Universidad de Buenos Aires
Facultad de Ingeniería

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

75.06/95.58 – Organización de Datos

Curso 02
1C 2023

Trabajo Práctico 2: Críticas cinematográficas

Grupo 14

Integrantes:

- *BOTOSHANSKY, ALAN* *108933*
- *BOTOSHANSKY, IVAN* *108897*
- *NIEVA, ULISES TADEO* *107227*

Corrector: Nacho

Fecha de entrega: *29* de *junio* del 2023

Universidad de
Buenos Aires
Facultad de
Ingeniería

Resumen

En el siguiente informe mencionaremos aspectos relativos a distintos modelos de procesamiento de lenguaje natural, como su concepción, sus características y sus resultados.

Desarrollo

El curso de acción elegido para tratar a los primeros modelos fue el mismo. Primero creamos uno por defecto para conocer el rendimiento "base" que deberíamos superar. Luego optimizamos sus hiperparámetros mediante Cross Validation, variando a su vez la técnica de vectorización entre CountVectorizer y TfidfVectorizer. Para Bayes Naive, Random Forest y XGBoost esto lo hicimos mediante un Pipeline, mientras que para la Red Neuronal lo llevamos a cabo a través de una de sus capas.

El mejor modelo de Bayes Naive que aplicó CountVectorizer resultó ser el mismo que se obtuvo por defecto, dando un F1 Score de 0.8411. En cambio, los modelos de dicho tipo que utilizaron TfidfVectorizer otorgaron todos valores cercanos o iguales a 0.86, acercándose más a este número conforme el parámetro *alpha* decrecía de 0.4 a 0.2, sin influencia aparente del valor del booleano del parámetro *fit prior.* La diferencia entre los modelos fue tan pequeña que los que mostraron un mejor rendimiento en el dataframe de test fueron aquellos con un F1 Score ligeramente más bajo, siendo el mejor en este apartado aquel con un F1 Score de 0.8599 en el set de validación y uno de 0.72029 en la competencia de Kaggle (set de test).

Por otro lado, en Random Forest el mejor modelo fue aquel que empleó CountVectorizer, dando un F1 Score de 0.8473. De hecho, otro modelo que poseía los mismos hiperparámetros que el anterior (entropía como criterio, 180 árboles y raíz cuadrada para la cantidad máxima de variables) pero que utilizó el otro vectorizador, dio un F1 Score de 0.8459 y una performance en el dataframe de test remarcablemente más baja, 0.70711 contra 0.71467 del mejor RF.

En XGBoost ocurrió algo particular. Por un lado, el mejor modelo en entrenamiento obtuvo un F1 Score de 0.8498 y sus hiperparámetros fueron 130 estimadores, una máxima profundidad de 21, un *learning rate* de 0.2 y Count como vectorizador. Sin embargo, no fue el mejor modelo en Kaggle, dando allí un puntaje de 0.70498. El que mejor resultó en la competencia consiguió 0.71951, habiendo alcanzado previamente un F1 Score de 0.8429 en validación, teniendo 170 estimadores, máxima profundidad de 14, una tasa de aprendizaje de 0.5 y un vectorizador Tfidf.

La Red Neuronal a emplear debía ser capaz de manejar datos de texto, por lo tanto, 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 pérdida que utilizamos fue entropía cruzada binaria y utilizamos Adam como optimizador. A su vez debía poseer flexibilidad para ajustar hiperparámetros, los cuales serían la cantidad de neuronas por capa, la cantidad de capas intermedias, las funciones de activación de las capas intermedias, la tasa de aprendizaje, la cantidad de epochs, el batch size y el vectorizador.

La mejor Red obtenida tras aplicar Cross Validation fue además el modelo que mejor resultado dio en la competencia, teniendo en entrenamiento un F1 Score de 0.8951 y un puntaje de 0.76313 en Kaggle. Sus hiperparámetros fueron 10 neuronas por capa, una capa intermedia, función de activación ReLU, una tasa de aprendizaje de 0.001, 5 epochs, un batch size de 500 y Count como vectorizador. En líneas generales, las Redes fueron los modelos que mejor performance demostraron y no hubo una diferencia notable entre qué vectorizadores utilizaron.

Para los ensambles híbridos, los modelos elegidos fueron Bayes Naive, Random Forest y XGBoost. En Voting, curiosamente, el modelo que menor F1 Score obtuvo en entrenamiento (0.8498), fue el que mejor resultado dio en la competencia (0.72843). El Stacking fue el segundo modelo con mejor rendimiento, dando el más destacado un F1 Score de 0.8818 en entrenamiento y 0.74588 en Kaggle. No existió relación entre los rendimientos de los modelos que componían los ensambles con el rendimiento final del ensamble en sí.

Conclusión

A diferencia de lo ocurrido en el trabajo práctico previo, aquí no existió mayoritariamente ni igualdad ni correlación entre el resultado que mostraba un modelo en su etapa de entrenamiento, respecto al puntaje que luego se obtenía al realizar una predicción basada en el conjunto de prueba. Además de existir una amplia diferencia entre los puntajes en validación y prueba, también pudimos observar que modelos que esperábamos que fueran superiores debido a su alto rendimiento en validación, luego eran sobrepasados por sus semejantes que habían obtenido un menor rendimiento en dicha etapa.