Girmar Laucho
Magdalena Romano

Tarea 2 - Introducción a la Ciencia de Datos
FING, 2025

Parte 1: Dataset y representación numérica de texto

1.

Antes de comenzar con el análisis es importante destacar algunas precisiones sobre los datos y la limpieza de los mismos. Como se detalló en la tarea 1 se tomó la decisión de trabajar únicamente los discursos del tipo “campaign speech” cuyas características en relación al lenguaje, cantidad de palabras y temas abordados son algo distintas a lo que pueden ser otras modalidades de discursos más interactivos. Para esta segunda tarea se mantuvo esta línea de trabajo y también se aplicó la función de limpieza de texto utilizada anteriormente.

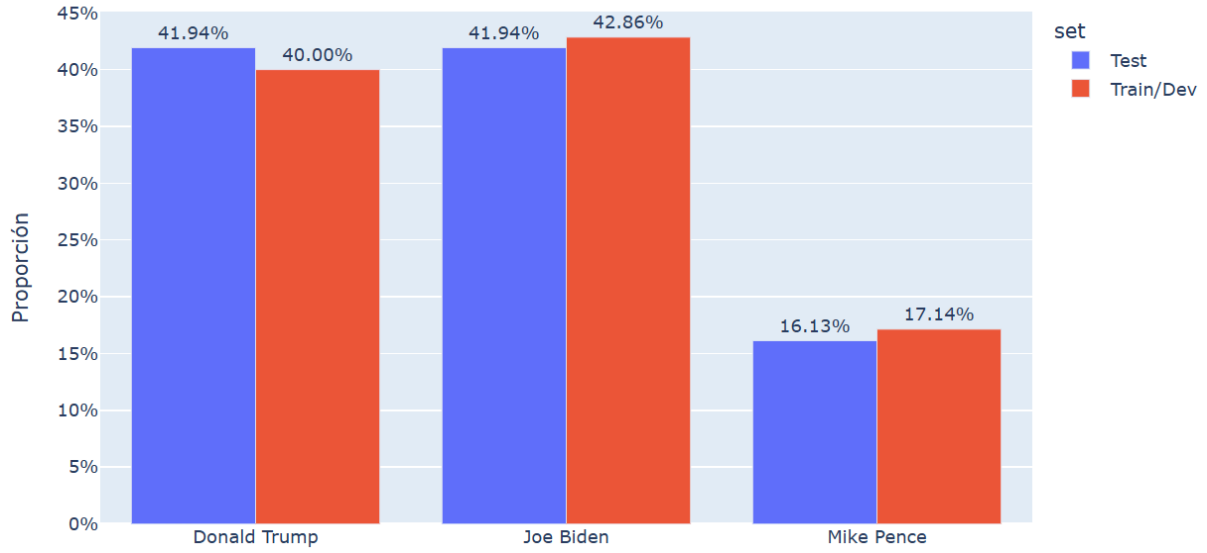
En esta limpieza se quitaron datos de los momentos en que se capturaba cada parte del discurso así como el nombre del orador en esos lugares específicos del texto; se normalizó el texto convirtiendo todas las palabras a minúsculas; a la función se le añadió eliminar paréntesis, espacios múltiples y signos de puntuación.

Es así entonces que para esta tarea se trabajó con una base de datos de 101 discursos de campaña que contenía los datos del texto limpio, en la variable denominada “CleanText” y el speaker correspondiente, para los 3 candidatos con mayor presencia.

2.

Estos datos se particionaron para generar un conjunto de test del 30% del total (31 documentos o discursos), utilizando muestreo estratificado. Entonces se forma un conjunto de entrenamiento, con 70 discursos y uno de test con 31 estratificados por speaker. Esto implica que los grupos formados mantienen las mismas proporciones entre sí, de acuerdo a esta variable y también acorde a lo que ocurría en el conjunto original. Se puede ver en el gráfico de la figura 1 que el balance de discursos de cada candidato es similar en los conjuntos de entrenamiento y testeo.

Figura 1: Balance de discursos de cada candidato en los conjuntos de train y test.



3.

Posteriormente se transformó el texto del conjunto de entrenamiento a la representación numérica de conteo de palabras o **bag of words**. Esta técnica consiste en contar en cada texto (o discurso) la presencia o ausencia de cada una de las palabras de un diccionario. Si el diccionario contiene M palabras, a cada texto se le asocia un vector de longitud M, se coloca 1 para cada palabra presente o la cantidad de ocurrencias que tiene en el texto y 0 si no está. Es posible utilizar un diccionario con las 10000 palabras más frecuentes de nuestro corpus de texto de entrenamiento. Nuestro conjunto de entrenamiento se compone de 70 textos y 11951 palabras en este corpus (aquí trabajaremos con esa cantidad, ya que no es sustantivamente mayor).

Al utilizar la técnica mencionada anteriormente la información de un texto o discurso quedará reducido a un vector de longitud de 11951 compuesto principalmente de 0 y valores mayores o iguales a 1 en las posiciones correspondientes a las palabras que están presentes en ese texto. La matriz de entrenamiento resultante tiene una dimensión de 70 x 11951, pero solo una parte de las entradas son distintas de cero. Por ejemplo, en el primer discurso solo hay 1043 palabras distintas, significa que en la primera fila hay 1043 elementos mayores o iguales a 1 (8,7%) y el resto 0. En toda la matriz la proporción de valores no nulos es 0,107 (10,7%).

Por eso esta matriz es sparse, ya que la mayoría de los valores son cero. Se puede almacenar eficientemente en formato de matriz sparse (en lugar de almacenar la matriz completa), almacenando la ubicación y los valores de las entradas distintas de cero. La figura 2 permite ejemplificar cómo funciona este almacenamiento. En el documento 0 (primer discurso) la palabra específica de nuestro diccionario ubicada en la columna 11658 aparece 5 veces en este documento. Esta forma de

almacenamiento reduce enormemente el uso de memoria RAM, ya que no se almacenan explícitamente una gran cantidad de ceros.

Figura 2: Ejemplos de valores almacenados en el documento 0, con la técnica BoW.

Coords	Values
(0, 11658)	5
(0, 5146)	2
(0, 5796)	24
(0, 5834)	50
(0, 4855)	18
(0, 10825)	161

4.

Un **n-grama** es una secuencia de n palabras. Los n-gramas, a diferencia de los unigramas (cuando se considera únicamente una palabra) permiten tener en cuenta algo del contexto, su contexto inmediato, aunque como desventaja se aumenta la dimensionalidad. En los modelos de n-gramas se estima la probabilidad de una palabra dadas las n-1 palabras previas, por lo que permiten asignar probabilidades a frases enteras.

En nuestro problema los n-gramas pueden ser útiles para el análisis de secuencias de los discursos de los candidatos, ya que pueden capturar patrones locales de los candidatos en el texto. Es posible que frases comunes de los candidatos, por ejemplo “alternative facts” de Trump, den más información para asociar más fácilmente con su candidato que si se observan como palabras independientes.

La técnica TF-IDF consiste en lograr una representación numérica que transforma un texto en vectores numéricos (como se explicó antes), con el adicional que refleja la importancia de un término (o una secuencia de palabras, en el caso de los n-gramas) en un documento teniendo en cuenta la colección entera de documentos. Es decir, pondera la frecuencia de la palabra viendo qué tan común es en general de todos los documentos. Con el TF-IDF palabras frecuentes en un documento pero raras en el corpus obtienen alto peso, mientras que las comunes quedan con peso bajo.

Al trabajar con unigramas y bigramas mediante *TfidfVectorizer*, obtuvimos que la cantidad total de términos de nuestro diccionario es de 149780, por lo que el tamaño de nuestra matriz TF-IDF de entrenamiento resulta de 70 x 149780, donde como se explicó anteriormente cada fila representa un documento, cada columna un término (unigrama o bigrama) del diccionario, pero en este caso los valores de la matriz son los pesos TF-IDF ponderado según su frecuencia y relevancia.

5.

A continuación se muestran mapas PCA y se analizan las diferencias en estas representaciones teniendo en cuenta parámetros relacionados al filtrado de stopwords, al uso de idf y de n-gramas. En primer lugar, comparando las figuras 3.1 y 3.2, ambas trabajan únicamente con unigramas y sin filtrado de stopwords y varían en el uso del IDF. En el global no se observan grandes diferencias, aunque en el primer caso parecen mezclarse un poco más algunos discursos de Pence y Biden. A su vez teniendo en cuenta las figuras 3.2 y 3.3, que comparan según el filtrado de stopwords, se identifican en la figura 3.3 clusters algo más compactos particularmente para los datos de Biden y Trump. Esto tiene sentido ya que al filtrar los stopwords, es decir, eliminar palabras comunes que no aportan significado, se reflejan mejor las diferencias semánticas entre documentos. Por otro lado, al comparar las figuras 3.3, 3.4 y 3.5, donde se introduce el análisis de los textos utilizando bigramas también, parecen verse aún más compactos los clusters, por lo que podemos suponer que hay bigramas significativos en este contexto.

Figura 3: Mapas PCA para las distintas configuraciones

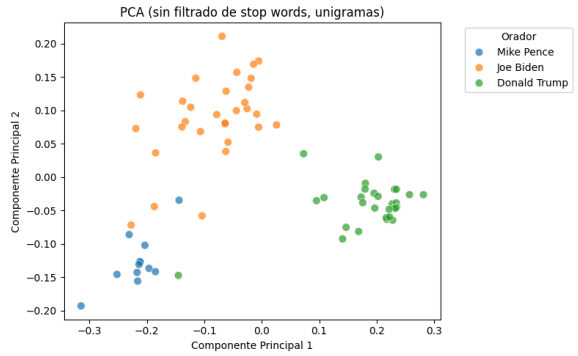


Figura 3.1: Mapa PCA sin filtrado de stopwords, sin uso de IDF, y unigramas

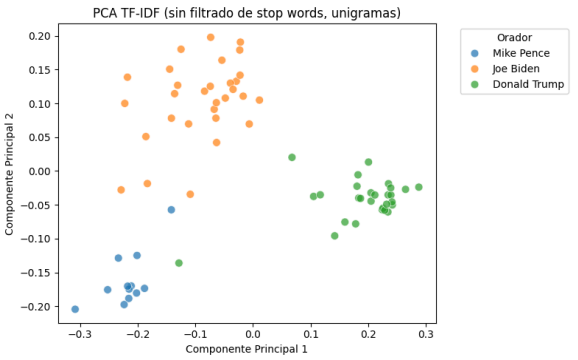


Figura 3.2: Mapa PCA sin filtrado de stopwords, con uso de IDF, y unigramas

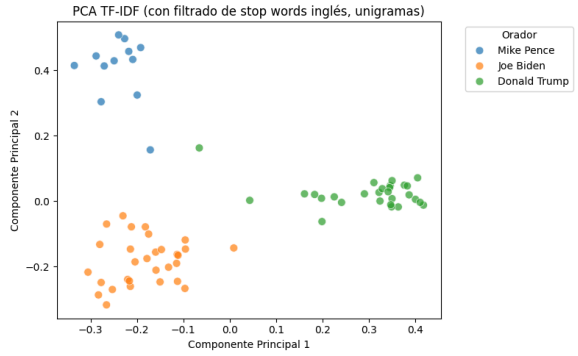


Figura 3.3: Mapa PCA con filtrado de stopwords, con uso de IDF, y unigramas

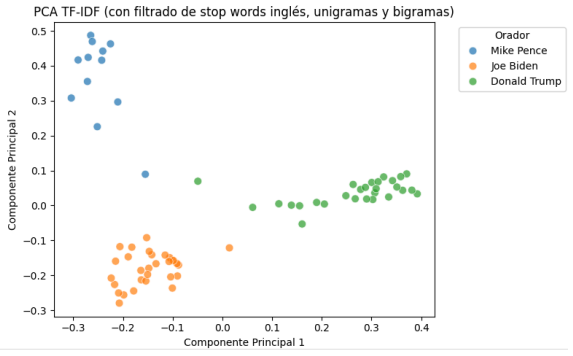


Figura 3.4: Mapa PCA con filtrado de stopwords, con uso de IDF, unigramas y bigramas

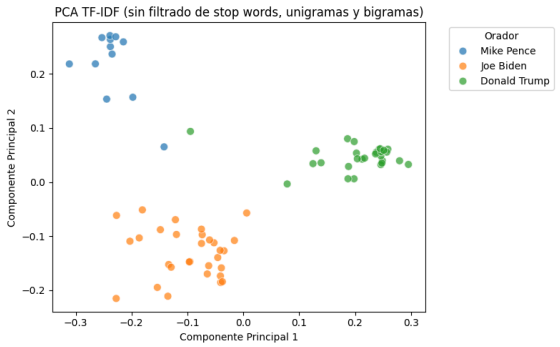
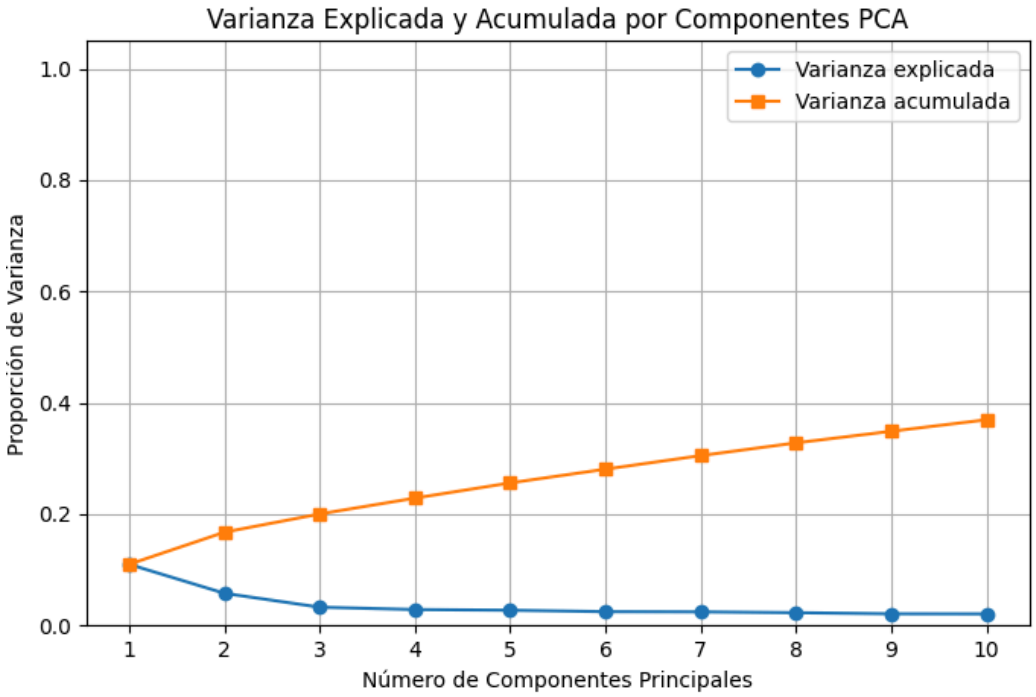


Figura 3.5: Mapa PCA sin filtrado de stopwords, con uso de IDF, unigramas y bigramas

Los distintos mapas PCA observados sugieren que en general se podría separar los candidatos usando únicamente las dos primeras componentes, ya que se ven los clusters bastante diferenciados. Para aportar a este análisis la figura 4 muestra que las dos primeras componentes explican solo menos de un 20% de la varianza total. Este análisis se realizó para el caso de no filtrado de stopwords, uso de IDF y unigramas y bigramas (el PCA correspondiente sería el de la figura 3.5). Esto puede sugerir que, a pesar de la baja varianza que capturan las dos primeras componentes, puede ser que estas mantengan información relevante para la tarea de clasificación.

Figura 4: Varianza explicada y acumulada por componentes PCA



Girmar Laucho
Magdalena Romano

Parte 2: Entrenamiento y Evaluación de Modelos

1.

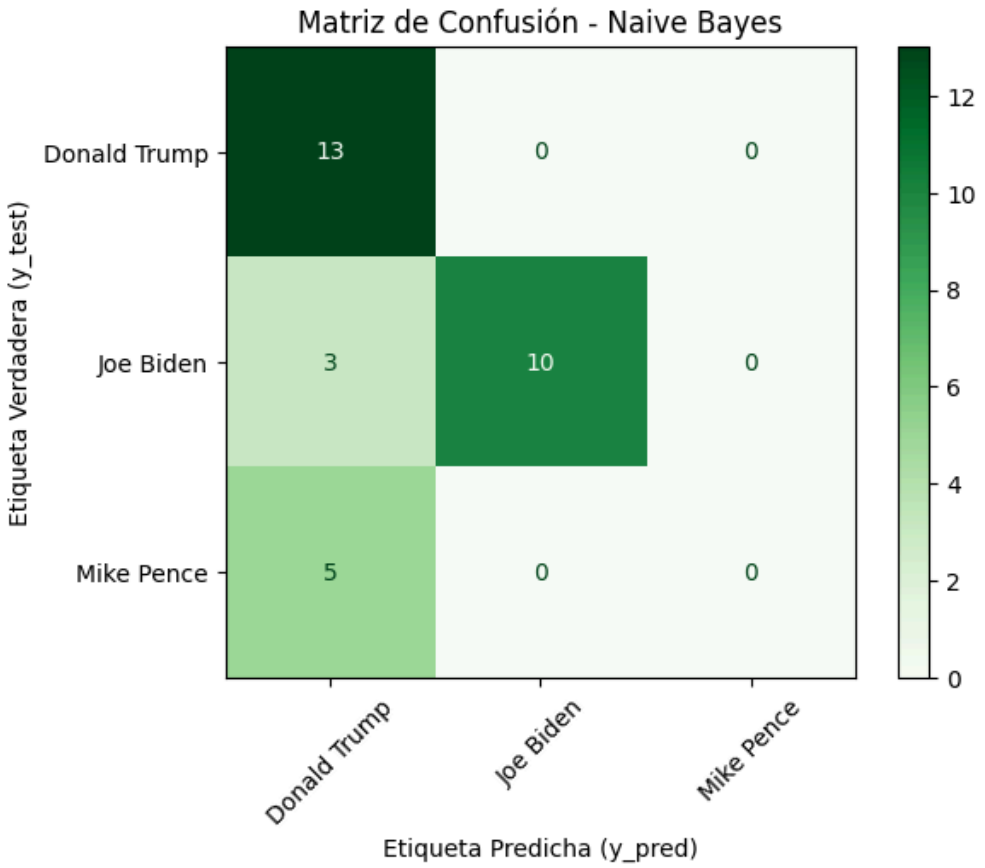
Según IBM (s.f.), el modelo multinomial Naive Bayes tiene como fundamento ser un clasificador probabilístico, que determina la probabilidad que un evento suceda tomando en consideración la ocurrencia de otro evento, esto permite que se puedan invertir las probabilidades condicionales. En este caso se determinará mediante el uso de los campos mencionados anteriormente (Clean Text y speaker).

El vectorizador para aplicar el modelo Multinomial Naive Bayes tiene tres parámetros, los mismos utilizados previamente para los distintos mapas PCA. Para esta parte se fijaron estos parámetros eliminando stopwords en inglés, asignando True a la variable "use_idf" y por último, trabajando con unigramas y bigramas. Luego, se ajusta el vectorizador al conjunto de entrenamiento para transformarlo y por último se transforma el texto de los datos “test”.
.
Al utilizar este modelo para clasificar los discursos del conjunto de test se obtuvieron los siguientes resultados que permiten analizar y evaluar el funcionamiento del modelo

Tabla N°1: Resumen de métricas del modelo multinomial Naive Bayes : Accuracy, Precision, Recall y F1-score

Clase	**Precision**	**Recall**	**F1-score**	**Support**
Donald Trump	0.62	1.00	0.76	13
Joe Biden	1.00	0.77	0.87	13
Mike Pence	0.00	0.00	0.00	5
Accuracy			**0.74**	31
Macro avg	0.54	0.59	0.54	31
Weighted avg	0.68	0.74	0.69	31

Figura 5: Matriz de Confusión para el modelo Multinomial Naive Bayes



Al desglosar la tabla tenemos que la primera columna “precision” indica en este caso los discursos que el modelo determinó corresponden al orador respecto a los que estimó correctamente. El único candidato donde la precisión fue 100% fue Biden, lo que indica que todos los discursos que el modelo predijo como de Biden efectivamente eran de él, mientras que el “recall” determinó que de los discursos reales de Joe Biden (13) el modelo únicamente identificó 10 (77%), mientras que el campo “f1-score” indica que que el promedio entre estos dos campos es bueno (al acercarse a 1).

El valor de la precision correspondiente a Trump sugiere que el modelo determinó correctamente 13 discursos pero le atribuyó 8 discursos de otros oradores, generando falsos positivos (3 de Biden y 5 de Mike Pence) (ver fig.5). Por lo que la precision fue de 0,62 (13/(8+13)).

Mientras que el campo “recall” fue de 1 lo cual indica que los 13 discursos fueron atribuidos correctamente a Donald Trump esto sugiere que no hubo ningún discurso de Trump que el modelo lo identificara con otro candidato. Esto puede indicar que el modelo clasifica correctamente los discursos de Donald Trump pero tiene dificultades con los discursos del resto de los oradores. Esto se puede validar al observar que el modelo no predice ninguno de los cinco discursos de Mike Pence y el modelo los atribuyó a Trump lo cual corresponde a lo mencionado en la tarea anterior en donde ambos fueron conformaron la dupla del partido republicano en las

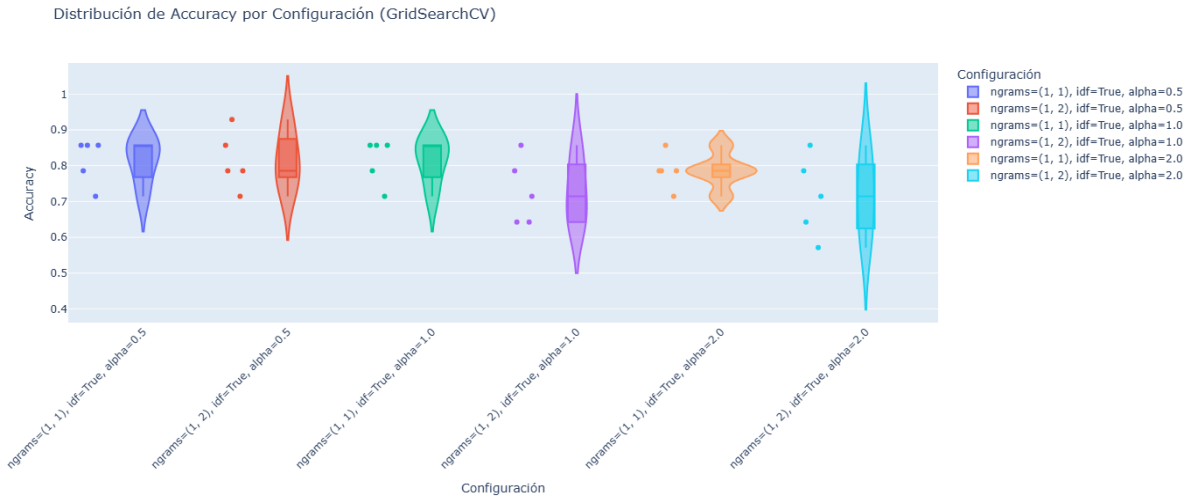
elecciones presidenciales de Estados Unidos del 2020 y sus discursos tenían muchas referencias a Trump (reflejado en las palabras mayor frecuencia).

Si únicamente se observa el valor “accuracy” la cual indica la exactitud del desempeño del modelo en este caso fue de 0.74 (estimó correctamente 23 de los 31 discursos), es un valor muy alto, lo cual puede no ser representativo ya que en uno de los oradores (Mike Pence) no clasificó ni estimó correctamente al orador ni sus discursos. Esto puede corresponder a que existe un desbalance de cantidad de discursos entre los oradores (la cantidad de discursos de Pence es notoriamente menor a los de Biden y Trump) y no le brinda al modelo una cantidad similar de entrenamiento entre los oradores, lo cual incrementa la probabilidad que asigne y clasifique erróneamente los discursos y su orador correspondiente.

2.

La validación cruzada o “cross-validation” es un método que permite comparar diferentes modelos de aprendizaje automático y generan una estimación de cómo funcionan en la práctica tomando en consideración nuevos datos. Esta técnica consiste en dividir el conjunto de datos de entrenamiento en subconjuntos, en este caso 5, y utilizar 4 de ellos para entrenar y el restante para evaluar el modelo, lo cual permite mitigar el impacto que puedan generar valores outliers dentro del set de datos y también minimiza el riesgo de sobreajuste. Otra característica positiva es que ,mejora el desempeño del modelo utilizando,determinando los rangos y parámetros que mejor se ajusten.

Figura 6: Gráfico comparativo de las distintas configuraciones



Para la confección del gráfico se utilizaron unigramas y bigramas, en todos los parámetros la variable use_idf fue igual a True ya que para el objetivo de este modelo se quiere utilizar el componente IDF para la transformación del texto lo cual

permitirá que exista una mejor discriminacion de los datos. El último parámetro es alpha, el cual es una constante que evita que la probabilidad de sea 0, si una palabra no aparece dentro de los datos de entrenamiento, lo cual permite una mejor generalización del set de datos, se utilizaron las constantes 0,5 (menor ajuste), 1 (equilibrado) y 2 (mayor ajuste) como los posibles ajustes que se deban aplicar.
En el gráfico se observa que los violines de color verde y azul son los que mejor se ajustan, ya que la mediana es alta y la dispersión de los datos es baja. Esto indica que el modelo funciona mejor con unigramas en lugar de bigramas y que el ajuste del parámetro alpha se debe realizar en 1 o 0,5. Se puede observa que, en al menos la mitad de las particiones de la validación cruzada, el modelo logró clasificar correctamente más del 80 % de los casos (la mediana de accuracy fue 0.85 en ambos parámetros).

Mientras que el violin de color celeste es el que tiene mayor dispersion de los datos y la mediana de accuracy en 0.71 (la más baja), lo cual determina que los parametros de ngrams=(1,2), idf=True y alpha=2 representan la peor configuración para el modelo, se puede inferir que el uso de bigramas no mejoran el rendimiento al incrementar igualmente la constante alpha a 2 puede generar mayores diferencias entre los datos, incrementando las diferencias entre las palabras poco representativas y no ser el indicado para entrenar el modelo.

3.
Tabla Nº2: Resumen de métricas para el modelo reentrenado: Accuracy, Precision, Recall, F1-score y Support

Configuración	**Accuracy**	**Clase**	**Precision**	**Recall**	**F1-score**	**Support**
ngrams=(1, 1), idf=True, alpha=0.5	0.814286	Donald Trump	0.68	1.00	0.81	13
		Joe Biden	1.00	0.92	0.96	13
		Mike Pence	0.00	0.00	0.00	5
		Macro avg	0.56	0.64	0.59	31
		Weighted avg	0.71	0.81	0.74	31

Al ejecutar nuevamente el modelo pero esta vez utilizando los mejores parámetros calculados anteriormente (ngrams=(1, 1), idf=True, alpha=0.5) se observan ligeras mejoras en los cálculos de “precision” y “recall” de los oradores Donald Trump y Joe

Biden. El modelo mejora, pero de forma muy leve, la precision de Trump de 0,62 a 0,68, lo que significa que el modelo sigue etiquetando como discursos de Trump a algunos que son de otros candidatos, al calcular los falsos positivos pasaron de 8 a 6, esto indica que el modelo sigue confundiendo discursos de otros oradores, en mayor proporción los discursos de Mike Pence (5) y uno de Joe Biden se los adjudica al orador Donald Trump creando falsos positivos y afectando el rendimiento general del modelo. El recall se mantuvo en 1, lo que indica que el modelo detectó el 100% de los discursos de las clases en el conjunto de prueba, es decir predijo correctamente los datos del test con los resultados del modelo.

Mientras que para los discursos de Joe Biden mejoró la estimación del “recall” predijo correctamente 12 discursos de 13, lo que representa el 92% del total de discursos, adjudicando 1 de sus discursos a Donald Trump como se mencionó anteriormente. De manera general mejoró la predicción en dos discursos respecto al modelo anterior que no utilizaban estos rangos y parámetros. La precisión se mantuvo en 1, lo que indica que asignó correctamente el orador los 13 discursos de Joe Biden.

La accuracy aumentó de 0.74 a 0.81, esto se debe a que el uso de estos parámetros mejoró la precisión en los discursos de Donald Trump y el recall en los de Joe Biden sin embargo en los discursos de Mike Pence el comportamiento fue el mismo, no hubo predicciones exitosas de sus discursos. En este último ejemplo se refleja una de las limitaciones del uso de “bag of words” o tf-idf, hay una importante diferencia de los casos de entrenamiento (discursos) de Mike Pence respecto al resto de los oradores.

Según Kowsari et al. (2019), una de las limitaciones del uso de “bag of words” es la codificación de cada palabra del vocabulario como un vector one-hot, lo que determina que el modelo crezca proporcionalmente al número de vectores, lo cual compromete la escalabilidad y el rendimiento del modelo. Con el set de datos que se están utilizando no tendríamos problemas pero si el dataset es más amplio implica mayores requerimientos de hardware, igualmente se pierde la relación semántica entre las palabras ya que permite asignar posiciones en el vector sin que represente alguna conexión entre las palabras. Mientras que el modelo TF-IDF, Kowsari et al. (2019), señalan como principales limitantes que no captura su posición en el texto ni el significado, esto puede repercutir negativamente ya que el modelo puede ponderar más bajo palabras que se repiten en los documentos pero no necesariamente es una “stopword” al igual que hay palabras que se escriben igual pero según el contexto el significado es diferente.

4.

En la librería de python scikit-learn se utilizó el modelo de regresión logística, el cual, según Kowsari et al. (2019), “predice probabilidades en lugar de clases”, esto quiere decir que brinda un umbral de certezas sobre un evento. Lo primero que hace

es vectorizar los datos, se añadieron rangos sobre la codificación (ngram_range=(1, 1), use_idf=True, stop_words='english'), manteniendo los rangos de la mejor configuración calculada anteriormente. Luego se seleccionó el modelo de regresión logística, se establece un número de máximas iteraciones del algoritmo de optimización en 1000 y el valor fijo establecido (random_state=42).

Posteriormente, se aplica la validación cruzada sobre el modelo de regresión logística, lo que representa que los datos se dividan en varios subconjuntos lo cual permite que el modelo se entrene con algunos y evalúe el resto, esto permite al modelo detectar la variación entre los subconjuntos. En este caso se seleccionó la validación cruzada de los datos de desarrollo que en total son 70 discursos, lo cual determina que el conjunto de datos de desarrollo se dividen en 5 partes iguales (14 discursos por subconjunto) y se realice entrenamientos y evaluaciones sobre dichas particiones, de los cuales 4 (56 datos en total) corresponden a datos de entrenamiento y uno (14) de prueba.Luego evalúa el “accuracy”, representando la cantidad de aciertos en las predicciones del modelo, los resultados fueron los siguientes:

Tabla N° 3: Resumen accuracy del modelo de regresión logística con validación cruzada

Métrica	**Valor**
Accuracy promedio (validación cruzada)	0.97
Accuracy por fold	[1.00, 0.92857143, 0.92857143, 1.00, 1.00]
Accuracy en test	1.00

La accuracy con los datos de desarrollo fue de 0.97, lo cual indica un acierto del 97% de predicción al evaluar el modelo mediante validación cruzada sobre los 70 discursos que conforman los datos de desarrollo (aproximadamente 13.58 de 14 discursos clasificados correctamente en las 5 particiones de subconjuntos).

Si se desglosa el accuracy según los 5 subconjuntos de datos, se observa que en 3 de los 5 folds la accuracy fue del 100%, mientras que en los dos restantes fue del 93%. Se puede inferir que existen folds con un mayor grado de complejidad que hace que el modelo no acierta el 100%. Sin embargo, esto es lógico, ya que siempre existe cierta heterogeneidad entre los datos, más aún cuando se trabaja con texto, como en este caso y un set de datos pequeño. Luego el modelo se entrena con los datos de desarrollo (70) y posteriormente realiza las predicciones con los datos “test” correspondiente a los speakers (31), los cuales son nuevos para el modelo, con estos datos la predicción del modelo fue del 100%, es decir que asignó correctamente los discursos a los datos de test en este caso los oradores.

Esto puede ser un indicativo de dos factores a considerar, el primero es que son muy pocos datos a evaluar y poco heterogéneos lo cual hace que para el modelo sea más fácil predecirlo, también puede indicar una fortaleza de los datos (limpieza y posterior vectorización). Con la finalidad de comparar con el modelo utilizado anteriormente se realizó una tabla con los siguientes resultados:

Tabla N° 4 Resumen de métricas del modelo de regresión logística : Accuracy, Precision, Recall, F1-score y Support

Clase	**Precision**	**Recall**	**F1-Score**	**Support**
Donald Trump	1.00	1.00	1.00	13
Joe Biden	1.00	1.00	1.00	13
Mike Pence	1.00	1.00	1.00	5
Accuracy			**1.00**	**31**
Macro avg	1.00	1.00	1.00	31
Weighted avg	1.00	1.00	1.00	31

Al comparar esta matriz con la del modelo anterior de Naive Bayes se observa que en ambos el accuracy es alto, sin embargo de los 5 datos de test correspondiente al orador Mike Pence el modelo de regresión logística acertó el 100% mientras que el modelo de Naive Bayes no acertó ninguno, indicando que este modelo presentan mayores dificultades al identificar estos discursos y se generen falsos positivos al atribuir los cinco discursos al orador Donald Trump, esto se debe a que dichos discursos tienen mucha referencia a Trump, similitud en las palabras que tienen mayor frecuencia y al tener menos datos que el resto de los oradores el modelo falla en estas predicciones ya que el modelo Naive Bayes se observa se ve más afectado por los desequilibrios en la cantidad de datos utilizados. De manera general el modelo de regresión logística según lo observado se adapta mejor a la heterogeneidad del número de datos debido.

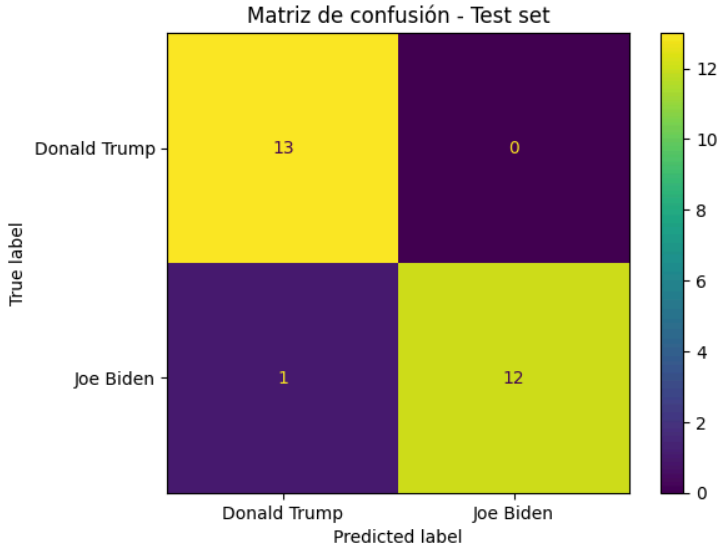
5.

Al evaluar nuevamente el modelo Naive Bayes filtrando por los dos candidatos con igual cantidad de datos de test (Joe Biden y Donald Trump),se obtuvo que el accuracy del modelo aumentó a casi el 100% (96%). En la siguiente tabla se muestran los resultados:

Tabla N°5 Resumen de métricas del modelo multinomial Naive Bayes filtrado por oradores Joe Biden y Donald Trump: Accuracy, Precision, Recall, F1-score y Support

Clase	**Precision**	**Recall**	**F1-Score**	**Support**
Donald Trump	0.93	1.00	0.96	13
Joe Biden	1.00	0.92	0.96	13
Accuracy			**0.96**	**26**
Macro avg	0.96	0.96	0.96	26
Weighted avg	0.96	0.96	0.96	26

Figura 7: Matriz de confusión del modelo multinomial Naive Bayes filtrado por oradores Joe Biden y Donald Trump



Al filtrar de los datos de test al orador Mike Pence los datos de Donald Trump obtienen el mayor beneficio ya que incrementa sustancialmente la precisión pasando del 68% (al ajustar los rangos y parámetros de alpha) al 93%, representando una mejora del 25% en la precisión del modelo, esto quiere decir que de las 14 predicciones de los discursos que el modelo atribuyó a Trump 13 eran correctos 1 era un falso positivo (correspondiente realmente a Joe Biden) (fig 7).

Este es uno de los factores que explican que las métricas de Biden se mantienen respecto al modelo anterior sin filtrar. Estos valores ratifican que el desbalance de datos no solo afectan a la predicción de la clase con menos datos (Mike Pence) sino a las clases que tienen alguna similitud en el contenido de los discursos, creando falsos positivos y atribuyendo discursos de Mike Pence erróneamente a Donald Trump, lo cual redujo sustancialmente la precisión en los discursos de Trump en el modelo anterior igualmente se puede inferir que los discursos de Biden no son similares a los de Pence y por ello no generó algún cambio en sus métricas, uno de sus discursos puede tener mayor relación con el orador Donald Trump eso podría explicar porque el recall se mantuvo en 0.92 (de los 13 discursos que se saben son de Biden el modelo atribuyó 1 a Trump).

Debido a que existe un desbalance en los datos de test, resulta conveniente entender cómo funcionan otras técnicas que permitan una mejor representación de las clases minoritarias en los modelos aplicados anteriormente, para ello se hará referencia al sobremuestreo y submuestreo. Ambas son técnicas de remuestreo que permiten equilibrar el conjunto de datos, la primera tiene como objetivo incrementar la cantidad de instancias de las clases desbalanceadas (minoritarias) o repetir algunas de estas, esto con el objetivo de reducir el desbalance entre los datos y que el modelo tenga suficientes datos que logren ser representativos de cada clase para evitar que únicamente sean favorecidas las clases mayoritarias (Mohammed et al. (2020)).

Esto se podría aplicar incrementando los discursos de Mike Pence, duplicándolos o añadiendo datos utilizando la técnica de Borderline-SMOTE, la cual consiste en aplicar k-nearest neighbors para generar datos sintéticos tomando como referencia la distancia entre el vector de características minoritarias y el vecino más cercano dentro de la misma clase Mujahid et al. (2024).
Mientras que el submuestreo tiene como objetivo reducir la cantidad de observaciones individuales (instancias) de la clase mayoritaria del conjunto de datos estudiado, esto permite equilibrar el set de datos utilizados. (Mohammed et al. (2020)).

6.

Una técnica alternativa para extraer features de texto es la denominada “transformers”, la cual no solo extrae features sino que mejora el rendimiento y la clasificación. Consiste en utilizar las relaciones entre todas las palabras de una secuencia, sin discriminar la distancia entre cada una de ellas. Esta técnica permite que al vectorizar cada palabra se le sume el vector correspondiente a su posición, este proceso no lo hace una vez sino que la realiza múltiples veces en paralelo lo que permite capturar diferentes tipos de relaciones entre las palabras. Mediante el uso del mecanismo de autoatención calcula tres vectores por palabra (Q,K,V), el primero tiene como función identificar que busca, el segundo como encontrarlo y el

último la información real, además permite identificar diferentes tipos de relaciones semánticas en paralelo(Vaswani et al., 2017).

Si se aplica está técnica a este set de datos se esperaría que logre identificar correctamente los discursos de la clase minoritaria (Mike Pence), ya que esta técnica puede calcular patrones y establecer dependencias de mayor complejidad que únicamente usar la frecuencia de palabras al establecer relaciones a larga distancia entre ellas. De esta manera se podrían reducir los falsos positivos en los discursos de Trump y probablemente afinar la precisión de los discursos de Biden.

Otra técnica relacionada, pero algo distinta es la de “word embedding”. Acá cada palabra se representa mediante un vector de dimensión n, densos pero de menor dimensión que en bag of words (acá cada palabra se representa con un vector de entre 50 y 500 de longitud). En esta técnica se intenta capturar el significado y el contexto de las palabras en sus valores. Para cada palabra, el “embedding” captura el significado de la palabra, por lo que palabras similares terminan con valores similares. Acá a diferencia de la técnica de bag of words, palabras con significado similar como universidad, facultad o instituto tienen representaciones similares o también palabras que pueden aparecer en contextos similares como comer, cocinar, cenar, almorzar, etc. también tienen representaciones cercanas.

Para nuestro problema si los oradores tienen un estilo muy marcado o usan términos particulares, quizás la técnica TF-IDF puede tener mejor precisión, ya que es sensible al vocabulario específico. En cambio, si los discursos tienen contenido semánticamente similar, pero expresado con diferente estilo o usando vocabulario distinto, los embeddings pueden capturar mejor esa variación y mejorar la predicción.

Bibliografía

IBM. (s. f.). Bag-of-Words. IBM Think.
Disponible en: https://www.ibm.com/es-es/think/topics/bag-of-words

IBM. (s. f.). Naive Bayes. IBM Think.
Disponible en: https://www.ibm.com/es-es/think/topics/naive-bayes

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2023). An Introduction to Statistical Learning with Applications in Python.

Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing (3rd ed.).

Jurafsky, D., & Martin, J. H. (2025). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models (3rd ed.). Online manuscript released January 12, 2025.
Disponible en: https://web.stanford.edu/~jurafsky/slp3

Kowsari, K., Jafari Meimandi, D., Heidarysafa, M., Mendu, S., Barnes, F., & Brown, D. (2019). Text Classification Algorithms: A Survey. arXiv.
Disponible en: https://arxiv.org/abs/1904.08067?

Linn, S. (s. f.). An introduction to word embeddings for text analysis.
Disponible en:
https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Mohammed, R., Rawashdeh, J., & Abdullah, M. A. (2020, April). Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. In 2020 11th International Conference on Information and Communication Systems (ICICS) (pp. 1–6). IEEE.

Scikit-learn. (s. f.). Cross-validation: evaluating estimator performance.
Disponible en: https://scikit-learn.org/stable/modules/cross_validation.html

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., &Kaiser, Ł. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems, 30.
Disponible en:
https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf