TP2: Críticas Cinematográficas - Grupo 02

Introducción

En el presente trabajo nos enfocamos en evaluar y desarrollar varios modelos de clasificación para procesamiento de lenguaje natural utilizando una colección de críticas cinematográficas en español. Nuestro objetivo es predecir el sentimiento de las críticas, categorizándolas como positivas o negativas.

Para ello, utilizamos dos tipos de conjuntos de datos, proporcionados por la cátedra: el conjunto de entrenamiento (train) y el conjunto de prueba (test). El conjunto de entrenamiento cuenta con las siguientes columnas:

- ID: Identificador de la crítica
- Review_es: Texto de la crítica en español
- Sentimiento: Sentimiento asociado (positivo o negativo)

El conjunto de prueba, por su parte, contiene las siguientes columnas:

- ID: Identificador de la crítica
- Review_es: Texto de la crítica en español

Para implementar estos enfoques, probamos los siguientes modelos: Bayes Naïve, Random Forest, XGBoost y una red neuronal utilizando Keras y TensorFlow. Adicionalmente confeccionamos un ensamble de los tres primeros. Para la mayoría de los modelos, utilizamos técnicas de normalización y vectorización de manera general, aplicando cada modelo con un tipo de normalización y un tipo de vectorización diferente entre las siguientes.

A su vez se empleó validación cruzada para buscar los hiperparametros que más incrementan la performance de cada uno. Luego cada sub-modelo

optimizado fue probado con su testing set (ya que se dividió el training set en dos nuevos sets de training y testing)

Finalmente, seleccionamos el mejor sub-modelo y lo reentrenamos con todo el training set, para luego predecir los sentimientos del set de testeo usado en la competencia de kaggle.

Pruebas realizadas

Realizamos múltiples pruebas que no fueron reflejadas en los modelos finales cuyos resultados fueron subidos a kaggle.

En general, se probaron diferentes setups para la búsqueda de los mejores hiperparámetros (sobretodo n estimators, learning rate, max depth) y probando todas las versiones del dataset que tenemos disponibles (con las diferentes técnicas de preprocesamiento), nos quedamos con aquellas que nos dieran los mejores resultados.

Particularmente, en la construcción del XGBoost los mejores resultados se obtuvieron sin hiperparametros modificados, por lo que descartamos esos en la versión final.

Por otro lado, también se realizaron pruebas de todos los modelos con la versión de sentimientos limpios, la cuál tiene aplicados métodos de reducción de palabras y busca un dataset más limitado, sin datos que generen ruido. Sin embargo, los resultados fueron muy similares sin importar si la entrada del modelo era ésta o la original.

Por último, se realizaron pruebas alternativas con un modelo de red neuronal completo, con un 25% de las palabras, que en nuestras métricas no tenían malos resultados (F1 Score y similares de aproximadamente 0.86) pero que en Kaggle tuvo muy malos resultados, en todas sus versiones, por lo que estos modelos fueron descartados.

Cuadro de Resultados

Medidas de rendimiento en el conjunto de TEST:

Modelo	F1-Test	Precision Test	Recall Test	Accuracy	Kaggle
Bayes Naive	0.87	0.87	0.87	0.87	0.71351
Random Forest	0.81	0.81	0.81	0.81	0.71040
XgBoost	0.83	0.83	0.83	0.83	0.71195
Red Neuronal	0.86	0.86	0.86	0.86	0.68734
Ensamble	0.92	0.92	0.92	0.92	0.73328

Descripción de Modelos

Preprocesamiento y Análisis Exploratorio

Realizamos varias etapas de preprocesamiento para transformar el texto en una representación adecuada para el análisis:

- **Tokenización:** Dividimos el texto en palabras individuales (tokens), utilizando nltk, lo que facilita el procesamiento posterior.
- **Eliminación de Stopwords:** Removemos palabras comunes y sin significado específico (como "y", "el", "en") que no aportan significado significativo al análisis.
- **Etiquetado de Partes del Discurso (POS tagging):** Asignamos etiquetas gramaticales a cada palabra para capturar información contextual adicional, usando herramientas de nltk y spacy.

- **Lematización:** Transformamos las palabras a su forma base (lemas) para reducir la dimensionalidad del texto y mejorar la consistencia. Esto es crucial para manejar variaciones de palabras (por ejemplo, "corriendo" se transforma en "correr").

A su vez, para adaptar el dataset a los modelos a utilizar, realizamos:

- **División de Datos:** Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba para poder realizar pequeñas predicciones con nuestros modelos y evaluar cuál era el más adecuado para la competencia.
- **Conjunto sin reseñas en inglés:** Un pequeño porcentaje de las reseñas se encontraba en un idioma diferente, por lo que se realizó una eliminación de las mismas, y se realizaron pruebas con y sin estas reseñas.
- **Extracción de Características:** Transformamos los datos de texto en una representación numérica adecuada utilizando TF-IDF (Term Frequency-Inverse Document Frequency), una técnica que asigna una importancia relativa a las palabras en función de su frecuencia en el documento y en el corpus completo.
- **Conjunto “sentimientos limpios”:** Se redujo la cantidad de palabras utilizadas al 25% del total para evitar el sobreajuste, pero que debe ser realizado con precaución ya que podría comprometer la efectividad del modelo en situaciones prácticas.

Ahondando en el último ítem, dataset “sentimientos limpios” tiene las siguientes columnas:

- ID: Identificador de la crítica
- Review_es: Texto de la crítica en español
- Sentimiento: Sentimiento asociado (positivo o negativo)
- Review_norm: Crítica reducida y normalizada

Para generar este dataset, aplicamos varios pasos de preprocesamiento. Primero, lematizamos el texto para conservar la forma base de las palabras.

Luego, limpiamos el texto eliminando caracteres especiales, caracteres no alfabéticos, diacríticos y palabras de un solo carácter. Finalmente, eliminamos las stop words.

Evaluación de los modelos

Evaluamos el rendimiento del modelo en el conjunto de prueba generado desde el conjunto train entregado por la cátedra, con una matriz de confusión y midiendo precisión, f1 score, recall y accuracy.
Se entrenó el modelo y se evaluó con diferentes conjuntos de datos, incluyendo los ya descritos en la sección anterior y los siguientes:

- **Bag of Words (BoW) con tokenización sin lematización**
- **Bag of Words (BoW) con tokenización con lematización**
- **N-gramas con tokenización con lematización**
- **N-gramas con tokenización sin lematización**
- **TF-IDF con tokenización sin lematización**
- **TF-IDF con tokenización con lematización**

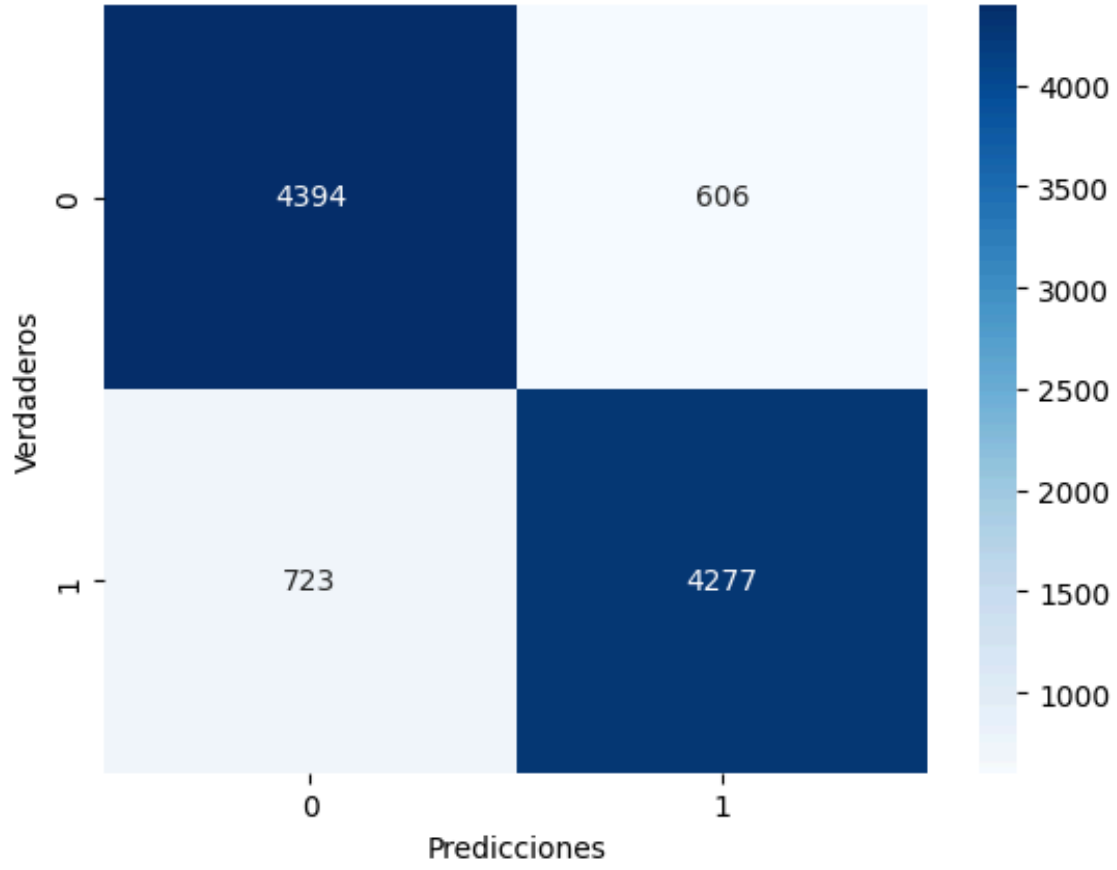
Cada configuración de datos se evaluó en un conjunto de validación y se compararon las matrices de confusión resultantes, por consiguiente, al ver como performa el mismo modelo con diferentes normalizaciones y vectorización decidimos cuál modelo era más adecuado para presentar en el kaggle.

Modelo 1: Bayes Naïve

Empezamos el trabajo con Bayes Naïve que es un modelo de clasificación basado en el teorema de Bayes, el cual asume independencia entre las características del conjunto de datos. Su eficiencia y precisión lo hacen adecuado para manejar grandes volúmenes de datos textuales. Utiliza la probabilidad condicional para predecir la categoría a la que pertenece un elemento dado, siendo especialmente útil en aplicaciones donde se requiere una rápida clasificación.

Para cada configuración, se ajustó un modelo Naive Bayes utilizando *MultinomialNB* con un parámetro de suavizado (alpha=1).

	precision	recall	f1-score	support
negativo	0.86	0.88	0.87	5000
positivo	0.88	0.86	0.87	5000
accuracy			0.87	10000
macro avg	0.87	0.87	0.87	10000
weighted avg	0.87	0.87	0.87	10000



Tras comparar el desempeño de cada modelo en el conjunto de validación, se identificó que el modelo de Naive Bayes utilizando N-gramas con tokenización sin lemas alcanzó el puntaje F1 más alto, registrando un valor de

0.87. Consideramos que es posible que haya sido ligeramente más efectivo al usar tokenización sin lemas por lograr un mayor equilibrio entre precisión y generalización. Bayes Naive fue un modelo fácil de implementar, comparado con la precisión de sus resultados.

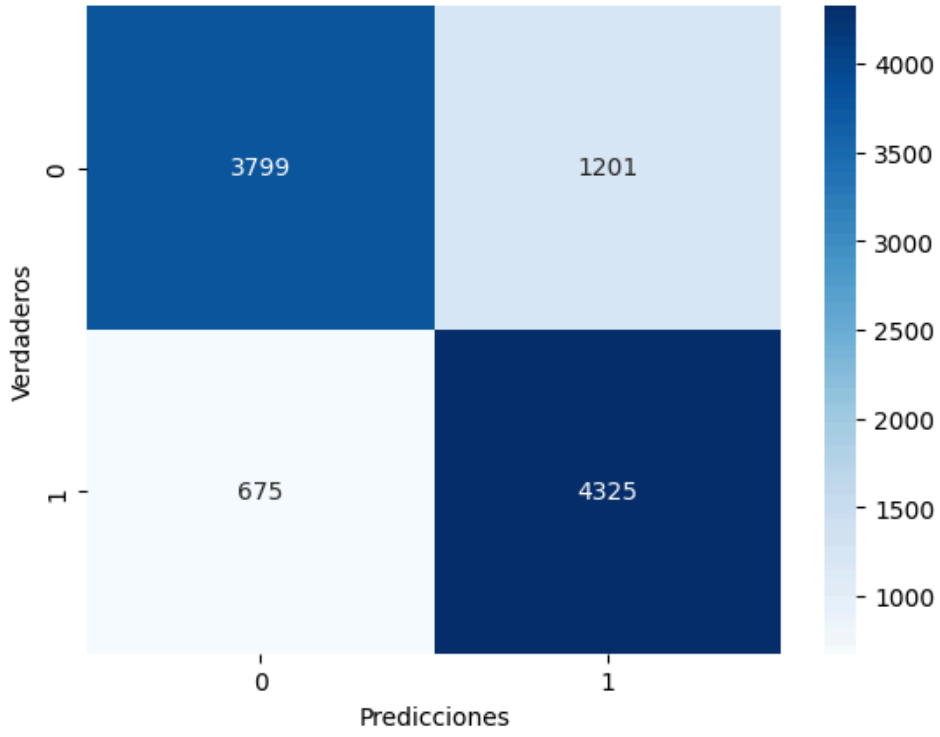
Modelo 2: Random Forest

Se siguió con Random Forest, el cual es un modelo de aprendizaje automático que pertenece a la familia de algoritmos de ensamble. Funciona construyendo múltiples árboles de decisión durante el entrenamiento y combinando sus predicciones.

Se comenzó entrenando un modelo de Random Forest sin hiperparámetros específicos, utilizando un conjunto de datos preprocesados con tokenización de N-gramas sin lematización. Luego de evaluar su rendimiento con la ayuda de una matriz de confusión, se procedió a buscar mejorar el rendimiento del modelo. Para ésto se realizó una búsqueda de hiperparámetros utilizando validación cruzada con K-fold (k=5). Se exploraron varios parámetros como la profundidad máxima del árbol, el criterio de división, el alpha de complejidad de poda, el número mínimo de muestras por división y el número de estimadores en el bosque. El mejor conjunto de hiperparámetros encontrado mediante búsqueda aleatoria se utilizó para refinar el modelo. Estos fueron los siguientes:

```
{'n_estimators': 80,
 'min_samples_split': 35,
 'max_depth': 17,
 'criterion': 'gini',
 'ccp_alpha': 0.0}
```

	precision	recall	f1-score	support
negativo	0.85	0.76	0.80	5000
positivo	0.78	0.86	0.82	5000
accuracy			0.81	10000
macro avg	0.82	0.81	0.81	10000
weighted avg	0.82	0.81	0.81	10000



Basados en éstas métricas, el modelo mejor performante es el que utiliza Tf-idf con tokenización sin lemmas, ya que obtuvo un F1 Score de 0.81.

Modelo 3: XG Boost

En tercer lugar abordamos XGBoost, una biblioteca de machine learning optimizada para implementar algoritmos de boosting basados en árboles de decisión conocido por su eficiencia y precisión.

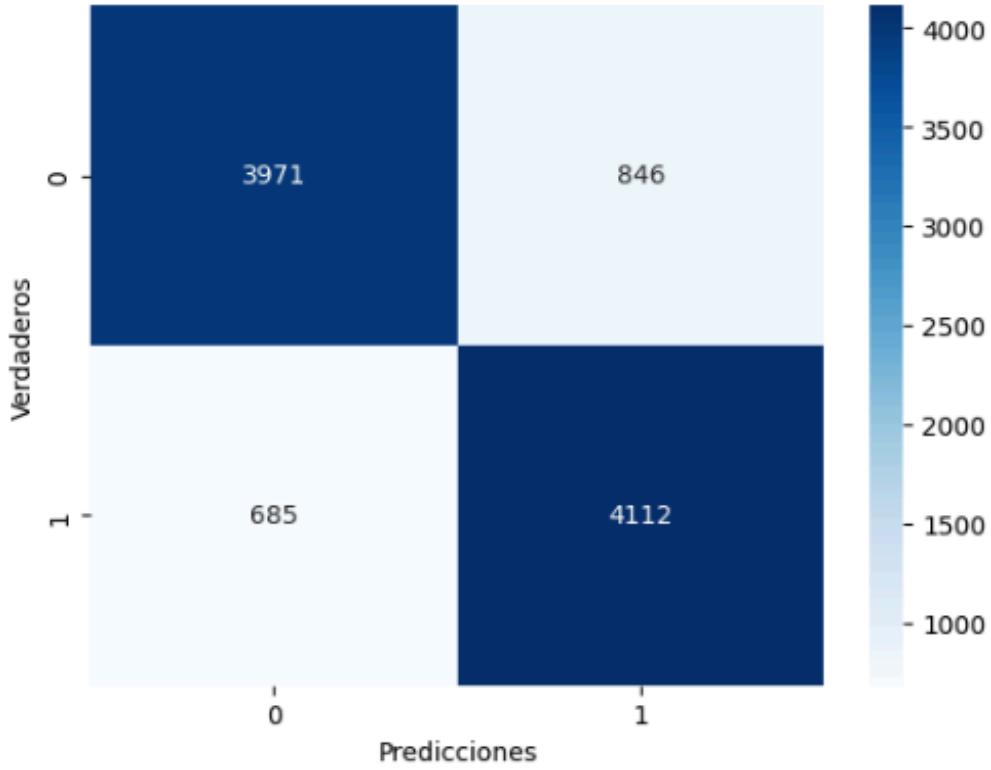
Inicialmente, se realizó la codificación de las etiquetas de las clases utilizando LabelEncoder para poder utilizar el modelo XGBoost.

Se llevó a cabo una búsqueda de hiperparámetros utilizando validación cruzada estratificada (k=5). Se exploraron diversas combinaciones de hiperparámetros incluyendo el número de estimadores, la tasa de aprendizaje, la profundidad máxima del árbol, la proporción de muestras utilizadas para entrenar cada árbol, y parámetros de regularización como gamma, lambda y alpha. La mejor combinación de hiperparámetros se seleccionó basándose en el puntaje F1 promedio obtenido durante la validación cruzada, resultando en:

```
{'subsample': 0.8157894736842105,
 'n_estimators': 90,
 'max_depth': 6,
 'learning_rate': 0.42653061224489797,
 'lambda': 1,
 'gamma': 0,
 'alpha': 0}
```

Después de comparar los resultados obtenidos en cada configuración, se identificó que la mayoría de las variantes alcanzaron un puntaje F1 de 0.84. A su vez, tampoco se vieron diferencias ante la utilización de los mejores hiperparametros o no. Este valor sugiere que el modelo de XGBoost es consistente en su rendimiento independientemente de la técnica de vectorización y preprocesamiento utilizada.

	precision	recall	f1-score	support
negativo	0.85	0.82	0.84	4817
positivo	0.83	0.86	0.84	4797
accuracy			0.84	9614
macro avg	0.84	0.84	0.84	9614
weighted avg	0.84	0.84	0.84	9614



Modelo 4: Red Neuronal

En nuestro proyecto de construcción de una red neuronal decidimos trabajar con el conjunto de datos "sentimientos_limpios".

Para empezar, transformamos cada crítica en una secuencia numérica, asignando números únicos a cada palabra y limitando nuestro vocabulario a las 12,500 palabras más comunes en las críticas. A continuación, estandarizamos la longitud de cada secuencia a 500 números, ya sea añadiendo ceros al final de las críticas más cortas o recortando las más largas, garantizando así que todas las entradas fueran uniformes y manejables por la red neuronal.

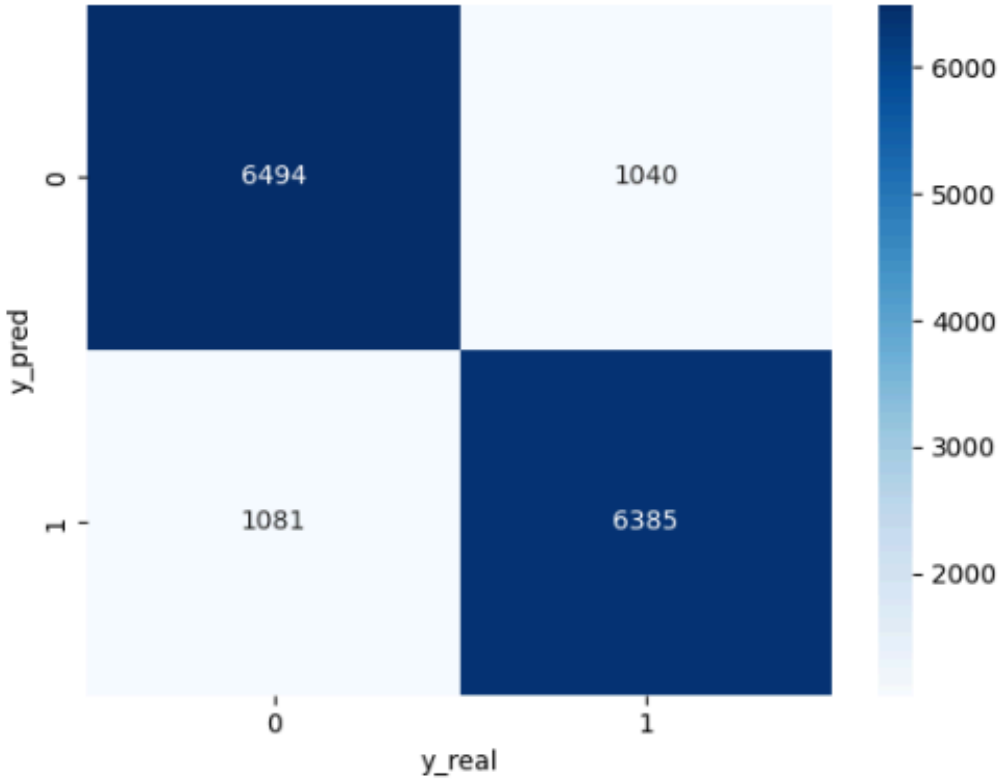
La arquitectura de nuestra red neuronal fue diseñada específicamente para procesar secuencias de texto. Empezamos con una capa de embedding que transforma las palabras en vectores densos, capturando así las relaciones semánticas entre ellas. Luego, implementamos una capa de convolución (Conv1D) para detectar patrones locales, seguida de dos capas de redes neuronales recurrentes (GRU) para procesar secuencias largas de manera efectiva.

Para evitar el sobreajuste durante el entrenamiento, introdujimos una capa de Dropout que desactiva aleatoriamente unidades de la red, promoviendo la generalización y robustez del modelo. Finalmente, añadimos una capa densa con activación sigmoide para generar la salida final, que indica la probabilidad de que una crítica sea positiva o negativa.

Compilamos nuestro modelo utilizando el optimizador Adam y la función de pérdida de entropía binaria, ideales para problemas de clasificación binaria como el nuestro. Evaluamos el desempeño del modelo utilizando el Área Bajo la Curva (AUC), una métrica que equilibra la sensibilidad y la especificidad del modelo, proporcionando una medida completa de su capacidad predictiva.

Implementamos también la técnica de early stopping para detener el entrenamiento una vez que el modelo alcanza una precisión deseada, evitando así el sobreentrenamiento y optimizando el tiempo de entrenamiento.

	precision	recall	f1-score	support
0	0.86	0.86	0.86	7534
1	0.86	0.86	0.86	7466
accuracy			0.86	15000
macro avg	0.86	0.86	0.86	15000
weighted avg	0.86	0.86	0.86	15000



Modelo 5: Ensamble

Por último en el apartado de Ensamble nos centramos en predecir el sentimiento de las críticas cinematográficas utilizando una combinación de

modelos de aprendizaje automático y técnicas avanzadas de procesamiento de texto, basándonos en dos enfoques de **ensamble.**

Entrenamos varios modelos de clasificación para predecir el sentimiento de las críticas que usamos en anteriores etapas del tp2:

- Naive Bayes: Un modelo probabilístico basado en el teorema de Bayes, adecuado para clasificación de texto.
- Random Forest: Un modelo de ensamble basado en árboles de decisión que mejora la precisión y reduce el sobreajuste.
- XGBoost: Un modelo de gradient boosting que optimiza el rendimiento utilizando técnicas de regularización.

Para mejorar el rendimiento de nuestras predicciones, implementamos dos enfoques de **ensamble**. Ensamble de Votación, que lo que hicimos fue que Combinamos las predicciones de Naive Bayes, Random Forest y XGBoost mediante dos técnicas:

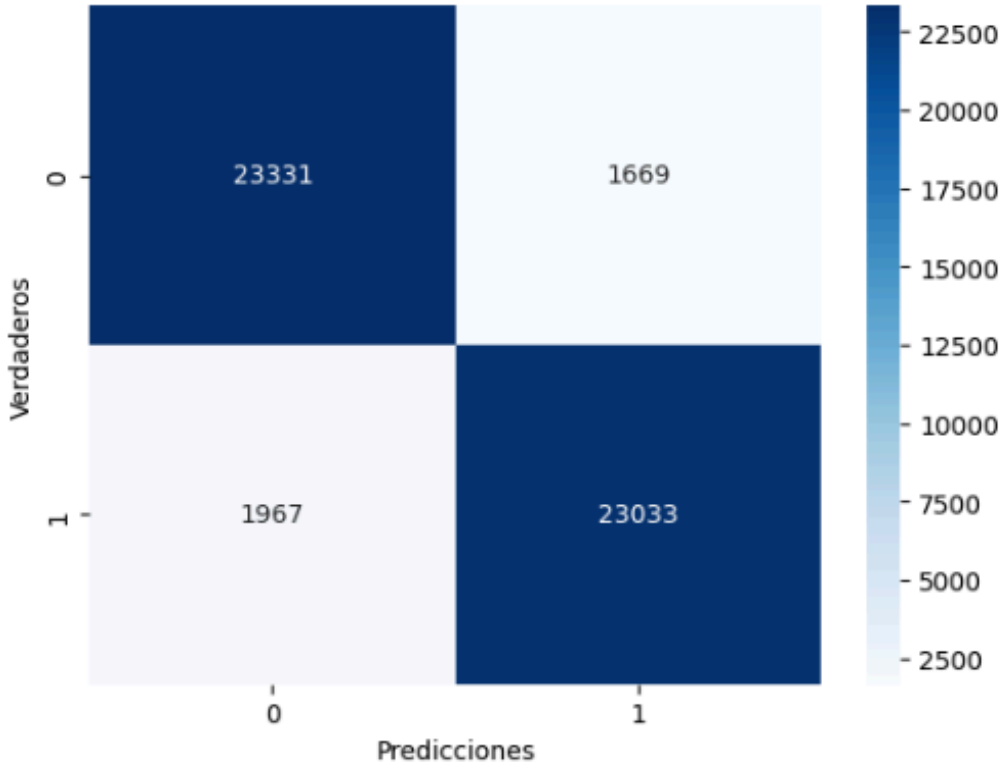
- Votación Hard: Cada modelo emite un voto y la clase con más votos es la predicción final.
- Votación Soft: Promediamos las probabilidades de las predicciones de cada modelo para obtener la predicción final.

Utilizamos un **Stacking** como StackingClassifier para combinar múltiples modelos de base (Naive Bayes, Random Forest y XGBoost) y un modelo de meta (regresión logística). Este enfoque entrena los modelos de base y utiliza sus predicciones como características para el modelo de meta, mejorando así la capacidad de generalización.

Los Conceptos Clave que utilizamos para poder desarrollar este apartado del tp con Ensamble:

- Validación Cruzada: Implementamos una validación cruzada de 5 pliegues (5-fold cross-validation) durante el stacking para asegurar una evaluación robusta.
- Ensemble Learning: Combina múltiples modelos para mejorar el rendimiento predictivo en comparación con los modelos individuales.
- Stacking: Una técnica de ensemble que utiliza modelos de base para generar predicciones que luego se utilizan como características para un modelo de meta.

	precision	recall	f1-score	support
negativo	0.92	0.93	0.93	25000
positivo	0.93	0.92	0.93	25000
accuracy			0.93	50000
macro avg	0.93	0.93	0.93	50000
weighted avg	0.93	0.93	0.93	50000



Conclusiones generales

En este trabajo práctico se realizaron diversas etapas de análisis, preprocesamiento y modelado de datos, cuyo objetivo fue predecir ciertas variables de interés a partir de un conjunto de datos inicial. A continuación, se presentan las conclusiones generales obtenidas a lo largo del proyecto:

Análisis Exploratorio de los Datos

Si bien fue fundamental realizar el análisis exploratorio de datos para entender mejor la estructura del dataset, identificar valores que puedan ser problemáticos (por ejemplo, reseñas en inglés, stopwords), muchas veces los modelos entrenados en sin hiperparametros obtuvieron mejores resultados. Aún así, consideramos importante que se trabajara con datos más limpios y relevantes, lo cual se reflejó en mejores métricas de evaluación.

Desempeño de los Modelos

El modelo que obtuvo el mejor desempeño en el subconjunto de TEST fue el Modelo Ensamble, el cual utilizó Bayes Naive, XGBoost y Random Forest. Este modelo superó a los demás en todas las métricas.
En Kaggle, el modelo con mejor desempeño también fue el Modelo Ensamble.

Rapidez de Entrenamiento

El modelo más sencillo y rápido de entrenar fue el Modelo Bayes Naive y el XGBoost (versión sin hiperparametros). A pesar de su simplicidad y velocidad, ambos mostraron un desempeño relativamente bueno en comparación con modelos más complejos, demostrando que a veces la simplicidad puede ser ventajosa en términos de recursos computacionales y tiempo.

Uso Productivo del Modelo

Consideramos que el Modelo de Ensamble (el de mejor desempeño) sería el más útil de forma productiva, por sus niveles de precisión que lo hacen más adecuado para aplicaciones prácticas. No obstante, se debe considerar la complejidad del modelo y los recursos necesarios para su despliegue. En ese sentido, el Modelo Bayes Naive (más rápido) podría ser considerado, sobretodo teniendo en cuenta que sus resultados no están tan alejados del mejor modelo.

Opciones No Exploradas

Debido a limitaciones de tiempo y recursos, algunas opciones quedaron fuera del alcance de este trabajo:

1. *Optimización de Hiperparámetros*

Se podrían haber implementado técnicas más avanzadas de optimización de hiperparámetros, como búsqueda bayesiana o algoritmos genéticos (nos hubiera gustado investigar y leer al respecto de otros modelos por nuestra cuenta), para encontrar configuraciones aún mejores.

2. Modelos Adicionales

Hubiese sido interesante probar otros algoritmos de aprendizaje automático, como redes neuronales profundas, que podrían captar patrones más complejos en los datos.

3. *Ensamble*

Si bien se exploró modelos de stacking, podría en el futuro probarse con modelos de blending, los cuales podrían mejorar aún más el desempeño general. A su vez, no nos quedó tiempo para probar todas las diferentes combinaciones de ensamble con 2 o más modelos que hubiéramos querido.

4. *Feature Engineering Avanzado*

Profundizar nuestros conocimientos sobre engineering avanzado podría ayudarnos a mejorar nuestros modelos. Al intentar mejorar los modelos nos dimos cuenta que obteniamos mejores resultados con aquellos que podíamos importar con scikit-learn.

Tareas Realizadas

Integrante	Tarea	Prom. Hs Semana
Aramayo Carolina	Preprocesamiento de datasets Lematización Tokenización Red Neuronal Informe	8hs
Utrera Maximo Damian	Ensamble Random Forest Bayes Naive Pruebas todos los modelos Informe	9hs
Villalba Ana Daniela	Preprocesamiento XGBoost Informe	8hs
Fiorilo Roy	Tokenización:N-gramas Red Neuronal Informe	1hs