Social Media Popularity Prediction

[*] Predicción de la popularidad de una imagen en redes sociales utilizando tanto la imagen como los datos del usuario que la publica

1st Bruno Castagnino Rossi
Universidad de San Andrés
Buenos Aires, Argentina
bcastagninorossi@udesa.edu.ar

2nd Dan Shlamovitz
Universidad de San Andres
Buenos Aires, Argentina
dshlamovitz@udesa.edu.ar

***Resumen*—Este trabajo aborda el desafío *Social Media Prediction 2019*, centrado en el análisis de datos de la red social Flickr. El objetivo principal es desarrollar un modelo predictivo capaz de anticipar la popularidad futura de una publicación a partir de sus características iniciales, como su contenido, metadatos y atributos contextuales. Para ello, exploramos diversas técnicas de machine learning, evaluando su desempeño en función de métricas clave. Los resultados obtenidos ofrecen una visión profunda sobre los factores que influyen en la popularidad en plataformas sociales.**

I. Introducción

En la era digital, las redes sociales han adquirido un rol fundamental como plataformas para la difusión de contenido y la interacción entre usuarios. Comprender y predecir la popularidad de una publicación en estas plataformas se ha convertido en un tema de creciente interés, tanto desde el punto de vista académico como para aplicaciones prácticas en marketing, diseño de estrategias de contenido y análisis de comportamiento en línea.

El desafío *Social Media Prediction 2019* proporciona un conjunto de datos de la red social Flickr, que incluye información relevante sobre publicaciones, como metadatos, etiquetas, y atributos contextuales. A partir de este conjunto de datos, el objetivo del presente trabajo es desarrollar un modelo predictivo que permita estimar la popularidad futura de una publicación basándose en sus características iniciales.

En este estudio, exploramos diversas técnicas de machine learning, evaluando su capacidad para capturar patrones significativos en los datos. Además, analizamos los factores que más contribuyen al éxito de una publicación, lo que permite obtener información valiosa sobre los mecanismos subyacentes que impulsan la viralidad en redes sociales.

En primer lugar, en la sección de Trasfondo teórico se presentan los módulos utilizados para generar modelos de predicción. Luego, en la sección Dataset, explicamos qué datos fueron utilizados y qué tecnicas se les aplicaron en pos de mejorar los modelos desarrollados. En la sección de Desarrollo y Análisis de resultados mostramos los modelos desarrollados y comparamos su performance. Por último, en la sección de conclusiones resaltamos los resultados más importantes y en la sección de Trabajo a Futuro presentamos posibles modificaciones o agregaciones que se podrían hacer para seguir investigando.

II. Trasfondo Teórico

II-A. Métrica de Performance

Para el desarrollo de este trabajo medimos la performace de los modelos realizados utilizando la métrica RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{D} (\hat{y} - y)^2}{D}} \tag{1}$$

Esta métrica permite penalizar los errores más grandes con el objetivo de desarrollar modelos robustos.

II-B. Módulos utilizados

Para generar features a partir de las imágenes, utilizamos el codificador (encoder) de imágenes de CLIP, basado en **ResNet50**, reentrenado con más de 400 millones de pares de imágenes y descripciones. Este modelo proyecta la imagen a un espacio de 512 dimensiones, alineándola con la descripción que mejor coincide (match).

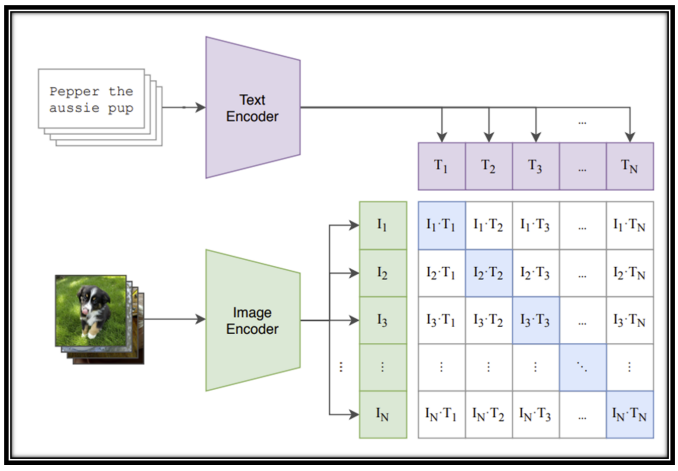

Figura 1. Funcionamiento del modelo CLIP

De este modelo solo nos quedamos con el encoder de imágenes.

Para las etapas de predicción, utilizamos los siguientes métodos:

- **MLP**
- **XGBoost**

- **CONV1D** combinado con **MLP**

Cada uno de ellos presenta características distintas que resultaron útiles para seleccionarlos de manera coherente en el desarrollo de los modelos.

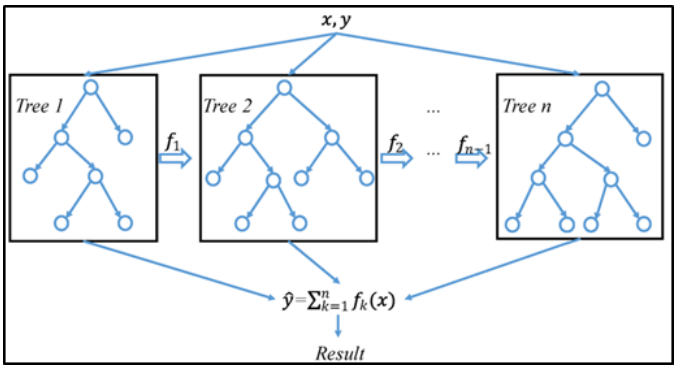

Figura 2. Funcionamiento de XG boost para regresión

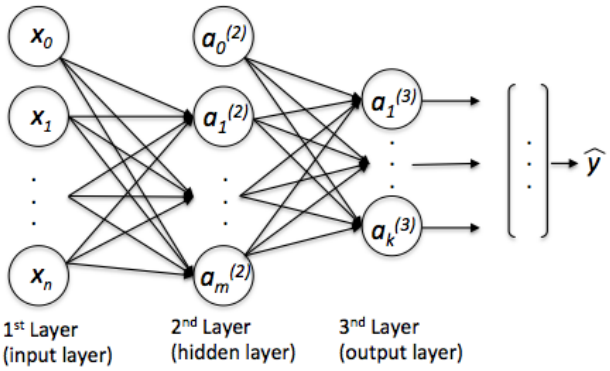

Figura 3. Funcionamiento de una MLP para clasificación (en nuestro caso reemplazamos la ultima capa por una capa lineal)

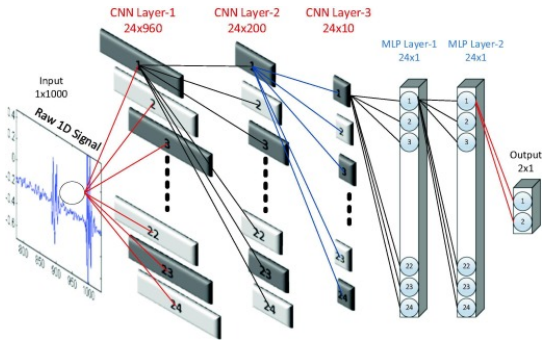

Figura 4. Funcionamiento de una estructura que convina CONV1D y MLP para clasificación (en nuestro caso reemplazamos la ultima capa por una capa lineal)

Para la reducción de dimensionalidad, utilizamos el método de Análisis de Componentes Principales (PCA), que reduce la dimensión de los datos manteniendo la mayor parte de la varianza. Dada una matriz de datos $X \in \mathbb{R}^{n \times m}$, PCA encuentra una matriz de componentes principales $W \in \mathbb{R}^{m \times k}$ tal que la representación reducida es:

$$Z = XW$$

donde Z es la representación de los datos en el espacio de menor dimensión k.

III. Dataset

El dataset provisto por la organización "SMP Challenge.^está compuesto de más de 400000 imágenes scrapeadas de la red social Flickr a lo largo de 2019 junto con metadata tanto de las imágenes en sí como de los usuarios que las publicaron. Finalmente, se provee una única columna numérica como target que se compone del "Popularity Score"de cada publicación. Este es un índice que se calcula aplicando una función (log) a la relación entre el número de views de una publicación y el tiempo transcurrido desde que fue posteada.

Debido al gran tamaño del dataset original, decidimos hacer un subsampling del mismo para facilitar el trabajo y el tiempo de cómputo a lo largo de nuestra experimentación. Se extrajeron 80000 muestras de imagen-metadata que luego se dividieron en sets de train, validation y test con los que se trabajó a lo largo de todo el proyecto. El set de train se compuso de 64800 instancias imagen-metadata , el de validation de 7200 y el de test de 8000. A continuación se muestran las subsecciones de datos con las que se trabajó.

III-A. Metadatos del usuario

Se enumeran los metadatos asociados al usuario que realizó la publicación original que se utilizaron:

- Número total de fotos publicadas por el usuario.
- Categoría del usuario dentro de la plataforma (Pro User vs. Non Pro User).

Para preparar estos datos para el procesamiento por nuestros modelos se realizó one-hot encoding de la columna **Categoría del usuario**. La columna con el número de fotos totales por usuario no se modificó porque ya era numérica.

III-B. Metadatos de la imagen

Se enumeran los metadatos asociados a cada imagen que se utilizaron:

- Categoría de la imagen (por ejemplo, "Weather-Season").
- Subcategoría de la imagen (por ejemplo, Raining").
- Concepto de la imagen (por ejemplo, Ümbrella").
- Timestamp de publicación de la imagen.

Para preparar estos datos para el procesamiento por nuestros modelos se realizó one-hot encoding de las columnas **Categoría**, **Subcategoría** y **Concepto**. La columna con el timestamp de la publicación fue convertida a one hot del día de la semana y mes de la publicación.

III-C. Imagen

Para el procesamiento de cada imagen, en primer lugar se realizó el cálculo de múltiples features de alto nivel a partir de cada una de ellas, que se enumeran a continuación:

- Clarity Contrast (Mide la claridad de la imagen en función del contraste de bordes).).
- Hue Count (Evalúa la diversidad de colores presentes en la imagen).
- Brightness Contrast (Mide el rango de brillo en la imagen).
- Color Entropy (Calcula la complejidad de la información de color en la imagen usando entropía).
- Composition Geometry (Mide la desviación del centroide de bordes en la imagen con respecto al centro de la misma).

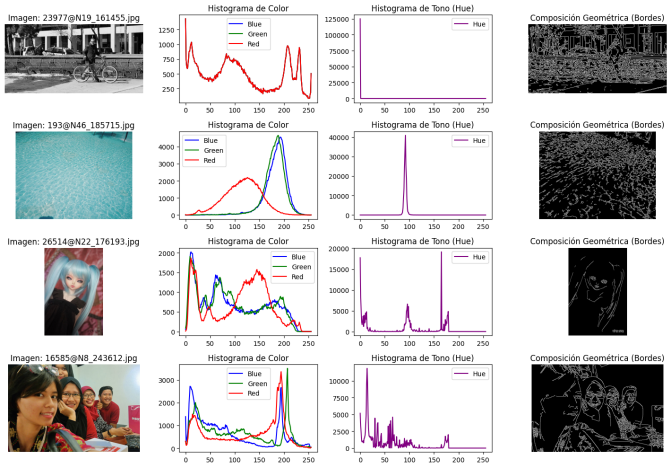

Figura 5. Evaluación de features de alto nivel de algunas de las imágenes del dataset.

Luego, se calculó un vector de features inherentes a cada imagen, utilizando el encoder del modelo CLIP que se explicó en la sección de trasfondo teórico. Para cada imagen, el encoder devolvió un vector numérico de dimensión 512x1. Concatenando este vector con las features de alto nivel calculadas previamente, se obtuvo un vector de 519x1 para cada imagen.

IV. Desarrollo y Análisis de Resultados

IV-A. Benchmark

Una vez realizada la limpieza y el curado de los datos, desarrollamos un modelo de referencia (benchmark). El objetivo de este modelo es ser sencillo y de rápida convergencia, proporcionando un punto de partida más sólido que un *random guess*. Para este modelo entrenamos un modelo **XGboost**

Para este modelo hicimos búsqueda de hiperparámetros por grid search con el set de validación y encontramos que lo mejor fue utilizar 2000 estimadores de profundidad máxima de 18 y un learning rate de 0.01.

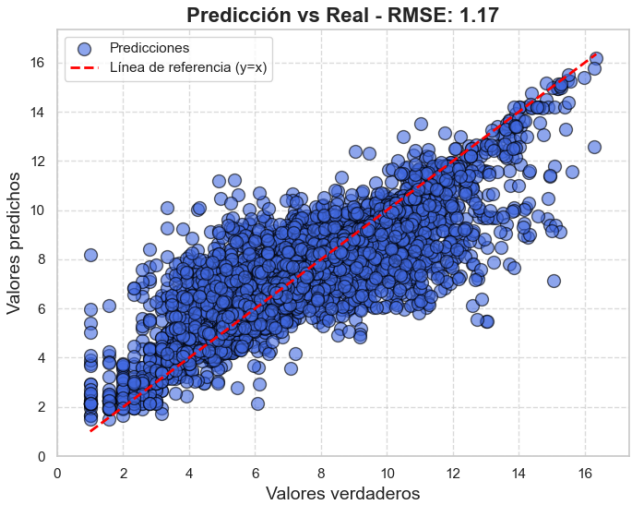

Figura 6. Valores reales vs. predichos para el modelo benchmark XGboost

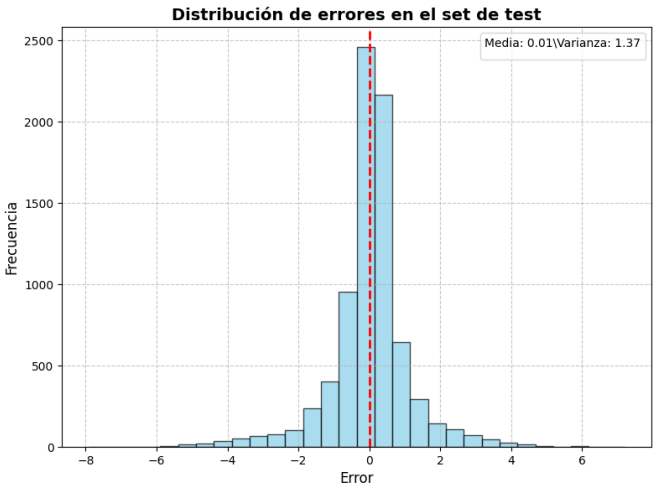

Figura 7. Distribución del error para el modelo benchmark XG boost

Hemos obtenido significativos resultados con este modelo, ya que el score máximo es 20 y el error cuadrático medio alcanzado representa solo el 6 % de dicho score. Es importante destacar que este modelo muestra una baja variabilidad en el error para el test set.

Para intentar mejorar el benchmark, utilizamos una técnica de feature engineering que consiste en realizar combinaciones no lineales entre columnas, dado que los árboles de decisión generan separadores lineales. Sin embargo, no obtuvimos mejores resultados.

Usando este modelo, aplicamos el método de *feature importance* proporcionada por la librería XGBoost. Esta técnica nos ayudó a identificar cuáles características que estaban más relacionadas con el rendimiento del modelo, lo que nos permitió enfocar nuestros esfuerzos en ellas para el desarrollo de modelos futuros y para entender mejor la correlación de la metadata con la popularidad de la imagen.

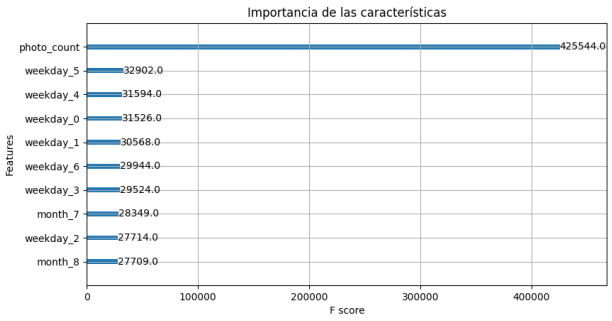

Figura 8. Feature importance para XGBoost

IV-B. XGBoost con CLIP y PCA

Como pudimos ver en la sección anterior, las predicciones del benchmark mostraron resultados sorprendentemente buenos considerando que no se utilizó en ningún momento la información de las imágenes. Es por este motivo que decidimos desarrollar un modelo que tomara la predicción inicial realizada por nuestro benchmark e intentara corregirla en base a los datos de las imágenes. Primero, concatenamos el encoding de las imágenes realizado por CLIP con los features de alto nivel y los datos tabulares. Luego, aplicamos PCA para la reducción de dimensionalidad y, finalmente, pasamos los datos reducidos por una etapa de predicción utilizando un modelo de boosting.

Utilizamos 2000 estimadores de profundiad máxima 12, learing rate de 0.01 y PCA a 300 dimensiones.

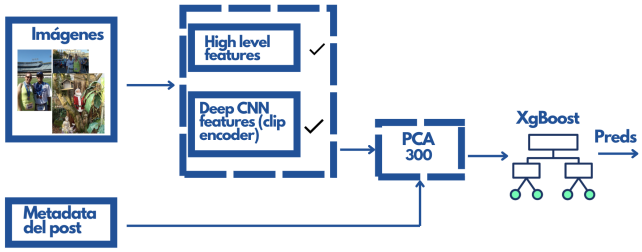

Figura 9. Estructura del modelo que combina la la metadata con el encoder de clip y las highlevel features y luego hace PCA para finalmente pasarlo por una etapa de predicción utilizando boosting.

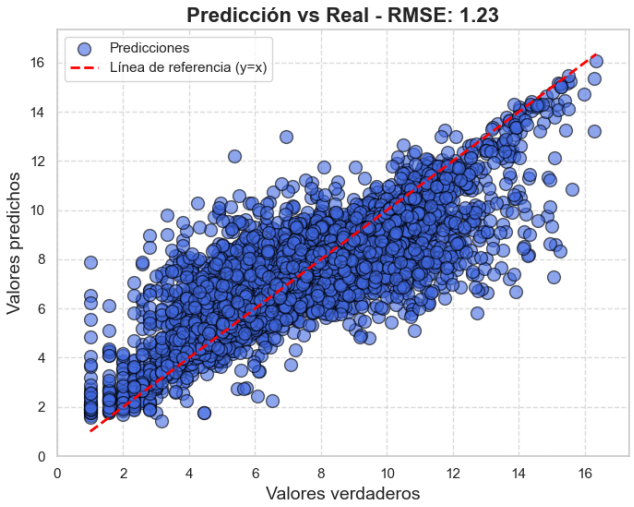

Figura 10. Valores reales vs. predichos para este modelo

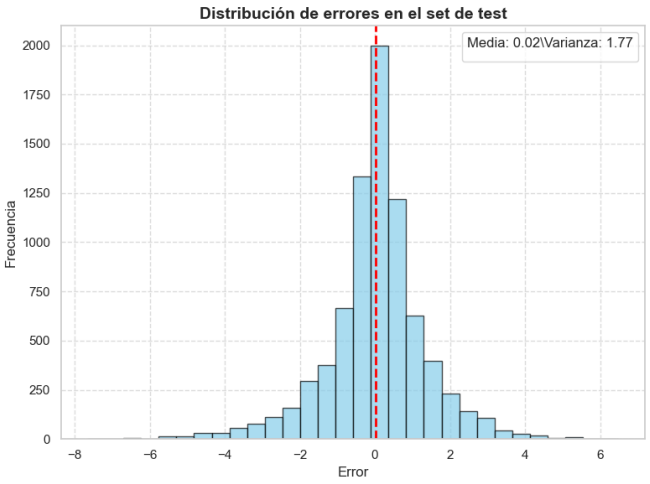

Figura 11. Distribución del error para este modelo

Los resultados que obtuvimos con este modelo fueron peores que el benchmark. Teorizamos que fue por que los hiperparámetros no buscados con el tiempo necesario ya que este método es significativamente más lento para converger que el benchmark propuesto anteriormente.

IV-C. XGBoost con CLIP y MLP

Luego, decidimos desarrollar un modelo ensamblado que tomara como input un nuevo vector (creado al concatenar la predicción realizada para cada imagen por nuestro benchmark con el vector de características de cada imagen (el vector de 512 obtenido con el encoder de CLIP más el vector de 7 features de alto nivel) y lo procesara mediante una red neuronal MLP fully connected.

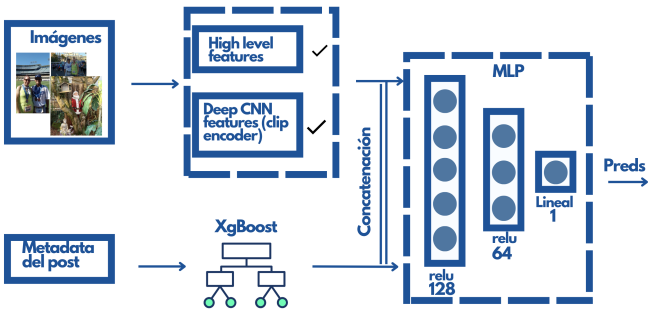

Figura 12. Estructura del modelo tipo Ensemble combinando predicción del benchmark XGBoost con features propias de la imagen + MLP

La arquitectura de la red y los hiperparámetros se eligieron en base a un grid search y los resultados obtenidos. La arquitectura final de la red utilizada fue de una primera capa fully connected de 128 neuronas luego de la cual se implementa batch normalization y una función de activación ReLU. Luego una capa FC de 64 neuronas (con su respectivo batch normalization y ReLU) y, por último, una única neurona de salida que produce la inferencia final.

Para el entrenamiento se utilizó el optimizador ADAM con un learning rate fijo de 0.0005 y se implementó regularización mediante dropout del 50 % en las dos primeras capas lineales. El entrenamiento se realizó por 30 epochs con batch size de 32. Finalmente, se probó la performance del modelo contra el set de test, como se evidencia en la figura 10.

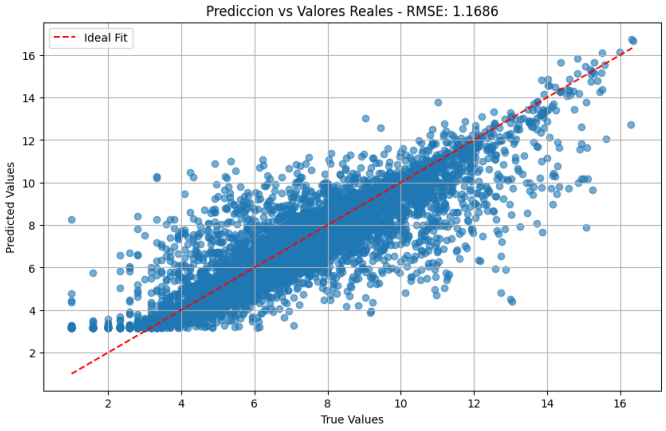

Figura 13. Valores predichos vs. reales según modelo Ensemble XGBoost + MLP

También evaluamos para el set de test la distribución del error de las predicciones, como podemos ver en la figura 11.

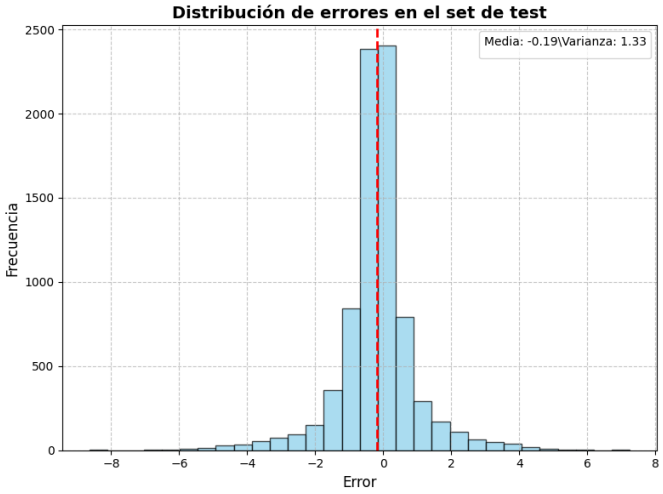

Figura 14. Distribución del error de las predicciones del modelo Ensemble XGBoost + MLP con el set de test

IV-D. XGBoost con CLIP CONV1D y MLP

Como experimento final decidimos intentar procesar los features obtenidos de las imágenes por una red convolucional unidimensional que nos permitiera correlacionarlas previo a introducir la predicción realizada por el modelo benchmark. Para esto, en primer lugar, tomamos para cada imagen el vector compuesto por el encoding realizado por el modelo CLIP concatenado con el vector de features de alto nivel y realizamos una reducción de su dimensionalidad con PCA manteniendo las 20 dimensiones más significativas. Luego, utilizamos este vector como input de la red convolucional unidimensional. A la salida de esta red se le concatenó la predicción realizada por el modelo benchmark y este vector compuesto se utilizó como input de una MLP. La arquitectura de este modelo puede verse en la figura 15.

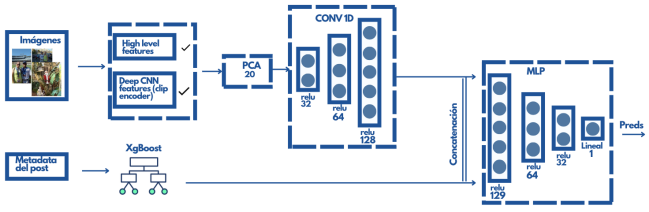

Figura 15. Arquitectura de la red Ensemble XGBoost + Conv1D + MLP

La idea de este modelo era que, mediante la utilización de una red convolucional unidimensional, se lograran obtener correlaciones significativas entre los features abstractos obtenidos por el encoder CLIP a partir de las imágenes.

Luego de cada capa convolucional se aplicó una capa de batch normalization. En cada capa se utilizó la ReLU como función de activación. Para evitar overfitting se utilizó un dropout del 75 % de las neuronas. Se utilizó el optimizador

ADAM con un learning rate de 0.0001 y weight decay. El entrenamiento se realizó por 20 epochs con batch size de 32.

Finalmente, se probó la performance del modelo contra el set de test, como se evidencia en la figura 16. La distribución del error sobre los datos de test se puede observar en la figura 17.

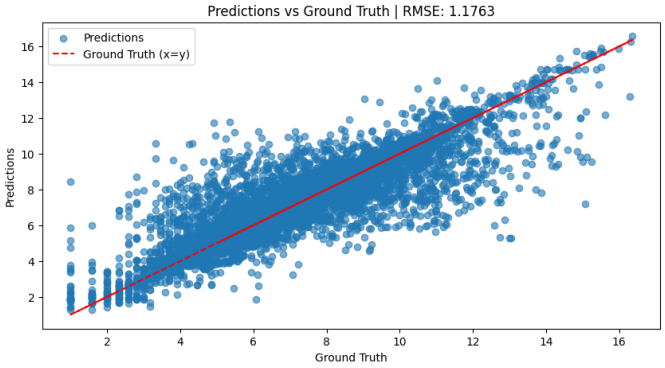

Figura 16. Predicciones vs. valores reales sobre set de test de arquitectura Ensemble XGBoost + Conv1D + MLP

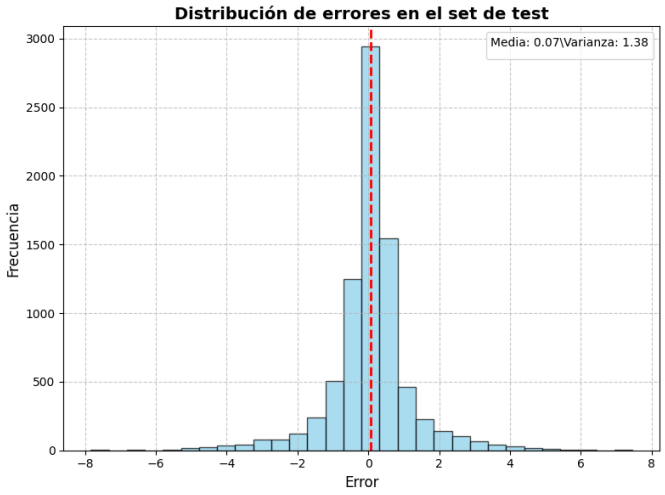

Figura 17. Distribución del error de predicción sobre set de test de arquitectura Ensemble XGBoost + Conv1D + MLP

V. Comparación Entre Modelos

Realizamos ahora una comparación entre todos los modelos investigados.

Modelo	Métricas en test set		
	RMSE	***Error medio***	***Varianza del error***
XGBoost, datos tabulares	1.178	0.01	1.37
XGBoost, CLIP vectors y PCA	1.238	0.02	1.77
MLP, XGBoost y CLIP vectors	**1.168**	-0.19	1.33
MLP, XGBoost y CLIP vectors	1.176	0.07	1.38

VI. Conclusiones

A lo largo de este trabajo hemos podido observar la performance de distintos modelos predictivos tomando en consideración distintos subsets de datos. Como se ha visto en la sección de desarrollo experimental, logramos obtener una performance muy buena utilizando solamente la metadata de la imagen y del usuario que realizó cada publicación. Mientras que este resultado nos resultó intuitivo (después de todo no debe extrañarnos que la publicación de una imagen de mala calidad por un usuario con millones de seguidores tenga más popularidad que la publicación de una imagen de excelente calidad por un usuario recién registrado en la plataforma), esperábamos observar un grado de impacto mucho mayor al introducir la información propia de las imágenes en nuestras predicciones.

Como puede observarse, solo obtuvimos una mejora muy leve en el error de predicción con el modelo que utiliza los features de las imágenes obtenidos por CLIP concatenados con los features de alto nivel y la predicción realizada por nuestro benchmark en una MLP respecto del modelo benchmark por sí solo. Pero la mejora resulta marginal y no se descarta que sea despreciable por completo, debiéndose a la distribución particular de los datos utilizados. Con ninguna de las otras arquitecturas logramos mejorar la performance obtenida con el benchmark.

Con esto podemos concluir que la popularidad de una publicación depende en su mayoría de quién la publica, cuándo y dónde. Lo que se publica parece ser secundario, o incluso insignificante según nuestros experimentos.

VII. Trabajo a Futuro

A partir de las conclusiones obtenidas en este proyecto, consideramos que una línea prometedora para continuar la investigación sería entrenar el modelo utilizando el conjunto de datos completo de 300000 imágenes, ya que actualmente estamos trabajando con un submuestreo. Además, la implementación de técnicas de cloud computing podría optimizar el proceso de entrenamiento, permitiendo una búsqueda más exhaustiva de hiperparámetros en los métodos que involucran redes neuronales multicapa (MLP), dado que estos modelos requieren más tiempo para converger.

Por otro lado, creemos que es crucial desarrollar una estructura que permita correlacionar la metadata con los datos de las imágenes antes de realizar las predicciones, lo que podría mejorar significativamente los resultados obtenidos.

Referencias

[1] W.-H. C. N.-H. Y. Fatma S. Abousaleh, Yu Tsao, "Multimodal deep learning framework for image popularity prediction on social media," 2019.

[1]