Facultad de Ingeniería
Universidad de Buenos Aires

Informe TP N°2 - Críticas Cinematográficas

JARVIS

2023

Primer cuatrimestre

Nombre	Padrón
Rando Julián Agustín	108642
Gomez Alejo	106421
Celano Minig Nicolás Franco	107666

Preprocesamiento

En el contexto de nuestro trabajo de análisis de sentimientos en el DataFrame de reviews de películas en español, realizamos una serie de pasos de preprocesamiento de datos para garantizar la calidad y relevancia de la información. A continuación, se detallan los pasos que llevamos a cabo y la razón detrás de cada uno de ellos:

- **Verificación de datos nulos:** Comenzamos por verificar si existían valores nulos en el DataFrame. Este paso es fundamental para asegurarnos de que los datos estén completos y no haya información faltante que pueda afectar nuestros análisis posteriores.

- **Balanceo de la distribución de datos:** Evaluamos la distribución de los datos para asegurarnos de que estuvieran balanceados. El balanceo de los datos es esencial para evitar sesgos en nuestros modelos de análisis de sentimientos y garantizar que tengamos una cantidad similar de ejemplos positivos y negativos.

- **Eliminación de reviews en otros idiomas:** Descartamos todas las reviews que no estaban en español. Esto se hizo para enfocarnos únicamente en las reviews en español, ya que ese era el objetivo principal de nuestro análisis.

- **Definición de expresión regular para identificar palabras con mezcla de mayúsculas y minúsculas:** Establecimos una expresión regular para identificar palabras que contengan una combinación de letras mayúsculas y minúsculas. Esta técnica nos permitió capturar palabras como "granDía" o "buenDia" en su forma adecuada y mantener la coherencia de las palabras en el análisis posterior.

- **Separación de palabras utilizando la expresión regular:** Utilizamos la expresión regular definida anteriormente para separar las palabras en el texto de las reviews. Esto nos permitió tokenizar el texto y obtener una lista de tokens individuales para su posterior procesamiento.

- **Limpieza de texto y tokenización:** Aplicamos técnicas de limpieza de texto, cómo eliminar caracteres especiales, números y símbolos no deseados. Luego, procedimos a la tokenización, que es el proceso de dividir el texto en unidades más pequeñas, en este caso, tokens.

- **Eliminación de tokens duplicados:** Identificamos y eliminamos los tokens duplicados presentes en las listas de tokens. Esto fue necesario para evitar redundancias y reducir la redundancia de información en nuestros datos.

- **Eliminación de outliers basados en la longitud del token (utilizando IQR, Q1 y Q3):** Realizamos una eliminación de outliers utilizando el rango intercuartílico (IQR) y los cuartiles Q1 y Q3 de la longitud de los tokens. Esta técnica nos permitió descartar los tokens cuya longitud se consideraba atípica y que podrían afectar nuestros análisis posteriores.

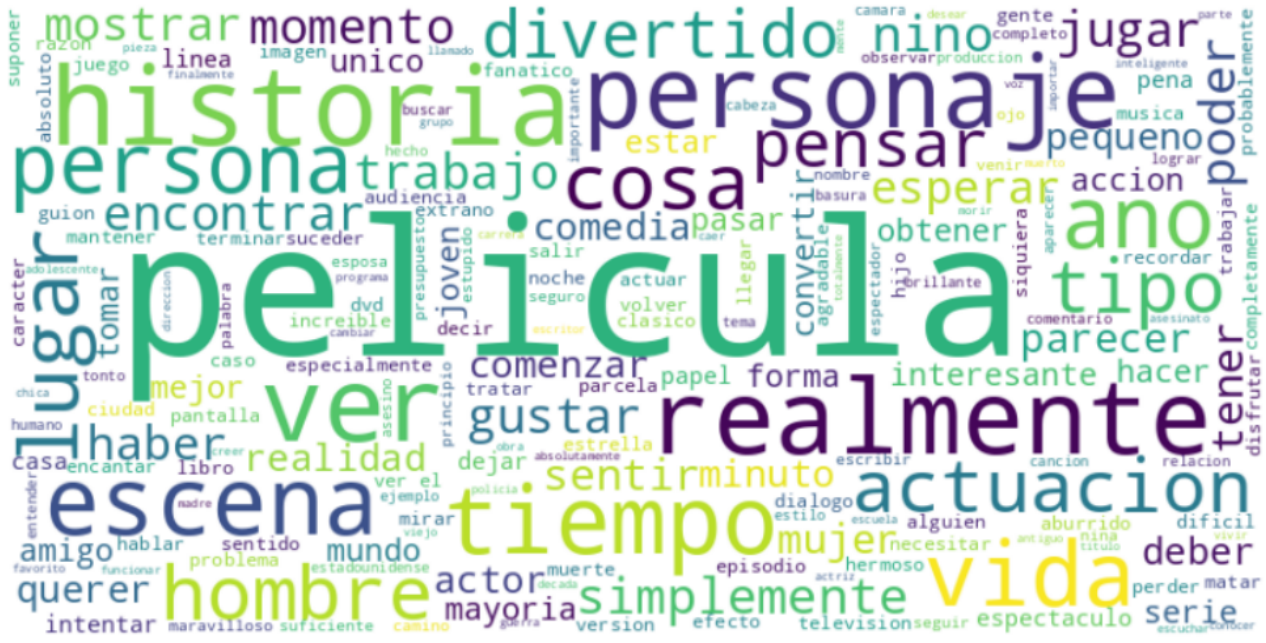
- **Observación de tokens y definición de umbral de frecuencia:** Realizamos un análisis de los tokens presentes en el dataframe y observamos sus frecuencias. Establecimos un umbral de frecuencia para eliminar aquellos tokens que no superan este umbral, con el objetivo de eliminar palabras poco frecuentes o ruido en nuestros datos. También establecimos un umbral de frecuencia máxima. Para establecer este umbral hicimos un mayor análisis de cuáles eran los tokens que estamos eliminando, ya que había tokens que podían ser muy frecuentes pero ser de importancia para la clasificación.

- **Eliminación de tokens con longitud menor o igual a 3:** Para mejorar la calidad y relevancia de los tokens, eliminamos aquellos con una longitud menor o igual a 3 caracteres. Esta decisión se tomó para descartar palabras demasiado cortas que podrían no tener un significado claro o ser poco informativas para nuestro análisis.

- **Mapeo del sentimiento positivo y negativo a 0 y 1:** Con el fin de estandarizar la representación del sentimiento en nuestro análisis, realizamos un mapeo del sentimiento positivo y negativo a los valores numéricos 0 y 1, respectivamente. Esta transformación facilita la posterior aplicación de modelos y algoritmos que requieren variables numéricas como entrada.

- **Eliminación de la columna de reviews:** Dado que ya hemos realizado todos los pasos necesarios de preprocesamiento y hemos extraído la información relevante en forma de tokens, eliminamos la columna de reviews original. Esto nos permite optimizar el uso de memoria y nos enfocamos únicamente en los datos necesarios para el análisis de sentimientos.

Obtuvimos una nube de palabras sobre los tokens de nuestro dataset



Modelos

XGBoost

Hiperparametros

A continuación, se detallan los hiperparámetros utilizados en el modelo y su justificación:

- ***learning_rate***: Es la tasa de aprendizaje del modelo, que controla la contribución de cada árbol al modelo final.
- ***max_depth***: Es la máxima profundidad de cada árbol en el modelo.
- ***n_estimators***: Es el número de árboles que se utilizarán en el modelo.
- ***gamma***: Es el parámetro de regularización que controla la reducción de la ganancia requerida para realizar una partición en un nodo del árbol.
- ***subsample***: Es la proporción de muestras utilizadas para entrenar cada árbol individualmente.
- ***colsample_bytree***: Es la proporción de características utilizadas para entrenar cada árbol individualmente.
- ***reg_alpha***: Es el término de regularización L1 aplicado a los pesos de las características.
- ***reg_lambda***: Es el término de regularización L2 aplicado a los pesos de las características.

Métricas obtenidas

Mejores hiperparámetros encontrados: {'alpha': 2.5, 'fit_prior': False}

	Train	Test	Kaggle
Accuracy	0.9780	0.8150	-
Recall	0.9780	0.8150	-
F1 score	0.9780	0.8150	0.6863
Precisión	0.9781	0.8150	-

Random Forest

Hiperparametros

A continuación, se detallan los hiperparámetros utilizados en el modelo y su justificación:

- ***n_estimators***: Es el número de árboles que se utilizarán en el modelo.
- ***criterion***: Es el criterio utilizado para medir la calidad de una división en los árboles.
- ***max_depth***: Es la máxima profundidad de cada árbol en el modelo.
- ***min_samples_split***: Es el número mínimo de muestras requeridas para realizar una división en un nodo interno. Sirve para controlar la profundidad del árbol y evitar el sobreajuste.
- ***min_samples_leaf***: Es el número mínimo de muestras requeridas para estar en una hoja del árbol. Sirve para controlar el sobreajuste y evitar hojas con pocas muestras.
- ***max_features***: Es el número máximo de características a considerar en cada división.

Métricas obtenidas

	Train	Test	Kaggle
Accuracy	0.8849	0.8272	-
Recall	0.8849	0.8272	-
F1 score	0.8849	0.8272	0.7152
Precisión	0.8849	0.8273	-

Bayes Nieve

Hiperparametros

A continuación, se detallan los hiperparámetros utilizados en el modelo y su justificación:

- ***alpha***: Es un parámetro de suavizado que se utiliza para evitar la probabilidad cero.
- ***fit_prior***: Es un parámetro booleano que indica si se deben estimar las probabilidades a priori de las clases o si se deben utilizar probabilidades uniformes.

Métricas obtenidas

	Train	Test	Kaggle
Accuracy	0.8774	0.8314	-
Recall	0.8744	0.8314	-
F1 score	0.8744	0.8314	0.7375
Precisión	0.8745	0.8314	-

Support Vector Machine

La consigna no pedía utilizar SVM, pero lo utilizaremos luego para realizar el ensamble.

Hiperparametros

- ***C***: Es un parámetro de regularización que controla el equilibrio entre el ajuste suave y el ajuste estricto de los datos de entrenamiento.
- ***kernel***: Especifica el tipo de función kernel utilizada.
- ***gamma***: Es un parámetro del kernel 'rbf' que controla el alcance de influencia de cada ejemplo de entrenamiento.

Métricas obtenidas

	Train	Test	Kaggle
Accuracy	0.9248	0.8296	-
Recall	0.9248	0.8296	-
F1 score	0.9248	0.8296	0.6910
Precisión	0.9248	0.8297	-

Ensamble Voting

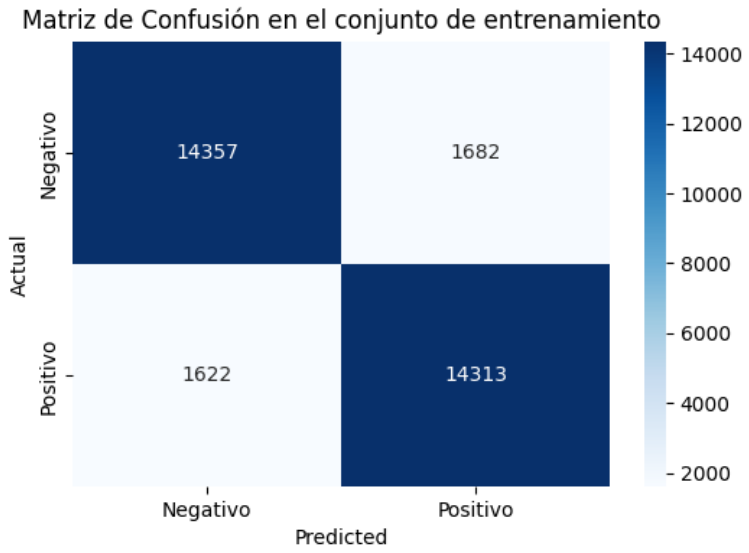
Modelos ensamblados

Se realizó el ensamble de SVM, Random Forest y Bayes Naieve. Los tres modelos son los mismos que se encuentran detallados en este informe.

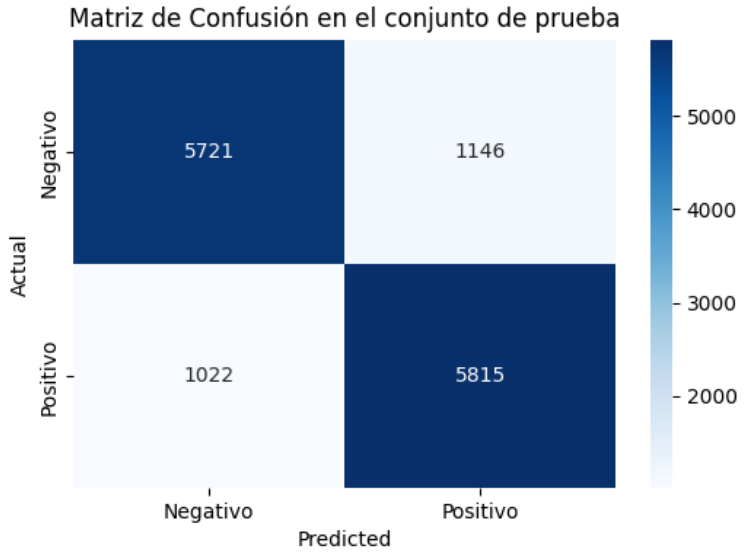
Métricas obtenidas

	Train	Test	Kaggle
Accuracy	0.8966	0.8417	-
Recall	0.8966	0.8417	-
F1 score	0.8966	0.8417	0.7313
Precisión	0.8966	0.8417	-

Podemos observar ambas matrices de confusión.



Vemos que se obtuvieron muy buenos resultados en el entrenamiento.



Podemos observar que existe algo de sobre ajuste.

Red Neuronal

La arquitectura de la red neuronal utilizada consiste en una secuencia de capas densamente conectadas, con la siguiente configuración:

- **Capa de entrada**: Dense layer con el número de neuronas especificado por el parámetro neurons, función de activación especificada por el parámetro activation, inicialización uniforme de pesos y un regularizador opcional.

- **Capas ocultas:** Cuatro dense layers con el número de neuronas especificado por el parámetro neurons_*i*, con *i* tomando valores entre 2 y 5 correspondiente al número de capa, función de activación especificada por el parámetro activation y un regularizador opcional.

- **Capa de salida**: Dense layer con 1 neurona, utilizando la función de activación 'sigmoid' para obtener la probabilidad de pertenencia a la clase positiva.

Esta arquitectura fue elegida para permitir la flexibilidad en la configuración del modelo y explorar diferentes combinaciones de hiperparámetros, como el número de neuronas en cada capa, la función de activación y la presencia de regularización. El uso de capas densas proporciona la capacidad de aprendizaje no lineal y la capacidad de modelar relaciones complejas entre las características de entrada y las etiquetas de salida.

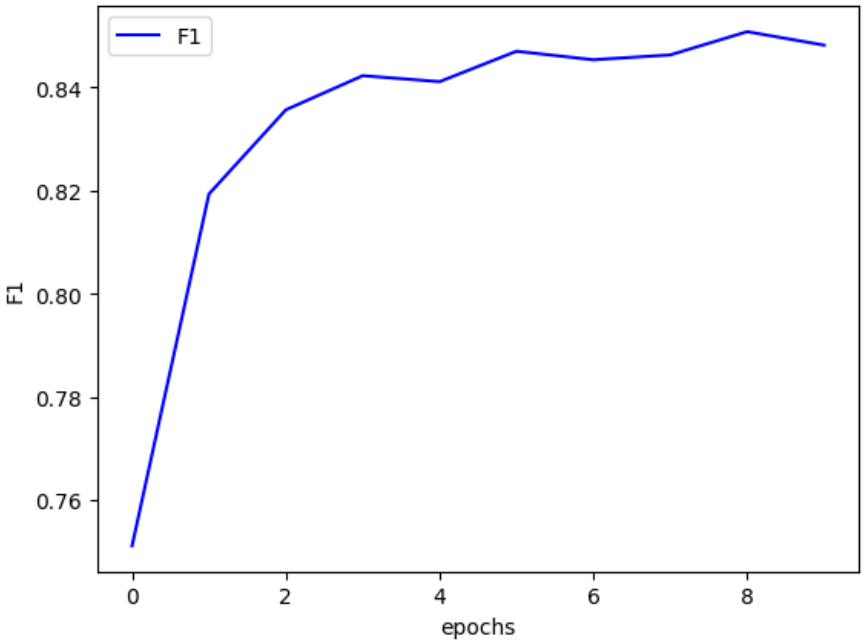
El modelo se compila con la función de pérdida 'binary_crossentropy', adecuada para problemas de clasificación binaria, y se define una métrica adicional 'accuracy' para evaluar la precisión del modelo durante el entrenamiento. También se puede incluir una métrica personalizada, como la puntuación F1, para evaluar el rendimiento del modelo.

Métricas obtenidas

	Train	**Test**	**Kaggle**
Accuracy	0.8995	0.7956	-
Recall	-	-	-
F1 score	0.9014	0.7997	0.7266
Precisión	0.8823	0.7820	-

En los puntajes F1 obtenidos en cada conjunto podemos observar un considerable overfitting, el cual podría haber sido reducido mediante una mejor optimización de los hiperparámetros correspondientes a la regularización de la red.

Aprovechando la inclusión personalizada de la puntuación F1, graficamos su cambio a medida que avanzan los ciclos de la red. Por pantalla podemos ver como su crecimiento es asintótico, aproximándose a valores cercanos a 1.


F1
0.84
0.82
0.80
0.78
0.76
F1
0
2
4
6
8
epochs