UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

INTRODUCCIÓN A LA CIENCIA DE DATOS

Tarea 2

Curso 2025

AUTOR:
Matías Izquierdo - C.I.: 5.288.879-7
Francisco Garchitorena - C.I.: 5.342.952-0

DOCENTES:
María Inés Fariello
Marcelo Fiori
Lorena Etcheverry
Guillermo Moncecchi
Graciana Castro

FECHA: 19 de junio de 2025

Tabla de contenidos

1. Dataset y representación numérica del texto **1**
- 1.1. Limpieza de texto . . . 1
- 1.2. Partición en train y test . . . 1
- 1.3. Bag of Words . . . 2
- 1.4. Mejoras en representación de texto . . . 3
  - 1.4.1. n-grama . . . 3
  - 1.4.2. TF-IDF . . . 3
- 1.5. PCA . . . 3

2. Entrenamiento y Evaluación de Modelos **7**
- 2.1. Modelo multinomial Naive-Bayes . . . 7
  - 2.1.1. Relación entre precisión, recall y la matriz de confusión . . . 9
  - 2.1.2. Análisis de los datos obtenidos . . . 9
- 2.2. Limitaciones del uso del *accuracy* . . . 10
- 2.3. Validación cruzada . . . 10
- 2.4. Regresión Logística . . . 12
- 2.5. Cambio de Candidato . . . 14
- 2.6. Word Embeddings . . . 15

1. Dataset y representación numérica del texto

1.1. Limpieza de texto

Para la realización de esta tarea se crea una versión reducida del dataset de discursos de los candidatos presidenciales de Estados Unidos para las elecciones de 2020. Se trabaja únicamente con los tres candidatos con mayor cantidad de discrusos: J. Biden, D. Trump y M. Pence. Previo a comenzar el tratamiento de datos se realiza una limpieza del texto tal y como se realizó en la primer tarea, se selecciona únicamente el texto del discurso dado por el orador, se eliminan signos de puntuación y números.

1.2. Partición en train y test

Con el objetivo de realizar un entrenamiento supervisado para predecir oradores en función del discurso, comenzamos separando los discursos en datos de entrenamiento (70 %) y de testeo (30 %), figura 1. Para esto utilizamos la función *train_test_split* de la librería scikit-learn, la cual realiza una separación de los datos. Incluímos el parámetro *random_state* para que la separación sea reproducible, y el parámetro *stratify* para que el muestreo sea estratificado (que mantenga la misma proporción de clases del conjunto original), como se puede observar en la tabla 1.

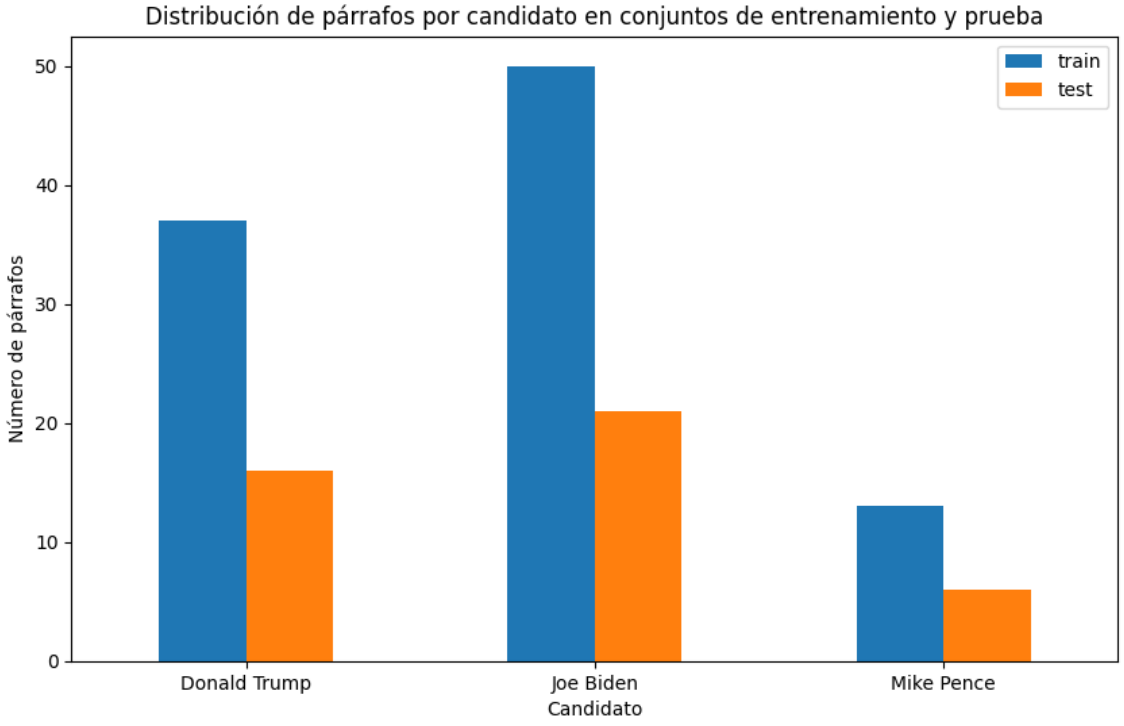

Figura 1: Cantidad de discursos por candidato en los conjuntos train y test

Candidato	**Train (%)**	**Test (%)**
Donald Trump	37.0	37.21
Joe Biden	50.0	48.84
Mike Pence	13.0	13.95

Tabla 1: Distribución porcentual de discursos por candidato en los conjuntos train y test

1.3. Bag of Words

Luego se transforma el texto a la representación numérica (en vectores) "bag of words" (BoW) utilizando la función *CountVectorizer* de scikit-learn. Esta técnica divide el texto en palabras para crear un vocabulario con todas las palabras (de forma única) existentes en todos los discursos. Luego se crea una matriz donde las filas son los discursos y las columnas las palabras, en los datos se coloca el número de veces que aparece la palabra en el discurso respectivo. Esta función permite, por ejemplo, configurar la BoW quitando las palabras vacías, separando por n-gramas o ponderar las palabras según su relevancia utilizando la técnica TF-IDF (*Term Frequency - Inverse Document Frequency*). Para esta etapa se utiliza la BoW por defecto, sin aplicar estas mejoras.

Para ejemplificar el concepto BoW supongamos una versión reducida del dataset, donde tenemos solo tres discursos:

1. "el clima está lindo hoy"
2. "hoy el día está feo"
3. "me gusta el clima feo"

El vocabulario sería: ['clima', 'día', 'está', 'feo', 'gusta', 'hoy', 'lindo', 'me'].

Y la representación en forma de BoW queda de la forma:

Discurso	**clima**	**día**	**está**	**feo**	**gusta**	**hoy**	**lindo**	**me**
"el clima está lindo hoy"	1	0	1	0	0	1	1	0
"hoy el día está feo"	0	1	1	1	0	1	0	0
"me gusta el clima feo"	1	0	0	1	1	0	0	1

Tabla 2: Ejemplo de representación Bag of Words

$$M = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Esta matriz es del tipo sparse ya que gran parte de los valores son cero (en discursos más largos esto se observa mejor). En un discurso, se usan pocas palabras del total del vocabulario por lo que la mayoría de las columnas (palabras) tienen valor 0 en cada fila (discurso). Almacenar todos esos ceros es ineficiente, por eso usamos una matriz sparse, que solo guarda los valores distintos de cero y sus posiciones.

1.4. Mejoras en representación de texto

Como forma de preprocesamiento del texto con el objetivo de obtener un mejor modelo se prueban tres técnicas: eliminación de *stopwords*, agrupación por Bigramas, ponderación por relevancia (TF-IDF).

1.4.1. n-grama

Un n-grama es una secuencia contigua de n elementos presentes en un texto. Los mismos son ampliamente utilizados en el procesamiento del lenguaje natural para representar texto en forma numérica. Algunos tipos de n-gramas comúnmente utilizados son los Unigramas, Bigramas y Trigramas. Los unigramas agrupan el texto de a un elemento (en este caso una palabra) mientras que los Bigramas y Trigramas de a dos y tres respectivamente. Estos últimos permiten obtener mayor información del contexto de la palabra pero, al aumentar la dimensión, también aumenta el requerimiento de recursos computacionales.

1.4.2. TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) es una técnica utilizada para representar numéricamente la importancia de un elemento dentro de un documento perteneciente a una colección. Esto lo hace mediante la ponderación por un valor que dice qué tan común o raro es encontrar el elemento en la colección, siendo útil para reducir el impacto de palabras que ocurren con mucha frecuencia.

Por un lado se tiene **TF** que mide cuántas veces aparece una palabra en un discurso. Por otro **IDF** mide cuán rara o informativa es una palabra en todo el conjunto. Finalmente TF-IDF es el resultado de multiplicar ambos factores.

1.5. PCA

Para finalizar con el preprocesamiento de los datos se realiza una PCA. De este modo reducimos la dimensionalidad del conjunto y podemos realizar visualizaciones en 2D. Entre otras cosas, esto nos permite ver cómo reaccionan los datos al aplicar las mejoras en la representación de texto descritas en la sección previa.

Cuanto mayor sea la cantidad de componentes principales de la PCA (dimensiones), mayor es la información (varianza) que retengo, pero pierdo capacidad de visualización.

El primer caso considerado —sin filtrado de stopwords, utilizando solo unigramas y sin aplicar el factor IDF— se observa en la Figura 2. Se observa que el valor de la componente principal 1 es casi constante, muy cercano al cero, con todos los datos agrupados en este valor. Esto muestra que los distintos discursos poseen baja diferencia entre candidatos. Por su parte, la componente principal 2 sí parece mostrar mayores diferencias, con Trump considerablemente separado del resto. Biden y Pence se agrupan mucho entre sí.

Al filtrar por stopwords y agrupar las palabras en bigramas se obtiene la Figura 3.

Aquí se separan más los datos para ambas componentes principales, se empieza a ver más diferenciación entre discursos.

Finalmente, al ponderar por relevancia (IDF = true), se obtiene mejor diferenciación entre candidatos. Esto porque se penalizan las palabras que aparecen mucho en todos los discursos y destaca las que son más distintivas. Esto se ve en la Figura 4.

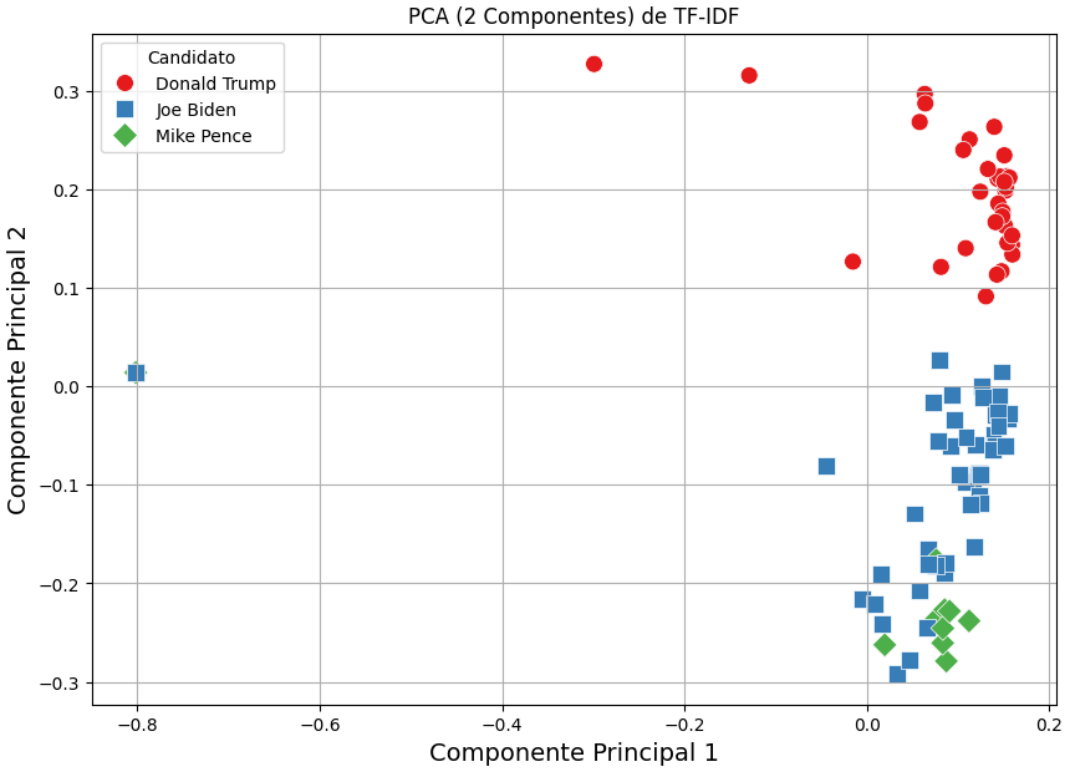

Figura 2: PCA pre filtrado de stopwords, idf = false y sin bigramas

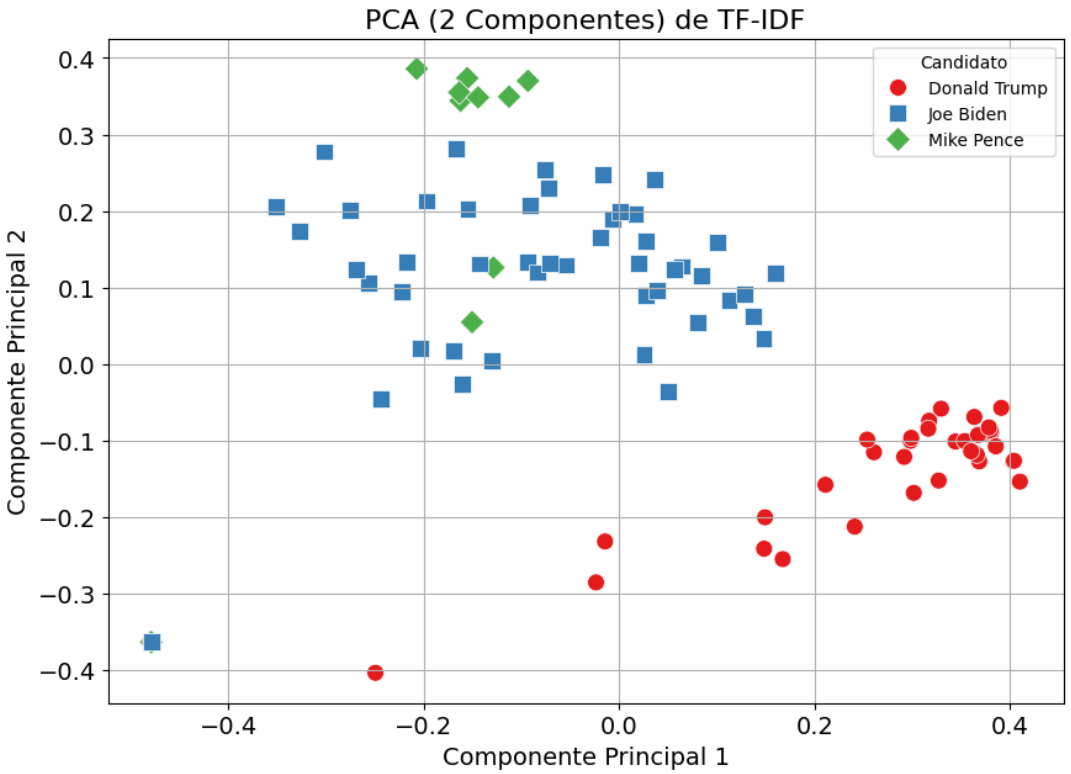

Figura 3: PCA post filtrado de stopwords, idf = false y con bigramas.

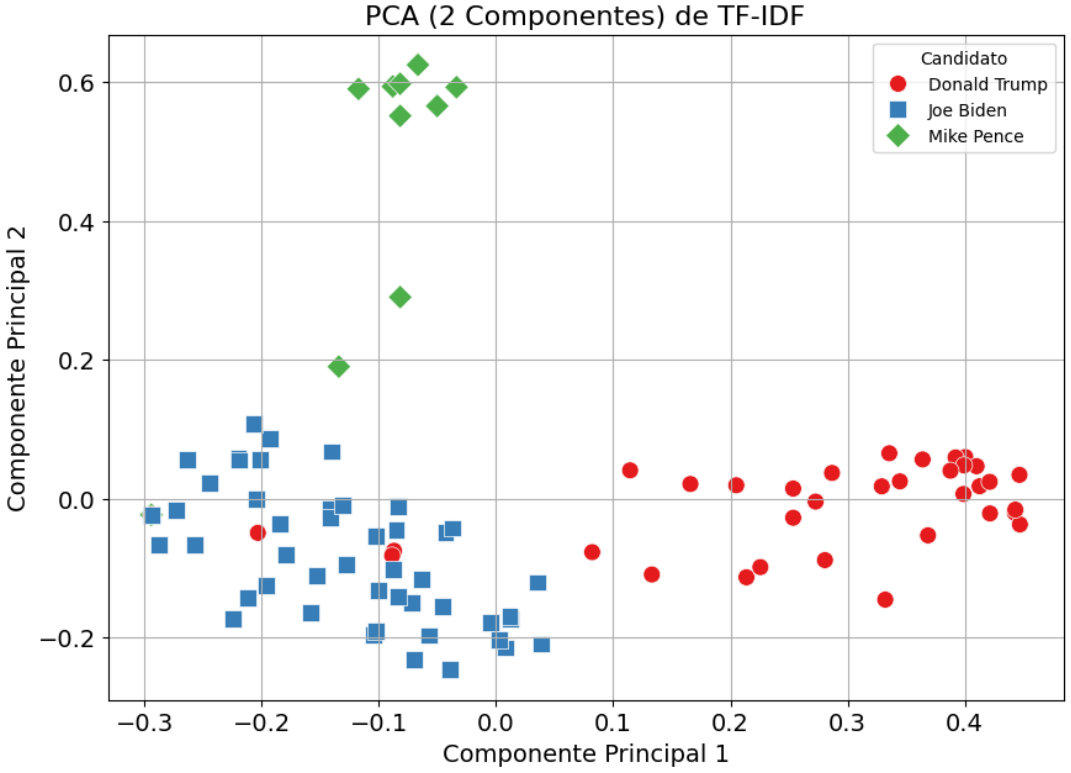

Figura 4: PCA post filtrado de stopwords, idf = true y con bigramas.

Para agregar información a la PCA se puede aumentar el número de componentes principales, a costa de perder capacidad de visualización. Sin embargo, cada componente agregada aumenta la información en menor medida que la anterior. En la Figura 5 y en la Figura 6 se visualizan la varianza explicada por componente y la varianza explicada acumulada, para el ejemplo final (el efecto es el mismo para todos los casos) donde se puede apreciar el efecto comentado.

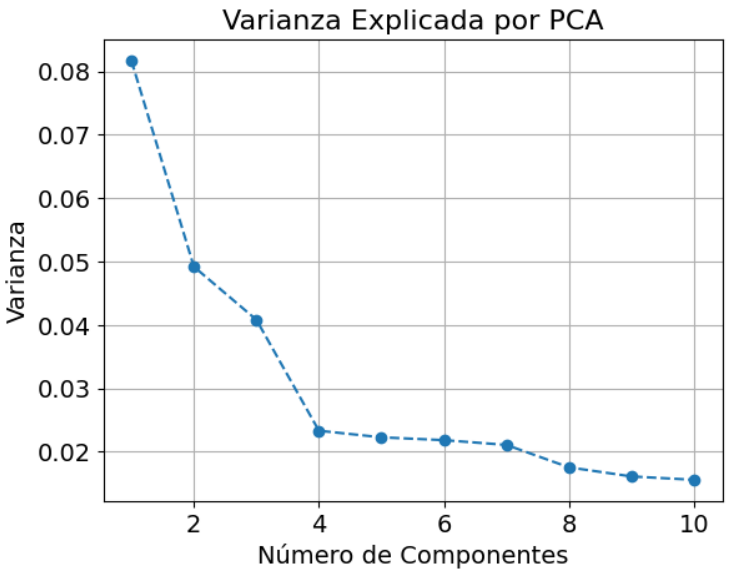

Figura 5: Varianza por componente post filtrado de stopwords, idf = true y con bigramas

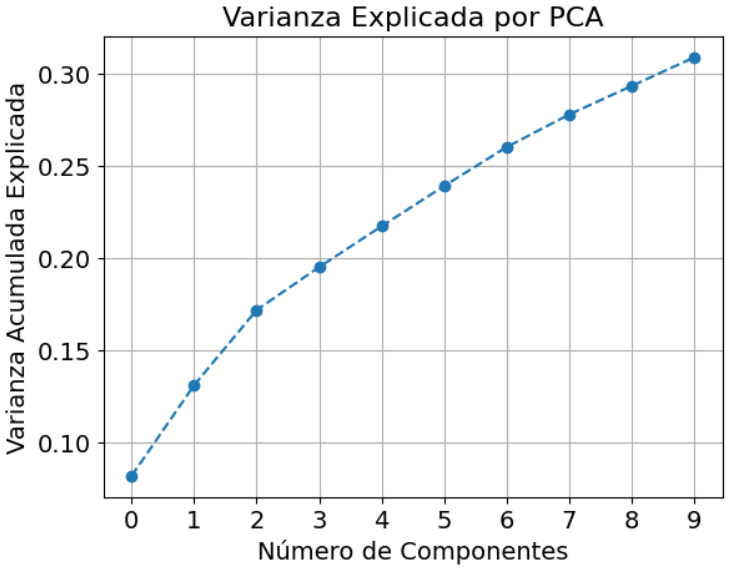

Figura 6: Varianza acumulada post filtrado de stopwords, idf = true y con bigramas

2. Entrenamiento y Evaluación de Modelos

2.1. Modelo multinomial Naive-Bayes

El modelo *Multinomial Naive-Bayes* es un modelo de clasificación probabilístico ampliamente utilizado en tareas de clasificación de texto. Es una herramienta de aprendizaje automático supervisado, que posee un buen rendimiento en su uso con conjuntos de datos multidimensionales [1]. Se basa en el Teorema de Bayes, utilizando probabilidades condicionales y en la asunción ingenua (naive) de independencia condicional entre las características (en este caso, las palabras de los discursos). El modelo asume que todas las variables (en este caso palabras) son independientes entre sí, dado que conocemos la clase (en este caso, los candidatos) [1].

Para la presente aplicación, se entrena el modelo con el conjunto de entrenamiento filtrado por "stopwords", agrupando tanto por Unigramas como Bigramas y utilizando el factor IDF.

Para analizar su funcionamiento, se utiliza el parámetro "accuracy", definido por la Ecuación 1.

$$accuracy = \frac{\text{Predicciones acertadas}}{\text{Predicciones totales}} \tag{1}$$

El valor obtenido en este caso, es de 0,77.

Con el fin de observar con mayor claridad el funcionamiento del modelo, se analiza la Matriz de Confusión, visible en la Figura 7. Esta matriz ilustra todas las combinaciones de resultados obtenidos, siendo estos:

- Verdaderos positivos (VP): El modelo predijo correctamente que el discurso es del candidato.
- Falsos positivos (FP): El modelo predijo que el discurso es del candidato, pero en realidad no lo es.
- Verdaderos negativos (VN): El modelo predijo correctamente que el discurso no es del candidato.
- Falsos negativos (FN): El modelo predijo que el discurso no es del candidato, pero en realidad sí lo es.

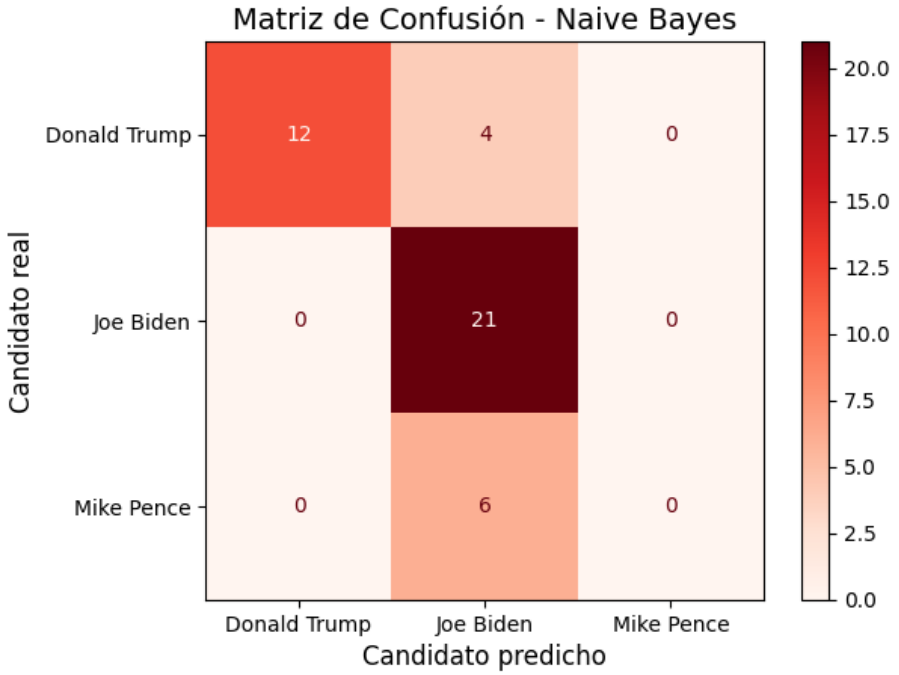

Figura 7: Matriz de confusión obtenida a partir del modelo Multinomial Naive-Bayes.

Con el fin de considerar el desbalance en la cantidad de discursos en las clases, se ilustra la matriz de confusión relativa, en la Figura 8.

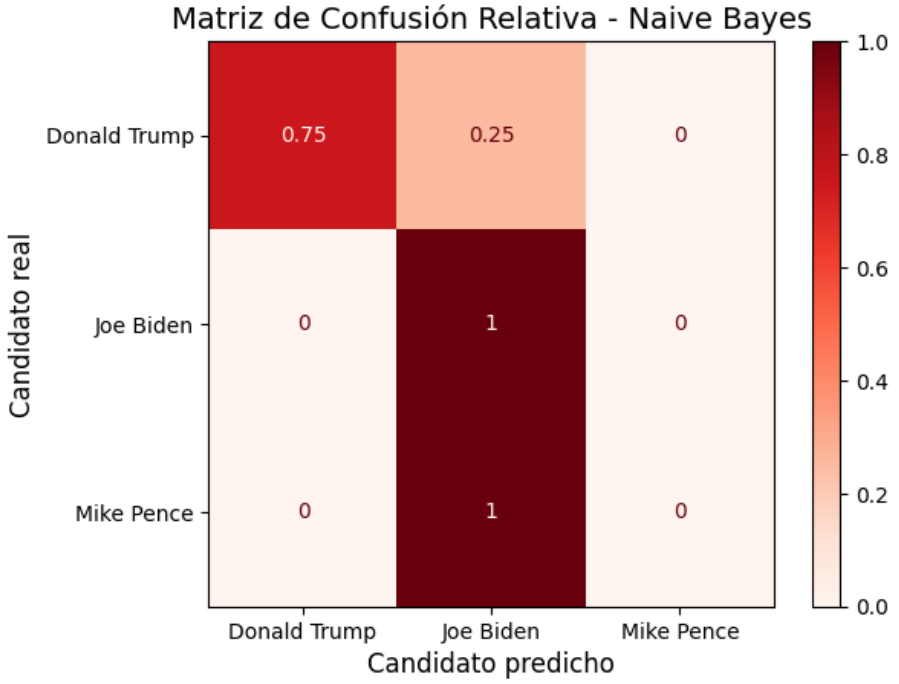

Figura 8: Matriz de confusión relativa obtenida a partir del modelo Multinomial Naive-Bayes.

Se observa que el modelo predice de forma excelente a Joe Biden, y falla en el total de los casos en el caso de Mike Pence, prediciendo que este candidato es Joe Biden. Si bien esto puede deberse a varias causas, la principal es que Mike Pence es el candidato con menos cantidad de discursos, por lo que tanto los datos de entrenamiento como los de test, son insuficientes.

2.1.1. Relación entre precisión, recall y la matriz de confusión

La matriz de confusión es una tabla que permite visualizar el desempeño de un modelo de clasificación. En el caso de un problema multiclase como el presente (con los candidatos Donald Trump, Joe Biden y Mike Pence), la matriz de confusión tiene una fila y una columna por cada clase (candidato).

- Las filas representan las clases reales (verdaderas).
- Las columnas representan las clases predichas por el modelo.
- Cada entrada (i, j) indica la cantidad de ejemplos de la clase real i que fueron predichos como clase j.

A partir de esta matriz, se derivan tres métricas clave:

Precisión

Indica la proporción de ejemplos predichos como pertenecientes a una clase que realmente lo son.

$$\text{Precisión} = \frac{VP}{VP + FP}$$

Se calcula por columna en la matriz de confusión.

Recall

Indica la proporción de discursos reales de una clase que fueron correctamente identificados.

$$\text{Recall} = \frac{VP}{VP + FN}$$

Se calcula por fila en la matriz de confusión.

2.1.2. Análisis de los datos obtenidos

A continuación se interpretan los valores obtenidos para cada candidato:

- **Donald Trump:** Precisión = 1.00, Recall = 0.75
  El modelo clasificó correctamente todos los discursos que predijo como de Trump (alta precisión), pero no detectó el 25 % de sus discursos reales (recall incompleto).

- **Joe Biden:** Precisión = 0.68, Recall = 1.00
  Todos los discursos reales de Biden fueron correctamente detectados (recall perfecto), pero el modelo clasificó incorrectamente otros discursos como suyos (precisión menor).
- **Mike Pence:** Precisión = 0.00, Recall = 0.00
  El modelo no fue capaz de identificar correctamente ningún discurso de Pence, ni como clase real ni como predicha.

Esto se refleja directamente en la matriz de confusión: Mike Pence no tiene valores en la diagonal principal, mientras que Joe Biden y Donald Trump sí presentan valores correctos (aunque con algunos errores en otras celdas).

Los valores de precisión y recall se observan en la Tabla 3.

Candidato	**Precisión**	**Recall**
Donald Trump	1.00	0.75
Joe Biden	0.68	1.00
Mike Pence	0.00	0.00

Accuracy general del modelo: 0.77

Tabla 3: Resultados precisión y recall del modelo Multinomial Naive-Bayes para los datos con representaciones TF-IDF.

2.2. Limitaciones del uso del *accuracy*

Como fue mencionado previamente, el *accuracy* es un indicador muy utilizado en este tipo de modelos, por su carácter intuitivo. Sin embargo, basarse únicamente en su valor puede llevar a conclusiones erróneas del rendimiento de un modelo. Esto se debe a que este indicador no tiene en cuenta el comportamiento en las distintas clases. Entonces, si se tienen una gran cantidad de datos de cierta clase, el modelo puede obtener un *accuracy* alto simplemente aprendiendo a predecir siempre esa clase, ignorando las clases con menor cantidad de datos. En este caso, por ejemplo, la clase "Joe Biden" contiene una cantidad significativamente mayor de ejemplos en comparación con "Mike Pence". Esta desproporción contribuye a un valor elevado de *accuracy*; sin embargo, un análisis más detallado de las predicciones revela que el modelo falla sistemáticamente al clasificar los discursos de Mike Pence, sin identificar correctamente ninguno de ellos.

2.3. Validación cruzada

La validación cruzada es una técnica utilizada para evaluar modelos de aprendizaje estadístico y seleccionar los mejores hiperparámetros (por ejemplo: stopwords, idf y n-gramas). Esta técnica evita que el modelo se sobreajuste al conjunto de entrenamiento. El funcionamiento se da de la siguiente manera:

- Se divide el conjunto de entrenamiento en k partes iguales (folds).

- Se entrena el modelo k veces, cada vez usando: (k-1) folds para entrenar, 1 fold para validar (que cambia en cada iteración).
- Se obtienen k valores de *accuracy*.
- Se calcula el promedio y desviación estándar, lo cual da una estimación robusta del desempeño del modelo.

Para esta sección se utilizó el texto filtrado por stopwords, ordenando en bigramas y con el factor IDF = True.

En este caso se utilizará "GridSearchCV", el cual busca automáticamente los mejores hiperparámetros usando validación cruzada. El hiperparámetro seleccionado para optimizar es:

- `clf__alpha`: Parámetro de suavizado de Laplace del clasificador Naive-Bayes. Valores más bajos implican menos suavizado.

La Tabla 4 ilustra los valores del hiperparámetro evaluado.

Hiperparámetro	**Valores evaluados**
`clf__alpha`	`0.001`, `0.01` `0.1`, `1`, `10`,

Tabla 4: Valores evaluados, mediante GridSearchCV, del hiperparámetro α.

Como indicador de performance se utilizó la *accuracy*, llegando al valor óptimo del hiperparámetro α reportado en la Tabla 5.

Hiperparámetro	**Valor óptimo**
`clf__alpha`	0.001
Accuracy promedio	**0.9300**

Tabla 5: Mejor valor del hiperparámetro α, encontrados mediante validación cruzada (GridSearchCV).

En la Figura 9 se ilustra la matriz de confusión y en la Tabla 6 las métricas, utilizando el valor óptimo del hiperparámetro *alpha*. Un factor importante a tener en cuenta, es el hecho de que el factor *alpha* (factor de suavizado) es el que realmente varía los resultados en la predicción del modelo. Si se deja su valor por defecto ($alpha = 1$), se obtienen los mismos resultados que en la Figura 8.

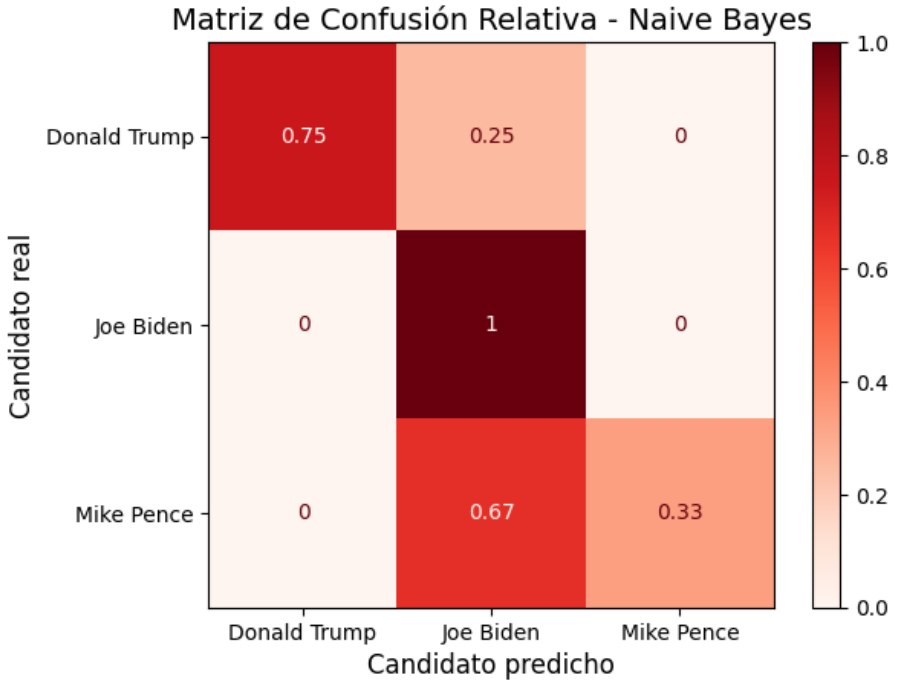

Figura 9: Matriz de confusión relativa con la mejor combinación de hiperparámetros, obtenida a partir del modelo Multinomial Naive-Bayes.

Candidato	**Precisión**	**Recall**
Donald Trump	1.00	0.75
Joe Biden	0.72	1.00
Mike Pence	1.00	0.33

Accuracy general del modelo: 0.81

Tabla 6: Resultados precisión y recall del modelo Multinomial Naive-Bayes utilizando la mejor combinación de hiperparámetros.

Por otra parte, se analiza el gráfico de violín visible en la figura 10.

En la Figura 10, se observa que los valores bajos de α (0.001 y 0.01) alcanzan mayores niveles de *accuracy*, con medianas cercanas a 0.9–1.0. También se observa mayor variabilidad (mayor dispersión), lo cual podría significar mayor sensibilidad a los datos. A medida que α aumenta, el rendimiento disminuye significativamente, siendo $\alpha = 10$ el peor caso. Esto indica que una regularización suave mejora el desempeño, mientras que una excesiva suavización (alta α) lo perjudica.

2.4. Regresión Logística

La regresión logística es un modelo lineal utilizado para tareas de clasificación. A diferencia de la regresión lineal, que predice valores continuos, la regresión logística estima la probabilidad de que una observación pertenezca a una determinada clase,

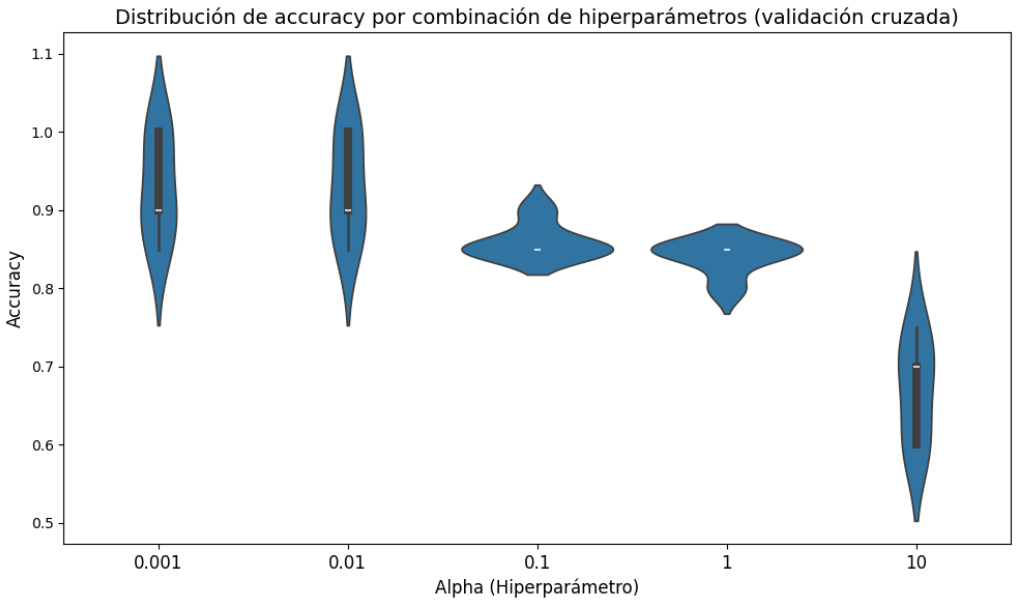

Figura 10: Distribución de *accuracy* para distintos valores del hiperparámetro α (validación cruzada).

aplicando la función sigmoide a una combinación lineal de las variables de entrada. Esta probabilidad se expresa como:

$$P(\text{clase} \mid \mathbf{x}) = \frac{1}{1 + e^{-\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)}}$$

El modelo puede extenderse a múltiples clases utilizando estrategias como "one-vs-rest" (una clase contra las demás), que es el enfoque por defecto en `scikit-learn`.

En este trabajo, se utiliza el modelo de regresión logística sobre las mismas *features* de texto que el clasificador Naive-Bayes. En la Figura 11 presentamos la matriz de confusión y en la tabla Tabla 7 el desempeño en términos de *accuracy*, precision, recall.

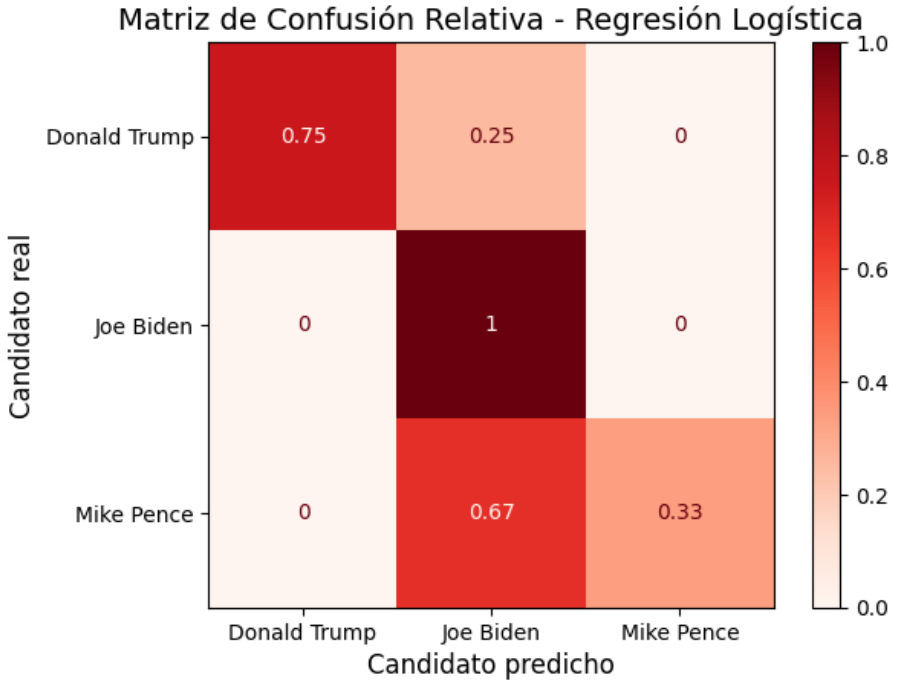

Figura 11: Matriz de confusión relativa obtenida a partir del modelo de Regresión Logística.

Candidato	Precisión	Recall
Donald Trump	1.00	0.75
Joe Biden	0.72	1.00
Mike Pence	1.00	0.33

Accuracy general del modelo: 0.81

Tabla 7: Resultados precisión y recall del modelo de Regresión Logística.

Observamos que este modelo tiene el mismo desempeño que el modelo de Naive-Bayes para el caso con mejores hiperparámetros (C = 30).

2.5. Cambio de Candidato

Para esta sección se modifica el dataset de trabajo cambiando el candidato Donald Trump por Bernie Sanders. La principal diferencia entre estos dos candidatos radica en la cantidad de discursos dictados y de palabras utilizadas. Por lo tanto, se está reemplazando un individuo con gran cantidad de datos por uno con una cantidad considerablemente menor.

Las consecuencias de este cambio se pueden ver tanto en la matriz PCA como en los resultados del modelo de predicción. En la Figura 12 se presenta la PCA con dos componentes para el nuevo trío de candidatos, ya aplicado el filtrado por stopwords, idf = true y bigramas.

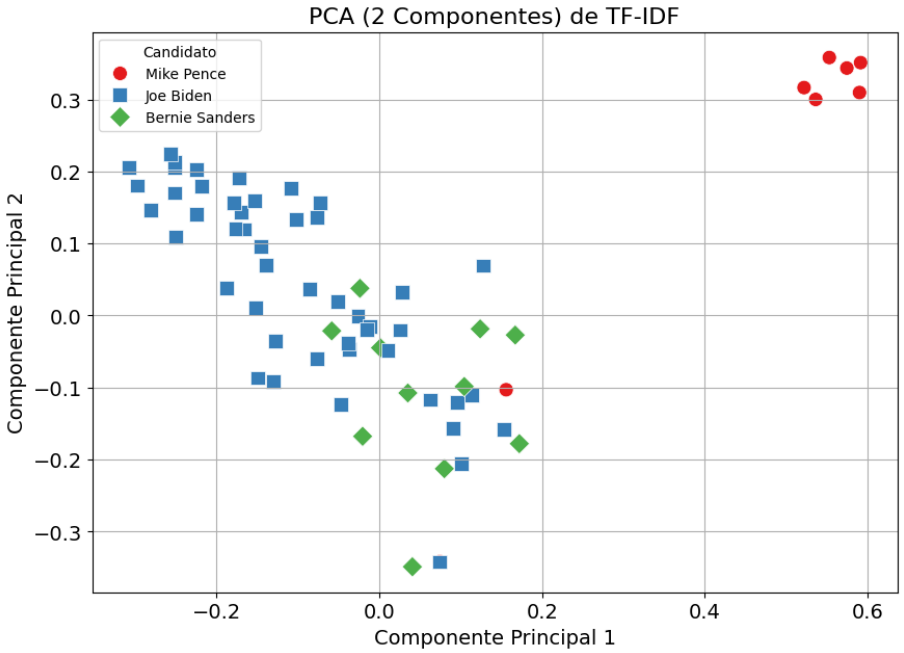

Figura 12: PCA con el cambio de candidato, con filtrado de stopwords, idf = true y bigramas

2.6. Word Embeddings

Una técnica alternativa a las representaciones tradicionales como bag-of-words o tf-idf es el uso de word embeddings, como por ejemplo *Word2Vec*, *GloVe* o *FastText*. Estas técnicas buscan representar cada palabra como un vector denso de baja dimensión, donde las posiciones del vector no corresponden a palabras específicas (como en bag-of-words), sino a propiedades semánticas aprendidas a partir del contexto en grandes conjuntos de texto.

A diferencia de las representaciones dispersas y de alta dimensión (sparse), los embeddings capturan similitudes semánticas entre palabras. Por ejemplo, en un embedding entrenado correctamente, los vectores de "presidente" y "ministro" estarán más cerca entre sí que los de "presidente" y "banana", lo cual permite al modelo aprender relaciones más profundas entre los textos.

Se espera que este tipo de representación mejore el rendimiento en tareas donde el significado de las palabras y su relación contextual es importante, como en análisis de sentimiento o clasificación de temas. Sin embargo, también requiere mayor complejidad computacional, más datos y un procesamiento más elaborado, por lo que puede no ser ideal para tareas introductorias o con datasets muy pequeños.

Referencias

[1] IBM Developer. Classifying data using the multinomial naive bayes algorithm, 2024. Accedido: 2025-06-09.