# TP2: Críticas Cinematográficas - Grupo 28

## Introducción

En este trabajo práctico, tenemos la misión de predecir si una crítica de películas en español será positiva o negativa. Para ello, se crearon modelos de clasificación que son capaces de analizar una porción de texto y detectar el sentimiento.

Antes de entrenar cada modelo, se realizó un preprocesamiento del dataset train.
Este dataset provisto por la cátedra consta de tres columnas:

- ID
- Review_es: son las críticas en español
- Sentimiento: es nuestra columna target y es la cual queremos predecir. Toma valores 'positivo' o 'negativo'.

Aparte el dataset consta de 50000 registros

Por otro lado, también tenemos un dataset test y que solo cuenta con las columnas de ID y review_es. La cantidad de registro que tiene es 8599.

Luego como preprocesamiento, se transformó la columna de sentimiento a valores numéricos. La categoría 'positivo' lo transformamos en 1 y la categoría 'negativo' en 0.
Aparte, utilizamos el método de bag of words para realizar el preprocesamiento de las críticas y así reducir el tamaño del conjunto de datos. Filtramos las stopwords usando la librería de NLTK, transformamos todo el texto en minúscula, eliminamos las puntuaciones, eliminamos todas las palabras que tengas un largo de 3 o menos caracteres y sus dígitos. Así obtenemos un dataset limpio y se puede procesar con más facilidad. Luego, se transforma los textos en vectores utilizando CountVectorizer

y/o TfidfVectorizer y quedandonos con las 10000 palabras mas frecuentes , las cuales se repiten por lo menos 20 veces.

## Cuadro de Resultados

| Modelo | F1-Test | Precision Test | Recall Test | Accuracy | Kaggle |
|---|---|---|---|---|---|
| **Bayes Naive** | 0.8348 | 0.8473 | 0.8227 | 0.8362 | 0.75014 |
| **Random Forest** | 0.8316 | 0.8168 | 0.8469 | 0.8274 | 0.76196 |
| **XgBoost** | 0.8371 | 0.8221 | 0.8525 | 0.8333 | 0.7177 |
| **Red Neuronal** | 0.8704 | 0.8599 | 0.8811 | 0.868 | |
| **Voting** | 0.8567 | 0.8491 | 0.8645 | 0.8545 | 0.7498 |
| **Stacking** | 0.8715 | 0.8602 | 0.8831 | 0.869 | 0.75169 |

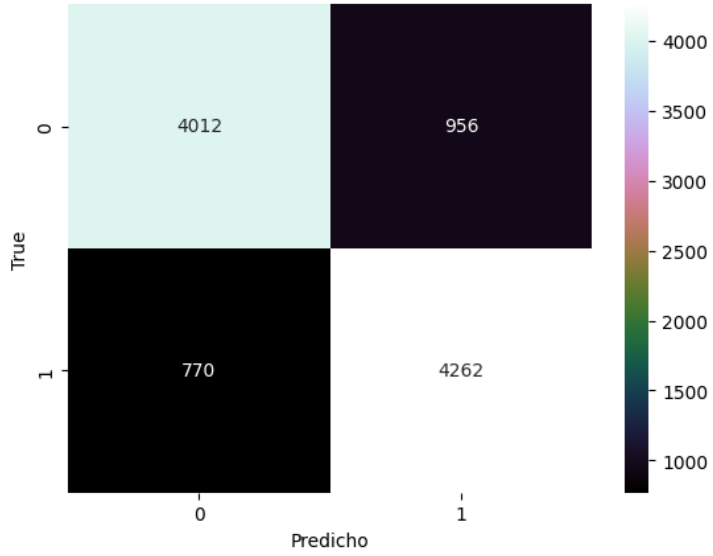
## Descripción de Modelos

A) Random Forest

Este modelo es el que mejor predicción tuvimos. Obtuvimos un 0.76 en kaggle.
Utilizando la técnica de preprocesamiento indicada anteriormente y vectorizando con TfidfVectorizer y el algoritmo RandomForestClassifier.
Aparte también se vectorizo con CountVectorizer pero se obtuvo peores resultados, aunque entre todos los modelos entrenados eran resultados muy aceptables.
Se optimizaron los hiperparámetros utilizando RandomSearchCV y obtuvimos los siguientes resultados:

- N_estimators : 376
- Min_samples_leaf: 1
- Max_features: log2
- Max_depth: 22
- Criterion: log_loss

Con todos estos hiperparametros obtuvimos nuestra mejor predicción.



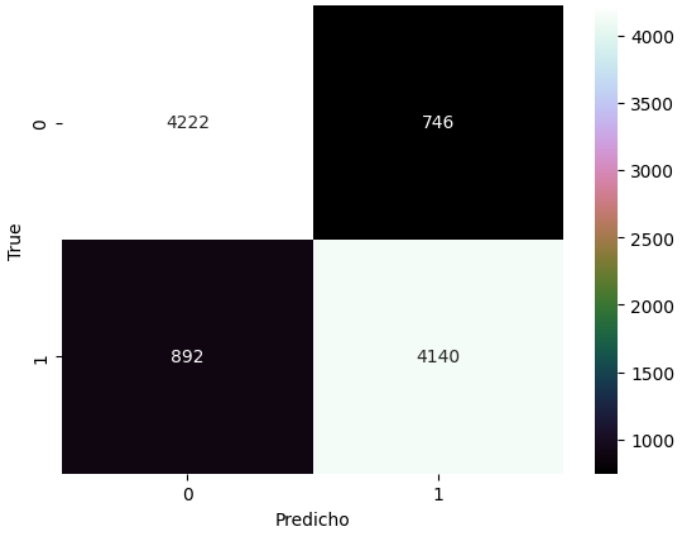

B) Bayes Naive Multinomial

Para este modelo se utilizó el algoritmo MultinominalNB y es el que estuvo más cerca de la predicción de random forest.

Utilizando la técnica de preprocesamiento indicada anteriormente y vectorizando con TfidfVectorizer. Aparte también se vectorizo con CountVectorizer pero se obtuvo peores resultados, aunque entre todos los modelos entrenados eran resultados muy aceptables.

Se optimizaron los hiperparámetros con RandomSearchCV con un valor de k fold 5 y obtuvimos los siguientes resultados:

- alpha : 4.067
- forcé_alpha: true
- fit_prior: false

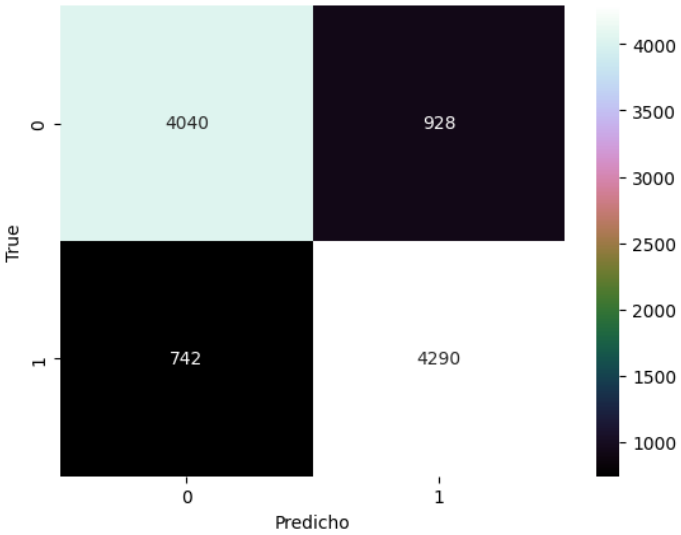



C) XGBoost

En este modelo no se obtuvieron buenos resultados. Se utilizo el algoritmo XGBClassifier. Es el modelo que más costo entrenarse.

Se pudo optimizar los hiperparametros con RandomSearchCV y llegamos a unos hiperpárametros aceptables pero la predicción no era muy buena.

- 'gamma': 0.4776
- 'learning_rate': 0.3252
- 'max_depth': 6,
- 'n_estimators': 124,
- 'subsample': 0.9883



D) Ensamble

Nosotros ensamblamos con los mejores modelos que obtuvimos con Random Forest, Bayes Naive Multinomial y XGBoost.
Obtuvimos una métrica buena aunque nunca se pudo superar a Random Forest.
Se utilizo un ensamble de Stacking Classifier con un meta modelo de regresión logística.
Es un ensamble que tardo mucho tiempo, pero se logró obtener aceptables resultados.

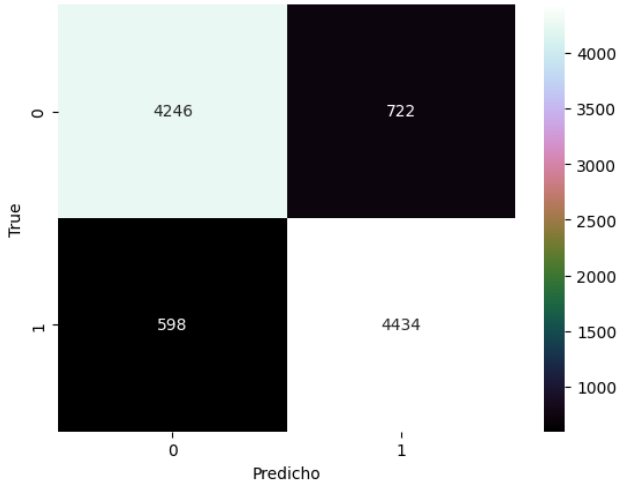
Aparte de Stacking, también probamos ensamblar con Voting pero los resultados obtenidos no fueron muy buenos aunque ensamblo en poco tiempo.

E) Redes Neuronales

Primero armamos una red simple estándar y luego optimizamos los hiperparametros con RandomSearch con el objetivo de obtener unos buenos hiperparametros y asi poder obtener una buena predicción.
La red que realizamos tuvo una capa de entrada, una capa de vectorización de texto, capas densas intermedias y una capa densa final con una función de activación sigmoide. La función de perdida que utilizamos fue entropía cruzada binaria y utilizamos Nadam como optimizador.
Los hiperparametros obtenidos utilizando RandomSearchCV son:

- units = 44
- mid_layers = 2
- activation = 'tanh',
- learning_rate = 0.001,
- vectorizer = 'count'

Esta red neuronal tuvo resultados no tan buenos aunque se podría haber optimizado un poco más.




## Conclusiones generales

En este trabajo, obtuvimos nuestra mejor predicción en Random Forest. El análisis exploratorio y preprocesamiento fue muy útil porque al correr los modelos sin procesar los datos nos daban unas performances muy bajas.

El modelo más sencillo y rápido de entrenar fue Bayes Naive Multinomial. Esto se debe por varias razones, pero además porque tiene pocos hiperpárametros para optimizar. Aunque sus valores fueron muy buenos.

El modelo que más tardo en optimizar fue XGBoost y fue difícil encontrar hiperpárametros con predicciones aceptables. Aparte también costo ensamblar los mejores modelos obtenidos con Random Forest, XGBoost y Bayes Naive Multinomial con Stacking.

## Tareas Realizadas

| Integrante | Promedio Semanal (hs) |
|---|---|
| Cecilia Jurgens | 8 hrs |
| Martin Schipani | 3 hrs |
| Marilyn Nicole Soto | 4 hrs |