TP2: Críticas Cinematográficas - Grupo 39

Introducción

- El problema a resolver consiste en determinar el sentimiento (positivo o negativo) de un conjunto de críticas cinematográficas a través de modelos de clasificación.

- Características del dataset:
  - Cantidad de registros: 50000
  - Cantidad de columnas: 3
  - Particularidades:
    - La cantidad de críticas positivas es igual a la cantidad de críticas negativas (25000 en ambos casos)
    - No posee valores faltantes

- Las técnicas utilizadas en el preprocesamiento fueron:
  - Vectorización: Vectorizamos las reviews del dataset de train y test para transformar las críticas de texto a una representación numérica para el entrenamiento de todos los modelos salvo la red neuronal.
  - Label Encoder: Codificamos las etiquetas de sentimiento para tenerlas en formato numérico (1 = positivo y 0 = negativo) para el entrenamiento de todos los modelos a excepción de Bayes Naive.

  Para redes neuronales tomamos de tamaño input_length = 1000 y 1500 por lo que podemos pensar que las primeras 1000 y 1500 palabras de la secuencia son más significativas como para poder predecir efectivamente el sentimiento.

Cuadro de Resultados

Modelo	F1-Test	Presicion Test	Recall Test	Kaggle
Bayes Naive	0.855	0.85	0.855	0.72843
Random Forest	0.835	0.83	0.83	0.73231
XgBoost	0.84	0.84	0.84	0.70691
Red Neuronal	0.87	0.87	0.875	0.74336
SVM polinómico	0.88	0.88	0.88	0.73347
Ensamble (Voting)	0.87	0.87	0.87	0.73657

Descripción de Modelos

- Bayes Naïve:

  Este modelo requiere una vectorización, que consiste en transformar las críticas que están en formato texto a una representación numérica, concretamente a un vector (bag of words) para que pueda ser entrenado. Para esto utilizamos un pipeline al cual le pasamos el método para vectorizar y el modelo Naïve Bayes. Y con esto ya puede ser entrenado con el dataset de las reviews

- Random Forest:

  Lo primero que hicimos fue optimizar los parámetros a través de un Random Search. Luego tomamos el mejor estimador y lo entrenamos con las reviews previamente vectorizadas en el preprocesamiento.

Los hiperparametros que utilizamos son: Bootstrap, ccp_alpha, class_weight, criterion, max_depth, max_featrures, min_sambles_leaf, min_samples_split, n_estimators.

- XGBoost:

  La característica de este modelo es su eficacia y capacidad para manejar conjuntos de datos grandes, su capacidad de regularización para evitar el sobreajuste y su velocidad en el entrenamiento.
  En este modelo utilizamos el dataset preprocesado con CountVectorizer y LabelEncoder. También optimizamos los hiperparametros utilizando un Random Search con scoring = f1, cv = 5 y los siguientes parámetros: learning_rate, max_depth, min_child_weight, subsample, colsample_bytree, gamma, n_estimators.

- Red neuronal:

  Utilizamos la librería keras.tensorflow para crear nuestro modelo de red neuronal. Para crearlo tokenizamos y secuenciamos el texto de las reviews y transformamos el sentimiento a un dato numérico negativo = 0, positivo = 1. Probamos 2 optimizadores Adam y Nadam
  Y al final utilizamos un threshold de 0.5 para dividir en positivo y negativo.

El mejor modelo fue el modelo 1 de redes neuronales entrenado con max_len = 1000 (ajusta las reseñas a una longitud máxima de 1000 palabras) y 5 epochs. En el modelo utilizamos una capa de Embedding. Esta capa se utiliza para convertir representaciones numéricas de palabras (índices enteros) en vectores densos de longitud fija. Luego una capa Flatten que aplana la salida de la capa de embedding a menos dimensiones y por último una capa de activación relu y otra sigmoid para la salida.
Como optimizador elegimos adam ya que dio un mejor resultado que Nadam cuando probamos en los diferentes modelos.

El ensamble de tipo Voting fue creado con los siguientes 3 modelos base: Random Forest, XgBoost y SVM polinómico. Este último modelo se creó porque no pudimos adaptar los modelos Red neuronal y Bayes Naive (utilizando scikeras) para que puedan ser soportados por VotingClassifier, ya que al no pertenecer a los modelos de sklearn no eran compatibles para poder ser usados como modelos base.

Conclusiones generales

- El análisis exploratorio nos sirvió para tener una mejor noción del conjunto de datos, ver que había misma cantidad de reseñas positivas que negativas, que la mayoría de reseñas tiene entre 1000 caracteres.
- Las tareas de preprocesamiento ayudaron a mejorar la performance solo de los modelos de redes neuronales ya que truncamos las reseñas a 1000/1500 palabras.
- El modelo SVM Polinomico fue el que tuvo mejor desempeño en TEST.
- El modelo de Red Neuronal fue el que tuvo mejor desempeño en Kaggle.
- El XGBoost con hiperparametros default performa un poco por debajo del modelo con hiperparametros optimizados pero es mucho más rápido(1 min vs 2hs).
- Al tratarse de críticas cinematográficas podría dar una idea del puntaje de una película fiable.
- Se podría mejorar probando optimizar más hiperparametros y modificar la arquitectura de la red neuronal probando más combinaciones ya que es muy personalizable.
- Analizando los resultados de todos los modelos vimos que todos predicen con al menos un 70% f1-score, ninguno destaca pero todos son aceptables.
- Otra conclusión es que en algunos casos aumentar las épocas resulta favorecedor y en otros no, independientemente del score de train.

Tareas Realizadas

Integrante	Promedio Semanal (hs)
Cattaneo Ariadna Antonella	6 horas
Agustin Ezequiel Sanchez Decouflet	6 horas
Franco Dario Mazzaro	6 horas