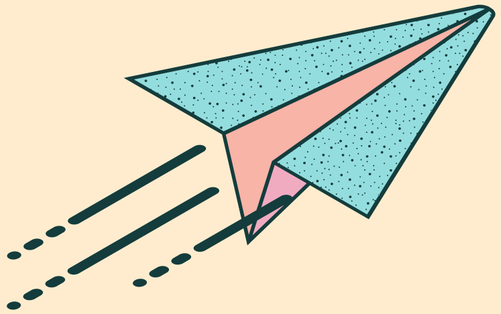


Procesamiento del Lenguaje Natural

AKINATOR LITERARIO

Agustín Jerusalinsky, Natali Lilienthal y Camila Borinsky

Contenido

OBJETIVO

OBJETIVO

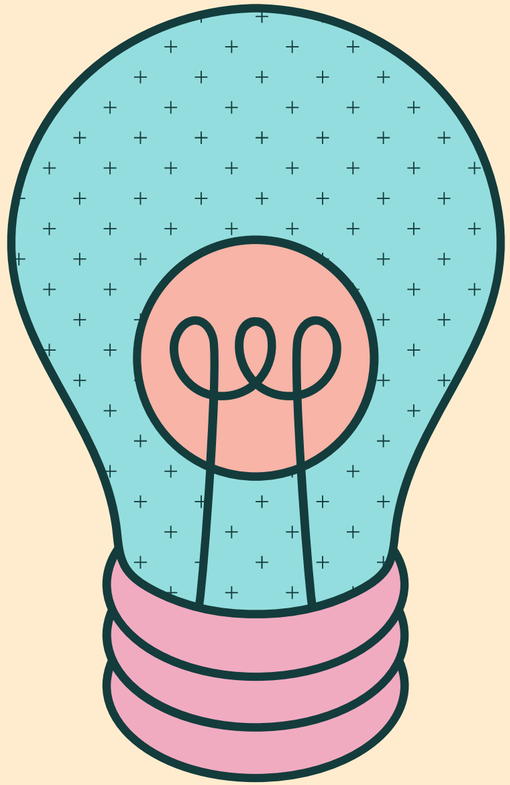
Entrenar un modelo con libros de **Project Gutenberg** de una cantidad limitada de autores para que pueda predecir el autor de un fragmento.

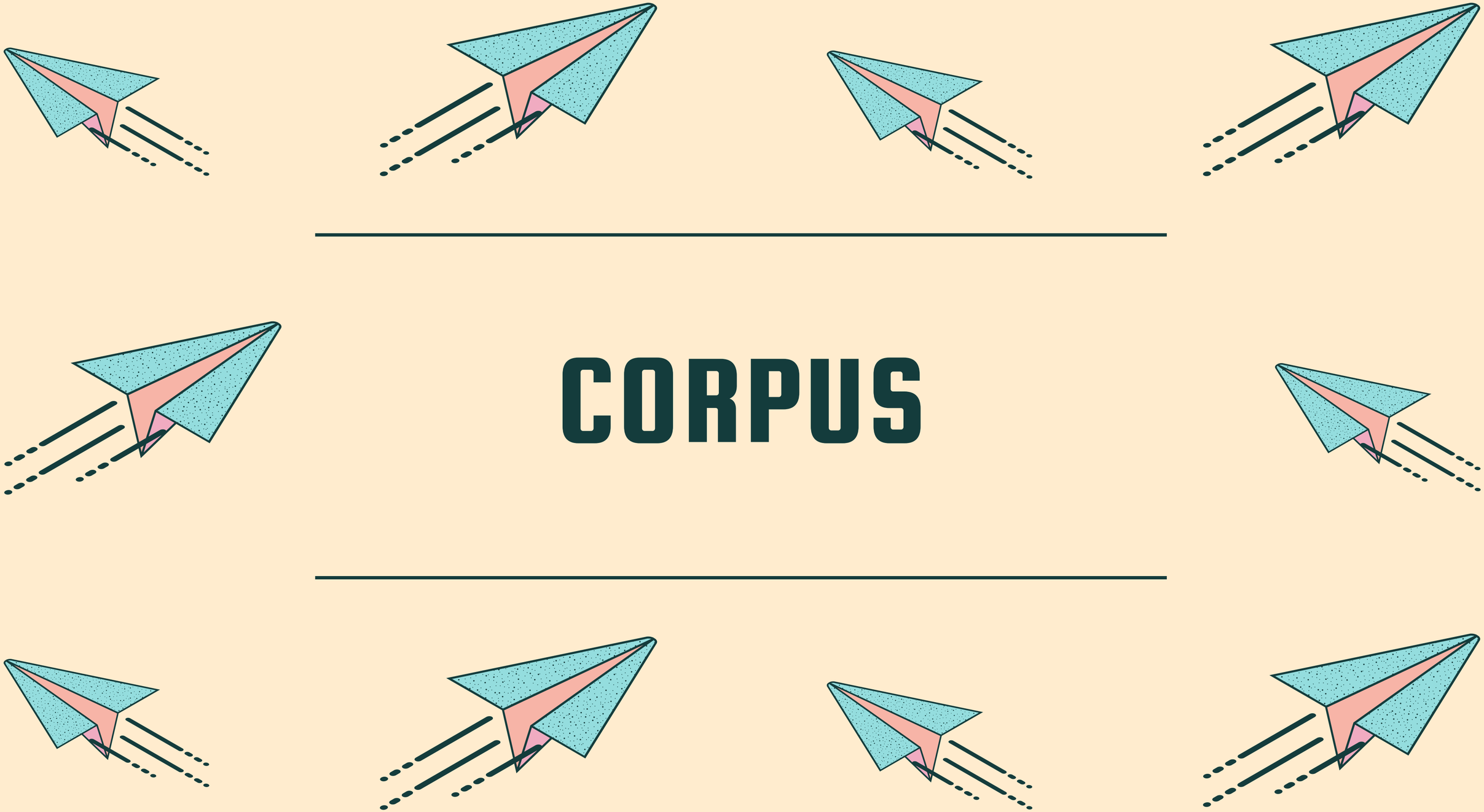
ENTRENAMIENTO

- Probar estrategia interpretable de clasificación
- Fine-tuning de un modelo pre-entrenado
- Clasificación sobre modelo pre-entrenado sin fine-tuning

EVALUACIÓN

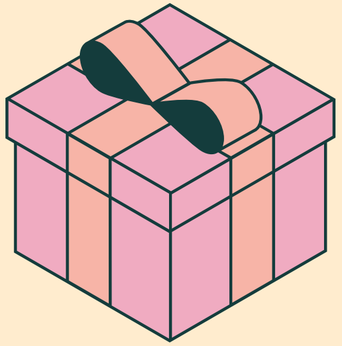
- Matriz de confusión
- Curva ROC
- Métricas: Precision, Accuracy, Recall y F1 score





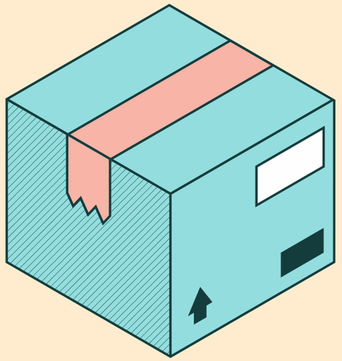
CORPUS

CORPUS

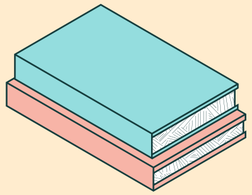


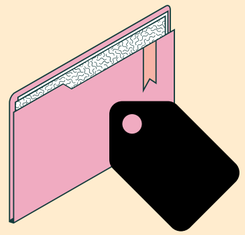
Armamos el corpus con una muestra de 6 autores (3 libros de cada uno)

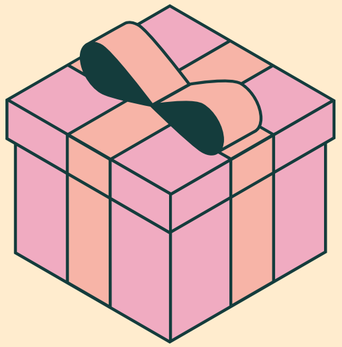
- Agatha Christie
- Jane Austen
- Louisa May Alcott
- Arthur Conan Doyle
- William Shakespeare
- Jules Verne



Dividimos el texto de un libro en párrafos. Cada párrafo es un input.





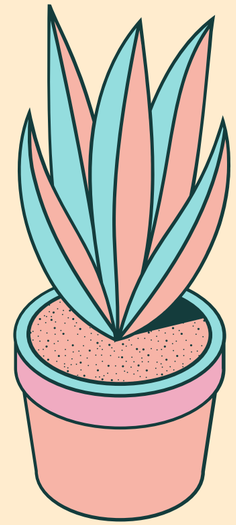


MODELOS Y RESULTADOS

TF-IDF Y BOW

MODELOS interpretables

Medida numérica para expresar cuán relevante es una palabra en un documento.



TF-IDF

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

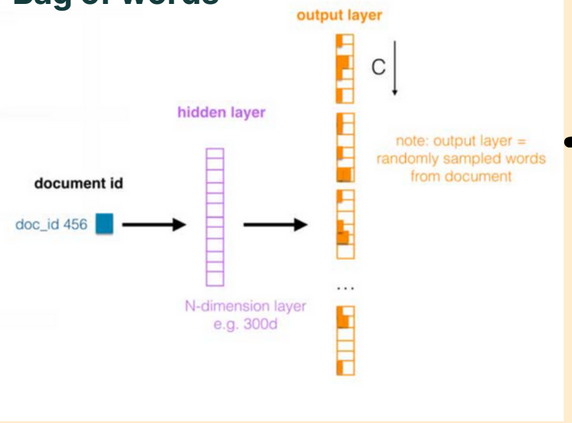
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

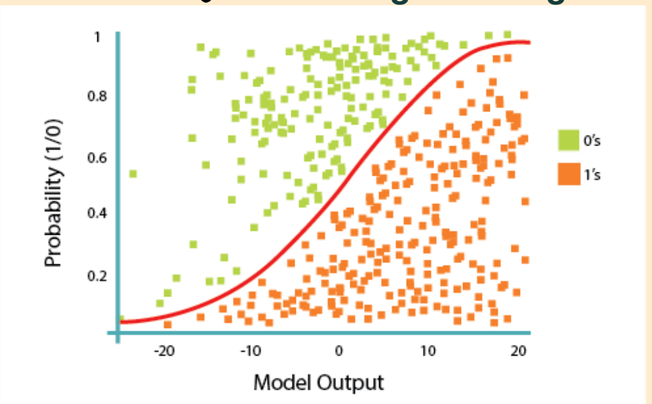
N = total number of documents

	de	gustan	hay	los	me	muchas	perros	razas
0	0.000000	0.546454	0.000000	0.546454	0.546454	0.000000	0.322745	0.000000
1	0.000000	0.000000	0.541343	0.000000	0.000000	0.000000	0.840802	0.000000
2	0.504611	0.000000	0.383770	0.000000	0.000000	0.504611	0.298032	0.504611

Bag of words



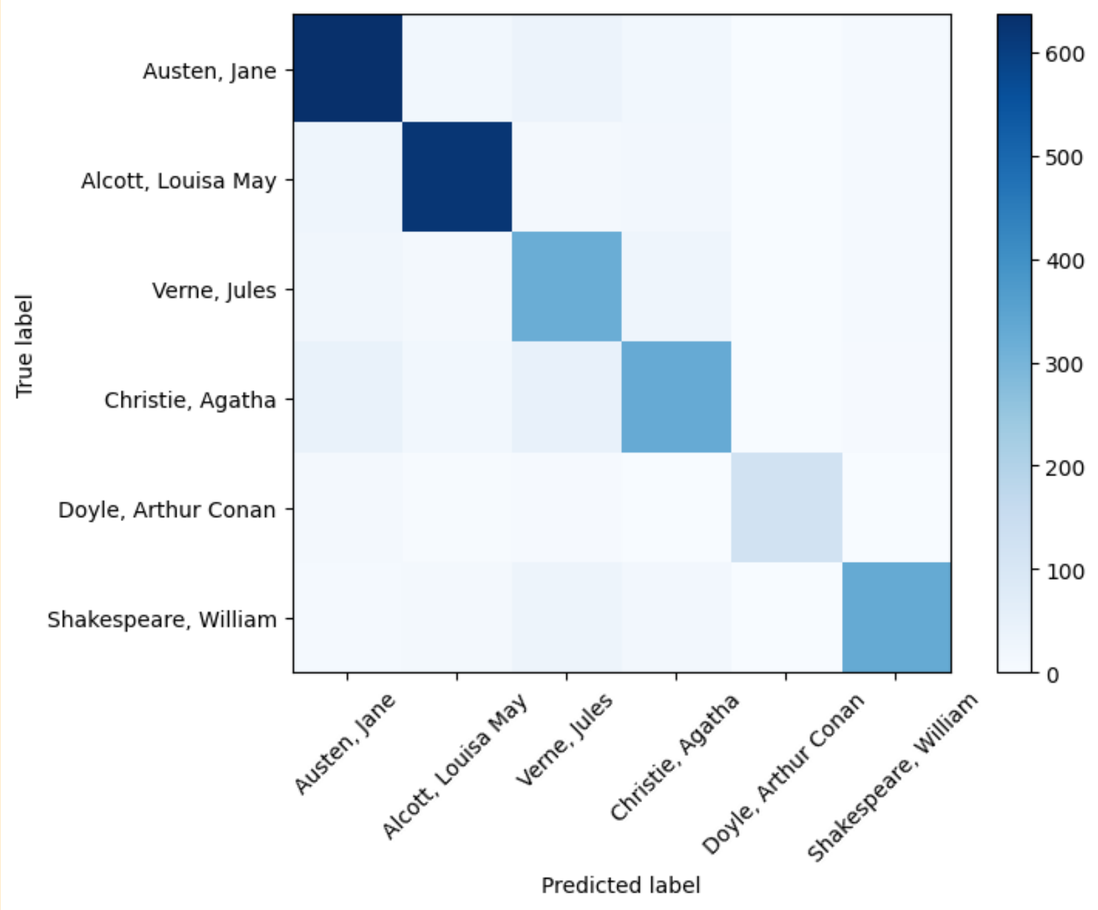
Regresión logística

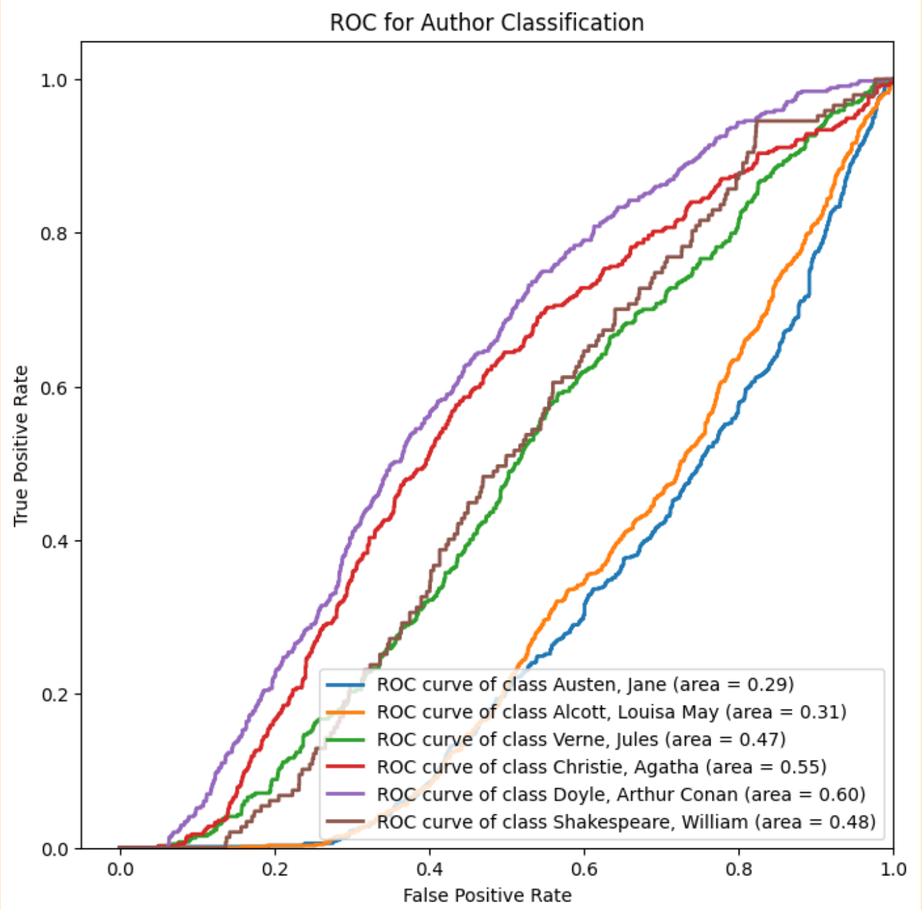


RESULTADOS TF-IDF

Matriz de confusión y curva ROC
Sin balancear

Tiempo 14.5s; 198 iteraciones; Sin balanceo

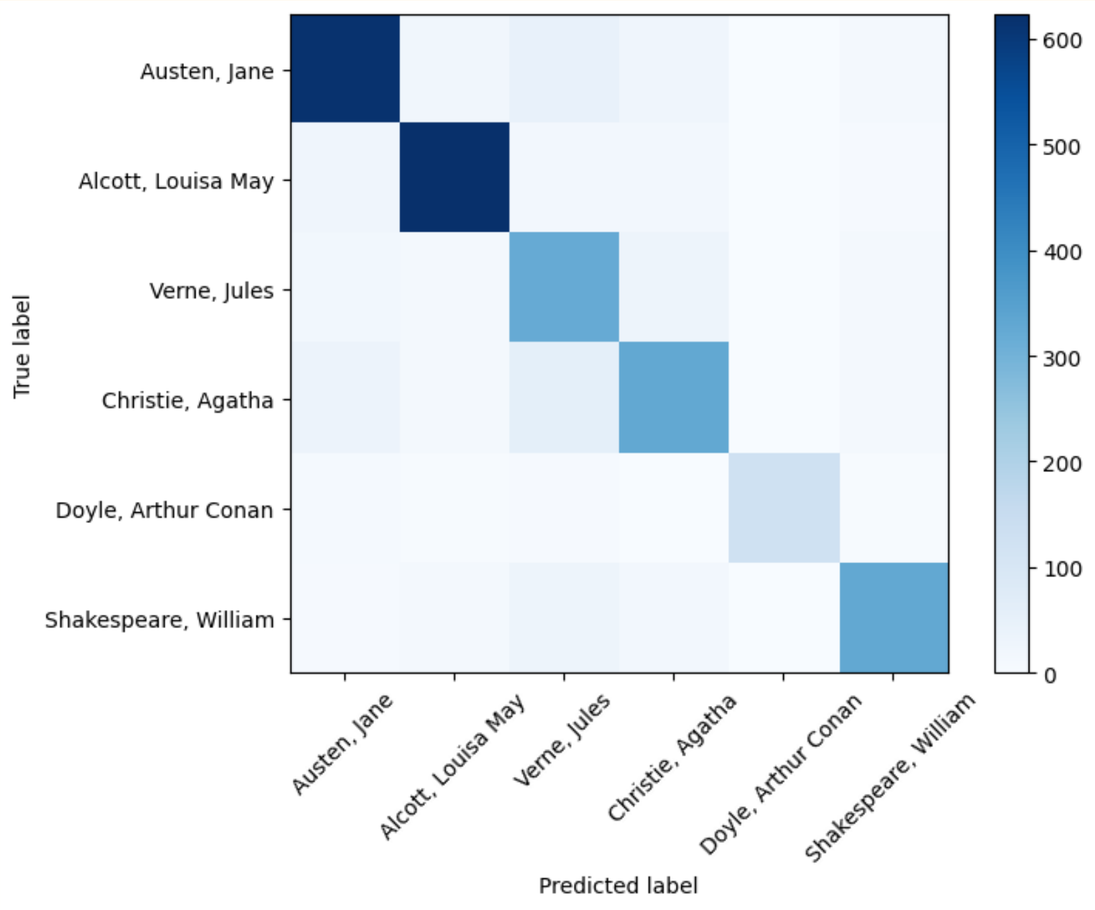


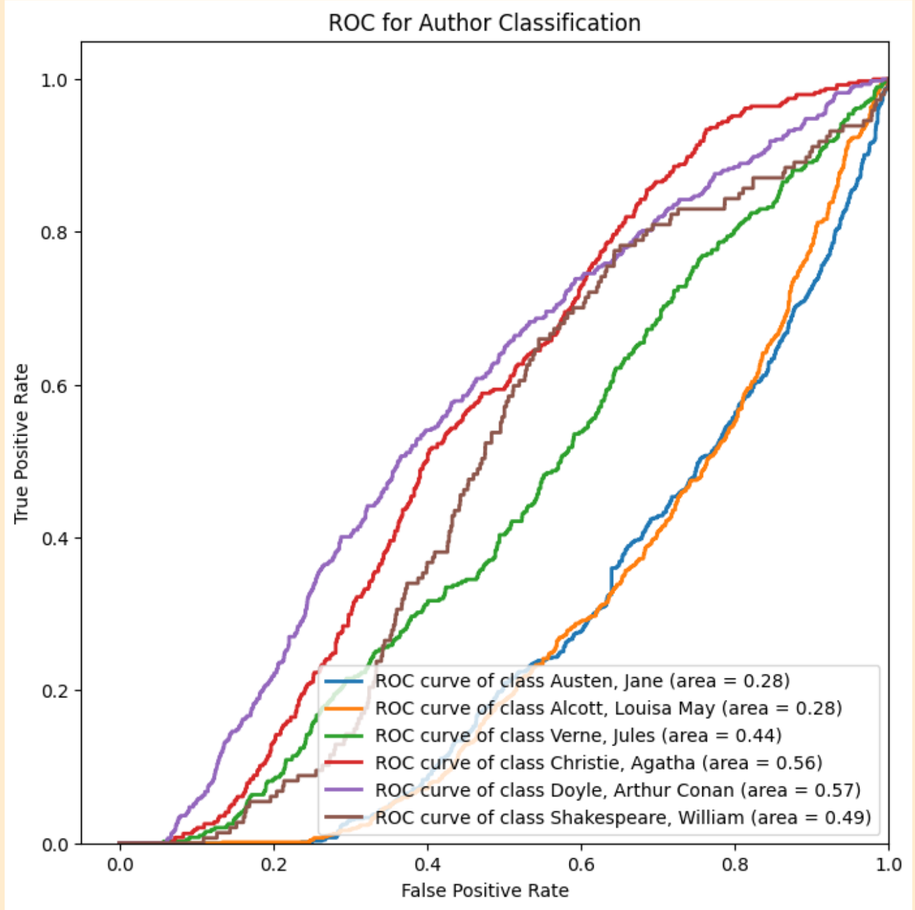


RESULTADOS BOW

Matriz de confusión y curva ROC
Sin balancear

Tiempo 22.3s; 305 iteraciones; Sin balanceo





RESULTADOS TF-IDF vs BOW

Métricas
Sin balanceo

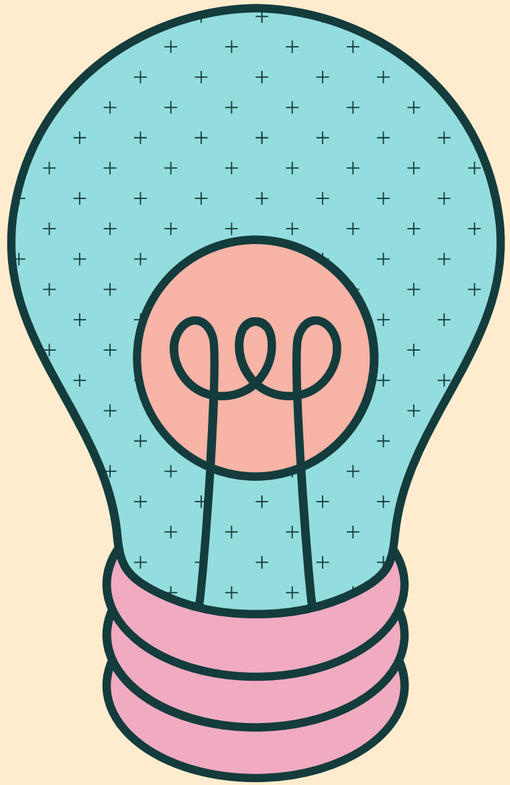
Parámetros	Tiempo 14.5s 198 iteraciones Sin balanceo	Tiempo 22.3s 305 iteraciones Sin balanceo
Test metrics	**Accuracy**: 0.845 **Recall**: 0.832 **Precision**: 0.859 **F1**: 0.843	**Accuracy**: 0.840 **Recall**: 0.833 **Precision**: 0.851 **F1**: 0.840
Train metrics	**Accuracy**: 0.921 **Recall**: 0.917 **Precision**: 0.928 **F1**: 0.922	**Accuracy**: 0.958 **Recall**: 0.959 **Precision**: 0.960 **F1**: 0.959

CONCLUSIÓN

Necesitamos balancear el dataset.

OPCIONES

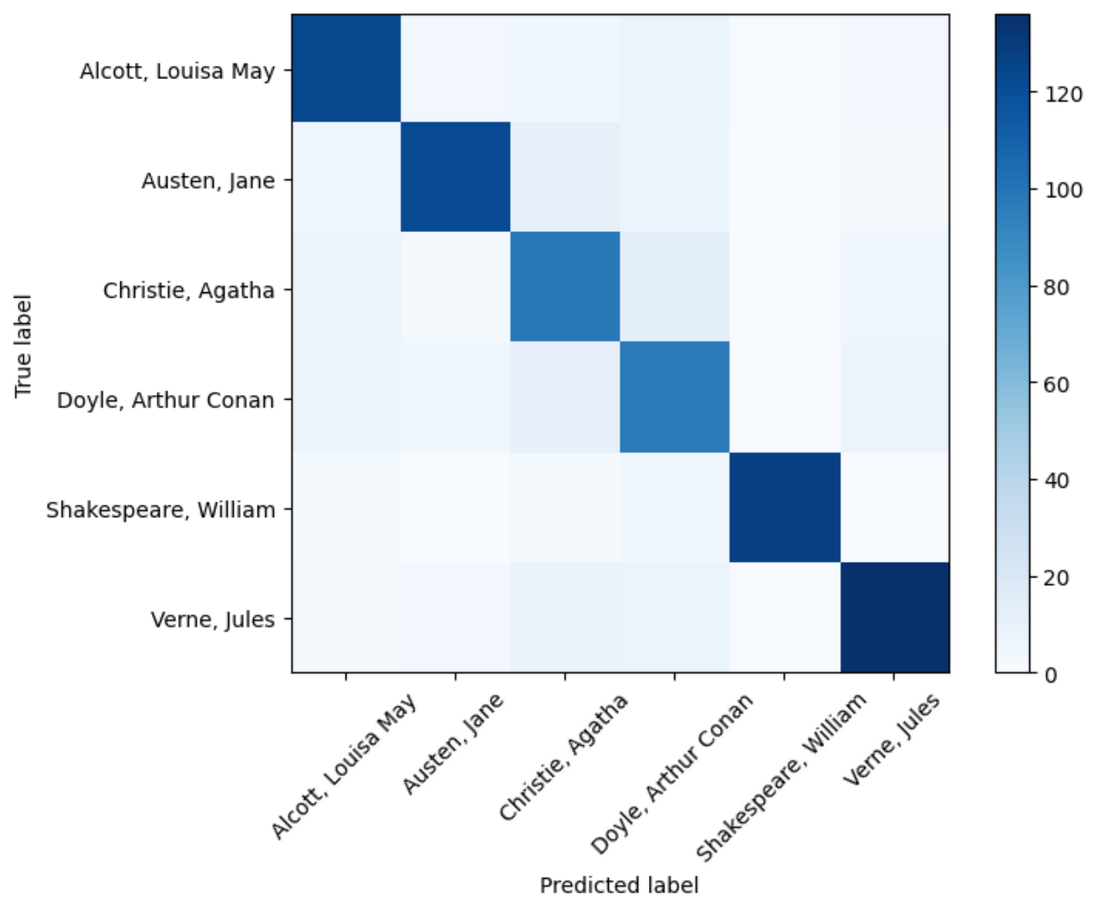
- Down-sampling: cortamos los datasets a la cantidad del que menos tiene.
- Up-sampling: Tomamos muestras con reemplazo para obtener la cantidad del autor con mayor cantidad de muestras.



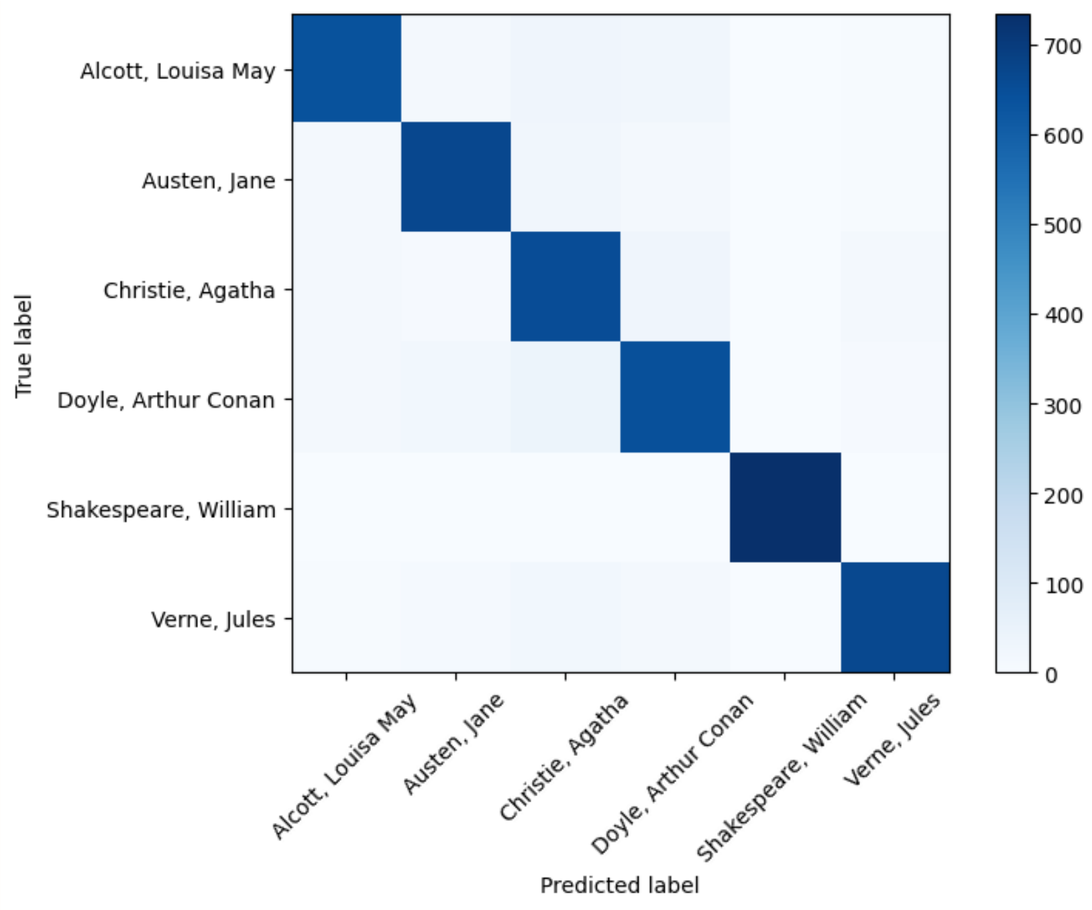
RESULTADOS TF-IDF

Matriz de confusión
Upsampling vs Downsampling

Tiempo 4.4s; 118 iteraciones; Downsampling 1410

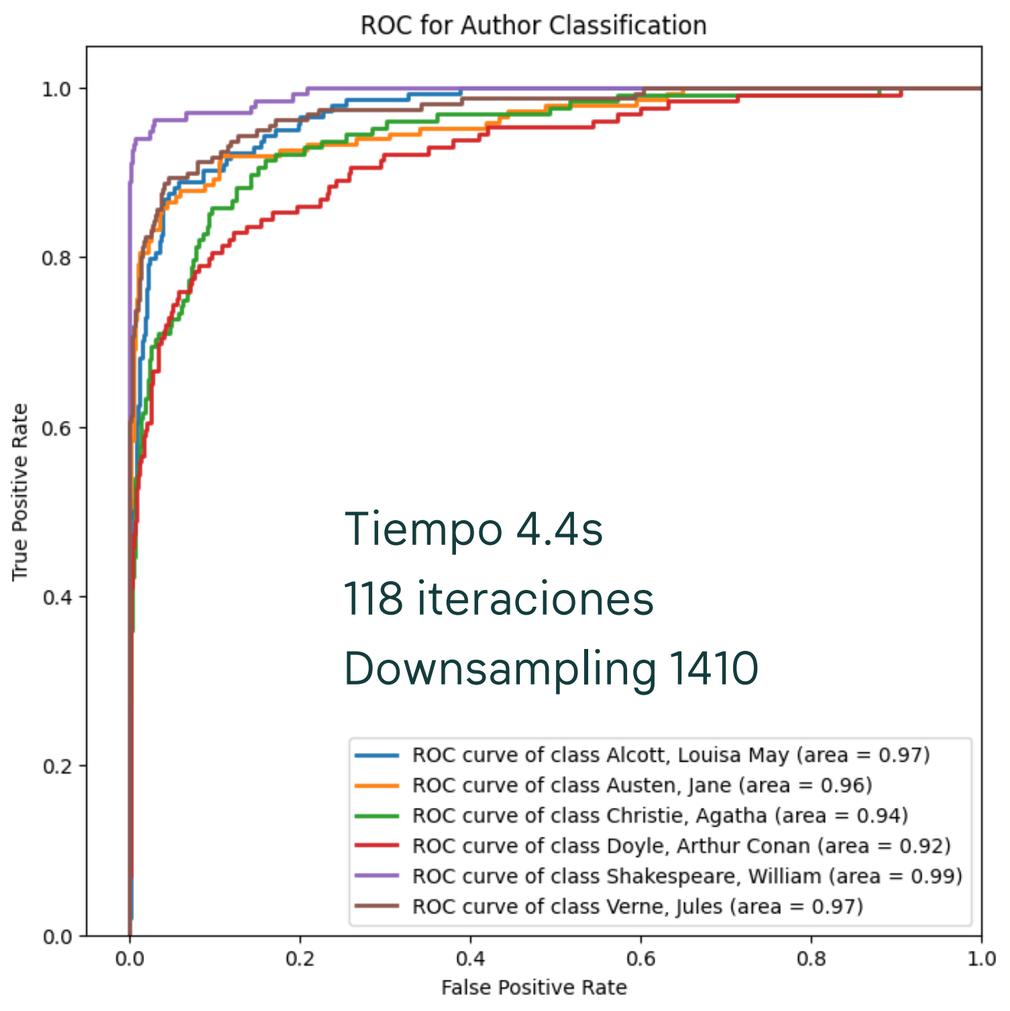


Tiempo 19.1s; 221 iteraciones; Upsampling 7228

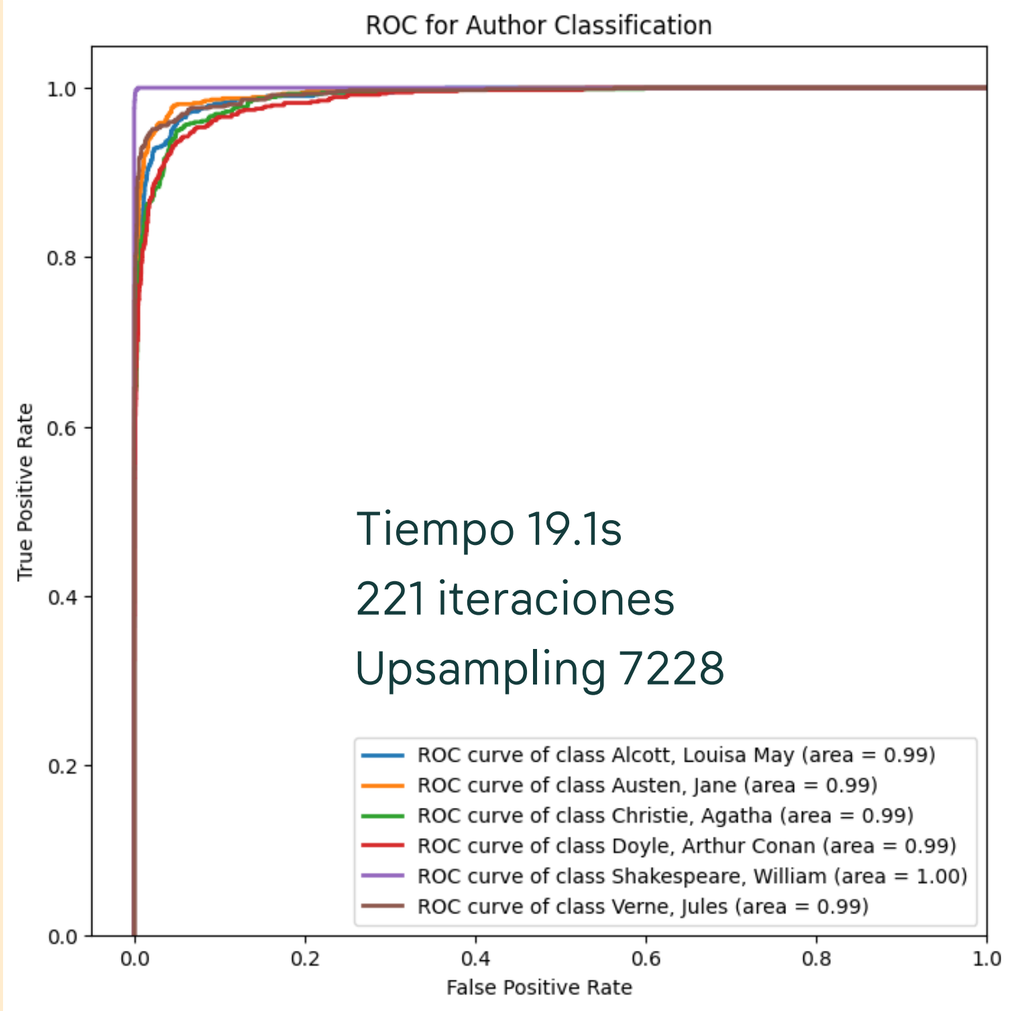


RESULTADOS TF-IDF

Curva ROC
Upsampling vs Downsampling



Tiempo 4.4s
118 iteraciones
Downsampling 1410



Tiempo 19.1s
221 iteraciones
Upsampling 7228

RESULTADOS TF-IDF

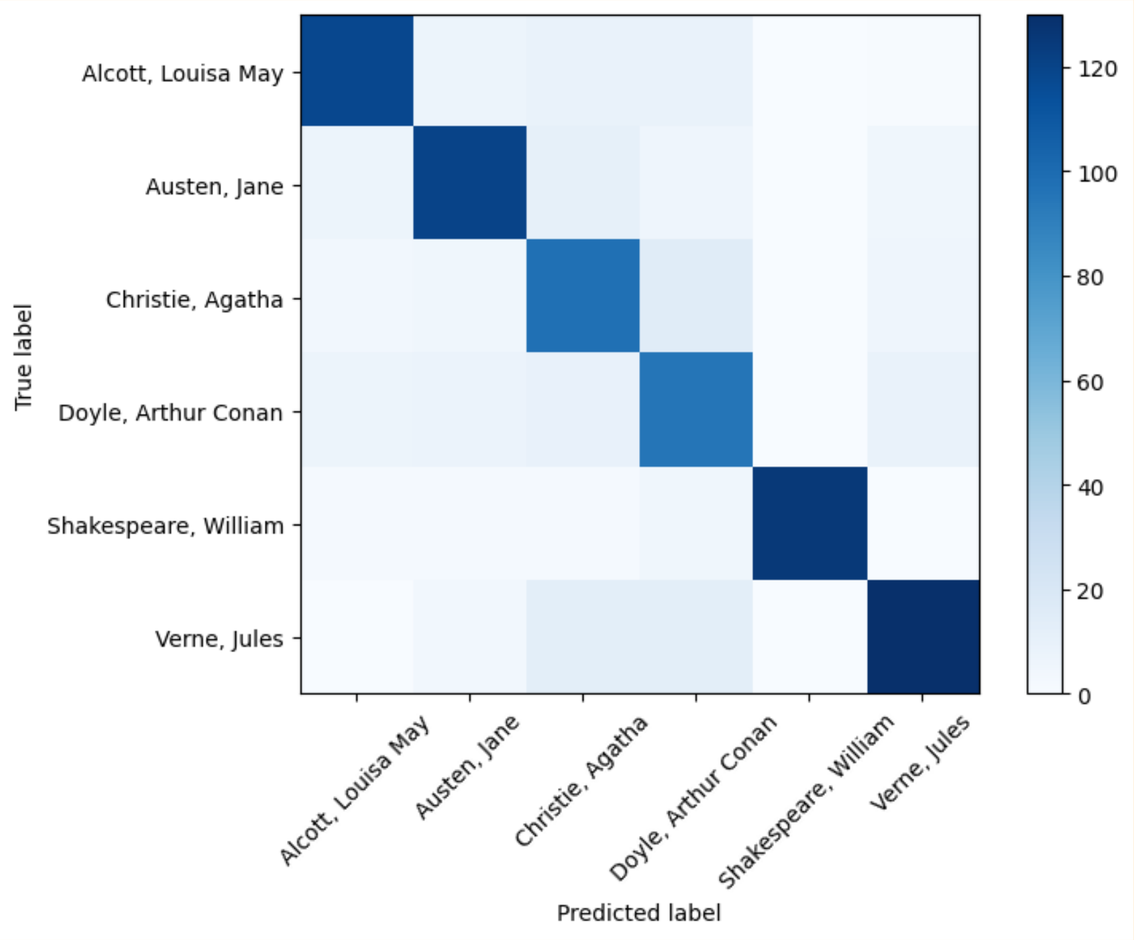
Métricas
Upsampling vs Downsampling

Parámetros	Tiempo 4.4s 118 iteraciones Downsampling 1410	Tiempo 19.1s 221 iteraciones Upsampling 7228
Test metrics	**Accuracy**: 0.831 **Recall**: 0.829 **Precision**: 0.831 **F1**: 0.830	**Accuracy**: 0.921 **Recall**: 0.921 **Precision**: 0.922 **F1**: 0.921
Train metrics	**Accuracy**: 0.933 **Recall**: 0.933 **Precision**: 0.933 **F1**: 0.933	**Accuracy**: 0.952 **Recall**: 0.952 **Precision**: 0.952 **F1**: 0.952

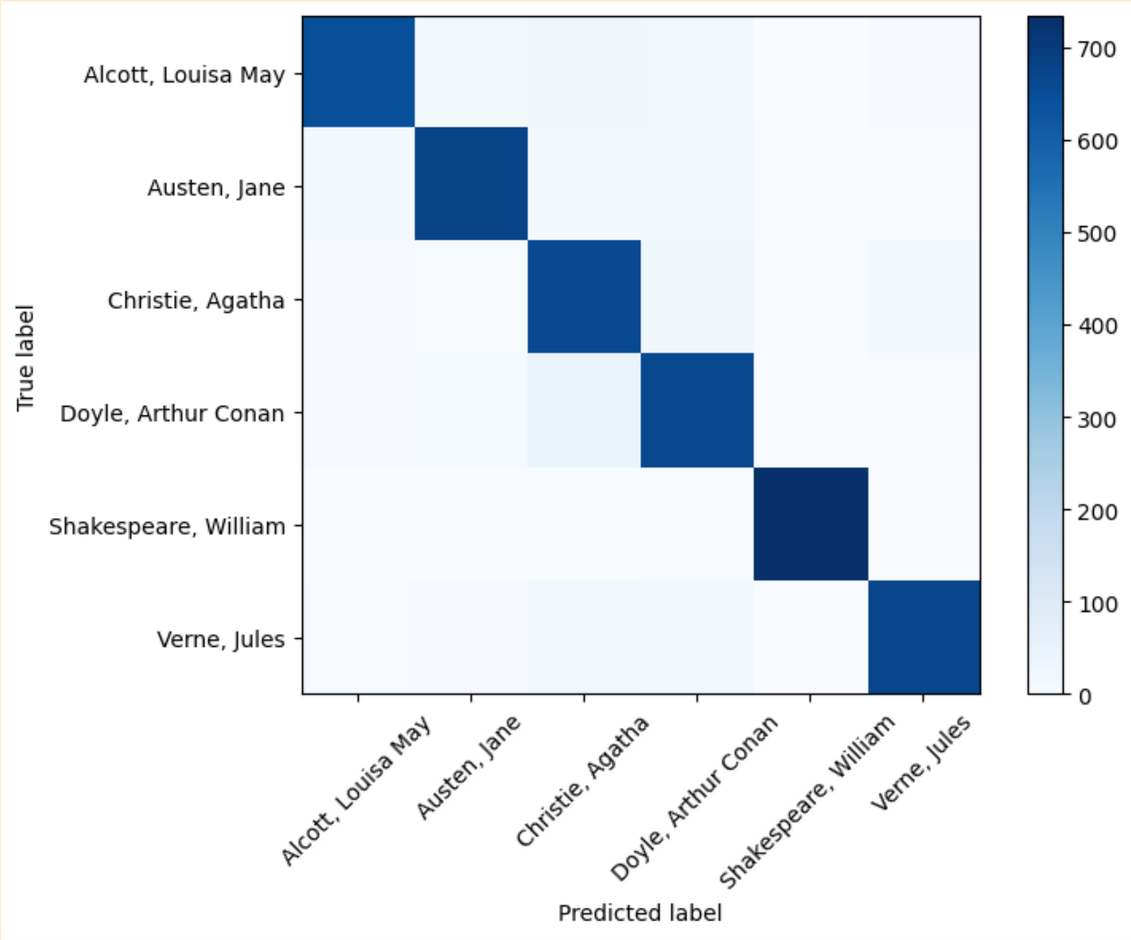
RESULTADOS BOW

Matriz de confusión
Upsampling vs Downsampling

Tiempo 6.6s; 205 iteraciones; Downsampling 1410



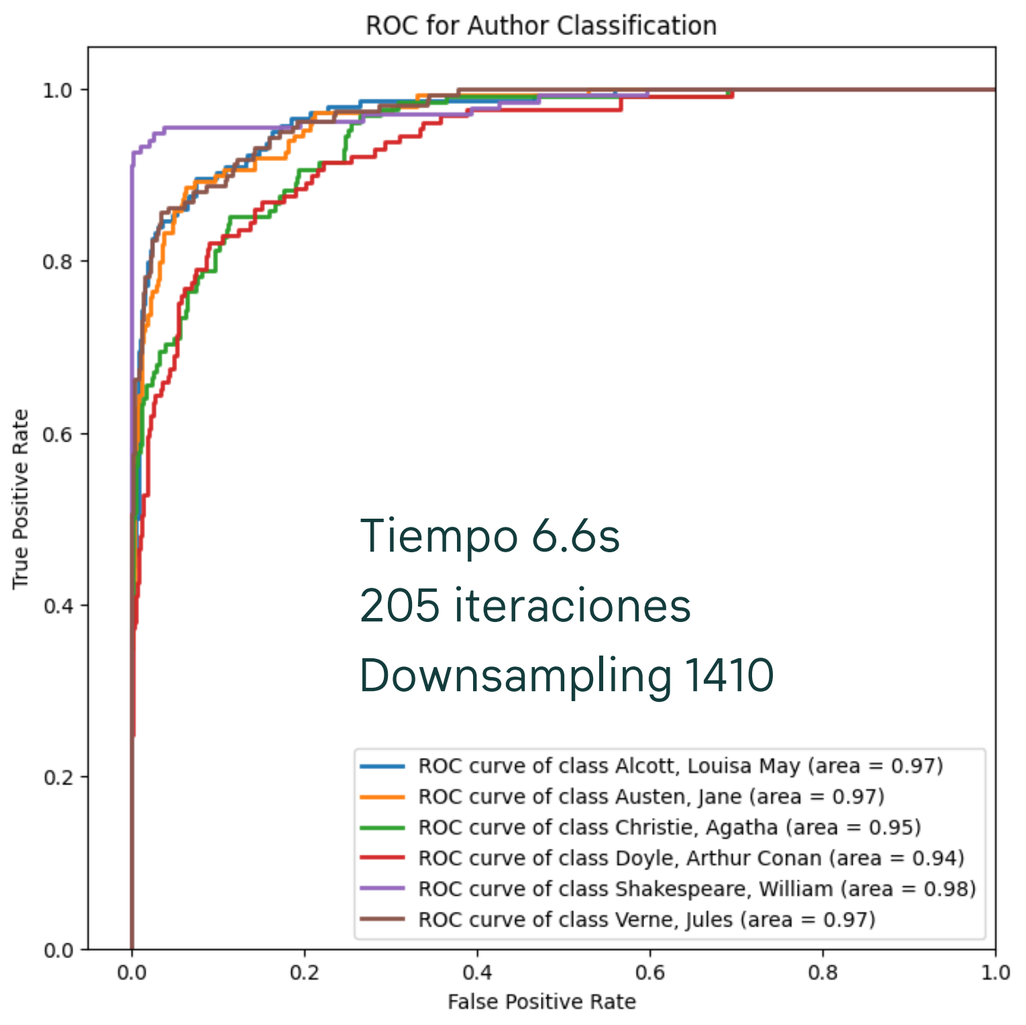
Tiempo 26s; 323 iteraciones; Upsampling 7228



RESULTADOS BOW

Curva ROC

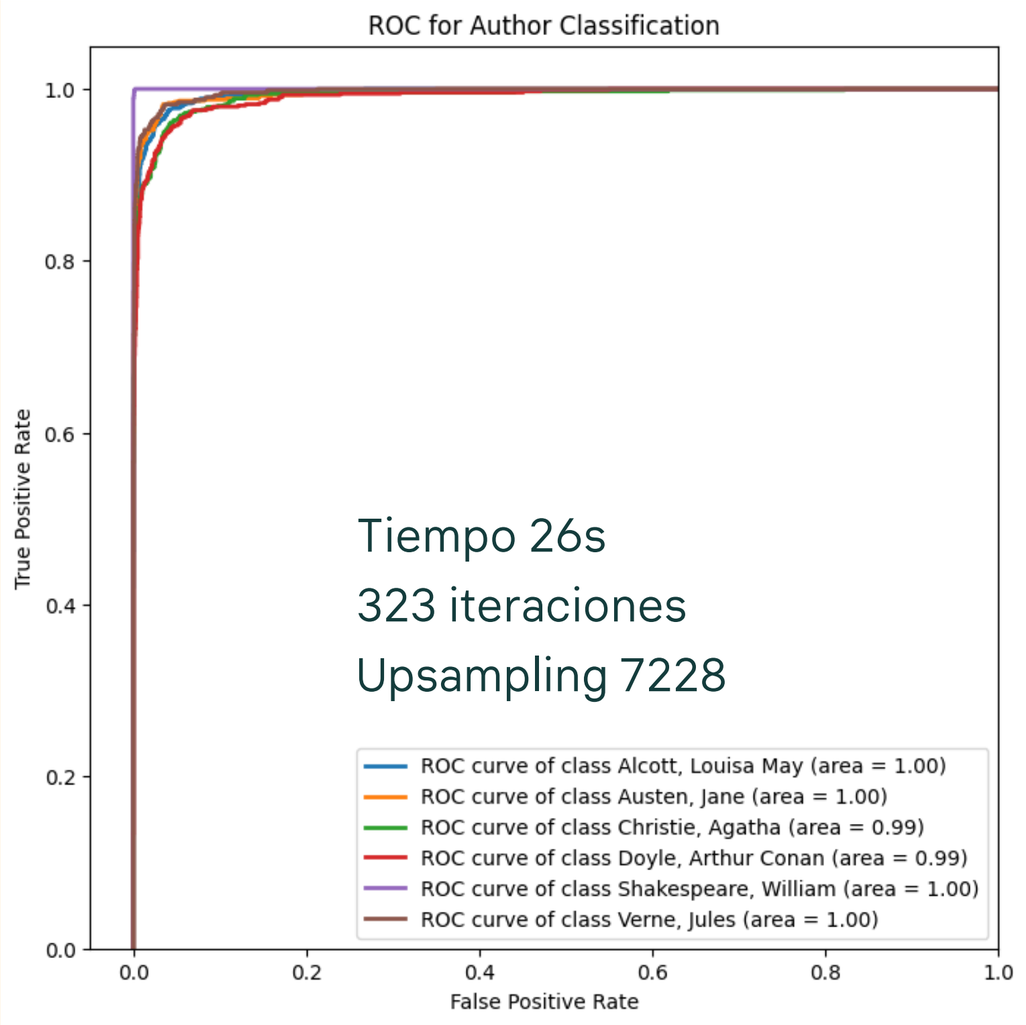
Upsampling vs Downsampling



Tiempo 6.6s

205 iteraciones

Downsampling 1410



Tiempo 26s

323 iteraciones

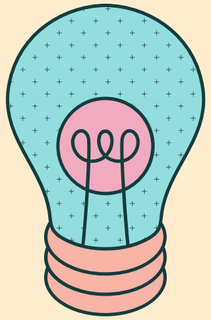
Upsampling 7228

RESULTADOS BOW

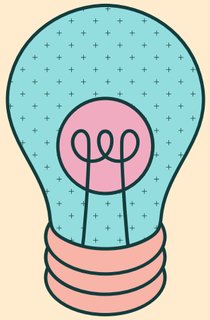
Métricas
Upsampling vs Downsampling

Parámetros	Tiempo 6.6s 205 iteraciones Downsampling 1410	Tiempo 26s 323 iteraciones Upsampling 7228
Test metrics	**Accuracy**: 0.811 **Recall**: 0.810 **Precision**: 0.815 **F1**: 0.811	**Accuracy**: 0.936 **Recall**: 0.935 **Precision**: 0.936 **F1**: 0.936
Train metrics	**Accuracy**: 0.973 **Recall**: 0.973 **Precision**: 0.975 **F1**: 0.974	**Accuracy**: 0.971 **Recall**: 0.971 **Precision**: 0.972 **F1**: 0.971

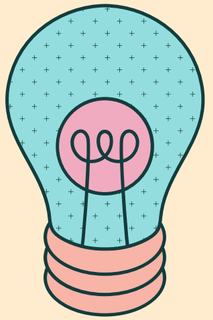
CONCLUSIONES



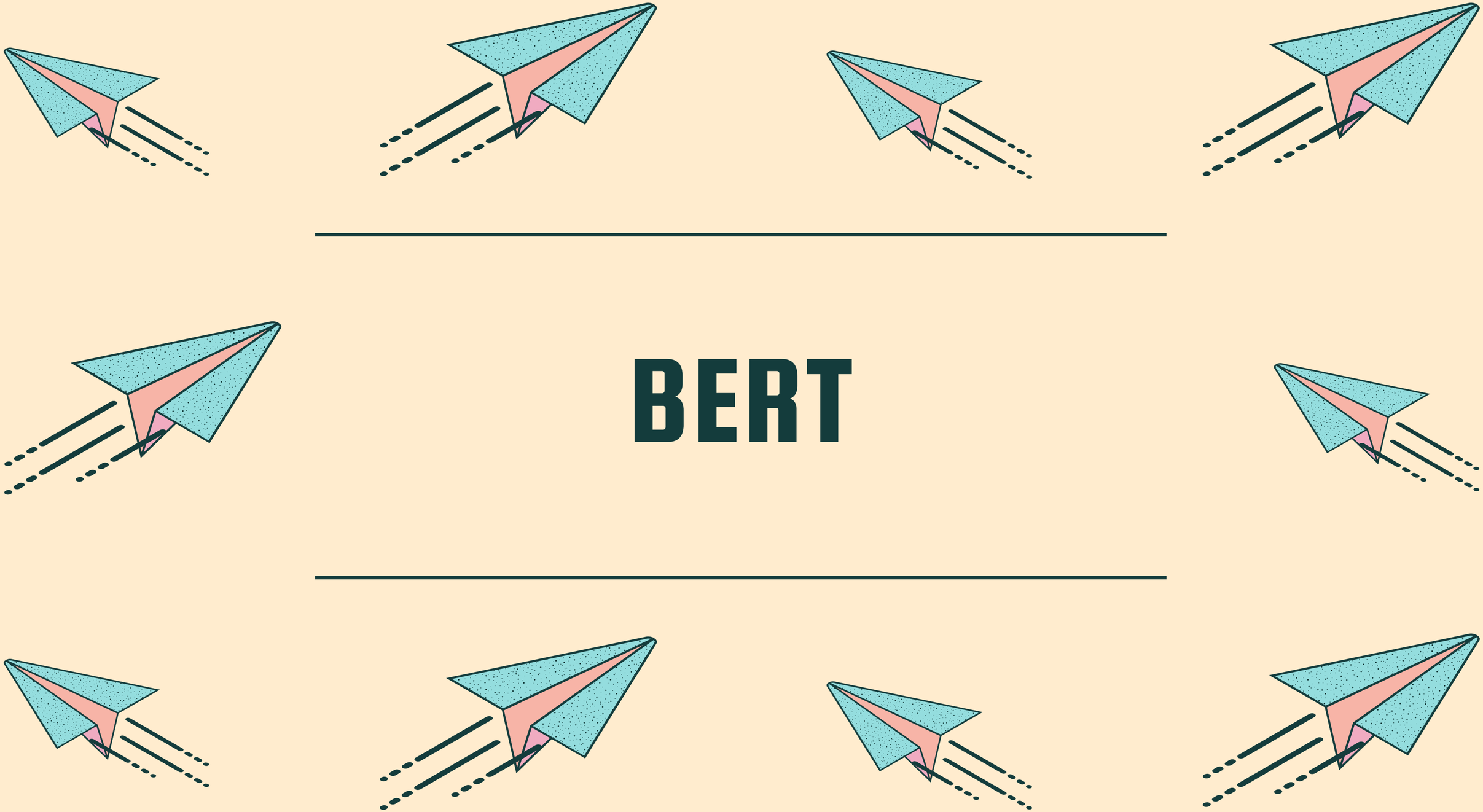
Balanceo con upsampling tiene mejores resultados porque el dataset es más grande.



Ambos modelos tienen resultados muy parecidos para este dataset.



Pueden aprender repetición de frases pero no estructuras sintácticas



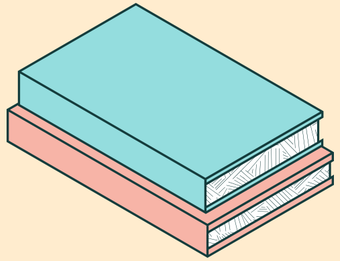
BERT

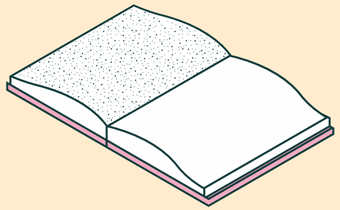
MODELO

Bert

"Bidirectional Encoder Representations from Transformers"

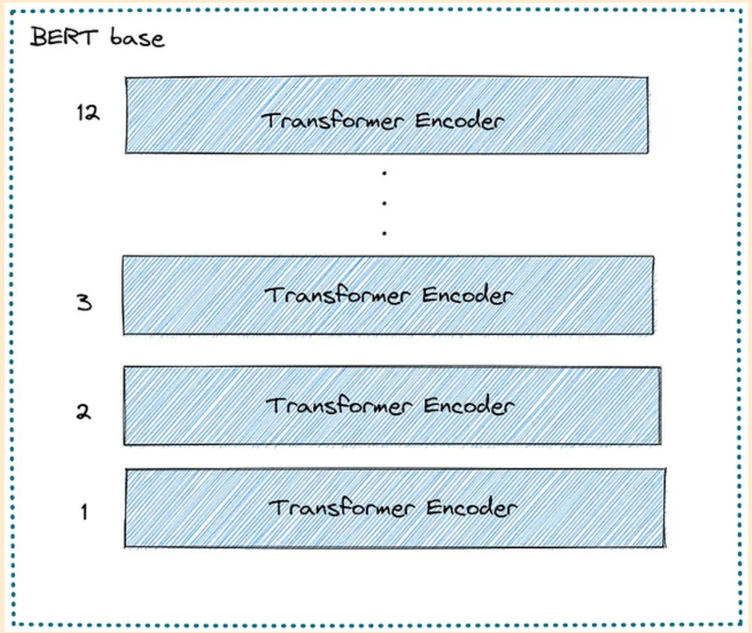
- Modelo desarrollado por Google en 2018
- Se basa en la arquitectura de Transformer
- Pre-entrenamiento: Masked Language Model" (MLM) y "Next Sentence Prediction" (NSP)
- Fine-tuning: Capa extra de clasificación con dataset especifico para la tarea





MODELO

INPUT → BERT → CLASIFICADOR



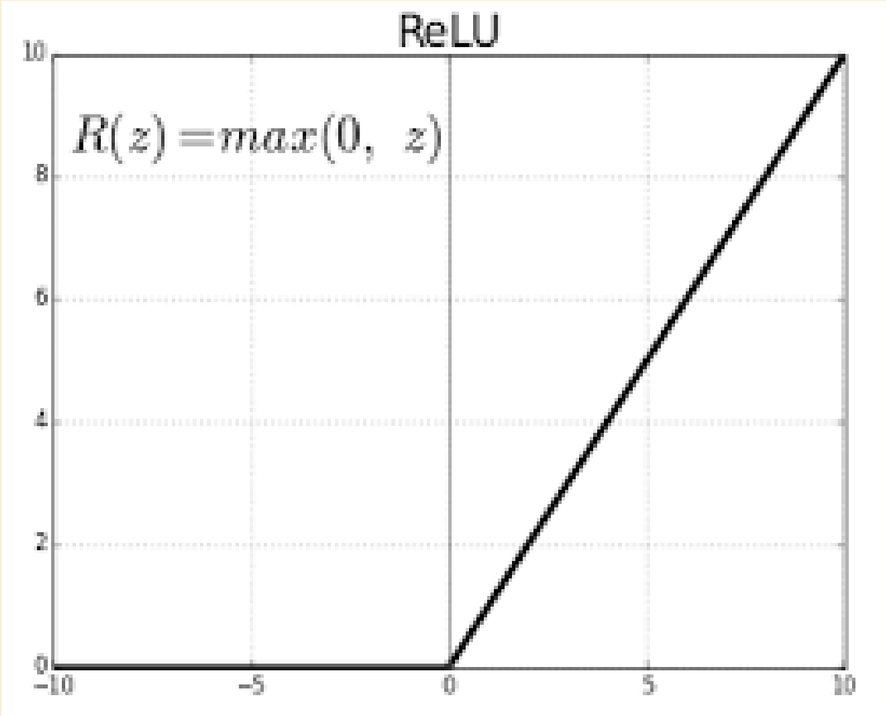
Text classification with BERT in pytorch

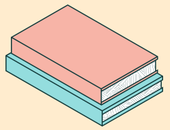
¿Por qué usamos BERT?

- Pre entrenado con un corpus mucho más grande que el nuestro.
- El pre-entrenamiento es bidireccional.

MODELO (fine tuning y RELU)

Usamos una capa de clasificación lineal y una relu sobre el modelo pre-entrenado de BERT (reentrenamos BERT).





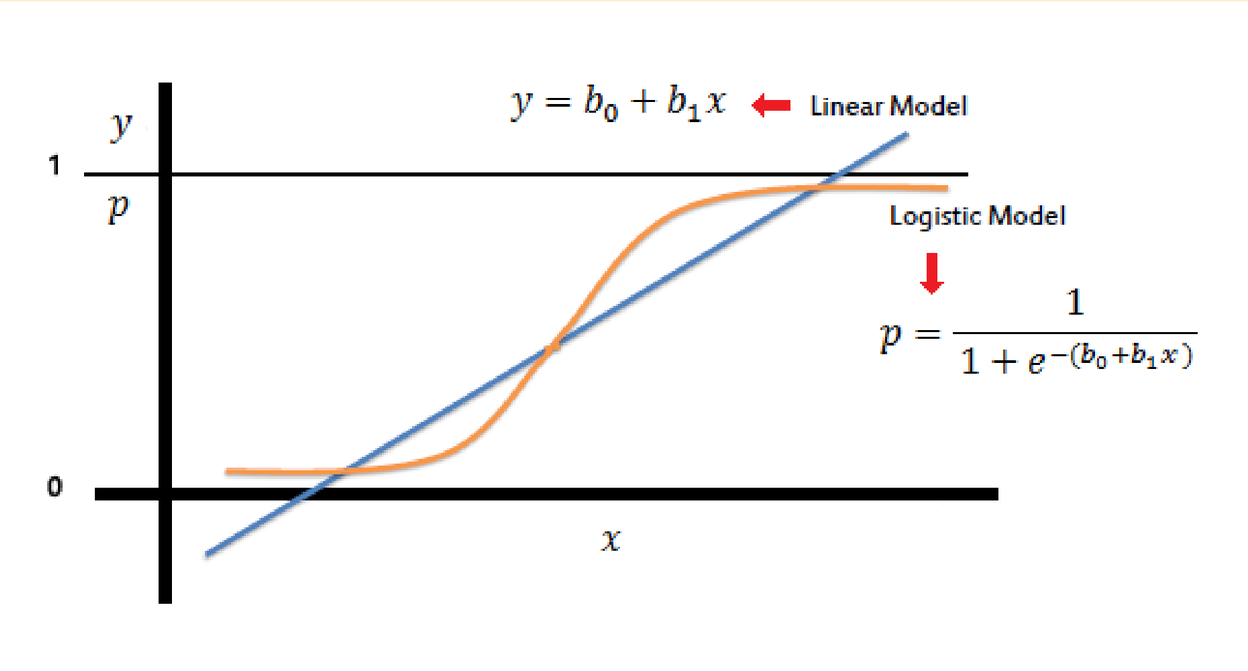
LINEAR + RELU

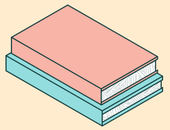
Input: Vector de embedding de [CLS]

Una combinación de una transformación lineal y una función de activación no lineal para clasificar datos en diferentes categorías.

MODELO (Logística)

Usamos una regresión logística sobre los embedding del modelo pre-entrenado de BERT (no reentrenamos BERT).





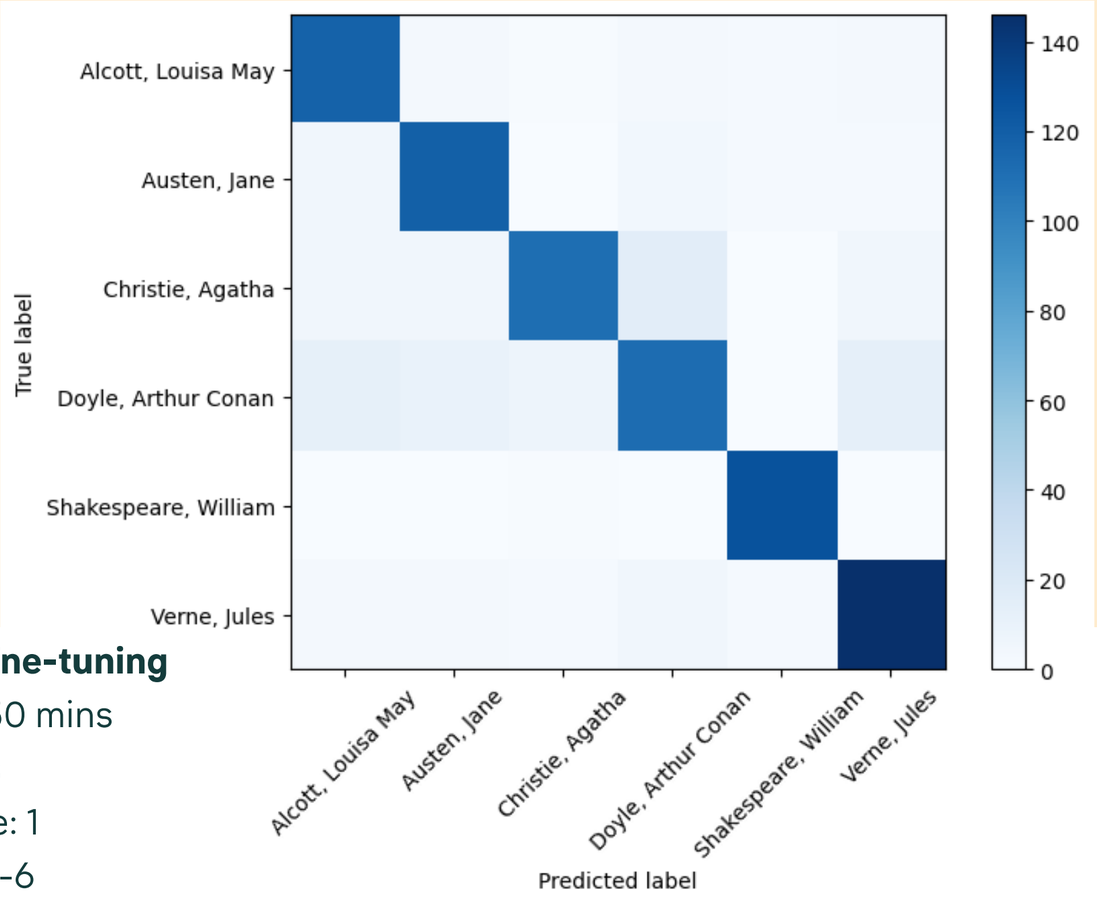
LOGÍSTICA

Input: Vector de embedding de [CLS]

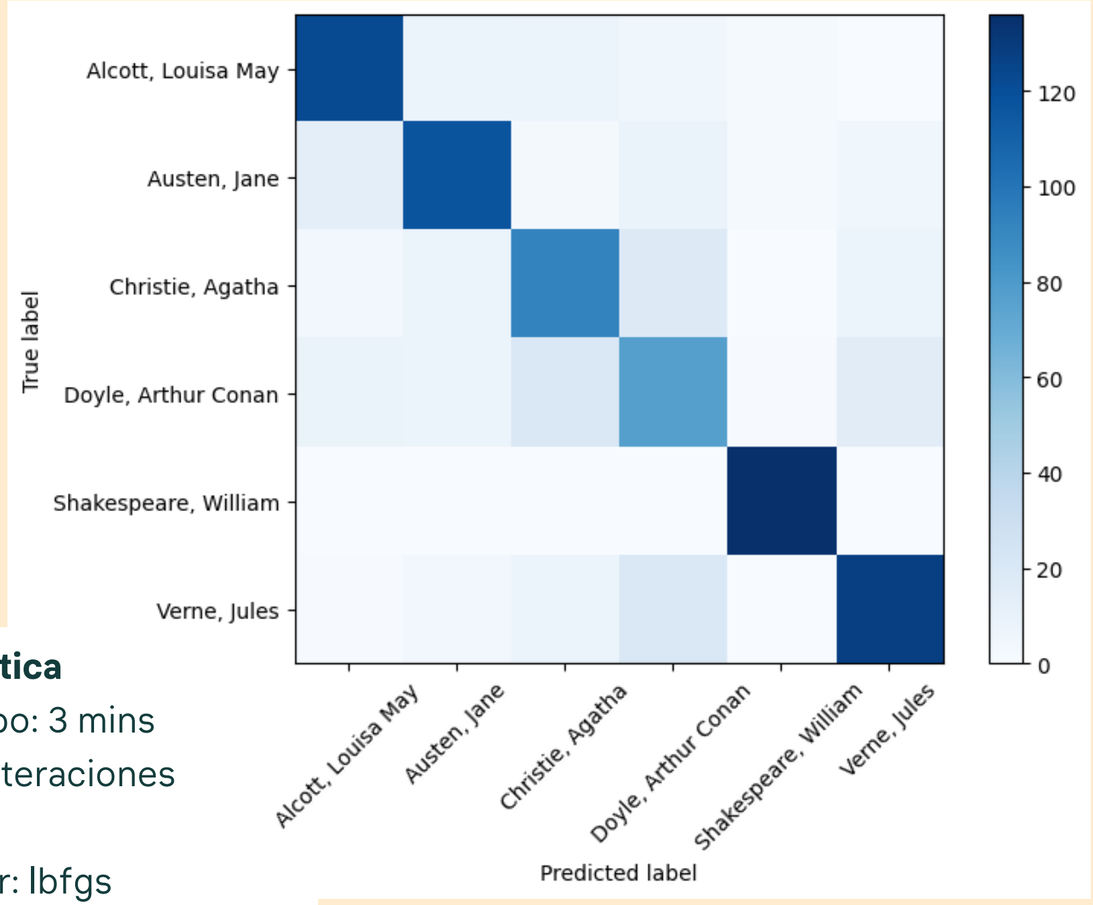
Regresión logística multiclase para clasificar datos en diferentes categorías.

RESULTADOS

Matriz de confusión
RELU y fine-tuning vs Logística



RELU y fine-tuning
Tiempo 50 mins
5 épocas
Batchsize: 1
LR: 1.00e-6
Seed: 42
Balanceado
Downsampling 1410

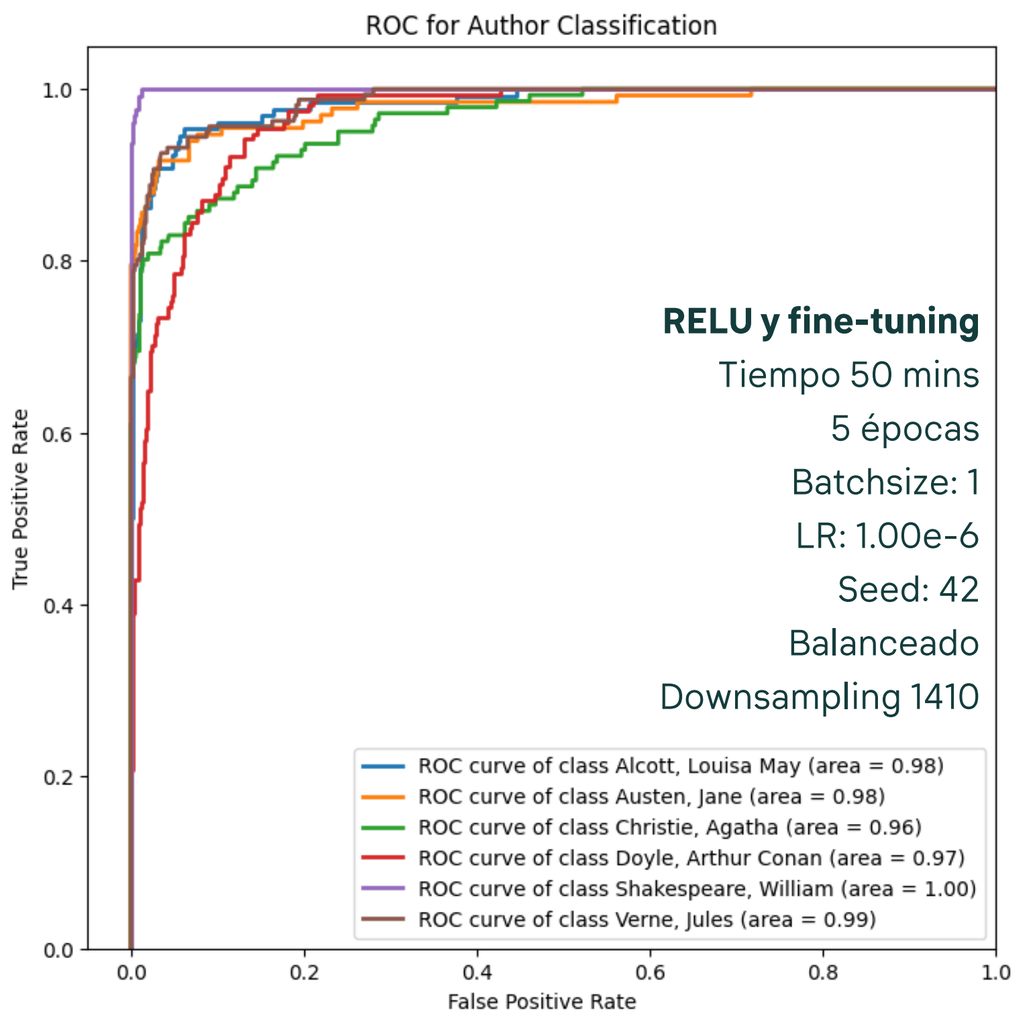


Logística
Tiempo: 3 mins
4614 iteraciones
C: 1
Solver: lbfgs
Balanceado
Downsampling 1410

RESULTADOS

Curva ROC

RELU y fine-tuning vs Logística



RELU y fine-tuning

Tiempo 50 mins

5 épocas

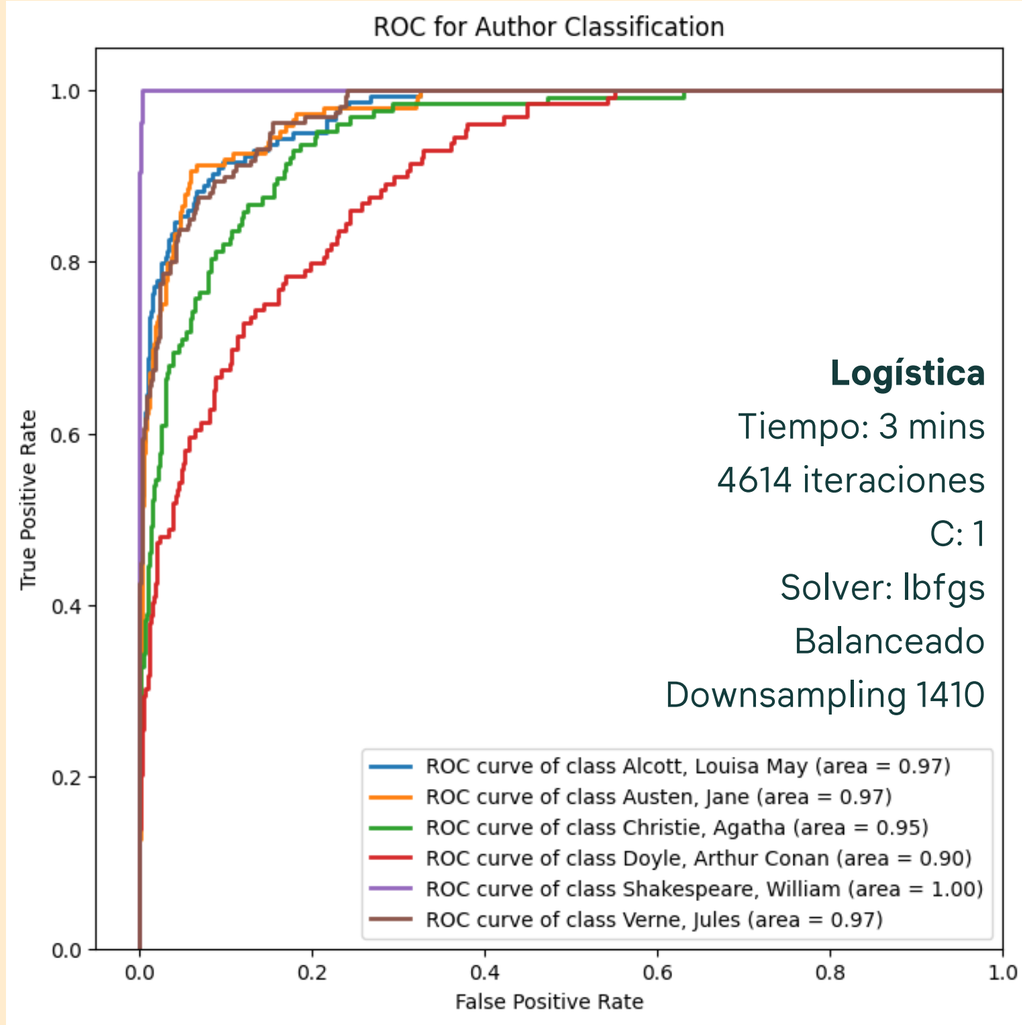
Batchsize: 1

LR: 1.00e-6

Seed: 42

Balanceado

Downsampling 1410



Logística

Tiempo: 3 mins

4614 iteraciones

C: 1

Solver: lbfgs

Balanceado

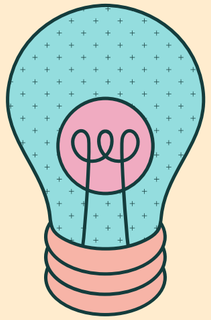
Downsampling 1410

RESULTADOS

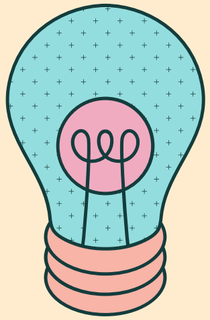
Métricas
RELU y fine-tuning vs Logística

Parámetros	**RELU y fine-tuning** Tiempo 50 mins 5 épocas Batchsize: 1 LR: 1.00e-6 Seed: 42 Balanceado Downsampling 1410	**Logística** Tiempo: 3 mins 4614 iteraciones C: 1 Solver: lbfgs Balanceado Downsampling 1410
Test metrics	**Accuracy**: 0.87 **Recall**: 0.87 **Precision**: 0.87 **F1**: 0.87	**Accuracy**: 0.79 **Recall**: 0.79 **Precision**: 0.79 **F1**: 0.79
Train metrics	**Accuracy**: 0.97 **Recall**: 0.97 **Precision**: 0.97 **F1**: 0.97	**Accuracy**: 0.85 **Recall**: 0.85 **Precision**: 0.85 **F1**: 0.85

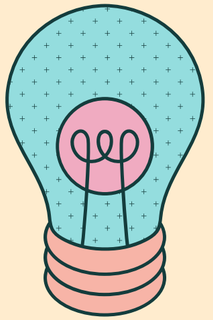
CONCLUSIONES



Reentrenando BERT se está aprendiendo un poco el "estilo" de escritura de cada autor



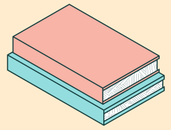
BERT no nos sirvió para utilizar como base sin reentrenarlo



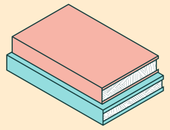
Fine-tuning de BERT con upsampling tardaba 1 hora por época

CONCLUSIONES

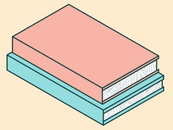
CONCLUSIONES



Pensamos que la versión sin fine-tuning iba a dar mejores resultados que tf-idf porque los embeddings tienen mucha más información



El dataset que elegimos no logra caracterizar a los autores por estilo, sino más bien por las palabras utilizadas



Limitación para aprender estilo en modelos TF-IDF y Bag of Words. Se basa únicamente en frecuencias de palabras pero son mucho más rápidos.

REFERENCIAS

REFERENCIAS

- Text classification with bert in pytorch
- Classify text with BERT | Text | TensorFlow
- Bert 101 - Hugging Face
- Project Gutenberg
- Gutenbergpy