Tarea
Introduccion a la Ciencia de Datos

Del Po, Diego
O'Flaherty, Julian

Junio 2025

1. Introduccion

El presente informe corresponde al desarrollo de la Tarea 2 del curso Introducción a la Ciencia de Datos. Este trabajo da continuidad a la Tarea 1, reutilizando datos y elementos de código previamente elaborados, y se centra en la aplicación práctica de conceptos fundamentales del aprendizaje automático y del procesamiento de texto. El objetivo principal es profundizar en el entendimiento del proceso completo de análisis de datos textuales, desde su representación numérica hasta el entrenamiento y evaluación de modelos de clasificación.

En esta tarea, se aborda el problema de clasificación de discursos de distintos candidatos (en específico de los tres candidatos con más discursos en el set de datos: Donald Trump, Joe Biden y Mike Pence), utilizando técnicas como *Bag of Words, Term Frequency - Inverse Document Frequency* (TF-IDF) y modelos supervisados dentro del entorno de *scikit-learn*. Asimismo, se exploran métodos para visualizar y comprender los datos mediante reducción de dimensionalidad a través de la técnica *Principal Component Analysis* (PCA) y se analizan los desafíos asociados al desbalance de clases, la selección de hiperparámetros y la interpretación de métricas como *accuracy, precision, recall y F1-Score.*

2. Pre-procesado y exploracion

Al igual que en la Tarea 1, el procesamiento del set de datos se basó en la columna *text*. A partir de dicha columna y con la ayuda de expresiones regulares, se extrajeron los candidatos y lo que dijo cada uno de ellos; si bien el set de datos original cuenta con una columna para identificar al orador (*speaker*), en muchas

ocasiones en los discursos interviene más de un orador, por lo que elegimos esta alternativa para quedarnos con lo que dijo únicamente el candidato. Además de lo anterior, una vez separados oradores y discursos, estos últimos fueron normalizados al igual que en la Tarea 1. Una vez obtenida las intervenciones de cada orador, se reconstruyo el discurso completo aislando unicamente lo que dijo el candidato.

Una vez realizadas las transformaciones necesarias, se procedió a separar el set de datos en dos, dejando un 70 % de los datos para entrenamiento y un 30 % de los datos para prueba. La figura 1 muestra cómo la distribución de candidatos queda prácticamente igual en ambos sets de datos, lo cual refleja una correcta separación de forma aleatoria.

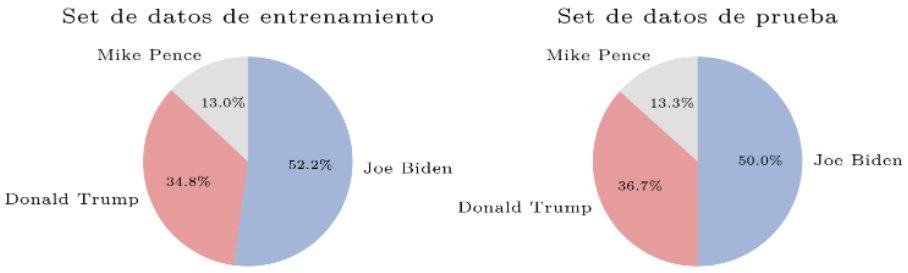

Figura 1: Distribución de candidatos en los set de datos de entrenamiento y prueba

2.1. Vectorizacion

El texto, en su forma natural (secuencias de caracteres y palabras), no es directamente procesable por la mayoría de los algoritmos, ya que estos requieren datos estructurados en forma de números. Por lo tanto, el paso inicial en cualquier tarea de modelado con texto es convertir el texto en una representación numérica que preserve su semántica o significado tanto como sea posible.

Una de las posibles técnicas de representación numérica de texto es el método de *Bag of Words*, el cual consiste en representar cada documento como un vector de longitud igual al tamaño del vocabulario donde cada posición del vector indica la frecuencia de una palabra en el documento, sin tener en cuenta el orden de las palabras. Por ejemplo, si el vocabulario es [*"president", "donald", "trump"*] y el documento es [*"the", "president", "trump"*], el vector sería [0, 1, 1]. El procesamiento de los datos a través de este método genera una matriz *sparse* (la mayoría de sus elementos son ceros), ya que el vocabulario suele ser muy grande en términos generales en comparación con el documento. En consecuencia, en

cada documento, la mayoría de las palabras (del vocabulario) no aparecen.

Un concepto que resulta importante en este contexto es el concepto de N-grama, el cual es una secuencia contigua de n elementos (normalmente palabras o caracteres) de un texto. Es decir, podemos aplicar la estrategia de *Bag of Words* a nivel de palabras, de n palabras o de n caracteres.

Una mejora de la estrategia *Bag of Words* es *TF-IDF* (*Term Frequency - Inverse Document Frecuency*), en el cual cada término de un documento recibe un peso que mide su importancia relativa en ese documento respecto al corpus completo. La idea principal de este método es restarle importancia a las palabras comunes (por ejemplo: *"the", "she", ït"*) que aparecen mucho en el texto pero que no aportan para identificar documentos. TF-IDF penaliza esas palabras comunes y resalta términos que son raros y distintivos de un documento.

$$tfid(t\ d\ D) = tf(t\ d)\ \ idf(t\ D) \tag{1}$$

$$tf(t\ d) = \frac{f_{t\ d}}{{}_{t\ \ d} f_{t\ d}} \tag{2}$$

$$idf(t\ D) = \log \frac{N}{|\{d : d \in D\ \ t \in d|} \tag{3}$$

En la ecuación (3) se presenta un ejemplo de cómo calcular el $t\ \ df$. En la ecuación (2) se define una forma de calcular la frecuencia de un término, donde $f_{t\ d}$ es la cantidad de veces que aparece un término en un documento y la suma en el divisor es el total de términos del documento. La ecuación (3) presenta un cálculo posible para el IDF, donde D es el conjunto de documentos, $N = \#D$ es el total de documentos, y el divisor es el número de documentos en donde t aparece. Notar que si una palabra aparece en todos los documentos, entonces $idf = \log 1 = 0$, es decir, esa palabra no nos aporta para distinguir un documento del otro.

Es esta última técnica la que se usa para vectorizar los textos en este trabajo.

2.2. Analisis PCA

Una vez vectorizados los textos, es de interés saber si es posible discernir entre candidatos a partir de estos, lo cual verificamos a través del análisis de componentes principales. En esencia, la técnica de PCA (*Principal Component Analysis*) busca reducir el número de variables (o características) en un conjunto de datos manteniendo la mayor cantidad de información posible. En la

figura 2 podemos ver cómo quedan distribuidos los candidatos según los dos principales componentes de los vectores TF-IDF. Es notorio cómo con tan solo dos componentes ya logramos una segregación de los candidatos bastante clara.

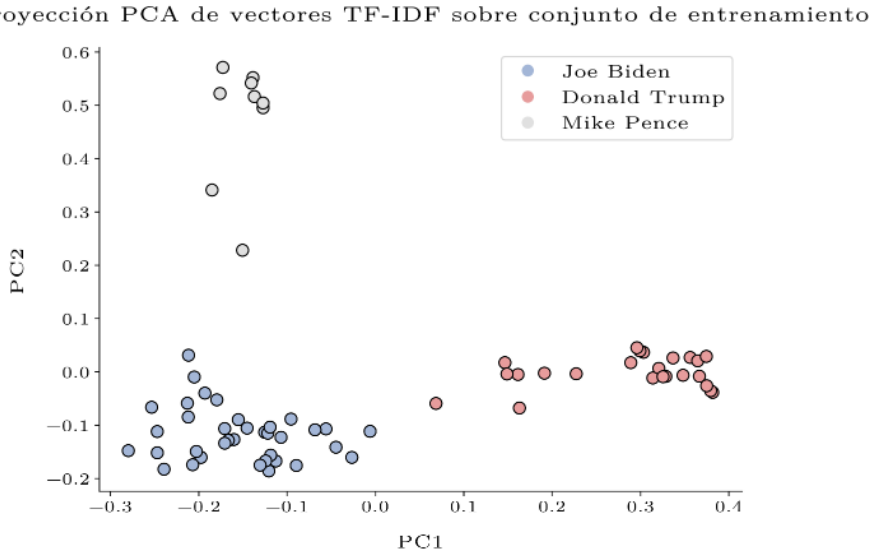

Figura 2: Análisis de los dos componentes principales en el set de datos de entrenamiento utilizando los parámetros *stop_words* para idioma inglés, *use_idf=True* y *ngram_range=(1,2)*

Por otra parte, resulta interesante notar en la figura 3 cómo, a pesar de que con solo dos componentes principales parecen segregarse de forma bastante clara los candidatos, estos solamente explican algo más del 10 % de la varianza explicada acumulada. Al parecer, aunque los primeros dos componentes principales solo expliquen una pequeña porción de la varianza total, es posible que capten la varianza más importante para separar a los candidatos.

Esto podría significar que, incluso si hay otras dimensiones en los vectores TF-IDF que explican más varianza total, esa varianza adicional podría estar relacionada con cuestiones que no importan tanto para la distinción entre oradores, como por ejemplo:

- Variaciones en los estilos discursivos de un mismo candidato. Es decir, diferencias de palabras o frases usadas por la misma persona en diferentes discursos.
- Variaciones debido a que los discursos tratan temas diferentes, independientemente del orador.

- Ruido o información irrelevante. Variabilidad que no contribuye a distinguir a los oradores.

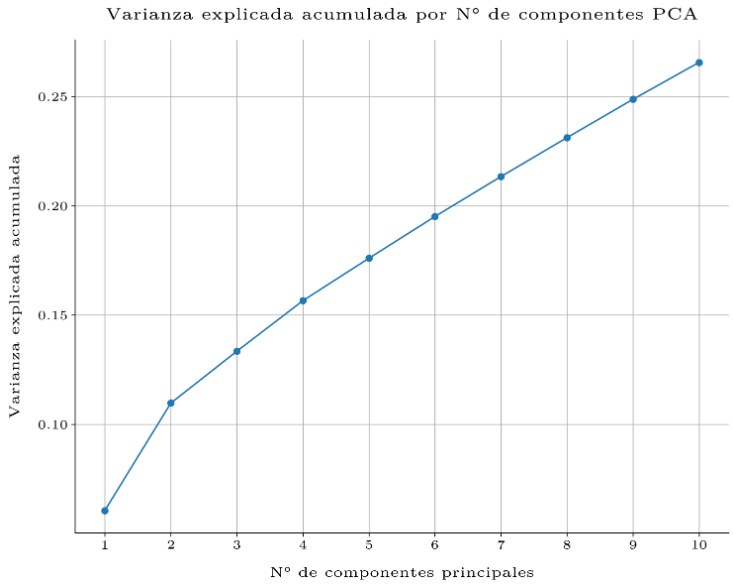

Figura 3: Varianza explicada acumulada según los 10 componentes principales.

Los dos primeros componentes principales, a pesar de su bajo porcentaje de varianza total explicada, evidentemente capturan las características lingüísticas que son más distintivas entre los tres oradores.

3. Entrenamiento y evaluacion

Una vez obtenidas las *features* para clasificación y validar la separabilidad de las mismas, entrenaremos modelos de clasificación. Para entrenarlos, se usará la separación de *train* (70 %) y *test* (30 %) antes mencionada, utilizando validación cruzada para la selección de parámetros.

3.1. Metricas

Para poder evaluar los modelos es necesario seleccionar métricas que capturen correctamente los objetivos del problema. Es tentador utilizar la exactitud (*accuracy*) para este problema, pero esta es muy susceptible a datos desbalanceados. Supongamos que el 95 % de los textos corresponden a un candidato. Si

nosotros predecimos todos los textos como ese candidato, vamos a tener un 95 % de exactitud, por lo que es importante utilizar múltiples métricas para medir el desempeño.

En nuestro caso, vamos a considerar 4 métricas multi-clase: la exactitud (4) (*accuracy*), la precisión (5) (*precision*), la exhaustividad (6) (*recall*) y el puntaje F1 7, que es la media armónica entre los dos. C es el conjunto de clases, TP son los verdaderos positivos, FP los falsos positivos y FN los falsos negativos.

$$Accuracy = \frac{1}{N} \sum_{i} 1\{\hat{y}_i \neq y_i\} \tag{4}$$

$$Precision = \frac{1}{C} \sum_{i=}^{C} \frac{TP_i}{TP_i + FP_i} \tag{5}$$

$$Recall = \frac{1}{C} \sum_{i=}^{C} \frac{TP_i}{TP_i + FN_i} \tag{6}$$

$$\text{F1-Score} = \frac{1}{C} \sum_{i=}^{C} \frac{2 \quad Precision_i \quad Recall_i}{Precision_i + Recall_i} \tag{7}$$

En donde:

- **Verdaderos Positivos TP):** El modelo predijo "Positivoz el valor real era "Positivo". Por ejemplo: El modelo predijo "Trumpz el discurso era de Trump.
- **Falsos Negativos FN):** El modelo predijo "Negativo"pero el valor real era "Positivo". Por ejemplo: El modelo no predijo "Trumpçuando el discurso era de Trump.
- **Falsos Positivos FP):** El modelo predijo "Positivo"pero el valor real era "Negativo". Por ejemplo: El modelo predijo "Trumpçuando el discurso era de Biden o Pence.
- **Verdaderos Negativos TN):** El modelo predijo "Negativoz el valor real era "Negativo". Por ejemplo: El modelo no predijo "Trumpz el discurso no era de Trump.

Estas métricas, aunque igual siguen siendo susceptibles al caso de un des-balance extremo, dan un panorama más claro del desempeño del modelo. Retomando

el ejemplo, notar que tanto *recall* como *precision* darían 0 para los candidatos que no son el mayoritario, por lo que el resultado sería $\frac{95}{C}$ donde C es el número de clases. Si solo tenemos 2 candidatos, pasamos a un desempeño de menos del 50 %. Para la selección de hiper-parámetros utilizaremos el puntaje F1, ya que depende tanto de la *precision* como del *recall*.

3.2. Validacion Cruzada

La validación cruzada es una estrategia común para evaluar modelos y seleccionar hiper-parámetros, especialmente útil cuando se dispone de un conjunto de entrenamiento reducido. La estrategia consiste en dividir los datos de entrenamiento en k conjuntos (*folds*), y realizando k entrenamientos donde se reserva uno de los conjuntos para validación y se entrena con los *k-1* restantes. Esto permite estimar el desempeño del modelo de forma más robusta y detectar posibles casos de sobreajuste.

Dada una métrica M, al usar validación cruzada se obtienen k muestras para cada métrica. Se define entonces:

$$M = \frac{1}{k} \sum_{i=}^{k} m_i$$

donde m_i es la métrica obtenida para cada conjunto.

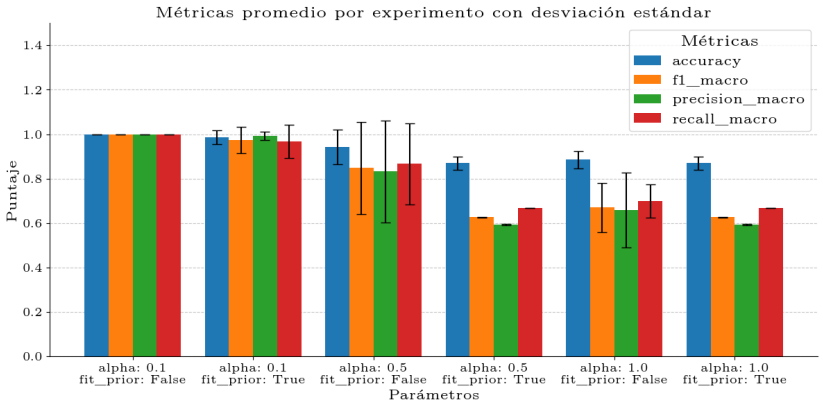

Figura 4: Resultados de las métricas durante la validación cruzada para al entrenar sobre los discursos, utilizando solo el texto dicho por candidato. Se incluye la desviación estándar en cada una de las barras.

En la Figura 4 se observan los resultados de la validación cruzada sobre las métricas anteriormente mencionadas, usando $k = 5$. Lo interesante a observar

en este gráfico es cómo algunos sets de parámetros son más propensos a sobreajustar los datos. A simple vista, vemos que para un set de parámetros las métricas dan siempre perfectas, lo cual es extraño pero no imposible. Fuera de este, vemos cómo los demás sets de parámetros obtienen resultados distintos para cada métrica, e incluso dentro de un set de parámetros y una métrica, se pueden observar resultados muy variados según el set de entrenamiento.

Una observación extra para destacar es que la *accuracy* siempre está por encima del resto de métricas. Esto valida el punto presentado anteriormente sobre el impacto del desbalance de clases en las métricas, y también valida la elección del puntaje F1 para la elección de modelo.

3.3. Modelos y Resultados

3.3.1. Multinomial Naive Bayes

El primer modelo utilizado para clasificar los discursos según el orador fue el *Naive Bayes Multinomial Classi er*. Este clasificador se basa en el Teorema de Bayes y asume una independencia condicional fuerte entre las características, dado el valor de la clase.

Como se mencionó anteriormente, se utilizó la validación cruzada para definir la mejor combinación de hiperparámetros para este modelo. La Figura 4 muestra que dicho procedimiento arrojó que la mejor combinación de hiperparámetros fue: *alpha* igual a 0,1 y *t_prior* igual a *False*. La mejor combinación en este caso no solamente implica que se obtuvieron mejores valores de los estadísticos, sino que éstos también presentaron una mayor estabilidad entre las cinco iteraciones.

Con los mencionados hiperparámetros, se volvió a entrenar el modelo con absolutamente todos los datos del set de entrenamiento y se obtuvo la matriz de confusión al aplicar el modelo sobre el set de datos de prueba. En la Figura 5 podemos ver que la *performance* del modelo fue perfecta, llegando a valores de *accuracy*, *precision* y *recall* del 100 % para todos los candidatos.

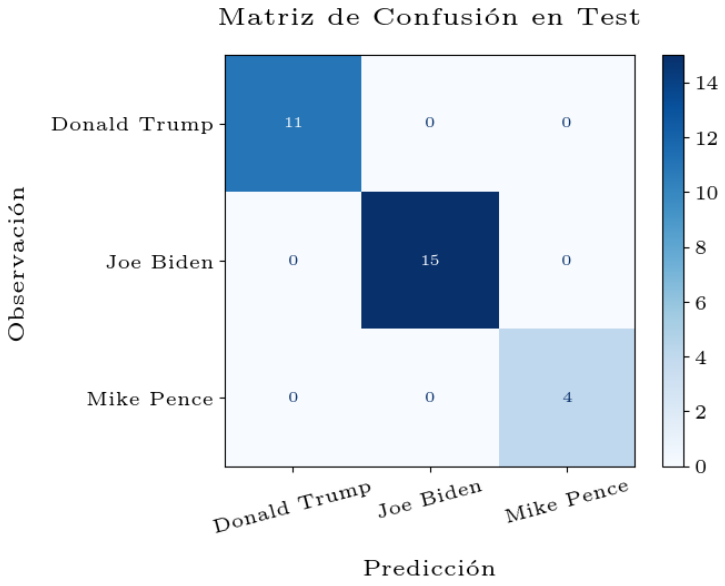

Figura 5: Matriz de confusión resultante de aplicar el Clasificador Multinomial Naive Bayes sobre set de datos de prueba.

3.3.2. Decision Tree

Se realizó también un experimento usando un árbol de decisión. Este modelo va creando un árbol que en cada nodo separa los datos en dos conjuntos de igual tamaño según cierto criterio. Al igual que para *Naive Bayes*, realizamos un *grid search* variando los siguientes parámetros:

- **criterion**: criterio para hacer la división. Opciones: *gini*, *entropy*.
- **max_depth**: profundidad máxima a la que puede llegar el arból. Mientras más chico, más regularizado está. Opciones: 10, 25, 50, 100, *Null* (crece ilimitadamente).
- **min_samples_split**: cuantas muestras tiene que haber como mínimo para hacer un split. Opciones: 2, 5, 10
- **splitter**: como realizar el split. Opciones: random, best.

Luego del *grid search*, se obtiene que la mejor combinación de hiperparámetros es *criterion* entropy, *max_depth* de 25, *min_samples_split* de 2 y *splitter* random. En la Figura 6 se observa la matriz de confusión obtenida. A diferencia del *Naive Bayes*, el árbol de decisión tiene peor *performance*, teniendo problemas para identificar a Trump.

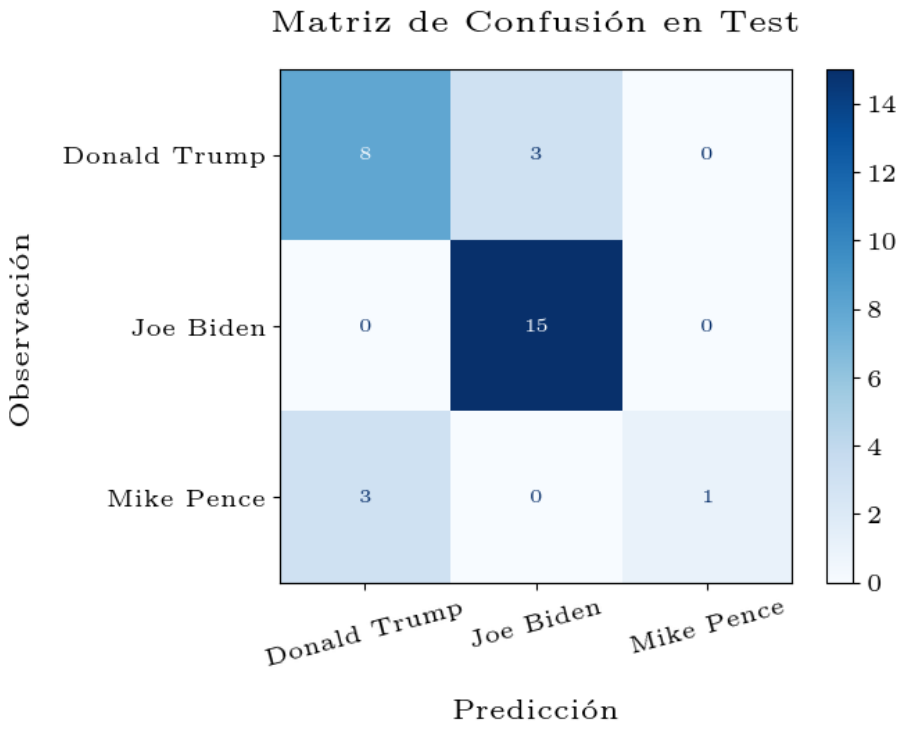

Figura 6: Matriz de confusión en los datos de test para el árbol de decisión con parámetros *entropy*, *max_depth* de 25, *min_samples_split* de 2 y *splitter* random

Esto probablemente se deba a la alta dimensionalidad de los datos de entrada, sumado a la baja cantidad de datos que se tienen, hace que el modelo de árbol de decisión no logre aprender lo suficiente para poder generalizar. Coincidentemente, la profundidad máxima es 25, ya que durante el *grid search* se vio la tendencia a sobreajustar que tienen estos modelos.

3.4. Explicabilidad

Vamos a hacer un análisis de los resultados obtenidos para el modelo de *Naive Bayes*. Algo que inmediatamente llamó la atención fue el desempeño perfecto en *test*. Es por esto que quisimos ver en qué se estaba basando el modelo para hacer las predicciones.

Naive Bayes es un modelo con una alta explicabilidad, ya que para cada N-grama tenemos el valor de *log-likelyhood* para cada clase, que es lo que usa para

hacer la predicción. Por lo tanto, podemos usar esos valores para ver cuales son las características más distintivas en cada discurso. Para hacer este cálculo, dada una clase se buscaron las *features* que tengan más diferencia con el promedio del resto de clases. En el Cuadro 1 se muestra los resultados obtenidos.

Donald Trump	**Joe Biden**	**Mike Pence**
fake (4.47)	significant (3.87)	president donald (5.15)
sleepy (4.15)	joey (3.68)	opening america (4.01)
fake news (4.11)	crisis (3.40)	trump stood (3.90)
sleepy joe (4.08)	beau (3.40)	women serve (3.76)
fantastic (3.94)	awful (3.39)	couldn proud (3.69)
horrible (3.61)	systemic (3.35)	republican majority (3.67)
sir (3.58)	god protect (3.33)	unleashed american (3.58)
great people (3.53)	making sure (3.32)	supporting law (3.58)
said sir (3.52)	build better (3.29)	susan (3.56)
crazy (3.52)	care act (3.24)	trump white (3.55)

Cuadro 1: Palabras que más distinguen a un candidato del resto. El número que acompaña a cada palabra es la distancia de la *log-likelyhood* de la frase para el candidato con el promedio, comparado con el promedio para el resto de candidatos.

Rápidamente se observan ciertas tendencias de los discursos de los candidatos que ayudan a discernirlos. Trump, por ejemplo, es el que más utiliza el ápodo despectivo *"sleepy joe"* para referirse a su competidor. Biden por su parte, hace uso de palabras más negativas, como *"crisis"* o *"awful"*, que es coherente con su postura de oposición, a diferencia de Pence y Trump. Con este rápido análisis validamos que los resultados tienen sentido, y descartamos problemas de fuga de datos (*data leak*), es decir, que datos que podrían identificar al orador (los tags de inicio de frase, por ejemplo) hayan quedado en los datos de entrenamiento y validación.

4. Impacto del desbalance de clases

Si hacemos el ejercicio de incorporar otro candidato, por ejemplo Kamala Harris, en lugar de uno de los dos candidatos con más discursos, por ejemplo Joe Biden, podemos notar un clásico problema de desbalance entre las clases del conjunto de datos. Con este cambio, Donald Trump queda como la clase predominante, mientras que Kamala Harris y Mike Pence tienen una proporción marginal de los discursos.

Los algoritmos de aprendizaje automático (incluido *Naive Bayes*) están diseñados para minimizar el error general. En un set de datos desbalanceado, minimizar el error general a menudo significa priorizar la clasificación correcta

de la clase mayoritaria. En ese sentido, el modelo podría volverse muy bueno en identificar discursos de Donald Trump (alta *precision* y *recall* para Trump), pero muy malo en identificar los de Kamala Harris o Mike Pence (baja *precision* y *recall* para las clases minoritarias), porque cometer un error en una clase minoritaria tiene un impacto mucho menor en el error total que un error en una clase mayoritaria.

En la Figura 7 podemos notar cómo la validación cruzada del Clasificador Multinomial Naive Bayes da resultados significativamente menores con un set de datos de entrenamiento desbalanceado. Lo más interesante es ver lo que venimos comentando, los valores de *accuracy* del modelo para toda combinación de hiperparámetros son bastante "satisfactorios", pero cuando pasamos a los demás estadísticos enseguida notamos que el modelo tiene un sesgo hacia la clase mayoritaria.

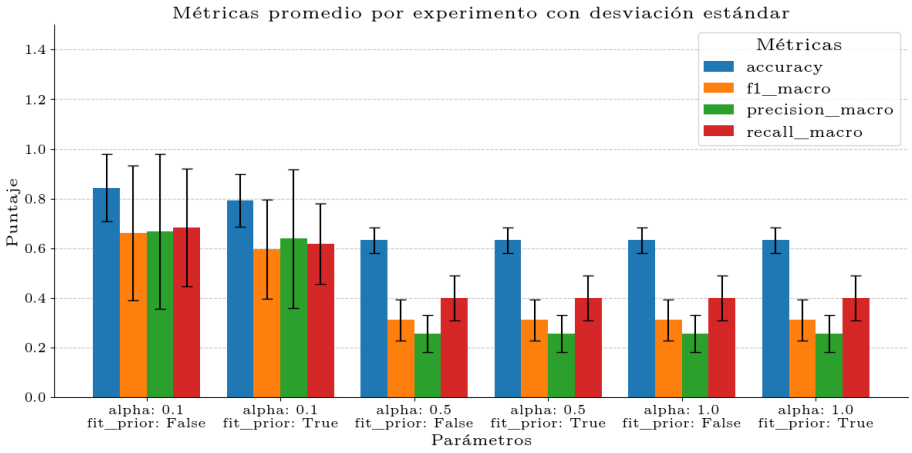

Figura 7: Resultados de las métricas durante la validación cruzada en un set de datos de entrenamiento desbalanceado. Se incluye la desviación estándar en cada una de las barras.

5. Alternativas para extraccion de features

A pesar de haber conseguido importantes resultados utilizando transformaciones con TF-IDF, esta técnica (al igual que la técnica BoW) tiene limitaciones en cuanto al análisis de texto. Por ejemplo, ambas técnicas tratan a un documento como una colección desordenada de palabras, donde la secuencia en la que aparecen las palabras es completamente ignorada. En consecuencia, estos modelos no pueden capturar la gramática, la sintaxis, las relaciones de dependencia o

las estructuras de las oraciones. Esto es una limitación severa para tareas que requieren una comprensión profunda del significado o la intención.

Además de las limitaciones en cuanto a capturar el sentido de los textos, las técnicas de BoW y TF-IDF generan una alta dimensión y dispersión debido a que el vocabulario de cualquier corpus de texto suele ser muy grande. No solo ello, sino que de forma posterior en cada documento de ese corpus es muy común que la mayoría de las palabras del vocabulario no aparezcan, lo cual genera matrices dispersas tal y como definimos al inicio de este informe. Las matrices dispersas y de alta dimensionalidad consumen mucha memoria y requieren más tiempo computacional para almacenar y procesar.

Los algoritmos de aprendizaje automático a menudo tienen dificultades para funcionar eficientemente en espacios de alta dimensionalidad, donde las distancias y similitudes se vuelven menos significativas. Un modelo con demasiadas características (en estos casos, palabras) puede ser propenso al sobreajuste, especialmente con conjuntos de datos de entrenamiento reducidos.

Una alternativa para extraer *features* de texto es utilizar *Word Embeddings*. En esencia, los *Word Embeddings* son utilizados en el dominio del Aprendizaje Automático para referirse a una representación vectorial **aprendida** que contiene información del valor de entrada. El concepto de *embedding* para texto se presenta en el paper de Word2Vec [2], donde a cada *token* de entrada se le asigna un vector que se aprende durante el entrenamiento de un modelo de aprendizaje profundo. Al final del entrenamiento, el vector obtenido para cada *token* capturó la semántica de la palabra.

Un tiempo después, se presentó una estrategia para obtener un vector que capture la semántica de toda una frase en vez de solo palabras. El modelo más usado con este propósito es BERT [1], el cual derivó en una familia de modelos de codificación adaptados a distintos idiomas y dominios. Estos modelos son de código abierto y están disponibles en una variedad de librerías, por ejemplo Sentece Transformers .

Uno de los modelos de *embeddings* más usados hoy en día es el de *OpenAI*. A diferencia de BERT, el modelo de *OpenAI* es propietario y secreto, aunque hicieron una publicación [3] donde explican la estrategia de entrenamiento para obtener un modelo que captura la semántica de un texto.

¿Cómo se usarían? Utilizando algún modelo de *embeddings* se obtiene un vector para cada texto. Con estos vectores como entrada, se puede entrenar un MLP simple para clasificar. Según la cantidad de datos y candidatos, se debe variar la complejidad de la capa de MLP.

`https://sbert net/`

6. Conclusiones

El presente trabajo se centró en la aplicación práctica de conceptos fundamentales del aprendizaje automático y el procesamiento de texto para abordar el problema de la clasificación de discursos de los principales candidatos: Donald Trump, Joe Biden y Mike Pence.

La fase de preprocesamiento y exploración de datos fue crucial, donde se transformaron los discursos a una representación numérica utilizando la técnica TF-IDF. Se observó que, a pesar de que los dos primeros componentes principales del PCA solo explicaban aproximadamente el 10 % de la varianza acumulada, estos componentes lograban una segregación visual bastante clara de los candidatos en la proyección bidimensional. Este fenómeno sugiere que la varianza más relevante para la discriminación entre oradores se concentra en estas dos dimensiones principales, independientemente de la varianza total de los datos originales que puede estar relacionada con variaciones intra-candidato, temas diversos o ruido.

En cuanto al entrenamiento y evaluación de modelos, se utilizó un clasificador Multinomial Naive Bayes. La selección de hiperparámetros se realizó mediante validación cruzada (k=5), enfocándose en el puntaje F1 para mitigar la susceptibilidad de la exactitud (*accuracy*) al desbalance de clases. Los resultados de la validación cruzada mostraron que, si bien la exactitud tendía a ser alta, otras métricas como *precision*, *recall* y *F1-score* en promedio eran más sensibles a los desbalances.

Sorprendentemente, el modelo Multinomial Naive Bayes entrenado con estos hiperparámetros alcanzó una performance perfecta en el conjunto de prueba, obteniendo 100 % en *accuracy*, *precision* y *recall* para todos los candidatos. Si bien estos resultados son excepcionales, se realizó un análisis de explicabilidad para validar la base de las predicciones. La inspección de las palabras más distintivas para cada candidato (como *"sleepy joe"* para Trump o *çrisis"* para Biden) validó que el modelo se estaba basando en características lingüísticas coherentes y significativas, ayudando a descartar problemas de fuga de datos.

Finalmente, se abordó el impacto del desbalance de clases, un desafío crucial en la clasificación. Al simular un escenario con Kamala Harris en lugar de Joe Biden, generando un desbalance importante, se observó claramente cómo el modelo tendía a priorizar la clasificación correcta de la clase mayoritaria (Donald Trump en ese caso), resultando en una baja *precision* y *recall* para las clases minoritarias (Kamala Harris y Mike Pence), a pesar de que la exactitud general se mantenía en buenos niveles. Esta observación refuerza la necesidad de utilizar métricas robustas como el *F1-Score* para una evaluación precisa en escenarios reales.

Se reconocieron las limitaciones inherentes a las técnicas de *Bag of Words* y

TF-IDF, como la pérdida del orden de las palabras, la incapacidad de capturar la gramática y el significado contextual, y los desafíos de alta dimensionalidad y dispersión de datos. Estas limitaciones señalan la relevancia de futuras exploraciones con técnicas de extracción de características más avanzadas, como los *Word Embeddings* densos y contextualizados.

En conclusión, este informe demuestra la efectividad del Naive Bayes con TF-IDF para la clasificación de oradores en un escenario relativamente equilibrado y la importancia crítica de una evaluación de métricas adecuada para identificar y mitigar los desafíos que surgen del desbalance de clases en problemas del mundo real.

Referencias

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Je rey Dean. Efficient estimation of word representations in vector space, 2013.

[3] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pre-training, 2022.