UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

TESIS DE MAESTRÍA

Sistema de aprendizaje automático para la clasificación de páginas web de acuerdo al interés de una entidad

---

Autor:
Ing. Naivy PUJOL MÉNDEZ

Tutor:
MSc. Joelsy PORVEN RUBIER

Trabajo final presentado en opción al título de Máster en Informática Avanzada

La Habana, 10 de diciembre de 2018

Declaración de autoría

Yo, **Ing. Naivy PUJOL MÉNDEZ**, con carné de identidad **91060637614** declaro por este medio que soy la autora principal del trabajo final de maestría «Sistema de aprendizaje automático para la clasificación de páginas web de acuerdo al interés de una entidad», desarrollada como parte de la **Maestría en Informática Avanzada** y que autorizo a la **Universidad de las Ciencias Informáticas** a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Y para que así conste, firmo la presente declaración jurada de autoría, a los: 10 de diciembre de 2018

Ing. Naivy Pujol Méndez

«*Never let your sense of morals prevent you from doing what is right.*»

Isaac Asimov, Foundation

Resumen

Entre las principales tendencias tecnológicas para 2018 se encuentran: la inteligencia artificial, el análisis inteligente y sus aplicaciones, propiciado por al aumento de la disponibilidad de los datos y la necesidad de su análisis. La Web es una de sus principales áreas de aplicación, fomentando el desarrollo de la minería Web para el descubrimiento de patrones en ella. En la Universidad de las Ciencias Informáticas, en la Dirección de Redes y Servicios Telemáticos, se almacena información sobre la navegación. Esta información puede ser utilizada para la clasificación de páginas Web de acuerdo al interés de la universidad, pero es mucha información sin procesar. Además, la clasificación de las páginas Web de interés: se realiza manualmente siendo menos que las que no lo son y no se analizan los datos existentes o su relación con los patrones de navegación de los usuarios. A partir de estas limitantes se plantea como objetivo de la investigación: desarrollar un sistema de aprendizaje automático apoyado en los patrones de navegación de los usuarios que permita clasificar las páginas Web de acuerdo al interés de una entidad. Después de un análisis del estado del arte, se concluyó que las técnicas de aprendizaje automático son las más utilizada, su selección depende de los datos y estudios experimentales sobre ellos. La minería Web, es la encargada de estudiar los datos provenientes de la Web y su proceso define las fases: preprocesamiento, descubrimiento de los patrones y análisis de los patrones. En la primera, se analiza el conjunto de entrenamiento y se realizan las transformaciones necesarias para mejorar los resultados de los clasificadores. En la segunda fase, se analiza los parámetros, funcionamiento y entrenamiento de los clasificadores y su salida es utilizada en la tercera fase para el análisis de los patrones, así como, la validación de la solución. Los métodos estadísticos aplicados, identificó que la única métrica que presenta diferencias significativas es *Acurracy* y el análisis de su promedio en los clasificadores comprobó que el clasificador $KNeighborsClassifier$ presenta los mejores resultado. Además, con los elementos teóricos y prácticos de la minería del uso de la Web, el aprendizaje automático y la clasificación automática, se desarrolló un clasificador $KNeighborsClassifier$ utilizando los patrones de navegación de los usuarios, que permite clasificar las páginas Web de acuerdo al interés de una entidad.

Agradecimientos

« Son muchas las personas a las que siempre voy estar agradecida: mi familia, mis amigos, mis compañeros e incluso algunas que no son más que simples conocidos.

»De cada uno de ellos he aprendido algo, por tanto, merecen estar aquí.

»Algunos han sido importantes y definitorios en la formación de mi personalidad, gustos e intéreses; ellos son los que han hecho de mí la persona que soy hoy.

»Otros han potenciado mi vocación por la curiosidad, el análisis y la investigación, han ampliando mi visión como profesional e incluso he aprendido a utilizarlo en mi formación personal.

»También están de los que simplemente he adquirido ideas aisladas, que desconocen que me he apropiado de ellas, pero que son lo suficientemente importantes como para insidir directamente en mí.

»También hay otros, que incluso se encuentran en más de una categoría o incluso en todas.

»No voy a mencionar ningún nombre. Porque cada una de estas personas, cuando lean estos agradecimientos, si lo piensa solo un poco; sabrá en cuál de estas categorías está.

»Por este motivo, simplemente me queda decir: **gracias**»

Índice general

Índice de figuras

Índice de tablas

Introducción

El avance de las Tecnologías de la Información y las Comunicaciones, ha elevado la cantidad de información almacenada a partir de las diferentes áreas de la vida común. Durante años, la empresa Gartner ha informado las tendencias de investigación de las tecnologías de la información y en octubre del 2017 se celebró el *Gartner Symposium/ITxpo 2017: innovation and disruption*. En esta ocasión, dieron a conocer las diez principales tendencias tecnológicas estratégicas para 2018 y entre ellas se encontraban: la inteligencia artificial, el análisis inteligente y sus aplicaciones. Con respecto a estos temas y en dicho evento David W. Cearley, vicepresidente de la compañía, expresó: «*En los próximos años, cada aplicación, aplicación y servicio incorporará inteligencia artificial en algún nivel ... para agregar a los datos valor comercial en nuevas versiones; en forma de análisis avanzados, procesos inteligentes y experiencias de usuario avanzadas.*»

El uso del análisis inteligente se ha fortalecido debido al crecimiento explosivo de los datos semi-estructurados y no-estructurados almacenados actualmente, por tanto, las organizaciones buscan formas de innovar apoyándose en la **ciencia de datos** (van der Aalst, 2014; Blei y col., 2017). El creciente interés de estas por extraer información y producir conocimiento a partir de la cantidad masiva de datos, ha propiciado el fortalecimiento de esta disciplina (Fayyad; Simoudis y col., 2017). Por otra parte, la disponibilidad de *BigData* ha fomentado en las organizaciones el análisis de datos con el fin de extraer conocimiento procesable y entendible que pueda utilizarse en la toma de decisiones y en la realización de predicciones sólidas con respecto a su modelo de negocio (Molina-Solana y col., 2017; Giama y col., 2018).

El sector empresarial utiliza el potencial predictivo del análisis de datos para mejorar la gestión estratégica (Agarwal y col., 2014), la eficiencia operativa (Cai y col., 2015) y el rendimiento financiero de las organizaciones (Ausloos y col., 2016; Newman y col., 2016). Pero no es la masividad lo que hace a los datos almacenados interesantes y/o planteen desafíos en la realización se su análisis, son los datos en sí y su comportamiento en tiempo real los que los convierten en componentes básicos en la búsqueda de conocimiento (van der Aalst, 2014; Bichler y col., 2017; Rupp y col., 2017).

Una característica importante de los datos almacenados es su diversidad (Schutt y col., 2013); pueden ser desde los tradicionales: numérico, categórico o binario, hasta más complejos como lo son: texto, datos de ubicación geográfica, de red, de sensores, imágenes objetivo o logs[1]. Por tanto, los principales desafíos que dan paso al fortalecimiento de la ciencia de datos están dados por la necesidad de analizar datos diversos e incompletos (Provost y col., 2013), desordenados (Schutt y col., 2013), de gran volumen (Agarwal y col., 2014; Ayankoya y col., 2014; Cai y col., 2015), que cambian en el tiempo (Kormos y col., 2017). Además, de la necesidad de encontrar información que

[1]Registros de datos a nivel de usuario y datos de eventos con marcas de tiempo

impulsen y apoyen la toma de decisiones de las organizaciones a partir de estos datos.

El término ciencia de datos se acuñó a principios del siglo XXI y se le atribuye a (Cleveland, 2001), donde lo define como una continuación de algunos campos de análisis de datos como la estadística y pone a la disciplina propuesta en el contexto de la informática y la Ciencia de la computación. Con el tiempo esta disciplina fue ampliada con la incorporación de nuevos elementos, (Hazen y col., 2014) propone una nueva definición que incorpora las nuevas herramientas proporcionada por las Tecnologías de la Información y las Comunicaciones: «*...campo emergente que combina, ciencias de la computación, estadística, matemática, y la experiencia ciencia del comportamiento, para generar conocimiento sobre los datos de las organizaciones.*» A la definición otros elementos han sido agregados posteriormente cómo su relación con la minería de datos (Provost y col., 2013), la minería de procesos (van der Aalst, 2016) y el aprendizaje automático (AA) (Schutt y col., 2013; Karpatne y col., 2017).

Las principales ventajas que aporta el uso de la ciencia de datos, según (Gan y col., 2016) son: la detección de patrones no descubiertos, mejorar la gestión y el control de la información, facilidades en el acceso y la explotación de lo datos, prevención de fugas de información y la detección de comportamientos inadecuados que causen errores. Como se mencionó anteriormente, generar conocimiento útil que permita comprender mejor el comportamiento de las organizaciones, es la principal ventaja de la ciencia de datos (Schoenherr y col., 2015). Pero según (Cadez y col., 2003), uno de los grandes desafíos de esta disciplina es la comprensión del comportamiento humano en el contexto de los «entornos digitales», como la Web.

Actualmente, las organizaciones que interactúan con la Web buscan la manera de hacer uso de estás ventajas a partir del uso de la información almacenada en los logs sobre el comportamiento de los usuarios y las páginas Web (Schutt y col., 2013). (Tarik y col., 2017) plantea que la exploración basada en logs generados a partir del uso de la Web son un punto de partida importante para ampliar la comprensión del comportamiento de las organizaciones que ofrecen servicios asociados a la misma. Y dentro de las técnicas estudiadas en la literatura, la clasificación automática juega un papel importante ya que la misma tiene como objetivo principal asignar a los datos en una de las varias clases predefinidas, lo que permite la estructuración de la información para análisis posteriores (S. Gupta y col., 2017).

Se realizó un diagnóstico preliminar para conocer las investigaciones relacionadas a la clasificación de páginas Web, después de realizada la misma se pudo constatar que las técnicas utilizadas dependen de las características de los datos y existen varias soluciones propuestas en la literatura (Tarik y col., 2017). Dentro de las investigaciones relevantes estudiadas referentes al tema se encuentran: la clasificación de patrones en la personalización Web utilizando el análisis de logs de acceso (Pierrakos y col., 2003; Shanmugam y col., 2015); la clasificación en multi-etiquetas predeterminadas utilizando aprendizaje automático (Patil y col., 2012); la clasificación binaria en cuanto a si tienen contenido inapropiado o no, utilizando métodos de minería de datos y aprendizaje automático (Kotenko y col., 2014); la predicción del patrones de navegación a partir de

la clasificación utilizando aprendizaje automático (Sathyadevan y col., 2014; Kansara, 2015); la clasificación para la reducción de la dimensionalidad de los enlaces (Dehankar y col., 2015). En las investigaciones anteriormente estudiadas los principales tipos de datos considerados son el contenido, las URL[2] y el contexto; en el proceso de clasificar las páginas Web.

Entre los retos identificados en la literatura con respecto a la clasificación de páginas Web se encuentran (Wong y col., 2002; Naidenova, 2018): debido a la heterogeneidad de los datos almacenados y su naturaleza semi-estructurada y no estructurada, es necesario utilizar diferentes tipos de técnicas para poder realizar los análisis. En dependencia del tipo de dato del que proviene la información es necesario analizar el contenido de las páginas Web, por lo tanto, las mismas tienen que estar accesibles para realizar los análisis. Debido al gran volumen de información, es necesarios el uso de métodos para reducir la dimensionalidad de las página que se deben clasificar para que permita reducir la complejidad computacional y el tiempo de búsqueda.

Otro de los retos, es el carácter dinámico de los datos en la Web que provocan que la clasificación pueda ser modificada en el transcurso del tiempo. Con el fin de brindar soluciones a estos problemas surge a principios de milenio una nueva área para su estudio denominada minería Web (MW).

La minería Web surge como una aplicación de técnicas de minería de datos para descubrir patrones en la Web y tiene como objetivo descubrir información útil o conocimiento de hipervínculos, contenidos de páginas y logs (Kosala y col., 2000; B. Liu, 2007). Aunque la MW utiliza muchas técnicas de minería de datos, como se mencionó anteriormente, no es puramente una aplicación de técnicas tradicionales debido a la heterogeneidad y la naturaleza semi-estructurada o no estructurada de los datos Web (Rao y col., 2017). En función de los principales tipos de datos utilizados, las tareas de la misma se pueden clasificar en tres tipos fundamentales: minería del contenido de la Web, (WCM, Web Content Mining), minería de la estructura de la Web (WSM, Web Structure Mining) y minería del uso de la Web, (WUM, Web Usage Mining) (Kosala y col., 2000; B. Liu y Chen-Chuan-Chang, 2004).

En ocasiones es necesario que la clasificación no se centre solamente en las características de las páginas Web, sino que se clasifiquen de acuerdo al comportamiento de los usuarios para realizar los análisis (Lopes y col., 2015). Dentro de la MW la minería del uso de la Web es la encargada de resolver este tipo de problemas, ya que la misma tiene como objetivos la detección y análisis automáticos de patrones en datos recopilados o generados, como resultado de interacciones del usuario con recursos Web en una o más páginas (Srivastava y col., 2015). Según (Rao y col., 2017), es una técnica efectiva para extraer conocimiento de los datos no estructurados.

El uso de minería de uso de la Web tiene varias aplicaciones debido a que no es necesario tener conocimiento previo de los recursos Web (Rao y col., 2017), entre ellas se encuentran: utilización en el comercio electrónico para hacer marketing personalizado (Pierrakos y col., 2003); utilización para la predicción mediante el reconocimiento de las actividades y patrones con el objetivo de mejorar la

[2]Uniform Resource Locator (URL por sus siglas en inglés), es el término coloquial para referirse a una dirección Web

relación organización-cliente (Parmar, 2015; Rao y col., 2017). A pesar de las ventajas que brinda la aplicación de la WUM, existen varios problemas a enfrentar con su uso, dentro de ellos se encuentran:

- Los logs contienen datos irrelevantes que no contribuyen a la extracción de información útil, por lo que requieren un preprocesamiento y estudio exploratorio (Rao y col., 2017).
- La utilización de datos personales genera algunas inquietudes en los usuarios (Rao y col., 2017).
- La gran cantidad de datos que es necesario analizar, es creciente, cambiante y evoluciona en el tiempo (Kiziloluk y col., 2017).

En la Universidad de las Ciencias Informáticas (UCI), en la Dirección de Redes y Servicios Telemáticos, se ha almacenado en logs con el volumen de datos necesarios para estudiar los datos y obtener patrones sobre las tendencias de navegación de los usuarios. Una de las áreas que requiere el análisis esta relacionada con el acceso a páginas Web de interés de la universidad. Para ello actualmente se tiene un registro de los dominios de interés de la universidad, a partir de los cuales se puede conocer si una página es de interés o no.

Por otra parte, el procedimiento llevado a cabo para clasificar las páginas Web de interés se realiza manualmente y una página es clasificada como de interés atrevés de una petición directa a la dirección. Las principales áreas encargadas de realizar estas peticiones son: formación, producción e investigación, pero por las característica de este procedimiento, no se conoce a priori las características que deben de cumplir una página Web para ser considerada de interés. Debido a al gran volumen de páginas que alberga la Web realizar un estudio de las páginas Web que podrían ser de interés y no han sido clasificadas, resulta un proceso costoso el cuál necesita ser automatizado. Por tanto, son muchas las páginas Web que podrían ser de interés y no son identificadas.

Con el fin de lograr una mejor clasificación de las páginas Web de interés se ha decidido utilizar el conjunto de logs almacenados con el comportamiento de la navegación de los usuarios, que permita tomar una acción proactiva en la clasificación de las páginas Web. El conjunto de datos, consiste en un histórico de la navegación de los usuarios y su comportamiento de manera general en cuanto a tiempo, tamaño de la petición, cantidad de peticiones y la URL. Además, se cuenta con el listado de los dominios de interés que han sido identificados hasta el momento de acuerdo al interés de la universidad.

Entre los principales retos identificados con respecto al análisis de los datos disponibles y la clasificación de las páginas Web de interés se encuentran: que el conjunto de datos contiene un gran volumen de información como para poder ser gestionado manualmente, con el propósito de descubrir patrones y conocimientos interesantes que permitan apoyar la toma de decisiones en la clasificación de las páginas. Otro elemento importante es la heterogeneidad de la distribución de las páginas Web de interés con respecto a las que no los son, por tanto, existe un desbalace entre los dos tipos valores en los que se puede realizar la clasificación. Después del análisis de la situación problemática, se identificaron las siguientes limitaciones:

- Se almacena mucha información pero no se realiza procesamiento de la misma, de manera que se aprovechen las tendencias de navegación de los usuarios en la clasificación de páginas Web de interés. Además, la identificación y clasificación de nuevas páginas Web de interés se realiza manualmente, provocando que sean muchas las que no se identifican.
- No se realiza análisis a partir de los datos existentes, la relación entre los patrones de navegación de los usuarios y la clasificación de páginas Web de interés.
- Desbalance de las páginas Web de interés clasificadas, provocando que existan menos elementos de la clase objetivo.
- Pueden existir páginas Web incorrectamente clasificadas, debido a que solo están clasificadas las que son de interés y no las que no lo son.

A partir de los elementos anteriormente planteados se identifica el siguiente **problema de investigación**: ¿Cómo clasificar las páginas Web apoyado en los patrones de navegación de los usuarios de acuerdo al interés de una entidad?

El **objeto de estudio**: el procesos de clasificación de páginas Web y el **campo de acción**: Algoritmos de la minería de uso de la Web para la clasificación de páginas Web.

Objetivo general:

Desarrollar un sistema de aprendizaje automático apoyado en los patrones de navegación de los usuarios que permita clasificar las páginas Web de acuerdo al interés de una entidad.

Objetivos específicos:

- Construir el marco teórico referencial de la investigación, relacionado al proceso de clasificación de páginas Web.
- Desarrollar un sistema automatizado que permita utilizar los patrones de navegación de los usuarios en la clasificación de páginas Web.
- Validar el sistema de aprendizaje automático desarrollado a partir de los métodos científicos definidos.

Para guiar la investigación se propone la siguiente **hipótesis científica**:

El desarrollo de un sistema de aprendizaje automático apoyado en los patrones de navegación de los usuarios permite clasificar las páginas Web de acuerdo al interés de una entidad.

En el anexo **A**, se puede observar la operacionalización de las variables a partir de la hipótesis planteada.

Para guiar y organizar el desarrollo de la investigación las técnicas y métodos científicos empleados fueron los siguientes (Hernández Sampieri y col., 2010):

Análisis-síntesis: para el estudio de las fuentes bibliográficas existentes referente a los temas del proceso de clasificación de páginas Web, la minería de uso de la Web y el aprendizaje automático; identificando los elementos más importantes que representen las tendencias actuales para dar solución del problema planteado. Se citó fundamentalmente bibliografía del periodo 2014-2018 para asegurar la actualidad de los elementos abordados.

Inductivo-deductivo: para el estudio de las principales tendencias y las técnicas de aprendizaje automático más utilizadas en la clasificación de páginas Web. Además, el estudio del procesos llevado a cabo por la minería del uso de la Web. Ambos estudios con el objetivo de determinar cuáles son las alternativas viables a incorporar en la presente investigación.

Método hipotético-deductivo: para guiar la investigación se hace uso de una hipótesis científica. A partir de la observación y el análisis del fenómeno en cuestión se formuló una hipótesis que será comprobada en el proceso de validación.

Observación: dirigida a la percepción del comportamiento de los diferentes clasificadores seleccionados para estudiar el comportamiento de acuerdo a los parámetros.

Medición: dirigida a la medición de la eficacia y la interpretabilidad de los clasificadores obtenidos por los métodos del estado del arte y por la solución que se propone.

Experimental: para realizar la validación, con técnicas estadísticas paramétricas y no-paramétricas, de los resultados obtenidos derivados de los resultados de las métricas obtenidas de los clasificadores.

El documento está estructurado en tres capítulos:

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN: en el capítulo se abordan los elementos que conforman el marco teórico referencial y que componen el objeto de estudio de la investigación. Se describen las principales cuestiones implícitas en la propuesta de solución relacionadas con los fundamentos teóricos para los procesos de clasificación de páginas Web. Además, de los aspectos medulares de la investigación relacionados a la clasificación de páginas Web, la minería de uso de la Web y el aprendizaje automático. Por último se realiza un análisis de las herramientas y tecnologías para el desarrollo del sistema automatizado y se formulan las conclusiones del capítulo.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN: en el capítulo se realiza un análisis del proceso de clasificación así como una breve descripción de los algoritmos a utilizar. Se realiza la descripción de las dos primeras etapas que proponen el proceso de la minería Web: preprocesamiento y

descubrimiento de patrones y se describen las características de la solución propuesta y su arquitectura. En la etapa de preprocesamiento se abordan los temas de la construcción del conjunto de entrenamiento, la limpieza y escalado de los datos, la reducción de la dimensionalidad y el desbalance de los mismos En la etapa de descubrimiento de patrones se analizan los algoritmos de clasificación seleccionados de acuerdo a su comportamiento interno, con el fin de encontrar parámetros eficientes en la clasificación de páginas Web.

CAPÍTULO 3. DESARROLLO Y VALIDACIÓN DE LA PROPUESTA: en el capítulo se realiza el análisis de los resultados obtenidos a partir de la evaluación de los resultados utilizando las métricas definidas. Además, se realiza la descripción de la última etapa que proponen el proceso de la minería Web: el análisis de los resultados utilizando un experimento para analizar la evaluación de los clasificadores utilizados mediante las métricas definidas en el capítulo 1. Por último, se describe los métodos estadísticos utilizados para el análisis comparativo entre los clasificadores y se arriban a las conclusiones.

Finalmente se presentan las **Conclusiones**, se emiten las **Recomendaciones**, se listan las **Referencias bibliográficas** y se incluyen los **Anexos.**

Capítulo 1

Fundamentación teórica de la investigación

Introducción

En el presente capítulo se abordan los elementos que conforman el marco teórico referencial y que componen el objeto de estudio de la investigación. Se describen las principales cuestiones implícitas en la propuesta de solución relacionadas con los fundamentos teóricos para los procesos de clasificación de páginasWeb. Además, de los aspectos medulares de la investigación relacionados a la clasificación de páginas Web, la minería de uso de la Web y el aprendizaje automático. Por último se realiza un análisis de las herramientas y tecnologías para el desarrollo del sistema automatizado y se formulan las conclusiones del capítulo.

1.1. Ciencia de los datos y la clasificación de páginas Web

Existe un creciente interés en las organizaciones por extraer información y producir conocimiento útil a partir de la cantidad masiva de datos creados diariamente (Fayyad; Candel y col., 2017). La disponibilidad de los mismos ha permitido a las organizaciones aprovechar el análisis inteligente de los datos con el fin de descubrir patrones que puedan utilizarse para la toma de decisiones y la realización de predicciones sólidas referentes a sus modelos de negocios (Molina-Solana y col., 2017; Giama y col., 2018).

A principios del milenio, con los nuevos avances de las Tecnologías de la Información y las Comunicaciones, surge la **ciencia de datos** como el área encargada de estudiar el comportamiento de los datos (Kosala y col., 2000). Según (Hazen y col., 2014), la ciencia de datos, se define como un campo emergente que combina: ciencias de la computación, la estadística, las matemáticas y la experiencia del área de aplicación; con el fin de obtener patrones a partir de los datos. A esta nueva área, con el tiempo ha sido enriquecida y otros elementos han sido agregados posteriormente cómo sus relaciones con la minería de datos (Provost y col., 2013), la minería de procesos (van der Aalst, 2016) y el aprendizaje automático (Karpatne y col., 2017; Chojnacki y col., 2017).

La ciencia de datos cuenta con una gran cantidad de herramientas provocando que existan soluciones diferentes, estrechamente relacionadas con el área de aplicación y las características particulares del problema a resolver (Foreman y col., 2014; Dalkir y col., 2017). Según (Cadez y col., 2003), entre los

grandes desafíos se encuentra: la comprensión del comportamiento humano en el contexto de los **entornos digitales**, como la Web.

Después de realizado un estudio se logró identificar que las organizaciones buscan la manera de utilizar las ventajas que brinda el uso de la ciencia de datos para el análisis de los logs, ya que estos son la principal fuente de información acerca de la actividad de la red, comportamiento de los usuarios y los sistemas (Schutt y col., 2013). Entre las principales ventajas que aporta el estudio del comportamiento de los logs se encuentran (Gan y col., 2016):

- Facilitar el acceso y la explotación de estos datos.
- Prevenir fugas de información y comportamientos inadecuados que causen errores.
- Detectar patrones a partir de la aplicación de técnicas de aprendizaje automático.
- Mejorar la gestión y el control de la información a partir de la información obtenida.

En los últimos años, el número de logs y la cantidad de datos en la Web ha crecido exponencialmente (Durant y col., 2006), permitiendo que las organizaciones utilicen la información generada con el fin de encontrar patrones que puedan ser utilizados para mejorar los servicios brindados. La ciencia de datos provee herramientas que permiten realizar el análisis inteligente del comportamiento de los logs. Después de realizado un diagnóstico preliminar se identificó la clasificación automática juega un papel importante, debido a que la misma permite la estructuración de la información para análisis posteriores.

1.1.1. Clasificación de páginas Web

Varias son las organizaciones que pueden generar conocimiento, a partir de la clasificación de los páginas Web que les permitan mejorar los servicios brindados. La clasificación tiene como objetivo principal asignar a los elementos de datos una de las varias clases predefinidas o categorías en particular (A. Gupta y col., 2014). Después de realizado un estudio sobre las investigaciones relacionadas a la clasificación de páginas Web, se pudo constatar que estas dependen de las características de los datos. Por tanto, son variadas las soluciones propuestas en la literatura y se encuentran estrechamente relacionadas con la fuente y tipo al que pertenecen los datos (Tarik y col., 2017).

Para conocer las tendencias en el estado del arte sobre la clasificación de páginas Web se realizó un análisis bibliográfico, teniendo en cuenta las investigaciones que tienen como objetivo la clasificación de páginas Web. Los elementos que se tuvieron en cuenta para la realización del análisis son: el año, los autores, el título, el objetivo de la investigación y las técnicas o métodos

utilizados en las soluciones propuestas. En el anexo B, se observa un resumen con los 19 principales artículos analizados.

En el caso del tipo de dato empleado, dentro de las investigaciones analizadas se pueden destacar: la estructura de la páginas (Glover y col., 2002), resúmenes de las páginas (Shen y col., 2004), el contenido (Kan y col., 2005; Rajalakshmi y col., 2011), las características de las URL (Bhalla y col., 2017), y la semántica (Saleh y col., 2017). En cuanto al el objetivo de la clasificación también son diferentes: en (zu Eissen y col., 2004) clasifica en varios géneros, en (Weibel, 2017) en temas y en (Saleh y col., 2017) de acuerdo a la semántica. Otro elemento importante observado en el análisis es que en ellas es necesario el acceso a la información de las páginas Web, tanto para analizar su estructura, las URL o la semántica.

Del análisis se pudo constatar que aunque utilizan la clasificación de páginas Web, el uso que le dan a la misma dependen del problema, así como el tipo de solución que utilizan. Además, en cuanto a las técnicas identificadas predomina el uso de técnicas de Inteligencia Artificial donde la mayoría de ellas son de aprendizaje automático. Dentro de las técnicas más utilizadas se encuentran *Naive Bayes* y *Support Vector Machine*, aunque lo que predomina es el uso de varias de estas técnicas de las cuales se escoge la de mejores resultados a partir de estudios experimentales.

En ocasiones es necesario que la clasificación no se centre solamente en las características de las páginas Web, sino que se clasifiquen de acuerdo al comportamiento de los usuarios (Lopes y col., 2015) y la información generada a partir de la navegación de grupos específicos. La exploración basada en log generados por los páginas Web es un punto de partida importante para ampliar la comprensión del comportamiento de las organizaciones (Tarik y col., 2017). El área encargada de descubrir automáticamente información en datos asociados a las páginas Web se denomina **minería Web**.

1.2. Minería Web

Según (Kosala y col., 2000), la minería Web es la aplicación de técnicas de minería de datos para encontrar patrones de la Web y tiene como objetivo descubrir información útil o conocimiento a partir del análisis de hipervínculos, contenidos de las páginas Web y de los logs (B. Liu, 2007). En la formación del área de estudio se tomaron dos enfoques diferentes para definir la minería Web. El primero fue una «*visión centrada en el proceso*», donde se describe como una secuencia de tareas (B. Liu, 2007). El segundo fue una «*visión centrada en los datos*» donde las decisión sobre el las técnicas a emplear estaban estrechamente relacionadas con el tipo de dato (B. Liu, 2007).

Otro elemento importante con respecto a la minería Web, es que la misma utiliza muchas técnicas de la minería de datos, pero no es puramente una aplicación de las técnicas tradicionales, debido a la heterogeneidad y la naturaleza semi-estructurada o no-estructurada de los datos en la Web (Rao

y col., 2017). Después de realizado un estudio en la literatura se pudo constatar que hay tres tipos de fuente de información que utiliza la minería Web (Rao y col., 2017):

- Uso: proveniente de los logs del servidor y el seguimiento de la actividad de los navegadores Web.
- Estructura: de los enlaces e hipervínculos entre las páginas Web.
- Contenido: de la información contenida en las páginas Web, tanto del contenido publicado cómo de su implementación.

Para analizar las características posen cada uno de estos tipos de fuente de información se realizó un resumen con los principales elementos encontrados en la literatura. En la tabla 1.1, se observa los elementos seleccionados, dentro de ellos se encuentran: tipo y fuente de datos, representación comúnmente y las técnicas utilizadas para descubrir los patrones según la bibliografía consultada.

TABLA 1.1: Resumen de las fuentes de información en de la minería Web (Elaboración propia).

	minería Web		
	Contenido	**Estructura**	**Uso**
Tipo de datos	No-estructurados y semi-estructurados	Hipervínculos	Interactivos, no-estructurados
Fuente de datos	Documentos de texto e hipertexto	Hipervínculos	Logs del servidor
Representación	Bolsa de palabras, n-gramas, frases, conceptos y ontologías	Grafos	Grafos y base de datos
Técnicas	Aprendizaje automático, estadística, procesamiento de lenguaje natural	Aprendizaje automático, teoría de grafos	Aprendizaje automático, estadística y teoría de grafos

En dependencia de la fuente de información, el tipo de representación de los datos difiere, así como las técnicas de análisis utilizadas para la extracción de los patrones. Por ello, en función de los principales tipos de datos utilizados en el proceso de descubrimiento de patrones, la minería Web se puede clasificar en tres tipos (Kosala y col., 2000; C. Liu y col., 2000; B. Liu y Chen-Chuan-Chang, 2004; B. Liu, 2007):

- Minería de contenido Web: extrae información útil o conocimiento de los contenidos de la página Web, tanto en la información que brinda directamente, como la que contiene implícita en su desarrollo.
- Minería de estructura Web: descubre conocimiento útil utilizando hipervínculos, que representan la estructura de la Web, utilizan las conexiones entre las páginas Web.

- Minería de uso Web: descubre los patrones a partir de la capturar, modelación y análisis los patrones de comportamiento y de los perfiles de los usuarios que interactúan con las páginas Web.

Dentro de la minería Web, como se muestra en la clasificación anterior, la minería de uso de la Web es la encargada de descubrir información útil en los datos que se almacenan en logs. La información con la que se cuenta para conocer si una página Web es de interés de la universidad o no, proviene de los logs de navegación de los usuarios. Debido al tipo de fuente de información de donde provienen los datos y el análisis realizado anteriormente, se puede concluir que la presente investigación se utilizará la minería de uso de la Web para la clasificación de páginas Web.

1.2.1. Minería de uso Web

La minería de uso Web (WUM, por sus siglas en inglés Web usage Mining), se refiere a la detección y análisis automáticos de patrones en datos recopilados o generados, como resultado de interacciones del usuario con recursos Web en uno o más páginas Web (Srivastava y col., 2015). Según (Rao y col., 2017), es una técnica efectiva para extraer conocimiento de los datos no estructurados y tiene varias ventajas que hacen atractivo su uso (Rao y col., 2017), las dos principales ventajas son: la utilización en el comercio electrónico para hacer marketing personalizado a partir de la preferencias de los usuarios (Pierrakos y col., 2003) y la utilización para la extracción de patrones que permiten establecer una mejor relación organización-usuarios (Rao y col., 2017). A pesar de estas ventajas que brinda la WUM, después de realizado un análisis en la literatura se identificaron que existen varios problemas en su uso entre ellas se encuentran (Rao y col., 2017; Kiziloluk y col., 2017):

- Los log contienen datos irrelevantes que influyen negativamente en la extracción de patrones, por lo tanto es necesario realizar un preprocesamiento de los datos.
- La utilización de datos personales genera algunas inquietudes éticas en cuanto a su uso, por tanto, es necesario utilizar un mecanismo de anonimización de la información.
- El gran volumen de datos que es necesario analizar se caracteriza por: su creciente y cambiante evolución y su naturaleza no estructurada.

Cada una de las limitantes mencionadas anteriormente es necesario tenerlas en cuenta en la solución propuesta. Otro elemento que es necesario analizar para utilizar la WUM son los pasos a seguir que permitan organizar el trabajo y que tenga en cuenta el tratamiento de a las limitantes anteriormente mencionadas. Para ello en el próximo acápite se realiza un análisis para la selección del proceso que guiará el desarrollo de la solución.

1.2.2. Proceso de la minería Web

Después de realizado una análisis de los diferentes procesos que guían el desarrollo de soluciones que utilizan la minería Web, se puede concluir que este guarda una estrecha relación con la **minería de datos** y describen tres pasos fundamentales: preprocesamiento, descubrimiento y análisis de los patrones (B. Liu, 2007; Hicks y col., 2016; Rao y col., 2017).

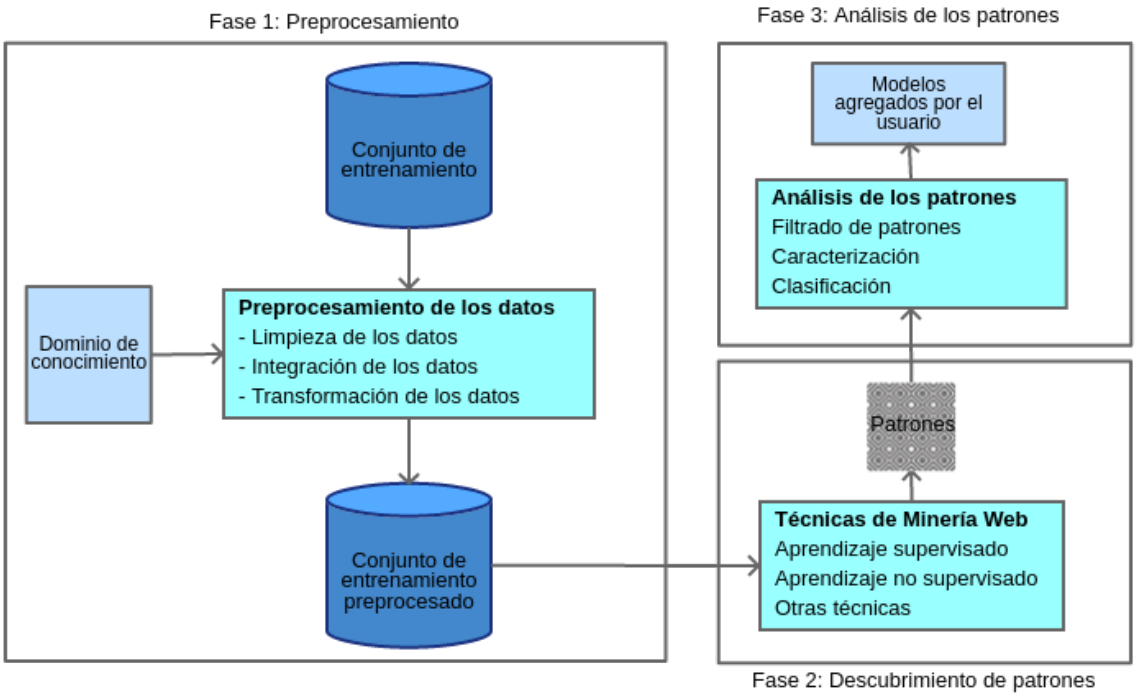


FIGURA 1.1: Proceso de la minería Web (B. Liu, 2007).

En la presente investigación se utiliza el propuesto por (B. Liu, 2007), mediante su utilización se facilita tener en cuanta las limitantes identificadas con respecto a la MUW. Además, incluye los principales elementos identificados en la literatura; comienza con la comprensión del dominio de aplicación del problema, identifica las fuentes de datos y los datos objetivos, para después dar paso al descubrimiento de los patrones y su posterior análisis (B. Liu, 2007). A continuación se muestra una descripción general de los tres pasos propuestos por el proceso (B. Liu, 2007):

- Preprocesamiento: En esta etapa, se conforma el conjunto de entrenamiento, los datos se limpian, se anonimizan y se dividen a partir de la información de cada usuario (Rao y col., 2017). Una de las cuestiones clave en esta etapa es preprocesamiento de datos de navegación en los logs con el fin de obtener los datos correctos para las etapas posteriores (Zhu y col., 2015). El objetivo principal es capturar, modelar y analizar los patrones de comportamiento a partir de la información del comportamiento de los usuarios con las páginas Web.

- Descubrimiento de los patrones: En esta etapa el descubrimiento de patrones a partir de diferentes técnicas que reflejen el comportamiento de los usuarios (Adhvaryu, s.f.). Dependiendo del tipo de problema que se desea solucionar, propiciado por la complejidad y tamaño de los datos es común el uso de aprendizaje automático en esta etapa.
- Análisis de los patrones: En esta etapa, los patrones y las estadísticas descubiertos se procesan, filtran y generan modelos agregados por de usuarios que se pueden utilizar como entrada para aplicaciones tales como motores de recomendación, herramientas de visualización y análisis Web y herramientas de generación de informes (Tarik y col., 2017).

En la figura 1.1, se observa cada una de las etapas y los principales elementos que se tienen en cuenta en cada una de ellas. También se muestran cuáles son las entradas en cada etapa y las relaciones entre cada una de ellas.

1.3. Preprocesamiento

El preprocesamiento de datos es un paso preliminar durante el proceso de la minería Web. Se encarga de realizar transformaciones a los datos brutos con el objetivo de mejorar su comprensión y uso sin perder su integridad (García; Ramírez-Gallego y col., 2016). La información a partir de la cuál se realiza el análisis de los patrones del comportamiento de la navegación de los usuarios, en la presente investigación, cuenta con tres característica que deben tenerse en cuenta en esta etapa: la alta dimensionalidad, el desbalance de los datos entre las dos clases y la dispersión no estructurada de la información.

Se realizó un estudio del arte sobre las herramientas que se utilizan para dar solución a las características mencionadas anteriormente. En el caso de la reducción de dimensionalidad existen varias técnica, pero la más utilizada en la bibliografía es la extracción de características ya que permite eliminar las que no son relevantes y no aportan cambios significativos en las soluciones (Aytuğ Onan, 2016). En el caso de los datos desbalanceados se revisaron las técnicas para resolver los problemas de desbalance, teniendo en cuenta que la la clase minoritaria es la que se desea predecir.

Otro elemento importante identificado en la literatura con respecto a la etapa de preprocesamiento utilizada para mejorar el rendimiento de los algoritmos es el escalado de los datos. Dentro de las técnicas utilizadas para realizar el escalado de los datos estudiadas en la bibliografía, la normalización es la más utilizada ya que permite comprimir o extender los valores de la variable de entradas para que se distribuyan en un rango definido (Schwarzenbach y col., 2015). La normalización de los datos se realiza para que todas las variables de entrada tengan el mismo tratamiento en el modelo y los coeficientes de un modelo no se escalan con respecto a las variables

de las entradas. En los próximos acápites, se analizarán las técnicas identificadas en la literatura para resolver los problemas anteriormente mencionados.

1.3.1. Extracción de características

Una gran cantidad de información con una amplia cantidad de características puede ser útil para descubrir los patrones en la práctica no sucede de esta manera. Según (Qi y col., 2009), algunas características pueden ser simplemente «ruido», por lo que no contribuyen (o incluso degradan) el proceso de extracción de patrones. También está el caso de que puedan existir características similares que no afecten el posible resultado pero al analizar aumenten la complejidad computacional del algoritmo (Li y col., 2015).

La selección de un subconjunto de características se conoce como extracción de características (Law y col., 2004). Según (Fernández Hernández y col., 2016): «*Tiene como objetivo obtener un conjunto de entrenamiento lo más reducido posible que permita clasificar con la misma o más calidad que con el conjunto de entrenamiento original*». Dentro de la razones identificadas en el literatura por la cuál es importante la extracción de características la fundamental, según (Aytuğ Onan, 2016), es que las características ruidosas pueden degradar el rendimiento de la mayoría de los algoritmos de aprendizaje cómo se mencionó anteriormente.

En el análisis en la literatura se pudo constatar que la extracción de las características: puede mejorar el rendimiento de los clasificadores entrenados con cantidades limitadas de datos e iguales resultados (Law y col., 2004); permite clasificadores más económicos, tanto en almacenamiento como en tiempo de cómputo, (Jing, 2014) y en muchos casos, puede conducir a modelos interpretables por los analistas a partir de patrones generales (Onan, 2016).

La extracción de características es particularmente importante para conjuntos de datos con un gran número de características (Law y col., 2004) y esta es uno de los elementos que caracterizan la fuente de información, debido a que se tiene en cuanta el valor de la relación de que cada usuario con una página Web. Cada uno de los usuarios se convierte en una característica, por tanto, se esta en presencia de un problema que poseé tantas características como usuarios disponibles.

Dentro de las técnicas para la extracción de características estudiadas se encuentran las siguientes:

- *Latent Dirichlet Allocation*(LDA): Este método identifica componentes, es decir combinación lineal de las variables observadas, que maximizan la separación de clases o varianza entre clases, cuando dicha información previa está disponible. Por tanto, solo puede ser utilizada en la clasificación supervisada, o sea cuando se conoce el valor de la clase objetivo para el conjunto de entrenamiento (Moro y col., 2015).

- *Principal component analysis*(PCA): Este método busca encontrar componentes que tengan en cuenta la varianza máxima en los datos, incluido el error y la varianza dentro de la variable. A diferencia de LDA, no tiene en cuenta la pertenencia a la clase y se utiliza cuando dicha información no está disponible (Shlens, 2014).
- *Factor Analysis*(FA): Este método intenta descubrir los factores latentes que explican la varianza compartida entre las variables observadas, excluyendo así el error y la varianza dentro de la variable. Idealmente, los factores latentes resultantes representan constructos subyacentes interpretables. Un elemento importante para utilizar FA es que debe asumirse que existe un modelo causal subyacente que induce la covarianza entre varias variables observadas (Yousefi y col., 2014).

De las técnicas estudiadas, tanto LDA como PCA no requieren ninguna noción previa de cómo las variables se relacionan entre sí, que sí es necesario en el caso del uso de FA, por tanto cómo no se conoce la relación entre las características no se debe utilizar esta técnica. Por otra parte, según (Martinez y col., 2001) una de las principales ventajas del uso del algoritmo PCA es que es menos sensible a diferentes conjuntos de entrenamiento, lo que brinda soluciones más estables. Debido a esta característica en la presente investigación se utiliza PCA para reducir la dimensionalidad porque facilita el proceso de extensión de la solución con el uso de conjuntos de entrenamiento diferentes.

1.3.2. Datos desbalanceados

En muchas de las aplicaciones del mundo real es necesario aprender de conjuntos de datos desbalanceados o sea que contienen muy pocas instancias de algunas de las clases en las que se quiere realizar la clasificación. En la mayoría de estos problemas la clase minoritaria o interesante generalmente produce clasificadores sesgados que tienen una precisión predictiva más alta sobre la(s) clase(s) mayoritaria(as), pero una precisión predictiva peor que la clase minoritaria (Bhagat y col., 2015; Wang y col., 2017).

La mayoría de las investigaciones sobre el comportamiento de los clasificadores estándares entrenados con conjuntos desbalanceados indican que en general, se puede esperar un decremento del rendimiento del clasificador en presencia de clases con un nivel de representación altamente desigual (Wang y col., 2017). Dicha pérdida de rendimiento es mayor cuánto mayor es el desequilibrio existente, que queda cuantificado por el Índice de prevalencia de la clase Positiva (PPR[1]) (García; Luengo y col., 2014).

En general, las técnicas estudiadas para el trabajo con datos desbalanceados se agrupan de acuerdo a la estrategia de trasformación de las instancias. Los tres tipos de técnicas identificadas fueron (Kumar y col., 2012; Cano y col., 2016; Timothy y col., 2017):

[1] Siglas provenientes del Inglés, Positive Prevalence Rate, que indica el porcentaje de instancias cuya etiqueta corresponde a la clase de interés

- Over-sampling: estás técnicas realizan una transformación del conjunto de entrenamiento $B \rightarrow B_0$ tal que, el nuevo conjunto tenga el PPR más cercano al balanceo absoluto.
- Under-sampling: estás técnicas construyen distintas hipótesis sobre el mismo conjunto de entrenamiento, ya sea utilizando distintos subconjuntos de entrenamiento ($B*$) obtenidos a partir de B por muestreo de instancias o utilizando una penalización sucesiva que corrija los errores de clasificación.
- Técnicas híbridas: combinan técnicas de Over-resampling y de Under-sampling para dar lugar a nuevos subconjuntos de entrenamiento ($B*$) seleccionados a partir de modelos que utilizan ambos tipos de técnicas.

En la presente investigación para reducir el tiempo de computo de la solución se escoge el uso de técnicas de under-sampling, las cuales disminuyen la cantidad de entradas de la clase negativa en este caso las páginas Web que no son de interés de la universidad. De dichas técnicas pueden dividirse en dos las de generación o en los de selección de prototipos. Estas técnicas reducen el número de muestras en las clases objetivo, pero las muestras restantes se seleccionan, del conjunto original.

Según (Dittman y col., 2014), el uso de cualquiera de estas técnicas no presenta diferencia estadística en el resultado del muestreo de datos. De estas técnicas las que toman muestras aleatorias ($RandomUnder - Sampling$), es la que presenta mejor rendimiento. Además de ser la más frecuentemente utilizada en la clasificación. Debido a lo anteriormente mencionado, se decide utilizar esta técnica para el tratamiento del desbalance de los datos.

1.4. Aprendizaje automático

En la etapa en el proceso de la MUW, descubrimiento de los patrones, como se identificó en el acápite 1.2.1 las técnicas de aprendizaje automático son las más utilizadas en la misma. Esto se debe a que es una herramienta indispensable para problemas donde es necesario automatizar el descubrimiento de conocimiento con poca interacción de los usuarios, un gran conjunto de datos y con poca información sobre las fuentes de datos (Cadez y col., 2003; Maloof, 2006; Tarik y col., 2017).

En la literatura existen varias definiciones para el **aprendizaje automático**[2], según (Bishop, 2006): «el aprendizaje automático investiga cómo las computadoras pueden aprender utilizando la información de los datos». Por otra parte (Maloof, 2006) lo define como: «*... un campo de la informática que da a las computadoras la capacidad de aprender sin ser explícitamente.*» Una de las definiciones más ampliamente utilizada en la comunidad científica para definir ML es la siguiente: «*Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y la medida de*

[2](ML, por sus siglas en inglés Machine Learning)

rendimiento R, si su desempeño en tareas en T, medido por R, mejora la experiencia», entre los autores que lo proponen se encuentran los siguientes: (Bien y col., 2007; Holzinger y col., 2014; Biehl y col., 2016; Vasilevich y col., 2017). Esta última definición es la que se utiliza en la presente investigación, debido a que la misma tiene en cuenta los elementos principales dentro del ML: tareas, rendimiento y entrenamiento.

ML es una disciplina con una amplia aplicación en la actualidad (Durrant y col., 2017; Carrasquilla y col., 2017; Fu y col., 2017; R. Liu y col., 2017; Voyant y col., 2017), ya que provee diferentes soluciones en dependencia de las características de la información.

Según (Nasrabadi, 2007), el ML se puede clasificar en tres tipos en dependencia de la forma en la que realiza el descubrimiento del conocimiento:

- **Aprendizaje supervisado:** es una técnica para descubrir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares: por una parte los datos de entrada y por otra, los resultados deseados. Se necesitan ejemplos de datos en los que se tengan entradas y salidas para poder utilizar este tipo de técnicas.
- **Aprendizaje no supervisado:** es una técnica donde se trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento anterior sobre los resultados esperados.
- **Aprendizaje semi-supervisado:** es una técnica para los conjuntos de entrenamiento donde a veces no se puede acceso directo a los resultados deseados, pero se puede obtener alguna medida de la calidad de una salida o después de la entrada.

La elección del tipo de técnicas de ML a utilizar está estrechamente relacionado con la naturaleza de la información disponible (Asatkar y col., 2017), por tanto, en la presente investigación se decide utilizar el aprendizaje supervisado. Esta elección se debe a que ya se encuentran las clases establecidas (si es de interés o no) y se tienen ejemplos de datos en los que se tienen entradas y salidas.

Dentro del aprendizaje supervisado la clasificación juega un papel importante y como se analizó en el acápite 1.1.1 es el objetivo de la investigación. Debido a esto a continuación se analizan las diferentes técnicas de clasificación y su uso en en la resolución de problemas.

1.4.1. Clasificación automática

La clasificación automática se define como la acción ejecutada por un sistema informático sobre un conjunto de elementos con el fin de ordenarlos en clases o categorías (Maldonado, 2017). El problema de clasificación se basa en agrupar y discriminar objetos, descritos mediante un conjunto de atributos,

ya sea construyendo las clases o asignando los objetos a clases previamente definidas (Bishop, 2006). Para el mismo, se establece que las clases deben ser mutuamente excluyentes, o sea, una unidad de observación no puede ser clasificada simultáneamente en dos clases (Nasrabadi, 2007), con la misma escala, y a su vez, el conjunto de clases debe ser exhaustivo, no pudiendo ninguna unidad de observación quedar sin tener una clase donde pueda ser clasificada (Bell, 2015).

Según (Molina de Armas y col., 2013), la clasificación se debe utilizar en problemas que cumples con las condiciones siguientes:

- Una población de datos que se dividen en dos o más clases de acuerdo con una taxonomía determinada.
- Un grupo de datos de los cuáles se conoce con anterioridad la clase a la que pertenecen.
- Un conjunto de datos de los cuáles se desea saber a qué clase pertenecen.

Por otra parte, después del análisis realizado se identificó que dentro de la tarea de clasificación supervisada se puede dividir en los siguientes tipos:

- Clasificación binaria: cuándo sólo son posibles dos valores o clases para Y.
- Clasificación múltiple o multi-clase: cuándo el conjunto de valores que puede tomar Y es superior a dos.

Después de analizados los tipos de clasificación automática y debido a que el problema de clasificación que aborda en el presente trabajo cuenta con sólo dos clases, página Web de interés o no. Por tanto, en la presente investigación se hará uso de clasificación binaria supervisada porque es la que se ajusta a la resolución del problema debido a las características del mismo.

1.4.2. Conceptos fundamentales

Para la construcción de un clasificador es necesario conformar criterios que permitan determinar el valor de la clase a partir de un conjunto de ejemplos, denominado conjunto de entrenamiento, de un cierto dominio D, aplicando un algoritmo de aprendizaje (Bishop, 2006). Esos criterios están basados en los valores de uno o varios de los pares (atributo, valor) que intervienen en la definición de los ejemplos. El uso de un modelo clasificador estará determinado por su capacidad para predecir o inferir la clase a la que pertenece una nueva entidad correspondiente al dominio de estudio.

Entre los conceptos fundamentales asociados a la clasificación que son necesarios analizar se encuentran los siguientes (Bishop, 2006): Un **conjunto de datos** D consiste en un registro de los

datos, que se describen mediante un conjunto de atributos $A = A1, A2, ..., A|A|$, donde $|A|$ denota el número de atributos o el tamaño del conjunto A.

Tiene un atributo de objetivo C, que se llama **atributo de clase** o característica. El atributo de clase C tiene un conjunto de valores discretos, es decir, $C = c_1, c_2, ..., c_{|C|}$, donde $|C|$ es el número de clases y $|C| >= 2$. Un valor de clase también se llama etiqueta de clase.

Cada **instancia** describe una pieza de experiencia pasada, en la literatura de aprendizaje automático y minería de datos, también se llama ejemplo, caso o vector. Un conjunto de datos D básicamente consiste en un conjunto de ejemplos o instancias conformado por los atributos del conjunto A y C.

En la clasificación, es necesario dividir el conjunto de datos en dos subconjuntos disjuntos: **conjunto de entrenamiento** a partir del cuál se aprende o construye un modelo mediante un algoritmo de aprendizaje y **conjunto de prueba** para evaluar la precisión del modelo (B. Liu, 2007). Es importante tener en cuenta que los datos de prueba no se utilizan para aprender el modelo de clasificación. Los ejemplos en los datos de prueba generalmente también tienen etiquetas de clase. Es por eso que los datos de prueba se pueden usar para evaluar al modelo aprendido verificando si la clase predicha para cada caso de prueba por el modelo es la misma que la clase real del caso de prueba (Bishop, 2006).

Dado un conjunto de datos D, el objetivo del aprendizaje es producir un **algoritmo de aprendizaje** para relacionar valores de atributos en A y clases en C para predecir los valores de clase de los datos futuros. También se le puede llamar función o modelo de clasificación, o simplemente clasificador (B. Liu, 2007).

Otro de los elementos fundamentales dentro de la clasificación son los pasos del aprendizaje en la figura 1.2 se muestran estos pasos según (B. Liu, 2007). En el paso de entrenamiento, un algoritmo de aprendizaje utiliza los datos de entrenamiento para generar un modelo de clasificación. En el paso de prueba, el modelo aprendido se verifica los resultados utilizando el conjunto de prueba para obtener la evaluación de la clasificación. Si la precisión es satisfactoria, se puede usar en tareas del mundo real para predecir casos futuros. Si la precisión no es satisfactoria, es necesario elegir un algoritmo de aprendizaje diferente y/o hacer un procesamiento adicional de los datos (primera etapa del proceso de minería Web).

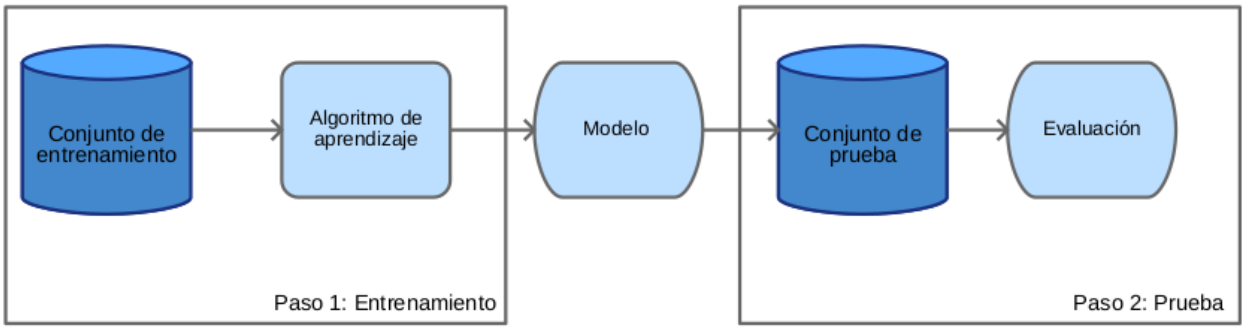


FIGURA 1.2: Pasos del aprendizaje automático (B. Liu, 2007).

1.4.3. Algoritmos de aprendizaje

Cómo se analizó en el acápite 1.1.1, en el proceso de clasificación de páginas Web, la elección del algoritmo de aprendizaje que se debe utilizar es un paso crítico. Varios técnicas dentro de la clasificación automática han sido estudiados. Dentro de ellos se encuentran los clasificadores bayesianos, las redes neuronales artificiales (ANN[3]), *Support Vector Machine* (SVM) y los ensamble de métodos. A continuación se realiza un análisis de estos clasificadores con el objetivo de realizar la selección de los que se utilizaran en la propuesta de solución.

Clasificador bayesiano: es un clasificador de patrones basado en las teorías estadísticas del aprendizaje (Maloof, 2006). Se calcula la probabilidad de cada hipótesis de los datos y se realiza la predicción a partir de esta, basándose en la teoría de decisión de Bayes. Dentro de las técnicas estudiadas se destacan en la clasificación las técnicas *Nearest Neighbor* y *Naive Bayes* (Al-Jefri y col., 2017). Dentro de estos algoritmos, *Naive Bayes* es ampliamente utilizado ya que requiere una pequeña cantidad de datos de entrenamiento para estimar los parámetros necesarios. Además, tienen muy buen rendimiento en comparación con otros métodos más complejos (Krichene, 2017).

En el caso de *Nearest Neighbor* logra un rendimiento constante, sin suposiciones a priori sobre las distribuciones a partir de las cuales se extraen los ejemplos de entrenamiento. En él, el número de muestras puede ser una constante definida por el usuario o variar en función de la densidad local de puntos. Por otra parte la distancia puede, en general, ser cualquiera de la métricas, la distancia euclidiana estándar es la opción más común utilizada en la literatura y la que se utilizará en la presente investigación (Tyagi y col., 2018). A pesar de su simplicidad, *Nearest Neighbor*, según lo estudiado en la literatura, han tenido éxito en una gran cantidad de problemas de clasificación, siendo un método no paramétrico, que a menudo es exitoso en situaciones de clasificación donde el límite de decisión es irregular.

[3]Por sus siglas en inglés, Artificial Neural Networks

Redes neuronales artificiales: Según (Raith y col., 2017), una ANN es una estructura compuesta de un número de unidades interconectadas. En ellas, la unidad análoga a la neurona biológica es el elemento procesador (PE[4]) o neuronas artificiales. Un PE tiene varias entradas y las combina, utilizando diferentes tipos de funciones. La suma de las entradas es modificada por una función de transferencia y el valor de la salida de esta función de transferencia se pasa directamente a la salida del próximo PE (Raith y col., 2017).

La salida del PE se puede conectar a las entradas de otras PE mediante conexiones ponderadas correspondientes a la eficacia de la sinapsis de las conexiones neuronales. Dentro de las principales técnicas estudiadas se encuentran: el perceptrón simple, el perceptrón multicapa (MLP [5]), los mapas auto-organizados, también conocidos como redes de Kohonen (Fei y col., 2017).

Dentro de las redes neuronales artificiales más utilizadas para la clasificación, según la literatura consultada se encuentra el perceptrón multicapa (Fei y col., 2017). Entre las ventajas de utilizar este método se encuentran: la capacidad para aprender modelos no lineales y en tiempo real (aprendizaje en línea) (Raith y col., 2017).

MLP es un algoritmo de aprendizaje supervisado que aprende una función $f(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^o$ por entrenamiento en un conjunto de datos, donde m es el número de dimensiones para la entrada y o es el número de dimensiones para la salida. Dado un conjunto de características $X = x\_1, x\_2, ..., x\_m$ y un objetivo se aprende una función no lineal para aproximar la clasificación.

La capa de entrada, consiste en un conjunto de neuronas $x\_i|x\_1, x\_2, ..., x\_m$ representando las características de entrada. Cada neurona en la capa oculta transforma los valores de la capa anterior con una suma lineal ponderada $w\_1x\_1 + w\_2x\_2 + ... + w\_mx\_m$, seguida de una función de activación no lineal $g(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$. La capa de salida recibe los valores de la última capa oculta y los transforma en valores de salida.

*Support Vector Machine***:** crea un hiperplano de separación óptima a partir de un conjunto de entrenamiento (Kim y col., 2017). Puede ser lineal o no lineal dependiendo de la función de optimización utilizada (kernel), por lo que se puede decir que es un caso específico de un clasificador lineal, sin embargo, su uso más común suele ser cuando el conjunto de datos no es linealmente separable (zu Eissen y col., 2004). Es considerado más fácil de usar que las redes neuronales y su meta es crear un modelo a partir de los datos de entrenamiento.

Entre las principales ventajas de estos métodos están: efectivo en espacios de alta dimensión; sigue siendo efectivo en casos donde el número de características es mayor que el número de muestras. Además, utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamados vectores de soporte), por lo que también es eficiente desde el punto de vista de uso de la RAM[6] (Kim

[4]Por sus siglas en inglés, Process Element

[5]Por sus siglas en inglés, Multilayer Perceptron

[6]Del inglés, Random-access memory, es una forma de almacenamiento de datos de la computadora que almacena los datos y el código de máquina que se está utilizando en ese momento.

y col., 2017).

También, se pueden especificar diferentes funciones del *kernel* para la función de decisión que pueden ser propios o especificados por el usuario, con el fin de obtener mejores resultados. Entre los kernels más usados según la literatura se encuentran: el lineal, el polinomial, el sigmoidal y la función de base radial.

Cómo se mencionó en el acápite 1.1.1, para la clasificación de páginas Web se utilizan en la mayoría de las investigaciones estudiadas varios de estos métodos y a partir de allí se realizan análisis estadísticos con el fin de utilizar el que mejores resultados arroje. Otras de las soluciones propuestas en la literatura es el uso de varios clasificadores mediante un ensamble de métodos que permita el uso de varios clasificadores débiles con el fin de obtener una mejor solución. En el próximo acápite, se realizará un análisis de los métodos de ensamble utilizados para la clasificación.

1.4.4. Ensamble de clasificadores

Para mejorar el rendimiento de los enfoques de clasificador se propone la combinación de múltiples clasificadores en el campo del aprendizaje automático. Estos se basan en la combinación de un grupo de clasificadores de modo que su fusión logra un rendimiento superior que los clasificadores independientes (Nanni y col., 2009). Por lo tanto, un ensamble de clasificadores es un conjunto de clasificadores, cuyas decisiones de clasificación individuales se combinan (Nanni y col., 2009).

La idea clave de varios métodos para construir un conjunto de clasificadores es modificar el conjunto de datos de entrenamiento, generando nuevos conjuntos de entrenamiento N_K, para construir clasificadores con los N_K nuevos conjuntos de entrenamiento, y luego los combina mediante una regla de decisión final (Aytuğ Onan, 2016). Existen varios métodos para realizar el ensamble de clasificadores. Según la literatura consultada, los principales métodos utilizados son los siguientes (Nanni y col., 2009):

- *Bagging* (empaquetado): Dado un conjunto de entrenamiento D, genera N_K nuevos conjuntos de entrenamiento $D1, ..., D_{NK}$. Al extraer aleatoriamente un subconjunto de los patrones de entrenamiento, cada nuevo conjunto D_i se usa para entrenar exactamente un clasificador.
- *Boosting* (reforzado): Mantiene un conjunto de pesos sobre el conjunto de entrenamiento original D y ajusta estos pesos después de que el algoritmo de aprendizaje básico aprende cada clasificador. Los ajustes aumentan el peso de los ejemplos que están mal clasificados por el algoritmo de aprendizaje básico y disminuyen el peso de los ejemplos que están clasificados correctamente. La familia de *Adaboost* es de los más representativos dentro de esta clasificación según la literatura.

Al igual que sucede con la selección del algoritmo de ML en el caso del ensamble de métodos sucede de igual manera. La mayoría de las investigaciones estudiadas utilizan varios de estos métodos y a partir de allí se realizan análisis estadísticos con el fin de utilizar el que mejores resultados arroje.

A partir del análisis realizado en los acápites 1.4.1, 1.4.3 y 1.4.4, se pudo concluir que la selección del algoritmo de ML depende de la naturaleza de los datos, pero es necesario la realización de estudios experimentales que permitan tomar una decisión al respecto a la selección. Por lo anteriormente planteado se propone que cada uno de estos algoritmos sean analizados en la propuesta de solución con el fin de realizar una comparación con los resultados obtenidos en la experimentación, para tomar una decisión con el objetivo de obtener los mejores resultados (Dietterich, 2000). La etapa de prueba del proceso de aprendizaje, anterior a la realización de las comparaciones de los diferentes resultados arrojados por los clasificadores. En esta etapa, es necesario el estudio de la evaluación del clasificador, por tanto en la próxima sección se realiza una análisis de los diferentes métodos de evaluación y las métricas utilizadas en la literatura.

1.5. Métodos de evaluación

Como anteriormente se mencionó en el acápite 1.4.2, el conjunto de entrenamiento se usa para aprender un clasificador y el conjunto de prueba se usa para evaluar el clasificador. El conjunto de entrenamiento no debe usarse en la evaluación ya que el clasificador está sesgado hacia el conjunto de entrenamiento, es decir, el clasificador puede **sobreajustar** los datos de entrenamiento, lo que da como resultado una precisión muy alta en el conjunto de entrenamiento pero baja precisión en el conjunto de prueba. El uso del conjunto de pruebas no vistas proporciona una estimación imparcial de la precisión de la clasificación (Voyant y col., 2017). Los principales métodos identificados en la literatura para dividir D en conjuntos de entrenamiento y prueba son los siguientes (B. Liu, 2007):

1. **Muestreo aleatorio:** del conjunto de entrenamiento D escoge un subconjunto de manera aleatoria para aprender y el resto para probar.
2. **Antes y después:** si los datos se recopilan a lo largo del tiempo, se pueden dividir en dos momentos: antes y después, la parte anterior de los datos para el aprendizaje y la parte posterior de los datos para la prueba.
3. **Muestreo aleatorio múltiple:** cuando el conjunto de datos disponibles es pequeño, el uso de los métodos anteriores puede no ser confiable porque el conjunto de prueba sería demasiado pequeño para ser representativo. Un enfoque para tratar el problema es realizar el muestreo aleatorio anterior n veces. Cada vez que se produce un conjunto de entrenamiento diferente y un conjunto de prueba diferente. Esto produce n precisiones y la precisión estimada final en los datos es el promedio de las n precisiones.

4. **Cross Validation (Validación cruzada):** cuando el conjunto de datos es pequeño, el método de validación cruzada de $n - fold$ se usa con mucha frecuencia. En este método, los datos disponibles se dividen en n subconjuntos disjuntos de igual tamaño. Cada subconjunto se usa luego como el conjunto de prueba y los subconjuntos $n - 1$ restantes se combinan como el conjunto de entrenamiento. Este procedimiento se ejecuta $n - fold$, lo que da n precisiones. La exactitud estimada final de aprendizaje de este conjunto de datos es el promedio de las n precisiones.

Para escoger el método de evaluación es necesario analizar la naturaleza de la información disponible como fuente de información. Aunque la cantidad de datos disponibles es relativamente grande solamente se tendrá en cuenta los logs de un día de navegación, además, debido al desbalance de los datos y las transformaciones definidas en el acápite 1.3 como parte del procesamiento dicha dimensionalidad se verá reducida.

Debido a que a priori no es posible conocer cuánto sera esta variación se decide utilizar Cross Validation como método de evaluación. Este enfoque puede ser computacionalmente costoso, pero como fija un conjunto de prueba arbitrario esto propicia que se disminuye su tiempo de cómputo.

Además, del método de evaluación también es necesario identificar las métricas asociadas a la clasificación. Con este propósito en el próximo acápite se estudiarán las métricas adecuadas para poder analizar la hipótesis planteada en la presente investigación.

1.5.1. Métricas de evaluación

Uno de los aspectos que se tiene en cuenta el proceso de evaluación de un clasificador es su precisión, el cual se encuentra relacionado con el porcentaje de casos clasificados correctamente (ecuación 1.1). Para ello es necesario definir ciertos criterios de comparación, y los resultados obtenidos dependerán en gran medida de dos aspectos, el método de clasificación seleccionado y el conjunto de datos disponible para entrenar el clasificador, teniendo en cuenta el grado de representatividad que tengan estos últimos sobre los datos reales.

En este contexto, aparecen varios procedimientos para seleccionar de los datos disponibles, los datos que son de entrenamiento y los que son de prueba, aspecto al que se refieren los métodos de evaluación. Por otra parte, se encuentran las diferentes formas de representar cuantitativamente la precisión de un clasificador, las cuáles se denominan métricas de evaluación.

TABLA 1.2: Matriz de confusión.

	Predicción positivos C_P	Predicción negativos C_N
Clase real positivos C_P	TP	FN
Clase real negativos C_N	FP	TN

De forma general las métricas de evaluación están basadas en los cálculos generados a partir de la matriz de confusión, tabla 1.2. La matriz o tabla de confusión expone los resultados de un procedimiento de clasificación o predicción. Si hay k categorías posibles, es una matriz kxk donde las filas denotan la verdadera categoría y las columnas indican la categoría a la que se asigna después de la predicción. El elemento ij define el número de muestras de la categoría i que fueron clasificados como categoría j, representando así los errores en la asignación de categorías a los patrones actuales.

Aquellas entradas en la diagonal de la matriz son correctas. Las entradas fuera de la diagonal son errores de clasificación y representan «confusión». Con la matriz es fácil ver si el sistema confunde dos categorías. La clase que le interesa al usuario se denomina comúnmente clase positiva y el resto, clase (s) negativa (s).

Las cuatro posibilidades que pueden resultar de una predicción cuando hay dos categorías «positivo o sí» y «negativo o no» son:

- Si el caso es positivo y es clasificado como positivo por el clasificador se denomina Verdadero Positivo (TP [7]).
- Si el caso es negativo y es clasificado como positivo se denomina Falso Positivo
- (FP [8]), o Error de Tipo I.
- Si el caso es negativo y es clasificado como negativo se denomina Verdadero Negativo (TN[9]).
- Si el caso es positivo y es clasificado como negativo se denomina Falso Negativo (FN [10]), o Error de Tipo II

Las principales métricas definidas a partir de la matriz de confusión que son utilizadas para evaluar los clasificadores son las siguientes:

[7]Del inglés True Positive
[8]Del inglés False Positive
[9]Del inglés True Negative
[10]Del inglés False Negative

- Accuracy (exactitud) describe la cantidad de clasificaciones correctas con respecto al total. Para realizar el cálculo de la misma se utiliza la ecuación siguiente:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.1}$$

- Precision (precisión) del clasificador, se corresponde con la proporción de casos positivos predichos que son correctos. Se calcula utilizando la ecuación:

$$precision = \frac{TP}{TP + FP} \tag{1.2}$$

  Cabe señalar que esta métrica puede no ser una medida adecuada para evaluar el comportamiento cuando el número de casos de un tipo es mucho mayor que el número de casos contrario, ya que este índice es global y no indica cómo la exactitud se reparte entre las diversas categorías individuales.

- Recall (exhaustividad) define cuán completa es la clasificación en la clase positiva. La misma se calcula mediante la siguiente ecuación:

$$recall = \frac{TP}{TP + FN} \tag{1.3}$$

  Esta métrica está estrechamente relacionada con la presición del clasificador, ya que permite tener una idea más completa acerca del clasificador.

- Area Under Curve (Área bajo la curva): Es una forma de evaluar el rendimiento de un clasificador es utilizando el área bajo las curvas ROC [11], donde se realiza una representación gráfica de la razón de verdaderos positivos (rTP), frente a la razón de falsos positivos (rFP), como ejes X e Y respectivamente. Dado que la rTP es equivalente a sensibilidad y larFP lo es a $(1 - especificidad)$, el gráfico ROC se define también como la representación de $(1 - especificidad)$ frente a la sensibilidad.

- f1_score en estadística es la medida de precisión que tiene un test. Se emplea en la determinación de un valor único ponderado de la presición y la exhaustividad. La misma se calcula mediante la siguiente fórmula:

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} \tag{1.4}$$

  El f1_score se considera como una media armónica que combina los valores de la precisión y de la exhaustividad y permite tener una idea general del desempeño del clasificador.

- Coeficiente Kappa de Cohen, es una medida estadística que ajusta el efecto del azar en la proporción de la concordancia observada. Esta métrica es calculada mediante la fórmula siguiente:

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \tag{1.5}$$

[11] Del inglés, Receiver Operator Characteristic

Donde $Pr(a)$ es el acuerdo observado relativo entre los observadores, y $Pr(e)$ es la probabilidad hipotética de acuerdo por azar, utilizando los datos observados para calcular las probabilidades de que cada observador clasifique aleatoriamente cada categoría. Si los evaluadores están completamente de acuerdo, entonces $\kappa = 1$. Si no hay acuerdo entre los calificadores distinto al que cabría esperar por azar, según lo definido por $Pr(e)), \kappa = 0$.

Cada una de estas medidas según (Benavoli y col., 2017), son efectivas para realizar una completa evaluación de la clasificación ya que tienen en cuenta los diferentes elementos que son necesario analizar en la clasificación. Por este motivo cada una de estas métricas serán utilizadas en la presente investigación para la evaluación de los clasificadores propuestos.

1.6. Herramientas y bibliotecas para la clasificación

Son varias las herramientas y bibliotecas utilizadas en el campo del aprendizaje automático y en la clasificación automática. A continuación, se analizarán las principales teniendo en cuenta el soporte al proceso de clasificación, flexibilidad de uso y comunicación con otros sistemas a partir de las característica de cada una de ellas.

1.6.1. MATLAB

MATLAB es una herramienta de programación cuyo nombre proviene de la abreviación de «MATrix LABoratory». Esta herramienta permite realizar cálculos numéricos con vectores y matrices. Como caso particular puede trabajar con números escalares reales o complejos, con cadenas de caracteres y con otras estructuras de información más complejas (M. official, 2018). Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones que permite el análisis de lo datos.

MATLAB es un programa de cálculo técnico y científico (M. official, 2018). Dispone de un código básico y de varias bibliotecas especializadas, en el caso específico de la clasificación se utiliza Math, Statistics y Optimization. Entre las principales limitantes para su utilización en la presente investigación se encuentran:

1. Es un producto propietario.
2. Es indispensable el uso de MCR[12] para que los archivos MATLAB funcionen correctamente.

[12]Del inglés, MATLAB Component Runtime

Debido a las limitantes anteriormente mencionadas en especial a la característica de poseer licencia propietaria no es aconsejable su uso como herramienta para el desarrollado de la propuesta de solución.

1.6.2. Weka

El sitio oficial de Weka (W. official, 2018), describe la herramienta como una colección de algoritmos de aprendizaje automático para la tarea de extracción de datos. Es un software libre distribuido bajo licencia GNU-GPL desarrollado en la Universidad de Waikato. Los algoritmos pueden ser aplicados directamente a un conjunto de datos o ser llamados desde código Java. Contiene herramientas para el pre-procesamiento, clasificación, regresión, clustering, reglas de asociación y visualización de datos.

Permiten personalizaciones para el código mediante el uso del lenguaje Java (W. official, 2018). La interfaz gráfica de Weka cuenta con cuatro formas de acceso a las diferentes funcionalidades de la aplicación, las cuáles son:

- Simple CLI [13], que permite el acceso a través de una consola de comandos a todas las opciones de Weka.
- Explorer, es la opción más intuitiva para nuevos usuario, pues dispone de varios paneles que dan acceso a las principales características del programa.
- Experimenter, permite la comparación sistemática de una ejecución de los algoritmos predictivos de Weka sobre una colección de conjuntos de datos.
- Knowledge Flow, soporta esencialmente las mismas opciones que la interfaz pero organizadas como un flujo.

Entre las principales limitantes del uso de Weka en la presente investigación según la bibliografía se encuentran (W. official, 2018):

1. Todas las técnicas fundamentan en la asunción de que el conjunto de entrenamiento está disponibles en un fichero de texto plano.
2. El conjunto de entrenamiento está descrito por un número fijo de atributos, normalmente numéricos o nominales aunque también se soportan otros tipos.
3. Poca flexibilidad para ajustar los parámetros y los algoritmos de manera general.
4. Posee pocas interfaces con otros lenguajes de programación excepto Java.
5. No es aconsejable su uso en conjunto de entrenamientos grandes.

Weka es una herramienta para familiarizarse con el aprendizaje automático en un contexto académico. Pero en la práctica puede ser bastante inflexible limitando su uso en las soluciones debido a las limitantes anteriormente mencionadas.

1.6.3. Software estadístico R

R es un lenguaje y entorno de programación para análisis estadístico y gráfico (R. official, 2018). Se trata de un proyecto de software libre, resultado de la implementación GNU del premiado lenguaje S. Se trata de uno de los lenguajes más utilizados en investigación por la comunidad estadística. A esto contribuye la posibilidad de cargar diferentes bibliotecas o paquetes con funcionalidades de cálculo y gráficas. A pesar de las flexibilidades que brinda por el hecho de ser un lenguaje de programación no está libre de limitantes a continuación se enumeran:

1. La mayor desventaja de R es que es más difícil de mantener si su código crece.
2. En difícil llevar a cabo las reglas impuestas por el lenguaje.
3. R es extremadamente flexible en su sintaxis, pero para el mantenimiento y/o para trabajar en equipo puede crear confusiones y provocar errores.
4. El proceso de optimización del código es complejo lo que lo hace relativamente lento.

Debido a las limitantes anteriormente mencionadas para el desarrollo con este lenguaje no se hará uso del mismo como herramienta para el desarrollado de la propuesta de solución.

1.6.4. Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Al igual que R brinda las facilidades de ser un lenguaje de programación, aunque Python en de propósito general, no solo académico (*Python.Org*, 2018). La mayoría de las bibliotecas de procesamiento utilizan las bibliotecas C o Fortran, aumentando la velocidad de las soluciones. Es fácil de aprender debido a su sintaxis constante y la forma en que refleja el lenguaje humano y/o sus equivalentes matemáticos.

La amplia cantidad de bibliotecas especializadas en función matemática y optimización lo convierten actualmente, en una herramienta muy atractiva para la solución de problemas asociados con la ciencia de datos y el ML en general. Debido a estas característica en la presente investigación se utilizará para la implementación de la solución el lenguaje de programación Python. Ya que el mismo cuenta con herramientas que le permiten lograr un equilibrio entre soluciones prácticas y

académicas. Además, es ampliamente reconocido en el ámbito de la ciencia de datos, la minería de datos y el aprendizaje automático.

Dentro de las bibliotecas de Python más importantes dentro del aprendizaje automático propuestas en la literatura se encuentran (Keita, 2017; Ferreira y col., 2017):

1. NumPy: especializada en cálculos matemáticos para la computación científica. Contiene entre otras funcionalidades: trabajo robusto con matrices N-dimensional, funciones sophisticated (broadcasting), herramientas para integrar el código C/C ++ y Fortran, álgebra lineal útil, transformada de Fourier y generación de números aleatorios, contenedor multidimensional de datos genérico y se pueden definir tipos de datos arbitrarios (*NumPy — NumPy*, 2018).

2. Pandas: es una biblioteca escrita en el lenguaje de programación Python para la manipulación de datos y el análisis. Entre las principales funcionalidades se encuentran (*Python Data Analysis Library — Pandas: Python Data Analysis Library*, 2018): herramientas para leer y escribir datos entre estructuras de datos en memoria y diferentes formatos: CSV, archivos de texto y bases de datos SQL; manipular con facilidad los datos desordenados en una forma ordenada; Combinación y unión de conjuntos de datos de alto rendimiento; optimizado para el rendimiento, con rutas de código críticas escritas en Cython o C.

3. SciKit-Learn: ofrece herramientas versátiles para el aprendizaje automático y análisis en cualquier campo de la ciencia y la ingeniería. Cuenta con varios algoritmos de clasificación, regresión y agrupación. Está diseñado para interoperar con las bibliotecas numéricas y científicas de Python, NumPy y SciPy. Además, poseé implementación de los algoritmos de ML más utilizados actualmente.

4. SciPy: es una biblioteca para las matemáticas, la ciencia y la ingeniería para Python. Contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, procesamiento de señal e imagen y otras tareas comunes en ciencia e ingeniería. Además, de un amplio conjunto de métodos estadísticos paramétricos y no-paramétricos.

5. Matplotlib: es una biblioteca de trazado 2D python que produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos a través de plataformas. Además, exportar los mismos en diversos formatos para su posterior utilización.

6. Imbalanced-learn: es una biblioteca para el trabajo con datos desbalanceados, se integra con las principales bibliotecas destinadas al aprendizaje automático en python. Además, sigue el mismo flujo de implementación que SciKit-Learn, facilitando la integración entre las mismas.

7. Jupyter: es una biblioteca que empaqueta diferentes medios en una solución de problemas: codifica, escribe y visualiza. Por estas características permite que el trabajo sea mas amigable, además, simplifica el intercambio de dicho trabajo, tanto para fines educativos como colaborativos.

8. Joblib: es una biblioteca que permite el almacenamiento e interacción de diferentes archivos. Es muy utilizado para exportar los modelos ya entrenados, así como los datos utilizados.

9. Hashlib: es una biblioteca que permite el uso y manipulación de funciones *hash* mediante el uso de los algoritmos más utilizados.

Python junto al conjunto de bibliotecas analizadas anteriormente hace que sea una buena opción para la realización de una solución de alta complejidad con un buen rendimiento utilizando un conjunto de datos grande. La decisión de utilizar para las implementaciones las herramientas anteriormente mencionadas partió del análisis que se realizó de las principales herramientas software de código abierto.

Los principales motivos de la selección son: incluyen una colección de algoritmos de aprendizaje automático dentro de los que se encuentran los identificados en los acápites de 1.4. Además, de brindar facilidades para el trabajo con conjuntos de entrenamientos, preprocesamiento, herramientas para la evaluación y la flexibilidad de incluir nuevos algoritmos o transformaciones en caso de ser necesarias utilizando el lenguaje.

Conclusiones parciales

- Después de un análisis del estado del arte sobre la clasificación de páginas Web se puede concluir que las técnicas de aprendizaje automático son las más utilizadas y la selección del algoritmo a utilizar está estrechamente relacionado con la naturaleza de los datos.
- Según las fuentes bibliográficas consultadas, la diversidad de algoritmos para la clasificación, impide llegar a un consenso único en cuál utilizar, por lo cual, se hace necesario el uso estudios experimentales para tomar una decisión al respecto.
- Con el fin de organizar el desarrollo de la investigación se hace uso del proceso de la minería Web que cuenta con tres pasos principales: preprocesamiento, descubrimiento de los patrones y análisis de los patrones.
- Después del estudio de las diferentes técnicas en la literatura se identificaron para su uso: Under-sampling para el trabajo con datos desbalanceados, *PCA* para la reducción de dimensionalidad, Cross Validation como métodos de evaluación, las métricas asociadas a la clasificación para la evaluación y Python como herramienta para la implementación de la solución.

Capítulo 2

Propuesta de solución

Introducción

En el presente capítulo se realiza un análisis del proceso de clasificación, así como, una breve descripción de los algoritmos a utilizar. Se realiza la descripción de las dos primeras etapas que proponen el proceso de la minería Web: preprocesamiento y descubrimiento de patrones. Se describen las características de la solución propuesta y su arquitectura.

En la fase de preprocesamiento se abordan los temas de la construcción del conjunto de entrenamiento, la limpieza y escalado de los datos, la reducción de la dimensionalidad y el desbalance de los mismos. En la fase de descubrimiento de patrones se analizan los algoritmos de clasificación seleccionados de acuerdo a su comportamiento interno, con el fin de encontrar parámetros eficientes en la clasificación de páginas Web.

2.1. Descripción de la propuesta de solución

La solución tiene como objetivo clasificar las nuevas páginas Web de acuerdo con el interés de la universidad. En función de lo anterior, la propuesta está encaminada a brindar una herramienta útil que permita la clasificación utilizando los patrones de navegación de los usuarios.

Dentro de los elementos analizados en el acápite 1.1.1, un elemento importante a tener en cuenta es las páginas las páginas Web no se encuentran clasificadas, pero se conoce cuáles son los dominios de interés. Por tanto, se puede saber cuales son páginas Web de interés y se asume que el resto no lo son. Por este motivo el primer paso dentro de la etapa de preprocesamiento será clasificar las páginas de acuerdo a la información con la que se cuenta. Uno de los problemas que pueden ocurrir es que puede existir un conjunto de páginas que se clasifiquen cómo que no son de interés, pero pudieran en realidad serlo. Identificar estás páginas Web es una de las aplicaciones en las que puede ser utilizados los clasificadores que resulten parte de esta investigación.

Cómo se analizó en el capítulo 1, en el acápite 1.2.2, se decidió utilizar el proceso de la minería Web; a partir de la descripción de la misma se definen las tres fases de la propuesta de solución:

- Fase de preprocesamiento: Se tendrán en cuenta los elementos que permitan la unificación de la fuente de información, así como la construcción del conjunto de entrenamiento. Posteriormente se realiza el escalamiento de los datos con el fin de eliminar los valores anormales que pueden influir negativamente en el aprendizaje de los algoritmos en la próxima fase. También se realiza la reducción de la dimensionalidad del conjunto de entrenamiento mediante la extracción de características y el algoritmo PCA definidos en el acápite 1.3.1. Y por último se analiza el desbalance de los datos utilizando las técnicas definidas en el acápite 1.3.2.
- Fase de descubrimiento de patrones: En este paso se realiza el análisis de los parámetros de los algoritmos identificados en acápites 1.4.3 y 1.4.4. Se describe la implementación de dichos algoritmos de aprendizaje y se realiza el entrenamiento de los clasificadores.
- Fase de análisis de patrones: Esta es la última fase en ella se evalúan los algoritmos utilizando las métricas definidas en el acápite 1.5.1. Además, se realiza el análisis estadístico de las métricas aplicadas con el fin de revisar los resultados de las métricas y se visualizan los resultados obtenidos.

Cada una de las fases anteriores se realizan en ese orden, donde la entrada de cada fase es la salida de la fase anterior. Para la implementación de cada una de estas fases se hace uso de las herramientas identificadas en el acápite 1.6. Además, se utiliza la función *sklearn.pipeline* propuesta por la librería $scikit - lean$, ya que esta permite que la salida de cada paso sea la entrada del próximo.

Para la implementación de la solución y del *pipeline*, se utilizó los aspectos relevantes para el desarrollo de soluciones de aprendizaje automático propuesto por (Zinkevich, 2017). Entre las que se puede destacar: diseñar el *pipeline* de manera que sea sólido de un extremo a otro; comenzar con objetivo razonable y pequeños; agregar nuevas funciones en caso de que sea necesario y por último, asegurar que las nuevas funcionalidades no afecten la estabilidad del *pipeline*.

2.2. Preprocesamiento

La fase de preprocesamiento es fundamental para el éxito de las fases posteriores, ya que sienta las bases para lograr un rendimiento adecuado en los algoritmos de ML. En la solución, la información utilizada proviene como se mencionó anteriormente de los logs histórico de navegación de los usuarios y además, del listado de dominios de interés definidos en la universidad.

Para el presente trabajo, con el fin de disminuir la complejidad computacional de la solución se utilizó cuenta los logs provenientes de un día de navegación; los cuales se encuentran almacenados en un archivo de formato texto plano, que utilizan espacio y el fin de línea como separadores. Los archivos utilizados se describen a continuación:

- Logs histórico de navegación: Archivo en texto plano que contiene los datos la navegación de los usuarios, entre los que se encuentran: URL, cantidad de peticiones, cantidad de datos, ip de origen y el usuario que las realiza. En el mismo se utiliza el fin de linea para separar cada una de las entradas y el espacio para separar cada uno de los datos.
- Dominios de interés: Archivo de texto plano que contiene el listado de los dominios de interés para la universidad. En el mismo cada entrada es un dominio y utilizan el fin de linea como separador.

En el caso de los logs del histórico de navegación, cuenta con varios elementos los cuáles son descritos en la tabla 2.1. En la tabla se realiza la descripción de cada una de los elementos a tener en cuanta para la construcción del conjunto de entrenamiento. En cuanto al archivo de los dominios de interés, solo cuenta con un campo, el dominio en sí.

TABLA 2.1: Descripción de las variables de los logs histórico de navegación de un día.

Nombre de la variable	Descripción de la variable
ts	Timestamp[5], instante de tiempo en que se realiza la petición
hits	Cantidad de peticiones realizadas
ip	Dirección ip del origen de la petición
tcp_resp	Respuesta que ofrece el protocolo TCP [6] a la petición
size	Tamaño de la respuesta de la petición
http_method	Método HTTP [7], que utiliza la petición
url	URL a la que se le realiza la petición
username	Usuario que realiza la petición
destiny	Destino de la petición
mimetype	MIME type [8], tipo de la respuesta de la petición.

A partir de la información disponible hay varios elementos a tener cuenta en la fase de preprocesamiento, para ello se definieron varios pasos con el objetivo de organizarlos. Para su definición se tuvo en cuenta los elementos planteado en el acápite 1.3 y las características de la información. A continuación, se presentan los pasos definidos para llevar a cabo dentro de la fase:

- Paso 1: Construcción del conjunto de entrenamiento
- Paso 2: Limpieza y escalado de los datos
- Paso 3: Extracción de características
- Paso 4: Datos desbalanceados

Cada uno de los pasos identificados anteriormente son descritos a continuación, con el objetivo de mostrar su uso en la solución propuesta.

2.2.1. Paso 1: Construcción del conjunto de entrenamiento

Para construir el conjunto de entrenamiento es importar la información de los logs del histórico de navegación para ello se utiliza la función *read_csv* de la biblioteca *Pandas* ya que con la misma se puede convertir la información obtenida en un $dataframe$ el cuál se encuentra optimizado para el trabajo con las bibliotecas $scikit-learn$ y $numpy$.

De la información disponible se tendrá en cuenta solamente la URL y los usuarios, ya que con ambos se realiza el cálculo de la relación entre los usuarios y las páginas Web. El resto de los elementos no serán utilizados ya que su almacenamiento afectaría el rendimiento de la solución.

Otro punto a tener en cuenta es que es necesario anonimizar los usuario, en este paso, con el fin de respetar la privacidad de los mismos; para ello se hace uso de la función hash $MD5$[9]

Una de las características de la fuente de información es que está compuesto por más de un archivo, cómo se mencionó en el acápite anterior y cada uno de ellos brinda información indispensable para solucionar el problema. En este paso, primero se realiza la unificación de la fuente de información, para ello es necesario buscar cuál es el elemento común de ambas fuentes y a partir de este elemento realizar la unificación, es este caso los elementos de unión es la URL.

Una URL es la cadena de caracteres con la cuál se asigna una dirección única a cada uno de los recursos de información disponibles en la Web (Sanagavarapu y col., 2018). La primera parte de la dirección indica qué protocolo utilizar, la segunda parte específica la dirección IP o nombre de dominio donde se localiza el recurso. Por otra parte, el dominio es una etiqueta de identificación asociada a un grupo de dispositivos o equipos conectados en la Web.

En el caso de los logs de navegación se cuenta con la URL de las peticiones y esta incluye el dominio. Para ello construyó una expresión regular a partir de la cual se extrae el dominio de cada una de las instancias. Mediante este proceso ocurre una reducción de la cantidad de instancias de la fuente de información y a partir de ella se puede hacer la construcción de la variable objetivo.

Para la extracción el dominio se utilizó la biblioteca $URLlib.parse$ perteneciente al estándar de Python, que es el lenguaje seleccionado en el acápite 1.6.4. En la figura 2.1, se puede observar la fuente de información inicial después de las transformaciones mencionadas anteriormente.

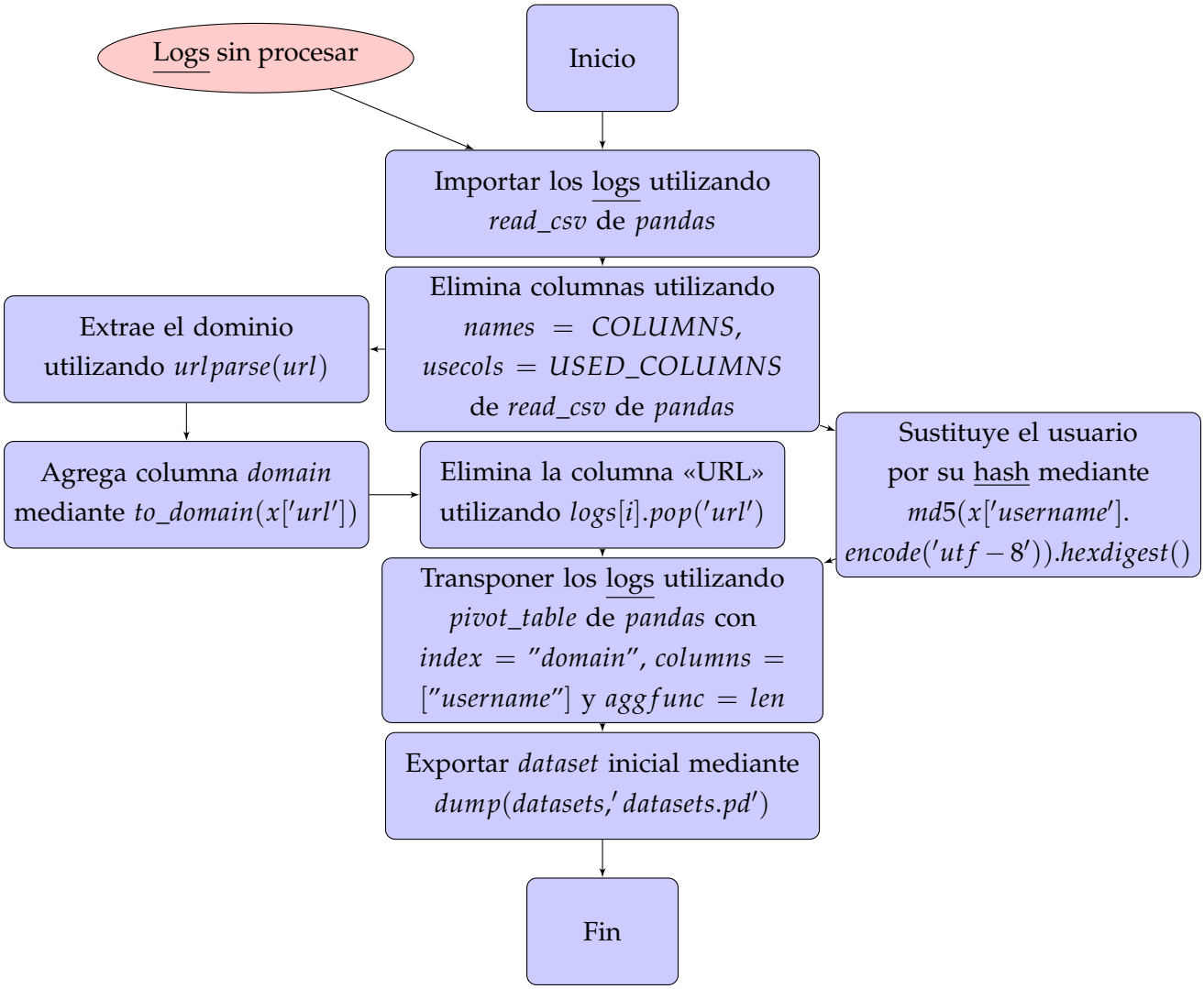
[9]En criptografía, $MD5$ (abreviatura de Message-Digest Algorithm 5 es un algoritmo de reducción criptográfico de 128 bits ampliamente utilizado.

FIGURA 2.1: Fuente de información inicial (Elaboración propia)

	username	domain
0	336d5ebc5436534e61d16e63ddfca327	s-media-cache-ak0.pinimg.com
1	336d5ebc5436534e61d16e63ddfca327	s-media-cache-ak0.pinimg.com
2	336d5ebc5436534e61d16e63ddfca327	s-media-cache-ak0.pinimg.com
3	64aebbf14d39e7ba378dc7077f2a6e84	clients1.google.com.cu
4	1881cda74e95358d2a7ddde989c092af	reverse1.upr.edu.cu

Por último, es necesario saber cuál es la cantidad de visitas que realiza cada usuario a un dominio en específico, para ello se utiliza la función *pivot_table*, también de *Pandas*, que permite transponer una matriz utilizando la columna *domain* como índice y la columna *username* como columnas.

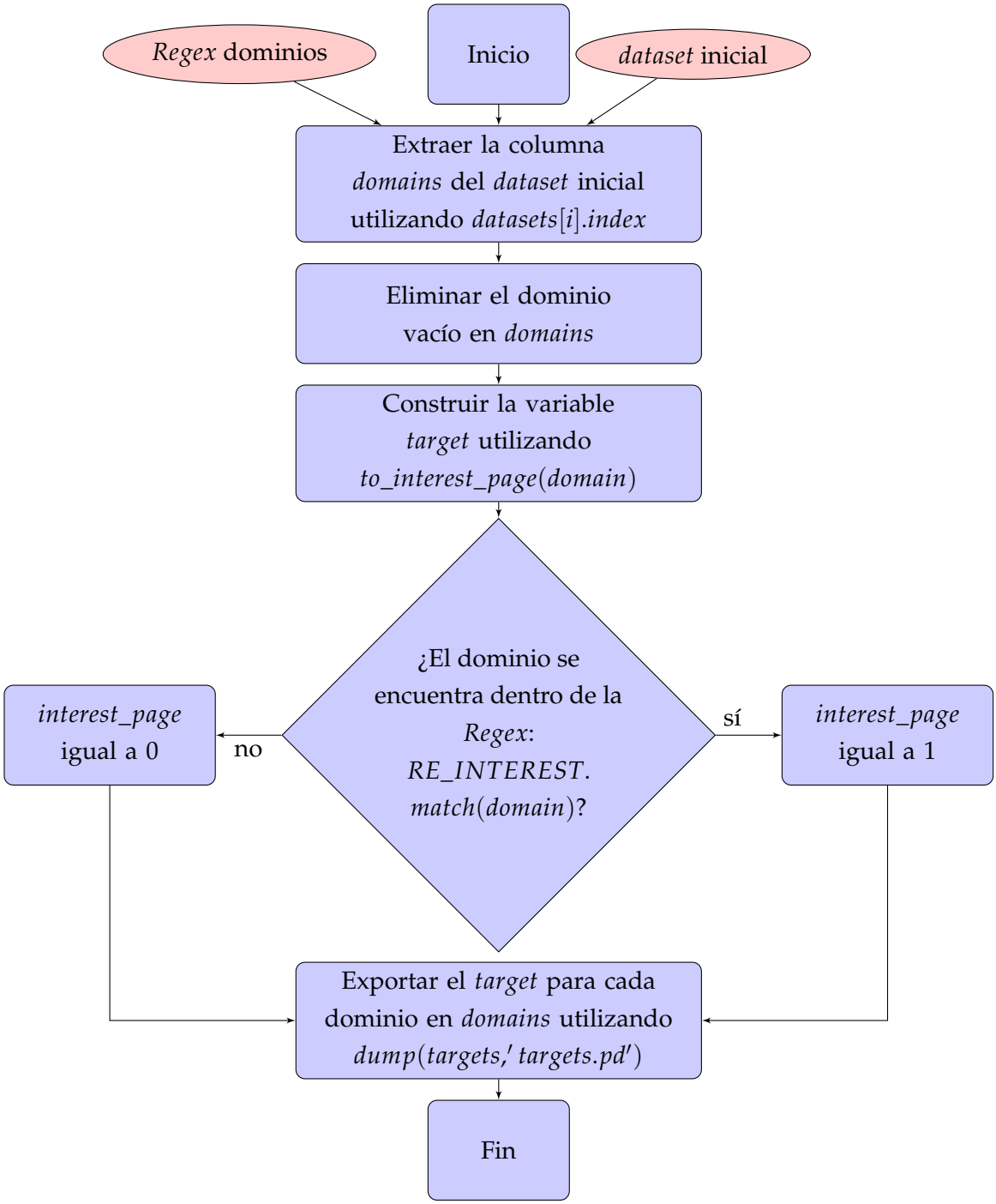
FIGURA 2.2: Construcción del conjunto de entrenamiento (Elaboración propia).



En la figura 2.2, se muestra el diagrama de flujo llevado a cabo en este paso para la obtención del conjunto de entrenamiento que se utilizará posteriormente. Además al conjunto de entrenamiento obtenido es necesario agregar la variable objetivo.

Para ello se construye una nueva variable denominada *interest_page*, la cuál, está relacionada con los dominios de interés como se mencionó anteriormente. Esta variable tendrá un valor binario, el mismo será $(1/s)$ si es una página de interés o $(0/no)$ si no lo es. El proceso llevado a cabo para la construcción de la variable, se puede observar en la figura 2.3.

FIGURA 2.3: Construcción de la variable objetivo (Elaboración propia).



2.2.2. Paso 2: Limpieza y escalado de los datos

Después del paso 1 donde se obtuvo *dataset* y *target*, le corresponde el paso 2 en el cual se realiza la limpieza y escalado de los datos. En este paso, las entradas del mismo son las dos obtenidas en el paso anterior. Cómo se analizó en el acápite 1.3, se utiliza la normalización en el proceso de escalado, ya que esta se realiza de modo que todas las variables de entrada tengan el mismo tratamiento en el modelo y los coeficientes de un modelo no se escalen con respecto a las unidades de las entradas.

La normalización se le realiza al *dataset* solamente, ya que el *target* contiene valores binarios por lo que no es necesario. Para ello se utilizará *sklearn.preprocessing.Normalizer* de la biblioteca $scikit - learn$, con la misma,cada instancia (es decir, cada fila del *dataset*) con al menos un componente distinto de cero se escala independientemente de otras instancias, para que su norma sea igual a uno. La salida de este paso es un *dataset* normalizado el cual será utilizado en los pasos posteriores. El algoritmo seguido para la implementación de este paso se puede observar en la próxima sección, debido a que se utiliza el *pipeline* ambos pasos se realizan de manera consecutiva.

2.2.3. Paso 3: Extracción de características

Las características o atributos son componentes importantes en muchos métodos de aprendizaje automático. Lograr que dichas características sean representativas, por lo general, es una tarea importante, especialmente porque su reducción incide positivamente en los tiempos de ejecución obteniendo igual o mejor exactitud en la clasificación (Aytuğ Onan, 2016).

La extracción de características es una técnica de procesamiento de datos cuyo objetivo es buscar un subconjunto de características que mejore el rendimiento del clasificador (Fernández Hernández y col., 2016). En algunos casos las características son explícitas, sin embargo, en otros casos es necesario que sean definidas y extraídas antes de aplicar algoritmos de minería Web.

Las características o atributos como se analizó en el acápite 1.3.1, para la reducción de la dimensionalidad del *dataset* se hace uso de la extracción de características mediante el algoritmo PCA. Para ello se hace uso de biblioteca *sklearn.decomposition.PCA*, esta función permite la reducción de la dimensionalidad los datos para proyectarlo a un espacio dimensional inferior eliminado las características irrelevantes.

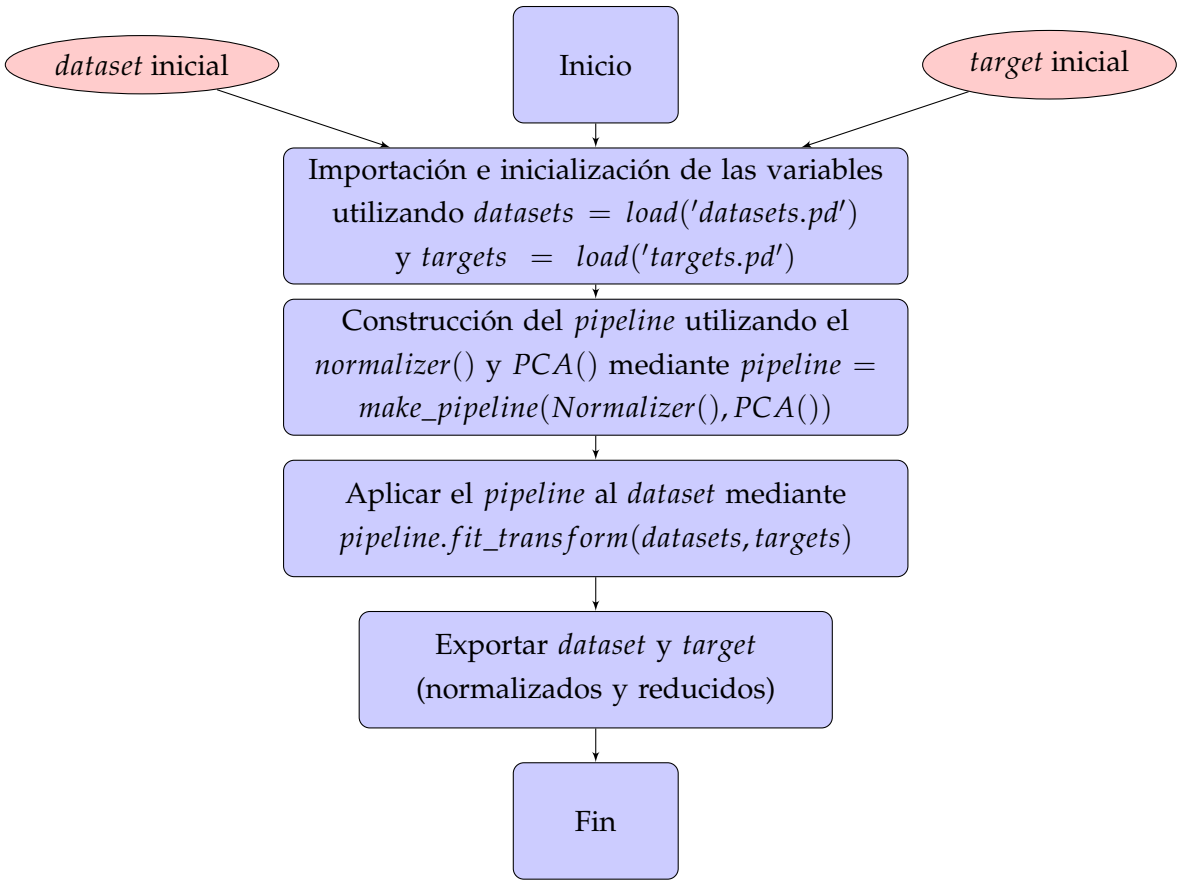
Entre las salidas que ofrece dicha función para realizar el análisis del comportamiento del mismo, según la documentación oficial de la biblioteca (*Scikit-Learn: Machine Learning in Python — Scikit-Learn 0.19.1 Documentation*, 2018), se encuentran:

- *n_components_*: Este atributo devuelve el número estimado de componentes después de ejecutado el algoritmo.

- *components_*: Ejes principales en el espacio característico, que representan la variación máxima en los datos, ordenados por *explanation_variance_*.
- *get_covariance*: Calcula la covarianza de los datos con el modelo generativo. El valor indica el grado de variación conjunta de dos variables aleatorias respecto a sus medias.

En la figura 2.4, se observa el diagrama de flujo seguido por el algoritmo para la reducción de la dimensionalidad utilizando PCA, precedido por la normalización de los datos como se mencionó en la sección anterior. La entrada del algoritmo es las salidas del paso anterior y la salida de este es un *dataset* donde las características han sido reducidas a los componentes que genera el modelo PCA.

FIGURA 2.4: Algoritmo para la reducción de la dimensionalidad utilizando PCA (Elaboración propia).

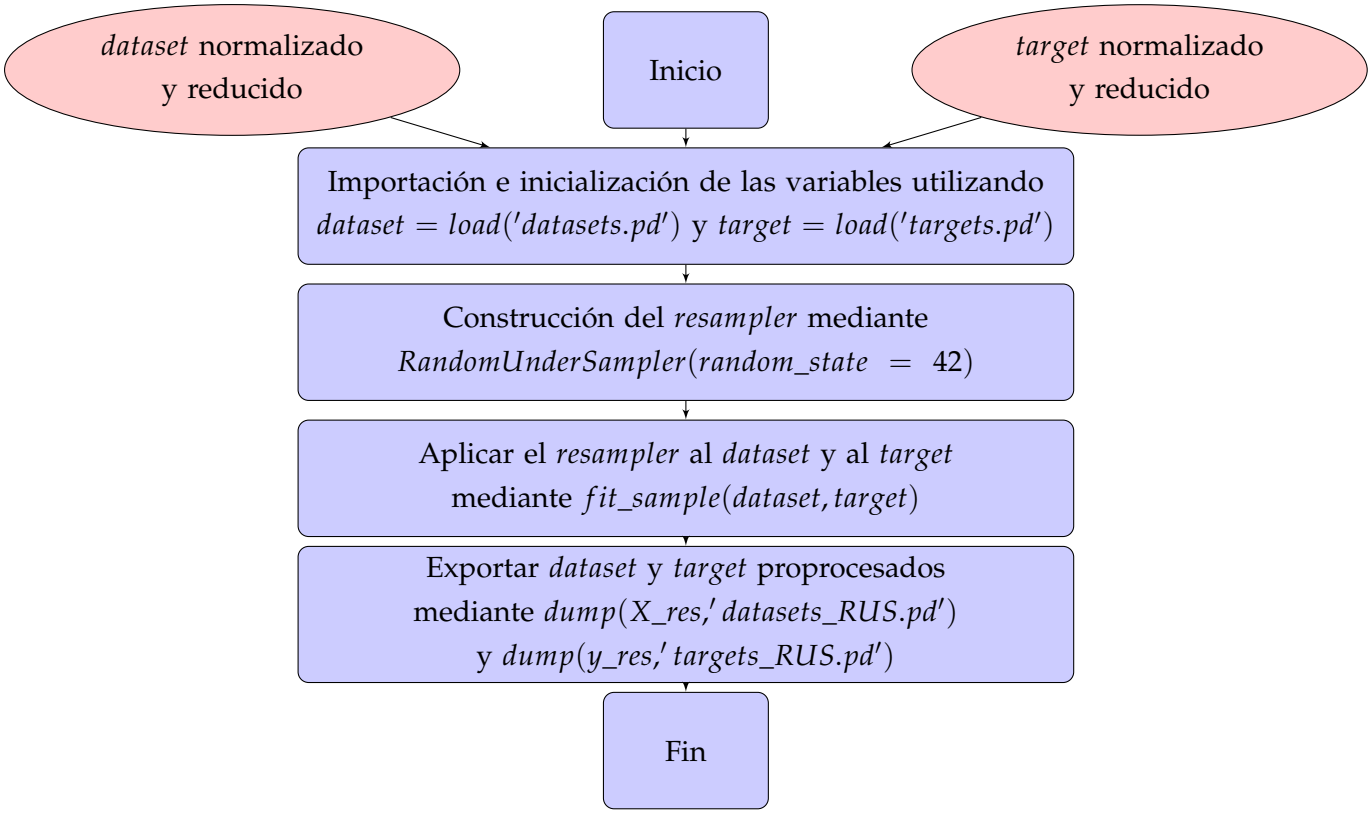


2.2.4. Paso 4: Datos desbalanceados

Para lograr el balance de las instancias del *dataset*, como se plantea en el acápite 1.3.2, se utiliza para la solución de la biblioteca $imbalanced - learn$, la cuál, se integra con la biblioteca $scikit - learn$ utilizando *pipeline*. Dentro de ella se utiliza el algoritmo *imblearn.under_sampling.RandomUnderSampler*, el mismo realiza un sub-muestreo aleatorio eliminando instancias de la clase mayoritaria. Como solamente se desea que se modifique la clase

mayoritaria se decidió utilizar el parámetro $ratio =' majority'$ propuesto en la biblioteca, que permite dicha funcionalidad. En la figura 2.5, se puede observar diagrama de flujo del algoritmo para el tratamiento de los datos desbalanceados, utilizando el algoritmo anteriormente mencionado.

FIGURA 2.5: Algoritmo para el tratamiento de datos desbalanceados (Elaboración propia).



En los subacápites de esta sección se analizaron los pasos que son necesarios para llevar a cabo la fase de preprocesamiento. Durante el mismo se definieron y caracterizaron las transformaciones necesarias con el fin de obtener el *dataset* y el *target* preprocesados, de manera que permita mejorar el rendimiento de los algoritmos de aprendizaje a utilizar en la próxima sección. En el anexo C, se puede observar la implementación de esta fase, utilizando las definiciones propuestas en esta sección, asi como las herramientas y bibliotecas definidas en el acápite 1.6.

2.3. Descubrimiento de patrones

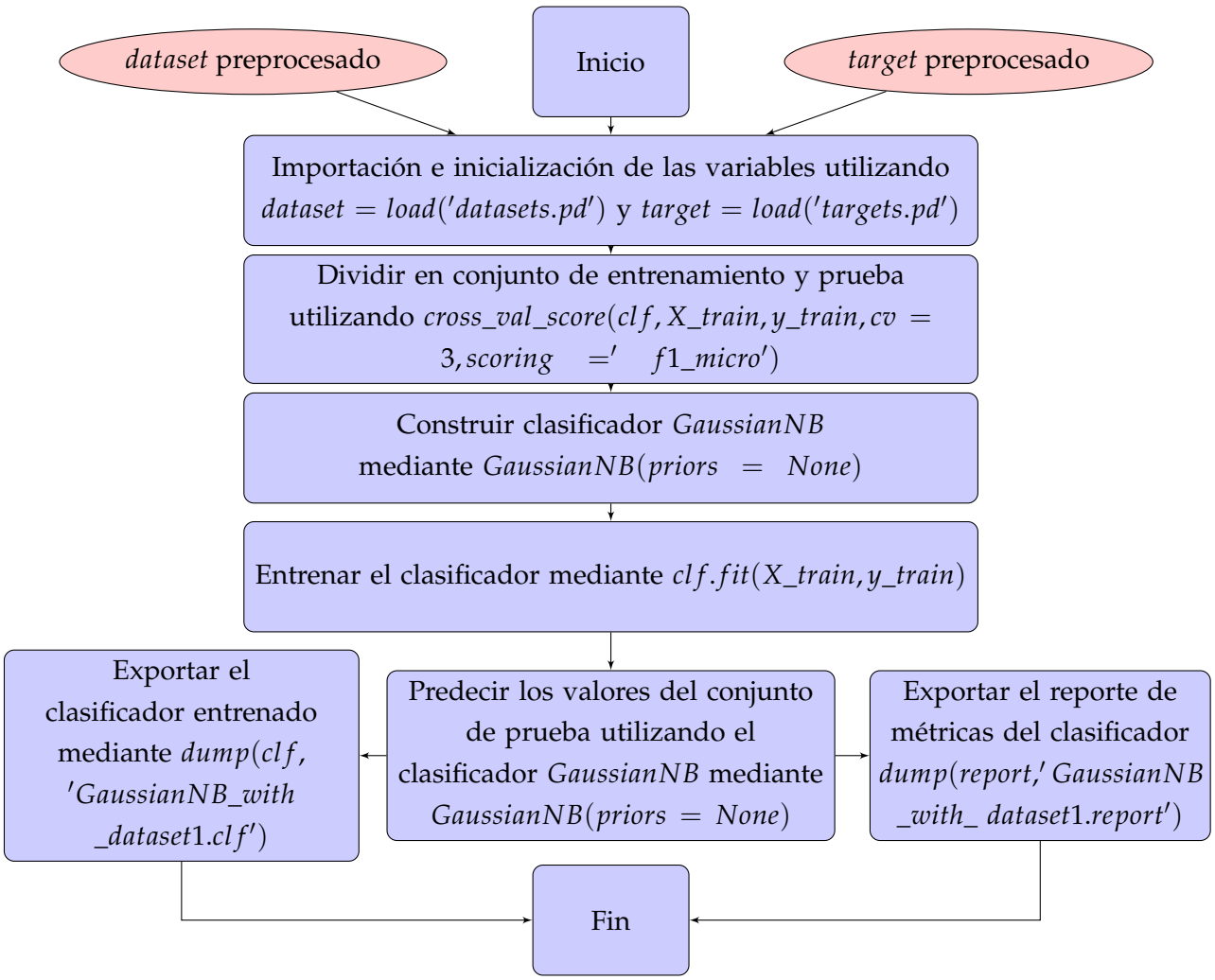
La segunda fase del proceso de la minería Web es el descubrimiento de patrones, en esta fase se buscar obtener patrones ocultos que reflejan el comportamiento típico de los usuarios (Adhvaryu, s.f.). Además, de como estos afectan el comportamiento de la clasificación de las páginas Web (Adeniyi y col., 2016).

Cómo se analizó en los acápites 1.4.3, 1.4.4, en esta fase se analizarán varios clasificadores, cuyas salidas posteriormente serán utilizadas posteriormente en la próxima fase: Análisis de los patrones. En el caso de la división del conjunto de entrenamiento para el aprendizaje de los clasificadores se utilizará *sklearn.model_selection.cross_val_score*. Esta función es la implementación de la biblioteca $scikit-learn$ para el método *Cross Validation* a utilizar en la solución como método de evaluación de los clasificadores, según lo definido en el acápite 1.5. A continuación, se realiza un análisis de los parámetros a utilizar en las técnicas definidas en dichos acápites, con el fin de obtener los mejores para la propuesta de solución.

2.3.1. *Gaussian Naive Bayes*

Los clasificadores bayesianos son un conjunto de algoritmos de aprendizaje supervisado basados en la aplicación del teorema de Bayes con la suposición de independencia entre cada par de características. Cómo se definió en el acápite 1.4.3, se utilizará *Naive Bayes* debido a su amplia aplicación, dentro de ellos *Gaussian Naive Bayes* es el más utilizado en la clasificación.

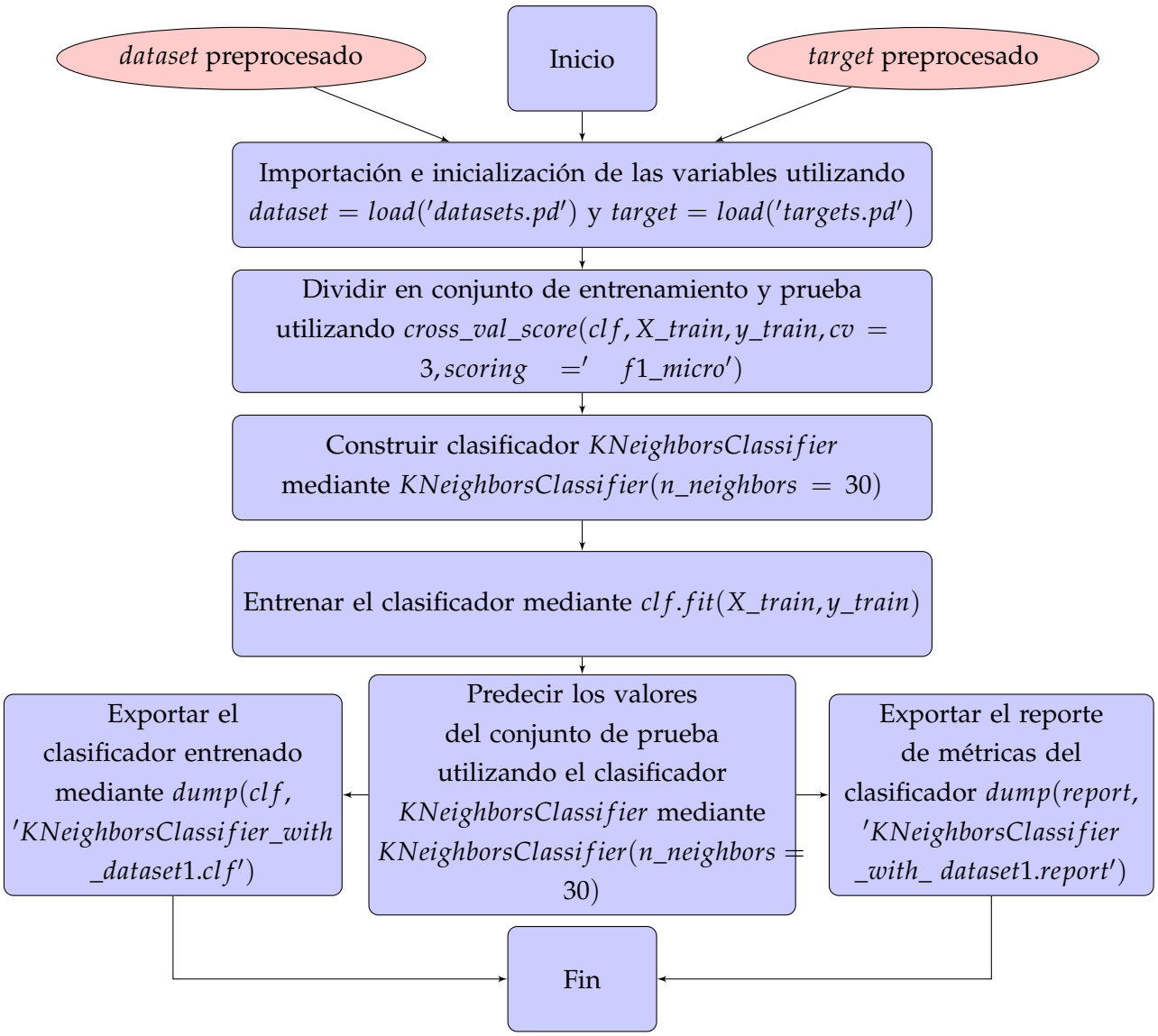
FIGURA 2.6: Algoritmo para utilizar el Gaussian Naive Bayes (Elaboración propia).



Por lo antes mencionado para la solución se utiliza *sklearn.naive_bayes.GaussianNB* de la biblioteca $scikit-learn$, esta biblioteca implementa el algoritmo *Gaussian Naive Bayes* optimizado para la clasificación. Como no existe ningún tipo de prioridad en el aprendizaje de las clases objetivo a la hora de realizar la clasificación se establece la variable $priors = None$, siendo este el único parámetro de la función. El diagrama de flujo que representa al algoritmo a seguir para el entrenamiento de del clasificador se puede observar en la figura 2.6.

2.3.2. *Nearest Neighbors*

FIGURA 2.7: El algoritmo para la clasificación utilizando el clasificador *Nearest Neighbors* (Elaboración propia).



El clasificador *Nearest Neighbors* es un tipo de aprendizaje basado en instancias o no generalizado: no intenta construir un modelo interno general, sino que simplemente almacena instancias de los

datos de capacitación. La clasificación se calcula a partir de un voto de mayoría simple de los vecinos más cercanos de cada punto: a un punto de consulta se le asigna la clase de datos que tiene la mayor cantidad de representantes dentro de los vecinos más cercanos del punto.

Dentro de esta familia de algoritmos, $KNeighborsClassifier$, es el método optimizado para su uso en la clasificación. En la solución se utiliza $sklearn.neighbors.KNeighborsClassifier$ de la biblioteca $scikit-learn$, en dicha implementación hay varios parámetros a tener en cuenta entre los que se encuentra la elección del valor k, o sea, la cantidad de vecinos. Este parámetro depende en gran medida de los datos: en general, una k más grande suprime los efectos del ruido, pero hace que los límites de la clasificación sean menos distintos. Para la elección de dicho valor se realiza un experimento utilizando $f1\_score$ como métrica de comparación y utilizando varios valores de k.

Otro de los parámetros está relacionado con la distribución de los pesos, debido a que no existe ningún estudio previo sobre la relevancia de las características se utiliza una distribución de pesos uniformes: es decir, el valor asignado a un punto de consulta se calcula a partir de un voto de mayoría simple de los vecinos más cercanos: $weights =' uniform'$.

Los demás parámetros del algoritmo es opcional su configuración y como no existe ninguna característica del problema que justifique su modificación, se utilizan los valores por predeterminado. En la figura 2.7, se observa diagrama de flujo del algoritmo utilizando el clasificador *Nearest Neighbors*.

2.3.3. Redes neuronales artificiales

Cómo se analizó en el acápite 1.4.3, dentro de las técnica de ANN se utiliza en la presente solución $sklearn.neural\_network.MLPClassifier$, que es la implementación de MLP para la clasificación dentro de la biblioteca $scikit-learn$. Este modelo optimiza la función de costos utilizando el gradiente de descenso estocástico, además utiliza el algoritmo $Backpropagation$ [10]para el aprendizaje.

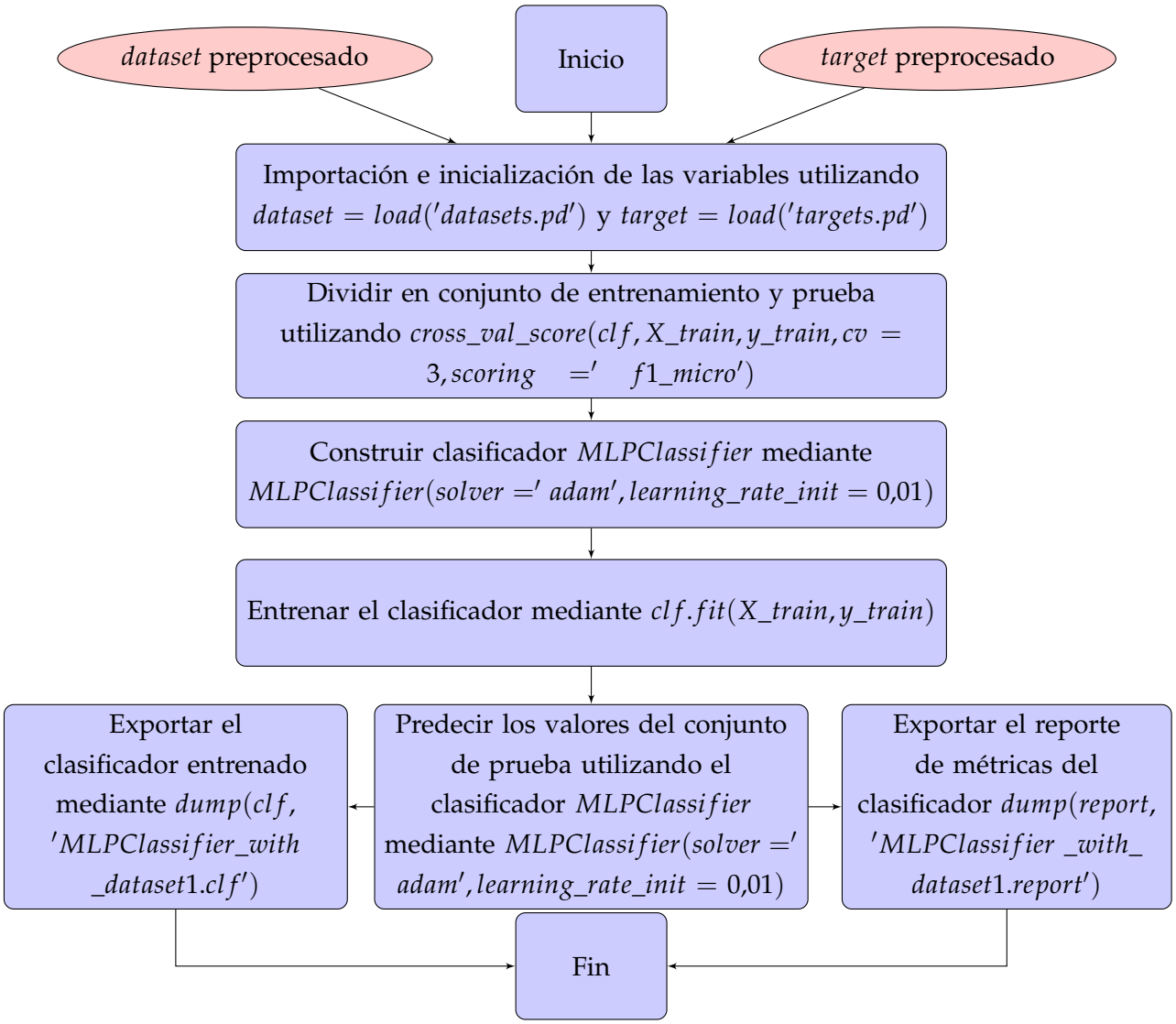
MLP con capas ocultas tiene una función de pérdida no convexa donde existe más de un mínimo local. Por lo tanto, diferentes inicializaciones de peso aleatorio pueden llevar a una precisión de validación diferente. Además, requiere ajustar varios hiper-parámetros como el número de neuronas, capas e iteraciones ocultas.

En la implementación $MLPClassifier$ los parámetros más importantes dentro de la función se encuentran: $learning\_rate$ que es el método de planificación para la actualización de los pesos y $solver$ se encarga de resolver la optimización de los pesos. Para el análisis de estos parámetros se realiza un experimento en el cual se compara con respecto al *score* de la red, después de aplicar

[10]Es un método utilizado en ANN para calcular los pesos que se utilizarán en la red. Su nombre es una abreviatura de «propagación de errores hacia atrás»

varios juegos de parámetros. Y en el análisis también se tienen en cuenta otras métricas como lo son: la media del *accuracy* y *loss_* es la pérdida calculada que genera la función de costo.

FIGURA 2.8: El algoritmo para la clasificación utilizando un perceptrón multicapa (Elaboración propia).

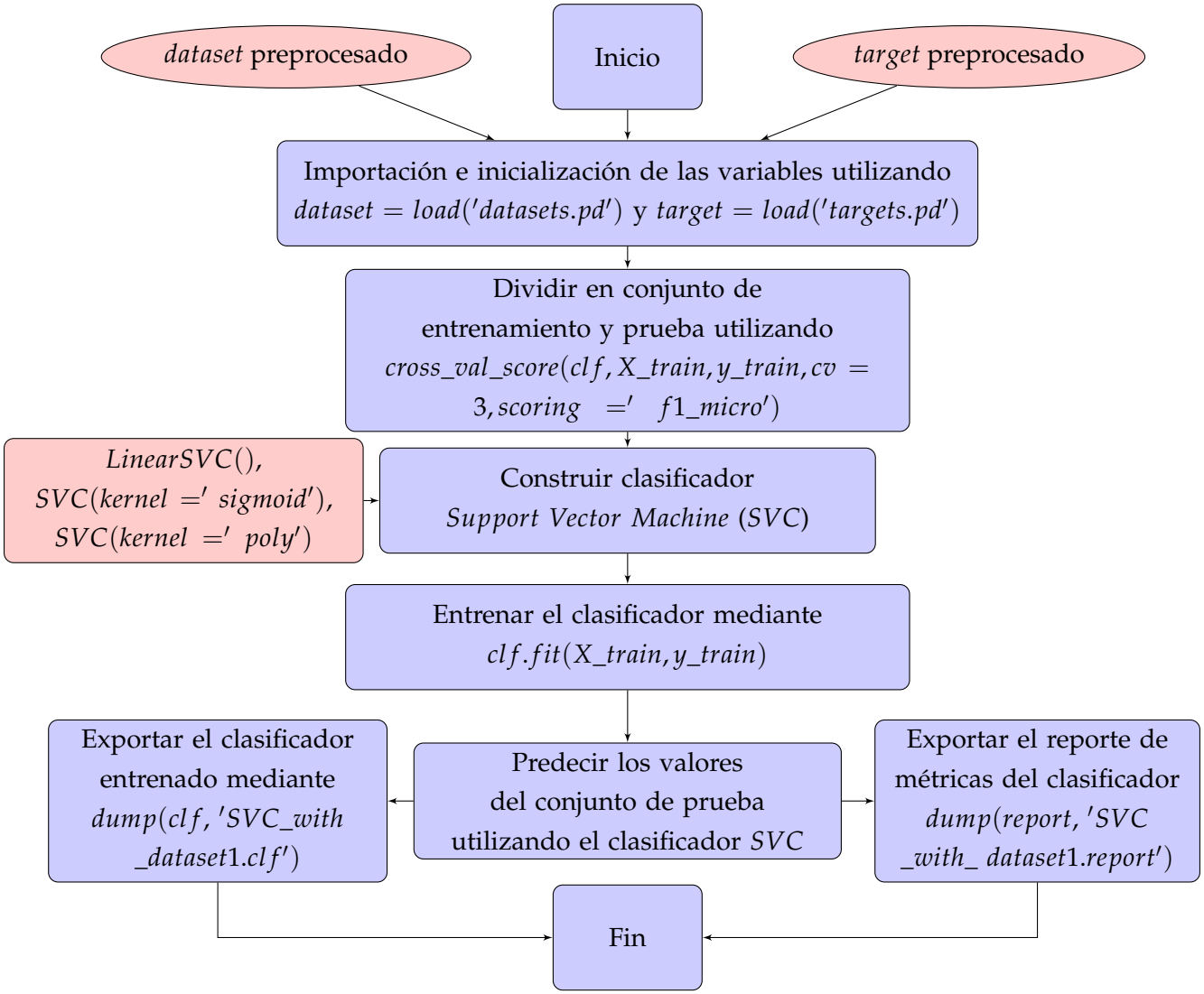


En el caso de los otros parámetros que son necesarios definir, en la presente investigación se utilizará $relu$, la función rectificada de la unidad lineal, devuelve $f(x) = max(0, x)$, ya que es ampliamente utilizado en la clasificación binaria según a literatura consultada. En el caso de los valores de $alpha$ o término de regularización se utilizará 0,0001, ya que estos también son comúnmente utilizados en la clasificación. En la figura 2.8, se observa diagrama de flujo del algoritmo utilizando para entrenar la red neuronal.

2.3.4. *Support Vector Machine*

Support Vector Machine (SVM) es un conjunto de métodos de aprendizaje supervisados utilizados comúnmente en para la clasificación. Cómo se analizó en el acápite 1.4.3, se utilizan $sklearn.svm.LinearSVC$ y $sklearn.svm.SVC$ de la biblioteca $scikit-learn$. La primera función es una implementación optimizada para utilizar el kernel lineal y la segunda es una implementación genérica que puede utilizar varios kernels. Las funciones *kernel* que se pueden utilizar en este último son las siguientes: linear: $\langle x, x' \rangle$; polynomial: $(\gamma \langle x, x' \rangle + r)^d$ (El valor de d representa el grado del polinomio, r los valores de los coeficientes definidos en el parámetro $coef0$) y sigmoid: $(\tanh(\gamma \langle x, x' \rangle + r))$ (Donde el valor de r es especificado por el parámetro $coef0$).

FIGURA 2.9: El algoritmo para la clasificación utilizando *Support Vector Machine* (Elaboración propia).



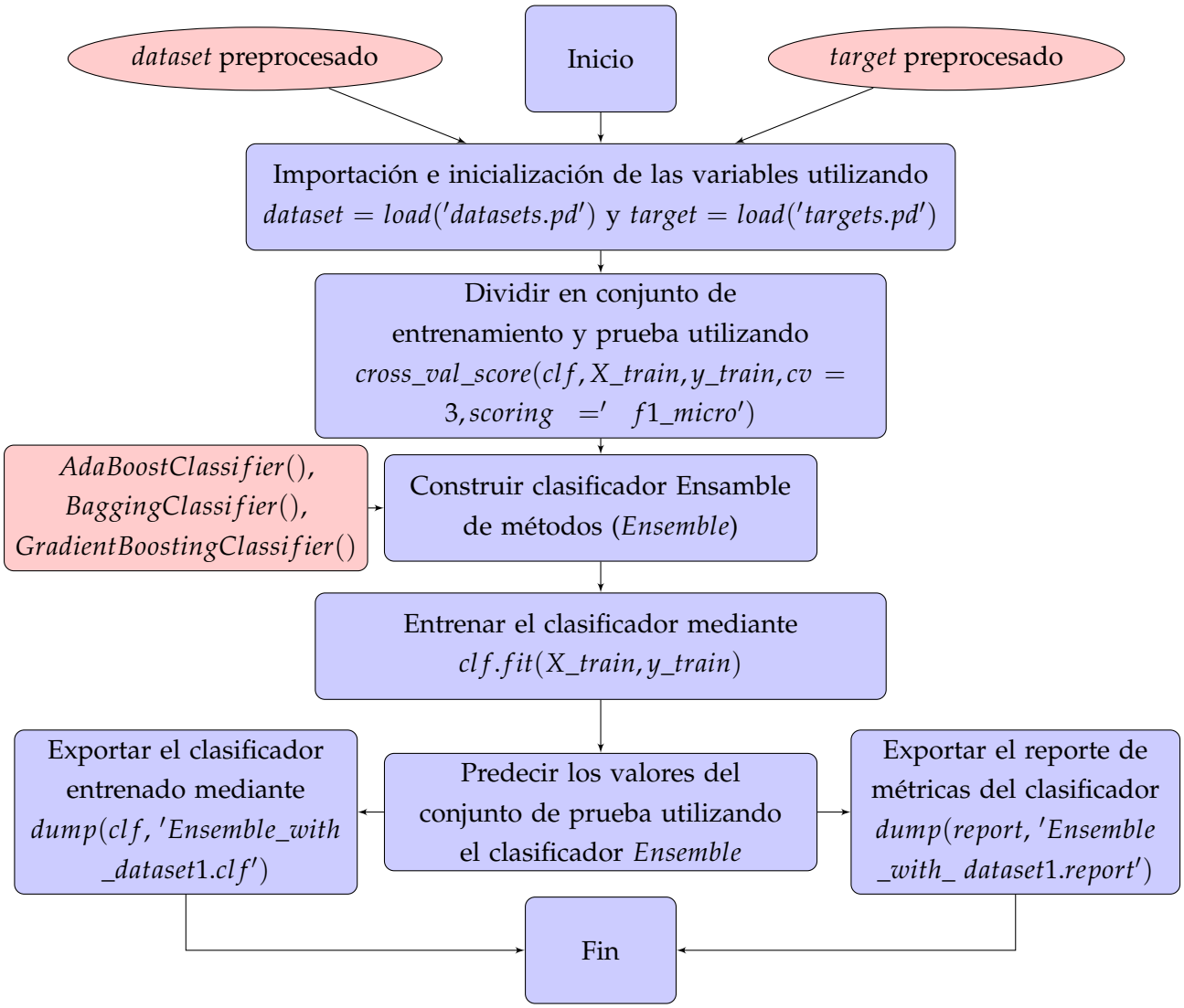
Para el análisis de las funciones kernel se realiza un experimento en el cuál se estudia el comportamiento de la clasificación con el uso de la métrica $f1\_score$. En la figura 2.9, se observa el

diagrama de flujo del algoritmo para el entrenamiento de los clasificadores utilizando *Support Vector Machine* con los tres principales *kernels* utilizados. Cada uno de los clasificadores fueron entrenados con el mismo conjunto de entrenamiento.

2.3.5. Ensamble de métodos

El objetivo de los ensambles métodos es combinar las predicciones de varios clasificadores entrenados con varios algoritmos de aprendizaje, para mejorar la generalización/robustez sobre un único clasificador. De los algoritmos analizados y seleccionados en el acápite 1.4.3, se utiliza las implementaciones propuestas por la biblioteca $scikit-learn$: $sklearn.ensemble.AdaBoostClassifier$, $sklearn.ensemble.BaggingClassifier$ y $sklearn.ensemble.GradientBoostingClassifier$. En la figura 2.10, se observa el diagrama de flujo del algoritmo para el entrenamiento de los métodos de ensamble, cada uno de ellos fueron entrenados con el mismo conjunto de entrenamiento.

FIGURA 2.10: El algoritmo para la clasificación utilizando ensamble de métodos (Elaboración propia).



En la fase de descubrimiento de patrones, se describieron los elementos fundamentales relacionados con los algoritmos de aprendizaje. Se analizaron los parámetros de cada uno de ellos y se propuso los experimentos para el análisis de alguno de ellos en la fase posterior. Además, se obtuvo los clasificadores entrenados a partir de los cuáles se pueden obtener los elementos necesarios en la fase posterior. En el anexo D, se puede consultar la implementación de esta fase, utilizando las herramientas analizadas en el capítulo 1 y las descripciones utilizadas en esta sección.

Conclusiones parciales

- En este capítulo se analizaron las dos primeras fases propuestas por el proceso de minería de la Web y cómo se aplica en realización de la presente investigación.
- Los pasos definidos en la fase de preprocesamiento fueron indispensables para organizar la implementación y obtener un conjunto de entrenamiento que facilitara su uso en la siguiente fase.
- En la fase de descubrimiento de patrones el análisis preliminar de los parámetros y funcionamiento de los algoritmos permitió tomar decisiones con el fin de mejorar el rendimiento de los mismos.
- Después de culminadas estas dos fases, las salidas obtenidas son fundamentales para la fase posterior ya que brindan la información necesaria para ser utilizada en el análisis de los patrones y la validación de la solución propuesta.

Capítulo 3

Desarrollo y validación de la propuesta

Introducción

En el presente capítulo se realiza el análisis de los resultados obtenidos, la evaluación de los resultados utilizando las métricas definidas y la validación de la propuesta. Además, se realiza la descripción de la última fase que propone el proceso de la minería Web: análisis de los patrones. Para ello se utiliza la experimentación para analizar el desempeño de los clasificadores entrenados en el capítulo 2, mediante las métricas definidas en el acápite 1.5.1.

Posteriormente se describen los valores de las métricas obtenidas a partir de los clasificadores entrenados. Por último se realiza se describen los métodos estadísticos utilizados para realizar el análisis comparativo entre las métricas y se arriban a las conclusiones.

3.1. Análisis de los patrones

Después de realizadas las primeras dos fases del proceso de la minería Web se realiza el análisis de los patrones, para ello en esta sección se realiza el análisis de los experimentos propuestos para la selección de los parámetros de los clasificadores. Además, se realiza el cálculo de las métricas, definidas en el acápite 1.5.1, sobre los clasificadores entrenados. Por último, se realiza un experimento para obtener los valores de la métricas de los clasificadores utilizados diferentes conjuntos de entrenamiento, cuyos resultados se analizan con respecto a las métricas definidas.

3.1.1. Análisis de los parámetros

Para dar inicio a las fase de análisis de los patrones, se analiza el resultado de los valores de los experimentos definidos en el capítulo 2. Para ello se describe los valores de los parámetros obtenidos en cada uno de los experimentos realizados. Para ello la fuente de información que se utilizó fue la información de los logs de navegación de los usuarios durante un día laboral, debido a las características del hardware disponible para su procesamiento. El mismo cuenta con 19632547 instancias, de las cuales se realizó una selección de un millón de instancias para ser utilizadas y conformar el conjunto de entrenamiento a utilizar, esta selección será denominada *Original*.

Preprocesamiento

Se tendrá en cuenta los resultados obtenidos en la fase de preprocesamiento, cómo se definió en el acápite 2.2, será organizado utilizando los cuatro pasos. Después de realizadas las transformaciones propuestas en el acápite 2.2.1, para la construcción del conjunto de entrenamiento, se obtuvieron el conjunto de entrenamiento y la variable objetivo que se describen a continuación:

- El *dataset* contiene una matriz con la cantidad de visitas que cada usuario realiza a un dominio determinado, ver la figura 3.1. Los valores de las características e instancias son las siguientes: 5910 dominios y 1171 usuarios.
- El *target* contiene una matriz unidimensional de 0 y 1, de tamaño 5910, en el cual se evidencia si el dominio es de interés(1) o no(0).

FIGURA 3.1: Fragmento del *dataset* (Elaboración propia).

username domain	0014505d8eeaf96094904267bb5d622f	001c14bed432cb1f1a4d327e778782be	002ec0d787c9b38899e33896al
	0	0	
0-edge-chat.facebook.com	0	0	
0-edge-chat.messenger.com	0	0	
0.client-channel.google.com	0	0	
0.gravatar.com	0	0	
0659febd873362761b06a9edf0261b0b.clo.footprintdns.com	0	0	
duolingo.com	0	1	
0701.static.prezi.com	0	0	
0761c191b4d318e0f929cd695feee60c.clo.footprintdns.com	0	0	
08db7a429ab82d6cd6a13b821738712a.clo.footprintdns.com	0	0	
0901.static.prezi.com	0	0	
0914.global.ssl.fastly.net	0	0	
0ea3dd69f8bd83905a0a0cd22d3a6424.clo.footprintdns.com	0	0	

En cuanto a los parámetros obtenidos después de ejecutado el paso 3: Extracción de características, definido en el acápite 2.2.3, se obtuvieron las siguientes métricas después de concluido el paso:

- *n_components_*: La cantidad de componentes es igual a la cantidad de características, por tanto, es igual a 1171.
- *components_*: Devuelve los 1171 componentes, ordenados por *explanation_variance_*.
- *get_covariance*: El mayor valor de varianza de los componentes es de 0,2397, por tanto, la diferencia entre los componentes es menor o igual que este valor.

En el último paso, paso 4: Datos desbalanceados, después de realizadas las transformaciones propuestas en el acápite 2.2.4, se obtuvieron los siguientes resultados generales: La cantidad de

instancias iniciales de la clase positiva($interest\_page = 1$) es de 157 entradas y 5753 entradas de la clase negativa($interest\_page = 0$). Después de realizado el paso se obtuvo 157 instancias, en ambas clases.

Después de concluida la fase se obtuvo la información necesaria para ser utilizada en el ajuste de los parámetros de la fase de Análisis de los patrones. La misma cuenta como salida con un fichero el cuál contiene la información preprocesada, tanto del *dataset* como del *target*.

Análisis de los patrones

Se tendrá en cuenta los resultados obtenidos en la fase de Análisis de los patrones, cómo se definió en el acápite 2.3. Para ello se analizará el comportamientode los clasificadores, a partir del la utilización de diferentes valores en los parámetros.

En el caso del clasificador *Nearest Neighbors*, según lo propuesto en el acápite 2.3.2, se realizó un experimento con el fin de determinar el número de cluster a utilizar. Para ello se realizó una comparación con varios valores aleatorio entre 5 y 120, como se observa en la tabla 3.1, los mejores resultados se obtuvieron con $n\_neighbors = 30$.

TABLA 3.1: Resumen de $f1\_score$ para diferentes valores de $n\_neighbors$.

Valor de $n\_neighbors$	$f1\_score$
5	0.7544848795489492
120	0.07534597642234751
20	0.9682214249103024
100	0.9359302921578677
30	**0.9774474628395694**
60	0.9697590978985136
50	0.9723218862121988

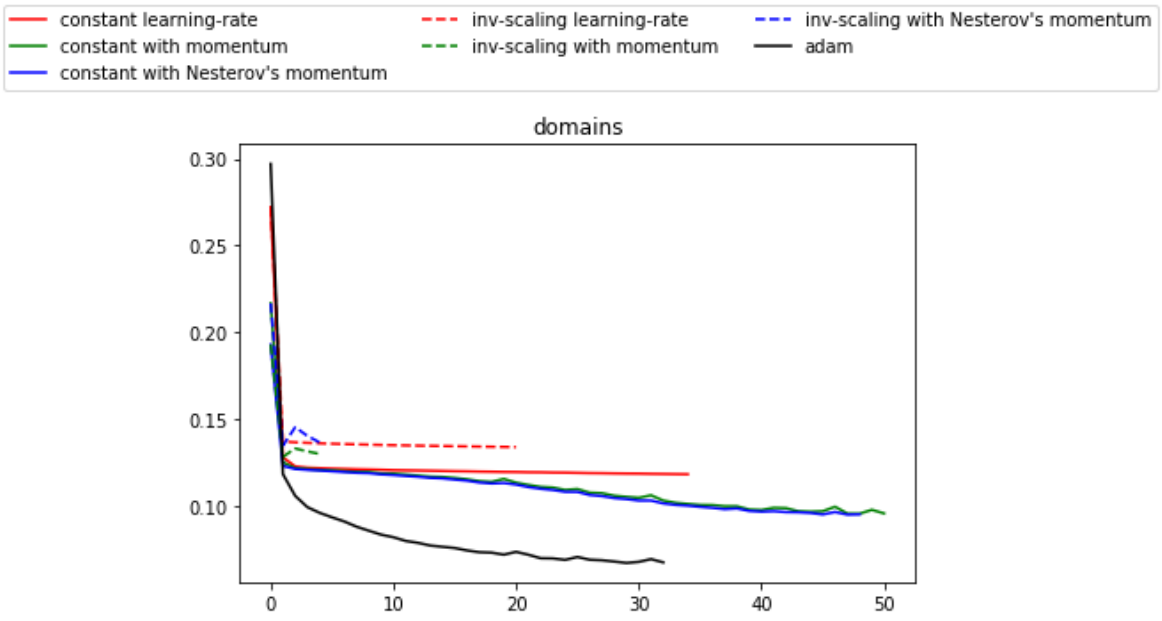
En el caso de la **Redes neuronales artificiales**, como se definió en el acápite 2.3.3 se analizó en comportamiento del clasificador $MLPClassifier$ con diferentes parámetros. En la tabla 3.2 y la figura 3.2, se puede observar los diferentes parámetros y los resultados se su desempeño en el conjunto de entrenamiento.

TABLA 3.2: Resumen de los parámetros para el *MLPClassifier*

Parámetros escogidos	*score*	*loss_*
'solver' :' sgd',' learning_rate' :' constant',' momentum' : 0,*' learning_rate_init'* : 0,2	0.9737	0.1184
'solver' :' sgd',' learning_rate' :' constant',' momentum' : ,9,*' nesterovs_momentum'* : *False,' learning_rate_init'* : 0,2	0.9788	0.0958
'solver' :' sgd',' learning_rate' :' constant',' momentum' : ,9,*' nesterovs_momentum'* : *True,' learning_rate_init'* : 0,2	0.9790	0.0953
'solver' :' sgd',' learnin_rate' :' invscaling',' momentum' : 0,*' learning_rate_init'* : 0,2	0.9734	0.1340
'solver' :' sgd',' learning_rate' :' invscaling',' momentum' : ,9,*' nesterovs_momentum'* : *True,' learning_rate_init'* : 0,2	0.9734	0.1301
'solver' :' sgd',' learning_rate' :' invscaling',' momentum' : ,9,*' nesterovs_momentum'* : *False,' learning_rate_init'* : 0,2	0.9734	0.1367
'solver' :' adam',' learning_rate_init' : 0,01	0.9847	0.0677

En la tabla 3.2 se puede observar que los mejores valores de *score* se logra con la utilización del *'solver' :' adam'* y en la figura se puede observar que este mismo método es el que más se acerca al mínimo local. Por tanto, esta combinación de parámetros serán los utilizados en el *MLPClassifier*

FIGURA 3.2: Comparación de los parámetros del *MLPClassifier* (Elaboración propia)



En el caso de *Support Vector Machine*, como se definió en 2.3.4 se analizará el comportamiento del clasificador con diferentes kernels. En la tabla 3.3, se encuentran los valores de *f1_score* obtenidos, los resultados obtenidos para los *kernels*: *bagging* y *polynomial* son muy bajos, pero se decide

mantenerlos para poder comprobar si con otros conjuntos de entrenamiento se obtienen resultados diferentes.

TABLA 3.3: Resumen de $f1\_score$ para diferentes valores de *kernels*

Kernel	$f1\_score$
$LinearSVC()$	0.7837006663249616
$SVC(kernel =' bagging')$	0.5511532547411584
$SVC(kernel =' gradient')$	0.62352639671963096

En esta sección se analizaron los parámetros obtenidos después de realizados los experimentos, por tanto, después de ella se realiza el entrenamiento de los clasificadores con los parámetros seleccionados. Estos clasificadores entrenados son exportados para su posterior uso en la evaluación y validación de la solución.

3.1.2. Diseño experimental

Los criterios para llevar a cabo la evaluación pueden ser intrínsecos o extrínsecos: los primeros evalúan los objetivos del sistema, o sea, clasificar las páginas Web de acuerdo al interés de una entidad; y los segundos evalúan el funcionamiento del sistema en sí. A continuación, se realiza la descripción de la propuesta de validación de los objetivos del sistema, apoyado en la hipótesis planteada.

La evaluación intrínsecos o sea de los objetivos del sistema de aprendizaje automático se tendrá en cuenta la variables independiente: «sistema de aprendizaje automático apoyado en los patrones de navegación de los usuarios.» La misma posee como dimensiones: la exactitud, la precisión y la calidad de la clasificación, las cuales se pueden observar en el anexo A. Las métricas seleccionadas para la evaluación de los clasificadores se corresponden con cada uno de los indicadores de cada dimensión, la relación entre métrica-indicador se puede observar en la tabla 3.4.

TABLA 3.4: Relación entre métrica e indicadores para la evaluación

Dimensión	**Indicador**	**Métrica**	**Fórmula**
Exactitud	Exactitud predictiva	*Acurracy*	$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
	Área bajo la curva ROC	*ROC*	Gráfico *ROC*
Precisión	Precisión predictiva	*Presition*	$precision = \frac{TP}{TP+FP}$
	Sensibilidad	*Recall*	$recall = \frac{TP}{TP+FN}$
Calidad	Influencia del azar	*KappaCohen*	$\kappa = \frac{\Pr(a)-\Pr(e)}{1-\Pr(e)}$

El procedimiento utilizado para realizar los experimentos que forman parte de la fase: Análisis de los patrones, incluye una serie de pasos los cuales son descritos a continuación:

1. Elegir un *datasets*.
2. Ejecutar cada uno de los clasificadores y obtener las métricas utilizando *cross_validation*.
3. Generar el reporte de las métricas de cada clasificador para el *dataset* seleccionado.
4. Realizar el análisis estadístico de las métricas obtenidas.

Los experimentos descritos en este acápite, fueron diseñados para estudiar las métricas después de la aplicación de las dos primeras fases del proceso de minería de la Web, definido en el acápite 1.2.2:

- Preprocesamiento
- Descubrimiento de patrones

En cuanto al hardware utilizado para llevar a cabo los experimentos e implementación de la propuesta de solución, es descrito a continuación:

- Memoria RAM: 3840MiB.
- CPU: Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz.
- Sistema Operativo: Debian 64 bits.

3.1.3. Datos empleados

Los datos empleados para la realización de la validación de los clasificadores provienen de los logs históricos de navegación de los usuarios, perteneciente a un día de almacenamiento, como se mencionó en el acápite 3.1. Para la realización de la experimentación se escogieron cinco muestras de un millón de entradas cada una; dentro de las cuáles se encuentra el *dataset Original* utilizado para el análisis y selección de los parámetros utilizados en las fases anteriores.

TABLA 3.5: Resumen de las características e instancias por *dataset*

Dataset	**Instancias**	**Características**
Original	5910	1171
*data*1	5210	1132
*data*2	9477	1729
*data*3	7781	1695
*data*4	5703	1110

Después de realizado el primer paso de la fase: Preprocesamiento, para cada uno de los *datasets* posee una cantidad de instancias (dominios) en el intervalos de $5000 < i < 10000$ y características

(usuarios) en el intervalo de $1000 < c < 2000$. Siendo los *datasets* *data*2 y *data*3 los de mayor tamaño. En la tabla 3.5, se puede apreciar las cantidades de características e instancias que poseen cada uno de los *datasets*.

TABLA 3.6: Distribución del tipo de variable objetivo por *dataset* antes y después de concluida la fase: Preprocesamiento

Dataset	**Clase negativa inicial**	**Clase positiva inicial**	**Clase negativa**	**Clase positiva**
Original	5753	157	157	157
*data*1	5053	157	157	157
*data*2	9168	309	309	309
*data*3	7543	238	238	238
*data*4	5513	190	190	190

En la tabla 3.6, se puede observar las distribución de la variable objetivo con respecto a si los dominios son de interés (Clase positiva) o no(Clase negativa). En ella se puede observar que al igual que con respecto a la cantidad de características e instancias, los *datasets* *data*2 y *data*3 los de mayor tamaño. Además, es evidente el desbalance de las clases en cada uno de los *dataset*. Después de concluida la fase: Preprocesamiento, se logra el balance entre la clase mayoritaria y la minoritaria.

3.1.4. Clasificadores empleados en la comparación

Un elemento importante en la fase: Análisis de los patrones, es la descripción de los clasificadores utilizados en los experimentos, utilizando los parámetros definidos en los acápites 2.3 y 3.1.1. Los clasificadores utilizados en las comparaciones son los siguientes:

- $GaussianNB(priors = None)$
- $KNeighborsClassifier(n\_neighbors = 30)$
- $MLPClassifier(solver =' adam', learning\_rate\_init = 0{,}01)$
- $LinearSVC()$
- $SVC(kernel =' sigmoid')$
- $SVC(kernel =' poly')$
- $AdaBoostClassifier()$
- $BaggingClassifier()$
- $GradientBoostingClassifier()$

3.1.5. Métricas empleadas

El último elemento en el diseño del experimento, son las métricas a utilizar con el fin de realizar un análisis exhaustivo los clasificadores. El objetivo de las comparaciones es analizar el comportamiento de los diferentes clasificadores en varios *datasets* mediante la aplicación de las métricas definidas en el acápite 1.5.1 y analizadas en el acápite 3.1.2.

Para el cálculo de dichas métricas se utilizó la biblioteca $scikit-learn$ y dentro de ella las funciones siguientes en correspondencia con ellas:

- *Accuracy*: para el cálculo de esta métrica se utiliza *metrics.accuracy_score*.
- *Precision*: para el cálculo de esta métrica se utiliza *metrics.precision_score*.
- *Recall*: para el cálculo de esta métrica se utiliza *metrics.recall_score*.
- *Area Under Curve*: para el cálculo de esta métrica se utiliza *metrics.roc_auc_score*.
- *Coeficiente Kappa Cohen*: para el cálculo de esta métrica se utiliza *metrics.cohen_kappa_score*.

Después de descritos los elementos que conformaron el experimento, se obtuvieron cada uno de los clasificadores seleccionados, entrenados por cada uno de los *datasets* seleccionados. A continuación, se da paso al análisis de los valores de las métricas obtenidas a partir de la evaluación de los clasificadores.

3.1.6. Resultados de la evaluación de las métricas

La evaluación de las métricas utilizando los clasificadores entrenados cada uno de los *datasets* bajo las condiciones descritas en el acápite anterior; arrojó los resultados que se pueden observar en las tablas 3.7, 3.8, 3.9, 3.10 y 3.11.

TABLA 3.7: Resumen de las métricas por clasificador para el *datasetoriginal*

Clasificador	*accuracy*	*precision*	*recall*	*ROC*	*Cohen Kappa*
$GaussianNB$	0.0337	0.7755	0.0645	0.6010	0.0172
$KNeighborsClassifier$	0.3878	0.0905	0.6013	0.0482	0.0433
$MLPClassifier$	0.0703	0.5306	0.1241	0.6749	0.0835
$LinearSVC$	0.0532	0.5102	0.0963	0.6381	0.0533
$SVC(kernel =' sigmoid')$	0.0501	0.3673	0.0882	0.5940	0.0461
$SVC(kernel =' poly')$	0.0316	0.7755	0.0608	0.5820	0.0132
$AdaBoostClassifier$	0.0390	0.4898	0.0723	0.5895	0.0270
$BaggingClassifier$	0.0416	0.4694	0.0764	0.5954	0.0317
$GradientBoostingClassifier$	0.0433	0.5102	0.0799	0.6100	0.0352

TABLA 3.8: Resumen de las métricas por clasificador para el *datasetdata*1

Clasificador	*accuracy*	*precision*	*recall*	*ROC*	*CohenKappa*
$GaussianNB$	0.8093	0.0621	0.3167	0.5719	0.0483
$KNeighborsClassifier$	0.9651	0.0458	0.5567	0.5	0.0335
$MLPClassifier$	0.782	0.0777	0.4833	0.6381	0.0786
$LinearSVC$	0.7645	0.0741	0.5	0.637	0.0727
$SVC(kernel =' sigmoid')$	0.9616	0.2857	0.0667	0.5303	0.0962
$SVC(kernel =' poly')$	0.957	0.2308	0.1	0.544	0.121
$AdaBoostClassifier$	0.6622	0.0547	0.5333	0.6001	0.0384
$BaggingClassifier$	0.736	0.0603	0.45	0.5982	0.0477
$GradientBoostingClassifier$	0.6593	0.0572	0.5667	0.6147	0.0434

TABLA 3.9: Resumen de las métricas por clasificador para el *datasetdata*2

Clasificador	*accuracy*	*precision*	*recall*	*ROC*	*CohenKappa*
$GaussianNB$	0.4239	0.0367	0.7907	0.6021	0.0186
$KNeighborsClassifier$	0.9725	0.0463	0.6045	0.534	0.0234
$MLPClassifier$	0.7084	0.0689	0.7674	0.7371	0.08
$LinearSVC$	0.7311	0.0763	0.7907	0.7601	0.0938
$SVC(kernel =' sigmoid')$	0.3897	0.038	0.8721	0.6241	0.0213
$SVC(kernel =' poly')$	0.8776	0.104	0.4535	0.6715	0.1303
$AdaBoostClassifier$	0.6579	0.0576	0.7442	0.6998	0.0588
$BaggingClassifier$	0.7618	0.0759	0.686	0.725	0.0918
$GradientBoostingClassifier$	0.7417	0.0682	0.6628	0.7034	0.0777

TABLA 3.10: Resumen de las métricas por clasificador para el *datasetdata*3

Clasificador	*accuracy*	*precision*	*recall*	*ROC*	*CohenKappa*
$GaussianNB$	0.6632	0.0533	0.6026	0.6338	0.0447
$KNeighborsClassifier$	0.9696	0.0643	0.5732	0.5	0.0562
$MLPClassifier$	0.7387	0.0594	0.5128	0.6293	0.0551
$LinearSVC$	0.7905	0.0848	0.6026	0.6995	0.1009
$SVC(kernel =' sigmoid')$	0.8407	0.081	0.4103	0.6322	0.0891
$SVC(kernel =' poly')$	0.8512	0.0914	0.4359	0.6501	0.1062
$AdaBoostClassifier$	0.6799	0.054	0.5769	0.63	0.0457
$BaggingClassifier$	0.6947	0.0609	0.6282	0.6625	0.059
$GradientBoostingClassifier$	0.7025	0.0647	0.6538	0.6789	0.0662

TABLA 3.11: Resumen de las métricas por clasificador para el *datasetdata*4

Clasificador	*accuracy*	*precision*	*recall*	*ROC*	*CohenKappa*
$GaussianNB$	0.6785	0.0568	0.5932	0.6373	0.0493
$KNeighborsClassifier$	0.9687	0.0556	0.6378	0.5	0.0498
$MLPClassifier$	0.7242	0.0611	0.5424	0.6362	0.0566
$LinearSVC$	0.7832	0.0605	0.4068	0.6011	0.0536
$SVC(kernel =' sigmoid')$	0.9065	0.0966	0.2373	0.5827	0.097
$SVC(kernel =' poly')$	0.923	0.0784	0.1356	0.542	0.0621
$AdaBoostClassifier$	0.6775	0.0537	0.5593	0.6203	0.0433
$BaggingClassifier$	0.754	0.059	0.4576	0.6106	0.0518
$GradientBoostingClassifier$	0.7317	0.0627	0.5424	0.6401	0.0596

Mediante el uso de la observación para analizar los resultados de las métricas en cada uno de los *datasets* por cada clasificador no se aprecian diferencias notables. Por tanto, se hace necesario utilizar métodos estadísticos con el objetivo de revisar el comportamiento de las métricas y decidir cuales presentan diferencias significativas en sus resultados y utilizarla para tomar una decisión.

3.2. Análisis estadísticos de los resultados

Con el objetivo de probar los resultados de cada una de las métricas en cada uno de los *datasets*, se prepararon los resultados obtenidos por los grupos. Cada grupo es definido por cada métrica utilizado, en cada uno de ellos tendrá cinco variables por cada uno de los *dataset*.

3.2.1. Supuesto de normalidad

Para poder realizar el análisis estadístico es indispensable contrastar la normalidad de un conjunto de datos. Con este objetivo existen diferentes pruebas dentro de ellas la prueba de Shapiro-Wilk es ampliamente utilizada ya que se considera una de las pruebas más potentes para el contraste de normalidad, en muestras pequeñas $(n < 30)$. La primera hipótesis a comprobar es el supuesto de normalidad de los datos de la cual se desglosan las siguientes hipótesis nula y alternativa:

- H_0: Los datos provienen de una distribución normal.
- H_1: Los datos no provienen de una distribución normal

La prueba propuesta por Shapiro y Wilk (1965), se considera la primera en detectar desviaciones de la normalidad debido a la asimetría (momento central de tercer orden), la curtosis (momento central

de cuarto orden) o ambas (Siegel, 1957). Para ello proponen un estadístico de prueba que posee en el numerador una combinación lineal de los estadísticos de orden de la muestra y en el denominador la varianza muestral. La hipótesis nula se rechazará si el valor del estadístico de prueba es demasiado pequeño, además el valor del mismo puede oscilar entre 0 y 1.

Para la realización de los cálculos se utilizó *scipy.stats.shapiro*, la misma define como valor de significancia $\alpha = 0{,}05$. Además, las comparaciones se realizan entre los valores de cada una de las métricas en los diferentes clasificadores con el fin de comprobar el comportamiento de las mismas.

TABLA 3.12: Resultados de la prueba de normalidad

Métrica	**Estadística**	**pvalue**	*Normal*
Accuracy	0.941845	0.679013	Sí
Precision	0.552182	0.000131	No
Recall	0.552182	0.000131	No
ROC	0.552182	0.000131	No
Coeficiente Kappa de Cohen	0.552182	0.000131	No

En la tabla 3.12, se observan los resultados obtenidos después de la aplicación de la prueba de Shapiro-Will. Se observa que la prueba de normalidad no da positiva en todos los casos, por tanto, los análisis comparativos de variables de concordancia y correlación deberán realizarse de forma no-paramétrica. La métrica *Accurancy* se distribuye de manera normal debido a que el *pvalue* es menor al valor de α y por tanto se acepta la hipótesis nula. En las restantes métricas el valor del *pvalue* es mayor que el valor α, se rechaza la hipótesis nula y por tanto, no siguen una distribución normal.

3.2.2. Análisis de k muestras independientes

El análisis de k muestras independientes se realiza para verificar si existen diferencias significativas en el valor de una métrica en dependencia del clasificador utilizado. En este apartado se presentan dos pruebas que permiten contrastar si los valores de cada métrica en cada clasificador proceden de una misma población, es decir, si un factor que subdivide la población de origen incide de forma significativa sobre el valor central de la población.

Las pruebas a utilizar tienen como objetivo verificar la diferencia entre los valores de las métricas en cada uno de los clasificadores y para la selección de las mismas en necesario tener en cuenta el supuesto de normalidad estudiado en el acápite anterior. Debido a que un grupo de métricas se comporta de manera normal y otro no, es necesario utilizar dos pruebas uno para cada grupo:

- Análisis de la Varianza (Anova), para las métricas que siguen una distribución normal.

- Prueba de Kruskal-Wallis, para las métricas que no siguen una distribución normal.

El análisis de la varianza (Anova)

La prueba Anova nos permite comparar las medias de r grupos, siendo r mayor o igual a 2. El modelo Anova presupone que las varianzas de los grupos son iguales y que los residuos o errores son aleatorios, independientes e idénticamente distribuidos siguiendo una ley normal con media 0 y desviación constante (Bradley, 1997).

La hipótesis nula de la prueba Anova de un factor es:

- H_0: Las medias de los k grupos son todas iguales.
- H_1: Al menos una de las medias es diferente.

Esta prueba se basa en la comparación de las sumas de cuadrados medias debidas a la variabilidad entre grupos y la debida a la variabilidad dentro de los grupos. Ambas sumas son estimaciones independientes de la variabilidad global, de manera que, si el cociente entre la primera y la segunda es grande, se tendrá mayor probabilidad de rechazar la hipótesis nula.

Para la realización de los cálculos se utilizó *scipy.stats.f_oneway* y las comparaciones se realizan entre los valores de cada una de las métricas en los diferentes clasificadores con el fin de comprobar el comportamiento de las mismas.

TABLA 3.13: Resultados de la prueba Anova

Métrica	**Estadística**	**pvalue**	H_0
Accuracy	6.783265	0.406006	Sí

En la tabla 3.13, se observan los resultados obtenidos después de la aplicación de la prueba de Anova sobre la métrica *Accurancy*. Se observa que el *pvalue* de la métrica es mayor que el valor de α y por tanto se acepta la hipótesis nula. Por tanto, las diferencias entre las medias son significativas después del análisis estadístico.

Prueba de Kruskal-Wallis

En el caso de las métricas que no siguen una distribución normal y el análisis de la varianza no es aplicable debido a incumplimientos de las suposiciones del modelo, es necesario aplicar la prueba de Kruskal-Wallis para el contraste de k medianas.

La prueba de Kruskal-Wallis es el método más adecuado para comparar poblaciones cuyas distribuciones no son normales. Las hipótesis planteadas por la prueba de Kruskal-Wallis son:

- H_0: Las k medianas son todas iguales
- H_1: Al menos una de las medianas es diferente

Para la realización de los cálculos se utilizó *scipy.stats.kruskal*, la misma define como valor de significancia $\alpha = 0{,}05$. Además, las comparaciones se realizan entre los valores de cada una de las métricas en los diferentes clasificadores con el fin de comprobar el comportamiento de las mismas.

TABLA 3.14: Resultados de la prueba de Kruskal-Wallis.

Métrica	**Estadística**	**pvalue**	H_0
Precision	0.334626	0.325006	No
Recall	2.514727	0.254027	No
ROC	0.827750	0.406006	No
Coeficiente Kappa de Cohen	0.869962	0.406006	No

En la tabla 3.14, se observan los resultados obtenidos después de la aplicación de la prueba de Kruskal-Wallis sobre las métricas que no siguen una distribución normal: *Precision*, *Recall*, *ROC* y *Coeficiente Kappa de Cohen*. Se observa que el *pvalue* de las métricas es mayor que el valor de α y por tanto, se acepta la hipótesis nula. Por tanto, las diferencias entre las medias no son significativas después del análisis estadístico.

3.2.3. Resultado del análisis estadístico

Después del análisis realizado en los acápites 3.2.2 y 3.2.2, se puede concluir que la única métrica que posee diferencias estadísticas significativas es: *Accurancy*. Por tanto, el análisis de los clasificadores con mejores resultados se puede obtener a partir del promedio de dicha métrica en cada uno de los *dataset*.

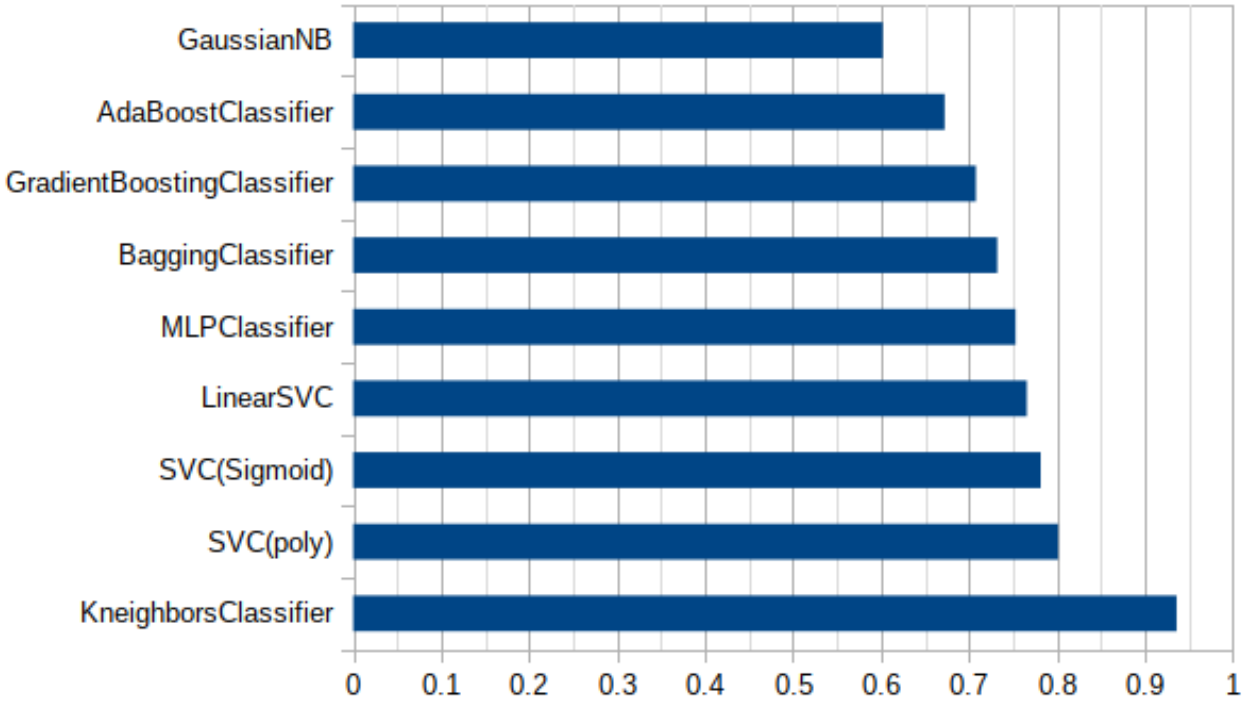
TABLA 3.15: Resumen del promedio de *Accuracy* para los clasificadores en los *datasets*.

Clasificador	*Original*	*data*1	*data*2	*data*3	*data*4	*Promedio*
GaussianNB	0.4352	0.8093	0.4239	0.6632	0.6785	0.60202
KNeighborsClassifier	0.8042	0.9651	0.9725	0.9696	0.9687	0.93602
MLPClassifier	0.8119	0.782	0.7084	0.7387	0.7242	0.75304
LinearSVC	0.7596	0.7645	0.7311	0.7905	0.7832	0.7657
$SVC(kernel =' sigmoid')$	0.8093	0.9616	0.3897	0.8407	0.9065	0.78156
$SVC(kernel =' poly')$	0.3983	0.957	0.8776	0.8512	0.923	0.80142
AdaBoostClassifier	0.6843	0.6622	0.6579	0.6799	0.6775	0.67236
BaggingClassifier	0.715	0.736	0.7618	0.6947	0.754	0.7323
GradientBoostingClassifier	0.7048	0.6593	0.7417	0.7025	0.7317	0.708

En la tabla 3.15, puede se observa el resumen del promedio de *Accuracy* para los clasificadores en todos los *datasets*. De los clasificadores entrenados $KNeighborsClassifier$ es el que mejores resultados promedio alcanzó, por tanto, será el seleccionado en la presente investigación.

Aunque $KNeighborsClassifier$, fue el algoritmo de mejores resultados, en la propuesta final se incluyen los demás clasificadores entrenados, para que los especialistas del área de aplicación, puedan apoyarse en sus predicciones para la toma de decisiones. En la figura 3.3, se representan los clasificadores del que peores resultados obtuvo al de mejores resultados utilizando la métrica *Accuracy*.

FIGURA 3.3: Promedio de *Accuracy* para los clasificadores (Elaboración propia).



Después de la utilización del algoritmo $KNeighborsClassifier$, el cual según lo anteriormente analizado presenta los mejores resultados generales en la clasificación. En la figura 3.4, se observa el resultado de la predicción del clasificador $KNeighborsClassifier$ entrenado con el $dataset = data3$ que fue el que mejores resultados obtuvo en cuanto la métrica *accuracy* como se muestra en la tabla 3.15. Para mostrar los resultados de manera visual se seleccion un conjunto de cinco URL, de las cuales, se encuentran dos que son de interés de la universidad, estas dos URL son *duolingo.com* y *redalyc.org*.

FIGURA 3.4: Ejemplo de salida del clasificador *KNeighborsClassifier* (Elaboración propia).

```
In [21]: from joblib import load

         pages = ['duolingo.com','0.gravatar.com', 'static.prezi.com','cio.footprintdns.com','redalyc.org']
         knn = load('KNeighbors.clf')

In [23]: knn.predict(pages)

Out[23]: [1, 0, 0, 0, 1]
```

La salida del método *predict*(), es el valor de la etiqueta predicha por el clasificador, donde el valor 1 es que es de interés y 0 en caso contrario. Como se puede observar el valor de predicción del clasificador *KNeighborsClassifier*, para la clasificación de las páginas Web de acuerdo al interés de una entidad en el ejemplo es correcto.

Conclusiones parciales

- En este capítulo se analizó la última fase propuesta por el proceso de minería de la Web y cómo se aplica en la realización de la presente investigación, lo cuál permitió la selección de los parámetros de los clasificadores, el cálculo de las métricas y el análisis del comportamiento de las dos primeras fases.
- La aplicación de los métodos estadísticos, para el análisis de las métricas obtenidas a partir de los diferentes *datasets*, permitió identificar que la única métrica que presenta diferencias significativas es *Acurracy*, por tanto, es la que se puede utilizar para comparar los clasificadores.
- El análisis del promedio de *Acurracy*, de cada clasificador en todos los *datasets*, permitió comprobar que *KNeighborsClassifier* presenta los mejores resultados, por tanto, será el utilizado en la propuesta de solución para la clasificación de páginas Web de acuerdo al interés de una entidad.

Conclusiones

La investigación realizada permite arribar a las siguientes conclusiones:

- Con los elementos teóricos y prácticos actuales de las ciencias informáticas en lo referido la minería del uso de la Web, el aprendizaje automático y la clasificación automática; se desarrolló un clasificador $KNeighborsClassifier$ utilizando los patrones de navegación de los usuarios, que permite clasificar las páginas Web de acuerdo al interés de una entidad.
- En la solución se utilizó el proceso definido por la minería Web con las fases: preprocesamiento, descubrimiento y análisis de patrones; lo cuál permitió la organización de su desarrollo e implementación.
- Los clasificadores entrenados obtuvieron resultados similares en el valor de las métricas utilizadas mediante la observación, lo que hizo necesario la utilización de métodos estadísticos para comprobar si existían diferencias significativas en los resultados en cuanto exactitud, precisión y calidad de la clasificación.
- La aplicación de los métodos estadísticos, para el análisis de las métricas obtenidas a partir de los diferentes *datasets*, permitió identificar que la única métrica que presenta diferencias significativas es *Acurracy*, por tanto, es la que se puede utilizar para comparar los clasificadores.
- El análisis del promedio de *Acurracy*, de cada clasificador en todos los *datasets*, permitió comprobar que $KNeighborsClassifier$ presenta los mejores resultados, por tanto, será el utilizado en la propuesta de solución para la clasificación de páginas Web de acuerdo al interés de una entidad.

Recomendaciones

Para la continuidad de la presente investigación se recomienda:

- Incorporar a la propuesta de solución el análisis incremental debido al dinamismo de la generación de los logs y las tendencias de navegación de los usuarios.
- Definir un método para el cálculo de la relación entre usuarios y páginas Web, que incorpore nuevos indicadores relacionados con la navegación de los usuarios.

Bibliografía

ADENIYI, D.A.; WEI, Z. y YONGQUAN, Y., 2016. Automated Web Usage Data Mining and Recommendation System Using K-Nearest Neighbor (KNN) Classification Method. *Applied Computing and Informatics*. Vol. 12, n.º 1, págs. 90-108. ISSN 2210-8327. Disponible desde DOI: 10.1016/j.aci.2014.10.001.

ADHVARYU, Rachit. A Review Paper on Web Usage Mining and Pattern Discovery. *Journal Of Information, Knowledge And Research In Computer Engineering*. Vol. 2, págs. 12.

AGARWAL, Ritu y DHAR, Vasant, 2014. Editorial—Big Data, Data Science, and Analytics: The Opportunity and Challenge for IS Research. *Information Systems Research*. Vol. 25, n.º 3, págs. 443-448. ISSN 1047-7047. Disponible desde DOI: 10.1287/isre.2014.0546.

ASATKAR, Bhagyesh P; WAGH, KP y CHATUR, PN, 2017. Classification of Web Pages Using Naive Bayesian Approach. En: *Classification of Web Pages Using Naive Bayesian Approach. International Conference on Recent Trends in Engineering and Science (ICRTES 2017)*. Vol. 20, 21st.

AUSLOOS, Marcel; CASTELLANO, Rosella y CERQUETI, Roy, 2016. Regularities and Discrepancies of Credit Default Swaps: A Data Science Approach through Benford's Law. *Chaos, Solitons & Fractals*. Vol. 90, págs. 8-17. ISSN 0960-0779. Disponible desde DOI: https://doi.org/10.1016/j.chaos.2016.03.002. Challenges in Data Science.

AYANKOYA, Kayode; CALITZ, Andre y GREYLING, Jean, 2014. Intrinsic Relations Between Data Science, Big Data, Business Analytics and Datafication. En: *Intrinsic Relations Between Data Science, Big Data, Business Analytics and Datafication. Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology*. New York, NY, USA: ACM, 192:192-192:198. SAICSIT '14. ISBN 978-1-4503-3246-0. Disponible desde DOI: 10.1145/2664591.2664619.

AYTUĞ ONAN, 2016. Classifier and Feature Set Ensembles for Web Page Classification. *Journal of Information Science* [online]. Vol. 42, n.º 2, págs. 150-165 [visitado 2018-02-05]. ISSN 0165-5515. Disponible desde DOI: 10.1177/0165551515591724.

BELL, Jason, 2015. *Machine Learning: Hands-on for Developers and Technical Professionals*. Indianapolis, IN: Wiley. ISBN 978-1-118-88906-0 978-1-118-88939-8 978-1-118-88949-7.

BENAVOLI, Alessio; CORANI, Giorgio; DEMŠAR, Janez y ZAFFALON, Marco, 2017. Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *The Journal of Machine Learning Research*. Vol. 18, n.º 1, págs. 2653-2688.

BHAGAT, R. C. y PATIL, S. S., 2015. Enhanced SMOTE Algorithm for Classification of Imbalanced Big-Data Using Random Forest. En: *Enhanced SMOTE Algorithm for Classification of Imbalanced Big-Data Using Random Forest. 2015 IEEE International Advance Computing Conference (IACC)*, págs. 403-408. Disponible desde DOI: 10.1109/IADCC.2015.7154739.

BHALLA, Vinod Kumar y KUMAR, Neeraj, 2017. An Efficient Multiclass Classifier Using On-Page Positive Personality Features for Web Page Classification for the Next Generation Wireless Communication Networks. *Wireless Personal Communications* [online]. Vol. 93, n.º 2, págs. 503-522 [visitado 2017-10-06]. ISSN 0929-6212, 1572-834X. ISSN 0929-6212, 1572-834X. Disponible desde DOI: 10.1007/s11277-016-3173-4.

BICHLER, Martin; HEINZL, Armin y van der AALST, Wil M. P., 2017. Business Analytics and Data Science: Once Again? *Business & Information Systems Engineering* [online]. Vol. 59, n.º 2, págs. 77-79 [visitado 2018-01-25]. ISSN 2363-7005, 1867-0202. ISSN 2363-7005, 1867-0202. Disponible desde DOI: 10.1007/s12599-016-0461-1.

BIEHL, Michael; HAMMER, Barbara y VILLMANN, Thomas, 2016. Prototype-Based Models in Machine Learning. *Wiley Interdisciplinary Reviews: Cognitive Science*. Vol. 7, n.º 2, págs. 92-111. ISSN 1939-5086. Disponible desde DOI: 10.1002/wcs.1378.

BIEN, Z. Zenn y LEE, Hyong-Euk, 2007. Effective Learning System Techniques for Human–robot Interaction in Service Environment. *Knowledge-Based Systems*. Vol. 20, n.º 5, págs. 439-456. ISSN 09507051. Disponible desde DOI: 10.1016/j.knosys.2007.01.005.

BISHOP, Christopher M., 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. ISBN 978-0-387-31073-2.

BLEI, David M. y SMYTH, Padhraic, 2017. Science and Data Science. *Proceedings of the National Academy of Sciences* [online]. Vol. 114, n.º 33, págs. 8689-8692 [visitado 2018-01-22]. ISSN 0027-8424, 1091-6490. ISSN 0027-8424, 1091-6490. Disponible desde DOI: 10.1073/pnas.1702076114.

BRADLEY, Andrew P, 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*. Vol. 30, n.º 7, págs. 1145-1159.

CADEZ, Igor; HECKERMAN, David; MEEK, Christopher; SMYTH, Padhraic y WHITE, Steven, 2003. Model-Based Clustering and Visualization of Navigation Patterns on a Web Site. *Data Mining and Knowledge Discovery* [online]. Vol. 7, n.º 4, págs. 399-424 [visitado 2018-02-10]. ISSN 1384-5810, 1573-756X. ISSN 1384-5810, 1573-756X. Disponible desde DOI: 10.1023/A:1024992613384.

CAI, Li y ZHU, Yangyong, 2015. The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *Data Science Journal* [online]. Vol. 14, n.º 0 [visitado 2018-01-25]. ISSN 1683-1470. Disponible desde DOI: 10.5334/dsj-2015-002.

CANO, Alberto; NGUYEN, Dat T; VENTURA, Sebastián y CIOS, Krzysztof J, 2016. ur-CAIM: improved CAIM discretization for unbalanced and balanced data. *Soft Computing*. Vol. 20, n.º 1, págs. 173-188.

CARRASQUILLA, Juan y MELKO, Roger G., 2017. Machine Learning Phases of Matter. *Nature Physics* [online]. Vol. 13, n.º 5, págs. 431-434 [visitado 2017-10-04]. ISSN 1745-2473. Disponible desde DOI: 10.1038/nphys4035.

CHOJNACKI, Alex; DAI, Chengyu; FARAHI, Arya; SHI, Guangsha; WEBB, Jared; ZHANG, Daniel T.; ABERNETHY, Jacob y SCHWARTZ, Eric, 2017. A Data Science Approach to Understanding Residential Water Contamination in Flint. En: *A Data Science Approach to Understanding Residential Water Contamination in Flint*. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, págs. 1407-1416. KDD '17. ISBN 978-1-4503-4887-4. Disponible desde DOI: 10.1145/3097983.3098078.

CLEVELAND, William S., 2001. Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics. *International Statistical Review*. Vol. 69, n.º 1, págs. 21-26. ISSN 1751-5823. Disponible desde DOI: 10.1111/j.1751-5823.2001.tb00477.x.

DALKIR, Kimiz y BEAULIEU, Marco, 2017. *Knowledge Management in Theory and Practice*. MIT Press. ISBN 978-0-262-03687-0.

DEHANKAR, S; WAGH, KP y CHATUR, PN, 2015. Web Page Classification Using Apriori Algorithm and Naïve Bayes Classifier. *International Journal*. Vol. 3, n.º 4.

DIETTERICH, Thomas G., 2000. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*. Vol. 40, n.º 2, págs. 139-157. ISSN 1573-0565. Disponible desde DOI: 10.1023/A:1007607513941.

DITTMAN, D.J.; KHOSHGOFTAAR, Taghi; WALD, Randall y NAPOLITANO, A, 2014. Comparison of data sampling approaches for imbalanced bioinformatics data, págs. 268-271.

DURANT, Kathleen T y SMITH, Michael D, 2006. Mining Sentiment Classification from Political Web Logs. En: *Mining Sentiment Classification from Political Web Logs*. *Proceedings of Workshop on Web Mining and Web Usage Analysis of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (WebKDD-2006), Philadelphia, PA*.

DURRANT, Robert J.; KIM, Kee-Eung; HOLMES, Geoffrey; MARSLAND, Stephen; SUGIYAMA, Masashi y ZHOU, Zhi-Hua, 2017. Foreword: Special Issue for the Journal Track of the 8th Asian Conference on Machine Learning (ACML 2016). *Machine Learning* [online]. Vol. 106, n.º 5, págs. 623-625 [visitado 2017-10-04]. ISSN 0885-6125, 1573-0565. ISSN 0885-6125, 1573-0565. Disponible desde DOI: 10.1007/s10994-017-5637-5.

FAYYAD, Usama M.; CANDEL, Arno; ARIÑO DE LA RUBIA, Eduardo; PAFKA, Szilárd; CHONG, Anthony y LEE, Jeong-Yoon, 2017. Benchmarks and Process Management in Data Science: Will We Ever Get Over the Mess? En: Benchmarks and Process Management in Data Science. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, págs. 31-32. KDD '17. ISBN 978-1-4503-4887-4. Disponible desde DOI: 10.1145/3097983.3120998.

FAYYAD, Usama M.; SIMOUDIS, Evangelos y SRIVASTAVA, Ashok, 2017. Foreword to the Applied Data Science: Invited Talks Track at KDD-2017. En: Foreword to the Applied Data Science. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, págs. 7-8. KDD '17. ISBN 978-1-4503-4887-4. Disponible desde DOI: 10.1145/3097983.3121426.

FEI, Yang y LI, Wei-qin, 2017. Improve Artificial Neural Network for Medical Analysis, Diagnosis and Prediction. *Journal of Critical Care* [online] [visitado 2017-09-01]. Disponible desde: http://www.sciencedirect.com/science/article/pii/S0883944117308468.

FERNÁNDEZ HERNÁNDEZ, Yumilka; HERNÁNDEZ, Yumilka Bárbara Fernández; CABRERA, Yaima Filiberto; PÉREZ, Rafael Bello; DOMINGUEZ, Mabel Frias y MOTA, Yaile Caballero, 2016. Efecto de La Selección de Rasgos En La Clasificación Basada En Prototipos. *Rev. Cuba. Cienc. Inform.* Vol. 10, n.º 4, págs. 83-96. ISSN 1994-1536. **urlalso**: http://rcci.uci.cu/?journal=rcci&page=article&op=view&path[]=1203.

FERREIRA, P.; SIMONS, C.; FERREIRA, P. y SIMONS, C., 2017. Machine Learning with Python. En: *Machine Learning with Python* [online]. Bristol, UK: Association of C and C++ Users [visitado 2018-03-14]. Disponible desde: http://www.accu.org.

FOREMAN, John W; JENNINGS, Greg y MILLER, Evan, 2014. *Data Smart: Using Data Science to Transform Information into Insight*. John Wiley & Sons.

FU, Bo; BANSOD, Sourabh; JAIN, Kunal; PRICE, Thomas; CHANDRASEKARAN, Deepak; GUPTA, Prachi; SINGH, Sarvjeet; CHEW, Sue Yi y XIE, Jierui, 2017. Dynamic Throttling of In-App Promotions to Reduce Marketing Spend Based on Machine-Learning. *Defensive Publications Series*. **urlalso**: http://www.tdcommons.org/dpubs_series/407.

GAN, Thomas; BALMAIN, Brendan y SIGBATULLIN, Artem, 2016. Formation Evaluation Logoff Results Comparing New Generation Mining-Style Logging Tools to Conventional Oil and Gas Logging Tools for Application in Coalbed Methane (CBM) Field Development. *Journal of Natural Gas Science and Engineering* [online]. Vol. 34, págs. 1237-1250 [visitado 2018-02-11]. ISSN 1875-5100. Disponible desde DOI: 10.1016/j.jngse.2016.07.070.

GARCÍA, Salvador; LUENGO, Julián y HERRERA, Francisco, 2014. *Data Preprocessing in Data Mining*. Springer. ISBN 978-3-319-10247-4.

GARCÍA, Salvador; RAMÍREZ-GALLEGO, Sergio; LUENGO, Julián; BENÍTEZ, José Manuel y HERRERA, Francisco, 2016. Big Data Preprocessing: Methods and Prospects. *Big Data Analytics* [online]. Vol. 1, págs. 9 [visitado 2017-09-01]. ISSN 2058-6345. Disponible desde DOI: 10.1186/s41044-016-0014-0.

GIAMA, E. y PAPADOPOULOS, A. M., 2018. Carbon Footprint Analysis as a Tool for Energy and Environmental Management in Small and Medium-Sized Enterprises. *International Journal of Sustainable Energy*. Vol. 37, n.º 1, págs. 21-29. Disponible desde DOI: 10.1080/14786451.2016.1263198.

GLOVER, Eric J.; TSIOUTSIOULIKLIS, Kostas; LAWRENCE, Steve; PENNOCK, David M. y FLAKE, Gary W., 2002. Using Web Structure for Classifying and Describing Web Pages. En: *Using Web Structure for Classifying and Describing Web Pages. Proceedings of the 11th International Conference on World Wide Web*. New York, NY, USA: ACM, págs. 562-569. WWW '02. ISBN 978-1-58113-449-0. Disponible desde DOI: 10.1145/511446.511520.

GUPTA, Ashika; ARORA, Rakhi; SIKARWAR, Ranjana y SAXENA, Neha, 2014. Web Usage Mining Using Improved Frequent Pattern Tree Algorithms. En: *Web Usage Mining Using Improved Frequent Pattern Tree Algorithms. Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference On*. IEEE, págs. 573-578.

GUPTA, Shashank y GUPTA, B. B., 2017. Cross-Site Scripting (XSS) Attacks and Defense Mechanisms: Classification and State-of-the-Art. *International Journal of System Assurance Engineering and Management* [online]. Vol. 8, n.º 1, págs. 512-530 [visitado 2018-01-31]. ISSN 0975-6809, 0976-4348. ISSN 0975-6809, 0976-4348. Disponible desde DOI: 10.1007/s13198-015-0376-0.

HAZEN, Benjamin T.; BOONE, Christopher A.; EZELL, Jeremy D. y JONES-FARMER, L. Allison, 2014. Data Quality for Data Science, Predictive Analytics, and Big Data in Supply Chain Management: An Introduction to the Problem and Suggestions for Research and Applications. *International Journal of Production Economics*. Vol. 154, págs. 72-80. ISSN 0925-5273. Disponible desde DOI: https://doi.org/10.1016/j.ijpe.2014.04.018.

HERNÁNDEZ SAMPIERI, Roberto; FERNÁNDEZ COLLADO, Carlos y BAPTISTA LUCIO, Pilar, 2010. *Metodología de La Investigación . México, DF*. México: McGraw Hill.

HICKS, Chelsea; BEEBE, Nicole y HALISCAK, Brandi, 2016. Extending Web Mining to Digital Forensics Text Mining. *AMCIS 2016 Proceedings*. **urlalso**: https://aisel.aisnet.org/amcis2016/ISSec/Presentations/19.

HOLZINGER, Andreas y JURISICA, Igor, 2014. Knowledge Discovery and Data Mining in Biomedical Informatics: The Future Is in Integrative, Interactive Machine Learning Solutions. En: Knowledge Discovery and Data Mining in Biomedical Informatics. *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics* [online]. Springer, Berlin, Heidelberg, págs. 1-18 [visitado 2018-01-19]. Disponible desde DOI: 10.1007/978-3-662-43968-5_1.

AL-JEFRI, Majed M.; EVANS, Roger; GHEZZI, Pietro y UCHYIGIT, Gulden, 2017. Using Machine Learning for Automatic Identification of Evidence-Based Health Information on the Web. En: *Using Machine Learning for Automatic Identification of Evidence-Based Health Information on the Web. Proceedings of the 2017 International Conference on Digital Health*. New York, NY, USA: ACM, págs. 167-174. DH '17. ISBN 978-1-4503-5249-9. Disponible desde DOI: 10.1145/3079452.3079470.

JING, Si-Yuan, 2014. A Hybrid Genetic Algorithm for Feature Subset Selection in Rough Set Theory. *Soft Computing* [online]. Vol. 18, n.º 7, págs. 1373-1382 [visitado 2017-09-01]. ISSN 1432-7643, 1433-7479. ISSN 1432-7643, 1433-7479. Disponible desde DOI: 10.1007/s00500-013-1150-3.

KAN, Min-Yen y THI, Hoang Oanh Nguyen, 2005. Fast Webpage Classification Using URL Features. En: *Fast Webpage Classification Using URL Features. Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, págs. 325-326. CIKM '05. ISBN 978-1-59593-140-5. Disponible desde DOI: 10.1145/1099554.1099649.

KANSARA, Richa Patel1 Akshay, 2015. WEB PAGES RECOMMENDATION SYSTEM BASED ON K-MEDOID CLUSTERING METHOD. *Development*. Vol. 2, n.º 5.

KARPATNE, Anuj y KUMAR, Vipin, 2017. Big Data in Climate: Opportunities and Challenges for Machine Learning. En: Big Data in Climate. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, págs. 21-22. KDD '17. ISBN 978-1-4503-4887-4. Disponible desde DOI: 10.1145/3097983.3105810.

KEITA, Moussa, 2017. *Data Science Sous Python: Algorithme, Statistique, DataViz, DataMining et Machine-Learning [Data Science with Python : Algorithm , Statistics , DataViz , DataMining and Machine - Learning]*. **urlalso**: https://ideas.repec.org/p/pra/mprapa/76653.html. MPRA Paper. University Library of Munich, Germany.

KIM, Yeongsu y LEE, Seungwoo, 2017. SVM-Based Web Content Mining with Leaf Classification Unit from DOM-Tree. En: *SVM-Based Web Content Mining with Leaf Classification Unit from DOM-Tree. Knowledge and Smart Technology (KST), 2017 9th International Conference On*. IEEE, págs. 359-364.

KIZILOLUK, Soner y OZER, Ahmet Bedri, 2017. Web Pages Classification with Parliamentary Optimization Algorithm. *International Journal of Software Engineering and Knowledge Engineering*. Vol. 27, págs. 499-513. ISSN 0218-1940. Disponible desde DOI: 10.1142/S0218194017500188.

KORMOS, Marton; COLLURA, Mario; TAKÁCS, Gabor y CALABRESE, Pasquale, 2017. Real-Time Confinement Following a Quantum Quench to a Non-Integrable Model. *Nature Physics* [online]. Vol. 13, n.º 3, págs. 246 [visitado 2018-01-25]. ISSN 1745-2481. Disponible desde DOI: 10.1038/nphys3934.

KOSALA, Raymond y BLOCKEEL, Hendrik, 2000. Web Mining Research: A Survey. *SIGKDD Explor. Newsl.* [online]. Vol. 2, n.º 1, págs. 1-15 [visitado 2018-02-11]. ISSN 1931-0145. Disponible desde DOI: 10.1145/360402.360406.

KOTENKO, Igor; CHECHULIN, Andrey; SHOROV, Andrey y KOMASHINSKY, Dmitry, 2014. Analysis and Evaluation of Web Pages Classification Techniques for Inappropriate Content Blocking. En: *Analysis and Evaluation of Web Pages Classification Techniques for Inappropriate Content Blocking. Industrial Conference on Data Mining*. Springer, págs. 39-54.

KRICHENE, Aida, 2017. Using a Naive Bayesian Classifier Methodology for Loan Risk Assessment: Evidence from a Tunisian Commercial Bank. *Journal of Economics, Finance and Administrative Science* [online]. Vol. 22, n.º 42, págs. 3-24 [visitado 2018-01-20]. ISSN 2077-1886. Disponible desde: http://www.scielo.org.pe/scielo.php?script=sci_abstract&pid=S2077-18862017000100002&lng=es&nrm=iso&tlng=en.

KUMAR, M y SHESHADRI, H, 2012. On the Classification of Imbalanced Datasets. *International Journal of Computer Applications*. Vol. 44.

LAW, M.H.C.; FIGUEIREDO, M.A.T. y JAIN, A.K., 2004. Simultaneous Feature Selection and Clustering Using Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. Vol. 26, n.º 9, págs. 1154-1166 [visitado 2018-03-20]. ISSN 0162-8828. Disponible desde DOI: 10.1109/TPAMI.2004.71.

LI, Huanhuan; CHEN, Quansheng; ZHAO, Jiewen y WU, Mengzi, 2015. Nondestructive detection of total volatile basic nitrogen (TVB-N) content in pork meat by integrating hyperspectral imaging and colorimetric sensor combined with a nonlinear data fusion. *LWT-Food Science and Technology*. Vol. 63, n.º 1, págs. 268-274.

LIU, Bing, 2007. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer Science & Business Media. ISBN 978-3-540-37881-5.

LIU, Bing y CHEN-CHUAN-CHANG, Kevin, 2004. Editorial: Special Issue on Web Content Mining. *SIGKDD Explor. Newsl.* [online]. Vol. 6, n.º 2, págs. 1-4 [visitado 2018-02-11]. ISSN 1931-0145. Disponible desde DOI: 10.1145/1046456.1046457.

LIU, Chang y ARNETT, Kirk P., 2000. Exploring the Factors Associated with Web Site Success in the Context of Electronic Commerce. *Information & Management*. Vol. 38, n.º 1, págs. 23-33. ISSN 0378-7206. Disponible desde DOI: https://doi.org/10.1016/S0378-7206(00)00049-5.

LIU, Ruoqian; YABANSU, Yuksel C.; YANG, Zijiang; CHOUDHARY, Alok N.; KALIDINDI, Surya R. y AGRAWAL, Ankit, 2017. Context Aware Machine Learning Approaches for Modeling Elastic Localization in Three-Dimensional Composite Microstructures. *Integrating Materials and Manufacturing Innovation* [online]. Vol. 6, n.º 2, págs. 160-171 [visitado 2017-10-04]. ISSN 2193-9764, 2193-9772. ISSN 2193-9764, 2193-9772. Disponible desde DOI: 10.1007/s40192-017-0094-3.

LOPES, Prajyoti y ROY, Bidisha, 2015. Dynamic Recommendation System Using Web Usage Mining for E-Commerce Users. *Procedia Computer Science* [online]. Vol. 45, págs. 60-69 [visitado 2018-02-10]. ISSN 1877-0509. Disponible desde DOI: 10.1016/j.procs.2015.03.086.

MALDONADO, Cesar Byron Guevara, 2017. *Desarrollo de Algoritmos Eficientes Para Identificación de Usuarios En Accesos Informáticos* [online] [visitado 2018-01-20]. Disponible desde: https://dialnet.unirioja.es/servlet/tesis?codigo=123061. http://purl.org/dc/dcmitype/Text. Universidad Complutense de Madrid.

MALOOF, Marcus A., 2006. *Machine Learning and Data Mining for Computer Security: Methods and Applications*. Springer Science & Business Media. ISBN 978-1-84628-253-9.

MARTINEZ, A. M. y KAK, A. C., 2001. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 23, n.º 2, págs. 228-233. ISSN 0162-8828. Disponible desde DOI: 10.1109/34.908974.

MOLINA DE ARMAS, Elvismary; PUPO MERIÑO, Mario; VÁZQUEZ, Leyva y YELANDI, Maikel, 2013. Aplicación informática que integra los procesos de generación, evaluación y uso de Clasificadores Bayesianos para dominios Bioinformáticos y Médicos. [online] [visitado 2018-03-03]. Disponible desde: http://repositorio.uci.cu/jspui/handle/ident/8015.

MOLINA-SOLANA, Miguel; ROS, María; RUIZ, M. Dolores; GÓMEZ-ROMERO, Juan y MARTIN-BAUTISTA, M. J., 2017. Data Science for Building Energy Management: A Review. *Renewable and Sustainable Energy Reviews*. Vol. 70, págs. 598-609. ISSN 1364-0321. Disponible desde DOI: https://doi.org/10.1016/j.rser.2016.11.132.

MORO, Sérgio; CORTEZ, Paulo y RITA, Paulo, 2015. Business intelligence in banking: A literature analysis from 2002 to 2013 using text mining and latent Dirichlet allocation. *Expert Systems with Applications*. Vol. 42, n.º 3, págs. 1314-1324.

NAIDENOVA, Xenia Alexandre, 2018. Incremental Approach to Classification Learning. *http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-2255-3.ch017* [online], págs. 191-201 [visitado 2017-10-06]. Disponible desde DOI: 10.4018/978-1-5225-2255-3.ch017.

NANNI, Loris y LUMINI, Alessandra, 2009. An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*. Vol. 36, n.º 2, Part 2, págs. 3028-3033. ISSN 0957-4174. Disponible desde DOI: https://doi.org/10.1016/j.eswa.2008.01.018.

NASRABADI, Nasser M., 2007. Pattern Recognition and Machine Learning. *Journal of Electronic Imaging*. Vol. 16. Disponible desde DOI: 10.1117/1.2819119.

NEWMAN, Russell; CHANG, Victor; WALTERS, Robert John y WILLS, Gary Brian, 2016. Model and Experimental Development for Business Data Science. *International Journal of Information Management*. Vol. 36, n.º 4, págs. 607-617. ISSN 0268-4012. Disponible desde DOI: 10.1016/j.ijinfomgt.2016.04.004.

NumPy — NumPy [online] [visitado 2018-03-05]. Disponible desde: http://www.numpy.org/.

OFFICIAL, Matlab. MathWorks - Makers of MATLAB and Simulink [online] [visitado 2018-03-05]. Disponible desde: https://www.mathworks.com/.

OFFICIAL, R. *R: The R Project for Statistical Computing* [online] [visitado 2018-03-05]. Disponible desde: https://www.r-project.org/.

OFFICIAL, Weka. Weka 3 - Data Mining with Open Source Machine Learning Software in Java [online] [visitado 2018-03-05]. Disponible desde: https://www.cs.waikato.ac.nz/ml/weka/.

ONAN, Aytuğ, 2016. Classifier and Feature Set Ensembles for Web Page Classification. *Journal of Information Science*. Vol. 42, n.º 2, págs. 150-165.

PARMAR, Devendra, 2015. SURVEY ON WEB USAGE MINING AND PRE-FETCHING. *Development*. Vol. 2, n.º 3.

PATIL, Ajay S y PAWAR, BV, 2012. Automated Classification of Web Sites Using Naive Bayesian Algorithm. En: *Automated Classification of Web Sites Using Naive Bayesian Algorithm*. *Proceedings of the International Multiconference of Engineers and Computer Scientists*. Vol. 1, págs. 519-523.

PIERRAKOS, Dimitrios; PALIOURAS, Georgios; PAPATHEODOROU, Christos y SPYROPOULOS, Constantine D., 2003. Web Usage Mining as a Tool for Personalization: A Survey. *User Modeling and User-Adapted Interaction* [online]. Vol. 13, n.º 4, págs. 311-372 [visitado 2018-02-11]. ISSN 0924-1868, 1573-1391. ISSN 0924-1868, 1573-1391. Disponible desde DOI: 10.1023/A:1026238916441.

PROVOST, Foster y FAWCETT, Tom, 2013. *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. .º'Reilly Media, Inc."

Python Data Analysis Library — Pandas: Python Data Analysis Library [online] [visitado 2018-03-05]. Disponible desde: https://pandas.pydata.org/.

Python.Org, 2018 [online] [visitado 2018-03-05]. Disponible desde: https://www.python.org/.

QI, Xiaoguang y DAVISON, Brian D., 2009. Web Page Classification: Features and Algorithms. *ACM Comput. Surv.* Vol. 41, n.º 2, págs. 12:1-12:31. ISSN 0360-0300. Disponible desde DOI: 10.1145/1459352.1459357.

RAITH, Stefan; VOGEL, Eric Per; ANEES, Naeema; KEUL, Christine; GÜTH, Jan-Frederik; EDELHOFF, Daniel y FISCHER, Horst, 2017. Artificial Neural Networks as a Powerful Numerical Tool to Classify Specific Features of a Tooth Based on 3D Scan Data. *Computers in Biology and Medicine* [online]. Vol. 80, págs. 65-76 [visitado 2017-09-01]. ISSN 0010-4825. Disponible desde DOI: 10.1016/j.compbiomed.2016.11.013.

RAJALAKSHMI, R. y ARAVINDAN, C., 2011. Naive Bayes Approach for Website Classification. En: *Naive Bayes Approach for Website Classification*. *Information Technology and Mobile Communication* [online]. Springer, Berlin, Heidelberg, págs. 323-326 [visitado 2017-10-03]. Communications in Computer and Information Science. ISBN 978-3-642-20572-9 978-3-642-20573-6. Disponible desde: https://link.springer.com/chapter/10.1007/978-3-642-20573-6_55. 00013 DOI: 10.1007/978-3-642-20573-6_55.

RAO, Rajinder Singh y ARORA, Jyoti, 2017. A Survey on Methods Used in Web Usage Mining.

RUPP, Ghislain M.; OPITZ, Alexander K.; NENNING, Andreas; LIMBECK, Andreas y FLEIG, Jürgen, 2017. Real-Time Impedance Monitoring of Oxygen Reduction during Surface Modification of Thin Film Cathodes. *Nature Materials* [online]. Vol. 16, n.º 6, págs. 640 [visitado 2018-01-25]. ISSN 1476-4660. Disponible desde DOI: 10.1038/nmat4879.

SALEH, Ahmed I.; RAHMAWY, Mohammed F. Al y ABULWAFA, Arwa E., 2017. A Semantic Based Web Page Classification Strategy Using Multi-Layered Domain Ontology. *World Wide Web* [online]. Vol. 20, n.º 5, págs. 939-993 [visitado 2018-02-10]. ISSN 1386-145X, 1573-1413. ISSN 1386-145X, 1573-1413. Disponible desde DOI: 10.1007/s11280-016-0415-z.

SANAGAVARAPU, Lalit Mohan; SARANGI, Sourav; Y, Raghu Reddy y VARMA, Vasudeva, 2018. Fine Grained Approach for Domain Specific Seed URL Extraction. En: *Fine Grained Approach for Domain Specific Seed URL Extraction* [online] [visitado 2018-03-08]. ISBN 978-0-9981331-1-9. Disponible desde: http://scholarspace.manoa.hawaii.edu/handle/10125/50111.

SATHYADEVAN, S.; SARATH, P. R.; ATHIRA, U. y ANJANA, V., 2014. Improved Document Classification through Enhanced Naive Bayes Algorithm. En: *Improved Document Classification through Enhanced Naive Bayes Algorithm*. *2014 International Conference on Data Science Engineering (ICDSE)*, págs. 100-104. Disponible desde DOI: 10.1109/ICDSE.2014.6974619.

SCHOENHERR, Tobias y SPEIER-PERO, Cheri, 2015. Data Science, Predictive Analytics, and Big Data in Supply Chain Management: Current State and Future Potential. *Journal of Business Logistics*. Vol. 36, n.º 1, págs. 120-132. ISSN 2158-1592. Disponible desde DOI: 10.1111/jbl.12082.

SCHUTT, Rachel y O'NEIL, Cathy, 2013. *Doing Data Science: Straight Talk from the Frontline*. .º'Reilly Media, Inc." ISBN 978-1-4493-6390-1.

SCHWARZENBACH, Heidi; MACHADO DA SILVA, Andreia; CALIN, George y PANTEL, Klaus, 2015. Data Normalization Strategies for MicroRNA Quantification. *Clinical Chemistry*. ISSN 0009-9147. Disponible desde DOI: 10.1373/clinchem.2015.239459.

Scikit-Learn: Machine Learning in Python — Scikit-Learn 0.19.1 Documentation [online] [visitado 2018-03-05]. Disponible desde: http://scikit-learn.org/stable/.

SHANMUGAM, Jagan y P RAJAGOPALAN, S, 2015. *A Survey on Web Personalization of Web Usage Mining*.

SHEN, Dou; CHEN, Zheng; YANG, Qiang; ZENG, Hua-Jun; ZHANG, Benyu; LU, Yuchang y MA, Wei-Ying, 2004. Web-Page Classification Through Summarization. En: *Web-Page Classification Through Summarization. Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, págs. 242-249. SIGIR '04. ISBN 978-1-58113-881-8. Disponible desde DOI: 10.1145/1008992.1009035.

SHLENS, Jonathon, 2014. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

SIEGEL, Sidney, 1957. Nonparametric statistics. *The American Statistician*. Vol. 11, n.º 3, págs. 13-19.

SRIVASTAVA, Mitali; GARG, Rakhi y MISHRA, P. K., 2015. Analysis of Data Extraction and Data Cleaning in Web Usage Mining. En: *Analysis of Data Extraction and Data Cleaning in Web Usage Mining. Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015)* [online]. New York, NY, USA: ACM, 13:1-13:6 [visitado 2018-02-11]. ICARCSET '15. ISBN 978-1-4503-3441-9. Disponible desde DOI: 10.1145/2743065.2743078.

TARIK, Boudheb; MAHMOUD, Djelloul Daouadji y ZAKARIA, Elberrichi, 2017. Classifying Web Pages by Aimed Nation Using Machine Learning. *International Journal of Organizational and Collective Intelligence (IJOCI)* [online]. Vol. 7, n.º 1, págs. 20-35 [visitado 2018-02-08]. ISSN 1947-9344 DOI: 10.4018/IJOCI.2017010102. Disponible desde DOI: 10.4018/IJOCI.2017010102.

TIMOTHY, G y col., 2017. study on the effects of unbalanced data when fitting logistic regression models in ecology. *Ecological indicators*.

TYAGI, Neha y GUPTA, Santosh Kumar, 2018. Web Structure Mining Algorithms: A Survey. En: Web Structure Mining Algorithms. *Big Data Analytics* [online]. Springer, Singapore, págs. 305-317 [visitado 2018-02-11]. Advances in Intelligent Systems and Computing. ISBN 978-981-10-6619-1 978-981-10-6620-7. Disponible desde DOI: 10.1007/978-981-10-6620-7_30.

Van der AALST, Wil M. P., 2014. Data Scientist: The Engineer of the Future. En: Data Scientist. *Enterprise Interoperability VI* [online]. Springer, Cham, págs. 13-26 [visitado 2018-01-21]. Disponible desde DOI: 10.1007/978-3-319-04948-9_2.

Van der AALST, Wil M. P., 2016. *Process Mining: Data Science in Action*. Springer. ISBN 978-3-662-49851-4.

VASILEVICH, Aliaksei S.; CARLIER, Aurélie; de BOER, Jan y SINGH, Shantanu, 2017. How Not To Drown in Data: A Guide for Biomaterial Engineers. *Trends in Biotechnology*. Vol. 35, n.º 8, págs. 743-755. ISSN 01677799. Disponible desde DOI: 10.1016/j.tibtech.2017.05.007.

VOYANT, Cyril; NOTTON, Gilles; KALOGIROU, Soteris; NIVET, Marie-Laure; PAOLI, Christophe; MOTTE, Fabrice y FOUILLOY, Alexis, 2017. Machine Learning Methods for Solar Radiation Forecasting: A Review. *Renewable Energy* [online]. Vol. 105, págs. 569-582 [visitado 2017-10-04]. ISSN 0960-1481. Disponible desde DOI: 10.1016/j.renene.2016.12.095.

WANG, Qi; LUO, ZhiHao; HUANG, JinCai; FENG, YangHe y LIU, Zhong, 2017. *A Novel Ensemble Method for Imbalanced Data Learning: Bagging of Extrapolation-SMOTE SVM* [online] [visitado 2018-04-02]. Disponible desde DOI: 10.1155/2017/1827016.

WEIBEL, Nadir, 2017. New Frontiers for Pervasive Telemedicine: From Data Science in the Wild to HoloPresence. En: *New Frontiers for Pervasive Telemedicine: From Data Science in the Wild to HoloPresence. Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*. ACM, págs. 276-281.

WONG, Wai-chiu y FU, Ada Wai-chee, 2002. Incremental Document Clustering for Web Page Classification. En: *Incremental Document Clustering for Web Page Classification. Enabling Society with Information Technology* [online]. Springer, Tokyo, págs. 101-110 [visitado 2018-01-31]. Disponible desde DOI: 10.1007/978-4-431-66979-1_10.

YOUSEFI, Mahyar; KAMKAR-ROUHANI, Abolghasem y CARRANZA, Emmanuel John M, 2014. Application of staged factor analysis and logistic function to create a fuzzy stream sediment geochemical evidence layer for mineral prospectivity mapping. *Geochemistry: Exploration, Environment, Analysis*. Vol. 14, n.º 1, págs. 45-58.

ZHU, Jieming; HE, Pinjia; FU, Qiang; ZHANG, Hongyu; LYU, Michael R. y ZHANG, Dongmei, 2015. Learning to Log: Helping Developers Make Informed Logging Decisions. En: Learning to Log. *Proceedings of the 37th International Conference on Software Engineering - Volume 1* [online]. Piscataway, NJ, USA: IEEE Press, págs. 415-425 [visitado 2018-02-10]. ICSE '15. ISBN 978-1-4799-1934-5. Disponible desde: http://dl.acm.org/citation.cfm?id=2818754.2818807.

ZINKEVICH, Martin, 2017. *Rules of Machine Learning: Best Practices for ML Engineering*. Technical Report.

Zu EISSEN, Sven Meyer y STEIN, Benno, 2004. Genre Classification of Web Pages. En: *Genre Classification of Web Pages. KI 2004: Advances in Artificial Intelligence* [online]. Springer, Berlin, Heidelberg, págs. 256-269 [visitado 2018-01-31]. Disponible desde DOI: 10.1007/978-3-540-30221-6_20.

Apéndice A

Operacionalización de las variables

Hipótesis:

El desarrollo de un sistema de aprendizaje automático apoyado en los patrones de navegación de los usuarios permite clasificar las páginas Web de acuerdo al interés de una entidad.

Variable independiente: Sistema de aprendizaje automático apoyado en los patrones de navegación de los usuarios.

TABLA A.1: Operacionalización de las variables independientes

Dimensión	**Indicadores**	**Métrica**
Exactitud	Exactitud predictiva	*Acurrancy*
	Área bajo la curva ROC	*ROC*
Precisión	Precisión predictiva	*Presition*
	Sensibilidad	*Recall*

Variable dependiente: clasificar las páginas Web de acuerdo al interés de una entidad.

TABLA A.2: Operacionalización de la variable dependiente.

Dimensión	**Indicadores**	**Unidad**
Calidad	Influencia del azar	*Coeficiente Kappa de Cohen*

Apéndice B

Clasificación de páginas web, una revisión bibliográfica

TABLA B.1: Clasificación de páginas web, una revisión bibliográfica

Año	Autor	Título	Objetivo de la investigación	Técnicas
2002	Sun, Aixin; Lim, EePeng; Ng, WeeKeong	Web Classification Using Support Vector Machine	Clasificación de páginas web	Support Vector Machine
2002	Glover, Eric J.; Tsioutsiouliklis, Kostas; Lawrence, Steve; Pennock, David M.; Flake, Gary W.	Using Web Structure for Classifying and Describing Web Pages	Clasificación y descripción de páginas web	Clustering
2002	Yu, Hwanjo; Han, Jiawei; Chang, Kevin ChenChuan	PEBL: Positive Example Based Learning for Web Page Classification Using SVM	Clasificación de páginas web	Support Vector Machine
2002	Wong, Waichiu; Fu, Ada Waichee	Incremental Document Clustering for Web Page Classification	Clasificación de páginas web	Clustering, DCtree
2004	Eissen, Sven Meyer zu; Stein, Benno	Genre Classification of Web Pages	Clasificación de páginas web en géneros	Neural Network, Support Vector Machines
2004	Shen, Dou; Chen, Zheng; Yang, Qiang; Zeng, HuaJun; Zhang, Benyu; Lu, Yuchang; Ma, WeiYing	Webpage Classification Through Summarization	Clasificación de páginas web	Algoritmos basados en resúmenes web
2005	Kan, MinYen; Thi, Hoang Oanh Nguyen	Fast Webpage Classification Using URL Features79	Clasificación de páginas web	Métodos basados en url
2006	Chen, RungChing; Hsieh, ChungHsun	Web page classification based	Clasificación de páginas web	Support Vector Machine (SVM)

Apéndice C

Fase: Preprocesamiento, implementación

FIGURA C.1: Código: etapa de Preprocesamiento. Parte 1 (Elaboración propia).

```
In [1]: %matplotlib inline
        import os
        import re
        import numpy as np
        import pandas as pd
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        import matplotlib.cm as cm

        from urllib.parse import urlparse
        from hashlib import md5
        from imblearn.under_sampling import RandomUnderSampler
        from joblib import dump

In [2]: # Definition of log structure and used
        COLUMNS = [
                'ts', 'hits', 'ip', 'tcp_resp',
                'size', 'http_method', 'url',
                'username', 'destiny', 'mimetype'
            ]
        USED_COLUMNS = {'url', 'username'}


        # Import and open archive domain
        base_path = os.getcwd()
        file_domains = os.path.join(base_path, 'domains')

        with open(file_domains) as domains:
        # Convert domains to regex
            RE_INTEREST =  re.compile(
                '^.*({})$'.format('|'.join(
                    re.escape(domain.replace(
                        '\n', '')) for domain in domains)))


        # Methods to preprocessing logs
```

FIGURA C.2: Código: etapa de Preprocesamiento. Parte 2 (Elaboración propia).

```
        # Clean domain
        def to_domain(url:str):
            if url.endswith(':443'):
                return url[:-4]
            if url.endswith(':80'):
                return url[:-3]
            return urlparse(url).netloc


        #  Extract domain from URL
        def to_interest_page(domain:str):
            return 1 if RE_INTEREST.match(domain) else 0

In [3]: # Import and open archive data. It's contain five archives from the dataset original
        data_file = [os.path.join(
            'file://',
            os.getcwd(),
            'data{}'.format(i)) for i in range(1,6)]

        # Convert datasets in panda Dataframe
        logs = [pd.read_csv(data_file[i],
                            delim_whitespace=True,
                            names=COLUMNS,
                            usecols=USED_COLUMNS)
                for i in range(5)]

In [4]: # Preprocessing all datasets
        for i in range(5):
        #   Extract domine from URL for all log
            logs[i]['domain'] = logs[i].apply(lambda x: to_domain(x['url']), axis=1)
        #     Remove the column "url" to the dataset
            logs[i].pop('url')
        #   Anonymize username for all log
            logs[i]['username'] = logs[i].apply(
                lambda x: md5(x['username'].encode('utf-8')).hexdigest(), axis=1)

In [5]: # Export preprocessed logs
        dump(logs,'total_logs.log')

Out[5]: ['total_logs.log']

In [6]: # Pivot all dataset using the number of visits per user
        datasets = [pd.pivot_table(logs[i],
                                   index="domain",
                                   columns=["username"],
                                   aggfunc=len, fill_value=0)
                    for i in range(5)]
```

FIGURA C.3: Código: etapa de Preprocesamiento. Parte 3 (Elaboración propia).

```
In [7]: # Remove anonymize user
        for i in range(5):
            datasets[i].pop(md5('-'.encode('utf-8')).hexdigest()) # unknown user :O

        # Export pandas datasets
        dump(datasets,'datasets.pd')

Out[7]: ['datasets.pd']

In [8]: # Build the target objetive for all dataset
        targets = [np.array([to_interest_page(domain)
                             for domain in datasets[i].index])
                   for i in range(5)]


        # Export padas target
        dump(targets,'targets.pd')

Out[8]: ['targets.pd']

In [9]: # Preprossesing datasets to remove unbalanced data
        rus = RandomUnderSampler(random_state=42)
        X_res = []
        y_res = []
        for i in range(5):
            temp1, temp2 = rus.fit_sample(datasets[i], targets[i])
            X_res.append(temp1)
            y_res.append(temp2)

        # Export preprossesing datasets and targets
        dump(X_res,'datasets_RUS.pd')
        dump(y_res, 'targets_RUS.pd')

Out[9]: ['targets_RUS.pd']
```

Apéndice D

Fase: Descubrimiento de patrones, implementación

FIGURA D.1: Código: etapa de Descubrimiento de patrones. Parte 1 (Elaboración propia).

```
In [1]: %matplotlib inline
        import os
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import matplotlib.cm as cm
        import matplotlib as mpl

        from joblib import load, dump
        from imblearn.pipeline import make_pipeline
        from sklearn.preprocessing import Normalizer
        from sklearn.decomposition import PCA
        from sklearn.svm import LinearSVC, SVC
        from sklearn.model_selection import train_test_split, cross_val_score
        from sklearn.naive_bayes import GaussianNB
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.neural_network import MLPClassifier
        from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
        from sklearn.metrics import roc_auc_score, cohen_kappa_score, classification_report
        from sklearn.ensemble import ExtraTreesClassifier, GradientBoostingClassifier
        from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier

In [3]: # Load preprocessed dataset
        X_res = load('datasets_RUS.pd')
        y_res = load('targets_RUS.pd')

In [4]: # Definition of metrics name
        METRICS_NAME = ('Accuracy',
                        'Precision',
                        'Recall',
                        'f1_score',
                        'ROC',
                        'Coeficiente Kappa de Cohen',)

        # Definition of classifier names
        CLASSIFIERS_NAME = (
            'GaussianNB',
```

FIGURA D.2: Código: etapa de Descubrimiento de patrones. Parte 2 (Elaboración propia).

```
        'KNeighbors',
        'MLP',
        'LinearSVC',
        'SVCSigmoid',
        'SVCPoly',
        'AdaBoost',
        'Bagging',
        'GradientBoosting')

    # Build classifiers
    CLASSIFIERS = [
        GaussianNB(priors=None),
        KNeighborsClassifier(n_neighbors=30),
        MLPClassifier(solver='adam',learning_rate_init=0.01),
        LinearSVC(),
        SVC(kernel='sigmoid'),
        SVC(kernel='poly'),
        AdaBoostClassifier(),
        BaggingClassifier(),
        GradientBoostingClassifier()]

In [6]: metrics_values = []

    for i in range(5):
    #     Build, fixed, transform and export pipeline
        pipeline = make_pipeline(Normalizer(), PCA())
        X = pipeline.fit_transform(X_res[i], y_res[i])
        dump(X, '{}.ppl'.format(i))

    #     Split the dataset in train and test
        X_train, X_test, y_train, y_test = train_test_split(X, y_res[i], test_size=0.33)

        metrics = []

        for clf, clf_name in zip(CLASSIFIERS, CLASSIFIERS_NAME):
    #         Definition of type of evaluation
            scores = cross_val_score(clf,X_train,y_train,cv=3,scoring='f1_micro')
    #     Train the classifier
            clf.fit(X_train,y_train)
    #     Predict the test dataset
            y_preds = clf.predict(X_test)
    # Build metrics
            METRICS = (accuracy_score(y_test, y_preds),
                       precision_score(y_test, y_preds),
                       recall_score(y_test, y_preds),
                       f1_score(y_test, y_preds, average='micro'),
                       roc_auc_score(y_test, y_preds),
                       cohen_kappa_score(y_test, y_preds),)
```

FIGURA D.3: Código: etapa de Descubrimiento de patrones. Parte 3 (Elaboración propia).

```
#       Calc metrics
        metrics.append([m for m in METRICS])
#     Export classifier
        dump(clf, '{} with {}.clf'.format(clf_name, i))
#     Create report from classification
        report = classification_report(
            y_test, y_preds, labels=None,
            target_names=None,
            sample_weight=None,
            digits=2)
#     Export report from classification
        dump(report, '{}_with_{}.report'.format(clf_name, i))
#     Save metrics for classifier
    metrics_values.append(metrics)
    plt.show()

# Export metrics
dump(metrics_values, 'metric_values.me')
```

Apéndice E

Fase: Análisis de los patrones, implementación

FIGURA E.1: Código: etapa de Análisis de los patrones. Parte 1 (Elaboración propia).

```
In [1]: %matplotlib inline
        import os
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import matplotlib.cm as cm
        import matplotlib as mpl

        from joblib import load, dump
        from imblearn.pipeline import make_pipeline
        from sklearn.preprocessing import Normalizer
        from sklearn.decomposition import PCA
        from sklearn.svm import LinearSVC, SVC
        from sklearn.model_selection import train_test_split, cross_val_score
        from sklearn.naive_bayes import GaussianNB
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.neural_network import MLPClassifier
        from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
        from sklearn.metrics import roc_auc_score, cohen_kappa_score, classification_report
        from sklearn.ensemble import ExtraTreesClassifier, GradientBoostingClassifier
        from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier

In [3]: # Load preprocessed dataset
        X_res = load('datasets_RUS.pd')
        y_res = load('targets_RUS.pd')

In [4]: # Definition of metrics name
        METRICS_NAME = ('Accuracy',
                        'Precision',
                        'Recall',
                        'f1_score',
                        'ROC',
                        'Coeficiente Kappa de Cohen',)

        # Definition of classifier names
        CLASSIFIERS_NAME = (
            'GaussianNB',
```

FIGURA E.2: Código: etapa de Análisis de los patrones. Parte 2 (Elaboración propia).

```
        'KNeighbors',
        'MLP',
        'LinearSVC',
        'SVCSigmoid',
        'SVCPoly',
        'AdaBoost',
        'Bagging',
        'GradientBoosting')

    # Build classifiers
    CLASSIFIERS = [
        GaussianNB(priors=None),
        KNeighborsClassifier(n_neighbors=30),
        MLPClassifier(solver='adam',learning_rate_init=0.01),
        LinearSVC(),
        SVC(kernel='sigmoid'),
        SVC(kernel='poly'),
        AdaBoostClassifier(),
        BaggingClassifier(),
        GradientBoostingClassifier()]

In [6]: metrics_values = []

    for i in range(5):
    #     Build, fixed, transform and export pipeline
        pipeline = make_pipeline(Normalizer(), PCA())
        X = pipeline.fit_transform(X_res[i], y_res[i])
        dump(X, '{}.ppl'.format(i))

    #     Split the dataset in train and test
        X_train, X_test, y_train, y_test = train_test_split(X, y_res[i], test_size=0.33)

        metrics = []

        for clf, clf_name in zip(CLASSIFIERS, CLASSIFIERS_NAME):
    #         Definition of type of evaluation
            scores = cross_val_score(clf,X_train,y_train,cv=3,scoring='f1_micro')
    #     Train the classifier
            clf.fit(X_train,y_train)
    #     Predict the test dataset
            y_preds = clf.predict(X_test)
    # Build metrics
            METRICS = (accuracy_score(y_test, y_preds),
                       precision_score(y_test, y_preds),
                       recall_score(y_test, y_preds),
                       f1_score(y_test, y_preds, average='micro'),
                       roc_auc_score(y_test, y_preds),
                       cohen_kappa_score(y_test, y_preds),)
```

FIGURA E.3: Código: etapa de Análisis de los patrones. Parte 3 (Elaboración propia).

```
#       Calc metrics
        metrics.append([m for m in METRICS])
#     Export classifier
        dump(clf, '{} with {}.clf'.format(clf_name, i))
#     Create report from classification
        report = classification_report(
            y_test, y_preds, labels=None,
            target_names=None,
            sample_weight=None,
            digits=2)
#     Export report from classification
        dump(report, '{}_with_{}.report'.format(clf_name, i))
#     Save metrics for classifier
    metrics_values.append(metrics)
    plt.show()

# Export metrics
dump(metrics_values, 'metric_values.me')
```