DETECCIÓN DE TROLLS EN TWITTER PARA EL CASO DE LA GUERRA DE UCRANIA MEDIANTE APRENDIZAJE NO SUPERVISADO

CONVOCATORIA: Ordinaria (abril de 2023)

ALUMNO / A: Héctor Asorey de Pablos

TUTOR / A: Dr. Pedro Concejero Cerezo

GRADO: Ingeniería de Software

ÍNDICE

AGRADECIMIENTOS

Gracias a mis padres, Manuel y Nuria, y a mi abuela, Celia, por apoyarme durante mis estudios de grado.

Agradecer también a mi tutor, Pedro Concejero Cerezo, por orientarme y guiarme para la realización de este proyecto.

Por ultimo, mi agradecimiento a mi profesor de Proyectos de Ingeniería de Datos, Pablo Ramos Criado, por la ayuda e ideas ofrecidas en el aspecto del desarrollo software y a mis compañeros de proyecto Sergio Esteban Tarrero, Miguel Herencia García y Manuel Jesús Cerezo Mazon.

RESUMEN

El auge de las redes sociales ha permitido conectar a personas de todas partes del mundo y permite a sus usuarios conocer otras realidades distintas. A pesar de esto, las redes sociales también se pueden utilizar con finalidades perjudiciales como la difamación, acoso o propagación de desinformación.

En los últimos años, han ocurrido numerosos escándalos en la red social Twitter, desde la interferencia de Rusia en asuntos vitales de otros países, como las elecciones estadounidenses de 2016 ó 2020, o en el asunto sobre la independencia de Cataluña tras el referéndum ilegal en 2017, hasta la propagación de mentiras sobre el Covid-19 y las vacunas. Debido a ello, la lucha contra la desinformación en redes sociales ha adquirido una gran importancia.

Actualmente, con la invasión rusa de Ucrania, se vive un momento político delicado a nivel mundial, siendo éste un ambiente muy propicio para la propagación de desinformación. El objetivo de este trabajo es desarrollar una herramienta que pueda detectar publicaciones y cuentas que propaguen desinformación sobre este tema concreto, mediante el uso de técnicas de procesamiento de datos y aprendizaje no supervisado. Así, se realizará una obtención de tweets que traten el tema de Ucrania, se procesarán para obtener información de esos tweets y se emplearán modelos de aprendizaje no supervisado para la detección de publicaciones o usuarios anómalos.

El fin último de este trabajo es, por tanto, el desarrollo de una aplicación que permita identificar tweets o usuarios con fines maliciosos para el caso de la guerra en Ucrania.

PALABRAS CLAVE

Twitter, troll, aprendizaje no supervisado, anomalía, ratio, métricas, embeddings.

ABSTRACT

The rise of social media has allowed for connecting people from all over the world and has made possible for users to learn about different realities. However, social media can also be used for harmful purposes such as difamation, harassment, or spreading misinformation.

In recent years, numerous scandals have occurred on the social media platform Twitter, from Russian interference in vital affairs of other countries, such as the 2016 or 2020 US elections or the issue of Catalonia's independence following the illegal referendum in 2017, to the wide spread of lies about Covid-19 and vaccines. Due to this, fighting against misinformation on social media has acquired great importance.

Currently, with the Russian invasion of Ukraine, there is a delicate political moment worldwide, which is a very conducive environment for the spread of misinformation. The aim of this work is to develop a tool that can detect posts and accounts that spread misinformation about this specific topic, using data processing techniques and unsupervised learning. Thus, a collection of tweets related to the Ukraine topic will be obtained, processed to obtain information from those tweets, and unsupervised learning models will be employed to detect anomalous posts or users.

The goal of this work, therefore, is the development of an application that allows the identification of tweets or users with malicious purposes regarding the Ukraine war.

KEYWORDS

Twitter, troll, unsupervised learning, anomaly, ratio, metrics, embeddings.

1. INTRODUCCIÓN

1.1 Motivación y contexto

Las redes sociales son servicios que permiten a personas comunicarse y compartir información por todo el mundo gracias a Internet.

En 1997 surge la que se considera la primera red social, en la cual los usuarios se creaban una cuenta, podrían tener una lista de amigos y comunicarse con ellos a través de un sistema de mensajería. A inicios de los 2000 surgieron más redes sociales: Friendster, MySpace y LinkedIn. La primera de ellas estaba orientada a usuarios que querían compartir intereses sobre el mundo de los videojuegos, y las otras dos tenían un enfoque más profesional, orientada al mundo laboral. En 2004 aparece una de las redes sociales más prolíficas y de mayor éxito, Facebook. Facebook, en su origen, nació como una red social para conectar a los estudiantes de Harvard. Twitter surge en 2006 en San Francisco, creada por Jack Dorsey, Noah Glass, Biz Stone y Evan Williams.

Twitter supuso una revolución informativa, pues cualquier ciudadano podía publicar noticias que ocurrieran en su país, logrando una "democratización" de la información. Sin embargo, esto ha resultado en números casos de desinformación y propagación de bulos o bien sin ningún interés más que provocar una desinformación generalizada, o bien con fines políticos. En 2016, Rusia, bajo el gobierno de Vladimir Putin, interfirió en las elecciones estadounidenses de 2016, en la cual miles de cuentas en redes sociales creadas por la IRA (Internet Research Agency, una compañía rusa) (Internet Research Agency - Wikipedia, Wikipedia, 2023) se hacían pasar por estadounidenses publicando contenidos en redes sociales con el objetivo de desinformar a la población, para perjudicar a la candidata demócrata Hillary Clinton y popularizar a Donald Trump. Este caso supuso un antes y un después, pues se vio el peligro que pueden llegar a suponer las redes sociales, y cómo pueden ser capaces de influenciar la opinión pública (The New York Times, Barnes, J.E, 2021, Russian Interference in 2020 Included Influencing Trump Associates, Report Says).

Otros casos muy sonados fueron la inferencia rusa en redes sociales, principalmente Twitter, para promover la salida de Reino Unido de la Unión Europea y en caso del

referéndum en Cataluña del 6 de octubre de 2017 con el objetivo de desestabilizar España, propagación de desinformación acerca del Covid-19 y las vacunas en época de pandemia o propaganda pro-rusa sobre la guerra de Ucrania.

Por ello, en una época donde abunda la información, y donde es más fácil que nunca convertir una mentira en verdad debido a la alta propagación que puede tener hoy en día cualquier noticia debido a las redes sociales y la interconexión global que estas permiten, es de vital importancia identificar los hechos reales de los que no, y proteger la verdad.

Asimismo, debido al conflicto armado actual entre Rusia y Ucrania, permite tener una primera toma de contacto con la nueva modalidad de guerra posible debido al rápido avance de las tecnologías de la información, la guerra híbrida. Se podrá visualizar como ambos bandos luchan no solo por motivos políticos y económicos, como en las guerras tradicionales, sino también, como ambos luchan para convertir en verdad su versión de los hechos mediante la manipulación en redes sociales publicando contenidos que tienen un interés concreto detrás, así como la posible evolución de la propaganda, intentando llegar al máximo número de personas posible, no solo dentro de las fronteras de cada país, sino también fuera de ellas, pudiendo llegar a cualquier parte del mundo.

Además, los trolls en Twitter no solo obedecen a conflictos políticos, sino que también su objetivo puede ser el de desprestigiar a una persona o marca en concreto, dificultando así la capacidad de éstos a obtener clientes. Por ejemplo, cuando se implementó por primera vez la funcionalidad de Twitter Blue, que te permite pagar mensualmente para obtener la marca de verificado, hubo usuarios que se hicieron pasar por empresas publicando contenido inadecuado (por ejemplo, el usuario @nIntendoofus se hizo pasar por Nintendo of America, una de las mayores empresas del sector del videojuego, publicó una imagen del personaje estrella de Nintendo, Super Mario haciendo un gesto inadecuado). También hubo numerosos casos de personas haciéndose pasar por Elon Musk, burlándose de él.

El objetivo de este trabajo será desarrollar una aplicación que pueda detectar cuentas dedicadas a la propagación de información falsa o con fines dañinos en la red social Twitter, con el fin último de proteger a los usuarios de este tipo de cuentas y de la desinformación que éstas propagan.

1.2 Planteamiento del problema

Este trabajo está orientado al desarrollo de una aplicación que permita identificar cuentas de la red social Twitter dedicadas a propagar desinformación, ya sean bots o personas reales quienes gestionan estas cuentas.

Como se ha expuesto anteriormente, la proliferación de estas cuentas en redes sociales supone un grave problema, pues pueden alterar la opinión pública acerca de cualquier tema motivada por objetivos personales o políticos, o difamar a una persona o empresa. La facilidad actual para generar y obtener información hace más importante que nunca la creación de herramientas que permitan al público general discernir entre datos verídicos y falsos, para evitar la propagación y aceptación de hechos falsos.

En los últimos años, se han visto numerosos ejemplos de intentos de alterar la opinión pública sobre determinados temas, siendo uno de los casos más sonados, la interferencia de Rusia en el panorama electoral estadounidense de 2016, mediante la propagación de informaciones falsas en redes sociales, como Twitter, concluyendo en la elección de Donald Trump como presidente, más afín con los intereses de Rusia que la candidata demócrata Hillary Clinton. Casos como este hacen ponen en manifiesto la necesidad de desarrollar herramientas que permitan proteger la verdad por encima de los intereses de personas, organizaciones o incluso, naciones.

Es importante desarrollar aplicaciones que puedan detectar de manera precisa este tipo de cuentas, pues a pesar de que el elevado número de cuentas troll daña la imagen de marca de Twitter, este tipo de cuentas sigue proliferando en esta red social, y pese a que Twitter eventualmente suspenda alguna de estas cuentas, para cuando lo ha hecho, esa cuenta ha tenido ya un impacto considerable.

1.3 Objetivos del trabajo

Con este trabajo, se pretende detectar cuentas dedicadas a desinformar al público general mediante la propagación de noticias falsas en la red social Twitter. A este tipo de cuentas se les denomina trolls. Para poder su correcta detección, se parte de las siguientes hipótesis:

- Mediante el uso de la API de Twitter, se podrán obtener los datos necesarios. De esta forma, no solo obtendremos el contenido de los tweets, sino también información relativa al tweet y al usuario que publica el tweet.
- El conjunto de datos obtenido permitirá la generalización del modelo, pues habrá variedad en los datos y serán representativos de la realidad.
- La limpieza de los datos tendrá como salida siempre la obtención de datos de calidad que permitan la obtención de nuevas métricas y el correcto entrenamiento de los modelos.
- Se obtendrán métricas de los datos que permitan una mayor precisión a la hora de distinguir entre cuentas de usuarios reales y cuentas troll.
- El uso de técnicas NLP como stopwords o Word embeddings permitirán que los modelos sean capaces de un mejor aprendizaje acerca de los patrones en los contenidos publicados que permitan distinguir entre usuarios reales y trolls. El uso de distancias entre textos combinado con los embeddings se utilizará para lograr una medida de lo parecido que son los tweets de un usuario, asumiendo qué si un usuario publica contenidos similares siempre, será un troll.
- Las anomalías detectadas serán casos de trolls, y no otro tipo de anomalías.

El empleo de técnicas de aprendizaje no supervisado centradas en la detección de anomalías combinado con un trabajo de preprocesado y preparación de los datos lograrán que algoritmos como Isolation Forest, Local Outlier Factor, One-class SVM, entre otros, alcancen buenas precisiones a la hora de detectar cuentas fraudulentas.

2. ESTADO DEL ARTE

2.1 Marco teórico del trabajo

Para el desarrollo de este trabajo, se necesitan múltiples herramientas derivadas del campo de la Inteligencia Artificial, como Natural Language Processing o modelos de aprendizaje no supervisado. Asimismo, también se necesitan herramientas para la extracción de conjuntos de datos.

2.1.1 NLP (Natural Language Processing)

El Procesamiento de Lenguaje Natural o en inglés Natural Language Processing (NLP, a partir de ahora se utilizará este acrónimo) es una rama de la Inteligencia Artificial que permite que los ordenadores puedan analizar, entender y generar texto o incluso hablar como los humanos (McCarthy, J., Minsky, M., Rochester, N., Shannon, C., 1956, A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence).

NLP tiene aplicaciones tan diversas como el análisis de sentimiento, que pretende identificar las emociones predominantes en un texto, pudiendo, por ejemplo, categorizar un texto como una opinión positiva o negativa, o los Word embeddings, que intentan representar las palabras en forma de vectores numéricos para su procesamiento por parte del ordenador. Otra aplicación puede ser la generación automática de texto, cuyas aplicaciones pueden ir desde chatbots que asistan a los usuarios, como ChatGPT, hasta la generación de textos con el estilo de algún autor conocido, como Shakespeare. Asimismo, el Procesamiento de Lenguaje Natural se puede utilizar para la traducción automática de textos y diálogos o para resumir artículos.

Uno de los mayores problemas que tiene Natural Language Processing tiene problemas para adaptarse a distintos idiomas. La mayor parte de la investigación acerca de NLP se desarrolla en inglés, lo cual dificulta su implementación con otras lenguas como podría ser el español o el francés, que pese a tener desarrollo, no se equipará con el desarrollo e investigación en inglés. También, a la hora de investigar en esta área, se debe tener en

cuenta también la dificultad del lenguaje, y se debe tener en cuenta que lenguas románicas como el español o el francés tienen reglas más complejas, lo cual dificulta el desarrollo en estos idiomas.

Para este trabajo, son claves dos conceptos de NLP, Word embeddings y sentiment analysis.

Word Embeddings

Word embeddings: técnica que pertenece a la rama de Procesamiento de Lenguaje Natural (NLP), que permite representar de forma numérica mediante vectores palabras, permitiendo así que características como el contexto de la palabra, relación con otras palabras o la similitud semántica y sintáctica con otras palabras puedan ser entendidas por una máquina. En resumen, un embedding es una representación de texto de tal manera que palabras parecidas tienen una representación vectorial parecida. Un embedding es una representación densa de una palabra en forma de vector.

Figura 2.1: Representación en el espacio de las palabras gracias al uso de Word embeddings

https://blogs.iadb.org/conocimiento-abierto/es/que-son-los-word-embeddings/

Por ejemplo, supongamos las siguientes palabras, perro y gato. Estas dos palabras hacen referencia a animales mamíferos cuadrúpedos, por lo que, al ser palabras similares, los embeddings deberían presentar similitud entre ambos, de tal manera que, si se representasen estas palabras en el espacio, deberían situarse cerca.

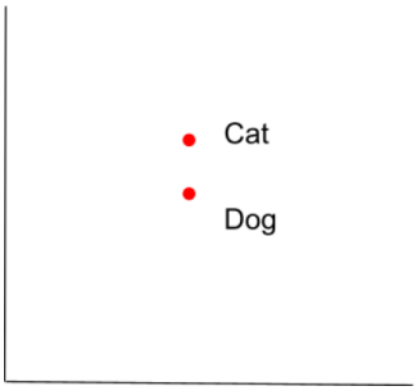

Figura 2.2: Representación de dos palabras con significados parecidos en el espacio gracias a embeddings
https://neptune.ai/blog/word-embeddings-guide

Algunos algoritmos para el desarrollo de Word Embeddings son Word2Vec y GloVe.

Word2Vec: método de creación de embeddings (Mikolov, T., et al., 2013, Efficient Estimation of Word Representations in Vector Space y Distributed Representations of Words and Phrases and their Compositionality), propuesto como mejora del modelo propuesto previamente por Bengio et al (Bengio, Y., et al., 2003, A Neural Probabilistic Language Model), para reducir la complejidad computacional. Mikolov et al propusieron dos modelos principales: CBOW y Continuous.

Modelo CBOW: para predecir el embedding de una palabra, se combinan las representaciones de las palabras que rodean a la que se quiere predecir. Así, se deben definir dos parámetros, la palabra a predecir y el número de palabras que rodean al target a tener en cuenta para la predicción, tanto por delante como por detrás.

Continuous Skip-gram: este modelo en vez de predecir la representación de una palabra en función de las palabras que la rodean, es decir, el contexto, intentará predecir aquellas palabras que pueden aparecer tanto antes o después de una palabra, en un determinado rango.

GloVe (Global Vectors por Word Representation): es un algoritmo de aprendizaje no supervisado desarrollado por la Universidad de Stanford para generar Word embeddings. (https://nlp.stanford.edu/projects/glove/, Pennington, J., Socher, R., Manning, C.D, 2014, GloVe: Global Vectors for Word Representation)

Sentiment Analysis

Sentiment Analysis, o análisis de sentimiento en español, es una técnica de Procesamiento de Lenguaje Natural que permite determinar el tono emocional subyacente en un texto (Pang, B., Lee, L., 2002, A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts). Mediante esta técnica se puede determinar si un texto tiene una connotación positiva, negativa o neutra, e incluso pueden determinar emociones más concretas, como puede ser enfado, alegría…

Hay tres ramas principales para el desarrollo de algoritmos de Sentiment Analysis:

- Basado en reglas: se parte de un diccionario en el que se define que palabras poseen un significado positivo y cuales poseen un significado negativo. A partir de estos diccionarios, el algoritmo itera sobre el texto, analizando el significado de cada palabra y asignando los pesos predefinidos en los diccionarios a cada palabra, para posteriormente calcular que tipo de palabras son las predominantes en ese texto. Habitualmente, se realizan una serie de transformaciones a los textos antes de aplicar el algoritmo de análisis de sentimiento. Estas transformaciones incluyen tokenizar el texto, que consiste en dividir una frase en las palabras que la componen, eliminación de stopwords, es decir, eliminar aquellas palabras que no añaden significado a una frase como pueden ser preposiciones, y lematizar, que consiste en obtener la raíz de las palabras, para así, por ejemplo, asociar una palabra en masculino y femenino a un solo token único en vez de a uno por la forma masculina y a otro por la forma femenina.
- Automático: no se parte de diccionarios predefinidos de palabras, sino que se utiliza una combinación de aprendizaje supervisado y no supervisado para la calificación de textos. Se divide en una fase de entrenamiento y en una de predicción. En la fase de entrenamiento, se entrena un modelo para asociar una entrada (en este caso, textos), a una salida (en este caso, como se haya etiquetado

manualmente el texto, si es positivo, negativo o neutro). En la fase de predicción, mediante transformación de textos a vectores, se asocia la etiqueta predicha por el modelo al texto. Se utilizan técnicas no supervisadas como Word embeddings para la transformación de cadenas de texto a vectores numéricos, de tal manera que palabras con significados parecidos tengan una representación espacial cercana. Para la clasificación de si un texto es negativo, positivo o neutro, se utilizan modelos de aprendizaje automático como Naive Bayes (Anderson, J., Mitchell, T., 1997, The Bayesian Classification Algorithm), Support Vector Machine (Vapnik, V., Chervonenkis, A., 1995, Support-Vector Networks) o Deep Learning.

- Híbrido: combina ambos métodos de realizar Sentiment Analysis, logrando una mayor precisión que los métodos anteriores.

El análisis de sentimiento, como el NLP, es muy dependiente del idioma de los textos sobre los que se quiera aplicar esta técnica, para la cual el inglés el idioma donde más desarrollo hay.

2.1.2 Machine Learning (Aprendizaje Automático)

El aprendizaje automático es rama de la Inteligencia Artificial enfocada al desarrollo de algoritmos y modelos estadísticos que permite a las máquinas aprender sobre los datos sin ser programadas para ello explícitamente.

Dependiendo de cómo se entrenen los modelos de aprendizaje automático, se distinguen tres enfoques principales: aprendizaje supervisado, no supervisado y por refuerzo.

Aprendizaje Supervisado

El aprendizaje supervisado es una técnica de aprendizaje automático que se basa en el uso de datos etiquetados para entrenar un modelo que pueda hacer predicciones precisas sobre nuevos datos. En el aprendizaje supervisado, se proporcionan al modelo un conjunto de

datos de entrada y su correspondiente etiqueta de salida, y el modelo aprende a relacionar la entrada con la salida deseada.

Existen varios algoritmos de aprendizaje supervisado que se utilizan para diferentes tareas, como la clasificación y la regresión. En la clasificación, el objetivo es predecir la etiqueta de clase de una entrada, mientras que, en la regresión, el objetivo es predecir un valor numérico. Los algoritmos de clasificación supervisada incluyen Árboles de decisión, KNN (Cover, T., Hart, P., 1967, Some Methods for Classification), Naive Bayes y SVM, mientras que los algoritmos de regresión supervisada incluyen regresión lineal, regresión polinómica y redes neuronales.

Una de las principales ventajas del aprendizaje supervisado es su capacidad para hacer predicciones precisas sobre nuevos datos, y permite obtener fácilmente una métrica de la precisión de acierto de los modelos. Al proporcionar datos etiquetados al modelo, se le enseña a reconocer patrones y relaciones en los datos, lo que le permite hacer predicciones precisas sobre datos no vistos previamente. Esto hace que el aprendizaje supervisado sea útil en una amplia gama de aplicaciones, como la detección de fraudes, la clasificación de correo no deseado, la identificación de imágenes y la predicción de resultados financieros.

Sin embargo, el aprendizaje supervisado también tiene algunas limitaciones. Requiere grandes conjuntos de datos etiquetados para entrenar el modelo de manera efectiva, y puede ser costoso y laborioso etiquetar grandes cantidades de datos. Además, el modelo solo puede hacer predicciones precisas sobre datos que son similares a los datos que se usaron para entrenarlo. Si el modelo encuentra un patrón o una relación que no se ajusta a los datos de entrenamiento, puede generar predicciones incorrectas.

Por tanto, el aprendizaje supervisado es una técnica que utiliza datos etiquetados para entrenar un modelo que pueda hacer predicciones precisas sobre nuevos datos.

Aprendizaje No Supervisado

El aprendizaje no supervisado es una rama del aprendizaje automático que se enfoca en encontrar patrones o estructuras en los datos sin la necesidad de tener etiquetas o respuestas predefinidas. A diferencia del aprendizaje supervisado, donde se entrena un

modelo para predecir una salida específica a partir de un conjunto de entradas y sus correspondientes etiquetas, en el aprendizaje no supervisado, el objetivo es descubrir patrones y relaciones ocultas en los datos.

Existen varios algoritmos de aprendizaje no supervisado que se utilizan para diferentes tareas, como la clasificación, la segmentación y la reducción de dimensionalidad. La agrupación o clustering es uno de los algoritmos más populares en el aprendizaje no supervisado, donde se busca agrupar los datos en diferentes conjuntos, de tal forma que los elementos dentro de cada conjunto sean similares entre sí y diferentes a los elementos de los demás conjuntos.

Otro algoritmo común en el aprendizaje no supervisado es la reducción de dimensionalidad, que se utiliza para disminuir el número de variables o dimensiones en los datos. Esto se logra a través de la identificación de las variables más importantes y la eliminación de aquellas que no contribuyen significativamente a la variabilidad en los datos.

Una de las principales ventajas del aprendizaje no supervisado es que no requiere la recopilación de grandes cantidades de datos etiquetados, lo que puede ser costoso y consume mucho tiempo. Además, el aprendizaje no supervisado puede ser muy útil para descubrir patrones y relaciones en los datos que no son evidentes a simple vista, lo que puede ayudar a los investigadores y analistas a identificar tendencias y patrones en los datos que podrían ser útiles en la toma de decisiones.

Por ello, el aprendizaje no supervisado se enfoca en descubrir patrones y relaciones en los datos sin la necesidad de etiquetas o respuestas predefinidas.

Aprendizaje Por Refuerzo

El aprendizaje por refuerzo es una técnica de aprendizaje automático que se enfoca en el desarrollo de modelos de inteligencia artificial capaces de aprender a través de la interacción con un entorno. En el aprendizaje por refuerzo, un agente de inteligencia artificial aprende a realizar una tarea específica a través de la retroalimentación que recibe del entorno en forma de recompensas o castigos.

La idea detrás del aprendizaje por refuerzo es similar a la forma en que los seres vivos aprenden. Cuando los seres vivos realizan una tarea, reciben retroalimentación en forma de recompensas o castigos según su desempeño. Los seres vivos aprenden a realizar las tareas gracias a la retroalimentación que reciben. El aprendizaje por refuerzo busca replicar este proceso de aprendizaje en los sistemas de inteligencia artificial.

En el aprendizaje por refuerzo, el agente de inteligencia artificial toma una acción en función del estado actual del entorno y recibe una recompensa o castigo según su desempeño. Si la acción del agente conduce a un resultado positivo, se le otorga una recompensa positiva. Si la acción del agente conduce a un resultado negativo, se le otorga un castigo negativo. Con el tiempo, el agente aprende a tomar decisiones que maximizan la cantidad de recompensas recibidas y minimizan la cantidad de castigos.

2.1.3 Modelos no supervisados para la detección de anomalías y clustering

Los modelos de aprendizaje no supervisado para detectar puntos anómalos están diseñados para poder detectar anomalías en un conjunto de datos sin haber realizado una etiquetación previa de los datos, ni haber definido previamente un conjunto de reglas que determine si un punto es o no anómalo.

Autoencoders

Autoencoders: son un tipo de red neuronal que se utiliza en aprendizaje no supervisado y en la detección de anomalías (Hinton, G, 2006, Reducing the Dimensionality of Data with Neural Networks). Funcionan codificando los datos de entrada en un espacio de características de menor dimensión (llamado espacio latente), y luego decodificando la codificación para reconstruir los datos originales. El objetivo de los Autoencoders es minimizar el error de reconstrucción entre los datos de entrada y la salida decodificada.

En la detección de anomalías, el Autoencoder se utiliza para aprender las características de datos normales y ser capaz de reconstruirlos a partir de un conjunto de entrenamiento.

Luego, se evalúa la capacidad del Autoencoder para reconstruir datos nuevos y no vistos. Si un dato tiene un error de reconstrucción alto en comparación con los datos de entrenamiento normales, se considera un punto atípico o anómalo.

Los Autoencoders son especialmente útiles en la detección de anomalías en datos de alta dimensionalidad, donde puede ser difícil encontrar patrones y características significativas.

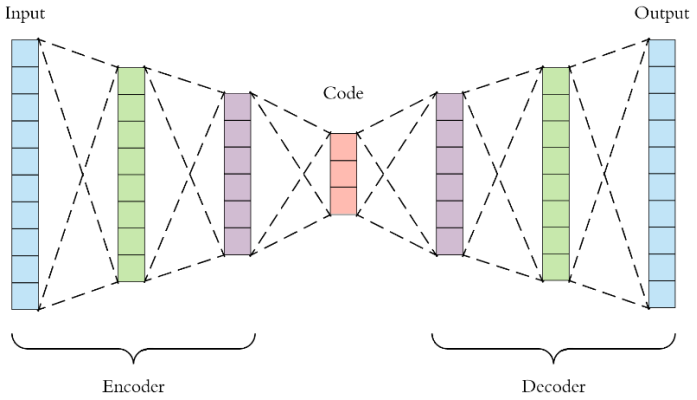

Figura 2.3: Representación gráfica del Autoencoder

http://www.cs.us.es/~fsancho/?e=232

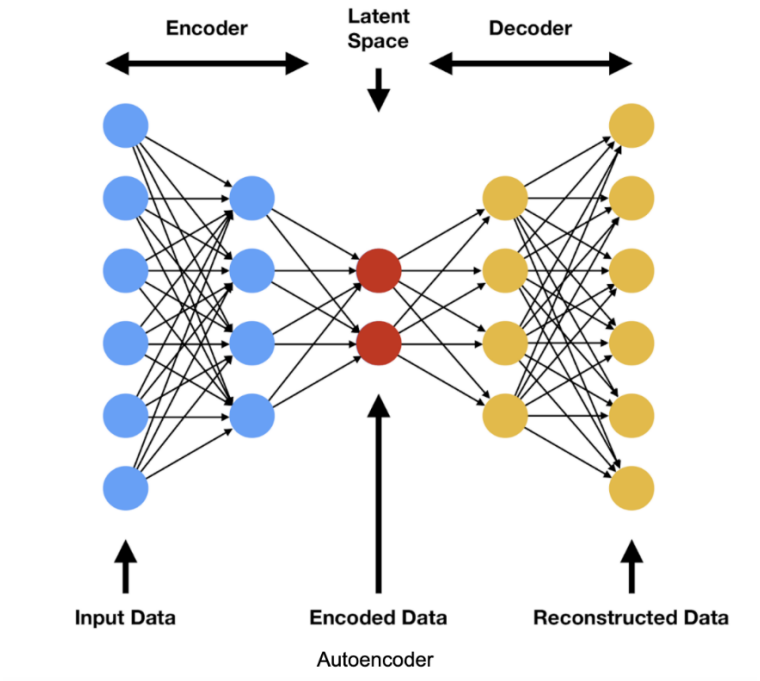

Figura 2.4: Representación gráfica de un Autoencoder

https://www.query.ai/resources/blogs/how-to-use-autoencoder-for-anomaly-detection/

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications With Noise): técnica de aprendizaje no supervisado para la clusterización (agrupación en grupos distintos) de datos, propuesta por Martin Ester, Hans-Peter Kriegel, Jorg Sander y Xiaowei Xu en 1996 (Ester, M., et al., 1996, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise). DBSCAN forma grupos de observaciones basándose normalmente en la distancia euclídea entre puntos para la agrupación y un número mínimo de puntos. A su vez, se considerarán anómalos aquellos puntos ubicados en regiones de baja densidad, es decir, regiones donde los puntos más cercanos están a gran distancia.

El algoritmo requiere dos parámetros principales, la distancia necesaria para considerar a dos puntos como parte del mismo grupo y el mínimo número de puntos que se necesita para formar una agrupación.

DBSCAN es un algoritmo que se puede utilizar para identificar patrones en un conjunto de datos, de tal manera que se puedan encontrar relaciones entre los datos.

En el caso de este trabajo se utilizará DBSCAN, junto a otros modelos, para intentar agrupar los tweets, de tal manera que aquellos tweets que no puedan ser categorizados en ningún grupo, se considerarán tweets troll, dedicados a la propagación de desinformación.

DBSCAN es un modelo robusto ante outliers, esto quiere decir, que las predicciones del modelo no se ven afectadas ante valores extremos. Además, la mayor ventaja que ofrece sobre algoritmos de clustering similares, como K-means, es que no es necesario definir de antemano el número de agrupaciones, lo cual suele ser un proceso arbitrario.

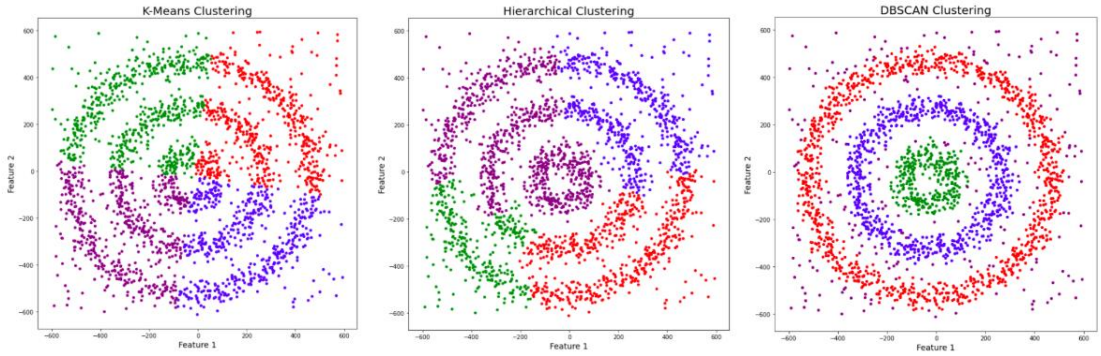

Figura 2.5: comparativa de la detección de outliers entre K-Means, Hierarchical Clustering y DBSCAN

https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/

Estas imágenes comparan el desempeño de distintos modelos no supervisados de clustering a la hora de agrupar observaciones y detectar anomalías. Siendo el morado el color asociado a las anomalías, y el resto de colores representando distintas agrupaciones, se observa como el DBSCAN agrupa de forma muy precisa los datos, y detecta bien anomalías frente a los otros dos modelos, que llegan a confundir datos que deberían pertenecer a una agrupación como anomalías.

One Class SVM (Support Vector Machine)

One-class SVM (Support Vector Machine): variación del modelo de aprendizaje supervisado SVM para la detección de anomalías de forma no supervisada, por lo que no utilizará datos etiquetados para su entrenamiento (Schölkopf, B., et al., 2000, Support Vector Method for Novelty Detection).

Un modelo One-Class SVM es un algoritmo de aprendizaje automático que se utiliza para detectar objetos anómalos o outliers en un conjunto de datos. Está diseñado para aprender un modelo de una sola clase de datos considerados normales, para que ese modelo sea capaz de identificar datos que se alejan significativamente de esa clase.

El funcionamiento de un One-Class SVM es similar al de otros algoritmos de clasificación, como SVM o Naive Bayes. Primero, se entrena el modelo con un conjunto de datos de entrenamiento que se considera normal. Luego, el modelo utiliza una función

de margen para identificar los datos que se alejan significativamente de la clase normal. Estos datos son considerados outliers o anómalos.

Una de las principales ventajas de los modelos One-Class SVM es que son muy eficaces para manejar datos con muchas dimensiones. Además, también son capaces de detectar anomalías en datos en los que la distribución no es conocida.

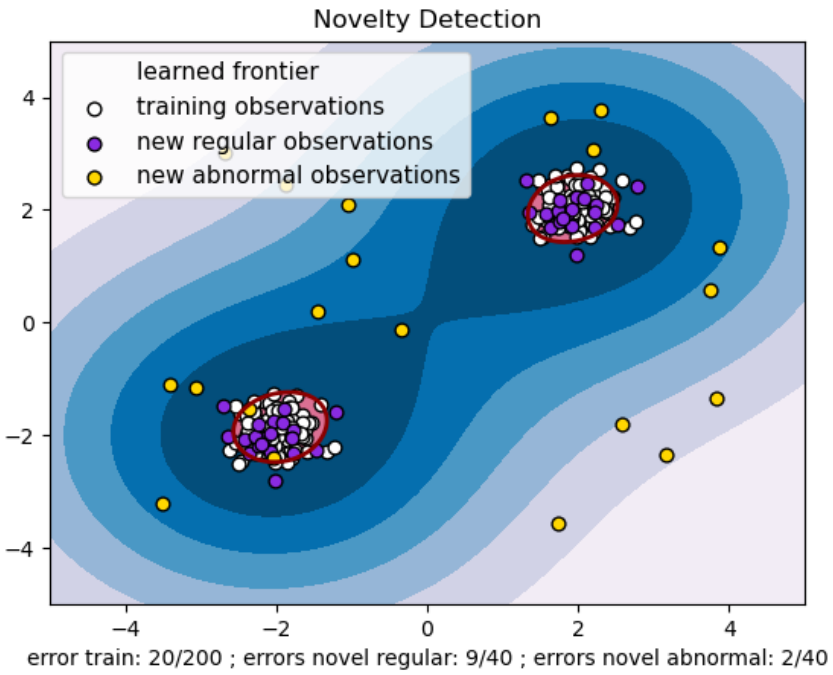

Figura 2.6: Representación gráfica de un modelo One-Class SVM
https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html

Local Outlier Factor

Local Outlier Factor: El modelo Local Outlier Factor (LOF) es un algoritmo de detección de anomalías utilizado para identificar puntos atípicos (outliers) en un conjunto de datos. LOF mide la desviación de densidad local de un punto de datos en comparación con sus vecinos cercanos. Un punto se considera atípico si su desviación de densidad local es significativamente mayor que la de sus vecinos cercanos (Breunig, M.M, Kriegel, H.P., Sander, J., 2000, LOF: Identifying Density-Based Local Outliers).

El algoritmo LOF funciona calculando la distancia entre los puntos de datos y sus vecinos cercanos, y luego utilizando estos valores de distancia para determinar la desviación de densidad local de cada punto. Se asigna una puntuación a cada punto en función de su desviación de densidad local, donde los puntos con puntuaciones más altas se consideran más atípicos.

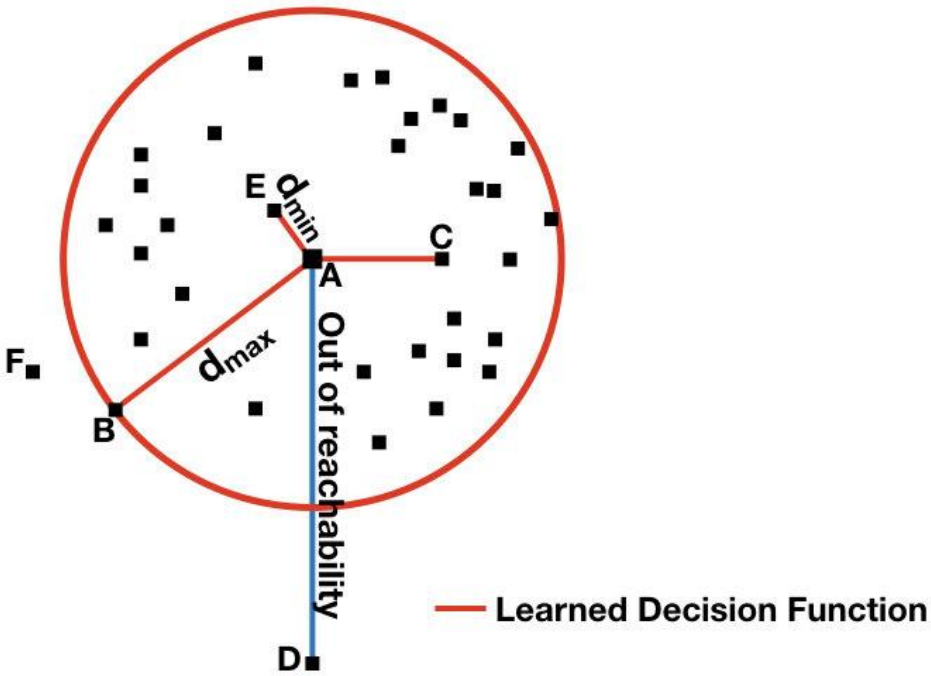

Figura 2.7: Representación de cómo determina LOF anomalías. Cada punto se compara con sus vecinos para determinar si es o no es anómalo.
https://www.researchgate.net/figure/Local-Outlier-Factor-Each-point-is-compared-with-its-local-neighbours-instead-of-the_fig3_328376861

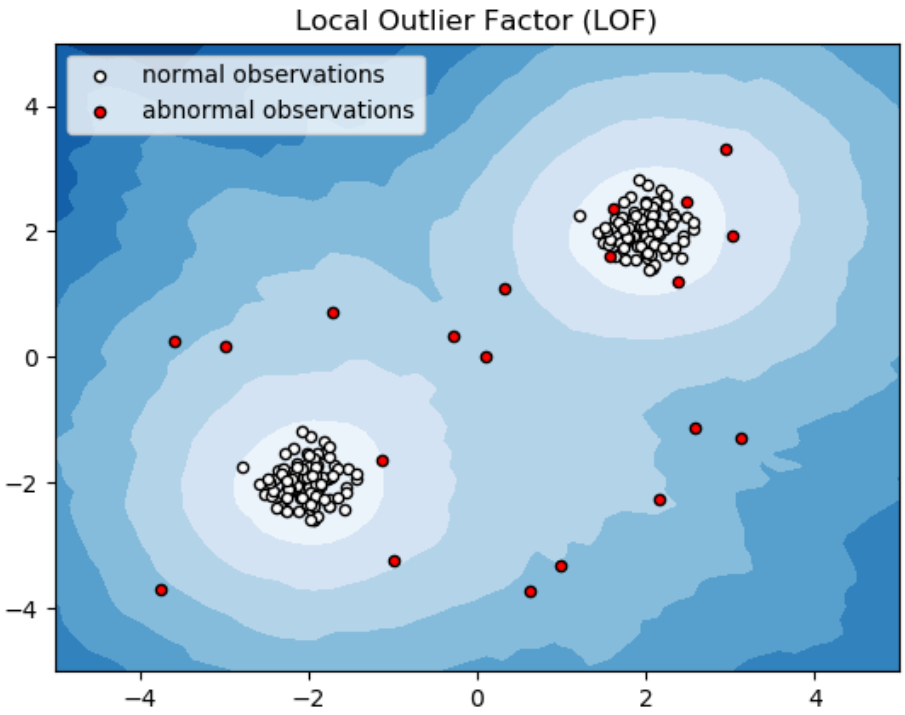

Figura 2.8: Representación gráfica de un modelo LOF con un dataset completo.
https://scikit-learn.org/0.19/auto_examples/neighbors/plot_lof.html

Isolation Forest

Isolation Forest: método de aprendizaje no supervisado para identificar anomalías. El modelo se compone de varios árboles binarios, los cuales se denominan Isolation Trees. Para entrenar el modelo, primero se elige una muestra de los datos de entrenamiento de forma completamente aleatoria, y se asigna esa muestra a un Isolation Tree. Este árbol escoge una característica al azar, y a partir de ella, el árbol se ramifica en función de un valor aleatorio contenido entre el valor mínimo y máximo que puede tener la característica escogida, llamada threshold. Así, los datos pasarán al nodo de la izquierda si el número asociado a esa característica es menor al valor del threshold, y pasarán al nodo derecho en caso contrario. Se repetirán estos pasos bien hasta que cada observación quede aislada o, si hay definido un valor de profundidad máxima, hasta que se llegue a ese valor. Todo esto se repetirá múltiples veces para crear múltiples Isolation Trees, que serán los que

formen el modelo de Isolation Forest (Liu, F.T., Ting, K.M., Zhou, Z.H, 2008, Isolation Forest).

A la hora de predecir si una observación es o no una anomalía, se pasará el dato a cada uno de los árboles que componen el modelo, y se asignará una puntuación (-1 en caso de anomalía, 1 en caso de no anomalía) en función de la profundidad a la que se ha tenido que llegar en cada árbol del modelo para alcanzar ese punto.

El modelo asigna las puntuaciones a las observaciones en función de un parámetro llamado contaminación, que es el porcentaje de anomalías esperadas en el conjunto de datos. Este parámetro toma un valor desconocido a priori, por lo que se deben probar distintos valores y comprobar en función de ellos el desempeño del modelo. A este tipo de parámetros se les denomina hiperparámetros.

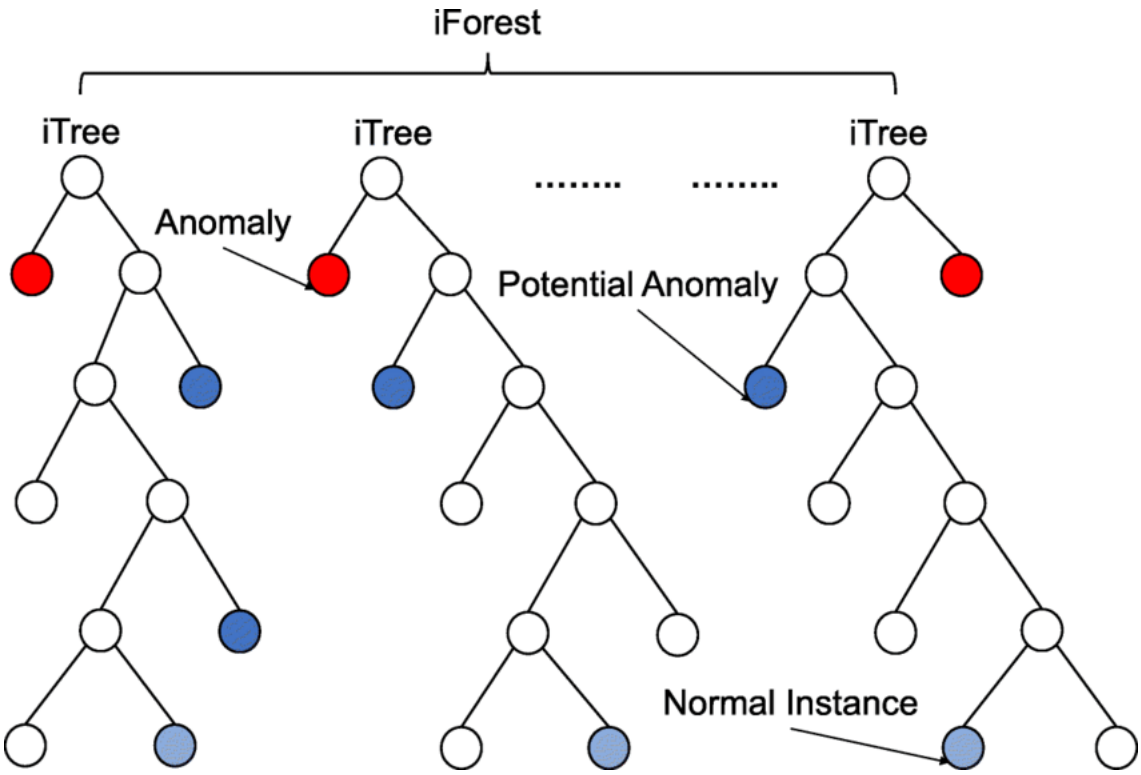

Figura 2.9: Representación gráfica de un modelo Isolation Forest.
https://www.researchgate.net/figure/Isolation-Forest-learned-iForest-construction-for-toy-dataset_fig1_352017898

2.1.4 Obtención de datos

Para obtener los datos con los que utilizar los modelos, o bien se utilizan datos recopilados en otros trabajos e investigaciones, o bien se obtiene un dataset propio para este trabajo.

Para obtener un dataset propio de tweets, se necesita acceso a la API de Twitter y un programa que se conecte a esa API y a través de ella, obtenga los datos que se necesiten.

API de Twitter

La API de Twitter es una interfaz de programación de aplicaciones (Application Programming Interface o API, por sus siglas en inglés) que permite a los desarrolladores acceder a los datos y funcionalidades de la plataforma de Twitter y utilizarlos en sus propias aplicaciones y servicios.

La API de Twitter ofrece una amplia gama de funciones que los desarrolladores pueden utilizar, incluyendo la posibilidad de leer y escribir tweets, acceder a los perfiles de los usuarios, buscar tweets por palabras clave o ubicación, acceder a las métricas asociadas a cada tweet, como el número de likes, número de retweets o número de contestaciones. Este servicio que ofrece Twitter a los desarrolladores puede ser utilizado para crear aplicaciones que tengan funcionalidades que integren Twitter o para la visualización y análisis de datos de Twitter.

Como se ha mencionado previamente, gracias a la API de Twitter, se pueden desarrollar aplicaciones dedicadas a la obtención masiva de tweets, para su posterior análisis.

Previamente, el acceso a la API de Twitter era gratuito, y ofrecía varios planes, Essential, Elevated y Academic. Las diferencias principales entre estos planes es el número de peticiones que se pueden realizar a la API en un tiempo determinado, el número de tweets al mes que se podían obtener y poder acceder no solo a los tweets recientes sobre un tema determinado, sino poder obtener cualquier tweet independientemente de la fecha de publicación de éste.

Con la reciente compra de la red social por parte de Elon Musk, se anunciaron cambios para la API de Twitter. Las funcionalidades que se van a añadir o mantener a día de hoy

se desconocen, pero si se planea eliminar el acceso gratuito a la API y sustituirlo por planes de pago, empezando el plan básico en 100 dólares al mes. Esto dificultaría el desarrollo de aplicaciones que utilicen la API de Twitter.

Scraper de tweets

Un tweet scraper es un programa o herramienta que se utiliza para extraer tweets u otra información de Twitter en grandes cantidades, o bien utilizando la API de Twitter o bien utilizando técnicas de scrapeo de páginas web.

Estas herramientas pueden ser utilizadas para la recopilación de datos de Twitter a gran escala, para su posterior análisis y visualización, intentando así lograr información valiosa de los datos recopilados. Así, se pueden tener distintos objetivos, como obtener información acerca de que temas son tendencia en la red social, conocer la opinión pública acerca de temas concretos, obtener patrones de comportamiento de los usuarios y así poder agruparlos en función de estos patrones, detección de información falsa y de cuentas dedicadas a desinformar mediante la propagación de noticias falsas entre otros posibles objetivos.

Un scrapeador de tweets puede extraer un sinfín de formación como el texto de los tweets, el nombre de usuario que ha publicado el tweet, la fecha y hora de publicación, el número de retweets y "me gusta"…, es decir, cualquier información relativa a un tweet. Pero la API también permite obtener información acerca de los usuarios, como si tiene descripción, foto de perfil, foto de banner, número de seguidores y número de usuarios a los que sigue. Así, este tipo de aplicaciones no solo permite obtener información acerca de las publicaciones, sino también sobre quién realiza esas publicaciones.

Asimismo, a la hora de desarrollar un scrapeador, se debe tener en cuenta las términos y condiciones de Twitter para el uso de la API, pues, por ejemplo, condiciona el uso de la información que se obtenga mediante su uso. También se debe tener en cuenta las normas y leyes estatales, pues pueden prohibir la recolección de datos de redes sociales. El scraping en España es legal siempre y cuando se cumplan los términos y condiciones que dictamina el propietario de la página web la cual se quiera scrapear.

2.2 Trabajos relacionados

En este apartado de trabajos relacionados, se mencionarán algunos de los avances para la detección de trolls en Twitter, así como algunas de las técnicas empleadas para lograr este objetivo.

Artículo 1

(Bot and Gender Detection of Twitter Accounts Using Distortion and LSA, Andrea Bacciu, Massimo La Morgia, Alessandro Mei, Eugenio Nerio Nemmi, Valerio Neri, and Julinda Stefa, 2019)

Este artículo intenta desarrollar una herramienta que permita detectar en Twitter cuentas que sean bots, es decir, cuentas automatizadas que generan mensajes de forma automática, siendo una de las finalidades de este tipo de cuentas hacer que parezcan personas interactuando en la red social y no un programa informático. También en este artículo se quiere ser capaz que, en caso de que la cuenta sea catalogada como persona y no bot, ser capaz de identificar su género, aunque para este trabajo, lo que realmente interesa es la detección de bots, pues hay muchos trolls que o bien son bots, o bien siguen patrones de comportamiento de bot.

Para ello, se parte de un dataset con cuentas que publican en inglés y con cuentas que publican en español. En el dataset inglés, hay 2880 cuentas que se utilizarán en el entrenamiento del modelo, y 1240 para el test. En caso del dataset en español, hay 2080 cuentas para el entrenamiento, y 920 para el test. De cada cuenta, hay 100 tweets en los datasets. Cada cuenta esta etiquetada si es bot o no, y si no es bot, el género de la persona que tiene la cuenta.

Lo primero para tener en cuenta son las características que se van a tener en cuenta para entrenar el modelo de cada usuario. Así, en el artículo enumeran las siguientes características: número medio de emoticonos utilizados, número medio de enlaces por publicación, número medio de hashtags, longitud promedio de los tweets y retweets, número medio de “;” utilizados, la media de las distancias coseno entre los tweets del

usuario, puntuación otorgada de media por el algoritmo Vader de Sentiment Analysis, catalogando cada tweet en positivo, neutro o negativo y distorsión de texto, ocultando los caracteres ASCII para centrarse en caracteres especiales y símbolos de puntuación, intentando así obtener patrones de escritura.

Una vez se tienen todas las características, se reduce la dimensionalidad utilizando PCA (Primary Component Analysis), cuyo objetivo es reducir el número de features manteniendo la mayor cantidad de información posible.

Una vez tiene los autores tienen los datos preparados, proceden al entrenamiento de los modelos, uno para las cuentas en inglés y otro para cuentas en español. Para inglés, utilizará un clasificador dividido en dos etapas. La primera, formada por un SVM junto con AdaBoost, un método ensemble de boosting (Boosting fue propuesto por Freund y Schapire en 1997 para mejorar predicciones de los modelos), de tal forma que va entrenando distintos modelos SVM, de tal manera que se van adaptando los clasificadores a los casos más complicados de categorizar, ajustando los pesos de aquellos casos que han sido mal categorizados. La segunda capa está formada por un Soft-Voting Classifier, que calcula a partir de las predicciones de la capa anterior, la clase con mayor probabilidad. En caso del modelo para español, solo se utiliza una SVM, pues según los autores, obtiene mejores resultados así que replicando la estructura anterior. El modelo en inglés obtiene resultados cercanos al 95% de accuracy, mientras que el modelo español obtiene resultados cercanos al 90%.

A pesar de que el objetivo era utilizar técnicas no supervisadas para la detección de trolls, pues no se parte de datasets previamente etiquetados y los autores del artículo emplean técnicas supervisadas, todo lo relacionado a las métricas a obtener de los usuarios resulta muy útil, pues permite obtener más información, permitiendo así que los modelos no supervisados obtengan más patrones para clasificar a los usuarios en trolls o no trolls, y así, mejorar la precisión de los modelos.

Artículo 2

(Identification of Twitter Bots Based on an Explainable Machine Learning Framework: The US 2020 Elections Case Study, Alexander Shevtsov, Christos Tzagkarakis, Despoina Antonakaki, Sotiris Ioannidis, 2021)

Este artículo tiene por objetivo la identificación de bots en Twitter en las elecciones estadounidenses del 2020. Para ello, utilizará un dataset que contiene tweets que contengan los hashtags más populares en relación con las elecciones, mediante el uso de la API de Twitter. El dataset contiene 15.6 millones de tweets y 3.2 millones de usuarios.

Debido a que se utiliza la API de Twitter para obtener el dataset, los datos obtenidos están sin etiquetar, y debido al gran coste que tendría categorizar a esos usuarios como trolls o no trolls, los autores hacen uso de herramientas ya desarrolladas para la etiquetación de cuentas, como Botometer (Varol et al) y BotSentinel. Los autores combinan los resultados de ambas herramientas para el etiquetado, siendo bots aquellos datos que ambas detecten como bot, humanos aquellos que ambos consideren humanos, y en los casos dudosos, es decir, donde se contradicen las predicciones de las herramientas, no se asigna a esos datos ninguna etiqueta. Debido a que Botometer tiene un límite de peticiones diarias, al contrario que BotSentinel, el flujo de trabajo para la etiquetación es el siguiente: se hacen peticiones a BotSentinel, se guarda la etiqueta que devuelve y se vuelve a pasar este dato o bien a BotSentinel o bien a la API de Twitter para comprobar si la cuenta está eliminada. De esta manera, se reducen el número de peticiones a realizar a Botometer para comprobar si es o no bot, pues cuentas que hayan sido suspendidas por Twitter se asume que eran bots.

Una vez tienen los datos etiquetados, proceden a la extracción de características de esos datos. Los autores agrupan estas características en cuatro grupos diferenciados, siendo estos, características de perfil, de contexto, de tiempo y de interacción.

En el grupo de características de perfil o de cuenta destacan el número de seguidores y de usuarios que siguen una cuenta, número de favoritos, longitud de la descripción, ubicación, uso de imagen de perfil o la similitud entre el nombre de usuario y el nombre de la cuenta (el nombre de usuario es modificable y es el que aparece siempre destacado,

mientras que el nombre de cuenta es el @ del usuario). La similitud se calcula utilizando la distancia de Jaccard.

En el grupo de características de contexto se extrae información como el número de URLs en los tweets, hahstags populares utilizados por el usuario, el propio contenido de los tweets del usuario, así como las palabras más frecuentes utilizadas por el usuario.

Las características de tiempo tienen como objetivo obtener la distribución de actividad de las cuentas en función del tiempo. Así, obtienen características como la actividad diaria y la actividad cada hora, el porcentaje de tweets diarios y el intervalo de tiempo en el que el usuario publica contenido.

Para las características de interacción, primero se realizan grafos, en el que los usuarios se relacionan mediante retweets. De esta manera, los usuarios serían los nodos y las aristas, los retweets que los relacionan. Una vez hecho esto, las características extraídas son estadísticas de los grafos, como los grados de entrada (corresponderían con el número de aristas que inciden en un nodo en concreto) y los grados de salida (corresponderían con el número de aristas que salen de un nodo) de cada usuario.

Hecho esto, los autores comparan el desempeño de modelos supervisados como Random Forest, Support Vector Machine y XGBoost (Extreme Gradient Boost). Utiliza una combinación de entrenamiento 80/20 (80% entrenamiento, 20% test) y cross-validation para comparar los modelos mencionados, y XGBoost obtiene los mejores resultados, logrando un AU-ROC del 97.9%

Los autores también hacen énfasis en que una posible nueva rama de investigación podría ser un análisis de texto sobre las publicaciones de los trolls para una buena identificación del tipo de contenido promovido por estas cuentas.

De este artículo, al igual que del anterior, se obtienen posibles métricas a introducir en este trabajo muy interesantes, como los hashtags que utiliza cada usuario, o la similitud entre nombre de usuario y cuenta, que pueden posibilitar un mejor desempeño de los modelos.

Artículo 3

(Detecting Social Media Bots with Variational AutoEncoder and k-Nearest Neighbor Xiujuan Wang, Qianqian Zheng, Kangfeng Zheng, Yi Sui, Siwei Cao y Yutong Shi, 2021)

El objetivo de este artículo es el desarrollo de una herramienta que pueda detectar bots en la red social Twitter, utilizando para ello Variational AutoEncoders (VAE) y algoritmos de detección de anomalías. El objetivo es que el VAE pueda aprender las características de los datos recogidos del dataset CLEF2019 (el mismo dataset utilizado en el artículo 1, con tweets en inglés y en español etiquetados como bot o no bot), de tal manera que el autor supone que aquellas características de usuarios no bots, no variarán demasiado tras la decodificación del AutoEncoder, al contrario que en caso de que el usuario sea un bot.

El autor decide antes de usar ningún algoritmo de aprendizaje extraer características de los datos para una mejor precisión de los modelos. Algunas de estas características son las siguientes:

Media de menciones por tweet: el autor supone que la media de menciones de usuarios reales será mayor que la de los bots, pues dice que cuentas que utilicen muchas menciones son susceptibles de ser cerradas por las redes sociales.

Media de emoticonos por tweet: según el autor, los bots o bien abusan de introducir emoticonos en los tweets, o bien no introducen ninguno.

Media de stopwords por tweet: el autor expone que el uso de stopwords en caso de bots es mucho menor a su uso por parte de usuarios reales, pues según él, las stopwords representan los hábitos de lenguaje de la persona.

Media de temas (hashtags) por tweet: los bots suelen utilizar todo tipo de hashtags para que su mensaje se propague al mayor número de usuarios posible, mientras que los usuarios suelen tener un número de temas menor, pues los intereses de cada usuario no suelen variar demasiado.

Similitud media de los tweets: el autor expone que los contenidos publicados por bots suelen ser siempre parecidos, en contraparte de los usuarios reales. Así, el autor utilizará la distancia coseno y TF-IDF (Term Frequency Inverse Document Frequency) para calcular la similitud entre tweets.

Media de símbolos de puntuación: se calculará esta métrica para los símbolos “,”, “.”, “;”, “””, “!”, “(“ y “)” (es decir, comas, puntos, puntos y comas, comillas, exclamaciones y paréntesis). Según el autor, hay diferencias en el uso medio de estos símbolos entre usuarios y bots.

Una vez hecho esto, se propone combinar el uso de kNN y VAE, de tal manera que el clasificador recibirá las características del tweet y las características que logra decodificar el AutoEncoder. Con ello, logra una precisión del 98,34%.

También experimenta con algoritmos de detección de anomalías, entre ellos: ABOD (Angle Based Outlier Detector), CBLOF (Cluster Based Local Outlier Factor), Feature Bagging, HBOS (Histogram Based Outlier Score) o Isolation Forest. De aquí, compara mediante el uso de la curva ROC el desempeño de los algoritmos, siendo los mejores ABOD, con un 94% de precisión, Isolation Forest, con un 90% y Feature Bagging con un 89,76%. CBLOF lográ una precisión similar a Feature Bagging (89,16%) y HBOS consigue una precisión muy baja en comparación con el resto, un 66,89%.

Concluye exponiendo que a pesar de que su método de detección (VAE combinado con kNN), una posible rama de mejora sería encontrar nuevas características que permitiesen una distinción más clara entre bots y no bots, y que se tendría que adaptar este tipo de modelos a más lenguas, para poder detectar bots en cualquier idioma.

De este artículo destaco el uso de métricas como la similitud media o el uso de símbolos de puntuación medio, así como el uso de los algoritmos de detección de anomalías. La comparación de las curvas ROC de estos algoritmos me permite tener una noción previa sobre qué algoritmos utilizar y qué algoritmos no utilizar a priori.

3. ASPECTOS METODOLÓGICOS

3.1 Metodología

Para el desarrollo de este trabajo se ha optado por el uso de metodología ágil (https://agilemanifesto.org/, Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Sutherland, J., 2001, Agile manifesto), permitiendo esto centrar los esfuerzos en el desarrollo software y responder bien ante el cambio. La metodología concreta empleada para el desarrollo es Scrum mezclado con Kanban.

De esta manera, se realizó un tablero Kanban para anotar las distintas tareas a realizar para la consecución del proyecto. En el tablero se distinguen tres columnas, la de tareas a realizar, la de tareas en proceso y la de tareas realizadas. Esto permite comprobar de forma visual el estado del proyecto y de las distintas tareas asociadas a éste. Asimismo, las iteraciones o sprints fueron de una semana de duración, donde todo el equipo se reunía para analizar el estado actual del proyecto, si era necesario volver atrás en alguna tarea en la que no se consiguió los resultados esperados y las próximas tareas a realizar.

En cuanto a los roles, Scrum define tres roles principales, el equipo de desarrollo, el scrum master y el product owner.

- Equipo de desarrollo: equipo autoorganizado y autónomo, por lo que son los responsables de realizar el trabajo necesario para cumplir los objetivos marcados en cada sprint. Son los encargados por tanto de realizar todas las tareas necesarias para completar el trabajo estipulado en cada sprint.
- Scrum master: es el responsable de garantizar que se está utilizando el marco de trabajo Scrum de forma correcta y ayuda al equipo de desarrollo en todo lo posible, para facilitar el flujo de trabajo de los desarrolladores.
- Product owner: es el encargado de garantizar que se entrega el máximo valor posible en el producto desarrollado, y dictamina al equipo de desarrollo los objetivos importantes que se deben alcanzar. El product owner define la visión del producto a ser desarrollado y participa en la planificación de los sprints, definiendo éste los objetivos a alcanzar en las iteraciones.

Gracias a este tipo de metodología, ha sido posible adaptarse bien ante cambios imprevistos, como retroceder a objetivos que se creían ya realizados, ya sea para solventar algún error o porque se debe volver a desarrollar pues no cumple con lo esperado, o modificar lo desarrollado para cumplir mejor con el objetivo.

3.2 Tecnologías empleadas

3.2.1 Entornos de programación

Python (v. 3.8.8)

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Fue creado por Guido van Rossum, y se lanzó por primera vez en 1991 (https://www.python.org/doc/, Python Software Foundation, 2021). Este lenguaje es utilizado principalmente para desarrollo web, con frameworks como Django y Flask, análisis y visualización de datos, con librerías como Pandas, NumPy o Matplotlib, centrándose las dos primeras en tratamiento y análisis de datos, y Matplotlib siendo utilizada para tareas de visualización, y tareas relacionadas con inteligencia artificial, con librerías como Tensorflow, Keras o Pytorch, facilitando a los programadores la creación de modelos de aprendizaje automático o inteligencia artificial.

Uno de los aspectos a destacar de Python es que es sencillo de utilizar, facilitando la comprensión del código a los programadores, gracias a una sintaxis clara y sencilla. A su vez, Python es un lenguaje muy flexible, que permite utilizarlo para todo tipo de ámbitos, desde el desarrollo de páginas web, al desarrollo de modelos de aprendizaje automático, y debido a ser multiplataforma, se puede utilizar en cualquier sistema operativo sin problema. Además, gracias a la gran comunidad de usuarios, motivada por ser Python un lenguaje de código abierto, no sólo hay un amplio catálogo de librerías desarrolladas por la comunidad para facilitar las labores de codificación a otros programadores, sino que hay gran cantidad de información disponible en caso de necesitar resolver cualquier problema.

Python es el lenguaje perfecto para la problemática que se está tratando, pues es uno de los lenguajes más populares para el análisis de datos y aprendizaje automático, gracias a la gran cantidad de librerías disponibles, como las mencionadas previamente.

Jupyter Notebooks

Jupyter Notebooks son entornos interactivos web que permiten trabajar con múltiples lenguajes de programación, como Python o R, que permiten combinar en un único documento código, texto y visualizaciones.

Los notebooks están compuestos por celdas, las cuales pueden contener código o texto. Estos notebooks permiten una rápida visualización de los resultados de ejecutar cierto código, pues debajo de las celdas de código se muestra el resultado de la ejecución. Gracias a esto, Jupyter Notebook es uno de los mejores entornos para probar código, como explorar algoritmos nuevos o separar una funcionalidad en partes para comprobar de forma rápida y visual que cada parte devuelve los resultados esperados.

En este trabajo se ha optado por utilizar Jupyter Notebooks para la limpieza, cálculo de nuevas métricas y puesta en marcha de los modelos, para poder comprobar de forma rápida y visual qué resultados se están obteniendo con la ejecución de cada parte del código, además de poder estructurar el código de una forma más clara gracias a que se puede añadir texto que indique qué función tiene cada conjunto de celdas.

3.2.2 Librerías de Python

Scikit-Learn (v. 1.2.2)

Scikit-Learn es una librería open source para el lenguaje de programación Python enfocada en el aprendizaje automático. Esta librería comenzó siendo un proyecto de un desarrollador francés, David Cournapeau, como proyecto para el Google Summer of Code de 2007 como una extensión de terceros para la librería SciPy. En 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort y Vincent Michel, tomaron las riendas

del proyecto, y lanzaron la primera versión de Scikit-Learn el 1 de febrero de 2010 (Pedregosa, F., et al., 2011, Scikit-learn: Machine learning in Python).

Scikit-learn proporciona una gran variedad de algoritmos de aprendizaje automático, tanto de aprendizaje supervisado, para tareas como la clasificación o la regresión, como de aprendizaje no supervisado, para clustering, reducción de dimensionalidad o detección de anomalías.

Scikit-Learn también ofrece otras funcionalidades, como normalizar y estandarizar los datos, herramientas para la evaluación del desempeño de los modelos de aprendizaje automático o búsqueda de hiperparámetros.

Además, esta librería se integra con otras librerías populares de Python muy útiles para la ciencia de datos, como Pandas, Numpy o Matplotlib, librerías dedicadas al procesamiento y visualización de datos.

De esta librería se utilizarán algunos de los modelos que ofrece para la detección de anomalías, en concreto DBScan, Isolation Forest, OneClassSVM y LocalOutlierFactor.

Pandas (v. 1.2.4)

Pandas es una librería open source para Python para el análisis y estructuración de datos (McKinney, W., 2010, Data structures for statistical computing in Python). Pandas ofrece tres estructuras de datos principales, series, dataframes y paneles. Las series son estructuras unidimensionales de datos, es decir, una única columna con datos, similar a un array de Python. Los dataframes son estructuras bidimensionales de datos, formada por filas o columnas, como una hoja de Excel o una tabla SQL. Un panel es una estructura tridimensional de datos.

Pandas ofrece a su vez funcionalidades para tratar con estas estructuras de datos, como filtrado, selección, transformación y eliminación de datos.

Pandas también ofrece compatibilidad con una gran variedad de tipo de datos, como ficheros CSV, XLS, JSON…, facilitando así la importación y exportación de datos de distinto tipo, permitiendo el tratamiento de datos procedentes de distintas fuentes.

Pandas se utilizará para la lectura y almacenamiento de los datos recogidos de la API de Twitter, almacenados en distintos ficheros CSV. También se empleará para el tratamiento de los datos y para añadir métricas calculadas a los datos recogidos. Además, gracias a la integración entre Pandas y Scikit-Learn, se podrán entrenar los modelos directamente con DataFrames de Pandas, sin necesitar una transformación de los datos previa.

NumPy (v. 1.21.0)

Numpy es una librería open-source para el lenguaje Python enfocada al trabajo con arrays y a distintas operaciones matemáticas, proporcionando herramientas eficientes para cálculos matemáticos (Van der Walt, S., Colbert, S. C., Varoquaux, G., 2011, The NumPy array: A structure for efficient numerical computation).

Gracias a la capacidad de manejar grandes volúmenes de datos y realizar operaciones matemáticas con ellos de forma rápida y eficiente, es una librería muy utilizada en el ámbito de ciencia de datos.

Otra característica de NumPy a tener en cuenta es su integración con otras librerías de Python como Pandas, Scikit-Learn o Matplotlib. NumPy constituye la base sobre la que se construyó Pandas, y debido a ello, las estructuras de datos propias de Pandas dependen de la implementación de los arrays de NumPy, siendo los valores de las columnas de los dataframes guardados como ndarrays, es decir, arrays específicos de NumPy. Scikit-Learn asume que los datos con los que se trata estén en arrays de NumPy o matrices dispersas de SciPy, pues este tipo de datos son los más eficientes en el caso de la mayoría de las operaciones.

Numpy ofrece una gran variedad de funcionalidades. Algunas a destacar serían la creación y manipulación de arrays y matrices, pudiendo crear matrices de ceros o unos o cambiar las dimensiones de las matrices, operaciones matemáticas y funciones de álgebra lineal, como funciones trigonométricas, redondeo, producto escalar y vectorial entre vectores o producto de matrices entre otras, y funciones estadísticas como medias, medianas, percentiles, desviación estándar…

En este trabajo se utilizará NumPy de forma indirecta debido al uso de Pandas y Scikit-learn, y de forma directa para realizar operaciones como detectar si hay valores infinitos

o NAs en alguna columna del dataframe de datos, para el cálculo de la similitud coseno, y para el redondeo de columnas float con el objetivo de no tener un alto número de decimales.

PyOD (v. 1.0.7)

PyOD es una librería para el lenguaje de programación Python enfocada en la rama de detección de anomalías del aprendizaje no supervisado. Incluye más de 40 algoritmos de detección de outliers, como AutoEncoders, LOF (Local Outlier Factor), ABOD (Angle-Based Outlier Detection) o Isolation Forest (Zhao, Y., Nasrullah, Z., Li, G., 2019, PyOD: A Python toolbox for scalable outlier detection).

PyOD también ofrece funciones para la evaluación de modelos, utilizando métricas como la precisión, la especificidad o el F1-Score, además de funciones para la selección de modelos, pudiendo seleccionar de forma automática el modelo que tiene un mejor desempeño para el conjunto de datos.

PyOD es compatible con librerías como NumPy, Pandas y Scikit-learn, por lo que se puede utilizar junto a estas librerías.

En este trabajo se utilizará PyOD para detección de anomalías mediante el uso de AutoEncoders, y no se podrán utilizar funciones de evaluación, pues partimos de datos completamente sin etiquetar, y por ello, tampoco podremos utilizar las funciones de selección autómatica del mejor modelo.

Tweepy (v. 4.12.1)

Tweepy es una librería open-source que facilita el acceso e interacción con la API de Twitter en Python, permitiendo el desarrollo de herramientas en Python que puedan descargar tweets con toda su información, buscar usuarios mediante IDs de twitter o enviar tweets de forma automatizada, entre otras posibles aplicaciones (Roesslein, J., 2010, Tweepy: Twitter for Python).

Tweepy simplifica la interacción con la API de Twitter, y simplifica el proceso de autenticación con la API, encargándose la librería de todo el proceso de autenticación y gestión de tokens de la API.

Además, Tweepy permite la obtención de datos en tiempo real, pudiendo así desarrollar aplicaciones que procesen y analicen tweets en tiempo real. Tweepy permite a los desarrolladores escoger la información a extraer, como tweets que contengan un determinado hashtag, todos los tweets de un usuario en específico o métricas relacionadas con los tweets.

En este trabajo se utilizará Tweepy para el desarrollo de un scrapeador de tweets por palabra clave. Así, se desarrollará una herramienta que permita la recolección de tweets de un tema específico. Este scrapeador recolectará múltiple información que nos ofrece la API de Twitter, en específico: el id del tweet, el texto del tweet, cuando se publicó el tweet, el idioma del tweet, entities, que contiene información como los hashtags empleados, desde donde se ha publicado el tweet (es decir, desde el cliente web de twitter, desde la app de iOS, desde la app de Android…), si es un quote (es decir, si un usuario ha tweeteado el tweet de una persona con un comentario añadido), si el tweet ha sido truncado (esto se da porque originalmente los tweets eran de 140 caracteres y ahora son de 280, y así la API ofrece compatibilidad con versiones anteriores), el id del tweet original en caso de ser una respuesta, el id del usuario que publicó el tweet original en caso de ser una respuesta, el nombre de usuario en caso de ser una respuesta, las coordenadas y el lugar desde el que se publicó el tweet, el número de retweets, el id del usuario que ha publicado el tweet y su nombre de usuario, la localización del usuario, la descripción del usuario, el número de seguidores del usuario, el número de cuentas a las que sigue el usuario, cuando se creó la cuenta el usuario, el número de likes del tweet, si la cuenta está verificada, la imagen de perfil del usuario, la imagen de banner del usuario y un diccionario con todos los datos del usuario, en caso de necesitar otros datos a posteriori.

RE (v. 2.2.1)

RE es una librería para Python que proporciona funcionalidades para emplear expresiones regulares. Las expresiones regulares permiten comprobar si un texto cumple un patrón

definido, o si dentro de un texto, hay alguna cadena de caracteres que cumpla un patrón específico. Además de encontrar cadenas de texto con determinados patrones, también ofrece funciones para reemplazar determinadas cadenas de caracteres con otra cadena o dividir una cadena de texto según la aparición de algún carácter o patrón específicos.

En este trabajo, se va a emplear esta librería para la búsqueda de elementos importantes en el tweet como los hashtags o las menciones, así como para la eliminación de ciertos caracteres del tweet como pueden ser emoticonos.

Sentence_transformers (v. 2.2.2)

Sentence_transformers es una librería para el lenguaje de programación Python enfocada en generar sentence embeddings, es decir, representaciones vectoriales de frases o textos (Reimers, N., Gurevych, I., 2019, Sentence-transformers: Multilingual sentence embeddings using BERT / RoBERTa / XLM-RoBERTa & Co. with PyTorch).

Esta librería ofrece varios modelos que han sido entrenados previamente, basados en la arquitectura BERT.

Esta librería se utilizará para transformar en vectores los textos que conforman los tweets para que los modelos puedan entrenar teniendo en cuenta el contenido de los tweets.

Matplotlib (v. 3.3.4)

Matplotlib es una librería para el lenguaje de programación Python diseñado para crear visualizaciones de datos (Hunter, J. D., 2007, Matplotlib: A 2D graphics environment). Matplotlib incluye una gran variedad de tipos de gráfico como histogramas, gráficos circulares, gráficos de barras, gráficos de dispersión…, entre otros.

Esta librería está pensada para poder ser utilizada con otras librerías enfocadas a la ciencia de datos, como Pandas o Scikit-Lern, permitiendo realizar visualizaciones sobre dataframes o resultados de los modelos aplicados de forma sencilla.

En este trabajo, se utilizará Matplotlib principalmente para visualizar la distribución de distintos datos.

Spacy (v. 3.5.0)

Spacy es una biblioteca de procesamiento del lenguaje natural (NLP) de código abierto para Python. Proporciona funciones para el análisis sintáctico, la lematización o el análisis semántico.

Spacy utiliza modelos estadísticos y de aprendizaje automático para comprender el lenguaje natural. Estos modelos se han entrenado con millones de palabras etiquetadas para aprender patrones y relaciones en el lenguaje natural. Los modelos se pueden personalizar para adaptarse a los requisitos específicos del proyecto de NLP.

Spacy también es compatible con una amplia gama de bibliotecas y herramientas de análisis de datos de Python, como Scikit-learn, lo que la convierte en una excelente opción para los proyectos de NLP en Python.

3.2.3 Aplicaciones para la gestión del equipo

GitHub

GitHub es una plataforma para el desarrollo de software que permite a sus usuarios almacenar, colaborar y realizar control de versiones de los proyectos software. Los proyectos se almacenan en repositorios, donde se puede almacenar tanto el código como la documentación y permite realizar un seguimiento de los cambios que se han ido realizando.

Se utilizará GitHub para almacenar y colaborar en los distintos códigos que se vayan realizando, teniendo ramas propias para cada miembro del equipo donde se irán subiendo los avances que haya hecho cada miembro, y una vez el código cumpla con todos los

requerimientos y sea revisado por todo el equipo, se subirá a la rama main. De esta manera, en la rama main solamente estará la última versión de los códigos.

El código desarrollado se puede encontrar en el siguiente repositorio de Github: https://github.com/Hertorias/DeteccionTrolls

OneDrive

OneDrive es un servicio de almacenamiento en la nube propiedad de Microsoft. Permite que los usuarios almacenen y compartan ficheros con otros usuarios, y gracias a estar en la nube, se puede acceder a esos ficheros desde cualquier dispositivo.

En este trabajo se utilizará OneDrive para almacenar todos los datos que se hayan scrapeado, teniendo así un lugar donde todos los miembros del equipo puedan acceder fácilmente para descargar los datasets que necesite para el desarrollo del proyecto.

Jira

Jira es una herramienta para la gestión de proyectos y seguimiento de problemas desarrollada por Atlassian. Esta herramienta permite a los equipos de desarrollo software planificar el progreso de distintos proyectos. Así, Jira permite la creación de tareas, historias de usuario, seguimiento del tiempo empleado en cada tarea o ver el estado de desarrollo de las distintas tareas y su prioridad.

4. DESARROLLO

El objetivo de este trabajo es, mediante la utilización de diferentes algoritmos de aprendizaje no supervisado para detección de anomalías, detectar cuentas troll a partir de la información contenida en los datasets obtenidos a través de la API de Twitter y de información que se agregará mediante la obtención de nuevas métricas antes de utilizar los modelos.

Un esquema del flujo del trabajo sería el siguiente:

Figura 4.1: Flujo de trabajo del proyecto.

4.1 Scrapeo y almacenamiento de los tweets

El primer paso para el desarrollo del trabajo es la obtención de datos. Se necesita una cantidad representativa de tweets para poder extraer conclusiones al final del proceso, una vez se hayan entrenado ya los modelos. Para la obtención de datos, hay dos opciones principales, o bien se parte de conjuntos de datos utilizados en otros trabajos, con dos problemas principales, la posible no actualidad de esos conjuntos de datos y que no contengan toda la información que se cree necesaria para la posterior obtención de nuevas métricas, o bien se obtienen datasets propios mediante la aplicación de la API de Twitter, que nos permite elegir tanto el tema sobre el que obtener tweets como la información que se quiere obtener de cada tweet.

El desarrollo del scraper lo llevó a cabo Sergio Esteban (Esteban, S., 2023, Desinformación y manipulación en redes sociales: detección de trolls en Twitter usando técnicas de inteligencia artificial, análisis de redes sociales y minería de texto). Al scraper habrá que indicarle sobre qué tema se quiere obtener tweets, y éste obtendrá, de aquellos tweets que contengan la palabra especificada, la siguiente información:

- ID numérico del tweet. Esto permitirá luego buscar los tweets en el propio Twitter.

- El texto que tiene el tweet.
- La fecha de publicación del tweet
- Idioma en el que se ha publicado el tweet. El idioma es detectado por un algoritmo propio de Twitter.
- Entities, que contiene información, como los hashtags empleados en el tweet o los enlaces presentes en el tweet.
- Desde dónde se ha publicado el tweet, es decir, si el tweet se ha publicado desde el cliente web, desde la app de iPhone, desde la app de Android…
- Si el tweet es un quote, es decir, si un usuario ha publicado el tweet de una persona con un comentario añadido.
- Si el tweet ha sido truncado (esto se da porque originalmente los tweets eran de 140 caracteres y ahora son de 280, y así la API ofrece compatibilidad con versiones anteriores).
- El id del tweet original en caso de ser una respuesta.
- El id del usuario que publicó el tweet original en caso de ser una respuesta.
- El nombre del usuario que publicó el tweet original en caso de ser una respuesta.
- Las coordenadas desde la que se publicó el tweet.
- El lugar desde el que se publicó el tweet.
- El número de personas que han dado retweet a un tweet.
- El número de favoritos que tiene el usuario.
- El id del usuario que ha publicado el tweet.
- El nombre del usuario que ha publicado el tweet.
- La localización del usuario.
- La descripción del usuario.
- El número de seguidores del usuario.
- El número de cuentas a las que sigue el usuario.
- La fecha de creación de la cuenta del usuario.
- Si la cuenta está verificada.
- Un enlace a la imagen de perfil del usuario.
- Un enlace a la imagen de banner del usuario.
- User entities, que contiene datos adicionales del usuario, en caso de necesitar otros datos a posteriori.

Algunos de estos campos no devuelven la información esperada, como todos los casos de localización, que en muchas ocasiones devuelve valores NA.

Los datos se almacenan en formato CSV, para así tratarlos con Pandas en la parte de procesamiento de los datos. Gracias a toda la información obtenida mediante la explotación de la API de Twitter, se cuenta con mucha información sobre cada tweet, que también posibilita la creación de información adicional para un mejor desempeño de los modelos, pues podrán detectar nuevos patrones en los datos.

Para la realización de este proyecto, se han scrapeado 1.3 millones de tweets que contengan la palabra "ukraine", pues se parte de la hipótesis de que al ser un tema de gran importancia pública y sobre el cual puede haber múltiples intereses políticos, habrá una gran presencia de trolls.

Para este trabajo se va a utilizar un conjunto de datos formado por 50000 tweets, debido al alto coste computacional de utilizar el conjunto completo (unas ocho horas en un ordenador con procesador Intel Core i7-7700 CPU 3.60GHz, 32GB de RAM y tarjeta gráfica NVIDIA GeForce GTX 1060 6GB) y para mayor facilidad a la hora de mostrar resultados. Todos los datasets extraídos se encuentran en un Onedrive compartido para todos los miembros del equipo.

4.2 Procesamiento de los tweets

El objetivo de esta fase del proyecto es obtener nueva información agregada de la información obtenida mediante el scrapeador y limpiar el texto de los tweets para aplicar embeddings para la posterior aplicación de los modelos.

Mediante el cálculo de métricas adicionales, se intenta perfilar de forma más precisa cada tipo de usuario, con el fin último de que los modelos aplicados sean capaces de detectar las cuentas anómalas de una manera más precisa que sin esta información agregada. Se van a calcular tanto métricas relacionadas con los tweets, como métricas relacionadas con los usuarios. De esta manera, las métricas calculadas son las siguientes:

- Ratio favoritos entre seguidores: con esta métrica se intenta analizar el alcance que tienen los tweets de un usuario. Se puede ver con esta ratio el nivel de

relevancia de la cuenta. De esta manera, ratios cercanas a cero pueden implicar que ni seguidores ni otras personas interactúan con los tweets, ratios cercanas a uno, que más o menos los seguidores interactúan con los tweets de esta cuenta, y ratios mayores que uno implican que es una cuenta con gran relevancia, pues alcanza tanto a sus seguidores como a otras cuentas. La métrica estaría definida por la siguiente fórmula:

$$ratio\_favs\_followers = \frac{user.favourite\_count}{user.followers\_count}$$

Se debe tener en cuenta que un usuario puede ver visto su número de seguidores o favoritos modificados si se scrapea la misma cuenta en distintos días.

- Ratio retweets entre seguidores: con esta métrica se intenta analizar el nivel de relevancia del tweet en relación con la cuenta que lo publica. Se cumplen las mismas premisas que con la métrica anterior, si la ratio es cercana a cero, se considerará como un tweet poco relevante incluso entre seguidores de esa cuenta, ratios cercanas a uno implicará un tweet relevante entre los seguidores de la cuenta, y ratios mayores que uno, tweets relevantes en general, tanto para seguidores como para otros usuarios. La fórmula para calcular esta métrica sería la siguiente:

$$ratio\_retweets\_followers = \frac{retweet\_count}{user.followers\_count}$$

Esta métrica presenta uno de los problemas de la métrica anterior, y es que el número de seguidores de la cuenta puede variar si la misma cuenta está presente en los datos extraídos como resultado de distintos scrapeos.

- Número de emoticonos en el tweet: se identificará el número de emoticonos utilizado en cada tweet, para intentar identificar tweets sospechosos. Tweets que contengan demasiados emoticonos, pues se piensa que un usuario normal no suele añadir muchos emoticonos a sus tweets, sino que intercalan el texto del tweet con algún que otro emoticono.

- Comprobación sobre si la cuenta se creó previa o posteriormente a la guerra de Ucrania: con esta métrica, se intenta captar todas aquellas cuentas que se hayan creado con el fin de difundir propaganda o desinformación sobre este conflicto. Para ello, gracias a que la API de Twitter nos permite obtener la fecha de creación de la cuenta, se comparará esta fecha de creación con la fecha en la que se inició el conflicto armado, el 24 de febrero del 2022.
- Número de tweets del usuario: con esta métrica se intenta determinar si una cuenta es muy activa en la red social o no a partir del número de tweets que publica. Esta métrica se combinará con otra métrica para determinar cuánto de activa ha estado esa cuenta en el tiempo. Para calcular esta métrica se contarán cuantas entradas en el dataframe tiene un id de usuario concreto.
- Comprobación de si un tweet tiene menciones en el texto o no: se comprobará si un usuario en un tweet menciona a otros usuarios o no, con el fin de saber si ese tweet está dirigido a alguien o no.
- Número de menciones de cada usuario: con esta métrica se mostrará a cuantos usuarios ha mencionado una cuenta en sus tweets. Una mención en un tweet es de la forma "@nombreDelUsuario".
- Fecha más temprana y tardía de publicación: estas métricas obtendrán las fechas de publicación más recientes y menos recientes de un usuario dentro del conjunto de datos. Se obtienen a partir del campo "tweet_created_at", facilitado por la API de Twitter. Con estas métricas, se calculará el número de días que han pasado entre estas dos publicaciones, para obtener nueva información.
- Intervalo de días entre publicaciones: con las dos métricas anteriores, se obtiene el intervalo de días entre la publicación más reciente y la menos reciente. Esto servirá para calcular el nivel de actividad de la cuenta en el tiempo, gracias a esta métrica y al número de tweets de cada usuario calculado previamente.
- Ratio del número de tweets entre el intervalo de días: esta métrica mostrará cuánto de activa ha sido una cuenta en un periodo de tiempo determinado. Esta ratio permite visualizar de forma sencilla posibles cuentas sospechosas. La fórmula para calcular esta métrica es la siguiente:

$$ratio\_tweets\_dias = \frac{num\_tweets}{days\_interval}$$

Por ejemplo, supongamos una cuenta que tiene 40 tweets en un periodo de dos días. La ratio sería 20. Esto significaría que esta cuenta, ha publicado 20 tweets cada día, es decir, un tweet cada hora y doce minutos. Esto puede implicar una cuenta bot, pues una persona normal no debería poder publicar un tweet casi cada hora, pues tendrá que ir a trabajar, ir al colegio, dormir…

- Idiomas de cada usuario: gracias a que Twitter identifica el idioma en el que se ha publicado el tweet según la documentación de la API con un algoritmo de aprendizaje, se pueden extraer de cada tweet de un usuario los idiomas en los que están publicados, obteniendo así los idiomas que ha empleado una cuenta en sus diversos tweets. Con ello, se obtendrá el número de idiomas en los que tuitea.
- Número de idiomas de cada usuario: gracias al número de idiomas, también se pueden detectar cuentas sospechosas. Según un estudio de Eurostat la oficina de estadística de la Unión Europea, el 35.4% solo habla la lengua materna, un 35.2% adicional habla además una lengua extranjera, un 21% adicional habla dos idiomas extranjeros y solo un 8.4% habla tres o más lenguas extranjeras. Esto hace pensar que, cuentas que hablen más de dos idiomas pueden ser sospechosas, y a más aumenten los idiomas en los que se publica, más sospechosa será una cuenta.
- Hashtags utilizados y número de ellos en el tweet: se obtienen dos métricas de este proceso, que hashtags han sido utilizados, y cuántos hashtags tienen los tweets. Tweets con muchos hashtags pueden ser sospechosos, pues son tweets con el objetivo de tener una rápida difusión, intentando que ese tweet aparezca a usuarios que buscan tweets con temas distintos.
- Comprobación de si un usuario tiene foto de perfil por defecto: mediante la API de Twitter, se puede obtener un enlace a la foto de perfil de cada usuario, y Twitter, en la documentación, muestra que enlace de foto de perfil tiene una cuenta con la imagen por defecto. Gracias a esto, se puede comprobar si se tiene una foto de perfil customizada. No tener una imagen de perfil se ha asociado mucho tiempo a cuentas troll, bots o no, aunque no es ninguna garantía, pues este tipo de cuentas ha evolucionado mucho desde su origen. Aun así, sigue habiendo casos de cuentas falsas no muy trabajadas que siguen cumpliendo la premisa de no tener imagen de usuario.
- Word embeddings: el objetivo de los embeddings es convertir un texto en números para que los modelos puedan aprender también del tweet. Debido a que a los

modelos no se le pueden pasar características de tipo texto, debemos transformar los tweets a vectores de números. Antes de aplicar los embeddings, para su mejor aplicación, se deben limpiar los tweets de caracteres No-Ascii y de hipervínculos, además de eliminar stopwords. Además, gracias a los embeddings, se calculará la similitud que tienen los tweets de cada usuario entre sí.

- Similitud coseno: gracias al empleo de los embeddings, se puede obtener una medida de lo parecidos que son dos textos en el espacio. Así, hay distintas funciones de cálculo de distancias para aplicar al contexto de medición de la similitud entre dos vectores, como la distancia de Jaccard, euclídea, de Manhattan o, la que se va a emplear, la distancia coseno. Con ello se obtendrá una lista de coeficientes de similitud entre pares de tweets.
- Media aritmética de la similitud coseno: con esta métrica, se obtiene la media de similitud entre todos los tweets, siendo 0 que no se parecen nada entre sí, y 1 que son iguales. Un usuario que tenga una media de similitud alta será sospechoso, pues esto implica que siempre suele publicar tweets sobre temas muy parecidos empleando palabras muy similares en todos los tweets. Con esto se podrá, por ejemplo, identificar usuarios que publiquen copypastas, es decir, un bloque de texto que se publique de forma habitual, incluso por más usuarios. Un ejemplo de un copypasta sería el típico mensaje de Whatsapp diciendo “Pasale este mensaje a diez contactos tuyos para librarte de un año de mala suerte”.
- Número de retweets para cada usuario en el dataframe: se va a calcular el número de retweets de cada usuario presentes en el conjunto de datos, para obtener una medida más acerca de la actividad de los usuarios.
- Análisis de sentimiento: sobre el texto del tweet previamente preprocesado, se utilizará un modelo de Sentiment Analysis para determinar por cada tweet si el contenido de éste es positivo o negativo. Para ello se utilizará el modelo “en_core_web_sm” de la librería Spacy, entrenado con textos escritos en Internet, como blogs, comentarios de usuarios o noticias. Se suele considerar que los trolls publican contenido negativo en las redes, para provocar respuestas emocionales de los usuarios, alcanzado así una mayor difusión. Así, se asignará una etiqueta numérica a los textos, siendo 1 un texto positivo y 0 un texto negativo.

Una vez se hayan calculado las métricas, se guardará en formato CSV el DataFrame obtenido.

ratio_tweets_dias	langs_tweeted	lang_number	num_hashtags	only_hashtags	is_image_default	is_rt	are_mentions_in_tweet
2.0	[en]	1	1	Zelensky	False	True	True
1.0	[en]	1	0		False	True	True
1.0	[en]	1	0		False	True	True
7.0	[en]	1	0		False	True	True

Figura 4.2: Ejemplo de algunas de las métricas calculadas sobre un dataset.

4.3 Aplicación de los modelos

Una vez se ha obtenido el DataFrame con todas las métricas, es el turno de definir los modelos a aplicar sobre ese conjunto de datos. Lo primero es seleccionar únicamente aquellos tweets que estén en inglés, pues se quiere que una característica presente en los modelos sean los textos transformados en embeddings, y los embeddings empleados están pensados para utilizarse en inglés. Lo segundo es escoger las columnas del DataFrame a utilizar, puesto que no utilizaremos ni toda la información descargada mediante la API de Twitter, ni todas las métricas, puesto que la implementación de estos modelos necesita que todas las características sean numéricas, no textos, y en caso de que haya mezcla entre números y listas de números, se tendrán que separar esas listas y añadir nuevas columnas al dataframe donde poder almacenar esa información presente en las listas. Además, algunas de las métricas calculadas previamente no estaban pensadas para ser utilizadas independientemente, sino para calcular otras métricas. Por ejemplo, la fecha de publicación más temprana y tardía tenía por objetivo calcular el intervalo de días de publicación, y éste, a su vez, tenía por objetivo calcular la ratio de tweets publicados por día.

Por ello, de todas las columnas del dataframe, se escogerán las siguientes:

- “embeddings”: almacena en forma de lista los embeddings del texto del tweet. Se tendrá que realizar una transformación de esa lista a columnas de dataframe, como se ha expuesto previamente.
- “retweet_count”: número de personas que han dado retweet a un tweet. Tipo entero.
- “user.followers_count”: el número de seguidores del usuario que publicó el tweet. Tipo entero.
- “user.friends_count”: el número de cuentas a las que sigue el usuario que publicó el tweet. Tipo entero.
- “user.favourites_count”: el número de favoritos que ha conseguido un usuario con sus tweets. Tipo entero.
- “ratio_favs_followers”: proporción entre el número de favoritos y seguidores de una cuenta. Tipo decimal.
- “ratio_retweets_followers”: proporción entre el número de retweets de un tweet y seguidores de una cuenta. Tipo decimal.
- “number_emojis”: número de emoticonos en cada tweet. Tipo entero.
- “created_after_war”: verifica si la cuenta fue creada antes o después del 24 de febrero de 2022. Tipo booleano.
- “num_tweets”: número de tweets que ha publicado la cuenta, presentes en el dataframe. Tipo entero.
- “ratio_tweets_dias”: proporción entre el número de tweets publicados por una cuenta y el intervalo de días presentes en el dataset. Tipo decimal.
- “lang_number”: número de idiomas que utiliza un usuario concreto. Tipo entero.
- “num_hahstags”: el número de hashtags que tiene el tweet. Tipo entero.
- “is_image_default”: si la cuenta que ha publicado el tweet tiene la imagen por defecto de Twitter. Tipo booleano.
- “is_rt”: si el tweet es un retweet. Tipo booleano.
- “are_mentions_in_tweet”: si en el tweet hay algún usuario mencionado. Tipo booleano.
- “num_mentions”: el número de menciones presentes en un tweet. Tipo entero.
- “num_mentions_user”: el número de menciones presentes en todos los tweets de un usuario. Tipo entero.

- “cosine_avg”: la media de la similitud de todos los tweets de un usuario utilizando la función de similitud coseno. Tipo decimal.
- “num_rts”: número de retweets de cada usuario. Tipo entero.
- “sentiment”: tono emocional subyacente en el tweet. 0 si positivo, 1 si negativo. Tipo booleano.

Una vez seleccionados los atributos del tweet con los que aplicar los modelos, toca transformar la columna de embeddings. Para ello, se creará un dataframe nuevo, en el que cada columna de ese dataframe sea un embedding, se concatena este nuevo dataframe con el viejo, y se elimina la columna que contenía los embeddings en forma de lista.

En un primer enfoque, se ha apostado por combinar estadísticas de los tweets y de los usuarios en un mismo dataframe, pero también se podrían aplicar los modelos con un dataframe que sólo contenga información sobre los tweets o con un dataframe que solamente contenga información sobre los usuarios. El objetivo de esta aproximación es que los modelos puedan identificar tanto anomalías en los contenidos publicados, como en los usuarios que publican los contenidos. También, se normalizarán los datos mediante el uso de MinMaxScaler de Scikit-learn.

Para los modelos de clustering, se utilizará un dataframe normalizado con todas las métricas mencionadas anteriormente, pero sin los embeddings, puesto que la alta dimensionalidad de los datos puede afectar al desempeño de los modelos, dificultando al modelo la distinción de características importantes para la agrupación de los datos. Para los modelos de detección de anomalías, no obstante, se utilizará un dataframe normalizado con los embeddings, puesto que se quiere detectar anomalías no solo por estadísticas del tweet o del usuario, sino también por el contenido del tweet.

El primer modelo para probar es DBSCAN, un método de clustering. Pese a que en todos los modelos residirá una dificultad considerable en la elección de valores para hiperparámetros, en DBSCAN es particularmente difícil. En este caso, se debe elegir un parámetro épsilon, que decide como de cercanos deben ser los puntos para ser considerados como parte de un cluster, y un parámetro minPoints, que decide el número de puntos necesarios para formar un cluster o región densa. Para intentar escoger los mejores hiperparámetros, se utiliza un gráfico de la distancia k en caso de épsilon, y en caso de minPoints, se pondrá de valor 50, para que cada agrupación conste al menos de

50 datos. Con estos hiperparámetros, DBSCAN obtiene muy malos resultados, agrupando los datos en un único cluster, pero detectando 4930 puntos anómalos.

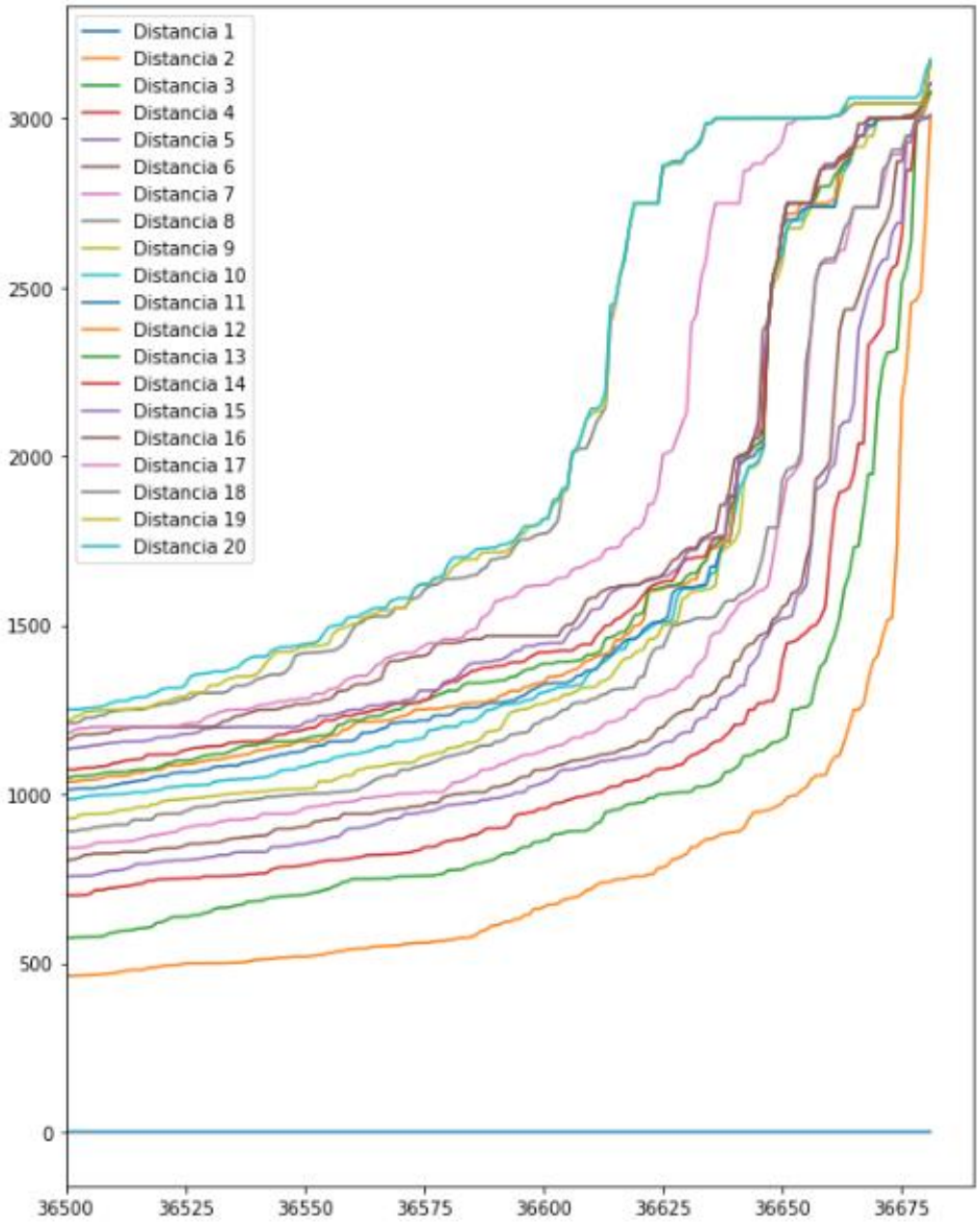

Figura 4.3: gráfica de las distancias k para elegir el valor de épsilon. Se deben escoger valores presentes en la curva de inflexión.

Con ello, se obtienen 21 agrupaciones y 823 datos anómalos.

```
Estimated number of clusters: 21
Estimated number of noise points: 823
```

Figura 4.4: resultados de aplicar DBSCAN con eps = 700 y min_samples = 50

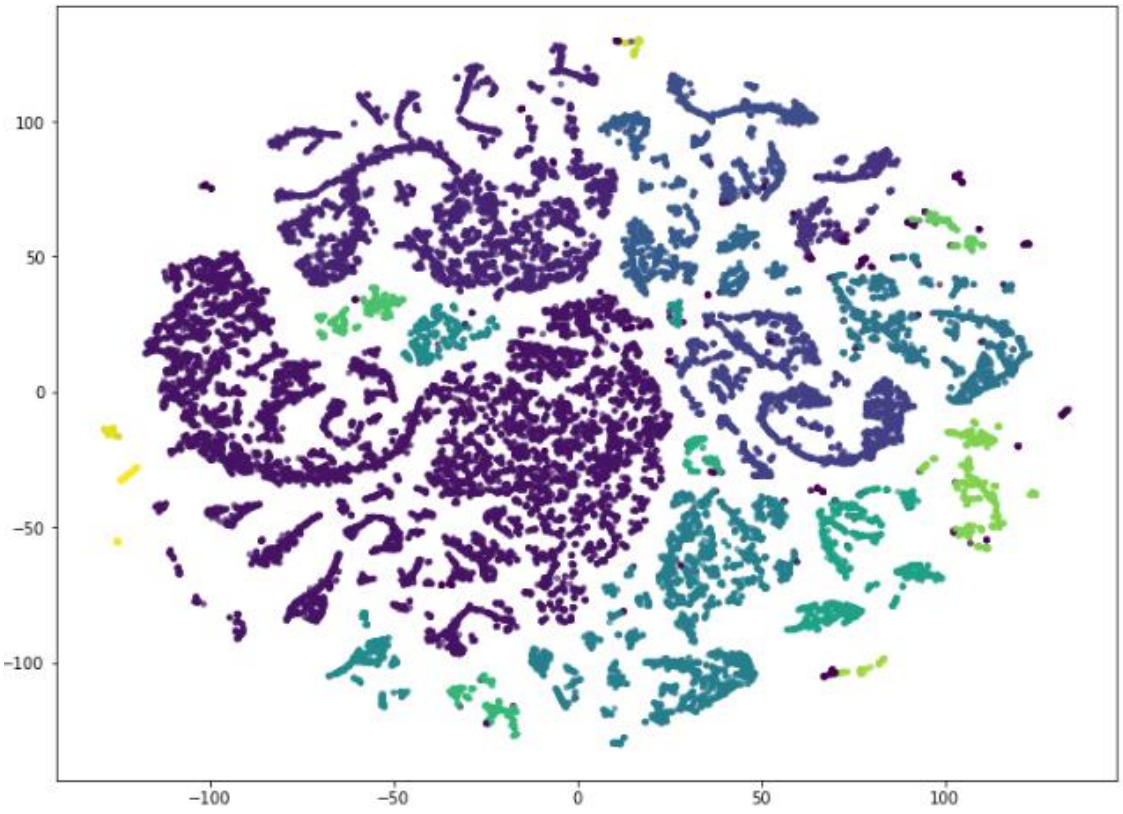

Figura 4.5: resultados de representar los datos en función de la agrupación a la que pertenecen mediante el color utilizando la técnica t-SNE

El siguiente modelo para probar es Isolation Forest. Los hiperparámetros serán los por defecto de Scikit-Learn a excepción del factor de contaminación. El factor de contaminación es la proporción de datos anómalos que se espera haya en el dataset. Se probarán el hiperparámetro por defecto, y el hiperparámetro con un valor de 0.15, pues un estudio de la universidad de Indiana determina que un 15% de los usuarios presentes en la red social son bots (Varol, O., et al., 2017, Online Human-Bot Interactions: Detection, Estimation, and Characterization). La primera prueba determina que solo hay 121 puntos anómalos, y la segunda prueba determina que 5503 puntos son anómalos. Los resultados que se guardarán son los de la segunda prueba, pues en los siguientes modelos se va a aplicar el mismo factor de contaminación.

```
Numero de anomalias segun Isolation Forest: 64
Numero de no anomalias segun Isolation Forest: 36618
```

Figura 4.6: resultados de aplicar Isolation Forest con los hiperparámetros por defecto incluyendo el factor de contaminación.

```
Numero de anomalias segun Isolation Forest: 5503
Numero de no anomalias segun Isolation Forest: 31179
```

Figura 4.7: resultados de aplicar Isolation Forest con los hiperparámetros por defecto, excepto el factor de contaminación, siendo este contamination = 0.15.

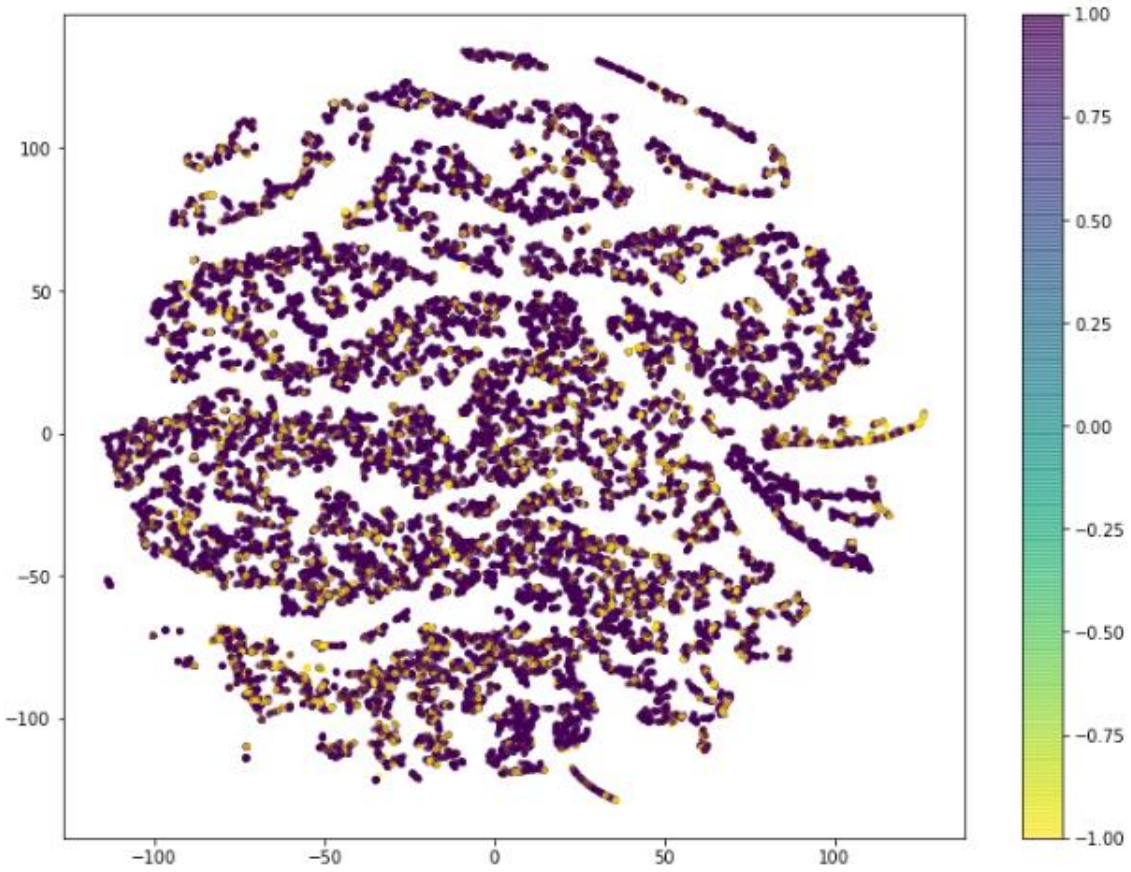

Figura 4.8: resultados de representar los datos obtenidos con Isolation Forest, teniendo una contaminación de 0.15, según si es o no anomalía utilizando t-SNE (amarillo = anomalía, morado = no anomalía)

Autoencoders para detección de anomalías es el siguiente modelo. Como se mencionó previamente, se va a partir de un factor de contaminación del 0.15. Así, el modelo de Autoencoder detecta 5503 puntos anómalos.

```
Numero de anomalias con Autoencoders: 5503
Numero de no anomalias con Autoencoders: 31179
```

Figura 4.9: resultados de aplicar Autoencoders, con el hiperparámetro contamination = 0.15.

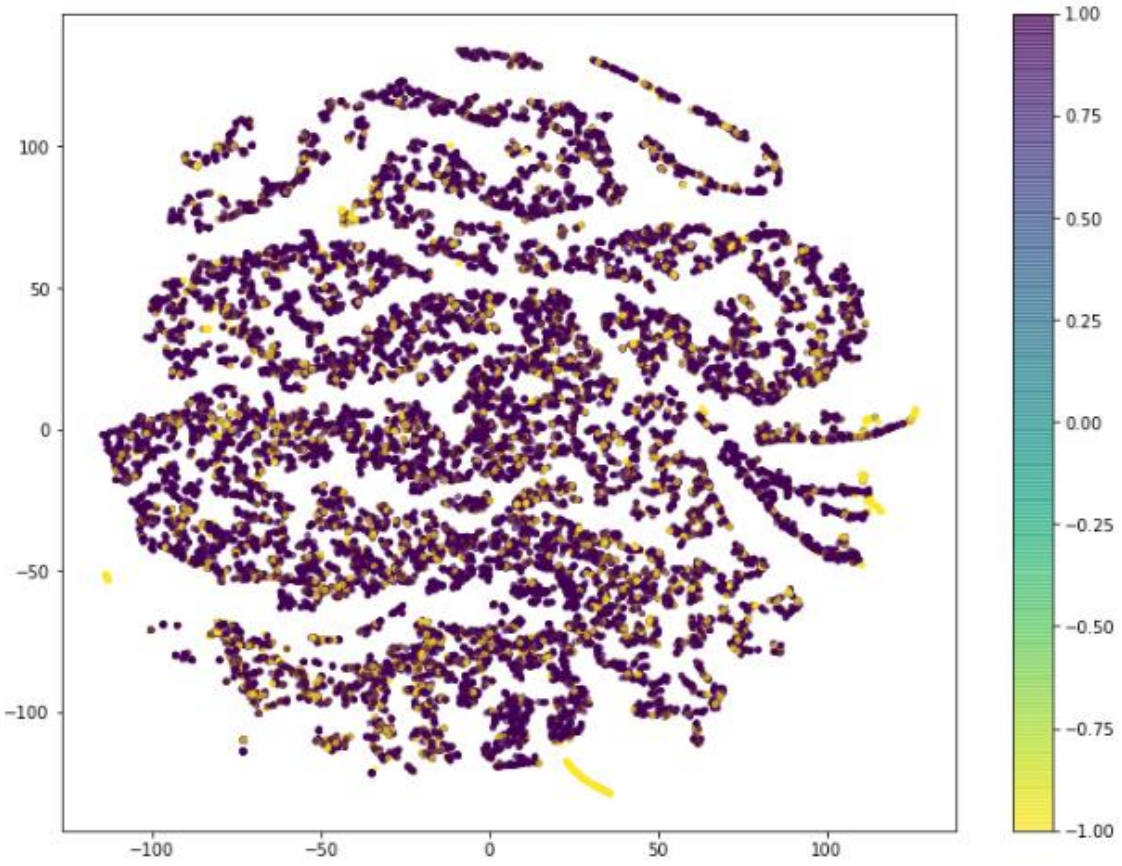

Figura 4.10: resultados de representar los datos obtenidos con Autoencoders, teniendo una contaminación de 0.15, según si es o no anomalía utilizando t-SNE (amarillo = anomalía, morado = no anomalía)

En One-Class SVM hay que elegir tres hiperparámetros, el kernel, el factor de contaminación y nu. Para el kernel, se escogerá rbf (radial basis function), el cual puede ser idóneo para datos no lineales como puede ser un dataframe.

```
Numero de anomalias con SVM: 10282
Numero de no anomalias con SVM: 26400
```

Figura 4.11: resultados de aplicar One-Class SVM con los hiperparámetros kernel = ‘rbf’, gamma = 0.05 y nu = 0.03.

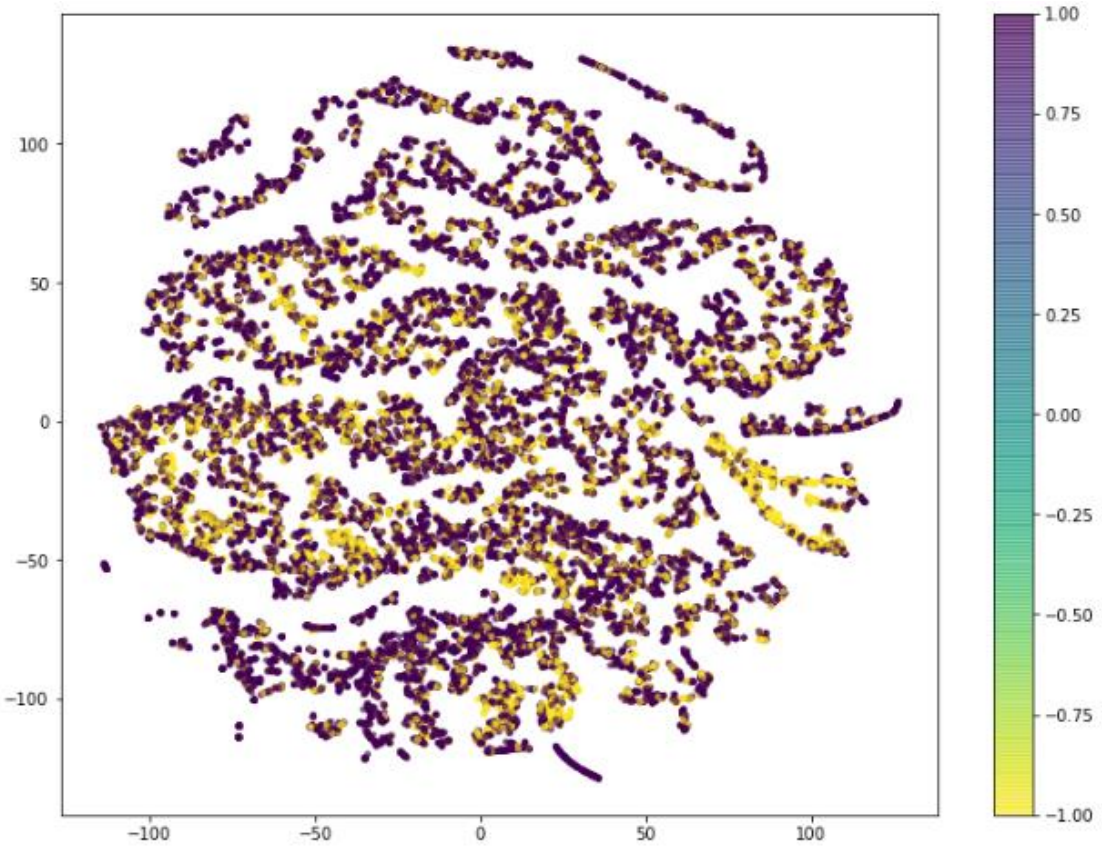

Figura 4.12: resultados de representar los datos obtenidos con One-Class SVM, teniendo una contaminación de 0.15, según si es o no anomalía utilizando t-SNE (amarillo = anomalía, morado = no anomalía)

Por último, el modelo Local Outlier Factor. En este modelo, el hiperparámetro n_neighbours tendrá el valor por defecto, pues un valor pequeño de este parámetro permite mayor detección de outliers. N_neighbours determina el número de puntos vecinos con los que se debe comparar el punto para calcular el LOF score, que es el que indica si un punto es o no un outlier.

```
Numero de anomalias con Local Outlier Factor: 5503
Numero de no anomalias con Local Outlier Factor: 31179
```

Figura 4.13: resultados de aplicar Local Outlier Factor con los hiperparámetros n_neighbours = 20 (por defecto) y contamination = 0.15.

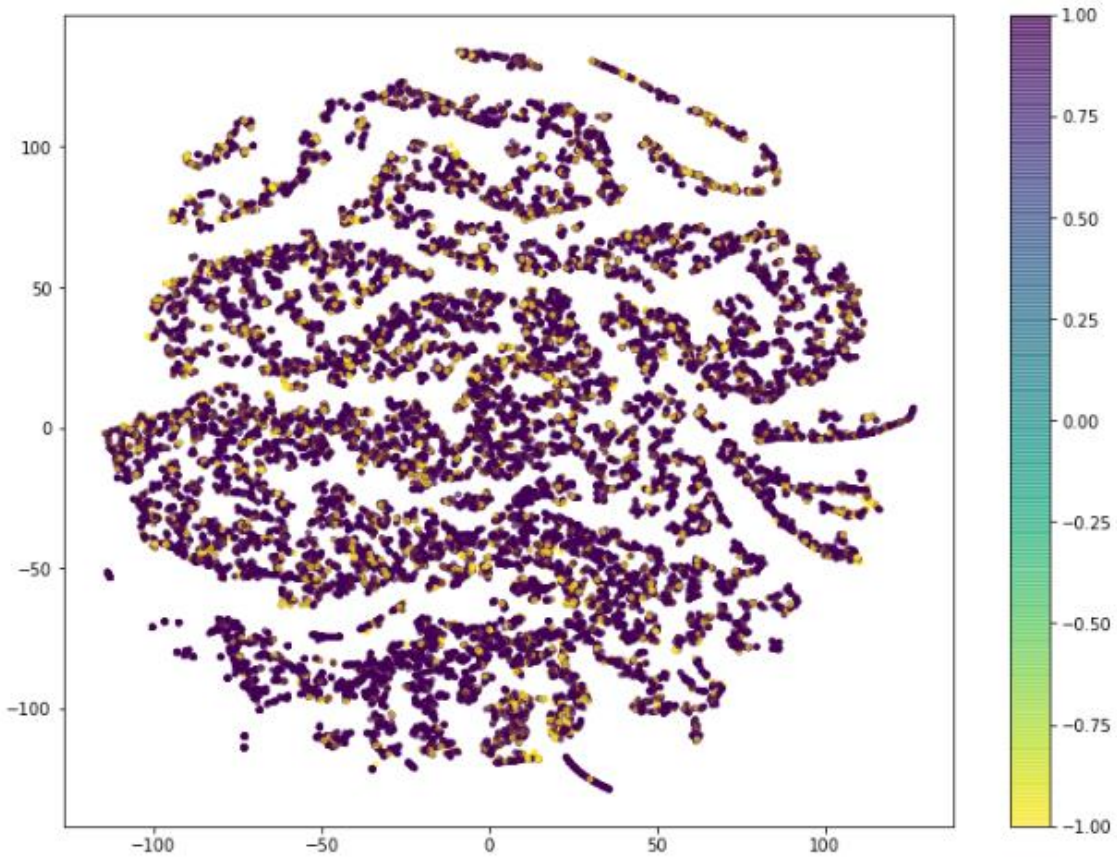

Figura 4.14: resultados de representar los datos obtenidos con Local Outlier Factor, teniendo una contaminación de 0.15 y n_neighbourse = 20, según si es o no anomalía utilizando t-SNE (amarillo = anomalía, morado = no anomalía)

Tras aplicar todos los modelos se añadirán al dataframe las predicciones de Isolation Forest, Autoencoders y Local Outlier Factor sobre cada fila de datos, pudiendo ver así qué modelos coinciden en sus predicciones y cuáles no, y analizando por qué lo marca como anomalía o no. Se ha descartado One Class SVM por la diferencia en la detección de anomalías respecto a los otros tres modelos. Además, se analizarán por separado los resultados de DBSCAN.

A su vez, se creará un dataframe por cada modelo, que solo contengan aquellos datos catalogados como anómalos por los modelos para su evaluación.

cosine_avg	num_rts	sentiment\r	anomaliesAE	anomaliesIF	anomaliesLOF	anomaliesSVM
0.305142	2	0	0	1	1	-1
0.000000	1	1	1	-1	1	1
0.000000	1	1	0	1	1	-1
0.436056	7	0	0	1	1	-1
0.000000	1	0	0	1	1	-1

Figura 4.15: dataframe del conjunto de datos con columnas de predicción añadidas.

4.4 Resultados y discusión

Una vez que los modelos han realizado las predicciones, éstas deben de ser analizadas para comprobar el desempeño de los modelos, puesto que, al ser aprendizaje no supervisado, no se pueden utilizar métricas propias del aprendizaje supervisado como el accuracy, recall o AUROC.

4.4.1 DBSCAN

DBSCAN ha sido capaz de agrupar los datos en veintiún agrupaciones más una agrupación de anomalías. Para intentar comprender el porqué de las agrupaciones, se va a calcular la media y la desviación típica de las principales métricas de tipo numérico. El objetivo de esto es poder distinguir entre clusters de usuarios normales y trolls, y si hay múltiples clusters de trolls, identificar qué es lo que los distingue de otros grupos.

Por ejemplo, uno de los grupos detectados por DBSCAN está caracterizado por usuarios que tuitean demasiado al día, 93 tweets al día de media, que mencionan a muchos usuarios en sus tweets (cada usuario menciona a otros 93 de media) y cada autor parece publicar tweets algo similares. Así, esto puede indicar que esta agrupación está detectando trolls. Cabe destacar también la diferencia de la ratio de favoritos por seguidor y la ratio de retweets por seguidor, 204 y 0.28 respectivamente. Este comportamiento es algo anómalo, pues no debería existir una diferencia tan significativa entre dos métricas

parecidas, pues ambas analizan el engagement que logra tener un usuario, es decir, la comunidad que crea.

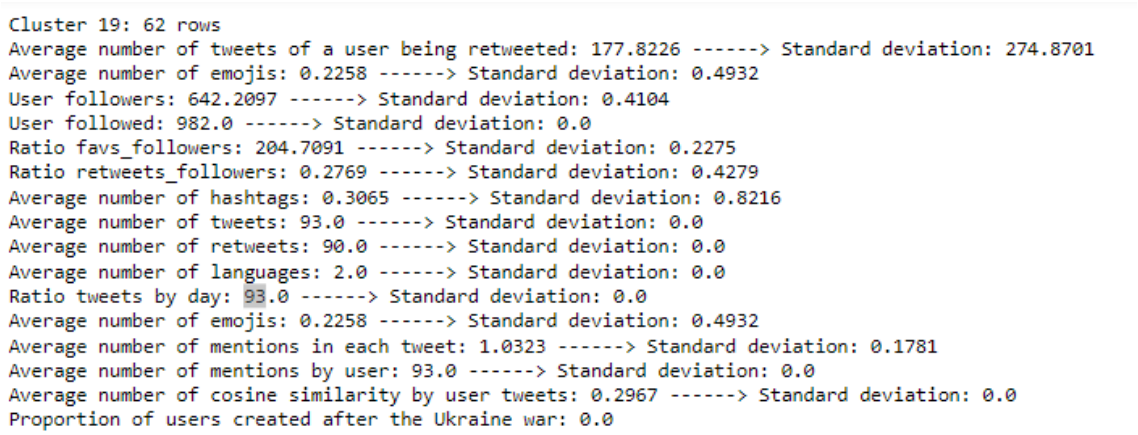

Figura 4.16: media y desviación típica de las métricas de una de las agrupaciones obtenidas con DBSCAN.

Otro ejemplo sería la agrupación caracterizada por tener una media de 0 favoritos, pero una media de retweet superior. Esto es raro, pues normalmente los tweets suelen tener más favoritos que retweets. Además, se ve que es un tipo de usuario muy activo con muchos tweets y la similitud entre tweets del mismo autor es muy elevada, de casi el 60%. Viendo esto, parece que este tipo de usuarios se dedican a propagar cadenas de texto (copypastas) por la red social.

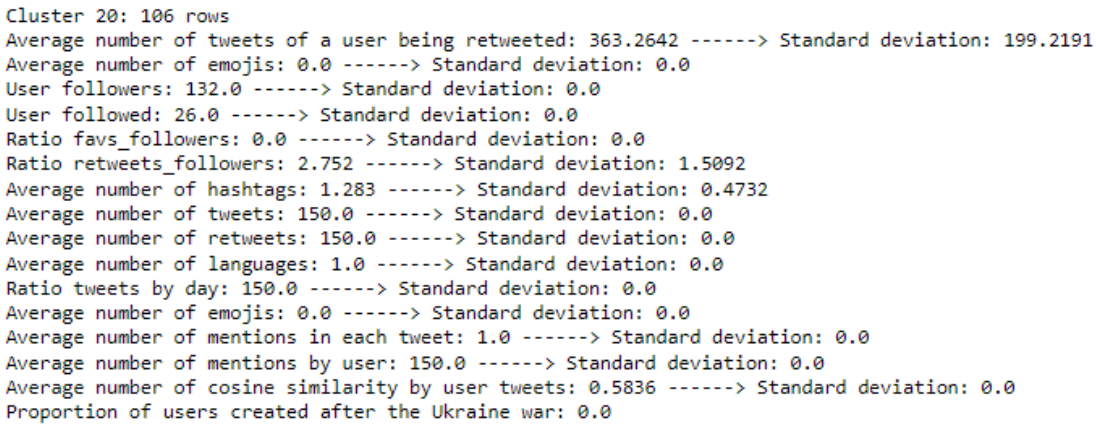

Figura 4.17: media y desviación típica de las métricas de una de las agrupaciones obtenidas con DBSCAN.

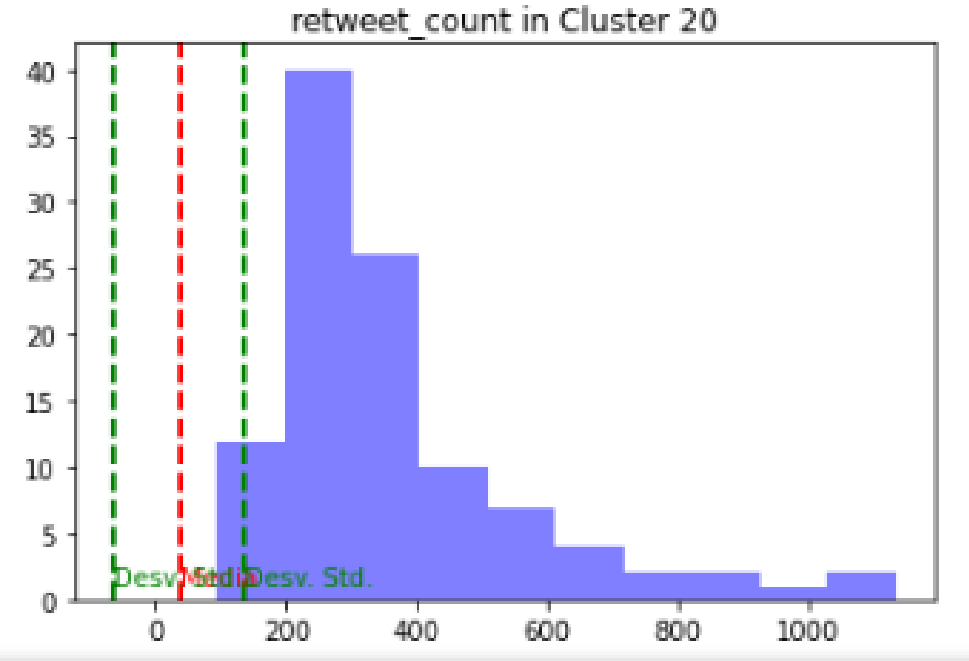

Figura 4.18: gráfico que muestra la distribución del número de seguidores de una de las agrupaciones obtenidas con DBSCAN

4.4.2 AutoEncoders

Con AutoEncoders, se ha detectado un usuario con un gran número de seguidores, pero pequeño de usuarios a los que sigue, cuyos tweets tienen un gran impacto, pues tienen una ratio muy alta de favoritos por seguidor, y una ratio considerable de retweets por seguidor, aunque mucho menor. Además, son usuarios que no suelen mencionar a otros usuarios en sus tweets, y que tienen una proporción de tweets al día alta pero razonable. Esta es una de las cosas que han permitido que los trolls proliferen, pues cada vez se adaptan mejor al esquema de un usuario real, dificultando así la diferenciación entre trolls y no trolls. Además, la proporción de usuarios creados tras la guerra de Ucrania es considerable, algo más de un 25% de estas cuentas son posteriores. Se puede apreciar también el bajo uso de hashtags, que se aproxima a cero hashtags de media en cada tweet.

```
Average number of retweets: 436.7385 ------> Standard deviation: 1858.5364
Average number of languages: 1.1307 ------> Standard deviation: 0.8083
User followers: 74495.0 ------> Standard deviation: 1201380.6388
User followed: 2719.342 ------> Standard deviation: 14063.7511
Ratio favs_followers: 262.5206 ------> Standard deviation: 2509.7836
Ratio retweets_followers: 44.4943 ------> Standard deviation: 412.7968
Average number of hashtags: 0.3524 ------> Standard deviation: 1.0538
Average number of tweets: 9.5341 ------> Standard deviation: 27.0061
Ratio tweets by day: 9.5341 ------> Standard deviation: 27.0061
Average number of emojis: 0.3147 ------> Standard deviation: 1.2765
Average number of mentions in each tweet: 0.9388 ------> Standard deviation: 0.9933
Average number of mentions by user: 10.5154 ------> Standard deviation: 33.0812
Average number of cosine similarity by user tweets: 0.1699 ------> Standard deviation: 0.2339
Proportion of users created after the Ukraine war: 0.2533
```

Figura 4.19: media y desviación típica de las métricas de un dataframe que solo contiene aquellos datos etiquetados como anómalos por Autoencoder.

Para comprender mejor porqué este modelo ha detectado estos tweets como anómalos, se va a comprobar la distribución de datos por cada métrica del dataframe con las anomalías, comparando los datos con la media y desviación típica de esa misma métrica con el dataframe que contiene todos los datos.

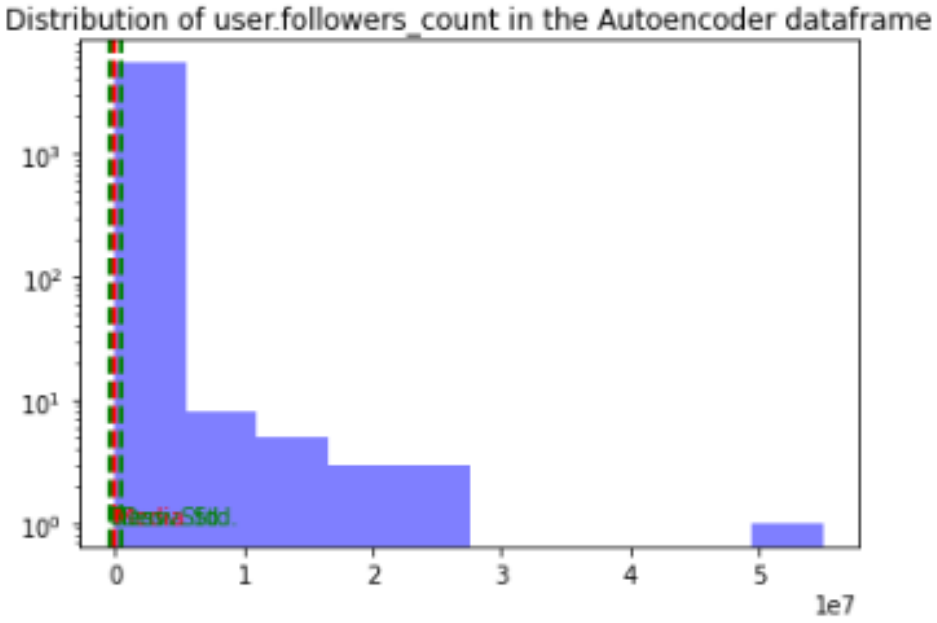

Figura 4.20: comparativa entre la distribución de datos de la métrica de seguidores en el dataframe de Autoencoder con la media y desviación típica de los seguidores en el dataframe general en escala logarítmica en base 10.

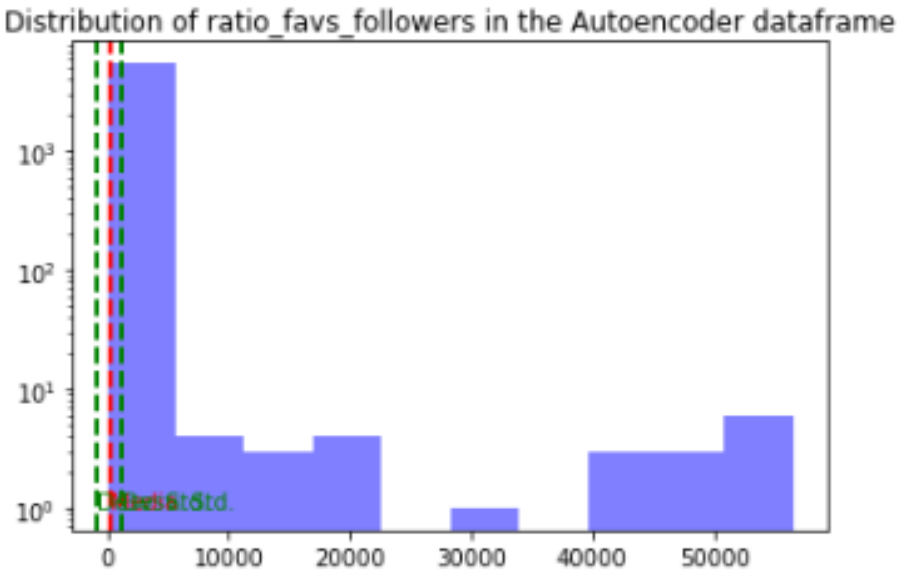

Figura 4.21: comparativa entre la distribución de datos de la métrica de retweets por seguidor en el dataframe de Autoencoder con la media y desviación típica de los seguidores en el dataframe general en escala logarítmica en base 10.

También se pueden mirar los tweets de las cuentas que el modelo Autoencoder ha catalogado como anómalas.

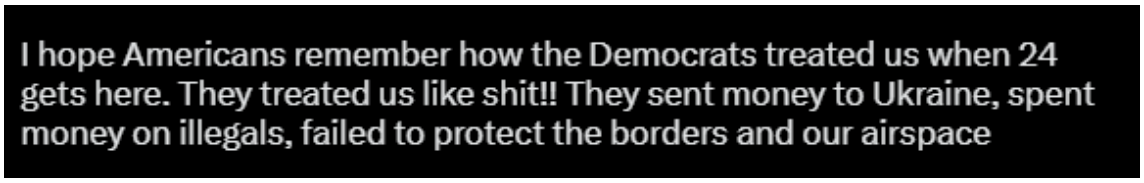

Figura 4.22: ejemplo de tweet identificado mediante Autoencoders.

4.4.3 Isolation Forest

El modelo de Isolation Forest ha catalogado como anomalías un tipo de usuario muy similar al que detecta Autoencoders. Así, se tiene un tipo de usuario con muchos seguidores, pero pocos usuarios seguidos, con unas ratios de favoritos y retweets elevadas.

```
Average number of retweets: 383.3278 ------> Standard deviation: 1223.2133
Average number of languages: 1.1072 ------> Standard deviation: 0.4609
User followers: 49985.7572 ------> Standard deviation: 1011762.435
User followed: 2135.1277 ------> Standard deviation: 8752.4717
Ratio favs_followers: 153.7932 ------> Standard deviation: 1035.4301
Ratio retweets_followers: 12.04 ------> Standard deviation: 137.3932
Average number of hashtags: 0.2831 ------> Standard deviation: 0.9123
Average number of tweets: 4.8437 ------> Standard deviation: 11.3779
Ratio tweets by day: 4.8437 ------> Standard deviation: 11.3779
Average number of emojis: 0.2368 ------> Standard deviation: 0.9044
Average number of mentions in each tweet: 0.8653 ------> Standard deviation: 0.7151
Average number of mentions by user: 4.4309 ------> Standard deviation: 11.9465
Average number of cosine similarity by user tweets: 0.1562 ------> Standard deviation: 0.2158
Proportion of users created after the Ukraine war: 0.2708
```

Figura 4.23: media y desviación típica de las métricas de un dataframe que solo contiene aquellos datos etiquetados como anómalos por Isolation Forest.

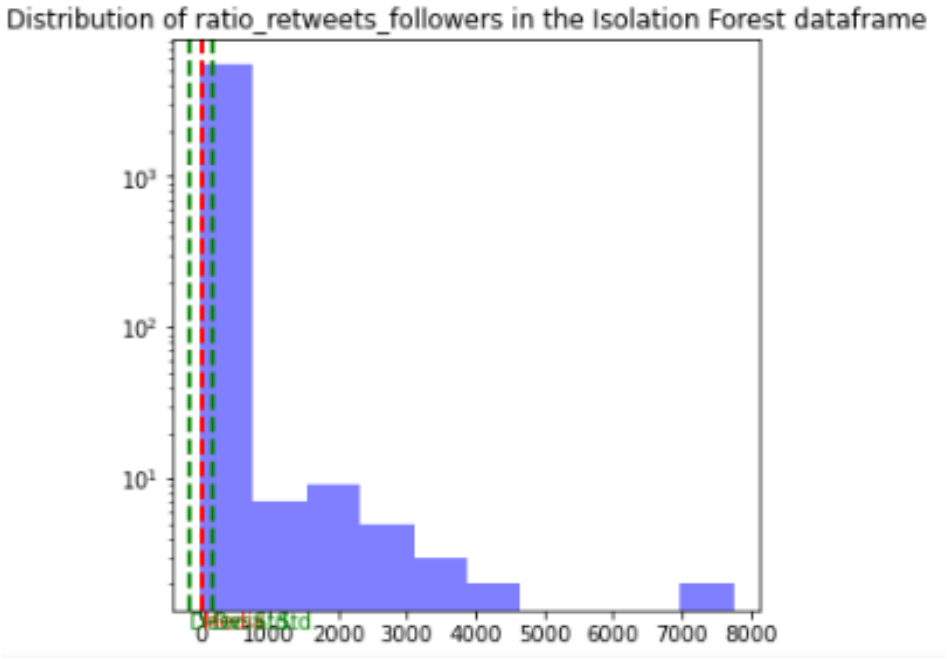

Figura 4.24: comparativa entre la distribución de datos de la métrica de retweets por seguidor en el dataframe de Isolation Forest con la media y desviación típica de los seguidores en el dataframe general en escala logarítmica en base 10.

Algunos de los tweets detectados por el modelo de Isolation Forest son también detectados por el modelo de Autoencoders. Un ejemplo de un tweet detectado por ambos puede ser el siguiente: “bbc lied about 911 bbc lied about iraq war bbc lied about libya war bbc lied about syria bbc lied about isis”

4.4.4 Local Outlier Factor

El modelo Local Outlier Factor, de nuevo, detecta un tipo de usuario parecido a lo que detectan los anteriores.

```
Average number of retweets: 622.3649 ------> Standard deviation: 1556.9584
User followers: 9154.3794 ------> Standard deviation: 228196.8589
User followed: 2074.1325 ------> Standard deviation: 6422.7058
Ratio favs_followers: 323.5415 ------> Standard deviation: 1976.9648
Ratio retweets_followers: 46.2249 ------> Standard deviation: 323.8446
Average number of hashtags: 0.2868 ------> Standard deviation: 0.8191
Average number of tweets: 6.0336 ------> Standard deviation: 12.5679
Ratio tweets by day: 6.0336 ------> Standard deviation: 12.5679
Average number of emojis: 0.2153 ------> Standard deviation: 0.7511
Average number of mentions in each tweet: 1.1599 ------> Standard deviation: 0.5648
Average number of mentions by user: 6.7456 ------> Standard deviation: 13.7406
Average number of cosine similarity by user tweets: 0.1737 ------> Standard deviation: 0.182
Proportion of users created after the Ukraine war: 0.2958
```

Figura 4.25: media y desviación típica de las métricas de un dataframe que solo contiene aquellos datos etiquetados como anómalos por Local Outlier Factor.

Local Outlier Factor a primera vista parece no detectar los mismos tweets que Autoencoders e Isolation Forest. Un tweet de ejemplo que ha detectado el modelo LOF es: “No Nazis in Ukraine they say? For those blind and deaf, here's yet another piece of evidence: On Valentine's day Zelensk…”

4.4.5 Resultados en común de los modelos

Al haber introducido las predicciones en el dataframe de datos, se va a considerar que para que un usuario sí sea una anomalía, al menos dos de los tres modelos deberán coincidir en la predicción (si solo un modelo de los tres considera un dato anomalía, sería una seguridad del 1/3 == 33%, si dos de tres, una seguridad del 2/3 = 66% y 3/3, una seguridad del 3/3 = 100%, según los modelos utilizados).

Así, se obtiene un conjunto de 4402 datos de los 50000 datos iniciales. Algunos de los tweets que han considerado anómalos son:

- “how about giving ohio some of those ukraine dollars ohio is fighting for its life”

- “bbc lied about 911 bbc lied about iraq war bbc lied about libya war bbc lied about syria bbc lied about isis”
- “check out medias dailys video tiktok ukraine is corrupt”
- “this is coming if you dont make a noise now get the fuck out of ukraine and taiwan its none of our god da…”
- “good russians are they good a perspective from kyiv in sho there are good good russians these are those who are fig…”
- “the number of divorces has increased sharply in russia during the past year according to official statistics while in…”
- “is the american dream dead the national park service is beginning to clear the homeless encampment at nohwest dcs mc…”
- “i would agree if you acknowledged the fact that dont paid hush money to porn stars stole from his o…”
- “us signals exit from ukraine on the horizon”

Como se puede ver en estos ejemplos, ha logrado detectar tweets bastante ofensivos (por ejemplo, el que se acusa a la BBC de haber mentido en temas muy complicados), tweets que no tienen que ver con el tema principal que se está tratando (por ejemplo, el tweet donde se dice que han incrementado los divorcios en Rusia) o tweets de información falsa (por ejemplo, en el que se insta a los EE.UU a salir de Ucrania).

También se pueden mirar aquellos tweets en los cuales todos los modelos están de acuerdo en que son anómalos. Si del conjunto de datos se obtienen únicamente aquellos que cumplan la condición de haber sido detectados como anomalías, se tiene un conjunto de datos nuevo formado sólo por 464 tweets. Es decir, solo un 10% de los datos contenidos en el dataframe en el que deben coincidir mínimo dos predicciones.

- “the germans are sending 88 tanks to ukraine seriously not 87 or 89 are they trolling”
- “parents worried about their child s pronouns in parents worried about the paycheck their child is receiving to make tik…”
- “i d like a refund of my federal taxes for the last 2 years that i have paid into for an incompetent biden administration”

- “so many phonies in a row lately at drowning street all desperately fighting to gain some popularity at home by…”
- “nato meanwhile you leave canada defenseless canadians going hungry homeless health care broken c…”
- “militants with isis patches have been spotted in the ranks of the armed forces of ukraine are they bringing th”

Viendo esto, parece que sí está detectando tweets troll, pero también a primera vista parece detectar tweets normales. Esto puede ser debido a que, aunque el texto del tweet sea normal, las métricas detrás del tweet o del usuario no sean normales. Por ello, no se puede descartar tampoco que muchos de los tweets que detectan los modelos como anomalías realmente sean anomalías, a pesar de que a primera vista no lo parezcan.

Podemos concluir que nuestros modelos sí están funcionando, aunque puede que no con el propósito inicial, que era únicamente detección de trolls que propagasen desinformación. En su lugar, parece haberse desarrollado una herramienta capaz de detectar anomalías generales, ya sean por propagar desinformación, por publicar contenidos que no tienen relación con el tema que se trata, o por tener unas métricas de usuario o tweet poco habituales.

Modelo	Nº de puntos anómalos	Nº de puntos no anómalos
Isolation Forest	5503	31179
Autoencoders	5503	31179
Local Outlier Factor	5503	31179
One-Class SVM	10282	26400

Figura 4.26: tabla con los resultados de la aplicación de los modelos de detección de anomalías. Estos resultados son muy dependientes de los hiperparámetros escogidos.

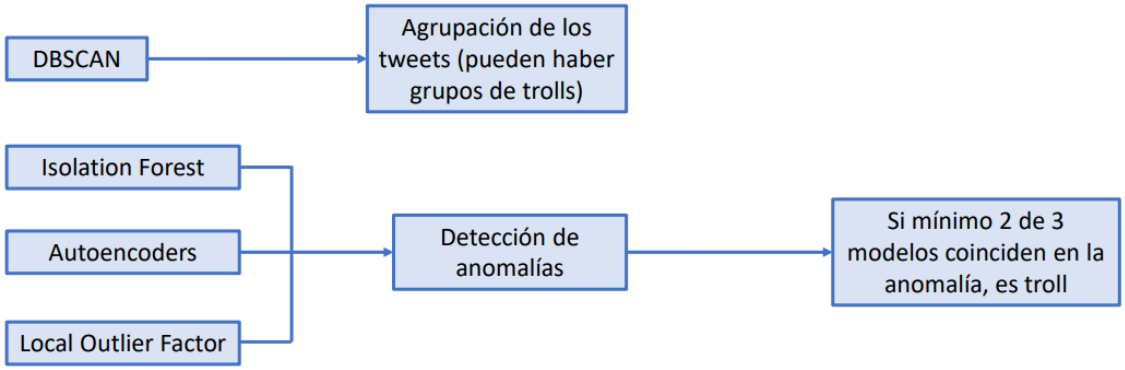

Figura 4.27: esquema de la aplicación de los modelos y sus resultados.

Métrica	Descripción
Número de retweets	Cuántos retweets tiene un tweet
Número de seguidores	Cuántos usuarios siguen a la cuenta
Número de seguidos	A cuántos usuarios sigue la cuenta
Ratio de favoritos entre seguidores	Proporción entre el número de favoritos que tienen todos los tweets de una cuenta y su número de seguidores
Ratio de retweets entre seguidores	Proporción entre los retweets que tiene un tweet de una cuenta y su número de seguidores
Número de hashtags utilizados	Cantidad de hashtags (temas) que contiene un tweet
Número de tweets por día	Cuántos tweets ha publicado una cuenta de media por día
Similitud coseno de los tweets	Dentro de los tweets publicados por una cuenta, cuánto de similares son entre ellos de media

Figura 4.28: tabla con las métricas más relevantes de los usuarios y su descripción

Como se puede apreciar en la tabla de resultados de los modelos, la detección de anomalías depende mucho de la selección de hiperparámetros, por lo que una posible línea de investigación sería combinar aprendizaje supervisado y no supervisado, de tal

manera que, las anomalías detectadas por los modelos no supervisados deberán ser etiquetados para posteriormente utilizar modelos supervisados. De esta manera, se evita tener que etiquetar el conjunto de datos inicial, y solo se etiquetarán pequeños subconjuntos, aprovechando aún la mayor ventaja del aprendizaje no supervisado, que es no tener que etiquetar grandes conjuntos de datos. Con la combinación de ambas técnicas, se podría obtener una herramienta más precisa a la hora de detectar cuentas troll.

5. CONCLUSIONES

Este Trabajo de Fin de Grado ha consistido en el desarrollo de un método para la detección de trolls en Twitter mediante aprendizaje no supervisado, pues es de vital importancia poder detectar este tipo de cuentas en el contexto actual, en el cual mucha gente recurre a redes sociales para informarse, y cada vez más, estas redes sociales contienen más desinformación y pueden ser usadas para alterar la opinión pública en temas complejos y relevantes.

Durante la realización de éste, se han encontrado diversos problemas. Así, en la primera etapa del proyecto se encontró el mayor número de problemas, siendo éstos la necesidad de una cuenta de Twitter con permisos de Academic Research en la API, pues las cuentas básicas pueden realizar muy pocas extracciones de tweets, y el anuncio sorpresa en febrero de que la API sería de pago, lo cual obligó al equipo a desarrollar un scraper y obtener datos lo más rápido posible, aunque al final aún se puede utilizar la API sin pagar. Además, se cambió el scraper para recolectar más información, por lo que hubo que recolectar nuevos datos desde cero. Uno de los mayores problemas se encontró en la fase de obtención de métricas, puesto que información que podría ser muy relevante, como la ubicación del usuario que publica un tweet, no ha sido posible obtenerla. Otro gran problema se encuentra en la elección de hiperparámetros de los modelos, pues es un factor que determina el aprendizaje de los modelos, y cuyos valores no puede ser determinado a partir de los datos, complicando así mucho su elección. Además, se debe tener en cuenta el problema de recursos computacionales, debido a los cuales no se ha utilizado el conjunto de datos entero (1.6 millones de datos). Esta gran cantidad de datos también deriva en un problema a la hora de extraer conclusiones de debido al alto volumen de datos a revisar.

Este trabajo ha logrado detectar trolls y otro tipo de anomalías, gracias al uso conjunto de información sobre el tweet y sobre el usuario que ha publicado ese tweet, pero, debido a que es aprendizaje no supervisado, no hay una medida de la efectividad de estos modelos. Debido a que no hay una manera precisa de saber lo buenos que son estos modelos, creo que este proyecto no debería ser usado para determinar de forma precisa qué cuentas son troll, sino más bien para marcar tweets como sospechosos a los usuarios, sirviendo así la herramienta desarrollada para advertir acerca de contenido posiblemente troll.

Por lo descrito anteriormente, se abre una línea de trabajo nueva sobre lo aquí desarrollado, que sería la implementación de sistemas de aprendizaje supervisado. Así, un conjunto de personas se dedicaría a etiquetar los datos marcados como anómalos por los modelos no supervisados, para posteriormente entrenar este otro tipo de modelos con esos datos etiquetados. Con ello, se podría obtener una herramienta más precisa, que ya pudiese servir no para advertir, sino para asegurar si una cuenta o publicación es troll. Además, otra posible línea de desarrollo podría ser la generación de nuevas métricas que ayuden a obtener aún más información sobre los usuarios, para un mejor desempeño de los modelos a la hora de detectar anomalías.

Por último, debido a que esta herramienta va a continuar en desarrollo posteriormente a la entrega de este TFG, algunas de las ideas mencionadas en el párrafo anterior puede que sean implementadas para la asignatura de Proyectos IV. Además, hay tareas pendientes de ser implementadas para el final de esta otra asignatura, como la puesta en marcha del código desarrollado en servicios de computación cloud como Amazon Web Services, para así disponer de los recursos necesarios para calcular las métricas y emplear los modelos con el conjunto entero de datos.

6. REFERENCIAS

6.1 Bibliografía

Bacciu, A., et al. (2019): Bot and Gender Detection of Twitter Accounts Using Distortion and LSA

Bashivan, D., et al. (2015): Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery

Breunig, M.M, Kriegel, H.P., Sander, J. (2000): LOF: Identifying Density-Based Local Outliers

Edosomwan, S., et al, (2011): The History of Social Media and its Impact on Business

Ester, M., Kriegel, H.P., Sander, J., Xiaowei, X. (1996): A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.

Hinton, G. (2006): Reducing the Dimensionality of Data with Neural Networks

Liu, F.T., Ting, K.M., Zhou, Z.H (2008): Isolation Forest

Rumelhart, D., Hinton, G., Williams, R. (1986): Learning Representations by Back-Propagating Errors

Sánchez, L., Miñana, T., Arias, J., Basanta-Val, P., Fuentes-Lorenzo, D., Congosto, M., Fernandez, N. (2015): PRIMEROS RESULTADOS HACIA LA DETECCION AUTOMATICA DE BOTS EN TWITTER

Schölkopf, B., et al. (2000): Support Vector Method for Novelty Detection

Shevtsov, A., Tzagkarakis, C., Antonakaki, D., Ioannidis, S. (2021): Identification of Twitter Bots Based on an Explainable Machine Learning Framework: The US 2020 Elections Case Study

Vapnik, V., Chervonenkis, A., (1995): Support-Vector Networks

Wang, X., Zheng, Q., Kangfeng, Z., Cao, S., Shi, Y. (2021): Detecting Social Media Bots with Variational AutoEncoder and k-Nearest Neighbor

6.2 Webgrafía

Agarwal, A. (2023): How to install and use spacy models?, https://www.projectpro.io/recipes/install-and-use-spacy-models

Akhtar, T. (2021): Introduction to BERT and its application in Sentiment Analysis, Medium, https://medium.com/analytics-vidhya/introduction-to-bert-and-its-application-in-sentiment-analysis-9c593e955560

Alam, M. (2020): Isolation Forest: A Tree-based Algorithm or Anomaly Detection, Towards Data Science, https://towardsdatascience.com/isolation-forest-a-tree-based-algorithm-for-anomaly-detection-4a1669f9b782

Alba, D., Wagner, K. (2022): Twitter Staff Grapple With Impostors Like Nintendo, Lilly, Bloomberg, https://www.bloomberg.com/news/articles/2022-11-11/musk-s-twitter-staff-shift-focus-from-midterms-to-brand-fakes#xj4y7vzkg

Autor desconocido (2020): How to Use Autoencoder for Anomaly Detection, https://www.query.ai/resources/blogs/how-to-use-autoencoder-for-anomaly-detection/

Autor desconocido (fecha desconocida): https://tweeterid.com/

Awasthi, S. (2020): Seven Most Popular SVM Kernels, https://dataaspirant.com/svm-kernels/

Barnes, J. E. (2021): Russian Interference in 2020 Included Influencing Trump Associates, Report Says, The New York Times, https://www.nytimes.com/2021/03/16/us/politics/election-interference-russia-2020-assessment.html

Beedle, M. (2001): Manifesto for Agile Software Development, https://agilemanifesto.org/

Bessi, A., Ferrara, E. (2016): Social bots distort the 2016 U.S Presidential election online discussion, https://firstmonday.org/ojs/index.php/fm/article/view/7090/5653

Bohutska, J. (2021): Anomaly Detection – How to Tell Good Performance from Bad, Towards Data Science, https://towardsdatascience.com/anomaly-detection-how-to-tell-good-performance-from-bad-b57116d71a10

Brownlee, J. (2017): What Are Word Embeddings for Text?, https://machinelearningmastery.com/what-are-word-embeddings/

Daityari, S. (2019): How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK), DigitalOcean, https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk

Twitter (fecha desconocida): Data dictionary: Standard v1.1, Twitter API Standard v1.1, https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

DataTechNotes (2020): Anomaly Detection Example with Local Outlier Factor in Python, https://www.datatechnotes.com/2020/04/anomaly-detection-with-local-outlier-factor-in-python.html

DataTechNotes (2020): Anomaly Detection Example with One-Class SVM in Python, https://www.datatechnotes.com/2020/04/anomaly-detection-with-one-class-svm.html

De la Hera, C. (2022): Historia de las redes sociales: cómo nacieron y cuál fue su evolución, https://marketing4ecommerce.net/historia-de-las-redes-sociales-evolucion/

Dhiraj, K. (2020): Anomaly Detection Using Isolation Forest in Python, Paperspace, https://blog.paperspace.com/anomaly-detection-isolation-forest/

Elmundodelosdatos (2021): Introducción al topic modeling con Gensim (III); similitud de textos, https://elmundodelosdatos.com/topic-modeling-gensim-similitud-textos/

Equipo de scikit-learn (2021): Demo of DBSCAN clustering algorithm, documentación de scikit-learn, https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html

Equipo de scikit-learn (2021): sklearn.cluster.DBSCAN, documentación de scikit-learn https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

Equipo de scikit-learn (2021): sklearn.ensemble.IsolationForest, documentación de scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Equipo de scikit-learn, (2021): sklearn.neighbors.LocalOutlierFactor, documentación de scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor.kneighbors

Eurostat (2021): Foreign language skills statistics, https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Foreign_language_skills_statistics#Number_of_foreign_languages_known

Explosion AI (fecha desconocida): Trained Models and Pipelines, https://spacy.io/models

Ganegedara, T. (2019): Intuitive Guide to Understanding GloVe Embeddings, Towards Data Science, https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010

Ganesan, K. (2018): Word2Vec: A Comparison Between CBOW, SkipGram & SkipGramSI, https://kavita-ganesan.com/comparison-between-cbow-skipgram-subword/#.ZB1yqHbMKUl

GrabNGoInfo (2022): One-Class SVM For Anomaly Detection, Medium, https://medium.com/grabngoinfo/one-class-svm-for-anomaly-detection-6c97fdd6d8af

Jeet, (2020): One Hot encoding of text data in Natural Language Processing, Medium, https://medium.com/analytics-vidhya/one-hot-encoding-of-text-data-in-natural-language-processing-2242fefb2148

Karani, D. (2018): Introduction to Word Embedding and Word2Vec, Towards Data Science, https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

Keepler (2019): Isolation Forest: The Star Algorithm for Anomaly Detection, https://keepler.io/2019/06/isolation-forest-the-star-algorithm-for-anomaly-detection/

Krishnan, A. (2019): Anomaly Detection with Isolation Forest & Visualization, Towards Data Science, https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2

Kulshrestha, R. (2019): NLP 101: Word2Vec – Skip-gram and CBOW, Towards Data Science, https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314

Kuo, C. (2019): Handbook of Anomaly Detection with Python Outlier Detection – (12) Autoencoders, Towards Data Science, https://towardsdatascience.com/anomaly-detection-with-autoencoder-b4cdce4866a6#:~:text=The%20autoencoder%20models%20are%20important,and%20has%20delivered%20superior%20results

Lawler, R. (2022): Elon Musk's response to fake verified Elon Twitter accounts: a new permanent ban policy for impersonation, The Verge, https://www.theverge.com/2022/11/6/23443871/elon-musk-twitter-permaban-impersonation-parody

Macedo, G. (2022): Guerra híbrida: ¿Qué es y qué relación tiene con el conflicto Ucrania-Rusia?, https://www.elperiodico.com/es/internacional/20220326/guerra-hibrida-que-es-rusia-ucrania-13435046

Milmo, D., Hern, A. (2022): Twitter bans comedia Kathy Griffin for impersonating Elon Musk, The Guardian, https://www.theguardian.com/technology/2022/nov/07/twitter-will-ban-permanently-suspend-impersonator-accounts-elon-musk-says-as-users-take-his-name

Mishra, P. (2022): Detecting Outliers with Angle-based Techniques in Python, Paperspace, https://blog.paperspace.com/outlier-detection-with-abod/

Mohammed, A. (2023): How to convert string categorical variables into numerical values using Label Encoder in python, https://www.projectpro.io/recipes/convert-string-categorical-variables-into-numerical-variables-using-label-encoder

Morrison, S. (2017): Up To 15 Percent of Twitter Users Are Bots, Study Says, https://www.vocativ.com/410517/twitter-bots-study/index.html

Mullin, T. (2020): DBSCAN Parameter Estimation Using Python, Medium, https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd

Naveira, A. (2021): Historia de Facebook: nacimiento y evolución de la red social de los (más de) 2.000 millones de usuarios, https://marketing4ecommerce.net/historia-de-facebook-nacimiento-y-evolucion-de-la-red-social/

Nichols, C. (2022): Fake Nintendo account trolls Twitter with Mario image as advertisers begin to drop from the app, https://www.marketing-beat.co.uk/2022/11/10/fake-nintendo-mario-twitter/

Panghal, A. (2019): Sentiment Analysis in Python using Keras, GloVe twitter Word embeddings and Deep RNN on a combined dataset, Medium, https://medium.com/@panghalarsh/sentiment-analysis-in-python-using-keras-glove-twitter-word-embeddings-and-deep-rnn-on-a-combined-580646cb900a

Twitter (fecha desconocida): Post, retrieve, and engage with Tweets, Twitter API Standard v1.1, https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/overview

Riva, M. (2023): Word Embeddings: CBOW vs Skip-Gram, https://www.baeldung.com/cs/word-embeddings-cbow-vs-skip-gram

Sahil (2021): Word Embedding: CBOW & Skip-gram, Medium, https://medium.datadriveninvestor.com/word-embedding-cbow-skip-gram-8262e22fa7c

Salton do Prado, K. (2017): How DBSCAN works and why should we use it?, Towards Data Science, https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80#:~:text=The%20minimum%20value%20for%20the,value%20that%20should%20be%20chosen.

Sancho, F. (2020): Variational AutoEncoder, http://www.cs.us.es/~fsancho/?e=232

Shahidedu7 (2022): How to Calculate Cosine Similarity in Python?, Geeks for Geeks, https://www.geeksforgeeks.org/how-to-calculate-cosine-similarity-in-python/

Sharma, A. (2020): How to Master the Popular DBSCAN Clustering Algorithm for Machine Learning, https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/

Singh Chawla, J. (2018): What is GloVe, Medium, https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b

Srivignesh, R. (2021): Anomaly Detection using AutoEncoders – A Walk-Through in Python, Anomaly Detection using AutoEncoders | A Walk-Through in Python (analyticsvidhya.com)

The Associated Press (2022): Elon Musk targets impersonators on Twitter after celebrities troll him, OPB, https://www.opb.org/article/2022/11/07/elon-musk-targets-impersonators-on-twitter-after-celebrities-troll-him/

Vishal (2021): Python Regex Split String Using re.split(), Pynative, https://pynative.com/python-regex-split/

Wolff, R. (2020): Quick Introduction to Sentiment Analysis, Towards Data Science, https://towardsdatascience.com/quick-introduction-to-sentiment-analysis-74bd3dfb536c

Yalçin, O. G. (2020): Sentiment Analysis in 10 Minutes with BERT and Tensorflow, Towards Data Science, https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671

Yan, P. (2021): A Comprehensive Python Implementation of GloVe, Towards Data Science, https://towardsdatascience.com/a-comprehensive-python-implementation-of-glove-c94257c2813d

Yildirim, S. (2020): Introduction to Linear Algebra with NumPy, Towards Data Science, https://towardsdatascience.com/introduction-to-linear-algebra-with-numpy-79adeb7bc060

Zhao, Y. et al (2017): All Models, documentación de PyOD, https://pyod.readthedocs.io/en/latest/pyod.models.html#pyod.models.abod.ABOD

Zhao, Y. et al (2017): Source code for pyod.models.auto_encoder, documentación de PyOD, https://pyod.readthedocs.io/en/latest/_modules/pyod/models/auto_encoder.html

6.3 Índice de imágenes

ANEXOS

Glosario

API: Application Programming Interface o Interfaz de Programación de Aplicaciones son un conjunto de protocolos y definiciones que permiten la comunicación entre dos aplicaciones software sin necesidad de entender cómo están implementados.

Anomalía: valor atípico que se desvía considerablemente del comportamiento típico dentro del conjunto de datos.

Aprendizaje no supervisado: rama del Machine Learning en el que los modelos intentan aprender de un conjunto de datos sin etiquetar. De esta manera, este tipo de modelos puede ser utilizado para descubrir patrones entre los datos ocultos a simple vista o para examinar grandes volúmenes de información.

BERT: Bidirectional Encoder Representations from Transformers, o Bert, es un framework de aprendizaje automático para NLP que permite a un ordenador entender el significado de las palabras basándose en el contexto. Gracias a BERT, se puede fácilmente extraer embeddings de textos.

Bots: tipo de cuenta automatizada gracias a un software que utilice la API de Twitter. Este tipo de cuentas pueden publicar contenido en periodos de tiempo predefinidos, dar favorito o retuitear tweets de otros usuarios. Los bots pueden tener fines no maliciosos, como cuentas automatizadas que cuentan los días hasta la salida de un producto, o fines maliciosos, como intentar alterar la opinión pública sobre un tema.

Clustering: técnica del aprendizaje no supervisado que busca dividir un conjunto de datos en subgrupos o agrupaciones de datos que tengan características similares entre sí, pero distintas con los otros grupos.

Clusters: en aprendizaje automático, un cluster es una agrupación de datos no etiquetados que comparten características en común.

Copypasta: cadena de texto que es copiado y pegado por todo Internet en sitios como foros, blogs o redes sociales.

CSV: El formato CSV (Comma Separated Values) es un formato de archivo utilizado para almacenar datos en forma de tabla, donde cada línea del fichero representa una fila y cada valor separado por un signo separador (habitualmente una coma), representa una columna.

Detección de anomalías: técnica del aprendizaje no supervisado cuyo objetivo en encontrar patrones anómalos entre los datos utilizando distintas métricas como la densidad de probabilidad o funciones de distancia como puede ser la distancia de Mahalanobis.

Embedding: representación de una palabra o frase en un vector formado por números reales.

Engagement: medida del grado de interacción de un usuario, marca o producto con sus seguidores en redes sociales. El engagement se puede medir mediante el número de favoritos, cuántas veces se ha compartido una publicación o número y tipo de comentarios de las publicaciones.

Favoritos: funcionalidad que permite a los usuarios mostrar si un tweet les ha gustado. Twitter renombró esta funcionalidad a "Me gusta".

Guerra híbrida: estrategia militar consistente en mezclar tácticas de guerra convencional con otro tipo de estrategias no convencionales, como la realización de ataques terroristas o emplear las nuevas tecnologías con fines como la propagación de noticias falsas, robo de información, intervención política o afectar a las infraestructuras críticas de empresas o de estados.

Hashtags: funcionalidad inventada por Twitter que permite clasificar los tweets por tema o palabra clave. De esta manera, los usuarios pueden buscar en la red social aquellos tweets que traten sobre un tema específico gracias a los hashtags.

Hiperparámetros: parámetros que determinan el proceso de aprendizaje de un modelo de aprendizaje automático.

Librería de programación: conjunto de archivos de código cuya finalidad es facilitar la labor de programar a los programadores, al proporcionar funcionalidades comúnmente utilizadas desarrolladas previamente por otros desarrolladores.

NLP: Natural Language Processing, o NLP, es una rama multidisciplinar que integra la lingüística, la informática y la inteligencia artificial con el objetivo de que los ordenadores puedan procesar, analizar y entender el lenguaje humano.

Ratio: razón o proporción entre dos magnitudes relacionadas entre sí.

Red social: plataforma que permite a usuarios y organizaciones conectarse mediante intereses, actividades o relaciones en común.

Retweet: funcionalidad de la red social Twitter que permite a un usuario publicar el tweet de otro usuario. El número de retweets se puede utilizar para medir la relevancia de un tweet o usuario en la red social

Scraper: programas informáticos cuya finalidad es extraer información de páginas web de manera automatizada.

Sentiment Analysis: técnica de Procesamiento de Lenguaje Natural que permite determinar el tono emocional que subyace dentro de un texto.

Similitud coseno: medida de la similitud entre dos vectores en el espacio.

Stopwords: palabras comúnmente utilizadas en cualquier idioma que otorgan poco significado, como, por ejemplo, preposiciones.

Tokenizar: técnica de Procesamiento de Lenguaje Natural para separar frases o párrafos en unidades más sencillas, como palabras.

Trolls: cuentas que publican contenidos en redes sociales ofensivos, polémicos, provocadores o fuera de contexto.

Twitter: red social cuyo concepto inicial era que los usuarios se comunicasen entre sí mediante mensajes cortos, llamados tweets. Con el paso del tiempo, Twitter no solo ha ido añadiendo funcionalidades como los “Me gusta” o los retweets, sino que también ha ido expandiendo el límite de caracteres que pueden tener los tweets, pasando de 140 en sus orígenes a 4000 con una suscripción a Twitter Blue.

Fragmentos de código

Se van a mostrar algunos fragmentos del código desarrollado para este proyecto en esta sección. El código completo se puede encontrar en el siguiente enlace: https://github.com/Hertorias/DeteccionTrolls

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

df['embeddings'] = df['text'].apply(model.encode)
```

Aplicación de embeddings a los tweets.

```
lista_text2 = []

for i in df['user.id']:
    lista_text2.append(df.loc[df['user.id'] == i,]['embeddings'].values)

lista_dist = []
from numpy.linalg import norm

for i in lista_text2:
    lista_aux = []
    for x in range(len(i)-1):
        div = (norm(i[x])*norm(i[x+1]))
        if div > 0:
            lista_aux.append(np.dot(i[x], i[x+1])/div)
    lista_dist.append(lista_aux)
```

Cálculo de la similitud coseno

```
def sentimentWithSpacy(x):
    doc = nlp(x)
    sentiment = doc._.blob.polarity
    sentiment = round(sentiment,2)

    if sentiment > 0:
      sent_label = 1 #Positive
    else:
      sent_label = 0 #Negative
    return sent_label

df['sentiment'] = df['text'].apply(sentimentWithSpacy)
```

Aplicación de Sentiment Analysis a los tweets.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 3000))

# normalize the DataFrame
df_norm = pd.DataFrame(scaler.fit_transform(dataframe2), columns=dataframe2.columns)
df_norm2 = pd.DataFrame(scaler.fit_transform(dataframe3), columns=dataframe3.columns)
```

Escalado del dataframe sin embeddings y del dataframe con embeddings.

```
db = DBSCAN(eps=700, min_samples=50).fit(df_norm)

labels = db.labels_

n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Estimated number of clusters: %d" % n_clusters_)
print("Estimated number of noise points: %d" % n_noise_)
```

Definición, entrenamiento y predicción del modelo DBSCAN.

```
model=IsolationForest(n_estimators=100, max_samples='auto', contamination='auto', max_features=1.0)
model.fit(df_norm2)

model.decision_function(df_norm2)

predictions = model.predict(df_norm2)
print(predictions)

anomalies = np.count_nonzero(predictions == -1)
notAnomalies = np.count_nonzero(predictions == 1)

print("Numero de anomalias segun Isolation Forest: " + str(anomalies))
print("Numero de no anomalias segun Isolation Forest: " + str(notAnomalies))
```

Definición, entrenamiento y predicción del modelo Isolation Forest.

```
from pyod.models.auto_encoder import AutoEncoder
atcdr = AutoEncoder(contamination=0.15)
history = atcdr.fit(df_norm2)

scores = atcdr.decision_function(df_norm2)
print(scores)

predAE = atcdr.predict(df_norm2)
print(predAE)

anomaliesAE = np.count_nonzero(predAE == 1)
notAnomaliesAE = np.count_nonzero(predAE == 0)

print("Numero de anomalias con Autoencoders: " + str(anomaliesAE))
print("Numero de no anomalias con Autoencoders: " + str(notAnomaliesAE))
```

Definición, entrenamiento y predicción del modelo Autoencoders.

```
svm = OneClassSVM(kernel='rbf', gamma=0.05, nu=0.03)
print(svm)

svm.fit(df_norm2)
predSVM = svm.predict(df_norm2)

print("Numero de anomalias con SVM: " + str(np.count_nonzero(predSVM == -1)))
print("Numero de no anomalias con SVM: " + str(np.count_nonzero(predSVM == 1)))
```

Definición, entrenamiento y predicción del modelo One-Class SVM.

```
lof = LocalOutlierFactor(n_neighbors=20, contamination=.15)

predLOF = lof.fit_predict(df_norm2)

print("Numero de anomalias con Local Outlier Factor: " + str(np.count_nonzero(predLOF == -1)))
print("Numero de no anomalias con Local Outlier Factor: " + str(np.count_nonzero(predLOF == 1)))
```

Definición, entrenamiento y predicción del modelo Local Outlier Factor.

```
dataframe2['anomaliesAE'] = predAE
dataframe2['anomaliesIF'] = predictions
dataframe2['anomaliesSVM'] = predSVM
dataframe2['anomaliesLOF'] = predLOF
```

Introducción de las predicciones en el dataframe.

Mapa conceptual