DETECCIÓN DE FAKE FOLLOWERS EN TWITTER MEDIANTE CARACTERÍSTICAS DE REDES SOCIALES

Máster Universitario en Analítica de Negocio y Grandes Volúmenes de Datos

Presentado por:
D. DANIEL FRANCISCO LÓPEZ

Dirigido por:
Dr. MARÇAL MORA CANTALLOPS

Alcalá de Henares, a 7 de Febrero de 2020

Detección de *fake followers* en *Twitter* mediante características de redes sociales

Francisco López, Daniel

7 de febrero de 2020

Índice general

1. Introducción **7**

1.1. Resumen 7

1.2. Abstract 8

1.3. Palabras clave 8

1.4. Preámbulo 9

1.5. Objetivos 10

2. Marco teórico **11**

2.1. Estado del arte 11

2.2. Definiciones generales 13

2.2.1. ¿Qué es un *fake follower*? 13

2.2.2. ¿Qué es la complejidad algorítmica temporal de un algoritmo? 14

3. Metodología **15**

3.1. Elección del *dataset* 16

3.1.1. Introducción 16

3.1.2. Características necesarias para el *dataset* candidato (criterios de elegibilidad) 16

3.1.3. Relación de *datasets* y verificación de su elegibilidad . . . 18

3.1.4. Elección del *dataset* finalista y estudio asociado 29

3.2. Elección de características para el modelo 29

3.2.1. Introducción a las características heredadas 30
3.2.2. Características heredadas del modelo de *Stringhini et al.* [1] 31
3.2.3. Características heredadas del modelo de *Yang et al.* [2] . . 35
3.2.4. Nuevas características de redes sociales 40
3.2.5. Relación final de las nuevas características de redes sociales evaluadas 51
3.3. Construcción de la red 52
3.4. Creación del modelo 52
3.4.1. Introducción 52
3.4.2. Métricas de evaluación del modelo 53
3.4.3. Elección del clasificador 53
3.4.4. Balance entre las clases de los datos 54
3.4.5. Proceso de creación del modelo 54
3.5. Construcción de caso real 60
3.5.1. Introducción 60
3.5.2. Obtención de *tweets* de *Twitter* 61
3.5.3. Filtrado de los datos 62
3.5.4. Obtención de usuarios, *friends* y *followers* 62
3.5.5. Precisiones sobre las APIs de *Twitter* 63

4. Resultados **65**
4.1. Resultados obtenidos en la creación del modelo 65
4.1.1. Fase I del modelo 65
4.1.2. Fase II del modelo 66
4.1.3. Fase III del modelo 67
4.1.4. Fase IV del modelo 68
4.2. Resultados del caso real 71
4.2.1. Datos de interés del conjunto de datos final 71
4.2.2. Detalles de la predicción del caso real 71

5. Discusión **85**

6. Conclusiones **91**

6.1. Propuestas de trabajo futuro 92

Capítulo 1

Introducción

1.1. Resumen

La información es muy importante para la comunicación y el progreso de las sociedades, permitiendo intercambiar conocimiento, sucesos, y generar opinión. En un contexto en el que las personas cada vez tienen más acceso a la información, se observa una tendencia hacia la desinformación, especialmente en el entorno de las redes sociales. Particularmente, la existencia de *fake followers* en *Twitter*, consigue aumentar la notoriedad de ciertas personas que los contratan de forma ilegítima, alterando artificialmente la esfera política, económica y social. En este entorno, se va a intentar mejorar un modelo preexistente de *machine learning* para la detección de *fake followers*, introduciendo al modelo nuevas características de redes sociales, basadas en el grado, la centralidad o el prestigio entre otros. Para la selección de estas características, será un factor relevante el crecimiento del coste de procesamiento conforme lo haga el volumen de los datos procesados, siendo la complejidad algorítmica un importante indicador de dicho coste. Posteriormente, se comparan crítica y cuantitativamente ambos modelos, con el fin de determinar si la inclusión de métricas de redes sociales redunda en una mejora significativa de la predicción de *fake followers*. Por último, se prueba el modelo desarrollado sobre un caso real de *tweets* sobre

los debates preelectorales de las elecciones del 10 de noviembre. La conclusión de este estudio es que, a pesar de que se obtiene una ligera mejora en la predicción, no justifica el enorme coste de extracción, almacenamiento, modelado y procesamiento de la red.

1.2. Abstract

The information is very important for the communication and the progress of the societies, allowing to exchange knowledge, events, and generate opinion. In a context in which people increasingly have access to information, there is a tendency towards misinformation, especially in the social network environment. Particularly, the existence of fake followers on Twitter that manages to increase the notoriety of certain people who hire them illegitimately, artificially altering the political, economic and social sphere. In thus environment it will be tried to improve a pre-existing machine learning model for the detection of fake followers, introducing new features of social networks, based on degree, centrality or prestige among others. For the selection of these characteristics, the growth of the processing cost will be a relevant factor according to the volume of the processed data, being the algorithmic complexity an important indicator of that cost. Subsequently, both models are critically and quantitatively compared, in order to determine whether the inclusion of social network metrics results in a significant improvement in the prediction of fake followers. Finally, the model developed on a real case of tweets about the pre-election debates of the elections of November 10th is tested. The conclusion of this study is that, although a slight improvement in the prediction is obtained, it doesn't justify the enormous cost of extraction, storage, modeling and processing of the network.

1.3. Palabras clave

machine learning; redes sociales; fake followers; clasificación; complejidad algorítmica

1.4. Preámbulo

La información es muy importante para la comunicación y el progreso de las sociedades, permitiendo intercambiar conocimiento, sucesos, y generar opinión. No obstante, en un contexto en el que las personas cada vez tienen más acceso a la información, se observa una tendencia hacia la desinformación, especialmente en el entorno de las redes sociales.

Particularmente, hay *bots* en *Twitter* que pretenden influenciar en la opinión pública mediante la publicación de *tweets*, lo que puede implicar cambios drásticos a distintos niveles (incluido el nivel político) sin que la sociedad se dé cuenta, influyendo o intentando influir en la misma tal y como se ha podido percatar tanto la prensa generalista como medios científicos [3][4][5][6]. Hay casos incluso en los que otros países presuntamente influyen en los resultados electorales del país [7][8].

La detección de estos *bots* puede ser de utilidad para optimizar las campañas electorales y hacer estudios sociológicos diversos, y es un problema de *data science* de sumo interés para la sociedad, ya que estos *bots* alteran la relevancia de las personas y de los temas, en base a intereses que no siempre tienen por qué ser legítimos. Además, hay que destacar que para el público en general también es un aspecto de importancia, dada la cantidad de usuarios de *Twitter* presentes en España (cerca de 5 millones) [9].

En este trabajo se va a trabajar en un tipo concreto de fraude: *los fake followers*. Estos, inflan la cantidad de seguidores de una cuenta dada, de tal manera que aumentan la notoriedad de la misma. Sin embargo, este tipo de comportamientos no son positivos, ya que pueden afectar al ámbito político, económico y social [3].

El propósito es mejorar un modelo de *machine learning* en el que, mediante métricas de redes sociales, con el objetivo de incrementar las prestaciones del mismo. Sin embargo, dada la ingente cantidad de información existente, será necesario tener en cuenta el coste de generación de las mismas.

1.5. Objetivos

El principal objetivo del presente trabajo es crear un modelo predictivo para la detección de *fake followers* en la red social de *Twitter* basado en características. Partiendo de las características previstas por otro modelo, añadiendo nuevas características de redes sociales y mediante un *dataset* existente, se creará un nuevo modelo de clasificación. Para la introducción de las características de redes sociales, se tendrá especialmente en cuenta el coste de procesamiento de las mismas. Este coste, se medirá mediante la complejidad algorítmica temporal, formando parte del conjunto de características solamente aquellas métricas que tengan coste a lo sumo lineal y que sean computables en un tiempo razonable (no superior a un día).

Una vez creado dicho modelo, se comparará con el modelo preexistente, para observar si existe mejora en alguno de los criterios de evaluación de modelos propuestos, y si es una mejora suficiente como para justificar el uso de las características de redes sociales (que tienen un coste mayor). También se detallarán las características que más aportaciones hacen, las que menos, y aquellas que hayan sido sustituidas entre ambos modelos.

Por último, se ejecutará dicho modelo sobre un *dataset* de un caso real: las interacciones en *Twitter* a través de *hashtags* relacionados con los debates preelectorales de las elecciones generales del 10 de noviembre de 2019 y se analizará el reparto de la clasificación entre usuarios legítimos y *fake followers* así como la distribución de las probabilidades de dicha clasificación.

Capítulo 2

Marco teórico

2.1. Estado del arte

Los medios de comunicación han evolucionado significativamente a lo largo de la historia. Con el auge de las redes sociales, las personas han pasado de ser meras lectoras de información a emplear diversas plataformas para compartir, modificar y discutir contenido de Internet [10]. Existen estudios como [10] que catalogan los medios sociales en bloques funcionales: presencia, compartición, relaciones, identidad, conversaciones, grupos y reputación. *Twitter*, podría ser catalogado como lugar de conversación, aunque en cierta manera también representa reputación (los perfiles, según el número de *seguidores*, el conocimiento del área sobre la que intercambian información, etcétera). Así pues, comprender y saber aprovechar las potencialidades de cada red social es importante para las empresas y la sociedad en general. Sin embargo, el uso indebido de dicho conocimiento, conlleva hacer un uso de estos bloques de forma ilegítima, pudiendo alterar el prestigio de ciertas cuentas intencionalmente. En el entorno de redes sociales, existen artículos sobre la influencia de ciertas métricas de redes sociales, como los seguidores, sobre el prestigio de las empresas [11]. También se ha comprobado que ciertos indicadores como los seguidores mencionados anteriormente, la cantidad de *tweets* o las interacciones son relevantes para la influencia

en *Twitter* [12]. A tenor de los estudios mencionados, se observa que la influencia y el prestigio en las redes crece conforme lo hacen el número de seguidores. En base a esto, han surgido servicios que buscan inflar dicho número de seguidores para aumentar la notoriedad, lo que se conoce como *fake followers* [13]. De forma paralela, han surgido iniciativas como [14] que buscan luchar contra estas prácticas, debido a la ilegitimidad de la influencia de quien las practica.

Inicialmente, las estrategias para la detección de *bots* y particularmente de *fake followers* consistían en reglas empíricas que los *bloggers* y personas del entorno de los medios de comunicación deducían en base a la mera observación de ciertos atributos o patrones de comportamiento de los usuarios [13]. Sin embargo, esta forma de detección es inviable debido a la gran cantidad de información y al continuo perfeccionamiento de los mismos [15]. Es por esto, que se empezaron a emplear algoritmos de *machine learning* para la detección de dichos *bots*.

Bastantes artículos, coinciden en este enfoque, intentando además explicar la aportación de las variables que emplean en los modelos. [16] observa durante 7 meses la interacción de los usuarios, introduciendo algunas de las características sospechadas antiguamente, como la longitud de los nombres, la frecuencia de publicación o incluso evalúa los *tweets* y su contenido en busca de enlaces. Introduce alguna métrica de redes sociales como puede ser el número de enlaces bidireccionales. [2] y [1] avanzan por el mismo campo, siendo [2] quien más propuestas realiza respecto a las variables derivadas del grafo de *friends* y de *followers*, teorizando y probando en pequeños grafos algunas métricas más complejas como pueden ser la centralidad de cercanía o el coeficiente local de agrupamiento. No obstante, decide que por el coste computacional hace inviable su cálculo. [13], aborda de forma más amplia la evolución en las técnicas empleadas para la detección de *fake followers*, no solamente describiendo de forma detallada el funcionamiento de ciertos sistemas de clasificación de usuarios en base a conjuntos de reglas y poniéndolos en contexto con los clasificadores en base a características, si no que ha cuantificado la aportación de las variables y la precisión de ambos. En la creación de un modelo de características, crea un

modelo basado en los modelos de [1] y [2] y posteriormente lo simplifica (reduciendo un poco los parámetros de evaluación del modelo) para reducir el coste de predicción (eso sí, suprimiendo las características que necesiten ser calculadas en base a una red de usuarios con sus relaciones de amistad, ya que una de las intenciones es reducir el coste de datos descargados debido a las limitaciones de la API de *Twitter* [17, 18]). Finalmente, se observa que [19] realiza un clasificador con características de redes sociales relativas a la centralidad, aunque no evalúa como tal el coste de las mismas.

La cuestión de la complejidad de los algoritmos, ha sido muy importante desde que *Alan Turing* ideara su máquina, debido a que es un marco de trabajo interesante que permite saber el coste de un algoritmo y sirve como marco teórico para comparar distintas implementaciones de un algoritmo [20], por lo que esta será tenida en cuenta al evaluar las métricas de redes sociales, basándose en bibliografía relativa a teoría de grafos [21] en la medida de lo posible. Tener en cuenta este criterio es una decisión muy importante en *data science* debido al crecimiento exponencial de la información [22].

2.2. Definiciones generales

En la presente sección se procede a explicar un par de conceptos que son necesarios para comprender el alcance del proyecto y el contexto en el se enmarca.

2.2.1. ¿Qué es un *fake follower*?

Un *fake follower* es aquella cuenta de *Twitter* que se ha creado con el propósito de inflar el número de seguidores de otra cuenta determinada [13]. Los *fake followers* son peligrosos porque alteran los conceptos de popularidad e influencia en las redes, pudiendo impactar en muchas áreas de las sociedad como es la política, la economía o la sociedad [13] en casos tales como [3, 4, 5, 6, 7, 8].

2.2.2. ¿Qué es la complejidad algorítmica temporal de un algoritmo?

La complejidad temporal de un algoritmo es una forma de medir cómo de rápido crece el tiempo que conlleva su computación conforme crece el tamaño del problema a resolver. Esta forma de medida tiene la particularidad de que no está ligada a las especificaciones de un ordenador concreto ni a los recursos que está empleando, si no que es una medida que solamente se centra en cómo está construido dicho algoritmo. No tener en cuenta los criterios anteriormente mencionados facilita mucho la comparación entre algoritmos y el entendimiento del desempeño del mismo [22, 20].

Capítulo 3

Metodología

Este capítulo versa sobre la necesidad de abordar los siguientes hitos en el presente trabajo:

1. Determinar los criterios que serán aplicados para la elección de un *dataset*.

2. Elegir un *dataset* entre el conjunto de *datasets* más relevantes de la comunidad, que cumpla con los criterios de elegibilidad anteriormente descritos, eligiendo junto al *dataset* el artículo científico que lo emplea.

3. Reproducir las características del modelo implementado en el trabajo de investigación del que se va a partir y generar las nuevas características de redes sociales derivadas de la información del *dataset*.

4. Generación de la propuesta del modelo incluyendo las características del modelo anterior así como las nuevas características enmarcadas dentro del ámbito de las redes sociales.

5. Obtención de datos de *Twitter* orientados a la construcción de un *dataset* sobre el que se ejecutará la predicción del modelo previamente generado y se observará la distribución de usuarios legítimos y de *fake followers*.

3.1. Elección del *dataset*

3.1.1. Introducción

Este capítulo tiene como objetivo la búsqueda y selección de un *dataset* para el entrenamiento del nuevo modelo y la posterior predicción de *bots*, extrayendo las métricas para alimentar dicho modelo y cotejar su validez.

Para ello, se van a analizar los principales trabajos y *papers* que se encuentren sobre esta materia, para ver los conjuntos de datos que aportan, y en caso afirmativo, elegir el que más convenga para el entrenamiento del modelo predictivo a desarrollar.

3.1.2. Características necesarias para el *dataset* candidato (criterios de elegibilidad)

Para que el *dataset* elegido cumpla correctamente el propósito para el que va a ser empleado, debe tener las siguientes características:

- Que sea un *dataset* con datos de la red social *Twitter* (es la red social elegida para realizar el análisis de *bots*).
- Que sea un *dataset* etiquetado indicando si el usuario es o no es un *bot*, para poder entrenar los algoritmos deseados.
- Que contenga toda la información necesaria para detectar si un usuario dado es un *bot* o no. En caso de que un *dataset* indicara dos columnas, una con el identificador del usuario y otra indicando si es *bot* o no lo es, habría que descargarlo, lo que puede suponer un problema de cara a la fiabilidad del modelo. Esto es debido a que se desea garantizar el valor y la veracidad de la información. Debido a que la red social *Twitter* es dinámica, podría suceder que los datos del perfil del usuario, *tweets* y cualquier otro elemento que pueda definir la probabilidad de que un usuario sea o fuese un *bot* se viera alterada. Por ejemplo, si aquel usuario catalogado como *bot*, decidiera abandonar la red social o alterara sustancialmente su

perfil o su información publicada (*tweets*), ya no se podría utilizar su información para el entrenamiento del modelo. Por lo tanto, los datos del usuario deben ser del mismo (o cercano) momento en que fue etiquetado, para evitar sesgo por las razones anteriormente expuestas.

- Debido a que se plantea mejorar el modelo en base a la introducción de características de redes sociales, las relaciones de amistad entre los usuarios deben estar presentes en el *dataset* para poder construir posteriormente el grafo que las modele.

Los datos que se desean obtener son los siguientes:

- Tweets del usuario.
- Datos de perfil del usuario.
- Identificadores de *friends* del usuario.
- Identificadores de *followers* del usuario.

Estos dos últimos requisitos serán necesarios para la creación posterior de una red que permita construir las características de redes sociales que se consideren.

Cumpliendo estas características, el *dataset* ya podrá ser empleado para generar las métricas adicionales en el ámbito de las redes sociales, que se enumerarán y expondrán posteriormente.

Los criterios anteriormente mencionados, son los criterios mínimos de elegibilidad necesarios para que un *dataset* pueda ser elegido. No obstante, la elección del *dataset* final se realizará observando métricas tales como el volumen de datos, la completitud (de los datos y de la documentación de los estudios asociados que hubiera), el área para el que ha sido extraído o la antigüedad del mismo.

3.1.3. Relación de *datasets* y verificación de su elegibilidad

A continuación se expondrá una relación de los *datasets* encontrados, de los que posteriormente se deberán elegidos aquellos que puedan ser candidatos. Para ello se ha elegido un repositorio especializado en este tipo de *datasets*.

La universidad de Indiana, en el marco de del Instituto de ciencia de las redes, tiene una inciativa denominada *Botometer*, en la que tienen un repositorio que promueve los principios de compartir *datasets* anotados en relación con la detección de *bots* en *Twitter*. Su motivación, que refleja muy bien el valor del estudio de estas áreas de conocimiento, se expone (traducida) a continuación [14]:

> La evidencia creciente sugiere que hay una cantidad creciente de contenido publicado en redes sociales que está siendo generado por entidades autónomas conocidas como *bots* sociales. Muchos *bots* realizan funciones útiles, pero hay un registro creciente de aplicaciones maliciosas de *bots* sociales. Creemos que es importante proveer *datasets* públicos y herramientas que ayuden a la identificación de *bots* sociales, ya que las herramientas de engaño y de detección están en una carrera armamentística.

Así pues, de este repositorio se extraerán varios *datasets*, para los que se verificará su elegibilidad.

Caverlee 2011

Este *dataset* contiene datos recolectados de *Twitter* desde el 30 de diciembre de 2009 hasta el 2 de agosto de 2010. El *dataset* fue utilizado en el artículo denominado *Seven months with the devils: a long-term study of content polluters on Twitter* [16].

El *dataset* está compuesto de tres ficheros para los usuarios legítimos y otros tres con el mismo tipo de información para los usuarios que son *bots*. Todos los

ficheros son en formato TSV [23]. La descripción de cada uno de los ficheros se realiza a continuación:

- Contenido de los usuarios (`content_polluters.txt` y `legitimate_users.txt`): Contiene en cada línea un perfil (usuario) de la red social.

  Los campos contenidos en dichos ficheros son los que se muestran a continuación:

  - **UserID:** Identificador interno del usuario (no es el identificador de *Twitter*).
  - **CreatedAt:** Fecha de creación del *tweet*.
  - **CollectedAt:** Fecha de recolección del *tweet*.
  - **NumberOfFollowings:** Número de seguidores.
  - **NumberOfTweets:** Número de *tweets*.
  - **LengthOfScreenName:** Longitud del nombre de usuario.
  - **LengthOfDescripcionInUserProfile:** Longitud de la descripción del nombre de usuario.

- Contenido del número de seguidores obtenidos periódicamente (`content_polluters_followings.txt` y `legitimate_users_followings.txt`): Contiene en cada línea para un perfil dado diversas muestras del número de seguidores que tiene. Los campos empleados son los que se muestran a continuación:

  - **UserID:** Identificador interno del usuario (no es el identificador de *Twitter*).
  - **SeriesOfNumberOfFollowings:** Lista de número de seguidores de un usuario obtenidos de forma periódica. Los elementos de esta lista son los identificadores internos, separados por coma.

- Contenido de los *tweets* (`content_polluters_tweets.txt` y `legitimate_users_tweets.txt`): Contiene en cada línea un *tweet*. Los campos empleados son los que se muestran a continuación:

- **UserId:** Identificador interno del usuario (no es el identificador de *Twitter*).
- **TweetID:** Identificador del *tweet*.
- **Tweet:** Contenido textual del *tweet*.
- **CreatedAt:** Fecha de creación del *tweet*.

Respecto al volumen de los datos, se detecta en el *dataset* las cantidades de registros (*tweets* y usuarios) mostradas en la siguiente tabla donde se muestran desglosando cuantos registros de información provienen de usuarios legítimos y cuantos de usuarios ilegítimos (*bots*):

Tipo de dato	#legítimos	#ilegitimos
Tweets declarados	3.263.238	2.380.059
Tweets contabilizados	3.259.693	2.353.473
Usuarios	19.276	22.223

Como se ha podido observar, es una red de *Twitter* en la que se distingue abiertamente entre los usuarios que son *bots* (y sus *tweets*) y aquellos que son legítimos, por lo que se cumplen los dos primeros criterios básicos de elegibilidad. Además, se observa que los datos están bastante completos ya que incluyen la información necesaria de perfiles y *tweets*. Sin embargo, no se cumpliría el último criterio de elegibidad, que estriba en la necesidad de la información de las relaciones de amistad entre los perfiles.

Se incluyen bastantes campos que probablemente serían de utilidad, aunque la antigüedad es significativamente elevada (datos de los años 2009 y 2010) y la red es de carácter generalista, esto es, no se hace un especial hincapié en eventos o interacciones de carácter político, lo que puede no ser demasiado favorable ya que se desconoce si existen posibilidades de que la red generada se vea influida por el tipo de información que se intercambia, y es razonable sospechar que pudiera haber un cierto sesgo en ese sentido.

En conclusión, debido a las razones anteriormente expuestas, se procede a descartar este *dataset*.

The Fake Project dataset

Este *dataset*, conocido también como `cresci-2015`, contiene datos recolectados de *Twitter*, tanto de cuentas genuinas como de *bots*. Ha sido generado por investigadores del *Institute of Informatics and Telematics of Italian National Research Council (CNR)* [24] y se ha empleado principalmente en [13].

El *dataset* está compuesto realmente por cinco *subdatasets* diferentes que se diferencian según el tipo de usuarios y según el tipo de evento capturado, que se enumeran a continuación y son expuestos con posterioridad [24]:

- TFP (the fake project): 100 % humanos.
- E13 (elecciones italianas de 2013): 100 % humanos.
- INT (intertwitter): 100 % *fake followers*.
- FSF (fastfollowerz): 100 % *fake followers*.
- TWT (twittertechnology): 100 % *fake followers*.

The Fake Project dataset — TFP - The fake project

En el caso del *subdataset* de "The Fake Project", los autores procedieron a crear una cuenta de *Twitter* denominada **@TheFakeProject** el 12 de diciembre de 2012, cuyo perfil indicaba el lema "Sígueme solo si NO eres falso" y explica que la iniciativa está vinculada con un proyecto de investigación promovido por investigadores del de investigadores del *Institute of Informatics and Telematics of Italian National Research Council, IIT-CNR*, en Pisa-Italia [24]. En la biografía de la cuenta, se publicó un enlace a la página web del proyecto (http://wafi.iit.cnr.it/thefakeproject/), donde se encuentran las instrucciones para unirse a la iniciativa y una descripción general sobre las motivaciones y objetivos del proyecto [24]. En una primera fase, los propietarios contactaron con otros investigadores y periodistas con el fin de dar publicidad a la iniciativa [24]. La versión *online* de un periódico popular italiano promovió el proyecto e invitó a las personas a unirse (se puede observar en [25] un extracto de esta

noticia en italiano) [24]. Periodistas y *bloggers* extranjeros también dieron apoyo a la iniciativa en sus respectivos países [24]. En doce días transcurridos entre el 12 y el 24 de diciembre, la cuenta fue seguida por 574 seguidores. A través de la API de *Twitter*, se rastreó diversa información pública de esos seguidores como los perfiles, y tu *timeline*, junto a la de sus *followers* y sus *friends* [24]. Para este conjunto de 574 perfiles de partida, se extrajeron un total de 616.193 *tweets* y 971.649 relaciones de amistad (identificadores de *followers* y *friends* [24]. Además, todos esos seguidores que se unieron voluntariamente a la iniciativa, también hicieron una fase de verificación, en la que cada seguidor recibió un mensaje directo de la cuenta de @TheFakeProject, con una URL que dirige a un *captcha*, único para cada seguidor. Se consideran humanos certificados los 469 perfiles que completaron con éxito el *captcha* [24].

The Fake Project dataset — E13 - Elecciones italianas de 2013

El *dataset E13* nació para apoyar una iniciativa de investigación para un estudio sociológico realizado en colaboración con la Universidad de Perugia y la Universidad Sapienza de Roma [13]. El estudio se centró en los cambios estratégicos en el panorama político de Italia en el periodo transcurrido entre el 2013 y el 2015 [13]. Los investigadores lograron identificar un total de 84.033 cuentas únicas de *Twitter* que empleaban el *hashtag* **#elezioni2013** en la redacción de sus *tweets* [24]. Estos *tweets* fueron obtenidos durante el periodo comprendido entre de 9 de enero y el 28 de febrero de 2013 [13]. Estas cuentas se han identificado mediante la búsqueda de términos en las cuentas tales como *blogger*, periodista, estratega o analista de medios de comunicación y congresista [13]. También se incluyeron en estas búsqueda los nombres de partidos políticos [13]. Posteriormente, de todas estas cuentas de usuario, se descartaron todas aquellas que estuvieran ligadas a la campaña electoral. Estas cuentas descartadas han sido las de políticos, candidatos, partidos, periodistas, blogeros, asociaciones, etcétera, que hayan estado oficialmente involucradas en política [13]. El resto de cuentas se clasifican como ciudadanos. De estas cuentas de ciudadanos se

extrajo una muestra, que fue analizada exhaustivamente por dos sociólogos de la Universidad de Perugia [13]. Se analizaron en profundidad las fotografías, biografías y su *timeline* con el fin de determinar si eran o no personas reales [13]. Solo si los dos sociólogos determinaban que una persona era real, se incluía en el *dataset E13* [13].

The Fake Project dataset — INT, FSF, TWT - *Datasets* de *fake followers*

Estos *subdatasets* fueron comprados en abril del 2008 en tres diferentes proveedores distintos [13]. Mientras que hubo dos proveedores que proporcionaron más *fake followers* de la cantidad por la que se había pagado, otro proveedor entregó la cantidad justa, pero entorno al 15 % de los *fake followers* ya estaban suspendidos por *Twitter* cuando se fue a recabar su información. Tras obtener los identificadores de estos *fake followers*, se obtuvo toda su información y pasaron a formar parte del *dataset* [13].

[13] reconoce que la obtención que se ha hecho de estos *fake followers* no es la más exhaustiva, si no que se ha realizado de esta forma para representar de forma ilustrativa todos los posibles conjuntos de *datasets* de *fake followers*. Pero detalla que estos proveedores han sido localizados de forma sencilla en los buscadores, por lo que pueden argumentar que su *dataset* representa aquello que fue fácilmente posible encontrar en la Web en el momento en que se buscó [13].

The Fake Project dataset — campos contenidos en el *dataset*

En cada *subdataset* hay varios ficheros CSV, que se componen de los campos que se muestran a continuación (que vienen de *Twitter* directamente):

- Fichero de *tweets* (`tweets.csv`): Este fichero contiene por cada línea un *tweet*. Su cabecera se enumera a continuación:

- created_at
- id
- text
- source
- user_id
- truncated
- in_reply_to_status_id
- in_reply_to_user_id
- in_reply_to_screen_name
- retweeted_status_id
- geo
- place
- retweet_count
- reply_count
- favorite_count
- num_hashtags
- num_urls
- num_mentions
- timestamp

- Fichero de usuarios (`users.csv`): Este ficheros contiene por cada línea un usuario. Su cabecera se enumera a continuación:

  - id
  - name
  - screen_name
  - statuses_count
  - followers_count
  - friends_count
  - favourites_count
  - listed_count
  - created_at
  - url
  - lang
  - time_zone
  - location
  - default_profile
  - default_profile_image
  - geo_enabled
  - profile_image_url
  - profile_banner_url
  - profile_use_background_image
  - profile_background_image_url_https
  - profile_text_color
  - profile_image_url_https
  - profile_sidebar_border_color
  - profile_background_tile

- `profile_sidebar_fill_color`
- `profile_background_image_url`
- `profile_background_color`
- `profile_link_color`
- `utc_offset`
- `protected`
- `verified`
- `description`
- `updated`
- `dataset`

- Fichero de seguidores (`followers.csv`): Este fichero contiene por cada línea para cada perfil seguidor. Su cabecera se enumera a continuación:
  - `source_id`
  - `target_id`
- Fichero de amigos (`friends.csv`): Este fichero contiene por cada línea un amigo. Su cabecera se enumera a continuación:
  - `source_id`
  - `target_id`

La cantidad de *tweets* y de perfiles para cada *dataset* se puede observar en la siguiente tabla:

dataset	tweets	usuarios
TFP	563.694	469
E13	2.068.038	1.481
INT	58.925	1.337
FSF	22.911	1.170
TWT	114.193	845

Como se ha podido observar, es una red de *Twitter* con los datos autocontenidos en la que se distingue abiertamente entre los usuarios que son *bots* (y sus *tweets*) y aquellos que son legítimos, obteniendo además una red de *friends* y de *followers*, por lo que se cumplen los criterios básicos de elegibilidad enunciados con anterioridad.

The Fake Project dataset (v2)

Este *dataset*, conocido como `cresci-2017` contiene datos recolectados de *Twitter*, tanto de cuentas genuinas como de *bots*. Es un conjunto de varios *datasets* obtenidos en diversas capturas. Ha sido lanzado por el *Institute of Informatics and Telematics of Italian National Research Council (CNR)* [24] y se ha empleado principalmente en [26, 27].

El *dataset* está compuesto realmente por *datasets* diferentes que se diferencian según el tipo de usuarios y según el tipo de evento capturado. Estos *datasets* se definen a continuación [24]:

- **genuine accounts:** Cuentas verificadas que están gestionadas por humanos.
- **social spambots #1:** Personas que *retwitean* los *tweets* de un candidato político italiano.
- **social spambots #2:** *Spammers* de aplicaciones de pago para móviles.
- **social spambots #3:** *Spammers* de productos a la venta en *Amazon.com*.
- **traditional spambots #1:** *Dataset* de entrenamiento de *spammers* utilizado por [2]
- **traditional spambots #2:** *Spammers* para URLs de estafa.
- **traditional spambots #3:** Cuentas automáticas que envían información no deseada sobre ofertas de trabajo.
- **traditional spambots #4:** Otro grupo de cuentas automáticas que envían información no deseada sobre ofertas de trabajo.
- **fake followers:** Simples cuentas de usuario que inflan el número de *followers* de otra cuenta.

A continuación se observa una tabla con el número de cuentas y *tweets* por cada *dataset*

nombre de grupo	cuentas	tweets	año
genuine accounts	3.474	8.377.522	2011
social spambots #1	991	1.610.176	2012
social spambots #2	3.457	428.542	2014
social spambots #3	464	1.418.626	2011
traditional spambots #1	1.000	145.094	2009
traditional spambots #2	100	74.957	2014
traditional spambots #3	433	5.794.931	2013
traditional spambots #4	1.128	133.311	2009
fake followers	3.351	196.027	2012

En cada *dataset* hay varios ficheros CSV, que se componen de los campos que se muestran a continuación (que vienen de *Twitter* directamente:

- Fichero de *tweets* (`tweets.csv`): Este fichero contiene por cada línea un *tweet*. Su cabecera se enumera a continuación:

  - `created_at`
  - `id`
  - `text`
  - `source`
  - `user_id`
  - `truncated`
  - `in_reply_to_status_id`
  - `in_reply_to_user_id`
  - `in_reply_to_screen_name`
  - `retweeted_status_id`
  - `geo`
  - `place`
  - `retweet_count`
  - `reply_count`
  - `favorite_count`
  - `num_hashtags`
  - `num_urls`
  - `num_mentions`
  - `timestamp`

- Fichero de usuarios (`users.csv`): Este ficheros contiene por cada línea un usuario. Su cabecera se enumera a continuación:

- id
- name
- screen_name
- statuses_count
- followers_count
- friends_count
- favourites_count
- listed_count
- created_at
- url
- lang
- time_zone
- location
- default_profile
- default_profile_image
- geo_enabled
- profile_image_url
- profile_banner_url
- profile_use_background_image
- profile_background_image_url_https
- profile_text_color
- profile_image_url_https
- profile_sidebar_border_color
- profile_background_tile
- profile_sidebar_fill_color
- profile_background_image_url
- profile_background_color
- profile_link_color
- utc_offset
- protected
- verified
- description
- updated
- dataset

Con respecto a los criterios de elegibilidad, se cumple que es un *dataset* que contiene datos de *Twitter*. También se cumple que es un *dataset* que permite distinguir por su estructuración entre *bots* y *no bots*, siendo además un *dataset* autocompletado, desde el punto de vista que los datos que se obtienen no se limita a los identificadores de usuario si no que se obtiene mucha más información que los investigadores emplean en sus artículos de investigación. Sin embargo, este *dataset* no es elegible para este proyecto debido a que el *dataset* no contiene información que permita construir una red para modelar el seguimiento entre las diversas cuentas (*friends* y *followers*).

Otros *datasets*

Los demás *datasets* localizados a través de *Botometer*, en base al análisis previo de los mismos, han sido descartados automáticamente sin necesidad de analizarlos de forma más detallada debido a que no cumplen el criterio mínimo de elegibilidad basado en la autocompletitud de la información. Particularmente, estos *datasets* contienen un listado de identificadores de usuario etiquetados. Este tipo de *datasets* no es válido para este estudio debido a que no contienen ninguna información que permita ser empleada para clasificar, y la descarga de dicha información, habiendo pasado un cierto tiempo, implica que probablemente una buena parte de esos *bots* no existan ya, dado que la red es dinámica. Debido a este dinamismo de la red, los usuarios clasificados como legítimos (en el caso de los *datasets* donde los hubiera), ya no serían iguales ni tendrían las mismas características dentro de la red. Estos factores, introducirían un sesgo adicional en el modelo: la calidad de la información, por lo que para evitarlo, se empleará un candidato que ya tenga todos los datos necesarios.

3.1.4. Elección del *dataset* finalista y estudio asociado

Debido a que solamente hay un *dataset* que cumple los criterios de elegibilidad, no cabe la realización de análisis de candidatos. Así pues, el *dataset* elegido es el promovido por [13] realizado por *Cresci et al.* Se empleará el mismo estudio como base para la realización del nuevo modelo incluyendo características de redes sociales (una vez realizado el análisis previo de las mismas). La clasificación se basará en la detección de *fake followers* debido a que el *dataset* está basado en la detección de los mismos.

3.2. Elección de características para el modelo

En esta sección se expondrá la totalidad de las características que van a ser introducidas en una primera iteración de desarrollo del modelo.

En primer lugar, se expondrán las características heredadas del artículo en

el que se basa el presente trabajo. Como se expuso en el capítulo anterior, el trabajo en el que se basa el trabajo, se denomina *Fame for Sale: efficient detection of fake Twitter followers* [13]. Se empleará esta investigación como punto de partida debido a que el *dataset* escogido es el único que cumple todos los parámetros de elegibilidad expuestos. Además, hace enfoque en medir qué características aportan más de otros dos trabajos en los que se basa cuando habla de crear un modelo basado en conjuntos de características [13, 2, 1], y procura la creación de un modelo que busca ser eficiente en la medida de lo posible, soslayando en última instancia las características que requieren más esfuerzo en ser obtenidas (las de redes sociales) [13].

El objetivo de este trabajo es no tener en cuenta este criterio de coste que subraya [13], por lo que además de las características que toma [13], basadas en [2] y [1], se introducirán otras medidas de redes sociales que se han visto a lo largo del máster.

3.2.1. Introducción a las características heredadas

Dentro de [13], se hace una evaluación de la ganancia de información de las variables principales seleccionadas por los autores de [13] (*Cresci et al.*), los modelos de [1] realizado por *Stringhini et al.* [2] realizado por *Yang et al.*:

- Características propuestas por Stringhini et al. [1] y seleccionadas e interpretadas por *Cresci et al.* [13]:
  - Número de amigos.
  - Número de *tweets*.
  - Similaridad de los *tweets*.
  - Ratio de *tweets* que contienen URLs.
  - Ratio entre amigos y seguidores.
- Características propuestas por *Yang et al.* [2] y seleccionadas e interpretadas por *Cresci et al.* [13]:

- Edad de la cuenta.
- Ratio de enlaces bidireccionales entre cuentas (amistad recíproca).
- Media de seguidores de los amigos.
- Media de *tweets* de los amigos.
- Cociente entre el número de amigos y la mediana de los seguidores de los amigos.
- Ratio entre el número de *tweets* enviados a través de una API respecto del total de *tweets*.
- Ratio entre el número de *tweets* enviados a través de una API y además contienen al menos una URL respecto del total de *tweets*.
- Similaridad de los *tweets*.
- Tasa de seguimiento (cómo de rápido se siguen nuevas cuentas).

A partir de todas las características anteriormente expuestas, [13] realiza un modelo que combina todas ellas. En esta sección se detallarán en primer lugar todas ellas, lo que permitirá con posterioridad crear el modelo de [13], al que se le añadirán las características de redes sociales que se indicarán con posterioridad.

Se recuerda que el objetivo es **verificar si el modelo que sea generado en el presente trabajo mejora cuantitativamente el realizado por [13]**, introduciendo en la medida de lo posible y procurando no variar la intención y el tratamiento de las métricas empleadas por [13], al que solamente **se añadirán las nuevas características de redes sociales**.

A continuación se detallarán con detenimiento estas características y las interpretaciones que han tenido que ser hechas para ser calculadas.

3.2.2. Características heredadas del modelo de *Stringhini et al.* [1]

A continuación se exponen las métricas que empleó *Cresci et al.* en [13] para su modelo, inspirado en el trabajo de *Stringhini et al.* en [1].

Número de amigos

Cresci et al. defiende en [13] en base a lo afirmado por *Stringhini et al.* [1] que un bajo número de amigos podría indicar con alta probabilidad que esa cuenta es un *fake follower*, reduciéndose esa posibilidad conforme se incrementa el número de amigos.

El número de amigos, es sencillo de obtener ya que se extrae como un campo directo `friends_count` en el *dataframe* destinado a la obtención de usuarios. Además, cuando se genere el caso real, también vendrá junto a la información de usuario que se extraiga a través de la API de *Twitter*, como se indicará en la sección destinada a exponer la extracción del caso real.

Número de tweets

De forma análoga a la característica anterior, *Cresci et al.* defiende en [13] junto a *Stringhini et al* en [1] que una cuenta con menos de 20 *tweets* podría ser probablemente un *fake follower*, reduciéndose esa posibilidad conforme se incrementa el número de *tweets*.

El número de *tweets*, es sencillo de obtener ya que se extrae como un campo directo `statuses_count` en el *dataframe* destinado a la obtención de usuarios. No obstante, también se puede obtener como el conteo del número de *tweets* agrupados por usuario. Esta última forma, aunque algo más costosa, garantiza que las métricas subyacentes al *dataframe* de *tweets* sean coherentes.

Similaridad de los *tweets*

Esta característica, como se arguye tanto en [1] como en [13], tiene como objetivo detectar la similaridad de los *tweets*, ya que los *fake followers* tienden a escribir mensajes mucho más parecidos que los usuarios legítimos.

La obtención de esta característica tiene algo más de complejidad, puesto que requiere realizar algo más de tratamiento de cadenas. En este caso, existen discrepancias entre la definición realizada por *Cresci et al.* en [13] y la definición realizada en [1] por *Stringhini et al.*

En primer lugar, *Stringhini et al.* definió la similaridad de un usuario mediante la fórmula 3.1 que arroja un número flotante, que cuando mayor sea implicará que mayor es la similaridad, y viceversa.

$$S = \frac{\sum_{p \in P} c(p))}{l_a l_P} \tag{3.1}$$

donde S es la similaridad de los *tweets*, P es el conjunto de combinaciones de *tweets* tomadas de dos en dos del usuario determinado sobre el que se calcula la similaridad, p es un par de mensajes tomados de P, $c(p)$ es la función que calcula cuántas palabras comunes hay en el par de *tweets* contenidos en el par p, l_a, es la longitud media de los *tweets* posteados por el usuario y l_P es el número de pares contenidos en P. La dificultad de esta fórmula estriba en la cantidad de combinaciones (y tratamiento de cadenas subyacente) que es necesario procesar conforme crece el número de *tweets*. Esta cantidad de combinaciones puede ser calculada de acuerdo al coeficiente binomial expuesto en la fórmula 3.2.

$$\begin{aligned} l_P &= \binom{n}{2} = \frac{n}{2 \cdot (n-2)!} = \frac{n \cdot (n-1) \cdot \cancel{(n-2)!}}{2 \cdot \cancel{(n-2)!}} \\ l_P &= \frac{n \cdot (n-1)}{2} \end{aligned} \tag{3.2}$$

donde n es el número de mensajes de cada usuario. Como se puede observar, estas combinaciones crecen de forma cuadrática, por lo que complican el tratamiento cuando hay muchos *tweets* para un solo usuario.

En este sentido, *Cresci et al.* opta en [13] por simplificar esta métrica, relajando el criterio, pasando dicha métrica de ser un valor flotante a ser booleano. En esta aproximación, solo se verifican los 15 *tweets* más recientes y se considera que la similaridad debe tomar valor verdadero si hay al menos cuatro palabras iguales en una de las combinaciones analizadas. Si esto se produce, la métrica arrojará valor verdadero directamente, independientemente de lo que suceda con el resto de combinaciones. De esta manera, el enfoque reduce las comparaciones a hasta 105 por cada usuario, ya que este límite se ve afectado por dos condiciones, un máximo de 15 *tweets* a analizar y que en cuando se encuentra

una combinación que arroja valor verdadero al criterio anteriormente expuesto, el algoritmo no sigue analizando.

Una optimización que se ha realizado ha sido generar de forma previa los *cuatrigramas* para cada *tweet* con NLTK [28] y almacenarlos en un conjunto cada cada *tweet*, ya que se reduce enormemente el procesamiento de cadenas y se agiliza la comparación, al ser esta una simple comparación de cadenas.

Ratio de *tweets* que contienen URLs

La razón de la introducción de esta característica estriba en que los *fake followers* suelen escribir URLs en una gran parte de los *tweets* que publican mientras que en los usuarios legítimos no se observa esta tendencia [1, 13].

El cálculo de esta métrica no es muy complejo. Simplemente se calcula una variable booleana que para cada *tweet* indique si hay más de una URL comparando con la variable directa *num_urls* y luego operando mediante *split-apply-combine* sobre los *tweets* y asignando a los usuarios.

Respecto a la implementación, primero se agruparán los *tweets* en función del identificador del usuario. En segundo lugar se crea una serie de *pandas* (que luego no se incluirá en el *dataframe*) que permitirá calcular una columna *booleana* que determine en base al número de URLs declaradas por *Twitter* en cada *tweet* mediante el campo *num_urls*, y por último se hará la media de estos valores booleanos (que *pandas* implícitamente transforma en enteros para hacer la operación) y se asignarán los valores a los usuarios.

Ratio amigos/seguidores

Un valor alto de esta característica podría indicar que el usuario es un *fake follower* tal y como indica *Cresci et al.* en [13]. Basándose en el trabajo de *Stringhini et al.* que concreta en [1] la siguiente conclusión razonada: si alguien es un *bot*, es probable que las personas a las que siga ese *bot* no correspondan al *bot* siguiéndole a él. Es por ello por lo que se esperan altos valores para *fake followers* y bajos para usuarios legítimos.

El algoritmo para calcular esta característica se limita a realizar un par de operaciones binarias mediante *broadcasting* de las series `friends_count` y `followers_count` con *pandas*, por lo que la operación será muy rápida y eficiente debido a que estas operaciones están aceleradas en *Pandas* por defecto [29].

3.2.3. Características heredadas del modelo de *Yang et al.* [2]

Edad de la cuenta

La edad es un indicador bastante fiel de la legitimidad de un usuario, ya que la probabilidad de la legitimidad del mismo es mayor conforme aumenta su edad, tal y como se afirma en [13], en [2] y en [30]. Además, es razonable pensar que un usuario ilegítimo antes o después es localizado por usuarios que lo denunciarán y provocarán que *Twitter* lo elimine, jugando en contra de estos usuarios el tiempo.

Para calcularlo, en el *dataset* de partida se observa que existen dos variables claves para la obtención de esta característica que se denominan `updated`, que contiene la fecha en que obtuvieron los datos, y `created_at`, que contiene la fecha en al que se creó la cuenta. Simplemente haría falta restar ambas fechas. Como no se dispone de esta información en el *dataset* extraído del caso real, simplemente se procederá a restarle una fecha conocida en la que fue realizada la extracción. La unidad en la que se reflejará esta métrica será en días, ya que en los trabajos referenciados anteriormente no se ha observado ninguna apreciación sobre este criterio.

Ratio de enlaces bidireccionales entre cuentas (amistad recíproca)

Esta métrica es muy interesante, porque es la primera métrica que se observa entre las expuestas hasta ahora que puede ser considerada de red social. Conceptualmente en el campo de conocimiento de las redes sociales, es considerado una diada que tiene la particular de que hay una relación recíproca entre

ambos miembros. Esta métrica está diseñada para arrojar valores más bajos en el caso de *fake followers* y más altos en el caso de usuarios legítimos [13]. Más concretamente, [2] menciona que esta métrica tiende a ser más alta porque un usuario tiende a seguir a sus propios seguidores que suelen ser amigos, familia o compañeros de trabajo, y por el contrario, es menos común que un usuario siga a un *follower* al que no conoce. Para evitar que un *fake follower* pueda contrarrestar esta métrica haciéndose seguidor de todos sus *followers* simplemente se divide el número de enlaces recíprocos entre el número de *friends*.

Esta métrica se conoce en el ámbito de redes sociales como reciprocidad [31]. En la sección destinada al análisis de redes sociales, se profundizará algo más sobre el valor de esta métrica, aunque no se calculará por estar aquí presente.

Se ha optado por calcular esta métrica partiendo de la unión de las dos tablas (la de *friends* y la de *followers*), de tal manera que el registro del campo destino de la tabla de *friends* coincida con el campo fuente de *followers*. Así pues, de esta forma, los registros de esta unión interna, serán los enlaces recíprocos.

Media de seguidores de los amigos

Según se refleja en [13], esta característica tiene como objetivo reflejar la calidad en la elección de los amigos y se espera un mayor valor para los usuarios legítimos que para los *fake followers*. Es obvio que los usuarios legítimos intentan seguir cuidadosamente a cuentas que seleccionan y que suelen tener un mayor grado de calidad [2]. Por el contrario, los *fake followers* suelen seguir a diversas cuentas de forma aleatoria [2]. Tanto [13] como [2] subrayan que esta métrica, por las razones anteriormente expuestas, será probablemente mayor en el caso de usuarios legítimos y menor en el caso de *fake followers*.

En esta característica, se opta por cruzar el *dataframe* de amigos con el de usuarios, con el fin de aproximarse a esta métrica. Podrá suceder que en muchos casos no haya datos de este usuario, por lo que la métrica sería nula, y se imputaría a cero (ya que la media u otras propiedades estadísticas que se pudieran inferir en base a la población de la que se poseen los datos, no es

representativa de los usuarios de los que no se tienen datos).

Media de *tweets* de los amigos

De acuerdo a [2], esta característica es análoga a la anterior y tiene como objetivo medir la calidad de la cuenta conforme a la calidad de los *tweets* de las cuentas a las que sigue. También es posible por parte de los *fake followers* "maquillar" esta característica, por lo que para evitar este problema, desarrollan otra que se explicará a continuación. Esta característica arrojará valores altos para cuentas legítimas y bajos para *fake followers* [13].

Para calcular esto, se deben agrupar los *tweets* por usuario, hacer la media, y luego hacer una unión con el *dataframe* de *friends*. En esta métrica es inevitable que si no se dispone de toda la información de todos los *friends*, habrá circunstancias en las que la métrica generada no sea del todo correcta. Otra forma de calcular esto sería mediante el uso del campo `statuses_count` del *dataframe* de usuarios, pero no se dispone de la información de todos los usuarios, por lo que se ha descartado esta forma de calcularlo, ya que no aporta una mayor corrección que la alternativa empleada.

Cociente entre el número de amigos y la mediana de los seguidores de los amigos

Esta característica tiene como objetivo medir el ratio entre el número de amigos y la mediana de los seguidores de los amigos.

El planteamiento que hace [2] de esta característica estriba en que los *fake followers* no pueden controlar la calidad de las cuentas a las que siguen, por lo que los valores de la mediana de *followers* serán típicamente bajos. Por lo tanto, debido al gran número de amigos que tiene cada *fake followers*, los valores de la característica presentada también serán altos [2]. Para las cuentas legítimas, para mostrar el análisis de esta característica, se pueden dividir en dos tipos diferentes: cuentas comunes (cuentas legítimas sin grandes cantidades de seguidores) y cuentas populares (cuentas legítimas con grandes cantidades

de seguidores). Para el primer tipo de cuentas, también pueden seguir a sus amigos, lo que genera un pequeño valor de la mediana de seguidores para cada amigo [2]. Sin embargo, como la cantidad de personas a las que sigue tampoco es alta, el resultado seguirá siendo alto [2]. Por otra parte, en el caso de las celebridades, estas escogerán como *friends* personas que sean también populares (que tengan un alto número de *followers*), lo que hará que la métrica resultante tenga un valor bajo también [2]. Además el uso de la mediana en vez de la media persigue dificultar la elusión de esta métrica a los *fake followers* [2]. Así pues, resumidamente, esta métrica debe tomar valores altos para los *fake followers* y bajos para las cuentas legítimas [13, 2].

Para el cálculo de esta métrica, se hará la unión interna del *dataframe* de *friends* y el de *followers*, se hará la mediana de *followers* agrupando por *friends* y posteriormente se hará el cociente correspondiente.

Ratio entre el número de *tweets* enviados a través de una API respecto del total de *tweets*

Esta característica es una de las menos complejas que intenta medir el grado de automatización a la hora de emitir los *tweets*. Es interesante medir el grado de automatización porque se ha demostrado que los *bots* suelen emplear API para postear los *tweets* [2, 32]. Esta mayor incidencia de las API's es un factor que permite la sospecha de que podrá ser un factor diferenciador a la hora de saber si un usuario es legítimo o no. Si se piensa con detenimiento, esto es normal, ya que para que el modelo de negocio de la creación de *fake followers* sea rentable, necesitará generar un gran volumen de estos, lo que es incompatible con un tratamiento manual [2]. Así pues, que un usuario tenga una alta tasa de *tweets* enviados es un indicio de que se tratará de un *fake follower* [2, 13].

Para el procesamiento de esta métrica, es necesario en primer lugar identificar para cada *tweet* si ha sido posteado a través de una API o no. Tras realizar este booleano, se agrupan los *tweets* por usuario y se hace la media de este campo (*pandas* hará una transformación dinámica de este campo a tipo

entero).

Ratio entre el número de *tweets* enviados a través de una API y que además contienen al menos una URL respecto del total de *tweets*

Esta característica, es análoga a la anterior, solo que se evalúa en cada *tweet* que además de ser emitido desde una API, contenga al menos una URL. La existencia de URLs se puede verificar cotejando que el campo directo `num_urls` sea mayor que cero. Al igual que en la característica anterior, la aparición de valores altos de esta métrica suele asociarse a que el usuario sea un *fake follower*, mientras que lo contrario es más propio de usuarios legítimos [13, 2].

Respecto al método de cálculo, no varía significativamente respecto a la característica anterior con la excepción de que el valor booleano auxiliar que se tiene que calcular por cada *tweet* tiene que tener en cuenta también la existencia de al menos una URL para tomar valor verdadero.

Similaridad de los *tweets* que provienen de una API

La similaridad según [13], es muy parecida a la empleada en [1], con la salvedad de que se ha empleado otra función de tokenización para ajustarse más a [2] pero manteniendo la filosofía de [13] en materia de acotar el número de *tweets* procesados e introduciendo la limitación de que estos *tweets* comparados deben proceder de una API.

El fundamento por el que se incluye esta métrica consiste en que los *fake followers* escribirán *tweets* más similares, y además en su mayoría provendrán de una API [13, 2].

Tasa de seguimiento (cómo de rápido se siguen nuevas cuentas)

Esta característica sugiere la velocidad en que un usuario sigue a otros usuarios [13, 2]. Debido a que es muy complicado obtener esta información (ya que si se hace de forma pura, el cálculo de esta característica supondría el uso de una serie temporal capturando las personas a las que sigue alguien de forma

regular), se hace una aproximación a esta métrica calculando el cociente entre el número de *friends* y la edad del usuario [13, 2]. Esta métrica suele arrojar un mayor valor para los *fake followers* mientras que los usuarios legítimos no [13, 2].

3.2.4. Nuevas características de redes sociales

A continuación, se proceden a explicar las nuevas características que se desean introducir en el modelo, exponiendo una justificación razonable de las razones que de forma hipotético-deductiva se esgrimen para su inclusión o exclusión en el modelo. Estas características serán relativas al conjunto de métricas de redes sociales, y serán extraídas de una red generada con los *followers* y los *friends* que se encuentran en el *dataset* escogido de entre los candidatos. Así mismo, para todas las características que se consideren de interés, acaben o no siendo *inputs* del modelo final, se expondrán y solo en aquellas características en las que se localicen referencias para ello o bien el algoritmo sea sencillo como para analizarlo manualmente, se incluirá un análisis de su complejidad algorítmica. Dado que para varias de las características, pueden existir varios algoritmos que permitan su obtención, se expondrá en la medida de lo posible la métrica que emplee *NetworkX*.

Como se podrá observar a continuación, las métricas que se emplean son aquellas que se pueden catalogar como métricas a nivel de nodo, ya que las que son a nivel de red no pueden ser asignadas a cada nodo, ya que representan información sobre la red en general, y no sobre el usuario.

Una vez expuestas dichas características, se procederá a su cálculo modelando los *datasets* elegidos (particularmente el de amigos y el de seguidores) en un grafo dirigido de *NetworkX*.

Como recordatorio previo, hay que recordar que no se conoce de antemano la interacción que tienen los *fake followers* entre sí, por lo que las explicaciones a la introducción de características de redes sociales, son una mera premisa. La validez (o no) de las mismas, se determinará con posterioridad analizando la

importancia que aporta a las mismas el modelo creado.

Grado, grado de entrada y grado de salida

Debido a que se presupone que un *fake follower* tendrá probablemente muchos amigos pero no tantos seguidores, debido a que probablemente no se siguen entre ellos y siguen a muchos perfiles (por razones de rentabilidad económica de los *bots*), es probable que tenga una gran cantidad de amigos y una cantidad baja de seguidores, por lo que la inclusión de los grados de entrada, de salida y el vecindario (siendo este la suma de ambos) puede ser de utilidad. En particular, tal y como dedujeron [1, 13], un bajo grado de salida o de entrada, puede ser un comportamiento producido por un *fake follower*. Esta métrica es equivalente al número de *friends* y de *followers*, pero se opta por calcularlo debido a que cotejando ambas columnas, hay algunas diferencias sutiles en los valores de ambas, por lo que se observará en función de la importancia que el modelo asigne a estas características si se mantienen o se suprimen en la iteración final de dicho modelo.

Para el cálculo de estas métricas se emplearán las funciones indicadas en [33, 34, 35]. Estas métricas, en base al código fuente de *NetworkX*, implican recorrer para cada nodo sus vecinos. Debido a que para cada nodo, sus vecinos (sucesores y predecesores) se almacenan en diccionarios [36], y los diccionarios, debido a que almacenan en un atributo su longitud (sin necesidad de iterarlos para calcularla), el cálculo de cada nodo de sus nodos sucesores y precedesores es de complejidad constante ($\mathcal{O}(1)$) [37], por lo que hacer esta operación para cada nodo implica globalmente un coste lineal, esto es, una complejidad temporal de ($\mathcal{O}(n)$) donde n es el número de nodos que contiene la red. Hay que destacar, que esta obtención del número de elementos de los diccionarios, solo es aplicable si el grafo no tiene sus aristas etiquetadas con pesos, ya que entonces la manera de calcularlo variaría [38]. Se expone en el código 1 un ejemplo de cálculo de esta métrica (concretamente el grado de entrada implementado por *NetworkX*) [38]. Además, se puede observar que los autores devuelven como ge-

nerador el resultado (con el empleo de la función `yield`), por lo que permiten que posteriores operaciones puedan procesarse paralelamente, y no es necesario esperar a que la totalidad de los nodos estén computados para calcular otra métrica derivada, evitando además guardar información innecesaria en memoria RAM de las etapas intermedias calculadas [39]. De hecho, realmente, para poder incorporar la métrica al *dataframe* de *pandas*, tendrá que ser transformada a diccionario, siendo solo entonces ejecutada, ya que el uso del generador provoca que la función no es ejecutada hasta que el generador no es iterado.

```
1  def __iter__(self):
2      weight = self._weight
3      if weight is None:
4          for n in self._nodes: #O(n)
5              preds = self._pred[n]
6              #Crea un generador, permite optimizar operaciones a
                ↪  posteriori, pasandose a otras funciones como
                ↪  flujo de datos, evitando iterar varias veces.
7              yield (n, len(preds)) # O(1) para len(preds)
8
9      else:
10         for n in self._nodes:
11             preds = self._pred[n]
12             deg = sum(dd.get(weight, 1) for dd in
                ↪  preds.values())
13             yield (n, deg)
```

Código fuente 1: Implementación de grado de entrada por *NetworkX*

Grado medio del vecindario

Esta característica permite medir el grado medio que tiene el vecindario de un nodo dado. Esto puede ser relevante para medir cómo de importante es ese nodo. Se entiende que un nodo es muy importante en la red, ya que viene a medir de los seguidores y amigos que tiene, a cuántas personas siguen de media. Se prevé que esta métrica sea mayor para los *fake followers* ya que estos por su naturaleza seguirán a muchas cuentas y (sobre todo) cuentas muy influyentes, mientras que los usuarios legítimos, suelen ser mucho más selectivos, ya que además de seguir a personas con gran influencia, seguirán también a familiares y amigos, que por lo general tendrán una menor cantidad de seguidores y amigos, lo que provocaría que la media cayera significativamente. Esta característica se puede calcular mediante una función de *NetworkX* [40, 41]. Con respecto a la complejidad, observando el código fuente de dicha función, resulta que es $\mathcal{O}(n)$ donde n es el número de nodos. Esta característica ha sido finalmente introducida en el modelo.

Algoritmos de centralidad: centralidad de grado

La centralidad de grado consiste en contar el número de nodos que están conectados a un nodo dado. Esto es equivalente al cálculo realizado anterior del grado con una salvedad: es una métrica normalizada (cuyo valor oscila entre 0 y 1 siempre y cuando no haya bucles de ningún nodo consigo mismo). Esto se hace dividiendo el grado de cada nodo entre el grado que tendría cada nodo de un grafo completo (un grafo completo es n-1 regular) [42], esto es, el grado máximo teórico de un grafo totalmente conexo y sin bucles de ningún grafo consigo mismo [43].

Las ventajas e inconvenientes de aplicar esta métrica, son las mismas que las indicadas en el caso de las métricas de grado, añadiendo otro posible factor: esta métrica podría ser útil para el modelo por estar normalizada. Con posterioridad, se observará qué importancia arroja el modelo a esta métrica respecto a las demás.

Desde el punto de vista de la complejidad algorítmica temporal de esta métrica, se puede observar en el código fuente 2, obtenido del código fuente de *NetworkX* [44], que emplea simplemente un bucle, para hacer la operación de normalización. Sin embargo, hace esta normalización sobre el cálculo empleando la función del grado expuesta anteriormente, por lo que se podría pensar que hay dos bucles anidados (lo que implicaría una complejidad cuadrática, esto es, $\mathcal{O}(n^2)$) Pero como se expuso anteriormente, la función que genera los grados por cada nodo, devuelve un generador, por lo que realmente no es ejecutada, y la métrica actual sigue teniendo una complejidad temporal de $\mathcal{O}(n)$, siendo n la cantidad de nodos que hay en la red.

```
1  def degree_centrality(G):
2      if len(G) <= 1:
3          return {n: 1 for n in G}
4
5      s = 1.0 / (len(G) - 1.0)
6      centrality = {n: d * s for n, d in G.degree()}
7      return centrality
```

Código fuente 2: Implementación de centralidad de grado por *NetworkX*

Algoritmos de centralidad: centralidad de intermediación o *betweenness centrality*

Esta métrica indica el número de rutas más cortas entre dos nodos que pasan por otro nodo dado [2]. [13] indica al igual que [2] que el fundamento intuitivo por el que se debería incluir esta métrica, es que debido a la acción de un *fake follower* de crear nuevas relaciones con cuentas desconocidas, este está creando de forma indirecta nuevas rutas, por lo que estaría acortando rutas entre otros nodos y llevaría a dicho *fake follower* a ocupar un lugar más central en la red

que otras cuentas legítimas.

Respecto a la complejidad algorítmica, el algoritmo que se emplea en *NetworkX* según su documentación [45] y según su código fuente [46], es el creado por *Ulrik Brandes*. En [47], *Brandes* menciona que la complejidad algorítmica espacial del algoritmo para el cálculo de esta métrica es $\mathcal{O}(n+m)$, mientras que la complejidad temporal es de $\mathcal{O}(nm)$ (en grafos no etiquetados), donde n es el número de nodos que conforman la red y m es el número de aristas.

La complejidad anteriormente expuesta, justifica su no inclusión. El uso de esta característica fue probado por [2] en un modelo simplificado, habiendo sido clasificada por su modelo de robustez con una alta robustez, pero en un modelo con mayor escala es inviable, por lo que [13] optó por no introducirlo. En el caso del trabajo actual, se mantendrá el mismo criterio, habiendo verificado previamente a esta toma de decisión el coste computacional real de esta característica.

Algoritmos de centralidad: centralidad de vector propio o *eigenvector centrality*

La centralidad de vector propio, es una métrica que tiene como objetivo medir la relevancia de un nodo en la red [48]. Particularmente indica que un nodo es importante si sus vecinos lo son [48]. Así pues, se conjetura que los usuarios con una mayor centralidad de vector propio serán usuarios legítimos (esto es porque son más influyentes porque se rodean de gente influyente), siendo aquellos usuarios que tengan menores valores *fake followers*. *NetworkX* permite procesar esta características mediante la función descrita en [49]. No se ha localizado ninguna referencia relativa a la complejidad que tiene esta implementación. No se ha hecho ingeniería inversa sobre el código fuente, debido a que es bastante complejo (interactúan varias bibliotecas). No obstante, se ejecutó y la duración de dicha ejecución era asumible. Posteriormente, se verifica que para otras utilidades de generación y procesamiento de grafos, dicha complejidad es de $\mathcal{O}(v+e)$ donde v es el número de nodos y e es el número de aristas [21]. Esta métrica ha sido

incluida finalmente en el *dataset*.

Algoritmos de centralidad: centralidad de cercanía o *closeness centrality*

Esta característica permite medir para un nodo dado, la distancia media de ese nodo a todos los nodos restantes [50, 51] . Así pues, un valor reducido podría ser positivo puesto que este nodo ocupa un lugar central en la red, mientras que un nodo con una medida alta significaría que está muy alejado de algunos de los nodos de la red [51]. Se puede plantear que un usuario legítimo en principio sería más central que un *fake follower*. La función que calcula esta característica está presente en *NetworkX* [51]. Sin embargo, esta métrica no se incluirá en el modelo, ya que tiene una complejidad superior a lineal, esto es, una complejidad de $\mathcal{O}(ve)$, donde v es el número de nodos y e es el número de aristas [21].

Coeficiente de agrupación o *clustering coefficient*

Esta característica, conocida más concretamente como coeficiente de agrupación local, permite definir para cada nodo, cómo de interconectados están sus vecinos. Si sus vecinos están totalmente conectados entre ellos (lo que sería un clique), este valor será 1 y si ninguno de sus vecinos está directamente conectado con algún otro vecino, ese valor será 0 [52].

Esta medida funciona empíricamente en la medida de que los usuarios sueles seguir a amigos, familiares o conocidos. Estas cuentas a las que sigue, es probable que estén conectadas entre sí [2]. Sin embargo, este principio de las redes sociales no se observa en el caso de los *fake followers* ya que su funcionamiento se suele basar en seguir a personas aleatoriamente, por lo que no suelen "devolver" esta amistad [2].

Respecto al coste de este algoritmo, en [2] se hicieron pruebas sobre una red pequeña para medir su robustez (que era elevada) pero no se hizo a gran escala, ya que según se afirma, el coste computacional es elevado. En [13] se indica que esta característica es omitida debido a su coste computacional. Observando el

código fuente de *NetworkX* en el fichero en el que se implementa esta característica, se puede observar que su complejidad algorítmica es de $\mathcal{O}(n^3)$, debido al triple bucle definido (realmente son cuatro bucles entre la función principal y la auxiliar, pero en uno de ellos no se genera el resultado, si no que se posterga con el uso de `yield` [39]). Además, en [52] se expone que la complejidad es la afirmada en el peor caso. En el caso de esta característica, se esperarían valores altos para usuarios legítimos y bajos para *fake followers* [13, 2]. Por las razones anteriormente expuestas, finalmente se opta por no introducir esta característica en el *dataset*.

Algoritmos de prestigio: *PageRank*

Esta característica es una de las que representa el prestigio de cada nodo en particular en base al prestigio de los nodos que le siguen y la cantidad de enlaces que contienen los mismos [53]. Inicialmente se etiquetan los nodos con cualquier valor (aunque hay estrategias para optimizar esta asignación inicial) [53] y luego se calcula el valor de los nodos hasta que se llega a la convergencia (o a un número de iteraciones máximas) [53, 54]. Fue desarrollado por *Larry Page*. Se puede calcular con *NetworkX* [54] aunque no se ha localizado la complejidad de este algoritmo en su implementación. Se ha observado que en otras bibliotecas que permiten modelar grafos, esta complejidad es de $\mathcal{O}(e)$, donde e es el número de aristas [21]. Finalmente ha sido introducida en el *dataset*, debido al carácter lineal de su complejidad y debido a que se ha logrado computar en el *dataset* original en un tiempo muy razonable.

Algoritmos de prestigio: *HITS*

Esta característica también representa el prestigio de cada nodo en particular, pero está compuesto de dos métricas: el valor de autoridades, que estima el valor del nodo en base a los enlaces entrantes, y el valor de *hubs* que estima el valor del nodo en base a los enlaces salientes [55, 56]. Al igual que el *PageRank*, el algoritmo es iterativo y finaliza cuando las métricas convergen (o llegan a

un número máximo de iteraciones) [55, 56]. Se puede calcular con *NetworkX* [54] aunque no se ha localizado la complejidad de este algoritmo en su implementación. La complejidad algorítmica genérica, al igual que *PageRank* es de $\mathcal{O}(e)$, donde e es el número de aristas [**?**]. Finalmente ha sido introducida en el *dataset*, debido al carácter lineal de su complejidad y debido a que se ha logrado computar en el *dataset* original en un tiempo muy razonable.

Reciprocidad

La reciprocidad en la ciencia de redes sociales, consiste en una métrica que analiza el número de enlaces que hay de forma mutua respecto a la cantidad de enlaces que hay [31]. Es muy importante el análisis de esta característica, ya que existen investigaciones que destacan el valor de las mismas [31]. Particularmente, cabría destacar que un usuario que tiene un alto grado de reciprocidad tiene mucho más en común con sus *friends*, mientras que uno que tenga una baja reciprocidad, tendrá menos en común, por lo que es más probable que sea un *fake follower*.

No se va a calcular esta métrica mediante *NetworkX* ya que ya se calculó anteriormente, mediante la replicación de las métricas del modelo de partida. La métrica en concreto, se llamó "tasa de enlaces bidireccionales".

Tríadas

Las tríadas son la menor estructura de redes sociales que puede ser considerada sociedad [57]. El análisis de las tríadas y la prevalencia de diferentes tipos de tríadas en las poblaciones ha sido un elemento básico de la sociometría y el análisis de redes sociales [57]. En las tríadas (dirigidas), podemos ver la aparición de tendencias hacia el equilibrio y la consistencia, la institucionalización, de las estructuras sociales (equilibrio y transitividad) [57]. Las tríadas son también las estructuras más simples en las que podemos ver el surgimiento de la jerarquía [57]. Es por estas razones, que verificar las triadas de las que forma parte cada nodo es una métrica que puede revestir importancia a la hora

de realizar un modelo, especialmente al hacer un modelo de *machine learning* donde la relación entre los usuarios puede ser clave.

En *NetworkX* no existe una métrica que obtenga directamente para cada nodo en cuantas triadas de cada tipo participa. Sin embargo, *NetworkX* tiene la función *triadic_census()* [58] que permite obtener para una red dada, la cantidad de triadas que hay que cada tipo, con una complejidad declarada de $\mathcal{O}(m)$, donde m es el número de aristas que tiene la red. Esta complejidad se basa en el uso del algoritmo propuesto por *Vladimir Batagelj* y *Andrej Mrvar* [59].

No obstante, en base a propuestas de ciertos usuarios [60] derivadas de esta funcionalidad de *NetworkX*, se implementó el algoritmo para almacenar por cada nodo las triadas de cada tipo en el *dataframe* de *Pandas*. Sin embargo, no se logró un algoritmo lo suficientemente rápido como para que a pesar de tener un comportamiento lineal tuviera un rendimiento aceptable (solamente se logró computar el uno por ciento de los datos tras 24 horas de proceso). Además de almacenar las participaciones por triada de cada nodo, se hicieron las siguientes intervenciones en aras de mejorar la rapidez del algoritmo:

- Se redujeron las iteraciones del primer bucle, minimizando las iteraciones innecesarias calculando solo las triadas partiendo de los nodos que forman parte del conjunto de usuarios que se desean introducir al modelo. Para ello, también se modificó la variable m que contiene el conjunto de elementos del grafo, numerados, para permitir hacer una poda y no recalcular triadas desde otros nodos. Esta modificación se hizo para que los nodos configurados en el primer nodo estuvieran los primeros en la numeración, para favorecer la probabilidad de que el algoritmo fuera podado más veces (reduciendo el tiempo medio de ejecución).

- Se prescindió de las triadas de tipo 003, que como se puede observar en la ilustración 3.1 son aquellas en las que todos los nodos están inconexos, ya que son triadas más caras de computar, pues hay que verificar la relación de cada nodo con los restantes nodos de la red. Por este mismo criterio, también se evitó la computación de los tipos 012 y 102, en la que solamente

dos nodos están conectados.

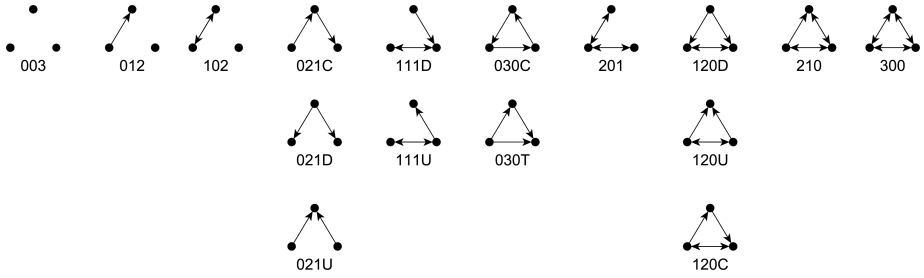

Ilustración 3.1: Tipos de triadas (imagen producida por [61])

Las intervenciones enumeradas anteriormente no surtieron efecto, por lo que propone esto como una posible línea de investigación en el futuro. La problemática estriba en el uso de los bucles junto a los conjuntos empleados por el algoritmo, que no aportan a priori un rendimiento significativo.

Cliques

Los *cliques* son subgrafos que tienen la característica de ser completos . El planteamiento inicial podría ser verificar para cada nodo, si forma parte de un clique con un tamaño particular, lo que proporcionaría una característica para determinar si existe una agrupación o no, que se entiende más propensa en el caso de usuarios legítimos, y menos propensa en el caso de *fake followers*. Sin embargo, no se calculará esta métrica por dos razones:

1. La primera razón es que habría que determinar de qué tamaño se desean calcular los cliques. No existe una respuesta universal, por lo que habría que, mediante ensayo y error, obtener el número de elementos que deben participar en los cliques, para que ofrecer mayor información al modelo (si es que la aportan).

2. La segunda razón es que, aun habiendo determinado este número concreto, la búsqueda de cliques es un problema np-completo [62], lo que implica

que habría que probar todos los subconjuntos del grafo para llegar a la solución, no habiendo una solución polinomial para llegar a dichos cliques.

3.2.5. Relación final de las nuevas características de redes sociales evaluadas

A continuación se muestra una tabla en la que para cada métrica de redes sociales de las analizadas en la sección anterior, detallando la complejidad algorítmica y si se incluye o no dicha métrica en la primera fase del modelo desarrollado. En la complejidad algorítmica expresada en dicha tabla, v es el número de nodos y e es el número de aristas.

Característica	**Complejidad**	**¿Incluido?**
Grado	$\mathcal{O}(n)$	Sí
Grado de entrada	$\mathcal{O}(v)$	Sí
Grado de salida	$\mathcal{O}(v)$	Sí
Grado medio del vecindario	$\mathcal{O}(v)$	Sí
Centralidad de grado	$\mathcal{O}(v)$	Sí
Centralidad de intermediación *betweenness centrality*	$\mathcal{O}(ve)$	No
Centralidad de vector propio *eigenvector centrality*	$\mathcal{O}(v+e)$	Sí
Centralidad de cercanía *closeness centrality*	$\mathcal{O}(ve)$	No
Coeficiente de agrupación *clustering coefficient*	$\mathcal{O}(v^3)$	No
page rank	$\mathcal{O}(e)$	Sí
HITS	$\mathcal{O}(e)$	Sí
Reciprocidad	–	No
Tríadas	$\mathcal{O}(e)$	No
Cliques	NP-completo	No

Como se puede observar, se van a introducir en la primera fase del modelo, todas las métricas cuyo desempeño en su complejidad temporal sea lineal. La única excepción a esta regla, es el uso de tríadas, que no se incluirán debido a que el algoritmo base es bastante lento (a pesar de ser lineal) y los retoques al mismo (se podía retocar puesto que no estaba en *NetworkX*) no surtieron efecto en hacer viable el procesamiento en un tiempo prudencial.

3.3. Construcción de la red

Para construir la red, se han fusionado los *dataframes* que modelan las relaciones de *friends* y de *followers* (invirtiendo el orden de las columnas de uno de ellos) y se ha generado la red en base a esas aristas. Respecto a la posible preocupación sobre la duplicidad de las aristas de *NetworkX*, la propia biblioteca se encarga de no incluir duplicados en la red.

De esta forma, se ha construido una red que modela las amistades y que permite construir las métricas de redes sociales elegidas en el apartado anterior.

3.4. Creación del modelo

En esta sección se va a detallar con detenimiento la creación del modelo, tanto los planteamientos previos como la explicación sucinta de su desarrollo en las distintas fases de refinamiento.

3.4.1. Introducción

La creación del modelo implica el uso de las características desarrolladas previamente así como el uso de un clasificador. En esta sección se detallará qué clasificador se empleará, como se evaluará su desempeño y los diversos pasos para obtener el clasificador final.

3.4.2. Métricas de evaluación del modelo

Antes de empezar con el entrenamiento del modelo predictivo de clasificación, es conveniente destacar ciertas métricas que permitirán evaluar el desempeño de dicho modelo. Dichas características se expondrán a continuación contextualizadas en el problema que se está resolviendo.

Precisión (*accuracy*)

La precisión representa la proporción de los resultados correctamente clasificados (tanto usuarios legítimos como *fake followers*) sobre el total de usuarios en el *dataset*.

Exactitud (*precision*)

La exactitud representa la proporción de usuarios detectados como *fake followers* que realmente lo son entre el total de resultados detectados como *fake followers*.

Exhaustividad (*recall*)

La exhaustividad representa la proporción de usuarios que han sido detectados como *fake followers* que realmente lo eran entre el total de resultados que son *fake followers*, hayan sido o no detectados.

Área bajo la curva (*AUC*)

Esta métrica representa el rendimiento del clasificador considerando el porcentaje de positivos ciertos comparado con el porcentaje de falsos negativos [13]. El AUC se emplea para resumir la curva ROC en un solo valor, de tal manera que cuanto más se acerca a 1 dicho valor, más eficaz es el clasificador [13].

3.4.3. Elección del clasificador

Debido a que los estudios previos analizados reportan consistentemente que *Random Forest* destaca sobre los demás clasificadores [13, 1, 2, 30, 16] y debido

también a sus características teóricas [63], con el fin de poder hacer comparación entre semejantes, sin variar el modelo (solo modificando las características de entrada al modelo), se opta por *Random Forest* como clasificador a emplear en el desarrollo de este modelo de *machine learning*.

3.4.4. Balance entre las clases de los datos

Debido al reparto más o menos balanceado del *dataset* entre la clase de usuarios legítimos (cerca del 40 % de los usuarios) y la de *fake followers* (algo más del 60 % de los usuarios), no se considera necesario alterar esta distribución de los datos para mejorar la eficiencia del modelo.

3.4.5. Proceso de creación del modelo

El modelo creado se ha realizado realizando cuatro iteraciones que serán detalladas a lo largo de esta sección. Respecto al software utilizado, hay que destacar que además del uso de *pandas* para el tratamiento de datos y de *NetworkX* para la creación de la red, se ha empleados *sklearn* para la medición, desarrollo y ejecución de los modelos de *machine learning*.

Tratamiento previo de los datos

En primer lugar, se realizan una serie de verificaciones sobre los datos, con el fin de determinar si están listos o no para ser introducidos en el modelo. Particularmente, se verifica que los datos no tengan ningún nulo y que sean del tipo de dato correcto. Hay que destacar que esta es una comprobación cautelar, ya que el tratamiento de los posibles nulos o valores que pudieran ser anómalos ya ha sido tenido en cuenta en el cálculo de las métricas.

Una vez que se ha comprobado que los pasos anteriormente mencionados son correctos, se pasa a verificar la correlación de las variables. En una primera fase de la verificación de la correlación, se observa la correlación de cada variables respecto a las demás, obteniendo los resultados que se pueden observar en la ilustración 3.2.

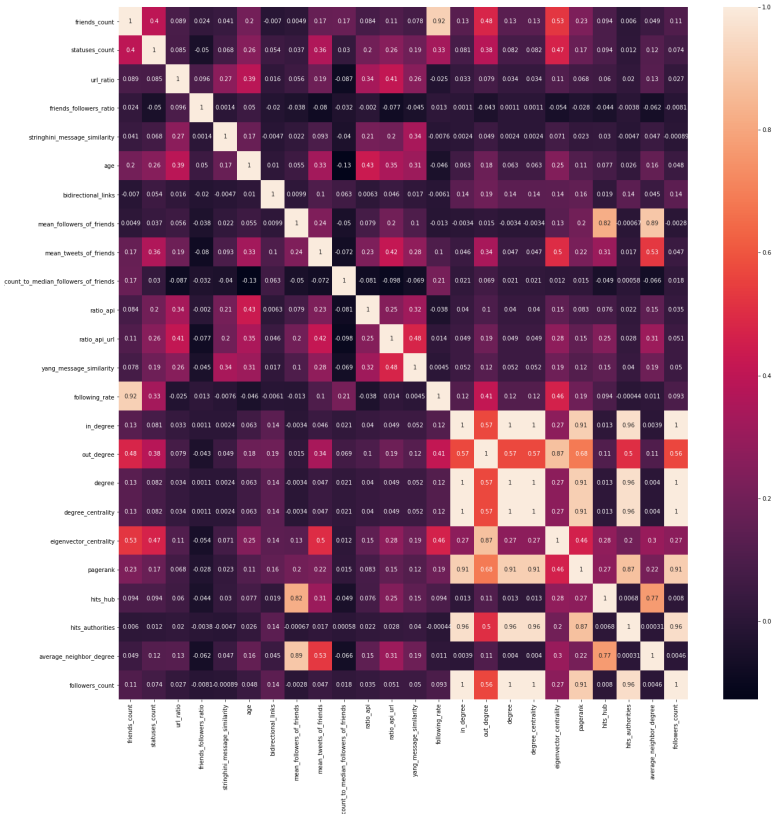

Ilustración 3.2: Verificación de la correlación (I)

Como se puede apreciar, hay variables que correlacionan exactamente 1, como pueden ser `friends_count`, `in_degree`, `degree` y `degree_centrality`. Se procede a suprimir todas exceptuando `followers_count`, ya que para las restantes es necesario computar los datos mientras que esta última proviene directamente del *dataframe*. Hay que destacar, que como se habrá podido observar en la sección relativa a las métricas, esta métrica relativa a los *followers* no aparecía. Esto es porque se ha introducido posteriormente para verificar si la correlación entre el grado de entrada y esta era elevada (como así se ha podido observar) y emplearla en caso afirmativo. Esto es debido a que cuando se realizó el cálculo de las variables, se verificó que las variables `followers_count` y `friends_count` no coincidían exactamente con el grado de entrada y salida respectivamente, aunque no diferían mucho. Dado que `friends_count` se heredaba de [13] vía [1], se explicó, mientras que la otra no se explicó porque no procedía del cálculo de una métrica mediante la red ni provenía del modelo de partida, si no que fue una medida discrecional.

Tras esta primera fase, se procede a hacer otra segunda fase, en la que se observa en la ilustración 3.3 que no hay ninguna variable que correlacione exactamente con otra que no sea ella misma. Habrá algunas que se acerquen bastante, pero no se van a suprimir, ya que pueden ser las que aporten pequeñas diferencias al modelo. Estas pequeñas diferencias pueden ser importantes ya que el margen de mejora del modelo preexistente realizado por [13] es reducido.

Fase I del modelo

Tras realizar la pertinente partición, se ejecuta el modelo y se evalúa tanto en el conjunto de entrenamiento con en el de test. Se observa que mediante una validación cruzada de tipo *10-fold* en la que se generan diez particiones ordenadas, en las que el 90 % de los datos son de entrenamiento y el 10 % restante se emplea para validación, se obtenía una precisión entorno a dos puntos porcentuales menor que lo arrojado por el modelo con la partición escogida aleatoriamente. Se decide investigar esto, ya que puede haber dos circunstancias

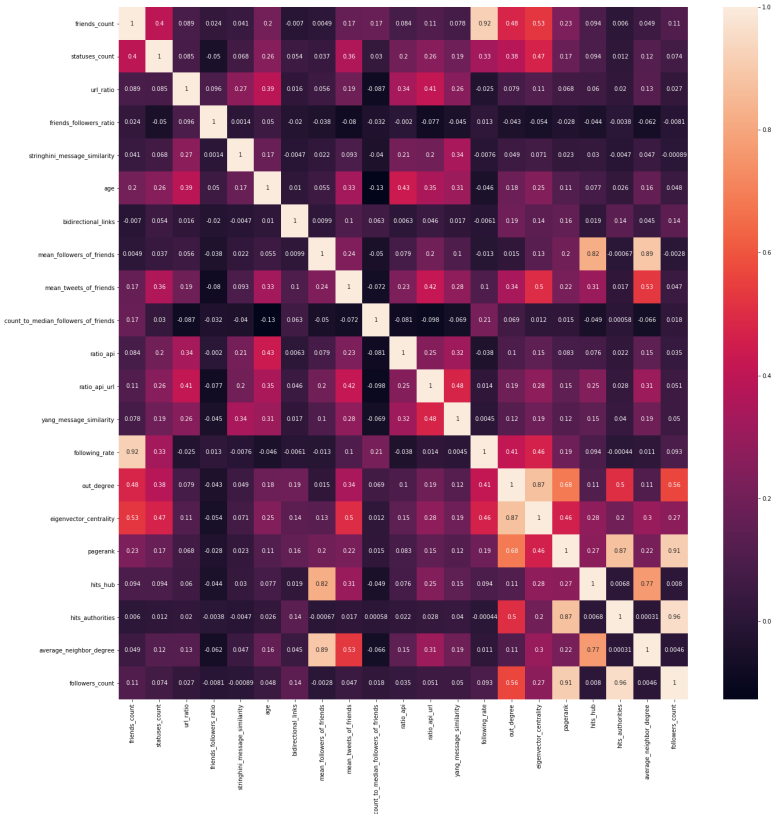

Ilustración 3.3: Verificación de la correlación (II)

que se pueden estar superponiendo:

- Que la partición escogida de forma pseudoaleatoria para entrenamiento y para validación tengan un reparto que beneficie especialmente los parámetros de evaluación del modelo.
- Que el *dataframe* de partida tenga los datos distribuidos de forma no uniforme.

Se va a comprobar en primer lugar este último aspecto, ya que es el más sencillo de probar. Para ello, se indica a la función de validación cruzada que indique la información del criterio medido (precisión) para cada una de las particiones generadas. Es allí, donde se observa, que el resultado decae porque una de las diez particiones tiene un resultado peor, afectando este *outlier* de forma significativa a la media. Para confirmar que esto se deba a una distribución no uniforme de los datos, se procede a realizar un *scatterplot* en base a la única variable conocida que puede arrojar la pista sobre esto (si el usuario es o no es un *fake follower*) para ver cómo se distribuye. Si se observa una nube en la que no se observa una tendencia, quiere decir que los datos fueron mezclados en el *dataframe*. Si se observa un par de líneas, quiere decir que ha sido concatenado.

Como se puede observar en la distribución mostrada en la ilustración 3.4, efectivamente, los datos no están uniformemente distribuidos, lo que justifica que una validación cruzada ordenada, no es suficiente para garantizar que la distribución tomada de los datos no haya sido beneficiosa. Es más, introduce un cierto sesgo, que luego a la hora de comparar modelos entre sí mediante esta validación cruzada no permite una buena comparabilidad con el modelo desarrollado en [13]. Particularmente, tras observar esto, y repasando el tratamiento previo realizado de los *datasets*, se observa que este fenómeno se produce por la fusión de los cinco subconjuntos de datos que conforman este *dataset*, ya que en su momento cuando se realizó no fue previsto este tipo de problemas.

Así pues, para solventar este problema se puede optar por dos soluciones. Una de ellas sería hacer una ordenación aleatoria del modelo, con el fin de que la validación cruzada *10-fold* no se vea afectada por la ordenación previa

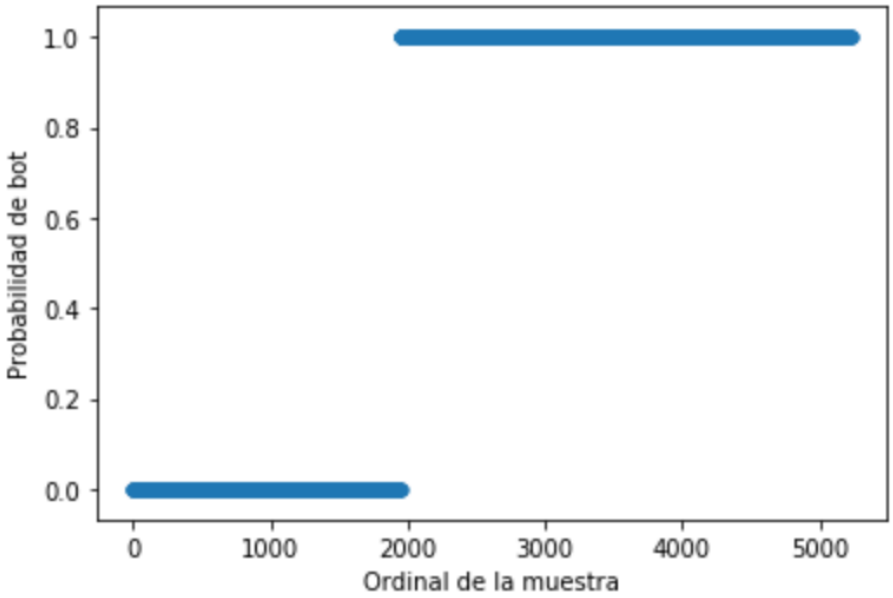

Ilustración 3.4: Distribución de los datos según la variable `is_bot`

de los datos contenidos en el *dataframe*. La otra solución, estriba en el uso de otra función de validación cruzada que obtenga las particiones distribuidas aleatoriamente. En este caso se opta por esta última, ya que permite probar otras formas de validación cruzada.

Tras emplear una validación cruzada que toma las particiones de forma aleatoria (manteniendo la proporción 90-10 mencionada anteriormente) con 50 particiones, se observa que las métricas son mucho más coherentes y son cercanas a los valores obtenido mediante la partición en los dos subconjuntos. No obstante, en los resultados, se mostrarán solamente los resultados obtenidos a través de la validación cruzada, al igual que hace [13], ya que estos datos, son más agnósticos de la partición realizada para hacer el entrenamiento y la validación al uso.

Fase II del modelo

En esta segunda fase, se suprimen las características que aportan menos de un 1 % al modelo, en base a la importancia de las mismas realizada por el propio modelo. Con esto se busca reducir la dimensionalidad del modelo, aunque teniendo como propósito no degradar significativamente la predictibilidad del mismo. De esta manera, se pasa de 21 variables a solo 12. Se ha observado que con la reducción del 43 % de las variables, el modelo no varía significativamente, mejorando en algunas métricas de evaluación y empeorando en otras.

Fase III del modelo

En esta fase, se simplificará el modelo, estableciendo ciertos parámetros de forma fija (mediante prueba y error) de tal manera, que se va observando su mejora. Estableciendo de una cierta manera el número de estimadores y la profundidad máxima, se consigue que no haya un deterioro significativo de los datos, a la vez que se consigue un modelo más simple (lo que beneficia a la hora de minimizar el *overfitting*.

Fase IV del modelo

En esta última fase del modelo, se propone mejorar otros parámetros, pero mediante el empleo de `GridSearchCV`. No se observa que haya habido ningún cambio respecto a la iteración anterior. Se ha probado en particular la verificación del número de características y la característica de *bootstrap* que determina para cada árbol construido si entrenarlo con una partición de los datos o con todo el conjunto [64].

3.5. Construcción de caso real

3.5.1. Introducción

Una vez que se ha creado el modelo final para la predicción de *fake followers*, se procede a emplear dicho modelo para analizar un caso real.

Dicho caso real, ha sido extraído en el contexto de las elecciones generales de España celebradas el 10 de noviembre de 2019. Para dicho caso real, se extrajo la información de *Twitter*, se seleccionó un subconjunto de la extracción relativo a los debates preelectorales y posteriormente se transformaron los datos para tener el mismo formato de entrada (CSV) que los *datasets* empleados para entrenar el modelo.

3.5.2. Obtención de *tweets* de *Twitter*

La extracción del *dataset* base de *Twitter* se ha basado en la extracción de los *tweets* de las siguientes dos maneras:

1. Se parte de una **lista de usuarios de confianza**, que tienen un marcado carácter político, y que además de capturarse todos los *tweets* que hayan escrito, *retwiteado* o respondido sus integrantes en tiempo real, se almacenarán los *hashtags* que empleen en una **lista de *hashtags* relevantes**. Esta lista de usuarios tiene la confianza suficiente para "crear temas de interés", ya que se presupone que sus interacciones serán de interés político, que es el objetivo a capturar.

2. Habrá otros usuarios, de los que se obtendrán sus *tweets* en tiempo real empleando como filtro que aparezcan en la lista de *hashtags* relevantes. Esto es, solo se insertará un *tweets* obtenido a través de esta captura si algunos de sus *hashtags* ya estaban previamente en la lista de *hashtags* relevantes, si no se descartará.

Cada vez que un *tweet* es almacenado, se incrementa un número (que se denominará **relevancia**) asociado a cada *hashtag* en base a la procedencia o no de la lista de usuario de confianza y al número de seguidores que tenga el autor del *tweet*. Dicha relevancia solo se podrá actualizar si una autoridad ya insertó previamente dicho *hashtag*. Para garantizar la captura de los temas más relevantes en el momento (lista de *hashtags* relevantes), el paso del tiempo debe reducir la relevancia de los mismos cuando se deje de hablar de ellos. Para esto existe

un **algoritmo de envejecimiento** que reduce periódicamente la relevancia. Cuando los temas son poco relevantes, hay un **umbral** bajo el que dejan de capturarse en la lista de usuarios generales.

Discrecionalmente, se podrán excluir de la lista de *hashtags* relevantes aquellos que no sean de interés para la extracción debido al marcado carácter generalista que tienen (tales como *#FelizJueves o #france* podrían ser dos ejemplos). Para cumplir este objetivo, existe una **lista de exclusión**.

Los *tweets* obtenidos mediante este algoritmo forman parte de la extracción base, en la que hay un montón de *tweets* que tenderán a ser de carácter político.

3.5.3. Filtrado de los datos

Tras obtener este *dataset* base de *tweets*, se han filtrado dichos *tweets* en base a los *hashtags* relativos a los debates. Para ello, se han extraído los *hashtags* más empleados por cada día, y se han escogido aquellos que tienen relación con dichos debates

Este *dataset* de *tweets* es el definitivo. No obstante, por limitaciones con la memoria RAM al cargar el grafo y procesarlo, solamente se han mantenido aquellos *tweets* que han sido escritos por usuarios de los que se han guardado más de 5 *tweets*, siendo el resto eliminados. Se emplea este criterio, porque cuanta más interacción haya, más interesante puede ser detectar si el usuario es legítimo o no.

3.5.4. Obtención de usuarios, *friends* y *followers*

Una vez que se han obtenido los *tweets*, hay que obtener el resto de información necesaria. En el caso de los usuarios, esa información se obtiene embebida en el propio *tweet*, por lo que no es necesario invocar a *Twitter* para obtener, puesto que con un tratamiento sobre la base de datos sería suficiente.

Lo único que faltaría sería la obtención de *friends* y followers, núcleo esencial para la construcción de la red en *NetworkX*. Dicha obtención, se realiza invocando a la API de *Twitter* por cada usuario al menos una vez a cada *endpoint*. Esta

información se almacena en una base de datos documental, para posteriormente (al igual que con los *tweets* y los usuarios) extraerla y almacenarla en ficheros con el mismo formato que el empleado para el entrenamiento del modelo.

Estos ficheros de relaciones de *friends* y *followers* conforman las aristas de la red que se va a crear. Los nodos son los usuarios de fuente y de destino mientras que una arista es cada línea de estos ficheros. Con estos ficheros importados como *dataframes*, se invoca una función de *NetworkX* que se encarga de crear los nodos necesarios (sin duplicados) y de crear las aristas necesarias en un grafo dirigido. De esta forma se genera una red de amistades.

Una vez creada la red, se podrá trabajar con ella para la obtención de las características de redes sociales que se requieran para todos sus nodos. No obstante, dado que para la mayoría de los nodos solamente se conoce la mera conexión con su vecindario, no se podrán categorizar todos ellos (al no tener la información de los *tweets* ni el resto de información de usuario que pueda ser relevante. En base a esto, en la sección de resultados que caracterice la red, se hará distinción entre el número de usuarios (el que se ha obtenido de los autores de los *tweets*, y que serán el objeto de clasificación) y el número de nodos (que salvo los usuarios anteriormente mencionados, solamente ayudarán formando parte de las métricas de redes sociales, pero no serán objeto de clasificación).

3.5.5. Precisiones sobre las APIs de *Twitter*

Respecto a la descarga de información de los *tweets*, se empleó la API de *Twitter Streaming*, que permitió descargar los datos en tiempo real, almacenándolos directamente como JSON en la base de datos. Respecto a los usuarios, estos vienen en el cuerpo del *tweet*, por lo que no es necesaria su extracción. Para la obtención de los *friends* y los *followers*, se hará invocando a la API de *Twitter*. Aquí es donde radica la mayor dificultad, ya que la API solamente permite 15 invocaciones cada 15 minutos y a su vez cada invocación solamente permite obtener hasta 1.000 usuarios [17, 18]. Esto hace que en usuarios grandes sea necesario paginar la petición, y que para muchos usuarios, el tiempo de

descarga se extienda, ya que la API actúa de cuello de botella en este sentido.

Capítulo 4

Resultados

En este capítulo se expondrán los resultados que se han obtenido. Estará dividida en dos secciones. La primera sección versará sobre los resultados obtenidos en la creación del modelo, indicando las características de evaluación del modelo así como todas las características empleadas como entradas del modelo en cada una de las iteraciones realizadas. La segunda versión consistirá en la muestra de aquellos detalles estadísticos que se consideren de interés sobre el caso real, ya que no podrá ser evaluado (por carecer de etiquetas que determinen la clasificación de los usuarios según su legitimidad).

4.1. Resultados obtenidos en la creación del modelo

Debido a que el modelo ha sido realizado en varias fases, se expondrán los datos para cada una de las iteraciones.

4.1.1. Fase I del modelo

Esta fase del modelo ha empleado las siguientes características:

- friends_count
- statuses_count
- url_ratio
- friends_followers_ratio
- stringhini_message_similarity
- age
- bidirectional_links
- mean_followers_of_friends
- friends_count_to_median_followers_of_friends
- ratio_api
- ratio_api_url
- yang_message_similarity
- following_rate
- out_degree
- eigenvector_centrality
- pagerank
- hits_hub
- hits_authorities
- average_neighbor_degree
- followers_count

Las métricas de evaluación obtenidas por el modelo son las siguientes (redondeando a las milésimas) obtenidas mediante 50 particiones aleatorias en las que los datos se reparten en un 90 % para el entrenamiento y un 10 % para la validación:

Métrica	**Valor**
Precisión (*accuracy*)	0,995
Exactitud (*precision*)	0,998
Exhaustividad (*recall*)	0,994
Área bajo la curva	1,000

El modelo empleado es un *Random Forest* sin ninguna modificación respecto a los parámetros por defecto en la implementación de *sklearn*.

4.1.2. Fase II del modelo

Esta fase del modelo ha empleado las siguientes características:

- `friends_count`
- `statuses_count`
- `url_ratio`
- `friends_followers_ratio`
- `age`
- `mean_followers_of_friends`
- `ratio_api_url`
- `following_rate`
- `eigenvector_centrality`
- `pagerank`
- `hits_authorities`
- `followers_count`

Las métricas de evaluación obtenidas por el modelo son las siguientes (redondeando a las milésimas) obtenidas mediante 50 particiones aleatorias en las que los datos se reparten en un 90 % para el entrenamiento y un 10 % para la validación:

Métrica	**Valor**
Precisión (*accuracy*)	0,996
Exactitud (*precision*)	0,998
Exhaustividad (*recall*)	0,995
Área bajo la curva	1,000

El modelo empleado es un *Random Forest* sin ninguna modificación respecto a los parámetros por defecto en la implementación de *sklearn*.

4.1.3. Fase III del modelo

Esta fase del modelo ha empleado las siguientes características:

- `friends_count`
- `statuses_count`
- `url_ratio`
- `friends_followers_ratio`
- `age`
- `mean_followers_of_friends`
- `ratio_api_url`
- `following_rate`

- `eigenvector_centrality`
- `hits_authorities`
- `pagerank`
- `followers_count`

Las métricas de evaluación obtenidas por el modelo son las siguientes (redondeando a las milésimas) obtenidas mediante 50 particiones aleatorias en las que los datos se reparten en un 90 % para el entrenamiento y un 10 % para la validación:

Métrica	**Valor**
Precisión (*accuracy*)	0,995
Exactitud (*precision*)	0,997
Exhaustividad (*recall*)	0,995
Área bajo la curva	1,000

El modelo empleado es un *Random Forest* con las siguientes modificaciones respecto a los parámetros por defecto en la implementación de *sklearn*:

- Número de estimadores (`n_estimators`): 6
- Profundidad máxima (`max_depth`): 7

4.1.4. Fase IV del modelo

Esta fase del modelo ha empleado las siguientes características:

- `friends_count`
- `ratio_api_url`
- `statuses_count`
- `following_rate`
- `url_ratio`
- `eigenvector_centrality`
- `friends_followers_ratio`
- `pagerank`
- `age`
- `hits_authorities`
- `mean_followers_of_friends`
- `followers_count`

Las métricas de evaluación obtenidas por el modelo son las siguientes (redondeando a las milésimas) obtenidas mediante 50 particiones aleatorias en las que los datos se reparten en un 90 % para el entrenamiento y un 10 % para la validación:

Métrica	**Valor**
Precisión (*accuracy*)	0,995
Exactitud (*precision*)	0,997
Exhaustividad (*recall*)	0,995
Área bajo la curva	1,000

El modelo empleado es un *Random Forest* optimizado con *GridSearchCV*, aunque dicha optimización no ha producido ningún cambio en los parámetros desde la fase anterior.

La importancia que asignó el modelo a las diversas características que permanecieron hasta esta fase en el modelo se detalla en la siguiente tabla:

Característica	**Importancia**
`friends_followers_ratio`	22,1 %
`hits_authorities`	20,63 %
`followers_count`	20,36 %
`statuses_count`	14,69 %
`friends_count`	7,49 %
`age`	7,23 %
`following_rate`	4,19 %
`pagerank`	2,34 %
`mean_followers_of_friends`	0,48 %
`url_ratio`	0,29 %
`eigenvector_centrality`	0,16 %
`ratio_api_url`	0,03 %

Respecto a las métricas de redes sociales generadas mediante un grafo de *NetworkX*, han pasado a formar parte del modelo las que se observan a continuación:

- `hits_authorities`, siendo esta la más importante de todas para el modelo, muy por encima de las restantes.
- `pagerank`.
- `eigenvector_centrality`, residual.

Las que finalmente no forman parte del modelo son las siguientes:

- `in_degree`: Eliminada en la fase previa de correlaciones.
- `out_degree`: Eliminada en la fase previa de correlaciones.
- `degree`: Eliminada en la fase previa de correlaciones.
- `degree_centrality`: Eliminada en la fase previa de correlaciones.
- `hits_hub`: Poco relevante para el modelo
- `average_neighbor_degree`: Poco relevante para el modelo.

Si se contabilizan todas las métricas de redes sociales (o aquellas derivadas de estas), y no solamente aquellas que provengan del grafo realizado, se observa que el modelo final aparecen las siguientes:

- `friends_count`
- `followers_count`
- `friends_followers_ratio`
- `mean_followers_of_friends`
- `eigenvector_centrality`
- `pagerank`
- `hits_authorities`

4.2. Resultados del caso real

A continuación se expondrán unas cifras del propio conjunto de datos extraído, para posteriormente exponer los datos finales de la predicción.

4.2.1. Datos de interés del conjunto de datos final

A continuación se observan algunos detalles del conjunto de datos empleado para el caso real.

Métrica	**Valor**
Número de usuarios	21.104
Número de *tweets*	227.374
Número de relaciones de *friends*	21.692.133
Número de relaciones de *followers*	41.934.844
Número final de nodos de la red	12.578.188
Número de aristas de la red	62.153.070

4.2.2. Detalles de la predicción del caso real

Se procede a exponer una serie de datos de interés sobre la predicción realizada por el modelo entrenado anteriormente de los usuarios que son legítimos y de los *fake followers*.

En la ilustración 4.1 se puede observar cómo se distribuyen los usuarios en el *dataset* analizado. Se puede observar que el clasificador cataloga como usuarios legítimos a 20.819 usuarios, lo que representa el 98,65 % Por el contrario, cataloga como *fake followers* a 285 usuarios, que suponen el 1,85 % restante de los datos.

A continuación, se puede observar en la 4.2 la distribución de la probabilidad de ser un usuario legítimo estratificada en 5 agrupaciones. Posteriormente se observa en una tabla la misma información, pero estratificada en 10 agrupaciones.

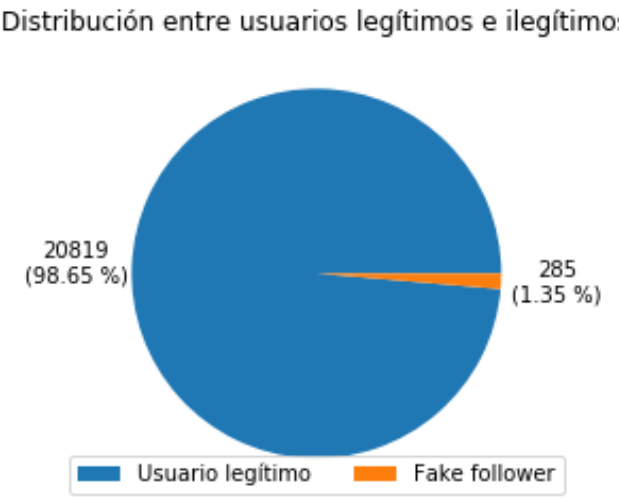

Ilustración 4.1: Distribución de usuarios según su legitimidad

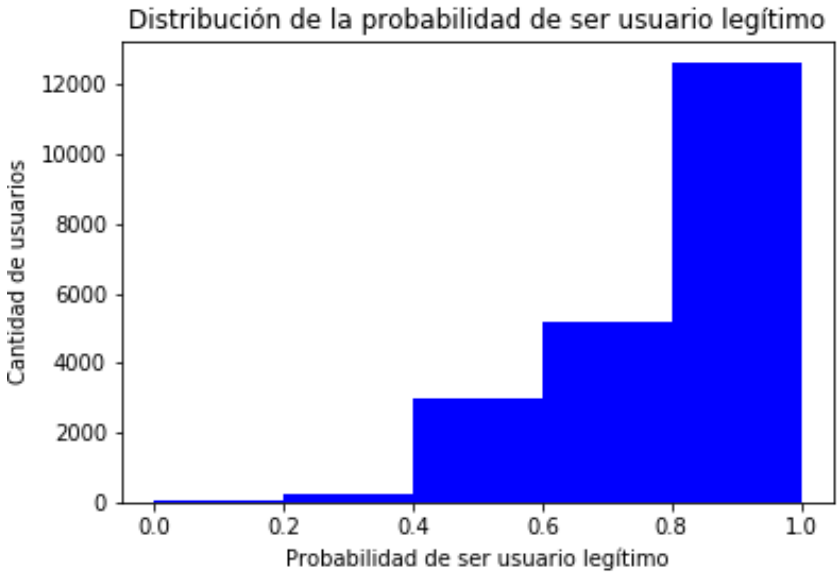

Ilustración 4.2: Distribución de la probabilidad de ser un usuario legítimo

Prob. legítimo	**Número de usuarios**
(90 %-100 %]	7.573
(80 %-90 %]	5.048
(70 %-80 %]	130
(60 %-70 %]	5.057
(50 %-60 %]	3
(40 %-50 %]	3.008
(30 %-40 %]	225
(20 %-30 %]	0
(10 %-20 %]	51
[0 %-10 %]	9

Por último, se puede observar en la 4.3 la distribución de la probabilidad de ser un usuario ilegítimo estratificada en 5 agrupaciones. También se la probabilidad estratificada en 10 agrupaciones.

Prob. *fake follower*	**Número de usuarios**
(90 %-100 %]	9
(80 %-90 %]	51
(70 %-80 %]	0
(60 %-70 %]	225
(50 %-60 %]	0
(40 %-50 %]	3.011
(30 %-40 %]	5.057
(20 %-30 %]	130
(10 %-20 %]	5.048
[0 %-10 %]	7.573

Respecto a la clasificación de usuarios, se pueden mencionar como ejemplos dos usuarios en los que el modelo ha clasificado con la máxima probabilidad que ha podido como usuarios legítimos:

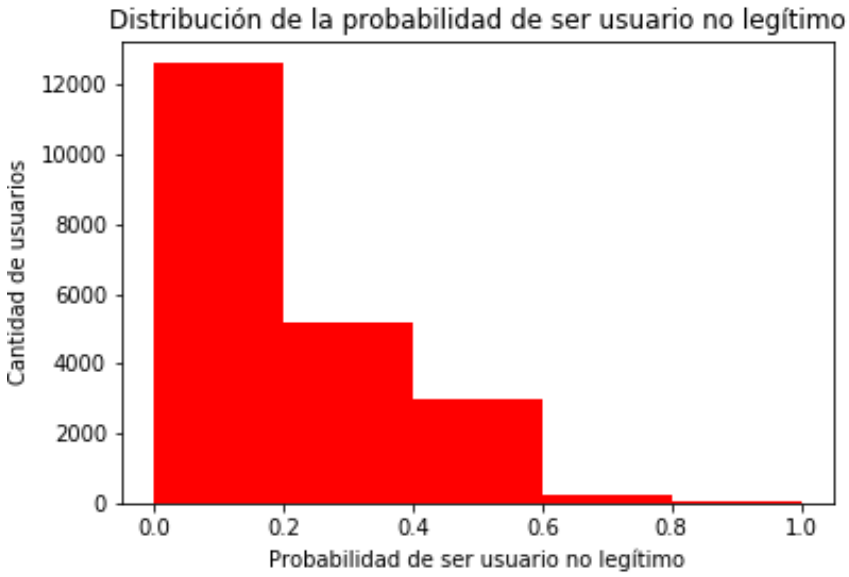

Ilustración 4.3: Distribución de la probabilidad de ser un usuario ilegítimo (*fake follower*)

- **@numanciapodemos**: cuenta oficial de una agrupación local del partido político Podemos.
- **@Mariaperezpere**: persona física.

De forma análoga se exponen un par de ejemplos de *fake followers*:

- **@Jorge_Mireia_TR**.
- **@pedro85939187**.

En el caso de la cuenta perteneciente a Podemos, se observa en su perfil (ilustración 4.4) que es una cuenta que efectivamente pertenece a Podemos. Su nombre está escrito correctamente y la biografía parece coherente. No se trata de una cuenta reciente. Además, se observa que tiene cierta cantidad de *friends* y *followers*. En dichos *friends* (ilustración 4.5) y *followers* (ilustración 4.6) no se observan (a simple vista) relaciones en las que esta cuenta pudiera ser un *fake follower*. Respecto a sus *tweets* se observa que suele *retwitear* cuestiones relativas al partido y sus intereses.

Sobre la cuenta de **@Mariaperezpere**, se observa en su perfil (ilustración 4.7) que es una persona física que indica su nombre y apellidos, así como su ubicación. Indica una biografía. Se observa que tiene una gran cantidad de *friends* y de *followers*. Dichos *friends* (ilustración 4.8) y *followers* (ilustración 4.9) no se observan (a simple vista) relaciones en las que esta cuenta pudiera ser un *fake follower*.

Pasando a las cuentas detectadas por el algoritmo como *fake followers*, si se intenta observar el caso de **@Jorge_Mireia_TR** se observa que dicha cuenta ya no existe (ilustración 4.10). Si se observa en los datos obtenidos, tiene 17 amigos y 3 seguidores. Además, solo tenía 8 *tweets*.

Por último, en el caso de **@pedro85939187**, se observa en su perfil (ilustración 4.11) que es una persona que no indica su nombre real ni tiene una fotografía de su persona. Además su nombre público tiene un montón de números. No indica ubicación ni biografía. Solamente tiene un seguidor, y sigue a dos cuentas. En la ilustración 4.13 se observa la cuenta que le sigue. Si se observa la

Ilustración 4.4: Perfil y biografía de **@numanciapodemos**

Ilustración 4.5: Muestra de *friends* de **@numanciapodemos**

Ilustración 4.6: Muestra de *followers* de **@numanciapodemos**

Ilustración 4.7: Perfil y biografía de **@Mariaperezpere**

Ilustración 4.8: Muestra de *friends* de **@Mariaperezpere**

Ilustración 4.9: Muestra de *followers* de **@Mariaperezpere**

Ilustración 4.10: Muestra de *tweets* repetidos de **@pedro85939187**

ilustración 4.12, se observa que las dos cuentas a las que sigue son las principales del partido político Ciudadanos. Como curiosidad, se observa además que hay *tweets* que se repiten hasta tres veces variando el *hashtag* (ilustración 4.14). Se observa además que los *tweets* son críticos en general con el gobierno. A tenor de estos datos, se puede tratar de un *fake followers* que además es un *bot*.

Ilustración 4.11: Perfil y biografía de **@pedro85939187**

Ilustración 4.12: Muestra de *friends* de **@pedro85939187**

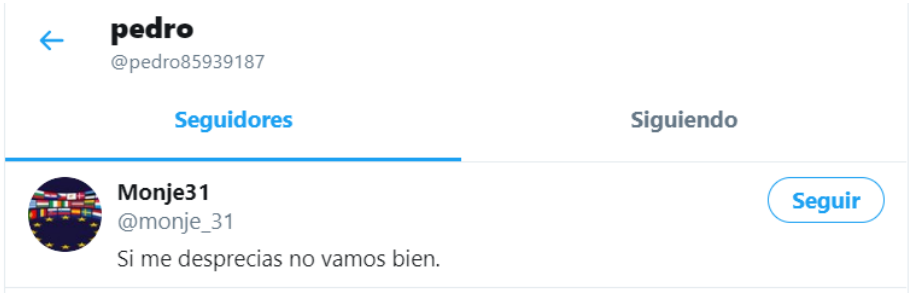

Ilustración 4.13: Muestra de *followers* de **@pedro85939187**

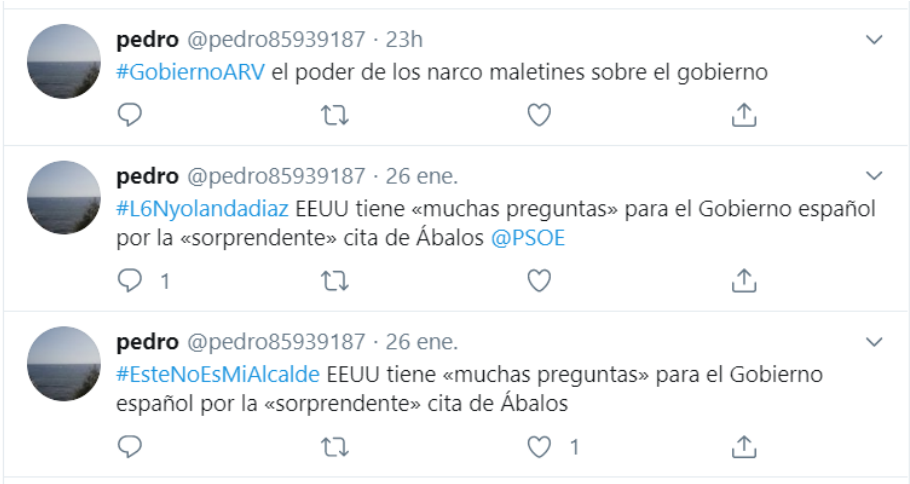

Ilustración 4.14: Muestra de *tweets* repetidos de **@pedro85939187**

Capítulo 5

Discusión

Un vez concluido el modelo, se procede a discutir las diferencias respecto al modelo de [13]. Posteriormente se expondrá la aplicabilidad al problema de detección de *fake followers*.

Los parámetros principales que evalúan el modelo de [13] basado en todas las características son los que se observan a continuación:

Métrica	**Valor**
Precisión (*accuracy*)	0,994
Exactitud (*precision*)	0,997
Exhaustividad (*recall*)	0,994
Área bajo la curva	0,999

Si se hace una comparación con los valores del modelo final propuesto realizado mediante la inclusión de métricas de redes sociales, se observa que existen ligeras mejoras en los siguientes aspectos:

- En el caso de la precisión (*accuracy*) se observa una mejora de una milésima.

- En el caso de la exactitud (*precision*) no se observan mejoras.

- Para la exhaustividad (*recall*) se observa nuevamente una mejora de una milésima.
- Para el área bajo la curva, se observa la misma mejora.

Estas mejoras detalladas, ponen de manifiesto que el modelo mejora muy ligeramente en la detección de más *fake followers* (por la exhaustividad) y en conjunto predice mejor habiendo menos falsos resultados (por la precisión).

No obstante, hay que destacar que las métricas de evaluación descritas en los resultados han sido redondeadas a la milésima para tener la misma precisión decimal que las propuestas en [13]. Si no se efectuara el redondeo, dichas diferencias serían algo más pequeñas.

En el caso de la importancia de las características, hay que destacar que en el modelo propuesto, finalmente ha habido menos características de redes sociales de las que inicialmente se proveyeron al modelo. Dichas características son mencionadas a continuación:

- `friends_count`, métrica directa que es equivalente al grado de salida de un nodo en una red.
- `followers_count`, métrica directa que es equivalente al grado de entrada de un nodo en una red.
- `friends_followers_ratio`
- `mean_followers_of_friends`, métrica que se asimila al grado medio del vecindario (*average neighbor degree*), y que es heredada del modelo inicial previsto por [13].
- **`eigenvector_centrality`**, centralidad de vector propio.
- **`pagerank`**, métrica de prestigio del algoritmo *PageRank*.
- **`hits_authorities`**, métrica de prestigio del algoritmo *HITS*.

De las mencionadas anteriormente, solo las 3 últimas se calcularon mediante el grafo generado (y son aportaciones propias), mientras que las dos primeras se

pudieron obtener directamente del *dataframe* de usuarios. Además, se introdujo al modelo (aunque no estaba previsto en [13]) la cantidad de *followers* para ver si correlacionaba o no con el grado de entrada. Como finalmente correlacionó con un coeficiente de correlación de 1, se suprimió el grado de entrada en base a criterios de coste. En relación a este criterio, algunas características de redes sociales no se introdujeron en el modelo final debido a que existían características que aportaban el mismo valor conceptual que las calculadas en el grafo y además tenían alta correlación, por lo que en estos casos se optó por suprimir aquellas que tenían un mayor coste de obtención (que en este caso en particular era la obtenida mediante el grafo, respecto a la obtenida directamente de *Twitter* que tenía coste cero).

Sin embargo, otras características se suprimieron debido a que en la primera iteración se observó que prácticamente no aportaban nada al modelo (menos de un 1 % en la primera iteración). Así pues, es curioso observar cómo la tasa de enlaces bidireccionales era la más importante en el modelo completo explicado en [13] mientras que en el propuesto ni siquiera llega a aparecer. Esta divergencia en la importancia de las características que se observa entre el modelo propuesto y el de partida puede ser debido a la influencia de las nuevas características, y también puede ser debido a la divergencia interpretativa que puede haber surgido entre las métricas originales interpretadas en [13] y las finalmente ejecutadas en el modelo propuesto.

También se observa que hay muchas características procedentes del modelo anterior que no se encuentran en el actual, a pesar de que tenían una ganancia de información significativa. Este es el caso de métricas como la tasa de enlaces bidireccionales, que era líder en ganancia de información (lo que hubiera hecho que el modelo le asociara una importancia significativa) y sin embargo, prácticamente no aporta nada al modelo actual. Se observa un fenómeno similar en el caso de las métricas de similaridad de los *tweets*, que sin ser las más importantes, tenían una relevancia media pero no están presentes en el modelo actual. Otro ejemplo de este fenómeno se puede observar en el caso del ratio de URLs, que ha pasado de tener una importancia media a no estar presente en el modelo actual.

Sin embargo, también se puede observar que hay características que tenían un gran valor predictivo por su ganancia de información en [13] y que lo siguen manteniendo en el modelo actual. Este es el caso del ratio *friends/followers* que tenía un alto valor para la predicción, y lo sigue teniendo. También sucede este fenómeno con el número de *tweets*. Profundizando en esto, se observa que la característica de *followers count*, que fue introducida como sustituto del grado de entrada con el fin de ser eficiente en el coste de generación de las características, tiene un valor significativo en la predicción, teniendo una importancia del 20 %. También hay una característica que ha destacado significativamente con un 20 %: el valor de autoridad en el algoritmo HITS. Esto pone de manifiesto la utilidad de este indicador en la medición de la autoridad (que está estrechamente ligado con el número y la calidad de los de enlaces entrantes). Sin embargo, su medida opuesta (medición de hub) no ha llegado a estar presente en el modelo. Esto podría indicar que para el modelo predictivo es más importante para medir la legitimidad de un usuario, que sus *followers* sean de calidad, antes que sus *friends* sean de calidad.

Un aspecto que tiene una relevancia apreciable es la edad. Aunque no es el factor que más determina la predicción, parece que el modelo avala que una cuenta con mayor edad, tiene mayor probabilidad de ser un usuario legítimo. Similares conclusiones se pueden obtener con la tasa de seguimiento.

Respecto al *PageRank*, aporta menos de lo esperado (entorno al 2 %) siendo realmente la medición del valor de autoridad de HITS la que está midiendo el prestigio.

A pesar de que el modelo actual mejora ligeramente los resultados respecto al modelo sobre el que se partía, no sería aconsejable su implementación en un sistema de producción ya que la magnitud de la mejora obtenida no justifica el enorme coste que supone implementarlo, especialmente al obtener los usuarios, almacenar la red en memoria RAM (que en el caso real se han superado los 50 GB para un caso real no tan grande) y al calcular las métricas, en las que se incurre en un mayor coste (aunque este sea lineal). Respecto a esto, [13] sugiere la generación de otro modelo derivado, que aunque obtiene resultados

algo inferiores, soslaya esta estructura de costes anteriormente mencionada.

Hay que destacar respecto al entrenamiento del modelo, que sus resultados puede estar sesgados por la calidad de las características (tanto las del *dataset* de entrenamiento como las del caso real) empleadas para predecir la categoría de los usuarios. Particularmente, en el caso de las métricas de redes sociales, estas mejorarán en la medida en que se logre profundizar en la red para obtenerlas (lo que a su vez incrementa el coste).

Sobre los resultados obtenidos en el caso real, se observa que un 1,85 % de los usuarios se han clasificado como *fake followers*. Se podría plantear si esta tasa es alta o baja, pero no se ha localizado un estudio concluyente sobre esto. Esta tasa probablemente dependerá del tipo de entorno en el que se extraiga la red, y de los personajes públicos que haya a su alrededor. Además dependiendo de las fuentes que intenten medir esto, puede haber resultados dispares. En [65], se afirma que los *fake followers* que tiene *Donald Trump* oscilan entre un 11 % y un 70 % según la herramienta empleada para obtener dicha medición. Así pues, es bastante complejo poner en contexto la magnitud de *fake followers* detectada por el modelo.

Conforme a los ejemplos de usuarios legítimos y de *fake followers* aportados en la sección de resultados, son simplemente el resultado del modelo. Aunque dichos ejemplos son una muestra de la predicción realizada por el modelo que representa a los usuarios que más categóricamente ha categorizado el modelo, no se entra a valorar en la certeza de que la predicción sea correcta y que la detección sea o no correcta, ya que es muy complicado valorar a un usuario particularmente. Como ilustración de esto, destacar que para valorar uno de los conjuntos de datos que han formado parte del *dataset* que se ha empleado para el entrenamiento de este modelo (#elezioni2013) y poder categorizar correctamente los usuarios legítimos, fueron necesarios dos sociólogos con poder de veto entre ellos, lo que muestra la dificultad que conlleva este tipo de verificaciones. Así pues, que el modelo califique en una categoría con el 100 % de probabilidad a un usuario, no necesariamente quiere decir que dicha decisión sea correcta (aunque sí que será bastante probable). Con respecto a la distribución

de la probabilidad de ser legítimo, se observa que los usuarios legítimos se detectan con una alta probabilidad en general, aunque hay una parte de usuarios que tiene un 50 % de probabilidades de ser de ambas clases. En estos casos, el clasificador los cataloga finalmente como usuarios legítimos. Respecto a los *fake followers*, se observa que todos los detectados como tal tienen en común que tienen una probabilidad de serlo de más del 60 % y una probabilidad de ser usuarios legítimos inferior al 40 %. Como se puede deducir (especialmente de aquellas clasificaciones donde la probabilidad está equilibrada entre ambas clases) es la dificultad que tiene la labor de detección de *fake followers*.

Capítulo 6

Conclusiones

En este trabajo se ha realizado un modelo predictivo para la detección de *fake followers* basado en [13], introduciendo características del ámbito de las redes sociales, incidiendo en el coste, En primer lugar, se ha buscado un *dataset* que tuviera todos los criterios necesarios para poder ser utilizado para calcular las características propuestas por [13] y basadas en [1, 2] así como las características basadas en redes sociales. Para el cálculo de las características de redes sociales, se ha tenido especial cautela a la hora de elegir las que tuvieran un coste razonable. El coste de las mismas se ha determinado averiguando la complejidad algorítmica de las métricas y escogiendo aquellas que tuvieran un coste algorítmico que a lo sumo fuera lineal. Tras ello, se ha creado y refinado un modelo de clasificación con estas métricas. A lo largo del proceso, se han eliminado características que no tenían relevancia para el modelo. En el modelo final se observa que no todas las métricas generadas en el ámbito de las redes sociales han pasado a formar parte del modelo, siendo solamente tres las características que han sido calculadas mediante la red generada que hayan entrado al modelo. Se hace un análisis comparativo entre el modelo realizado y el de partida, concluyendo que la mejora es mínima. Esta mejora mínima, se entiende que no justifica el enorme coste de extracción, almacenamiento, modelado y procesado de la red.

Las limitaciones que se han encontrado a la hora de realizar esta propuesta, son la dificultad de encontrar *datasets* adecuados para modelar la red necesaria para realizar este tipo de modelos. Esta limitación probablemente se derive de otra localizada durante la extracción del caso real. Esta dificultad es la del coste (temporal) a la hora de extraer la información de *Twitter*, especialmente elevado (15 peticiones por cada 15 minutos) cuando se extraen los *friends* y los *followers*. Otra limitación es la baja eficiencia especial de representar el grafo en memoria RAM. Mientras que el grafo almacenado mediante *pickle* ocupa 1,97 GB, se observa que al leerlo, el proceso pasa a ocupar en memoria RAM aproximadamente unas 40 GB, que se incrementa al realizar ciertas operaciones con dicho grafo.

6.1. Propuestas de trabajo futuro

El área de detección de fraude, de *bots* y de *fake followers* es una área que está en alza. Se puede plantear continuar el avance de este área a través de otras formas de afrontar el problema. Se puede plantear el uso de series temporales para detectar el flujo de información (ya sea de *tweets* o de *usuarios*. Se puede plantear profundizar también mediante el análisis de los mensajes intercambiados, siendo el procesamiento de lenguaje natural clave en esta estrategia. Por último, otras posibilidades podrían hacer uso de técnicas como el *deep learning*.

Se ha observado que hay una escasez de *datasets* que contengan los conjuntos de datos de *friends* y *followers*, por lo que planificar y desarrollar *datasets* con estas características puede ser un aporte significativo que permite expandir las posibilidades de creación de modelos predictivos en base a características de redes sociales.

Respecto a las características de redes sociales, ha habido una métrica que tiene un coste lineal de extracción y que no ha sido implementada: las triadas. Como se indicó en la metodología, *NetworkX* no dispone por el momento de un algoritmo que extraiga para cada nodo los tipos de triadas en las que participa, siendo la implementación más cercana la del censo triádico [58]. Habría que

plantear la optimización del cálculo de esta característica, utilizando estructuras de datos más eficientes. Un posible planteamiento para afrontar este problema puede venir de la mano del *backtracking*, reduciendo la creación y destrucción de conjuntos en su algoritmo. Esta característica podría ser muy prometedora por las implicaciones que tiene en la sociedad respecto a la transitividad y la tendencia al balance de los nodos que la conforman [57].

Por último, si se quiere partir del modelo actual, se puede plantear la prueba de otros modelos de clasificación alternativos al *Random Forest*, con el fin de determinar si otro modelo puede mejorar las métricas obtenidas mediante los criterios de evaluación expuestos. Otra posibilidad, estriba en el análisis de la complejidad algorítmica espacial además de la temporal abordada en el trabajo.

Índice de códigos fuente

1. Implementación de grado de entrada por *NetworkX* 42
2. Implementación de centralidad de grado por *NetworkX* 44

Índice de ilustraciones

3.1. Tipos de triadas (imagen producida por [61]) 50
3.2. Verificación de la correlación (I) 55
3.3. Verificación de la correlación (II) 57
3.4. Distribución de los datos según la variable `is_bot` 59

4.1. Distribución de usuarios según su legitimidad 72
4.2. Distribución de la probabilidad de ser un usuario legítimo 72
4.3. Distribución de la probabilidad de ser un usuario ilegítimo (*fake follower*) 74
4.4. Perfil y biografía de **@numanciapodemos** 76
4.5. Muestra de *friends* de **@numanciapodemos** 77
4.6. Muestra de *followers* de **@numanciapodemos** 78
4.7. Perfil y biografía de **@Mariaperezpere** 79
4.8. Muestra de *friends* de **@Mariaperezpere** 80
4.9. Muestra de *followers* de **@Mariaperezpere** 81
4.10. Muestra de *tweets* repetidos de **@pedro85939187** 82
4.11. Perfil y biografía de **@pedro85939187** 83
4.12. Muestra de *friends* de **@pedro85939187** 83
4.13. Muestra de *followers* de **@pedro85939187** 84
4.14. Muestra de *tweets* repetidos de **@pedro85939187** 84

Bibliografía

[1] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," 12 2010, pp. 1–9.

[2] C. Yang, R. Harkreader, and G. Gu, "Empirical evaluation and new design for fighting evolving twitter spammers," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1280–1293, Aug 2013.

[3] F. Peinado, "Una red de cuentas falsas de twitter promueve a vox en campaña," Abril 2019. [Online]. Available: https://elpais.com/politica/2019/04/25/actualidad/1556203502_359349.html

[4] El Periódico, "'bots' de twitter generaron «contenido violento» para influir en el 1-o, según un estudio," Noviembre 2018. [Online]. Available: https://www.elperiodico.com/es/politica/20181120/bots-twitter-contenido-violento-referendum-1o-7158708

[5] M. Stella, E. Ferrara, and M. De Domenico, "Bots increase exposure to negative and inflammatory content in online social systems," Noviembre 2018. [Online]. Available: https://www.pnas.org/content/115/49/12435

[6] R. Terrassa, "El 'spam' político o cómo manipulan los partidos las redes sociales," Marzo 2016. [Online]. Available: https://www.elmundo.es/espana/2016/03/10/56d42f14268e3e10648b4660.html

[7] B. Logan, "Twitter found more than 50,000 russia-linked accounts that actively shared election-related material — and trump inter-

acted with them hundreds of times," Enero 2018. [Online]. Available: https://www.businessinsider.com/twitter-found-more-russian-bots-trump-interacted-with-many-2018-1?IR=T

[8] A. Badawy, E. Ferrara, and L. Kristina, "Analyzing the digital traces of political manipulation: The 2016 russian interference twitter campaign," Enero 2018. [Online]. Available: https://www.businessinsider.com/twitter-found-more-russian-bots-trump-interacted-with-many-2018-1?IR=T

[9] "Iv estudio sobre usuarios de facebook, twitter e instagram en españa," The social media family. [Online]. Available: https://www.emprendedores.es/gestion/a26421488/facebook-instagram-twitter-redes-sociales-mas-utilizadas-espana-2019/

[10] J. H. Kietzmann, K. Hermkens, I. P. McCarthy, and B. S. Silvestre, "Social media? get serious! understanding the functional building blocks of social media," *Business Horizons*, vol. 54, no. 3, pp. 241 – 251, 2011, sPECIAL ISSUE: SOCIAL MEDIA. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007681311000061

[11] R. Wang and Y. Jin, "An empirical study on the relationship between the followers' number and influence of microblogging," in *2010 International Conference on E-Business and E-Government*, May 2010, pp. 2014–2017.

[12] I. Anger and C. Kittl, "Measuring influence on twitter," in *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, ser. i-KNOW '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2024288.2024326

[13] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: efficient detection of fake twitter followers," *Decision Support Systems*, vol. 80, pp. 56–71, December 2015.

[14] O. Varol, "Bot repository." [Online]. Available: https://botometer.iuni.iu.edu/bot-repository/index.html

[15] E. G. Ellis, "Fighting instagram' $1.3 billion problem—fake followers," WIRED. [Online]. Available: https://www.wired.com/story/instagram-fake-followers/

[16] K. Lee, B. D. Eoff, and J. Caverlee, "Seven months with the devils: a long-term study of content polluters on twitter," in *In AAAI Int'l Conference on Weblogs and Social Media (ICWSM)*, 2011.

[17] "Get friends/ids - twitter developers." [Online]. Available: https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-friends-ids

[18] "Get followers/ids - twitter developers." [Online]. Available: https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-followers-ids

[19] A. Mehrotra, M. Sarreddy, and S. Singh, "Detection of fake twitter followers using graph centrality measures," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, Dec 2016, pp. 499–504.

[20] S. A. Cook, "An overview of computational complexity," *Commun. ACM*, vol. 26, no. 6, p. 400–408, Jun. 1983. [Online]. Available: https://doi.org/10.1145/358141.358144

[21] N. Meghanathan, *Graph Theoretic Approaches for Analyzing Large-Scale Social Networks*, ser. Advances in Wireless Technologies and Telecommunication (2327-3305). IGI Global, 2017. [Online]. Available: https://books.google.es/books?id=YYwtDwAAQBAJ

[22] P. Berba, "Time complexity for data scientists," Towards DataScience. [Online]. Available: https://towardsdatascience.com/time-complexity-for-data-scientists-664d00e57724

[23] U of MN Internet Gopher Team, "http://www.iana.org/assignments/media-types/text/tab-separated-values." [Online]. Available: http://www.iana.org/assignments/media-types/text/tab-separated-values

[24] "My information bubble datasets." [Online]. Available: http://mib.projects.iit.cnr.it/dataset.html

[25] M. Munafo, "Twitter, quanti falsi profili: il cnr ora va a caccia dei fake - repubblica.it." [Online]. Available: http://goo.gl/zNC2k

[26] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proceedings of the 26th International Conference on World Wide Web Companion*, ser. WWW '17 Companion. International World Wide Web Conferences Steering Committee, 2017, pp. 963–972. [Online]. Available: https://doi.org/10.1145/3041021.3055135

[27] S. Cresci, R. D. Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Social fingerprinting: Detection of spambot groups through dna-inspired behavioral modeling," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, pp. 561–576, 2018.

[28] "ngrams - nltk - python documentation - kite." [Online]. Available: https://kite.com/python/docs/nltk.ngrams

[29] "Essential basic functionality - pandas 0.25.3 documentation." [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html

[30] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots + machine learning," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '10. New York, NY, USA:

Association for Computing Machinery, 2010, p. 435–442. [Online]. Available: https://doi.org/10.1145/1835449.1835522

[31] D. Garlaschelli and M. I. Loffredo, "Patterns of link reciprocity in directed networks," *Physical Review Letters*, vol. 93, no. 26, Dec 2004. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.93.268701

[32] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: human, bot, or cyborg?" in *ACSAC '10*, 2010.

[33] N. Developers, "networkx.digraph.in_degree - networkx documentation." [Online]. Available: https://networkx.github.io/documentation/stable/reference/classes/generated/networkx.DiGraph.in_degree.htm

[34] ——, "networkx.digraph.out_degree - networkx documentation." [Online]. Available: https://networkx.github.io/documentation/stable/reference/classes/generated/networkx.DiGraph.out_degree.ht

[35] ——, "networkx.digraph.degree - networkx documentation." [Online]. Available: https://networkx.github.io/documentation/stable/reference/classes/generated/networkx.DiGraph.degree.html

[36] "Digraph class - networkx - github." [Online]. Available: https://github.com/networkx/networkx/blob/master/networkx/classes/digraph.py

[37] B. Rafael Gallardo, "Week eight: Complexity of python operations," Donald Bren School of Information and Computer Sciences, University of California, Irvine. [Online]. Available: https://www.ics.uci.edu/~brgallar/week8_2.html

[38] "Reportviews - networkx - github." [Online]. Available: https://github.com/networkx/networkx/blob/master/networkx/classes/reportviews.py

[39] "The python yield keyword explained - python tips." [Online]. Available: https://pythontips.com/2013/09/29/the-python-yield-keyword-explained/

[40] "networkx.algorithms.assortativity.average_neighbor_degree." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/netwo

[41] "networkx.algorithms.assortativity.average_neighbor_degree." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/netwo

[42] S. Bose, "Regular graph in graph theory - geeks for geeks." [Online]. Available: https://www.geeksforgeeks.org/regular-graph-in-graph-theory/

[43] N. Developers, "networkx.algorithms.centrality.degree_centrality - networkx documentation." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algori

[44] "Degree algorythms - networkx - github." [Online]. Available: https://github.com/networkx/networkx/blob/master/networkx/algorithms/centrality/degree_alg.p

[45] "networkx.algorithms.centrality.betweenness_centrality." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.al

[46] "Betweenness centrality measures - networkx - github." [Online]. Available: https://github.com/networkx/networkx/blob/master/networkx/algorithms/centrality/betweenness

[47] U. Brandes, "A faster algorithm for betweenness centrality," *The Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001. [Online]. Available: https://doi.org/10.1080/0022250X.2001.9990249

[48] M. E. J. Newman, "The mathematics of networks," Center for the Study of Complex Systems, University of Michigan, Ann Arbor. [Online]. Available: http://ccb-class.pbworks.com/f/newman_network_math.pdf

[49] "networkx.algorithms.centrality.eigenvector_centrality." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.al

[50] M. Franceschet, "Closeness centrality," Department of Mathematics and Computer Science. University of Udine. [Online]. Available: https://www.sci.unich.it/ francesc/teaching/network/closeness.html

[51] "networkx.algorithms.centrality.closeness_centrality." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrali

[52] M. Franceschet, "Local clustering," Department of Mathematics and Computer Science. University of Udine. [Online]. Available: https://www.sci.unich.it/ francesc/teaching/network/clustering.html

[53] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[54] "networkx.algorithms.link_analysis.pagerank_alg.pagerank." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorith

[55] "networkx.algorithms.link_analysis.hits_alg.hits." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.link_an

[56] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, pp. 604–632, September 1999. [Online]. Available: http://doi.acm.org/10.1145/324133.324140

[57] R. A. H. M. Riddle, *Introduction to social network methods.* University of California, Riverside, 2005, ch. 8. Embedding. [Online]. Available: https://faculty.ucr.edu/ hanneman/nettext/C8_Embedding.html

[58] "networkx.algorithms.triads.triadic_census." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.triads.t

[59] V. Batagelj and A. Mrvar, "A subquadratic triad census algorithm for large sparse networks with small maximum degree," *Social Networks*, vol. 23, no. 3, pp. 237 – 243, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378873301000351

[60] "networkx - get the list of triad nodes, who fall under the category of individual triadic census - stack overflow." [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.triads.t

[61] M. Cugmas, A. Ferligoj, and A. Žiberna, "Generating global network structures by triad types," *PLOS ONE*, vol. 13, 10 2017.

[62] K. R. Abrahamson, "12.6. the clique problem," Department of Computer Science, East Carolina University. [Online]. Available: http://www.cs.ecu.edu/karl/6420/spr16/Notes/NPcomplete/clique.html

[63] L. Breiman and A. Cutler, "Random forests." [Online]. Available: https://www.stat.berkeley.edu/ breiman/RandomForests/cc_home.htm

[64] "3.2.4.3.1. sklearn.ensemble.randomforestclassifier," scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[65] K. McElwee, "Fake-follower calculators misinform users, journalists with dubious statistics." towards data science, Septiembre 2019. [Online]. Available: https://towardsdatascience.com/fake-follower-calculators-misinform-users-journalists-with-dubious-statistics-659b60fc4d5a