UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

PREDICCIÓN DEL ENGAGEMENT EN INSTAGRAM APLICADO AL MUNDO DEL RUNNING

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Primo Aparici, Alba

Tutor/a: Fernández Diego, Marta

Cotutor/a: Baviera Puig, Tomás

CURSO ACADÉMICO: 2021/2022

Resumen

Hoy en día las redes sociales se han convertido en una herramienta que nos acerca a las personas, nos ayudan a entender los comportamientos de los consumidores y pueden ser muy útiles para extraer información que a simple vista no somos capaces de ver. En concreto, Instagram está abriendo a las empresas un nicho de consumidores cada vez más joven.

En el presente proyecto se va a abordar la predicción del *engagement* de los posts de Instagram en el área del *running* a través de distintos modelos de aprendizaje automático. No es fácil saber si un *post* va a tener éxito o no antes de publicarlo. Pero si las empresas son capaces de anticipar la acogida del *post*, podrán mejorar su interacción con la comunidad de manera más eficiente. En cualquier caso, existen multitud de factores que hacen que un *post* pueda acabar convirtiéndose en viral.

Para conseguir nuestro objetivo, se va a construir un repositorio con imágenes relacionadas con el mundo del *running* proporcionadas por la propia aplicación junto con todos los datos asociados a cada *post*. Una vez recogidos estos datos, se seguirá la metodología CRISP-DM para abarcar desde la definición de necesidades del cliente de este proyecto de análisis de datos hasta la evaluación de los resultados. Por último, se detallarán las aplicaciones comerciales/empresariales que podría tener una herramienta que incorporará nuestro modelo.

Palabras clave: *Engagement; redes sociales; Instagram; comunicación estratégica; running; Machine Learning; Deep Learning; Neural Networks*

Abstract

Today, social networks have become a tool that brings us closer to people, helps us to understand consumer behaviour and can be very useful for extracting information that we are not able to see with the naked eye. Instagram is opening up an increasingly younger consumer niche for companies.

This project will focus on predicting the engagement of Instagram posts in running using different machine learning models. It is not easy to know whether a post is going to be successful or not before it is published. But if companies can anticipate the reception of the post, they will be able to improve their interaction with the community more efficiently. In any case, there are many factors that can make a post go viral.

To achieve our goal, we are going to build a repository with images related to the world of running provided by the application itself along with all the data associated with each post. Once this data has been collected, the CRISP-DM methodology will be followed to cover everything from the definition of the client's needs for this data analysis project to the evaluation of the results. Finally, we will detail the commercial/business applications that a tool incorporating our model could have.

Key words: *Engagement; Social networks; Instagram; Strategic communication; running; Machine Learning; Deep Learning; Neural Networks*

Resum

Hui dia les xarxes socials s'han convertit en una eina que ens acosta a les persones, ens ajuden a entendre els comportaments dels consumidors i poden ser molt útils per a extraure informació que a simple vista no som capaces de veure. En concret, Instagram està obrint a les empreses un nínxol de consumidors cada vegada més jove.

En el present projecte s'abordarà la predicció del *engagement* dels posts d'Instagram en l'àrea del *running* a través de diferents models d'aprenentatge automàtic. No és fàcil saber si un *post* tindrà èxit o no abans de publicar-lo. Però si les empreses són capaces d'anticipar l'acolliment del *post,* podran millorar la seua interacció amb la comunitat de manera més eficient. En qualsevol cas, existeixen multitud de factors que fan que un *post* puga acabar convertint-se en viral.

Per a aconseguir el nostre objectiu, es construirà un repositori amb imatges relacionades amb el món del *running* proporcionades per la pròpia aplicació juntament amb totes les dades associades a cada *post*. Una vegada recollits aquestes dades, se seguirà la metodologia *CRISP-DM per a abastar des de la definició de necessitats del client d'aquest projecte d'anàlisi de dades fins a l'avaluació dels resultats. Finalment, es detallaran les aplicacions comercials/empresarials que podria tindre una eina que incorporarà el nostre model.

Paraules clau : *Engagement; xarxes socials; Instagram; comunicació estratègica; running; Machine Learning; Deep Learning; Neural Networks*

Índice general

Índice de figuras

Índice de tablas

1.Introducción

1.1 Objetivos

El principal objetivo del presente proyecto es conseguir un modelo que nos permita predecir con suficiente precisión el *engagement* en *posts* de Instagram.

Este objetivo se descompone en los siguientes:

- Profundizar en el concepto de *Engagement* para su predicción en un modelo de aprendizaje profundo.
- Creación de un conjunto de datos a partir a la información proporcionada por Instagram.
- Profundizar en la preparación y preprocesado de datos.
- Crear un modelo de red neuronal convolucional enfocado a un problema de clasificación para la extracción de características.
- Crear un modelo de red neuronal enfocado a un problema de regresión.
- Comparar los resultados de las distintos modelos predictivos propuestos.
- Identificar las distintas aplicaciones de nuestro modelo entrenado.

1.2 Motivación

Nos encontramos en un mundo en cambio constante: La tecnología está evolucionando a pasos agigantados y con ello, las estrategias de negocio mejoran dejando atrás técnicas o estrategias obsoletas.

Sin duda uno de los cambios más revolucionarios negocio-tecnología son las redes sociales y su creciente integración en estrategias de *marketing* o comunicación, causa directa de un incremento en ventas.

Actualmente, un 93 % de los encuestados en un estudio elaborado por *eMarketer e Insider Intelligence*, compañía de investigación líder a nivel mundial centrada en la transformación digital, asegura que utilizó Instagram en sus estrategias de *marketing*. [1]

Las redes sociales han cambiado la manera de aportar valor a los productos. Por ejemplo, hoy en día que una publicación en tu Instagram tenga un *engagement* elevado, es decir, que la interacción con los usuarios de la aplicación a través de los *likes* y los comentarios sea alta, aporta mucho más valor a un producto de lo que nos podemos imaginar [2]. Esto se ve potenciado con la colaboración de influencers que, cada vez son más.

Es curiosa la manera en que los usuarios nos dejamos influenciar por lo que vemos en redes sociales o más bien por lo que pretenden transmitir los sujetos interesados en conseguir *engagement* en sus publicaciones. La interacción entre usuarios es una acción muy ligada a los sentimientos, cómo nos hace sentir aquello que vemos, como lo interpretamos nosotros. ¿Se merece un *'like'* el *post* que acabo de ver? ¿Me transmite lo suficiente como para dejar un comentario? Y la pregunta del millón, ¿sería capaz una máquina de predecir la interacción con cualquier *post*? ¿Son relevantes los elementos o patrones que se puedan encontrar en las imágenes sin que el usuario sea consciente? Pues en este trabajo pretendemos empezar a dar respuesta a alguna de estas preguntas, uniendo el *Marketing* como área empresarial que más se acerca a las redes sociales y la parte tecnológica más en auge hoy en día, como el *big data* o la inteligencia artificial.
Aunque este trabajo es extrapolable a distintos sectores, áreas empresariales, públicos objetivos… se ha orientado a una actividad que ahora mismo es tendencia en Valencia, y que además tiene un gran impacto económico en la ciudad: el *running.*
Además, este grupo social se relaciona directamente con la salud, un tema en el que toda la investigación posible nunca será suficiente. Y aunque en este trabajo, no se investigue esa rama, el dataset construido podría ser de utilidad.
También es interesante este grupo social porque entra en juego la salud, uno de los conceptos considerados de vital importancia en la vida.

1.3 Impacto Esperado

Tras la finalización del presente proyecto se espera conseguir un modelo que nos permita predecir con suficiente precisión el *engagement* en *posts* de Instagram.
Desde el punto de vista empresarial, estos modelos podrían ayudar a las empresas a mejorar el *engagement* de sus *posts,* y esto se podría ver traducido en un incremento de clientes.

A través de este TFG se pretende aclarar qué factores son los más influyentes para conseguir *engagement* y averiguar hasta qué punto un modelo sería capaz de predecirlo.

Además, estos modelos podrían facilitar campañas de publicidad en redes sociales, ya que, a la hora de elegir qué imágenes complementan la campaña, contaríamos con este software que nos confirma que esa imagen va a funcionar, o dicho de otro modo va a tener un buen *engagement*.

También es relevante el papel que puede tener la extracción de conocimiento de las imágenes a la hora de analizar el comportamiento de los usuarios a través de los resultados y la relación con los metadatos. Podrían ayudar a los equipos de trabajo a conocer cómo impactan sus *posts* en la comunidad de Instagram.

En conclusión, se trata de un estudio idóneo para ayudar a tomar decisiones respecto a publicaciones en redes sociales.

1.4 Metodología

La metodología en la que nos hemos basado para este proyecto ha sido CRISP-DM (*Cross-Industry Standard Process for Data Mining*). Este modelo está orientado a proyectos profesionales de análisis de datos [3]. Trabajar con esta metodología nos ayuda a estructurar y planificar el proyecto enfocándonos en los datos, uno de los principales pilares del proyecto. Se trata de un modelo estándar formado por 6 fases como muestra la Figura 1.

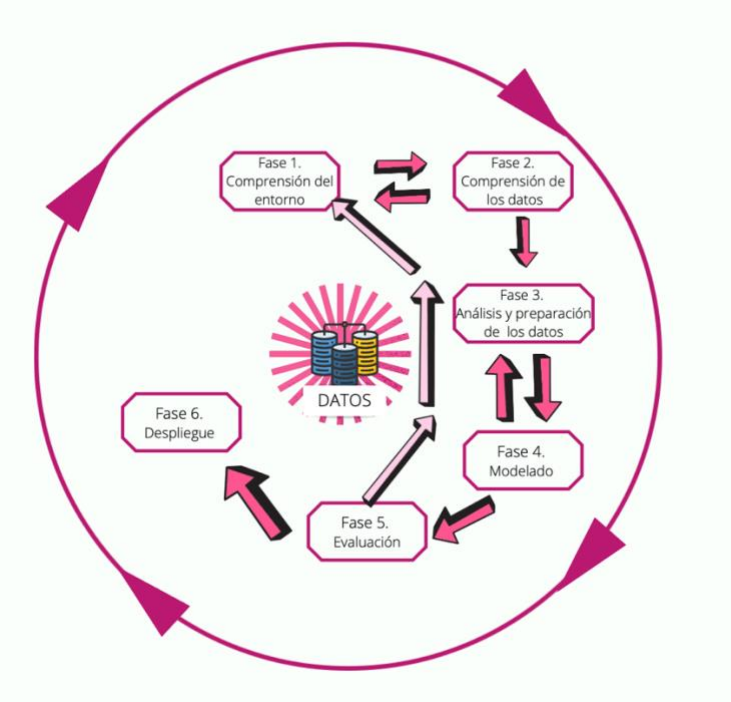

Figura 1 Fases metodología CRISP-DM

Fuente: Elaboración propia.

A continuación, se explica brevemente cada fase y como las hemos adaptado al proyecto:

- Fase 1. Comprensión del entorno/negocio: En esta primera fase se establecen los objetivos del proyecto y cómo se pretende trabajar cada una de las problemáticas expuestas. En nuestro caso, en esta primera fase definimos objetivos, metodologías y tecnologías usadas en el proyecto. También

corresponde a esta fase el Estado del arte donde se exponen el contexto relacionado con este proyecto. Además, se investigará si está a nuestro alcance recolectar los datos necesarios para poder solucionar el problema planteado.

- Fase 2. Comprensión de los datos: Esta segunda fase sirve para familiarizarse y entender los datos con los que se va a trabajar. De esta manera se pueden identificar problemas no previstos, descubrir subconjuntos interesantes de datos o afinar qué datos van a ser relevantes para el estudio. En nuestro caso, en esta fase hemos tratado de entender cómo se relacionaban los datos disponibles con la interacción de los usuarios. También incluimos en esta fase un script desarrollado con la finalidad de facilitar la descarga tanto de imágenes como posts de Instagram necesarios para el estudio.

- Fase 3. Análisis y preparación de los datos: Desde la preparación del dataset hasta la selección de atributos para el estudio. En esta fase se prepara el conjunto de datos (transformación y limpieza) para el modelo. En nuestro caso, en esta fase se incluye la selección de atributos a tener en cuenta para los modelos propuestos, así como la limpieza del data-set antes de ser utilizado.

- Fase 4. Modelado: En esta fase, se seleccionan y aplican las técnicas de modelado que sean pertinentes al problema (cuantas más mejor), y se calibran sus parámetros a valores óptimos. Aquí es donde hemos desarrollado diferentes modelos predictivos que han sido entrenados con el dataset elaborado anteriormente.

- Fase 5. Evaluación: En esta etapa del proyecto, se seleccionan los modelos que cumplen con un nivel suficiente de predicción. Antes de proceder al despliegue final del modelo, es importante evaluarlo a fondo y revisar que se han tenido en cuenta todas las casuísticas posibles. Al final de esta fase, se obtienen las principales conclusiones del proyecto, así como líneas futuras de trabajo.

- Fase 6. Despliegue: Puesta en producción del modelo desarrollado. En nuestro caso esta fase no aplica.

1.5 Estructura de la memoria

Para estructurar este proyecto nos hemos guiado por las fases de la metodología CRISP-DM antes mencionadas con el fin de facilitar la compresión del proceso seguido.

A continuación, describimos brevemente de cada uno de los capítulos que componen este documento:

1. Introducción: en este primer apartado se introduce brevemente el contexto del trabajo a realizar, la motivación por la que se ha decidido llevar a cabo este TFG, los objetivos que se pretenden conseguir y la metodología utilizada para ello.

2. Estado del arte: en este apartado se da una breve explicación de los elementos más importantes para realizar el trabajo: *Machine Learning,* redes sociales y *engagement.*

3. Herramientas: En este capítulo se describen las tecnologías que han sido necesarias para llevar a cabo el proyecto.

4. Análisis del problema: en este capítulo se describe de forma detallada la problemática a resolver y cómo se ha abordado.

5. Preparación y comprensión de los datos: En este apartado se detalla cómo se ha construido el conjunto de datos que posteriormente será entrenado. Además, se hará el preprocesado de datos y la elección de variables para los modelos.

6. Modelado: En este apartado se desarrollarán y evalúan los modelos y se estudiaran a fondo los resultados obtenidos para determinar qué modelo es más adecuado.

7.Evaluaciones: En este apartado se analiza si han alcanzado los objetivos propuestos y se realiza una valoración del proyecto en su conjunto.

8. Conclusiones: En este apartado se analiza si han alcanzado los objetivos propuestos y se realiza una valoración del proyecto en su conjunto. Además, se describirán posibles líneas de desarrollo que podrían derivarse de este primer proyecto.

9. Bibliografía: Por último, se referencian todas las fuentes bibliográficas utilizadas para la realización del proyecto.

Por último, cabe mencionar que todos los ficheros y bases de datos utilizados a lo largo del trabajo se pueden localizar en el siguiente repositorio creado en específico: [https://github.com/albaap007/TFG_INF].

2. Estado del arte

En este epígrafe se puede encontrar una revisión de los conceptos relacionados con el estudio. También nos ayuda a entender la situación actual de las tecnologías utilizadas.

2.1 *Machine Learning*

La inteligencia artificial es un conjunto de algoritmos y técnicas que pueden ser usadas para resolver problemas que los humanos resolvemos intuitivamente, pero que son realmente complejas para un ordenador [4]

Aunque cada vez la inteligencia artificial está más presente en nuestras vidas, lleva muchos años formando parte de nuestro entorno. Por ejemplo, en 1997 la computadora Deep Blue de IBM ya ganó jugando al ajedrez a Garry Kasparov [4], campeón mundial de ajedrez, o una calculadora, muy comunes en nuestro día a día, es una máquina en la que se aplica Inteligencia artificial. Cabe puntualizar que en muchas ocasiones se asocia inteligencia artificial con reconocimiento de sentimientos, pero esto es un mito.

Dentro de la inteligencia artificial podemos encontrar dos tipos de aprendizaje. Por un lado, el aprendizaje basado en reglas en el que se le ofrece al modelo una serie de características para que así la "máquina" encuentre cuál es la solución correcta. Por otro lado, el aprendizaje basado en ejemplos, más conocido como *Machine Learning.* Este aprendizaje alimenta su modelo con ejemplos y la máquina tiene que ser capaz de encontrar patrones para encontrar la solución óptima y de ahí, extraeríamos las características más relevantes.

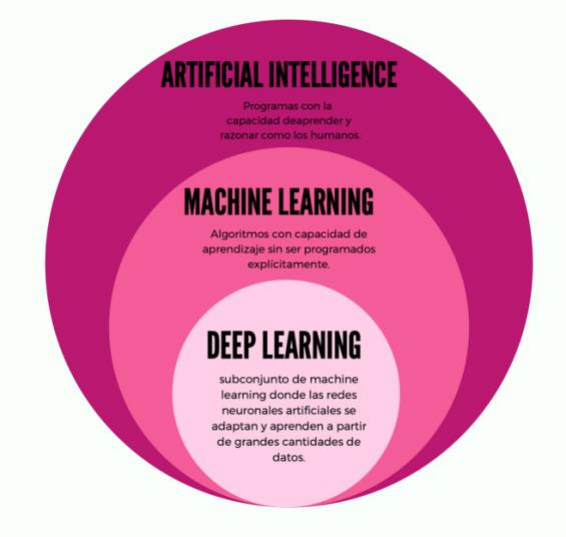

Figura 2 Jerarquía Inteligencia Artificial

Fuente: Elaboración propia.

Profundizando un poco más encontramos el *Deep Learning* o Entrenamiento profundo. Se trata de un sub-tipo de *Machine Learning*. Este sub-tipo hace por nosotros la extracción de características, es decir, son las redes neuronales las encargadas de encontrar patrones, extraer las características y clasificar o predecir los datos. Esta técnica es muy eficiente cuando se dispone de una gran cantidad de datos [5].

Centrándonos en Machine Learning, Arthur Samuel, uno de los primeros líderes estadounidenses en el campo de los juegos de ordenador y la inteligencia artificial, acuñó el término "aprendizaje automático" en 1959 mientras trabajaba en IBM. Definió el aprendizaje automático como "el campo de estudio que da a los ordenadores la capacidad de aprender sin ser programados explícitamente". Sin embargo, no existe una definición universalmente aceptada para el aprendizaje automático. A continuación damos dos definiciones más.

1. El aprendizaje automático consiste en programar los ordenadores para que optimicen un criterio de rendimiento a partir de datos de estudio o de experiencias pasadas. Tenemos un modelo definido por unos parámetros, y el aprendizaje es la ejecución de un programa informático para optimizar los parámetros del modelo utilizando los datos de entrenamiento o la experiencia pasada. El modelo puede ser predictivo para hacer predicciones en el futuro, o descriptivo para obtener conocimiento de los datos, o ambas cosas [6].
2. El campo de estudio conocido como aprendizaje automático se ocupa de la cuestión de cómo construir programas informáticos que mejoren automáticamente con la experiencia [7].

El proceso de aprendizaje ya sea por parte de un humano o de una máquina, puede dividirse en cuatro componentes, a saber: almacenamiento de datos, abstracción, generalización y evaluación. La figura 3 ilustra los distintos componentes y las etapas del proceso de aprendizaje.

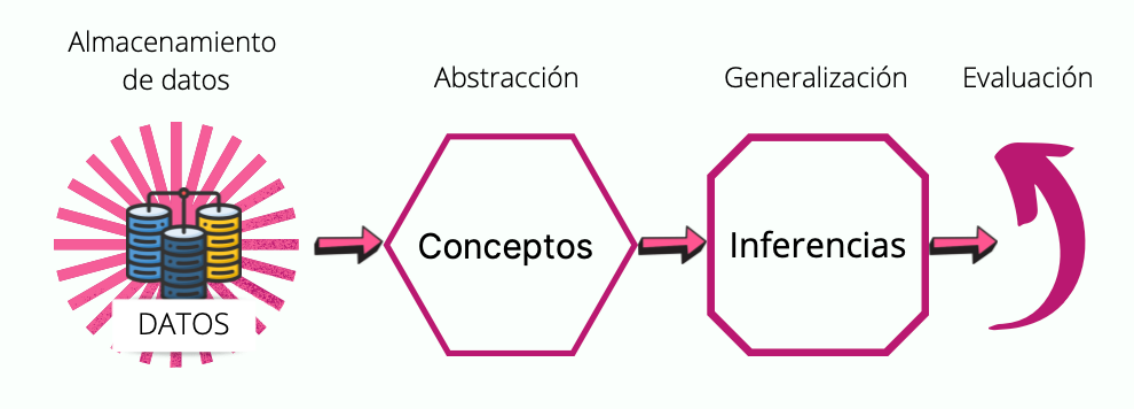

Figura 3 Etapas proceso aprendizaje

Fuente: Elaboración propia.

1. Almacenamiento de datos
   Las facilidades para almacenar y recuperar grandes cantidades de datos son una parte importante del proceso de aprendizaje. Tanto los humanos como las computadoras utilizan el almacenamiento de datos como base para el razonamiento avanzado.
   En los humanos, los datos se almacenan en el cerebro y se recuperan mediante señales electroquímicas. Las computadoras usan discos duros, memoria flash, memoria de acceso aleatorio y otros dispositivos similares para almacenar datos además de otras tecnologías para recuperarlos.

2. Abstracción
   El segundo componente del proceso de aprendizaje se llama abstracción. La abstracción es el proceso de extraer conocimiento sobre los datos almacenados. Esto implica crear conceptos generales sobre todos los datos. La creación de conocimiento implica la aplicación de modelos conocidos y la creación de nuevos modelos. El proceso de ajustar un modelo a un conjunto de datos se llama entrenamiento. Una vez que se entrena el modelo, los datos se transforman en una forma abstracta que resume la información original.

3. Generalización
   El tercer componente del proceso de aprendizaje se llama generalización. El término generalización describe el proceso de convertir el conocimiento sobre los datos almacenados en una forma que pueda usarse para operaciones futuras. Estas operaciones se realizarán en tareas similares pero no idénticas a las vistas anteriormente. En pocas palabras, el objetivo es descubrir los atributos de datos más relevantes para futuras tareas.

4. Evaluación
   La evaluación es el componente final del proceso de aprendizaje. Es el proceso de proporcionar retroalimentación a los usuarios para medir la utilidad de lo aprendido. Esta retroalimentación se utiliza para mejorar el proceso general de aprendizaje.

Como con cualquier método, hay diferentes formas de entrenar algoritmos de aprendizaje automático, cada una con sus propias ventajas y desventajas. Para entender los pros y los contras de cada tipo de aprendizaje automático, primero debemos ver con qué tipo de datos se entrenan los algoritmos. En el aprendizaje automático, hay dos tipos de datos: los datos etiquetados y los datos no etiquetados.
Los datos etiquetados tienen tanto los parámetros de entrada como los de salida en un patrón completamente legible por la máquina, pero requieren mucho trabajo humano para etiquetar los datos, para empezar. Los datos sin etiquetar sólo tienen uno o ninguno de los parámetros en un formato legible por la máquina. Esto anula la necesidad de trabajo humano, pero requiere soluciones más complejas.
También hay algunos tipos de algoritmos de aprendizaje automático que se utilizan en casos de uso muy específicos, pero hoy en día se utilizan tres métodos principales.

2.1.1 El Aprendizaje supervisado

El aprendizaje supervisado es uno de los tipos más básicos de aprendizaje automático. En este tipo, el algoritmo de aprendizaje automático se entrena con datos etiquetados. Aunque los datos deben estar etiquetados con precisión para que este método funcione, el aprendizaje supervisado es extremadamente potente cuando se utiliza en las circunstancias adecuadas.

En el aprendizaje supervisado, el algoritmo de ML recibe un pequeño conjunto de datos de entrenamiento para trabajar. Este conjunto de datos de entrenamiento es una parte más pequeña del conjunto de datos más grande y sirve para dar al algoritmo una idea básica del problema, la solución y los puntos de datos con los que debe tratar. El conjunto de datos de entrenamiento también es muy similar al conjunto de datos final en sus características y proporciona al algoritmo los parámetros etiquetados necesarios para el problema [6].

A continuación, el algoritmo encuentra relaciones entre los parámetros dados, estableciendo esencialmente una relación de causa y efecto entre las variables del conjunto de datos. Al final del entrenamiento, el algoritmo tiene una idea de cómo funcionan los datos y la relación entre la entrada y la salida.

A continuación, esta solución se despliega para su uso con el conjunto de datos final, del que aprende de la misma manera que del conjunto de datos de entrenamiento. Esto significa que los algoritmos de aprendizaje automático supervisado seguirán mejorando incluso después de ser desplegados, descubriendo nuevos patrones y relaciones a medida que se entrenan con nuevos datos.

2.1.2 El aprendizaje no supervisado

El aprendizaje automático no supervisado tiene la ventaja de poder trabajar con datos no etiquetados. Esto significa que no es necesario el trabajo humano para hacer que el conjunto de datos sea legible por la máquina, lo que permite que el programa trabaje con conjuntos de datos mucho más grandes.

En el aprendizaje supervisado, las etiquetas permiten al algoritmo encontrar la naturaleza exacta de la relación entre dos puntos de datos cualquiera. Sin embargo, el aprendizaje no supervisado no dispone de etiquetas para trabajar, lo que da lugar a la creación de estructuras ocultas. Las relaciones entre los puntos de datos son percibidas por el algoritmo de forma abstracta, sin que se requiera la intervención de los seres humanos.

La creación de estas estructuras ocultas es lo que hace que los algoritmos de aprendizaje no supervisado sean versátiles. En lugar de un enunciado de problema definido y establecido, los algoritmos de aprendizaje no supervisado pueden adaptarse a los datos cambiando dinámicamente las estructuras ocultas. Esto ofrece más desarrollo posterior a la implementación que los algoritmos de aprendizaje supervisado [4].

2.1.3 El aprendizaje por refuerzo

El aprendizaje por refuerzo se inspira directamente en la forma en que los seres humanos aprenden de los datos en su vida. Se trata de un algoritmo que se mejora a sí mismo y aprende de las nuevas situaciones mediante un método de ensayo y error. Los resultados favorables se estimulan o "refuerzan", y los no favorables se desaniman o "castigan".
Basado en el concepto psicológico de condicionamiento, el aprendizaje por refuerzo funciona poniendo al algoritmo en un entorno de trabajo con un intérprete y un sistema de recompensas. En cada iteración del algoritmo, el resultado de la salida se da al intérprete, que decide si el resultado es favorable o no.
En caso de que el programa encuentre la solución correcta, el intérprete refuerza la solución proporcionando una recompensa al algoritmo. Si el resultado no es favorable, el algoritmo se ve obligado a reiterar hasta encontrar un resultado mejor. En la mayoría de los casos, el sistema de recompensa está directamente ligado a la eficacia del resultado.
En los casos de uso típicos del aprendizaje por refuerzo, cómo encontrar la ruta más corta entre dos puntos de un mapa, la solución no es un valor absoluto. En cambio, adquiere una puntuación de eficacia, expresada en un valor porcentual. Cuanto más alto sea este valor porcentual, mayor será la recompensa que reciba el algoritmo. Así, el programa se entrena para dar la mejor solución posible a cambio de la mejor recompensa posible.

2.2 Algoritmos de aprendizaje automático

A grandes rasgos tenemos dos tipos de problemas, clasificación y regresión. La clasificación consiste en agrupar los datos en clases. Cuando el algoritmo de aprendizaje supervisado etiqueta los datos de entrada en dos clases distintas, se denomina clasificación binaria. La clasificación múltiple consiste en clasificar los datos en más de dos clases. Sin embargo, la regresión es el tipo de aprendizaje supervisado que aprende de los conjuntos de datos etiquetados y es capaz de predecir una salida de valor continuo para los nuevos datos dados al algoritmo. Se utiliza cuando la salida requerida es un número, como el dinero o la altura, etc. A continuación, nos centraremos en los principales algoritmos de aprendizaje.

- *Decision Tree*: Los árboles de decisión clasifican basándose en los valores de las características. Utilizan el método de la ganancia de información y descubren qué característica del conjunto de datos proporciona la mejor información, la convierten en el nodo raíz y así sucesivamente hasta que son capaces de clasificar cada instancia del conjunto de datos. Cada rama del árbol de decisión representa una característica del conjunto de datos. Es uno de los algoritmos más utilizados para la clasificación [8].

- *Naives Bayes*: Los algoritmos de Naive Bayes asumen que las características del conjunto de datos son todas independientes entre sí. Funcionan muy bien en conjuntos de datos grandes. Los grafos acíclicos dirigidos (DAG) se utilizan para la clasificación [8].

- *Support Vector Machine* (SVM): Los algoritmos SVM se basan en la teoría del aprendizaje estadístico de Vap Nik. Utilizan funciones Kernel que son un concepto central para la mayoría de las tareas de aprendizaje. Estos algoritmos crean un hiperplano que se utiliza para clasificar las dos clases entre sí.

- Regresión lineal: Este algoritmo asume que existe una relación lineal entre las 2 variables, Entrada (X) y Salida (Y), de los datos de los que ha aprendido. La variable de entrada se denomina variable independiente y la de salida, variable dependiente. Cuando se pasan datos no vistos al algoritmo, éste utiliza la función, calcula y asigna la entrada a un valor continuo para la salida [8].

- Regresión logística: Este algoritmo predice valores discretos para el conjunto de variables independientes que se le han pasado. Realiza la predicción asignando los datos no vistos a la función logit que se ha programado en él. El algoritmo predice la probabilidad de los nuevos datos, por lo que su salida se encuentra entre el rango de 0 y 1 [8].

- KNN (*K-Nearest Neighbors*): KNN es un método no paramétrico utilizado para la clasificación. Aunque también puede ser utilizado para regresión. El principio de este algoritmo es que los datos conocidos se ordenan en un espacio definido por las características seleccionadas. Cuando se suministra un nuevo dato al algoritmo, éste compara las clases de los k datos más cercanos para determinar la clase del nuevo dato. La mayor ventaja de la clasificación KNN es su simplicidad, además de ser un método eficiente. Sin embargo, a pesar de su eficiencia, los tiempos de cálculo pueden ser largos con grandes bases de datos, la determinación del número de vecinos a utilizar (k) requiere ensayo y error y el algoritmo es débil con los valores atípicos, lo que puede afectar fuertemente a su eficiencia [8].

- *Boosting*: es una técnica general para crear un conjunto de modelos. Se suele utilizar con árboles de decisión y requiere mucho cuidado en su aplicación. En los modelos de regresión lineal, los residuos se examinan a menudo para ver si se puede mejorar el ajuste. Boosting lleva este concepto mucho más allá y se ajusta a una serie de modelos, en los que cada modelo sucesivo busca minimizar el error del modelo anterior. Hay varias variantes del algoritmo, entre ellas las más frecuentes son Adaboost, gradient Boosting y XGBoost(se trata del software más utilizado de dominio público para boostng [8].

2.2.1 Redes neuronales

Para las redes neuronales decidimos extendernos un poco más dada su relevancia en nuestro proyecto.

A pesar de que da la sensación de que estamos hablando de algo novedoso, esta técnica tiene origen en 1943 por McCulloch & Pitt. Algunas de las técnicas más

utilizadas hoy en día como el algoritmo *Perceptron* de 1958 o el *Backpropagation Werbos* de 1974 no fueron utilizadas antes por la falta de recursos Hardware [4].

Las redes neuronales son un conjunto de algoritmos, modelados a la manera del cerebro humano, que están diseñados para reconocer patrones. Interpretan los datos sensoriales a través de una especie de percepción mecánica, etiquetando o agrupando la información en bruto. Los patrones que reconocen son numéricos, contenidos en vectores, a los que deben traducirse todos los datos del mundo real, ya sean imágenes, sonidos, textos o series temporales [4].

Las redes neuronales nos ayudan a agrupar y clasificar. Se puede pensar en ellas como una capa de agrupación y clasificación sobre los datos que se almacenan y gestionan. Ayudan a agrupar los datos no etiquetados según las similitudes entre las entradas de ejemplo, y clasifican los datos cuando tienen un conjunto de datos etiquetados con el que entrenar.

2.3 Procesamiento de imágenes

El procesamiento digital de imágenes consiste en la manipulación de imágenes mediante ordenadores digitales. Su uso ha aumentado exponencialmente en las últimas décadas. Sus aplicaciones van desde la medicina hasta el entretenimiento, pasando por el procesamiento geológico y la teledetección. Los sistemas multimedia, uno de los pilares de la moderna sociedad de la información, dependen en gran medida del procesamiento digital de imágenes.

La disciplina del procesamiento digital de imágenes es muy amplia y abarca tanto técnicas de procesamiento digital de señales como técnicas específicas de las imágenes. Una imagen puede considerarse como una función f (x, y) de dos variables continuas x e y. Para ser procesada digitalmente, tiene que ser muestreada y transformada en una matriz de números. Dado que un ordenador representa los números con una precisión finita, estos números tienen que ser cuantificados para ser representados digitalmente. El procesamiento de imágenes pues, consiste en la manipulación de esos números de precisión finita. El procesamiento de imágenes digitales puede dividirse en varias clases: mejora de imágenes, restauración de imágenes, análisis de imágenes y compresión de imágenes. En la mejora de imágenes, se manipula una imagen, sobre todo mediante técnicas heurísticas, para que un observador humano pueda extraer información útil de ella. Las técnicas de restauración de imágenes tienen como objetivo procesar imágenes corruptas de las que se tiene una descripción estadística o matemática de la degradación para poder revertirla. Las técnicas de análisis de imágenes permiten procesar una imagen de forma que se pueda extraer información de ella automáticamente. Ejemplos de análisis de imágenes son la

segmentación de imágenes, la extracción de bordes y el análisis de textura y movimiento.

Con redes neuronales básicas ya podemos trabajar con imágenes si las convertimos en un vector, pero, esto no es suficiente cuando hablamos de una gran cantidad de píxeles, de ahí surgen las redes convolucionales. La idea de la convolución es sin ir más lejos pasarle un "filtro" (*kernel*) a cada píxel de la imagen. Los valores de este *kernel* son los valores que tiene que aprender mi modelo, es decir los coeficientes.

Uno de los hiperparámetros que se introduce en las redes convolucionales es el *stride* que nos dice los saltos que toma el filtro para pasar de un píxel a otro.

Uno de los problemas que surgen es que los filtros no tienen en cuenta los bordes por lo que las imágenes se ven reducidas perdiendo así información. Lo que se hace para solucionar este problema es una técnica conocida como *Zero padding,* que no es más que añadir a nuestra imagen un borde de 0 preservando así toda la información de la fotografía al pasar el filtro en las primeras capas.

La estructura de estas redes se caracteriza por los bloques convolucionales: la salida (mapa de activación) de la primera capa convolucional será la entrada de la segunda capa, y así sucesivamente

La salida de la segunda capa convolucional serán las activaciones que representan características de más alto nivel.

Como muestra la Figura 7, dentro de estas redes neuronales encontramos los siguientes tipos de capas:

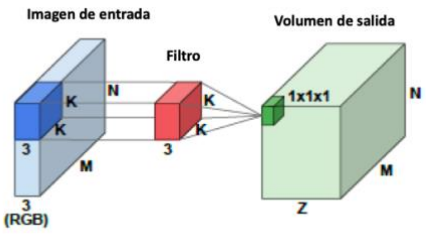

Figura 4 Ejemplo Capa convolucional

Fuente: Diapositivas curso Deep Learning ofrecido por la upv.

- Capa convolucional
- Capa de activación (relu, sigmoid, softmax…)
- Capa *pooling*: para reducir tamaño así menos parámetros y menos coste computacional. Además, así controlamos el *overfitting.* Aunque existen varias formas para hacer esta reducción la máx típica es el Max pooling. De esta forma estoy reduciendo información, pero de forma controlada.
- Capa *Fully-connected*: nuestro top model.

2.4 *Engagement* en redes sociales

La aparición de las plataformas de medios sociales ha modificado drásticamente el papel de los clientes de las empresas, que han pasado de ser observadores pasivos de los contenidos a participantes activos, que ahora son los coproductores y cocreadores de contenidos a través de sus interacciones y comportamientos en línea. El comportamiento que refleja el compromiso con los medios sociales incluye la creación, la contribución o el consumo de contenido relacionado con la marca por parte de los clientes dentro de una red social. El grado de compromiso difiere de los tipos simples de compromiso (por ejemplo, "me gusta" una publicación en Instagram) a los tipos más altos de compromiso de los clientes en las actividades de cocreación (por ejemplo, la publicación de reseñas) [9].

La creciente prevalencia de las redes sociales ha hecho que tanto los académicos como los profesionales se centren en el concepto de compromiso en las plataformas de redes sociales. El compromiso de los clientes se ha estudiado en muchos campos, como la psicología, la educación, la gestión, el marketing y los sistemas de información.

En el contexto del marketing digital, algunos investigadores como Brodie y Hollebeek [10] el concepto de compromiso e intentaron validar empíricamente sus escalas de medición. Van Doorn [11] lo definen como "las manifestaciones de comportamiento de un cliente que tienen un enfoque de marca o de empresa, más allá de la compra." Existen varios objetos focales del compromiso del cliente, entre los que se encuentran las ofertas de productos o servicios, las actividades y los eventos y los medios de comunicación. Algunos reconocieron las propiedades del *engagement,* como la intensidad, y examinaron sus diversos aspectos y antecedentes. El *engagement* es interactivo y dependiente del contexto y se entiende a través del examen de cada uno de sus diversos aspectos, conocidos como experiencias de servicio. Según Dolan y Conduit [9], existen seis tipos de comportamientos de compromiso con los medios sociales en las páginas de fans: crear, contribuir, destruir (conocidos como comportamientos de compromiso activo), consumir, dormitar y desapegarse (conocidos como formas pasivas y/o más individualizadas de compromiso).

Las características interactivas del comportamiento de compromiso pueden conducir a diferentes niveles de intensidad. En un estudio más reciente sobre los niveles de intensidad del *engagement,* Dolan y Conduit sugieren dos tipologías que engloban los seis grupos de comportamiento (referidos anteriormente). Según argumentan, puede ser pasivo (bajo) o activo (alto) y también positivo o negativo. El compromiso pasivo se define a través del comportamiento de un miembro que navega por un grupo en línea y aprovecha al máximo los beneficios accesibles, mientras que no participa en ninguna actividad de la comunidad. Por el contrario, el compromiso activo se determina a través del comportamiento de los miembros altamente interesados en comprometerse en una comunidad online participando en actividades, creando mensajes, difundiendo información y proporcionando apoyo emocional a los demás.

Existen diferentes métricas para medir la intensidad de la pasividad o la activación del compromiso con los medios sociales. Por ejemplo, Alhabash [12] describen el gusto (como "respuesta afectiva") y los comentarios como ("deliberación activa y pública"), como comportamientos activos en los medios sociales, mientras que la lectura de contenidos y los clics son ejemplos de comportamientos de compromiso pasivo.

3. Herramientas

En esta sección se enumeran y explican brevemente los elementos que han sido necesarios para la realización de este proyecto. Estos elementos los hemos dividido en *Hardware, Software* y conjunto de datos.

3.1 Hardware

Para la realización de este trabajo se ha utilizado un MacBook Pro de 14'' con un procesador Apple M1Pro con CPU de 8 núcleos, 6 núcleos de rendimiento y 2 de eficiencia; GPU de 14 núcleos, *Neural Engine* de 16 núcleos y 200 GB/s de ancho de banda de memoria [13].

Además, para el entrenamiento de las redes neuronales se ha hecho uso de *Google Colaboratory*. Se trata de una plataforma online ofrecida por Google que permite ejecutar código Python utilizando el *hardware* del propio Google en la nube, lo cual ha permitido acelerar el proceso de entrenamiento respecto a la ejecución en local.

3.2 Software

3.1.1 Lenguaje de programación y entorno de desarrollo

El lenguaje escogido para realizar este proyecto ha sido Python en su última versión disponible (3.11), tanto para la obtención de datos como para el entrenamiento de los modelos propuestos de *Deep Learning.*

Los principales motivos por los que se ha elegido este lenguaje se detallan a continuación:

- Se trata de un lenguaje muy sencillo y fácil de aprender
- El curso realizado de *Deep Learning* requería conocimientos en Python
- Una de las herramientas utilizadas para la construcción del Data-set requiere Python
- Ofrece un montón de posibilidades en relación a librerías
- Es uno de los programas más utilizados mundialmente y podría facilitarnos una futura integración con *software* ya existente en el futuro.

El entorno de desarrollo elegido para la realización del proyecto es *VisualStudio Code*, uno de los entornos más conocidos y fáciles de utilizar. Ofrece un montón de extensiones que facilitan el desarrollo y es muy intuitivo.

Para la creación de los modelos se ha trabajado en el entorno de *Google Colaboratory* mencionado anteriormente.

3.1.2 Librerías utilizadas

A continuación, se detallarán las librerías que han sido utilizadas para la realización del proyecto:

- TensorFlow: Es un f*ramework* desarrollado y mantenido por Google que permite la ejecución de operaciones matemáticas, mediante diagramas de flujo de datos, de una forma optimizada en una CPU o GPU (en GPU todo más rápido). Es flexible, escalable y en constante actualización y mantenimiento.

- Keras: El objetivo de la biblioteca es acelerar la creación de redes neuronales. Para conseguir este objetivo Keras funciona como una API que permite acceso a frameworks como TensorFlow: para ello, Keras no funciona como un *framework* independiente, sino como una interfaz de uso intuitivo (API) que permite acceder a varios *frameworks* de aprendizaje automático y desarrollarlos. Entre los *frameworks* compatibles con Keras, se incluyen Theano, Microsoft Cognitive Toolkit (anteriormente CNTK) y TensorFlow.

- Pandas: Pandas es una biblioteca de Python que se utiliza para trabajar con conjuntos de datos. Tiene funciones para analizar, limpiar, explorar y manipular datos. Pandas nos permite analizar *big data* y sacar conclusiones basadas en teorías estadísticas. Además, puede limpiar conjuntos de datos desordenados y hacerlos legibles y relevantes.

- NumPy: Numpy te permite crear una estructura universal de datos para facilitar su análisis e intercambio de varios algoritmos. Implementa vectores multidimensionales y matrices para almacenar grandes cantidades de datos. Cuenta con funciones matemáticas y estructuras de datos de alto nivel.

- Scikit-learn: Scikit-learn (Sklearn) es la biblioteca más útil y robusta para el *machine learning* en Python. Proporciona una selección de herramientas eficientes para el aprendizaje automático y el modelado estadístico, incluyendo la clasificación, la regresión, la agrupación y la reducción de la dimensionalidad a través de una interfaz consistente en Python. Esta biblioteca, escrita en gran parte en Python, está construida sobre NumPy, SciPy y Matplotlib.

- OpenCV: es una librería de código abierto para la visión por ordenador y el aprendizaje automático. OpenCV fue construido para proporcionar una infraestructura común para las aplicaciones de visión por ordenador y para acelerar el uso de la percepción de la máquina en los productos comerciales. La biblioteca cuenta con más de 2500 algoritmos optimizados, que incluyen un amplio conjunto de algoritmos de visión por ordenador y aprendizaje automático, tanto clásicos como de última generación. Estos algoritmos pueden utilizarse para detectar y reconocer caras, identificar objetos y clasificar acciones humanas en vídeos...

- Instaloader: se trata de una librería de código libre que te permite descargar imágenes y datos de la aplicación de Instagram de una manera intuitiva y sencilla. Aunque se trata de una librería muy útil presenta ciertas limitaciones que se comentaran posteriormente.

4. Análisis del problema

El crecimiento exponencial del uso de las redes sociales en distintas áreas profesionales como el *marketing* han hecho que los creadores de estas aplicaciones pongan precio al uso de funcionalidades especiales para empresas, como por ejemplo para poder poner anuncios en la propia plataforma o tener un mayor alcance de sus publicaciones.

Además, la última novedad de Instagram, que se encuentra ahora en fase experimental en Estados Unidos, es la oferta de suscripciones a creadores de contenido de renombre. A través de esta suscripción los usuarios podrán acceder a contenido exclusivo no disponible para los demás usuarios. Por lo tanto, ya no se podrá consumir de manera gratuita todo el contenido de la aplicación. Uno de los mayores atractivos de la utilización de redes sociales en campañas de Marketing es el bajo coste con el que se consigue llegar a un público mayor. Y ese coste, cada vez es más alto para conseguir *engagement.*

Otro problema es que el tipo de información compartida en redes sociales cada vez es más de tipo multimedia, es decir, los usuarios comparten fotos, videos, reels, streaming… Una gran parte de la información que se genera está en datos multimedia, y todavía no se está explotando.

Existen varias aplicaciones que proporcionan análisis y métricas sobre cuentas, como por ejemplo *metricol* [14] aunque no se utiliza toda esta información multimedia. Sin embargo, solo hemos dado con dos soluciones que parecen utilizar este tipo de datos, una de ellas orientada a clasificación y la otra, simplemente, especializada en imágenes de viajes.

- *LikelyAI* [15] fue desarrollada por antiguos empleados de Facebook y Google, extrae miles de puntos de datos de una imagen y reconoce los patrones populares. Objetos, colores, emociones, formas, iluminación, tamaño y posición: todos ellos son puntos de datos. El usuario tiene que seleccionar varias fotos, luego el algoritmo evalúa cada una de ellas y finalmente la aplicación clasifica las fotos y muestra los resultados.

- *Beautiful Destinations* [16], una marca de viajes y estilo de vida que tiene más de 10,5 millones de seguidores en 180 países construyó un algoritmo [17] que indica cuántos *likes* y comentarios recibirá una foto. Crearon el algoritmo recopilando un montón de fotos de muchas redes sociales (Instagram, Flicker y Pinterest) que tenían reacciones para hacer correlaciones. Este software ayuda a los *community managers* a elegir la foto de Instagram que más *likes* obtendrá. El hecho es que la aplicación es sólo para uso de la empresa, no es pública. El algoritmo está entrenado sólo con fotos de viajes y estilo de vida, y por esta razón, el algoritmo funciona mejor en este contexto. *Beautiful Destinations* ya está trabajando en una nueva tarea: predecir el compromiso de la audiencia con los vídeos.

La primera solución a las problemáticas comentadas anteriormente es la explotación de todos los datos de los que se dispone, ya que de esta manera se generaría mucho más conocimiento. Este incremento del conocimiento puede traducirse en un rendimiento más optimo en las redes sociales, y así, permitir un crecimiento orgánico sin necesidad de aumentar los costes.

Dada la amplitud del problema planteado, en este proyecto se va a delimitar el escenario del problema:

- La red social elegida para el estudio será Instagram, una popular aplicación de redes sociales centrada en compartir fotos y vídeos. Existe desde 2010 y ha mantenido un alto nivel de popularidad gracias a la incorporación de nuevas funciones innovadoras, como las Historias de Instagram, las compras, los Reels, etc. Nos hemos decantado por esta red social, entre otras cosas, por el alto porcentaje de uso en estrategias de marketing (ya mencionado en el apartado 1.2) y la gran cantidad de contenido multimedia que ofrece.
- En línea con *Beautiful Destinations*, este TFG se centrará en un solo dominio para predecir la popularidad de las imágenes. En concreto, nos hemos decantado por el sector de running, un deporte del que cada vez se habla más y en el último año, registró un crecimiento del 5,13% en nuestro país [18].

Para ponerle solución a los problemas planteados se va a realizar una aproximación de un sector concreto prediciendo el *engagement* incluyendo características extraídas de los datos multimedia, en concreto, de las imágenes. En definitiva, el objetivo consistirá en predecir el *engagement* en Instagram a través de los *likes*, ya que se trata de la acción más común realizada por el usuario.

Para ello, el primer paso ha sido construir el dataset como que detallamos en el siguiente apartado. Guiándonos con la metodología CRISP-DM, estudiaremos la predicción del *engagement* a través de diferentes modelos, observando cómo se comportan distintas combinaciones de variables. Además, incluiremos conocimiento extraído de las imágenes recogidas, con la intención de comprobar si es relevante la información que nos proporcionan las imágenes a la hora de entrenar los diferentes modelos.

A través de los siguientes apartados de la memoria se irá detallando paso a paso y de manera ordenada todo el proceso seguido a lo largo de este proyecto.

5. Preparación y comprensión de los datos

Para realizar este estudio se ha creado un *dataset* propio a partir de los datos proporcionados por la APP de Instagram. El proceso de creación del *dataset* ha hecho que se profundice e indague extensamente sobre la API de esta aplicación, así como el manejo y el tratamiento de los datos.

En una primera lluvia de ideas, en la que se decidió como se iban a obtener los datos de la aplicación para el proyecto, se propuso identificar los *posts* que contuvieran el *hashtag* #running para así, descargar la imagen junto con los metadatos necesarios.
Tras una búsqueda para ver qué opciones se tenía a la hora de conseguir la información de interés, encontramos un software libre que facilitaba la extracción de información de Instagram a través de la API Instaloader [19].

Cabe mencionar que se estudió la viabilidad de utilizar directamente la API de Instagram para generar el *dataset*, pero esta opción fue descartada al no obtener los permisos necesarios para la extracción de toda la información. Además, el proceso hubiera sido mucho más costoso en lo que al tiempo se refiere y hubiéramos seguido teniendo la limitación de peticiones permitidas a la API.

Al empezar a utilizar un pequeño *script* que facilitaba la descarga y el manejo de metadatos obtenidos, nos dimos cuenta de la limitación de peticiones que se permite hacer a la API. De este modo, solo se conseguían alrededor de 30 posts cada 24 horas. Esto se producía porque, a través del *hashtag* se accede al nodo de información del *post,* pero no al del usuario que lo ha publicado. Para acceder al nodo de información donde están los datos del propio usuario, se tenía que hacer el doble de llamadas a la API y eso producía una recopilación muy lenta de información. Esta casuística nos llevó a plantear el problema desde otro punto de vista. Dado que el problema estaba identificado, después de muchas pruebas decidimos acceder directamente al nodo de información de los usuarios, consiguiendo así alrededor de 200 posts en 24 horas.

Así pues, se decidió obtener el conjunto de datos a través de los *posts* que tienen los usuarios en su perfil, independientemente de que tuvieran el *hashtag* #running. Para ello, el primer paso fue identificar que usuarios nos podían proporcionar imágenes interesantes para nuestro predictor. Visto desde el punto de vista empresarial, los usuarios que promocionan productos y los pueden llevar a generar ventas son los *influencers* [20]. Por esta razón, se decidió que el conjunto de datos para este proyecto iba a estar formado por los *posts* publicados por *influencers* en el mundo del *running.*

A partir de este punto, comienza el proceso de creación del *dataset,* que se detallará a continuación. En este apartado se especificará el proceso seguido desde la identificación de influencers hasta la obtención del dataset final que nutrirá el modelo de redes neuronales.

5.1 Identificación de influencers

En esta primera fase, se seleccionaron los *influencers* que iban a formar parte del estudio. La primera decisión que se tomó fue no limitar la búsqueda de i*nfluencers* solo al territorio nacional. Esta decisión se respalda por la cantidad de datos que necesitamos obtener para entrenar el modelo de redes neuronales. Si bien, se han diferenciado para enriquecer el conjunto de datos obtenido y hacer un posterior análisis de esta característica.

Una vez tomada esta decisión, se inició la búsqueda, a través de internet, de publicaciones hechas en redes sociales o en páginas web especializadas en temas como el *running*, donde se listaran los influencers de running más importantes o reconocidos tanto a nivel mundial como a nivel nacional. También se incluyeron aquellas publicaciones ofrecidas por páginas web especializadas en búsqueda de influencers, como inBeat [21]. Con estos criterios conseguimos un total de 11 publicaciones que se enumeran en la Tabla 1.

	Fuente	**Título**	**Ámbito**
1	[22]	Los 11 bloggers de running más influyentes de España	Nacional
2	[23]	Las instagramers más influyentes del universo running en España	Nacional
3	[24]	8 cuentas de runners en Instagram que deberías estar siguiendo	Nacional
4	[25]	Los runners españoles más influyentes en Instagram	Nacional
5	[26]	Principales influencers del running en las redes sociales	Internacional
6	[27]	Los 6 runners más influyentes en Instagram	Internacional
7	[21]	Top 10 Runners Instagram Influencers In Spain In 2021	Nacional
8	[28]	10 global running influencers you should know	Internacional
9	[29]	Top 150 Running Instagram Influencers most followed	Internacional
10	[30]	Los corredores más influyentes del running en España	Nacional

Tabla 1 Publicaciones consultadas para la selección de influencers

Fuente: Elaboración propia.

Una vez revisadas las distintas publicaciones, se listaron todos los usuarios de Instagram en un fichero de texto, obteniendo un total de 248 perfiles. El primer paso fue eliminar los perfiles duplicados, ya que en varias publicaciones se mencionaban los mismos perfiles. El número de usuarios repetidos ascendió a 31, dejando así un total de 217 perfiles para la extracción de datos.
Para obtener los datos referentes a los 217 perfiles, se utiliza la función *get_data_influencers(file.csv)* desarrollada en Python. Esta función recibe un archivo CSV como parámetro en el cual están listados todos los perfiles de Instagram sin la arroba "@" y sin espacios en blanco. Cada fila es un usuario, y de forma iterativa va leyendo las líneas del CSV y extrayendo los datos asociados a ese usuario haciendo uso de Instaloader. Una vez recuperados los datos, generamos un fichero formado con los datos extraídos.

0_lista_influencers1_resumen

num	username	full_name	business	verified	bio	category	followers	followees	nPost	nIgtv	reels
1	heatherunz	Heather Schulz	0	0	Running through life❤️🏃‍♀️ Coach 🏃‍♀️ Psych Teacher 📚 Track Shack 🌟 26.2 x 40 Sub 3hrs x 13 Boston x 13.1 26.2 2:54 13.1 1:21		92464	521	2266	3	1
2	lucy_bartholomew	Lucy Bartholomew	0	1	Fresh air seeker 🌬️ Adventure lover⛰️ Plant muncher 🌱		105003	96	3200	23	1
3	sarahall3	Sara Hall	1	1	American record holder- Half marathon 🇺🇸 Soccer Mom to 4 from 🇺🇸 Wife to @ryanhall3	Creators & Celebrities	167841	432	1256	1	1
4	stephrothstein	Stephanie Rothstein Bruce	1	1	Professional runner for HokaOneOne NAZ Elite, PRO Compression, mom, Celiac, Co-conspirator for Picky Bars, coach of Running w/ Bruces!	Creators & Celebrities	96182	762	2354	8	1
5	trackclubbabe	KIM	0	0	RUN ❤️er [26.2- 6:08🔥3:11⚡13.1- 1:29] here to help ya get FASTER & LOVE running 💌trackclubbabe@gmail.com track your miles w/ @runkeeper app 👇		128816	283	2064	3	1
6	ian.morgan	Ian Morgan | Ultra Runner	0	0	🏴Ultra Runner ❤️🤍 Mi mujer @frangibar Based in Sitges/Spain 🇪🇸 Traveler🌍 Coffee❤️ @cimalpofficial @ultraxco @sequentworld		121772	2330	2604	20	1
7	foodfitnessflora	Flora Beverley 🌺🌱	0	1	Fitness | Food | Sustainability 👁️ Collab: Hannah@wmodel.co.uk 🏠 @backtobristol @leosbox 🍵 florabeverley@gmail.com @theforwardlab writer 🤍YOUTUBE/BLOG⬇		114672	1221	4144	6	1
8	des_linden	desiree linden	1	1	201∞ Boston Marathon Champion. 2xUS Olympian. Coffee aficionado @coffeebyit. Whiskey connoisseur. Music junkie. Book nerd. Travel enthusiast.	Creators & Celebrities	181959	1155	496	7	1
9	runrx	RunRx | Learn to run pain free	1	0	RunRxStrong Membership Skill-Strength-Injury prevention Learn how to run and stay pain free! 👇@runrx #runrx	Personal Goods & General Merchandise Stores	211008	100	6736	103	1
10	karagoucher	Kara Goucher	1	1	Mom, wife to @adamgoucher, obsessed runner. "Be the change." 2 x Olympian, World Championship 🥉Athlete-Advisor @oiselle, @altrarunning	Creators & Celebrities	188103	502	1875	4	1
11	jennifalconer	Jenni Falconer	1	1	@smoothradio Breakfast 💜 Founder podcast 'RunPod'🏃‍♀️🎧 Co-Founder @kollohealth Runner/Golfer/Mum Enquiries-tap ⬇️ https://tinyurl.com/57yju8u4	Creators & Celebrities	247719	6263	5914	53	1
12	tommy_rivs	tommy_rivs 🇲🇽	1	1	keep calm & rage on #teamrivs	Creators & Celebrities	327573	1077	1058	17	1
13	courtneydauwalter	Courtney Dauwalter	1	1	Ultra runner with a love for sunshine, long inseams, and candy. 🏃‍♀️ @salomonrunning 🧃 @tailwindnutrition 👑 @kodiakcakes	Creators & Celebrities	337598	931	198	3	1
14	shalaneflanagan	Shalane Flanagan	1	1	Mom💙Nike Coach🏃‍♀️ BTC|4XOlympian|Olympic Silver Medalist |NYC Marathon champion|American record holder|NYTimes best selling author X3 Run Fast Eat Slow	Creators & Celebrities	420429	407	1173	10	1
15	corroysoymujer	Erica ··· Corro y Soy Mujer ···	0	0	🎤📰Periodista 🏃‍♀️🏃Runner 📚 ¡He escrito un libro! 🐕 Sí a los retos ⏳Postureo 🏃‍♀️ Mi otro proyecto ⬇️ 🤘@malditorunning 🤘 Mi libro 📖📍		32343	2669	3197	6	1
16	pitufollow	SERGIO TURULL	0	1	🎁 Adidas Athlete 🚀 Foodspring -15%: pitufollowFSG 📩 sergiopitufollow@gmail.com 📖 Autor #Corroporquecaminarmecansa		92309	963	3360	25	1
17	martaggrun	*Marta Guerrero*	0	0	🚵 Ultra runner LOEKS 💚 Rank de 6 🦄 **ASICS Frontrunner** 🏃‍♀️ **Spain** 🥇 @coros_iberia ⚡ @compexspain 🚀 @aonijie.es 📩 *martaggrun@gmail.com*		38491	3501	1151	3	1
18	bielrafols	Biel Ràfols	1	0	🔸Sports Mkt @salomon.spain 🏔️ 🔸Orienteering lover 🗺️🧭 🔸Camera-runner 🏃‍♀️📹 shortur.at/vFNV3	General Interest	20143	987	1400	68	1

Figura 5 Datos obtenidos sobre los influencers

Fuente: Elaboración propia.

Como se puede observar en la figura 5, los datos recuperados son:

- username: El nombre de usuario.
- full_name: El nombre completo del usuario.
- nPost de la cuenta: El número de posts total visibles que el usuario ha publicado.
- Bio: Descripción de la cuenta.
- Followers: El número de perfiles que siguen al usuario.
- Followees: El número de perfiles que el usuario sigue.
- igtvCount: El número de vídeos subidos a igtv.
- business: Devuelve '1' si la cuenta es un perfil de negocio y '0' en caso contrario.
- category: La categoría de negocio si se trata de un perfil de negocio.
- verified: Devuelve '1' si la cuenta es un perfil verificado y '0' en caso contrario.

Al ejecutar la función, no se encontraron 6 perfiles. Después de verificar que los perfiles no existían, seguimos adelante con una lista con 211 perfiles.

Para conseguir un dataset lo más consistente posible se estableció que, del listado inicial, nos interesaban aquellos perfiles que tuvieran más de 10.000 seguidores, ya que son los perfiles que consiguen mayor *engagement* [31]. Filtrando los datos como se muestra en la figura 6, es decir, seleccionando solo los perfiles con más de 10.000 seguidores, nos deja un total de 165 perfiles al excluir 46 usuarios por no cumplir dicha condición.

```
import pandas as pd

df = pd.read_csv("lista_influencers_datos.csv", sep= ';', index_col= 'num',  on_bad_lines='skip')

#filas con la columna followers mayor que 10.000

df_mas_10000 = df_V[df_V["followers"] > 10000]

df_mas_10000.to_csv('influencers_mas_10mil_datos.csv')
```

Figura 6 Filtrado de datos para afinar el dataset a perfiles con más de 10.000 seguidores

Fuente: Elaboración propia.

Por último, fue necesario inspeccionar los 165 perfiles uno a uno desde la App de Instagram para asegurarnos de que los usuarios compartían contenido sobre running frecuentemente. Este paso se ha llevado a cabo porque en los listados previos había perfiles como el de Cristina Pedroche o Patricia Montero. Estos perfiles no comparten contenido sobre running ya que su actividad principal es otra.

Por ejemplo, en la figura 7 (izquierda) se puede observar como el perfil de Cristina Pedroche, más conocida por los atuendos utilizados para despedir el año, comparte publicaciones aleatorias sobre su vida. No se trata de un perfil enfocado a compartir publicaciones sobre running o temáticas relacionadas. Sin embargo, en la ilustración 3 (derecha) se muestra el perfil de Kelly Holmes en el que, sí se pueden localizar diferentes publicaciones de ella corriendo, lo que nos confirma que se trata de una cuenta donde efectivamente se puede consumir contenido relacionado con el running.

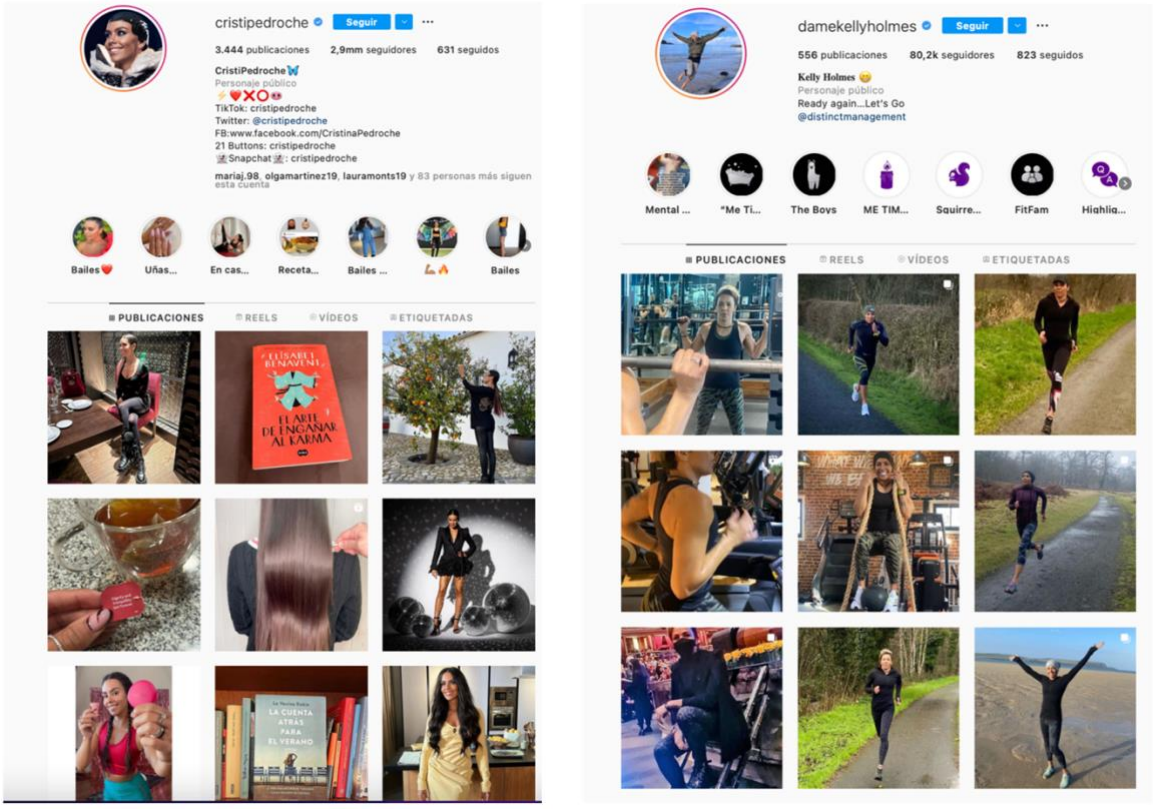

Figura 7 Perfiles de Cristina Pedroche y Kelly Holmes a 26/01/2022.

Fuente: Elaboración propia.

Al comprobar todos los perfiles de la lista previa (influencers_mas_10mil_datos.csv) se han identificado 17 perfiles en los que no se comparte contenido relacionado con el running en sus 30 posts más recientes, por tanto, no se ha contado con ellos para la creación del dataset. Resumiendo, la lista final de influencers consta de 149 perfiles.

5.2 Descarga de los posts

El siguiente paso, con los influencers ya identificados, es decidir cómo se van a descargar los posts de cada usuario.

En primer lugar, se tomó la decisión de descargar los posts de un año entero, del 1 de enero de 2021 al 31 de Diciembre se 2021. La razón principal es que de cara a un análisis futuro se pueda disponer del comportamiento cronológico de los usuarios.

Con el objetivo de optimizar el proceso de descarga y automatizarlo lo máximo posible, construimos un pequeño script: *get_post_and_data.py*. Este script consta de tres funciones, la principal y dos de descarga. La función principal contiene el flujo y lógica seguida para todo el proceso de descarga y almacenamiento de la información. En cambio, las dos funciones restantes son las llamadas al propio Instaloader.

Para entender bien como hemos enfocado este proceso vamos a ilustrarlo con el esquema de la figura 8

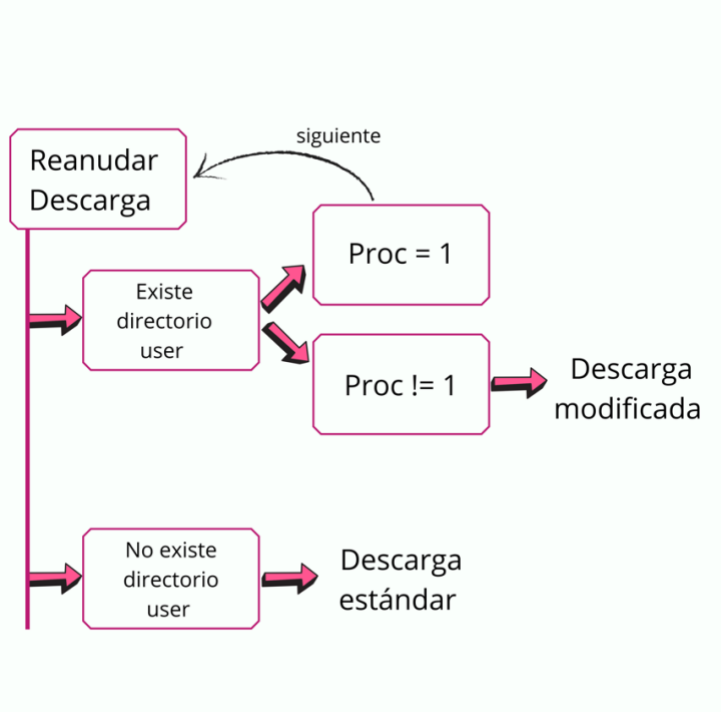

Figura 8 Lógica del script get_post_and_data.py

Fuente: Elaboración propia.

La lógica seguida para la construcción de la función principal se ha hecho de manera que podamos controlar donde se interrumpe la descarga cuando llegamos al límite de peticiones hechas a la API en un día. Para almacenar todas las imágenes que

descarguemos se ha decidido crear una carpeta por usuario, de esa manera podemos tener localizadas las imágenes por usuario para futuros análisis.

A la función *reanudar_descarga* le pasamos como parámetro un fichero CSV con 2 columnas: “user” y “proc”. La primera contiene todos los perfiles de los cuales queremos descargar *posts*, la segunda inicialmente está toda a 0. Esta columna se ha añadido para controlar cuando se ha terminado de procesar un usuario, es decir, se han descargado todas sus imágenes de enero a diciembre de 2021. Lo primero que hace la función es leer este fichero, y fila a fila vamos trabajando. Por cada fila comprueba si existe una carpeta que se llame como el “user”, si no existe es que ese usuario aún no se ha empezado a procesar, por lo que creamos la carpeta e iniciamos la descarga estándar. Esta descarga recibe como parámetros el usuario a descargar y las fechas inicial y final para que se descarguen los posts y la información asociada entre esas fechas. Por cada post descargado grabamos la información en un fichero csv, que se explicará más adelante. En cambio, si ya existe una carpeta con el nombre de usuario que contiene esa fila tenemos que comprobar si ese usuario está completamente procesado (proc = 1) o no. En el primer caso, no se hará nada y se pasará a leer la siguiente fila del fichero. En el segundo caso, nos encontramos en la casuística de que el usuario empezó a procesarse, pero no terminó. Para poder optimizar el proceso y no volver a descargar todo el usuario al completo, recuperamos la información del último post descargado que tenemos almacenada en el fichero csv donde guardamos los datos una vez descargados el *post*. De este fichero recuperamos una lista con los identificadores de las imágenes descargadas y la fecha del último *post* descargado. Y con esta información llamaremos a la función *descarga modificada*, que en este caso además recibirá como parámetro la lista con los identificadores de las imágenes descargadas. Ahora la fecha final que le pasamos a la función será la recuperada del fichero anterior. Con estas ligeras modificaciones conseguimos que las peticiones a la API sean en un rango de fechas más reducido y, por otro lado, a la hora de la descarga solo se descargarán aquellos *posts* que no estén en la lista de identificadores.

Asumimos el coste computacional de estas modificaciones porque orientado de esta manera cubrimos la posibilidad de que un usuario haya publicado más de un post el mismo día, así que nos aseguramos de que ni perdemos información (pasándole como parámetro la última fecha del post descargado -1) ni la duplicamos.

Este proceso ha sido largo, dada la limitación de la API de Instagram, ocupando un periodo de aproximadamente 4 meses. Una vez finalizada la descarga completa, con 24.787 posts, pasamos a hacer la limpieza correspondiente de los datos para su posterior uso.

5.3 Fichero de datos csv

En el presente apartado se explicarán todos los datos obtenidos de la descarga. Cabe mencionar que se ha descargado una cantidad de datos superior a la que se va a utilizar en el proyecto dado que se pretende ampliar este trabajo en el futuro TFG de ADE.

El dato más importante de la descarga es el contenido multimedia del post, que puede ser solo una imagen, un grupo formado por un conjunto de entre 2 a 10 imágenes (sidecar) o un vídeo. Dicho esto, en presente proyecto solo se van a utilizar los *posts* de una imagen única ya que son de los únicos *posts* que podemos hacer la extracción de características conforme se ha planteado el proyecto.

En la figura 9, se puede observar la estructura estándar de un post en Instagram:

Figura 9 Ejemplo post de Instagram

Fuente: Elaboración propia.

Asociados a cada post y en consecuencia a cada imagen tenemos los siguientes datos:

- post: El nombre del archivo multimedia que descargamos, en formato UTC.
- Shortcode: id del post.
- User: Alias del usuario que ha compartido la publicación.
- Fecha: Día, mes y año en la que se publicó el post.
- Hora: Hora exacta con minutos y segundos de la publicación.
- Tipo: Específica si el tipo de post que se ha descargado es imagen, vídeo o sidecar.
- Mediacount: Número de imágenes en un sidecar.
- Likes: Número de “me gustas” que ha recibido el post.
- Comentarios: Número de comentarios que ha recibido el post.
- Texto: Se trata del texto bajo la foto, escrito por el propio usuario.
- Hashtags: una lista de los hashtags que aparecen en el texto de la imagen.
- Sponsored: Es 0 si el post no es patrocinado y 1 si se trata de un post patrocinado.
- Sponsor: En caso de ser un post patrocinado, quien es el patrocinador o colaborador.
- Menciones: Una lista de los usuarios mencionados en el texto de la imagen.
- Etiquetas: Una lista de los usuarios etiquetados en la imagen.
- videoView: En caso de ser un vídeo, las visitas que ha tenido.
- videoTime: En caso de ser un vídeo, la duración que tiene.

5.4 Preprocesado

De todas las variables recuperadas, para nuestro proyecto solo nos quedaremos con las numéricas y categóricas, descartando las de tipo texto que no vamos a procesar en este TFG. Además, solo nos quedamos las que son relevantes para nuestro proyecto. Por ejemplo, las variables de videoView o videoCount solo son aplicables cuando el post es un vídeo. Como no vamos a trabajar con los vídeos recuperados, estas variables, aunque son numéricas, no formaran parte de nuestro dataset para el entrenamiento de modelos. En la tabla 2 se especifican las variables conseguidas con la descarga de los posts y su tipo:

VARIABLE	TIPO
Post	texto
Shortcode	texto
User	texto
Fecha	datetime
Hora	datetime
Tipo	categórica
Mediacount	numérica
Likes	numérica
Comentarios	numérica
Texto	texto
Hashtags	texto
Sponsored	categórica
Sponsor	texto
Menciones	texto
Etiquetas	texto
VideoView	numérica
VideoTime	numérica

Tabla 2 Variables asociadas a los posts

Fuente: Elaboración propia.

Por lo comentado anteriormente, de las variables de la tabla 2 solo trabajaremos con: fecha, hora, *likes*, comentarios y *sponsored*. A estas hay que añadir los datos recuperados del perfil previos a la descarga, también relevantes para nuestro trabajo. De igual modo, de la tabla 3 seleccionaremos las variables numéricas y categóricas, descartando las de tipo texto:

VARIABLE	TIPO
Username	texto
Full_name	texto
nPost	numérica
Bio	texto
Followers	numérica
Followees	numérica
IgtvCount	numérica
Business	categórica
Category	categórica
Verified	categórica

Tabla 3 Variables asociadas al usuario

Fuente: Elaboración propia.

Uno de los primeros pasos para construir el dataset ha sido concatenar ambas tablas de datos a través del *user*, dado que es la variable común a ambas tablas. Una vez unidas ambas tablas, empezamos la limpieza de los datos y la elección de variables dependientes e independientes.

Teniendo en cuenta el propósito de nuestro proyecto, que es la predicción del *engagement*, hay dos variables con las que podemos medirlo: los *likes* y los comentarios. Para simplificar el problema decidimos centrarnos en los *likes* como variable dependiente, ya que como se ha comentado antes es la función más realizada por los usuarios. De esa forma, ya no podemos tener en cuenta para el estudio los comentarios como variable independiente dado que al igual que los *likes* son variables de las cuales no conocemos su valor antes de que el post haya sido publicado.

Una vez elegida la variable dependiente, nuestro *dataset* quedaría con las variables que aparecen en la tabla 4:

VARIABLE	TIPO	
Likes	numérica	Dependiente
Sponsored	categórica	Independiente
nPost	numérica	Independiente
Followers	numérica	Independiente
Followees	numérica	Independiente
igtvCount	numérica	Independiente
Business	categórica	Independiente
Category	categórica	Independiente
Verified	categórica	Independiente
Fecha	datetime	Independiente
Hora	datetime	Independiente

Tabla 4 Variables escogidas para el proyecto.

Fuente: Elaboración propia.

En este punto cabe indicar que hubiera sido más coherente eliminar del dataset todos aquellos *posts* que no son de tipo imagen antes de empezar con la generación de características. Sin embargo, estas características son necesarias para el TFG posterior por lo que se ha seguido trabajando hasta el último momento con el *dataset* completo.

Finalmente, antes de pasar a la generación de características, han sido localizados los valores perdidos ('NaN') en el dataset. En nuestro caso, solo se han dado en la variable 'category', donde se nos devolvía un NaN si la categoría no estaba especificada. Para solucionarlo, hemos sustituido los 'NaN' por la etiqueta 'Other'. como se ve en la figura:

```
[14] # Sustituimos los NaN por la etiqueta 'Other'
     df['category'] = df['category'].fillna('Other')
```

Figura 10 Extracto Script para sustituir NaN

Fuente: Elaboración propia.

Por último, después de hacer las pruebas correspondientes detectamos la presencia de *outliners*. En la figura x se observa como conforme el número de *likes* es mayor los datos se dispersan más.

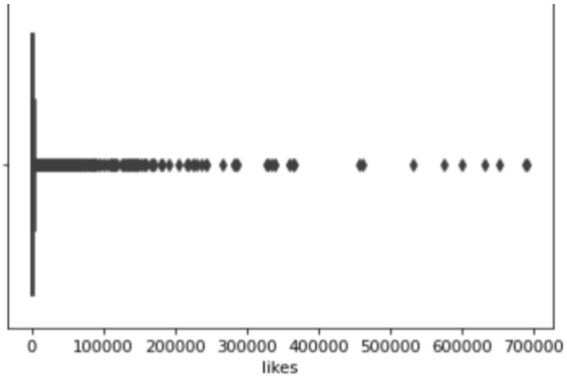

Figura 11 Boxplot de likes

Fuente: Elaboración propia.

No ha sido fácil determinar hasta que número de *likes* incluir en nuestro estudio y para tomar una decisión se entrenaron distintos modelos con distintos rangos para ver de qué manera se comportaba el modelo. Sin embargo, dichas pruebas no se detallan en este trabajo dado que el propósito de este es otro.

5.5 Generación de características

5.5.1 Características relacionadas con el tiempo

Dado que la fecha y la hora recogidos del post tal y como vienen son difícilmente manejables, las hemos transformado de manera que aporten valor a nuestro modelo.

generando las siguientes nuevas variables:

- *day*: Día de la semana en el que se publicó el post.
- *month*: Mes del año en el que se publicó el post.
- *hour_of_day*: Hora del día en el que se publicó el post.

Para ello hemos utilizado el siguiente código que aparece en la figura 12:

```
import csv
fechas= []
hora = []
format = []
horaf = []
meses = []
mesf = []


with open('/content/drive/My Drive/Curso_DL_CFP/proyecto/datos_nc_modelos/0_dataset_resumen_c.csv', 'r') as File:
                        reader = csv.reader(File, delimiter=';')
                        # Omitir el encabezado
                        next(reader, None)
                        for row in reader:

                            fechas.append(datetime.strptime(row[4], '%Y-%m-%d'))
                            hora.append(datetime.strptime(row[5], '%H:%M:%S'))

                        print(fechas)
                        print(hora)
                        for fec in fechas:
                          format.append(fec.strftime('%A'))
                          meses.append(fec.strftime('%m'))
                        print(format)
                        print(meses)
                        for hr in hora:
                          horaf.append(hr.strftime('%H'))
                        print(horaf)

                        df['day'] = format
                        df['month'] = meses
                        df['hour_of_day'] = horaf
```

Figura 12 Extracto código creación características day, month, hour_of_day.

Fuente: Elaboración propia.

5.5.2 Características extraídas de las imágenes

Uno de los propósitos de este proyecto es conseguir información a través de las imágenes. Aunque se pueden extraer características inimaginables, como por ejemplo, el color que predomina en la imagen, qué sentimiento transmite o incluso si aparecen marcas, en esta ocasión hemos decidido escoger variables que aporten valor al dominio del estudio, el sector del running.

Por esta razón, las variables que hemos decidido extraer de la imagen son, por un lado, el número de caras que aparecen, ya que, aunque se trata de un deporte individual suele practicarse en grupo. Y esta información podría ser relevante para el estudio. Por otro lado, hemos querido clasificar las imágenes para saber si se mostraba al aire libre o en un espacio cerrado.

Antes de plantear la extracción de las características mencionadas hemos procedido a eliminar del dataset todos aquellos posts que no fueran una sola imagen, como se ha

comentado anteriormente Para la eliminación de estos *posts* hemos seguido los siguientes pasos:

1) Eliminación de las filas con tipo de post 'GraphVideo' o 'GraphSidecar':

```
[ ]  data = df[df['tipo']== 'GraphVideo'].index
     df.drop(data, inplace = True)

     dat = df[df['tipo']== 'GraphSidecar'].index
     df.drop(dat, inplace = True)
```

Figura 13 Extracto código eliminación de filas

Fuente: Elaboración propia.

2) Eliminación de los archivos multimedia del directorio donde se han descargado. Al hacer la descarga todos los archivos se guardaban con su extensión correspondiente:

2021-11-03_16-43-52_UTC.jpg	3 nov 2021 17:43	317 KB	Imagen JPEG
2021-11-03_16-43-52_UTC.txt	3 nov 2021 17:43	1 KB	Texto
2021-10-31_11-49-19_UTC.jpg	31 oct 2021 12:49	341 KB	Imagen JPEG
2021-10-31_11-49-19_UTC.txt	31 oct 2021 12:49	710 bytes	Texto
2021-10-30_09-03-32_UTC.jpg	30 oct 2021 11:03	130 KB	Imagen JPEG
2021-10-30_09-03-32_UTC.mp4	30 oct 2021 11:03	10,3 MB	Vídeo MPEG-4
2021-10-30_09-03-32_UTC.txt	30 oct 2021 11:03	706 bytes	Texto
2021-10-28_17-19-21_UTC_1.jpg	28 oct 2021 19:19	239 KB	Imagen JPEG
2021-10-28_17-19-21_UTC_2.jpg	28 oct 2021 19:19	253 KB	Imagen JPEG
2021-10-28_17-19-21_UTC_3.jpg	28 oct 2021 19:19	449 KB	Imagen JPEG

Figura 14 Tipos de ficheros descargados

Fuente: Elaboración propia.

Como se observa en la figura 14, tenemos los archivos que finalizan con 'UTC.jpg' que corresponden a los posts de una sola imagen y serán los que utilicemos en nuestro proyecto. Los videos tienen su extensión .mp4. Y los *sidecars* los localizamos porque las imágenes que pertenecen al post están enumeradas por lo que a diferencia de los *posts* con solo una imagen (que terminan todas …UTC.jpg) es que siempre vendrá algún número delante de la extensión .jpg. Por último, tenemos los archivos con extensión .txt que corresponden a los textos que acompañan al contenido multimedia del *post* y podrán ser procesados en proyectos futuros.

Al ejecutar el script de la figura 15, conseguimos quedarnos solo con las imágenes necesarias para la extracción de características, que son, un total de 14.782 imágenes.

```
ImageDataPath = "/Users/albaprimoaparici/Desktop/dataset"
fr = os.listdir(ImageDataPath)
x = fr.sort

for dir1 in os.listdir(ImageDataPath)[0:]:
    print(dir1)

    files = os.listdir(ImageDataPath + '/' + dir1)
    for file in files:
        if file.endswith("UTC.jpg"):
            continue
        else:
            os.remove( ImageDataPath + '/' + dir1 + '/' + file)
```

Figura 15 Extracto script para dejar en el directorio archivos necesarios

Fuente: Elaboración propia.

5.5.2.1 Número de caras

Para la generación de esta característica se ha decidido utilizar la librería OpenCV, ya que cuenta con clasificadores para la detección de rostros y es fácil de implementar. Siguiendo la documentación parametrizamos los valores necesarios. En concreto, era necesario establecer un valor para los parámetros *scaleFactor* y *minNeighbors*.

En primer lugar, *ScaleFactor* nos indica cuanto se va a reducir la imagen en porcentaje. Por ejemplo, en nuestro caso en cada iteración se reducirá un 30 %. Hay que tener cuidado ya que si se trata de un número muy pequeño como 1,01(cada iteración se reducirá un 1%), el tiempo de procesamiento será mayor y además el modelo será propenso a detectar rostros donde no los hay. En el caso de ser muy grande, por ejemplo 1,5(cada iteración se reducirá un 50%), nos encontraremos con el caso contrario, el modelo será menos flexible y por lo tanto no será capaz de detectar todos los rostros que puedan aparecer.

En la figura 16 se ha manipulado el parámetro *scaleFactor* para ver cómo se comporta sin modificar el parámetro *minNeighbors.*

Figura 16 Resultados de ejecución del detector de cararas utilizando distintos scaleFactor

Fuente: Elaboración propia.

Se observa claramente que cuanto menor es este factor (arriba a la izquierda) más rostros falsos detecta. En la fotografía en concreto de la Figura solo se muestran dos rostros y cuando *scaleFactor* coge tanto el valor 1,3 como 1,5 detecta los rostros correctamente.

Si utilizamos una imagen más compleja, en la que aparecen muchos más rostros, con un valor mayor para este parámetro perdemos información. Esto lo comprobamos en la figura 17. Cuando el parámetro toma el valor 1.5 deja de detectar una de las caras que aparecen en la imagen.

Figura 17 Resultado ejecución detector de caras con distintos scaleFactor

Fuente: Elaboración propia.

Con las comprobaciones mostradas anteriormente concluimos que el valor que más se adapta para la detección de rostros es 1,3.

En segundo lugar, el parámetro *minNeighbors* nos indica cuantas detecciones como mínimo debe haber del mismo rostro en el cómputo total de todas las operaciones para ser candidato a ser un rostro detectado. Es decir, en cada iteración tenemos un rectángulo que va recorriendo la imagen, y en cada iteración hace *n* detecciones. Para que se considere que el rostro ha sido detectado al final de la ejecución, si el parámetro *minNeighbors* vale 25, el rostro en cuestión tendrá que haberse detectado en 25 iteraciones distintas. Si, por lo contrario, este parámetro vale 1, se considerará rostro detectado con que solo haya aparecido en una iteración. Esto nos indica que conforme mayor sea el valor, más exigente será la detección de rostros.

En la figura 18 se ha manipulado el parámetro *minNeighbors* para ver cómo se comporta sin modificar el parámetro *scaleFactor.* En este caso, cuando el parámetro coge el valor 3 ya nos devuelve un resultado correcto.

Figura 18 Resultado ejecución detector de caras con distintos minNeighbors

Fuente: Elaboración propia.

Para asegurarnos de que se trata de un valor óptimo hemos vuelto a hacer la prueba con una imagen más compleja, al igual que con el parámetro anterior. En la figura 19, bajo a la izquierda vemos como sigue detectando falsos rostros. En cambio, a la derecha vemos como detecta solo los rostros que aparecen en la imagen.

Figura 19 Resultado ejecución detector de caras con distintos minNeighbor

Fuente: Elaboración propia.

Con estas pruebas queda justificada la selección de valores de los parámetros para el script que utilizaremos a continuación.

Una vez elegidos los valores de los parámetros que va a tomar la función, para recorrer todas las carpetas donde están almacenadas las casi 15.000 imágenes con las que trabajamos hemos seguido la misma estructura que para la eliminación de archivos que

no necesitábamos. Además, hemos preparado la extracción de dos variables distintas con este script. Por un lado, una primera variable numérica llevará el conteo de los rostros que se detecten en las imágenes. Y, por otro lado, la segunda variable será booleana, siendo falsa cuando no se encuentre ningún rostro y verdadera cuando se encuentre al menos un rostro.

Como puede observarse en la figura 20, se ha añadido un controlador para la excepción *ValueError*, ya que después de varias pruebas descubrimos que cuando salta esta excepción es que no está encontrando ningún rostro en la imagen. Esto indica que cuando salte la excepción nuestra variable booleana será falsa y el número de rostros será 0.

```
import cv2
import numpy as np
from google.colab import drive
from google.colab.patches import cv2_imshow
import os
# Montamos la unidad de Drive
drive.mount('/content/drive') #(X)
ImageDataPath = "/content/drive/My Drive/Curso_DL_CFP/proyecto/dataset"
faces = []
post = []
is_people = []
people = ''
for dir1 in os.listdir(ImageDataPath):

        files = os.listdir(ImageDataPath + '/' + dir1)
        fotos = []
        for file in files:
            if file.endswith(".jpg"):
                original_image = cv2.imread(ImageDataPath + '/' + dir1 + '/'+ file)
                if original_image is not None:
                    # Convertir imagen a escala de grises
                    image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)

                    # Creando clasificador
                    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

                    # Detección de rostros usando el clasificador
                    detected_faces = face_cascade.detectMultiScale(image=image, scaleFactor=1.3, minNeighbors=4)

                      try:
                        row, columns = np.shape(detected_faces)
                        print('N º caras detectadas de frente :  ', row)
                        people = 'true'
                      except (ValueError):
                        row = 0
                        people = 'false'

                    faces.append(row)
                    post.append(file)
                    is_people.append(people)

                else:
                    print(f'En error occurred while trying to load {original_image}')
```

Figura 20 Extracto código detección de caras

Fuente: Elaboración propia.

Por último, la información extraída es almacenada en un fichero de datos (figura 21) que después se concatenará con las demás características ya conseguidas.

```
import csv
feature_faces = []
with open("/content/drive/My Drive/Curso_DL_CFP/proyecto/datos_nc_modelos/num_faces.csv", 'w', encoding='UTF-8') as archivo:
        writer = csv.DictWriter(archivo, fieldnames=['post', 'faces', 'is_people'], delimiter=',')
        writer.writeheader()
        for tupla in zip(post, faces, is_people):
          fila = {}
          fila['post'] =  tupla[0]
          fila['faces'] = tupla[1]
          fila['is_people'] = tupla[2]
          feature_faces.append(fila)

        writer.writerows(feature_faces)
```

Figura 21 Extracto código para guardar las características de la detección de caras

Fuente: Elaboración propia.

5.5.2.2 Espacio cerrado o abierto

En esta ocasión decidimos plantear la creación de una red neuronal que nos clasificara nuestras imágenes en *outdoor* o *indoor*. Para ello seguimos los siguientes pasos:

1. Búsqueda de un *dataset* ya existente para el entrenamiento de la red neuronal.
2. Manipulación y preprocesado de las imágenes para que el modelo pudiera trabajar de manera eficiente con ellas.
3. Elección de la arquitectura para nuestro modelo.
4. Entrenamiento y evaluación del *accuracy*.

En el primer paso se investigó si ya existía algún dataset útil para nuestro problema o por lo contrario teníamos que construirlo con un proceso de *web Scraping*. La búsqueda resultó exitosa y conseguimos un dataset de 800 imágenes clasificadas en outdoor e indoor [32].

A continuación, preparamos la entrada a nuestra red neuronal, es decir las imágenes. Para ello, lo primero que hicimos fue cargar el dataset en *Google colab* y crear las etiquetas para la clasificación. Después, utilizamos la técnica denominada *Data augmentation* para generar datos artificialmente a partir de perturbaciones en los datos ya existentes. De esta manera, conseguimos aumentar el dataset entre otros beneficios.

La arquitectura elegida para el modelo fue AlexNet, figura x, que consta de ocho capas: cinco capas convolucionales y tres capas totalmente conectadas. Además, se caracteriza por:

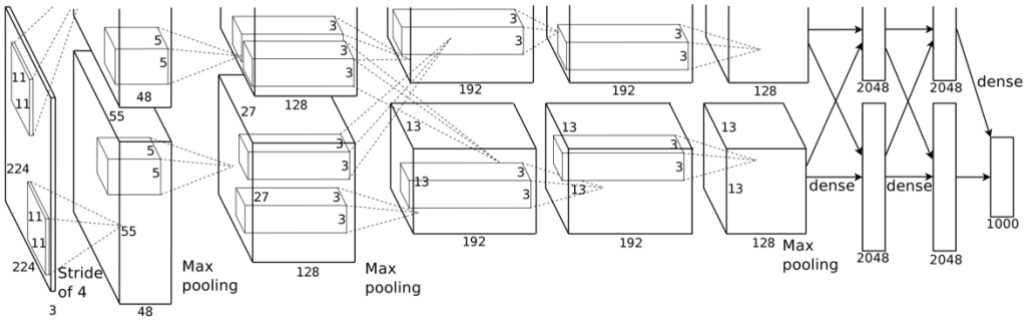

Figura 22 Arquitectura CNN AlexNet

Fuente: [33].

- AlexNet utiliza unidades lineales rectificadas (ReLU) en lugar de la función tanh, que era el estándar en ese momento. La ventaja de ReLU está en el tiempo de entrenamiento; una CNN que utiliza ReLU fue capaz de alcanzar un error del 25% en el conjunto de datos CIFAR-10 seis veces más rápido que una CNN que utiliza tanh [33].

- Tradicionalmente, las CNN "agrupan" las salidas de los grupos de neuronas vecinos sin que se produzca un solapamiento. Sin embargo, cuando los autores introdujeron el solapamiento, observaron una reducción del error de alrededor del 0,5% y descubrieron que los modelos con agrupación solapada suelen tener más dificultades para el sobreajuste.

Otro aspecto importante es la reducción de sobre entrenamiento. Para lidiar con este problema, a parte del *Data augmentation* mencionado arriba, la arquitectura AlexNet

introduce *Dropout o Desconexión.* Esta técnica consiste en "apagar" las neuronas con una probabilidad predeterminada (por ejemplo, el 50%). Esto significa que cada iteración utiliza una muestra diferente de los parámetros del modelo, lo que obliga a cada neurona a tener características más robustas que puedan ser utilizadas con otras neuronas aleatorias.

Una vez configurada la arquitectura de AlexNet a nuestro dataset procedimos a entrenarla. En el scrjpt «*_generación_caracteristicas.ipynb*», que puede encontrarse en el repositorio mencionado en la estructura de la memoria (apartado 1.5), se incluye toda la preparación, configuración y entrenamiento del modelo.

Consiguiendo una *accuracy* mayor al 80 % en la evaluación como puede observarse en la figura 23, estamos en disposición de utilizar este modelo para predecir la característica en cuestión.

```
[11]  # evaluate the model
      cvscores = []
      scores = AlexNet.evaluate(val_datagen, verbose=0)
      print("%s: %.2f%%" % (AlexNet.metrics_names[1], scores[1] * 100))
      cvscores.append(scores[1] * 100)
      print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))

      accuracy: 83.55%
      83.55% (+/- 0.00%)
```

Figura 23 Evaluación modelo CNN

Fuente: Elaboración propia.

El esquema seguido para la extracción de la información de las imágenes es el ya comentado en la característica anterior, pero en este caso integrando el modelo entrenado como se muestra en las figuras 24 y 25. La primera figura muestra la función donde se integra el modelo y en la segunda se muestra el uso de esta función para la extracción de conocimiento.

```
labelNames = ["indoor", "outdoor"]

def predict_image(image, model):

  # Expandimos las dimensiones (32, 32, 3) a (1, 32, 32, 3)
  image = np.expand_dims(image, axis=0) #(X)

  # Clasificación de la imagen empleando el modelo
  # Realizamos la predicción
  resultado = AlexNet.predict(image) #(X)
  # Nos quedamos con la clase que presente una probabilidad mayor y buscamos la etiqueta en el vector labelNames
  idx = np.argmax(resultado)
  label = labelNames[idx]
  res = label
  acc = np.max(resultado)

  return res, acc
```

Figura 24 Función predict_image

Fuente: Elaboración propia.

```
ImageDataPath = "/content/drive/My Drive/Curso_DL_CFP/proyecto/dataset"
label = []
post = []
acc = []
for dir1 in os.listdir(ImageDataPath):
    files = os.listdir(ImageDataPath + '/' + dir1)
    fotos = []
    for file in files:
        if file.endswith(".jpg"):
           original_image = cv2.imread(ImageDataPath + '/' + dir1 + '/'+ file, cv2.IMREAD_COLOR) # Leo imagen con OPENCV
           if original_image is not None:

              image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB) # Por defecto la carga en BGR, la convierto a RGB
              # Pre-procesamos tal y como he hecho para la fase de entrenamiento con las muestras de CIFAR10
              imgage = image.astype("float") / 255.0 #(X)
              # Re-escalamos la imagen al tamaño con el que fue entrenada la red (comando cv2.resize)
              img_res = cv2.resize(imgage, (32, 32)) #(X)
              # Predecimos la imagen pasando como parámetros a la función predict_image: la imagen y el modelo
              res = predict_image(img_res, AlexNet)#(X)

              label.append(res[0])
              post.append(file)
              acc.append(res[1])



    else:
      print(f'En error occurred while trying to load image')
```

```
import csv
feature = []
with open("/content/drive/My Drive/Curso_DL_CFP/proyecto/datos_nc_modelos/in_out.csv", 'w', encoding='UTF-8') as archivo:
        writer = csv.DictWriter(archivo, fieldnames=['post', 'label', 'acc'], delimiter=',')
        writer.writeheader()
        for tupla in zip(post, label, acc):
          fila = {}
          fila['post'] =  tupla[0]
          fila['label'] = tupla[1]
          fila['acc'] = tupla[2]
          feature.append(fila)

        writer.writerows(feature)
```

Figura 25 Extracto código detección de espacio

Fuente: Elaboración propia.

En esta ocasión, se ha almacenado también la probabilidad de acierto con la que ha elegido la etiqueta de cada imagen. La razón principal es que esta variable podría servirnos para ajustar nuestro dataset y filtrar posts en caso de que fuera necesario.

5.6 Preparación dataset para entrenamiento de modelos

Una vez extraída la información de las imágenes con la que vamos a trabajar, vamos a hacer las modificaciones pertinentes para poder entrenar nuestros modelos. La principal, la codificación de las variables categóricas para que nuestros modelos sean capaces de entender esta información.

La tabla 5 muestra las variables con la que vamos a trabajar finalmente, así como el tipo de estas. La última columna muestra los valores que puede coger cada característica.

VARIABLE	TIPO	Datos
Likes	numérica	[0, …, 690534]
Sponsored	categórica	[0, 1]
nPost	numérica	[0, …, 8763]
Followers	numérica	[0, …, 11112558]
Followees	numérica	[0, …, 7489]
IgtvCount	numérica	[0, …, 127]
Business	categórica	[0, 1]
Category	categórica	['Other', 'Creators & Celebrities', 'Personal Goods & General Merchandise Stores', 'General Interest', 'Publishers', 'Non-Profits & Religious Organizations']
Verified	categórica	[0, 1]
Hour_of_day	categórica	[0, …, 23]
Day	categórica	[Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday]
Month	categórica	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Num_faces	numérica	[0, …, 33]
Is_people	categórica	[true, false]
Space	categórica	[*outdoor*, *indoor*]

Tabla 5 Variables finales para entrenar los modelos.

Fuente: Elaboración propia.

Como acabamos de comentar, es necesario transformar las variables categóricas para que el modelo sea capaz de interpretarlas. Para ello, se va a utilizar *One-Hot-Encoding*, la técnica más extendida para la codificación de variables.

En este método, asignamos cada categoría a un vector que contiene 1 y 0, denotando la presencia o ausencia de la característica. El número de vectores depende del número de categorías de las características. Algunas de las ventajas de usar *One-Hot-Encoding* son:

- No hace ninguna suposición sobre la distribución de la categoría.
- Preserva la información original de la variable categórica.
- Maneja nuevas categorías en el conjunto de pruebas.
- Adecuado para la regresión lineal.

En la figura 26 se muestra como se ha hecho uso de la librería *scikit learn* para la codificación de características.

```
from sklearn.preprocessing import LabelEncoder
#one hot encoding
from numpy import asarray
from sklearn.preprocessing import OneHotEncoder
# definimos datos
data_day = dataset_train[['day']].values
data_month = dataset_train[['month']].values
data_hour_of_day = dataset_train[['hour_of_day']].values
data_category = dataset_train[['category']].values
data_is_people = dataset_train[['is_people']].values
data_label = dataset_train[['label']].values
# definimos one hot encoding
encoder = OneHotEncoder(sparse=False)
# transformamos datos
onehot_day = encoder.fit_transform(data_day)
onehot_month = encoder.fit_transform(data_month)
onehot_hour_of_day = encoder.fit_transform(data_hour_of_day)
onehot_category = encoder.fit_transform(data_category)
onehot_is_people = encoder.fit_transform(data_is_people)
onehot_label = encoder.fit_transform(data_label)
```

```
dataset_train['day'] = onehot_day
dataset_train['month'] = onehot_month
dataset_train['hour_of_day'] = onehot_hour_of_day
dataset_train['category'] = onehot_category
dataset_train['is_people'] = onehot_is_people
dataset_train['label'] = onehot_label
```

Figura 26 Codificación OneHotEncoding

Fuente: Elaboración propia.

En este punto, las variables ya están preparadas para poder trabajar con ellas en los modelos propuestos en la siguiente sección.

6.Modelado

En este apartado se describirán los diferentes modelos utilizados, así como el modelo tomado como referencia.

6.1 *Baseline*

Lo primero ha sido establecer un *baseline* para tener un punto de partida que nos permita evaluar nuestros modelos comparándolos con un caso simple.

Para ello, se ha utilizado el estimador Knn, ya que se trata de un método sencillo que suele adaptarse bien a cualquier problema con sus parámetros por defecto. Los valores que cogen estos parámetros por defecto son:

- *n_neighbors* = 5, que especifica cuántos vecinos comprobará para determinar la relación con un punto de consulta específico. En este caso, si k=5, la instancia se asignará a sus 5 vecinos más cercanos.
- *algorithm:* 'auto', que especifica el algoritmo utilizado para calcular los vecinos más cercanos. En este caso a ser auto el valor por defecto escogerá el algoritmo más adecuado en función de los valores pasados al método de ajuste.
- *Leaf_size*= 30, que especifica el tamaño de hoja que se pasa al algoritmo de cálculo. Esto puede afectar a la velocidad de construcción y consulta, así como a la memoria necesaria para almacenar el árbol. El valor óptimo depende de la naturaleza del problema.
- P=2, que especifica el parámetro de potencia para la métrica de Minkowski. Cuando p = 1, esto equivale a utilizar manhattan_distance (l1), y euclidean_distance (l2) para p = 2. Para p arbitrario, se utiliza minkowski_distance (l_p).
- *Metric*= 'minkowski', que especifica La métrica de distancia a utilizar para el árbol. Con p=2 es equivalente a la métrica euclidiana estándar.

Con los valores detallados arriba y las características obtenidas antes de la extracción de características de las imágenes, en una primera ejecución obtuvimos los valeres mostrados en la figura 27. Como se ve tenemos 3 formas de evaluar el modelo:

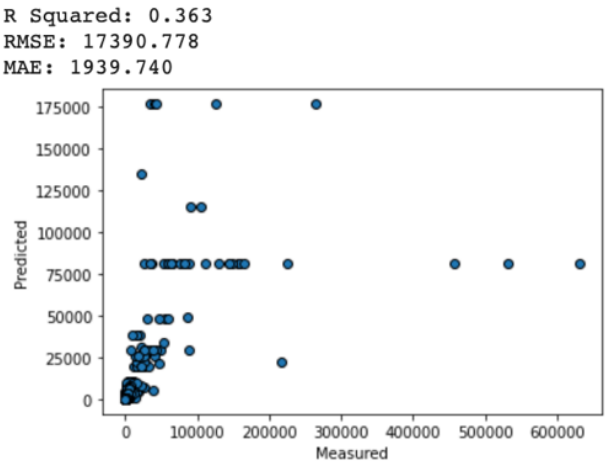

Figura 27 Diagrama de dispersión modelo Knn

Fuente: Elaboración propia.

1. El r2, que es la forma más extendida en evaluación de modelos predictivos. El R-cuadrado evalúa la dispersión de los puntos de datos en torno a la línea de regresión ajustada. También se denomina coeficiente de determinación, o coeficiente de determinación múltiple para la regresión múltiple. Para el mismo conjunto de datos, los valores de R-cuadrado más altos representan menores diferencias entre los datos observados y los valores ajustados. Se trata del porcentaje de la variación de la variable dependiente que explica un modelo lineal. El R-cuadrado está siempre entre el 0 y el 100%:

    - El 0% representa un modelo que no explica nada de la variación de la variable de respuesta en torno a su media. La media de la variable dependiente predice la variable dependiente tan bien como el modelo de regresión

    - El 100% representa un modelo que explica toda la variación de la variable de respuesta en torno a su media

Normalmente, cuanto mayor sea el R2, mejor se ajusta el modelo de regresión a las observaciones.

2. RMSE: es la desviación estándar de los errores que se producen cuando se realiza una predicción sobre un conjunto de datos. En el RMSE, los errores se elevan al cuadrado antes de ser promediados. Esto implica básicamente que el RMSE asigna un mayor peso a los errores más grandes. Esto indica que el RMSE es mucho más útil cuando hay grandes errores que afectan drásticamente al rendimiento del modelo. Normalmente, cuanto más bajo es el RMSE mejor ajustado está el modelo.

3. MAE: mide la magnitud media de los errores en un conjunto de predicciones, sin considerar si se trata de un valor negativo o no. El MAE toma la media de este error de cada muestra de un conjunto de datos y da el resultado. Se suele utilizar cuando el rendimiento se mide sobre datos de variables continuas. Da un valor

lineal, que promedia las diferencias individuales ponderadas por igual. Cuanto menor sea el valor, mejor es el rendimiento del modelo.

Por ello, se escoge el r2 para la evaluación de los modelos, ya que tanto el MAE como el RMSE presentan limitaciones. El MAE suelte utilizarse cuando todas las variables son continuas, en nuestro caso hay variables categóricas y el RMSE no describe únicamente el error medio y tiene otras implicaciones que son más difíciles de descifrar y comprender.

El r2 obtenido en esta primera ejecución fue de 36,3 %, lo que nos dejaba un margen de mejora bastante amplio. Sin embargo, en una segunda ejecución después de eliminar los outliners el algoritmo knn mejoró el r2 más de un 35%. Como se muestra en la figura 28, el r2 consigue un r2 de 74,3%. Además, también se observa la abrupta mejora del RMSE y el MAE.

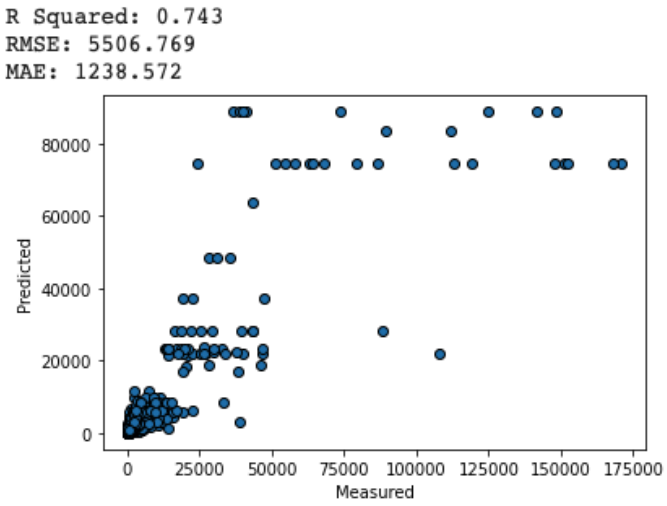

Figura 28 Diagrama de dispersión modelo Knn después de eliminar outliners

Fuente: Elaboración propia.

6.2 Análisis de correlación

De cara a la propuesta de modelos, se ha realizado un análisis de correlación para identificar qué variables pueden funcionar mejor para la predicción del *engagement.*

El análisis de correlación es un método estadístico que se utiliza para descubrir si existe una relación entre dos variables o conjuntos de datos, y la intensidad de dicha relación. Este tipo concreto de análisis es útil cuando un investigador quiere establecer si existen posibles conexiones entre las variables [8]. A menudo se malinterpreta que el análisis de correlación determina la causa y el efecto; sin embargo, no es así porque otras variables que no están presentes en la investigación pueden haber influido en los resultados.

Si se encuentra correlación entre dos variables, significa que cuando hay un cambio sistemático en una variable, también hay un cambio sistemático en la otra; las variables se alteran juntas en un determinado periodo de tiempo. Si se encuentra correlación, dependiendo de los valores numéricos medidos, ésta puede ser positiva o negativa.

- Existe correlación positiva si una variable aumenta simultáneamente con la otra, es decir, los valores numéricos altos de una variable se relacionan con los valores numéricos altos de la otra.
- Existe una correlación negativa si una variable disminuye cuando la otra aumenta, es decir, los valores numéricos altos de una variable se relacionan con los valores numéricos bajos de la otra.

El coeficiente de Pearson es la medida de la correlación y oscila (según la correlación) entre +1 y -1. +1 indica la mayor correlación positiva posible y -1 la mayor correlación negativa posible. Por lo tanto, cuanto más se acerque el coeficiente a cualquiera de estos números, más fuerte será la correlación de los datos que representa. En esta escala, el 0 indica que no hay correlación, por lo que los valores más cercanos a cero ponen de manifiesto una correlación más débil/pobre que los más cercanos a +1/-1.

En la figura 29 se observa como las 5 variables más correlacionadas con el *engagement*, y, por tanto, las que más se tendrán en cuenta a la hora de proponer modelos son: Followers, verified, nIgtv, business y hour_of_day. Sin embargo, cabe precisar que realmente la única variable que podría considerarse correlacionada con nuestra variable objetivo es Followers.

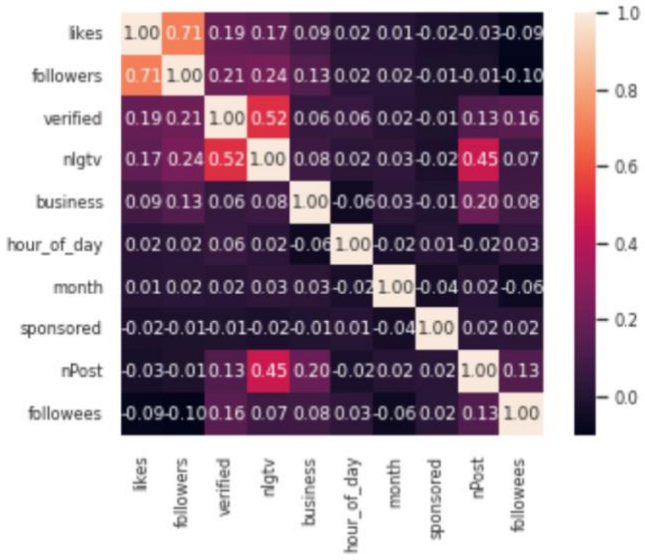

Figura 29 Estudio de Correlación

Fuente: Elaboración propia.

6.3 Propuesta de modelos

Una vez establecido el modelo de referencia, se han propuesto varios experimentos para ver cómo se comportan distintos modelos con relación a las variables con las que trabajamos.

A nivel de variables se ha decidido hacer 4 grupos, en cada grupo el conjunto de características variará. De esta manera podremos estudiar qué valor aportan las características extraídas de las imágenes al entrenamiento de un modelo predictivo.

Los 4 grupos de variables son los siguientes:

- Las variables de nuestro dataset sin incluir las extraídas de las imágenes. A este grupo lo denominamos csv.
- El conjunto de todas las variables, incluyendo las extraídas de las imágenes: csv + img
- Las 5 variables extraídas del análisis de correlación: 5csv
- Las 5 variables extraídas del análisis de correlación dependiente incluyendo las características extraídas de la propia imagen: 5csv + img

Por otro lado, estos grupos de variables se entrenarán con distintos modelos. Se proponen dos algoritmos de regresión más potentes que el KNN, así como distintas arquitecturas de ANN.

Los dos algoritmos elegidos vienen de la técnica Boosting, de la que se ha hablado en el apartado 2.2. En concreto, vamos a trabajar con GradientBoosting y XGBoost.

- GradientBoosting: este algoritmo es uno de los más potentes en el campo del aprendizaje automático. Puede utilizarse para predecir no sólo la variable objetivo continua (como regresor), sino también la variable objetivo categórica (como clasificador). Cuando se utiliza como regresor, la función de coste es el error cuadrático medio (MSE). Está técnica se basa en intuir cual va a ser el siguiente modelo mejor. Cuando se combina con los modelos anteriores, minimiza el error de predicción global. La idea clave es establecer los resultados objetivo para este próximo modelo con el fin de minimizar el error [8].

- XGBoost: Se trata de una implementación específica del método Gradient Boosting que utiliza aproximaciones más precisas para encontrar el mejor modelo de árbol. Además, presenta otras ventajas como la rapidez de entrenamiento y la posibilidad de ser distribuido a través de clústeres, es decir, puede ser paralelizado [8].

La selección de parámetros para las arquitecturas ANN se ha elegido después de unas pruebas ad-hoc, que no se adjuntan en este proyecto dada su larga extensión.

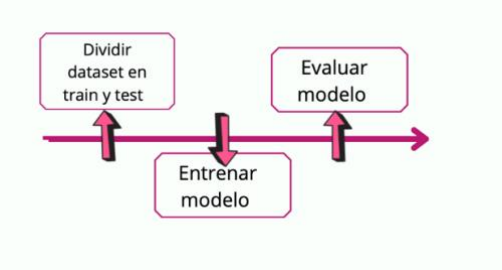

Figura 30 Metodología experimentos

Fuente: Elaboración propia.

Como puede apreciarse en la figura 30, exponemos la metodología estándar de los experimentos realizados que, empieza por la división del dataset y termina con la obtención de *accuracy* que evalúa el modelo.

En la primera fase los datos se reparten en dos subconjuntos, el de entrenamiento y el de validación o prueba. Se ha decidido utilizar el 80 % de los datos para el entrenamiento y el 20 % restante para la evaluación de rendimiento de los modelos. Además, el parámetro *random_state,* nos permite que la evaluación del modelo no varíe en distintas ejecuciones con las mismas condiciones.

Con la función de la librería *Scikit-learn* que se muestra en la figura 31, los datos a utilizar se dividen en cuatro:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figura 31 División del dataset

Fuente: Elaboración propia.

- Xtrain, Ytrain: En estas variables se almacenará cada elemento a utilizar durante el entrenamiento. Por un lado, en la variable Y se recogerá cada valor de la variable dependiente. Por otro lado, en la estructura X se almacenarán el resto de las variables para cada caso.

• Xtest, Ytest: Estos dos subconjuntos son similares a los anteriores, pero en este caso son los elementos utilizados para la evaluación del modelo.

Una vez hecha esta partición los datos utilizados en el entrenamiento son 11.403 posts y para test 2851 posts.

En la siguiente fase se entrenarán los distintos modelos. La siguiente tabla 6 recoge las distintas técnicas, así como las variables que serán utilizadas en cada uno de los modelos.

	Técnicas	Variables
Modelo 1	Gradient Boosting	csv
Modelo 2	Gradient Boosting	Csv + img
Modelo 3	Gradient Boosting	5 csv
Modelo 4	Gradient Boosting	5 csv + img
Modelo 5	XGBoost	csv
Modelo 6	XGBoost	Csv + img
Modelo 7	XGBoost	5 csv
Modelo 8	XGBoost	5 csv + img
Modelo 9	ANN	csv
Modelo 10	ANN	Csv + img
Modelo 11	ANN	5 csv
Modelo 12	ANN	5 csv + img

Tabla 6 Modelos distintos que se van a entrenar.

Fuente: Elaboración propia.

7.Evaluación de modelos

Antes de la evaluación de los modelos definitivos, se adjunta a modo complementario la tabla 7 con los resultados obtenidos en las primeras pruebas. A raíz de estos resultados detectamos la pertinencia de gestionar los *outliers* e incluimos las instrucciones de preprocesado necesarias para eliminarlos.

	CSV	CSV + IMG	5 CSV	5 CSV + IMG
Gradient Boosting	0.466909	0.461334	0.475178	0.478520
XGBoost	0.466808	0.465137	0.475202	0.477918
ANN	0.442	0.449	0.434	0.441

Tabla 7 Precisión modelos antes de eliminar outliners

Fuente: Elaboración propia.

Sin entrar en mucho detalle con estos resultados preliminares ya observamos que, aunque las diferencias son mínimas, las características que mejor funcionan son las de la última columna (5 csv + img) en el caso de los dos primeros modelos. Sin embargo, en el caso de ANN, se ha obtenido un resultado mejor con el conjunto completo de características. Además, cabe destacar que en el caso de las redes neuronales los modelos entrenados con las características extraídas de las imágenes mejoran respecto de los mismos modelos sin estas características. A pesar de seguir siendo modelos con una precisión muy baja, todos están por encima del resultado proporcionado por el modelo Knn(0,363).

En la tabla 8 mostramos los resultados de los experimentos detallados en el apartado anterior con todo el preprocesado hecho, es decir incluyendo la eliminación de los *outliners.*

	CSV	CSV + IMG	5 CSV	5 CSV + IMG
Gradient Boosting	0.770572	0.762233	0.773482	0.772484
XGBoost	0.772233	0.766626	0.773494	0.773143
ANN	0.685	0.711	0.666	0.670

Tabla 8 Precisión modelos después de eliminar outliners

Fuente: Elaboración propia.

Tanto para el algoritmo Gradient Boosting como para el XGBoost el mejor resultado se obtiene cuando le estamos pasando las 5 características más relacionadas. En cambio, en el caso de las redes neuronales el mejor resultado se obtiene cuando se utilizan todas las variables, incluyendo las extraídas de las imágenes.

El mejor resultado obtenido es 0.773494, con el algoritmo XGBoost y las 5 variables más correlacionadas con el *engagement*. Con el resultado obtenido podemos decir que hemos conseguido un modelo predictivo con una precisión aceptable (más del 70 %). Un muy buen punto de partida para próximas investigaciones. Respecto de nuestro modelo de referencia, ha aumentado la precisión hasta casi un 7%

Por otro lado, podemos observar una mejora relativa con el uso de características extraídas de imágenes en el algoritmo propuesto con redes neuronales. A pesar de esta pequeña mejora, no podemos afirmar que este tipo de características tengan influencia real en el entrenamiento de modelos. Si bien es cierto, que en el caso de los modelos *boosting* empeoran levemente cuando se añaden estas características.
Además, si tenemos en cuenta nuestro *baseline*, el único modelo con una precisión mejor es el modelo 10, es decir, el que utiliza todas las variables. Estos resultados podrían ser debidos tanto a la configuración de parámetros como a la cantidad de datos utilizada para el entrenamiento de los modelos.

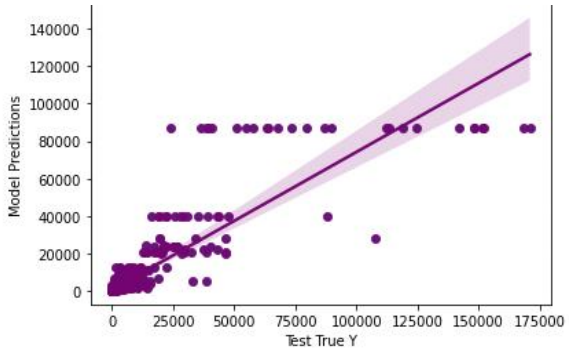

Figura 32 Diagrama de dispersión modelo XGBoost 5 csv.

Fuente: Elaboración propia.

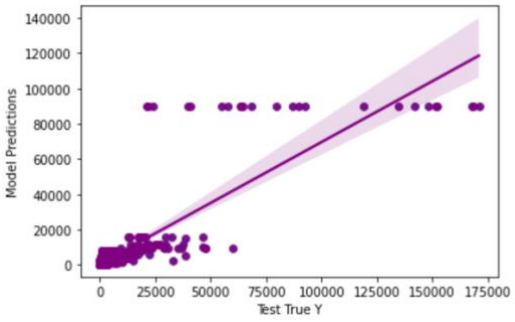

Figura 33 Diagrama de dispersión modelo ANN todo csv + img.

Fuente: Elaboración propia.

Si comparamos los diagramas de dispersión, se observa que con el predictor XGBoost (figura 32) los resultados se ajustan mejor, hay más linealidad que en la figura 33, que muestra el gráfico de dispersión del modelo ANN.

Esta pérdida de linealidad podría deberse a la introducción de las variables extraídas de las imágenes, ya que no existe una relación aparente entre estas variables y el número de *likes* que ha obtenido el *post*. Además, al hacer el estudio de correlación observamos que solo había una variable que realmente estuviera correlacionada con nuestra variable objetivo, que eran los *followers.*

Pero sin duda, lo que más llama la atención en esas gráficas es la línea continua que aparece entre los 80.000 y los 100.000 *likes.* Al llegar a este punto el modelo está prediciendo el mismo valor para todos los posts. Esto se podría explicar con la existencia de un sub-conjunto de los datos con los mismos valores en casi todas las variables, lo que podría hacer que el modelo predijera el mismo número de *likes* para todos los posts. Si, además de las variables asociadas al usuario, otras variables tienen el mismo valor para todos los *posts* se explica el comportamiento del modelo. En la figura 34 se ejemplifica con los valores que forman esa línea continua en los diagramas anteriores.

	Test True Y	Model Predictions	followers	nIgtv	business	verified	hour_of_day
404	36498	87302.546875	11112558.0	62.0	0.0	0.0	1.0
423	67967	87302.546875	11112558.0	62.0	0.0	0.0	1.0
603	113196	87302.546875	11112558.0	62.0	0.0	0.0	1.0
610	54864	87302.546875	11112558.0	62.0	0.0	0.0	1.0
706	89685	87302.546875	11112558.0	62.0	0.0	0.0	1.0
722	170940	87302.546875	11112558.0	62.0	0.0	0.0	1.0
904	73737	87302.546875	11112558.0	62.0	0.0	0.0	1.0
998	124762	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1017	86880	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1072	148037	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1212	41165	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1305	79465	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1520	63172	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1533	38722	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1586	39865	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1663	112137	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1840	148408	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1953	24220	87302.546875	11112558.0	62.0	0.0	0.0	1.0
1961	141835	87302.546875	11112558.0	62.0	0.0	0.0	1.0
2144	57803	87302.546875	11112558.0	62.0	0.0	0.0	1.0
2172	64020	87302.546875	11112558.0	62.0	0.0	0.0	1.0
2221	119211	87302.546875	11112558.0	62.0	0.0	0.0	1.0
2288	168157	87302.546875	11112558.0	62.0	0.0	0.0	1.0

Figura 34 Ejemplo misma predicción para mismo número de variables.

Fuente: Elaboración propia.

Podemos ver perfectamente como todas las variables toman el mismo valor, siendo evidente que todas las publicaciones pertenecen al mismo usuario y, además, los *posts* fueron publicados todos a la misma hora. Por esta razón, el modelo predice el mismo

número de *likes* para todos los *posts.* Esto nos indica que existen factores que no se están teniendo en cuenta e influyen en el comportamiento de nuestra variable dependiente. Además, esto nos indica que realmente el modelo está ajustándose a los datos del usuario que publicó el *post*, y no al propio *post.*

Para profundizar un poco más, y poder determinar qué modelo puede adaptarse mejor a la predicción de *engagement* vamos a analizar los primeros 20 resultados obtenidos con los datos de test, comparándolos con el valor real.

Con propósito de no ser redundante, y dado que en todas las opciones XGBoost ha demostrado ser más potente que Gradient Boosting cogeremos como referencia el modelo con un resultado más alto con este tipo de técnica. Y por otro lado, dado que se trata de una técnica diferente cogeremos el mejor resultado obtenido por la arquitectura ANN para analizar los resultados obtenidos en profundidad.

A la izquierda de la figura 35 encontramos las predicciones hechas por la ANN y a la derecha encontramos las predicciones del modelo que usa XGBoost, Viendo estos resultados se corrobora que las predicciones hechas por el modelo XGBoost se acercan más al valor real que los valores predichos por la ANN.

	Test True Y	Model Predictions
0	733	344.600189
1	2422	1089.782593
2	1004	784.154907
3	1098	880.457520
4	2577	1067.915283
5	5857	5518.607422
6	749	1158.257935
7	5030	8257.225586
8	7429	2327.132568
9	654	344.710175
10	173	547.335938
11	2405	1735.348022
12	1623	8257.225586
13	995	2278.593506
14	626	890.044556
15	12632	8062.309570
16	460	603.534912
17	686	671.428345
18	1222	1108.062256
19	17	490.103119

	Test True Y	Model Predictions
0	733	452.267670
1	2422	1648.900024
2	1004	798.538208
3	1098	1431.406738
4	2577	1441.434570
5	5857	5415.688965
6	749	945.778625
7	5030	3409.335449
8	7429	5171.046387
9	654	452.267670
10	173	367.512329
11	2405	3337.780029
12	1623	3409.335449
13	995	1850.809448
14	626	617.289551
15	12632	5828.248047
16	460	635.548401
17	686	798.538208
18	1222	1226.435181
19	17	321.186188

Figura 35 Izquierda: modelo ANN; Derecha: XGBoost

Fuente: Elaboración propia.

Aunque a grandes rasgos y en todo momento hemos visto como se ajusta mejor el modelo XGBoost entrenado con las 5 variables más correlacionadas. El hecho de que de estas 5 variables, solo haya una que proporciona información no proveniente del perfil, la hora de publicación del día, va a hacer que el modelo tienda a predecir el mismo valor de *likes* para todos los *posts* del mismo usuario, ya que, el modelo es capaz de encontrar ese patrón. Por ello, desechamos la opción de utilizar este modelo, dejando,

así como la opción más viable el modelo ANN utilizando todas las variables, incluidas las extraídas de las propias imágenes. Esta técnica ha sido capaz de obtener el mejor resultado con todo el conjunto de variables, e incluso podría estudiarse la viabilidad de eliminar algunas de las variables asociadas a los usuarios y generar nuevas para conseguir explicar de una manera más obvia el comportamiento de los *likes.*

Dada la complejidad de la variable objetivo y los diferentes factores que le influyen, haber conseguido una precisión alrededor del 70 % en esta primera toma de contacto cumple con nuestras expectativas.

A pesar de que la aplicación de estos modelos predictivos en casos reales podría ser prematura dada la precisión obtenida, toda la información conseguida puede ser de gran utilidad para la toma de decisiones en estrategias de *social media*, cosa que se abordará en el TFG de ADE.

8.Conclusiones

8.1 Principales aportaciones

Varios de los objetivos era profundizar y aprender sobre las metodologías utilizadas en el proyecto, objetivos que se han conseguido de manera muy satisfactoria.

En un inicio se propuso abordar el mundo de las redes sociales, en concreto el concepto de *engagement,* durante la trayectoria del proyecto se ha profundizado en este concepto y al mismo tiempo hemos conseguido traducir este concepto para poder trabajar con él en modelos predictivos. Para ello construimos una base de datos con un montón de características relacionadas con este concepto, cumpliendo así con el objetivo de recopilar datos de la propia *app* de Instagram para la investigación. Después de la extracción masiva de datos concluimos que es posible aplicar las técnicas utilizadas para cualquier área o dominio.

Una vez conseguidos los datos trabajamos con ellos en el preprocesado y manipulaciones correspondientes para conseguir una buena precisión predictiva. Han sido estas técnicas las que nos han permitido mejorar la precisión de nuestros modelos casi el doble. Por tanto, podemos afirmar la importancia que tiene el proceso de preparación de los datos para el éxito en los modelos propuestos.

A su vez, se ha cumplido con los objetivos de crear varios modelos de redes neuronales con distintos propósitos. Por un lado, podemos concluir que el uso de redes neuronales convolucionales es una buena técnica para la extracción de características en imágenes ya que se ha conseguido una precisión elevada (más del 80%). Por otro lado, hemos trabajado con varios modelos de regresión para predecir el *engagement* de los *posts* en el sector del running, consiguiendo una precisión mayor al 70 % después de hacer un buen preprocesado de datos. Por tanto, concluimos que estás técnicas podrían ser utilizadas para ayudar a la toma de decisiones en estrategias de marketing en redes sociales. Si bien, hemos como la cantidad de variables asociadas a los perfiles influyeran demasiado en la predicción del mismo valor de *likes.* Por lo que, deja una línea de investigación abierta para intentar nutrir nuestros modelos con variables asociadas directamente a los posts y eliminar algunas variables más relacionadas con el usuario.

Por último, se han comparado todos los modelos propuestos para determinar la influencia que podía tener el uso de características extraídas de las imágenes en el entrenamiento de los modelos. Los resultados obtenidos no contienen diferencias relevantes por lo que es difícil concluir que realmente tienen algún efecto en el entrenamiento de modelos. Es algo que se deberá seguir estudiando para sacar conclusiones válidas.

Por todo ello consideramos este trabajo una buena aproximación inicial tanto a la investigación del *engagement* enfocada al sector del running como a lo que podría ser, en el futuro, un modelo predictivo de *engagement* de los posts en Instagram.

8.2 Relación con los estudios cursados

Tras la realización del presente proyecto se han abordado distintas áreas de estudio y especialidades. Este trabajo ha supuesto un aprendizaje continuo. Desde la primera lluvia de ideas hasta la última corrección la evolución ha sido muy gratificante y se ha intentado resolver el problema de manera innovadora y desde un punto de vista analítico y reflexivo, consiguiendo así unos resultados mucho más óptimos que en las primeras ejecuciones.

Centrándonos en las distintas áreas estudiadas en la carrera, encontramos una fuerte relación con asignaturas de la rama de sistemas de información. A través de asignaturas como Sistemas de información estratégicos me familiarice con los procesos ETL así como el manejo de grandes volúmenes de datos.

También hay una fuerte relación con asignaturas como introducción a la estadística, donde se empezaron a introducir de forma muy simple modelos de regresión.

Por último, la asignatura que me acercó al mundo de la inteligencia artificial y trata las distintas utilidades que existen dentro de este campo, fue sistemas inteligentes. Fue el contenido de esta asignatura el que me motivó a ampliar los conocimientos en esta área y aplicarlos a un problema real.

8.3 Limitaciones del trabajo

Este trabajo ha supuesto un verdadero reto. En el inicio de este, era difícil hacerse una idea de todos los procesos que abarcaba y todo el trabajo y conocimiento necesario para llevarlo a cabo. Si bien, hemos intentado adaptarlo al máximo a nuestras posibilidades, hemos tenido que lidiar con algunas limitaciones.

En primer lugar, cuando empezó el proyecto mi formación en algunas áreas no era completa, ya que se propusieron tecnologías con las que nunca había trabajado. Por ello, realicé varios tutoriales de Python y sus librerías e hice un curso especializado en *Deep learning* para el reconocimiento de señales e imágenes ofrecido por la UPV lo que me ayudó a asentar unas bases y poder sacarle el máximo partido a este proyecto.

También ha sido realmente complicada la investigación sobre el problema específico que se presenta. El uso de técnicas de *machine learning* es algo bastante novedoso lo que limita bastante la información. Pero, además, en el escenario donde nosotros estábamos interesados, el running, no se encontró ninguna evidencia de que alguna vez haya sido investigado este dominio con estas técnicas. Esto ha supuesto ir ‘a ciegas’ en varias ocasiones. No disponer de información suficiente ha sido limitante en varias ocasiones.

Otra de las limitaciones que más ha influido en la realización del proyecto ha sido el acceso a datos. El límite de peticiones a la API ha hecho que este proceso fuera lento y no pudiéramos ampliar el dataset antes de la entrega.

8.4 Líneas futuras

Este proyecto abre muchos frentes de investigación. Lo abordado en este estudio es solo una pequeña parte de todo lo que se podría desarrollar. Al fin y al cabo, nuestro TFG es un punto de partida. A continuación, detallaremos los distintos caminos que puede tomar el trabajo elaborado:

1. Mejora de los modelos propuestos utilizando nuevas variables y ajustando los parámetros de la manera óptima.
2. Extracción de nuevas características a través de las imágenes, como por ejemplo el color predominante, la detección de marcas…
3. Utilización de los datos de tipo texto que acompañan a las imágenes.
4. Ampliación de dicho estudio a tipos de post como vídeos o *sidecars.*
5. Aplicación de este proceso a otros núcleos sociales de interés.
6. Análisis de toda la información recopilada para poder hacer recomendaciones, además de ofrecerles la predicción de *engagement*.

En el TFG de ADE, actualmente en desarrollo, se van a elaborar una serie de propuestas para mejorar la comunicación en Instagram, dirigidas a las empresas con interés en vender sus productos a la comunidad runner. En concreto, se estudiarán los factores que influyen en el *engagment*. Estas propuestas se van a realizar tras analizar toda la información obtenida en el presente proyecto. Después, se elaborarán diferentes modelos estadísticos para identificar aquellas variables que más influyen en el *engagement.* A partir de estos resultados, se propondrán pautas de actuación para las empresas relacionadas con el running con el fin de mejorar su marketing de influencers.

9.Bibliografía

[1] Redacción PuroMarketing, «En qué redes sociales se están centrando las empresas en su estrategía de Marketing con influencers,» 22 06 2021. [En línea]. Available: https://www.puromarketing.com/42/35446/redes-sociales-estan-centrando-empresas-estrategia-marketing-influencers.

[2] E. Armas, «12 joyas 'beauty' que se hicieron virales gracias a la magia de TikTok,» 21 02 2022. [En línea]. Available: https://smoda.elpais.com/belleza/12-joyas-beauty-que-se-hicieron-virales-gracias-a-la-magia-de-tiktok/100474610/image/100474612.

[3] N. Hotz, «CRISP-DM: Towards a Standard Process Model for Data Mining,» 16 04 2022. [En línea]. Available: https://www.datascience-pm.com/crisp-dm-2/.

[4] A. Bosch Rue, J. Casas Roma y T. Lozano Bagen, Deep learning : principios y fundamentos, Barcelona: Editorial UOC, 2019.

[5] Mckinsey & Company, «An executive's guide to AI,» 2020. [En línea]. Available: https://www.mckinsey.com/business-functions/quantumblack/our-insights/an-executives-guide-to-ai.

[6] E. Alpaydin, Introduction to Machine Learning, Cambridge, Massachusetts: The MIT Press, 2014.

[7] T. Mitchell, Machine Learning, McGraw Hill.

[8] P. Bruce, A. Bruce y A. Gedeck, Estadística práctica para ciencia de datos con R y Python, MARCOMBO, 2022.

[9] R. Dolan, J. Conduit, C. Frethey-Bentham, J. Fahy y S. Goodman, «Social media engagement behavior: A framework for engaging customers through social media content,» *European Journal of Marketing,* p. Vol. 53 No. 10, 2019.

[10] R. Brodie y L. Hollebeek, «Customer Engagement: Conceptual Domain, Fundamental Propositions, and Implications for Research,» *Journal of Service Research,* pp. 252-271, 2011.

[11] J. L. K. Van Doorn, «Customer engagement behavior: Theoretical foundations and research directions,» *Journal of Service Research,* pp. 253-266, 2010.

[12] S. Alhabash, A. McAlister y A. Hagerstrom, «Between Likes and Shares: Effects of Emotional Appeal and Virality on the Persuasiveness of Anticyberbullying Messages on Facebook,» *Cyberpsychology, Behavior, and Social Networking,* pp. 175-182, 2013.

[13] Apple, «Apple,» 2021. [En línea]. Available: https://www.apple.com.

[14] metricool, «metricool,» 2022. [En línea]. Available: https://metricool.com/es/.

[15] LikekyAI, «LikelyAI,» 2017. [En línea]. Available: https://www.likelyai.com.

[16] Beautiful Destinations, «Beautiful Destinations,» 2022. [En línea]. Available: https://beautifuldestinations.com.

[17] A. Krueger, «This Company Can Predict The Number Of Likes An Instagram Photo Will Get,» 21 07 2016. [En línea]. Available: https://www.forbes.com/sites/alysonkrueger/2016/07/21/this-company-can-predict-the-number-of-likes-an-instagram-photo-will-get/?sh=e8cc55f5a47b.

[18] Munideporte, « La práctica del running aumentó en España un 5,13% en el último año,» 27 05 2021. [En línea]. Available: https://www.valgo.es/blog/la-practica-del-running-aumento-en-espana-un-513-en-el-ultimo-ano?elem=264420.

[19] A. Graf, «instaloader.github.io,» 2022. [En línea]. Available: https://instaloader.github.io. [Último acceso: 01 2022].

[20] REAL ACADEMIA ESPAÑOLA, «RAE,» [En línea]. Available: https://www.rae.es/observatorio-de-palabras/influencer. [Último acceso: 01 2022].

[21] InBeat, «InBeat,» 2021. [En línea]. Available: https://www.inbeat.co.

[22] Runnea, «Los 11 bloggers de running más influyentes de España,» 14 10 2021. [En línea]. Available: https://www.runnea.com/articulos/running-news/2017/03/bloggers-running-influyentes-espana-2506/.

[23] A. Roldán, «Las instagramers más influyentes del universo running en España,» 13 09 2018. [En línea]. Available: https://www.runnea.com/articulos/running-news/2018/09/instagramers-influyentes-universo-running-espana-3805/.

[24] C. M. Pastor, «8 cuentas de runners en Instagram que deberías estar siguiendo,» [En línea]. Available: https://www.carreraspopulares.com/noticia/8-cuentas-de-runners-en-instagram-que-deberias-estar-siguiendo.

[25] C. C. Ungría, «Los runners españoles más influyentes en Instagram,» 28 10 2018. [En línea]. Available: https://www.trecebits.com/2018/10/28/los-runners-espanoles-mas-influyentes-en-instagram/.

[26] Affde, «Principales influencers del running en las redes sociales,» 05 10 2021. [En línea]. Available: https://www.affde.com/es/top-running-influencers.html.

[27] El comerico, «Los 6 runners más influyentes en Instagram,» 14 11 2018. [En línea]. Available: https://elcomercio.pe/deporte-total/running/guia-runner/fotos-6-runners-influyentes-instagram-noticia-577183-noticia/.

[28] G. Godfrey, «10 global running influencers you should know,» [En línea]. Available: https://www.fashionmonitor.com/blog/wD/10-global-running-influencers-you-should-know.

[29] Feedspot Media Database Team, «Top 150 Running Instagram Influencers most followed,» 9 12 2020. [En línea]. Available: https://blog.feedspot.com/running_instagram_influencers/.

[30] Libertad Digital, «Los corredores más influyentes del running en España,» 31 08 2016. [En línea]. Available: https://www.libertaddigital.com/deportes/mas-deporte/2016-08-31/los-corredores-mas-influyentes-del-running-en-espana-1276575752/.

[31] M. Sicilia, M. Palazon, I. Lopez y M. Lopez, Marketing en redes sociales, Madrid: Esic Editorial, 2021.

[32] A. Nadian-Ghomsheh, «Indoor-outdoor image classification using dichromatic reflection model and Haralick features,» 2017. [En línea]. Available: https://figshare.com/articles/dataset/Indoor-Outdoor_dataset/4595323/1.

[33] A. Krizhevsky, I. Sutskever y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» 2012. [En línea]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[34] M. &. Company, «An executive's guide to AI,» 2020. [En línea]. Available: https://www.mckinsey.com/business- functions/mckinsey-analytics/our-insights/an-executives-guide-to-ai.

[35] A. Brooks, «10 Best Running Instagram Accounts to Follow ASAP,» [En línea]. Available: https://www.runtothefinish.com/top-instagram-runners-inspire/.

10.Anexo

10.1 ODS

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero.**				x
ODS 3. **Salud y bienestar.**		x		
ODS 4. **Educación de calidad.**				x
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento.**				x
ODS 7. **Energía asequible y no contaminante.**				x
ODS 8. **Trabajo decente y crecimiento económico.**	x			
ODS 9. **Industria, innovación e infraestructuras.**		x		
ODS 10. **Reducción de las desigualdades**.			x	
ODS 11. **Ciudades y comunidades sostenibles**.				x
ODS 12. **Producción y consumo responsables.**		x		
ODS 13. **Acción por el clima.**				x
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**				x
ODS 16. **Paz, justicia e instituciones sólidas.**				x
ODS 17. **Alianzas para lograr objetivos.**				x

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Los Objetivos de Desarrollo Sostenible que se pueden vincular a este proyecto son, en primer lugar, el trabajo decente y crecimiento económico. Con la investigación que hemos llevado a cabo y el resultado conseguido estos modelos predictivos podrían ayudar a las empresas a tomar decisiones adecuadas en sus estrategias de marketing en redes sociales. De esta manera podrían incrementar su *engagement* y en consecuencia sus ventas.

Con una relación menos directa, este proyecto se ha desarrollado tras un dominio que se relaciona directamente con la salud y el bienestar, el *running.* Este grupo social realiza una actividad que presenta grandes beneficios para la salud y aunque, este proyecto se ha centrado en la aplicación de modelos predictivos con objetivo de predecir el *engagement* de los posts, el dataset construido es un material que podría ser valido para estudios relacionados con la salud. Además, dicho dataset se ha almacenado en la nube y puede ser utilizado por cualquier persona sin necesidad de gastar memoria en su disco duro, este hecho relaciona nuestro proyecto con el compromiso de consumo responsable.

Otro de los objetivos que está presente en el proyecto es el de industria, innovación e infraestructura ya que se ha construido un dataset específico para nuestro trabajo y no hay evidencia de que se haya investigado aplicando las técnicas y modelos utilizados en el presente proyecto.

Por último, cabe mencionar una pequeña relación con la reducción de las desigualdades ya que en el estudio se han utilizado datos de toda la población sin importar la nacionalidad, género o cultura de los perfiles seleccionados.