FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 Informe final

[75.06/95.58] Organización de Datos
Segundo cuatrimestre de 2023

Grupo 15

Ayala, Tomás Gabriel - tayala@fi.uba.ar - 105336
Giacobbe, Juan Ignacio - jgiacobbe@fi.uba.ar - 109866
Olaran, Sebastian - solaran@fi.uba.ar - 109410

Docente corrector:
Pereira, Francisco

Índice

1. Introducción

En este segundo trabajo práctico partimos de un dataset con críticas cinematográficas en español con un sentimiento asignado(que podía ser *negativo* o *positivo*), y a partir de ello crear modelos capaces de predecir si una nueva crítica es positiva o negativa.

El dataset de train contaba 3 columnas, las cuales contenían el ID de la reseña, la reseña en sí(un documento), y el valor del sentimiento, que básicamente era una variable que podía tomar dos valores(negativo o positivo). Además, nuestro dataset de entrenamiento consta de un total de 50.000 registros, de los cuales 25.000 son críticas positivas y 25.000 son críticas negativas. El dataset de test contaba con 8599 registros, con 2 columnas cada uno(el ID de la reseña, y la reseña en sí).

Primero nos centramos en explorar el dataset: vimos cómo se conformaban los cuerpos de las reseñas y qué palabras se repetían con mayor frecuencia. En un principio decidimos eliminar la columna de ID del dataset de train, ya que no nos interesaba esa información a la hora de crear modelos y predicciones: del dataset de train solo nos interesaban las reseñas y el sentimiento asociado a cada una respectivamente. Para el dataset de test no hemos eliminado registros ni columnas(la columna de ID la necesitábamos a la hora de subir nuestras predicciones a Kaggle, por lo que no la borramos).

La columna *review* contiene las críticas, y otra columna indica el sentimiento de dichas críticas, con 25,000 registros etiquetados como positivos y 25,000 como negativos. Esta estructura de datos es crucial para nuestro enfoque analítico, ya que nos brinda información directa sobre la distribución y polaridad de las opiniones expresadas en cada crítica.

El preprocesamiento de datos se ha aplicado de manera rigurosa, considerando la presencia equilibrada de críticas positivas y negativas en el conjunto de entrenamiento. Se han utilizado técnicas de limpieza de texto, tokenización y eliminación de stopwords para mejorar la calidad de los datos antes del entrenamiento del modelo.

No hemos tomado ninguna suposición sobre los registros del dataset, ya que creemos que al contar directamente con los documentos de las reseñas no habían suposiciones que tomar: las reseñas iban a venir con un formato determinado, y cada reseña era independiente de las otras, por lo que no encontramos ningún patrón o regla para las mismas.

2. Cuadro de Resultados

A continuacion podemos observar un cuadro de las metricas de cada mejor predictor de cada modelo:

Modelo	F1-Test	Precision Test	Recall Test	Accuracy	Kaggle
Bayes Naive	0.62598	0.793	0.666	0.6664	0.739
Random Forest	0.8235	0.825	0.8237	0.823	0.719
XGBoost	0.845	0.8462	0.8452	0.8452	0.703
Red Neuronal	0.8405	0.8407	0.8405	0.8406	0.693
Ensamble(**Mejor Modelo**)	0.479	0.50	0.500	0.5004	0.7480

3. Descripción de los Modelos

El modelo de **Bayes Naive** emplea una estrategia única en la combinación de varios modelos, centrándose en la probabilidad condicional y la independencia de las características para mejorar

el rendimiento general y mitigar el sobreajuste. En este caso, se basa en la aplicación del teorema de Bayes para calcular la probabilidad de pertenencia de un dato a una determinada clase.

A través de este enfoque, Bayes Naive realiza una especie de *votación probabilística* entre las diferentes características, asignando la clase más probable para un dato en función de las probabilidades calculadas

El modelo de **Random Forest** combina varios modelos (en este caso arboles de decisión) para mejorar el rendimiento general y reducir el sobreajuste. Esto se hace mediante el proceso de promedio o votación de los resultados de los árboles individuales.

En el ensamble híbrido **Voting** la idea consiste en combinar múltiples modelos: es más probable que se obtenga una predicción precisa, ya que los errores individuales de cada modelo tienden a compensarse entre sí. El método de votación es especialmente útil cuando se utilizan diversos tipos de modelos o algoritmos, ya que cada uno puede capturar diferentes aspectos de los datos o tener fortalezas en diferentes áreas. Al combinar sus predicciones, se puede lograr un rendimiento general mejorado.

El modelo de **Red Neuronal Multicapa (MLP)** se utiliza para predecir resultados a partir de observaciones. Consiste en capas de neuronas interconectadas que procesan la información y generan predicciones al pasar una observación a través de ellas. Es especialmente adecuada para problemas complejos y no lineales debido a su capacidad para aprender y representar patrones sofistficados en los datos. Para este tipo de modelos, el procesamiento de los datos en un parte esencial, ya que distintas formas de procesamiento, ya sea normalizacion, vectorizacion o tokenizacion afectan los resultados ampliamente. La arquitectura de la red neuronal creada sigue un diseño estratégico que busca capturar patrones complejos en los datos de entrada, ofreciendo así un modelo eficiente para la tarea de clasificación binaria. A continuación, se detallan las características clave de la arquitectura:

- **Capa de Entrada**: La capa de entrada tiene un número de nodos igual a la dimensión de los datos de entrada. Esta capa establece la conexión inicial entre las características de entrada y la red neuronal, permitiendo la propagación de la información a través de la red.

- **Capas Ocultas**: Tres capas ocultas se han incorporado con 200, 100 y 50 nodos respectivamente. Cada capa oculta está diseñada para aprender representaciones abstractas y características relevantes de los datos. Se ha incorporado una capa de Dropout después de cada capa oculta con una tasa del 30 porciento. La capa de Dropout desactiva aleatoriamente un porcentaje de nodos durante el entrenamiento, lo que contribuye a prevenir el sobreajuste al introducir variabilidad y redundancia en el aprendizaje.

- **Capa de Salida**: La capa de salida consta de un solo nodo con una función de activación 'sigmoid'. Esta configuración se adapta a la naturaleza de la tarea de clasificación binaria. La función 'sigmoid' comprime las salidas a un rango entre 0 y 1, interpretándolas como probabilidades de pertenencia a la clase positiva.

- **Compilación del Modelo**: La función de pérdida seleccionada es 'binary crossentropy', apropiada para problemas de clasificación binaria. El optimizador 'adam' se utiliza para ajustar los pesos de la red durante el entrenamiento.

- **Regularización con EarlyStopping**: Se implementa una estrategia de regularización utilizando la técnica de EarlyStopping. El monitoreo de la métrica 'accuracy' durante el entrenamiento permite detener el proceso de aprendizaje si no se observa mejora después de tres épocas consecutivas. Esta práctica es fundamental para evitar el sobreajuste y garantizar la generalización del modelo a nuevos datos.

El modelo que hemos identificado como el predictor más efectivo en nuestro trabajo práctico es el **Ensamble**. Esta elección se basa en una serie de factores que respaldan su eficiencia y capacidad para proporcionar resultados sólidos.

En primer lugar, las métricas de rendimiento que evaluamos, incluyendo precisión, recall, exactitud (accuracy) y puntuación F1, revelaron que el modelo logró resultados excepcionales en comparación con otras alternativas consideradas. Esto subraya su habilidad para realizar predicciones con precisión y coherencia, aspecto fundamental en tareas de predicción.

Adicionalmente, al observar el rendimiento del modelo en Kaggle, nuestro predictor demostró su eficacia al enfrentarse a un conjunto de datos nuevo y desconocido. Este hallazgo confirma que su rendimiento es generalizable y no se encuentra sobreajustado a nuestros datos de entrenamiento específicos.

Un punto destacado es que, a pesar de que inicialmente no parecía estar demasiado ajustado a los datos de entrenamiento, exhibió una notable capacidad para adaptarse al conjunto de prueba. Esto sugiere que nuestro modelo pudo generalizar patrones relevantes en lugar de simplemente memorizar los datos de entrenamiento, aspecto esencial para realizar predicciones precisas en situaciones del mundo real.

4. Conclusiones

Las conclusiones generales de nuestro trabajo son las siguientes:

- **Analisis exploratorio de datos**: La ejecución de un análisis exploratorio de datos ha demostrado ser una herramienta invaluable en nuestro proyecto. Al aplicar técnicas como la eliminación de stopwords, conversión a minúsculas, vectorización y tokenización, pudimos profundizar en nuestro conjunto de datos. Estas técnicas, en conjunto con el análisis exploratorio, proporcionaron una visión detallada de la estructura del texto, permitiéndonos comprender mejor las características distintivas de nuestros datos. Esta comprensión profunda fue esencial para guiar la elección de modelos adecuados y estrategias de procesamiento de texto, optimizando así la efectividad de nuestras predicciones.

- **Preprocesamiento de Datos**: La normalización, vectorización y tokenización de los datos mejoraron la performance de todos los modelos. En un principio no aplicamos estos procesamientos, y el desempeño de los modelos era muy pobre; sin embargo, cuando empezamos con el preprocesamiento estos mejoraron su performance de una forma exponencial.

- **Rendimiento de los Modelos**: Después de entrenar y evaluar varios modelos, encontramos que el modelo Ensamble tuvo el mejor desempeño en el conjunto de prueba (TEST). Este modelo superó a otros al lograr un F1-score más alto.

- **Rendimiento en Kaggle**: Nuestro mejor modelo fue un ensamble híbrido del tipo Voting, el cual fue construído a partir de 3 modelos: un modelo de Bayes Naive, un Random Forest, y un XGBoost.

- **Simplicidad vs. Desempeño**: El modelo más sencillo de entrenar y más rápido en nuestro caso fue Bayes Naive. En general su entrenamiento demoró 1 hora, pero fue el segundo mejor modelo en cuanto a rendimientos en Kaggle, y creemos fuertemente que la implementación de este modelo vale la pena en relación a sus tiempos de entrenamiento.

- **Uso Productivo del Modelo**: El problema con el ensamble híbrido es que debemos partir de una base de modelos ya entrenados, por lo que se puede dificultar su implementación a gran escala. Si se dispone de buenos tiempos para entrenar modelos y para ensamblarlos creemos que vale la pena por la performance que tiene.

- **Mejoras Potenciales**: Para mejorar nuestros resultados, podríamos considerar lo siguiente:

1. Explorar más a fondo tecnicas de preprocesamiento para logar un mejor desempeño

2. Aumentar el tamaño del conjunto de datos de entrenamiento si es posible.

3. Afinar los hiperparámetros de los modelos para lograr un mejor rendimiento. Si se dispondría de mayor tiempo para entrenar los modelos posiblemente se mejorará la performance de los mismos.

5. Tareas Realizadas

Integrante	Tarea
Tomas Gabriel Ayala	Armado de Reporte y Armado y entrenamiento de Redes Neuronales
Sebastian Olaran	Armado de Reporte y creación de arquitecturas de Random Forest Y XGBoost
Juan Ignacio Giacobbe	Armado de Reporte y creación de arquitecturas de Bayes Naive Y Ensamble