Trabajo practico 2

Bayes Naive

Preprocesamiento

Fue uno de los modelos que mejor respondió al preprocesamiento, ya que sin este tenía solo 0.68 (pasó a tener un final de 0.72) de precisión en Kaggle, también respondió mejor al tfidvectorizer que al countvectorizer.

Se quitaron las stop words, se realizo stemming. Para tokenizar se uso el TfVectorizer de sklearn el mismo vectorizer se encargo de aplicar la stop words pasadas por parametro.

De este modelo se destaca la relación entre la simpleza para crearlo y su efectividad.

Hiperparametros

Hiperparametro	Descripcion	Valor utilizado
force_alpha	Parámetro que indica si se debe forzar el valor del hiperparámetro alfa.	False
fit_prior	Parámetro booleano que indica si se deben ajustar las probabilidades a priori de las clases durante el entrenamiento.	True
alpha	Parámetro de suavizado. El suavizado ayuda a evitar problemas de probabilidad cero y mejora la generalización del modelo.	1.0

Resultados

Precison	Recall	F1-score	Puntaje kaggle
0.85	0.86	0.85	0.72

XgBoost

Preprocesamiento

Se quitaron las stop words y para tokenizar se utilizo CountVectorized de sklearn.

Hiperparametros

Hiperparametro	Descripcion	Valor utilizado
n_estimators	Número de árboles de decisión (estimadores) que se crearán durante el entrenamiento	527
learning_rate	Tasa de aprendizaje que controla la contribución de cada árbol	0.4
max_depth	Profundidad máxima de cada árbol	4
objective	Función de pérdida	binary:logistic

Resultados

Precision	Recall	F1-score	Puntaje kaggle
0.85	0.88	0.87	0.72

Red neuronal

Preprocesamiento

Para reducir un poco el vocabulario, se decidio aplicar stemming (reducir las palabras a su forma raìz o base), usando el PorterStemmer.

Luego se utilizo CountVectorized, para realizar la tokenizaciòn de las palabras.

Parametros del tokenizer: min_df=0.001 y max_feature=10000.

Se establecio un max de 10000 palabras, para evitar el consumo excesivo.

Arquitectura

- Capa de incrustacion (embedding)
    - Esta capa se utiliza para aprender representaciones vectoriales densas de las palabras en un espacio de baja dimensionalidad.
- Capa de aplanamiento (flatten)

    - Esta capa se utiliza para convertir la salida tridimensional de la capa de incrustación en un vector unidimensional.
- Capa densa (dense)
    - Esta es una capa completamente conectada en la que cada neurona está conectada a todas las neuronas de la capa anterior. En este caso, la capa densa tiene una sola neurona. La función de activación utilizada es la función sigmoide (`sigmoid`), que produce una salida entre 0 y 1, lo que se interpreta como la probabilidad de pertenecer a la clase positiva.

Layer	Output shape	Param #
Embedding	(None, 1000, 200)	2000000
Flatten	(None,2000000)	0
Dense	(None,1)	2000001

Total params: 4.000.001

Hiperparametros

La busqueda de los hiperparametros se realizo manualmente por cuesitones de saturacion de memoria ram de Google Colab.

Hiperparametro	Descripcion	Valor utilizado
input_dim	Numero neuronas de entrada	10000
output_dim	Numero de neuronas de salida	200
input_length	Especifica la longitud máxima de la secuencia	10000
epochs	Cantidad de veces que el modelo recorrerá todo el conjunto de entrenamiento durante el proceso de entrenamiento.	3
batch_size	Número de ejemplos de entrenamiento que se utilizarán en cada iteración del proceso de entrenamiento	200
activation_function	Función de activación	sigmoid
loss	Función de pérdida utilizada para medir el error entre las salidas predichas y las salidas reales durante el entrenamiento	binary_crossentropy
optimizer	Optimización utilizado para ajustar los pesos de la red neuronal durante el entrenamiento	rmsprop

Resultados

Precision	Recall	F1-score	Puntaje kaggle
0.89	0.85	0.872	0.758

Random Forest

Preprocesamiento

Se quitaron las stop words, se probó tanto stemming como lemmatizing y dio mejores resultados el último, por lo que se realizó lemmatizing. Para tokenizar se usó el CountVectorizer (Ya que dio mejores resultados que el Tfidf) de sklearn. El mismo vectorizer se encargó de aplicar las stop words pasadas por parámetro. Para lemmatizing se utilizó el WorldNetLemmatizer.

Hiperparámetros

Hiperparámetro	Descripción	Valor utilizado
n_estimators	Número de árboles de decisión que se utilizarán. En este caso, se establece en 100, lo que significa que se entrenarán 100 árboles de decisión.	100
min_sample_split	Controla el número mínimo de muestras requeridas para dividir un nodo interno del árbol de decisión. Un valor más alto de min_samples_split puede ayudar a prevenir el sobreajuste al hacer que el árbol de decisión sea menos complejo.	4
criterion	Función de calidad que se utilizará para medir la calidad de una división en el árbol de decisión.	entropy

Resultados

Precision	Recall	F1-score	Puntaje kaggle
0.84	0.86	0.85	0.71

Ensamble

Preprocesamiento

Eliminacion de stop words y aplicacion de stemming. Se uso el CountVecotrizer de sklearn y el PorterSteemer de nlkt.

Hiperparametros

Los hiperparametros utilizados son los mejores que se obtuvieron en la busqueda de los modelos anteriores.

Modelos utilizados

- Random Forest
- Bayes Naive
- XgBoost

Resultados

Precision	Recall	F1-score	Puntaje kaggle
0.87	0.86	0.87	0.73

Comentario: se entreno una version con estos tres modelos y una red neuronal, sin embargo no hubo una mejoria no en las metricas de train ni en las de kaggle. (f1-score: 0.87) (kaggle-score: 0.72)

Tareas realizadas

Tarea	Integrante
Desarrollo del modelo Bayes Naive	Axel Apaza
Desarrollo del modelo XGBoost	Alan Cristian Goyzueta
Desarrollo del modelo Random Forest	Valentin Juan Gonzalez
Desarrollo de La red Neuronal	Alan Cristian Goyzueta, Franco Gazzola
Desarrollo del Emsable	Todos participaron
Informe	Alan Cristian Goyzueta, Franco Gazzola

Aclaracion: si se quiere ejecutar todo el contenido de colab, se debe tener una carpeta llamada TP2 en el drive “MI Unidad”, con el dataset de train.