Tarea 2 - Introducción a la Ciencia de Datos

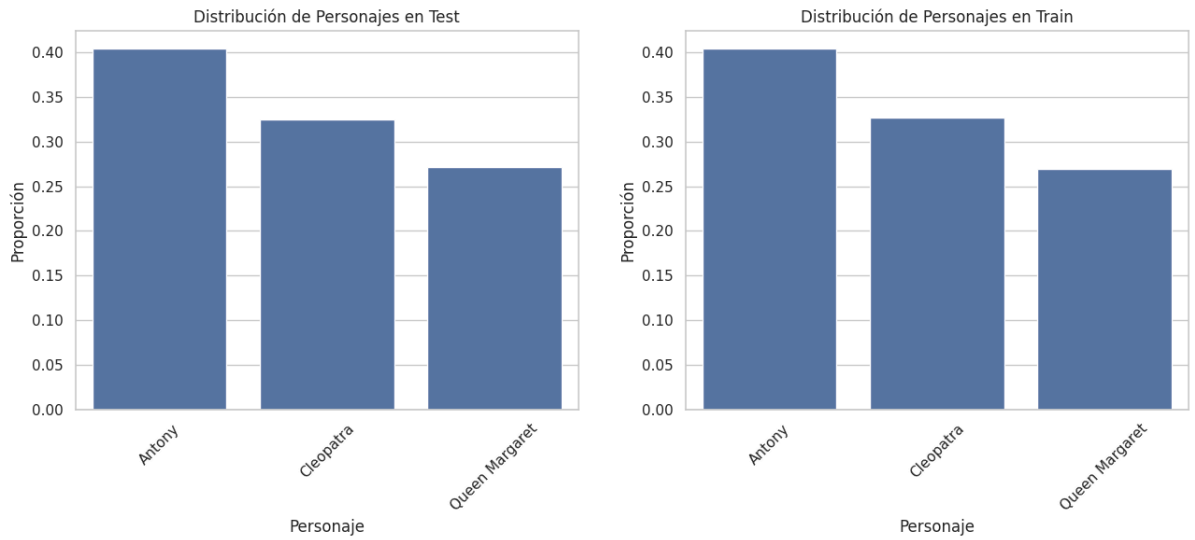
Agustina Añasco - Horacio Solé

Parte 1: Dataset y representación numérica de texto

1. En el notebook de la tarea, se crea un Dataset reducido de sólo 3 personajes. Se espera
que utilice su propia versión de la función clean_text() de la Tarea 1. Parta los datos para generar un conjunto de test del 30% del total, utilizando muestreo estratificado.

> **Sugerencia: utilice el parámetro stratify de la función train_test_split de scikit-learn, fijando también random_state para obtener resultados reproducibles.**

2. Genere una visualización que permita verificar que el balance de párrafos de cada personaje es similar en train y test.



3. Transforme el texto del conjunto de entrenamiento a la representación numérica (features) de conteo de palabras o bag of words. Explique brevemente cómo funciona esta técnica y muestre un ejemplo. En particular explique el tamaño de la matriz resultante, y la razón por la que es una sparse matrix.

Bag of Words es una técnica básica del procesamiento de lenguaje natural (NPL) que nos permite analizar la tokenización de palabras.
El texto es un conjunto de datos no estructurado y para que una computadora lo pueda interpretar es necesario poder estructurarlo. Bag of Words nos permite tomar nuestro texto y separarlo y dividirlo en tokens. Cada token podría verse como una palabra o una representación de la palabra, y la idea es extraer información de nuestros textos a través de

estos tokens para conocer, por ejemplo, la cantidad de palabras que hay en un texto y la frecuencia con la cual se repiten.
CountVectorizer nos ayuda a crear nuestra Bag of Words, creando una matriz dispersa es posible trabajar de manera óptima cuando tenemos muchos datos, debido a que la cantidad de memoria usada depende de la cantidad de palabras que estemos analizando, y esta función lo convierte en matrices donde las palabras aparecen como valores (0 y 1).

En el ejercicio, utilizando la siguiente linea de comandos:

```
count_vect = CountVectorizer(stop_words=None, ngram_range=(1,1))
X_train_counts = count_vect.fit_transform(X_train)
X_train_counts
```

Se obtiene una matriz de 438x2807, es decir, que hay 438 filas (documentos) y 2807 columnas (palabras únicas) en el conjunto de training. El resultado también arroja que la matriz presenta 10831 elementos no nulos.

4. Explique brevemente qué es un n-grama. Obtenga la representación numérica Term Frequency - Inverse Document Frequency. Explique brevemente en qué consiste esta transformación adicional.

Un n-grama es un conjunto de *n* elementos consecutivos en un documento de texto, que puede incluír palabras, números, símbolos y puntuación. Los modelos de n-gramas son útiles en los análisis de texto donde la secuencia de palabras es relevante. Su modelado es una de las técnicas que permite convertir el texto de un formato no estructurado a un formato estructurado.
Si contamos la cantidad de veces que cada n-grama único aparece en un determinado documento podemos crear un modelo lingüístico, esto se conoce como bolsa de n-gramas.

Por su parte, TF-IDF es una técnica que evalúa la relevancia de cada palabra dentro de un documento o conjunto de documentos, combinando dos métricas: Term Frequency (TF) que aprecia la frecuencia de una determinada palabra, e Inverse Document Frequency (IDF) que tiene en cuenta la relevancia de dicha palabra en el texto. Por lo tanto, esta técnica, además de contar la frecuencia de las palabras como también lo hace Bag of Words, también lo relaciona con la importancia de las mismas en el texto o conjunto, lo que permite dar más peso a las palabras relevantes y menos a aquellas comunes, permitiendo un modelo que capture mejor el significado de los datos.

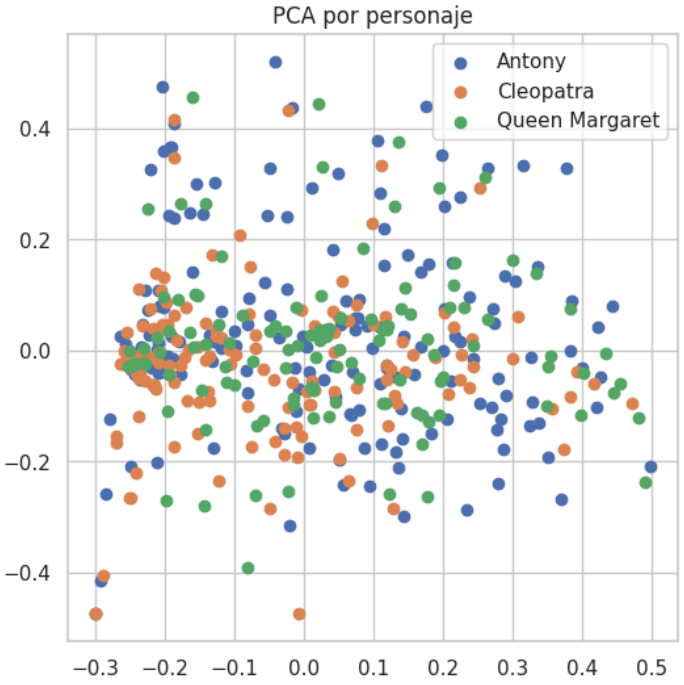
Resultados similares a los obtenidos en la línea de comandos del punto anterior se obtuvieron al utilizar esta técnica:

```
tf_idf = TfidfTransformer(use_idf=False)
X_train_tf = tf_idf.fit_transform(X_train_counts)
X_train_tf
```

5. Muestre en un mapa el conjunto de entrenamiento, utilizando las dos primeras componentes PCA sobre los vectores de tf-idf. Analice los resultados y compare qué sucede si utiliza el filtrado de stop_words para idioma inglés, el parámetro

use_idf=True y ngram_range=(1,2). Opcionalmente, también puede analizar qué sucede si no elimina los signos de puntuación.
● ¿Se pueden separar los personajes utilizando sólo 2 componentes principales?
● Haga una visualización que permita entender cómo varía la varianza explicada a medida que se agregan componentes (e.g: hasta 10 componentes).

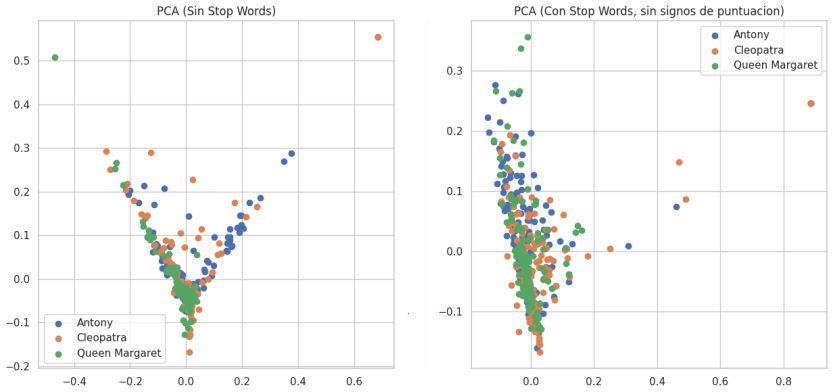
El PCA es una técnica de reducción de dimensionalidad que transforma los datos originales a un nuevo conjunto de variables, llamadas “componentes principales”. Por lo general, las primeras componentes principales capturan la mayor parte de la variabilidad en los datos.
El siguiente gráfico de dispersión de los datos reducidos está hecho en base a las dos primeras componentes principales (la primera en el eje X y la segunda en el eje Y), estos ejes permiten ver las combinaciones lineales de las variables que maximizan la varianza de los datos. Por lo tanto, cada punto en la gráfica representa una observación en el espacio de las componentes principales, para cada personaje.
En el gráfico de dispersión realizado a partir de dos componentes principales no se encuentran grandes diferencias entre los personajes. Se observa una mayor concentración de los puntos pertenecientes a Cleopatra y Antony hacia la izquierda, mientras que los puntos pertenecientes a Queen Margaret se ven distribuídos de manera uniforme en el gráfico. Esto podría mostrar una similitud en los textos correspondientes a los diálogos de estos personajes, por ejemplo, que compartan características similares en su vocabulario o expresiones, lo cual es capturado por los componentes principales del PCA, a diferencia de los guiones de Queen Margaret, que parece presentar una mayor variabilidad en sus expresiones.



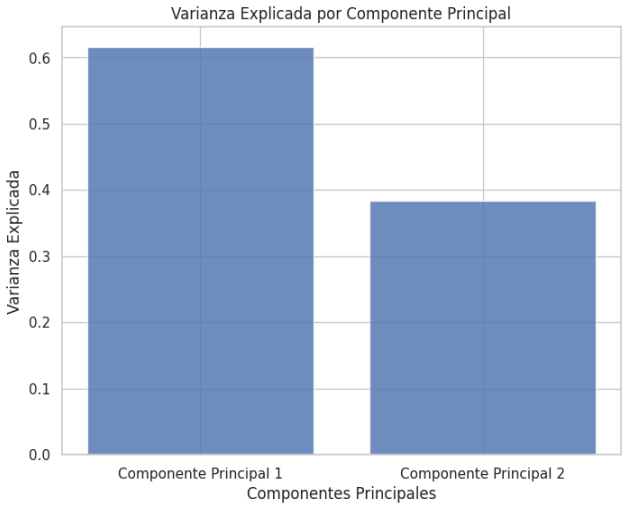
Las *stop words* (palabras vacías) son palabras que normalmente pueden (o no) filtrarse o eliminarse del texto durante el procesamiento de datos, dependiendo del análisis que estemos realizando y el modelo elegido. Estas palabras son comunes en el idioma y generalmente no aportan un significado adicional al contenido del texto. Ejemplos típicos de stop words en inglés incluyen palabras como "a", "an", "the", "in", "on", "at", etc. El propósito

de eliminar las stop words durante el preprocesamiento de texto es mejorar la calidad del análisis al reducir la dimensionalidad del espacio de características y eliminar el ruido innecesario que no contribuye significativamente a la semántica del texto. Esto puede resultar en modelos más eficientes y precisos al centrarse en palabras que tienen un mayor peso.

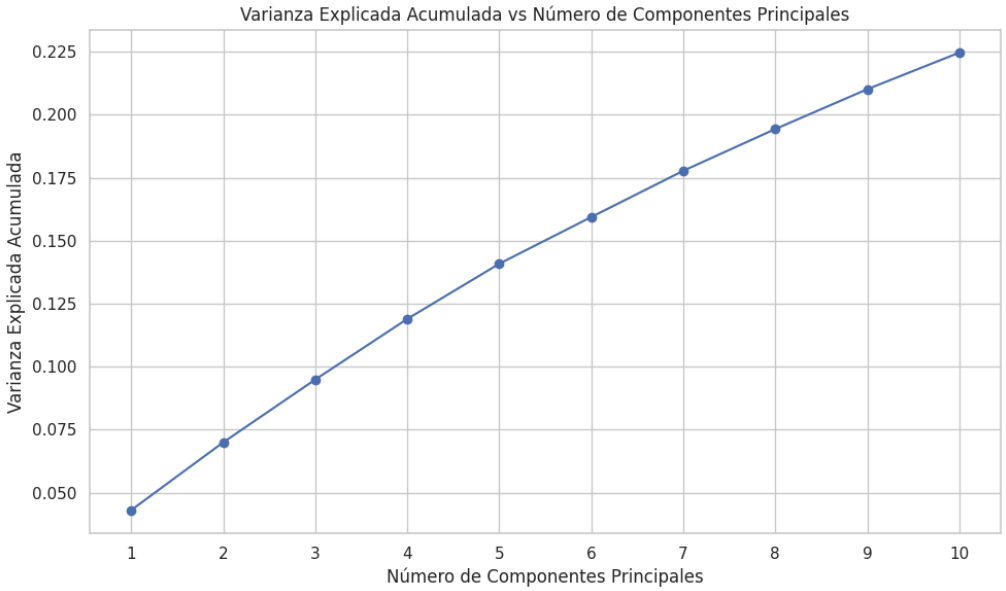
Al eliminar las stop words (primero de los tres gráficos siguientes) vemos que los puntos tienden a acumularse en el punto 0,0 del Primer Componente y luego se dispersan hacia arriba en forma de V, principalmente para Antony y en menor medida para Cleopatra. esto seguramente se deba a la reducción del ruido causado anteriormente por la presencia de estas palabras de menor relevancia. A diferencia de los otros, las palabras correspondientes a Queen Margaret tienden a acumularse hacia la izquierda. Por lo que, esto permite diferenciar un poco mejor las características de los datos de cada personaje.
Si dejamos las stop words pero variamos los signos de puntuación vemos una marcada acumulación de los puntos hacia la izquierda para los tres personajes. Esto podría demostrar cómo se reduce la variabilidad de las palabras cuando contemplamos las palabras comunes (que se repiten con mucha frecuencia). Por otra parte, también nos muestra que existe cierta dominancia de las stop words en los textos analizados. Se remarca la diferencia del análisis teniendo en cuenta las stop words y sin ellas, y los resultados tan diferentes que se pueden obtener con ello.



Al observar la varianza explicada, los resultados indican que los dos componentes principales obtenidos son capaces de capturar la mayoría de la variabilidad en los datos originales, lo cual tomamos como un buen indicio para poder separar efectivamente las clases (en este caso, los personajes) utilizando estas dos dimensiones principales (Varianza explicada acumulada: 0.616).



Las gráficas de varianza explicada acumulada permiten ver cómo se acumula la varianza por cada componente adicional. Podemos ver cómo la curva va en aumento a medida que se agregan más componentes, con una pendiente al principio ya que los primeros componentes explican la mayor parte de la varianza total. A medida que se van agregando componentes la pendiente tiende a volverse constante, ya que cada nuevo componente explica una proporción más pequeña de la varianza restante.

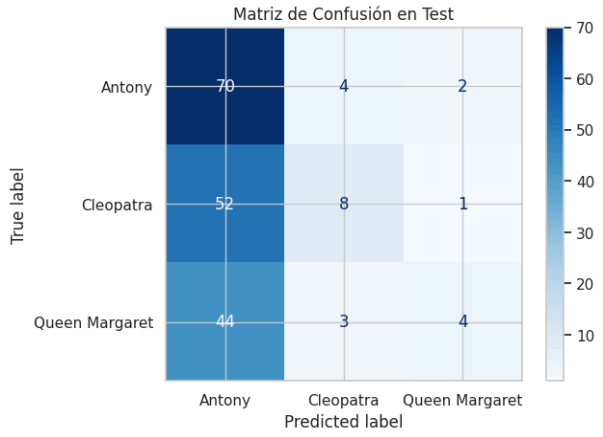


Parte 2: Entrenamiento y Evaluación de Modelos

1. Entrene el modelo Multinomial Naive Bayes, luego utilícelo para predecir sobre el

conjunto de test, y reporte el valor de accuracy y la matriz de confusión. Reporte el valor de precision y recall para cada personaje. Explique cómo se relacionan estos valores con la matriz anterior.
● ¿Qué problemas puede tener el hecho de mirar sólamente el valor de accuracy?
● Considere qué sucedería con esta métrica si el desbalance de datos fuera aún mayor entre personajes.
● Sugerencia: utilice el método from_predictions de ConfusionMatrixDisplay para realizar la matriz.

Se obtuvieron los siguientes resultados:



	precision	recall	f1-score	support
Antony	0.42	0.92	0.58	76
Cleopatra	0.53	0.13	0.21	61
Queen Margaret	0.57	0.08	0.14	51
accuracy			0.44	188
macro avg	0.51	0.38	0.31	188
weighted avg	0.50	0.44	0.34	188

El modelo de Naive Bayes multinomial tiene un *accuracy* de aproximadamente 0,646 en el conjunto de entrenamiento. Esto indica que, en promedio, el modelo predice correctamente alrededor del 64.6% de las etiquetas de los datos de entrenamiento, que es un rendimiento aceptable pero está lejos de la excelencia.
Tener en cuenta solo el calor del accuracy puede llevar a errores de interpretación en el modelo, ya que un accuracy elevado no significa que el modelo esté funcionando bien en todas las clases, el mismo puede fallar, principalmente en aquellas clases minoritarias. Por otra parte, hay que tener presente que el modelo utilizado puede estar sobreajustado (overfitted), capturando mayor ruido. Comparar el accuracy entre el conjunto de entrenamiento y prueba puede ayudar a identificar este problema.

La *precisión* y el *recall* son métricas que permiten evaluar el modelo además del accuracy. Así también, vienen de la mano con la matriz de confusión, la cual detalla los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos para cada clase en estudio. Esta gráfica nos permite ver qué tan buenas predicciones está haciendo el modelo clasificador. La precisión mide la exactitud de las predicciones positivas hechas por el modelo para cada clase (número de verdaderos positivos dividido el total de verdaderos positivos y falsos positivos). Por otro lado, el recall o sensibilidad mide la capacidad del modelo para encontrar todos los positivos reales para cada clase. Por lo tanto, la matriz de confusión es la que proporciona los valores necesarios para el cálculo de estas dos métricas.

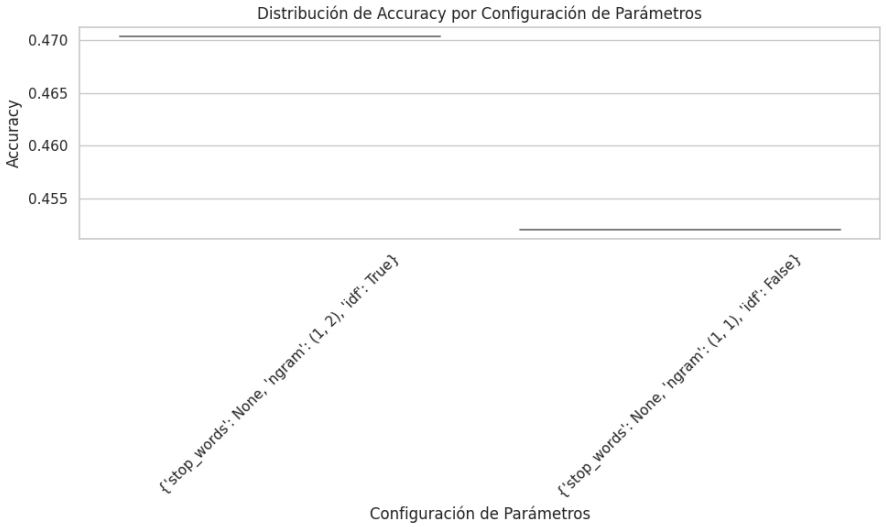
Observando de a uno, podemos ver que Antony presenta un alto recall (0,92), lo que significa que el modelo utilizado es bueno identificando correctamente la mayoría de los positivos reales de este personaje, pero la baja precisión (0,42) nos dice que el modelo tiene una alta tasa de falsos positivos (es decir, que muchos datos de los otros personajes son identificados como si fueran de Antony, incorrectamente, lo que podría significar que el modelo esté sobreajustado).
Por su parte, Cleopatra presenta bajo recall y una precisión bastante baja (0,13 y 0,53 respectivamente). Y por último, Queen Margaret también presenta bajos estos parámetros (recall de 0,08 y precisión de 0,57), por lo que al modelo le cuesta mucho identificar correctamente los datos de este personaje. Por lo tanto, el modelo parece ser eficiente identificando a Antony, pero no a los otros personajes.

2. Explique cómo funciona la técnica de validación cruzada o cross-validation. Interprete y complete el código de ejemplo para la búsqueda de hiper-parámetros. Genere una
visualización que permita comparar las métricas (e.g: accuracy) de los distintos modelos entrenados, viendo el valor promedio y variabilidad de las mismas en todos los splits
(e.g: en un gráfico de violín).

La Validación Cruzada permite evaluar el rendimiento de un modelo. Implica dividir el conjunto de datos (finito) en partes iguales (k partes) llamadas folds. Luego, se entrena y se evalúa el modelo k veces utilizando un fold diferente como conjunto de validación en cada iteración y el resto como conjunto de entrenamiento. Esto permite obtener múltiples medidas de rendimiento del modelo y reduce el riesgo de sobreajuste al usar diferentes particiones de datos. Se obtuvieron los siguientes resultados:

```
Parámetros: {'stop_words': None, 'ngram': (1, 2), 'idf': True}
Accuracy promedio: 0.4704 ± 0.0282

Parámetros: {'stop_words': None, 'ngram': (1, 1), 'idf': False}
Accuracy promedio: 0.4521 ± 0.0275
```

Por lo que, al parecer, el uso de bigramas y aplicar la transformación TF-IDF es beneficioso para este modelo, mejorando el accuracy promedio levemente que cuando usamos unigramas y sin aplicar IDF. Además, las desviaciones estándar son similares indicando que la variabilidad del rendimiento entre las diferentes particiones de validación cruzada es comparable entre los dos conjuntos.



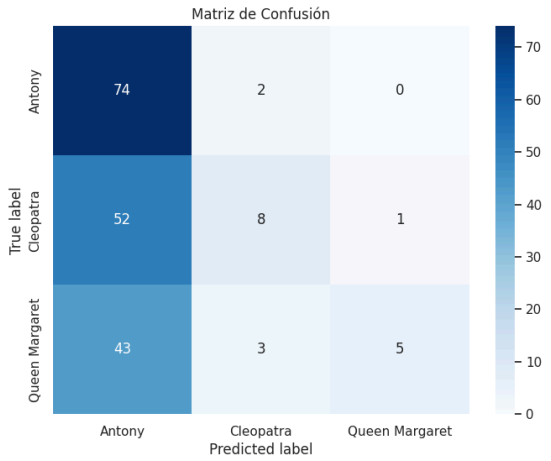
Nota al lector: no conocemos los comandos correctos para hacer el gráfico de violines correspondiente.

3. Elija el mejor modelo (mejores parámetros) y vuelva a entrenar sobre todo el conjunto de entrenamiento disponible (sin quitar datos para validación). Reporte el valor final de las métricas y la matriz de confusión. Discuta las limitaciones de utilizar un modelo basado en bag-of-words o tf-idf en cuanto al análisis de texto.

A partir de los resultados de la Validación cruzada pudimos determinar que los mejores parámetros para nuestro conjunto de datos eran los siguientes:

```
Parámetros: {'stop_words': None, 'ngram': (1, 2), 'idf': True}
Accuracy promedio: 0.4704 ± 0.0282
```

Por lo que entrenamos el modelo Naive Bayes multinomial utilizando estos mismos. Se obtuvieron los siguientes resultados:



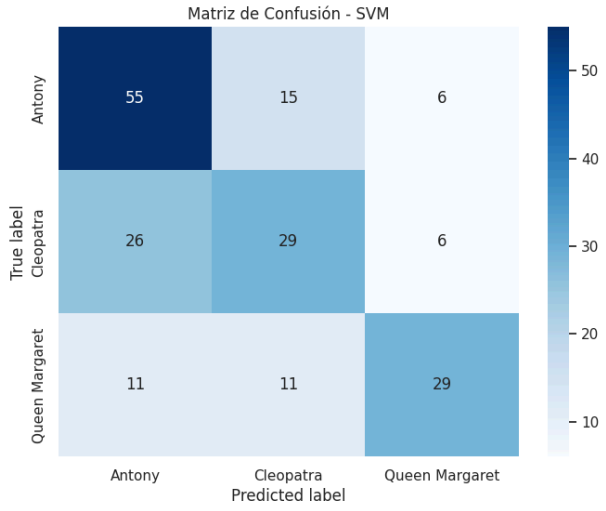

```
Reporte de Clasificación:
                precision    recall  f1-score   support

       Antony       0.44      0.97      0.60        76
    Cleopatra       0.62      0.13      0.22        61
Queen Margaret      0.83      0.10      0.18        51

     accuracy                           0.46       188
    macro avg       0.63      0.40      0.33       188
 weighted avg       0.60      0.46      0.36       188
```

Comparando con la primera matriz de confusión obtenida, podemos ver que el ajuste de los parámetros mejora la precisión con la que el modelo logra identificar a Antony, ya que el aumento de predicciones correctas pasó de 70 a 74 y se redujo el número de errores. Esto se ve reflejado en el aumento de la precisión y el recall para este personaje.
Por otra parte, los valores de Queen Margaret presentan una leve mejora, vemos que aumentó significativamente la precisión pero el recall no tuvo grandes variaciones.
Respecto a Cleopatra, no vemos cambios relevantes.
Por lo tanto, el cambio de los parámetros ha mostrado una mejora en la precisión, principalmente para los primeros dos personajes. Aún así, si bien el modelo presentó mejoras, la confusión entre los personajes sigue siendo significativa.

Es necesario tener en cuenta que, al tratar con modelos basados en Bag of Words no se contempla el contexto en el que aparecen las palabras, cosa que muchas veces es necesaria para la correcta comprensión durante el análisis de datos. Si bien TF-IDF mejora un poco esto, priorizando las palabras de mayor relevancia, no logra capturar las relaciones contextuales entre las palabras. En ambas puede aparecer ruido y sobreajuste. Por último, es necesario tener formas de representar los datos obtenidos en representaciones gráficas no muy complejas, ya que de por sí ambas técnicas presentan representaciones que se pueden volver complejas si el número de palabras es muy grande, dificultando también de por sí el entrenamiento muchas veces.

4. Evalúe al menos un modelo más (dentro de scikit-learn) aparte de Multinomial Naive Bayes para clasificar el texto utilizando las mismas features de texto. Explique brevemente cómo funciona y compare los resultados con el anterior.

Se implementó un modelo SVM (Support Vector Machine) es un algoritmo de aprendizaje supervisado que busca, al igual que el modelo anterior, encontrar la mejor manera de separar diferentes clases en un espacio multidimensional. Es un modelo más complejo y flexible que el Naive Bayes multinomial, también requiere tener una idea clara del ajuste de hiper-parámetros para su correcto uso. Como ventaja, presenta mayor precisión y menor riesgo de sobreajuste, pero tambièn una complejidad computacional elevada cuando se trata de conjuntos de datos muy grandes.

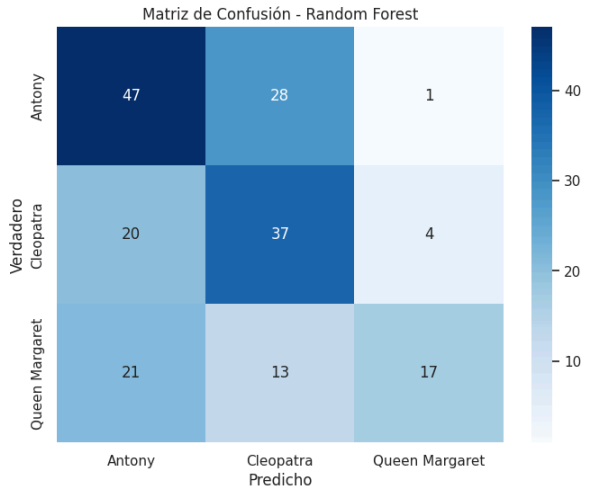



```
Resultados del SVM:
Accuracy: 0.6011

Matriz de Confusión:
[[55 15  6]
 [26 29  6]
 [11 11 29]]

Reporte de Clasificación:
                precision    recall  f1-score   support

       Antony       0.60      0.72      0.65        76
    Cleopatra       0.53      0.48      0.50        61
Queen Margaret      0.71      0.57      0.63        51

     accuracy                           0.60       188
    macro avg       0.61      0.59      0.60       188
 weighted avg       0.60      0.60      0.60       188
```

Comparando los resultados obtenidos del modelo SVM con los obtenidos utilizando Naive Bayes multinomial anteriormente, podemos ver claramente cómo las métricas de Antony disminuyen. es decir que Naive Bayes clasifica con mayor precisión a este personaje. Sin embargo, el modelo SVM presenta una mayor precisión al clasificar a Cleopatra y Queen Margaret.

Por otra parte, también probamos con el modelo Random Forest. A diferencia de Naive Bayes y SVM que son modelos individuales, este algoritmo funciona construyendo múltiples árboles de decisión durante el entrenamiento y promediando los resultados para obtener un resultado final. Normalmente, suele usarse cuando se tienen conjuntos de datos grandes y complejos, y no se conoce mucho sobre los mismos.

Se obtuvieron los siguientes resultados:




```
Resultados del Random Forest:
Accuracy: 0.5372

Matriz de Confusión:
[[47 28  1]
 [20 37  4]
 [21 13 17]]

Reporte de Clasificación:
                precision    recall  f1-score   support

       Antony       0.53      0.62      0.57        76
    Cleopatra       0.47      0.61      0.53        61
Queen Margaret      0.77      0.33      0.47        51

     accuracy                           0.54       188
    macro avg       0.59      0.52      0.52       188
 weighted avg       0.58      0.54      0.53       188
```

Vemos que Random Forest tiene un desempeño inferior a Naive Bayes pero mejor que SVM al realizar la clasificación de Antony. Para Cleopatra las métricas obtenidas son mejores que con Naive Bayes, y no muy diferentes de los resultados obtenidos con SVM. Por último, para Queen Margaret se observa un rendimiento de Random Forest que es menor que el obtenido con SVM pero mucho mejor que los resultados de Naive Bayes.

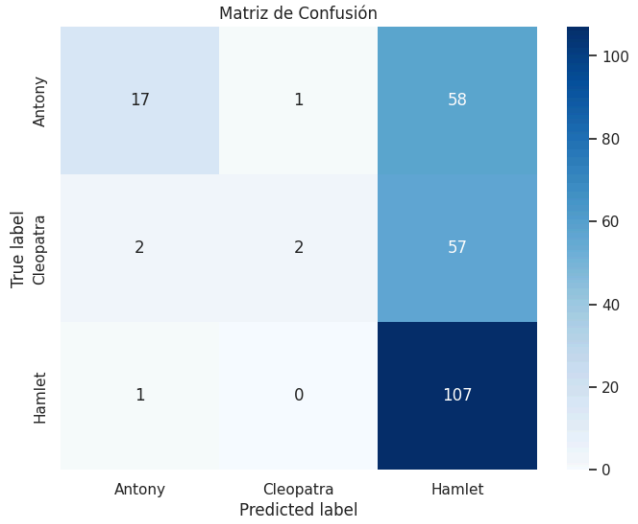
5. Evalúe el problema cambiando al menos un personaje. En particular, observe el (des)balance de datos y los problemas que pueda generar, así como cualquier indicio que pueda ver en el mapeo previo con PCA. Puede ser útil comentar acerca de técnicas como sobre-muestreo y submuestreo, no es necesario implementarlo.

Cambiamos el personaje de Queen Margaret por Hamlet.

El accuracy del Naive Bayes multinomial es de 0,619. Al ajustarlo mediante Cross-Validation obtuvimos los siguientes resultados:

```
Parámetros: {'stop_words': None, 'ngram': (1, 2), 'idf': True}
Accuracy promedio: 0.4701 ± 0.0252

Parámetros: {'stop_words': None, 'ngram': (1, 1), 'idf': False}
Accuracy promedio: 0.4824 ± 0.0262
```

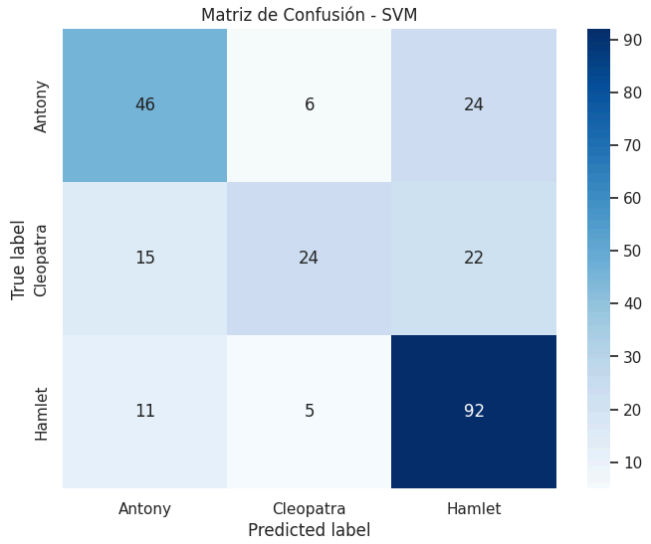
Y, con estos parámetros mejorados, obtuvimos la siguiente matriz:



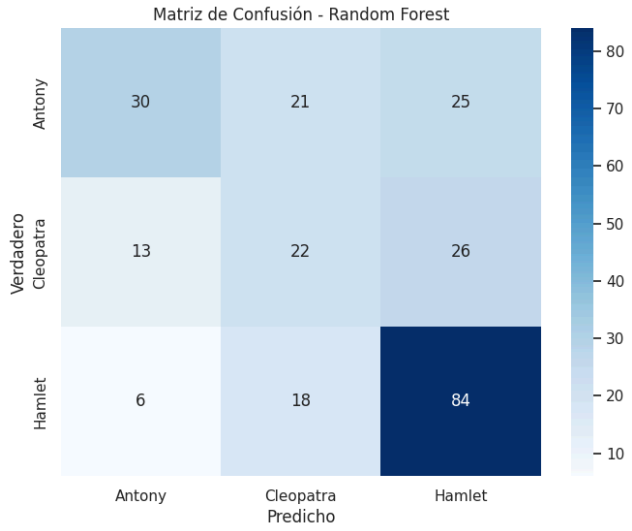

```
Accuracy: 0.5143

Matriz de Confusión:
[[ 17   1  58]
 [  2   2  57]
 [  1   0 107]]

Reporte de Clasificación:
              precision    recall  f1-score   support

      Antony       0.85      0.22      0.35        76
   Cleopatra       0.67      0.03      0.06        61
      Hamlet       0.48      0.99      0.65       108

    accuracy                           0.51       245
   macro avg       0.67      0.42      0.36       245
weighted avg       0.64      0.51      0.41       245
```

El modelo muestra una alta precisión para Antony, pero un recall bastante bajo para este personaje; muestra métricas bastante bajas para Cleopatra y un recall significativo para Hamlet, pero la precisión es bastante baja. Por lo que, el modelo parece ajustarse bien a Hamlet pero no a los demás personajes, siendo Cleopatra la más dificil de predecir. Estos resultados pueden indicar que las clases no están representadas de manera equitativa en el conjunto de datos, pudiendo producirse sesgos e inclinarse hacia la clase mayoritaria.

Al analizarlo mediante SVM y Random Forest:




```
Resultados del SVM:
Accuracy: 0.6612

Matriz de Confusión:
[[46  6 24]
 [15 24 22]
 [11  5 92]]

Reporte de Clasificación:
              precision    recall  f1-score   support

      Antony       0.64      0.61      0.62        76
   Cleopatra       0.69      0.39      0.50        61
      Hamlet       0.67      0.85      0.75       108

    accuracy                           0.66       245
   macro avg       0.66      0.62      0.62       245
weighted avg       0.66      0.66      0.65       245
```




```
Resultados del Random Forest:
Accuracy: 0.5551

Matriz de Confusión:
[[30 21 25]
 [13 22 26]
 [ 6 18 84]]

Reporte de Clasificación:
              precision    recall  f1-score   support

      Antony       0.61      0.39      0.48        76
   Cleopatra       0.36      0.36      0.36        61
      Hamlet       0.62      0.78      0.69       108

    accuracy                           0.56       245
   macro avg       0.53      0.51      0.51       245
weighted avg       0.55      0.56      0.54       245
```

En estos casos vemos que SVM muestra el mejor desempeño para este conjunto de datos, con un accuracy de 66.12%. Este modelo tiene una distribución más equilibrada y los otros dos presentan sobreajustes, sesgados a la clase mayoritaria. Esto también se ve reflejado en las métricas de cada personaje en los distintos modelos.

6. Busque información sobre al menos una técnica alternativa de extraer features de texto.
Explique brevemente cómo funciona y qué tipo de diferencias esperaría en los resultados. No se espera que implemente nada en esta parte.

Una técnica alternativa podría ser Word Embeddings. Esta técnica representa las palabras como vectores de números en un espacio de varias dimensiones. La técnica es capaz de capturar el contexto, la similitud semántica y sintáctica (género, sinónimos, etc) de una palabra, creando vectores donde las palabras con significados similares estén cerca unas de otras. Word2Vec, GloVe y FastText son técnicas comunes para crear estos vectores.
Como veíamos hoy, Bag of Words y TF-IDF no tienen en cuenta el contexto y la semántica de las palabras en el texto como tal, pero Word Embeddings sí, lo que puede mejorar la precisión de los modelos significativamente. Además, los vectores son más compactos, por lo que la cantidad de datos se reduce pero sin perder información relevante para los análisis. Como posible desventaja, hay que tener en cuenta que para entrenar buenos modelos de esta técnica se necesitan grandes volúmenes de texto.