Introducción a la ciencia de datos

Tarea 2

Guido Pereyra
Matias López

Introducción

En este informe, exploramos la tarea de clasificar párrafos según el personaje que los dijo en una selección de obras de Shakespeare. Utilizamos técnicas de vectorización como Bag of Words (BoW) y Term Frequency-Inverse Document Frequency (TF-IDF), y evaluamos el rendimiento de diferentes modelos de clasificación, incluyendo Multinomial Naive Bayes y Support Vector Machines (SVM). Nuestro objetivo es identificar las limitaciones y ventajas de estos métodos y proponer mejoras basadas en nuestros hallazgos.

Además, analizamos otras técnicas avanzadas de extracción de características textuales, como Word2Vec y GloVe, y su potencial para mejorar la clasificación en comparación con los métodos tradicionales. Aplicamos Análisis de Componentes Principales (PCA) para reducir la dimensionalidad y facilitar la visualización de los datos, observando cómo diferentes configuraciones de parámetros afectan la capacidad de nuestros modelos para distinguir entre personajes.

El informe también detalla la aplicación de técnicas de validación cruzada y GridSearchCV para optimizar los hiperparámetros de los modelos, y la evaluación del impacto de diferentes configuraciones en el rendimiento. Finalmente, se exploran técnicas de sobre-muestreo y submuestreo para abordar el sesgo en los datos y mejorar la capacidad del modelo para generalizar a las clases minoritarias, este último punto sólo de forma teórica.

A lo largo del informe, presentamos los resultados de múltiples entrenamientos, evaluamos las métricas de rendimiento y discutimos las implicaciones de nuestros hallazgos para el desarrollo de modelos de clasificación de texto más efectivos y robustos.

Metodología

a. Preparación de los Datos

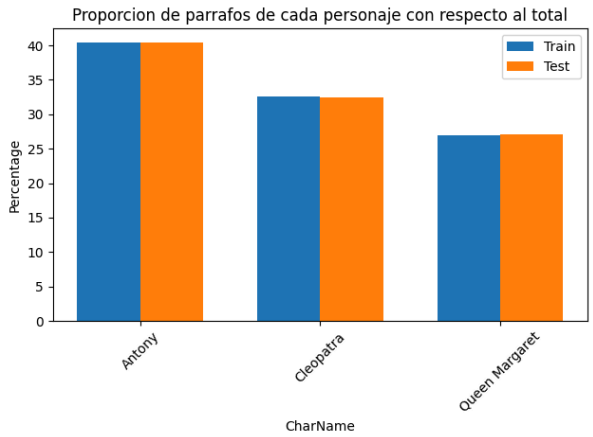
Los datos utilizados consisten en párrafos etiquetados según el personaje que los dijo, extraídos de la base de datos de obras de Shakespeare disponible en db.relational-data.org. Partiendo de la tabla ***paragraphs***, lo primero que hicimos fue aplicar la función ***clean_text*** a la columna ***PlainText*** para reducir el ruido generado por caracteres especiales, signos de puntuación, mayúsculas, entre otros. Las reglas aplicadas en esta limpieza fueron las siguientes:

- Convertir todo el texto a minúsculas.
- Eliminar caracteres especiales como saltos de línea y tabulaciones.
- Eliminar todos los signos de puntuación.
- Eliminar todos los dígitos.
- Eliminar stop_words de:
    - Inglés Moderno
    - Inglés Antiguo
- Eliminar palabras de 1 o menos caracteres.

Después de limpiar el texto, usamos la tabla characters para armar un conjunto de datos con solo dos columnas: ***CharName*** y ***CleanText***, donde:

- CharName representa el nombre del personaje.
- CleanText contiene el contenido del párrafo con la función clean_text aplicada.

Finalmente, dividimos los datos en conjuntos de entrenamiento y prueba para evaluar el rendimiento de los modelos. Utilizamos la función ***train_test_split*** de la librería ***sklearn***, estableciendo parámetros como ***random_state*** para asegurar la reproducibilidad de los resultados, ***stratify*** para mantener el balance de los personajes entre los conjuntos de entrenamiento y prueba, y ***test_size*** en 0.3 para tener un conjunto de test del 30%.



b. Vectorización del Texto

Para utilizar diferentes modelos de clasificación y predecir a qué personaje corresponde cada párrafo, aplicamos dos técnicas principales de vectorización: Bag of Words (BoW) y TF-IDF.

Bag of Words (BoW) es una técnica de procesamiento de lenguaje natural que convierte el texto en vectores numéricos, tratando el texto como una colección de palabras sin considerar su orden ni la gramática. Esta técnica presenta limitaciones como la incapacidad de reconocer sinónimos o variaciones y la generación de vectores grandes y dispersos si el vocabulario es amplio. En BoW, podemos contar tanto las ocurrencias como las frecuencias de las palabras.

El tamaño de la matriz resultante está determinado por el número de párrafos en el conjunto de datos (filas) y el número de palabras únicas en todos los párrafos del conjunto de datos (columnas). En nuestro ejemplo, utilizando los personajes Antony, Cleopatra y Queen Margaret, los tamaños de nuestros conjuntos de entrenamiento y prueba fueron los siguientes:

- Tamaño del conjunto de entrenamiento: 438 párrafos.
- Tamaño del conjunto de prueba: 188 párrafos.

La matriz resultante de la vectorización del conjunto de entrenamiento tiene 438 filas y 2532 columnas, correspondientes al número de palabras únicas.

Al utilizar solo tres personajes, ya obtenemos una matriz de dimensiones significativas. Si ampliamos el conjunto de datos para incluir muchos más personajes, la matriz resultante sería de un tamaño considerablemente mayor, lo que podría dificultar su manejo en memoria. Esto se debe a que no solo se añadirían más filas, sino que también el vocabulario se expandirá, generando muchas más columnas. Al verificar el tipo de datos de nuestra matriz, observamos que son elementos de tipo “numpy.int64”, lo que implica una ocupación en memoria calculada de la siguiente manera:

- Filas * columnas * 8 bytes = tamaño de matriz en bytes
    - 438 filas * 2532 columnas * 8 bytes = 9202368 bytes = **9 MB**

Esta consideración es importante al planificar el análisis de datos y la implementación de modelos de procesamiento de lenguaje natural en conjuntos de datos más extensos.

Denominamos estas matrices como "matrices dispersas" (***sparse matrix***) porque cada fila se expande en tantas columnas como palabras únicas hay en todo el conjunto de datos, pero no todas las palabras aparecen en cada párrafo, lo que provoca que muchas columnas tengan valores de cero.

Para la segunda parte de la vectorización, como mencionamos al principio, usamos ***TF-IDF***. Esta técnica consiste en convertir una matriz de recuento de palabras (bag of words) en una representación TF-IDF (Term Frequency - Inverse Document Frequency). Esta transformación ajusta las frecuencias de las palabras en los documentos para reflejar no solo su frecuencia en un documento específico, sino también la frecuencia inversa de los documentos en los que aparecen. Esto ayuda a reducir el peso de las palabras comunes y a resaltar las palabras que son más informativas y distintivas.

Configuramos los parámetros para incluir diferentes rangos de ***n-gramas*** (unigramas y bigramas) y el uso de stop words en inglés. Los n-gramas son secuencias de N elementos contiguos extraídos de un texto (palabras, caracteres o sílabas) y se utilizan para modelar secuencias de texto y capturar patrones de uso del lenguaje. Estos se utilizan comúnmente en el procesamiento de lenguaje natural y en el análisis de texto.

Ejemplo de matriz final:

En una de las matrices resultante, se pueden observar dos características importantes:

- **Matriz dispersa:** Muchos elementos tienen valores de cero.
- **Limitaciones de métodos elementales:** Palabras como "young", "younger" y "youth" no se unifican, lo que podría ser beneficioso en algunos análisis.

```
# Mostrar la representación TF-IDF
print("Representación TF-IDF:")
print(X_train_tf.toarray())

# Mostrar las palabras correspondientes
print("Palabras:")
print(count_vect.get_feature_names_out())
```

```
Representación TF-IDF:
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
Palabras:
['abhorring' 'abides' 'abjects' ... 'young' 'younger' 'youth']
```

Una de las principales limitaciones de los modelos basados en Bag-of-Words (BoW) y TF-IDF es la pérdida de contexto. Ambos métodos tratan a las palabras de manera independiente, sin poder capturar relaciones semánticas ni dependencias gramaticales. Esto significa que no pueden distinguir entre diferentes significados de la misma palabra (polisemia) ni agrupar palabras con significados similares (sinonimia). Además, no son capaces de capturar estructuras complejas, relaciones de dependencia o la estructura narrativa de los textos.

Otra limitación significativa es la alta dimensionalidad que generan estos modelos, produciendo vectores muy dispersos sin capacidad de reducción debido a la naturaleza independiente de las palabras. Esto no solo dificulta el procesamiento y almacenamiento, sino que también puede afectar la escalabilidad cuando se aplican a grandes volúmenes de texto. En resumen, aunque BoW y TF-IDF son útiles para ciertas aplicaciones, su incapacidad para captar el contexto y manejar la alta dimensionalidad limita su efectividad en análisis de texto más avanzados.

Al estar entrenando un modelo que busca predecir quién dijo cada párrafo, no creo que estas técnicas de procesamiento de lenguaje natural que no toman en cuenta el contexto ni las estructuras narrativas sean de gran utilidad. Sin embargo, pueden proveer alguna utilidad si encontramos personajes con un lenguaje muy específico que los distingue de los demás. En estos casos, las diferencias en la frecuencia de términos y el uso de vocabulario particular podrían ser suficientes para lograr cierta precisión en la predicción.

Otras técnicas:

Existen otras técnicas que podrían ser usadas para extraer características de los textos de Shakespeare, las cuales pueden ayudar a capturar más contexto. Estas técnicas son especialmente útiles en análisis de sentimientos, sistemas de recomendación y traducciones automáticas. Dos de las técnicas más destacadas son Word2Vec y GloVe.

Word2Vec es una técnica de aprendizaje automático desarrollada por Google que se utiliza para aprender representaciones vectoriales de palabras en un texto. Los vectores generados capturan las relaciones semánticas entre palabras, lo que significa que palabras con significados similares están cerca en el espacio vectorial. Word2Vec utiliza dos métodos principales:

- **Skip-gram**: Predice el contexto de una palabra dada.
- **CBOW (Continuous Bag of Words)**: Predice una palabra dado su contexto.

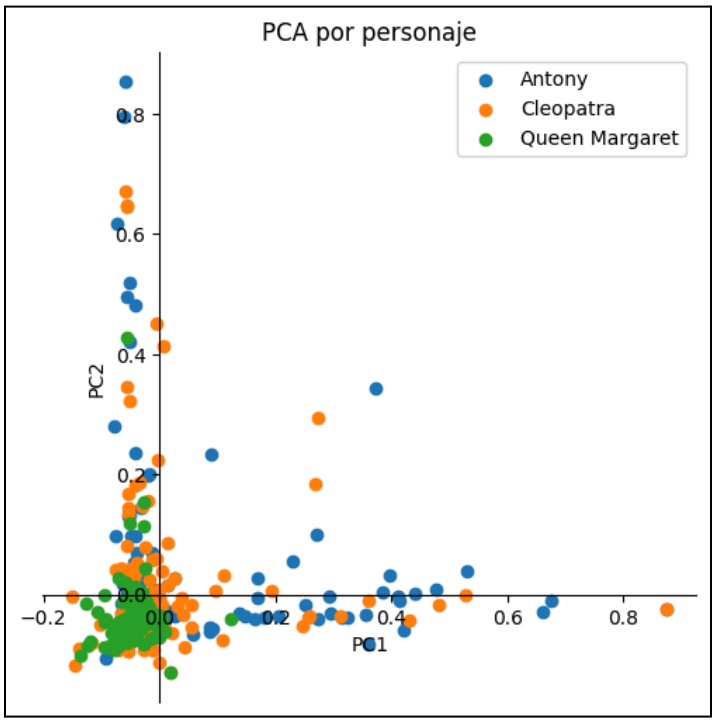
GloVe es una técnica desarrollada por Stanford para obtener representaciones vectoriales de palabras. A diferencia de Word2Vec, que aprende vectores basándose en la predicción de palabras en un contexto local, GloVe se basa en el análisis global de la coocurrencia de palabras en un texto. GloVe construye una matriz de co-ocurrencia de palabras y luego aplica factorización de matrices para aprender los embeddings.

Comparación con Métodos Basados en Bag-of-Words (BoW)

- **Captura de Semántica**: BoW no captura relaciones semánticas entre palabras, mientras que Word2Vec y GloVe sí lo hacen.
- **Dimensionalidad Reducida**: Los embeddings de palabras reducen la dimensionalidad del espacio de características comparado con el BoW tradicional.
- **Relaciones Sintácticas y Semánticas**: Word2Vec y GloVe capturan relaciones tanto sintácticas como semánticas entre palabras, lo que mejora la calidad de las características extraídas.

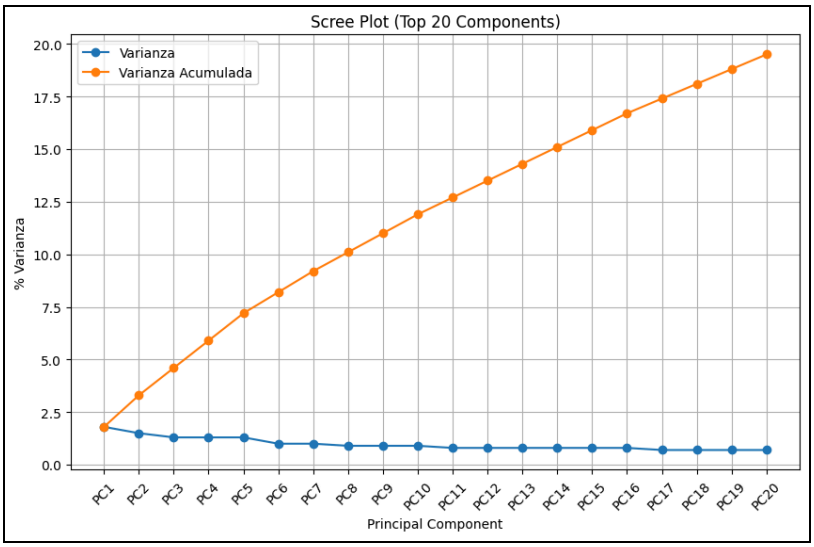
c. Análisis de Componentes Principales (PCA)

Luego de preparar los datos y vectorizar el texto, aplicamos PCA (Análisis de Componentes Principales) para reducir la dimensionalidad y visualizar la separación entre los personajes en un espacio bidimensional. Inicialmente, utilizamos n-grams (1,1), con las opciones de stop words e IDF desactivadas, obteniendo los siguientes resultados:



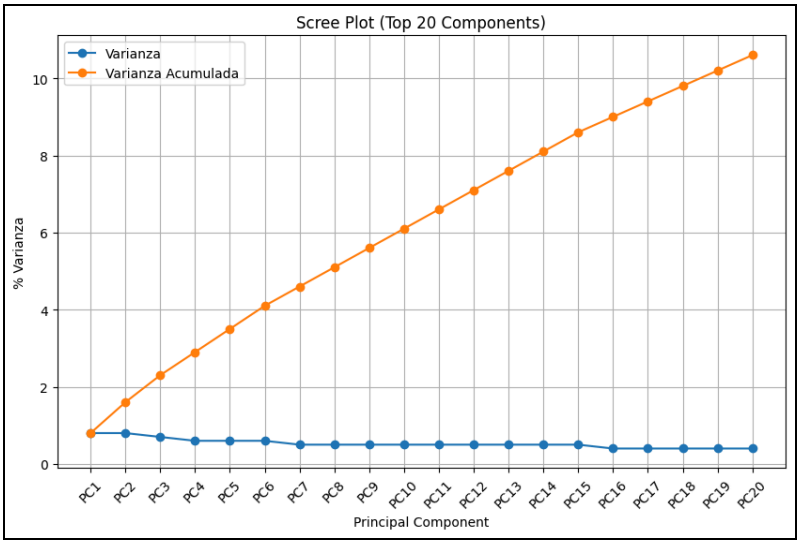
Al mapear las dos primeras componentes principales del conjunto de entrenamiento, observamos una alta superposición entre los personajes, lo que impide la formación de clústeres definidos para cada uno de ellos. Esta falta de separación clara sugiere que las primeras dos componentes no son suficientes para distinguir adecuadamente entre Antony, Cleopatra y Queen Margaret.

Además, el scree plot refuerza esta observación, mostrando que las primeras 20 componentes principales no explican ni el 20% de la varianza total. Este bajo porcentaje de varianza acumulada confirma que no podemos diferenciar significativamente a los personajes utilizando estos componentes principales.



Variando los parámetros de la vectorización utilizando n-grams (1,2), con las opciones de stop words e IDF activadas, los resultados no son mucho más prometedores:

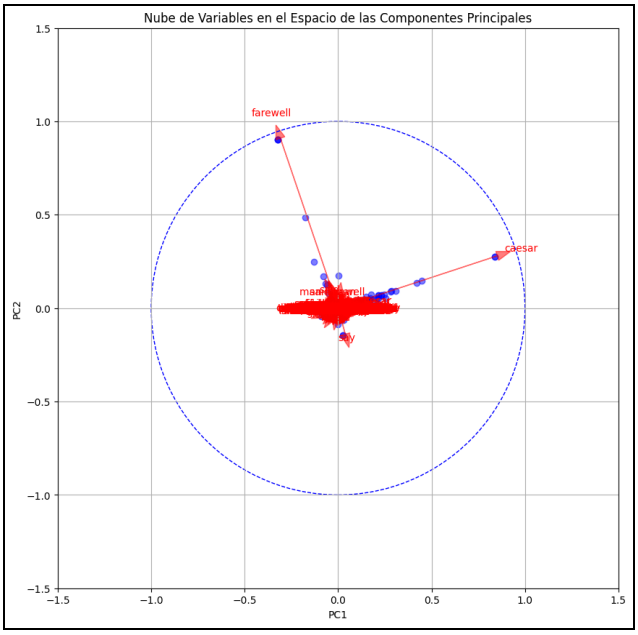
En este caso, la agregación de bigramas y el uso de TF-IDF con filtrado de stop words llevó a una mayor dispersión de la varianza entre las componentes principales y a juntar mucho más los puntos a la hora de graficar las primeras dos componentes. Las 20 primeras componentes explican un poco menos del 10% de la varianza total, en contraste con casi un 20% obtenido previamente.



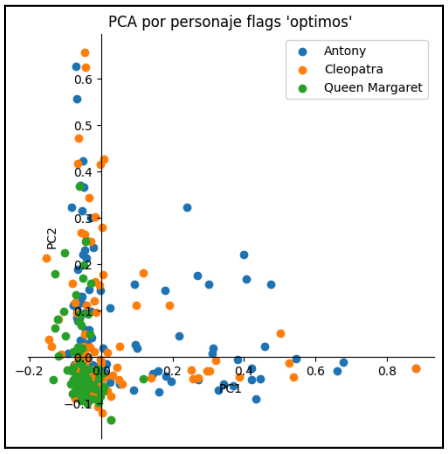
Este fenómeno puede explicarse por la reducción del ruido que generaban algunas stop words, que, aunque no son informativas, contribuyen significativamente a la varianza total. Además, la inclusión de bigramas incrementó la dimensionalidad del análisis, distribuyendo la varianza en más dimensiones. El uso de Inverse Document Frequency redistribuye el peso de las palabras comunes a las menos frecuentes, provocando una mayor desconcentración de la varianza en las primeras componentes.

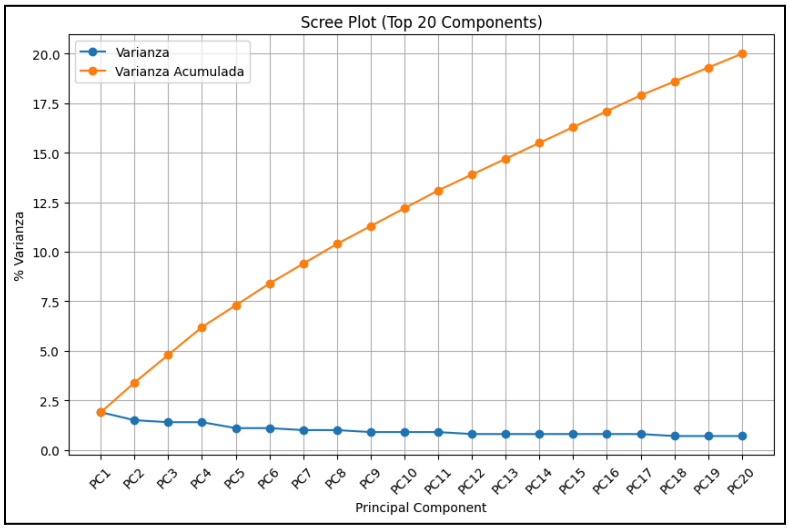
Si bien estos resultados no necesariamente son negativos, pueden indicar que la representación de los datos es más rica y detallada, lo que podría mejorar el rendimiento de los modelos de machine learning en tareas específicas como la clasificación. Sin embargo, también implica que puede ser necesario considerar más componentes principales para capturar una cantidad significativa de varianza y lograr una buena separación de los datos.

Para complementar el análisis anterior, generamos una 'Nube de Variables en el Espacio de las Componentes Principales', donde se puede observar el peso de ciertas palabras en las componentes PC1 y PC2. Este gráfico proporciona una visualización clara de cómo determinadas palabras influyen en la representación PCA de los textos de Shakespeare. Las palabras farewell y caesar son las más influyentes en las dos primeras componentes principales, mientras que la alta concentración de palabras cerca del centro sugiere que muchas palabras no contribuyen significativamente a la diferenciación entre personajes. Al considerar la baja varianza explicada por cada componente y la representación de las componentes mediante palabras específicas, se refuerza la idea de la dificultad para agrupar a los personajes utilizando solo unas pocas componentes principales.



Luego de la validación cruzada y el entrenamiento del modelo usando todas las combinaciones de los parámetros de vectorización (tema que se verá en las secciones posteriores), la que tuvo mejor rendimiento fue **{"stop_words": 'english', "ngram": (1,1), "idf": False}** y la misma muestra una representación de las primeras dos componentes principales que si bien no es ideal pero es más prometedor que los anteriormente vistos ya que los puntos se encuentran más dispersos y la varianza acumulada de las primeras veinte llega a representar el 20%.





Modelos de Clasificación

Una vez realizada la preparación de los datos, la vectorización del texto y el análisis PCA, continuamos con el entrenamiento del modelo para predecir quién dijo cada uno de los párrafos de las obras de William Shakespeare.

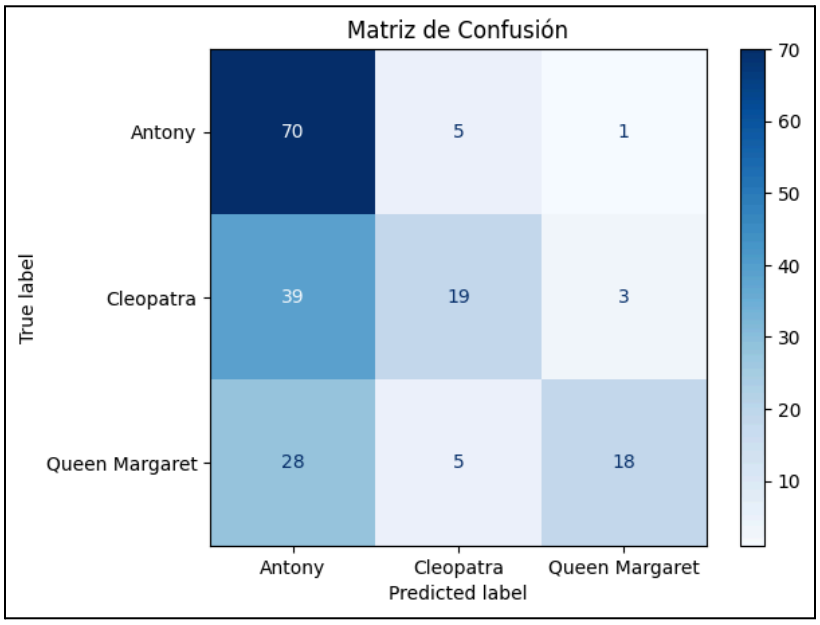
Para estudiar diferentes rendimientos, entrenamos dos modelos diferentes aplicando técnicas de validación cruzada en la búsqueda de encontrar los parámetros de la vectorización que nos den los mejores rendimientos, usando como referencia tanto las métricas de precisión cómo f1-score. Los dos modelos que usamos fueron:

- Multinomial Naive Bayes (NB)
- SVM

Primer entrenamiento

Iniciamos el primer entrenamiento usando Naive Bayes (NB) y los parámetros iniciales de vectorización, los cuales son stop-word y idf activados, y ngrams = (1,2). Los resultados, como era previsto, no fueron muy optimistas:

Personaje	Precision	Recall	F1-score	Support
Antony	51.095%	92.105%	65.728%	76.0
Cleopatra	65.517%	31.148%	42.222%	61.0
Queen Margaret	81.818%	35.294%	49.315%	51.0



Comparando los resultados de las columnas “precision” y “recall” podemos detectar varios problemas que pueden surgir si sólo miramos una métrica en particular y no todo el conjunto de

métricas que le da contexto al resultado final de las predicciones. El ejemplo de Antony es el más claro: podemos ver que hay una precisión baja pero un recall alto, lo que muestra que el modelo predijo que muchísimos párrafos eran de Antony. Esto llevó a equivocarse mucho, pero al mismo tiempo a adivinar la mayoría de los párrafos que realmente eran de él, ya que tiene un 92% de recall. Sin embargo, esto parece ser más por una cuestión de volumen de predicciones que de precisión en las mismas, lo que puede exponer un sesgo del modelo hacia predecir la clase Antony.

Un comportamiento diferente se observa con Cleopatra y Queen Margaret, donde el modelo fue mucho más conservador al predecir párrafos de su autoría, lo que se refleja en una precisión más alta pero un recall mucho más bajo. Todo lo anterior también se puede notar en los resultados de la métrica F1-score, que es la media armónica de la precisión y el recall, proporcionando una medida única que toma en cuenta ambos aspectos.

Actualmente estamos trabajando con un set de datos de tres personajes donde hay un desbalance entre Antony y los otros dos protagonistas, ya que los párrafos de Antony representan un 40% del total de todos los párrafos, como se puede apreciar en el (1) del informe. Este desbalance hace que la métrica F1-score sea aún más importante, ya que ayuda a tener una medida más balanceada del rendimiento del modelo. Cuanto más desbalanceado sea el conjunto de datos, mayores diferencias habrá entre la precisión y el recall de los personajes en nuestras predicciones.

Mirar solo el valor de accuracy puede ser problemático porque puede dar una falsa sensación de buen rendimiento, especialmente en casos de clases desbalanceadas. Un alto valor de accuracy puede simplemente reflejar el hecho de que el modelo está prediciendo bien la clase mayoritaria, ignorando las minoritarias.

En un escenario con un mayor desbalance de datos, la métrica de accuracy se volvería aún menos representativa del verdadero rendimiento del modelo, ya que podría ser alta a pesar de que el modelo no está clasificando correctamente las clases minoritarias.

Segundo entrenamiento

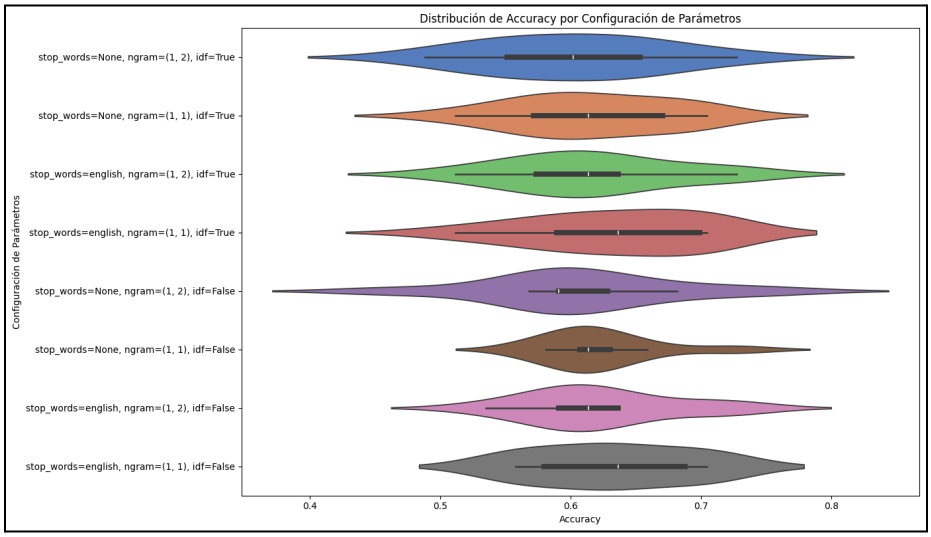
Para el segundo entrenamiento, volvemos a usar Naive Bayes (NB), pero esta vez, en lugar de elegir nosotros mismos los parámetros de la vectorización del texto, vamos a analizar la performance de cada una de las combinaciones de dichos parámetros para buscar la óptima en términos de accuracy y f1-score. Para esto, utilizaremos la técnica de validación cruzada junto a la función GridSearchCV para encontrar los mejores hiperparametros del modelo para cada set de entrenamiento.

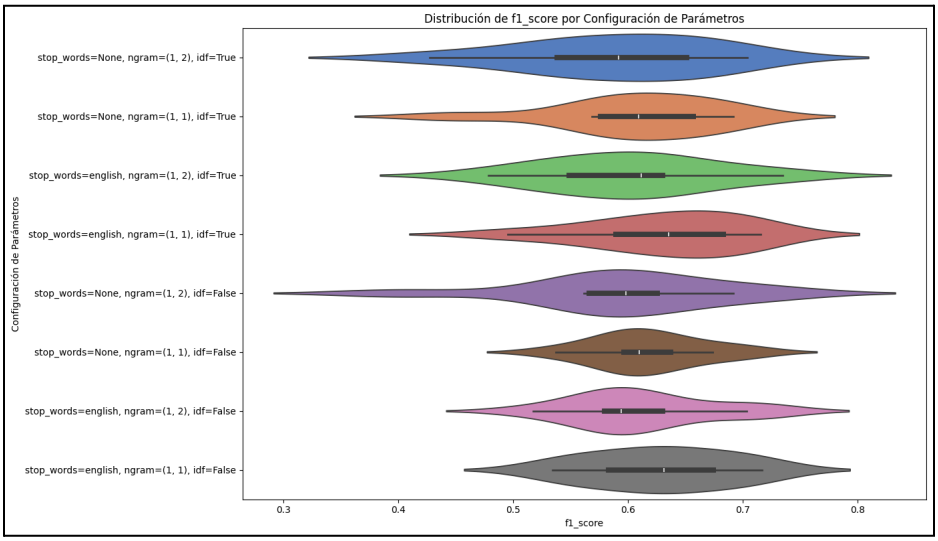
La **validación cruzada** es una técnica utilizada en machine learning para evaluar la capacidad de generalización de un modelo. En lugar de dividir los datos en un único conjunto de entrenamiento y uno de prueba, la validación cruzada permite una evaluación más robusta y confiable mediante la división de los datos de entrenamiento en múltiples subconjuntos. Esto

implica dividir los datos en varios subconjuntos y realizar múltiples entrenamientos y evaluaciones.

Los datos se dividen en X subconjuntos de tamaño aproximadamente igual, luego se realizan iteraciones del proceso de entrenamiento y evaluación. En cada iteración se usa uno de los X conjuntos como conjunto de prueba y los restantes X-1 conjuntos se usan como entrenamiento. Al final de dicho proceso, se promedian todas las métricas de evaluación para tener una estimación más confiable del rendimiento del modelo.

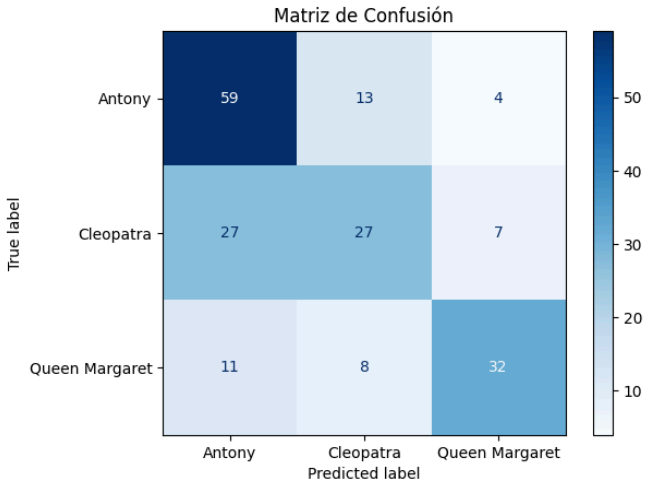
Aplicando cross-validation con un n_split = 10 podemos observar que la configuración con stop words en inglés activadas, IDF desactivado y solo unigramas (violín gris) muestra la mejor performance del modelo, tanto en términos de Accuracy como de F1-score. Esto se debe a que presenta una mediana alta y una distribución de los datos (anchura) ubicada en los valores más altos. Con respecto a los hiperparametros de NB se setea fit_prior=False.





Hecha la validación cruzada en búsqueda de los hiperparametros más adecuados, procedemos a entrenar nuevamente el modelo usando NB obteniendo los siguientes resultados:

Personaje	Precisión	Recall	F1-score	Soporte
Antony	60.825%	77.632%	68.208%	76
Cleopatra	56.250%	44.262%	49.541%	61
Queen Margaret	74.419%	62.745%	68.085%	51



En el segundo entrenamiento, observamos mejoras significativas en el rendimiento del modelo en comparación con el primero. Para el personaje de Antony, la precisión aumentó de 51.09% a 60.82%, aunque el recall disminuyó de 92.10% a 77.63%. Este equilibrio resultó en un incremento del F1-score de 65.72% a 68.20%, indicando una mejor discriminación del modelo. En el caso de Cleopatra, aunque la precisión disminuyó de 65.51% a 56.25%, el recall mejoró notablemente de 31.148% a 44.262%, lo que llevó a un aumento del F1-score de 42.22% a 49.54%. Finalmente, para Queen Margaret, si bien la precisión disminuye, el recall mejoró, pasando de 81.81% a 74.41% y de 35.29% a 62.74%, respectivamente. Esto resultó en un aumento significativo del F1-score de 49.31% a 68.08%. Estos resultados reflejan un modelo más equilibrado y efectivo en la clasificación de los párrafos, mejorando notablemente su capacidad de generalización y precisión en la identificación de los personajes.

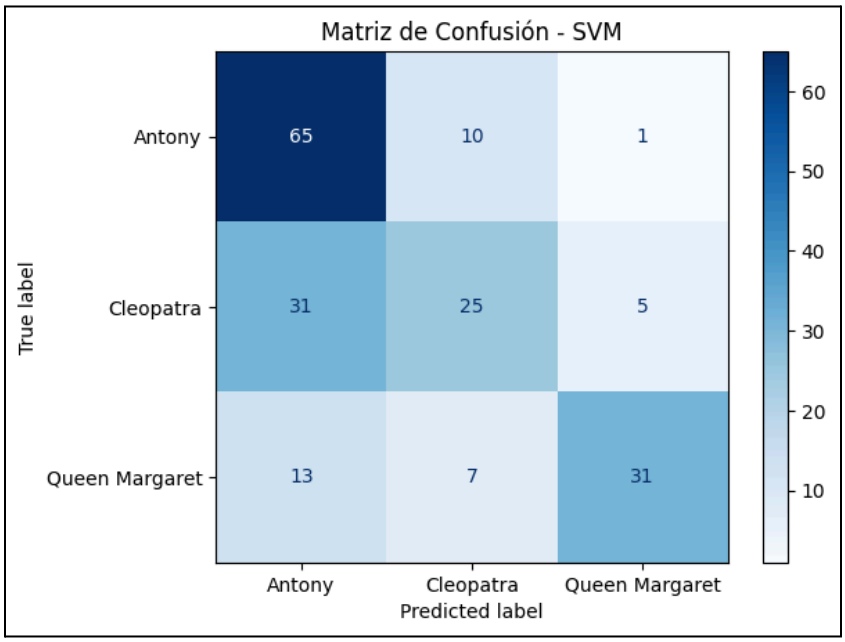
Tercer entrenamiento

En este tercer entrenamiento, usamos SVM (Support Vector Machine) para clasificar el texto de los personajes. SVM es una de las técnicas más populares para la clasificación de texto debido a su capacidad para manejar datos de alta dimensionalidad y su eficacia en diversas tareas de clasificación. Los datos textuales suelen ser de alta dimensionalidad, especialmente cuando se representan mediante técnicas como Bag-of-Words (BoW) o TF-IDF. SVM maneja bien estas altas dimensiones y puede encontrar hiperplanos de separación eficaces en espacios de características complejos.

Los vectores de características generados por BoW y TF-IDF son a menudo dispersos (sparse), con muchos ceros. SVM puede manejar eficientemente estos datos dispersos, lo que lo hace adecuado para aplicaciones de procesamiento de lenguaje natural (NLP).

Para la primera prueba, usamos los parámetros stop-words y idf activados, y ngrams = (1,2). Los resultados mejoraron al no ser tan conservador en las predicciones de los párrafos que no pertenecen a Antony, mejorando el F1-score tanto de Cleopatra como de Queen Margaret. Esto aumentó la precisión de estas clases sin penalizar el recall.

Personaje	Precision	Recall	F1-score	Support
Antony	60.0%	86.0%	70.0%	76
Cleopatra	60.0%	41.0%	49.0%	61
Queen Margaret	84.0%	61.0%	70.0%	51

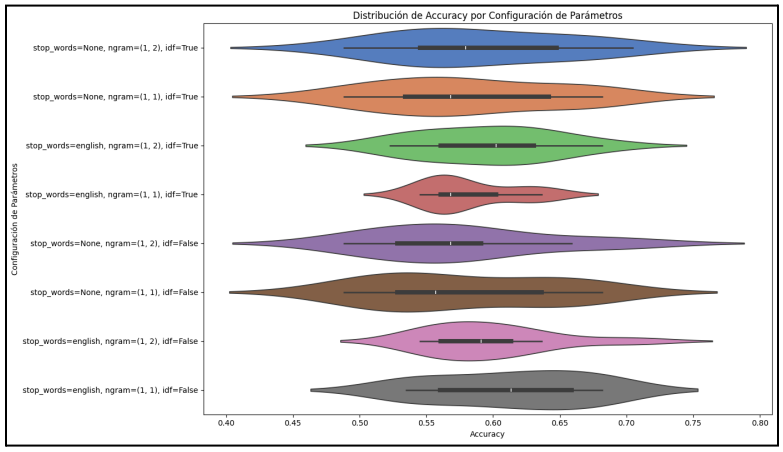


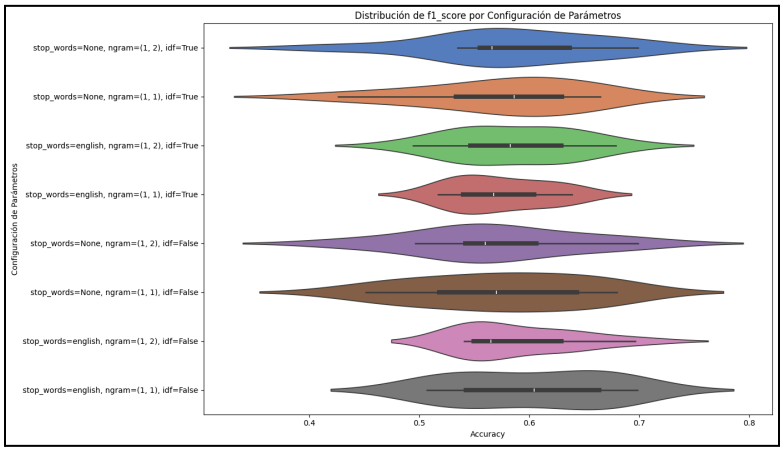
Cuarto entrenamiento

En este cuarto intento volvimos a hacer validación cruzada pero sumandole tambien la funcion GridSearchCV de SVN para iterar sobre los diferentes parámetros que pueden ser utilizados en dicho modelo como C , gamma y kernel.

- **C**: Controla la penalización de los errores. Un valor alto intenta evitar errores a toda costa, mientras que un valor bajo permite más errores con un margen mayor.
- **gamma**: Controla la influencia de cada punto de entrenamiento en los kernels no lineales. Un valor alto significa que cada punto tiene una influencia pequeña, mientras que un valor bajo significa que cada punto tiene una influencia grande.
- **kernel**: Define la función de transformación de los datos. Los kernels comunes son linear, polynomial, RBF y sigmoid.

Una vez calculadas las métricas resultado de cada uno de los entrenamientos representadas por los siguiente gráficos de violín:

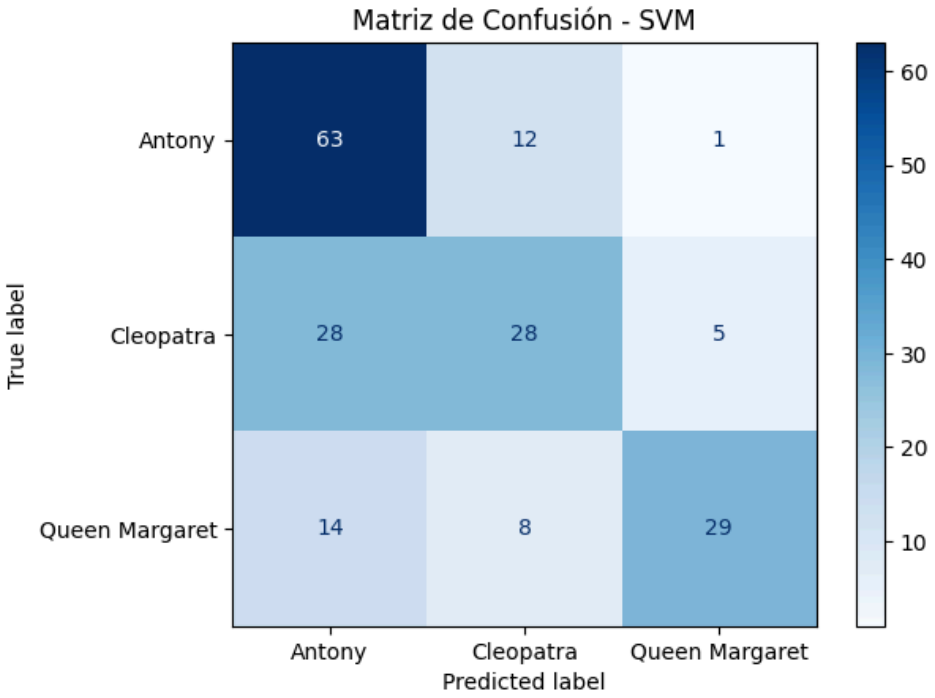




Se puede apreciar que la combinación “stop_words=english, ngram=(1,1), idf=False” es la mejor, ya que tanto en el gráfico de Accuracy como en el de F1-score, esta configuración tiene una mediana alta. Esto indica que, en promedio, esta configuración proporciona buenos resultados. Además, la dispersión de los valores de Accuracy y F1-score es menor y está concentrada en valores altos, lo que sugiere que el modelo con esta configuración tiene un rendimiento consistente y robusto.

Revisando los hiperparametros de gridSearchCV que lograron las mejores performance con la configuración anteriormente mencionada, vemos que estos fueron C:1 , gamma: 1 y kernel : linear.

Personaje	Precision	Recall	F1-score	Support
Antony	60.0%	83.0%	70.0%	76
Cleopatra	58.0%	46.0%	51.0%	61
Queen Margaret	83.0%	57.0%	67.0%	51

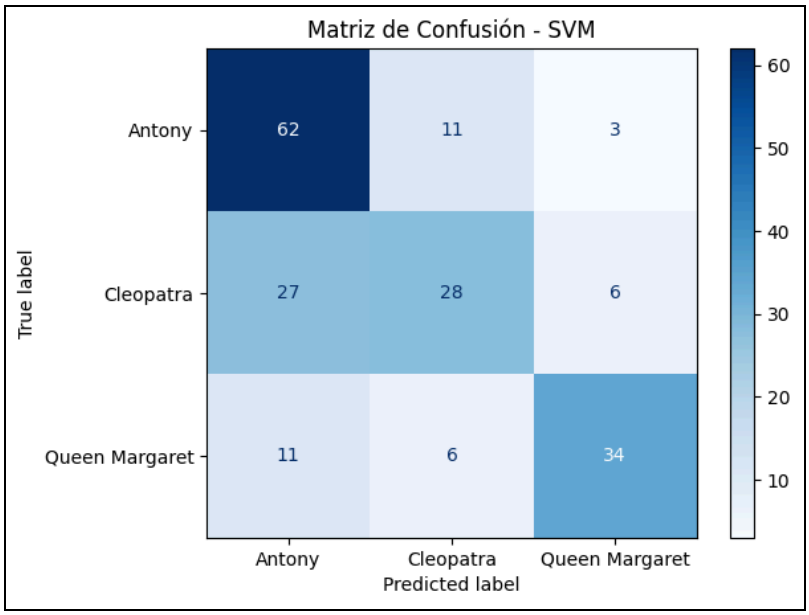


Donde no se vieron grandes variaciones con los resultados del anterior entrenamiento.

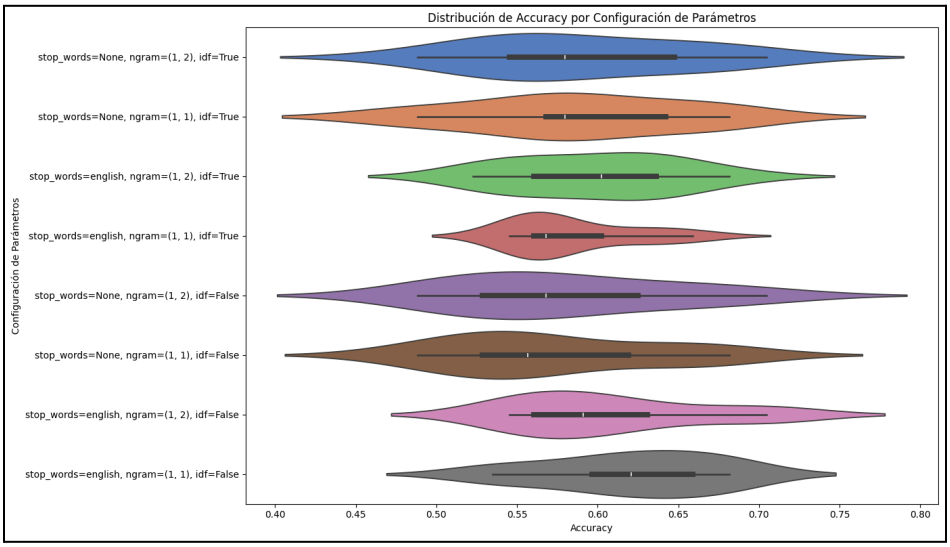
Quinto entrenamiento

Con motivos experimentales, intentemos entrenar un modelo SVN usando parámetros de vectorización completamente distintos como “stop_words=None, ngram=(1,2), idf=True” y C:100 , gamma: 0.01 y kernel : rbf , obteniendo los siguientes resultados que superan estrechamente los vistos en el entrenamiento anterior:

Personaje	Precision	Recall	F1-score	Support
Antony	62.0%	82.0%	70.0%	76
Cleopatra	62.0%	46.0%	53.0%	61
Queen Margaret	79.0%	67.0%	72.0%	51

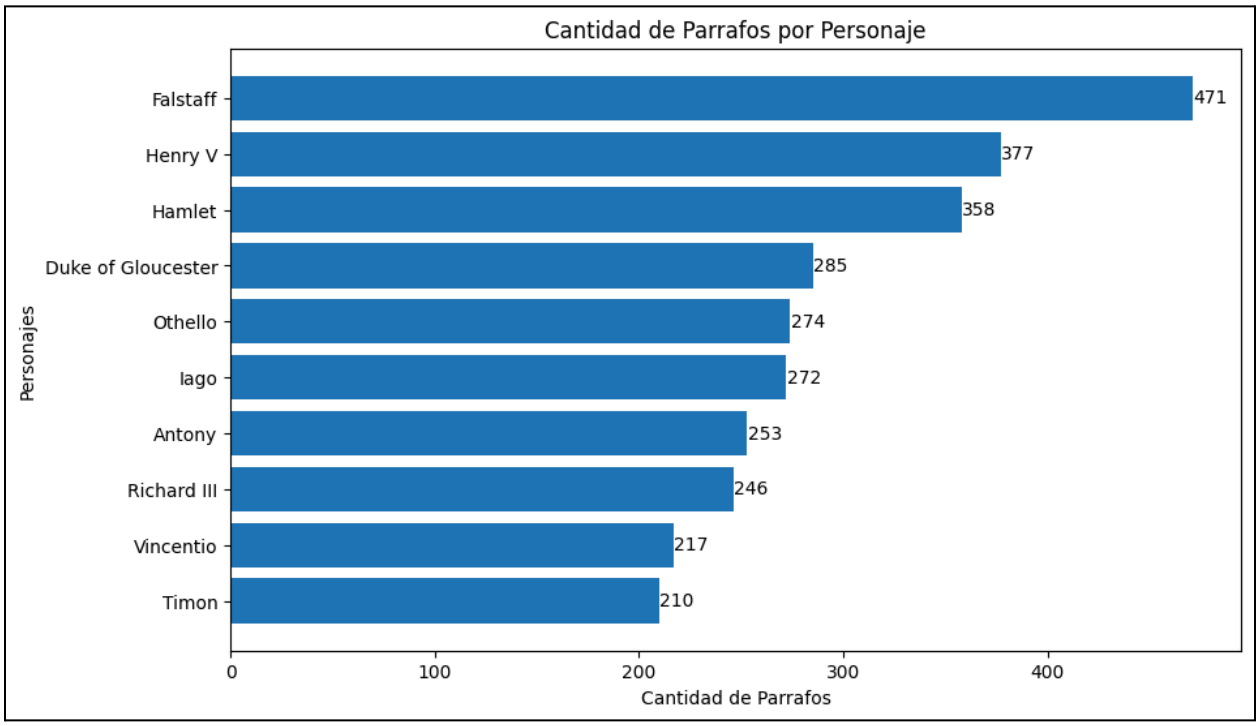


Podemos ver cómo en estos últimos resultados se balanceo un poco la métrica F1-score, lo que podría ser una solución adecuada si buscamos una recall decente sin penalizar la precisión. En primera instancia, no hubiese elegido estos hiperparametros ya que, como se ve en la imagen (violín azul), la dispersión es alta tanto de accuracy como en el f1-score pero alcanzaba los más altos valores.



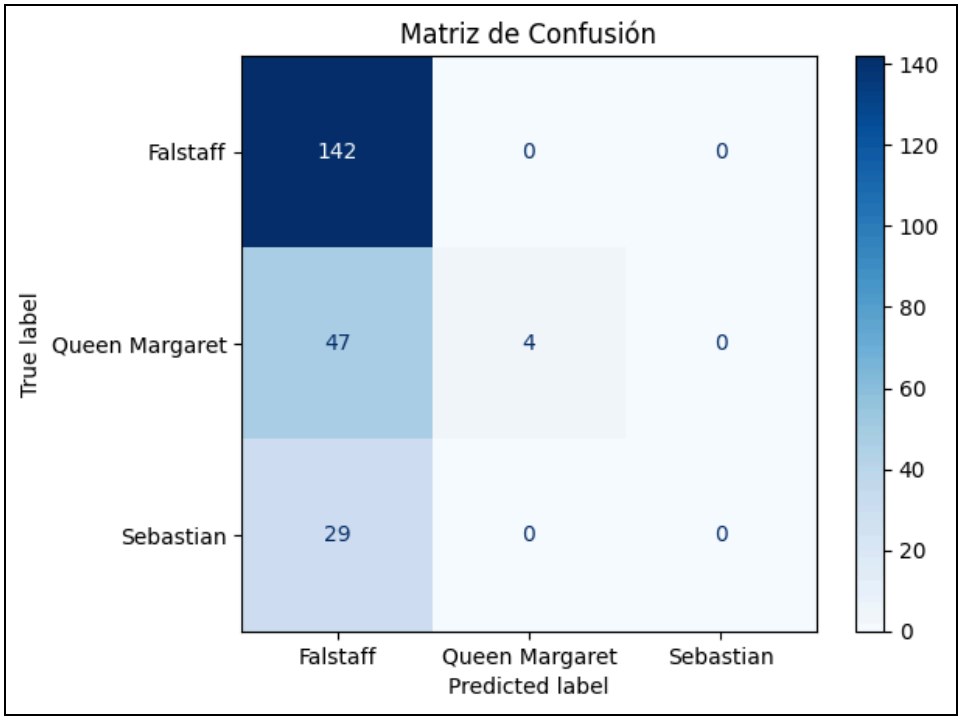
Sexto entrenamiento

En este sexto intento, se busca experimentar el entrenamiento de un modelo NB con un set de datos completamente desbalanceado eligiendo para eso al personaje Falstaff que sabemos que es el que lidera la tabla de cantidad de párrafos y a Sebastian que aparece en la parte baja de dicha tabla con solo 98 párrafos en su nombre:



Para realizar este entrenamiento, utilicé los parámetros de vectorización que me dieron mejores resultados en las pruebas anteriores: “stop_words='english', ngram=(1,1), idf=False”. Los resultados, como se esperaban, fueron los de un modelo que solo sabe predecir Falstaff.

Personaje	Precision	Recall	F1-score	Support
Falstaff	65.138%	100.000%	78.889%	142
Queen Margaret	100.000%	7.843%	14.545%	51
Sebastian	0%	0%	0%	29

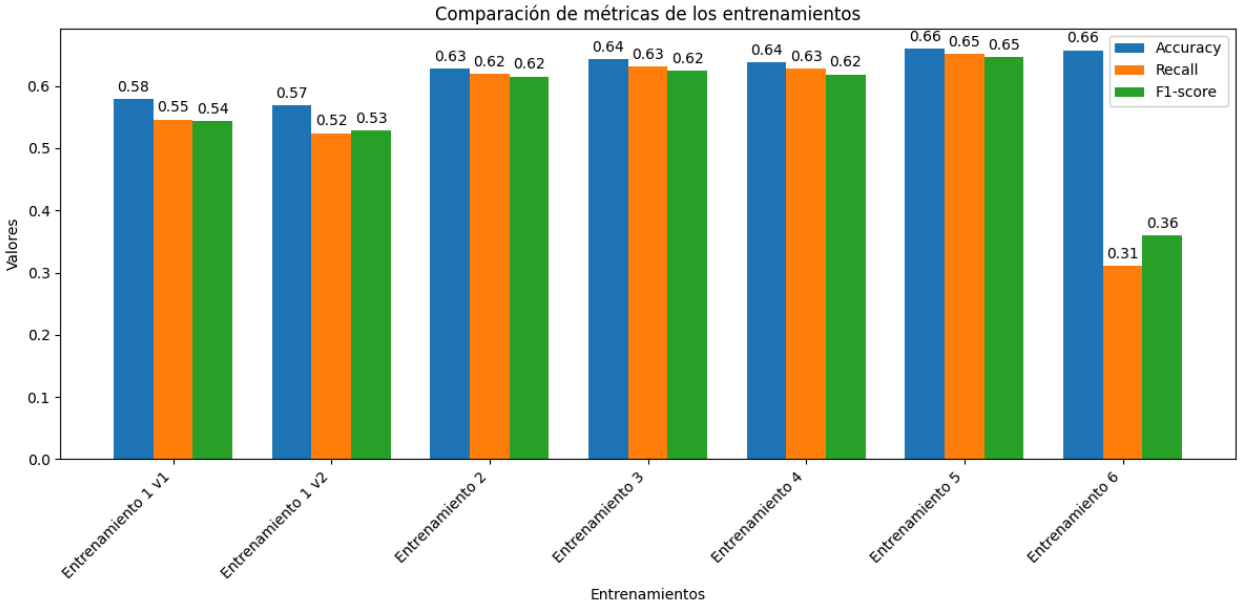


La matriz de confusión muestra claramente que el modelo predijo correctamente todos los casos de Falstaff (142) y no predijo correctamente ningún caso de Sebastian. Además, de los 51 casos de Queen Margaret, el modelo solo identificó correctamente 4, lo que se traduce en un recall de 7.84%. Esto es indicativo de que el modelo fue extremadamente conservador con los personajes Queen Margaret y Sebastian, resultando en una precisión del 100% para Queen Margaret pero un recall muy bajo, y en el caso de Sebastian, tanto la precisión como el recall fueron del 0%.

Estos resultados reflejan un modelo completamente sesgado, con un recall del 100% para Falstaff y una precisión del 65.13%, lo que significa que mientras el modelo es capaz de identificar todos los casos de Falstaff, tiene una tasa significativa de falsos positivos para los otros personajes. El F1-score de 78.88% para Falstaff indica un buen equilibrio entre precisión y recall para esta clase específica, pero los F1-scores extremadamente bajos para Queen Margaret y Sebastian (14.5% y 0%, respectivamente) destacan la incapacidad del modelo para generalizar más allá de la clase dominante.

Este sesgo puede ser abordado mediante técnicas de sobre-muestreo y submuestreo. El sobre-muestreo implica aumentar el número de ejemplos en las clases minoritarias, lo que podría ayudar a mejorar la capacidad del modelo para reconocer Queen Margaret y Sebastian. Métodos como SMOTE (Synthetic Minority Over-sampling Technique) generan nuevas instancias sintéticas de las clases minoritarias para equilibrar el conjunto de datos. Por otro lado, el submuestreo (undersampling) reduce el número de ejemplos en la clase dominante (Falstaff en este caso) para igualar el número de ejemplos de las clases minoritarias, aunque esto puede llevar a pérdida de información si no se realiza cuidadosamente.

Resumen de las métricas de todos los entrenamientos:



Conclusion

A lo largo de este informe, hemos explorado la clasificación de párrafos según el personaje que los dijo en una selección de obras de Shakespeare. Utilizando técnicas de vectorización como Bag of Words (BoW) y Term Frequency-Inverse Document Frequency (TF-IDF), y evaluando el rendimiento de modelos de clasificación como Multinomial Naive Bayes y Support Vector Machines (SVM), hemos podido identificar las ventajas y limitaciones de cada método.

Nuestros resultados iniciales mostraron que, si bien los métodos tradicionales de BoW y TF-IDF pueden ser efectivos en ciertos contextos, tienen limitaciones significativas, especialmente en la captura del contexto y las relaciones semánticas entre palabras. Esto se reflejó en la alta dimensionalidad y la dispersión de los vectores, lo que complicó el análisis y la interpretación de los resultados.

El análisis de componentes principales (PCA) nos permitió visualizar la separación entre personajes, aunque con limitaciones debido a la superposición de los datos y a baja representación en la varianza de cada componente.

La validación cruzada y el uso de GridSearchCV fueron cruciales para optimizar los hiperparámetros de los modelos, mejorando su precisión y capacidad de generalización.

Los entrenamientos con diferentes configuraciones de parámetros mostraron que la configuración óptima para nuestro conjunto de datos era "stop_words='english', ngram=(1,1), idf=False". Esta configuración logró un equilibrio adecuado entre precisión y recall, destacando la importancia de la selección cuidadosa de los parámetros de vectorización.

Finalmente, abordamos de forma teórica el sesgo presente en el modelo mediante técnicas de sobre-muestreo y submuestreo, destacando la necesidad de equilibrar las clases en el conjunto de datos para mejorar la capacidad de generalización del modelo cuando el set de datos de entrenamiento no se encuentra balanceado entre los personajes.

En resumen, nuestros hallazgos subrayan la importancia de seleccionar técnicas de vectorización y modelos adecuados para la tarea específica de clasificación de texto. Además, destacan la necesidad de considerar técnicas avanzadas y métodos de optimización de hiper parámetros para mejorar el rendimiento de los modelos en contextos como los que presenta este laboratorio, como también la importancia de la preparación y el balanceo de los set de datos previo a cualquier entrenamiento.