Facultad de Ciencias Económicas y Empresariales

Love Brand: la personalización de la marca y éxito del engagement para el caso de TikTok.

Autor: Pilar Folqué Brier

Tutor: Ana Isabel Jiménez Zarco

MADRID | abril 2025

RESUMEN

Las marcas conocidas como *Love Brand* son aquellas que son capaces de generar una reacción diferente en el consumidor. Donde se llega a crear un vínculo emocional y un compromiso. Se trata de un elemento muy importante para tener en cuenta en el mundo actual debido a la ventaja competitiva que proporciona frente a otras marcas. Ya que es gracias a ello, que se consigue tener lo que se conoce como "*engagement*" donde el usuario es fiel al producto. Esto mismo ocurre con las experiencias personalizadas que ofrece la aplicación de *TikTok*. Es por ello por lo que, el presente trabajo tiene como objetivo estudiar este fenómeno. Donde se busca entender cuáles son los factores que hace que se desarrolle un nivel de vinculación hacia la marca provocando este tipo de reacciones en el comportamiento del usuario.

La metodología utilizada se ha basado en la aplicación de técnicas de visualizaciones de datos sobre diferentes bases de *TikTok* además de aplicaciones de diferentes modelos de *Machine Learning*. Donde las principales conclusiones han sido que se consiguió entender en profundidad el "*engagement*" en *TikTok*. Variables como los "*likes"*, visualizaciones, compartidos o cometarios muestran una clara relación lineal y un comportamiento muy parecido. Donde, además, se detectó una alta correlación entre las interacciones y la visibilidad del contenido. Se destaca la importancia del uso de hashtags para tener un mayor alcance y para la creación de nichos. Los mejores modelos de *machine learning* detectados que aplicar para estos casos son *XGBoost* y *Random Forest*. Además de la importancia del fenómeno del "boca a boca" y de la relación inversa del número de visitas con la duración del vídeo.

Palabras clave: *Love Brand*, *TikTok*, engagement, vínculo, experiencia personalizada, ventaja competitiva

ABSTRACT

Brands known as *Love Brands* are the ones that can generate a different type of reaction in the consumer. Where an emotional bond and commitment are created by the consumer. This is a very important aspect to consider in today's world. Due to the competitive advantage that it provides in comparison to other brands. It is thanks to this, that an "engagement" is created where the user keeps its loyalty to the product. This also happens with the *TikTok* app and the personal experiences that it offers. That is why the main objective of this project is to study this phenomenon. Where the idea is to search for the different factors that help develop a high level of attachment to the brand. Provoking these reactions on the user's behaviour.

The methodology used for the research was based on data visualization techniques on different TikTok datasets, as well the implementation of *Machine Learning* models. And part of the conclusion was the deep understanding of the engagement on *TikTok*. A clear relationship was observed between variables such as likes, views, shares and comments and sharing a very similar behaviour. There was a strong correlation identified between interactions and content visibility. It was also highlighted; how important it was the use of hashtags for both achieving greater reach and creating niche communities. The most effective machine learning models identified were *XGBoost* and *Random Forest*. There was a significant importance for the use of "word-of-mouth" and an inverse relationship between video views and its duration.

Key Words: *Love Brand*, *TikTok*, engagement, attachment, personalized experience, competitive advantage

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

Este capítulo tiene como objetivo hacer una introducción al trabajo de fin de grado desarrollado. Tratando de clasificar la información más relevante como es el contexto en el que se encuentra, los objetivos a desarrollar, la metodología y la estructura del trabajo que se va a seguir.

1.1 CONTEXTO

El análisis de datos es una de las prácticas más habituales en la actualidad para poder entender la situación en la que se encuentra una marca con respecto a su industria. En este caso se trata de la información de *TikTok* y de cómo interpretar la información para entender el vínculo que se tiene. Las marcas ahora tienen un papel fundamental que es el de saber interactuar con el cliente y entender qué percepción tiene el cliente mediante la representación de la marca.

1.2 OBJETIVOS

Lo que se busca con el desarrollo de este trabajo es poder entender las interacciones que tienen el cliente con la aplicación y cómo esto se puede enfocar en una buena aplicación de los servicios de *TikTok* para poder desarrollar una mejor vinculación con la marca. Y poder ver en qué consiste la vinculación que tiene un usuario con la marca.

1.3 METODOLOGÍA

Para poder desarrollar este trabajo me voy a centrar en identificar qué factores y patrones pueden afectar al comportamiento del consumidor por medio de un análisis de diferentes conjuntos de datos que recopilan información muy relevante para la herramienta y que pueden ayudar a identificar qué factores son los más relevantes. Todo ello para entender la

vinculación que existe entre el usuario y la marca. Gracias a un análisis de datos e interpretación de diferentes visualizaciones.

1.4 ESTRUCTURA DEL TRABAJO

El desarrollo de este proceso de investigación va a consistir en tres partes fundamentalmente. Primero de todo, en la definición de todos los conceptos necesarios para poder entender el tipo de análisis que se está haciendo. Lo segundo, será el desarrollo de un análisis de datos sobre diferentes fuentes. Se va a tratar de analizar y visualizar una gran variedad de información que traten la temática sobre las interacciones de los usuarios con la aplicación de *TikTok*. Y, por último, se llevará a cabo una conclusión que trate de abordar la manera en la que los datos se analicen y cómo interpretar la relación entre el usuario y la aplicación.

Capítulo 2. MARCO TEÓRICO

2.1 CONCEPTO

La idea principal que se tiene detrás de este proyecto es entender el mecanismo que hay detrás del nivel de afecto que tiene el cliente hacia la marca. En este caso en concreto se trata de *TikTok* y el cliente es el usuario. Esto es lo que se va a conseguir identificar mediante un análisis exhaustivo de las interacciones de usuarios con la aplicación en diferentes contextos que se recopila de diferentes fuentes de datos.

El nivel de afecto en el que me quiero enfocar es en el de "amor hacia la marca" para entender qué factores son los que más afectan al cliente para el desarrollo de este vínculo emocional. Entender el nivel de afecto "emocional" que puede llegar a tener un cliente hacia la marca. Todo esto permite a las empresas, que quieran comunicarse y desarrollar la publicidad de sus productos mediante *TikTok*, entender todos los factores que afectan para poder desarrollar esa relación afectiva con el cliente. La identificación de estos patrones puede hacer que se desarrolle un vínculo afectivo que haga que se creen oportunidades de negocio muy llamativas para la propia marca en si como *TikTok* o para una empresa como usuario de *TikTok* que haga uso de la propia herramienta para vender sus productos.

2.2 DEFINICIÓN DE UNA *LOVE BRAND*

Las "*love brands*" son aquellas marcas que han sido capaces de desarrollar un fuerte vínculo con su cliente que hace que, a la hora de elegir entre diferentes productos va a hacer que acabe eligiendo el de su marca preferida que en este caso es el de la "*love brand*". Para una organización, la marca se acaba convirtiendo en su activo más importante ya que es lo que les define y es su símbolo de identidad. Es por ello por lo que se debe de tener muy claro que se quiere hacer con la marca a nivel estratégico. Las claves más importantes para poder tener éxito en el desarrollo de este tipo de marcas se especifican en poder desarrollar un

marco de coherencia y consistencia en el que se establece una relación con el público que se tiene como objetivo y se le da la prioridad al cliente. Varios expertos destacan que gran parte de la problemática se podría acotar mediante un buen cuidado de sus empleados y su nivel de felicidad en la empresa (Hosteltur, 2024).

Hago, además, hincapié en que la conexión con el cliente siempre se encuentra en constante cambio y se mantiene constantemente adaptándose. Este tipo de empresas prestan mucha atención en mantener la conexión con el cliente. Es importante tenerle muy presente en todas las fases del proyecto a lo largo de todo el proceso de creación de un producto (Hosteltur, 2024).

2.3 TIPOS DE LOVE BRAND

La clasificación de "*love brands*" se basa en el sentimiento que producen en el usuario. Es bueno fijarse en las marcas a nivel mundial que más éxito han tenido en este ámbito. Esta recopilación se ha hecho a base de ver qué dicen las personas sobre las marcas.

Para poder analizar y entender las categorías que tienen este tipo de marcas, hay que identificar el enfoque emocional que se le da a la marca y cómo se siente el usuario. Un ejemplo para una marca que genera inspiración es *Nike* con su lema de "*Just do It*" con la idea de crear un mensaje de motivación que se pueda dirigir a todas las personas. Con la idea de representar a la idea como "entrenador" (García, 2024).

Por otro lado, se tienen las marcas que generan experiencias que luego los usuarios recuerdan y hacen que se vuelvan a consumir. Este es el caso de *Starbucks* en el que el cliente tiene una experiencia diferente a la de otro tipo de cafetería al encontrar no solo en una gran variedad de servicios que evolucionan a partir del café si no que la experiencia del local produce una satisfacción por el ambiente que se genera dentro y por la rapidez en la que se ofrece el servicio. Sobre todo, en su manera de promover un servicio que refleja la excelencia en el servicio al cliente (Moon y Quelch, 2005).

Se puede ver otro tipo de "*love brands*" como son las marcas que se han convertido en un icono. Este es el caso de *Coca-Cola*. Una bebida cuyo conjunto en su totalidad representa todo un icono en la cultura global y en el arte de la actualidad. Parte del éxito como icono viene del desarrollo de un buen diseño de la botella que le permitió poder convertirse en un símbolo además de su famoso logo. Ya que el logo es un elemento fundamental de todas las conocidas "*love brands*". La empresa lleva desde hace mucho tiempo, con la estrategia de convertirse en una bebida que se mantenga constante a lo largo de los tiempos. El diseño y todo su conjunto exterior le permitió convertirse en un icono de la cultura *pop* (The Coca-Cola company).

Por otro lado, se tienen las marcas que son expertas en expresar humanidad y desarrollar empatía. Y desarrollan una experiencia muy personalizada. Es el caso de *IKEA*. Se establece que parte del cariño que se le tiene a la marca viene de lo que se expresa en el "boca a boca". *IKEA* es capaz de establecer un fuerte vínculo emocional por la experiencia que generaba mediante diferentes estímulo y sentidos. Generando un fuerte impacto que es lo que hace que los clientes recomienden diferentes productos (Rodrigues y Brandão, 2020).

Es interesante entender el tipo de amor a la marca que generan "*engagement*". Como se acaba de mencionar, las "*love brands*" son marcas que generan un tipo de impacto en la persona. Puede ser una marca que es capaz de inspirar y generar una motivación para poder alcanzar algunos objetivos en concreto. Otras marcas generan una experiencia sobre el usuario gracias a la compra de su producto, que puede ser para consumir, y que consigue generar una asociación emocional.

Y como también se ha podido ver, hay marcas que consiguen "enganchar" al cliente y que son un icono y representan algo fundamental de la cultura. Este tipo de marcas consiguen vincularse a través de una conexión de momentos especiales para el usuario. Y luego hay otras marcas que consiguen generar una experiencia sensorial gracias a la capacidad de hacer al cliente sentirse identificado y ser parte de algo. La idea es conseguir generar algún tipo de vínculo emocional con el consumidor. Y se le puede dar diferentes enfoques y conseguir

construir una relación fuerte y consolidada. A continuación, se va a profundizar más en el tema de cómo de ventajoso es generar "*engagement*" en el usuario.

2.4 VENTAJAS DEL ENGAGEMENT

La generación de un compromiso por parte del consumidor se ha definido como un fenómeno psicológico. Se necesita generar algún tipo de "conexión a la marca" que acaba traduciéndose en una mayor disposición a pagar un producto sin depender de una subida de precio. Además de que se propague entre consumidores al hacerse recomendaciones entre diferentes productos. Se ve claramente que se tiene un alto impacto a nivel financiero, al producirse este tipo de fenómenos. Debido a que el cliente está dispuesto a pagar por ello. Además, si se llega a producir un error, no se percibe como algo tan grave (Park et al., 2006).

Otra razón por la que es interesante tener en cuenta el "*engagement*" es porque es capaz de sacar información relevante para poder elaborar un plan estratégico a la hora de gestionar la marca (Bolton, 2012).

2.5 ANTECEDENTES DEL ENGAGEMENT

Se identifican como antecedentes al compromiso como marca, a la identidad de la marca en sí y su sentido de comunidad. Siendo estos los que más influencian en la creación de un amor por la marca. Esto consigue generar lealtad hacia el producto que vende la marca en sí. (Bergkvist y Bech-Larsen, 2010).

Otro elemento fundamental es el apego hacia la marca que consigue generar una conexión emocional fuerte. Se conocen como bases fundamentales de apego gracias a las experiencias, las eficiencias en la cotidianeidad y la fuerte identificación con un símbolo. Además de tener una confianza en la marca para poder continuar fomentando el comportamiento. Algunos factores que pueden fomentar un mejor "*engagement*" son los niveles de consistencia para poder generar valor, la experiencia que realiza el usuario y el sentimiento de comunidad que ayudan a crear una sensación de "colectividad" (Bolton, 2012).

2.6 ALGORITMO PARA DETECTAR ENGAGEMENT

En los siguientes capítulos se va a desarrollar un algoritmo de análisis de información de diferentes fuentes de datos que busca encontrar la manera de interpretar el "*engagement*" de la aplicación de *TikTok*. Entre las preguntas que se pretenden resolver en este análisis se encuentra identificar cuáles son los principales factores que determinan el nivel de "engagement" en la aplicación de *TikTok*. Se pretende analizar la manera de entender el uso de *hashtags*, además de lo visible y viral que puede llegar a ser un contenido. Se va a buscar también analizar la relación entre el número de seguidores y lo viral que puede ser un vídeo. Además, del impacto de la duración del vídeo a nivel de cómo interactúa con el usuario. Además, se busca predecir a través de modelos de *Machine Learning* la capacidad de predicción de la variable "*engagement*". Se busca también entender cómo afecta el grado de personalización del algoritmo para que los usuarios permanezcan en la plataforma.

Capítulo 3. ANÁLISIS EMPÍRICO

3.1 CONTEXTO SOBRE TIKTOK

TikTok es una red social que aparece en 2018 que se caracteriza por un algoritmo de carácter muy personalizado para cada usuario con capacidad para comprender lo que le gusta a cada persona que interactúa con la herramienta. Es conocida por su alto "*engagement*" con el usuario. Es por ello por lo que la mayoría de las empresas se quieren beneficiar de su uso. El objetivo, en parte, de este trabajo de investigación es entender en qué se basa esta capacidad de conectar con los usuarios de la aplicación. Se trata de una forma de conseguir alcanzar una mayor audiencia (Peña-Fernández et al., 2022).

La capacidad de "*engagement*" se basa principalmente en el uso de un algoritmo que es capaz de detectar mediante patrones los gustos del usuario en función de cómo actúe con la herramienta. Su atractivo entre los jóvenes es algo que otras aplicaciones como *Twitter* o *Instagram* han estado intentando conseguir mediante sus adaptaciones (Peña-Fernández et al., 2022).

Es importante entender el origen de *TikTok*, en 2018, parte de la base "*Musial.ly*". Donde la franja de edad de la mayoría de los usuarios se encontraba entre los 10 y los 29 años. En 2020 llegó a alcanzar más de 45 millones coincidiendo con el inicio de la pandemia por "*COVID-19*" donde llegó a ser la segunda aplicación más descargada superando a otro tipo de plataformas como *Snapchat* o *Twitter* (Peña-Fernández et al., 2022).

El motivo por el que se puede considerar una herramienta con gran capacidad de "*engagement*" se debe a la capacidad que tiene para desarrollar la comunicación. Con la particularidad de crear contenido mediante vídeos desde un enfoque narrativo. Parte de su técnica para conseguir grandes capacidades de viralización se basa en sus mecanismos de creación de *hashtags* que funcionan como canales de difusión (Peña-Fernández et al., 2022).

Es decir, que parte del éxito de la *TikTok* con respecto a la capacidad que tiene de involucrar al usuario se basa en el algoritmo que usa, que está creado con Inteligencia Artificial. Gracias al uso de esta, se ha conseguido llegar a un consumo de una herramienta de carácter muy personalizado para el usuario. Donde la principal base de su análisis se fundamenta en el estudio de los cambios generados en los "*likes*", comentarios y visualizaciones (Peña-Fernández et al., 2022).

Parte de la manera por la cual consigue mantener esta relación con el usuario se basa en que el usuario participa en "directo" con el contenido que quiere ver. Es decir, que es él el que decide qué es lo que quiere consumir. A diferencia de otras redes sociales que se basan en lo que otros seguidores quieren ver. Esta manera de interactuar, donde el usuario toma la iniciativa sobre su propio contenido a consumir se ha convertido en la estrategia más innovadora que se está llevando a cabo en el ámbito de la comunicación. Y es que esta innovadora manera de adaptar contenido es lo que han estado haciendo diferentes medios de comunicación (Peña-Fernández et al., 2022).

Un ejemplo muy claro de esto último que se acaba de mencionar es el del paródico del "*Washington Post*" que ha sido pionera en el uso de la herramienta para poder optimizar su contenido. Se ha hecho uso de información aportada por este periódico como parte de la investigación de este proyecto. Esto se debe a las diferentes aplicaciones que tiene TikTok para el campo del periodismo. Es por eso por lo que es interesante ver la manera en la que los medios utilizan la herramienta de *TikTok* para ver cómo consiguen conectar con un público más joven para poder adaptarse a la nueva realidad de la comunicación (Peña-Fernández et al., 2022).

Las variables más importantes para poder entender este fenómeno son quién está realmente presente en *TikTok*, qué interacciones realizan con la herramienta, qué tienen en común las distintas comunidades de usuarios que consumen contenido similar y cuál es la continuación de la creación de contenido similar. Se han identificado la creación de comunidades gracias al uso de "*hashtags*". Y esto se ha podido comprobar con nichos de contenido periodístico. Donde se concentran usuarios interesado de la materia en cuestión. Donde al tener un

contenido muy especializado, se aumentan los niveles de engagement. Siendo un elemento clave para el éxito en la plataforma (Peña-Fernández et al., 2022).

Se ha descubierto que el algoritmo utilizado en *TikTok*, a diferencia de otros algoritmos de recomendación, personaliza el porcentaje de vídeos que recomienda en función de intereses pasados. Es decir, es capaz de generar diferente una proporción de recomendaciones en base a intereses pasados en función del usuario que se tratase (Vombatkere et al., 2024).

Es por ello por lo que se basa en identificar los puntos clave para entender la capacidad de personalización de la aplicación. Se encuentran dos posibles corrientes que son la parte de la exploración donde se muestran los nuevos intereses de usuarios y en la parte donde se reflejan los intereses previos que tenía el usuario. Después de analizar una investigación donde se comparaba datos reales sacados de *TikTok* con datos generados por "*bots*" automatizados. Se llegó a la conclusión de que se personalizaba entre un 30 y un 50% de los primeros 1000 vídeos que se mostraban a los usuarios. Dónde más se veía el impacto de la personalización era en los "*follows*" y "*likes*". Se destacó, que los vídeos con la etiqueta de "*exploit*" tenían mayor puntuación para la personalización ya que eran de carácter mucho más específico. Por otro lado, se vio que el uso de muchos hashtags en vídeos eran vídeos de carácter más personalizado que los vídeos con menos uso de *hashtags* (Wang y Guo, 2023).

Otro factor muy interesante es ver cuál es la relación ente la conciencia del algoritmo con la motivación para el uso de TikTok entre jóvenes, y se destacó que el nivel sobre la conciencia de los algoritmos era moderado. Identificando que la manera que tenían los algoritmos de recomendar conseguía aumentar la permanencia de los usuarios en la plataforma. Generando lo que se conoce como burbujas de filtro que consigue que se refuercen los intereses de los usuarios y haciendo que se limite la diversidad en el contenido (Wang y Guo, 2023).

Con respecto a la implementación de la inteligencia artificial en el algoritmo de *TikTok*, se destaca que la aplicación basa parte de su éxito en la personalización del contenido debido al uso de algoritmos basados en IA que permiten llegar a la optimización de contenido que se muestra luego al usuario. Aunque, sin embargo, no hay mucha transparencia por parte de

TikTok sobre cuáles son las normas que se rigen detrás del algoritmo de "Para Ti". Sin embargo, sí que se ha expresado que se hace uso de una IA con parámetros de entrada sobre las interacciones, los gustos, las descripciones de los vídeos, entre muchos otros. Y que el sistema se iba retroalimentando constantemente con la información que proporcionaba el usuario. Consiguiendo así mejores resultados a la hora de recomendar, conforme más interacciones se generaban. Se destaca la falta de transparecía a la hora de indicar cómo se consigue la capacidad de personalización que tiene (Sidorenko-Bautista et al., 2024).

3.2 ANÁLISIS SOBRE LA MARCA

Para el caso en concreto del análisis, lo enfoqué en el desarrollo de resultados de diferentes fuentes de datos: todos ellos procedentes de la plataforma de "*Kaggle*". Por un lado, se tenía la base de datos de "*dataset_free-tiktok-scrapper_2022-07-27_21-44-20-266.xlsx*" que contenía 1200 elementos que se basa en contenido sobre "*hashtags*" de TikTok. Con las diferentes variables de medición como son el avatar, "*digg*", *fans*, seguidores, *id*, "*nickNames*", firmas, verificaciones, videos, comentarios de verificación, momento en el que se creó, descargas, además de muchas otras características como efectos de los *stickers* o los *hashtags* (Muhammad, 2023).

Otra base de datos que analicé fue la de "*tiktok_dataset.csv*" con 19382 ejemplares. Esta base de datos muestra información sobre reportes de vídeos de usuarios y comentarios que además incluyen reclamaciones de estos. Esta base de datos se basó en un principio para utilizarse con el fin de realizar investigaciones y con un objetivo de carácter pedagógico. Donde cada fila representa el reporte de un usuario. Con variables como el "*status*", el *id* del video, la duración en segundos del vídeo, en número de visitas, compartidos, "*likes*" y descargas entre otras (Muhammad, 2023).

Por otra parte, la base de datos de "*trending.csv*" con variables que ya he mencionado como el número de cometarios, número de compartidos, número de reproducciones entre muchos otros además del nombre del usuario y su *id*. Se trata de los 1000 vídeos de TikTok con de

carácter más *trending* sacados mediante "*scrapping*". Son un conjunto de datos analizados para un perfil personalizado de *TikTok* (Van De Ven, 2020).

Por otro lado, también he analizado otra base de datos que es "*tiktok_top_1000.csv*" que tiene 1000 muestras y clasifica los datos en función del país, el *ranking*, la cuenta, el título, el número de "*likes*" y de visitas entre muchas otras características. Se puede observar que claramente se trata de muchas etiquetas que se les pone a los datos para poder clasificarlos de tal manera que permita poder proporcionar información sobre factores muy relevantes con respecto a *TikTok*. Se trata de los 1000 "*influencers*" más relevantes del mundo (Kanawattanachai).

Por otro parte, analicé la base de datos de "*omnibuslaw_video.csv*" con más variables interesante como la longitud del vídeo, el número de "*likes*", el número de compartidos o el número de cometarios. Además de muchas otras variables que ya se han mencionado antes. Esta base de datos tenía como objetivo medir el éxito de un vídeo en la red social en función del número de "*likes*", compartidos y comentarios. Se incluían otra serie de parámetros como el identificador del vídeo, la descripción, compartidos, comentarios y el enlace al vídeo. Se querían explorar factores que hacen que un vídeo se haga popular en *TikTok* (The Devastator).

Y por último la base de datos de "*washingotnopost_videos.csv*" que contenía información que ya he mencionado en otras bases de datos como *IDs* o parámetros sobre el vídeo como longitud, número de "*likes*", o el número de comentarios. Al igual que el *dataset* que acabo de mencionar, el objetivo de su uso es para hacer análisis de preferencias de redes sociales. Con la idea de analizar las preferencias del usuario para las redes sociales y predecir futuras tendencias (The Devastator).

3.3 MÉTODOS

A la hora de desarrollar los resultados sobre la marca, lo desarrollé en dos partes fundamentales que son un análisis de las visualizaciones que venían de los datos y una

implementación de diferentes modelos de "*machine learning*" para poder sacar información de los datos.

3.3.1 ANÁLISIS DE LAS VISUALIZACIONES

A continuación, voy a mostrar gran parte de las visualizaciones realizadas para este análisis.

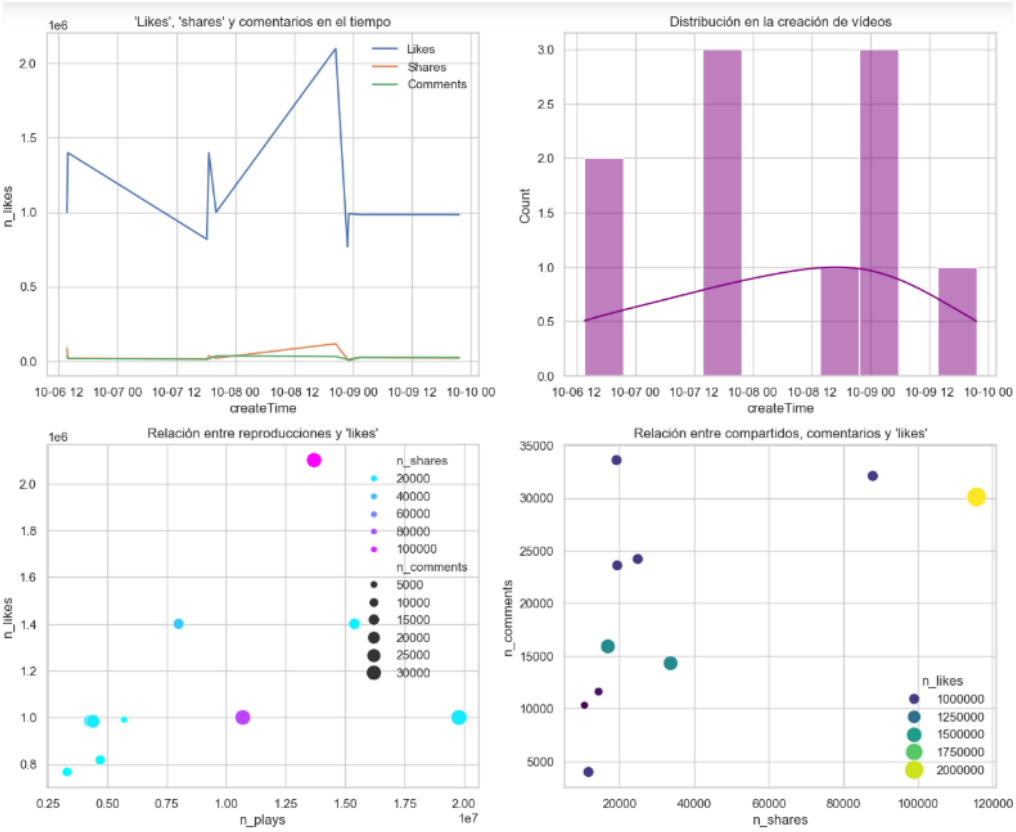
Figura 1: Distribución ancho de vídeos TikTok de Free TikTok Scrapper

Fuente: Elaboración Propia

Al observar la Figura 1, destaca por que la duración del ancho de banda de los vídeos en píxeles sigue una distribución normal centrada en torno 550 píxeles. Seguido de una segunda pequeña distribución centrada cerca de 700 píxeles. Pero, que, sin embargo, su tamaño es muy inferior. Con una proporción que no se concuerda con la realidad. Se observa, que realmente esta distribución temporal tiene mucho sentido ya que son vídeos de *TikTok* cuya duración es limitada y suele ser muy similar. Es por ello por lo que se observan dos

distribuciones normales. Se destacan dos claras tendencias con respecto a lo de la distribución de los datos.

Figura 2: Relación entre variables para "Omnibuslaw Videos"



Fuente: Elaboración Propia

La figura superior muestra mucha información relevante sobre diferentes parámetros que se recopilan en la base de datos de "*Omnibuslaw Videos*". Entre ellos, la distribución del número de "*likes*", de compartidos y de comentarios a lo largo del tiempo. Esto es lo que se está representando en la imagen superior de la izquierda. Se observa un pico el día 10-08 en el número de "*likes*" del año 2020 durante el mes de octubre. Esto se puede deber en parte a que fue durante ese año que se aprobó la conocida ley en el parlamento de Indonesia que fue promulgada más adelante en el mes de noviembre (Secretaría de estado de comercio, 2025). La tendencia del número de comentarios también aumentó durante este periodo, aunque no

de una manera tan pronunciada como con el número de me gustas. Se observa un importante pico de bajada después del día de la aprobación de esta.

En la imagen superior derecha, se muestra una distribución temporal de la fecha de la creación de vídeos. Donde se destacan tres fechas importantes que son el seis, el siete y el nueve de octubre. Siguiendo una distribución que es muy parecida a la normal.

Las dos visualizaciones inferiores tratan sobre la correlación entre diferentes métricas como son el número de visitas, de "*likes*", de compartidos y de comentarios. Con respecto a la imagen de la izquierda, se observa una relación entre el número de reproducciones y de "*likes*" con respecto a la relación que hay con los compartidos y con el número de comentarios. Hay una relación relativamente directa debido a que a medida que aumenta el número de reproducciones también lo hace el número de me gustan. Aunque también se observan puntos donde el número de likes es bajo, pero el número de reproducciones se mantiene alto. Esto se debe a que sea un vídeo muy visto, pero con mala reacción por parte de la audiencia. Es importante destacar que los puntos donde la magnitud es mayor se tratan de que con un número alto de compartidos y de comentarios, se mantiene alto el número de reproducciones y de me gusta.

Con respecto al gráfico inferior derecho, se observa una relación con respecto a la cantidad de compartidos, comentarios y "likes". Mostrando qué relación hay entre los compartidos y los comentarios. Y se observa que hay una cierta relación lineal entre estas dos variables ya que a medida que uno de los dos aumenta también lo hace el siguiente. Con respecto a la relación que se muestra con los colores del número de me gusta, se observa que también hay una relación directa con las dos variables que acabo de mencionar ya que a medida que hay un mayor número de interacciones también hay un mayor número de me gustas. Esto se puede deber a que se genera mayores reacciones sobre el contenido.

Figura 3: Estadísticos Métricas Más Importantes Para Free TikTok Scrapper

	playCount	shareCount	videoMeta/duration
count	1.200000e+03	1.200000e+03	1200.000000
mean	5.468718e+07	1.569660e+05	29.741667
std	5.784957e+07	2.546967e+05	28.546173
min	9.225000e+03	0.000000e+00	0.000000
25%	1.447500e+07	1.717500e+04	13.000000
50%	3.690000e+07	6.515000e+04	19.000000
75%	7.085000e+07	1.971750e+05	41.000000
max	3.548000e+08	2.800000e+06	356.000000

Fuente: Elaboración Propia

Con respecto a las métricas sobre los estadísticos de la base de datos analizada de "*Free TikTok Scrapper*", se destaca que las variables más representativas son el número de reproducciones o como la nombran "*playCount*", "*shareCount*" que corresponde a la cuenta del número de compartidos y "*videoMeta/duration*" que representa a la duración del vídeo en segundos. Destaca por la alta variabilidad en las reproducciones y los compartidos llegando a unos máximos muy altos de 3.54 y 2.8 millones respectivamente.

Figura 4: Estadístico Métricas Más Importantes Para Omnibuslaw Videos

	n_likes	n_shares	n_comments	n_plays
count	1.000000e+01	10.000000	10.000000	1.000000e+01
mean	1.144300e+06	35440.000000	19966.900000	9.000000e+06
std	3.957884e+05	36202.952611	10192.110265	5.663725e+06
min	7.670000e+05	10600.000000	3969.000000	3.300000e+06
25%	9.838250e+05	15025.000000	12275.000000	4.475000e+06
50%	9.951000e+05	19300.000000	19750.000000	6.850000e+06
75%	1.300000e+06	31500.000000	28625.000000	1.295000e+07
max	2.100000e+06	115700.000000	33600.000000	1.980000e+07

Fuente: Elaboración Propia

Para las métricas de la base de datos de "*Omnibuslaw Videos*", se destaca como métricas más relevantes el número de "*likes"*, el número de compartidos y el número de reproducciones. Debido a que se trata de una base de datos menor, la muestra es muy

pequeña, pero se puede ver que el número de "*likes*" más alto asciende a 2 millones con un máximo de 115 mil compartidos.

Figura 5: Correlación datos Free TikTok Scrapper

	playCount	shareCount	videoMeta/duration
playCount	1.000000	0.546308	-0.106476
shareCount	0.546308	1.000000	-0.095679
videoMeta/duration	-0.106476	-0.095679	1.000000

Fuente: Elaboración Propia

En la figura superior se muestran los valores de correlaciones entre variables para el caso del *dataset* sobre "*Free Tiktok Scrapper*". Se ve que hay una clara correlación entre "*shareCount*" y "*playCount*". Mientras que existe una relación inversa de aproximadamente un valor de "-0.1" para entre las variables de "*videoMeta/duration*" y "*playCount*". Y luego, por otro lado, para "*videoMeta/duration*" y "*shareCount*".

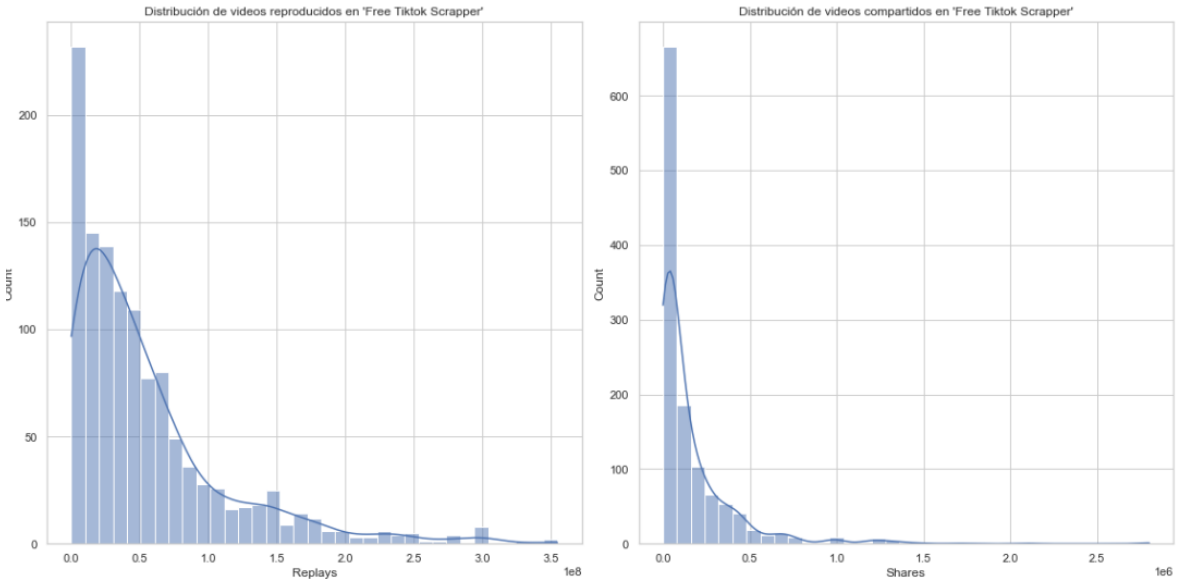
Figura 6: Correlación para Omnibuslaw Videos

	n_likes	n_shares	n_comments	n_plays
n_likes	1.000000	0.701418	0.304627	0.476969
n_shares	0.701418	1.000000	0.604274	0.329372
n_comments	0.304627	0.604274	1.000000	0.582052
n_plays	0.476969	0.329372	0.582052	1.000000

Fuente: Elaboración Propia

Para el caso de la imagen superior se muestran los valores entre correlaciones de variables para el caso del dataset de "*omnibuslaw_videos_metrics*" que representa una correlación muy alta entre las variables de "*n_likes*", y "*n_shares*". Por otro lado, se observan otras correlaciones por encima del 0.5 para el caso de "*n_shares*" y "*n_comments*". Ocurre lo mismo para "*n_comments*" y "*n_plays*". Por otro lado, algunas de las correlaciones que se pueden apreciar que no tienen ningún valor significativo son para el caso de "*n_comments*" y "*n_likes*" y también para el caso de "*n_plays*" y "*n_likes*" y por último para el caso de "*n_shares*" y "*n_plays*".

Figura 7: Distribución de Vídeos Compartidos y Distribuidos en Free Tiktok Scrapper



Fuente: Elaboración Propia

Con respecto a las visualizaciones que se muestran en la parte superior, se ve que en las distribuciones para los videos de *TikTok* en general hay una clara tendencia hacia la izquierda: se observa una distribución sesgada a la derecha.

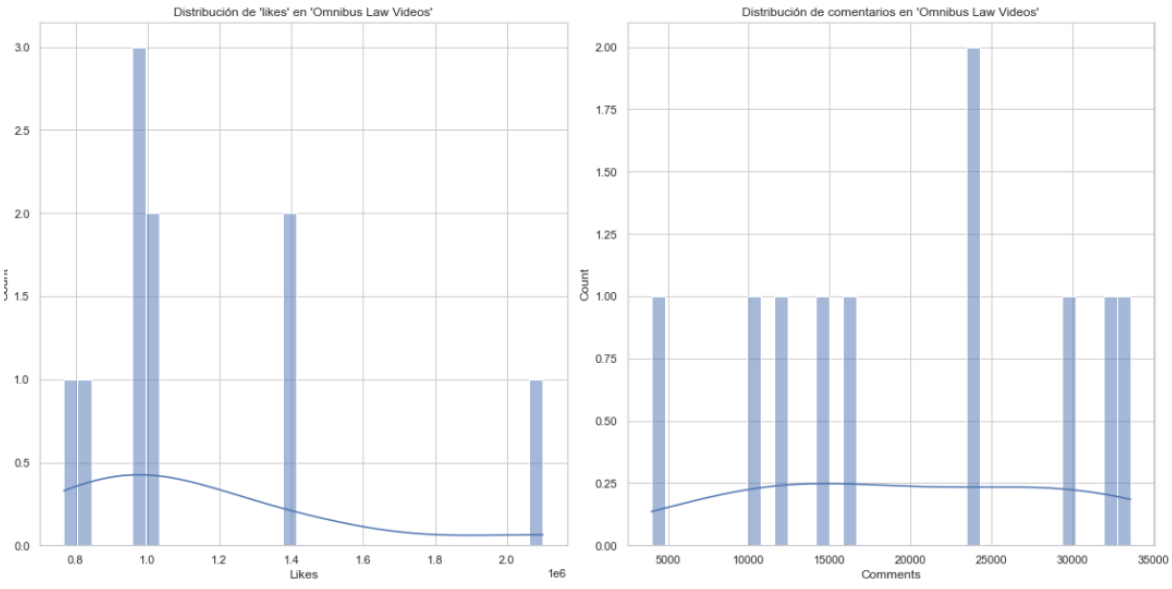
Se observa también que lo normal es que la cantidad de vídeos que haya disponible es de bajo número de reproducciones, y que, inusualmente a medida que el número de reproducciones aumenta, las reproducciones en si disminuyen.

Muy pocos videos son los que llegan a alcanzar un gran número de visualizaciones, y esto es cuando se empieza a encontrar en el rango de millones. La gran mayoría de estos vídeos no llegan a este gran alcance. Y es evidente de que cuando un vídeo llega a alcanzar un gran número de visualizaciones, se convierte en un vídeo viral.

Por otro lado, al observar la gráfica de los vídeos compartidos de esta misma base datos, se ve que el número de "*shares*" tiene un comportamiento muy similar al del número de "visitas".

Esto implica que no hay tantos vídeos compartidos y que a media que el número aumenta, se ve una mayor disminución de los compartidos. Entre las implicaciones de esto, se encuentra que los videos se comparten si al usuario realmente le gustan o si encuentra en ellos algo que realmente le llame la atención. Se observa que hay ciertos videos que hacen que se compartan más que otros.

Figura 8: Distribución de Likes y de Comentarios en Omnibuslaw Videos



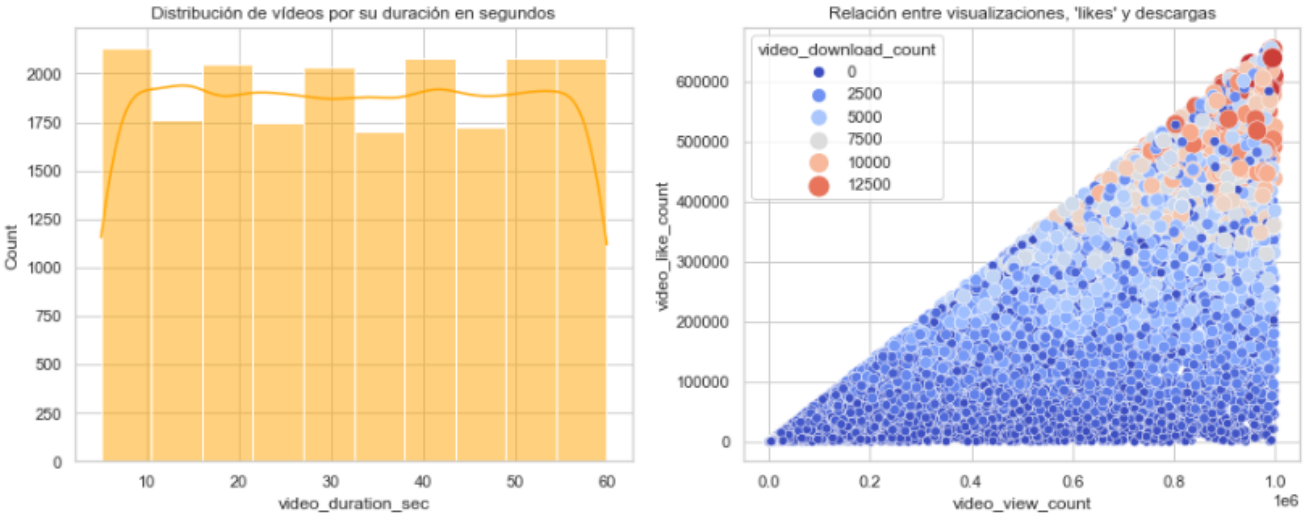
Fuente: Elaboración Propia

Con respecto a los vídeos sobre la ley "*Omnibus*", se observa que hay más dispersión con respecto a los "*likes*" con algunos importantes. Esto implica que hay diferentes grupos con más "*likes*". Y es gracias a este tipo de distribución se observa que hay estos grupos que sobresalen por alcanzar el gran número de "*likes*". Esto indica cómo la variabilidad del contenido y cómo de atractivo sea el contenido, hace que se le consiga dar más "*likes*" o menos.

Con respecto a la manera en la que se distribuyen los comentarios de los videos de la ley "*Ómnibus*" se ve que se distribuye de manera uniforme. Y únicamente mostrando un pico claro. Todos esto se podría deber a que hace comentarios a vídeos si se corresponde con un comportamiento con mayor consistencia ya que hay muchas más interacciones sin importar realmente la calidad del vídeo.

Como conclusión a todas estas observaciones, se observa una interacción con alto valor por aquellos videos que lleguen a tener algún tipo de valor. Hay ciertos tipos de vídeos que consiguen obtener un alto número de interacciones. Es muy interesante analizar de qué tipos de vídeo se trata esto. Con intuición se puede llegar a ver que se puede deber a su calidad y relevancia. Además, de donde fueron publicados.

Figura 9: Visualizaciones del TikTok Dataset



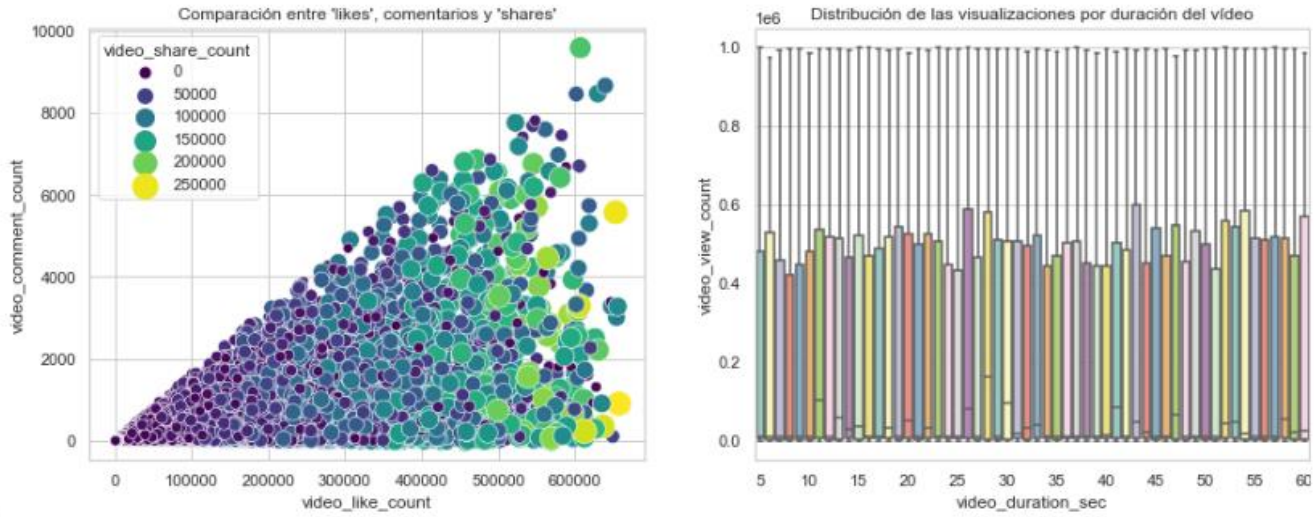
Fuente: Elaboración Propia

En la figura superior se muestran algunos datos relevantes sobre la tabla de "*TikTok Dataset*". En la imagen de la izquierda, se observa una relación del gráfico de barras que muestra la frecuencia de la duración de los vídeos en segundos. Con la línea de densidad que aparece, se observa una distribución uniforme que sufre de ligeras variaciones. Se ve que hay unas duraciones muy similares entre vídeos sin ver una tendencia hacia algún tipo de visualización en concreto. Se asemeja a una distribución uniforme en el sentido de que la distribución sigue unos parámetros muy parecidos, pero no llega a ser perfecta.

Con respecto al gráfico de la derecha, se observa un número de visitas, de "*likes*" y de descargas. Se trata de otro gráfico de dispersión donde se muestra la relación entre el cuenteo del número de visitas a un vídeo y en el eje Y el número de "*likes*" en los vídeos. Se observa una clara relación lineal lo que implica que, a mayor número de visualizaciones, mayor número de "*likes*". Esto significa que para poder llegar a mayor número de reacciones hay

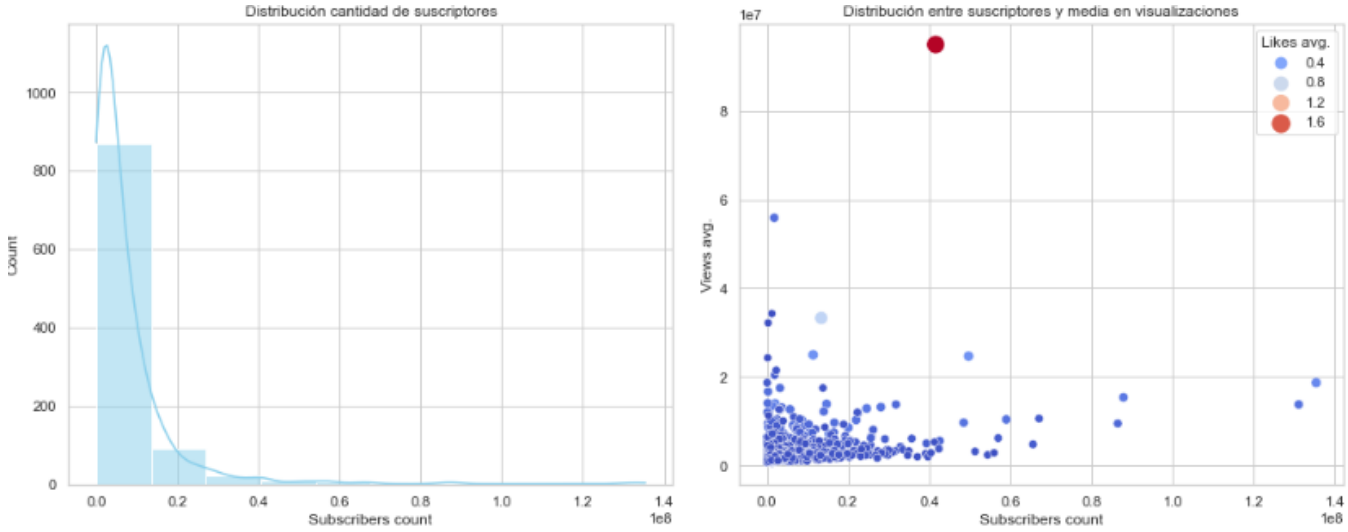
que conseguir un alto impacto mediante el número de visualizaciones, y de esta manera conseguir enganchar a los usuarios con contenido que les llame la atención y les guste. Queda clara la relación entre descargas, "*likes*" y visualizaciones.

Figura 10: Visualizaciones de TikTok Data Set



Fuente: Elaboración Propia

Figura 11: Distribución número de suscriptores y media de visualizaciones de Tiktok Top 1000

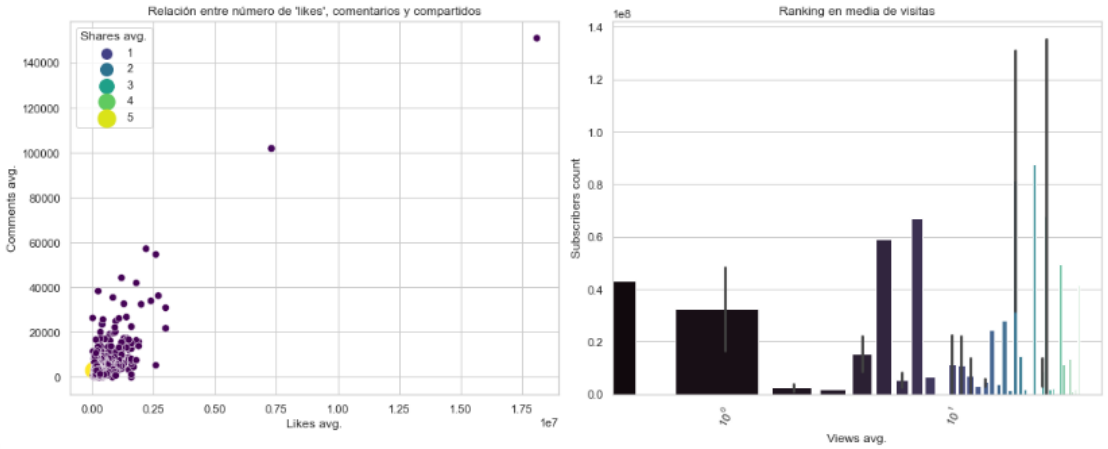


Fuente: Elaboración Propia

La imagen superior muestra visualizaciones que corresponden al dataset de "*Tiktok Top 1000*". En la imagen de la izquierda, se observa la relación con la distribución del número de personas que se convierten en seguidoras. Se observa que la mayoría de muestras no llegan a tener una gran cantidad de seguidores. Es decir, que el éxito en *TikTok* no está relacionado con el número de personas que te sigan. Se ve representada una distribución con sesgo hacia la derecha, implicando esto que son solo unos pocos usuarios los que consiguen llegar a alcanzar un gran número de seguidores.

Con respecto a la imagen de la derecha, se ve una relación entre el número de suscriptores con la media de las visitas. Donde la mayoría de los puntos se encuentran en la parte inferior de la izquierda. Con una media de "*likes*" muy baja. Esta imagen muestra otra importante característica con respecto a la cantidad de los suscriptores. Se observa que se puede tener un número alto de suscriptores y que la cantidad de me gustas no tiene por qué ser relevante. Es decir, que la información sobre el número de me gustas no tiene por qué significar nada en concreto. Se observa también que ocurre lo mismo para el número de seguidores y de visualizaciones. Ya que no se ve que este segundo parámetro tenga relación directa con la cantidad de seguidores.

Figura 12: Visualizaciones para TikTok Top 1000



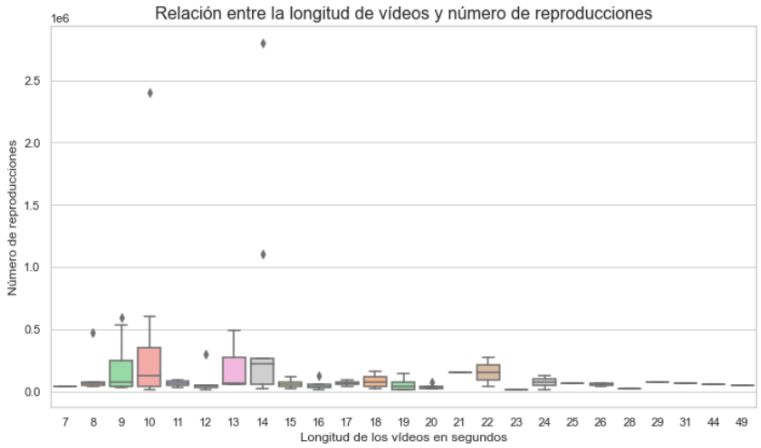
Fuente: Elaboración Propia

En la imagen superior se muestran dos visualizaciones que también forman parte del *dataset* "*Tiktok Top 1000*". En la imagen izquierda se muestra la relación entre el número de "*likes*",

comentarios y compartidos. Se ve que hay una clara concentración en los valores más pequeños. Además, de que se observa una cierta tendencia lineal. Además de que el número de compartidos suele ser de uno. La media de comentarios y de "*likes*" no suele superar los 40000 y los 0.25 respectivamente.

Por otro lado, se tiene la visualización de la izquierda donde se observa el ranking de suscriptores en representación con la media de visitas. Se observa un cierto engrosamiento para el caso de la media de visitas de 1 con una cuenta de suscriptores de 0.3. Otros casos interesantes que se desarrollan son las altas tendencias de media de visitas que superan el valor de 10 y que se encuentras por encima de 1.4. Para este caso también se observa una cierta relación lineal las variables de "*Views avg*" y "*Suscriptors count*". Se observa que se concentra mucha información en los altos valores.

Figura 13: Relación entre el número de reproducciones y longitud del vídeo para el Washington Post

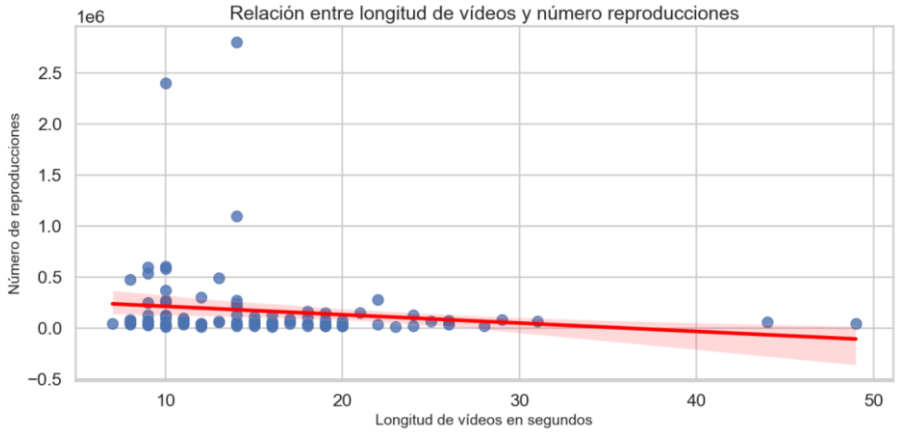


Fuente: Elaboración Propia

Con respecto a la imagen superior, se muestra una relación de la duración de los vídeos con el número de reproducciones para el caso de las bases de datos del "*Washington Post*". Tienen una duración que va desde siete hasta casi cincuenta segundos. Y llegan a alcanzar los 2.5 millones de visualizaciones. Se observa una distribución temporal que va de los 8 a los 14 segundos llegando a una mayor variabilidad en las reproducciones. Se ven, a parte, algunos *outliers* donde se observan gran cantidad de visualizaciones. Por otro lado, cuando

los vídeos son más largos, llegando a los 17 segundos no se alcanzan tan altos niveles de reproducciones y hay menos varianza. Este último caso no llega a ser tan viral. Se ve que los vídeos con menor duración son los que llega a alcanzar mayor atención. Sería bueno enfocarse en vídeos de menos de 15 segundos.

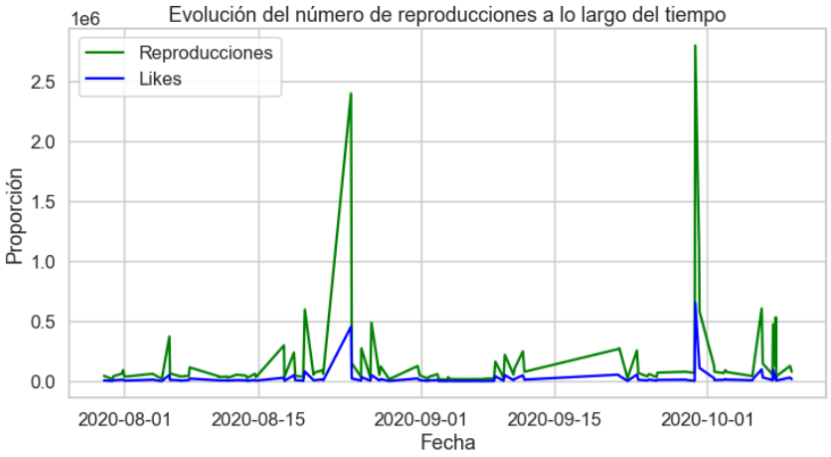
Figura 14: Relación entre el número de reproducciones y longitud del vídeo para el Washington Post



Fuente: Elaboración Propia

La visualización que se muestra en la parte superior también forma parte del "*Washington Post*" y se observa otra vez la relación de la longitud del vídeo con el número de visualizaciones, pero representado sin el gráfico en forma de cuartiles. Se observa una clara relación inversa ya que a medida que la longitud del vídeo aumenta, el número de visualizaciones disminuye. Siendo la mayoría de los casos de hasta 30 segundos y viéndose aquellos vídeos considerados *outliers* que van hasta 25 millones de visualizaciones que corresponden con los que acabo de nombrar. Por otro lado, hay que destacar de esta visualización que la mayoría de las visualizaciones no llegan a alcanzar más de 500 mil visualizaciones.

Figura 15: Evolución del número de "plays" y de reproducciones a lo largo del tiempo para Washington Post



Fuente: Elaboración Propia

En la imagen superior se muestra cómo se distribuye el número de reproducciones y de "*likes*" a lo largo del tiempo. Con respecto a los datos hay que destacar, se observan tres acontecimientos que son. Primero de todo, que el número de reproducciones y de me gusta sigue la misma tendencia. Y que en momentos puntuales como en agosto y octubre del 2020, hay creación de vídeos virales. Estos casos se ven que la reacción en el número de reproducciones es estrictamente mucho más alta.

3.3.2 MODELOS DE MACHINE LEARNING

A continuación, voy a hablar sobre los modelos de *Machine Learning* implementados. La idea es aplicar todos los posibles modelos sobre algunas de las fuentes de datos que se acaban de analizar para poder interpretar toda la información que aparecen sobre ellos.

Con respecto a la base de datos de "*Free TikTok Scrapper*", se trata de un conjunto de datos con información relevante de grupos de variables como "*authorMeta*", "*videoMeta*" que contienen información sobre los seguidores, la verificación y la biografía, ya que son variables que están agrupadas dentro de esas dos categorías. Además, de algunas métricas estadísticas como "*playCount*", "*shareCount*" o "*searchHasthag/views*". Debido al tipo de modelo que se trata, se podría implementar un modelo de regresión para predecir factores como el número de visitas, "*likes*" o compartidos. Para ello sería bueno implementar

modelos cómo el de "**Regresión Lineal"**[1] y "**Random Forest"**[2]. Además, con estas bases de datos también se puede medir el "*engagement*" de los usuarios con la aplicación mediante el uso de variables como "*searchHashtag/name*" o "*searchHastag/views*" donde se podrían aplicar modelos de **"Regresión Logística"** [3]o de **"Support Vector Machine"**[4].

A continuación, voy a desarrollar todos estos modelos con respecto a la base de datos mencionada. El modelo inicial que se desarrolló fue un modelo de regresión lineal que tenía como variable "objetivo" la variable de *"searchHashtag/views"* que representa el número de visitas y se analizará con respecto a variables independientes de contenido cuantificable que son el caso de *"authorMeta/following"*, *"videoMeta/duration"*, *"authorMeta/fans"*, *"commentCount"*. Este primer modelo desarrolla unas primeras medidas muy grandes como el error cuadrático medio o la R cuadrado del 30% que implicaría ser un modelo que explica el 30% de las variables. En la imagen inferior se muestra el modelo final analizado donde se observa que el número de visitas tienen una correlación de carácter positivo con el resto de las características relacionadas con el vídeo.

Figura 16: Modelo Regresión Lineal para "Free TikTok Scrapper"

OLS Regression Results

Dep. Variable:	searchHashtag/views	R-squared:	0.294
Model:	OLS	Adj. R-squared:	0.293
Method:	Least Squares	F-statistic:	399.2
Date:	Sun, 16 Mar 2025	Prob (F-statistic):	1.65e-74
Time:	20:17:17	Log-Likelihood:	-28060.
No. Observations:	960	AIC:	5.612e+04
Df Residuals:	958	BIC:	5.613e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-9.483e+10	4.79e+10	-1.981	0.048	-1.89e+11	-8.9e+08
commentCount	1.056e+07	5.29e+05	19.979	0.000	9.53e+06	1.16e+07

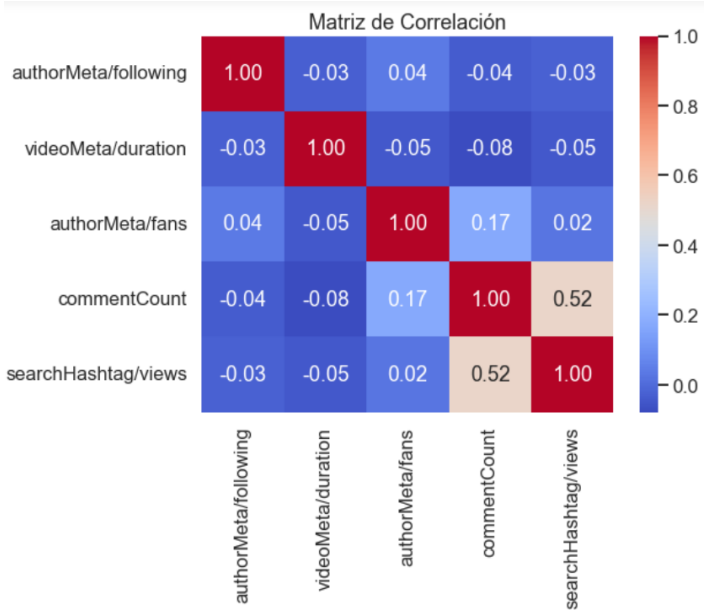
Omnibus:	816.657	Durbin-Watson:	1.952
Prob(Omnibus):	0.000	Jarque-Bera (JB):	27678.217
Skew:	3.721	Prob(JB):	0.00
Kurtosis:	28.230	Cond. No.	1.12e+05

Fuente: Elaboración Propia

[1] Modelo de *machine learning* que tiene como objetivo pasar lo más cerca posible del conjunto de puntos dado (Herbrich y Graepel).
[2] Método de clasificación que se basa en la combinación de diferentes árboles de decisión (Medina-Merino y Ñique-Chacón, 2017).
[3] Predicción de variable categórica a partir de variables independientes (Pérez-Obregón y Romero-Díaz, 2018).
[4] Separación óptima mediante el uso de *kernels* para conseguir transformar los datos (Herbrich y Graepel).

A continuación, voy a desarrollar la correlación entre estas variables para entender bien cómo se relacionan.

Figura 17: Matriz de Correlación para Free TikTok Scrapper



Fuente: Elaboración Propia

Con respecto a la figura superior, se observa una correlación muy elevada entre las variables de "*searchHashtag/views*" y "*commentCount*" con un valor de 0.52 lo cual sugiere que está muy relacionado el número de comentarios con el número de visitas. Esto es una buena señal, porque además no existe una alta correlación entre las variables que se utilizan para predecir. Además, con la figura que representa la correlación con el número de visitas se observa que hay una correlación muy baja con las variables de "*authorMeta/following*" y "*videoMeta/duartion*". Se refuerza la idea de que la viralidad se encuentra influenciada por el número de veces que se comparte un vídeo.

Es interesante probar a únicamente poner la variable que realmente es más explicativa. Para este último modelo se observa que realmente no es un gran modelo en ese sentido. Es por ello por lo que voy a utilizar otros modelos.

Figura 18: Intervalos de Confianza Coeficientes del Modelo

	2.5%	97.5%
const	-1.887641e+11	-8.900651e+08
commentCount	9.525827e+06	1.160100e+07

Fuente: Elaboración Propia

Al analizar el intervalo de confianza para la variable que se ha utilizado para predecir el modelo, se ve que los valores de "*commentCount*" oscilan entre 9.52 para el 2.5% de los casos y por otra parte el 97.5% de los casos ya están en 11,601,000. Esto implica que hay un gran rango de valores a analizar. Se observa un grado alto de incertidumbre. Varía mucho a la hora de impactar en las variables objetivo.

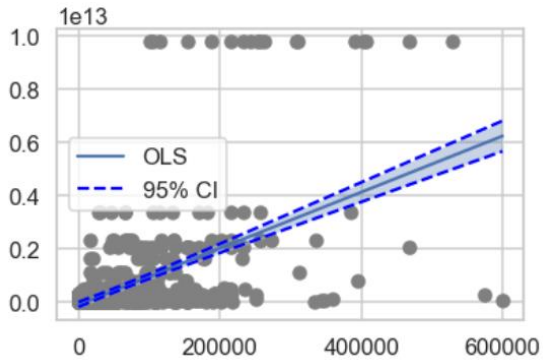
Figura 19: Predicciones Intervalo de Confianza del 95%

	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
230	1.077712e+12	4.922917e+10	9.811023e+11	1.174321e+12	-1.275321e+12	3.430744e+12
1171	6.498935e+11	3.971261e+10	5.719598e+11	7.278273e+11	-1.702446e+12	3.002233e+12
1099	-7.079533e+10	4.716832e+10	-1.633605e+11	2.176983e+10	-2.423665e+12	2.282074e+12
208	8.421476e+11	4.294183e+10	7.578767e+11	9.264185e+11	-1.510410e+12	3.194706e+12

Fuente: Elaboración Propia

En la imagen superior se muestran las predicciones de intervalos de confianza para el caso de la media y muchos otros parámetros. Se observan todas sus métricas. Esto también se puede observar en la imagen inferior. Se observa un alto grado de variabilidad.

Figura 20: Predicciones con intervalos de confianza al 95%



Fuente: Elaboración Propia

Figura 21: Error Cuadrático Medio Final Modelo

```
El error (rmse) de test es: 1047436543456.8076
```

Fuente: Elaboración Propia

El modelo que se aplicó a continuación fue un *Random Forest*. Se aplicó el logaritmo a la variable de visitas para ver si se puede compensar la desigualdad en la distribución de los datos. Al aplicarlo mediante la función de "*log1p*" de *numpy* siendo capaz el modelo de explicar el 44% de los casos. El error cuadrático medio se ha conseguido reducir unas pocas décimas.

Figura 22: Modelo Random Forest "Free TikTok Scrapper"

```
RandomForestRegressor
RandomForestRegressor(max_features=1, n_estimators=10, n_jobs=-1,
                      random_state=123)
```

Fuente: Elaboración Propia

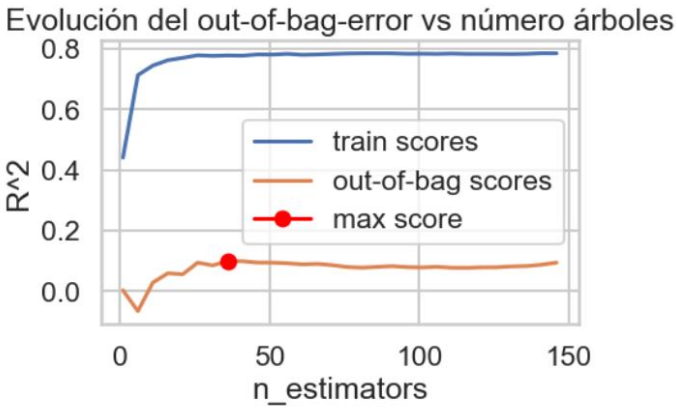
Figura 23: Error Cuadrático Medio Inicial

```
El error (rmse) de test es: 1047436543456.8076
```

Fuente: Elaboración Propia

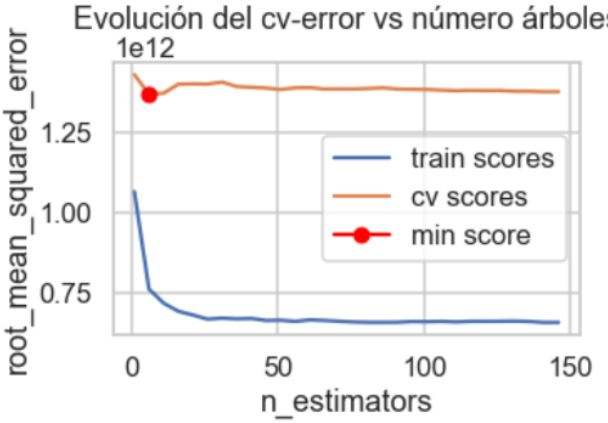
Como se puede observar con la información de la figura superior, se destaca la mejora que se obtiene a la hora de predecir en comparación con la regresión lineal. Consiguiendo llegar a tener una mejor capacidad para explicar las variables del modelo.

Figura 24: Evolución del out-of-bag-error vs número árboles para "Free TikTok Scrapper"



Fuente: Elaboración Propia

Figura 25: Evolución del cv-error vs número de árboles



Fuente: Elaboración Propia

Al haber desarrollado una estimación de la evolución del "*out-of-bag-error*" en comparación con el número de árboles, se observa una tendencia a estabilizarse a partir de los 40 estimadores. Ocurro algo muy parecido al analizar la evolución del "*cv-error*" en comparación con el número de árboles. El error tiende a estabilizarse, siendo el error más bajo para el caso del error cuadrático medio de aproximadamente 1.38.

Figura 26: Grid Search basado en Out-Of-Bag Error

	oob_r2	max_depth	max_features	n_estimators
3	0.270548	3.0	5.0	150.0
4	0.270548	3.0	7.0	150.0
5	0.270548	3.0	9.0	150.0
6	0.109902	10.0	5.0	150.0

Fuente: Elaboración Propia

Se ha podido observar, con las imágenes superiores, la capacidad de estabilización del modelo a partir de un cierto valor para los árboles. Se consigue obtener un valor óptimo para poder desarrollar una mejor predicción sin llegar a sobre ajustar el modelo.

Figura 27: Mejores Hiperparámetros

```
-----------------------------------------
Mejores hiperparámetros encontrados (oob-r2)
-----------------------------------------
oob_r2            0.270548
max_depth         3.000000
max_features      5.000000
n_estimators    150.000000
Name: 3, dtype: float64
```

Fuente: Elaboración Propia

Figura 28: Grid Search basado en Validación Cruzada

	param_max_depth	param_max_features	param_n_estimators	mean_test_score	std_test_score	mean_train_score	std_train_score
3	3	5	150	-1.207997e+12	2.416997e+11	-1.009100e+12	5.637566e+10
4	3	7	150	-1.207997e+12	2.416997e+11	-1.009100e+12	5.637566e+10
5	3	9	150	-1.207997e+12	2.416997e+11	-1.009100e+12	5.637566e+10
6	10	5	150	-1.354119e+12	2.229714e+11	-6.532202e+11	3.741371e+10

Fuente: Elaboración Propia

Figura 29: Mejores Hiperparámetros Encontrados con Validación Cruzada

```
----------------------------------------
Mejores hiperparámetros encontrados (cv)
----------------------------------------
{'max_depth': 3, 'max_features': 5, 'n_estimators': 150} : -1207997
396284.5935 neg_root_mean_squared_error
```

Fuente: Elaboración Propia

Se consigue obtener unos mejores hiperparámetros gracias al uso de "**Grid Search"**[5] y de la validación cruzada. Consiguiendo obtener los valores óptimos para la profundidad del árbol y cuántos estimadores se necesitan para poder maximizar el rendimiento del modelo.

Figura 30: Importancia por Permutación

	importances_mean	importances_std	feature
1	6.378266e+11	2.487248e+10	commentCount
0	0.000000e+00	0.000000e+00	const

Fuente: Elaboración Propia

Se observa mediante la importancia de la permutación, cómo ha ido evolucionando el modelo "**XGBoost"**[6], se observa una mejora en la capacidad de predecir en comparación con versiones anteriores. Realizando un mejor ajuste de las variables.

Figura 31: Aplicación logarítmica a modelo de Random Forest

```
MSE: 4.482369931013603
MAE: 1.530697822740772
RMSE: 2.1171608184107327
R² Score: 0.3536934942458617
MAPE: 0.06325093959029106
```

Fuente: Elaboración Propia

Al realizar una aplicación logarítmica, se logra una capacidad de predicción con valores de errores mucho más bajos. Lo cual implica un mejor ajuste para el modelo. Esto también se

[5] Técnica utilizada para la definición de hiperparámetros del algoritmo SVR (Ríos et al., 2019).
[6] Se trata del método de 'Extreme Gradient Boosting' y se trata de un método mucho más eficiente que Gradient Boosting por tener mayor diversidad y obtener datos dispersos (Chen y He, 2025)

puede observar al evaluar el valor de los parámetros de "**R2"** [7]para las distintas variables. Que se puede observar en la imagen inferior.

Figura 32: R2 scores

```
[-0.55918004  0.35714339 -0.49393791 -0.05892484 -0.14523253]
```

Fuente: Elaboración Propia

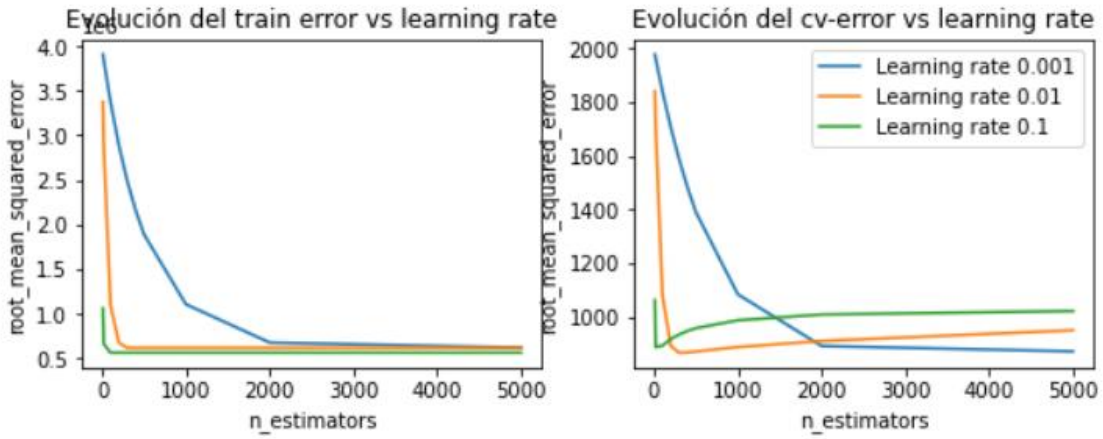
A continuación, voy a hablar sobre la aplicación de un modelo de **"Gradient Boosting"** [8] a la base de datos de "*TikTok Dataset*".

Figura 33: Hist Gradient Boosting Regressor

```
HistGradientBoostingRegressor
HistGradientBoostingRegressor(max_iter=10, random_state=123)
```

Fuente: Elaboración Propia

Figura 34: Evolución Learning Rate del modelo



Fuente: Elaboración Propia

[7] Proporción explicación de variable dependiente que es explicada por la variación de variable independiente para el caso del modelo de regresión lineal (Figueiredo et al., 2011).
[8] Aprendizaje automático para modelos muy complejos y personalizables (Natekin y Knoll, 2013)

En la imagen superior, se observa una evolución de los hiperparámetros del modelo y se muestran posibles valores óptimos. Mostrando la relación entre el número de estimadores y el error cuadrático medio. Se destaca que un posible valor óptimo para la ratio de aprendizaje puede ser de 0.01 y cómo número de estimadores unos 200. Esto mismo se puede ver observado en las siguientes tres figuras donde se analiza los posibles valores óptimos donde se indican estos mismos valores aproximadamente de los que se acaba de hablar.

Figura 35: Tabla del "Grid Search" basado en validación cruzada

	param_l2_regularization	param_learning_rate	param_loss	param_max_depth	mean_test_score	std_test_score	mean_train_score	std_train_score
56	10	0.01	squared_error	3	-857.491929	24.579890	-815.916680	15.592269
8	0	0.01	squared_error	3	-857.552980	24.635263	-815.741472	15.418526
32	1	0.01	squared_error	3	-857.773981	24.389896	-816.501592	15.982709
40	1	0.10	squared_error	3	-857.921449	25.247504	-809.852537	13.493479

Fuente: Elaboración Propia

Figura 36: Mejores hiperparámetros encontrados (cv)

```
----------------------------------------
Mejores hiperparámetros encontrados (cv)
----------------------------------------
{'l2_regularization': 10, 'learning_rate': 0.01, 'loss': 'squared_e
rror', 'max_depth': 3} : -857.4919287289699 neg_root_mean_squared_e
rror
```

Fuente: Elaboración Propia

Figura 37: Número de árboles del modelo

```
Número de árboles del modelo: 346
```

Fuente: Elaboración Propia

Figura 38: Importancia de cada variable

	importances_mean	importances_std	feature
2	942.693120	5.177588	video_comment_count
1	640.778355	3.330704	video_like_count
3	4.740643	0.086670	video_view_count
0	1.455521	0.371182	video_duration_sec

Fuente: Elaboración Propia

Figura 39: Hiperparámetros Evaluados

	param_booster	param_learning_rate	param_max_depth	param_subsample	mean_test_score	std_test_score	mean_train_score	std_train_score
17	gbtree	0.01	3	1.0	-857.748174	56.838359	-824.932034	27.978529
29	gbtree	0.10	3	1.0	-859.788490	56.161838	-828.675077	29.619767
28	gbtree	0.10	3	0.5	-860.186494	56.631548	-829.866055	26.411415
16	gbtree	0.01	3	0.5	-860.295496	57.599785	-831.566388	27.144687

Fuente: Elaboración Propia

Figura 40: Error (RMSE) de prueba

```
----------------------------------------
Mejores hiperparámetros encontrados (cv)
----------------------------------------
{'booster': 'gbtree', 'learning_rate': 0.01, 'max_depth': 3, 'subsa
mple': 1} : -857.7481736418819 neg_root_mean_squared_error
Número de árboles incluidos en el modelo: 329
```

Fuente: Elaboración Propia

Figura 41: LightGBM

	param_boosting_type	param_learning_rate	param_max_depth	param_n_estimators	param_subsample	mean_test_score	std_test_score	mean_train_score	std_train_score
67	gbdt	0.010	3	500	1.0	-855.827074	51.107919	-809.038531	22.951554
66	gbdt	0.010	3	500	0.5	-855.827074	51.107919	-809.038531	22.951554
23	gbdt	0.001	3	5000	1.0	-856.032138	51.051182	-808.769961	22.753686
22	gbdt	0.001	3	5000	0.5	-856.032138	51.051182	-808.769961	22.753686

Fuente: Elaboración Propia

Al aplicar el modelo de "**LightGBM**", se observan unos resultados muy similares a los mostrados anteriormente. Además de una mayor eficiencia a nivel computacional. Donde se consigue manejar mejor grandes volúmenes de datos. Con la aplicación de unos mejores hiperparámetros.

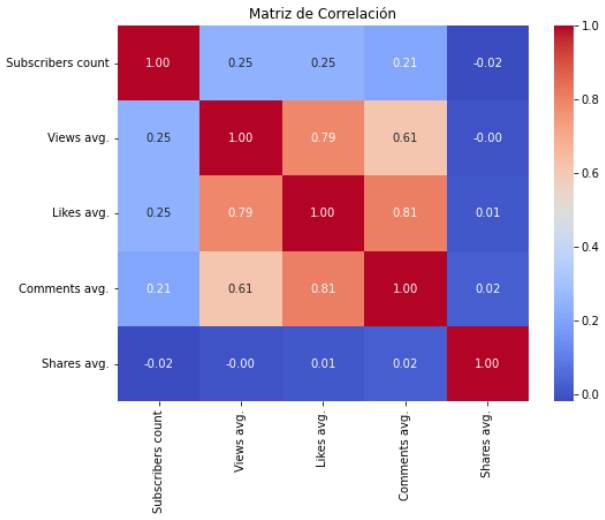
Figura 42: Mejores hiperparámetros encontrados

```
----------------------------------------
Mejores hiperparámetros encontrados (cv)
----------------------------------------
{'boosting_type': 'gbdt', 'learning_rate': 0.01, 'max_depth': 3, 'n
_estimators': 500, 'subsample': 0.5} : -855.8270735757934 neg_root_
mean_squared_error
```

Fuente: Elaboración Propia

Con respecto a la base datos de "*TikTok Top 1000*", lo primero que se ha analizado es la matriz de correlación entre variables.

Figura 43: Matriz correlación para TikTok Top 1000

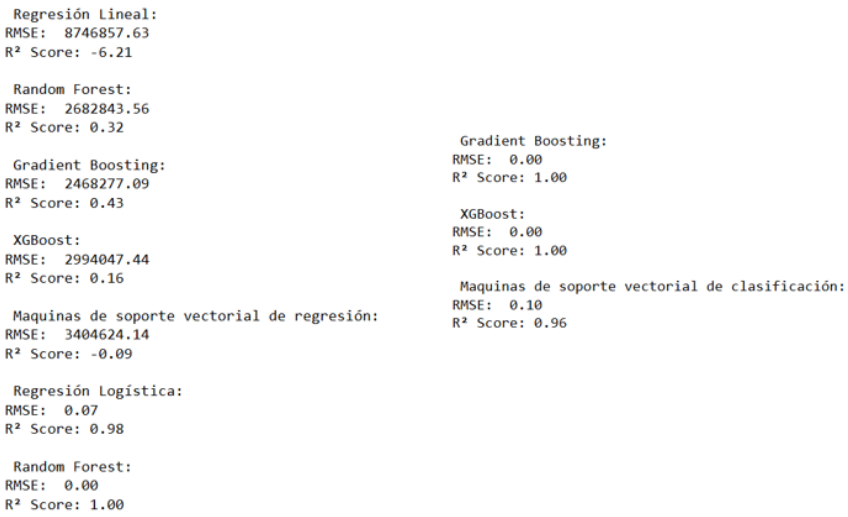


Fuente: Elaboración Propia

Con respecto a la correlación de variables para este dataset, se observa cierta correlación interesante entre algunas variables como son la media entre visualizaciones, "*likes*" y comentarios. Se ve una relación muy importante entre estas tres variables. Implicando una muy alta correlación. Debido a la distribución de diferentes variables, es posible aplicar diferentes tipos de modelos. Es por ello por lo que se puede un modelo de regresión donde se intenta predecir el número de visualizaciones (*"Views avg.*") a partir de algunas de las variables como son la cuenta del número de suscriptores, la medía de "*likes*", la media de comentarios y la media del número de compartidos. Siendo todas ellas variables de carácter continuo. Para este mismo conjunto de datos se van a llevar a cabo diferentes modelos de

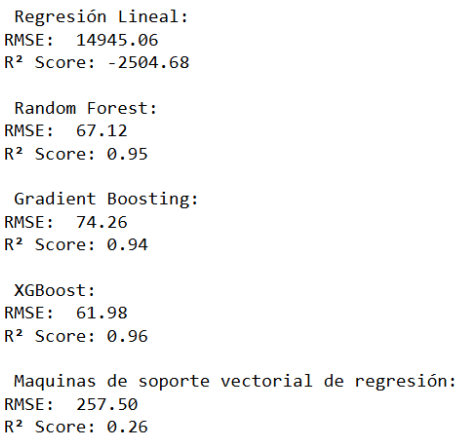
carácter de regresión como son el modelo de regresión lineal, un modelo *random forest*, un modelo *gradient boosting*, un *XGBoost* y un vector de soporte de regresión. Con un conjunto de 100 regresores y cómo parámetro de random estate el número 42. Para todos ellos se sacó la métrica el error cuadrático medio y el *R2*. En todos ellos se obtuvo un valor de *R2* muy parecido que encontraba en un rango de entre 0.17 a 0.19 de ratio de explicación. Llegando a explicar en torno a un 20% de las variables.

Figura 44: Modelos para "TikTok Top 1000"



Fuente: Elaboración Propia

Figura 45: Modelo mejorado para TikTok Top 1000



Fuente: Elaboración Propia

A continuación, procedí a realizar un modelo de clasificación para predecir si se fuese a tener muchos seguidores o no. Siendo muchos seguidores por encima de 5 millones. Creando una variable de carácter binario siendo 1 si la cuenta de suscriptores está por encima de este número. Las variables que se utilizaron como variables independientes fueron la media de las visitas, la media de los "*likes*", la media de los comentarios o la media de los compartidos. Los modelos de clasificación que se han utilizado en este caso son modelos *de regresión logística, random forest, gradient boosting, XGBoost* y *SVM*. Todos ellos consiguiendo un valor de *accuracy* relativamente bueno ya que conseguía explicar en torno a un 60% de los casos. Siendo el mejor modelo el de máquinas de soporte virtual con una precisión del 64%.

Para poder llevar a cabo un análisis de variables más precisos para este conjunto de datos. Lo que hice fue aplicar la transformación logarítmica para algunas variables que podían no estar siendo tan bien capturadas por el sesgo. Este es el caso de la cuenta de los suscriptores y la media de las visualizaciones.

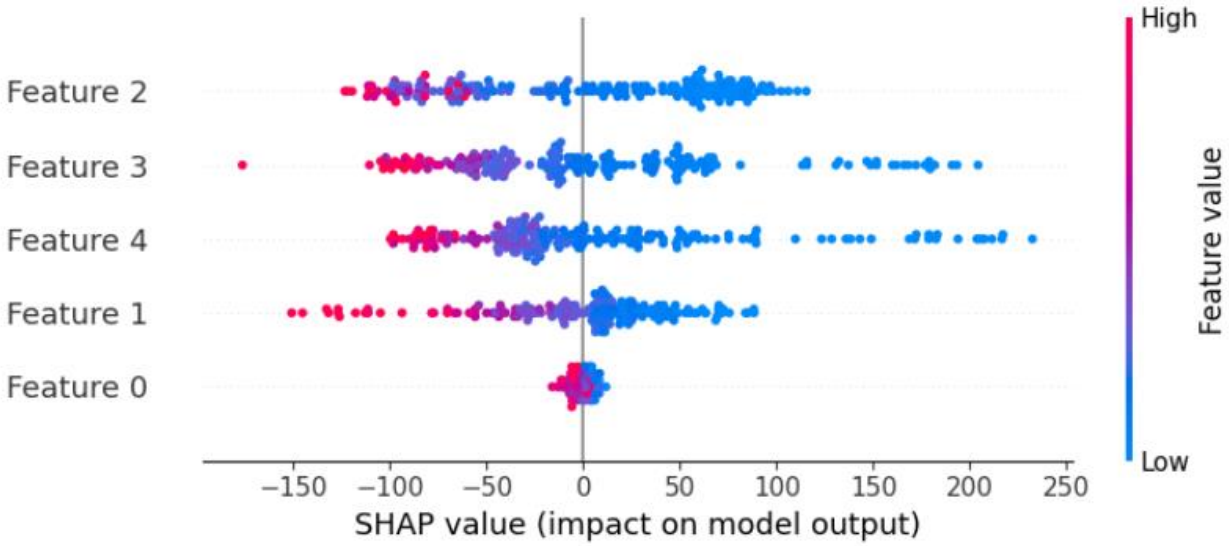
Además, he creado nuevas variables como el "*Engagement Rate*" que hace una media ponderada que agrupa la media de "me gustas", comentarios y compartidos entre la suma de los suscriptores. Además de otras métricas como la ratio de viralización que hace la división de la media de compartidos por la media de visualizaciones. Otra métrica creada sería la interacción por seguidor que es equivalente a la suma de la media de "me gustas" con la suma de la media de comentarios dividido por la cuenta de los suscriptores.

El objetivo ahora va a ser utilizar como variable objetivo la variable "*Rank*". Es por ello por lo que únicamente se van a utilizar variables numéricas para llevar a cabo todos los modelos. Los modelos analizados fueran un *random forest, gradient boosting, SVR* y *XGBoost*. Todos ellos consiguieron alcanzar un *R2* mucho más elevado llegando a alcanzar el 96% de precisión a la hora de explicar el modelo. Todas ellas menos en el caso del *SVR*. Esto se ve observado en la figura 45. Ocurre una tendencia muy parecida para los parámetros en el *RMSE* donde se observan unos valores también muy parecidos que se encuentran entre el 120 y el 140. Se desvía un poco para el caso del *SVR*. Siendo el modelo con mejores prácticas

el *XGBoost*, seguido del *Random Forest* y el *Gradient Boosting* con unos valores muy similares. Se conservan valores muy bajos de error cuadrático medio.

A continuación, llevé a cabo un análisis de **"SHAP"** [9]para el mejor modelo que es para este caso el *XGBoost*.

Figura 46: Análisis SHAP XGBoost



Fuente: Elaboración Propia

El gráfico de *SHAP* que se muestra en la parte superior representa cómo de importante y qué impacto tienen cada una de las características. La más influyente de las características fue la tercera mientras que la menos fue la primera. El color representa, por una parte, que la característica tiene un valor alto de observación para el caso del rojo y un valor bajo de observación para el caso del azul.

El gráfico muestra algunas características tienen más impacto sobre el modelo que otras. Llegando a alcanzar valores que van aumentando progresivamente. Se ve claramente que algunas llegan a afectar de una manera más dispersa al modelo. Ya que el *SHAP* sirve para

[9] *Shapley Additive Explanations*: marco explicativo sobre modelos de aprendizaje automático (Mosca, 2022).

observar de qué manera cambia la predicción del modelo teniendo en cuenta o no las diferentes características.

3.4 RESULTADOS

En este apartado se va a desplegar información relevante sobre el análisis que se acaba de llevar a cabo. Por una parte, se va a analizar las características más relevantes que se han observado en las visualizaciones ejecutadas. Para, a continuación, mostrar una tabla a modo de resumen con todos los modelos y métricas utilizadas.

Gracias al análisis llevado a cabo en la implementación de visualizaciones para poder entender mejor el comportamiento de la información de las distintas bases de datos utilizadas, se pudo detectar la distribución de la normal centrada en torno a los 500 píxeles que se veía en la figura 1. Y que, además, se llega a observar otra en torno a los 600 píxeles de ancho de banda.

Se observa que la tendencia entre los "*likes*" y los compartidos que se puede ver en la figura 2. Y que se sigue una tendencia muy parecida: bajo el mismo patrón pero que el número de "*likes*" lo hace con una magnitud claramente mayor que el número de veces que se compartía, esto se puede observar al ver la reacción de la aprobación en la ley de "*Ómnibus*" de ese año.

Aquí también, se pudo apreciar que se tenía una relación lineal muy directa entre diferentes parámetros interesantes como el número de "*likes*", el número de reproducciones y el número de comentarios. Donde los valores suelen estar concentrados en los mismos rangos. Destacando la idea de los puntos con mayor magnitud son aquellos con mayor concentración de "*likes*", compartidos, reproducciones y comentarios. Generando mayor capacidad de interacción sobre el contenido.

Al analizar diferentes estadísticos, que se mostraban en la figura 3 y la figura 4, las diferentes bases de datos como "*Free TikTok Scrapper*" y "*Omnibuslaw Videos*": se observa una clara relevancia en las variables que medían el número de reproducciones, de compartidos y la duración del vídeo. En la figura 5 se puede apreciar que el número de compartidos y de

reproducciones están muy relacionadas. Se observa una relación inversa entre la duración del vídeo y el número de reproducciones.

En la figura 6, se puede ver una clara relación muy alta entre el número de compartidos y de comentarios, y el número de comentarios y de reproducciones. Se observa em la figura 7, también, que la distribución del número de compartidos y de reproducciones tiene un sesgo hacia la derecha. Este comportamiento también es muy similar entre estos dos parámetros. Sin embargo, la frecuencia del número de “likes” y de comentarios no sigue esta misma tendencia. Destacando que el número de comentarios sí que suele seguir una tendencia más constante como se puede ver en la figura 8.

Hay una relación muy constante con respecto a la duración de los vídeos: es muy parecida en la mayoría de los casos. Se observa una clara relación lineal entre el número de visualizaciones, "*likes*" y descargas. Algo muy parecido se puede ver al comparar el número de cometarios, de "*likes*" y de compartidos. Con la misma relación lineal detectada. Esto se puede ver reflejado en la figura 10. Donde al comparar la duración del vídeo en segundos con el número de visualizaciones, se observa una relación estable ya que el número de visualizaciones se mantiene constante a lo largo de los diferentes valores de duración del vídeo.

En la figura 11 se ve reflejado que el número de personas suscritas, que para este caso concreto correspondería al número de seguidores en las cuentas sigue las mismas tendencias en la distribución normal que se podía observar en la distribución del número de compartidos y de reproducciones. Destacando que la cantidad de seguidores no llega a ser nunca de una gran magnitud. El éxito en *TikTok* no se encuentra en tener un gran número de seguidores. No se observa una relación muy fuerte entre el número de seguidores y el número de visitas.

En la figura 12 se vuelve a observar el patrón en la concentración de valores: en este caso en concreto para "*likes*" y comentarios. Por otro lado, se ve una frecuencia más grande para los valores de "*Views avg*" más altos. Se puede detectar que a medida que la duración del vídeo es menor, el número de visitas aumenta.

Donde la mayoría de los vídeos se concentraban en valores más bajos de duración en las reproducciones de los vídeos. Por último, se observa en la figura 15 el mismo patrón de

comportamiento entre "*likes*" y reproducciones que se veía también en la figura 2. Sin embargo, aquí se compararon dos variables distintas: teniendo mayor reacción en las reproducciones que en el número de "*likes*". Se puede decir que parte de la capacidad de generar "*engagement*", está relacionado con el nivel de actividad y de comunidad que sea capaz de generar el creador de contenido.

A continuación, se muestra una tabla a modo de resumen con todos los modelos de *Machine Learning* que se han aplicado:

Tabla 1: Resumen Modelo de Machine Learning Aplicados

Base de datos	**Modelo de Machine Learning**	**Métricas de precisión**	**Observaciones**
Free TikTok Scrapper	Regresión Lineal	R2=0.293	Predicción de la relación de visitas por búsqueda en "*hashtags*" en función de la cuenta de comentarios.
Free TikTok Scrapper	*Random Forest Regressor*	R2=0.35	Predicción de la relación de visitas por búsqueda en "*hashtags*" en función de la cuenta de comentarios.

TikTok Dataset	*Histogram Gradient Boosting Classifier*	Puntuación en prueba estándar: 51.1.	Predicción del número de descargas de vídeos en función de la duración de los vídeos, el número de "*likes*" por vídeo, el número de comentarios y el número de visitas.
TikTok Top 1000	SVC	R2=0.96	Predicción del número de visitas en función del número de suscriptores, "*likes*", cometarios y "*shares*".
TikTok Top 1000	*Random Forest*	R2= 0.95	Predicción de la variable "*Rank*" en función del *engagement rate*, "*viral ratio*" y la interacción por parte del usuario.
TikTok Top 1000	*Gradient Boosting*	R2= 0.94	Predicción de la variable "*Rank*" en función del engagement rate, "*viral ratio*" y la

			interacción por parte del usuario.
TikTok Top 1000	*XGBoost*	R2=0.96	Predicción de la variable "*Rank*" en función del *engagement rate*, "*viral ratio*" y la interacción por parte del usuario.
TikTok Top 1000	SVR	R2=0.26	Predicción de la variable "*Rank*" en función del *engagement rate*, "*viral ratio*" y la interacción por parte del usuario.

Fuente: Elaboración propia

Después de haber aplicado todos estos diferentes modelos de *Machine Learning*, se observa una correlación positiva entre el número de visitas y el número de comentarios. Además de la alta correlación entre el número de compartidos y el número de visitas. Lo cual tiene mucho sentido. Destacando que para poder encontrar mayor "*engagement*" y viralidad, es necesario desarrollar una mayor capacidad de compartidos. Aportando además una gran variabilidad entre el número de visualizaciones que se puede ver reflejado en el intervalo de confianza desarrollado.

Hay que destacar que los modelos de mejor rendimiento fueron "*XGBoost*" para la predicción de la variable "*Rank*". Seguidos de los modelos de "*Random Forest*" y de "*Gradient Boosting*" donde se obtuvieron valores de *R2* también muy altos llegando a ser

por encima del 90%. Se puede percibir la idea de que los modelos que se basan los árboles tienden a entender mejor las relaciones que existen entre los datos.

Ya que se consigue un error mucho más estable gracias al uso de un número de árboles más adecuado para entrenar el modelo. Esto puede ayudar a conseguir un modelo mejor sin necesidad de sobre ajustar el modelo. Y tras aplicarle a este mismo modelo el análisis de *SHAP* para la base de datos de "*TikTok Top 1000*", se detecta que el número de "*likes*" es de las variables más importantes a la hora de predecir el "*engagement*" que se genera en el usuario.

Capítulo 4. CONCLUSIÓN

Después de haber llevado a cabo un trabajo de investigación y de análisis de diferentes fuentes de datos donde se aplicaron diversos modelos de *machine learning*. Se ha conseguido entender el funcionamiento del "*engagement*" en la aplicación de *TikTok*. Además, de identificar la manera de ver el impacto que tiene el nivel de viralidad del contenido que se ve en la plataforma. Donde se ha demostrado lo importante que es que se consiga personalizar el algoritmo a la hora de retener usuarios. Se tiene que poder generar un entorno en que los intereses individuales se puedan ver reforzados gracias al contenido considerado relevante.

A través del análisis de las visualizaciones generadas, pude identificar una serie de patrones y de comportamientos que se repetían en diferentes conjuntos de datos. Entre ellos se encuentran que hay ciertas variables que sigue una misma distribución normal a la hora de analizar su frecuencia de aparición. Entre estas variables se encuentran los suscriptores, "*likes*", número de reproducciones y número de compartidos. Además de la relación lineal entre visualizaciones, "*likes*", descargas, comentarios y compartidos. Se ha podido ver que existía una concentración de valores. Además de que se ha podido ver que todas las variables de interacción siempre siguen el mismo comportamiento. Donde las que tienen un mayor nivel de reacción son los "*likes*" y las reproducciones.

Entre los principales hallazgos que he obtenido de este análisis, se encuentra la fuerte correlación entre el número de interacciones, ya sea el número de "*likes*", compartidos o cometarios con respecto a la visibilidad que tiene ese contenido. Además, se ha conseguido demostrar lo efectiva que es la estrategia de utilizar "*hashtags*" para conseguir un mayor alcance en los vídeos. Consiguiendo así que llegue a alcanzarse otras audiencias que consuman contenido del mismo estilo. Es por ello por lo que, gracias al uso de los *hashtags*, se ha conseguido llegar a un público más grande. Dónde, además, se ha demostrado la importancia de qué se intenta expresar mediante el uso del "*hashtag*" para poder crear comunidades.

Se ha desarrollado también una comparación en las interacciones del contenido. Teniendo en cuenta el número de seguidores y sin tenerlo en cuenta. Se observa que, aunque no haya sido un factor de los más determinante a la hora de estimar la relación con el usuario. Si que ha favorecido a que se mantenga una relación más sólida con un número más grande de usuarios. Ya que esto también favorece a lo que se ha mencionado a lo largo de este proyecto que se conoce como el fenómeno del "boca o boca" que se produce entre diferentes usuarios que consumen productos similares. Lo cual hace que se potencie una mejor propagación de los vídeos.

Con respecto al análisis de la duración de los vídeos, se ha conseguido demostrar que exista una relación inversamente proporcional entre la duración de estos y el número de reproducciones. Consiguiendo captar mayor atención por parte de los usuarios en vídeos más cortos. Lo cual hace que se favorezca a una mayor tasa de retención por parte de los usuarios. Además de que se obtenga mayor número de visualizaciones.

Además, hay que destacar que, para poder llegar a conseguir un mayor nivel de compromiso por parte del usuario, es muy importante que sienta una confianza por parte del recíproco. Y que esto se consigue en parte gracias a la generación de contenido específico que hace que se generen "nichos" de contenido para usuarios. Esto se ha demostrado a través de la creación de contenido específico donde se lograba un mayor nivel de "*engagement*". Y que se apoyaba en el uso de *hashtags*.

Con respecto a los modelos de *Machine Learning* que se han utilizado para analizar los datos, se obtuvo que los mejores modelos eran los árboles de decisión tales como "*XGBoost*" o *"Random Forest"* que conseguían explicar más del 90% de las variables. Y gracias a ellos se puedo demostrar que las variables más influyentes a la hora de predecir el "*engagement*" eran el número de "*likes*" y el número de compartidos.

Es por ello por lo que se puede concluir con que, para poder optimizar la herramienta para conseguir una mayor "*engagement*", se debe de enfocar en un uso de la aplicación manteniendo el enfoque en diversos temas fundamentales. Donde entre ellos se encuentra la magnitud de las interacciones y la capacidad la personalización en el algoritmo. Además de

la duración del vídeo. Dándole especial importancia al tema del uso de "*hashtags*" ya que es clave para poder desarrollar una estrategia efectiva a la hora de querer crear un sentimiento de comunidad en la red social. Se puede resumir con que lo más importante para generar un mayor engagement es que el contenido de la plataforma esté personalizado para el usuario en sí.

*** Declaración de Uso de Herramientas de Inteligencia Artificial Generativa en Trabajos Fin de Grado**

ADVERTENCIA: Desde la Universidad consideramos que ChatGPT u otras herramientas similares son herramientas muy útiles en la vida académica, aunque su uso queda siempre bajo la responsabilidad del alumno, puesto que las respuestas que proporciona pueden no ser veraces. En este sentido, NO está permitido su uso en la elaboración del Trabajo fin de Grado para generar código porque estas herramientas no son fiables en esa tarea. Aunque el código funcione, no hay garantías de que metodológicamente sea correcto, y es altamente probable que no lo sea.

Por la presente, yo, Pilar Folqué Brier, estudiante de 5º de Ingeniería en Tecnologías de las Telecomunicaciones y Business Analytics de la Universidad Pontificia Comillas al presentar mi Trabajo Fin de Grado titulado "LOVE BRAND: LA PERSONALIZACIÓN DE LA MARCA Y ÉXITO DEL ENGAGEMENT PARA EL CASO DE TIKTOK", declaro que he utilizado la herramienta de Inteligencia Artificial Generativa ChatGPT u otras similares de IAG de código sólo en el contexto de las actividades descritas a continuación:

1. **Brainstorming de ideas de investigación**: Utilizado para idear y esbozar posibles áreas de investigación.
2. **Referencias**: Usado juntamente con otras herramientas, como Science, para identificar referencias preliminares que luego he contrastado y validado.
3. **Metodólogo**: Para descubrir métodos aplicables a problemas específicos de investigación.
4. **Interpretador de código**: Para realizar análisis de datos preliminares.
5. **Corrector de estilo literario y de lenguaje**: Para mejorar la calidad lingüística y estilística del texto.
6. **Sintetizador y divulgador de libros complicados**: Para resumir y comprender literatura compleja.
7. **Revisor**: Para recibir sugerencias sobre cómo mejorar y perfeccionar el trabajo con diferentes niveles de exigencia.
8. **Traductor**: Para traducir textos de un lenguaje a otro.

Afirmo que toda la información y contenido presentados en este trabajo son producto de mi investigación y esfuerzo individual, excepto donde se ha indicado lo contrario y se han dado los créditos correspondientes (he incluido las referencias adecuadas en el TFG y he explicitado para que se ha usado ChatGPT u otras herramientas similares). Soy consciente de las implicaciones académicas y éticas de presentar un trabajo no original y acepto las consecuencias de cualquier violación a esta declaración.

Fecha: abril 2025

Firma: Pilar Folqué Brier

Capítulo 5. BIBLIOGRAFÍA

[1] Amat, J. (2020, octubre). *Gradient Boosting con Python.* Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py09_gradient_boosting_python

[2] Amat, J. (2020, octubre). *Random Forest con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py08_random_forest_python

[3] Amat, J. (2020, octubre). *Regresión lineal con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py10-regresion-lineal-python

[4] Bergkvist, L., & Bech-Larsen, T. (2010). Two studies of consequences and actionable antecedents of brand love. *Journal of Brand Management*, 17 504-518.

[5] Bolton, R.N. (2012). Customer Engagement: Opportunities and Challenges for Organizations. *Journal of Service Research* 14(3): 272-274.

[6] Chen, T. & He, T. (2025, 26 de marzo) xgboost: eXtreme Gradient Boosting.

[7] Figueiredo, D., Junior, S., & Rocha, E. (2011, noviembre) Wha t is R2 all about?

[8] García, M. (2024, 14 de octubre). Nike Swoosh, la historia del logo más famoso del mundo. *Brandemia*. Recuperado de https://brandemia.org/nike-la-historia-del-logo-mas-famoso-del-mundo

[9] Herbrich, R., & Graepel T. Chapman & Hall/CRC Machine Learning & Pattern Recognition Series. *CRC Press*. Cambridge, UK.

[10] Hosteltur. (2024, 14 de octubre) Cómo convertirse en una "*Love Brand*" y no morir en el intento. *Hostertur*. Recuperado de https://www.hosteltur.com/166136_como-convertirse-en-una-love-brand-y-no-morir-en-el-intento.html

[11] Huseyn, R. (2023) Dataset from TikTok. TikTok Dataset for EDA, Statistical and Predictive Modelling. *Kaggle*. Recuperado de https://www.kaggle.com/datasets/raminhuseyn/dataset-from-tiktok

[12] Kanawattanachai, P. Top 1000 TikTok Influencers Ranking: Explore top 1,000 TikTok Influencers. *Kaggle*. Recuperado de https://www.kaggle.com/datasets/prasertk/top-1000-tiktok-influencers-ranking/data

[13] Medina-Merino, R.F., & Ñique-Chacón C.I. (2017, 23 de septiembre) Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python. *Instituto Nacional de Estadística e Informática*. Lima, Perú.

[14] Moon, Y. & Quelch, J. (2005, 14 de julio) Starbucks*:* Brindando servicios al cliente. *Harvard Business School*.

[15] Mosca, E., Szigeti, F., Tragianni, S., Gallagher, D., & Groh, G. (2022, octubre) SHAP-Based Explanation Methods: A Review for NLP Interpretability. *International Committee on Computational Liguistics*.

[16] Muhammad, M.A. (2023) TikTok Dataset: TikTok popular hashtags dataset. *Kaggl*e. Recuperado de https://www.kaggle.com/datasets/muhammadanasmahmood/tiktok-dataset

[17] Natekin, A., & Knoll A. (2013, 4 de diciembre) Gradient boosting machines, a tutorial. *Department of Informatics, Technical University Munich, Garching, Munich, Germany*.

[18] Park, C.W.; Macinnis, D.J., & Priester, J. (2006): Brand attachment: Constructs, consequences, and causes. *Foundations and Trends in Marketing*, Vol. 1, nº 3, pp. 191-230.

[19] Peña-Fernández, S., Larrondo-Ureta, A., & Morales-i-Gras, J. (2022). Current affairs on TikTok. Virality and entertainment for digital natives. *Profesional de la información*, v. 31, n. 1, e310106.

[20] Pérez-Obregón, J., & Romero-Díaz, T. (2018). Análisis del rendimiento académico mediante regresión logística y múltiple. *Revista Electrónica de Conocimientos, Saberes y Prácticas,* 1(2), 33-42.

[21] Ríos, J., Ulla G., & Borello A. (2019, octubre) Aplicación de Regresión con Vectores de Soporte en un Sistema Recomendador de Actividades Sociales. *Congreso Argentino de Ciencias de la Computación. Inteligencia Artificial, Departamento de Ingeniería en Sistemas de Información, Universidad Tecnológica Nacional, Facultad Regional Córdoba*.

[22] Rodrigues, C., & Brandão, A. (2020). Measuring the effects of retail brand experiences and brand love on word of mouth: a cross-country study of IKEA brand. *The International Review of Retail, Distribution and Consumer Research*, *31*(1), 78–105.

[23] Secretaría de estado de comercio. Ministerio de economía, comercio y empresa. (2025, 17 de mayo). *Barrera al comercio: Limitaciones a la Inversión*. Recuperado de https://barrerascomerciales.comercio.gob.es/es-es/paises/paginas/paises-barrera.aspx?a=669&b=351

[24] Sidorenko-Bautista, P., Lacasa, P., & Matsumoto, M. (2024). La inteligencia artificial (IA) en TikTok: la plataforma habla sobre su uso [Artificial intelligence (AI) on TikTok: Theo platform talks about its use]. *Infonomy*, 2(3) e24040.

[25] The Coca-Cola company. *The History of the Coca-Cola Contour Bottle. The Creation of a Cultural Icon*. Recuperado de https://www.coca-colacompany.com/about-us/history/the-history-of-the-coca-cola-contour-bottle

[26] The Devastator. TikTok: What's trending and why? A dataset for studying user preferences in social media. *Kaggle*. Recuperado de https://www.kaggle.com/datasets/thedevastator/tiktok-what-s-trending-and-why

[27] The Devastator. TikTok: What's trending and why? A dataset for studying user preferences in social media. *Kaggle*. Recuperado de https://www.kaggle.com/datasets/thedevastator/tiktok-what-s-trending-and-why?select=washingtonpost_videos.csv

[28] Van De Ven, E. (2020) TikTok Trending Videos: First 1000 trending videos scrapped from TikTok. *Kaggle*. Recuperado de https://www.kaggle.com/datasets/erikvdven/tiktok-trending-december-2020/data

[29] Vombatkere K., Mousavi S, Zannettou S., Roesner F., & Gummadi P. (2024, 13-17 de mayo). TikTok and the Art of Personalization: Investigating Exploration and Exploitation on Social Media Feeds. *Proceedings of the ACM Web Conference 2024 (WWW '24),* Singapore, Singapore. ACM, New York, NY, USA, 9 páginas.

[30] Wang, X., & Guo, Y. (2023). Motivations on TikTok addiction: The moderating role of algorithm awareness on young people. *Profesional de la información*, v. 32, n. 4, e320411.

Capítulo 6. ANEXO

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
from wordcloud import WordCloud
import plotly.express as px
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,
silhouette_score, root_mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import classification_report,
confusion_matrix,ConfusionMatrixDisplay
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
FunctionTransformer, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor,
RandomForestClassifier, GradientBoostingRegressor,
GradientBoostingClassifier
from sklearn.impute import SimpleImputer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.model_selection import RepeatedKFold,ParameterGrid
from sklearn.inspection import permutation_importance
from sklearn.feature_selection import SelectKBest, f_regression
from xgboost import XGBRegressor
import shap
from mpl_toolkits.mplot3d import Axes3D
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.svm import SVC,SVR
from collections import Counter
import statsmodels.api as sm
from sklearn.model_selection import cross_val_score
import multiprocessing
from lightgbm.sklearn import LGBMRegressor
from joblib import Parallel, delayed
from multiprocessing import cpu_count
free_tiktok_scraper = pd.read_excel('./dataset_free-tiktok-scraper_2022-
07-27_21-44-20-266.xlsx')
omnibuslaw_videos = pd.read_csv('./omnibuslaw_videos.csv')
tiktok_dataset = pd.read_csv('./tiktok_dataset.csv')
tiktok_top_1000 = pd.read_csv('./tiktok_top_1000.csv')
trending = pd.read_csv('./trending.csv')
washingtonpost_videos = pd.read_csv( './washingtonpost_videos.csv')
sns.set(style="whitegrid")
```

```
# Histograma
plt.figure(figsize=(10, 6))
sns.histplot(free_tiktok_scraper['videoMeta/width'], bins=10, kde=True,
color='blue')
plt.title('Distribución del ancho de la duración de los vídeos de free
tiktok scrapper')
plt.xlabel('Proporción en píxeles')
plt.ylabel('Frecuencia')
plt.show()


# Nombre usuario 'Free Tiktok Scrapper'
free_tiktok_scraper['username'] =
free_tiktok_scraper['webVideoUrl'].str.extract(r'/@([^/]+)/video')

# Frecuencia nombre por usuario
user_counts = free_tiktok_scraper['username'].value_counts().head(10)

# Gráfico barras usuario más frecuente
plt.figure(figsize=(10, 6))
user_counts.plot(kind='bar', color='orange')
plt.title('Frecuencia nombre de usuario')
plt.xlabel('Usuario')
plt.ylabel('Frecuencia en vídeos')
plt.xticks(rotation=45)
plt.show()
#omnibuslaw_videos
#Variable temporal
omnibuslaw_videos['createTime'] =
pd.to_datetime(omnibuslaw_videos['createTime'])
sns.set(style="whitegrid")
# 4 visualizaciones
fig, axs = plt.subplots(2, 2, figsize=(15, 12))
#Distribución de "likes", "shares" y comentarios en el tiempo
sns.lineplot(data=omnibuslaw_videos, x='createTime', y='n_likes',
label='Likes', ax=axs[0, 0])
sns.lineplot(data=omnibuslaw_videos, x='createTime', y='n_shares',
label='Shares', ax=axs[0, 0])
sns.lineplot(data=omnibuslaw_videos, x='createTime', y='n_comments',
label='Comments', ax=axs[0, 0])
axs[0, 0].set_title("'Likes', 'shares' y comentarios en el tiempo")
#Distribución temporal en la creación de videos
sns.histplot(data=omnibuslaw_videos, x='createTime', bins=10, kde=True,
ax=axs[0, 1], color='purple')
axs[0, 1].set_title('Distribución en la creación de vídeos')
#Relación entre reproducciones y "likes"
sns.scatterplot(data=omnibuslaw_videos, x='n_plays', y='n_likes',
ax=axs[1, 0], hue='n_shares', size='n_comments', palette='cool',
sizes=(40, 200))
axs[1, 0].set_title("Relación entre reproducciones y 'likes'")
#Relación entre los compartidos, cometarios y "likes"
sns.scatterplot(data=omnibuslaw_videos, x='n_shares', y='n_comments',
size='n_likes', hue='n_likes', palette='viridis', sizes=(50, 300),
ax=axs[1, 1])
axs[1, 1].set_title("Relación entre compartidos, comentarios y 'likes'")
```

```
# Estadísticos
# TikTok Data
free_tiktok_scraper_metrics = free_tiktok_scraper[['playCount',
'shareCount', 'videoMeta/duration']]
free_tiktok_scraper_metrics_summary =
free_tiktok_scraper_metrics.describe()
# Omnibus Law Videos Data
omnibuslaw_videos_metrics = omnibuslaw_videos[['n_likes', 'n_shares',
'n_comments', 'n_plays']]
omnibuslaw_videos_metrics_summary = omnibuslaw_videos_metrics.describe()
free_tiktok_scraper_metrics_summary
omnibuslaw_videos_metrics_summary
# Correlación Free Tiktok Scraper
free_tiktok_scraper_correlation = free_tiktok_scraper_metrics.corr()
# Correlación Omnibuslaw Videos
omnibuslaw_videos_correlation = omnibuslaw_videos_metrics.corr()
free_tiktok_scraper_correlation
sns.set(style="whitegrid")
# Distribución para "replays" y "shares"
fig, axs = plt.subplots(1, 2, figsize=(16, 8))
sns.histplot(free_tiktok_scraper['playCount'], bins=35, kde=True,
ax=axs[0])
axs[0].set_title("Distribución de videos reproducidos en 'Free Tiktok
Scrapper'")
axs[0].set_xlabel("Replays")
sns.histplot(free_tiktok_scraper['shareCount'], bins=35, kde=True,
ax=axs[1])
axs[1].set_title("Distribución de videos compartidos en 'Free Tiktok
Scrapper'")
axs[1].set_xlabel("Shares")
plt.tight_layout()
plt.show()
# Distribución para likes y comentarios en Omnibus Law Videos
fig, axs = plt.subplots(1, 2, figsize=(16, 8))
sns.histplot(omnibuslaw_videos['n_likes'], bins=35, kde=True, ax=axs[0])
axs[0].set_title("Distribución de 'likes' en 'Omnibus Law Videos'")
axs[0].set_xlabel("Likes")
sns.histplot(omnibuslaw_videos['n_comments'], bins=35, kde=True,
ax=axs[1])
axs[1].set_title("Distribución de comentarios en 'Omnibus Law Videos'")
axs[1].set_xlabel("Comments")
plt.tight_layout()
plt.show()
#Visualizaciones de tiktok_dataset
fig, axs = plt.subplots(2, 2, figsize=(15, 12))

#Distribución de la duración de los vídeos
sns.histplot(data=tiktok_dataset, x='video_duration_sec', bins=10,
kde=True, color='orange', ax=axs[0, 0])
axs[0, 0].set_title("Distribución de vídeos por su duración en segundos")
#Relación entre views, likes y descargas
sns.scatterplot(data=tiktok_dataset, x='video_view_count',
y='video_like_count', size='video_download_count',
hue='video_download_count', palette='coolwarm', sizes=(40, 200),
ax=axs[0, 1])
```

```
axs[0, 1].set_title("Relación entre visualizaciones, 'likes' y
descargas")
#Comparación entre likes, comentarios y shares
sns.scatterplot(data=tiktok_dataset, x='video_like_count',
y='video_comment_count', hue='video_share_count',
size='video_share_count', palette='viridis', sizes=(50, 300), ax=axs[1,
0])
axs[1, 0].set_title("Comparación entre 'likes', comentarios y 'shares'")
#Distribución de las visualizaciones por duración
sns.boxplot(data=tiktok_dataset, x='video_duration_sec',
y='video_view_count', palette='Set3',ax=axs[1, 1])
xticks = plt.xticks()[0]  # Obtener posiciones etiquetas
plt.xticks(xticks[::5])  # Mostrar 1 de 5 etiquetas
axs[1, 1].set_title('Distribución de las visualizaciones por duración del
vídeo')
#Visualizaciones para tiktok top 1000

fig, axs = plt.subplots(2, 2, figsize=(15, 12))
#Distribución cantidad de suscriptores
sns.histplot(data=tiktok_top_1000, x='Subscribers count', bins=10,
kde=True, color='skyblue', ax=axs[0, 0])
axs[0, 0].set_title('Distribución cantidad de suscriptores')
#Distribución entre suscriptores y media de visualizaciones
sns.scatterplot(data=tiktok_top_1000, x='Subscribers count', y='Views
avg.', size='Likes avg.', hue='Likes avg.', palette='coolwarm',
sizes=(40, 200), ax=axs[0, 1])
axs[0, 1].set_title('Distribución entre suscriptores y media en
visualizaciones')
#Relación entre número de 'likes', comentarios y compartidos
sns.scatterplot(data=tiktok_top_1000, x='Likes avg.', y='Comments avg.',
size='Shares avg.', hue='Shares avg.', palette='viridis', sizes=(50,
300), ax=axs[1, 0])
axs[1, 0].set_title("Relación entre número de 'likes', comentarios y
compartidos")
top_data = tiktok_top_1000.sort_values('Views avg.',
ascending=False).head(50)
sns.barplot(data=top_data, x='Views avg.', y='Subscribers count',
palette='mako')
axs[1, 1].set_title('Ranking en media de visitas')
axs[1, 1].set_xlabel('Views avg.')
axs[1, 1].set_ylabel('Subscribers count')

axs[1, 1].tick_params(axis='x', rotation=60)
xticks = axs[1, 1].get_xticks()
axs[1, 1].set_xticks(xticks[::5])
axs[1, 1].set_xscale('log')
plt.tight_layout()
plt.show()
#Washington
plt.figure(figsize=(10, 6))
sns.boxplot(data=washingtonpost_videos, x="video_length", y="n_plays",
palette="pastel")
plt.title("Relación entre la longitud de vídeos y número de
reproducciones", fontsize=16)
plt.xlabel("Longitud de los vídeos en segundos", fontsize=12)
```

```
plt.ylabel("Número de reproducciones", fontsize=12)
plt.tight_layout()
plt.show()
sns.set(style="whitegrid", context="talk")
plt.figure(figsize=(12, 6))
sns.regplot(data=washingtonpost_videos, x="video_length", y="n_plays",
scatter_kws={"s": 100}, line_kws={"color": "red"})
plt.title("Relación entre longitud de vídeos y número reproducciones",
fontsize=18)
plt.xlabel("Longitud de vídeos en segundos", fontsize=14)
plt.ylabel("Número de reproducciones", fontsize=14)
plt.tight_layout()
plt.show()
washingtonpost_videos["video_time"] =
pd.to_datetime(washingtonpost_videos["video_time"], unit='s')
#Conversión de la variable tiempo al formato de fecha
plt.figure(figsize=(12, 6))
plt.plot(washingtonpost_videos["video_time"],
washingtonpost_videos["n_plays"], label="Reproducciones", color='green')
plt.plot(washingtonpost_videos["video_time"],
washingtonpost_videos["n_likes"], label="Likes", color='blue')
plt.legend()
plt.title("Evolución del número de reproducciones a lo largo del tiempo")
plt.xlabel("Fecha")
plt.ylabel("Proporción")
plt.show()
# Regresión lineal para predecir el número de visitas
X = free_tiktok_scraper[['authorMeta/following', 'videoMeta/duration',
'authorMeta/fans', 'commentCount']]  # Variables predictoras
y = free_tiktok_scraper['searchHashtag/views']  # Variable objetivo



def convert_integer(value):
    value = str(value).upper().strip()  # Convertir a string por
seguridad
    if value[-1] == 'K':  # Miles
        return int(float(value[:-1]) * 1_000)
    elif value[-1] == 'M':  # Millones
        return int(float(value[:-1]) * 1_000_000)
    elif value[-1] == 'B':  # Mil millones
        return int(float(value[:-1]) * 1_000_000_000)
    else:
        return int(float(value))  # Si no hay sufijo, solo convertir a
entero
free_tiktok_scraper['searchHashtag/views'] =
free_tiktok_scraper['searchHashtag/views'].astype(str).apply(convert_inte
ger)
```

```
X_train, X_test, y_train, y_test = train_test_split(
                                       X, y,
                                       train_size   = 0.8,
                                       random_state = 1234,
                                       shuffle      = True)
X_train = X_train.select_dtypes(include=[np.number])
# Modelo
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.OLS(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
# Selección de Variables para predecir el número de visitras
X = free_tiktok_scraper[['commentCount']]  # Variables predictoras
y = free_tiktok_scraper['searchHashtag/views']  # Variable objetivo
free_tiktok_scraper['searchHashtag/views'] =
free_tiktok_scraper['searchHashtag/views'].astype(str).apply(convert_inte
ger)
X_train, X_test, y_train, y_test = train_test_split(
                                       X,
                                       y,
                                       train_size   = 0.8,
                                       random_state = 1234,
                                       shuffle      = True)
# Modelo
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.OLS(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())
#Correlación
correlation_matrix = free_tiktok_scraper[['authorMeta/following',
'videoMeta/duration',
                                          'authorMeta/fans',
'commentCount', 'searchHashtag/views']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Matriz de Correlación")
plt.show()
# Intervalos de confianza para los coeficientes del modelo
intervalos_ci = modelo.conf_int(alpha=0.05)
intervalos_ci.columns = ['2.5%', '97.5%']
intervalos_ci
# Predicciones con intervalo de confianza del 95%
predicciones =
modelo.get_prediction(exog=X_train).summary_frame(alpha=0.05)[10]
```

[10] Amat, J. (2020, octubre). *Regresión lineal con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py10-regresion-lineal-python

```
predicciones.head(4)
# Predicciones con intervalo de confianza del 95%
predicciones =
modelo.get_prediction(exog=X_train).summary_frame(alpha=0.05)
predicciones['x'] = X_train.loc[:, 'commentCount']
predicciones['y'] = y_train
predicciones = predicciones.sort_values('x')
# Gráfico del modelo#
================================================================================
=====
fig, ax = plt.subplots(figsize=(6, 3.84))
ax.scatter(predicciones['x'], predicciones['y'], marker='o', color =
"gray")
ax.plot(predicciones['x'], predicciones["mean"], linestyle='-',
label="OLS")
ax.plot(predicciones['x'], predicciones["mean_ci_lower"], linestyle='--',
color='blue', label="95% CI")
ax.plot(predicciones['x'], predicciones["mean_ci_upper"], linestyle='--',
color='blue')
ax.fill_between(predicciones['x'], predicciones["mean_ci_lower"],
predicciones["mean_ci_upper"], alpha=0.3)
ax.legend();
# Error de test del modelo
#
================================================================================
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog=X_test)
rmse = root_mean_squared_error(y_test, predicciones)
print(f"El error (rmse) de test es: {rmse}")
#Random Forest
random_forest_model = RandomForestRegressor(
            n_estimators = 10,
            criterion    = 'squared_error',
            max_depth    = None,
            max_features = 1,
            oob_score    = False,
            n_jobs       = -1,
            random_state = 123)
random_forest_model.fit(X_train, y_train)
# Error de test del modelo inicial
predicciones = modelo.predict(X_test)
rmse = root_mean_squared_error(y_true=y_test, y_pred=predicciones)


print(f"El error (rmse) de test es: {rmse}")[11]
```

[11] Amat, J. (2020, octubre). *Regresión lineal con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py10-regresion-lineal-python

```
train_scores = []
oob_scores   = []# Valores evaluados
estimator_range = range(1, 150, 5)
# Bucle para entrenar un modelo con cada valor de n_estimators y extraer
su error de entrenamiento y de Out-of-Bag.
for n_estimators in estimator_range:
    modelo = RandomForestRegressor(
                n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 1,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123 )
    modelo.fit(X_train, y_train)
    train_scores.append(modelo.score(X_train, y_train))
    oob_scores.append(modelo.oob_score_)
fig, ax = plt.subplots(figsize=(5, 3))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, oob_scores, label="out-of-bag scores")
ax.plot(estimator_range[np.argmax(oob_scores)], max(oob_scores),
marker='o', color = "red", label="max score")
ax.set_ylabel("R^2")
ax.set_xlabel("n_estimators")
ax.set_title("Evolución del out-of-bag-error vs número árboles")
plt.legend();
print(f"Valor óptimo de n_estimators:
{estimator_range[np.argmax(oob_scores)]}")
# Validación empleando k-cross-validation y neg_root_mean_squared_error
train_scores = []
cv_scores    = []# Valores evaluados
estimator_range = range(1, 150, 5)
# Bucle para entrenar un modelo con cada valor de n_estimators y extraer
su error de entrenamiento y de k-cross-validation.
for n_estimators in estimator_range:
    modelo = RandomForestRegressor(
               n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 1,
                oob_score    = False,
                n_jobs       = -1,
                random_state = 123 )
    # Error de entrenamiento
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X=X_train)
    rmse = root_mean_squared_error(     y_true  = y_train,
            y_pred  = predicciones,
           )
    train_scores.append(rmse)[12]
```

[12] Amat, J. (2020, octubre). *Random Forest con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py08_random_forest_python

```
    # Error de validación cruzada
    scores = cross_val_score( estimator = modelo,
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
                cv        = 5)
    # Se agregan los scores de cross_val_score() y se pasa a positivo
    cv_scores.append(-1*scores.mean())
# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(5, 3))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, cv_scores, label="cv scores")
ax.plot(estimator_range[np.argmin(cv_scores)], min(cv_scores),
        marker='o', color = "red", label="min score")
ax.set_ylabel("root_mean_squared_error")
ax.set_xlabel("n_estimators")
ax.set_title("Evolución del cv-error vs número árboles",pad=20)
plt.legend();
print(f"Valor óptimo de n_estimators:
{estimator_range[np.argmin(cv_scores)]}")
# Grid de hiperparámetros evaluados
param_grid = ParameterGrid(
                {'n_estimators': [150],
                 'max_features': [5, 7, 9],
                 'max_depth'   : [None, 3, 10, 20] })
# Loop para ajustar un modelo con cada combinación de hiperparámetros
# ==============================================================
resultados = {'params': [], 'oob_r2': []}
for params in param_grid:
    modelo = RandomForestRegressor(
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123,
                ** params)
    modelo.fit(X_train, y_train)
    resultados['params'].append(params)
    resultados['oob_r2'].append(modelo.oob_score_)
    print(f"Modelo: {params} ✓")
# Resultados
resultados = pd.DataFrame(resultados)
resultados = pd.concat([resultados,
resultados['params'].apply(pd.Series)], axis=1)
resultados = resultados.drop(columns = 'params')
resultados = resultados.sort_values('oob_r2', ascending=False)
resultados.head(4)# Mejores hiperparámetros por out-of-bag error
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (oob-r2)")
print("----------------------------------------")
print(resultados.iloc[0,0:])
# Grid de hiperparámetros evaluados [13]
```

[13] Amat, J. (2020, octubre). *Random Forest con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py08_random_forest_python

```
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20]}
# Búsqueda por grid search con validación cruzada
grid = GridSearchCV(
            estimator  = RandomForestRegressor(random_state = 123),
            param_grid = param_grid,
            scoring    = 'neg_root_mean_squared_error',
            n_jobs     = cpu_count() - 1,
            cv         = RepeatedKFold(n_splits=5, n_repeats=3,
random_state=123),
            refit      = True,
            verbose    = 0,
            return_train_score = True)
grid.fit(X=X_train, y=y_train)
resultados = pd.DataFrame(grid.cv_results_)# Resultados
resultados.filter(regex = '(param.*|mean_t|std_t)') \
    .drop(columns = 'params') \
    .sort_values('mean_test_score', ascending = False) \
    .head(4)# Mejores hiperparámetros encontrados mediante validación
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
importancia = permutation_importance(
                estimator    = modelo_final,
                X            = X_train,
                y            = y_train,
                n_repeats    = 5,
                scoring      = 'neg_root_mean_squared_error',
                n_jobs       = cpu_count() - 1,
                random_state = 123)
# Se almacenan los resultados (media y desviación) en un dataframe
df_importancia = pd.DataFrame(
                    {k: importancia[k] for k in ['importances_mean',
'importances_std']})
df_importancia['feature'] = X_train.columns
df_importancia.sort_values('importances_mean', ascending=False)
14
```

[14] Amat, J. (2020, octubre). *Random Forest con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py08_random_forest_python

```
y = np.log1p(free_tiktok_scraper['searchHashtag/views'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=123)  #Aplicando el logaritmo en el modelo de random forest
random_forest_model = RandomForestRegressor(n_estimators=100,
random_state=42)
random_forest_model.fit(X_train, y_train)
y_pred = random_forest_model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R² Score:", r2_score(y_test, y_pred))
print("MAPE:", np.mean(np.abs((y_test - y_pred) / y_test)))
r2_scores = cross_val_score(random_forest_model, X, y, cv=5,
scoring='r2')
print(r2_scores)
#Voy a quedarme solo con las variables de "tiktok_dataset" que sean
numéricas y quitarle los valores nulos
tiktok_dataset = tiktok_dataset.dropna(subset=['video_download_count'])
tiktok_dataset=
tiktok_dataset[['video_download_count','video_duration_sec',
'video_like_count','video_comment_count','video_view_count']]
```

```
# División de los datos en entrenamiento y test
#
==========================================================================
=====
X_train, X_test, y_train, y_test = train_test_split(
                                        tiktok_dataset.drop(columns =
"video_download_count"),

tiktok_dataset['video_download_count'],
                                        random_state = 123
                                    )
# Creación del modelo
# ==========================================================================
modelo = HistGradientBoostingRegressor(
            max_iter     = 10,
            loss         = 'squared_error',
            random_state = 123
         )
# Entrenamiento del modelo
# ======
modelo.fit(X_train, y_train)
# Validación empleando k-cross-validation y neg_root_mean_squared_error
# ==========================================================================
train_scores = []
cv_scores    = [] # Valores evaluados
max_iter_range = range(1, 500, 25)
# Bucle para entrenar un modelo con cada valor de n_estimators y extraer
su error de entrenamiento y de k-cross-validation.
for max_iter in max_iter_range:
    modelo = HistGradientBoostingRegressor(
                max_iter     = max_iter,
                random_state = 123)  # Error de train
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X = X_train)
    rmse = mean_squared_error(
            y_true  = y_train,
            y_pred  = predicciones)
    train_scores.append(rmse) # Error de validación cruzada
    scores = cross_val_score(
                estimator = modelo,[15]
```

[15] Amat, J. (2020, octubre). *Gradient Boosting con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py09_gradient_boosting_python

```
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
                cv        = 5,
                n_jobs    = multiprocessing.cpu_count() - 1,
            )# Se agregan los scores de cross_val_score()y pasa positivo
    cv_scores.append(-1*scores.mean())
print(f"Valor óptimo de n_estimators:
{max_iter_range[np.argmin(cv_scores)]}")
# Validación empleando k-cross-validation y neg_root_mean_squared_error
resultados = {} # Valores evaluados
learning_rates_grid = [0.001, 0.01, 0.1]
max_iter_grid   = [10, 20, 100, 200, 300, 400, 500, 1000, 2000, 5000]
for learning_rate in learning_rates_grid: # Bucle
    train_scores = []
    cv_scores    = []
    for n_estimator in max_iter_grid:
        modelo = HistGradientBoostingRegressor(
                    max_iter      =  n_estimator,
                    learning_rate = learning_rate,
                    random_state  = 123)
        modelo.fit(X_train, y_train) # Error de train
        predicciones = modelo.predict(X = X_train)
        rmse = mean_squared_error(
                y_true  = y_train,
                y_pred  = predicciones
               )
        train_scores.append(rmse)
        scores = cross_val_score( # Error de validación cruzada
                    estimator = modelo,
                    X         = X_train,
                    y         = y_train,
                    scoring   = 'neg_root_mean_squared_error',
                    cv        = 3,
                    n_jobs    = multiprocessing.cpu_count() - 1
                 )
        cv_scores.append(-1*scores.mean())# Se agregan los scores de
cross_val_score()
    resultados[learning_rate] = {'train_scores': train_scores,
'cv_scores': cv_scores}
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(9, 3)) #Gráfico
for key, value in resultados.items():
    axs[0].plot(max_iter_grid, value['train_scores'], label=f"Learning
rate {key}")
    axs[0].set_ylabel("root_mean_squared_error")
    axs[0].set_xlabel("n_estimators")
    axs[0].set_title("Evolución del train error vs learning rate") [16]
```

[16] Amat, J. (2020, octubre). *Gradient Boosting con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py09_gradient_boosting_python

```
    axs[1].plot(max_iter_grid, value['cv_scores'], label=f"Learning rate
{key}")
    axs[1].set_ylabel("root_mean_squared_error")
    axs[1].set_xlabel("n_estimators")
    axs[1].set_title("Evolución del cv-error vs learning rate")
    plt.legend();# Grid de hiperparámetros evaluados
param_grid = {'loss'             : ['squared_error', 'absolute_error'],
              'learning_rate'    : [0.001, 0.01, 0.1],
              'max_depth'        : [3, 5, 10, 20],
              'l2_regularization': [0, 1, 10]
             }
# Búsqueda por grid search con validación cruzada
grid = GridSearchCV(
        estimator  = HistGradientBoostingRegressor(
                         max_iter            = 1000,
                         random_state        = 123,
# Activación de la parada temprana
                         early_stopping      = True,
                         validation_fraction = 0.1,
                         n_iter_no_change    = 10,
                         tol                 = 1e-7,
                         scoring             = 'loss',
                     ),
        param_grid = param_grid,
        scoring    = 'neg_root_mean_squared_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=3, n_repeats=1,
random_state=123),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
grid.fit(X = X_train, y = y_train)
# Resultados
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param.*|mean_t|std_t)') \
    .drop(columns = 'params') \
    .sort_values('mean_test_score', ascending = False) \
    .head(4)
# Mejores hiperparámetros por validación cruzada
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
# Número de árboles del modelo final (early stopping)
print(f"Número de árboles del modelo: {grid.best_estimator_.n_iter_}")[17]
```

[17] Amat, J. (2020, octubre), *Gradient Boosting con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py09_gradient_boosting_python

```
importancia = permutation_importance(
                estimator    = modelo_final,
                X            = X_train,
                y            = y_train,
                n_repeats    = 5,
                scoring      = 'neg_root_mean_squared_error',
                n_jobs       = multiprocessing.cpu_count() - 1,
                random_state = 123
            )
# Se almacenan los resultados (media y desviación) en un dataframe
df_importancia = pd.DataFrame(
                    {k: importancia[k] for k in ['importances_mean',
'importances_std']}
                )
df_importancia['feature'] = X_train.columns
df_importancia.sort_values('importances_mean', ascending=False)
param_grid = {'max_depth'        : [None, 1, 3, 5, 10, 20],
              'subsample'        : [0.5, 1],
              'learning_rate'    : [0.001, 0.01, 0.1],
              'booster'          : ['gbtree']
             }
# Crear conjunto de validación
np.random.seed(123)
idx_validacion = np.random.choice(
                    X_train.shape[0],
                    size=int(X_train.shape[0]*0.1), #10% de los trai
                    replace=False
                 )
X_val = X_train.iloc[idx_validacion, :].copy()
y_val = y_train.iloc[idx_validacion].copy()
X_train_grid = X_train.reset_index(drop = True).drop(idx_validacion, axis
= 0).copy()
y_train_grid = y_train.reset_index(drop = True).drop(idx_validacion, axis
= 0).copy()
# XGBoost necesita pasar los paramétros específicos para a.fit()
fit_params = {
              "eval_set": [(X_val, y_val)],
              "verbose": False
             }# Búsqueda por grid search con validación cruzada
grid = GridSearchCV(
        estimator  = XGBRegressor([18]
```

[18] Amat, J. (2020, octubre), *Gradient Boosting con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py09_gradient_boosting_python

```
                        n_estimators          = 1000,
                        early_stopping_rounds = 5,
                        eval_metric           = "rmse",
                        random_state          = 123
                    ),
        param_grid = param_grid,
        scoring    = 'neg_root_mean_squared_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=3, n_repeats=1,
random_state=123),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
grid.fit(X = X_train_grid, y = y_train_grid, **fit_params) # Resultados
#
==========================================================================
=====
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param.*|mean_t|std_t)') \
    .drop(columns = 'params') \
    .sort_values('mean_test_score', ascending = False) \
    .head(4)
# Mejores hiperparámetros por validación cruzada
#==============================================================
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
# Número de árboles del modelo final (early stopping)
n_arboles_incluidos = len(grid.best_estimator_.get_booster().get_dump())
print(f"Número de árboles incluidos en el modelo: {n_arboles_incluidos}")
```

```
# Grid de hiperparámetros evaluados
param_grid = {'n_estimators'     : [100, 500, 1000, 5000],
              'max_depth'        : [-1, 1, 3, 5, 10, 20],
              'subsample'        : [0.5, 1],
              'learning_rate'    : [0.001, 0.01, 0.1],
              'boosting_type'    : ['gbdt']
             }
# Búsqueda por grid search con validación cruzada
grid = GridSearchCV([19]
```

[19] Amat, J. (2020, octubre). *Gradient Boosting con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py09_gradient_boosting_python

```
        estimator  = LGBMRegressor(random_state=123),
        param_grid = param_grid,
        scoring    = 'neg_root_mean_squared_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=3, n_repeats=1,
random_state=123),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
grid.fit(X = X_train_grid, y = y_train_grid)
# Resultados
#======================================================================
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param.*|mean_t|std_t)') \
    .drop(columns = 'params') \
    .sort_values('mean_test_score', ascending = False) \
    .head(4) # Mejores hiperparámetros por validación cruzada
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
# Error de test del modelo final
modelo_final = grid.best_estimator_
predicciones = modelo_final.predict(X = X_test,)
rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones
       )
print(f"El error (rmse) de test es: {rmse}") [20]
```

[20] Amat, J. (2020, octubre), *Gradient Boosting con Python*. Ciencia de datos. Recuperado de https://cienciadedatos.net/documentos/py09_gradient_boosting_python

```
plt.figure(figsize=(8, 6))
sns.heatmap(tiktok_top_1000[['Subscribers count', 'Views avg.', 'Likes
avg.', 'Comments avg.', 'Shares avg.']].corr(), annot=True,
cmap="coolwarm", fmt=".2f")
plt.title("Matriz de Correlación")
plt.show()
#Reemplazo de valores no numéricos por NA para quitar valores no
numéricos
tiktok_top_1000[['Subscribers count', 'Views avg.', 'Likes avg.',
'Comments avg.', 'Shares avg.']] = \
    tiktok_top_1000[['Subscribers count', 'Views avg.', 'Likes avg.',
'Comments avg.', 'Shares avg.']].apply(pd.to_numeric, errors='coerce')

# Quitar valores NA
tiktok_top_1000 = tiktok_top_1000.dropna()
tiktok_top_1000_without_views=tiktok_top_1000[['Subscribers count',
'Likes avg.', 'Comments avg.', 'Shares avg.']]
#Apliación de diferentes modelos de machine learning
X_train, X_test, y_train, y_test = train_test_split(
                                        tiktok_top_1000_without_views,
                                        tiktok_top_1000['Views avg.'],
                                        test_size=0.2,
                                        random_state = 123)
#Escalar datos de entrenamiento y de test
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
#Aplicación de modelos de regresión
regresores = {
    "Regresión Lineal": LinearRegression(),
    "Random Forest": RandomForestRegressor(n_estimators=82,
random_state=42),
    "Gradient Boosting": GradientBoostingRegressor(n_estimators=82,
random_state=42),
    "XGBoost": xgb.XGBRegressor(n_estimators=82, random_state=42),
    "Maquinas de soporte vectorial de regresión": SVR()
}
for nombre, modelo in regresores.items():
    modelo.fit(X_train, y_train)
    y_prediccion = modelo.predict(X_test)
    print(f"\n {nombre}:")
    print(f"RMSE:  {mean_squared_error(y_test, y_prediccion) **
0.5:.2f}")
    print(f"R² Score: {r2_score(y_test, y_prediccion):.2f}")
#Clasificación para la predicción de "High Followers"
#Se crea la variable 'x'
tiktok_top_1000['Top Followers'] = np.where(tiktok_top_1000["Subscribers
count"] > 5_000_000, 1, 0) #Se le da un valor de 1 si el número de
suscriptores es mayor a 5,000,000
X_train, X_test, y_train, y_test = train_test_split(
                                        tiktok_top_1000_without_views,
                                        tiktok_top_1000['Top Followers'],
                                        test_size=0.2,
                                        random_state = 123)
#Aplicación de modelos de clasificación
```

```
clasificadores = {
    "Regresión Logística": LogisticRegression(max_iter=500),
    "Random Forest": RandomForestClassifier(n_estimators=82,
random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(n_estimators=82,
random_state=42),
    "XGBoost": xgb.XGBClassifier(n_estimators=82, random_state=42),
    "Maquinas de soporte vectorial de clasificación": SVC()
}
for nombre, modelo in clasificadores.items():
    modelo.fit(X_train, y_train)
    y_prediccion = modelo.predict(X_test)
    print(f"\n {nombre}:")
    print(f"RMSE:  {mean_squared_error(y_test, y_prediccion) **
0.5:.2f}")
    print(f"R² Score: {r2_score(y_test, y_prediccion):.2f}")
# Transformación logarítmica a variables con sesgo
tiktok_top_1000 = pd.read_csv('./tiktok_top_1000.csv')
tiktok_top_1000 = tiktok_top_1000.drop(columns = ['Country',
'Account','Title','Link','Scraped'])
tiktok_top_1000['Subscribers count'] =
np.log1p(tiktok_top_1000['Subscribers count'])
tiktok_top_1000['Views avg.'] = np.log1p(tiktok_top_1000['Views avg.'])
# Creación de nuevas métricas
tiktok_top_1000['Engagement Rate'] = (tiktok_top_1000['Likes avg.'] +
tiktok_top_1000['Comments avg.'] + tiktok_top_1000['Shares avg.']) /
tiktok_top_1000['Subscribers count']
tiktok_top_1000['Viral Ratio'] = tiktok_top_1000['Shares avg.'] /
tiktok_top_1000['Views avg.']
tiktok_top_1000['Interacción por usuario'] = (tiktok_top_1000['Likes
avg.'] + tiktok_top_1000['Comments avg.']) / tiktok_top_1000['Subscribers
count']
X_train, X_test, y_train, y_test = train_test_split(
                                        tiktok_top_1000.drop(columns =
'Rank'),
                                        tiktok_top_1000['Rank'],
                                        test_size=0.2,
                                        random_state = 123)
#Escalar datos de entrenamiento y de test
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
selector = SelectKBest(score_func=f_regression, k=5) #Selección de las 5
características más relevantes tanto para el entenamiento como para el
test
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)
#Aplicación de modelos de regresión
regresores = {
    "Regresión Lineal": LinearRegression(),
    "Random Forest": RandomForestRegressor(n_estimators=82,
random_state=42),
    "Gradient Boosting": GradientBoostingRegressor(n_estimators=82,
random_state=42),
    "XGBoost": xgb.XGBRegressor(n_estimators=82, random_state=42),
```

```
    "Maquinas de soporte vectorial de regresión": SVR()
}
for nombre, modelo in regresores.items():
    modelo.fit(X_train, y_train)
    y_prediccion = modelo.predict(X_test)
    print(f"\n {nombre}:")
    print(f"RMSE:  {mean_squared_error(y_test, y_prediccion) **
0.5:.2f}")
    print(f"R² Score: {r2_score(y_test, y_prediccion):.2f}")
#SHAP para el mejor modelo encontrado
mejor_modelo = regresores['XGBoost']
explicador = shap.Explainer(mejor_modelo, X_train_selected)
valores_shap = explicador(X_test_selected, check_additivity=False)
shap.summary_plot(valores_shap, X_test_selected)
```