Informe de Justificación de Decisiones

Este informe detalla las decisiones tomadas durante el desarrollo de un modelo predictivo de redes neuronales.

Preprocesamiento de Datos Tabulares

El archivo CSV proporcionado contiene información tabular relevante. Se llevaron a cabo varias etapas de preprocesamiento para garantizar la calidad y coherencia de los datos. Una de las primeras tareas fue la normalización de las columnas de latitud y longitud, utilizando una escala Min-Max. Esto permitió que las métricas geográficas estuvieran dentro de un rngo comparable, evitando que las diferencias de escala influyeran en el modelo.

Adicionalmente, se creó una nueva columna denominada target, calculada a partir de la fórmula:

$$target=(Likes+Bookmarks−Dislikes) / (Likes+Bookmarks+Dislikes)$$

Esta fórmula equilibra las interacciones positivas y negativas, asegurando que el target refleje la proporción relativa de interacciones en lugar del volumen absoluto. Los valores se normalizan dentro del rango [-1, 1], lo que facilita su uso en un modelo de regresión.

De esta forma, un alto nivel de aprobación (más Likes y Bookmarks) genera valores cercanos a 1, mientras que un predominio de interacciones negativas (Dislikes) produce valores cercanos a -1. Este enfoque elimina el sesgo del volumen total de interacciones, permitiendo que el modelo enfoque su análisis en la relación entre las métricas positivas y negativas.

Para procesar las variables categóricas presentes en el conjunto de datos, se utilizó la técnica de codificación one-hot encoding, que transforma dichas variables en representaciones binarias, preparándolas para su uso en redes neuronales.

Dividir los datos en subconjuntos de entrenamiento, validación y prueba fue una decisión crucial para garantizar la robustez del modelo. Inicialmente, se utilizó la función train_test_split para separar los datos en 80% para entrenamiento y 20% para prueba, empleando una semilla fija (random_state=10) para asegurar la reproducibilidad. Posteriormente, el conjunto de entrenamiento se subdividió en entrenamiento (80%) y validación (20%).

Se utilizaron Data Loaders con un tamaño de lote de 32, lo que facilitó el manejo eficiente de los datos durante el entrenamiento. Este tamaño de lote es estándar y ofrece un buen equilibrio entre el uso de memoria y la estabilidad en la actualización de los parámetros.

Preprocesamiento de Imágenes

En el caso de las imágenes, el preprocesamiento incluyó varias transformaciones diseñadas para mejorar la capacidad de generalización del modelo. Las imágenes se redimensionaron a 128x128 píxeles y se normalizaron utilizando la media y desviación estándar de los valores de píxeles en el conjunto de datos. Esto garantiza que las diferencias en el rango de valores no afecten el rendimiento de la red neuronal.

Además, se aplicaron transformaciones como rotación aleatoria de hasta 30 grados, volteo horizontal y vertical. Estas técnicas introducen variabilidad en los datos y ayudan a mitigar el sobreajuste, al aumentar la diversidad en el conjunto de entrenamiento.

Transformaciones de Imágenes para CNN

En el modelo basado en redes neuronales convolucionales (CNN), las imágenes se procesaron con un enfoque adicional para capturar características locales. La arquitectura incluye tres capas convolucionales con un aumento progresivo en el número de filtros (16, 32, 64) y un kernel de 3x3. Cada capa convolucional se combina con una función de activación ReLU y una operación de MaxPooling, que reduce la dimensionalidad manteniendo las características más relevantes.

Finalmente, se implementó una capa de Adaptive Global Pooling que reduce la salida a una representación fija (1x1), facilitando la conexión con la capa de salida. Esta capa utiliza la función de activación Tanh para normalizar las predicciones dentro del rango [-1, 1], consistente con los valores objetivo.

La tasa de aprendizaje se fijó en 0.0001, tras observar que valores mayores generaban fluctuaciones significativas en las métricas de entrenamiento. También se implementó early stopping para evitar sobreajustes, con las métricas de pérdida y validación almacenadas para un análisis posterior.

Hemos configurado el número de épocas en 100 y activado el early stopping con una paciencia de 10 épocas. La elección de una paciencia de 10 se debió a que, con un valor de paciencia de 5, el entrenamiento se detenía en la época 23, lo que parecía no ser el mínimo global, sino un punto prematuro del proceso de optimización. Al aumentar la paciencia a 10, se permitió que el modelo entrenara durante más épocas, dando más tiempo para encontrar una solución más óptima y evitar un paro prematuro en el que aún se podrían haber logrado mejoras adicionales.

Definición de la Red Neuronal FC

La red neuronal diseñada consta de tres capas totalmente conectadas, acompañadas de capas de normalización por lotes, funciones de activación ReLU y una capa de Dropout. Esta arquitectura fue elegida por su capacidad para aprender representaciones complejas en problemas de regresión.

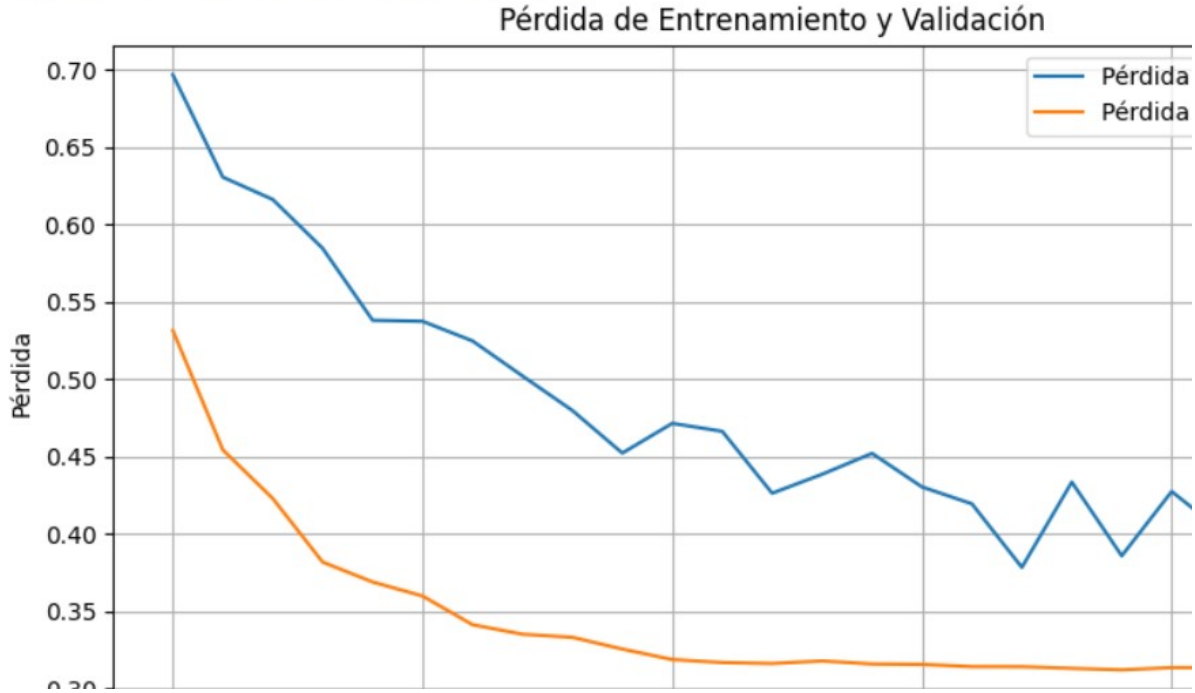
La normalización por lotes estabiliza el entrenamiento al reducir la sensibilidad a las inicializaciones de los parámetros, mientras que la función de activación ReLU introduce no

linealidad, permitiendo al modelo capturar relaciones complejas en los datos. Por último, la capa Dropout con un 50% de probabilidad evita el sobreajuste al desactivar aleatoriamente conexiones durante el entrenamiento.

Para optimizar el entrenamiento, se seleccionó la función de pérdida de error cuadrático medio (MSELoss), ideal para tareas de regresión. Como optimizador, se utilizó Adam, que combina eficiencia y adaptabilidad en la actualización de parámetros. El entrenamiento se realizó durante un máximo de 50 épocas, con una estrategia de early stopping que detuvo el proceso si no se observaban mejoras en la pérdida de validación durante cinco épocas consecutivas.

El mismo modelo con lr=0.0001 y función de activación Tanh en las capas, nos dio un resultado peor:

```
Época [25/50], Pérdida Entrenamiento: 0.4026, Pérdida Validación: 0.3167
Patience counter: 5/5
Deteniendo entrenamiento por Early Stopping...
```



Modelo de Ensamblado

Finalmente, se desarrolló un modelo de ensamblado que combina las salidas de los modelos CNN y de redes totalmente conectadas (FC). Esta arquitectura incluye una red totalmente conectada con dos capas intermedias (32 y 16 unidades) y una capa de salida con activación Tanh. La normalización por lotes y el Dropout (30%) se emplearon para estabilizar el entrenamiento y reducir el sobreajuste. Un dropout mayor nos daba peores resultados.

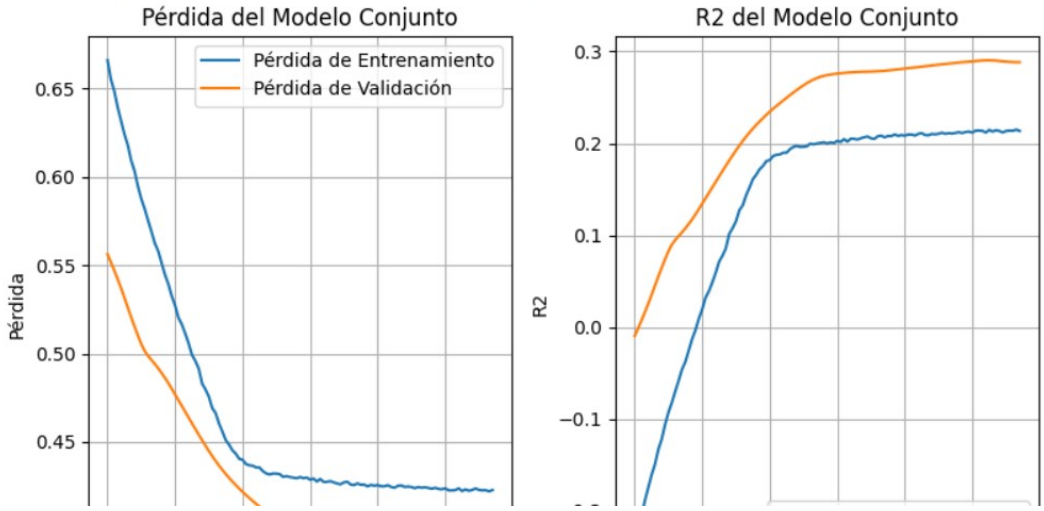
El modelo tiene 150 épocas ya que varias veces durante el entrenamiento con 100 épocas nos quedamos cortos.

Se eligió nuevamente la función de pérdida MSELoss, junto con Adam como optimizador, estableciendo la tasa de aprendizaje en 0.001 tras probar con diferentes valores. Como

métrica de evaluación secundaria, se utilizó el coeficiente de determinación (R^2), evaluando la calidad del ajuste del modelo.

Intentamos ajustar los hiperparamentos y el modelo, con una arquitectura de dos capas intermedias (16 y 8 unidades) recibimos resultados parecidos.

```
Modelo guardado después de la época 113.
Época 113/1000, Pérdida Entrenamiento: 0.4227, Pérdida Validación: 0.3919, R2 Entrenamiento: 0.2137, R2 Validació
Modelo guardado después de la época 114.
Época 114/1000, Pérdida Entrenamiento: 0.4220, Pérdida Validación: 0.3920, R2 Entrenamiento: 0.2149, R2 Validació
Early stopping activado en la época 115. No hubo mejora en la pérdida de validación durante 10 épocas.
```



Modelo Preentrenado ResNet-50

Tras la implementación inicial de nuestro modelo decidimos explorar arquitecturas más avanzadas para abordar las limitaciones del modelo inicial. Para ello, integramos un modelo preentrenado basado en **Transformers**.

Para aprovechar un modelo avanzado, se utilizó ResNet-50 preentrenado, adaptándolo para tareas de regresión.

Las imágenes se redimensionaron a 224x224 píxeles y se normalizaron con los valores utilizados en el preentrenamiento de ResNet-50 (mean=[0.485, 0.456, 0.406],

std=[0.229, 0.224, 0.225]). Las capas preentrenadas de ResNet-50 se congelaron, entrenando únicamente la capa final, que fue modificada para predecir valores continuos.

El optimizador Adam y la función de pérdida MSELoss fueron utilizados nuevamente, mientras que el error absoluto medio (MAE) se empleó como métrica secundaria para evaluar el rendimiento. También se implementó early stopping**,** asegurando un entrenamiento eficiente y evitando el sobreajuste.

La incorporación de Transformers en nuestro pipeline no solo permitió una mejora significativa en el rendimiento, sino que también resaltó la capacidad de esta arquitectura para capturar patrones complejos y generalizar mejor en comparación con los modelos tradicionales.

Conclusión

En este trabajo, comparamos el rendimiento de dos enfoques distintos para abordar el problema planteado: un modelo híbrido basado en una red completamente conectada (FC+CNN) y un modelo más avanzado basado en Transformers preentrenados.

La red basada en Transformers mostró un rendimiento superior en cuanto a precisión, con un valor de 0.4354 en el conjunto de prueba, a pesar de presentar una pérdida de prueba relativamente baja, 0.3028. Este comportamiento sugiere que la arquitectura de Transformers, fue más eficaz para abordar la tarea en cuestión, logrando una mejor generalización en los datos de prueba.

Por otro lado, la FC+CNN mostró una mayor pérdida en el conjunto de prueba (0.3235) **y** un R^2 de 0.3887**,** lo que indica un desempeño inferior en comparación con los Transformers.

En general, los resultados indican que los Transformers ofrecen una ventaja significativa en términos de precisión y capacidad predictiva. De esta manera, la elección del modelo dependerá del balance entre los recursos disponibles y el nivel de precisión requerido para la tarea específica.