Universidad Nacional de Luján

Clasificación de flujos de datos continuos y multi etiquetados

Tesina de grado presentada para optar al título de
Licenciado en Sistemas de Información

Juan Cruz Cardona

Director: Santiago Banchero

2021

CLASIFICACIÓN DE FLUJOS DE DATOS CONTINUOS Y MULTI ETIQUETADOS

La clasificación multi-etiquetas es un paradigma de aprendizaje supervisado que generaliza las técnicas clásicas de clasificación para abordar problemas en donde cada instancia de una colección se encuentra asociada a múltiples etiquetas. La mayor parte de los trabajos de investigación han sido realizados en contextos de aprendizaje por *batch*. Los ambientes de flujo continuo de datos (o *streaming*) presentan nuevos desafíos a esta área debido a las limitaciones de tiempo de respuesta y almacenamiento que acarrean. A esto se agrega la naturaleza evolutiva de este tipo de escenarios, que obligan a los algoritmos a adaptarse a cambios de concepto. En la presente investigación se aplican algoritmos de clasificación multi-etiquetas a colecciones estructuradas y no estructuradas. Los experimentos se llevarán a cabo en ambientes simulados de *streaming* de datos para conocer el impacto que produce este contexto sobre los resultados de la clasificación y acoplar el modelo a escenarios del mundo real. A su vez, se partirá de estas colecciones de datos para generar instancias sintéticas y así producir flujos potencialmente infinitos. Por último, se abordarán estrategias de ensambles de algoritmos en búsqueda de una mejora en la calidad de la tarea de predicción de objetos no observados por el modelo. De esta manera, se proveerá a la comunidad de nuevos estudios experimentales sobre algoritmos y colecciones ya conocidos del área de clasificación multi-etiquetas, de manera tal de extender el conocimiento sobre su rendimiento bajo escenarios evolutivos y de naturaleza variable.

Palabras claves: clasificación, multi-etiquetas, *streaming*, algoritmos, flujos.

Índice general

1. INTRODUCCIÓN

1.1. Fundamentos

En los últimos años ha habido un aumento considerable de datos de diversa índole y generados por fuentes heterogéneas. Según los autores Gantz y Reinsel, el volumen total de datos creados y replicados en el mundo durante el año 2011 supera los 1.8 ZB (zettabytes) y se ha estimado que duplica cada dos años [14]. Los avances en el área de Tecnología de la Información (TI) han contribuido a una continua producción de datos y expansión del campo digital, tal es el caso para la red social *Facebook*, la cual recibe cada hora un flujo de 10 millones de fotos que publican sus usuarios [26]. A estas grandes colecciones de datos se las conoce como *big data* y acarrean nuevas oportunidades y desafíos al campo de las ciencias de la computación. En cuestiones económicas, un análisis a gran escala en búsqueda de tendencias en el comportamiento de los usuarios o clientes de un sistema puede dar una ventaja competitiva en el mercado y, en adición, proveer de un servicio valioso a la comunidad. Potencialmente, la *big data* puede ser una fuente que proporcione a la comunidad de conocimiento nuevo sobre el mundo en el que habita, o como ha mencionado Fayyad, Piatetsky-Shapiro y Smyth en su escrito sobre el descubrimiento de conocimiento:

> "Los datos que percibimos de nuestro ambiente son la evidencia básica que usamos para construir teorías y modelos sobre el universo en el que vivimos" [1]

Sin embargo, volúmenes masivos de datos tornan obsoletos los tradicionales métodos manuales de análisis de datos y surge la necesidad de desarrollar técnicas automatizadas para extraer patrones en los datos y obtener conocimiento. Con este fin, se han desarrollado técnicas en las áreas de minería de datos y aprendizaje de máquinas que abordan estas colecciones en búsqueda de conocimiento válido y útil. Dichas técnicas se han enfocado en el aprendizaje por *batch* [12], lo que significa que el algoritmo dispone de la colección completa, almacenada en disco, y con la cual genera un modelo a partir de una o múltiples iteraciones sobre todos los datos. No obstante, el aprendizaje por *batch* trae aparejada una dificultad en su misma definición: requiere de todos los datos de la colección presentes y accesibles en todo momento, lo cual no siempre es posible. Además se suma una limitante que es clave en el contexto actual de alta disponibilidad de datos: hoy en día una buena parte de los datos generados proviene de flujos continuos o "*streamings*" de datos [2]. Estos flujos son potencialmente ilimitados, arriban de a una instancia por vez, y son analizados con restricciones altas de tiempo de procesamiento y de memoria. Tal es el caso para aplicaciones de sensores, monitoreo de redes y administración de tráfico, flujo de clics de un usuario en la web, redes sociales, entre otros. Los algoritmos de aprendizaje que actúen en este entorno dinámico deben contar con mecanismos que permitan manejar cambios en la naturaleza o distribución de los datos, tanto para incorporar datos nuevos, como para descartar los datos antiguos. Por estas razones, se torna necesario que las aplicaciones basadas en clasificación en tiempo real adapten sus operaciones de entrenamiento y predicción para lograr mejores resultados [39].

Dentro del área de minería de datos, una de las principales tareas es la de clasificación, la cual consiste en entrenar un modelo que sea capaz de asignar una única etiqueta a una instancia desconocida. No obstante, existen problemas de clasificación en donde múltiples

[1] "Data we capture about our environment are the basic evidence we use to build theories and models of the universe we live in" [11, p. 2]. Traducción propia.

etiquetas son necesarias para caracterizar una instancia. Por ejemplo, una noticia de diario referida al accidente aéreo que sufrió el plantel de fútbol del club Chapecoense puede ser clasificado en la categoría de "Fútbol" tanto como en la de "Tragedias". Del mismo modo, un video documental sobre la vida de Borges puede anotarse como "Biografía", "Literatura" o incluso "Buenos Aires" si se mostraran imágenes de la ciudad. Este tipo de problemas es llamado Clasificación multi-etiquetas (MLL)[2] y representa un nuevo paradigma de aprendizaje automático, con sus propios retos por afrontar y que aún no ha sido suficientemente explorado en proyectos de investigación.

Una clasificación multi-etiqueta permite conocer el grado de correlación entre una instancia de la colección y una o más etiquetas. Esta cualidad significa un mayor poder de generalización con respecto a la clasificación tradicional de única etiqueta, ya que puede abarcar esos mismos problemas y otros de mayor número de etiquetas. Además existen algoritmos que aprovechan la correlación entre etiquetas para mejorar la eficiencia de la clasificación y la calidad de la predicción.

El campo de MLL se ha desarrollado considerablemente en los últimos años pero hasta el momento muchos de estos trabajos se han llevado a cabo en ambientes estáticos de aprendizaje por *batch* [35], en consecuencia, se hace necesario encarar nuevos proyectos que aborden clasificaciones MLL en contextos de *streaming* de datos. El desafío entonces consiste en crear clasificadores que sean capaces de manejar un inmenso número de instancias y adaptarse al cambio, a la vez que estar preparados para hacer tareas de predicción en cualquier momento, y todo esto en un contexto de altas restricciones de tiempo de respuesta y memoria.

1.2. Descripción del tema de estudio

1.2.1. Clasificación Multi-etiquetas

Tradicionalmente, el aprendizaje supervisado ha consistido en asociar una instancia o ejemplo a una única etiqueta. Dicho ejemplo es una representación de un objeto del mundo real, y por lo tanto, consta de características o atributos particulares. La etiqueta corresponde a un significado semántico o concepto que lo caracteriza. La tarea de clasificación entonces, reside en aprender una función que permita enlazar ejemplos no observados con una etiqueta. Es preciso notar aquí que dicha definición encubre la restricción de que cada instancia pertenece a una única etiqueta, o dicho de otra manera, cada objeto del mundo real se asocia a un único concepto y ningún otro. Sin embargo, existen problemas de clasificación donde más de una etiqueta puede ser asignada a un ejemplo. La anterior presunción no se amolda a problemas complejos donde un objeto pueda tener más de un significado simultáneamente.

Tareas de este tipo pueden surgir en áreas como las de categorización de texto, recuperación de información musical, clasificación semántica de escenas, anotación automática de videos o clasificación de genes y funciones proteicas. A modo de ejemplo, en el campo mencionado de clasificación semántica de escenas, la foto de un paisaje que ilustra una montaña y una playa puede asociarse a las categorías de 'playa' y 'montaña', simultáneamente [16]; en bioinformática, cada gen puede ser asociado a clases según su función, tales como 'metabolismo', 'transcripción' o 'síntesis proteica' [46]; por último, en recuperación de información musical una pieza sinfónica puede tener *tags* como 'Mozart', 'piano' o 'clásica'.

[2] Siglas provenientes de su abreviación en inglés, Multi-label learning

Este nuevo paradigma es llamado 'Clasificación multi-etiquetas' y ataca problemas con las siguientes características [16]:

- El conjunto de etiquetas es previamente definido y tiene un significado interpretable por un humano.
- El número de etiquetas es limitado y no mayor que el número de atributos.
- En caso que el número de atributos sea grande, se debe poder aplicar estrategias de reducción de atributos.
- El número de ejemplos puede ser grande.
- Las etiquetas pueden estar correlacionadas. Esto significa que se pueden aplicar técnicas que exploten estas relaciones con el objetivo de reducir los tiempos de procesamiento de los algoritmos.
- La distribución de los datos puede estar desbalanceada, es decir, que una etiqueta puede tener un mayor número de ejemplos que otras.

Asimismo, surge un desafío a superar: el conjunto de etiquetas posible crece exponencialmente ante cada nueva adición de una etiqueta. Por ejemplo, si se tuvieran 20 etiquetas, la cantidad posible de conjuntos de etiquetas distintos excedería el millón (2^{20}). Esto implica un tamaño exorbitante del espacio de salida y, en consecuencia, costos computacionales altos. En ese sentido, se ha buscado desarrollar algoritmos que aprovechan las correlaciones o dependencias entre etiquetas. Por ejemplo, la probabilidad de que una noticia que contiene los términos 'pelota' y 'gol' sea anotada con la etiqueta 'fútbol' sería mayor que si se etiquetara con la etiqueta 'tenis'. Zhang y Zhang clasifican estos algoritmos en tres grupos según la estrategia de correlación aplicada [46]:

Estrategia de primer orden La tarea de MLL es dividida en q tareas de clasificación binarias, siendo q el número de etiquetas de la colección.

Estrategia de segundo orden La tarea de MLL se basa en la generación de relaciones de a pares de etiquetas ya sea por *rankings* entre clases relevantes y no relevantes o por interacción entre pares de etiquetas.

Estrategia de alto orden La tarea de MLL considera relaciones de alto orden entre etiquetas.

Las estrategias de primer orden son conceptualmente simples y eficientes pero logran resultados de menor calidad ya que no consideran correlaciones. Las estrategias de segundo orden tienen un mayor poder de generalización pero no todos los problemas de MLL pueden ser abarcados. El último grupo, por su parte, modela las correlaciones más potentes pero conlleva un costo computacional alto.

En el último tiempo, muchos son los algoritmos que han sido desarrollados para atacar el problema de la clasificación MLL. La comunidad de investigación ha aceptado la taxonomía definida por Tsoumakas y Katakis, para estudiar y clasificar los distintos algoritmos de la literatura [16]. La misma propone dos grandes grupos:

Métodos de transformación del problema Este tipo de algoritmos transforman el problema de clasificación MLL en un problema de clasificación tradicional. Ejemplos típicos de este grupo son Binary Relevance (BR) [41] y Classifier Chains (CC) [35]. Ambos convierten la tarea en una de clasificación binaria.

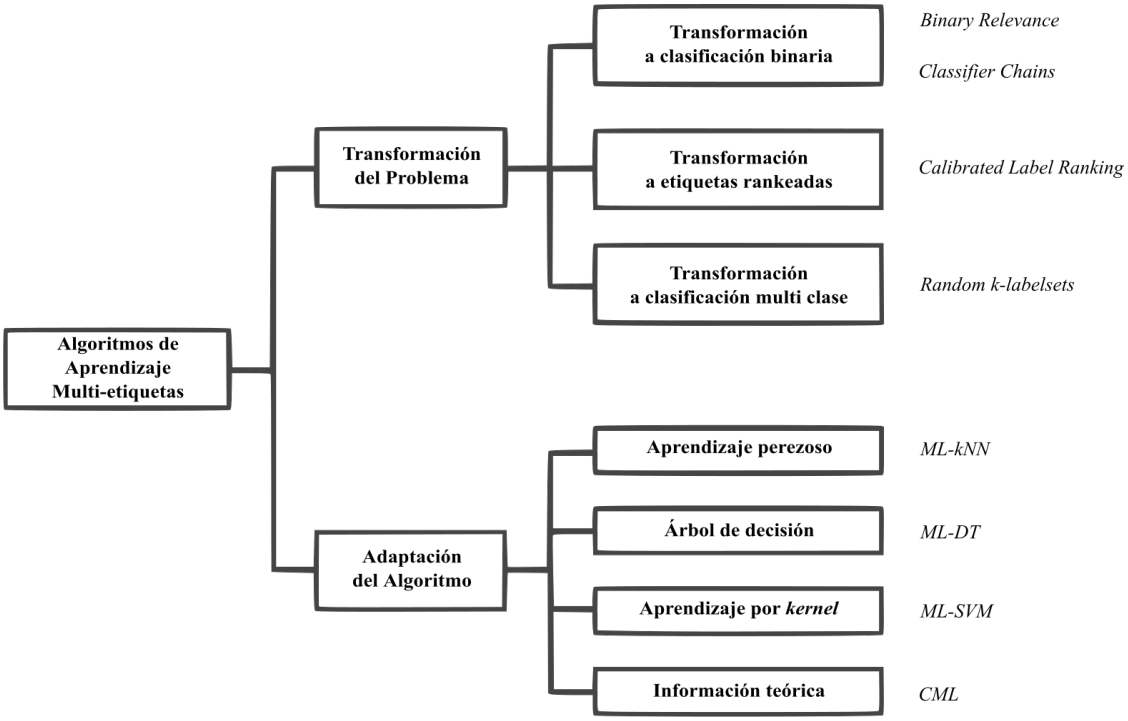

Fig. 1.1: Categorización de los algoritmos de MLL más representativos.

Métodos de adaptación del algoritmo Son algoritmos que toman métodos tradicionales de aprendizaje, como árboles de decisión o *naive bayes*, y los adaptan a la tarea de clasificación MLL.

A modo ilustrativo, la figura 1.1 es un diagrama de la taxonomía de algoritmos confeccionado por Zhang y Zhou [47].

1.2.2. Flujos continuos de datos

Hoy en día, los datos pueden ser generados por elementos de continuo monitoreo del medio, tales como sensores o archivos de registro. La clasificación de flujos continuos de datos o *streamings* se enfoca en problemas de este tipo, en donde objetos del mundo real son analizados en tiempo real. Un *stream* se considera como una secuencia ordenada de datos que fluye a alta velocidad y es teóricamente infinita. A continuación se describe brevemente las características de este tipo de datos y los requisitos que debe cumplir un algoritmo para poder tratar con ellos.

1.2.2.1. Características

En escenarios de *streamings*, los datos deben cumplir con las siguientes características [12]:

- Los datos están disponibles a través de flujos continuos e ilimitados en el tiempo, a diferencia del aprendizaje por *batch* donde los datos son acotados.
- Las regularidades subyacentes de los datos no son estacionarias sino que pueden evolucionar. En otras palabras, la distribución de los datos es susceptible a cambios en el tiempo.

- La data ya no es considerada independiente e idénticamente distribuida.
- La data está situada tanto en el espacio como en el tiempo. La lectura y procesamiento de los datos debe ser lo suficientemente veloz como para procesar el siguiente dato, de otro modo, el dato ya no podrá ser procesado en el futuro.

1.2.2.2. Requisitos de los algoritmos

Para hacer frente a estas características de los datos, los sistemas o algoritmos deben contar con los siguientes requisitos [19]:

- El tiempo para procesar cada registro debe ser constante y pequeño.
- Debe usar un tamaño fijo de memoria principal y no dependiente de la cantidad de registros ya procesados.
- El modelo es generado a partir de una única pasada sobre los datos.
- Debe contar con un modelo listo para realizar predicciones en cualquier momento.
- Idealmente deberá producir un modelo equivalente o casi idéntico al que hubiera sido producido en un ambiente de *batch*.
- Debe mantenerse actualizado ante evoluciones o derivas de concepto en los datos.

Los algoritmos que cumplen con estas cualidades son llamados algoritmos de aprendizaje adaptativo y, aplicados a tareas de clasificación como el de multi-etiquetas, pueden aprovecharse para entrenar un mayor número de objetos y predecir en cualquier instante.

1.3. Motivación

Ante la necesidad de hacer frente a un contexto global de generación masiva de datos y a ritmo acelerado, se hace preciso fortalecer las técnicas de aprendizaje automático actualmente presentes en el campo. En este escenario ya no es posible contar con todos los datos almacenados físicamente y la idea de generar un modelo completo para luego evaluarlo en una fase posterior debe ser reemplazada por una en donde el modelo esté siempre listo para realizar predicciones y al mismo tiempo ser capaz de re-entrenarse y recalcular las métricas de evaluación ante cada nueva instancia abordada. Todo esto en un contexto cambiante, de alta disponibilidad y de limitación en el espacio de almacenamiento. Si bien existen métodos de clasificación para flujos continuos que han dado resultados satisfactorios, aún es un campo conveniente de ser abordado. Asimismo, reproducir los experimentos realizados y fortalecer las técnicas y herramientas actuales puede ser beneficioso para lograr estudios precisos y pormenorizados que sean valiosos tanto para la comunidad científica en sí misma, como también para la sociedad en general.

Por otro lado, si bien existen en el mundo real infinidad de datos multi-etiquetados aún no es posible hallar colecciones disponibles al público que cuenten con todas las características de un flujo continuo de datos. Uno de los enfoques abordados es convertir las colecciones existentes en flujos tales que arriben en conjuntos predefinidos y a lo largo del tiempo. De esta manera los algoritmos pueden ser utilizados para realizar clasificaciones en un ambiente similar al de un escenario de *streaming*. Sin embargo, estas colecciones tienen un número limitado de instancias y por lo tanto no cumplen con la condición de ser teóricamente infinitos. Es entonces aquí donde surgen las técnicas de generación sintética

de instancias, que buscan reproducir la distribución subyacente de los datos para simular colecciones de datos del mundo real. La contracara de este enfoque es que, si bien existen técnicas y herramientas para generar datos etiquetados, buena parte de ellos son solo aplicables para instancias de una única etiqueta y los que logran generar datos multi-etiquetados no han sido lo suficientemente explorados en el área. Hasta el momento, los generadores de instancias multi-etiquetadas son capaces de generar datos cercanos a los de colecciones del mundo real [32] y brindan la posibilidad de realizar estudios relativamente certeros de algoritmos de clasificación [36]. No obstante, debe notarse también que si bien se han obtenido colecciones sintéticas en sí mismas aún no han logrado generar instancias para una colección en concreto, respetando sus cualidades particulares y que las distinguen de otras, tales como la co-ocurrencia de etiquetas, la densidad y cardinalidad de las etiquetas y la relación entre las etiquetas y sus atributos, por mencionar algunas. De lograr esta aproximación se podrán realizar estudios sobre el impacto de los algoritmos sobre flujos de datos de naturaleza distintiva, o en otras palabras, entender en qué medida un algoritmo es más apropiado que otro para un conjunto de datos en un determinado contexto.

Esta última idea mencionada, es decir, el ser capaz de hallar las fortalezas y debilidades de un algoritmo de MLL en un contexto determinado es clave para evaluar la clasificación y entender los resultados obtenidos. Estudios como el de Sousa y Gama [39] o el de Read y col. [36] se han topado con que algoritmos de menor complejidad pueden ser competitivos o incluso superar las métricas de otros algoritmos más complejos. Esta variabilidad en los resultados no solo contrae la necesidad antes mencionada de realizar más estudios al respecto, sino que también abre las puertas a incursionar en soluciones de ensambles de algoritmos. Estos ensambles han dado probada muestra de potenciarse ante la diversidad de resultados obtenidos por sus estimadores base [31], ya que son capaces de disminuir el error total mediante estrategias combinativas. De cualquier manera, las estrategias de ensamble existentes para flujos continuos no han recibido la misma atención que aquellas aplicadas sobre ambientes de *batch* y queda mucho camino por recorrer.

1.4. Objetivos

El presente trabajo tiene por objetivo principal estudiar el impacto de distintos algoritmos de clasificación sobre datos multi-etiquetados en ambientes de *streaming* provenientes de fuentes de distinto origen y naturaleza, en particular se seleccionan colecciones de datos que son puntos de referencia en la literatura y que poseen características distintivas entre sí, tales como el número de etiquetas, el número de atributos, o la cantidad de instancias. A su vez, es necesario convertir estos datos a flujos continuos y a este fin se generan instancias sintéticas que sean fieles a estas características mencionadas, aplicando técnicas existentes pero también extendiéndolas para detectar co-ocurrencias entre etiquetas. De esta manera, se buscan obtener representaciones óptimas de las colecciones. Finalmente, se llevarán a cabo clasificaciones con algoritmos clásicos y con soluciones de ensambles, en búsqueda de maximizar los valores de las métricas de evaluación en cada escenario. Adicionalmente, se diseñan distintas configuraciones de ensambles, variando los estimadores base y probando distintas implementaciones. Con esto último en mente, se desarrolla una versión del algoritmo de mayoría de voto para el lenguaje Python y se compara su rendimiento contra el de las implementaciones de Java .

describir qué es este algoritmo de mayoría de voto

referencia bibliografica

no estan los expe-

Con esto en mente, se listan a continuación los objetivos particulares del trabajo:

- Obtener colecciones de datos que cumplan con las propiedades requeridas para considerarse un flujo continuo. Las características deben variar entre colecciones. Cada

colección debe tener las instancias propias del juego de datos e instancias sintéticas potencialmente ilimitadas.

- Generar flujos continuos de datos a partir de la colección proporcionada replicando su número de etiquetas, atributos e instancias, su cardinalidad y densidad de etiquetas y la co-ocurrencia entre dos etiquetas.
- Ejecutar algoritmos de clasificación de MLL, para obtener modelos y realizar evaluaciones sobre los resultados. Todo esto sobre distintos escenarios de flujos continuos.
- Proponer una solución de ensambles a partir de la combinación de algoritmos seleccionados de la literatura.

1.5. Aportes

El presente trabajo de investigación aborda el campo de aprendizaje por multi-etiquetas a partir de la experimentación y evaluación de técnicas y algoritmos de la literatura sobre colecciones de naturaleza cambiante. MLL es un paradigma emergente de aprendizaje supervisado cuyas características implícitas abren paso a nuevos desafíos que derivan del crecimiento exponencial de etiquetas y sus combinaciones, y del costo computacional de entrenar y consultar el modelo. También suelen presentarse otras propiedades como la alta dimensionalidad, data evolutiva y desbalanceada o dependencia entre etiquetas, las cuales implican una re-significación de las técnicas y métodos tradicionales del área de minería de datos.

El paradigma de MLL ha dado muestras de su eficiencia en términos de tiempos de configuración y ejecución de las tareas, bajo diversos campos de aplicación tales como los de categorización de texto, diagnósticos médicos, minería de redes sociales o análisis de datos químicos, y se mantiene en constante expansión hacia nuevos dominios de aplicación. Asimismo, su continua integración a problemas de diversa naturaleza ha contribuido a alimentar esta tendencia.

La tarea de clasificación de *streaming* de datos se enfoca en problemas donde objetos del mundo real son generados y procesados en tiempo real. Datos de este tipo, y que además poseen múltiples etiquetas, son frecuentes en escenarios del mundo real tal como sitios de publicación de imágenes, correos electrónicos o portales de noticias. Abordar este tipo de problemas implica que los algoritmos sean capaces de identificar cambios de concepto en los datos y adaptarse al nuevo contexto. De lograr esto, se podrá generar modelos más sólidos, ya que se cuenta con un mayor número de objetos, y que se encontrarán aptos para predecir en cualquier momento. Surge entonces el reto de crear clasificadores que actúen en ambientes de altas restricciones computacionales y sean capaces de manejar un inmenso número de instancias, lidiar con evoluciones en los datos y estar listos para resolver tareas de predicción en tiempo real.

A diferencia de otros trabajos de investigación recientes, este proyecto lleva a cabo estudios experimentales sobre el tema de clasificaciones multi-etiquetas, para hallar las fortalezas y debilidades de distintos algoritmos de aprendizaje sobre distintos tipos de colecciones, con miras a aportar de un mayor conocimiento empírico sobre el tema a la comunidad científica especializada en tareas de clasificación de flujos de datos multi-etiquetados. El presente trabajo espera contribuir al estudio de métodos y técnicas asentadas, pero también examinar algoritmos exitosos del campo de aprendizaje por *batch*, particularmente los de ensambles de estimadores, a fin de extender su funcionalidad a ambientes de flujos continuos, analizar su desempeño y determinar en qué medida son aptos o no para este tipo de ambientes.

1.6. Organización del Trabajo

Describir las distintas secciones del trabajo

2. PRELIMINARES

En este capítulo se presenta el marco teórico de este trabajo, dando un panorama general de cada una de las disciplinas abordadas e introduciendo los conceptos básicos y fundamentales para entender el proyecto. Se comienza con la definición de la taxonomía del campo de estudio, luego

describir las siguientes secciones

2.1. Taxonomía del Campo de Estudio

En pocas palabras, el presente trabajo de investigación se enmarca en las áreas de *big data* y minería de datos, con aplicación en escenarios de *streaming* o flujos continuos de datos y abordando clasificaciones multi-etiquetas.

La figura 2.1 es un esquema que ilustra la taxonomía del campo de estudio y la interrelación entre las áreas de investigación involucradas.

2.2. Aprendizaje Automático

El aprendizaje automático, también conocido por su término en ingles "*Machine Learning*", se enmarca dentro del área de la Inteligencia Artificial (IA) y estudia cómo las computadoras pueden "aprender" o mejorar su rendimiento meramente a partir de datos y sin la intervención de un ser humano. La idea detrás de esta disciplina es lograr reconocer patrones subyacentes en los datos y tomar decisiones basándose en ellos. Por ejemplo, un problema de aprendizaje automático es el de reconocer dígitos escritos a mano a partir de un conjunto de ejemplos (ver figura 2.2). Aquí se tienen un conjunto de imágenes, cada una representando un dígito del 0 al 9, y el objetivo es construir un modelo que sea capaz de detectar de qué dígito se trata. Otro ejemplo es el de hallar documentos de texto que son relevantes a una consulta del usuario. En este caso el modelo recibe un conjunto acotado de términos, los cuales describen una necesidad de información del usuario, y el modelo debe ser capaz de retornar los documentos que satisfacen la consulta.

Estos problemas se suelen categorizar en aprendizaje supervisado o no supervisado, de acuerdo a si se conoce o no de antemano el concepto o etiqueta que define a los datos. Se desarrollará más sobre este punto en las próximas secciones . De entre los problemas de aprendizaje supervisado se destaca aquí el de clasificación, el cual será descrito a continuación.

expandir la definición de supervisada vs no supervisada

2.3. Clasificación

2.3.1. Definición

La clasificación es una tarea de minería de datos muy popular que consiste en hallar modelos que describen la o las clases intrínsecas de los datos. La clase corresponde a un concepto que representa al dato y es una etiqueta categórica, es decir, un valor discreto de entre un conjunto de valores previamente conocidos. Estos modelos, también llamados clasificadores, son capaces de predecir la clase a la que corresponden datos previamente desconocidos. Por ejemplo, se puede construir un modelo de clasificación para categorizar nuevos correos electrónicos de acuerdo a si se trata de correo basura (también conocido como "*spam*") o no. Dicho análisis puede ayudar a obtener un mayor entendimiento de

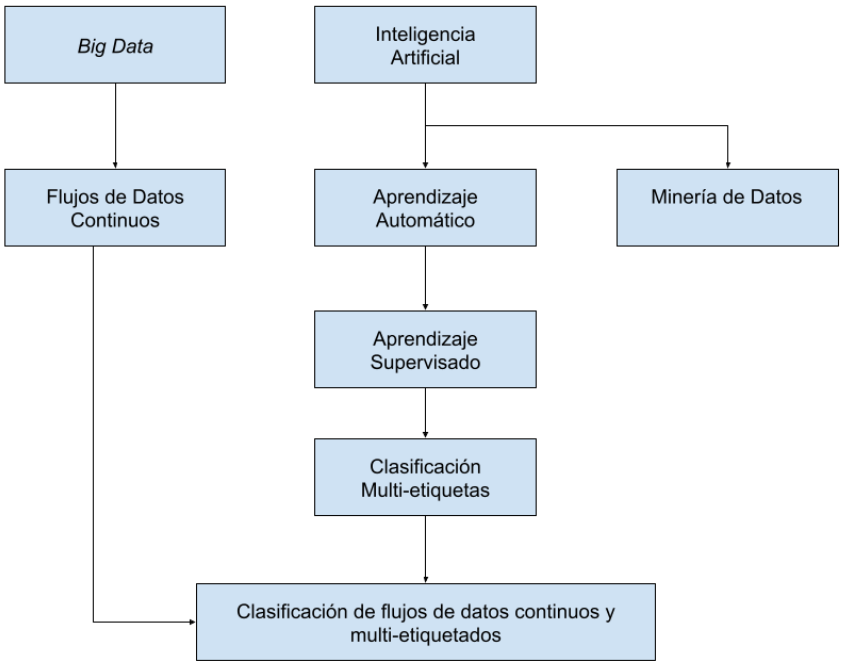

Fig. 2.1: Taxonomía del campo de estudio.

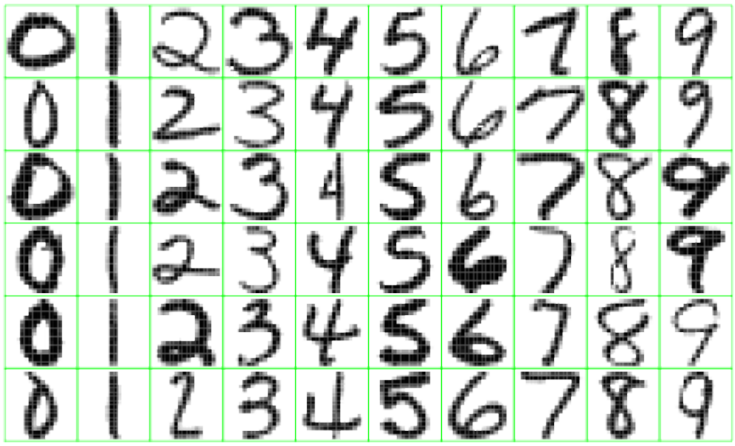

Fig. 2.2: Dígitos escritos a mano. Fuente: *The Elements of Statistical Learning* (2009).

los datos a alto nivel. Las tareas de clasificación han sido aplicadas en áreas tales como las de aprendizaje automático, reconocimiento de patrones o estadística.

En un principio, buena parte de los algoritmos se ejecutaban en memoria, con la limitación de espacio de almacenamiento que eso conlleva. Investigaciones más recientes han desarrollado técnicas para escalar los algoritmos de tal manera que puedan manejar datos de mayor tamaño, alojados en memoria, en disco o procesados bajo demanda. Las aplicaciones para este tipo de tareas son numerosas y entre ellas se encuentran las de detectar fraudes o realizar diagnósticos médicos, entre otras.

La clasificación de datos consta de dos etapas, una de aprendizaje y otra de clasificación o predicción. Durante la tarea de aprendizaje se construye el modelo de clasificación el cual describe un determinado número de clases o conceptos. También se conoce esta etapa como la de entrenamiento ya que se selecciona un subconjunto de los datos, llamado conjunto de entrenamiento, que consta de instancias o tuplas seleccionadas aleatoriamente y con una o más etiquetas asociadas. Formalmente, el problema de clasificación puede ser formulado de la siguiente manera. Se recibe un conjunto etiquetado de instancias, tupas o ejemplos de la forma (X, y) donde cada tupla es un vector $X = (x_1, x_2, \ldots, x_n)$, siendo cada valor una característica distintiva, atributo o *feature* de la instancia. El vector y por su parte toma un valor de entre n clases diferentes.

Este tipo de tareas se engloban dentro del campo de aprendizaje supervisado ya que para cada instancia la etiqueta es conocida de antemano, y es aprovechada para guiar o, siguiendo la metáfora, "supervisar" el aprendizaje del clasificador. Esta es la diferencia principal contra algoritmos de aprendizaje no supervisado, en los cuales la etiqueta no es conocida y se deben aplicar técnicas para salvar esta restricción.

La primera etapa de una clasificación puede ser vista también como el aprendizaje de una función $y = f(X)$ que pueda predecir la clase y para una tupla X. Por ejemplo, X podría ser un mensaje de correo y la etiqueta y la decisión de si se trata de un correo basura o no. Desde esta perspectiva queremos aprender una función que sea capaz de distinguir las clases subyacentes. Usualmente, esta asociación es llevada a cabo por algoritmos de aprendizaje, los cuales internamente usan funciones matemáticas o reglas de decisión que les permiten procesar los atributos de entrada y generar una salida acorde. Algunos ejemplos de este tipo de algoritmos son los árboles de decisión, *naive* bayes, perceptrón, entre otros. Más adelante se retomará sobre este punto para describir en detalle algunos algoritmos representativos del campo.

En la segunda etapa el modelo es usado para clasificar y realizar predicciones sobre datos desconocidos. A este fin, se calcula un valor que refleja la calidad del clasificador y es denominado "métrica de evaluación". Una de ellas es la exactitud o *accuracy* pero no es la única. Durante la etapa de entrenamiento esta estimación puede ser imprecisa, tomando un valor que tiende a ser "optimista" o que da un valor de exactitud mayor al rendimiento real. Esto sucede porque el clasificador puede llegar a incorporar anomalías particulares en el conjunto de datos de entrenamiento, las cuales no tienen que ver tanto con el dominio de aplicación en el cual se enmarca la tarea, sino más bien con "ruido", datos erróneos o simplemente instancias que no reflejan correctamente los objetos del mundo real. Este fenómeno es llamado "sobreajuste" u "*overfit*" y se han diseñado técnicas para reducirlo. Una de ellas consiste en separar de entre los datos de la colección completa, un subconjunto conocido como "conjunto de prueba" o de *testing* que no se usa durante el entrenamiento y a partir del cual se realizan predicciones y se calculan las métricas de evaluación.

Así pues, la tarea de evaluación es fundamental ya que es la vía a partir de la cual se determina qué algoritmos o técnicas son más apropiados que otros para un problema en particular. Asimismo, provee la información necesaria para corregir o ajustar los

parámetros de los algoritmos y así obtener modelos más robustos.

En definitiva, las etapas de aprendizaje y predicción se aplican consecutivamente con el objetivo de lograr generar un clasificador capaz de predecir con éxito las etiquetas de instancias nuevas y a priori desconocidas por el modelo.

2.3.2. Algoritmos

Como se mencionó en la sección anterior, una de las etapas de la clasificación consiste en generar un modelo capaz de clasificar instancias no observadas. En esta etapa de aprendizaje, se pueden aplicar diversos tipos de algoritmos de clasificación de acuerdo a la naturaleza de la tarea en particular que se desea abordar. A continuación, se describen algunos de estos algoritmos en detalle a fin de ahondar sobre el concepto de clasificación en el aprendizaje automático. Además, más adelante estos algoritmos serán particularmente relevantes para el desarrollo del presente trabajo de investigación.

2.3.2.1. *Naive* Bayes

Naive Bayes es uno de los algoritmos que pertenecen a la familia de clasificadores probabilísticos y se destaca por ser computacionalmente simple, interpretación, y al mismo tiempo brinda un rendimiento competitivo en comparación con otros modelos más complejos. Se dice que es un clasificador estadístico ya que se basa en el teorema de Bayes. La idea es computar una probabilidad para cada una de las clases, basada en los atributos de la instancia y seleccionar aquella de mayor probabilidad. El término "*naive*" es el inglés para el término "*ingenuo*" y nace de la presunción que hace el algoritmo de que los atributos son independientes entre sí, o condicionalmente independientes. Esta presunción raramente se cumple en los escenarios donde se aplica pero contribuye a su simplicidad computacional y a su velocidad durante el entrenamiento.

Para entender cómo funciona este algoritmo, es bueno abordar primero el teorema de Bayes. Formalmente, se define de la siguiente manera:

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)} \tag{2.1}$$

En esta ecuación, el vector X es una tupla definida tal como en la sección anterior y en términos bayesianos representa la "evidencia". $P(X)$, por lo tanto, es la probabilidad de que la tupla contenga los atributos que efectivamente posee. Por su parte, H es la hipótesis de que la tupla pertenece a una determinada clase y $P(H)$ su probabilidad. Esta es conocida como probabilidad "a priori". De la misma manera, $P(H|X)$ es la probabilidad de que la hipótesis H sea cierta bajo la evidencia X. A esta se la llama probabilidad "a posteriori" con H condicionada por X y es el valor que se quiere determinar en una tarea de clasificación. Finalmente, $P(X|H)$ indica la probabilidad de que la tupla tenga unos atributos determinados dado que se satisface la hipótesis.

A partir de dicha definición, y de forma similar, se expresa la ecuación de *Naive* Bayes de la siguiente manera:

$$P(C_i \mid X) = P(X \mid C_i)P(C_i) \tag{2.2}$$

Aquí el término $P(X)$ es descartado ya que se asume constante para todas las clases. La hipótesis H es representada como C_i que es un valor de la tupla $C = (C_1, C_2, \ldots, C_q)$, donde q es el número de clases. La presunción "ingenua" es aplicada para el cálculo del término $P(X \mid C_i)$ gracias a lo cual se puede definir de la siguiente manera:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots, \times P(x_n \mid C_i) \tag{2.3}$$

Finalmente, el modelo seleccionará la clase que maximice el valor de probabilidad y esa será la salida final del algoritmo.

Como se ha dicho anteriormente, la simplicidad, velocidad computacional y su competitividad en métricas de exactitud hacen de *Naive* Bayes un algoritmo destacado en el campo de aprendizaje automático [44] y ha sido aplicado para problemas diversos, tales como el de hallar errores en programas de computación [1], predecir enfermedades del corazón [10] o detectar ataques en una red de computadoras [20].

2.3.2.2. Árboles de Decisión

Árboles de decisión es un modelo de clasificación que se destaca por ser de fácil interpretación e intuitivo para el ser humano. De hecho, se puede generar una representación gráfica del árbol generado para asistir a la comprensión del modelo y así entender a más alto nivel cómo se comporta durante una predicción. En cuanto a su estructura, un árbol de decisión contiene nodos, cada uno representando un atributo de la colección. Estos nodos se conectan con otros nodos a partir de enlaces o "ramas" que representan un valor o un rango de valores de ese atributo. Los nodos de menor jerarquía son llamados "hojas" y contienen la clase de la predicción, y el nodo de mayor jerarquía es llamado "raíz". Al momento de predecir una instancia nueva, la clasificación se realiza de la siguiente manera: se toma la instancia nueva, la cual no tiene una etiqueta asociada, y los valores de sus atributos son comparados contra los del árbol, luego se traza un camino desde el nodo raíz hasta la hoja. Finalmente, la clase que contiene la hoja es seleccionada y será parte de la predicción resultante.

Figura de un árbol

Los árboles de decisión se generan a partir de un algoritmo de inducción. Existen varios de estos algoritmos pero todos son variantes que han sido diseñadas bajo un misma principio: construir el árbol de una manera "voraz"[1], comenzando desde el nodo raíz (conocido como enfoque *top-down*) y eligiendo en cada paso el atributo más informativo o que maximice alguna medida de ganancia de información. Entre estos algoritmos de inducción vale destacar los siguientes:

ID3 Son las siglas de "*Iterative Dichotomiser 3*" y fue desarrollado en 1986 por Ross Quinlan. Consiste en crear un árbol de múltiples vías, buscando para cada nodo el atributo categórico que lance la mayor ganancia de información para las clases categóricas. Los árboles crecen en un tamaño máximo y luego se realiza el paso de poda para mejorar el poder de generalización del modelo sobre datos desconocidos.

C4.5 Es la evolución del algoritmo ID3. La principal mejora con respecto a su predecesor es que elimina la restricción de que los atributos deban ser categóricos. Esto lo consigue particionando el valor continuo en rangos o en un conjunto de intervalos discretos. A su vez, C4.5 convierte el árbol entrenado en conjuntos de reglas de decisión.

[1] Se le llama voraz o *greedy* a un algoritmo que busca hallar la opción óptima en cada paso y, de esta manera, alcanzar la solución general óptima para resolver un problema. Esto lo diferencia de algoritmos como los de *backtracking*, los cuales exploran distintas posibilidades y pueden volver al inicio en búsqueda de una mejor solución.

CART Son las siglas de "*Classification and Regression Tree*" y es un algoritmo muy similar al C4.5 pero que soporta clases numéricas, lo cual permite que este algoritmo pueda ser utilizado para resolver problemas de regresión.

Una tarea fundamental en la generación de un árbol es definir un criterio de división. El objetivo del criterio de división es seleccionar el mejor atributo en cada paso y existen diversas técnicas para abordar el problema. Una de ellas es la de "Ganancia de Información", usada por el algoritmo ID3. La técnica de ganancia de información busca seleccionar el atributo que posee mayor variabilidad o representatividad de los datos y se sustenta en el cálculo de la entropía o medida de desorden. La idea de fondo es hallar el atributo que reduzca la entropía esperada. La entropía en el conjunto de datos D se calcula de la siguiente manera:

$$Entropia(D) = -\sum_{i=1}^{q} p_i \log_2(p_i) \tag{2.4}$$

Nótese que p_i corresponde a la probabilidad de que una tupla de D corresponda a la clase C_i. A partir de la entropía, se define la ganancia de información como:

$$Ganancia(A) = Entropia(D) - \sum_{j=1}^{v} \frac{\|D_j\|}{\|D\|} \times Entropia(D_j) \tag{2.5}$$

Aquí el atributo A divide al conjunto de datos en v particiones, siendo v los valores posibles que toma A. D_j es el subconjunto de los datos cuyas tuplas poseen el valor v del atributo A, siendo $\|D_j\|$ su cardinalidad o número de instancias del subconjunto. Al dividir este término por la cardinalidad del conjunto de datos, se obtiene un valor que representa el peso de la partición y que es aplicado sobre la entropía esperada. Este proceso se repite para todos los atributos y, una vez obtenidos los valores de ganancia para cada uno de ellos, se elige aquel que maximiza la ganancia y finalmente el atributo seleccionado será el criterio de separación en el nodo.

El algoritmo C4.5 introdujo una mejora en esta técnica llamada "Razón de Ganancia". La misma busca disminuir uno de los efectos adversos que provoca la técnica de ganancia de información, esta es, que tiende a favorecer a atributos con un mayor número de valores posibles. La razón de ganancia, en primer lugar, reemplaza la fórmula $Entropia(D)$ por la siguiente expresión:

$$EntropiaRG_A(D) = -\sum_{j=1}^{v} \frac{\|D_j\|}{\|D\|} \times \log_2(\frac{\|D_j\|}{\|D\|}) \tag{2.6}$$

A partir de allí, el cálculo de la razón de ganancia hace uso de la ganancia y de la entropía y se formula como:

$$RazonGanancia(A) = \frac{Ganancia(A)}{EntropiaRG_A(D)} \tag{2.7}$$

Finalmente, el atributo de mayor razón de ganancia es seleccionado como criterio de corte y se continúa el calculo con los siguientes subnodos.

Subsección para sgd?, svm?, perceptrones?

Aplicaciones de árboles de decisión?

2.3.2.3. Ensambles

Figura de ensamble

Los ensambles son un conjunto de clasificadores cuyas salidas son combinadas entre sí con el objetivo de realizar mejores predicciones que cualquiera de ellos individualmente. En pocas palabras, el enfoque de ensambles consiste en generar k clasificadores llamados "clasificadores base", desde un mismo algoritmo o no, y entrenarlos con distintos subconjuntos de la colección de entrenamiento original. Dada una tupla nueva, cada clasificador devuelve su propia predicción, llamada "voto", y luego el ensamble combina cada uno de estos votos siguiendo algún método de combinación elegido, de forma tal de producir una predicción final óptima.

La aplicación de ensambles en problemas de clasificación nace de la imposibilidad de generar un único modelo capaz de generalizar lo suficiente como para lograr un rendimiento perfecto. Ante la presencia de datos ruidosos, atípicos o erróneos los clasificadores pueden tender a clasificar mejor para un subconjunto de datos y no tan bien para otros. Este escenario es aprovechado por el enfoque de ensambles ya que su éxito tiene correlación directa con la existencia de "diversidad" en la clasificación, distinguiendo el concepto de diversidad como la existencia de variabilidad entre los modelos, entre hiper-parámetros o entre particiones del conjunto de datos. En definitiva se entiende que, cuanto mayor es esta diversidad, mayor es la probabilidad de aislar los posibles errores particulares de un modelo, y al suceder esto, el error terminará siendo filtrado por el ensamble en la clasificación final. En consecuencia, se espera lograr una disminución del error total de la clasificación así como también una mayor exactitud en la predicción, comparando contra la salida individual de cada clasificador base. Sumado a esto, un enfoque de ensambles abre la posibilidad de distribuir y/o paralelizar el cómputo de la predicción, pudiendo así mejorar los tiempos de ejecución durante el entrenamiento.

En suma, existen distintos tipos de ensamble de acuerdo a su construcción y arquitectura. A continuación se describen tres de ellos: los ensambles de tipo "*bagging*", los de tipo "*boosting*" y los de tipo "*stacked*".

Bagging Esta es una de las primeras técnicas de ensambles conocidas y fue introducida por Breiman[6]. La misma se desarrolla de la siguiente manera: dado un conjunto de entrenamiento D con n tuplas, *bagging* genera un número m de nuevos conjuntos de datos de entrenamiento, cada uno con n tuplas. Para esto se toman tuplas del conjunto original de manera aleatoria y con reemplazo, es decir que puede haber tuplas repetidas y otras que no están incluidas en el nuevo conjunto. Luego a partir de cada conjunto nuevo, se entrena un clasificador M_i. Cada clasificador puede ser del mismo tipo ya que la diversidad está dada por los datos. En la etapa de clasificación, cada modelo M_i genera una predicción que cuenta como un voto. El ensamble cuenta los votos y elige la clase con mayor cantidad de votos, siendo esta la decisión final del ensamble.

Boosting En la técnica de *boosting* se asigna un peso a cada tupla de entrenamiento y se generan un conjunto de clasificadores, cada uno a partir del anterior. A diferencia del método de *bagging*, *boosting* trabaja siempre sobre el mismo conjunto de datos y la variabilidad está dada por los pesos que son asignados. El proceso es el siguiente: para el primer modelo de clasificación, M_i, los pesos son inicializados en un mismo valor para todas las tuplas. Una vez que se entrena este modelo, los pesos son actualizados de tal manera que el siguiente clasificador $M_i + 1$ trate de manera particular a las tuplas mal clasificadas por M_i. De ese modo se busca llegar a una clasificación correcta en las sucesivas iteraciones. Finalmente, el modelo de ensamble combina los

votos de cada clasificador individual. Cabe notar que el peso de cada voto también es ponderado de acuerdo al rendimiento del clasificador base.

Stacking La técnica de *stacking* fue desarrollada por Wolpert[45] y consiste en entrenar un nuevo clasificador de acuerdo a las predicciones realizadas por otros modelos, tomando la salida de estos modelos como entrada, de tal manera de lograr hallar una combinación que produzca una mejor predicción. Este tipo de ensambles puede ser visto como un conjunto de capas. La primera capa consta de un ensamble de clasificadores que aprenden a partir de los datos de entrenamiento. Esta capa no necesariamente usa clasificadores del mismo tipo, mismos hiper-parámetros o particiones de la colección iguales, quedando estos detalles a cargo de quien diseña esta capa. La siguiente capa es el clasificador individual, o meta-clasificador, que se alimenta de las salidas de los clasificadores de la capa inferior y realiza el aprendizaje a partir de las clases producidas por estas salidas y las clases reales.

Una de las tareas a tener en cuenta durante el entrenamiento de un ensamble es la de combinar las salidas de cada modelo en una salida final. La estrategia más común y simple es la de mayoría de voto, la cual normalmente es aplicada por los métodos de *bagging*. No obstante, existen múltiples métodos de combinar los votos, e incluso no siempre un ensamble de tipo *bagging* debe aplicar esta estrategia. Por ejemplo, algunos clasificadores pueden decidir producir una salida solo en el caso de que más de la mitad de ellos coincidan, o incluso ser más restrictivos y obligar a que la coincidencia sea total. El enfoque de *boosting* por su parte, pondera al voto de acuerdo a los pesos que calcula, dando predominio a determinadas instancias. También se suele dar un mayor peso a determinados clasificadores por sobre otros. Este tipo de métodos se los denomina "mayoría de voto ponderada" y pueden llevar a un rendimiento superior si es aplicada en el escenario adecuado.

Aplicaciones de ensambles en la literatura

2.3.3. Evaluación

Llevar a cabo evaluaciones de rendimiento sobre los modelos es un aspecto importante del aprendizaje automático ya que nos permite conocer en qué medida un algoritmo es superior a otro para resolver una tarea. Particularmente, la tarea de clasificación es un desafío que se presenta en un contexto cambiante y evolutivo, donde nuevas herramientas surgen y se actualizan constantemente. Incluso la composición y estructura de los modelos de clasificación varía según la familia de algoritmos aplicadas, y es esperable que los conceptos extraídos de un modelo de tipo árbol tengan particularidades que lo diferencien de modelos de redes neuronales o modelos probabilísticos. Y del mismo modo, es esperable que alguno de estos modelos tenga un mejor rendimiento que otro en un determinado escenario, o incluso que sea mejor que un modelo generado por el mismo algoritmo pero con distintos hiper-parámetros. La tarea de evaluación es la que permite detectar estas particularidades y sacar provecho de los algoritmos para obtener aún mejores modelos.

A su vez, es importante estudiar las métricas de evaluación existentes y llevar adelante estrategias que nos permitan obtener medidas de evaluación confiables y que no hayan sido sesgadas por los datos que se usaron durante el entrenamiento. Y del mismo modo, entender los factores que provocaron un valor de métrica puede ser el paso inicial para hallar mejoras al modelo que optimizan su capacidad de predicción en el futuro.

Por consiguiente, a continuación se estudian algunas de las estrategias llevadas a cabo durante la evaluación para evitar sesgos, así como también las métricas que se calculan durante este proceso.

2.3.3.1. Métricas

Las métricas de evaluación se usan para conocer la habilidad predictiva de un modelo de clasificación. Se van a describir aquí las métricas más conocidas del campo y que proveen, a su vez, un primer vistazo de lo que serán las métricas usadas para evaluar algoritmos para datos multi-etiquetados en las futuras secciones del escrito.

Antes de comenzar es importante aclarar algunos términos que serán usados para definir las métricas. Se entiende como "ejemplos positivos" a aquellas instancias cuya etiqueta pertenece a la clase de interés en el problema de estudio y "ejemplos negativos" como aquellas que no pertenecen a dicha clase. Al mismo tiempo, se derivan cuatro conceptos que estarán presentes en las fórmulas para calcular las métricas, los cuales son:

Verdaderos Positivos (VP) Son los ejemplos positivos que fueron correctamente clasificados como positivos.

Verdaderos Negativos (VN) Son los ejemplos negativos que fueron correctamente clasificados como negativos.

Falsos Positivos (FP) Son los ejemplos negativos que fueron incorrectamente clasificados como positivos.

Falsos Negativos (FN) Son los ejemplos positivos que fueron incorrectamente clasificados como negativos.

Una vez obtenidos estos conceptos se pueden calcular una serie de métricas a partir de ellos. Estas métricas son:

Exactitud La exactitud o *accuracy* es la proporción de ejemplos correctamente clasificados sobre el número total de instancias y a mayor el valor de exactitud mejor es el rendimiento del clasificador. Se define como:

$$exactitud = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.8}$$

Tasa de Error A la inversa de la exactitud, la tasa de error es la proporción de ejemplos incorrectamente clasificados sobre el número total de instancias y a menor el valor de la tasa de error mejor es el rendimiento del clasificador. Se define como:

$$tasaError = \frac{FP + FN}{VP + VN + FP + FN} \tag{2.9}$$

Precisión La precisión es la proporción de ejemplos que fueron clasificados como positivos y que efectivamente lo son. Mayor es el valor de precisión mejor es el rendimiento del clasificador. Se dice que es una medida de exactitud y se define como:

$$precision = \frac{VP}{VP + FP} \tag{2.10}$$

Exhaustividad La exhaustividad o *recall* es la proporción de ejemplos positivos que fueron clasificados como positivos. Mayor es el valor de exhaustividad mejor es el rendimiento del clasificador. Se dice que es una medida de completitud y se define como:

$$exhaustividad = \frac{VP}{VP + FN} \tag{2.11}$$

Medida-F1 La medida-F1 o *f1-score* es una medida que integra las métricas de precisión y exhaustividad tomando la media harmónica entre ambas. Mayor el valor de esta medida, mejor es el rendimiento del clasificador. Se define como:

$$medidaF1 = \frac{2 \times precision \times exhaustividad}{precision + exhaustividad} \tag{2.12}$$

Por otro lado, existen métricas para evaluar la eficiencia del clasificador en términos de velocidad y consumo de memoria. Estas métricas cobran especial importancia en ambientes de flujos continuos de datos, donde el volumen de datos es grande, la velocidad en la que arriban es alta y los recursos escasean. A continuación se añade una breve descripción de ambas.

Velocidad La velocidad se refiere al costo computacional de generar el modelo y realizar predicciones, en general se deja fuera del cálculo los tiempos invertidos en cargar la colección en memoria, realizar tareas de normalización y pre-procesamiento sobre los datos y otras etapas de la clasificación. A menor el tiempo de ejecución, mejor es el rendimiento del algoritmo.

Consumo de Memoria El consumo de memoria es un indicador de los requerimientos de memoria aproximados para almacenar el modelo, así como también un indicador de resguardo para estimar el posible consumo de memoria durante la ejecución y asegurarse que el algoritmo se mantiene en actividad. A menor el consumo de memoria, mayor es la eficiencia del modelo.

2.3.3.2. Estrategias

Una vez que se define la métrica o el conjunto de métricas apropiadas para medir el rendimiento de los clasificadores, el siguiente desafío es seguir un procedimiento de pruebas capaz de lograr resultados de evaluación que puedan ser generalizables a conjunto de datos aún no observados. Las técnicas de "*Holdout*" y "Validación Cruzada" son dos de las técnicas más populares para evaluar la habilidad predictiva de los clasificadores y se describen a continuación.

Holdout En este método un conjunto de instancias es separado de la colección y se reserva para evaluar el rendimiento del clasificador. Este subconjunto es distinto del conjunto de datos de entrenamiento usado para generar el modelo y es llamado "conjunto de pruebas o *testing*". Una vez entrenado, el clasificador recibe las instancias del conjunto de pruebas pero sin incluir las etiquetas. La salida del clasificador son las etiquetas de cada instancia. Finalmente, las etiquetas producidas durante la predicción y las etiquetas reales de cada instancia se combinan y se calculan las métricas de evaluación. Esta técnica se basa en la idea de que, separando los datos que se usan durante el entrenamiento de aquellos usados durante la predicción, se logra una independencia en los datos que derivará en un mayor grado de generalización en el modelo.

Por contrapartida, el enfoque de *Holdout* tiene la limitación de que para lograr generalizar requiere de un número de instancias considerable. Esta idea proviene del hecho de que muy pocos datos en el conjunto de entrenamiento puede derivar en predicciones pobres, pero por el contrario, muy pocos datos en el conjunto de pruebas tampoco es recomendable, ya que podría resultar en medidas de rendimiento poco fiables. A esto se suma una de las dificultades más comunes en el área de aprendizaje automático: la falta de disponibilidad de colecciones grandes de datos del mundo real.

citar fuente

Por estas razones, ha tomado peso el uso de técnicas de muestreos de datos para reutilizar instancias de entrenamiento y de prueba. Una de ellas es la de validación cruzada.

Validación Cruzada de K iteraciones La técnica de validación cruzada, más conocida por el inglés *k-fold cross-validation*, consiste en particionar el conjunto de datos en k subconjuntos de manera aleatoria y en la forma $\{d_1, d_2, \ldots, d_k\}$, siendo cada uno de los subconjuntos mutuamente excluyentes entre sí y de igual o similar tamaño. El proceso itera sobre cada uno de los subconjuntos para generar k modelos. En la primera iteración i, se separa el subconjunto d_i y se entrena el modelo con los restantes subconjuntos para luego medir el rendimiento con él. En la segunda iteración, se repite este procedimiento pero usando como pruebas al subconjunto d_{i+1}, y así en cada iteración. Así pues, cada subconjunto es usado una vez para probar el modelo. Finalmente, se toman los k modelos generados y se promedian las métricas.

De esta técnica derivan otras similares como "*leave-one-out*" en donde cada subconjunto es conformado por $n-1$ instancias, siendo n el tamaño de la colección, dejando un elemento fuera del subconjunto y que será usado para validar el modelo. El proceso se repite n veces y se combinan los modelos tal como en el método de validación cruzada.

Otra técnica similar es la de "Validación Cruzada Estratificada", la cual consiste en generar subconjuntos de entrenamiento y pruebas que respeten la representación de clases existentes en la colección inicial. Por ejemplo, si una de las clases del problema aparece en el 25 % de las instancias, este método asegura que al generar las particiones esa clase continuará siendo representada por el 25 % de las tuplas. Esta técnica puede ser útil para problemas donde existe una diversidad que es necesario reflejar en las particiones para obtener medidas precisas.

En general, los investigadores recomiendan usar validación cruzada con $k = 10$ ya que se suelen conseguir estimaciones menos sesgadas sin incurrir en costos computacionales demasiado altos.

2.4. Clasificación Multi-etiquetas

A diferencia del aprendizaje automático tradicional, que usa datos de etiqueta única para representar objetos del mundo real, cada instancia en el aprendizaje multi-etiquetas representa un único objeto pero puede contener más de una etiqueta. Por consiguiente, la tarea de clasificación consiste en hallar una función que logre asignar a cada objeto, nuevo y desconocido, el conjunto de etiquetas que lo caracteriza.

En este apartado se da una definición formal, se detallan las características de un conjunto de datos multi-etiquetados y se describen algunos métodos tradicionales de clasificación multi-etiquetas junto con sus ventajas, desventajas, aplicaciones y motivaciones.

2.4.1. Definición

Asumiendo que $X = \mathbb{R}^d$ denota el espacio de instancias d dimensional, y que $Y = \{y_1, y_2, \ldots, y_q\}$ denota el espacio de etiquetas con q etiquetas posibles, la tarea de clasificación multi-etiquetas consiste en entrenar un conjunto $D = \{(x_i, Y_i) \mid 1 \leq i \leq m\}$ para hallar una función h tal que $h : X \to 2^y$. A su vez, X_i es un vector de atributos d dimensional definido como $(x_{i1}, x_{i2}, \ldots, x_{id})$. Y_i, por su parte, es el conjunto de etiquetas asociadas a la instancia X_i. Luego, para cada instancia desconocida $x \in X$ el clasificador h predice $h(x) \subseteq Y$ que representa el conjunto de etiquetas hallado para x.

A su vez, se definen un conjunto de métricas que describen el grado de multi-etiquetado que tiene un conjunto de datos dado, o en otras palabras, hasta qué punto cada ejemplo posee más de una etiqueta. Algunas de ellas son:

Cardinalidad de etiquetas: Es el promedio de etiquetas por instancia del conjunto de datos. Se define como:

$$CardE(D) = \frac{1}{m}\sum_{i=1}^{m} \|Y_i\| \tag{2.13}$$

Por lo tanto, a mayor el valor de cardinalidad, mayor es el número de etiquetas de una instancia. Por ejemplo, si $CardE = 1$, entonces la mayoría de ejemplos tiene una única etiqueta y, por consiguiente, se puede decir que la colección tiene un grado bajo de multi-etiquetado.

Densidad de etiquetas: Es la cardinalidad de etiquetas normalizada al número total de etiquetas de D y se define como:

$$DenE(D) = \frac{CardE(D)}{\|Y\|} \tag{2.14}$$

Así pues, un valor alto de densidad significaría que cada instancia puede ser una buena representación de las etiquetas del conjunto. De la misma manera, un valor bajo suele implicar dispersión, esto es, que la mayoría de las instancias tienen un subconjunto acotado de las etiquetas.

Diversidad de etiquetas: Es el número de conjuntos de etiquetas unívocos que aparecen en instancias de D. Se define como:

$$DivE(D) = \|\{Y \mid \exists x : (x, Y) \in D\}\| \tag{2.15}$$

Aquí la interpretación es que, a mayor el valor de diversidad, menor es la constancia con la que las etiquetas aparecen en las instancias. De manera similar a la cardinalidad, el valor de diversidad también puede normalizarse por el número de instancias del conjunto de datos:

$$DivEProm(D) = \frac{DivE(D)}{\|D\|} \tag{2.16}$$

2.4.2. Algoritmos

Como se había anticipado en la sección 1.2.1, la tarea de aprendizaje sobre datos multi-etiquetados puede ser encarada siguiendo dos grandes enfoques, llamados “Transformación del Problema” y “Adaptación del Algoritmo”. A continuación se describen ambos enfoques y algunos de sus algoritmos más representativos.

2.4.2.1. Transformación del Problema

Esta categoría engloba al conjunto de algoritmos que abordan el problema de clasificación multi-etiquetas transformándolo en múltiples problemas de clasificación de única etiqueta, lo cual permite aplicar algoritmos de clasificación convencionales. Tres de estos métodos son particularmente relevantes para este trabajo: “Binary Relevance (BR)”, “Classifier Chains (CC)” y “Label Powerset (LP)”.

Binary Relevance (BR) El algoritmo de Relevancia Binaria, conocido como *Binary Relevance* en la literatura, es un enfoque que consiste en descomponer la tarea de clasificación MLL en $\|q\|$ clasificadores binarios, independientes y de etiqueta única. A partir de esta transformación se puede seleccionar cualquier algoritmo de clasificación como clasificador base del problema (ver los algoritmos presentados en la sección 2.3.2). Cada clasificador binario g_j es entrenado con todas las instancias de la colección pero incluyendo solo la etiqueta j, la cual se activa o desactiva de acuerdo a si es relevante a la instancia. Luego la predicción de una instancia desconocida se realiza combinando las salidas de cada clasificador individual, esto es:

$$Y = y_j \mid g_j(x) > 0, 1 \leq j \leq q \tag{2.17}$$

Llegado el caso en que ninguno de los clasificadores retornen etiquetas activas, el conjunto Y será vacío.

Este enfoque se dice que es de primer orden (ver sección 1.2.1) y no tiene en cuenta la correlación o interdependencias entre etiquetas. Este es uno de los principales inconvenientes de este algoritmo ya que, en este tipo de problemas de MLL, es usual hallar que determinadas etiquetas se activan en conjunto con mayor probabilidad. Pese a ello, BR es un enfoque muy utilizado [47] ya que es simple de implementar, intuitivo y computacionalmente poco costoso en comparación con algoritmos que sí tienen en cuenta la relación entre etiquetas.

Classifier Chains (CC) Las cadenas de clasificadores o *Classifier Chains* [35] es una técnica que convierte el problema de MLL en una "cadena" de problemas de clasificación binaria, tal que el siguiente clasificador de la cadena posee las predicciones de los anteriores. En principio, la división del conjunto de datos es similar a la que se hace en el enfoque anterior, designando un clasificador por cada etiqueta. Durante el entrenamiento, el clasificador inicial, seleccionado aleatoriamente, usa de entrada los atributos originales, tal como el clasificador BR. Luego la salida de este clasificador es añadida al espacio de atributos como un atributo más de cada instancia, para que posteriormente, estos atributos sean la entrada del siguiente clasificador, el cual también es seleccionado al azar. Este proceso es repetido hasta completar todos los clasificadores. Como se puede observar, lo que se produce es un "encadenamiento" de clasificadores, el cual no es accidental y tiene como fin conservar la dependencia entre etiquetas: en el entrenamiento se van acumulando las salidas de los clasificadores anteriores de tal manera que el siguiente clasificador absorbe la correlación entre las etiquetas de los anteriores clasificadores.

Cabe notar que en esta técnica cobra especial importancia el ordenamiento de los clasificadores ya que este orden tiene un impacto directo sobre el resultado de la predicción. En otras palabras, si el ordenamiento de clasificadores se modifica, el modelo final otorgará resultados diferentes. Para salvar esta dificultad se han propuesto modelos como el de Ensamble de Classifier Chains (ECC). El mismo genera un conjunto de modelos de CC con distintos ordenamientos y entrenados con diversos subconjuntos de datos, generados con reemplazo o no. Durante la predicción, cada cadena produce un conjunto de etiquetas, que son los votos, y la salida final será computada por un algoritmo que combine cada salida individual.

Label Powerset (LP) El conjunto de potencias de etiquetas o *Label Powerset* [42] es una técnica que se encarga de transformar el problema de MLL en uno de clasificación multi-clase y así poder abordarlo con algoritmos de este tipo. La clasificación multi-clase es un enfoque usado para tratar con ejemplos en donde la etiqueta es única pero cuenta con más

de dos clases. Un ejemplo de este tipo de problemas es el de análisis de sentimiento de texto, en donde las clases pueden ser "positivo", "negativo" y "neutral".

En LP cada etiqueta indica el subconjunto de etiquetas de la instancia. Esto es beneficioso en cuanto a que se logra preservar la dependencia entre etiquetas. Sin embargo, el modelo tiene algunas dificultades. En primer lugar, el espacio de clases posibles es exponencial y su cantidad de clases puede llegar a ser de $2^{\|q\|}$ como máximo. A su vez, pueden llegar a arribar ejemplos con una combinación de etiquetas que el modelo no recibió durante el entrenamiento, por lo cual no logra generalizar lo suficiente y se lo considera un modelo incompleto. A fin de sobrepasar estas complicaciones se desarrolló la técnica de "Conjuntos Podados" o "Pruned Set (PS)". La misma consiste en preservar para la clasificación aquellos subconjuntos de etiquetas que son más frecuentes en la colección, y eliminar los demás. Con esto se logra disminuir considerablemente el espacio de clases y disminuye la complejidad computacional, tanto durante el entrenamiento como durante la predicción.

2.4.2.2. Adaptación del Algoritmo

Además del enfoque de transformación del problema, algunos autores abordan la clasificación multi-etiquetas a partir de la adaptación de algoritmos clásicos y bien conocidos a este tipo de escenarios. La categoría engloba al conjunto de algoritmos que acometen el problema de MLL mediante la modificación de algoritmos de etiqueta única para que sean capaces de manejar la nueva naturaleza de los datos en estas tareas. Las modificaciones que se introducen pueden variar en complejidad según el algoritmo tratado y las características de la colección. Se han adaptado una diversa cantidad de algoritmos incluyendo aquellos basados en redes neuronales, árboles, métodos probabilísticos, entre otros [18]. Por mencionar algunos ejemplos, Gargiulo, Silvestri y Ciampi usan redes neuronales profundas para clasificar documentos de texto libre y comparan su funcionamiento variando el número de etiquetas y aprovechando su estructura jerárquica [15]. Asimismo, Tanaka y col. agregan al algoritmo de BR la capacidad de capturar relaciones entre etiquetas a través del uso de árboles de decisión y lo aplican en el área de la genómica [40].

En lo que confiere a ambientes de flujos continuos de datos una de las técnicas más populares en la literatura es la de Árbol de Hoeffding o *Hoeffding Tree (HT)* [9], que a diferencia de los algoritmos convencionales de árboles de decisión, aborda los datos de manera incremental. Así pues, en lugar de realizar decisiones de corte de acuerdo a los datos previamente almacenados, el algoritmo de HT espera a tener una cantidad suficiente de instancias para realizar el corte, con un cierto grado de confianza. Esto significa que ya no es necesario guardar todos los datos de la colección y que mantener una serie de estadísticas es suficiente para realizar la clasificación. Otra de las propiedades a destacar de este método es que en teoría se puede entrenar un árbol cuyo rendimiento se aproxime al generado en un ambiente de *batch*, con la suficiente cantidad de datos [3].

A partir de esto, y buscando sacar provecho de las ventajas mencionadas, Read y col. decidieron adaptar este algoritmo a problemas de multi-etiquetas y desarrollaron el algoritmo llamado Árbol de Hoeffding Multi-etiquetado o *Multi-label Hoeffding Tree (MLHT)* [36]. Esto lo consiguen a partir de rediseñar la fórmula de ganancia de información (ver fórmula 2.7) de tal manera de reflejar en el cálculo de la entropía el impacto de todas las clases a las cuales el ejemplo no pertenece. Desde que fue introducido en el año 2012, MLHT ha sido usado como modelo de comparación en reiteradas oportunidades [39] y es uno de los métodos más populares para atacar problemas de MLL en ambientes de flujo continuo de datos.

Otro algoritmo muy popular para abordar este tipo de problemas y que también se

basa en árboles de decisión incrementales es el llamado iSoup-Tree (siglas del inglés *Incremental Structured Output Prediction Tree*) [28]. El mismo fue desarrollado inicialmente para hacer regresión de múltiples objetivos y luego fue adaptado a tareas de MLL. Una de las novedades que introduce es el uso de un perceptrón adaptativo en las hojas del árbol, lo cual le da la versatilidad de poder trabajar tanto con problemas de clasificación como de regresión.

Explicación de enfoques de ensambles? (ebr, ecc, elp)

2.4.3. Evaluación

Como se describió en la sección 2.3.3.1, en escenarios de clasificación tradicionales el poder de generalización de un modelo es evaluado a partir de métricas tales como la exactitud o la exhaustividad las cuales trabajan con una única etiqueta. Una predicción multi-etiquetada, por el contrario, carga con la complejidad de que cada ejemplo puede ser asociado a más de una o dos etiquetas. Esto implica que ya no se puede trabajar con la presunción de que una clasificación solo puede ser correcta o incorrecta. De lo contrario, se incursionaría en una evaluación excesivamente rigurosa. Por este motivo, en el campo de MLL se han diseñado una serie de métricas nuevas y específicas para abordar este tipo de problemas, y se clasifican en dos grupos: existen las métricas "basadas en ejemplos", que son computadas por cada instancia y luego promediadas; y las métricas "basadas en etiquetas", las cuales son calculadas para cada etiqueta por separado y devuelven el promedio micro o el promedio macro.

A continuación se describe en detalle ambas categorías, incluyendo métricas presentadas por cada enfoque, junto con su definición formal y una interpretación de la evaluación que proveen.

2.4.3.1. Métricas basadas en Ejemplos

En esta categoría se hallan las métricas que evalúan la calidad de la clasificación ejemplo por ejemplo y determinan qué tan buena es la clasificación sobre los distintos ejemplos. Tienen en común que comparan las etiquetas de la predicción contra las etiquetas reales para cada instancia y luego combinan la salida aplicando la media sobre el total de instancias. Entre ellas se encuentran las siguientes:

Hamming Loss Esta es una de las métricas más populares en la literatura y se encarga de evaluar la proporción de pares instancia-etiqueta incorrectamente clasificados, es decir, toma en cuenta para el conteo tanto si se marcó como irrelevante una etiqueta relevante o si, por el contrario, se activó una etiqueta que no era relevante. Se define como:

$$HLoss = \frac{1}{m} \sum_{i=1}^{m} \|h(x_i) \triangle Y_i\| \tag{2.18}$$

Aquí el símbolo "$\triangle$" es la diferencia simétrica entre conjuntos. Tener en cuenta también que cuanto menor es el valor de *hamming loss* mejor es el rendimiento del clasificador.

De manera complementaria, algunos autores de la literatura han decidido usar la métrica "*hamming score*" que es un derivado del *hamming loss* y se calcula como:

$$HScore = 1 - HLoss \tag{2.19}$$

Exactitud del Subconjunto Se la conoce también como *exact-match* en la literatura y es la métrica equivalente a la exactitud en clasificaciones de etiqueta única (ver "Exactitud" en sección 2.3.3.1). El cómputo toma en cuenta la proporción de instancias que fueron clasificadas de manera exacta, es decir, que todas sus etiquetas fueron correctamente clasificadas o, en otras palabras, que el subconjunto de etiquetas de la predicción es idéntico al subconjunto de etiquetas reales. En términos formales, se calcula como:

$$exactitudSubconjunto = \frac{1}{m}\sum_{i=1}^{m} \|h(x_i) = Y_i\| \tag{2.20}$$

Es considerada una métrica excesivamente rigurosa, especialmente para colecciones donde el número de etiquetas es alto.

Exactitud basada en ejemplos La exactitud basada en ejemplos también es conocida como "Índice de Jaccard" y consiste en tomar la proporción de etiquetas activas y correctamente clasificadas con respecto al total de etiquetas activas.

$$exactitudEj = \frac{1}{m}\sum_{i=1}^{m} \frac{\|Y_i \cap h(x_i)\|}{\|Y_i \cup h_(x_i)\|} \tag{2.21}$$

Precisión, Exhaustividad y Medida-F1 (basadas en ejemplos) Las métricas de precisión, exhaustividad y medida-F1 presentadas en la sección 2.3.3.1 también tienen sus equivalentes para datos multi-etiquetados y se definen de la siguiente manera:

$$precisionEj = \frac{1}{m}\sum_{i=1}^{m} \frac{\|Y_i \cap h(x_i)\|}{\|h_(x_i)\|} \tag{2.22}$$

$$exhaustividadEj = \frac{1}{m}\sum_{i=1}^{m} \frac{\|Y_i \cap h(x_i)\|}{\|Y_i\|} \tag{2.23}$$

$$medidaF1Ej = \frac{2 \times precisionEj \times exhaustividadEj}{precisionEj + exhaustividadEj} \tag{2.24}$$

2.4.3.2. Métricas basadas en Etiquetas

Hasta aquí, todas las métricas mencionadas son calculadas individualmente para cada instancia y luego promediadas por el total de instancias. No obstante, también existen otros tipos de métricas, las basadas en etiquetas, que se encargan de evaluar la calidad de la clasificación etiqueta por etiqueta y determinan qué tan buena es la clasificación sobre las distintas etiquetas. En síntesis, comparan las etiquetas de la predicción contra las etiquetas reales, cada una por separado y luego combinan el rendimiento entre todas las etiquetas. Dicha combinación se realiza aplicando el promedio micro o el promedio macro. Más adelante se explicará en qué consiste cada una de ellas.

De manera similar a lo que sucedía con las métricas de la sección 2.3.3.1, donde para definirlas fue necesario introducir el concepto de verdaderos positivos, verdaderos negativos, falsos negativos y falsos positivos, las métricas basadas en etiquetas también se fundamentan en esos conceptos pero su fórmula varia para adaptarse al escenario de MLL.

$$VP_j = \|\{x_i \mid y_j \in Y_i \wedge y_j \in h(x_i), 1 \leq i \leq m\}\| \tag{2.25}$$

$$FP_j = \|\{x_i \mid y_j \notin Y_i \wedge y_j \in h(x_i), 1 \leq i \leq m\}\| \tag{2.26}$$

$$VN_j = \|\{x_i \mid y_j \notin Y_i \wedge y_j \notin h(x_i), 1 \leq i \leq m\}\| \tag{2.27}$$

$$FN_j = \|\{x_i \mid y_j \in Y_i \wedge y_j \notin h(x_i), 1 \leq i \leq m\}\| \tag{2.28}$$

Tal como sucede con el aprendizaje tradicional, bajo esta definición las evaluaciones basadas en etiquetas logran satisfacer la condición:

$$VP + FP + VN + FN = m \tag{2.29}$$

A partir de estos cuatro conceptos se pueden derivar cualquiera de las métricas definidas en la sección 2.3.3.1. O dicho en términos formales, sea $B(VP_j, FP_j, VN_j, FN_j)$ una función cuyo dominio es $B \in \{exactitud, precision, exhaustividad, medidaF1\}$, las métricas pueden ser obtenidas siguiendo dos estrategias:

Promedio Macro

$$B_{macro}(h) = \frac{1}{q}\sum_{j=1}^{q} B(VP_j, FP_j, VN_j, FN_j) \tag{2.30}$$

Promedio Micro

$$B_{micro}(h) = B(\sum_{j=1}^{q} VP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} VN_j, \sum_{j=1}^{q} FN_j) \tag{2.31}$$

El enfoque de promedio macro tiene una semejanza al enfoque basado en ejemplos en cuanto a que ambas estrategias promedian por el mismo valor a todas sus etiquetas/ejemplos, siendo en este caso el número total de etiquetas. Esto se traduce en que esta categoría de métricas le asigna a cada etiqueta el mismo peso, sin importar la frecuencia de la misma.

Por el contrario, el enfoque de promedio micro sí provoca una ponderación en las etiquetas y la contribución de cada etiqueta al valor final no será el mismo. Esto sucede porque la métrica se computa solo una vez, habiendo ya calculado la cantidad de etiquetas para cada uno de los cuatro conceptos, lo que hace que aquellas etiquetas que se activan en escasas ocasiones vayan a tener una menor injerencia y por tanto serán sobrepasadas por aquellas etiquetas más frecuentes.

2.5. Clasificación de Flujos Continuos de Datos

En esta sección se profundiza sobre algunos conceptos referidos a flujos continuos de datos y su tratamiento, de forma tal de llevar a cabo tareas de clasificación de manera exitosa. Como se describió en la sección 1.2.2, un flujo continuo cuenta con características específicas que obligan a los algoritmos de aprendizaje a adaptarse a nuevos requerimientos. A continuación se define formalmente el concepto de flujo, se extiende el análisis sobre la tarea de evaluación para dar cuenta de su aplicación en este escenario, se describe el concepto de dato sintético y se presentan algunos algoritmos y técnicas para generar datos sintéticos útiles para clasificaciones de etiqueta única y de multi-etiquetas.

2.5.1. Definición

Un flujo continuo de datos o *Data Stream* es un conjunto de datos ordenados, que arriban en el tiempo y son potencialmente infinitos. Un *stream* se define como:

$$S = \{s_0, s_1, \ldots, s_t, \ldots, s_N\} \tag{2.32}$$

Donde s_t es la instancia presente en el tiempo t y s_N es el último dato avistado en el *stream* pero que no necesariamente representa el final del flujo. Cada instancia d_t posee un conjunto de atributos y etiquetas y se puede definir tal como se hizo en la sección 2.3.1 de tratarse de datos uni-etiquetados o como en la sección 2.4.1 de tratarse de datos multi-etiquetados.

El objetivo de la tarea de clasificación en este escenario es el mismo que para escenarios de *batch*, esto es, hallar una función capaz de enlazar instancias nuevas con sus etiquetas correspondientes, con la salvedad de que se debe lograr bajo restricciones de tiempo y espacio de almacenamiento, además de otras vicisitudes presentadas por las características mismas de un flujo continuo de datos (ver sección 1.2.2.1). Del mismo, deben ser capaces de lidiar con cambios en la distribución de los datos, deben estar listos para realizar predicciones en cualquier momento y otros requisitos ya mencionados en la sección 1.2.2.2. Estas características implican que obtener soluciones exactas es poco probable y es necesario aplicar técnicas y metodologías especiales de evaluación sobre los modelos, para reducir el error posible.

sección para explicar concept drift?

2.5.2. Evaluación

Como se describió en la sección 2.3.3, la tarea de evaluación para aprendizaje en ambientes de *batch* consiste en dividir al conjunto de datos en uno o más subconjuntos de entrenamiento y pruebas. El ambiente de *stream*, por su parte, presenta un desafío extra y es que no permite llevar a cabo esta división debido a que no se cuenta con los datos previamente almacenados. Además se debe tener en consideración la naturaleza incremental y evolutiva de los datos: a medida que pasa el tiempo pueden surgir nuevas etiquetas, ciertos atributos pueden dejar de tener peso en la predicción o incluso algunas reglas de decisión en el modelo pueden llegar a perder relevancia. Por lo tanto, se han presentado dos enfoques nuevos, similares a las estrategias definidas en la sección 2.3.3.2 pero acondicionados a los escenarios de flujos continuos de datos.

Evaluación holdout La evaluación por retención o *holdout* es un método derivado de la técnica de validación cruzada pero adaptado a ambientes de flujos continuos. Consiste en usar una parte del *stream* como conjunto de entrenamiento y, periódicamente, extraer una serie de conjuntos de prueba, llamados conjuntos de *holdout*, que son usados para computar las métricas de evaluación y que, por tanto, no deben haber sido observados por el modelo previamente. *Holdout* es un método que requiere contar con flujos continuos lo suficientemente grandes para que la evaluación sea precisa, lo cual no siempre es posible y es uno de los impedimentos que hacen que esta estrategia tenga menor trascendencia que otras.

Evaluación Prequential La técnica de evaluación *Prequential* o *test-then-train* consiste en realizar la evaluación de cada instancia primero y luego usarla para actualizar el modelo. En consecuencia, ya no hay una división en subconjuntos de datos independientes sino que todas las instancias son usadas para evaluar y luego clasificar, en

un mismo instante de tiempo. A diferencia del enfoque de *holdout*, no es necesario que el conjunto de datos sea grande y la evaluación *prequential* permite alimentar al modelo con todos los datos de la colección, lo que se traduce en un aprovechamiento máximo del *stream*. Por estas razones, el enfoque *prequential* tiende a ser el más usado en este campo de estudio.

En cuanto a las métricas de rendimiento utilizadas, en el ámbito de clasificaciones de flujos continuos se usan las mismas métricas de clasificaciones por *batch* con la salvedad que el cómputo se hace de manera incremental. En otras palabras, a diferencia de los métodos clásicos de validación cruzada donde la evaluación se hace en una única pasada, una vez generado el modelo final, aquí se debe calcular la métrica en cada nueva instancia y los resultados se van acumulando.

En definitiva, las métricas son las mismas que fueron presentadas en las secciones 2.3.3.1 para datos de etiqueta única y 2.4.3 para datos multi-etiquetados.

2.5.3. Datos sintéticos

Generar datos sintéticos es una práctica frecuente en la literatura para simular ambientes de flujo continuos de datos, el principal motivo es la falta de colecciones de *streams* del mundo real que sean lo suficientemente grandes y que al mismo tiempo cumplan con todos los requisitos necesarios para evaluar algoritmos en este escenario [21]. Pese a esta restricción, se han hallado ventajas comparativas en la aplicación de flujos sintéticos en el análisis y evaluación de algoritmos, entre ellas se encuentran las siguientes:

- Tiene un costo de almacenamiento relativamente menor.
- Cumplen con el requisito de ser teóricamente infinitos.
- Su generación es automatizable lo cual facilita su reproducibilidad entre experimentos.
- Es posible introducir cambios de concepto artificiales para realizar un análisis incisivo de los algoritmos de clasificación bajo escenarios dinámicos y cambiantes.
- Ayudan en el ámbito académico y científico a realizar estudios y experimentos más abarcativos.

En la actualidad existen varios generadores de flujos sintéticos que logran cumplir con los requisitos necesarios, aquí se describen dos de ellos: “Generador de Árbol Aleatorio (RTG)” y “Generador de Función Radial Base (RBF)”.

Generador de Árbol Aleatorio (RTG) El generador se basa en la técnica presentada por Domingos y Hulten [9] y consiste en producir un *stream* partiendo de un árbol construido aleatoriamente. A fines de generar el árbol, el método selecciona un atributo al azar para realizar el corte y posteriormente le asigna una clase aleatoria a cada hoja. A partir de este árbol se van a generar las instancias sintéticas. Primero se asignan valores aleatorios a los atributos, siguiendo una distribución uniforme, y luego con esos valores se atraviesa el árbol para hallar las clases de la etiqueta. Teniendo en cuenta que las instancias son generadas y clasificadas según un modelo con estructura de árbol, en teoría este método favorece a algoritmos del tipo de árboles de decisión.

Generador de Función Radial Base (RBF) El generador produce un *stream* de función de base radial aleatoria. El método actúa de la siguiente manera: se generan un número fijo de centroides y cada centro tiene una posición aleatoria, una única desviación estándar, una clase y un peso. Para generar una instancia se selecciona un centro al azar, teniendo en cuenta el peso asociado, de tal manera de favorecer a los centros con mayor peso. La clase del ejemplo es determinada por el centroide elegido. El siguiente paso es seleccionar una dirección tal que aleje los valores de atributo del punto central. La dirección es obtenida aleatoriamente, siguiendo una distribución gaussiana cuya desviación estándar es determinada por el centroide elegido. El resultado, en términos geométricos, es una hiperesfera en donde cada ejemplo rodea un punto central con densidades variables [21]. Este método surge de la necesidad de generar conceptos que no favorezcan a modelos de tipo árbol, tal como sí lo hacía la técnica RTG.

En lo que refiere a datos multi-etiquetados, las herramientas existentes son más reducidas. Read, Pfahringer y Holmes han presentado un marco general de trabajo para la generación de flujos continuos multi-etiquetados [32] con el objetivo de generar datos realistas o que se aproximen a instancias del mundo real. El procedimiento consiste en tomar la salida obtenida por los algoritmos generadores de datos de etiqueta única y transformarla en datos multi-etiquetados, y esto lo hacen obedeciendo a fenómenos, comportamientos o cualidades intrínsecas a las colecciones multi-etiquetadas del mundo real. La idea es que si estos fenómenos capturados en datos reales se cumplen en grado similar para los datos sintéticos entonces se ha logrado generar una colección de datos de valor. Los fenómenos presentados son los siguientes:

Sesgo de etiquetas Es el fenómeno por el cual existen etiquetas que se presentan en los datos más frecuentemente que otras. A diferencia de colecciones de etiqueta única, puede haber más de una etiqueta por instancia lo que lleva a que este fenómeno se magnifique en colecciones multi-etiquetadas. A su vez, en colecciones de texto es muy común encontrar unas pocas etiquetas que predominan y otras que pertenecen a subconjuntos de instancias muy específicas. Por ejemplo, una etiqueta como "`Ficción`" es muy probable que sea relevante en colecciones de cuentos literarios e incluso que figure junto con otras etiquetas, por ejemplo {`Ficción, Drama`}. No sucede lo mismo con etiquetas tales como "`Elefantes`", que probablemente aparezcan de manera aislada para este conjunto de datos. Por lo tanto, si se desea generar una colección sintética de texto es posible que se quiera ajustar el valor del parámetro de sesgo de etiquetas para que sea mayor al de otro tipo de colecciones.

Distribución de etiquetas La distribución de etiquetas tiene que ver con la forma en que la cardinalidad de etiquetas se distribuye a lo largo de la colección. La cardinalidad de etiquetas, tal como fue formulada en la ecuación 2.13, es la cantidad de etiquetas promedio por instancia y es una de las métricas usadas para conocer el grado de multi-etiquetado de los datos (ver sección 2.4.1). Observar la distribución de etiquetas es útil para entender la composición de dicha cardinalidad y se calcula tomando el número de veces que se repite cada posible tamaño de subconjuntos de etiquetas. Basándose en este fenómeno se distinguen dos tipos de colecciones, "A" y "B". Las colecciones de tipo A son aquellas donde la cardinalidad de etiquetas es muy cercana a uno pero resultan ser multi-etiquetadas por la existencia de ejemplos donde el etiquetado único generaría ambigüedad y se resolvió añadiendo etiquetas. Tal es el caso para colecciones de artículos periodísticos [24] o de imágenes para realizar

detección de objetos [5]. Por contrapartida, las colecciones de tipo "B" son aquellas donde existe más de una etiqueta por instancia y suelen situarse en dominios abarcativos. Es el caso para colecciones de funciones genómicas, por ejemplo, en la cual se espera que los genes tengan múltiples funciones [8]. Otros ejemplos son los de correos electrónicos [22] y conceptos semánticos [38].

Relación entre etiquetas Esta cualidad tiene que ver con el concepto de interdependencia entre etiquetas, descripto en la sección 1.2.1, y se trata de capturar la aparición mutua de etiquetas en los ejemplos de tal manera de reflejar el grado de dependencia de las etiquetas en el dominio del problema. La idea es que un generador de instancias sintéticas debe ser capaz de asignar subconjuntos de etiquetas, ya no de manera aleatoria, sino respetando esta relación subyacente. Esto se puede lograr diseñando una matriz cuadrada de doble entrada que guarde las probabilidades condicionales entre pares de etiquetas.

Espacio de atributos Así como existen relaciones entre etiquetas que los algoritmos explotan para generar datos sintéticos de calidad, también puede analizarse el espacio de atributos y las interdependencias entre sí para optimizar los generadores. El estudio realizado por Read, Pfahringer y Holmes halló efectos o particularidades frecuentes en los datos: uno de ellos es el efecto "atributo-etiqueta" por el cual hay atributos que de aparecer en un ejemplo activan una etiqueta. El ejemplo dado surgió en una colección de artículos de noticias tecnológicas [35], en donde se observó que los atributos "`linux`" y "`mobile`" estaban fuertemente emparentados con las etiquetas "`Linux`" y "`Mobile`", respectivamente. Otro efecto hallado es el denominado "atributo-combinación" por el cual la aparición de un atributo activa un subconjunto de etiquetas en simultáneo. Por ejemplo, en una colección de noticias [24] se descubrió que el atributo "`arms`" ocurre frecuentemente con las etiquetas {`politics.guns`, `misc.religion`}. Del mismo modo, existe el "efecto aleatorio", el cual engloba a los atributos que no proveen información significativa sobre la presencia de etiquetas o combinaciones de etiquetas. En definitiva, los generadores pueden sacar provecho de este fenómeno otorgando parámetros que permitan ajustar el grado de presencia de los efectos mencionados y así lograr colecciones de datos sintéticos más realistas.

3. METODOLOGÍA

Conforme a los objetivos planteados en la sección 1.4 se presentan las colecciones y algoritmos que serán evaluados junto con el escenario de flujos continuos diseñado a este fin.

Se seleccionan tres colecciones de datos de multi-etiquetas que han sido puntos de referencia en otros trabajos de investigación. Cada uno de ellos es transformado en un flujo continuo de datos que cumple con las características presentadas en la sección 1.2.2.1. A su vez, se generan instancias sintéticas que sean fieles a las cualidades subyacentes de los datos originales. Para ello, se ha diseñado un algoritmo basado en la implementación de Read, Pfahringer y Holmes [34] pero que hace uso de la matriz de probabilidades condicionales entre pares de etiqueta para respetar sus interdependencias.

La etapa de entrenamiento se lleva a cabo con algoritmos de clasificación multi-etiquetas que han sido adaptados a ambientes de flujos continuos para hacer frente a las cualidades incrementales inherentes a este contexto. Se seleccionan algoritmos de la familia de "Transformación del problema" y "Adaptación del algoritmo" , junto con soluciones de ensamble, a fines explorativos y para extender el conocimiento sobre sus fortalezas y debilidades. Los experimentos se realizan con algoritmos implementados en el lenguaje de programación Python. Aquellos que no soportan datos de múltiples etiquetas han sido acondicionados a ese fin de acuerdo al diseño especificado en la literatura y consultando las respectivas implementaciones en otros lenguajes de programación, de hallarse estas disponibles al público.

por el momento no se experimentó con algoritmos AA en Python

De manera complementaria, se diseña una solución de ensambles llamada Ensamble Fijo por Mayoría Ponderada (EFMP) , basada en las implementaciones existentes. La misma usa como clasificadores base tres algoritmos de MLL diferentes, que se mantienen fijos durante todo el entrenamiento. Además, el ensamble mantiene una ponderación de cada clasificador de acuerdo a su rendimiento, penalizando por cada etiqueta mal clasificada, y la combinación de los votos se lleva a cabo por mayoría de voto ponderada. La implementación se basa en la presentada por Kolter y Maloof [23], quienes también ponderan los clasificadores pero usan un único tipo de clasificador base y no contemplan problemas de múltiples etiquetas. Los experimentos se realizan con dos versiones, una de ellas se entrena con todas las instancias del subconjunto de entrenamientos y la otra tomando muestreos siguiendo la distribución de *poisson*, tal como se realiza para ensambles del tipo de *Oza bagging* [29].

nombre tentativo

Para la etapa de evaluación se sigue la estrategia "*Prequential*", descripta en la sección 2.5.2, y se aplican métricas basadas en etiquetas y ejemplos, se mide la eficiencia de los modelos en términos de velocidad y espacio de almacenamiento y se analizan los resultados obtenidos.

El marco metodológico de este proyecto se ajusta a los procedimientos efectuados previamente por otros investigadores de la literatura [28, 39, 7, 48, 36]. Con ello se busca expandir el conocimiento empírico de los algoritmos al mismo tiempo que proporcionar nuevos estudios que sean contrastables con los ya existentes. Se destaca el trabajo de Read y col. [36], quienes analizan algoritmos multi-etiquetas con flujos reales y sintéticos, pero a diferencia de este trabajo, generan instancias sintéticas para colecciones nuevas y sin basarse en colecciones reales específicas. La implementación del generador que usaron en sus experimentos se encuentra disponible al público [33] y ha sido el punto de partida para desarrollar la técnica aquí propuesta. Büyükçakır, Bonab y Can, por su parte, no

generan flujos sintéticos pero conducen experimentos similares en lo que respecta al análisis de algoritmos, poniendo el foco en modelos de ensambles. La implementación de sus experimentos también ha sido liberada al público [1].

3.1. Técnicas Propuestas

En esta sección se describen dos técnicas implementadas para este trabajo: un generador de instancias sintéticas para flujos continuos de datos y el diseño de una solución de ensambles para realizar clasificaciones.

3.1.1. Generación de Flujos Sintéticos

El generador presentado es un algoritmo que emplea técnicas probabilísticas para hallar dependencias entre etiquetas y reproducirlas en las nuevas instancias. La existencia de interdependencias entre etiquetas ha sido explorada reiteradas veces en la literatura [41, 34] y se ha demostrado que existen dependencias condicionales e incondicionales, esto es, etiquetas que dependen entre sí dado uno o más atributos de una instancia (dependencia condicional), y etiquetas cuya dependencia existe para todo el conjunto de instancias (dependencia incondicional). Estas dos cualidades son directamente extraídas de las colecciones reales y a partir de ellas se genera el espacio de atributos y etiquetas que, al fusionarse, constituyen la instancia sintética.

La dependencia incondicional parte de la idea de que hay etiquetas que se activan en conjunto con frecuencia y otras que son mutuamente excluyentes. Véase el caso de las etiquetas "`Ficción`" y "`No Ficción`", por ejemplo, que son excluyentes en el dominio de géneros literarios. Para capturar esta relación se acude al concepto de probabilidad a priori y probabilidad condicional de etiquetas. La probabilidad a priori de una etiqueta es obtenida a partir de observar su frecuencia relativa en la colección normalizada por la Cardinalidad de etiquetas. La frecuencia relativa se formula de la manera tradicional:

Chequear fórmula con director!

$$FrecRelE_j = \frac{1}{m}\sum_{i=1}^{m} y_{i,j} \tag{3.1}$$

La normalización toma en cuenta el valor de cardinalidad de etiquetas del conjunto de datos, esto bajo la recomendación de los autores del trabajo de referencia [36].

$$NormCardE = \frac{1}{CardE}\sum_{j=1}^{q} FrecRelE_j \tag{3.2}$$

Donde $CardE$ se define tal como en 2.13, esto es:

$$CardE(D) = \frac{1}{m}\sum_{i=1}^{m} \|Y_i\| \tag{3.3}$$

Luego, la probabilidad a priori de la etiqueta j se expresa de la forma:

$$P(E_j) = \text{mín}\,(1, \frac{FrecRelE_j}{NormCardE}) \tag{3.4}$$

El resultado es un vector $[P(E_1), P(E_2), \ldots, P(E_q)]$.

A partir de $P(E_j)$ se puede calcular la matriz condicional θ sobre los pares de etiquetas, esto es, $\theta_{j,k} = P(Y_j = 1 \mid Y_k = 1)$, donde $1 \leq j \leq L$ y $1 \leq k \leq L$ con $j \neq k$. Con el vector

L es menor o igual a j en la bibliografía, es un error?

[1] `https://github.com/abuyukcakir/gooweml`

de probabilidades a priori y extrayendo las co-ocurrencias de cada par de etiquetas en toda la colección, es posible obtener cada valor de la matriz θ, aplicando la probabilidad condicional:

$$P(Y_j = 1 \mid Y_k = 1) = \frac{P(Y_k = 1 \cap Y_j = 1)}{P(Y_k)} \tag{3.5}$$

Luego, la dependencia entre etiquetas es modelada como la distribución conjunta:

$$p_\theta(y) = P(y_1) \prod_{j=2}^{q} P(y_j \mid y_{j-1}) \tag{3.6}$$

Posteriormente, se realiza la generación del conjunto de etiquetas para la instancia sintética. El algoritmo 1 muestra las instrucciones ejecutadas para concretar esta tarea. Cabe aclarar que *sample*() retorna un índice de etiqueta de acuerdo a una función de masa de probabilidad basada en las probabilidades a priori, y *random*() produce un número aleatorio de distribución uniforme.

Algorithm 1: Algoritmo de generación del conjunto de etiquetas para una instancia sintética

Input: q: Número de etiquetas de la colección, p: vector de probabilidades a priori, $p_\theta(y)$: función definida en fórmula 3.6
Output: y: las etiquetas generadas.

```
y ← ∅_q
j ← sample(p)
y_j ← 1
i ← 0
while i < q do
    if i = j then continue
    y' ← y
    y'_i ← 1
    if p_θ(y') > random() then y ← y'
    i ← i + 1
end
```

Una vez generado el conjunto de etiquetas resta generar los valores de atributos para la instancia. Para ello se retoma el concepto ya mencionado de "Dependencia Condicional", para conocer en qué medida la presencia de un atributo activa una o más etiquetas en la instancia, o expresado en términos formales, hallar el término $P(y|x)$ tal que:

$$P(y|x) = P(x|y)P(y) \tag{3.7}$$

Como el cálculo de la probabilidad conjunta es altamente compleja se define una función de mapeo $\zeta[a] \mapsto y_a$, donde y_a es la combinación de etiquetas más probable para el atributo a. La función θ se obtiene a través de muestreos sucesivos del generador de etiquetas, y guardando las A combinaciones más frecuentes, siendo el número total de atributos. Al mismo tiempo, el vector x candidato es obtenido usando un generador binario tal como los descriptos en la sección 2.5.3. El algoritmo 2 muestra un pseudocódigo de cómo se completa el proceso. Notar que el generador binario g produce dos vectores de atributos candidatos, uno por cada clase, luego si la combinación de etiquetas para el atributo a es un subconjunto de las etiquetas generadas se toma el valor del vector de atributos positivos. Caso contrario, se toma del vector de negativos.

Finalmente, la instancia sintética se forma a partir de la salida de ambos algoritmos, siendo de la forma (x, y). Este proceso será repetido para cada instancia que se solicite al generador a fin de generar el flujo sintético para la colección dada. El objetivo es obtener colecciones sintéticas que se asemejen a datos del mundo real, por lo tanto, la evaluación de los resultados se hará mediante un análisis de sus cualidades en relación con fenómenos hallados en datos reales (ver sección 2.5.3), y se contrastan los datos generados en este marco contra los producidos en el trabajo de referencia.

Algorithm 2: Algoritmo de generación del conjunto de atributos para una instancia sintética

Input: A: Número de atributos de la colección, g: Generador de atributos, ζ: Función de mapeo.
Output: x: El vector de atributos generado.

```
x ← ∅_A
positivos ← g(1)
negativos ← g(0)
i ← 0
while i < A do
    if ∃q : ζ[a] ⊆ y_q then
        x_i ← positivos_i
    else
        x_i ← negativos_i
    end
    i ← i + 1
end
```

3.1.2. Algoritmo de Ensamble

El Ensamble Fijo por Mayoría Ponderada (EFMP) es una estrategia de ensamble en ambientes de flujos continuos que pondera a los clasificadores base de acuerdo a su rendimiento y ajusta los pesos en cada predicción, a fin de optimizar la exactitud y eficiencia de la respuesta, y al mismo tiempo mantenerse actualizado frente a los cambios de concepto. Además es un ensamble que permite definir clasificadores base modelados a partir de algoritmos de clasificación diferentes para explotar la variabilidad en los mismos. La técnica de ponderación se basa en la presentada por Kolter y Maloof para la estrategia *Dynamic Weighted Majority* (DWM) [23] y fue ajustada para soportar datos de múltiples etiquetas. Otra de las cualidades del ensamble DWM es que agrega y elimina clasificadores base dinámicamente de acuerdo a su ponderación. Sin embargo, el costo computacional acarreado es notorio y los nuevos modelos añadidos son de un mismo tipo y no permite variarlos. En consecuencia, se presenta la estrategia EFMP como posible alternativa junto con una variación del mismo, EFMP2, que muestrea instancias según la distribución *poisson*. A continuación, se describen ambas técnicas en detalle.

EFMP mantiene un conjunto fijo de m clasificadores base, cada uno con un vector de pesos $W_k = [w_0, w_1, \ldots, w_q]$, donde $1 \leq k \leq m$ y $w_{k,j}$ representa el peso del clasificador k para la etiqueta j. En el entrenamiento del modelo se reciben n instancias donde $n = \|D\|$ para la estrategia simple. Además se definen los parámetros p, que es la cantidad de instancias observadas entre actualizaciones de los pesos, y β, que representa el factor en el que se decrece el peso $w_{k,j}$ ante cada clasificación errónea. β es un valor definido en el dominio $0 \leq \beta \leq 1$ y toma el valor 0,5 por defecto. Todos los pesos son inicializados en 1.

El proceso de aprendizaje se lleva a cabo de la siguiente manera: al arribar una instancia i, EFMP se la asigna a cada uno de los m clasificadores. En primer lugar se realiza la actualización de pesos y si un clasificador C_k no predice correctamente una etiqueta j, su peso $w_{k,j}$ será multiplicado por el factor β. Luego, se entrena cada clasificador con la instancia nueva y se repite el procedimiento con la siguiente. El parámetro p es usado durante esta etapa y determina los períodos entre los cuales no se deben actualizar los pesos. Una vez completado un período se normalizan los pesos de manera tal que el máximo peso entre etiquetas es uno. El algoritmo 3 presenta el pseudocódigo de este proceso en detalle.

Algorithm 3: Algoritmo de entrenamiento y ajuste de pesos para Ensamble Fijo por Mayoría Ponderada (EFMP)

Input: $\{X, Y\}$: Conjunto de entrenamiento, n: Número de instancias de entrenamiento, q: Número de etiquetas, β: Factor de decrecimiento de los pesos, p: Período entre actualizaciones de los pesos, C: Clasificadores base, W: Pesos de los clasificadores

```
i ← 0
m ← ||C||
while i < n do
    if i mód p = 0 then
        k ← 0
        while k < m do
            y_i ← predecir(C_k, X_i)
            j ← 0
            while j < q do
                if y_{i,j} ≠ Y_{i,j} then
                    W_{k,j} ← β * W_{k,j}
                end
                j ← j + 1
            end
            k ← k + 1
        end
        W ← escalarPesos(W)
    end
    k ← 0
    while k < m do
        entrenar(C_k, X_i, Y_i)
        k ← k + 1
    end
    i ← i + 1
end
```

Durante la etapa de predicción cada clasificador retorna su voto $v_{i,j,k}$ y el ensamble realiza la combinación computando la suma ponderada para cada etiqueta. La ecuación es la siguiente:

$$y_{i,j} = \frac{1}{\sum_{k=1}^{m} w_{j,k}} \sum_{k=1}^{m} v_{i,j,k} * w_{j,k} \tag{3.8}$$

Las etiquetas cuyo valor superen un umbral de 0,5 serán activadas:

$$y_{i,j} = \begin{cases} 1, & \text{si } y_{i,j} \geq 0{,}5 \\ 0, & \text{de otro modo.} \end{cases} \tag{3.9}$$

EFMP2 introduce una única variación, durante el entrenamiento. Cuando arriba una instancia i, cada uno de los clasificador base entrena el modelo $Poisson(1)$ veces con dicha instancia. Esta técnica es conocida como *online bagging* y se ha probado que se aproxima a la estrategia tradicional de *bagging* en *batch*, cuando se entrena con ejemplos de distribución similar [29].

explicar: como se adaptaron los algoritmos? el script usado para entrenar y predecir? los archivos generados con los resultados?

Explicar más sobre el análisis y evaluación de los flujos sintéticos??

4. EXPERIMENTOS Y RESULTADOS

escribir intro de capitulo

4.1. Colecciones

Se seleccionan colecciones de datos multi-etiquetas del mundo real que han sido aplicados previamente en la literatura para evaluar la capacidad predictiva de los modelos de clasificación [28, 36, 7]. La tabla 4.1 enumera sus características principales, incluyendo métricas que describen su grado de multi-etiquetado (ver sección 2.4.1). Una descripción detallada de cada una se lista a continuación:

20ng [1]: Es una colección que consta de casi 20 mil publicaciones provenientes de grupos de noticias y que abordan 20 tópicos diferentes [24]. La colección es de texto y fue preprocesada para formar 1006 atributos numéricos.

Enron [2]: Es una colección de correos electrónicos seleccionados de entre los 500 mil generados por empleados de la compañía eléctrica *Enron* y filtrados durante una investigación por corrupción [22]. Su tamaño, que no supera los 2000 elementos, no es lo suficientemente grande para ser considerado un flujo continuo voluminoso, pero sí cuenta con otras propiedades como la inclusión de fechas y una evolución de los datos en el tiempo [36]. Las etiquetas se dividen en cuatro grupos, según su género (acuerdos laborales, correos meramente personales, etc); según la información que incluyen, esto es, si el correo contiene enlaces externos, adjuntos, reenvíos, etc; según el tono emocional que reflejan y según el tópico principal que abordan.

Mediamill [3]: Es una colección generada a partir de 80 horas de video provenientes de transmisiones de noticias durante noviembre de 2004 [38]. Se seleccionaron más de 43 mil ejemplos y fue manualmente etiquetada con 101 conceptos, que pueden visualizarse en la figura 4.1.

Nombre	Dominio	N	A	L	LC	LD
20ng	Texto	19300	1006	20	1,029	0,051
Enron	Texto	1702	1001	53	3,378	0,064
Mediamill	Video	43907	120	101	4,376	0,043

Tab. 4.1: Colecciones multi-etiquetas y sus características. N: número de instancias; A: número de atributos; L: número de etiquetas; LC: cardinalidad de etiquetas; LD: densidad de etiquetas.

Estas son sólo tres de las colecciones usualmente abordadas en la literatura y se han seleccionado con el objetivo de diversificar el análisis. Enron es una colección de pocas instancias pero muchas etiquetas, 20ng a la inversa, cuenta con pocas etiquetas pero muchas instancias; y Mediamill, finalmente, es la colección con más instancias que hay disponible y cuenta también con un número relativamente alto de etiquetas.

[1] `https://www.uco.es/kdis/mllresources/`
[2] `http://sourceforge.net/projects/mulan/files/datasets/enron.rar`
[3] `https://sourceforge.net/projects/mulan/files/datasets/mediamill.rar`

Fig. 4.1: Los 101 conceptos semánticos asociados a la colección Mediamill.

Durante la ejecución de experimentos, cada colección será convertida a un flujo sintético. Además, se generará una versión sintética de cada una, siguiendo la técnica descripta en la sección 3.1.1.

4.2. Software

A continuación se describen las herramientas de software que fueron utilizadas para la implementación y ejecución de los experimentos.

scikit-multiflow [4] Es una librería disponible para el lenguaje de programación Python que provee un *framework* para implementar y comparar algoritmos de aprendizaje automático en ambientes de flujos continuos de datos. Incluye pero no se limita a problemas de clasificación multi-etiquetas [27].

MOA [5] Massive Online Analysis (MOA) es un *framework* para realizar minería de datos sobre flujos continuos de datos, implementada en Java y de código libre. Incluye algoritmos de evaluación y de aprendizaje automático como clasificadores, regresores, o de *clustering*, pudiendo ser aplicados a problemas de clasificación de etiqueta única o multi-etiquetas. También incluye herramientas para generar datos sintéticos. Tanto MOA como scikit-multiflow facilitan la reiteración de experimentos con distintas configuraciones, así como la comparación de resultados y la extensión de funcionalidad [4].

[4] `https://scikit-multiflow.github.io/`
[5] `https://moa.cms.waikato.ac.nz/`

scikit-learn [6] Es una librería del lenguaje de programación Python que brinda herramientas para realizar evaluación, visualización y análisis de resultados [30].

Mulan [7] Es una librería del lenguaje Java especializada en aprendizaje por multi-etiquetas. Mulan incluye una variedad de colecciones de datos multi-etiquetas que han sido la fuente de otros trabajos de la literatura [43].

nombres de herramientas van en cursiva?

La herramienta MOA es usada para generar los flujos sintéticos y provee del marco de trabajo en el cual se implementó el algoritmo de generación descripto en 3.1.1. Los algoritmos de clasificación fueron implementados en Python y están disponibles bajo la librería *scikit-multiflow*. La solución de ensamble EFMP también fue implementada en esta librería. *scikit-learn*, por su parte, provee la implementación de las métricas basadas en etiquetas, y las colecciones de datos fueron extraídas de mulan.

4.3. Hardware

Se ha recibido apoyo del Centro de Investigación, Docencia y Extensión en TIC de la Universidad Nacional de Luján (CIDETIC), el cual ha proveído de equipos de altas prestaciones que han proporcionado la capacidad de cómputo necesaria para llevar a cabo este proyecto. El equipamiento facilitado cuenta con dos nodos de 12 núcleos cada uno, el CPU es un Intel Xeon X5675 de 3.07 GHz de velocidad de procesamiento, 12 Mb de memoria caché y 6 núcleos. El espacio de almacenamiento disponible es de 1 Tb y la memoria RAM es de 142 Gb.

Chequear estos datos

El Sistema Operativo instalado es Ubuntu 18.04 LTS y cuenta con la versión 3.6.9 de Python, el instalador de paquetes Pip en su versión 20.3.3 y Java 1.8.

4.4. Algoritmos

Se realizan los experimentos usando algoritmos multi-etiquetas disponibles en la librería scikit-multiflow junto con las implementaciones de ensambles presentadas en este trabajo: Ensamble Fijo por Mayoría Ponderada (EFMP) y su variación EFMP2. Entre los algoritmos del tipo de transformación del problema se seleccionan los de Binary Relevance (BR), Classifier Chains (CC) y Multi-label Hoeffding Tree (MLHT). Tanto BR como CC usan *naive* bayes como modelo de clasificación base y MLHT es ejecutado en su versión basada en Label Powerset (LP), siguiendo los procedimientos de Read y col. [36].

En lo que respecta a soluciones de ensamble, los modelos de EFMP contarán ambos con tres clasificadores base, siendo estos los mencionados en el párrafo anterior, es decir, CC, BR y MLHT. La comparación se hará contra el algoritmo Dynamic Weighted Majority (DWM), tal como ha sido definido por sus autores [23] pero adaptado a ambientes de multi-etiquetas (ver sección 3.1.2), y se suman al análisis los algoritmos de Ensamble de Binary Relevance (EBR) y Ensamble de Classifier Chains (ECC), tal como fueron definidos por Oza [29] y también han sido extendidos para soportar problemas de MLL [35]. Los tres algoritmos de ensamble extraídos de la literatura son configurados con diez clasificadores base de *naive* bayes, para imitar los experimentos conducidos por otros autores de la literatura [28, 36, 7].

La tabla 4.2 es un resumen de los algoritmos seleccionados junto con los clasificadores base configurados, la referencia bibliográfica y la clave que será usada en las tablas de resultados.

[6] `http://scikit-learn.org/stable/index.html`
[7] `http://mulan.sourceforge.net/index.html`

Clave	Algoritmo	Clasificadores base	Referencia
BR	Binary Relevance	*naive* bayes	Tsoumakas y Katakis [41]
CC	Classifier Chains	*naive* bayes	Read y col. [35]
MLHT	Multi-label Hoeffding Tree	Hoeffding Tree (HT)	Read y col. [36]
EFMP	Ensamble Fijo por Mayoría Ponderada	BR, CC y MLHT	Sección 3.1.2
EFMP2	Ensamble Fijo por Mayoría Ponderada 2	BR, CC y MLHT	Sección 3.1.2
DWM	Dynamic Weighted Majority	*naive* bayes (10 copias)	Kolter y Maloof [23]
EBR	Ensamble de Binary Relevance	*naive* bayes (10 copias)	Read y col. [35]
ECC	Ensamble de Classifier Chains	*naive* bayes (10 copias)	Read y col. [35]

Tab. 4.2: Métodos de clasificación multi-etiquetas seleccionados para ambientes de flujos continuos de datos.

4.5. Métricas de Evaluación

En la evaluación de algoritmos de clasificación se usan el conjunto de métricas que han sido utilizadas en otros trabajos de la literatura, tanto en escenarios de flujos [39, 48, 28] como en *batch* [25, 46, 16] y fueron descriptas en la sección 2.4.3. Estas son:

Métricas Basadas en Ejemplos : *Hamming score*, exactitud del subconjunto, exactitud (ejemplos), precisión (ejemplos), exhaustividad (ejemplos) y medida-f1 (ejemplos).

Métricas Basadas en Etiquetas : Micro-exactitud, micro-precisión, micro-exhaustividad, micro-f1, macro-exactitud, macro-precisión, macro-exhaustividad y macro-f1.

Métricas de Eficiencia : Velocidad y consumo de memoria.

La medición de velocidad comienza en el momento que inicia la predicción y entrenamiento del modelo por primera vez y finaliza cuando el clasificador termina de procesar la última instancia de la colección. Por lo tanto, quedan afuera las etapas de evaluación, carga de la colección en memoria, generación del flujo y configuración del entrenamiento. El consumo de memoria también es monitoreado durante la ejecución del entrenamiento y predicción y toma en cuenta la estructura completa del modelo y todos sus componentes, incluyendo pesos e hiper-parámetros propios y de sus clasificadores base.

Los flujos sintéticos son analizados teniendo en cuenta los fenómenos propios de colecciones del mundo real. A ese fin se estudia el sesgo de etiquetas, la relación entre etiquetas, la distribución de etiquetas y el espacio de atributos (ver sección 2.5.3).

es probable que este análisis se separe en una sección aparte y se aborde con mayor profundidad

4.6. Configuración Experimental

En lo que respecta a modelos de aprendizaje automático, los experimentos fueron desarrollados en el lenguaje Python usando la librería *scikit-multiflow*. Los algoritmos de transformación del problema se aplican tal como han sido implementados en la librería con la salvedad del MLHT, al que debió introducirle una modificación para manipular la predicción, se usaba un arreglo disperso para representar las etiquetas activadas, lo cual producía un desbordamiento de memoria en el entrenamiento de colecciones grandes como la de mediamill. Se lo suplantó por una estructura de representación densa. En cuanto a los modelos de ensambles, se adaptaron las implementaciones existentes de EBR, ECC y DWM para soportar múltiples etiquetas y para ello se debió modificar la etapa de combinación de votos para hacer frente a la nueva dimensionalidad de los datos. Por lo demás, la configuración de los algoritmos es la definida en la sección 4.4.

Para la etapa de evaluación se aplica la técnica de evaluación secuencial predictiva (*prequential*) con ventanas deslizantes, tal como se recomienda para ambientes de flujos continuos [13]. Ante cada ejemplo o ventana de ejemplos arribada el modelo primero realiza la predicción y luego el entrenamiento. Finalmente las métricas de evaluación son calculadas una vez procesados todos los ejemplos de la colección y a partir de todas las predicciones producidas. Notar que a partir de esta técnica el modelo predice y entrena todas las instancias, y no solo un subconjunto de ellas como sucede con la estrategia de *holdout*. La ventana deslizante se configura en $w = \frac{N}{20}$, es decir, se divide el número total de instancias del flujo en 20 ventanas, siguiendo las directivas de Read y col. [36]. Los resultados de la evaluación son agrupados según los tipos de métrica usados, para facilitar el análisis.

Por otro lado, los flujo de datos sintéticos fueron generados a partir de las colecciones 20ng, enron y mediamill, y cada una cuenta con cien mil instancias. Sus atributos se generan usando el algoritmo Generador de Función Radial Base (RBF) (ver sección 2.5.3).

4.7. Resultados

4.7.1. Flujos Continuos Sintéticos

Completar valores faltantes

	20ng					Enron				
Nombre	N	L	LC	LD	Nombre	N	L	LC	LD
20NG	19300	20	1,029	0,051	ENRON	1702	53	3,378	0,064
MOA	19300	20	3,397	0,170	MOA	1702	53	3,707	0,070
JC	19300	20	1,067	0,053	JC	1702	53	2,330	0,044
JC_80k	80000	20	1,062	0,053	JC_100k	100000	53	2,321	0,044

	Mediamill			
Nombre	N	L	LC	LD
MEDIAMILL	43907	101	4,376	0,043
MOA	43907	101	4,071	0,040
JC	43907	101	2,435	0,024
JC_500k	500000	101	2,439	0,024

Tab. 4.3: Características de las colecciones sintéticas generadas. N: número de instancias; L: número de etiquetas; LC: cardinalidad de etiquetas; LD: densidad de etiquetas.

4.7.2. Clasificaciones

La metodología propuesta permitió evaluar los diferentes algoritmos de clasificación multi-etiqueta para los diferentes *streams* utilizando las configuraciones sin ensambles, los ensambles de referencias y los métodos de ensamble propuestos. Los resultados se dividen en métricas de ajustes del modelo basadas en ejemplos (tabla 4.4), métricas basadas en etiquetas (tabla 4.5), y por último las métricas de eficiencia (tabla 4.6) que cuantifican el tiempo de procesamiento y espacio de almacenamiento de los modelos. Se marca en negrita la celda correspondiente al modelo que obtuvo el mejor valor de métrica para la correspondiente colección de datos. Para cerrar la sección se hace una comparativa contra experimentos de la literatura de referencia.

4.7.2.1. Resultados para Métricas Basadas en Ejemplos

	Exact-match			*Hamming score*			*Hamming loss*		
Stream	20ng	Enron	Mediamill	20ng	Enron	Mediamill	20ng	Enron	Mediamill
BR	0.228	0.018	0.001	0.952	0.912	0.711	0.048	**0.088**	**0.289**
CC	0.244	0.017	0.005	0.954	0.919	0.905	0.046	0.081	0.095
MLHT	**0.315**	**0.055**	**0.048**	0.934	0.928	**0.957**	**0.066**	0.072	0.043
DWM (BR)	0.171	0.021	0.001	**0.956**	0.938	0.800	0.044	0.062	0.200
DWM (CC)	0.167	0.024	0.014	0.956	**0.939**	0.936	0.044	0.061	0.064
EBR	0.153	0.012	0.001	0.953	0.928	0.726	0.047	0.072	0.274
ECC	0.115	0.012	0.001	0.952	0.938	0.923	0.048	0.062	0.077
EFMP	0.241	0.032	0.012	0.954	0.928	0.946	0.046	0.072	0.054
EFMP2	0.220	0.039	0.004	0.955	0.936	0.928	0.045	0.064	0.072

	Accuracy			*Precision*			*Recall*			*F-score (F1)*		
Stream	20ng	Enron	Mediamill	20ng	Enron	Mediamill	20ng	Enron	Mediamill	20ng	Enron	Mediamill
BR	0.952	0.912	0.711	0.295	0.326	0.093	**0.373**	**0.365**	**0.651**	**0.329**	0.345	0.163
CC	0.954	0.919	0.905	0.294	0.337	0.229	0.339	0.353	0.485	0.315	0.344	0.311
MLHT	0.934	0.928	**0.957**	**0.327**	0.269	**0.488**	0.321	0.127	0.378	0.324	0.172	**0.426**
DWM (BR)	**0.956**	0.938	0.800	0.187	0.305	0.129	0.197	0.183	0.623	0.192	0.229	0.214
DWM (CC)	0.956	**0.939**	0.936	0.178	0.308	0.350	0.183	0.185	0.466	0.181	0.231	0.400
EBR	0.953	0.928	0.726	0.185	0.320	0.099	0.213	0.269	0.649	0.198	0.292	0.171
ECC	0.952	0.938	0.923	0.136	0.375	0.291	0.149	0.227	0.583	0.142	0.283	0.388
EFMP	0.954	0.928	0.946	0.302	0.380	0.404	0.361	0.351	0.418	0.328	**0.365**	0.411
EFMP2	0.955	0.936	0.928	0.252	**0.389**	0.303	0.277	0.303	0.516	0.264	0.340	0.382

Tab. 4.4: Resultados de métricas basadas en ejemplos sobre los *streams* seleccionados para cada algoritmo evaluado.

Los valores de F1 obtenidos para la evaluación basada en ejemplos (Tabla 4.4) muestra que EFMP y EFMP2 fueron mejores que los *baselines* de ensambles en todos los casos. Además, superó a los que no utilizan ensambles en el *stream* de Enron. En los casos de 20ng, EFMP fue superado en un 0.001 % por BR y en Mediamill MLHT superó a EFMP en un 0.015 %. Para el caso de *Exact-match* el modelo dominante es MLHT, lo cual es un resultado en consonancia con otros estudios de la literatura, y los modelos propuestos se ubican en segundo lugar para dos de las tres colecciones. En cuanto al *Hamming score* los resultados son muy similares entre colecciones de datos, con los modelos de DWM sacando una leve ventaja para 20ng y Enron pero siendo superado por EFMP y MLHT para Mediamill.

4.7.2.2. Resultados para Métricas Basadas en Etiquetas

La métrica de *F-score macro* muestra resultados favorecedores para los modelos presentados. EFMP2 obtuvo un valor superior para Mediamill y EFMP fue el mejor para 20ng y el segundo mejor para Enron, por centésimas de diferencia con respecto al modelo BR. En lo que respecta a la comparativa entre soluciones de ensambles, EFMP y EFMP2 superan a las demás para todas las colecciones evaluadas. En relación a los modelos de DWM, estos muestran una disparidad entre los valores de precisión y *recall*. Vease por ejemplo el caso de DWM (CC) para 20ng, donde logra más de un 80 % de precisión (mejor clasificador) pero un *recall* por debajo del 20 % (segundo peor clasificador). Mismo caso pero a la inversa con DWM (BR) para Mediamill, el cual consigue un 40 % de *recall* (tercer mejor clasificador) pero apenas un 0.006 % de precisión (tercer peor clasificador). Esta imparidad logra suavizarse en los modelos de EFMP presentados y se ve reflejado en mejores valores de *f-score*.

También son favorecedores los valores de las métricas de *F-score micro* para los modelos

	Precision (macro)			*Recall* (macro)			*F-score* (macro)		
Stream	20ng	Enron	Mediamill	20ng	Enron	Mediamill	20ng	Enron	Mediamill
BR	0.604	0.106	0.062	**0.373**	0.110	**0.553**	0.461	**0.108**	0.111
CC	0.667	0.113	0.065	0.340	**0.097**	0.150	0.450	0.105	0.091
MLHT	0.546	0.005	0.074	0.318	0.016	0.030	0.402	0.008	0.043
DWM (BR)	0.781	0.121	0.064	0.196	0.031	0.418	0.314	0.049	0.111
DWM (CC)	**0.814**	0.114	0.068	0.183	0.030	0.111	0.299	0.047	0.084
EBR	0.687	0.100	0.062	0.214	0.068	0.531	0.326	0.081	0.111
ECC	0.753	0.119	0.049	0.150	0.046	0.112	0.250	0.067	0.068
EFMP	0.649	0.114	0.074	0.361	0.082	0.072	**0.464**	0.096	0.073
EFMP2	0.699	**0.143**	**0.096**	0.276	0.059	0.173	0.396	0.083	**0.124**
	Precision (micro)			*Recall* (micro)			*F-score* (micro)		
Stream	20ng	Enron	Mediamill	20ng	Enron	Mediamill	20ng	Enron	Mediamill
BR	0.552	0.330	0.096	**0.373**	**0.364**	**0.673**	0.445	0.347	0.168
CC	0.597	0.364	0.224	0.340	0.353	0.485	0.433	0.358	0.306
MLHT	0.345	0.283	**0.509**	0.318	0.079	0.342	0.331	0.124	**0.410**
DWM (BR)	0.773	0.552	0.131	0.197	0.178	0.642	0.313	0.269	0.218
DWM (CC)	**0.796**	**0.569**	0.328	0.183	0.179	0.463	0.298	0.272	0.384
EBR	0.633	0.412	0.101	0.214	0.285	0.671	0.320	0.337	0.175
ECC	0.644	0.534	0.299	0.150	0.242	0.579	0.243	0.333	0.395
EFMP	0.586	0.417	0.384	0.361	0.339	0.403	**0.447**	**0.374**	0.393
EFMP2	0.644	0.498	0.305	0.276	0.293	0.511	0.387	0.369	0.382

Tab. 4.5: Resultados de métricas basadas en etiquetas sobre los *streams* seleccionados para cada algoritmo evaluado.

presentados. EFMP es el mejor para las colecciones 20ng y Enron y es apenas superado por MLHT para el *stream* de Mediamill. En la comparativa de métodos de ensambles, EFMP es el mejor modelo para todos las métricas a excepción del caso de *recall* para Mediamill (donde EBR obtiene el mejor valor), y los casos de precisión para Enron y 20ng, donde DWM (CC) logra una clara diferencia. Al respecto de este ultimo modelo se puede hacer las mismas consideraciones en cuanto a la disparidad entre precisión y *recall*.

4.7.2.3. Resultados para Métricas de Eficiencia

	Tamaño del modelo (kb)			Tiempo de ejecución (segundos)		
Stream	20ng	Enron	Mediamill	20ng	Enron	Mediamill
BR	31.6	**82.9**	**18.9**	**1:43:26**	**0:21:28**	**2:08:42**
CC	31.9	85.1	26.7	1:45:00	0:27:05	2:52:36
MLHT	**22.8**	284.7	306.0	2:22:30	0:58:16	19:05:49
DWM (BR)	305.5	782.9	185.3	21:23:37	5:15:03	1 día, 4:21:27
DWM (CC)	308.4	802.7	262.4	21:03:02	5:05:56	1 día, 12:49:14
EBR	316.1	826.8	189.2	14:14:56	3:41:00	20:13:01
ECC	319.0	847.6	267.2	14:35:33	4:03:16	1 día, 1:23:01
EFMP	86.4	452.8	351.7	5:41:58	1:45:51	1 día, 0:47:17
EFMP2	84.7	359.2	262.2	5:36:55	2:03:59	1 día, 6:27:51

Tab. 4.6: Resultados de métricas de eficiencia sobre los *streams* seleccionados para cada algoritmo evaluado.

Tal como es de esperar, la tabla 4.6 muestra que los modelos *baselines* que no son

soluciones de ensambles hacen un uso de espacio significativamente menor que los modelos de ensambles y logran tiempos de ejecución menores. Sin embargo, en la comparativa entre ensambles, los algoritmos propuestos (EFMP y EFMP2) reducen tanto el espacio de almacenamiento como el tiempo de procesamiento de los *streams* de Enron y 20ng. Las métricas de tiempo se reducen significativamente para 20NG y Enron, mientras que en Mediamill EBR hace un uso significativamente menor de tiempo que los ensambles presentados.

4.7.2.4. Comparativa contra Literatura de Referencia

El resultado de los métodos propuestos muestra que son competitivos respecto a la literatura de referencia. En particular, para las pruebas realizadas con el conjunto de datos 20NG los valores de F-Score basado en ejemplos obtenidos superan en 2.3 % a [28] pero no son mejores que otros como [39, 7, 37]. En las pruebas realizadas con Enron, los métodos propuestos superan en 10.74 % a [28], duplican el rendimiento de [37] y son superados por [39] en un 22.6 %. Finalmente, para el conjunto de datos Mediamill tanto [39] como [37] superan nuestra propuesta.

5. CONCLUSIONES

ABCDEF

BIBLIOGRAFíA

[1] Ömer Faruk Arar y Kürşat Ayan. «A feature dependent Naive Bayes approach and its application to the software defect prediction problem». en. En: *Applied Soft Computing* 59 (oct. de 2017), págs. 197-209. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2017.05.043. URL: http://www.sciencedirect.com/science/article/pii/S1568494617303083 (visitado 11-01-2021).

[2] Albert Bifet y Gianmarco De Francisci Morales. «Big Data Stream Learning with SAMOA». En: *2014 IEEE International Conference on Data Mining Workshop*. 2014.

[3] Albert Bifet y col. *Machine Learning for Data Streams with Practical Examples in MOA*. Mar. de 2018. ISBN: 978-0-262-03779-2.

[4] Albert Bifet y col. «MOA: massive online analysis». En: *Journal of Machine Learning Research* 11 (mayo de 2010).

[5] Matthew Boutell y col. «Learning multi-label scene classification». En: *Pattern Recognition* 37 (sep. de 2004), págs. 1757-1771. DOI: 10.1016/j.patcog.2004.03.009.

[6] Leo Breiman. «Bagging predictors». en. En: *Machine Learning* 24.2 (ago. de 1996), págs. 123-140. ISSN: 1573-0565. DOI: 10.1007/BF00058655. URL: https://doi.org/10.1007/BF00058655 (visitado 23-01-2021).

[7] Alican Büyükçakır, Hamed Bonab y Fazli Can. «A Novel Online Stacked Ensemble for Multi-Label Stream Classification». En: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (oct. de 2018). arXiv: 1809.09994, págs. 1063-1072. DOI: 10.1145/3269206.3271774. URL: http://arxiv.org/abs/1809.09994 (visitado 12-02-2021).

[8] Sotiris Diplaris y col. «Protein Classification with Multiple Algorithms». en. En: *Advances in Informatics*. Ed. por Panayiotis Bozanis y Elias N. Houstis. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, págs. 448-456. ISBN: 978-3-540-32091-3. DOI: 10.1007/11573036_42.

[9] Pedro Domingos y Geoff Hulten. «Mining High-Speed Data Streams». En: *Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (nov. de 2002). DOI: 10.1145/347090.347107.

[10] Uma Dulhare. «Prediction system for heart disease using Naive Bayes and particle swarm optimization». En: *Biomedical Research* 29 (ene. de 2018). DOI: 10.4066/biomedicalresearch.29-18-620.

[11] Usama M Fayyad, Gregory Piatetsky-Shapiro y Padhraic Smyth. *Advances in Knowledge Discovery and Data Mining*. Ed. por Fayyad, Usama M. and Piatetsky-Shapiro, Gregory and Smyth, Padhraic and Uthurusamy, Ramasamy. Section: From data mining to knowledge discovery: an overview. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996.

[12] João Gama. *Knowledge Discovery from Data Streams*. 2010.

[13] João Gama, Raquel Sebastião y Pedro Pereira Rodrigues. «On evaluating stream learning algorithms». en. En: *Machine Learning* 90.3 (mar. de 2013), págs. 317-346. ISSN: 1573-0565. DOI: 10.1007/s10994-012-5320-9. URL: https://doi.org/10.1007/s10994-012-5320-9 (visitado 16-02-2021).

[14] J Gantz y D Reinsel. «Extracting value from chaos». En: *IDC IView* (2011), págs. 1-12.

[15] Francesco Gargiulo, Stefano Silvestri y Mario Ciampi. «Deep Convolution Neural Network for Extreme Multi-label Text Classification». En: *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies*. 2018.

[16] Eva Gibaja y Sebastian Ventura. «A Tutorial on Multi-Label Learning». En: *ACM Computing Surveys* 47 (2015).

[17] Trevor Hastie, Robert Tibshirani y Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. en. 2.ª ed. Springer Series in Statistics. New York: Springer-Verlag, 2009. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7. URL: https://www.springer.com/gp/book/9780387848570 (visitado 12-01-2021).

[18] Francisco Herrera y col. *Multilabel Classification*. Ene. de 2016. ISBN: 978-3-319-41110-1. DOI: 10.1007/978-3-319-41111-8.

[19] Geoff Hulten, Laurie Spencer y Pedro Domingos. «Mining Time-changing Data Streams». En: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '01. San Francisco, California: ACM, 2001, págs. 97-106.

[20] Harsha K. Kalutarage y col. «Detecting stealthy attacks: Efficient monitoring of suspicious activities on computer networks». en. En: *Computers & Electrical Engineering* 47 (oct. de 2015), págs. 327-344. ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2015.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0045790615002384 (visitado 11-01-2021).

[21] Richard Kirkby. *Improving Hoeffding Trees*. en. 2007. URL: /paper/Improving-Hoeffding-Trees-Kirkby/56283855992584581eb9c0eb4413b47be496b94e (visitado 30-01-2021).

[22] Bryan Klimt y Yiming Yang. «The Enron Corpus: A New Dataset for Email Classification Research». en. En: *Machine Learning: ECML 2004*. Ed. por David Hutchison y col. Vol. 3201. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, págs. 217-226. ISBN: 978-3-540-30115-8. DOI: 10.1007/978-3-540-30115-8_22. URL: http://link.springer.com/10.1007/978-3-540-30115-8_22 (visitado 15-06-2020).

[23] J. Zico Kolter y Marcus A. Maloof. «Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts». En: *The Journal of Machine Learning Research* 8 (dic. de 2007), págs. 2755-2790. ISSN: 1532-4435.

[24] Ken Lang. «NewsWeeder: Learning to Filter Netnews». en. En: *Machine Learning Proceedings 1995*. Ed. por Armand Prieditis y Stuart Russell. San Francisco (CA): Morgan Kaufmann, ene. de 1995, págs. 331-339. ISBN: 978-1-55860-377-6. DOI: 10.1016/B978-1-55860-377-6.50048-7. URL: http://www.sciencedirect.com/science/article/pii/B9781558603776500487 (visitado 01-02-2021).

[25] Gjorgji Madjarov y col. «An extensive experimental comparison of methods for multi-label learning». en. En: *Pattern Recognition* 45.9 (sep. de 2012), págs. 3084-3104. ISSN: 00313203. DOI: 10.1016/j.patcog.2012.03.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320312001203 (visitado 15-06-2020).

[26] V Mayer-Schonberger y K Cukier. *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. An Eamon Dolan book. Houghton Mifflin Harcourt, 2013.

[27] Jacob Montiel y col. «Scikit-Multiflow: A Multi-output Streaming Framework». En: *arXiv:1807.04662 [cs, stat]* (jul. de 2018). arXiv: 1807.04662. DOI: 10.5555/3291125.3309634. URL: http://arxiv.org/abs/1807.04662 (visitado 15-02-2021).

[28] Aljaž Osojnik, Panče Panov y Sašo Džeroski. «Multi-label classification via multi-target regression on data streams». en. En: *Machine Learning* 106.6 (jun. de 2017), págs. 745-770. ISSN: 1573-0565. DOI: 10.1007/s10994-016-5613-5. URL: https://doi.org/10.1007/s10994-016-5613-5 (visitado 15-06-2020).

[29] N. C. Oza. «Online bagging and boosting». En: *2005 IEEE International Conference on Systems, Man and Cybernetics*. Vol. 3. ISSN: 1062-922X. Oct. de 2005, 2340-2345 Vol. 3. DOI: 10.1109/ICSMC.2005.1571498.

[30] Fabian Pedregosa y col. «Scikit-learn: Machine Learning in Python». En: *arXiv:1201.0490 [cs]* (jun. de 2018). arXiv: 1201.0490. URL: http://arxiv.org/abs/1201.0490 (visitado 15-02-2021).

[31] Robi Polikar. «Polikar, R.: Ensemble based systems in decision making. IEEE Circuit Syst. Mag. 6, 21-45». En: *Circuits and Systems Magazine, IEEE* 6 (oct. de 2006), págs. 21-45. DOI: 10.1109/MCAS.2006.1688199.

[32] J. Read, B. Pfahringer y G. Holmes. *Generating Synthetic Multi-label Data Streams*. en. 2009. URL: /paper/Generating-Synthetic-Multi-label-Data-Streams-Read-Pfahringer/147e3bc5f3c03884a8ba6d5420dc100834424c5d (visitado 31-01-2021).

[33] Jesse Read. *MOA: moa.streams.generators.multilabel.MetaMultilabelGenerator Class Reference*. 2012. URL: https://www.cs.waikato.ac.nz/~abifet/MOA/API/classmoa_1_1streams_1_1generators_1_1multilabel_1_1_meta_multilabel_generator.html (visitado 20-02-2021).

[34] Jesse Read, Bernhard Pfahringer y Geoff Holmes. «Multi-label Classification Using Ensembles of Pruned Sets». En: *2008 Eighth IEEE International Conference on Data Mining*. 2008.

[35] Jesse Read y col. «Classifier chains for multi-label classification». En: *Mach. Learn.* 85.3 (2011), págs. 333-359.

[36] Jesse Read y col. «Scalable and efficient multi-label classification for evolving data streams». en. En: *Machine Learning* 88.1 (jul. de 2012), págs. 243-272. ISSN: 1573-0565. DOI: 10.1007/s10994-012-5279-6. URL: https://doi.org/10.1007/s10994-012-5279-6 (visitado 17-06-2020).

[37] Martha Roseberry y Alberto Cano. «Multi-label kNN Classifier with Self Adjusting Memory for Drifting Data Streams». en. En: *Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*. ISSN: 2640-3498. PMLR, nov. de 2018, págs. 23-37. URL: http://proceedings.mlr.press/v94/roseberry18a.html (visitado 18-08-2021).

[38] Cees G. M. Snoek y col. «The challenge problem for automated detection of 101 semantic concepts in multimedia». en. En: *Proceedings of the 14th annual ACM international conference on Multimedia - MULTIMEDIA '06*. Santa Barbara, CA, USA: ACM Press, 2006, pág. 421. ISBN: 978-1-59593-447-5. DOI: 10.1145/1180639.1180727. URL: http://portal.acm.org/citation.cfm?doid=1180639.1180727 (visitado 15-06-2020).

[39] Ricardo Sousa y João Gama. «Multi-label classification from high-speed data streams with adaptive model rules and random rules». En: *Progress in Artificial Intelligence* (2018).

[40] Erica Akemi Tanaka y col. «A multi-label approach using binary relevance and decision trees applied to functional genomics». en. En: *J. Biomed. Inform.* 54 (2015), págs. 85-95.

[41] Grigorios Tsoumakas y Ioannis Katakis. «Multi-Label Classification». En: *Int. J. Data Warehouse. Min.* 3.3 (2007), págs. 1-13.

[42] Grigorios Tsoumakas, Ioannis Katakis e I. Vlahavas. «Random k-Labelsets for Multi-Label Classification». En: *IEEE Trans. Knowl. Data Eng.* 23 (jul. de 2011), págs. 1079-1089. DOI: 10.1109/TKDE.2010.164.

[43] Grigorios Tsoumakas y col. «MULAN: A Java library for multi-label learning». En: *Journal of Machine Learning Research* 12 (jul. de 2011), págs. 2411-2414.

[44] Indika Wickramasinghe y Harsha Kalutarage. «Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation». en. En: *Soft Computing* (sep. de 2020). ISSN: 1433-7479. DOI: 10.1007/s00500-020-05297-6. URL: http://link.springer.com/10.1007/s00500-020-05297-6 (visitado 11-01-2021).

[45] David H. Wolpert. «Stacked generalization». en. En: *Neural Networks* 5.2 (ene. de 1992), págs. 241-259. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80023-1. URL: http://www.sciencedirect.com/science/article/pii/S0893608005800231 (visitado 23-01-2021).

[46] Min-Ling Zhang y Kun Zhang. «Multi-label learning by exploiting label dependency». En: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*. 2010.

[47] Min-Ling Zhang y Zhi-Hua Zhou. «A Review On Multi-Label Learning Algorithms». En: *IEEE Trans. Knowl. Data Eng.* 26 (2014), págs. 1819-1837.

[48] X. Zheng y col. «A Survey on Multi-Label Data Stream Classification». En: *IEEE Access* 8 (2020). Conference Name: IEEE Access, págs. 1249-1275. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2962059.