Trabajo Práctico 2 : Críticas cinematográficas

Fecha de entrega: 29/06/2023

Preprocesamiento

Se analizaron las reseñas mediante una detección de idioma y se eliminaron 1817 registros que resultaron estar en inglés.

Vectorización

Se realizaron dos predicciones con vectorizadores distintos: TfidfVectorizer y CountVectorizer.

	TfidfVectorizer	CountVectorizer
F1 Score	0.858	0.850
Accuracy	0.855	0.848

Dado que el TfidfVectorizer dió mejores resultados, es el que se decidió utilizar en lo sucesivo

Así mismo, se configuró el vectorizador con **stopwords** en español, se removieron **acentos** y se limitó la **cantidad de palabras** de 127.614 a 40.000. De esta forma se utilizan las de mayor frecuencia para evitar el **overfitting**.

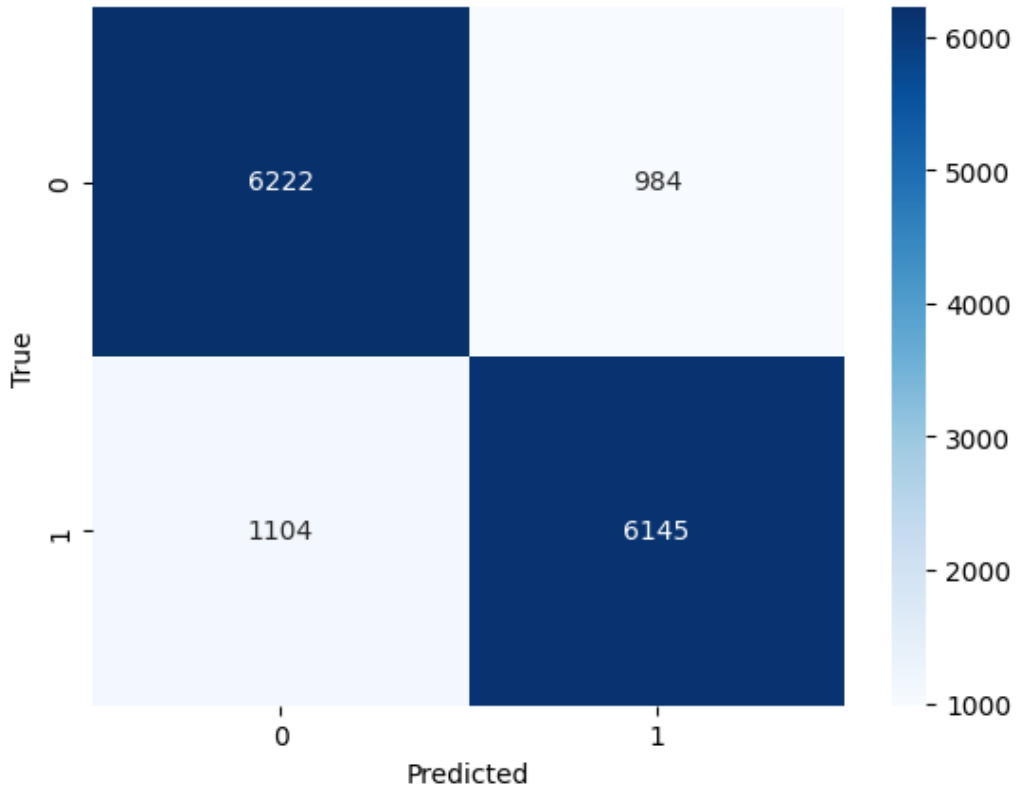
Esta reducción además redujo considerablemente el **tiempo de entrenamiento** y tamaño de los modelos, llevándolos aproximadamente a la mitad.

Bayes Naïve

Se utilizó el clasificador MultinomialNB.
Luego de optimizar hiperparámetros con RandomSearch cross validation, se definieron:

- alpha: 1 (default)
- forced_alpha: True
- class_prior: None (default)
- fit_prior: False

	precision	recall	f1-score
0	0.849	0.863	0.856
1	0.862	0.848	0.855
accuracy			0.856



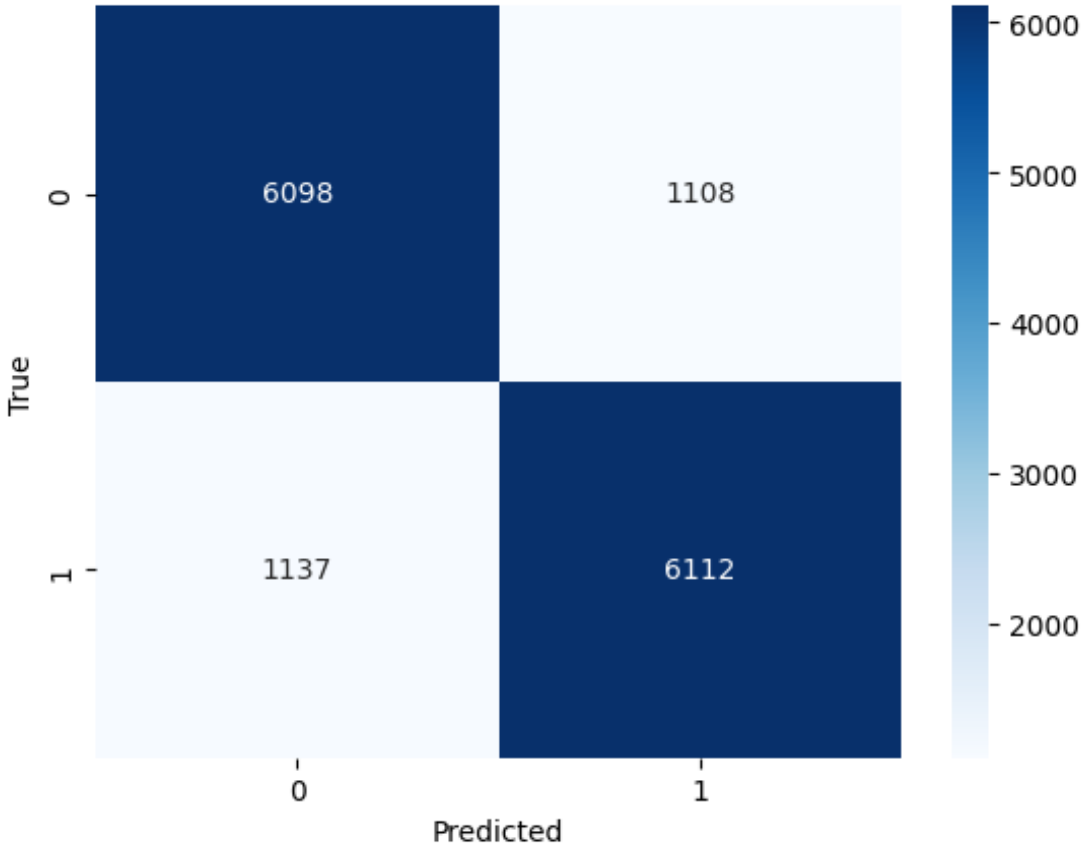
Score en Kaggle: 0.72727
Nombre modelo: bayes_naive_best_vectorizer4.sav

Random Forest

Luego de entrenar el modelo y buscar los mejores hiperparámetros, se seleccionaron los siguientes:

- Criterion: entropy
- min_samples_leaf: 1 (default)
- min_samples_split: 16
- n_estimators: 100 (default)

	precision	recall	f1-score
negativo	0.843	0.846	0.845
positivo	0.847	0.843	0.845
accuracy			0.845



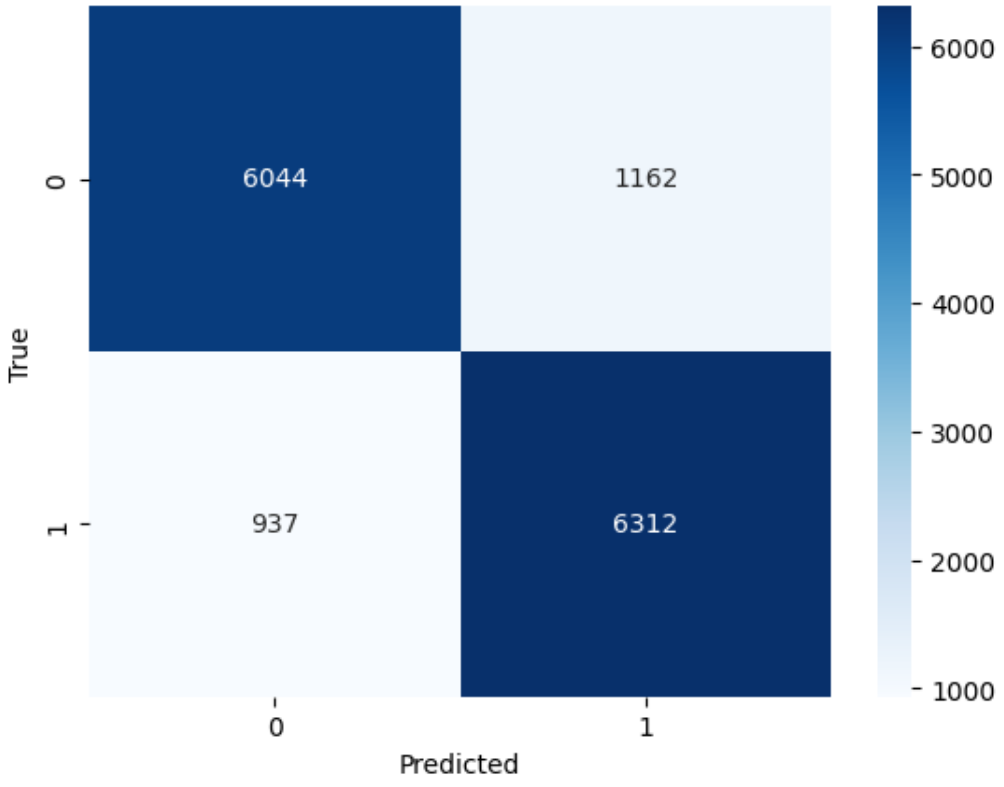
Score en Kaggle: 0.70595
Nombre modelo: random_forest_best-filtrado.sav

XGBoost

Luego de entrenar el modelo y buscar los mejores hiperparámetros, se seleccionaron los siguientes:

- Max_depth: 9
- Learning_rate: 0.1
- n_estimators: 300
- Subsample: 0.6
- Gamma: 0.1

	precision	recall	f1-score
0	0.866	0.839	0.852
1	0.845	0.871	0.857
accuracy			0.855



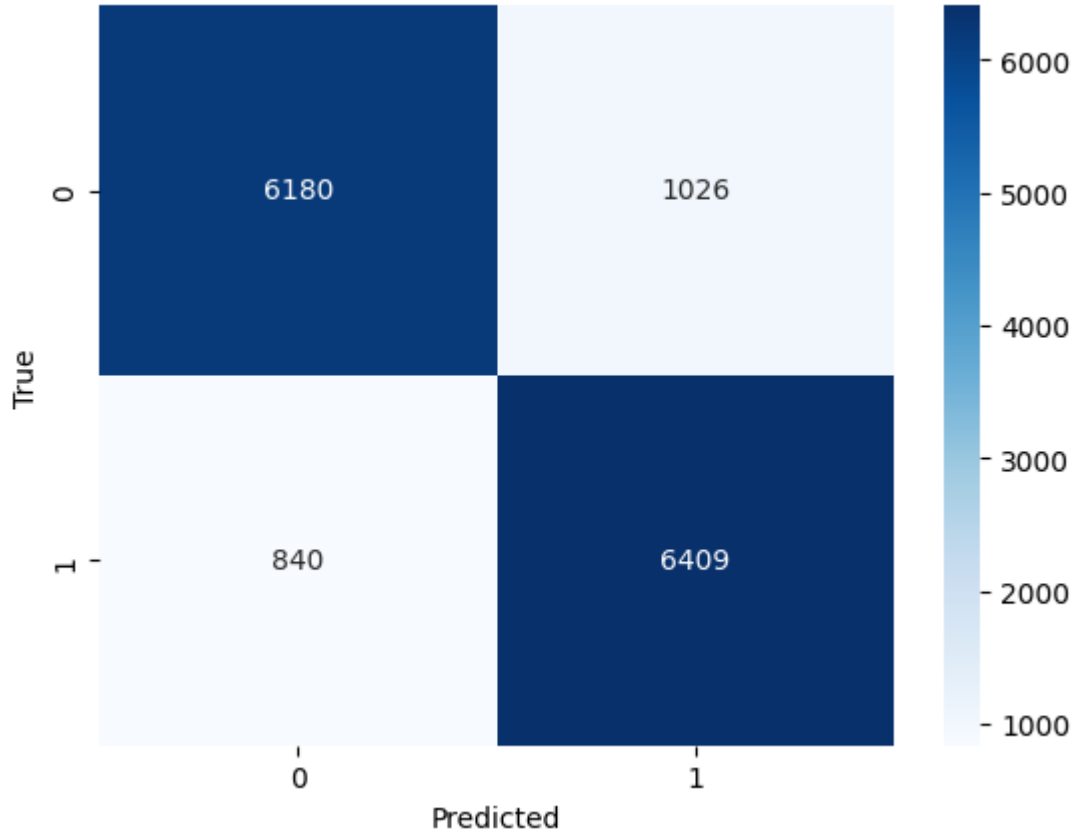
Score en Kaggle: 0.70943
Nombre modelo: xgboost_best-filtrado-2.sav

Ensamble Voting

Se realizó el ensamble con los 3 modelos entrenados anteriormente.
Se optimizaron los hiperparámetros y se obtuvo:

- voting: soft
- flatten_transaform: True

	precision	recall	f1-score
0	0.880	0.858	0.869
1	0.862	0.884	0.873
accuracy			0.871



Score en Kaggle: 0.7261
Nombre modelo: voting_hiperparametros.sav

Red Neuronal

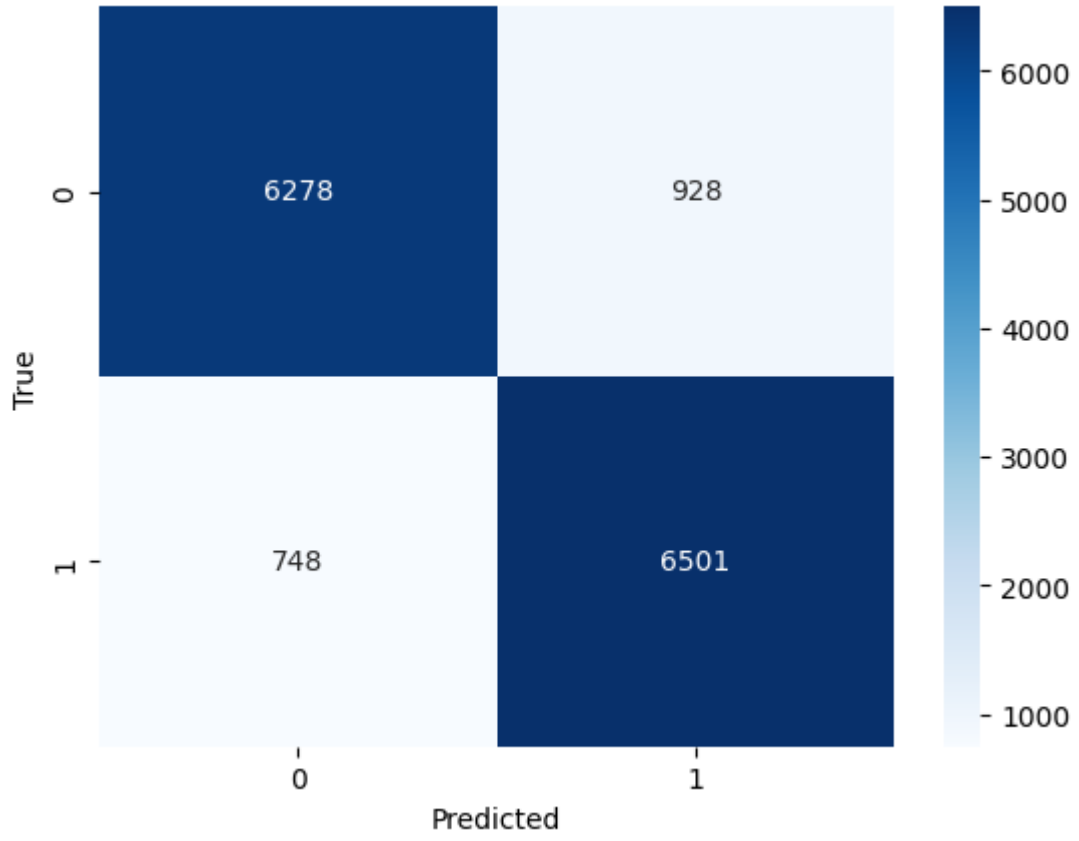
Arquitectura:

Inicialmente se armó una red neuronal con:

- Capa de entrada que recibe string
- Capa de vectorización
- Capa intermedia de 100 neuronas con función de activación relu
- Capa de salida de 1 neurona con función de activación sigmoid para categorizar de forma binaria

- Epochs: 5
- Batch size: 500
- Optimizer: Adam, learning_rate: 0.001

	precision	recall	f1-score
0	0.894	0.871	0.882
1	0.875	0.897	0.886
accuracy			0.884



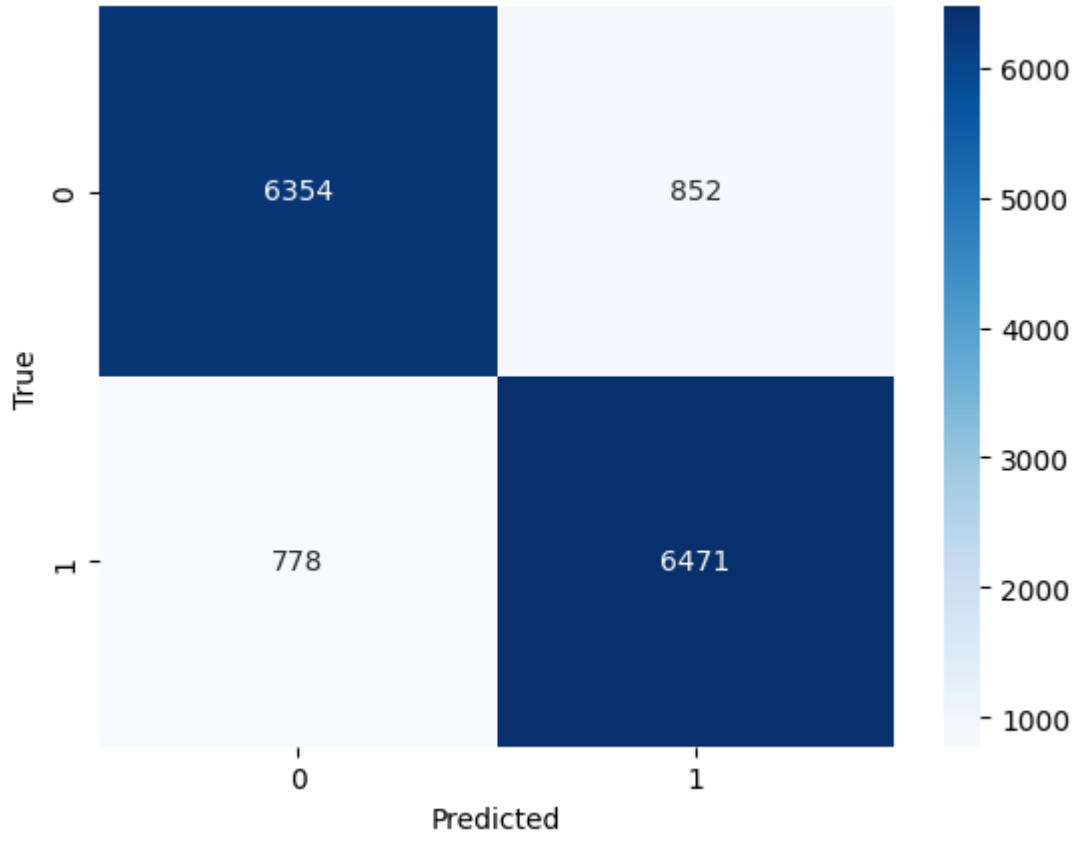
Score en Kaggle: 0.70943

Nombre modelo: red1.sav

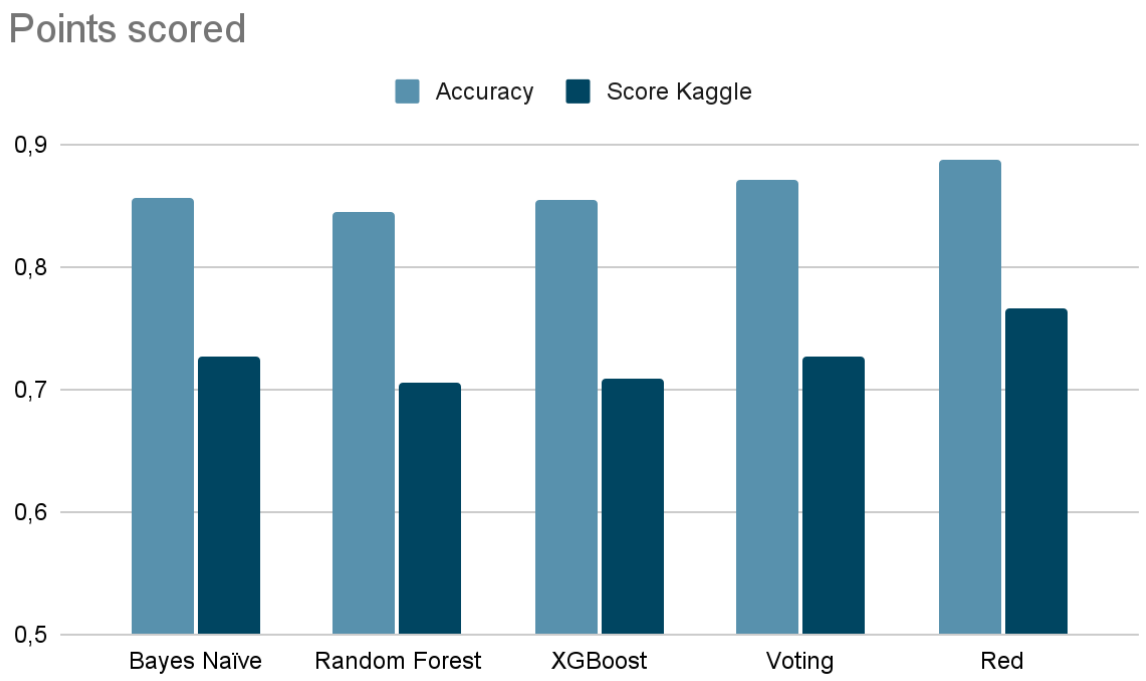
Luego se optimizaron los valores de epoch y batch_size y se obtuvo:

- Epochs: 1
- Batch size: 500

	precision	recall	f1-score
0	0.891	0.882	0.886
1	0.884	0.893	0.888
accuracy			0.887



Score en Kaggle: 0.76662
Nombre modelo: best_red.sav



Para mejorar:

Sin dudas profundizaría en el preprocesamiento del texto, para obtener una fuente con menor ruido y más reducida, para optimizar el tiempo de entrenamiento.

La red neuronal tiene mucho potencial para mejorar la predicción. En este caso se optimizaron únicamente los epochs y batch_size. Se podría optimizar también la cantidad de capas intermedias, su cantidad de neuronas y el learning_rate. (por falta de tiempo no se realizó).

Conclusiones

Luego de realizar el análisis de sentimiento entrenando distintos modelos, destaco el rendimiento del ensamble voting, que basándose en 3 modelos previamente entrenados, logró una mejora en el accuracy de aprox 1,5%

Más destacable aún fue la performance de la red neuronal, la cual alcanzó un accuracy de 0.887, mejorando la predicción sustancialmente al lograr un score de 0.766 en Kaggle.