U N I V E R S I D A D A U T Ó N O M A D E M A D R I D

La SEI
School of Economic Intelligence & International Relations
Escuela de Inteligencia Económica y RRII
Universidad Autónoma de Madrid

Marzo 2023

Volumen 13
ISSN 2660-7352
Editados por la Escuela de Inteligencia Económica y RRII
Editor Jefe / *Chief Editor*: Ángel Rodríguez García-Brazales

REPORTS DE INTELIGENCIA ECONÓMICA Y RELACIONES INTERNACIONALES

Identificación automática de bots en Twitter basada en contexto

Luis Santos Sanz

Desde hace algunos años, las redes sociales se han consolidado como canal para divulgar información, siendo para muchos de sus usuarios medios con los que estar informado de los acontecimientos que les rodean y a la vez, participar activamente de ellos, ya sea directamente o mediante bots. El objetivo del presente estudio es analizar el empleo de algoritmos de clasificación de redes convolucionales basadas en grafos para identificar, de forma automática, si una cuenta de Twitter es un usuario humano, o por lo contrario se trata de un bot.

Universidad Autónoma de Madrid UAM

La SEI
School of Economic Intelligence & International Relations
Escuela de Inteligencia Económica y RRII
Universidad Autónoma de Madrid

Título: *Identificación automática de bots en Twitter basada en contexto*

Autor: Luis Santos Sanz[1]

Volumen nº: 13

Fecha: 20 de marzo de 2023

ISSN 2660-7352

Reports de Inteligencia Económica y Relaciones Internacionales

Editor Jefe: Ángel Rodríguez García-Brazales

Editada por la:

Escuela de Inteligencia Económica y Relaciones Internacionales
Universidad Autónoma de Madrid
Campus de Cantoblanco
C/. Francisco Tomás y Valiente, nº 5, Módulo 10, despacho 303
28049 MADRID (SPAIN)

[1] Contacto: Luis Santos Sanz. Escuela de Inteligencia Económica y Relaciones Internacionales. Universidad Autónoma de Madrid. E-mail: santos.sanz.luis@gmail.com

Contenidos

Resumen

Desde hace algunos años, las redes sociales se han consolidado como canal para divulgar información, siendo para muchos de sus usuarios medios con los que estar informado de los acontecimientos que les rodean y a la vez, participar activamente de ellos. El impacto que generan en la vida cotidiana es tan grande que llegan a influir sobre las opiniones y tendencias de las personas, considerándose herramientas útiles que permiten la interacción con otros usuarios y disponen la capacidad para emitir contenidos a un gran número de receptores de forma inmediata. Sin embargo, este potencial puede emplearse de forma malintencionada para tergiversar y manipular la información y así la opinión de quienes la consumen, divulgando información falsa, rumores, o incluso promoviendo campañas de desinformación a gran escala alterando la opinión pública de la sociedad. Uno de los pilares tecnológicos fundamentales que permiten esta difusión a gran escala son los denominados bots, que se definen como cuentas automatizadas creadas y/o controladas por botmasters. Desde el punto de vista de la seguridad estas noticias falsas, se extienden rápidamente por la red y no solo aplican en la intoxicación de la información, si no que pueden ser la puerta de entrada de ataques como *spam*, *phishing* o incluso *malware,* haciendo indispensable disponer de medios para su identificación.

El presente estudio, "Identificación automática de bots en Twitter basada en contexto", se centrará en Twitter, una plataforma de microblogging online que permite a sus usuarios comunicación directa a través de publicaciones de un máximo de 280 caracteres denominados tweets. El objetivo que se persigue es analizar el empleo de algoritmos de clasificación para identificar, de forma automática si una cuenta de Twitter es un usuario humano, o por lo contrario se trata de un bot.

Con este fin, se ha examinado la posibilidad de utilizar redes convolucionales basadas en grafos, analizando su uso en contexto. Para ello se han generado grafos basados en las relaciones de los perfiles de Twitter, como las de seguimiento (seguidores/seguidos), o en base a los retweets o replies generados. Estos grafos han sido utilizados como entrada de algoritmos de aprendizaje automático que fuesen capaces de diferenciar bots de perfiles legítimos utilizando la estructura de dicha red, además de la información propia de cada perfil. Siendo el contexto de interés la denominada inteligencia de fuentes abiertas (OSINT), por lo que se ha trabajado con datos disponibles públicamente en Twitter, analizando desde las herramientas existentes para la extracción de datos hasta el detalle de la información disponible en cada caso.

Tras el proceso, se puede concluir en última instancia que, los clasificadores que combinan los atributos de nodo y las relaciones de los usuarios expuestas en grafos aportan mayor precisión al inducir datos que no han sido vistos durante el entrenamiento, que los modelos que no los contemplan.

Summary

In the past few years, social media has been consolidated as a channel to spread information, being for many of its users a means to be informed of the events that surround them and at the same time, to actively participate in them. The impact they have on daily life is so great that they influence people's opinions and trends and are considered useful tools that allow interaction with other users and have the capacity to broadcast content to a large number of recipients instantly. However, this potential can be used maliciously to distort and manipulate information and thus the opinion of those who consume it, spreading false information, rumors, or even promoting large-scale disinformation campaigns, altering the public opinion of society. One of the fundamental technological pillars that enable this large-scale spread of information are the so-called bots, which are defined as automated accounts created and/or controlled by botmasters. From the point of view of security, this fake news spread quickly through the network and not only apply in the intoxication of information but can also be the gateway for attacks such as spam, phishing or even malware. This makes it essential to have the means to identify them.

This study, "Automatic identification of bots on Twitter based on context", will focus on Twitter, an online microblogging platform that allows its users to communicate directly through posts of a maximum of 280 characters called tweets. It aims to analyze the use of classification algorithms to identify whether a Twitter account is a human user or a bot.

As a means to achieving this, the possibility of using graph-based convolutional networks has been examined, analyzing their use in context. To do this, graphs have been generated based on Twitter profile relationships, such as follower/followed, or based on the retweets and replies generated. These graphs have been used as input for machine learning algorithms capable of differentiating bots from legitimate profiles using the structure of the network, in addition to the information of each profile. As the so-called Open-Source Intelligence (OSINT) is the focus point, we have worked with publicly available data on Twitter, analyzing from the existing tools, from the data extraction to the detail of the information available in each case.

After the process, it can be ultimately concluded that classifiers that combine node attributes and user relationships exposed in graphs, provide higher accuracy by inducing data that has not been seen during training than models that do not contemplate them.

Palabras clave: Redes Sociales, Twitter, bot, campañas de desinformación, aprendizaje automático, grafos, GraphSAGE.

Key words: Social Networks, Twitter, Bot, influence campaigns, machine learning, graphs, GraphSAGE

1. Introducción

Desde hace algunos años, las Redes Sociales (RRSS) se han consolidado como canal para divulgar información, extendiéndose por momentos a más sectores de la sociedad, siendo para muchos usuarios un medio con el que estar informado de los acontecimientos que les rodean y a la vez participar activamente en la difusión de las noticias o contenidos que puedan ser de su interés. Sin embargo, este potencial también puede emplearse de forma malintencionada para tergiversar y manipular la información y así la opinión de quienes la consumen, divulgando información falsa, rumores, o incluso promoviendo campañas de desinformación a gran escala alterando la opinión pública de la sociedad. Uno de los pilares tecnológicos fundamentales que permiten esta difusión a gran escala son los denominados bots, que se definen como cuentas automatizadas creadas y/o controladas por botmasters empleados para facilitar este tipo fines.

Desde el punto de vista de la seguridad, estas *fake news* o noticias falsas, también llamadas bulos, se extienden de manera muy rápida por la red y no solo aplican en la intoxicación de la información, si no que pueden ser la puerta de entrada de ataques como *spam*, *phishing* o incluso *malware* [1]. El foco de este proyecto se pondrá en Twitter, una plataforma de microblogging online que permite a sus usuarios comunicación directa a través de publicaciones de un máximo de 280 caracteres denominados tweets. Cuyo objetivo es explorar las posibilidades en la construcción de clasificadores de aprendizaje automático que consideren el contexto de los nodos en la red social. Llegando a determinar si un perfil de Twitter es un usuario humano, o por el contrario se trata de una cuenta gestionada por un bot.

En el presente estudio se quiere prestar especial atención a las relaciones entre los usuarios y la conceptualización de Twitter como una gran red, donde los usuarios son nodos, con características particulares, que se relacionan estrechamente entre si mediante interacciones. Por ello, y con los medios disponibles, se quieren buscar las mejores opciones para generar grafos que las puedan representar.

Así las cosas, el proceso abarca desde la recopilación de información de la red social, hasta el análisis de modelos de aprendizaje automático disponibles que puedan aprovechar el contexto de la red y entrenamiento de estos. Modelos que deberán poder aplicar el concepto de Redes Neuronales en Grafos (GNN) [2].

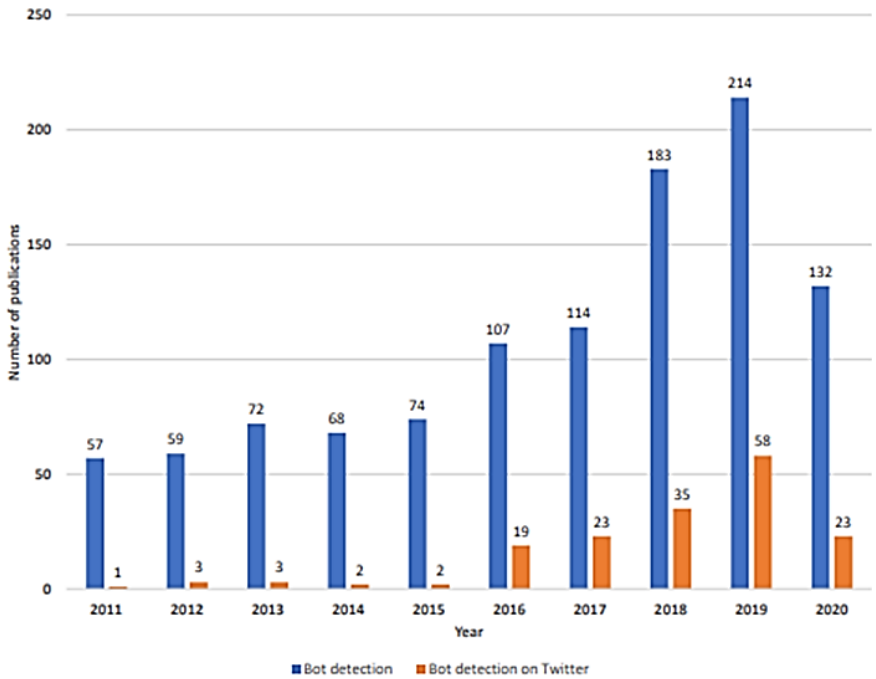
2. Bots en Twitter

En los últimos años, las redes sociales se han consolidado como canal para generar y difundir contenidos, extendiéndose por momentos a más sectores de la sociedad y abanicos mayores en términos de edad de sus usuarios. Para muchos de ellos, son un medio con el que estar informados de los acontecimientos que les rodean y a la vez participar activamente en la difusión de las noticias o contenidos que puedan ser de su interés. Siendo tal la relevancia que fue en el año 2009 cuando se publicó en Twitter una noticia antes que apareciera en los medios de comunicación ordinarios, marcando así tendencias que se mantienen hasta ahora.

El impacto generado por las RRSS en la vida cotidiana es enorme y puede influir de diversas maneras sobre las opiniones y tendencias de quienes las emplean. Son herramientas útiles que permiten la interacción con otros usuarios y poseen la capacidad de emitir contenidos a un gran número de receptores de forma rápida. Pero estas difusiones se consideran de baja calidad cuando lo que transmiten pueden ser rumores o simplemente información falsa. Es decir, se pueden aprovechar estas capacidades de manera mal intencionada para tergiversar y manipular la información y así la opinión de quienes la consumen promoviendo campañas de desinformación a gran escala. Uno de los pilares tecnológicos que permiten esta difusión a gran escala son los denominados bots, que utilizan las RRSS para este tipo de fines [3]. Se reconocen grandes campañas de bots aplicando sobre elecciones políticas para manipular a los electores a lo largo de todo el mundo, como en las elecciones US 2016 [4] o en el caso de España en 2019 [5].

Desde el punto de vista de la seguridad, estas noticias falsas o bulos, se extienden de forma muy rápida por la red y no solo se aplican en la intoxicación de la información, si no que pueden ser la puerta de entrada de ataques como *spam*, *phishing* o incluso *malware*. Por ello, se hace indispensable disponer de medios para la identificación temprana de este tipo de cuentas evitando que difundan información y puedan provocar cualquier efecto sobre los usuarios. En la propia plataforma se estima que 48 millones de cuentas (es decir, un 8,5% de las cuentas activas) son en realidad bots [6], aunque en otros estudios se afirma que esta cifra puede ser mayor [7].

Figura 2-1: Volumen de publicaciones para la detección de bots en Twitter en los últimos años.
(Fuente: Imagen publicada en [6], sección 1)



La intromisión en las elecciones de Estados Unidos en 2016, hito que promueve un aumento en las investigaciones para la identificación de cuentas automatizadas bots, también provoca una reacción de Twitter en la que decide cambiar su API (Aplicación de Programación de Aplicaciones) para reducir el impacto de las cuentas bots. Además, al limitar el acceso a su API también consiguieron reducir el número de usuarios activos, según información del Washington Post [8]. En la figura 2-1, obtenida de [6] se puede observar el incremento en los estudios para afrontar la detección de bots en los últimos años.

2.1. Twitter como red social

Twitter fue creada por el equipo de **Jack Dorsey** en 2006 [9] con la finalidad de enviar fragmentos cortos de texto en los que se pueden añadir enlaces, vídeos, imágenes, etc. La longitud máxima inicial de los mismos fue de 140 caracteres hasta el año 2018, a partir del cual se aumentó hasta los 280 actuales. Una de las ventajas que tiene es que es rápido de escribir y las publicaciones son inmediatas, por lo que puede ser usada para compartir vivencias de sus usuarios en el momento.

A continuación, se presentan algunas características generales sobre el empleo de Twitter, según **Statista GmbH** [10], un portal de estadística en línea alemán que permitirá ver la envergadura de la red social.

Figura 2-2: Países con más usuarios de Twitter

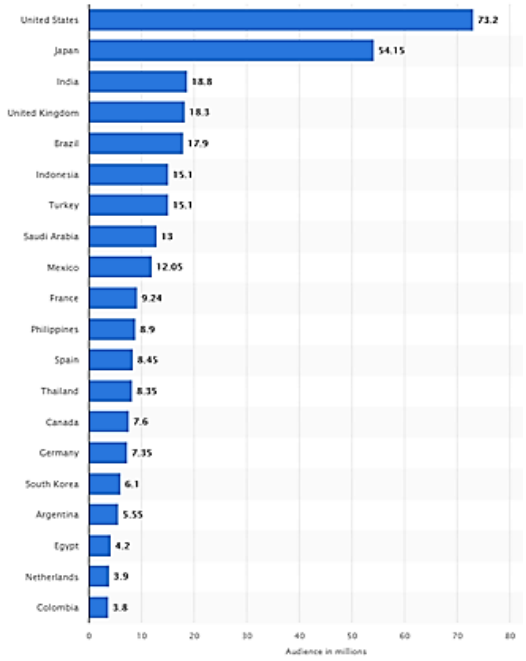
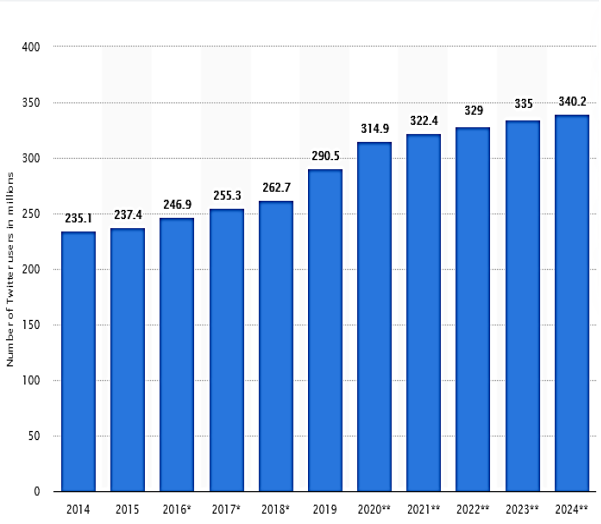


Figura 2-3: Número estimado usuarios de Twitter a nivel global entre 2014 y 2024 (millones)



En la imagen de la izquierda (Figura 2-2) se observan los países que más usuarios de Twitter tienen en millones de usuarios a fecha de abril 2021. En la imagen de la derecha (Figura 2-3) puede verse el número estimado de usuarios a nivel global desde el año 2014 hasta una estimación en 2024.

Otros datos de relevancia son los arrojados por Brandwatch [11] (ver Tabla 2-1), empresa de Inteligencia del consumidor digital.

Tabla 2-1: Estadísticas de Twitter facilitadas por Brandwatch.

DETALLE	VOLUMEN
Usuarios activos por mes	326 millones
Número de cuentas creadas	1.300 millones
Bot activos	23 millones
Número tweets generados diariamente	500 millones
Volumen de usuarios que acceden desde el móvil	80%

En el "*Anexo A"* se expone un pequeño glosario de terminología y posibilidades de la red social para familiarizarse con la misma y entender conceptos que se manejarán a lo largo del documento. Esta información se desglosará en dos partes, una relacionada directamente con el perfil de los usuarios y otro con la terminología y posibles características de los tweets.

2.2. Cuentas automatizadas y Bots

Siguiendo distintos tipos de definiciones, en general, se entiende como bot a un software o programa informático elaborado para llevar a cabo determinados tipos de actividades repetitivas como si de un ser humano se tratase. En el contexto de Twitter se identifica como bot a un software que controla una cuenta a través de la API de dicho servicio. Estos programas pueden generar tweets, retweets, marcar favoritos otros contenidos, así como seguir, dejar de seguir y enviar mensajes directos a otros usuarios, es decir, emular por completo el comportamiento de una cuenta manejada de forma única por humanos.

Sin embargo, aunque se acaba de indicar una definición un poco burda de bot, se debe entrar en más detalles para **separar lo que se entiende como bot malicioso y lo que son meros automatismos de actividades cotidianas que simplifican la labor de los usuarios** (ya que actualmente las RRSS también son empleadas bajo los paraguas de la publicidad, marketing y otras actividades profesionales). Por esto, se debe prestar especial atención a los automatismos que sí permite la plataforma para separar claramente qué se considera un bot malicioso y qué no. A continuación, se presentan las actividades que realmente están prohibidas en la plataforma según Twitter [12]:

- Empleo malicioso de las automatizaciones para interrumpir conversaciones de los usuarios con la pretensión de hacer publicidades o hacer del tema tendencia.
- Tratar de generar impulso y amplificación artificial en cualquier conversación.
- Completar cualquier movimiento de generación de contenidos e interacción de forma masiva o exagerada.
- Cualquier actividad con fines de *spam*, así como el denominado *hashtag cramming* (empleando hashtags que no tienen relación con el tweet).

Se menciona de nuevo que no todos los tipos de automatización vulnerarán los servicios de Twitter ni se considerarán maliciosos.

2.3. Enfoques en la detección de Bots

Las bases de la investigación para la identificación y clasificación de este tipo de cuentas se centran en el concepto de que los humanos y los bots presentan detalles y comportamientos diferentes. De esta forma se espera poder distinguir si una cuenta es un bot, independientemente de lo sofisticado que éste sea [13]. Es importante entender que estas técnicas de clasificación deben estar en constante evolución, ya que los bots también evolucionan como se ha descrito en [14].

De forma general, las aproximaciones empleadas para esta clasificación en Twitter utilizan modelos de aprendizaje automático con dos enfoques diferenciados en su estudio. Uno centra el foco en el **contenido generado** por la cuenta, y el otro en el **contexto social** de la misma.

2.3.1 Basados en contenido

Este tipo de enfoques son considerablemente usuales como punto de partida para estudios y clasificaciones en redes sociales, apoyándose en los contenidos generados por sus usuarios. En general estas tendencias se apoyan en PLN [15] (Procesamiento del Lenguaje Natural) dadas sus capacidades para extraer información del texto.

Estas técnicas basan su estudio en las diferencias existentes entre los tweets publicados por humanos y los que publican cuentas automatizadas en aspectos como longitud, contenido, estilo y variedad temática, así como el desarrollo de distintos patrones en la escritura. Exponiendo ejemplos, algunos estudios relacionados emplean PLN para realizar análisis de sentimientos en conjuntos de datos; y otros aplican sobre la predictibilidad del contenido generado por una cuenta determinada como en [16], entre muchos otros.

2.3.2 Basado en contexto social

A diferencia de los basados en contenidos, este tipo de enfoque, métodos y técnicas basan su estudio en las características propias de las cuentas de usuario. Muchas de ellas pueden extraerse a través de sus perfiles, como son el número de seguidores, conjunto de cuentas a las que sigue, volumen de tweets generados, etc.

Otro objetivo que presenta esta corriente es entender a los usuarios como puntos dentro de un gran grafo o red. De forma que los usuarios serán los nodos de la misma y se prestará atención a los detalles y actividades que los relacionan, como la publicación de retweets, menciones a otros usuarios, replies de tweets, etc. Se apoyan, por tanto, en la distinta actividad social que completan los bot, y qué los distingue de los usuarios humanos. Además, la sofisticación continua de las cuentas automatizadas requiere extraer información para evaluar las dinámicas de las interacciones sociales y no solo de características individuales como se sugiere en [17]. Será este enfoque el que se quiere emplear en el presente estudio.

2.4. Herramientas de clasificación rápida de cuentas de Twitter

Aunque estos programas evalúan multitud de parámetros y tienen una fiabilidad muy amplia en su clasificación, desde el propio Twitter reseñan que su enfoque puede ser limitado. Esto se debe a que suelen emplear la información pública de la cuenta que puede verse en Twitter para identificar características como el nombre, niveles de interacción del servicio, ubicación, descripciones, detalles del perfil, aspectos sobre el contenido generado, etc. Por tanto, esto podría no ser del todo eficaz en la clasificación, ya que no toma en cuenta las tendencias sociales del momento para comparar el comportamiento anterior de los perfiles a clasificar. Dos herramientas online de libre disposición serían las siguientes:

- **BOTOMETER** [18] Desarrollada por el Instituto de Ciencias de la Red [19] y el Centro de Investigación de Redes y Sistemas Complejos [20] de la Universidad de Indiana de Estados Unidos, esta herramienta nace para combatir los bots en Twitter y la desinformación y riesgos que provocan. Se trata de un servicio basado en aprendizaje que permite la rápida clasificación de una cuenta aportando su identificador o nombre, y retorna su clasificación. Botometer ha sido empleado como herramienta en sí y como objeto de estudio en distintas investigaciones [21].

Esta herramienta también permite determinar si los seguidores de una cuenta concreta son bots, ya que se ha demostrado que éstos tienen a agruparse. El algoritmo emplea los datos disponibles en Twtitter para evaluar más de 1200 características en su clasificación, como sus relaciones sociales, menciones, hashtag, línea temporal de actividad generada, etc., incluyendo distintas métricas sobre la red.

A lo largo del presente proyecto se ha empleado Botometer para extraer características en 3 conjuntos de datos diferentes. En el *Anexo B* se expone un ejemplo de uso, donde se detallan los requisitos para poderlo emplear, así como los resultados devueltos por la herramienta.

- **BOT SENTINEL** [22], es una plataforma de acceso libre elaborada para identificar y rastrear cuentas bot en Twitter. Como Botometer, se apoya en el aprendizaje automático para evaluar cuentas como confiables o no confiables agregando datos y resultados a un repositorio disponible públicamente para consulta. Además, al clasificar este tipo de cuentas, dichas herramientas también colaboran arrojando datos sobre la credibilidad de la información difundida por los usuarios en la red, ayudando a combatir la desinformación y la expansión de bulos.

2.5. Aprendizaje Automático en Grafos

Vivimos en un mundo completamente conectado, en el que se generan constantemente cantidades enormes de información bajo cualquier ámbito, desde el transporte, hasta las comunicaciones, el ocio o las redes sociales. Todo forma una malla en la que sus elementos están conectados mediante diversos vínculos. Al fijarse en este paradigma parece inmediato pensar que toda esta información puede representarse con estructuras de datos en forma de grafo de forma natural y flexible.

En el campo del aprendizaje automático, uno de los desafíos que ha limitado que algoritmos como las Redes Neuronales y Redes Neuronales Profundas adopten los grafos en problemas productivos es la dificultad de escalado a grandes conjuntos de datos.

Con los avances en las redes neuronales convolucionales (CNN) se completaron esfuerzos para adaptar este modelo de aprendizaje profundo para codificar estructuras de grafos disponiendo dos enfoques diferentes sobre cómo aplicar la operación de convolución en base a si el dominio de trabajo es o no euclidiano.

Estos Graph Convolutional Network (GCN) [23], consideran convoluciones espectrales, y han evolucionado en los últimos años para aprender sobre datos estructurados en grafos. Se han podido aplicar en diferentes campos, aunque es cierto que hasta el momento la investigación se ha centrado en generar nuevos algoritmos y probarlos en conjuntos de datos bastante reducidos, tradicionalmente redes de citas como **Cora**, **PubMed** o **CoauthorsCS.** No invirtiendo, en general, esfuerzos para enfocar estos algoritmos a gran escala hasta hace relativamente poco tiempo.

En varias de las primeras investigaciones, arquitecturas como GCN, ChebNet o GAT (Graph Attemption Network) [24] se entrenaron aplicando descenso por gradiente de lote completo, obligando a cargar en memoria la matriz de adyacencia del grafo y atributos de nodo, requiriendo demasiados recursos prácticamente en cualquier campo de estudio.

Uno de los primeros algoritmos que realmente fue novedoso es GraphSAGE [25] ya que, no solo se trata de un algoritmo inductivo, sino que también emplea el muestreo de vecindarios junto con entrenamiento en mini Bach. Este algoritmo, aunque presenta algunos inconvenientes, como se mostrará a lo largo del documento, será uno de los focos del estudio del proyecto.

3. Proyecto

El **objetivo final** del proyecto es analizar la posibilidad de utilizar algoritmos de clasificación de aprendizaje automático basados en grafos para identificar si una cuenta de Twitter es un usuario humano, o por lo contrario, se trata de una cuenta automatizada bot. Para lograrlo, se plantea cubrir los siguientes hitos:

- **Elección dataset de cuentas Twitter previamente etiquetadas** en estudios anteriores. Debido a que los métodos de aprendizaje automático a utilizar en el proyecto se engloban en el área de aprendizaje supervisado, para utilizarlos se necesitan conjuntos de datos (en este caso, perfiles de Twitter) previamente etiquetados. Es decir, cada una de las cuentas debe disponer una etiqueta que la clasifique como humana o bot.

- **Elección de herramientas para la descarga de información.** Será preciso determinar las limitaciones que se puedan encontrar en estos programas para seleccionar los que mejor cubran las necesidades del proyecto. Se utilizarán para completar el proceso de la descarga de información de los conjuntos previamente indicados, nutriendo los identificadores de los usuarios con el resto de los atributos asociados a su perfil y actividad en la red social.

- **Limpieza y Análisis de los datos obtenidos**. Se identificarán los principales atributos para formar el vector de características de cada uno de los usuarios. Dichas características estarán enfocadas en aspectos del perfil de usuario, así como su contexto social, siendo uno de los objetivos parciales el ajuste de dichos atributos para poder generar un grafo que relacione la mayor cantidad de usuarios posibles. Se cubrirán varias métricas como usuarios retuiteados o replies realizadas, aportando información sobre el contexto social y la forma en la que se relacionan.

- **Elección y entrenamiento de los modelos de aprendizaje automático**. Una vez preparados los datos y grafos, se deberán elegir los algoritmos adecuados para entrenar los modelos. Se utilizarán modelos basados en GNN (Graph Neural Network) [26], esperando encontrar en ellos buenos resultados añadiendo información sobre la forma en que se relacionan sus usuarios mediante el grafo.

- **Estudio de los resultados obtenidos.** Tras el entrenamiento y ajuste del modelo se buscará comparar los resultados obtenidos con modelos que no empleen la estructura de grafos en la clasificación. Observando así si se aporta mejoras en la precisión de la clasificación de bot.

A la hora de estudiar la aplicabilidad de los métodos de detección automática, un factor a considerar es el esfuerzo requerido para obtener los datos y entrenar los modelos. En este sentido, se analizó el tiempo necesario para su descarga considerando una infraestructura modesta. Algunos aspectos del marco de trabajo son susceptibles de ser escalados en la medida en la que se dispongan de mayores infraestructuras.

Así mismo, las tareas previas de extracción de datos se enmarcan conceptualmente en el contexto de fuentes de información **OSINT** (*Open Source Intelligence*) que provienen, en este caso concreto, de conjuntos de datos disponibles en línea y redes sociales, gratuitos, públicos y desclasificados. En cuanto a metodología se refiere y dada la naturaleza del estudio, se ha optado por seguir, en general, las líneas marcadas por **CRISP-DM** *(Cross Industry Standard Process for Data Mining)* [27], entre otras metodologías como **SEMMA** *(Sample, Explore, Modify, Model, Assess)* [28].

3.1. Herramientas de soporte

Para el desarrollo del estudio se han empleado diferentes tecnologías y plataformas, pudiendo utilizar indistintamente otras para el mismo propósito se ha realizado en **Python** [29], **Anaconda** [30], **Jupyter Notebook** [31] empleado **Tensorflow** [32] y **Keras** [33]. Aprovechando el completo ecosistema que forman para resolver cualquier problema de aprendizaje automático. Cabe hacer una mención especial a **Stellar Graph** [34], librería de código abierto que contiene una gran cantidad de algoritmos de aprendizaje automático basado en grafos, **Pandas** [35] y **Gephi** [36].

Por otro lado, se presentan las herramientas empleadas para llevar a cabo la descarga de información necesaria de Twitter, entre las que se encuentran **Tweepy** [37], una librería Python con la que acceder a las distintas *APIs* de Twitter, ***API streaming***, empleado para acceder en tiempo real a los tweets públicos, ***API Search*** para el acceso a los últimos 1500 tweets en la última semana y ***API Rest*** que permitirá el acceso a los últimos 3200 tweets. Todas se podrán emplear desde Tweepy, pero sus accesos estarán limitados por ventanas de tiempo [38] (motivo por el que se ha descartado su empleo en el proyecto). También se ha empleado **Twint** [39], una herramienta Python para la descarga de información de tweets y perfiles de usuario sin emplear la API de Twitter, a través del método del *crawling*. La principal ventaja de la herramienta es que puede superar fácilmente los límites de las ventanas de descarga de la API de Twitter, aunque es algo inestable en la recopilación de datos (no siempre generará la misma información, en el *Anexo C* se aportan más detalles sobre las características). Por esto, Twint ha sido la herramienta empleada para el raspado de datos durante el proyecto. No obstante, también se han explorado otras posibilidades como **scweet** [40] que funciona sobre ***Selenium*** [41], paquete empleado para automatizar la interacción del navegador web desde Python (pueden encontrarse más detalles sobre los métodos empleados y la información resultante en el *Anexo D).*

4. Obtención y análisis de datos

Como se ha comentado, en esta fase se quiere trabajar con aprendizaje automático supervisado, por lo que se necesitan conjuntos de datos previamente etiquetados para el estudio. Esto implica que cada uno de los usuarios debe tener su identificador de Twitter y la categoría que lo clasifica en una cuenta humana o bot (ver Tabla 4-1).

Tabla 4-1: Estructura de conjuntos etiquetados

ID	LABEL
Identificador de usuario en Twitter en formato numérico	**BOT / HUMAN**

Es importante destacar la dificultad de encontrar grandes conjuntos de datos que puedan servir para fines académicos y exploratorios [6] ya que la mayoría pertenecen a dataset privados o de difícil acceso. Dicho esto, los dos conjuntos etiquetados empleados durante el desarrollo del proyecto son los siguientes:

1. **Cresci-rtbust-2019**: descargado de un repositorio de **Botometer** [42], se trata de un conjunto de datos con 759 cuentas etiquetadas. Este extracto viene de un estudio [43] donde el conjunto original contaba con todos los retweets llevados a cabo entre el 17 y el 30 de junio de 2018 en lengua italiana. Formaban un total de 9,989,819 retweets, compartidos por 1,446,250 usuarios distintos.

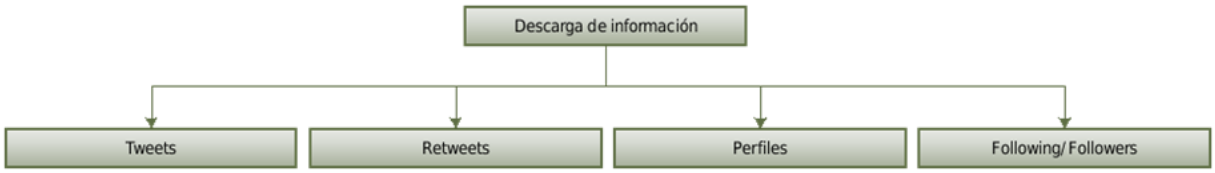
2. **Twitter_human_bots_dataset**: descargado de **Kaggle** [44], este conjunto de datos proviene de un recopilatorio del repositorio de dataset etiquetados de **Botometer**. Su autor no detalla

más información sobre la tipología de las cuentas automatizadas. Cuenta con un total de 37438, aunque sólo se utilizarán parte de ellos.

4.1. Obtención de datos

La información completa relacionada con un perfil de Twitter solo puede obtenerse mediante el pago a esta plataforma, no obstante, con las herramientas descritas anteriormente se pueden conseguir los detalles necesarios para este tipo de estudios (en esta sección se describirán todos los atributos disponibles públicamente). El proceso a seguir se define como **Web Scrapping,** o raspado web, y se trata de una técnica aplicada mediante la extracción de información de plataformas web. Se destaca esta etapa como una de las más importantes, ya que la cantidad de datos recopilados, el tiempo que requiere su obtención, así como la calidad de estos definirán el alcance y posibilidades del resto del proyecto.

Figura 4-1: Estructura descarga conjuntos de usuarios en Twitter



El esquema planteado en la descarga de los conjuntos es el que se ve en la Figura 4.1, para cada usuario se obtendrán los datos de su perfil, línea temporal de tweet generados, retweets y el listado de usuarios seguidos y seguidores. Como se detalla más adelante, no toda la información estará disponible en todos los conjuntos. En la Tabla 4.2 se pueden observar los atributos concretos descargados para cada grupo.

El objetivo inicial es poder descargar la línea temporal de tweets de los usuarios y la lista de sus seguidos y seguidores. Esto no ha sido posible de obtener en todos los conjuntos ya que como se indica en el siguiente foro de discusión [45], Twitter había anunciado el 8 de mayo 2020 el cierre de su versión heredada de su antiguo tema para escritorio el 1 de junio 2020 y el cierre de **Legacy Mobile Twitter el 15 de diciembre de 2020** (M2 mobile web). De manera que sólo se pueden utilizar una serie de navegadores descritos en su Centro de Ayuda [46] limitando las capacidades de Twint, la herramienta elegida para la descarga. Revisando el código de Twint, se ha detectado el uso de métodos de dicha versión "*mobil* no *js"* (ya cerrada) para la obtención de seguidores y seguidos, **por lo que no puede utilizarse para este fin**.

Tabla 4-2: Identificadores obtenidos por ámbito

TWEET Y RETWEET	PERFIL	FOLLOWERS/FOLLOWING
id, conversation_id, created_at ,date, time, timezone, user_id, username, name, place, tweet, language, mentions, urls, photos, replies, count,retweets, count,likes, count, hashtags, cashtags, link, retweet, quote_url, video, thumbnail,near, geo, source, user_rt, id, user_rt, retweet_id, reply_to, retweet_date, translate, trans_src, trans_dest	id, name, username, bio, location, url, join, date, join_time, tweets, following, followers, likes, media, private, verified, profile_image_url, background_image	Listado en String Following y Followers

No obstante, también se investigó la librería **Scweet** que funciona sobre *Selenium* para cubrir la necesidad, aunque el tiempo empleado en la descarga resultó excesivo para extenderlo a grandes conjuntos de datos.

4.1.1. Dataset 1

En esta primera descarga se quiso raspar la línea completa de tweets de los usuarios, llegando a descargar en el caso de los más activos hasta 20.000 tweet. Dado que no se pudo emplear Twint para los seguidores/seguidos se utilizó la librería S**cweet,** no volviéndose a emplear para el resto de las descargas debido al excesivo tiempo que requiere (precisa el empleo de una cuenta registrada con el posible riesgo de baneo asociado). Más adelante, se decidió nutrir este conjunto con los datos de los retweets para poder hacer pruebas más completas. Los elementos descargados en cada caso, así como el tiempo empleado se observan en la *Tabla 4-3*. Es importante notar que los tiempos de descarga descritos corresponden a infraestructura particular sobre la que se ha implementado el análisis. Aunque es posible que estos tiempos no sean trasladables a otros contextos, se considera importante conocerlos para entender la aplicabilidad de los métodos estudiados, así como sus limitaciones.

Tabla 4-3: Detalles descarga conjunto 1 de datos

ÁMBITO	LIBRERÍA	ELEMENTOS DESCARGADOS	TIEMPO ESTIMADO
Tweets	Twint	601	12 h
Perfiles	Twint	646	2.5 h
Seguidores / Seguidos	Scweet	531	63 h
Retweets	Twint	606	5 h
Volumen completo dataset: 531 usuarios			

4.1.2. Dataset 2

Las necesidades del proyecto en ese instante requerían de un conjunto más grande de información para que los algoritmos de aprendizaje automático pudiesen funcionar de manera más precisa. Dado que no se podía disponer de los follows/following para generar los grafos, y tras la lectura de artículos relacionados, se decidió incluir un nuevo conjunto que contuviese retweets y replies realizados por los usuarios. Es decir, en vez de crear una red donde las relaciones entre los nodos (perfiles) sean las relaciones de seguimiento en la red social, los enlaces del grafo representarán acciones de retweets y replies. En la siguiente tabla se encuentran los detalles de los elementos descargados por conjunto y el tiempo que ha llevado. La descarga de los retweets está limitada a 3200 elementos y se redujo la línea temporal de los usuarios a 2400 tweet por perfil (límite diario de publicaciones permitido por Twitter).

Tabla 4-4: Detalles descarga conjunto 2 de datos

ÁMBITO	LIBRERÍA	ELEMENTOS DESCARGADOS	TIEMPO ESTIMADO
Tweets	Twint	13.403	77 h
Perfiles	Twint	15.684	9-10 h
Retweets	Twint	14.672	94 h
Volumen completo dataset: 11.664 usuarios			

4.1.3. Datos extraídos en base a hashtags

Una vez construidos los modelos, se quería medir su eficacia en un entorno real reducido. Para ello se descargaron los tweets que contuviesen el hashtag "#StopVacunas" y con ello identificar así las cuentas que más veces habían publicado con ese hashtag. En la *Figura 4-2* puede verse el circuito seguido para conseguir el conjunto de datos. Se descargan 250 tweet y se toman los 15 usuarios que más veces han empleado dicho hashtag. Con esta información se extrae para cada cuenta el time-line de retweets y replies y se identifican a su vez los 15 usuarios con los que más interactúan (a los que más replies y

retweets han realizado) pudiendo obtener así el nuevo conjunto y grafo de pruebas. Véase Tabla 4.5 para los detalles del conjunto.

Figura 4-2: Flujo descarga y etiquetado de usuarios con Botometer.

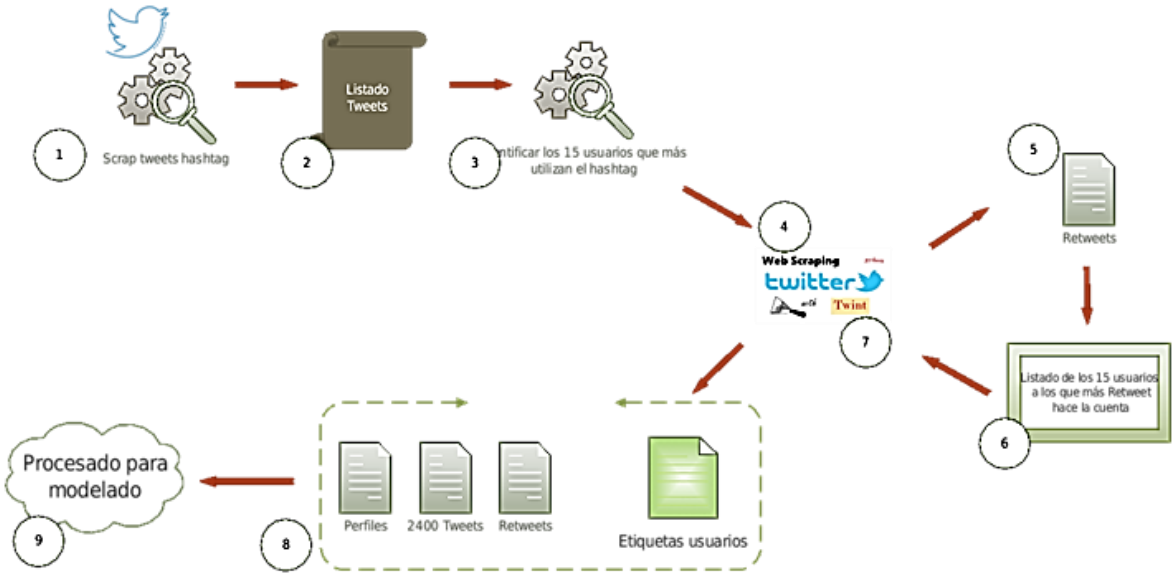
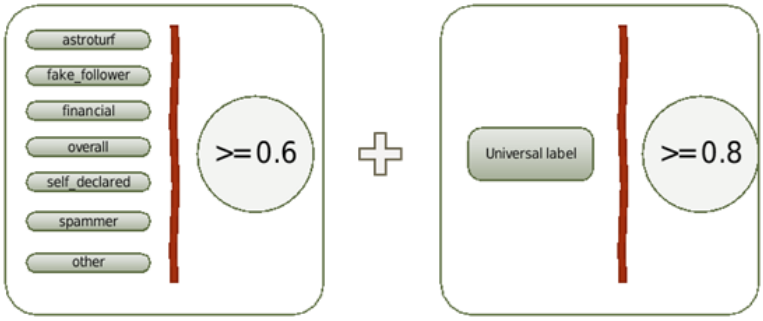


Tabla 4-5: Detalles descarga conjunto de datos sobre hashtag.

ÁMBITO	LIBRERÍA	ELEMENTOS DESCARGADOS
Tweets	Twint	145
Perfiles	Twint	146
Retweets	Twint	144
Etiquetas	Botometer	154
Volumen completo dataset: 126 usuarios		

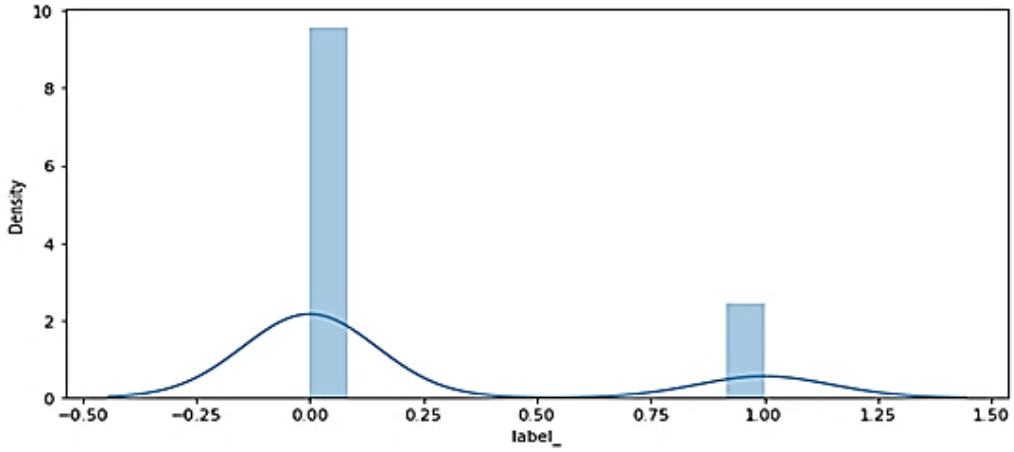
Por otra parte, al emplear aprendizaje automático supervisado, también deben etiquetarse estos nuevos usuarios. Para ello, Botometer ofrece un servicio que retorna para cada perfil una clasificación que lo referencia con diferentes tipos de bots y una clasificación universal. Con estos detalles sólo se considerarán cuentas automatizadas bots aquellas que dispongan un valor mayor a 0.6 en cualquiera de las clasificaciones y más de 0.8 en la etiqueta universal (ver figura 4.3). En el *Anexo B* se describe un ejemplo y se detalla cada categoría retornada por Botometer.

Figura 4-3: Etiquetado usuarios



En la figura 4.4 puede verse el volumen de usuarios bot frente a humanos etiquetados en este conjunto. Son un total de 137 cuentas útiles formadas por 30 bots y 107 usuarios humanos.

Figura 4-4: Distribución usuarios legítimos (0) y bots (1).



4.2. Construcción de grafos de usuarios en base a distintos contextos

Un grafo, es una estructura de datos que presenta dos elementos básicos, **nodos y bordes,** de manera que los nodos representan entidades en los datos, en este caso, usuarios de Twitter. Mientras que los **bordes** representan las relaciones existentes entre dichas entidades, pudiendo denotarlo como G = (V, E) donde **v** es un conjunto de nodos y **e** el de aristas. Como se ha mencionado a lo largo del documento las relaciones se basarán en: ***Follows/Following, Replies y Retweets***.

Los elementos de un grafo se pueden clasificar siguiendo varias pautas. Si todos los tipos de nodo del grafo son similares y solo existe un tipo de orden, se tratará de un grafo **homogéneo,** como en este caso ya que la información de los nodos y los bordes será siempre la misma (aunque cada uno de los nodos del grafo pueda presentar atributos propios que lo diferencian del resto). Además, la relaciones que se producen entre los usuarios son dirigidas, es un usuario quién genera una actividad sobre otro y no al revés, de forma que los grafos a utilizar en el proyecto serán también **dirigidos.**

La naturaleza de las redes sociales es cambiante, y por tanto también debería serlo el grafo, aunque para este proyecto será estático, como una foto tomada en un instante concreto de tiempo y evaluada con las características que presente. En general, los **problemas** que se pueden resolver con los grafos son [47]:

- **Clasificación de los nodos** o inferencia de atributos de nodo, **que será el caso a tratar.**
- **Predicción de enlaces** o inferir relaciones perdidas u ocultas.
- **Detección de comunidades** o inferencia de grupos de nodos en base a la estructura de la red y relaciones existentes entre sus nodos.
- **Clasificación gráfica**, o discriminación entre grafos de diferentes clases.

Al extraer la información para preparar los grafos se han formado redes con muchísimas relaciones, en el caso del conjunto 2 llegando a varias decenas de millones. No obstante, estos datos deben filtrarse para que solo aparezcan aquellos usuarios que disponen etiqueta e información descargada.

En el caso de los following/followers parece evidente tomar el listado de cada uno de los usuarios para concatenarlo y filtrarlo, pero en el caso de los replies y retweets hay que definir una estrategia. Para este proyecto, se han contemplado los retweets y replies realizados por la cuenta a examinar, ya que se trata del nodo que genera la acción bajo estudio y se dispone de toda la información relevante al usuario. Se descartan así los que le hayan realizado a él, ya que no se dispone de los identificadores de

estos usuarios en el conjunto de datos y requeriría realizar otras descargas sobre estos nuevos perfiles objetivo. En el dataset, se itera sobre cada uno de los datos de los usuarios extrayendo los identificadores que cumplan los requisitos:

- **@Retweet:** Twint genera una columna "retweet" cuyo valor será verdadero si esa acción es un retweet. En ese caso se podrá capturar el identificador de la columna *user_rt_id*. Debido a los problemas para manejar el formato de los identificadores también se ha extraído del campo "tweet" el nombre del usuario al que se retwitea, siempre precedido de *"RT @Name:…"* antes del cuerpo del tweet. Se relacionarán el nombre e identificador obtenidos para asegurar que todo es correcto.
- **@Reply:** En este caso se extrae la información de la columna *reply_to* ya que si se trata de un reply vendrán identificados en formato simple JSON los valores [{*'screeen_name': 'screen_name',' name': 'name', 'id':'id'*}...{}] del usuario o usuarios sobre los que se completa esta acción.

En la Tabla 4.6. se muestra el formato de los grafos, en el que se incluye el nodo origen "*source*", el nodo destino "*target*" y la etiqueta "weight" que representa el número de veces que se repite el enlace. Es decir, el número de veces que un usuario ha realizado un retweet o replie sobre otro.

Tabla 4-6: Formato de los grafos.

SOURCE	TARGET	WEIGHT
id	**id**	**int**

Seguidamente, la tabla 4-7 muestra todos los detalles de los grafos generados, así como las estadísticas obtenidas con la herramienta Gephi.

Tabla 4-7: Detalles de los grafos generados.

ESTADÍSTICAS	CONJUNTO 1			CONJUNTO 2	
	FOLLOWS	RETWEETS	REPLIES	RETWEETS	REPLIES
Ratio humanos / bot	33.33 / 60.73	47.42 / 52.58	39.27 / 60.73	79.9 / 20.1	90.9 / 9.1
Nodos	348	523	354	10340	7220
Enlaces	8535	19080	1590	106294	43144
Grado	24.5	36.48	4.49	10.28	5.9
Modularidad	0.12	0.15	0.40	0.68	0.6
Número de comunidades	19	28	21	328	232
Nodos poco conectados	15	24	15	300	202
Nodos fuertemente conectados	131	298	215	8005	5062
Diámetro de la Red	10	16	10	22	26
Longitud media de camino	1.98	2.07	3.85	5.65	6.21

En las siguientes figuras puede explorarse visualmente el aspecto de los conjuntos, se marcan en color verde los usuarios legítimos y en color rosa las cuentas bots. Para la disposición de los grafos se ha empleado *ForceAtlas2*, un algoritmo de vector de fuerza que agrupa los nodos de manera que los que tienen más conexiones entre ellos estarán más próximos en el grafo y los que tengan menos aparecerán más alejados. Puede observarse de esta forma que los bots tienen cierta tendencia para agruparse entre sí (ver figuras 4-5, 4-6 y 4-7 referentes al Dataset 1, y las 4-8 y 4-9 al Dataset 2). Adicionalmente se han combinado ambos grafos en cada conjunto y se han visualizado de la misma forma con Gephi, se pueden encontrar en el *Anexo F*.

Figura 4-5: Grafo Dataset 1 Follows.

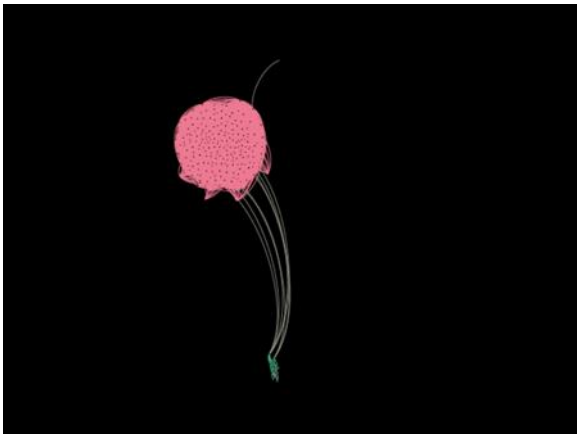


Figura 4-6: Grafo Dataset 1 Replies.

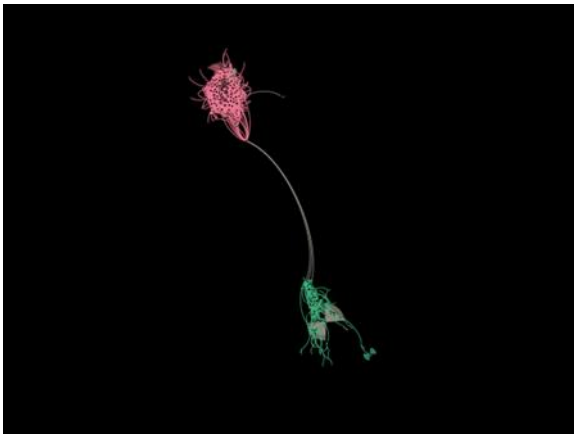


Figura 4-7: Grafo Dataset 1 Retweets.

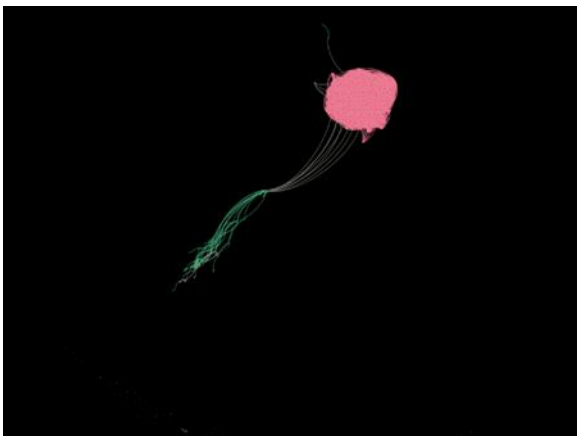


Figura 4-8: Grafo Dataset 2 Retweets.

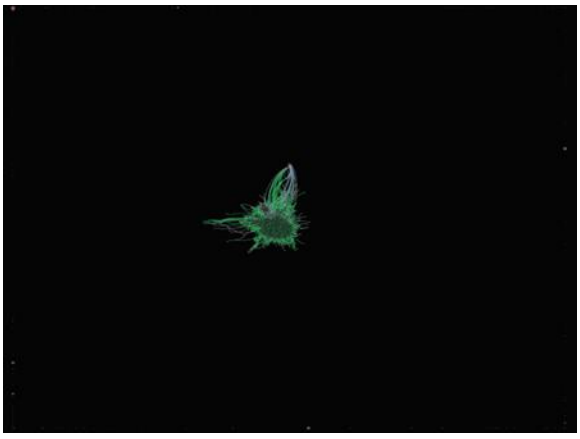
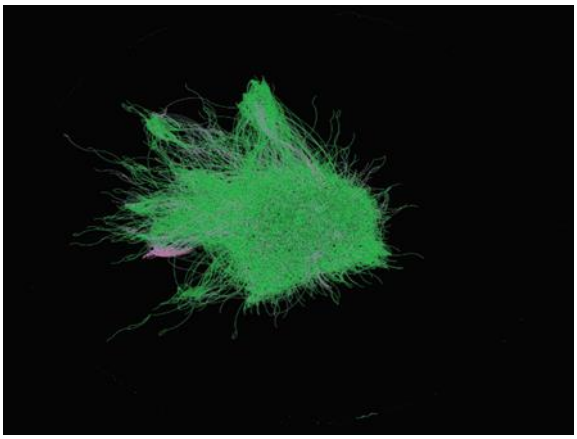


Figura 4-9: Grafo Dataset 2 Replies



4.3. Atributos de usuarios

Clasificar cuentas en Twitter con aprendizaje automático consiste en estudiar y evaluar atributos de los usuarios que son clave para diferenciarlos entre sí. Se trata de una tarea delicada que debe llevarse a cabo de manera precisa e iterativa para conseguir los mejores resultados. De la lectura de distintos autores en literatura previa [7] [48] [49] [50] se toman una serie de atributos e ideas bastante significativos que podrían agruparse de múltiples formas atendiendo a la naturaleza de los datos, por ejemplo:

- **Características de red,** que recogen detalles en la difusión de información y han sido incorporadas mediante los grafos descritos en la *sección 4-2*.

- De los propios **metadatos** de las cuentas de Twitter:

    - **Perfil verificado en Twitter,** en general las cuentas bots no suelen disponer de la verificación.
    - **Tiempo desde la creación de la cuenta,** habitualmente las cuentas de los bots son más recientes, tienden a borrarse para crear otros o simplemente son detectadas y eliminadas. Además, estas cuentas tienen objetivos concretos que suelen variar con el paso del tiempo y evolución tecnológica.
    - **Proporción seguidos/seguidores.** Las características derivadas de estos atributos también podrían agruparse en otra categoría propia relacionada con los contactos sociales de un perfil. No obstante, actualmente es difícil determinar la ratio estándar que pueden presentar las cuentas legítimas, ya que se han popularizado las cuentas de influencia y de generación de contenidos. Sin embargo, es habitual que una cuenta bot siga a muchos más usuarios de los que le siguen a él.

- O directamente relacionados con los momentos elegidos en las **publicaciones y consumos de contenidos**, destacando:

    - **Volumen, frecuencia y horario de publicaciones.** En cuanto al volumen será posible encontrar cuentas legítimas que publiquen muchos tweets y presenten gran actividad y bots que publiquen poco, pero en general, la capacidad de publicación de las cuentas automatizadas es muy superior a la de los humanos. Además, muchos bots suelen tener programadas unas horas fijas de publicación mientras que los humanos no, como se ejemplifica en las *figuras 4-11 y 4-12*.
    - **Tiempo entre actividad.** Las cuentas humanas retuitean contenidos cuando les resultan curiosos o interesantes, mientras que existen bots que retuitean contenidos generados por cuentas objetivo. De esta forma cuando un bot detecta que la cuenta objetivo ha publicado un tweet, lo retuitea automáticamente en tiempos casi instantáneos, mucho más reducidos en comparación con el de los seres humanos.
    - **Proporción retweet/tweet**. Las cuentas bots son más propensas a reaccionar al contenido generado por otros usuarios antes que a la generación de contenido propio.

- Por otro lado, y aunque no corresponden directamente a los objetivos del estudio, se han valorado e incluido algunas métricas sencillas relacionadas con el **contenido generado** por las cuentas:

    - **Inclusión de enlaces a la web u otras redes sociales.** Es habitual entre los bots tener enlaces externos en sus publicaciones incluyendo contenidos de *spam*.
    - **Léxico empleado.** El lenguaje utilizado por los bots suele ser más pobre, empleando menos palabras que los humanos y con métricas más repetitivas.
    - **Longitud y número de palabras.** En su mayoría, son tweets mucho más cortos los generados por cuentas automáticas en comparación a los humanos.

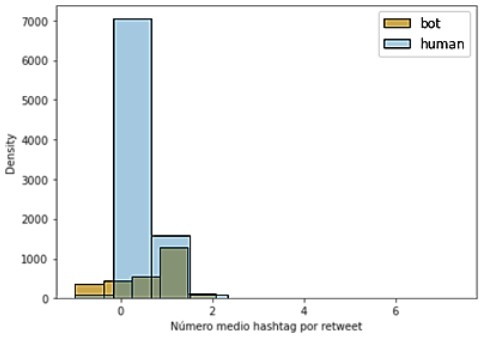
4.3.1. Vector de características desarrollado

En esta sección sólo se explicará el proceso seguido para el Dataset 2, el más grande, ya que los demás tienen las mismas métricas para las pruebas asociadas. Una vez capturados y agrupados los datos, se generan una serie de características para cada uno de los grupos (perfiles, tweet y retweet) apoyándose en la información descrita en el punto anterior.

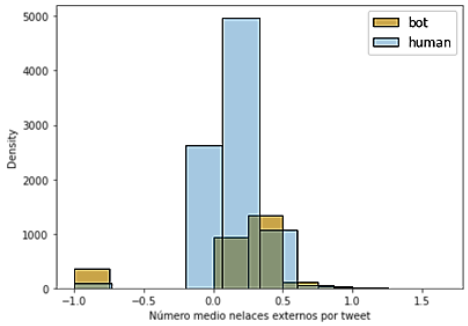
El objetivo será encontrar aquellos atributos que sean diferentes en las cuentas manejadas por los humanos de las cuentas automatizadas bots. En las siguientes imágenes (ver Figura 4-10) se exponen algunos histogramas de los datos para tomar una idea de los datos manejados. Se presenta el número medio de hashtags por retweet, número medio de urls incluidos en la publicación de tweets, ratio de retweets encontrados en el conjunto de retweets por usuario o las publicaciones en una franja de tiempo determinada (de 10.00 a 12.00 de la mañana). En azul se describe el comportamiento de las cuentas de humanos y en color marrón las de los bots:

Figura 4-10: Histogramas varios atributos Dataset 2.

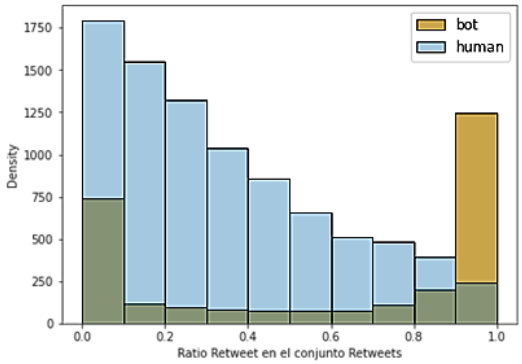
A: Número medio de Hastag por retweet



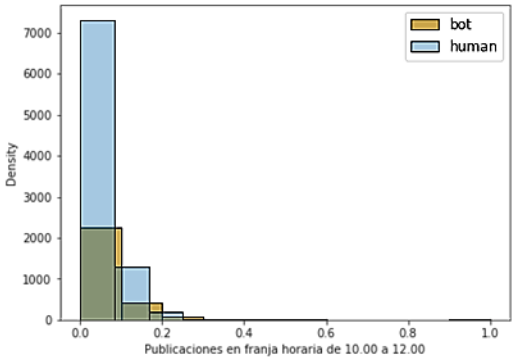
B: Número medio de enlaces por tweet



C: Ratio de retweet en el conjunto de retweet



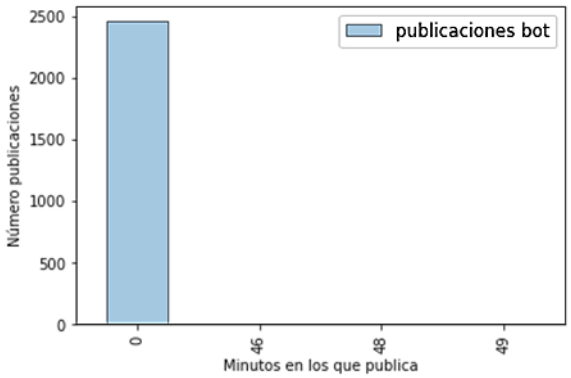
D: Publicaciones en franja horaria de 10,00 a 12,00



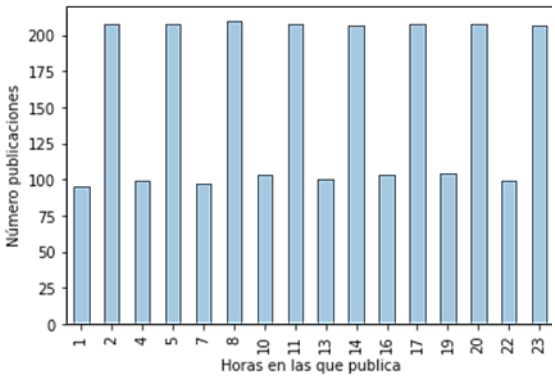
Seguidamente se muestra uno de los ejemplos más claros encontrados en la exploración manual de los datos, referenciado con las horas y volumen de publicación de las cuentas. En la Figura 4-11 se presentan las publicaciones completadas por el bot con identificador *787405734442958848,* reflejándose cómo publica en las horas en punto, en los 3-4 primeros segundos de cada hora y sigue una tasa de 100 publicaciones en las horas impares y 200 en las horas pares.

Figura 4-21: Histogramas de minutos y horas en las que publica un bot

A: Minutos



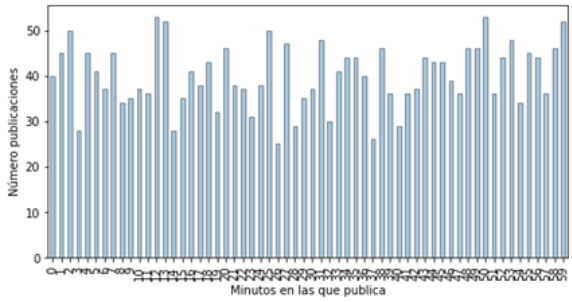
B: Horas



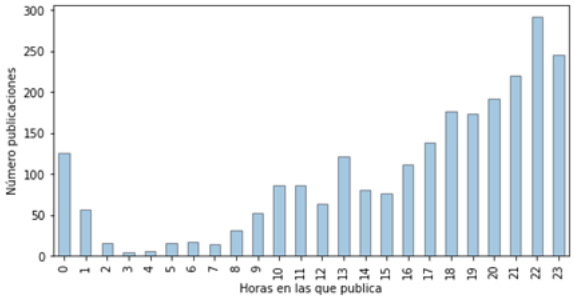
En contrapartida las publicaciones de los humanos suelen ser más aleatorias, y encontrar mucha menos actividad en las horas de la noche, como en el siguiente ejemplo (identificador *464781334*) de la Figura 4-12:

Figura 4-12: Histogramas de minutos y horas en las que publica un humano

A: Minutos



B: Horas



Finalmente se expone el conjunto de atributos empleados para la clasificación de los modelos, que consta de un total de 55 características, incluida la de etiqueta, descrito en la tabla 4-8, y se exponen el resto de los detalles, como la matriz de confusión, importancia de las variables, tipos más relevantes de las mismas o transformaciones aplicadas en el *Anexo E*.

Tabla 4-8: Atributos del vector de características de cada cuenta.

PERFILES	TWEETS	RETWEETS
id	id	id
pr_following	tw_longitud_bruta_media_tweet	rt_ratio
pr_followers	tw_replies_count_media	rt_time_to_rt_mean
pr_likes	tw_retweets_count_media	rt_time_to_rt_desv_tip
pr_media	tw_likes_count_media	rt_time_to_rt_var
pr_real_name	tw_replies_count_desv_tip	rt_time_to_rt_desv_norm
pr_join_date	tw_retweets_count_desv_tip	rt_retweets_count_media
pr_age_account	tw_likes_count_desv_tip	rt_retweets_count_desv_tip
pr_len_name	tw_replies_count_var	rt_retweets_count_des_med
pr_len_username	tw_retweets_count_var	rt_num_medio_urls
pr_len_bio	tw_likes_count_var	rt_num_medio_hashtags
pr_len_url	tw_replies_count_desv_med	rt_num_medio_mentions
pr_location	tw_retweets_count_des_med	rt_num_med_videos_por_tweet
pr_tff_ratio	tw_likes_count_desv_med	
	tw_num_medio_hashtags_por_tweet	
	tw_num_medio_mentions_por_tweet	
	tw_hora_1, tw_hora_2, tw_hora_3, tw_hora_4, tw_hora_5, tw_hora_6	
	tw_hora_7, tw_hora_8, tw_hora_9, tw_hora_10, tw_hora_11, tw_hora_12	
	tw_med_num_enlaces_por_tweet	
	tw_longitud_media_tweet	
	tw_n_palabras_med_tweet	

5. Modelos empleados

A lo largo de esta sección se describirán los modelos analizados y entrenados en la construcción de los clasificadores Desde el inicio del proyecto, el objetivo ha sido estudiar las posibilidades en el aprendizaje automático que permitan abordar la información albergada tanto en los atributos de las cuentas de los usuarios de la red social, como los detalles extraíbles de sus propias relaciones. Para ello se ha empleado **aprendizaje automático supervisado,** es decir, las cuentas con todos sus atributos deben estar identificados por una etiqueta que será la que debe predecir el modelo. La etiqueta estará presente en los datos de entrenamiento, de manera que se pueda crear una función capaz de predecir el valor de la misma para cualquier objeto de entrada al modelo tras haber realizado varios ejemplos durante el proceso. Los algoritmos se entrenan con un histórico de información del que aprenden para asignar la etiqueta de salida a nuevos valores. En este caso el tipo de variable objetivo será categórica: humano / bot.

Durante el presente documento se ha señalado a GraphSAGE como el principal modelo a utilizar, pero éste no se conocía desde el principio y se ha seguido una fase de investigación para encontrar distintas posibilidades.

El concepto de **graph embbed** o incrustaciones de grafos consiste en la transformación de grafos a un vector o conjunto de vectores, de manera que éstos capturen la topología del grafo, las relaciones entre vértices y distinta información relevante. Una primera clasificación los puede diferenciar en base a si su aprendizaje es **transductivo** o **inductivo,** donde los primeros aprenden un vector de incrustación único por nodo, pero cuando cambia la estructura del grafo es necesario volver a reaprender cada uno de los vectores de características, siendo imposible generalizar para nodos invisibles o no vistos durante el entrenamiento. Sin embargo, **los métodos inductivos** aprenden el modo de agregación entre nodos y el modelo de agregación de la vecindad se puede utilizar para aprender los detalles de los nuevos nodos sin requerir un proceso de capacitación adicional. También se puede hacer una clasificación en base al ámbito de las incrustaciones realizadas, por ejemplo:

- **Incrustaciones de vértices o nodos.** Que codifica cada nodo con su propia representación vectorial, bastante empleados para la predicción en nodos y enlaces. Los algoritmos valorados y probados en este ámbito han sido:
    - **Word2vec** [51], técnica para el procesamiento del lenguaje natural que emplea un modelo de red neuronal para aprender asociaciones de grandes conjuntos de texto denominados **corpus**.
    - **Node2vec** [52], algoritmo utilizado para generar representaciones vectoriales de nodos en un grafo. Éste aprende representaciones de baja dimensión para nodos completando recorridos aleatorios desde un nodo marcado objetivo. Este algoritmo implica 2 pasos:
        - **Caminatas aleatorias** de segundo orden para generar secuencias de frases a partir de un grafo. Dichas frases son listas de identificadores de nodos y el conjunto de todas forman un **corpus.** Estas caminatas aleatorias son similares al algoritmo DeepWalk [53] al que se le aplican modificaciones para preservar bien el vecindario local de los nodos explorados.
        - En esta segunda fase se utiliza el corpus para aprender vectores de incrustación para cada nodo. Cada nodo será una palabra en un diccionario de tamaño similar al número de objetivos y se puede emplear Word2Vec para calcular los vectores de incrustación.

Sin embargo, se descartó le empleo de Node2Vec en el proyecto ya que su utilización conllevaría implementar un sistema que pudiese concatenar la **información de los atributos de nodo** a los generados para la incrustación en el grafo, así como tener que utilizar un modelo después para realizar la clasificación final, debido a que es un algoritmo transductivo.

- **Incrustaciones de grafos**. Donde se representará el grafo completo con un solo vector, empleados para realizar predicciones a nivel de grafo, por ejemplo, en comparación de estructuras químicas.

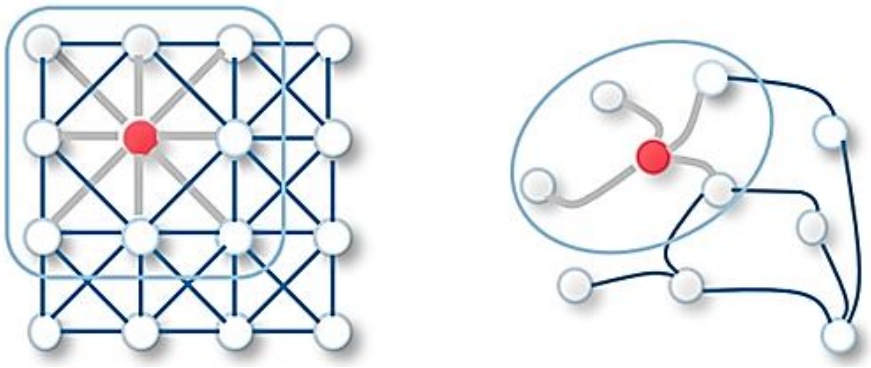
    - **Graph2Vec** [54], es una modificación de la variante de node2vec que aprende incrustaciones de subgrafos en grafos. Esto significa que dichas incrustaciones se calculan para un conjunto fijo de grafos en forma de matriz (o tabla de búsqueda), produciendo una incrustación únicamente válida para los grafos disponibles en el momento del entrenamiento. Con un enfoque basado en tres pasos, **muestreo de todos los subgrafos del grafo,** es decir de todos los nodos alrededor del nodo objetivo con una profundidad de exploración determinada. **Entrenamiento del modelo**, entendiendo el grafo de forma similar a un documento en el símil a word2vec, donde cada palabra es un conjunto de subgrafos proporcionando los grafos en la entrada como vectores one-hot. Y **Computación de incrustaciones,** proporcionando un identificador del grafo como un vector one-hot en la entrada, siendo la incrustación el resultado de la capa oculta. Al final los grafos con subgrafos similares tendrán incrustaciones parecidas.

El enfoque planteado para el proyecto con graph2vec consistía en la construcción de un grafo de relaciones por cada nodo. Para simplificar el exceso de información y relaciones de cada uno ellos, se pensó en utilizar la **centralidad de vector propio** [55] filtrando así las relaciones más relevantes para cada uno de sus usuarios. No obstante, fue nuevamente descartado, ya que todo este sistema es meramente transductivo y obliga en última instancia al empleo de otro clasificador para que sea entrenado con los vectores obtenidos.

Finalmente se puso el foco en algoritmos basados en GNN con carácter inductivo. GNN es un tipo determinado de red neuronal que puede funcionar directamente sobre grafos, recibiendo parámetros de entrada codificada como vectores, matrices o tensores que representarán dichos grafos. Éstas podrán aprender información más compleja ya que incorporan las relaciones existentes entre los nodos de la red. Hace no demasiados años las primeras aproximaciones de las redes neuronales solo podían utilizar datos regulares o euclidianos mientras que la mayoría de los datos estructurados en grafos disponen estructuras no euclidianas. Por ello, dada la falta de regularidad en las mismas ha llevado a las GNN a evolucionar como versiones generalizadas de CNN para poder funcionar con datos no euclidianos. Siendo ejemplos de estos algoritmos GCN (Graph Convolutional Network), GAT (Graph Attemption Network) o GraphSAGE.

Estas GCN son redes neuronales convolucionales de grafos de varias capas, que realizan operaciones similares a las CNN tradicionales en 1 o 2 dimensiones inspeccionando los nodos vecinos (como en el caso de las imágenes, donde se podrá decir que las relaciones de cada pixel en una posición dada, será siempre igual sobre la de sus pixeles o nodos vecinos exceptuando bordes de imagen). La diferencia reside en que el número de conexiones para GNN cambia en función del nodo a explorar y a su vez los nodos están desordenados (para un ejemplo, véase la F*igura 5-1*)

Figura 5-1: Red neuronal convolucional (izquierda) frente a red convolucional en grafo (derecha) [56]



5.1. Modelos tradicionales de referencia

Como se viene indicando a lo largo del documento, el objetivo del proyecto es tratar de aprovechar y entender la valiosa información que se encuentra en las relaciones entre las cuentas de Twitter, es decir, en su contexto social. A fin de comparar si los algoritmos son capaces de aprovecharla se utilizarán también modelos de aprendizaje automático más tradicionales, que solo clasifiquen las cuentas en base a los atributos de usuario y comparar así los resultados obtenidos. Se han empleado los siguientes modelos:

- **Random Forest o Bosque de decisión aleatorio** [57]. Se trata de un conjunto de árboles de decisión combinados de manera que no todos los árboles vean los mismos datos (*bagging*). Eso hace que cada árbol se entrene con distintas muestras del conjunto para el mismo problema. Así, cada uno de los árboles aportará una clasificación y el resultado será la clase con más votos arrojada por el bosque. En tareas de clasificación se denomina *soft-voting*, dando más importancia a los árboles que aporten valores más seguros. Se suelen emplear para atributos categóricos con dos posibilidades, de manera que este criterio medirá la información que aporta cada dato en el conjunto, siendo útiles tanto en problemas de regresión como de clasificación.

  Este modelo se utilizará en el proyecto dado que entre sus **ventajas** se podrían destacar que la preparación de los datos es mínima, puede manejar miles de variables identificando las más importantes, y en una de las salidas del modelo puede arrojar la importancia de las variables, utilizando esto para afinar el vector de características de los usuarios. Entre sus limitaciones se podría encontrar una tendencia al sobre ajuste en el entrenamiento, de manera que aprendan perfectamente los datos de entrenamiento, pero su generalización a los datos reales no sea tan buena.

- **SVM (Support Vector Machine)** [58]**,** es un algoritmo de aprendizaje automático que puede emplearse en tareas de regresión, clasificación y detección de valores atípicos. Un SVM construye un hiperplano o conjunto de hiperplanos en un espacio de alta dimensionalidad. Tratará de disponer el hiperplano o **vector de soporte** entre las dos clases a predecir lo más distantes posible haciendo de frontera en el mismo. Así los elementos que se encuentren a un lado pertenecerán a una clase y los que se encuentren al otro a la otra.

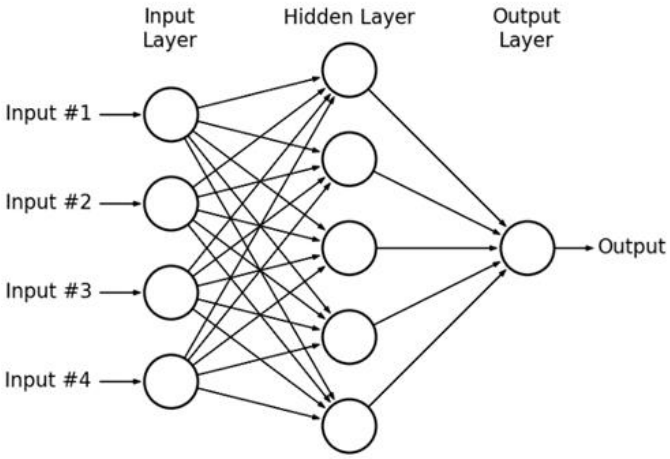
  Se ha empleado ya que los clasificadores SVM ofrecen una buena precisión y realizan predicciones más rápidas que otros algoritmos como *Naive Bayes*, además funcionan de manera diferente a las regresiones logísticas y se pueden emplear distintos kernel predefinidos en las librerías habituales.

- **MLP (MultiLayer Perceptron)** [59], es una Red Neuronal Artificial (ANN) compuesta por más de un perceptrón. Un perceptrón es un clasificador lineal, un algoritmo que clasifica la entrada en dos categorías siendo habitualmente un vector de características multiplicado por pesos **w**, al que se le añade sesgo **b**: $y = w \cdot x + b$. De manera que produce una salida única basada en varias entradas de valor real al formar una combinación lineal usando sus pesos de entrada.

  Un MLP consta al menos de 3 capas de nodos, una de **entrada**, para recibir la señal, una de **salida**, que aportará la predicción o clasificación sobre la entrada y entre estos dos, un número arbitrario de capas **ocultas** de nodos de activación no lineal, ya que los MLP están completamente conectados. Cada nodo en cada capa se conectará con ciertos pesos a los nodos de la capa siguiente y a excepción de la capa de entrada los nodos son neuronas que utilizan funciones de activación no lineales.

  Pueden utilizarse para tareas de clasificación, reconocimiento, predicción y aproximación de patrones. En la siguiente figura 5-2 se observa la estructura que sigue un MLP con una sola capa oculta de 6 nodos:

Figura 5-2: Ejemplo estructura Perceptron Multicapa extraído de [60], sección 2.3.



5.2. GraphSAGE

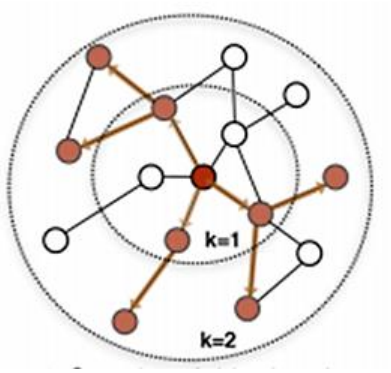
GraphSAGE (SAmple and aggreGatE) es un algoritmo iterativo que aprende las incorporaciones en grafos para cada nodo en un grafo determinado. Fue uno de los primeros algoritmos capaz de crear dichas incrustaciones de forma inductiva y en la actualidad se puede emplear de manera supervisada y no supervisada utilizándose para la clasificación de nodos, enlaces, detección de comunidades, análisis de redes, etc. La clave de su enfoque es que aprende cómo agregar información de características de la vecindad local de un nodo.

Así, el objetivo es aprender una representación para cada nodo apoyada en combinaciones de sus nodos vecinos. Se asume que los nodos que residen en el mismo vecindario deben tener incrustaciones parecidas. Como cada nodo puede tener asociado un vector de atributos, se podrá ejecutar una capa de GraphSAGE para **k** iteraciones, por lo que habrá una representación de nodo en cada iteración **k.**

El funcionamiento de GraphSAGE se divide en tres fases diferenciadas:

- **Construcción de contexto.** El algoritmo dispone un hiperparámetro **k** que determina la profundidad de la vecindad a explorar para cada uno de los nodos (ver Figura 5-3). Un k=1 señala que solo los nodos adyacentes serán contemplados, y k=2 explorara nodos a una distancia 2 del nodo origen. Es necesario tener en cuenta el diámetro de la red a explorar, ya que un k demasiado grande puede afectar incrustaciones de los demás nodos y diluir la del nodo objetivo, y un k muy bajo, inferior a 2 hará funcionar la red como un simple MLP.

Figura 5-3: Exploración de comunidades en base al parámetro K [25]



- **Agregación y actualización de información.** Como los nodos pueden ser representados por sus vecinos, la incrustación de un nodo dado puede ser descrita por las combinaciones de los vectores de las incrustaciones de nodos de su vecindario. Por cada iteración del algoritmo se obtendrá una nueva representación de dicho nodo. En la siguiente figura se expone el pseudocódigo del artículo original:

Figura 5-4: Pseudocódigo original algoritmo GraphSAGE [25]

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE debe en parte su inductividad a sus funciones de agregación, de entre las originales del artículo se reflejan dos de las empleadas en este proyecto ***mean*** y ***pool***, aunque en los resultados solo se plasman ***maxpool*** (los resultados en los conjuntos de datos dados han sido muy similares).

- **Agregador mean:** en este caso únicamente se utiliza el operador de medida media y el funcionamiento será casi equivalente a la regla de propagación convolucional empleada en GCN transductivo, por lo que se puede considerar convolutivo.

Figura 5-5: Función de agregación mean [25]

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}).$$

- **Agregador pool:** El vector de cada uno de los nodos vecinos se alimenta de manera independiente a través de una red neuronal completamente conectada, después se agregará la información en el conjunto vecino.

Figura 5-6: Función de agregación pool [25]

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\}),$$

Una vez obtenida la representación agregada para un nodo basada en sus vecinos, se actualiza el nodo actual mediante una combinación de su representación anterior y la representación agregada. Así la función de actualización es un marcador de posición que podrá ser como los agregadores, desde funciones simples de promedio hasta redes neuronales.

- **Función de pérdida**. El algoritmo debe calcular los pesos de los agregadores e incorporaciones, de forma que los nodos vecinos tengan incrustaciones parecidas mientras que los nodos lejanos o diferentes tengan incrustaciones dispares. En la siguiente figura se observa la función de pérdida indicada en el artículo original para el **aprendizaje no supervisado**. denotándose en la primera parte de la ecuación cierta similitud para los nodos **u, v** si se encuentran próximos en el grafo. En la segunda

parte, por el contrario, se dispone una variable Q que representa el número de muestras negativas y en general hace cumplir lo contrario, si los nodos **u,v** están distantes sus incrustaciones serán dispares. La inclusión de épsilon será para no disponer log(0).

Figura 5-7: Función de pérdida GraphSAGE [25]

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right)$$

En **aprendizaje supervisado** la función objetivo podrá calcularse siguiendo una predicción regular como entropía cruzada para tareas de clasificación de nodos, como en los ejemplos seguidos en este proyecto.

Uno de los **inconvenientes** que se pueden encontrar en los muestreos de GraphSAGE es la redundancia de nodos, lo que podría provocar que se calculen los mismos valores de forma reincidente. Se calcularían varias veces las mismas incrustaciones de nodos y se procesarían en el lote, aumentándolo de tamaño y generando mayores costes computacionales.

6. Pruebas y resultados obtenidos

A fin de entender en profundidad los resultados generados, se ha utilizado Botometer para intentar determinar el tipo de bot que predomina en cada conjunto de datos, justificando sus semejanzas o diferencias con los resultados de los modelos. Para ello, se ha obtenido un subconjunto totalmente aleatorio de 100 cuentas clasificadas como bots de cada uno. Pueden observarse los resultados en la *Tabla 6-1* (en el *Anexo B* se encuentra un ejemplo de uso de Botometer con la descripción de cada una de las categorías):

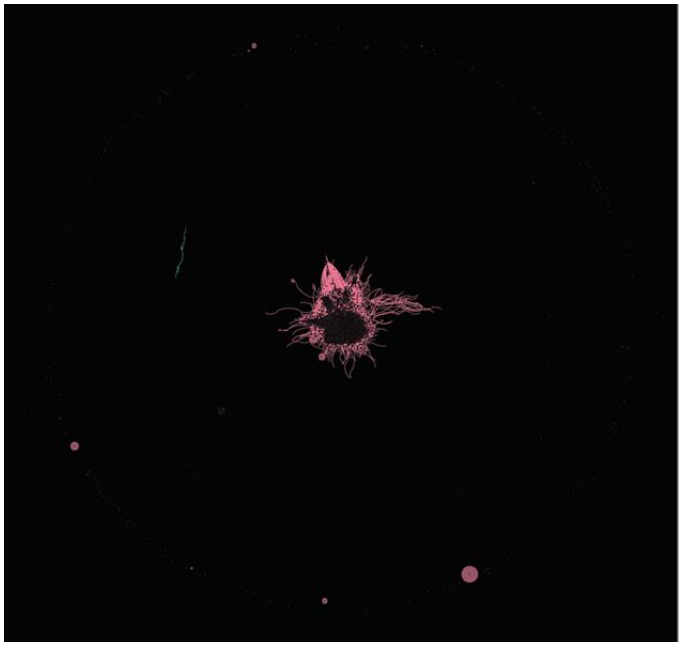
Tabla 6-1: Clasificación Botometer subconjunto 100 cuentas aleatorias de cada dataset.

VALORES CLASIFICACIÓN	CONJUNTO 1	CONJUNTO 2
Etiqueta universal	0.805	0.828
Astroturf	0.068	0.109
Financial	0.256	0.119
Self_declared	0.172	0.068
Fake_follower	0.396	0.213
Spammer	0.201	0.062
Overall	0.560	0.731
Other	0.507	0.778

Al disponer de dos conjuntos de datos diferenciados se realizarán pruebas de los modelos entrenados sobre un conjunto y probados en el otro. En el caso de GraphSAGE, el objetivo es demostrar sus capacidades inductivas con datos no visualizados durante el entrenamiento del modelo. Y los algoritmos más tradicionales servirán para tener referencias en cuanto a precisión.

En adelante, serán denominadas pruebas sobre comunidad aislada. En la *Figura 6-1* se puede observar en color verde el Dataset 1 y en color rosado el Dataset 2, viendo como ambos no están relacionados. Se expone el conjunto sin filtrar, pero se entiende que en las pruebas se utilizarán con los cribados aplicados a retweet y replies. Además, algunas cuentas del Dataset 1 se encontraban en el Dataset2, por lo que se ha decidido eliminarlas del Dataset 1, reduciéndolo a 321 nodos útiles para retweets y 206 nodos para replies.

Figura 6-1: Sistema de prueba con comunidades aisladas.



Por último, se menciona la importancia de definir el alcance del problema dentro de la red, ya que si se trata de un **problema de corto alcance,** como en las redes sociales, las predicciones se basarán en información de corto alcance arrojada por el vecindario cercano al nodo. Las predicciones no mejorarán agregando información de nodos muy distantes por lo que se recomienda completar estas tareas con redes poco profundas [61].

En la siguiente tabla, proporcionada por el artículo [62] y basado en la publicación [63], se puede observar como el rendimiento de modelos basados en GCN pierden efectividad conforme se agregan más capas profundas y se deja otro artículo de referencia [64], en el que se expone el mismo fenómeno. Por esto, en este proyecto se utilizará la estructura básica de GraphSAGE con una capa profunda por nivel de muestreo.

Tabla 6-2: Resultados clasificación GCN en la red de citas CoauthorsCS con varias layer. Publicado en [62].

Model	2-Layer	4-Layer	8-Layer	16-Layer	32-Layer	64-Layer
GCN-res	88.18±1.59	86.50±1.87	84.83±1.93	78.60±4.28	59.82±7.74	39.71±5.15
PairNorm	79.98±3.80	82.32±2.79	81.52±3.66	82.29±2.62	81.91±2.45	81.72±2.82
NodeNorm	**89.53**±1.29	88.60±1.36	88.02±1.67	88.41±1.25	88.30±1.30	87.40±2.06

6.1. Métricas de referencia

Para poder evaluar el rendimiento de los modelos será necesario definir métricas que sean capaces de valorar la calidad de sus predicciones. A lo largo del proyecto se han empleado y analizado las siguientes:

- **Matriz de Confusión,** esta métrica servirá para evaluar el modelo en base a los siguientes resultados obtenidos donde:
    - **TP** (*True positive*) Verdaderos positivos, representa el número de positivos clasificados correctamente como positivo.

- **FP** (*False Positive*) Falsos Positivos, será la cantidad de negativos clasificados incorrectamente como positivos.
- **FN** (*False Negative*) Falso negativo, volumen de positivos clasificados incorrectamente como negativos.
- **TN** (*True Negative*) Verdadero negativo, cantidad de negativos clasificados de forma correcta como negativos.

Tabla 6-3: Información aportada por una matriz de confusión.

		Predicción	
		Positivo	Negativo
Observación	Positivo	TP	FP
	Negativo	FN	TN

- **Precisión.** Esta métrica determinará la fracción de casos clasificados de forma correcta como positivos frente a todos los ejemplos que el modelo ha clasificado como positivos.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall,** medirá el porcentaje de los ejemplos positivos etiquetados correctamente entre todos los positivos reales. A la hora de interpretar estos valores habrá que tener presente el balanceo o no de las etiquetas del conjunto, ya que si una etiqueta predomina sobre otra se podrán obtener buenos resultados, aunque el modelo no sea del todo óptimo.

$$Recall = \frac{TP}{TP + FN}$$

- **Accuracy,** genera el porcentaje total de aciertos del modelo y será una buena métrica sobre conjuntos de datos balanceados.

$$Accuracy\ (Acc) = \frac{TP + TN}{TP + FN + TN + FP}$$

- **F1-Score,** combina las medidas de precisión y recall en un solo parámetro, siendo útil sobre conjuntos de datos no balanceados.

$$F1 = 2 \cdot \frac{Precision\ x\ Recall}{Precision + Recall}$$

6.2. Modelos tradicionales

Las pruebas fueron completadas con los siguientes modelos: *RandomForest*, *SVM* (Support Vector Machine) y *MLP* (Multilayer Perceptron)

6.2.1. Modelo RandomForest

Para ajustar rápidamente el modelo se ha utilizado **GridSearchCV** [65]. GridSearchCV es una clase disponible en *scikit-learn* [66] que sirve para evaluar y seleccionar parámetros e hiperparámetros de los modelos de manera sistemática. Se configura una matriz en la que se incluyen varios valores de los atributos del modelo y se ejecuta sobre una porción de los datos, retornando la mejor configuración posible para éstos. Se consigue así encontrar un mejor punto de partida para el entrenamiento y evaluación.

Se han incluido en la matriz valores para los parámetros: **max_depth,** que determina la profundidad máxima del árbol, **max_features** para garantizar que todos los árboles se entrenen con muestras aleatorias de los datos, y utilizado una serie de predictores en cada división: ***auto,*** los utilizará todos, ***sqrt*** empleará la raíz del total y ***log2*** tomará el logaritmo del total de predictores.

N_estimators, será el número de árboles que contendrá el modelo. En la siguiente figura se podrá encontrar la matriz generada y el listado de parámetros óptimos retornado por *GridSearch*:

Figura 6-2: Matriz GridSearch para RandomForest (izquierda) y resultados óptimos calculados (derecha).

```
params = {
 'n_estimators': [50, 200, 350, 500],
 'max_features': ['auto', 'sqrt', 'log2'],
 'max_depth' : [5, 6, 7, 8, 9, 10],
 'criterion' :['gini', 'entropy']
}
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': 10,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 500,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

Con la configuración indicada, se realiza la primera prueba sobre el Dataset 2 completo (11664 elementos) empleando el 80% de los datos para entrenamiento y el 20% restante para validación. Se comparan los resultados obtenidos utilizando los datos del conjunto transformados mediante **yeo-yonson** [67] y eliminando los atributos que han aportado menos valor frente a las 75 características con los datos en bruto.

Figura 6-3: Métricas RF sin normalizar (izquierda) frente a datos normalizados (derecha).

	precision	recall	f1-score	support
0	0.91	0.98	0.94	1770
1	0.90	0.68	0.78	563
accuracy			0.90	2333
macro avg	0.90	0.83	0.86	2333
weighted avg	0.90	0.90	0.90	2333

	precision	recall	f1-score	support
0	0.90	0.98	0.94	1770
1	0.90	0.67	0.77	563
accuracy			0.90	2333
macro avg	0.90	0.82	0.85	2333
weighted avg	0.90	0.90	0.90	2333

El valor *Acc* obtenido es de **0.905** con los datos sin normalizar frente al **0.902** ya transformados. Como se adelantó en la *sección 5.2* estos resultados no son sorprendentes ya que *Random Forest* no requiere

aplicar grandes esfuerzos en el preprocesado. En la *Figura 6-3* se observan el resto de las métricas generadas.

En ambos resultados puede apreciarse como el valor de recall o sensibilidad es mucho más bajo para la clasificación de los bot, debido en parte al desbalanceo en las etiquetas. Se ha tratado de corregir de distintas formas, utilizando el atributo ***class_weight = balanced,*** que utiliza los valores para ajustar los pesos inversamente proporcionales a las frecuencias de clase en los datos. La precisión en la clase de bot no ha mejorado pero los valores de recall son algo superiores. En la siguiente figura se exponen las métricas obtenidas para los datos transformados empleando *class_weight,* donde se comprueba que la métrica de recall mejora algo, aunque se pierde en la clasificación con la clase 1 (bot):

Figura 6-4: Métricas Dataset 2, datos transformados y empleo class_weight.

	precision	recall	f1-score	support
0	0.92	0.96	0.94	1770
1	0.85	0.72	0.78	563
accuracy			0.90	2333
macro avg	0.88	0.84	0.86	2333
weighted avg	0.90	0.90	0.90	2333

Adicionalmente al empleo de *class_weight*, se ha probado a reducir el muestreo de la clase predominante "*Subsampling*", empleando la función **NearMiss** con distribuciones de los datos:

- Distribución de 0.4: ({0: 5632, 1: 2253})
- Distribución de 0.5: ({0: 4506, 1: 2253})

Figura 6-5: Metricas Dataset 2 con subsampling 0.4 (izquierda) y 0.5 (derecha).

	precision	recall	f1-score	support
0	0.90	0.73	0.81	1770
1	0.47	0.75	0.57	563
accuracy			0.73	2333
macro avg	0.68	0.74	0.69	2333
weighted avg	0.80	0.73	0.75	2333

	precision	recall	f1-score	support
0	0.91	0.97	0.94	1770
1	0.88	0.68	0.77	563
accuracy			0.90	2333
macro avg	0.89	0.83	0.85	2333
weighted avg	0.90	0.90	0.90	2333

De las métricas arrojadas por las figuras anteriores, se puede sacar la conclusión que la reducción en los datos del conjunto funciona peor que el balanceo en los pesos al entrenamiento. Ya que los resultados generados empeoran según se elimina una porción mayor de los datos.

Tabla 6-4: Importancia atributos en Random Forest.

ATRIBUTO	IMPORTANCIA
rt_ratio_reply	0.133686
rt_ratio	0.128946
rt_num_medio_hashtags	0.062988
rt_num_medio_urls	0.056187
pr_media	0.055574
tw_n_palabras_lexico	0.047165
pr_followers	0.027544
tw_likes_count_desv_tip	0.022074
tw_likes_count_var	0.021887
tw_replies_count_desv_tip	0.021673

Random Forest también ofrece un listado de los atributos en base a su importancia en la clasificación. En la *Tabla 6-4* se muestra un extracto de los 10 más relevantes para la clasificación llevada a cabo con los datos transformados. Estos listados han servido en el ajuste del vector de características final.

Como el objetivo es poder comparar los resultados de este modelo con GraphSAGE, se truncarán los datos elaborando los dataset referentes a **retweet** (*10340 usuarios*) y **replies** (*7220 usuarios*), pues como se vio en la *sección 4.3* el número de usuarios de cada uno es diferente.

Tabla 6-5: Resultados obtenidos con Random Forest.

Random Forest	Retweet Datos NO procesados	Retweet Datos procesados	Reply Datos NO procesados	Reply Datos procesados
Accuracy Dataset 2	0.913	0.918	0.938	0.929
Accuracy sobre comunidad aislada	-	0.456	-	0.373

A continuación, se entrenan ambos modelos con los Datos del Dataset 2 y se prueba la precisión sobre el Dataset 1 (preparado el conjunto de replies y el de retweet) emulando una comunidad de usuarios aislada. La precisión en las pruebas realizadas se refleja en la *Tabla 6.5*.

En la siguiente imagen (Figura 6-6) se muestra el resto de las estadísticas generadas en las pruebas sobre el conjunto pequeño.

Figura 6-6: Replies (izquierda) y Retweets (derecha)

	precision	recall	f1-score	support
0	0.37	1.00	0.54	77
1	0.00	0.00	0.00	129
accuracy			0.37	206
macro avg	0.19	0.50	0.27	206
weighted avg	0.14	0.37	0.20	206

	precision	recall	f1-score	support
0	0.45	0.99	0.62	145
1	0.67	0.01	0.02	176
accuracy			0.45	321
macro avg	0.56	0.50	0.32	321
weighted avg	0.57	0.45	0.29	321

Los resultados en el conjunto de replies son francamente pobres, generando todas las clasificaciones como cuentas humanas, cuando 129 de los valores evaluados son cuentas bots. Pasa algo parecido en el caso del conjunto de retweets, pues en el ejemplo solo ha clasificado 2 cuentas bots bien frente a 174 mal.

Como conclusión, de este modelo se ha obtenido información importante para afinar el vector de características. Por otro lado, los resultados sobre los modelos originales han sido muy buenos, pero no es capaz de generalizar con datos no vistos en la fase de entrenamiento.

6.2.2. SVM (Support Vector Machine

Como en el caso anterior, se emplea GridSearchCV para tratar de ajustar el punto de partida en el entrenamiento del modelo. En la *Figura 6-7* se presenta la matriz de GridSeachCV y los parámetros retornados por éste.

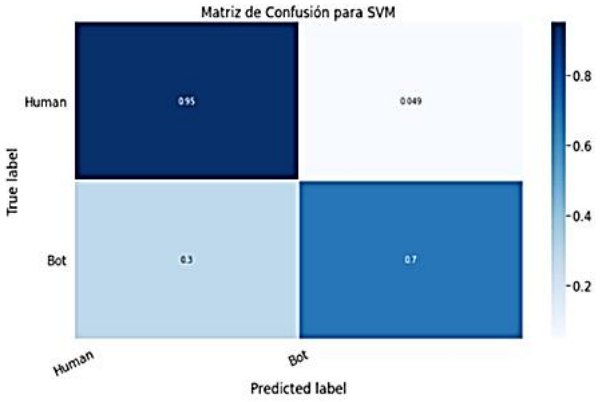
Figura 6-7: Matriz GridSearch para SVM (izquierda) y resultados óptimos calculados (derecha)

```
params = {'C': [0.001,0.01, 0.1, 1, 10, 100, 1000],
 'gamma': [10, 1, 0.1, 0.01, 0.001, 0.0001],
 'kernel': ['rbf']}
```

```
{'C': 10,
 'break_ties': False,
 'cache_size': 200,
 'class_weight': None,
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 3,
 'gamma': 0.0001,
 'kernel': 'rbf',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}
```

Con este modelo también se ha probado la importancia de la transformación de los datos a la entrada. Con todos los usuarios del Dataset 2 y los datos en bruto se ha obtenido una precisión *Acc* de **0.751** frente al **0.889** con los datos transformados. En la *Figura 6-8* se puede observar la matriz de confusión generada para esta última prueba, así como el resto de métricas:

Figura 6-8: Matriz de confusión y métricas SVM Dataset 2 completo.



	precision	recall	f1-score	support
0	0.91	0.95	0.93	1752
1	0.83	0.70	0.76	581
accuracy			0.89	2333
macro avg	0.87	0.83	0.84	2333
weighted avg	0.89	0.89	0.89	2333

Aplicando las técnicas anteriormente descritas para el desbalanceo de clases, se consigue un score algo inferior, *Acc* 0.876 pero con valores de sensibilidad al cambio de clases bastante mayores, a costa de perder precisión en la clase de bot. Se pueden observar estos detalles en la siguiente *Figura 6-9*.

Figura 6-9: Métricas Dataset 2 SVM aplicando medidas para desbalanceo de clases

	precision	recall	f1-score	support
0	0.95	0.89	0.91	1752
1	0.71	0.85	0.77	581
accuracy			0.88	2333
macro avg	0.83	0.87	0.84	2333
weighted avg	0.89	0.88	0.88	2333

En la *Tabla 6-6* se muestran los resultados sobre el Dataset 2 con las agrupaciones de los usuarios en base a replies y retweet, así como la precisión de los modelos en las "comunidades aisladas".

Tabla 6-6: Resultados con SVM

SVM	Retweet Datos NO procesados	Retweet Datos procesados	Reply Datos NO procesados	Reply Datos procesados
Accuracy Dataset 2	0.796	0.899	0.904	0.926
Accuracy sobre comunidad aislada	-	0.459	-	0.387

Con SVM se puede concluir que la transformación previa de los datos sí arroja mejoras sustanciales en la clasificación, siendo más notable para el conjunto de usuarios asociado a los retweet. También se observa cómo pierde rendimiento al utilizar el modelo con datos nuevos.

Figura 6-10: Métricas de clasificación SVM

	precision	recall	f1-score	support
0	0.45	0.97	0.62	143
1	0.64	0.04	0.08	175
accuracy			0.46	318
macro avg	0.54	0.51	0.35	318
weighted avg	0.55	0.46	0.32	318

En la *Figura 6-10* se muestran las métricas generadas por el conjunto de retweet en la comunidad aislada, siendo llamativo los resultados de *Recall*. Estos valores atípicos son debidos tanto a que el modelo no es capaz de generar los hiperplanos de separación como al desbalanceo de la muestra (número de humanos frente a bots). Clasificando de forma incorrecta 168 bot frente a 7 bien. Sin embargo, clasifica adecuadamente 139 cuentas de humanos y sólo 4 mal. Por tanto, la sensibilidad del modelo para clasificar bots es muy baja y predice casi todo como cuentas legítimas.

6.2.3. MLP (Multilayer Perceptron)

Los parámetros utilizados para la configuración de las pruebas sobre MLP han sido los siguientes: "*Activation: relu*" (la función de unidad lineal rectificada devuelve f (x) = max (0, x), un *batch_size* de ejecución de 200, *shuffle* verdadero para poder mezclar los datos como ya se ha descrito y solver *adam* (optimizador estocástico basado en gradientes). Se ha utilizado en alguna prueba *lbfgs*, y aunque se han obtenido resultados muy parecidos ha sido un poco peor en los primeros ensayos, por lo que se ha empleado *adam* en todas las presentadas.

En primer lugar, se completa una ejecución con el total de datos del Dataset 2 (11664 usuarios) para generar una referencia con 64 nodos por capa obteniendo las siguientes precisiones según el número de capas utilizadas:

- 1 capa, *accuracy* ***0.879***
- 2 capas, *accuracy* ***0.864***
- 3 capas, *accuracy* ***0.866***

Seguidamente se realizan las pruebas sobre los conjuntos de datos objetivo de retweets y replies empleando 1, 2 y 3 capas respectivamente. Una vez completados se han probado los mejores modelos creados sobre el conjunto de la comunidad aislada. En la *Tabla 6-7* se puede ver el valor *accuracy* obtenido en cada caso.

Tabla 6-7: Resultados con MLP

SVM	1 layer	2 layer	3 layer
Accuracy Dataset 2 Replies	0.919	0.915	0.904
Accuracy sobre comunidad aislada Replies	0.441	-	-
Accuracy Dataset 2 Retweet	0.891	0.888	0.890
Accuracy sobre comunidad aislada Retweets	0.485	-	-

En todas las pruebas completadas por este modelo se obtienen más o menos los mismos resultados, independientemente del grupo de usuarios bajo el estudio. El número de capas no parece mejorar el resultado (incluso habiendo completado pruebas aisladas sobre 8-10 capas) y las clasificaciones de los usuarios humanos son siempre mejores, con resultados superiores tanto en *precision, recall* o *f1-score*, siendo en torno al 0.70 para los bot mientras que para cuentas humanas se posiciona sobre el 0.90.

Figura 6-11: Métricas MLP comunidad aislada Replies (izquierda) y Retweets (derecha)

	precision	recall	f1-score	support
0	0.40	0.95	0.56	77
1	0.82	0.14	0.24	129
accuracy			0.44	206
macro avg	0.61	0.54	0.40	206
weighted avg	0.66	0.44	0.36	206

	precision	recall	f1-score	support
0	0.46	0.83	0.59	145
1	0.59	0.20	0.30	176
accuracy			0.49	321
macro avg	0.53	0.52	0.45	321
weighted avg	0.53	0.49	0.43	321

En ambos sistemas se han obtenido valores *Acc* similares y las matrices de confusión presentan comportamientos parecidos. En el conjunto de replies se han clasificado de forma correcta 73 usuarios humanos frente a 4 mal, y 18 bots correctamente frente a 111 mal. Por su parte, en el conjunto de retweets se consigue clasificar adecuadamente 120 humanos frente a 25 de manera incorrecta, y 36 bots bien frente a 140 incorrectos. Sin embargo, si existe una diferencia notable en cuanto a la precisión por cada clase, donde se ve más diferencia en las replies.

6.3. Resultados Dataset 1 con GraphSAGE

En las siguientes subsecciones se presentarán las pruebas realizadas sobre el Dataset 1 de usuarios con GraphSAGE.

6.3.1. Contexto en base a relaciones Following/Followers

Se aprovecha este primer ejemplo con GraphSAGE para explicar cómo se han gestionado los parámetros del algoritmo empleando la librería *Stellargraph*. Para comenzar, se deben preparar los datos y grafo asociado, pudiendo utilizar distintos formatos para los grafos como *NetworkX* [68] aunque en este caso se han preparado directamente sobre *Dataframes* de *Pandas* [33]. Para los atributos de cada nodo, se dispondrá el **id** de usuario como índice del conjunto para poder enlazar sus atributos con las relaciones que se encuentren dentro del grafo.

Una vez preparado, se generará un objeto ***StellarDiGraph (features, graph)***. *StellarDiGraph* se referirá a grafos dirigidos y representará el conjunto de datos utilizado por el modelo.

Para facilitar los datos del grafo a Keras, la biblioteca implementa un generador especializado en el modelo, en este caso para predicción de nodos sobre grafos dirigidos con GraphSAGE, **D*irected-***

GraphSAGENodeGnerator. Hay que pasarle variables como ***batch_size*,** para la parte del entrenamiento y los vectores ***in_samples*** y ***out_samples***. Éstos representan el número de nodos que se deben muestrear por nivel, por ejemplo, en la *Figura 6-12* se explorarán 3 niveles (k=3) y se muestrearán 15 nodos de entrada y salida en el primer nivel y 10 en los demás.

Figura 6-22: Variables para definir muestreo en GraphSAGE

```
in_samples = [15,10,10]
out_samples = [15,10,10]
```

El parámetro ***weighted*** referencia a las conexiones del grafo. Si se incluyen pesos comenzará tomando los vértices que tengan un mayor peso para ese nodo, tanto de entrada como de salida. Si fuese falso, la exploración se realizaría de forma aleatoria, aunque parece razonable entender que, si una conexión entre nodos se repite muchas veces, tendrá más probabilidades de muestreo, incluso de llegar a repetirse.

Figura 6-33: Parámetros DirectedGraphSAGENodeGenerator

```
generator = DirectedGraphSAGENodeGenerator(g, batch_size, in_samples, out_samples, weighted=True)
```

Con el método *generator.flow* se crean iteradores sobre los nodos que se utilizarán tanto para entrenamiento como para validación y se puede utilizar el parámetro *shuffle* para mezclar los datos de entrenamiento antes de cada época.

Figura 6-14: Parámetros generator.flow

```
train_gen = generator.flow(train_data.index, train_targets, shuffle=True)
test_gen = generator.flow(test_data.index, test_targets)
```

Cerrando esta explicación se listan los agregadores disponibles: *MeanAggregator, MeanPoolingAggregator, MaxPoolingAggregator, AttentionalAggregator* y se menciona que el tamaño de ***layer_sizes*** debe ser igual que los parámetros *in_samples* y *out_samples* pues determina el número de características ocultas en cada capa del modelo. Se presenta el fragmento de código donde se ensambla el modelo antes de su entrenamiento.

Figura 6-15: Código ensamblado DirectedGraphSAGE

```
base_model = DirectedGraphSAGE(
    layer_sizes=[64, 64, 64], generator=generator, bias=True,
    dropout=0.5, aggregator = MaxPoolingAggregator
)
x_inp, x_out = base_model.in_out_tensors()

prediction = layers.Dense(units=1, activation="sigmoid")(x_out)

model = Model(inputs=x_inp, outputs=prediction)
```

Seguidamente se expondrán los resultados obtenidos en esta fase del proyecto. Se destaca la importancia de comprender los detalles del grafo de relaciones en cada caso, para interpretar como configurar los atributos del algoritmo y los resultados generados. No obstante, en esta sección se tomará una configuración similar en todas las pruebas para poder comparar los resultados de las relaciones de igual

manera. El valor para *layer_size* será de 64 y se completarán exploraciones para k=2 y k=3 con el agregador *MaxPooling*, a 50 épocas y con un *batchsize* de 64. El número de nodos a explorar será de 15 en el primer nivel y 10 en los demás. Esta configuración se mantendrá en las secciones 6.3.1, 6.3.2 y 6.3.3.

Con el grafo de relaciones basado en following/followers el objeto *StellarDiGraph* prepara el siguiente objeto, en el que se pueden ver el número de nodos contenidos y de relaciones. El vector de características tiene 55 atributos (como en todas las pruebas que se realizarán) y el volumen de pesos por relación en el vector será de 1.

Figura 6-46: Objeto StellarDiGraph Dataset 1 grafo following/followers

```
StellarDiGraph: Directed multigraph
 Nodes: 348, Edges: 8535

 Node types:
  default: [348]
    Features: float32 vector, length 55
    Edge types: default-default->default
```

Para poder obtener las métricas de resultados, será necesario utilizar el generador definido anteriormente (*test_gen*). Las predicciones serán el resultado de la capa *softmax*. En la Tabla 6-8 se expone una imagen que presenta el ejemplo con 10 de los resultados de esta capa frente a la etiqueta de verdad para cada uno de los nodos

Tabla 6-8: Ejemplo salida capa softmax y etiquetas de verdad

id	Predicted	True
979117618295656450	0.953765	1
3020402578	0.981794	1
1268082668	0.092453	0
448110696	0.955254	1
2930775934	0.127782	1
460216069	0.073474	0
422924590	0.981986	1
825028579767390208	0.981055	1
895869750	0.121476	0
804074275376431105	0.973563	1

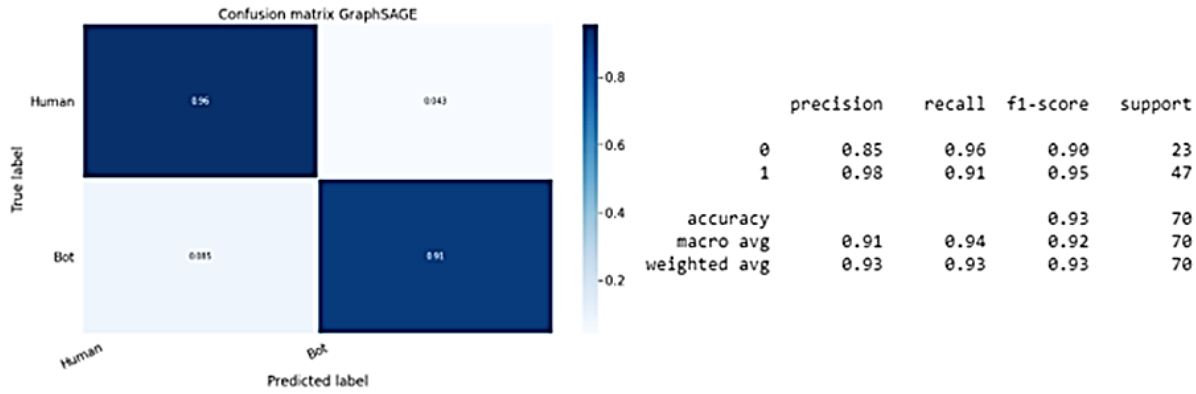
Finalmente, en la tabla 6.9 se pueden encontrar los resultados *accuracy* y *loss* en los grafos Following/Followers.

Tabla 6-9: Resultados GraphSAge Dataset 1 conjunto Following/Followers

GraphSAGE	K = 2		K = 3	
	Acc	Loss	Acc	Loss
Following/Followers	0.926	0.168	0.925	0.218

El mejor resultado se puede encontrar para K=2 para el que se muestran el resto de las métricas (Figura 6-17):

Figura 6-57: Métricas GraphSAGE Dataset 1 conjunto Following/Followers (k=2)



	precision	recall	f1-score	support
0	0.85	0.96	0.90	23
1	0.98	0.91	0.95	47
accuracy			0.93	70
macro avg	0.91	0.94	0.92	70
weighted avg	0.93	0.93	0.93	70

Como se muestra en la Figura 6.17, atendiendo a la matriz de confusión se han clasificado de forma correcta 22 cuentas de humanos y una sola de manera incorrecta. En el caso de los bots, han sido 4 los fallos en la clasificación frente a 43 aciertos.

No obstante, el problema es que el conjunto de datos es muy pequeño y el volumen de usuarios bot está desbalanceado. Empleando un 80% de los datos para el entrenamiento únicamente se cuenta con 70 usuarios para las validaciones, y de éstos en la mayoría de los ejemplos solo 20-23 usuarios legítimos. Este problema se repetirá a lo largo de esta subsección y es el motivo de haber generado nuevas descargas y conjuntos más grandes.

6.3.2. Contexto basado en Retweets

Con la configuración indicada en el apartado anterior se presenta el objeto *StellarDiGraph* generado para las pruebas:

Figura 6-18: Objeto StellarDiGraph Dataset 1 grafo Retweets

```
StellarDiGraph: Directed multigraph
 Nodes: 523, Edges: 19080

 Node types:
  default: [523]
    Features: float32 vector, length 55
    Edge types: default-default->default

 Edge types:
    default-default->default: [19080]
        Weights: range=[1, 1547], mean=8.42678, std=36.7264
        Features: none
```

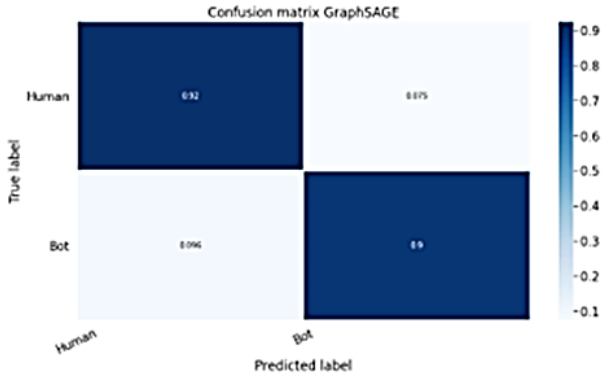
Seguidamente se muestran los resultados de *Accuracy* y *Loss* para el grafo de retweets:

Tabla 6-10: Resultados GraphSAGE Dataset 1 conjunto Retweet

GraphSAGE	K = 2		K = 3	
	Acc	Loss	Acc	Loss
Retweets	0.923	0.286	0.885	0.332

Y se presentan el resto de las métricas y matriz de confusión para el mejor modelo conseguido con k=2.

Figura 6-19 Métricas y Matriz de confusión GraphSAGE Dataset 1 conjunto Retweets (k=2)



	precision	recall	f1-score	support
0	0.91	0.92	0.92	53
1	0.92	0.90	0.91	52
accuracy			0.91	105
macro avg	0.91	0.91	0.91	105
weighted avg	0.91	0.91	0.91	105

En este conjunto se encuentran unos resultados más parejos en cuanto a la clasificación de bot y humanos. Es cierto que el número de cuentas del conjunto sigue siendo pequeño pero el grafo es mucho más nutrido.

La matriz de confusión indica que se han clasificado correctamente 49 cuentas legítimas, fallando únicamente en 4. Por el contrario, se han clasificado 47 cuentas bots y sólo 5 de ellas de manera incorrecta. Según las estadísticas expuestas en la *Tabla 4-6* el diámetro de la red en estos datos es de 16, por lo que se ha intentado hacer una prueba con k=4 para ver si la precisión cambia. Sin embargo, esto no ha sido posible por falta de capacidad de cómputo. Se ha empleado **GoogleColab** en Notebooks con aceleradores gráficos, pero los tiempos estimados de las sesiones son reducidos. La previsión de 1 época superaba las 2 horas y los tiempos de ejecución de los cuadernos no pueden superar las 12h antes de desconectarse.

6.3.3. Contexto basado en Replies

Según la configuración marcada se presenta el objeto *StellarDiGraph* generado para el grafo de replies:

Figura 6-60: Objeto StellarDiGraph Dataset 1 grafo Replies

```
StellarDiGraph: Directed multigraph
 Nodes: 354, Edges: 1590

 Node types:
  default: [354]
    Features: float32 vector, length 55
    Edge types: default-default->default

 Edge types:
    default-default->default: [1590]
        Weights: range=[1, 228], mean=3.61824, std=10.2222
        Features: none
```

A continuación, se presentan los resultados obtenidos con estos datos:

Tabla 6-81: Resultados GraphSAGE Dataset 1 conjunto Replies

GraphSAGE	K = 2		K = 3	
	Acc	Loss	Acc	Loss
Replies	**0.943**	0.188	0.915	0.313

En este conjunto el grafo tiene muy pocas relaciones, sólo 1590 enlaces y el grado medio de los nodos es inferior a cinco, con un valor medio menor a 4 para los nodos de origen e inferior a 6 para los nodos de destino. Se ha preparado una prueba reduciendo los parámetros exploratorios para que el algoritmo no tenga que reponer los nodos faltantes mediante reemplazo en los muestreos:

- *in_samples* = [6,5]
- *out_samples* = [4,3]

El valor *accuracy* es de 0.929 y la pérdida de 0.255. En la siguiente figura se muestran el resto de las métricas generadas donde se puede apreciar que no se ha encontrado mejora respecto a las exploraciones anteriores encontrando valores similares en todos los scores. Se han clasificado de forma correcta 22 cuentas de usuarios humanos frente a 2 mal etiquetadas, y 44 bots adecuadamente frente a 3 fallos en la clasificación.

Figura 6-71: Objeto StellarDiGraph Dataset 1 grafo Replies

	precision	recall	f1-score	support
0	0.88	0.92	0.90	24
1	0.96	0.94	0.95	47
accuracy			0.93	71
macro avg	0.92	0.93	0.92	71
weighted avg	0.93	0.93	0.93	71

Se cierra la sección 6.3 con una tabla resumen de los valores obtenidos en cada una de las pruebas, concluyendo que éstas han sido muy similares en todos los casos, pero han funcionado algo mejor con exploración de vecindario de dos niveles.

Tabla 6-92: Resultados GraphSAGE Dataset 1

GraphSAGE	K = 2		K = 3	
	Acc	Loss	Acc	Loss
Following/Followers	0.926	0.168	0.925	0.218
Retweets	0.923	0.286	0.885	0.332
Replies	0.943	0.188	0.915	0.313

El problema detectado a lo largo de esta sección es que se disponen de tan pocos datos que los resultados pueden variar incluso dependiendo de su ejecución, independientemente de las opciones empleadas para corregir el desbalanceo y otros factores de los modelos. Los resultados son similares en todos los casos, pero se necesitan más perfiles de usuarios para poder sacar conclusiones.

6.4. Resultados Dataset 2 con GraphSAGE

Seguidamente, se muestran los resultados obtenidos en las pruebas realizadas sobre el Dataset 2. El objetivo es entrenar modelos que se puedan exportar a cualquier conjunto de datos. Para utilizar cuentas que no se han contemplado durante la fase de entrenamiento, es necesario volver a generar el objeto ***StellarDiGraph*** para que contenga todos los datos, los nuevos y los que ya estaban disponibles, así como el grafo de relaciones completo, pudiendo incluir subgrafos inconexos. Después será necesario crear el generador para poder pasar toda esta información al motor *Keras*, como ya se ha explicado:

Figura 6-22: Nuevo generador para inducción de nodos

```
generator = DirectedGraphSAGENodeGenerator(g, batch_size, in_samples, out_samples, weighted=True)
```

Así como un nuevo generador con los datos a evaluar para poderlos incluir en las clases *predict* extrayendo evaluaciones y métricas:

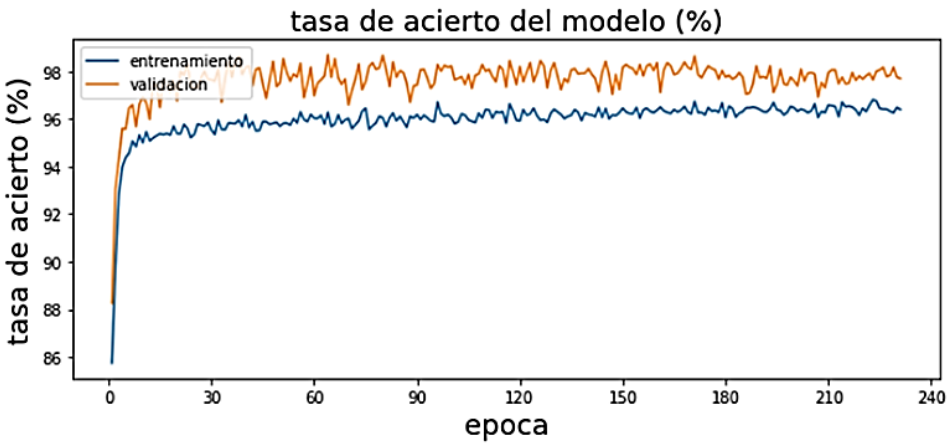
Figura 6-23: Parámetros generator.flow de los nodos a inducir

```
generator.flow(user_features_.index, user_targets_)
```

Un factor importante, es que la configuración de estos objetos y generadores deberá ser la misma que se definió en el modelo de origen, pero con los datos nuevos. Por tanto, dependiendo del objetivo buscado podría ser preferible entrenar el modelo con parámetros más generales al problema que a los datos concretos disponibles en el entrenamiento. Ya que si se ajustan los parámetros de exploración a unos datos puede que no sean los idóneos en los nuevos conjuntos.

Uno de los problemas encontrados en el entrenamiento del modelo con las primeras muestras, fue visualizar la línea de validación del modelo en todo momento por encima de la de entrenamiento. Se expone un ejemplo de una muestra entrenando para k=2 sobre el conjunto de retweets.

Figura 6-84: Tasas entrenamiento GraphSAGE conjunto Retweets (k=2)



Aunque se han obtenido valores *Acc 0.970 y Loss 0.075*, muy buenos, no se ha considerado que el modelo sea correcto. Para tratar de evitar este comportamiento, se han modificado los parámetros de ejecución. Por un lado, se han aumentado los valores de *layer_sizes* (número de dimensiones o nodos en cada capa) y *batch_sizes* (relacionado con el particionado de los datos de entrenamiento en mini lotes para pasarlos por la red).

Por otro lado, se ha tratado el desequilibrio de las clases mediante *compute_class_weight* de *Sklearn* para aplicar las ponderaciones de clase necesarias en conjuntos no balanceados. Se proporciona un diccionario donde las claves son clases y los valores los pesos de la clase correspondiente. Estos valores podrán incorporarse en el momendo del entrenamiento sobre el parámetro *class_weight*. También se ha reducido el valor de ***drop_out***, este valor establece de forma aleatoria las unidades de entrada como 0 con una frecuencia determinada en cada paso durante el entrenamiento. Ayudando así a evitar el sobreajuste, aunque en este caso concreto ha parecido mermar los resultados en esta fase.

Tabla 6-103: Resultados GraphSAGE Dataset 2

GraphSAGE	K= 1		K = 2		K = 3	
	Acc	Loss	Acc	Loss	Acc	Loss
Retweets	0.908	0.228	0.955	0.095	0.943	0.167
Subconjunto Dataset 1 Retweets	0.806	0.400	0.875	0.294	0.844	0.459
Subconjunto Hashtag Retweets	0.816	0.415	0.898	0.283	0.836	0.345
Replies	0.876	0.300	0.929	0.227	0.898	0.302
Subconjunto Dataset 1 Replies	0.791	0.405	0.878	0.270	0.810	0.554
Subconjunto Hashtag Replies	0.666	0.609	0.754	0.567	0.587	1.113

De las acciones y pruebas descritas la que mejor resultado ha proporcionado es el ajuste de las clases. Por ello, tras aplicarlo los valores de *drop_out* se volvieron a poner en 0.4, 0.45, casi los iniciales (0.5). La Tabla 6.13 presenta los resultados obtenidos con GraphSAGE sobre el Dataset 2, así como las pruebas de inducción de nodos sobre el Dataset 1 y el conjunto extraído a partir del *hashtag* "#StopVacunas" etiquetado con ayuda de Botometer.

En las siguientes subsecciones se aportarán el resto de métricas y detalles sobre los resultados generados. En todos los casos se ha utilizado un *batch_size* de 256, *layer_size* en cada caso de 64 y como agregador *MaxPool*.

6.4.1. Contexto en base a Retweets

En esta sección se desglosan los resultados obtenidos en base al parámetro de muestreo de vecindarios ***k***. En el caso de los retweets, se han entrenado los modelos empleando un 70% de los datos, ya que se cuentan con más de 10.000 nodos y 100.000 enlaces para realizar el entrenamiento y el restante 30% para validación.

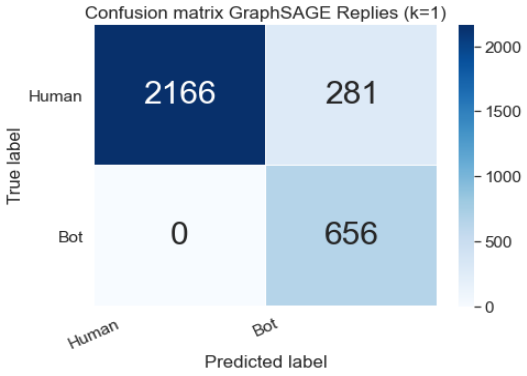
- **K=1,** se ha utilizado un muestreo *in_samples* 15 y out_*samples* 10. Las métricas arrojadas por el modelo son las siguientes (Figura 6-25):

Figura 6-95: Métricas de clasificación GraphSAGE, conjunto retweets (k=1)

	precision	recall	f1-score	support
0	1.00	0.89	0.94	2447
1	0.70	1.00	0.82	656
accuracy			0.91	3103
macro avg	0.85	0.94	0.88	3103
weighted avg	0.94	0.91	0.91	3103

Dado que la precisión para los humanos y *recall* para los bots es de 1 en cada caso, se observa el *f1-score*, con mejores valores para la clasificación de cuentas humanas que bots. Según la matriz de confusión, los bots han sido etiquetados de forma correcta, por el contrario, se han clasificado 281 cuentas humanas erróneamente, como se puede observar en la *Figura 6-26*.

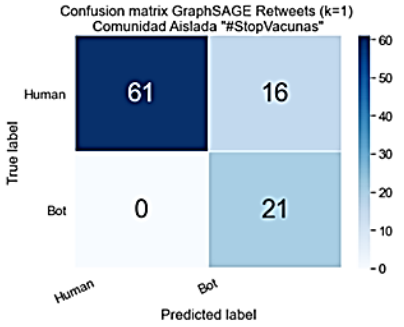
Figura 6-106: Matriz de confusión



Seguidamente se exponen las métricas y matriz de confusión de la prueba de inducción sobre las cuentas etiquetadas con ayuda de Botometer y extraídas a partir del hashtag "#StopVacunas", que ha sido el que ha generado mejores resultados:

Figura 6-117: Métricas y Matriz de confusión GraphSAGE "StopVacunas" conjunto Retweets (k=1)

	precision	recall	f1-score	support
0	1.00	0.79	0.88	77
1	0.57	1.00	0.72	21
accuracy			0.84	98
macro avg	0.78	0.90	0.80	98
weighted avg	0.91	0.84	0.85	98



A pesar de perder rendimiento ofrece resultados aceptables, manteniendo la misma dinámica en el acierto sobre las cuentas bots en estos nuevos datos, clasificando correctamente el total de perfiles bots.

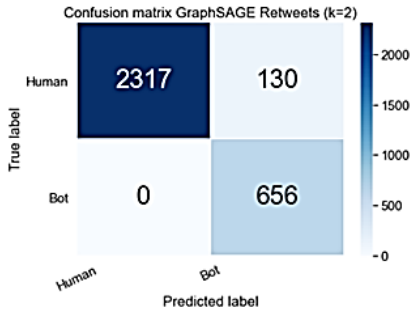
- **K=2**, este ha sido el modelo que ha generado los valores más precisos de todas las pruebas. Además, atendiendo a las relaciones del grafo, se han probado valores de exploración en los nodos *target* de [20, 20] aunque los resultados han sido muy similares a los expuestos. Se ha completado dicha prueba ya que los valores medios del grado de los nodos son bajos (10), pero el rango máximo de algunos nodos *source* tienen valores de 66 enlaces y los de *target* de hasta 2167. Se quería observar si ampliando la exploración en estos últimos nodos se mejoraba el resultado, pero no ha sido así. Finalmente, los valores de muestreo son:

    - *in_samples* 13,13,13
    - *out_samples* 9,9,9

    En la siguiente figura se exponen las métricas y matriz de confusión obtenidas:

Figura 6-28: Métricas y Matriz de confusión GraphSAGE Dataset 2 conjunto Retweets (k=2)

	precision	recall	f1-score	support
0	1.00	0.95	0.97	2447
1	0.83	1.00	0.91	656
accuracy			0.96	3103
macro avg	0.92	0.97	0.94	3103
weighted avg	0.97	0.96	0.96	3103



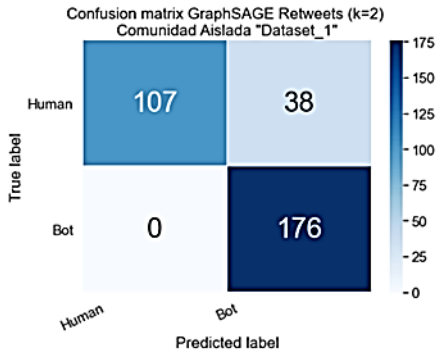
Se experimentan resultados algo parecidos a los expuestos para retweets con k=1. El valor de *recall* para los bot es muy bueno, ya que no se clasifican como falsos bot, sin embargo, si lo hace con los humanos.

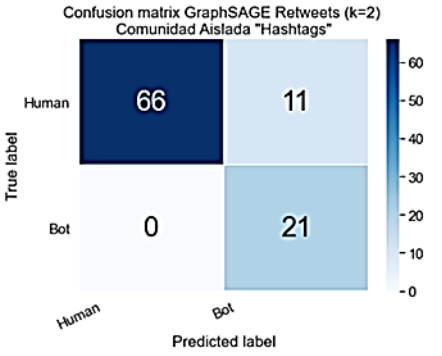
Ahora, se compararán los resultados encontrados para la comunidad aislada del Dataset 1 (izquierda) y conjunto StopVacunas (derecha):

Figura 6-29: Métricas y Matriz de confusión Dataset 1 (izquierda) y StopVacunas (derecha), GraphSAGE Retweets (k=2).

	precision	recall	f1-score	support
0	1.00	0.74	0.85	145
1	0.82	1.00	0.90	176
accuracy			0.88	321
macro avg	0.91	0.87	0.88	321
weighted avg	0.90	0.88	0.88	321

	precision	recall	f1-score	support
0	1.00	0.86	0.92	77
1	0.66	1.00	0.79	21
accuracy			0.89	98
macro avg	0.83	0.93	0.86	98
weighted avg	0.93	0.89	0.90	98





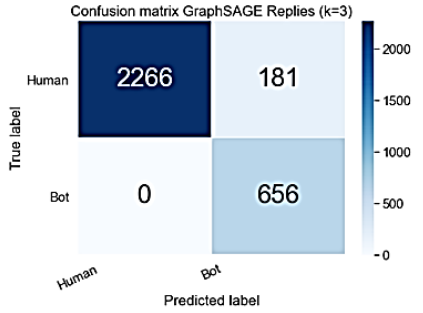
En ambos casos se han clasificado adecuadamente las cuentas bot, generando valores de recall o sensibilidad casi perfectos, pero clasificando erróneamente como bot cuentas humanas. **Se destaca que sí se han conseguidos buenos resultados sobre datos que no se habían visto en el entrenamiento, a diferencia del resto de modelos tradicionales probados.**

- **K=3,** se han utilizado los siguientes valores para el muestreo de vecindarios de cada nodo:
    - *in_samples* 13,13,13
    - *out_samples* 8,8,8

A continuación, se presenta la matriz de confusión y las métricas obtenidas para este modelo:

Figura 6-120: Métricas y Matriz de confusión GraphSAGE Dataset 2 conjunto Retweets (k=3)

	precision	recall	f1-score	support
0	1.00	0.93	0.96	2447
1	0.78	1.00	0.88	656
accuracy			0.94	3103
macro avg	0.89	0.96	0.92	3103
weighted avg	0.95	0.94	0.94	3103



Explorando el conjunto de retweets con una profundidad k=3 se observa, como en los dos casos anteriores, valores perfectos de recall para los bots y precisión en los humanos. Ha clasificado de forma correcta todas las cuentas automatizadas bots, pero etiquetando 181 falsos bots, cuentas de humanos mal etiquetadas. No obstante, los valores de f1-score son ligeramente mejores en la etiqueta de los humanos por la relación de los datos.

En última instancia se comparan las métricas obtenidas para la prueba de inducción sobre el Dataset 1 (izquierda) y sobre el conjunto StopVacunas (derecha), que ha sido la que mejores resultados ofrece:

Figura 6-131: Métricas Dataset 1 (izquierda) y StopVacunas (derecha) Retweets (k=3)

	precision	recall	f1-score	support
0	1.00	0.65	0.79	145
1	0.78	1.00	0.87	176
accuracy			0.84	321
macro avg	0.89	0.82	0.83	321
weighted avg	0.88	0.84	0.83	321

	precision	recall	f1-score	support
0	1.00	0.83	0.91	77
1	0.62	1.00	0.76	21
accuracy			0.87	98
macro avg	0.81	0.92	0.84	98
weighted avg	0.92	0.87	0.88	98

Los resultados son bastante parecidos, aunque los scores son mejores en el Dataset 1. En ambos conjuntos se pierde precisión respecto al modelo original, en torno al 10-11 % pero funcionan mucho mejor que cualquiera de los algoritmos tradicionales explorados. Por otro lado, se mantiene la dinámica vista, clasifica mejor las cuentas bots que las legítimas.

Se puede **concluir esta sección** afirmando que el algoritmo GraphSAGE es **capaz de predecir cuentas automatizadas bot en entornos con usuarios que no se habían mostrado antes al modelo, probando tanto en el subconjunto del Dataset 1 como en los usuarios extraídos del empleo del hashtag StopVacunas con resultados aceptables en ambos casos.**

Son modelos bastante precisos, aunque todos siguen un cierto patrón, no clasifican falsos bots, pero si fallan en las cuentas humanas. Estos falsos positivos (clasificar como bots cuentas legítimas) se podrían resolver a través de algún método complementario; por ejemplo, se podría pedir los usuarios "sospechosos" que verifiquen su legitimidad a través de un captcha, o sistema similar. Por otra parte, al tratarse de un pequeño sesgo y aparentemente estable, sería sencillo tenerlo en cuenta a la hora de hacer análisis sobre la difusión de noticias en RR.SS.

Así mismo, se destaca la importancia de definir bien los valores de muestreo ya que los tiempos requeridos para k=3 han sido de magnitudes 7 y 8 veces superiores a k=2 en estos ejemplos. Se muestran los costes de ejecución en tiempo y memoria según el artículo [2], donde n es el número de nodos, k el

número de capas, *s* el tamaño del lote, *r* el número de vecinos muestreados por cada nodo y *d* las dimensiones de las características ocultas del nodo. En graphSAGE no es necesario incluir el número de las conexiones del grafo para este cálculo.

$$\textbf{Tiempo: } \boldsymbol{O(r^k nd^2)}$$

$$\textbf{Memoria: } \boldsymbol{O(sr^k d + kd^2)}$$

A partir de estos resultados se puede inferir que para el conjunto de datos analizados el punto óptimo es k=2, donde se tiene una buena precisión sin incurrir en grandes costes computacionales.

6.4.2. Contexto en base a Replies

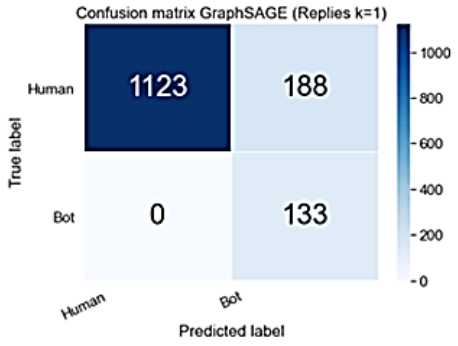
Seguidamente se presentan los resultados obtenidos en base al parámetro de muestreo ***k*** para los conjuntos de datos basados en replies. En este caso se ha utilizado un 80% de los datos para entrenamiento y el restante 20% para validación.

- **K=1,** empleando los siguientes valores para muestreo:
    - *in_samples* 15
    - *out_samples* 10.

En la *Figura 6-32* se pueden encontrar las métricas resultantes y la matriz de confusión. La precisión registrada para los bots es baja, 0.41 frente a la mostrada por los usuarios legítimos, 1.00. Dado que el valor de *recall* para los bots es perfecto habrá que fijarse en *f1-sore,* en el que se ven mejores resultados para la etiqueta humana.

Figura 6-142: Métricas y Matriz de confusión GraphSAGE Dataset 2 conjunto Replies

	precision	recall	f1-score	support
0	1.00	0.86	0.92	1311
1	0.41	1.00	0.59	133
accuracy			0.87	1444
macro avg	0.71	0.93	0.75	1444
weighted avg	0.95	0.87	0.89	1444



En la matriz de confusión se observa que se han clasificado correctamente todos los bots, de manera que los fallos se han encontrado en los humanos, clasificando erróneamente 188 frente a 1123 casos correctos.

A continuación, se presentan las métricas obtenidas sobre la comunidad aislada del Dataset 1 (izquierda) y conjunto StopVacunas (derecha):

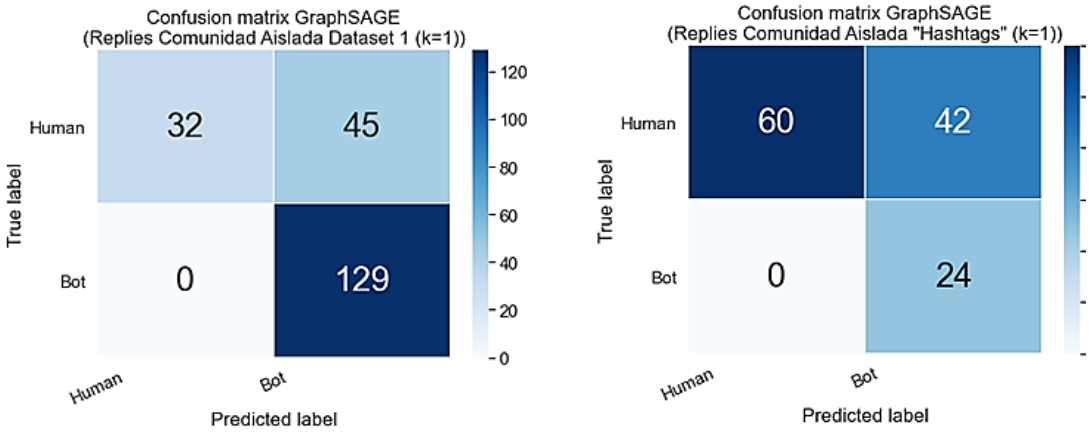
Figura 6-153: Métricas Dataset1 (izquierda) y StopVacunas (derecha). GraphSAGE, Replies (k=1)

	precision	recall	f1-score	support
0	1.00	0.42	0.59	77
1	0.74	1.00	0.85	129
accuracy			0.78	206
macro avg	0.87	0.71	0.72	206
weighted avg	0.84	0.78	0.75	206

	precision	recall	f1-score	support
0	1.00	0.59	0.74	102
1	0.36	1.00	0.53	24
accuracy			0.67	126
macro avg	0.68	0.79	0.64	126
weighted avg	0.88	0.67	0.70	126

Las cifras arrojan peores resultados para el conjunto StopVacunas, pero atendiendo a las matrices de confusión éstos pueden ser por la distribución de los datos. Es decir, para del Dataset 1 se llegan a clasificar incorrectamente 45 cuentas legítimas frente a sólo 32 bien, mientras que para StopVacunas se clasifican correctamente 60 cuentas humanas y solo 42 mal. En los dos casos acierta con el total de cuentas bots, pero en el primer ejemplo existen 129 casos y en StopVacunas sólo 24. En la siguiente figura se exponen las matrices de confusión de los dos ejemplos descritos:

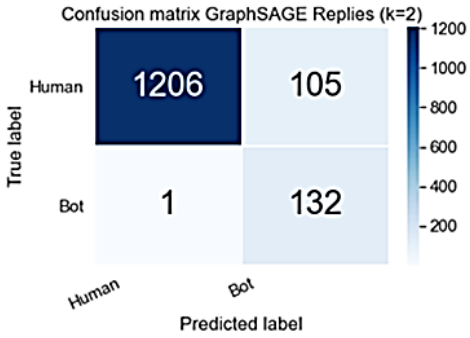
Figura 6-164: Matriz de confusión Dataset1 (izquierda) y StopVacunas (derecha). GraphSAGE, Replies (k=1)



- **K=2,** con valores exploratorios de vecindario: *in_samples* (15,15) y *out_samples* (10,10). En la siguiente figura se pueden observar las métricasy matriz de confusión obtenidas para este modelo:

Figura 6 35: Métricas y Matriz de confusión GraphSAGE, Replies (k=2)

	precision	recall	f1-score	support
0	1.00	0.92	0.96	1311
1	0.56	0.99	0.71	133
accuracy			0.93	1444
macro avg	0.78	0.96	0.84	1444
weighted avg	0.96	0.93	0.94	1444



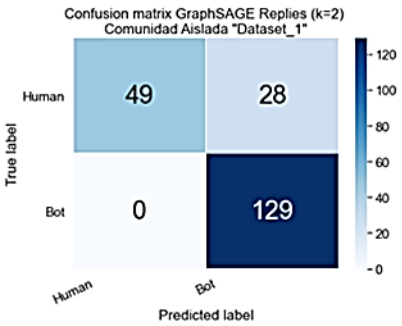
Este es el modelo que mejores resultados ha ofrecido para el conjunto de datos relacionados con la acción de replies de los usuarios. Los valores de precisión son algo peores para la categoría de los bots frente a la de humanos. Clasifica adecuadamente 1206 cuentas legítimas frente a 105 incorrectas, y 132 bots correctamente frente a sólo 1 mal.

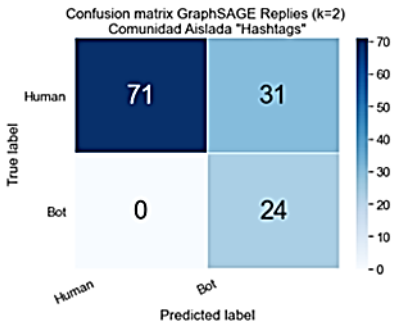
Seguidamente se presentan las métricas y matrices de confusión obtenidas para las comunidades aisladas, a la izquierda para el Dataset 1 y a la derecha para el conjunto extraído en base a hashtag.

Figura 6-36: Métricas y Matriz de confusión Dataset 1 (izquierda) y StopVacunas (derecha), GraphSAGE Replies (k=2)

	precision	recall	f1-score	support
0	1.00	0.64	0.78	77
1	0.82	1.00	0.90	129
accuracy			0.86	206
macro avg	0.91	0.82	0.84	206
weighted avg	0.89	0.86	0.86	206

	precision	recall	f1-score	support
0	1.00	0.70	0.82	102
1	0.44	1.00	0.61	24
accuracy			0.75	126
macro avg	0.72	0.85	0.71	126
weighted avg	0.89	0.75	0.78	126





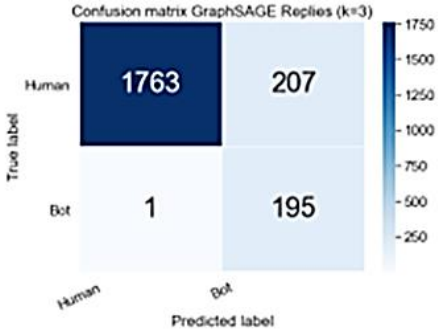
Las principales diferencias encontradas en estos dos casos es el valor de precisión en cuanto a la clasificación de cuentas bots, mucho menor sobre la comunidad del conjunto StopVacunas que en el Dataset 1.

- **K=3**, para el que se ha empleado la siguiente configuración en la exploración de vecindarios de los nodos:
    - *in_samples* 10,10,10
    - *out_samples* 8,8,8

Seguidamente se exponen las métricas y matriz de confusión obtenidas:

Figura 6-37: Métricas y Matriz de confusión GraphSAGE, replies (k=3)

	precision	recall	f1-score	support
0	1.00	0.89	0.94	1970
1	0.49	0.99	0.65	196
accuracy			0.90	2166
macro avg	0.74	0.94	0.80	2166
weighted avg	0.95	0.90	0.92	2166



En este ejemplo de graphSAGE sobre el grafo de replies, se han clasificado 1763 cuentas legítimas de manera correcta frente a 207 mal, y un total de 195 bot adecuadamente frente a sólo 1 incorrecta. Así, los valores de precisión para los humanos y *recall* para la etiqueta de bots registran valores casi perfectos.

A continuación, se exponen las figuras con las métricas para las comunidades aisladas, a la derecha el Dataset 1 y a la izquierda el grupo StopVacunas:

Figura 6-178: Métricas Dataset1 (izquierda) y StopVacunas (derecha). GraphSAGE, replies (k=3)

	precision	recall	f1-score	support
0	1.00	0.49	0.66	77
1	0.77	1.00	0.87	129
accuracy			0.81	206
macro avg	0.88	0.75	0.76	206
weighted avg	0.85	0.81	0.79	206

	precision	recall	f1-score	support
0	1.00	0.42	0.59	102
1	0.29	1.00	0.45	24
accuracy			0.53	126
macro avg	0.64	0.71	0.52	126
weighted avg	0.86	0.53	0.57	126

En este caso sí se observan diferencias en los resultados generados, ya que el valor *accuracy* es 22 puntos menor en el caso del conjunto de datos extraído mediante hashtag, un 0.81 frente a 0.58. En la métrica anterior se reflejan valores de precisión mucho más bajos para la clasificación de cuentas bots.

Finalmente, atendiendo a los resultados expuestos en la *Tabla 6-13*, los mejores resultados se han obtenido en el modelo con muestreo de vecindarios k=2 en ambos conjuntos. Vecindarios de un solo nivel no son capaces de explorar en detalle el vecindario del nodo y hacen funcionar la red como un simple MLP. Por el contrario, muestreos de vecindario más profundos afectan a las incrustaciones de los demás nodos diluyendo la del nodo objetivo, empeorando ligeramente los resultados y requiriendo una notable mayor capacidad de cómputo.

Sería complicado hacer una comparación sobre las relaciones basadas en retweets y replies ya que las diferencias en el volumen de las muestras pueden influir en los datos arrojados por los modelos. En la siguiente tabla se muestra el número útil de nodos y enlaces en cada conjunto.

Tabla 6-114: Diferencias en el volumen de las muestras

Estadísticas	Dataset 1		Dataset 2		StopVacunas	
	Retweets	Replies	Retweets	Replies	Retweets	Replies
Nodos	321	206	10340	7220	98	126
Enlaces	7922	719	106294	43144	295	620

No obstante, con los resultados obtenidos se puede concluir que las relaciones de retweets son mejores en cuanto a la clasificación de cuentas bots. Los modelos asumen más semejanza en las clasificaciones ofrecidas sobre el Dataset1 y StopVacunas en el caso de retweets, pues para replies el conjunto generado mediante hashtag ofrece significativamente peores cifras en todos los modelos.

Este resultado tiene sentido con la interpretación de la semántica de ambas acciones: mientras que un retweet es claramente un apoyo o aprobación del tweet correspondiente, un replie puede representar un apoyo, pero también una respuesta contraria. **Por lo tanto, la relación de retweet sería más consistente y permitiría obtener mejores conclusiones**.

7. Conclusiones

A lo largo del desarrollo de este trabajo se ha examinado la posibilidad de detectar cuentas automatizadas bot en Twitter mediante el empleo de redes convolucionales basadas en grafos. Es decir, se pretendía entender las oportunidades que esta tecnología, relativamente novedosa, puede aportar al análisis de

RR.SS., así como encontrar las posibles limitaciones de su aplicación. La intuición indicaba que un método que pudiera tener en cuenta las relaciones de un nodo con los demás (el contexto) podría generar mejores clasificadores que uno que sólo tuviera en cuenta la información de nodos individuales. El objetivo era analizar su uso en contexto: considerar desde la obtención de datos necesarios para su utilización hasta el tipo de conclusiones que se podría extraer al final del proceso. Con estas premisas, se exploraron los diferentes grafos que se pueden construir a partir de la información de una red social como Twitter. Así, se puede construir un grafo basado en las relaciones de seguimiento (seguidores/seguidos), en base a los retweets o en base los replies, entre otros. En cada caso, al grafo resultado fue considerado como la entrada a un algoritmo de clasificación que fuera capaz de diferenciar bots de perfiles legítimos usando la estructura del grafo, además de la información propia de cada perfil.

Por supuesto, un enfoque basado en análisis de datos presupone la disponibilidad de grandes conjuntos de datos, preferentemente etiquetados y necesariamente fiables. Este requisito resultó, a la postre, más difícil de satisfacer de lo que originalmente se había supuesto. El contexto de interés era la denominada inteligencia de fuentes abiertas (OSINT) y por ese motivo se insistió en trabajar en datos disponibles públicamente. En este punto es interesante destacar que, aunque la gran mayoría de los datos disponibles en Twitter se consideran públicos y gratuitos, en la práctica Twitter pone limitaciones a la obtención de grandes volúmenes de dichos datos. De hecho, las condiciones para hacer fueron cambiadas por Twitter durante el desarrollo del proyecto, lo que entre otras cosas imposibilitó el uso previsto de la herramienta Twint para la descarga de redes de seguidores. A pesar de esto, Twint fue la herramienta que mejores resultados dio para la descarga de grandes volúmenes de datos respecto de otros aspectos de los perfiles públicos en Twitter.

Respecto del interrogante principal de esta investigación, se puede afirmar que los clasificadores que combinan los atributos de nodo y las relaciones de los usuarios expuestas en grafos **aportan mayor precisión al inducir datos que no han sido vistos** durante el entrenamiento que los modelos tradicionales. Se pueden utilizar en la clasificación de cuentas no vistas hasta el momento sin necesidad de reentrenar el modelo gracias a sus capacidades inductivas. No obstante, será obligado descargar todos los datos necesarios, así como el procesado y extensión de grafos y atributos, ya que se requiere de toda la información para la clasificación (el grafo del modelo original concatenado a los nuevos datos a clasificar).

También se puede afirmar que el muestreo de vecindarios profundos no aporta mejores resultados en la clasificación, y por el contrario los tiempos de cómputo crecen de manera exponencial. Así, según las pruebas realizadas, el punto óptimo consiste en utilizar 2 niveles de relacionamientos (K=2) para los análisis.

En cuanto a las relaciones estudiadas sobre los usuarios, es complicado establecer un criterio definitivo sobre cuál de las dos (redes de retweets vs. redes de replies) resulta más certera para clasificar bot, ya que el volumen de datos utilizado por los modelos ha sido dispar. Sin embargo, los resultados parecen apoyar la idea de que los retweets tienen una semántica más clara y por lo tanto ofrecen un mejor criterio de clasificación.

Por otra parte, es interesante considerar cómo se podrían utilizar los resultados obtenidos en un contexto práctico. Es decir, ¿cómo podría mejorar la lucha contra las fakenews su tuviésemos capacidad de detectar bots automáticamente y a gran escala? ¿Afecta el porcentaje de error estimado en los clasificadores dicho uso?

Una primera solución sería construir aplicaciones de lectura (clientes Twitter) que filtraran o al menos advirtieran sobre aquellas publicaciones cuya difusión está ampliada mediante el uso de bots. En

este sentido, se podría asociar un porcentaje de verosimilitud a cada publicación de acuerdo al porcentaje de bots supuestamente detectados en su difusión.

Otra solución sería denunciar a la plataforma Twitter aquellos perfiles que consideramos bots y que están siendo utilizados para difundir información. Aquí, para evitar denunciar a usuarios legítimos, se podría considerar un paso previo en donde le pedimos al perfil que confirme su legitimidad, por ejemplo, a través del uso de Captchas.

Una tercera alternativa es, al hacer análisis estadísticos sobre el alcance de determinadas publicaciones, corregir los datos obtenidos para descontar el efecto de los supuestos bots, considerando para ello que la propia predicción tiene una probabilidad de error asociada.

Finalmente hay que destacar que el campo está en constante evolución. En particular, las cuentas automatizadas evolucionan día a día para completar sus fines sin ser descubiertas. Por ello, es probable que no todos los resultados mostrados en este trabajo puedan ser extrapolados a cualquier conjunto de datos y usuarios. En particular, es posible que el conjunto de atributos a utilizar para la clasificación automática varíe rápidamente en el tiempo. Sin embargo, consideramos que aquellas propiedades distintivas relacionadas con el contexto de cada nodo son más estables y difíciles de simular por un bot, por lo que se cree que el núcleo de las conclusiones obtenidas si es generalizable y tendrá cierta estabilidad en el tiempo.

8. Referencias

[1] A. Sanzgiri, A. Hughes and S. Upadhyaya, "Analysis of Malware Propagation in Twitter," *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, 2013, pp. 195-204, doi: 10.1109/SRDS.2013.28.

[2] Wu Z, Pan S, Chen F, et al. A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems. 2021 Jan;32(1):4-24. DOI: 10.1109/tnnls.2020.2978386.

[3] Haustein, Stefanie & Bowman, Timothy & Holmberg, Kim & Tsou, Andrew & Sugimoto, Cassidy & Larivière, Vincent. (2014). Tweets as Impact Indicators: Examining the Implications of Automated "bot" Accounts on Twitter. Journal of the Association for Information Science and Technology. 67. 10.1002/asi.23456.

[4] Luceri, L., Giordano, S., & Ferrara, E. (2020). Detecting Troll Behavior via Inverse Reinforcement Learning: A Case Study of Russian Trolls in the 2016 US Election. Proceedings of the International AAAI Conference on Web and Social Media, 14(1), 417-427. Retrieved from https://ojs.aaai.org/index.php/ICWSM/article/view/7311

[5] Javier Pastor-Galindo, Mattia Zago, Pantaleone Nespoli, Sergio López Bernal, José A. Ruipérez Valiente, Alberto Huertas Celdrán, Manuel Gil Pérez, Gregorio Martínez Pérez, Félix Gómez Mármol, «Spotting political social bots in Twitter: A use case of the 2019 Spanish general election«, Special Issue on Data Analytics and Machine Learning for Network and Service Management, IEEE Transactions on Network and Service Management, 2020

[6] Samper-Escalante, Luis D., Octavio Loyola-González, Raúl Monroy, and Miguel A. Medina-Pérez 2021. "Bot Datasets on Twitter: Analysis and Challenges" Applied Sciences 11, no. 9: 4105. https://doi.org/10.3390/app11094105

[7] Varol, O.; Ferrara, E.; Davis, C.; Menczer, F.; Flammini, A. Online Human-Bot Interactions: Detection, Estimation, and Characterization. In Proceedings of the 2017 Eleventh International AAAI Conference on Web and Social Media (ICWSM'17), Montréal, QC, Canada, 15–18 May 2017; pp. 280–289.

[8] Twitter reveals its daily active user numbers for the first time. [En línea]. Disponible: https://www.washingtonpost.com/technology/2019/02/07/twitter-reveals-its-daily-active-user-numbers-first-time/. [Último acceso: 1 agosto 2021].

[9] Página principal de Twitter. [En línea]. Disponible: https://twitter.com. [Último acceso: 17 agosto 2021].

[10] Statista GmbH. [En línea]. Disponible: https://es.statista.com. [Último acceso: 5 mayo 2021].

[11] 58 Estadísticas de Twitter. [En línea]. Disponible: https://www.brandwatch.com/es/blog/58-estadisticas-twitter. [Último acceso: 5 mayo 2021].

[12] ¿Bot o no? Los hechos sobre la manipulación de la plataforma en Twitter. [En línea]. Disponible: https://blog.twitter.com/es_la/topics/company/2020/Bot-o-no-Los-hechos-sobre-la-manipulacion-de-la-plataforma-en-Twitter. [Último acceso: 10 mayo 2021].

[13] Chu, Zi & Gianvecchio, Steven & Wang, Haining & Jajodia, Sushil. (2010). Who is Tweeting on Twitter: Human, Bot, or Cyborg? Proceedings - Annual Computer Security Applications Conference, ACSAC. 21-30. 10.1145/1920261.1920265.

[14] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M., "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race", 2017.

[15] Procesamiento del Lenguaje Natural (Sociedad Española para el Procesamiento del Lenguaje Natural). [En línea]. Disponible: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/index. [Último acceso: 7 Junio 2021].

[16] Przybyła, Piotr. (2019). Detecting Bot Accounts on Twitter by Measuring Message Predictability.

[17] Pozzana, Iacopo and Emilio Ferrara. "Measuring Bot and Human Behavioral Dynamics." *Frontiers in Physics* (2020).

[18] Botometer, An OSoMe proyect. [En línea]. Disponible: https://botometer.osome.iu.edu. [Último acceso: 20 agosto 2021].

[19] Indiana University NetworkScience Institute. [En línea]. Disponible: https://iuni.iu.edu. [Último acceso: 10 junio 2021].

[20] CNetS, Center for Complex Networks and Systems Research (Indiana University Bloomington). [En línea]. Disponible: https://cnets.indiana.edu. [Último acceso: 10 junio 2021].

[21] Yang, K.-C., Varol, O., Davis, C. A., Ferrara, E., Flammini, A., and Menczer, F., "Arming the public with artificial intelligence to counter social bots", 2019.

[22] Bot Sentinel. [En línea]. Disponible: https://botsentinel.com. [Último acceso: 20 agosto 2021].

[23] Kipf, T. N. and Welling, M., "Semi-Supervised Classification with Graph Convolutional Networks", 2016.

[24] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y., "Graph Attention Networks".

[25] William L. Hamilton, Rex Ying, Jure Leskovec (2017). Inductive Representation Learning on Large Graphs. Publication: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems December 2017 Pages 1025–1035

[26] Zhou, J., "Graph Neural Networks: A Review of Methods and Applications", 2018.

[27] Wirth, R. & Hipp, Jochen. (2000). CRISP-DM: Towards a standard process model for data mining. Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining.

[28] Rodríguez Montequín, M. T., Álvarez Cabal, J. V., Mesa Fernández, J. M., González, V. (2003). Metodologías para la realización de proyectos de Data Mining. VII Congreso Internacional de Ingeniería de Proyectos, 2003 Pamplona, pp 0257-0265

[29] Python. [En línea]. Disponible: https://www.python.org. [Último acceso: 10 Julio 2021].

[30] Anaconda (Data science tecnology for human sensemaking). [En línea]. Disponible: https://www.anaconda.com. [Último acceso: 25 mayo 2021].

[31] Jupyter Notebook. [En línea]. Disponible: https://jupyter.org. [Último acceso: 25 mayo 2021]

[32] Tensorflow, Plataforma de extremo a extremo de código abierto para aprendizaje automático. [En línea]. Disponible: https://www.tensorflow.org/?hl=es-419. [Último acceso: 25 agosto julio 2021].

[33] Keras. [En línea]. Disponible: https://keras.io. [Último acceso: 25 agosto julio 2021].

[34] Librería Stellargraph. [En línea]. Disponible: https://www.stellargraph.io. [Último acceso: 1 septiembre 2021].

[35] Pandas. [En línea]. Disponible: https://pandas.pydata.org. [Último acceso: 10 agosto 2021].

[36] Gephi (The Open Graph Viz Platform. [En línea]. Disponible: https://gephi.org. [Último acceso: 25 julio 2021].

[37] Tweepy (An easy-to-use Python library for accessing the Twitter API). [En línea]. Disponible: https://www.tweepy.org. [Último acceso: 1 julio 2021].

[38] Twitter, Rate limitis Developer Platform. [En línea]. Disponible: https://developer.twitter.com/en/docs/twitter-api/v1/rate-limits. [Último acceso: 26 junio 2021].

[39] Twint Project. [En línea]. Disponible: https://github.com/twintproject. [Último acceso: 26 junio 2021].

[40] Scweet (A simple and unlimited twitter scraper with Python). [En línea]. Disponible: https://github.com/Altimis/Scweet. [Último acceso: 26 junio 2021].

[41] Selenium automates browsers. [En línea]. Disponible: https://www.selenium.dev. [Último acceso: 15 junio 2021].

[42] Bot Repository Dataset Botometer. [En línea]. Disponible: https://botometer.osome.iu.edu/bot-repository/datasets.html. [Último acceso: 15 julio 2021].

[43] Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., and Tesconi, M., "RTbust: Exploiting Temporal Patterns for Botnet Detection on Twitter", 2019.

[44] Twitter Bot Accounts. [En línea]. Disponible: https://www.kaggle.com/davidmartngutirrez/twitter-bots-accounts?select=twitter_human_bots_dataset.csv. [Último acceso: 1 agosto 2021].

[45] Cierre de la versión Legacy Mobile Twitter el 15 de diciembre 2020. [En línea]. Disponible: https://www.reddit.com/r/Twitter/comments/k28b6w/legacy_mobile_twitter_version_will_shut_down_on. [Último acceso: 10 mayo 2021].

[46] Twitter Help Center, Navegadores Soportados por twitter.com. [En línea]. Disponible: https://help.twitter.com/en/using-twitter/twitter-supported-browsers. [Último acceso: 7 mayo 2021].

[47] Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., and Murphy, K., "Machine Learning on Graphs: A Model and Comprehensive Taxonomy", 2020.

[48] Haustein, Stefanie & Bowman, Timothy & Holmberg, Kim & Tsou, Andrew & Sugimoto, Cassidy & Larivière, Vincent. (2014). Tweets as impact indicators: Examining the implications of automated "bot" accounts on Twitter. Journal of the Association for Information Science and Technology. 67. 10.1002/asi.23456.

[49] Kosmajac, Dijana and Keselj, Vlado (2019). Twitter Bot Detection using Diversity Measures. Association for Computational Linguistics (sep 2019)

[50] Antenore, M., Camacho-Rodriguez, J. M., and Panizzi, E., "A comparative study of Bot Detection techniques methods with an application related to Covid-19 discourse on Twitter", 2021.

[51] Word2vec Mikolov, T., Chen, K., Corrado, G., and Dean, J., "Efficient Estimation of Word Representations in Vector Space", 2013.

[52] Node2Vec: Scalable Feature Learning for Networks. A. Grover, J. Leskovec. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

[53] Randomwalk: Xia, F., Liu, J., Nie, H., Fu, Y., Wan, L., and Kong, X., "Random Walks: A Review of Algorithms and Applications", 2020.

[54] Graph2vec: Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S., "graph2vec: Learning Distributed Representations of Graphs", 2017.

[55] Bihari, Anand & Pandia, Manoj. (2015). Eigenvector centrality and its application in research professionals' relationship network. 2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, ABLAZE 2015. 10.1109/ABLAZE.2015.7154915.

[56] Understanding Graph Convolutional Networks for Node Classification. [En línea]. Disponible: https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b. [Último acceso: 20 Agosto 2021].

[57] Scikit Learn, Random Forest Classifier. [En línea]. Disponible: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Último acceso: 18 Julio 2021].

[58] Evgeniou, Theodoros & Pontil, Massimiliano. (2001). Support Vector Machines: Theory and Applications. 2049. 249-257. 10.1007/3-540-44673-7_12MLP

[59] E. Wilson and D. W. Tufts, "Multilayer perceptron design algorithm," Proceedings of IEEE Workshop on Neural Networks for Signal Processing, 1994, pp. 61-68, doi: 10.1109/NNSP.1994.366063.

[60] Hassan, Hassan & Negm, Abdelazim & Zahran, Mohamed & Saavedra, Oliver. (2015). ASSESSMENT OF ARTIFICIAL NEURAL NETWORK FOR BATHYMETRY ESTIMATION USING HIGH RESOLUTION SATELLITE IMAGERY IN SHALLOW LAKES: CASE STUDY EL BURULLUS LAKE. International Water Technology Journal. 5.

[61] Wu, F., Zhang, T., Holanda de Souza, A., Fifty, C., Yu, T., and Weinberger, K. Q., "Simplifying Graph Convolutional Networks", 2019

[62] Do we need Deep graph neural networks. [En línea]. Disponible: https://towardsdatascience.com/do-we-need-deep-graph-neural-networks-be62d3ec5c59. [Último acceso: 25 Ago 2021].

[63] K. Zhou et al. Effective training strategies for deep graph neural networks (2020). arXiv:2006.07107.)

[64] Y. Rong et al. DropEdge: Towards deep graph convolutional networks on node classification (2020). In Proc. ICLR. An idea similar to DropOut where a random subset of edges is used during training.

[65] GridSeachCV. [En línea]. Disponible https://scikit-learn.org/stable/modules/grid_search.html. [Último acceso: 25 Ago 2021].

[66] Scikit-Learn [En línea]. Disponible: https://scikit-learn.org/stable. [Último acceso: 28 Agosto 2021].

[67] Weisberg, Sanford. (2001). Yeo-Johnson Power Transformations. Weisberg, Stanford. (2001). Yeo-Johnson Power Transformations.

[68] NetworkX. [En línea]. Disponible: https://networkx.org. [Último acceso: 28 agosto 2021].

9. Anexos

Anexo A: Detalles y nomenclatura de Twitter

A continuación, se presenta la información básica que maneja Twitter y su nomenclatura sobre los detalles básicos del perfil de los usuarios y las acciones que pueden realizar:

- **Perfil**:
    - **Nombre de usuario:** Será el nombre de la cuenta del usuario, con la que se registró en la aplicación y con el que se le podrá referenciar de diferentes formas, como precediéndolo a través de @.
    - **Nombre:** Es el nombre real del usuario, o el que éste quiera indicar en la cuenta de Twitter.
    - **Descripción o Biografía:** Se corresponde con el texto o detalles que los usuarios quieran poner sobre sí mismos en sus cuentas. Esta información será visible por cualquier otro usuario de la red y podrá ser pública o privada.
    - **Imagen de perfil**: Los usuarios, si lo desean, podrán incluir fotos personalizadas en su descripción de la cuenta.
    - **Localización**: Es una funcionalidad de Twitter que viene desactivada por defecto, pero se puede incluir la localización geográfica o geolocalización tanto para las cuentas como para incluirlo en tweets.
    - **Media**: reflejará listados en los que se verán los últimos movimientos de los usuarios. Podrán verse los tweets publicados, tweet y respuestas, fotos y videos, así como las publicaciones que hayan sido marcadas "me gusta" por el usuario.

- **Contenido y acciones**:
    - **Tweet**: se corresponde con cada uno de los mensajes publicados. Cada uno de ellos tiene una longitud máxima de 280 caracteres y puede agregar contenido multimedia como fotos, videos, gifs, etc.
    - **Retweet (RT)** 🔁: consiste en la republicación de un tweet ya generado, un usuario puede retuitear sus propios tweets o los de otras personas.
    - **Reply:** Implica responder a un tweet generado por otra persona o sobre un hilo con más mensajes.
    - **Follower**: Se trata de los usuarios que siguen a una cuenta determinada. Así podrán ver las actualizaciones que ésa realice y recibirán alertas con las nuevas publicaciones o generación de contenidos.
    - **Following**: Se referencia así en esta red social a la acción de seguir a otro usuario
    - **Hashtag:** Conjunto de caracteres precedidos por una almohadilla (#) y permiten referenciar un mensaje o términos en este *microblog*, facilitando así la identificación de términos y temáticas distintas.
    - **Time Line (línea de tiempo de un usuario)**: Representa la cronología de actividad realizada por un usuario, tweets y retweets generados, así como las características propias de cada uno de ellos.
    - **Me Gusta**: Es únicamente una característica que pueden utilizar los usuarios sobre los tweet y contenidos generados y puede servir para marcar el nivel de aceptación en la red de la publicación.

Anexo B: Ejemplo del uso de Botometer

Durante el desarrollo del proyecto se ha empleado el servicio de **Botometer** para clasificar algunas cuentas de Twitter que no estaban etiquetadas.

Para poderlo utilizar, una vez instalado se debe disponer de una cuenta para desarrolladores en Twitter (en el *Anexo D* se detalla cómo obtener la cuenta y los accesos), además de registrarse para conseguir una RapidAPI Key disponible en *https://rapidapi.com*

Con todo preparado, se crea un código como el que se muestra a continuación, en el que se generan las autenticaciones y se crea el objeto para poder invocar el servicio de Botometer:

Figura B-1: Ejemplo RapidAPI Key para Botometer

```
import botometer

rapidapi_key = "fabxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx8fbbd"
twitter_app_auth = {
    'consumer_key': "fNinxxxxxxxxxxxxxxxxxxxpTa",
    'consumer_secret': "rHaxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxhbIT",
    }

bom = botometer.Botometer(wait_on_ratelimit=True,
                          rapidapi_key=rapidapi_key,
                          **twitter_app_auth)
```

Con el objeto creado, se invocará el método ***check_account (id)*** con el identificador del usuario que se quiera revisar, obteniendo los siguientes datos. Se marcan en rojo los que realmente se han utilizado en este estudio para la clasificación:

Figura B-2: Ejemplo resultado clasificación Botometer


```
{'cap': {'english': 0.8021481695167405, 'universal': 0.8055586579935085},
 'display_scores': {'english': {'astroturf': 0.2,
   'fake_follower': 2.0,
   'financial': 3.8,
   'other': 1.6,
   'overall': 3.8,
   'self_declared': 0.0,
   'spammer': 0.6},
  'universal': {'astroturf': 0.0,
   'fake_follower': 3.4,
   'financial': 2.4,
   'other': 2.0,
   'overall': 3.4,
   'self_declared': 0.0,
   'spammer': 0.6}},
 'raw_scores': {'english': {'astroturf': 0.04,
   'fake_follower': 0.4,
   'financial': 0.75,
   'other': 0.31,
   'overall': 0.75,
   'self_declared': 0.0,
   'spammer': 0.11},
  'universal': {'astroturf': 0.0,
   'fake_follower': 0.67,
   'financial': 0.48,
   'other': 0.39,
   'overall': 0.67,
   'self_declared': 0.0,
   'spammer': 0.11}},
 'user': {'majority_lang': 'en',
  'user_data': {'id_str': '1732080566', 'screen_name': 'SearestInc'}}}
```

Como puede observarse, Botometer devuelve las puntuaciones para cada una de las clasificaciones de bot, así como la etiqueta de clasificación general específicos para cuentas en lengua inglesa, o para características universales. También genera todas las evaluaciones para scores que van del [0,1] y de [0,5]. Seguidamente se presenta el tipo de bot sobre el que Botometer ofrece clasificación:

- **Fake_follower**: bots específicos que sirven para aumentar el número de seguidores de otras cuentas, aumentando así la importancia o relevancia de las mismas.
- **Financial**: se trata de cuentas que publican empleando **cashtags**, una herramienta que permite búsquedas específicas en sitios oficiales relacionadas con las inversiones, empresas y mercado financiero.
- **Astroturf:** cuentas y bots políticos previamente etiquetados a mano que participan en distintos tipos de campañas generando y eliminando contenidos.
- **Spammer**: identificados como bot que generan contenido spam repetidamente.
- **Self_declared**: bots autoidentificados procedentes de repositorios como *botwiki.org*.
- **Other**: son otro tipo de bot que no figuran en las categorías anteriores. Han sido etiquetados de forma manual en otros estudios o advertidos por los usuarios.

Anexo C: Ejemplo del uso de Tweepy

Tweepy es una biblioteca de Python para acceder a la API oficial de Twitter, por lo que se debe solicitar una cuenta como desarrollador y obtener las claves del token de consumidor y el secreto. Solicitar este tipo de cuentas es un proceso que puede durar varios días, ya que el equipo de Twitter debe analizar la solicitud y dejar claro para qué se van a utilizar los posibles datos descargados y bajo qué ámbito se analizarán, empresarial, académico, etc. Si se les presentan dudas remitirán la solicitud para volver a completar los campos que estimen oportunos.

Una vez se dispone de la cuenta y las claves, se generarán las autenticaciones con el *OAuth*, con el que la aplicación identifica las peticiones y puede generar las respuestas *JSON*. En el siguiente ejemplo se muestra un pequeño código con el que autenticarse y crear el objeto necesario para el empleo de Twint.

Figura C-1: Autenticación en Tweepy

```
access_token = "159xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxipjzNTWE"
access_token_secret = "jFLS8xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxnldeux"
consumer_key = "xxxxxxxxxxxxxxXiebh75xxx"
consumer_secret = "xxxxxxxxxxxxLGodxxxxxxxMXxFat1eXzAxxxxxxxxxP0xxIT"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

Los métodos empleados más comunes son:

- ***GET followers_ids.*** Retorna una lista con los identificadores de los followers de una cuenta.
- ***GET users_show***. Genera información detallada sobre el perfil.
- ***GET lookup_users.*** Retorna los detalles de los usuarios solicitados (100 usuarios por solicitud)
- ***GET user_timeline***. Genera los 20 estados más recientes publicados por el usuario.

Por ejemplo, la información retornada para cualquier tweet generado es similar al reportado por Twint, formado por los siguientes atributos:

Tabla C-1: Identificadores descarga Tweepy

Identificadores descarga Tweepy		
created_at	in_reply_to_status_id	contribuyentes
id	in_reply_to_status_id_str	is_quote_status
id_str	in_reply_to_user_id	retweet_count
full_text	in_reply_to_user_id_str	favorite_cou
truncated	in_reply_to_screen_name	nt
display_text_range	usuario	favotirtos
entidades	geo	retweeted
extended_entities	coordenadas	posiblemente_sensible
'source'	lugar	lang

A continuación, se presenta un ejemplo para descargar el último tweet generado por la cuenta "@UAM_EPS" donde puede verse toda la información recopilada:

Figura C-2: Datos descargados de un tweet con Twint

```
screen_name ="UAM_EPS"
api.user_timeline(screen_name, count=1)
```

```
[Status(_api=<tweepy.api.API object at 0x000002166BE2BFD0>, _json={'created_at': 'Mon Jul 19 13:01:33 +0000 2021', 'id': 1417
7341863002113, 'id_str': '1417107341863002113', 'text': 'Máster Inter-Universitario en Métodos Formales en Ingeniería Informá
ca: https://t.co/2cewrukZNx', 'truncated': False, 'entities': {'hashtags': [], 'symbols': [], 'user_mentions': [], 'urls': [{'
rl': 'https://t.co/2cewrukZNx', 'expanded_url': 'https://www.uam.es/CentroEstudiosPosgrado/MU_Metodos_Formales_en_Ingenieria_I
formatica/1446792411488.htm?language=es_ES&nDept=4&pid=1446755975574&pidDept=1446755975831', 'display_url': 'uam.es/CentroEstu
ios…', 'indices': [74, 97]}]}, 'source': '<a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a>', 'in_reply_
o_status_id': 1417107340453683201, 'in_reply_to_status_id_str': '1417107340453683201', 'in_reply_to_user_id': 1394549807281213
40, 'in_reply_to_user_id_str': '1394549807281213440', 'in_reply_to_screen_name': 'UAM_EPS', 'user': {'id': 13945498072812134
'id_str': '1394549807281213440', 'name': 'UAM EPS', 'screen_name': 'UAM_EPS', 'location': 'Campus Cantoblanco, Madrid', 'descr
ption': 'Twitter oficial de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid', 'url': 'https://t.co/TDW3l
GiH', 'entities': {'url': {'urls': [{'url': 'https://t.co/TDW3l01GiH', 'expanded_url': 'https://www.uam.es/ss/Satellite/Escuel
Politecnica/es/home.htm', 'display_url': 'uam.es/ss/Satellite/E…', 'indices': [0, 23]}]}, 'description': {'urls': []}}, 'prote
ted': False, 'followers_count': 77, 'friends_count': 1, 'listed_count': 0, 'created_at': 'Tue May 18 07:06:41 +0000 2021', 'fa
ourites_count': 0, 'utc_offset': None, 'time_zone': None, 'geo_enabled': False, 'verified': False, 'statuses_count': 11, 'lan
g': None, 'contributors_enabled': False, 'is_translator': False, 'is_translation_enabled': False, 'profile_background_color':
'F5F8FA', 'profile_background_image_url': None, 'profile_background_image_url_https': None, 'profile_background_tile': False,
'profile_image_url': 'http://pbs.twimg.com/profile_images/1394550945590165504/3Rk7rIjY_normal.jpg', 'profile_image_url_https':
'https://pbs.twimg.com/profile_images/1394550945590165504/3Rk7rIjY_normal.jpg', 'profile_link_color': '1DA1F2', 'profile_sideb
r_border_color': 'C0DEED', 'profile_sidebar_fill_color': 'DDEEF6', 'profile_text_color': '333333', 'profile_use_background_ima
e': True, 'has_extended_profile': True, 'default_profile': True, 'default_profile_image': False, 'following': False, 'follow_r
quest_sent': False, 'notifications': False, 'translator_type': 'none', 'withheld_in_countries': []}, 'geo': None, 'coordinate
s': None, 'place': None, 'contributors': None, 'is_quote_status': False, 'retweet_count': 2, 'favorite_count': 1, 'favorited':
False, 'retweeted': False, 'possibly_sensitive': False, 'lang': 'es'}, created_at=datetime.datetime(2021, 7, 19, 13, 1, 33), i
=1417107341863002113, id_str='1417107341863002113', text='Máster Inter-Universitario en Métodos Formales en Ingeniería Informa
ica: https://t.co/2cewrukZNx', truncated=False, entities={'hashtags': [], 'symbols': [], 'user_mentions': [], 'urls': [{'url':
'https://t.co/2cewrukZNx', 'expanded_url': 'https://www.uam.es/CentroEstudiosPosgrado/MU_Metodos_Formales_en_Ingenieria_Infor
tica/1446792411488.htm?language=es_ES&nDept=4&pid=1446755975574&pidDept=1446755975831', 'display_url': 'uam.es/CentroEstudio
s…', 'indices': [74, 97]}]}, source='Twitter Web App', source_url='https://mobile.twitter.com', in_reply_to_status_id=1417107
0453683201, in_reply_to_status_id_str='1417107340453683201', in_reply_to_user_id=1394549807281213440, in_reply_to_user_id_str
='1394549807281213440', in_reply_to_screen_name='UAM_EPS', author=User(_api=<tweepy.api.API object at 0x000002166BE2BFD0>, _js
n={'id': 1394549807281213440, 'id_str': '1394549807281213440', 'name': 'UAM EPS', 'screen_name': 'UAM_EPS', 'location': 'Campu
Cantoblanco, Madrid', 'description': 'Twitter oficial de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid'
'url': 'https://t.co/TDW3l01GiH', 'entities': {'url': {'urls': [{'url': 'https://t.co/TDW3l01GiH', 'expanded_url': 'https://ww
w.uam.es/ss/Satellite/EscuelaPolitecnica/es/home.htm', 'display_url': 'uam.es/ss/Satellite/E…', 'indices': [0, 23]}]}, 'descri
tion': {'urls': []}}, 'protected': False, 'followers_count': 77, 'friends_count': 1, 'listed_count': 0, 'created_at': 'Tue May
18 07:06:41 +0000 2021', 'favourites_count': 0, 'utc_offset': None, 'time_zone': None, 'geo_enabled': False, 'verified': False
'statuses_count': 11, 'lang': None, 'contributors_enabled': False, 'is_translator': False, 'is_translation_enabled': False, 'p
ofile_background_color': 'F5F8FA', 'profile_background_image_url': None, 'profile_background_image_url_https': None, 'profile_
ackground_tile': False, 'profile_image_url': 'http://pbs.twimg.com/profile_images/1394550945590165504/3Rk7rIjY_normal.jpg', 'p
ofile_image_url_https': 'https://pbs.twimg.com/profile_images/1394550945590165504/3Rk7rIjY_normal.jpg', 'profile_link_color':
'1DA1F2', 'profile_sidebar_border_color': 'C0DEED', 'profile_sidebar_fill_color': 'DDEEF6', 'profile_text_color': '333333', 'p
ofile_use_background_image': True, 'has_extended_profile': True, 'default_profile': True, 'default_profile_image': False, 'fol
owing': False, 'follow_request_sent': False, 'notifications': False, 'translator_type': 'none', 'withheld_in_countries': []},
d=1394549807281213440, id_str='1394549807281213440', name='UAM EPS', screen_name='UAM_EPS', location='Campus Cantoblanco, Madr
d', description='Twitter oficial de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid', url='https://t.co/T
W3l01GiH', entities={'url': {'urls': [{'url': 'https://t.co/TDW3l01GiH', 'expanded_url': 'https://www.uam.es/ss/Satellite/Escu
laPolitecnica/es/home.htm', 'display_url': 'uam.es/ss/Satellite/E…', 'indices': [0, 23]}]}, 'description': {'urls': []}}, prot
cted=False, followers_count=77, friends_count=1, listed_count=0, created_at=datetime.datetime(2021, 5, 18, 7, 6, 41), favouri
s_count=0, utc_offset=None, time_zone=None, geo_enabled=False, verified=False, statuses_count=11, lang=None, contributors_enab
ed=False, is_translator=False, is_translation_enabled=False, profile_background_color='F5F8FA', profile_background_image_url=N
ne, profile_background_image_url_https=None, profile_background_tile=False, profile_image_url='http://pbs.twimg.com/profile_im
```

Con Tweepy, únicamente hay que tener presente las limitaciones en cuanto a la descargas, ya que se definen ventanas de 15 minutos para cada una, en las que se podrá capturar distinta información en base al método empleado.

Anexo D: Ejemplo del uso de Twint

Twint es una herramienta escrita en Python que sirve para descargar información de tweets y perfiles de usuario sin emplear la API de Twitter. En cuanto a los tweets, se podrán descargar empleando tres métodos:

- ***twint.run.Search***, que busca los tweet del usuario.
- ***twint.run.Profile***, raspando la línea de tiempo del usuario.
- ***twint.run.Favorites***, descargando los tweet favoritos del usuario.

Con cualquier método de los descritos, será necesario indicar en la configuración de la descarga que se incluyan los retweet si se desea disponer de los mismos en el time_lime generado.

Sobre los usuarios, como ya se ha comentado, no se podrán descargar los seguidos y seguidores (***twint.run.Followers y twint.run.Following***) y para descargar los atributos de una única cuenta se podrá emplear el método ***twint.run.Lookup.***

Para conseguir las descargas de todos los listados de usuarios se han empleado indistintamente los id de usuario como sus nombres, y ha funcionado de manera correcta en ambos casos. Por ejemplo, si se quisiesen extraer los datos del perfil de la cuenta de Twitter de la @EPS_UAM obtendríamos la siguiente información disponible públicamente a través de Twitter:

Figura D-1: Datos descargados de un perfil con Twint

```
1394549807281213440 | UAM EPS | @UAM_EPS | Private: False | Verified: False | Bio: Twitter oficial de la Escuela Politécnica
perior de la Universidad Autónoma de Madrid | Location: Campus Cantoblanco, Madrid | Url: https://t.co/TDW3l01GiH | Joined: 2
1-05-18 07:06:41 UTC | Tweets: 11 | Following: 1 | Followers: 77 | Likes: 0 | Media: 1 | Avatar: https://pbs.twimg.com/profil
images/1394550945590165504/3Rk7rIjY_normal.jpg
```

O descargar el último tweet generado por la cuenta, cuyo detalle se muestra en la *Tabla E-2*. Se destaca que las opciones de *run.Profiles* son más estables que las de *run.Search*, ya que con estas últimas en algunas ocasiones no se ha descargado toda la información disponible.

Tabla D-1: Ejemplo elementos descargados de un Tweet con Twint

Atributos	Valores descargados
id	1,41711E+18
conversation_id	1,41711E+18
created_at	2021-07-19 15:01:33 Hora de verano romance
date	19/07/2021
time	15:01:33
timezone	200
user_id	1,39455E+18
username	uam_eps
name	UAM EPS
place	-
tweet	MÃ¡ster Inter-Universitario en MÃ©todos Formales en IngenierÃa InformÃ¡tica: https://t.co/2cewrukZNx
language	es

mentions	[]
urls	['https://www.uam.es/CentroEstudiosPos-grado/MU_Metodos_Formales_en_Ingenieria_Informatica/1446792411488.htm?language=es_ES&nDept=4&pid=1446755975574&pidDept=1446755975831']
photos	[]
Replies_count	0
Retweets_count	2
Likes_count	1
hashtags	[]
cashtags	[]
link	https://twitter.com/UAM_EPS/status/1417107341863002113
retweet	FALSE
quote_url	-
video	0
thumbnail	-
Near	-
Geo	-
Source	-
User_rt_id	-
User_rt	-
Retweet_id	-
Reply_to	[]
Retweet_date	-
Translate	-
trans_src	-
trans_dest	-

Anexo E: Detalle de los datos y vector de características

A lo largo del estudio, se han generado paulatinamente distintas características sobre los conjuntos de datos, a fin de encontrar las que aporten mayor valor a los modelos manteniendo un esfuerzo razonable en su obtención. En el cuerpo del documento se ha presentado un vector con 55 atributos, pero se llegó a generar uno con 75. En la siguiente tabla se pueden observar cada una de ellas:

Tabla E-1: Características generadas por grupo

PERFILES	TWEETS	RETWEETS
id	**id**	**id**
pr_tweets	tw_longitud_bruta_media_tweet	rt_ratio_reply
pr_following	tw_desv_promedio_long_tweet	rt_ratio
pr_followers	tw_replies_count_media	rt_time_to_rt_mean
pr_likes	tw_retweets_count_media	rt_time_to_rt_desv_tip
pr_media	tw_likes_count_media	rt_time_to_rt_var
pr_private	tw_replies_count_desv_tip	rt_time_to_rt_desv_norm
Pr_verified	tw_retweets_count_desv_tip	rt_replies_count_media
pr_real_name	tw_likes_count_desv_tip	rt_retweets_count_media
pr_real_username	tw_replies_count_var	rt_likes_count_media
pr_join_date	tw_retweets_count_var	rt_replies_count_desv_tip
pr_age_account	tw_likes_count_var	rt_retweets_count_desv_tip
pr_len_name	tw_replies_count_desv_med	rt_likes_count_desv_tip
pr_len_username	tw_retweets_count_des_med	rt_replies_count_var
pr_len_bio	tw_likes_count_desv_med	rt_retweets_count_var
pr_len_url	tw_num_medio_hashtags_por_tweet	rt_likes_count_var
pr_location	tw_num_medio_mentions_por_tweet	rt_replies_count_desv_med
pr_bio	tw_hora_1, tw_hora_2, tw_hora_3, tw_hora_4, tw_hora_5, tw_hora_6	rt_retweets_count_des_med
pr_background_image	tw_hora_7, tw_hora_8, tw_hora_9, tw_hora_10, tw_hora_11, tw_hora_12	rt_likes_count_desv_med
pr_tff_ratio	tw_med_num_enlaces_por_tweet	rt_num_medio_urls
	tw_longitud_media_tweet	rt_num_medio_hashtags
	tw_n_palabras_lexico	rt_num_medio_mentions
	tw_n_palabras_med_tweet	rt_num_med_videos_por_tweet

A continuación, se describen algunos atributos empleados por cada uno de los grupos de descarga:

- **Para el conjunto de Tweet:** De la información descargada respecto a los tweets generados por los usuarios, se obtienen datos del contenido y patrones a la hora de publicar de cada uno de ellos. Aunque no es objetivo estudiar en profundidad el contenido de los tweets, si se emplean algunos datos con estas características.

  Para encontrar patrones horarios en la publicación, se generan 12 campos con el porcentaje de publicaciones en rangos de dos horas, de manera que el primero cubre desde las 00.00h de la noche hasta las 02.00h de la madrugada y el último desde las 22.00h hasta las 23.59h.

Por otro lado, se extrae la longitud media completa de los tweets, así como la longitud media después de limpiar el contenido eliminando los enlaces a páginas web externas, elementos de puntuación, números, espacios extras u otros contenidos como emoticonos. Aprovechando este análisis se ha generado el volumen medio de palabras empleados en el vocabulario de cada cuenta, es decir, el nivel léxico utilizado y el número medio de palabras que incluye cada publicación.

También se recogen varias estadísticas de media, desviación típica, desviación normal, y varianza del número de hashtags utilizados en los tweets, número de replies incluidos en los mismos, número de retweets, contenido audiovisual, enlaces a páginas externas, etc.

Es importante destacar que todas las medias se generan en base a los datos descargados, siendo un máximo de 2400 tweet por usuario.

- **Referentes al perfil:** En esta descarga de información, Twint aporta valores sobre el número de tweets generados por el usuario, número de seguidores y cuentas a las que sigue, así como el número de "me gusta" o el volumen de contenido audiovisual que genera la cuenta. Todos estos atributos se incorporan como valores enteros al vector.

  También se encuentran variables para determinar si la cuenta es privada, está verificada, tiene disponible la localización, o si han completado el fondo del perfil con una imagen o descripción biográfica. Estos detalles se incluyen en el vector como variables booleanas que indicarán si se dispone o no de estos datos. En caso afirmativo, se contabiliza la longitud de los campos para para disponer de estos detalles en la clasificación, como en la descripción de la biografía o inclusión de *urls* para enlaces externos.

  Finalmente se incluyen el nombre de la cuenta y nombre de usuario, de los que se aporta la longitud del nombre y descripción, además de utilizar ***gender.Detector()***, una biblioteca que puede identificar si el nombre y apodo de las cuentas son nombres de mujer, hombre o desconocidos. Con esta información se puede catalogar si se trata de nombres de personas o no. También se incluye la fecha de creación de la cuenta con lo que se calcula la edad de la misma en días hasta el momento del procesado de los datos y se incluye el ratio *followers/following (tff)*.

- **Referentes a los Retweet:** En este conjunto de datos Twint descarga en su mayoría retweets, replies y algún tweet normal generando conjuntos de un máximo de 3200 elementos. Para dar cualquier estadística habrá que realizar los filtros correspondientes para saber de qué tipo de elemento se trata.

  Entre otra información, se desea encontrar el tiempo medio que tarda una cuenta en realizar un retweet. Twint ofrece en distintas columnas el momento de creación del tweet y en el que se realiza el retweet, por lo que se incluye la resta de estos valores en el vector como tiempo en segundos.

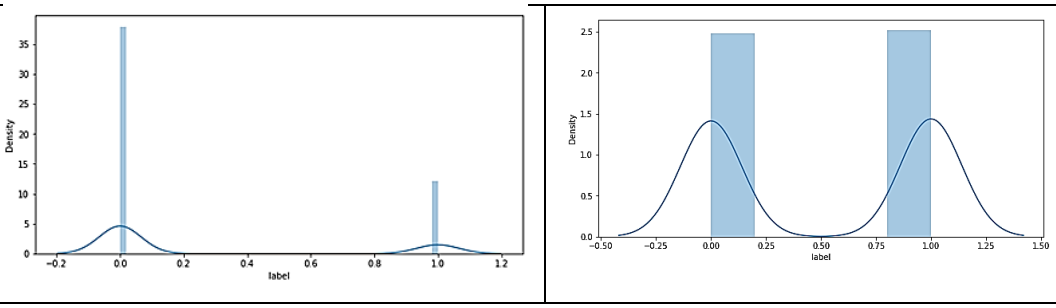
  Por otro lado, se capturarán los ratios de retweet y replies del conjunto descargado, y las estadísticas empleadas en el conjunto de los tweet para incluir diferentes métricas de los enlaces a url, hashtag, likes, videos, menciones, etc.

Todas las características descritas han llegado a formar un vector con 75 elementos para cada uno de los nodos. A continuación, se expone la matriz de confusión de estos datos, en la que se puede ver la correlación que tienen entre ellos y sobre el atributo "*label*", la etiqueta que cataloga si un perfil es un bot o un humano:

Figura E-1: Matriz de confusión Dataset 2

Antes de iniciar cualquier proceso de limpiado, es imprescindible identificar la densidad de la población de cuentas bot de los conjuntos. Debe comprenderse que al generar los grafos de relaciones muchos de los usuarios de los que se dispone información no estarán incluidos, y la relación bot/humanos en los atributos de los nodos presentarán cambios dependiendo de las relaciones que se procesen (retweet y replies).

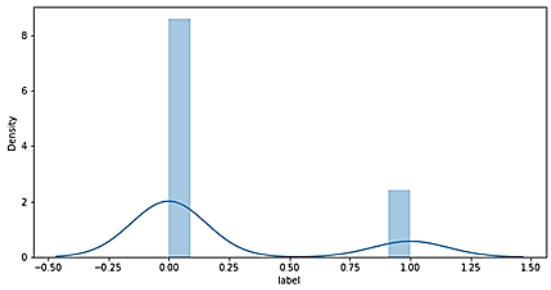
Figura E-2: Densidad bot/humanos en el Dataset 2 (izquierda) y Dataset 1 (derecha).



En la imagen E-2 se pueden ver las distribuciones para el Dataset 1, donde se observa una distribución muy parecida, conteniendo 588 usuarios siendo el 50.34% de los mismos cuentas legítimas y el 49.65% restante bots. También se presenta el Dataset 2, con 11664 elementos y un 75.85% de usuarios humanos frente al 24.14% bots, algo más desbalanceado.

Por último, se visualizan los datos del conjunto etiquetado mediante Botometer, con 137 usuarios etiquetados siendo 30 de ellos bot frente a 107 usuarios humanos (Figura E-3):

Figura E-3: Densidad bot/humanos en el Conjunto etiquetado con Botometer



Para favorecer el entrenamiento de los algoritmos, se han eliminado todos los elementos incorrectos o nulos, aplicando en los campos de texto la longitud numérica de los mismos o transformaciones **one shot** en los casos que proceda. Después se aplica una **transformación de datos yeo-yonson,** que permite manejar valores negativos. Se trata de una transformación monótona de los datos empleando funciones de potencia. Se suele utilizar para estabilizar la varianza y hacer que los datos tengan una distribución más normal, mejorando la validez de las medidas de asociación y distintas correlaciones. A continuación, se presenta la forma en que se aplica sobre los datos

Figura E-4: Transformación yeo-johnson de los datos

```
pt = preprocessing.PowerTransformer(method="yeo-johnson", standardize=True)
df_values_log = pt.fit_transform(df_values)
```

Se han empleado los listados de correlación sobre la variable objetivo ordenados de mayor a menor, de forma que se pueda ver qué variables están más relacionadas directamente con la etiqueta que distingue las cuentas legítimas de las bot. Esto se repite para cada uno de los grupos de información generados con los grafos, obteniendo datos diferentes en cada uno y completándose a su vez con los datos normalizados y sin normalizar. Dichos listados permitirán eliminar las variables menos relacionadas para conseguir que los modelos funcionen mejor. En la *Tabla E-3* se muestra un ejemplo con los datos sin transformar del Dataset 2. Esto ayuda a mejorar paulatinamente el vector de características para las pruebas finales.

Tabla E-3: Ejemplo listados de correlación de variables con la etiqueta objetivo

Atributo	Correlación sobre label		Atributo	Correlación sobre label
Label	1.0		pr_followers	-0.0676
rt_ratio	0.335		tw_likes_count_media	-0.0697
rt_num_medio_hashtags	0.1902		tw_retweets_count_des_med	-0.0737
tw_hora_6	0.1408		tw_likes_count_desv_med	-0.0758
tw_hora_5	0.1227		tw_hora_2	-0.0759
tw_hora_7	0.0959		tw_replies_count_desv_med	-0.0773

rt_time_to_rt_mean	0.0725		tw_re-tweets_count_desv_tip	-0.0876
pr_tweets	0.0657		tw_hora_11	-0.088
rt_time_to_rt_desv_norm	0.0653		tw_likes_count_desv_tip	-0.0886
tw_med_num_enlaces_por_tweet	0.0632		rt_retweets_count_media	-0.0986
pr_media	0.0581		tw_hora_12	-0.1005
tw_hora_8	0.0579		rt_re-tweets_count_des_med	-0.1102
tw_hora_4	0.0572		pr_likes	-0.114
rt_time_to_rt_desv_tip	0.0418		tw_longitud_bruta_me-dia_tweet	-0.1223
pr_following	0.0243		rt_num_medio_mentions	-0.1733
rt_time_to_rt_var	0.0182		pr_location	-0.1739
pr_real_name	0.0045		tw_longitud_media_tweet	-0.1833
tw_num_medio_mentions_por_tweet	0.0042		tw_n_pala-bras_med_tweet	-0.1833
rt_num_medio_urls	-0.0037		pr_len_url	-0.1901
pr_len_username	-0.0071		pr_len_bio	-0.2047
pr_private	-0.009		pr_age_account	-0.2133
tw_replies_count_var	-0.0093		rt_num_med_vid-eos_por_tweet	-0.2474
rt_retweets_count_var	-0.0126		rt_likes_count_desv_med	-0.2578
pr_real_username	-0.0165		rt_re-plies_count_desv_med	-0.2578
tw_num_medio_hashtags_por_tweet	-0.0177		rt_likes_count_media	-0.2578
tw_likes_count_var	-0.0183		rt_replies_count_media	-0.2578
tw_retweets_count_var	-0.0197		pr_bio	-0.2584
pr_tff_ratio	-0.0204		rt_likes_count_var	-0.2669
tw_hora_3	-0.0273		rt_replies_count_var	-0.2669
tw_hora_10	-0.0384		rt_likes_count_desv_tip	-0.2669
pr_len_name	-0.0471		rt_replies_count_desv_tip	-0.2669
rt_retweets_count_desv_tip	-0.0527		pr_verified	-0.2774
tw_hora_9	-0.0534		pr_background_image	-0.2873
tw_replies_count_desv_tip	-0.0625		tw_desv_prome-dio_long_tweet	-0.3317
tw_hora_1	-0.0626		rt_ratio_reply	-0.335
tw_replies_count_media	-0.0645		tw_n_palabras_lexico	-0.3967
tw_retweets_count_media	-0.0647			

Tras todo el proceso descrito se eliminan las siguientes características (Tabla E-4) dejando solo los atributos mostrados en la sección 4.

Tabla E-4: Atributos eliminados del vector de características

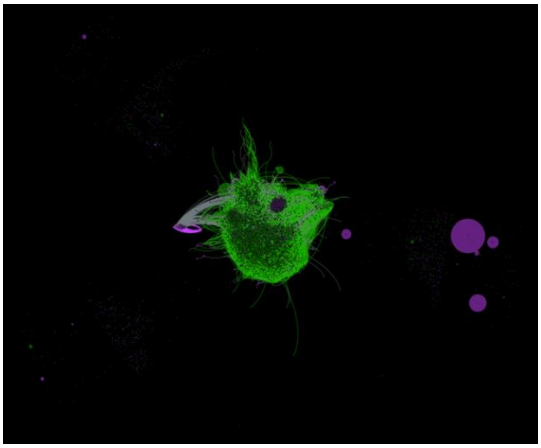
rt_likes_count_media	pr_background_image	tw_n_tweet_capturado
rt_replies_count_desv_tip	tw_desv_promedio_long_tweet	pr_tweets
rt_likes_count_desv_tip	rt_ratio_reply	rt_retweets_count_var
rt_replies_count_var	tw_n_palabras_lexico	rt_replies_count_var
rt_likes_count_var	pr_real_username	rt_likes_count_desv_tip
rt_replies_count_desv_med	pr_private	rt_replies_count_desv_tip
rt_likes_count_desv_med	pr_bio	pr_verified
rt_replies_count_desv_med	rt_likes_count_var	rt_replies_count_media
	rt_likes_count_media	

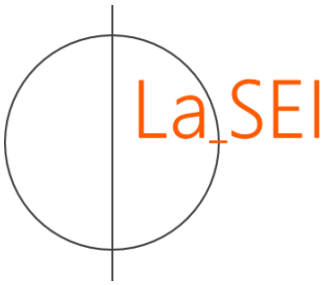
Anexo F: Grafos combinados de Retweets y Replies del Dataset 2

En este anexo se presenta, para los Dataset 1 y 2 la combinación de actividades generadas por sus usuarios en cuanto a replies y retweets se refiere:

Figura F-1: Grafo Dataset 1 con relaciones combinadas

Figura F-2: Grafo Dataset 2 con relaciones combinadas





Reports de Inteligencia Económica y Relaciones internacionales

[ISSN 2660-7352]

PUBLICACIONES DE LA ESCUELA DE INTELIGENCIA ECONÓMICA DE LA UAM


La_SEI
School of Economic Intelligence & International Relations
Escuela de Inteligencia Económica y RRII
Universidad Autónoma de Madrid