Universidad de la República
Facultad de Ingeniería

Introducción a la ciencia de datos

Tarea 2 - William Shakespeare

Julio de 2024

Juan Montesano
Manuel Padín

Introducción

El presente trabajo es la continuación de la tarea 1. En el mismo se exploran diferentes métodos para clasificar párrafos según el personaje al cual están asociados en una selección de obras de Shakespeare. A lo largo del trabajo se utilizaron técnicas para la vectorización de palabras como lo son Term Frequency-Inverse Document Frequency (TF-IDF) y Bag of Words(BoW). También se exploraron diferentes modelos de clasificación incluyendo Multinomial Naive Bayes y Support Vector Machines (SVM)

El informe también detalla la aplicación de técnicas de validación cruzada para optimizar los hiper parámetros de los modelos.

Se presentan resultados de múltiples entrenamientos y evaluaciones de métricas de los mismos.

Dataset seleccionado

Nuestro dataset está compuesto por párrafos de 3 personajes a los cuales se aplicó una separación estratificada para obtener un dataset de entrenamiento y otro de test.

Personaje	Cantidad total	Train	Test
Antony	253	177	76
Cleopatra	204	143	61
Queen Margaret	169	118	51

Tabla 1 Cantidad de párrafos por personajes

Balance del conjunto train-test

Se realiza el corte estratificado de párrafos de los personajes para el conjunto de train y test.

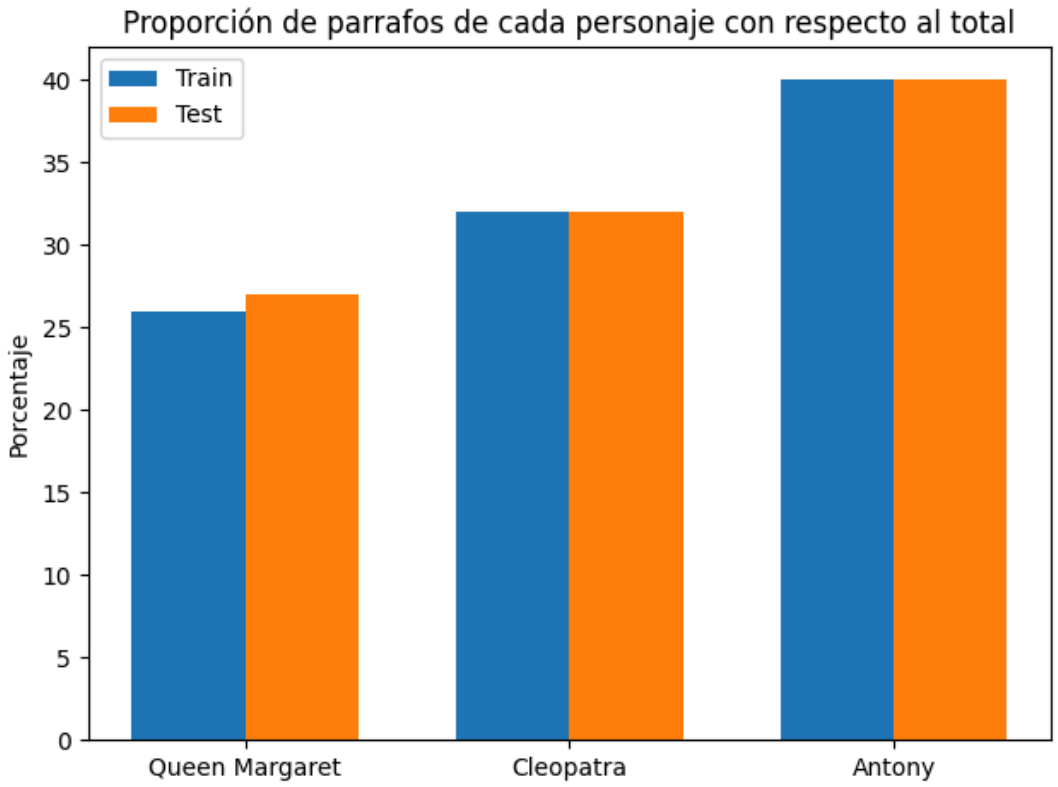

Figura 1 Proporciones correspondientes a cada set

Bag of Worlds

La técnica “Bag of Words” transforma un conjunto de textos en una matriz numérica donde cada fila i hace referencia a cada texto y cada columna j a una palabra diferente, en cada posición de la matriz tendremos la cantidad de ocurrencias de la palabra j en el texto i.
Esta es una técnica para transformar textos en valores numéricos y así poder aplicar técnicas de ciencia de datos.

Ejemplo, para estos dos textos:
“El gato come, el loro no come”
“El perro come”
Tendremos 6 palabras y la matriz resultará como la tabla 2.

El	gato	come	loro	no	perro
2	1	2	1	1	0
1	0	1	0	0	1

Tabla 2 Explicación del método Bag of Words

Es fácil ver cómo puede crecer esta matriz si aumenta la cantidad de palabras y de textos con lo cuál es un problema. Podemos estimar el tamaño de esta matriz para el inglés, Oxford English Dictionary posee 250.000 palabras, si suponemos un conjunto de 100.000 textos y si cada posición de la matriz utiliza 4 bytes para representar el entero, necesitaremos 93 GB de memoria RAM.
Por esta razón se utiliza la estructura sparse matrix para almacenarla ya que muchos valores serán 0 porque los textos contienen un conjunto reducido de palabras del inglés.

TF-IDF

Los N-grama son una secuencia consecutiva de N gramas, en lenguaje natural los gramas pueden ser letras o palabras. Esto es útil porque tenemos información de la relación entre los gramas cercanos, para lenguaje natural podemos construir modelos probabilísticos como por ejemplo qué palabra continúa a una secuencia de N palabras. Con Bag of Words esta información de la relación entre palabras cercanas se pierde.

Tf-idf nos da una idea de qué tan relevante es un término en un documento con relación a todos los demás documentos. Es directamente proporcional a la cantidad de ocurrencias del término en el documento e inversamente proporcional a la cantidad de documentos que contiene el término. Por lo tanto un tf-idf alto indica que el término es importante en el documento actual porque aparece mucho en el mismo pero no en otros, por el contrario un tf-idf bajo indica que el término es poco relevante porque aparece poco o nada en el documento actual o porque aparece mucho en otros documentos. Un término puede ser una palabra, n-grama, etc.

PCA

Procedemos a usar PCA para explorar nuestro data set. En la figura 2 mostramos un PCA con idf=False, stop_words=False, ngram_range=(1,1), con esta configuración tenemos 2.807 features (cantidad únicas de palabras).
Se observa que es muy difícil poder clasificar cada personaje con alguna técnica de separación del espacio, los párrafos de cada personaje están mezclados en el espacio.

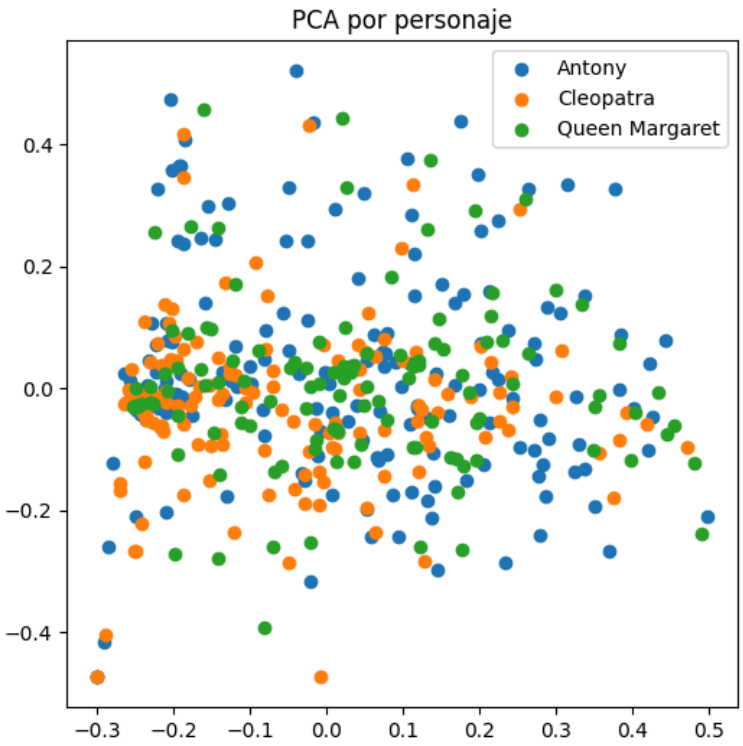

Figura 2 PCA con idf=False, stop_words=False, ngram_range=(1,1)

En la figura 3 podemos ver un PCA con idf=True, stop_words='english', ngram_range=(1,2), con esta configuración tenemos 8.489 features, aumentó por la utilización de bi-gramas. Aún se observa difícil separar cada personaje en el espacio en 2D aunque se comienza a agrupar en sectores, Queen Margaret posee una agrupación diferente con respecto a los otros dos personajes.

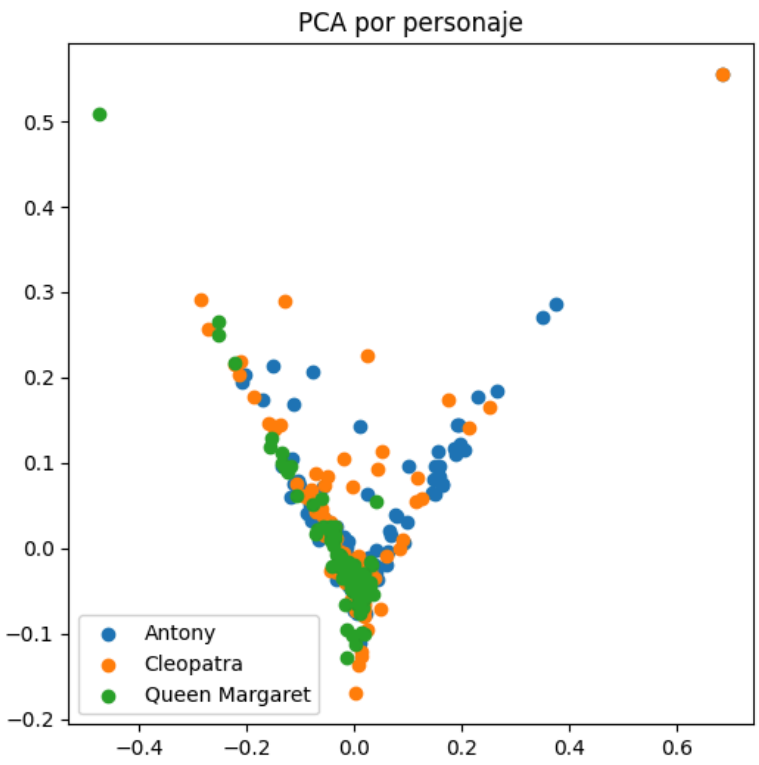

Figura 3 PCA idf=True, stop_words='english', ngram_range=(1,2)

Tal vez con solo 2 dimensiones no hay suficiente información para separar los personajes, en la figura 4 podemos ver la varianza explicada con 10 componentes de PCA , con los dos primeros no llegamos al 2% de la información, los otros 8.487 componentes explican el 98%.

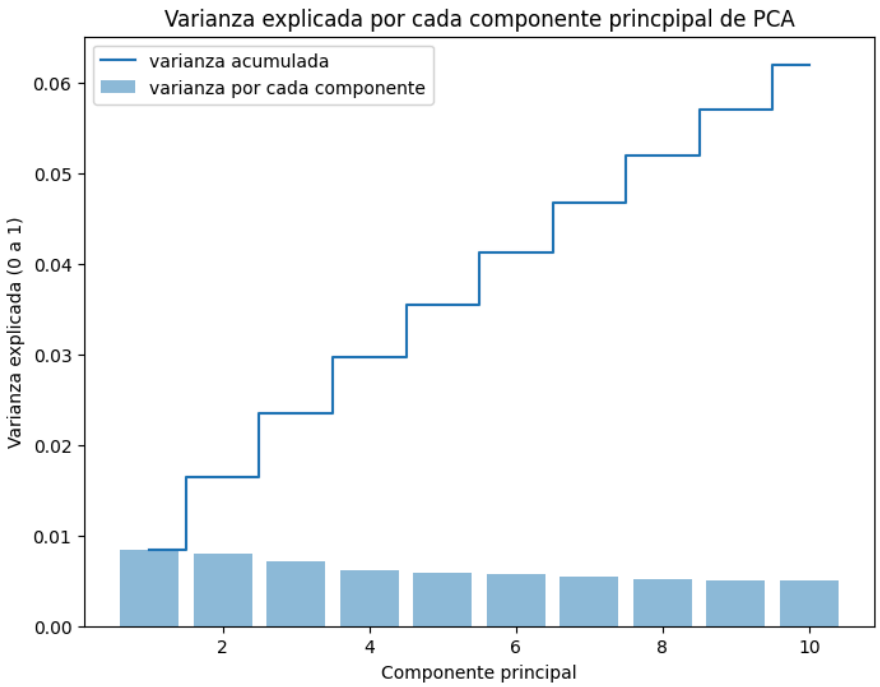

Figura 4 Varianza por componente de PCA

MNB

Para evaluar los modelos tendremos como referencia la figura 5 que nos enseña una matriz de confusión.

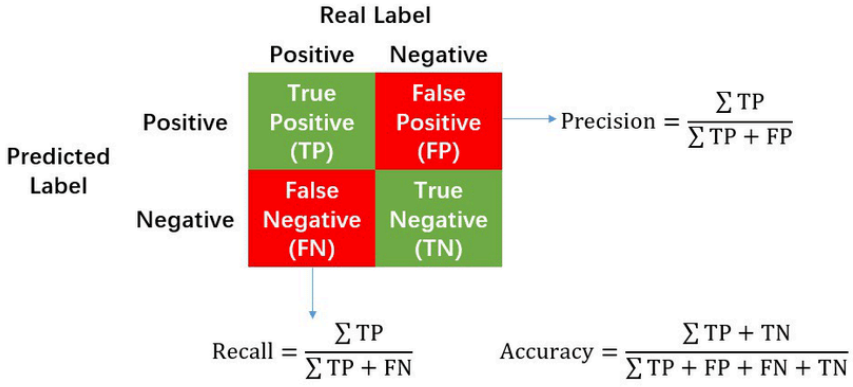

Figura 5 Explicación de matriz de confusión

Para clasificar los 3 personajes decidimos entrenar un modelo Multinomial Naive Bayes que arrojó las siguientes métricas:

Train accuracy: 96%
Test accuracy: 57 %

Clase / Personaje	Precision	Recall
Cleopatra	70%	34%
Queen Margaret	95%	35%
Antony	50%	91%

Tabla 3 métricas por categoría MNB

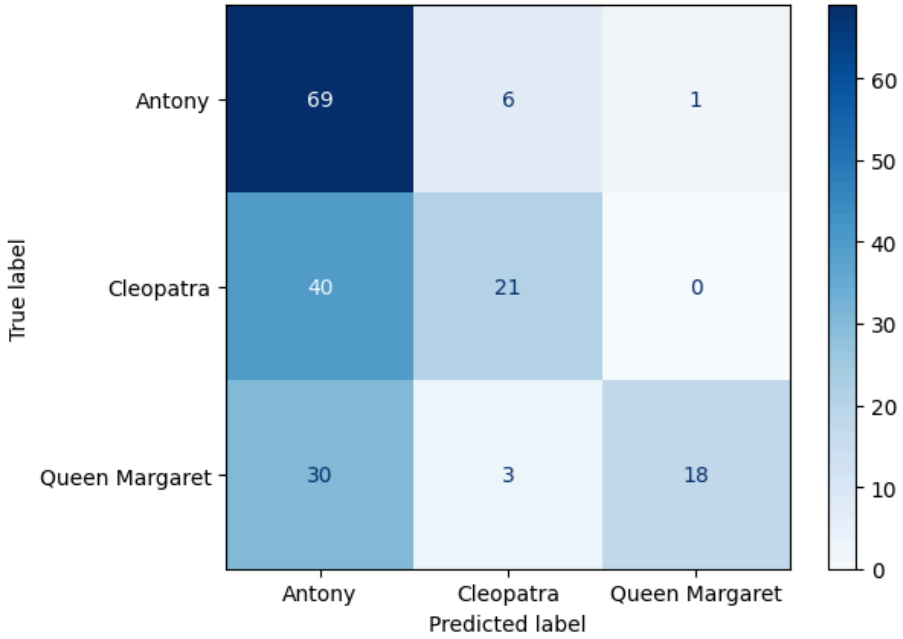

Figura 6 Matriz de confusión para

La precision nos da una métrica de lo bueno que fue prediciendo los valores positivos mientras que recall sobre el cubrimiento de los positivos.
Para Antony el recall es bueno ya que detectó al 91% de su clase correctamente pero a costa de su precisión la cuál fue del 50%.
Para Queen Margaret sucedió lo contrario, solo el 35% de su clase pudo clasificar, en cambio fue muy preciso porque acertó al 95% de la clase positiva.

Si utilizamos solo accuracy no estamos contabilizando la buena o mala predicción en una clase con pocas instancias, por ejemplo si tenemos la clase A con el 99% de las instancias y B con solo el 1%, si nuestro modelo es tonto y clasifica todo como A tendremos un accuracy de 99% y pensaremos que funciona muy bien cuando claramente no.

Este problema es muy común cuando queremos clasificar clases que ocurre con una probabilidad baja pero la queremos detectar (ejemplo una bomba en aeropuerto, enfermedad grave en pacientes, deserción estudiantil, etc).

Cross-validation

Cuando tenemos un dataset y queremos realizar varios experimentos en el mismo, por ejemplo para entrenar y validar varios modelos de ML, para aumentar la independencia de los datos podemos utilizar la técnica de Cross-validation.
Cross-validation genera K sub-conjuntos (folds) de datos iguales con lo cuál podemos realizar K experimentos con diferentes dataset de train y test. Esto es posible eligiendo un subconjunto como test y el resto como train.

Se probó 7 conjuntos de hiper-parámetros diferentes con k-fold = 4.
Se ve que los experimentos 2 y 3 son los que mejor accuracy alcanzaron, ambos utilizando english como stop word y ngram de 1 (solo palabras), el 3 levemente mejor el cuál tiene idf=True.

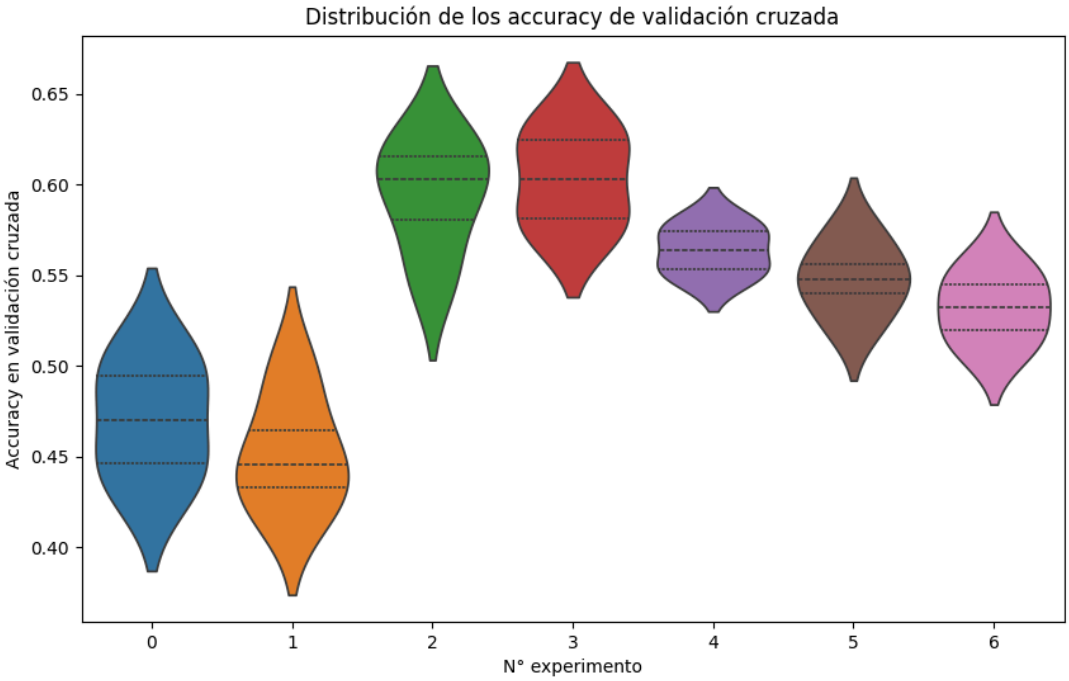

Figura 7 Accuracy en validación cruzada de MNB

MNB optimizado

Volvemos a entrenar y validar con la mejor configuración evaluada a través del cross validation:

- stop_words: 'english'
- ngram: (1,1)
- idf: True

Lo cuál arrojó las siguientes métricas:

Train accuracy: 92.9%
Test accuracy: 58.0 %

Clase / Personaje	Precision	Recall
Cleopatra	62%	34%
Queen Margaret	95%	41%
Antony	51%	88%

Tabla 4 métricas por categoría MNB optimizado

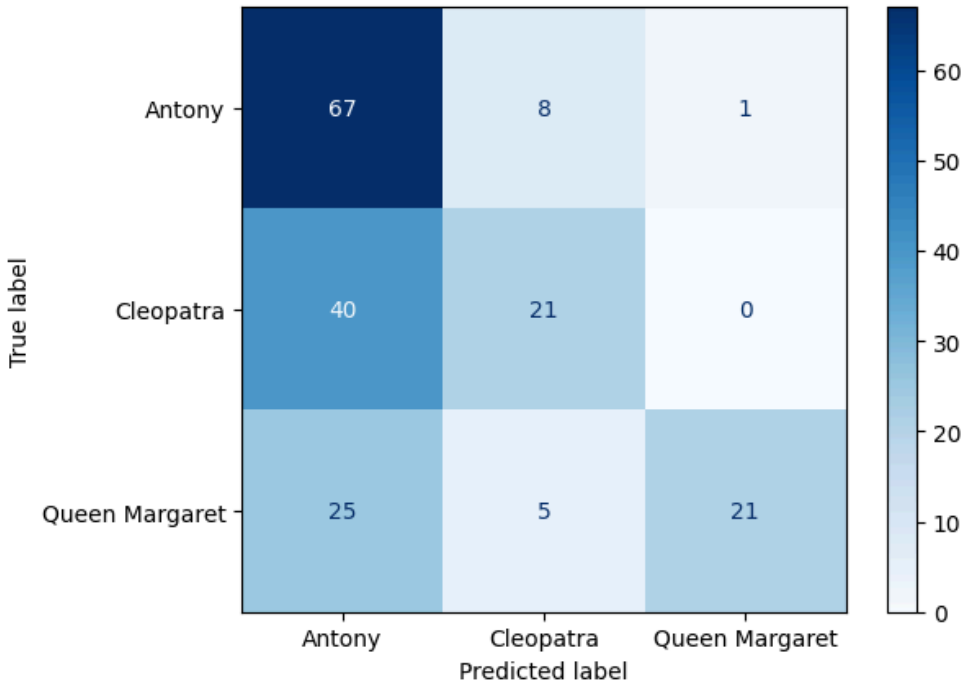

Figura 8 Matriz de confusión MNB optimizado

Podemos observar que la mejora alcanzada con los mejores hiper-parámetros para Naive Bayes es levemente superior al de la parte 2.1, esto nos sugiere que es una limitación de Naive Bayes o de los features extraídos (tf-idf).
Bag of words y tf-idf contabilizan palabras o segmentos muy cortos de oraciones (n-gramas) lo cual nos está haciendo perder información de cómo un personaje desarrolla un párrafo completo y cómo se relacionan las palabras en él.

SVM

Para este entrenamiento usamos SVM para clasificar el texto de los personajes. Esta técnica es muy popular ya que maneja muy bien datos de alta dimensión sin perder mucha eficacia. Al estar usando TF-IDF estamos representando al texto con una dimensionalidad bastante alta por lo que nos pareció apropiado usar esta técnica.

Las técnicas de SVM construyen un conjunto de hiperplanos en un espacio de alta dimensionalidad permitiéndonos en este caso clasificar información, también tienen la particularidad de trabajar bien con datos aislados como es nuestro caso al tratarse de una sparse-matrix en el caso del texto procesado.

Como se ve en la figura 9, usamos diferentes valores con l a función GridsearchCV para encontrar los parámetros que nos devolvieran los mejores resultados de prueba en la validación cruzada.
Los parámetros explorados con este método fueron los siguientes:

- Kernel:Define la función de transformación de los datos
- Stop words: Que palabras son tenidas en cuenta como stop words
- Ngram: El tamaño de los n gramas
- Idf: Si se emplea idf o no

También se exploró en un estudio paralelo modificar el parámetro C habiéndose explorado los siguientes valores: 0.1, 1, 10, 100 y 1000. Obteniendo mejores resultados con C=1 que es el que viene por default

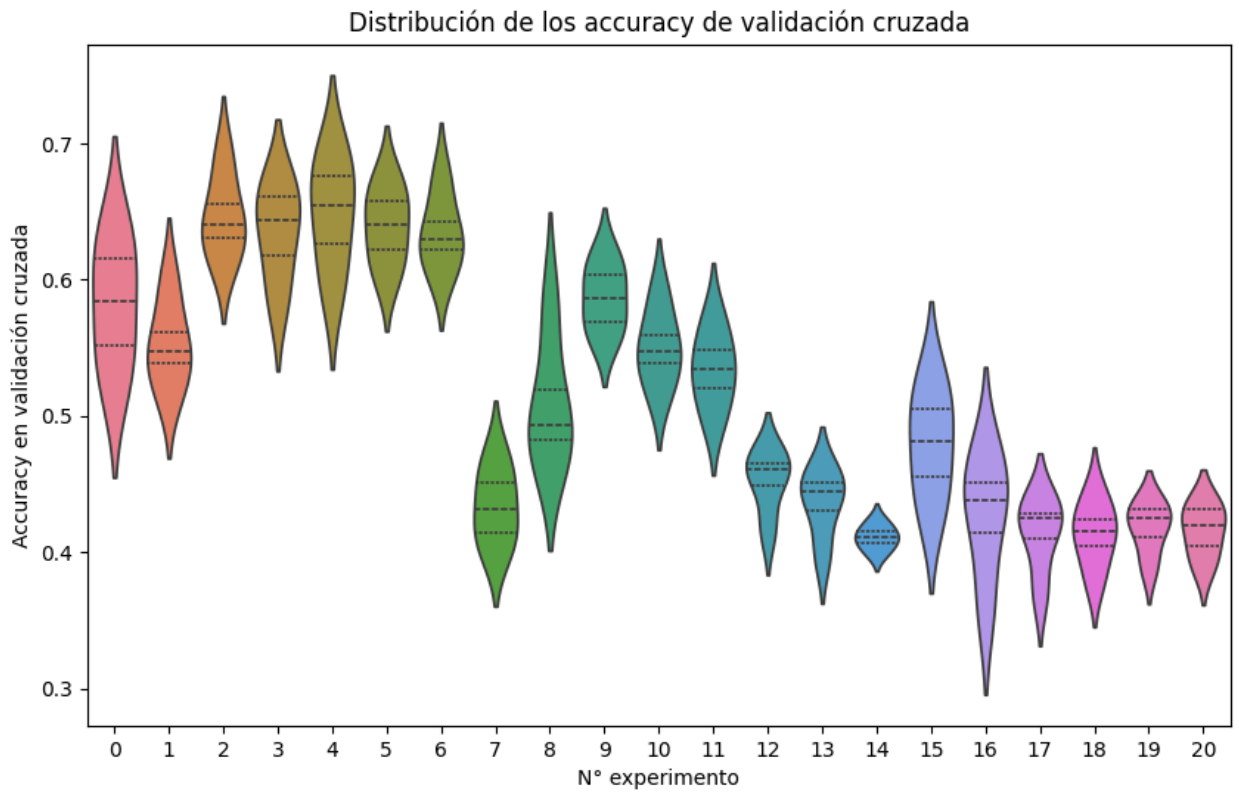

Figura 9 Accuracy en validación cruzada de MNB

Los parámetros que terminaron obteniendo los mejores resultados son:

- kernel: linear
- stop_words: english
- ngram: (1, 2),
- idf: False

Obteniendo:

Train Accuracy: 96,3%
Test Accuracy: 63,3%

Clase / Personaje	Precision	Recall
Cleopatra	54%	44%
Queen Margaret	85%	68%
Antony	58%	75%

Tabla 5 Métricas de SVM optimizado

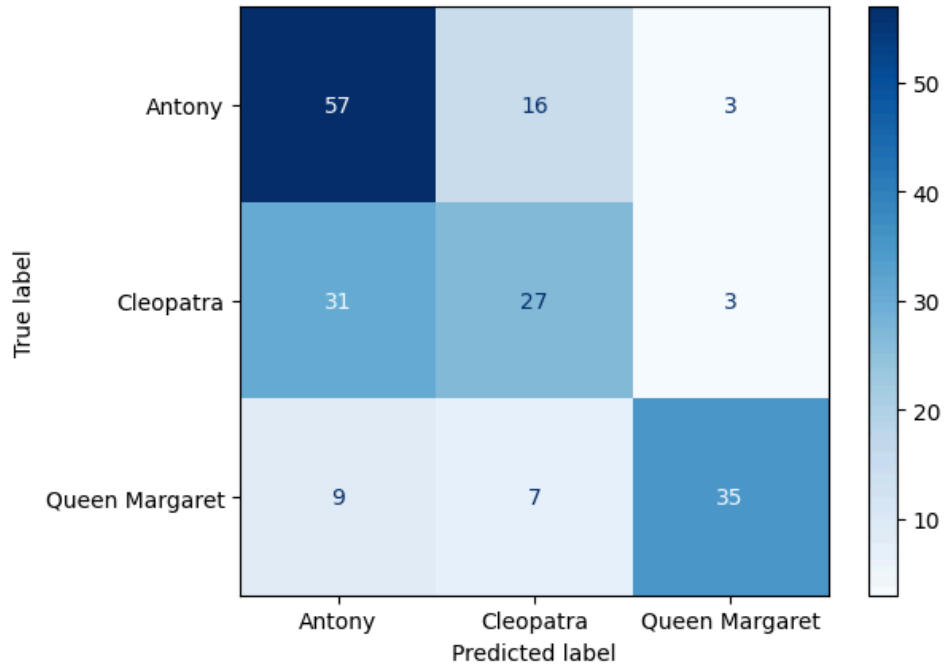

Figura 10 Matriz de confusión para SVM optimizado

En este entrenamiento con SVM optimizado obtuvimos mejoras significativas con respecto a MNB optimizado. Es importante notar que si bien la precisión disminuyó para el caso de Cleopatra y Queen Margaret mejoró significativamente el Recall, En el caso de Antony se dió al revés, mejoró la precisión y empeoró el Recall. Teniendo todos los cambios en cuenta nos encontramos con un modelo más equilibrado y efectivo para clasificar los párrafos habiendo obtenido más poder de generalización y precisión

Cambio de personaje

Para analizar este caso escogimos al personaje Poet porque contiene muchos párrafos y creemos que la estructura del mismo es muy diferente a los otros dos.
El algoritmo alcanzó 68% de accuracy en test pero si analizamos la matriz de confusión vemos cómo el desbalance en cantidad de párrafos afectó la correcta clasificación del modelo. El modelo “aprendió” a clasificar siempre como Poet, haciendo esto es capaz de alcanzar un buen accuracy pero tendrá un pésimo resultado en la clasificación de los otros dos personajes.
Técnicas de submuestreo nos pueden ayudar a igualar hacia abajo la cantidad de párrafos de Poet y que los tres personajes estén en igualdad de condiciones. Si tenemos pocos datos y no podemos perderlos podemos aplicar técnicas de data augmentation la cual iguala hacia arriba la cantidad de párrafos de los otros dos, hay que tener cuidado cómo se generar dichos datos, deben ser lo más fiel a los datos reales y no caer en el error de duplicarlos.

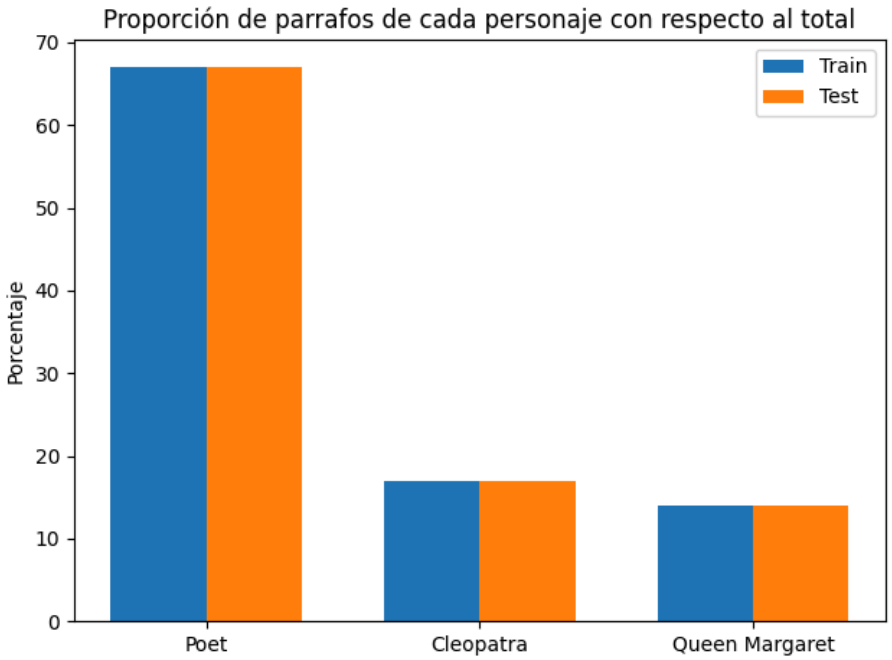

Figura 11 Proporción de párrafos con personaje nuevo

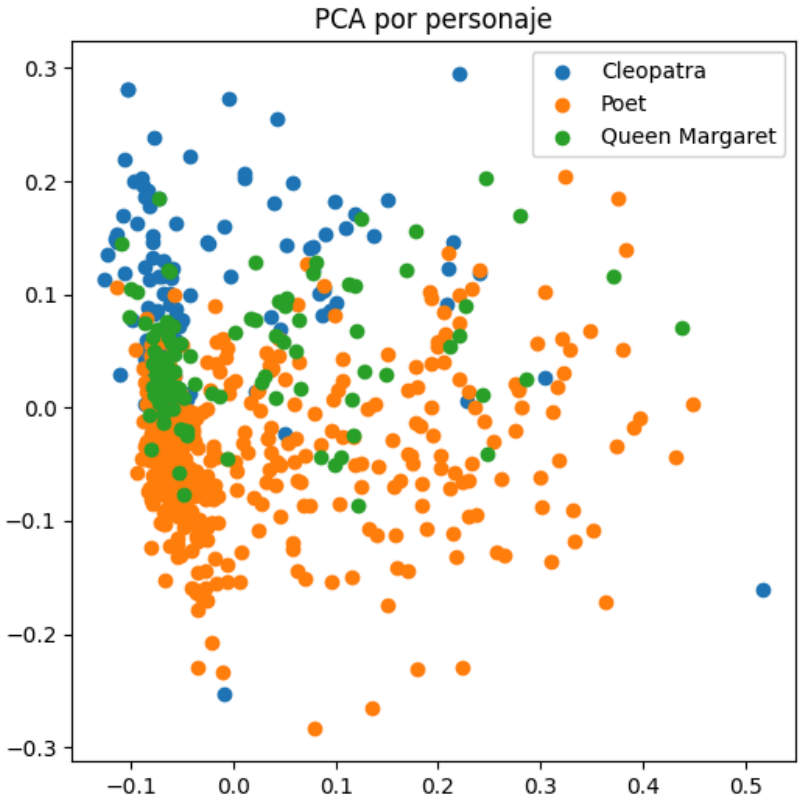

Figura 12 PCA con nuevo personaje

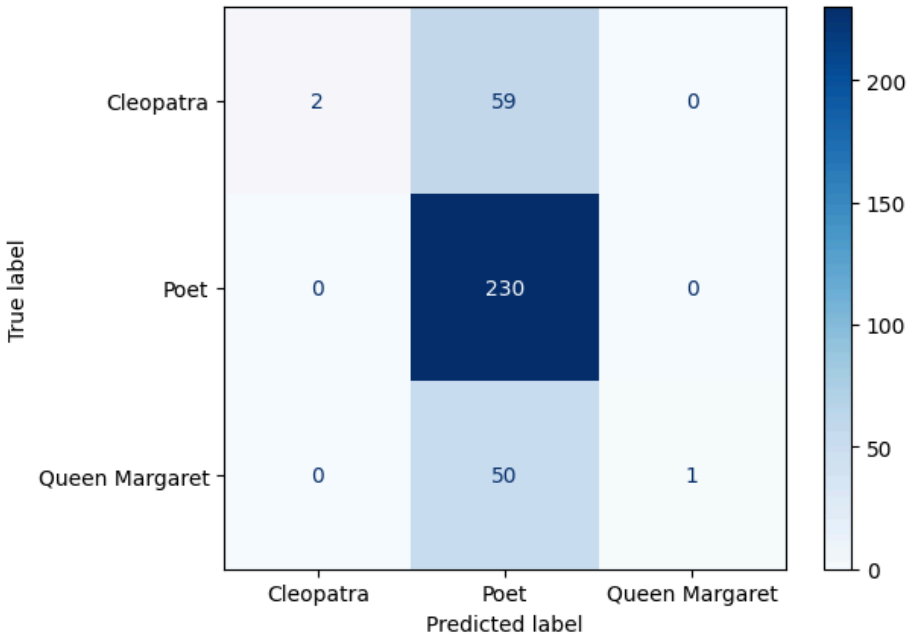

Figura 13 Matriz de confusión con nuevo personaje

Otro análisis que nos pareció interesante es encontrar la combinación de 3 personajes la cuál alcance el mayor accuracy en test para el modelo Naive Bayes, los personajes fueron Vincentio, Henry VI y Brutus. Creemos que estos tres personajes tienen una forma de expresarse muy diferente entre ellos y por esa razón este resultado, es un ejemplo práctico de cómo utilizar Aprendizaje Automático aplicado a la ciencia de datos y extraer conocimiento.

Train accuracy: 98.6%
Test accuracy: 83.8%

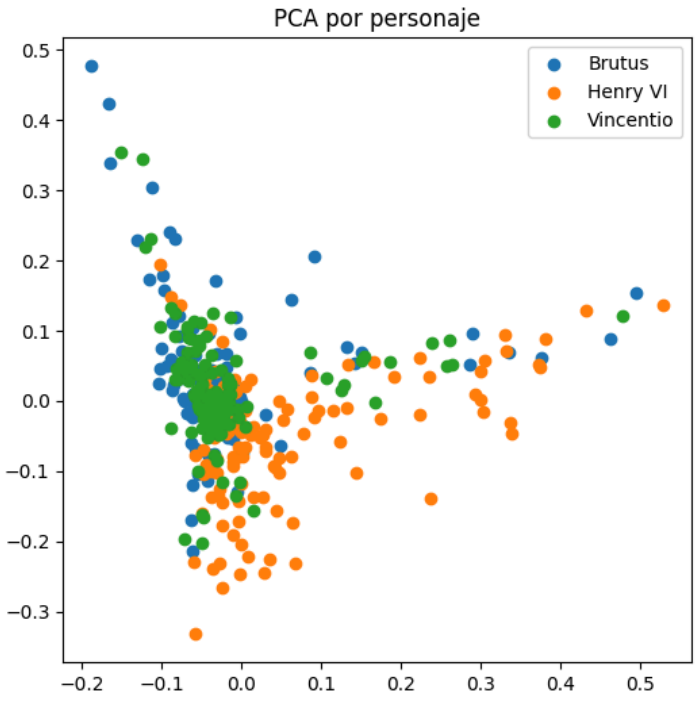

Figura 14 PCA con mejores personajes

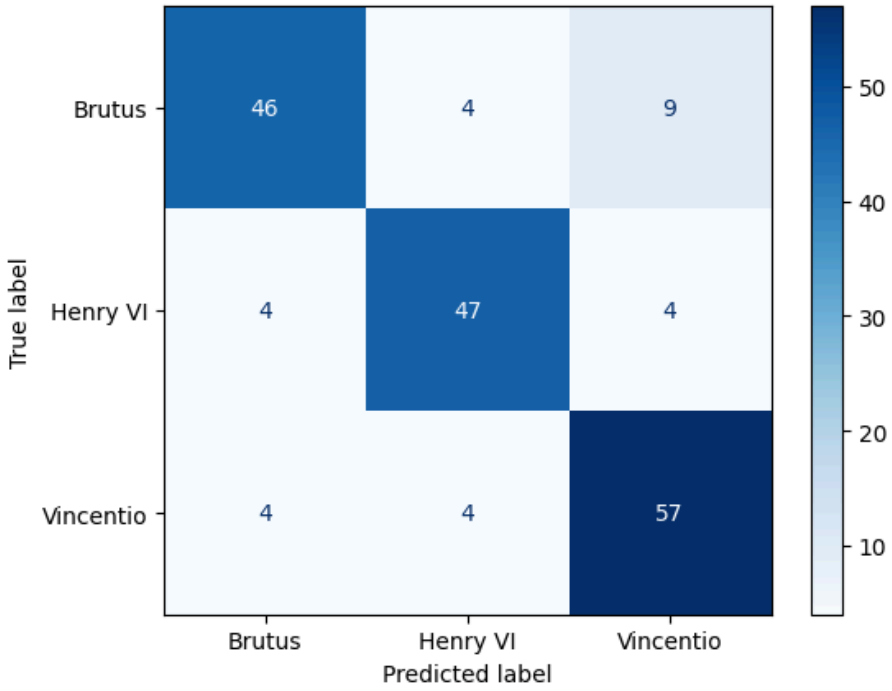

Figura 15 MNB optimizado

Otras técnicas

Si bien TF-IDF es una herramienta poderosa existen otras técnicas que pueden extraer más características de los textos y son empleadas ampliamente en análisis de sentimientos y traducciones automáticas

Una de estas técnicas es GloVe, la misma usa información de la co-ocurrencia de las palabras a lo largo de un texto, la misma construye una matriz de co-ocurrencia de palabras y luego usa propiedades de matrices para el embebido.

La principal ventaja es la sensibilidad al contexto global que presenta el algoritmo, no siendo necesario que se escriban palabras cerca para darse contexto entre ellas, Otra ventaja importante es la captura de relaciones semánticas, palabras cercanas tienen vectores de representación cercanos. Una ventaja un tanto mejor es que que las técnicas como GloVe producen vectores de más baja dimensionalidad en relación a las técnicas de BoW(Bag of Words).

Fasttext

Primero realizamos un análisis de la evolución del train y test accuracy de este algoritmo, a partir del epoch 200 test comienza a estabilizarse y en el epoch 950 alcanza su máximo en 59%.
En la figura 16 se observa que existe overfitting por la gran brecha entre train y test, no tuvimos tiempo de aplicar técnicas de regularización para reducirlo.

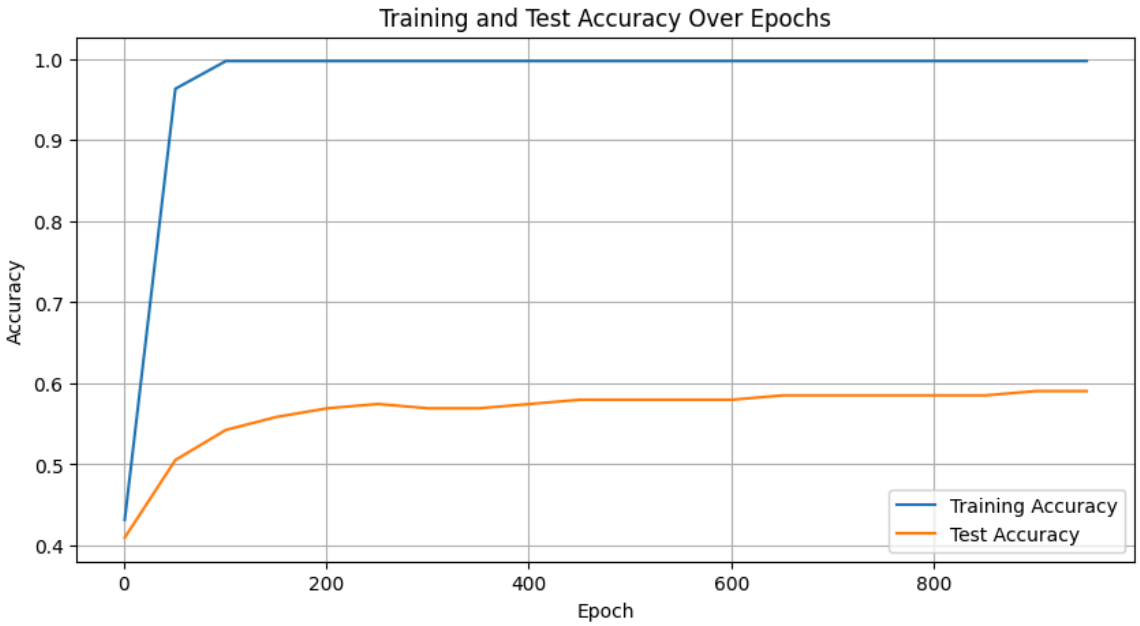

Figura 16 Progresión de train y test accuracy a lo largo de los epochs

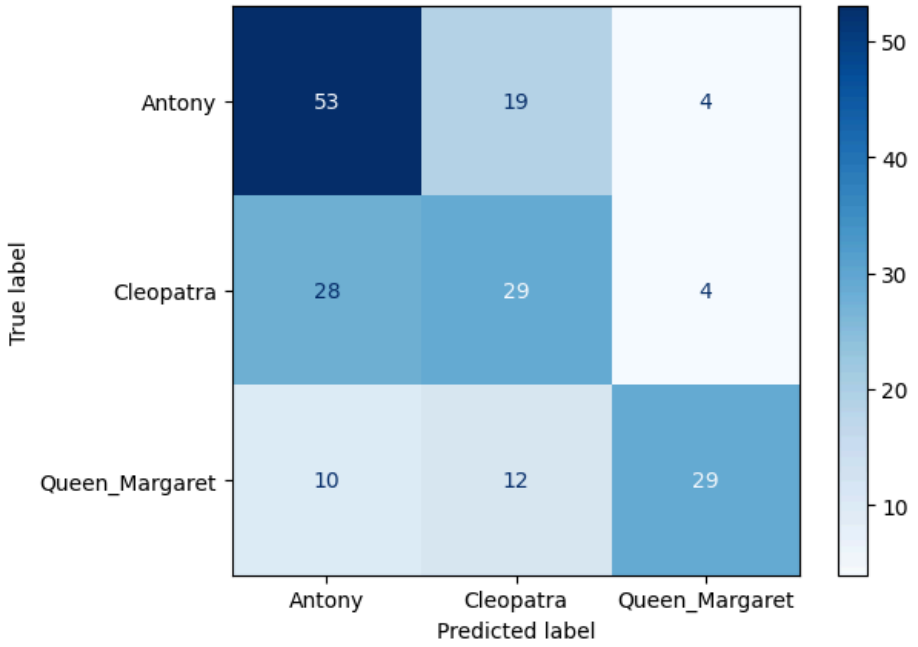

Figura 17 Matriz de confusión con FastText