FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN CIENCIA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2024/2025

Trabajo de Fin de Máster

TITULO: Modelos de "Machine Learning" para la detección ideológica en Twitter: Evaluación y aplicación práctica

***Alumno*: Gerardo Javier Benavides Martínez**

***Tutor*: Inmaculada Gutiérrez García-Pardo**

Junio de 2025

UNIVERSIDAD COMPLUTENSE
MADRID

UNIVERSIDAD COMPLUTENSE MADRID

Declaración Responsable sobre Autoría y Uso Ético de Herramientas de Inteligencia Artificial (IA)

Yo, Gerardo Benavides

Con DNI: 17549092T

declaro de manera responsable que el/la presente:

- ☐ Trabajo de Fin de Grado (TFG)
- X Trabajo de Fin de Máster (TFM)
- ☐ Tesis Doctoral

Titulado/a

Modelos de “Machine Learning” para la detección ideológica en Twitter: Evaluación y aplicación práctica

es el resultado de mi trabajo intelectual personal y creativo, y ha sido elaborado de acuerdo con los principios éticos y las normas de integridad vigentes en la comunidad académica y, más específicamente, en la Universidad Complutense de Madrid.

Soy, pues, autor del material aquí incluido y, cuando no ha sido así y he tomado el material de otra fuente, lo he citado o bien he declarado su procedencia de forma clara -incluidas, en su caso, herramientas de inteligencia artificial-. Las ideas y aportaciones principales incluidas en este trabajo, y que acreditan la adquisición de competencias, son mías y no proceden de otras fuentes o han sido reescritas usando material de otras fuentes.

Asimismo, aseguro que los datos y recursos utilizados son legítimos, verificables y han sido obtenidos de fuentes confiables y autorizadas. Además, he tomado medidas para garantizar la confidencialidad y privacidad de los datos utilizados, evitando cualquier tipo de sesgo o discriminación injusta en el tratamiento de la información.

En Madrid a Junio de 2025

FIRMA

Resumen

La era de la información trae consigo el creciente uso de redes sociales, este trabajo evalúa la capacidad de distintos modelos de aprendizaje automático y profundo para interpretar emociones y clasificar contenido ideológico en mensajes publicados en Twitter. La metodología combina dos enfoques: uno experimental, basado en el entrenamiento y comparación de modelos supervisados sobre un conjunto de datos etiquetado (Kaggle); y otro aplicado, donde se implementa un modelo de tipo Zero-Shot sobre un corpus real recolectado durante la campaña electoral de 2024 mediante "*web scraping*". Se analizan modelos basados en algoritmos clásicos (SVM, Regresión Logística, "*Naive Bayes*", "*Random Forest*", "*Gradient Boosting*", "*Ridge Classifier*" y "*Passive-Aggressive*") y avanzados basados en lenguaje (BETO, SBERT, "*Zero-Shot*" con BART). Los resultados permiten valorar la precisión, escalabilidad y aplicabilidad práctica de cada técnica, demostrando que estas herramientas son útiles para automatizar el análisis de sentimientos, emociones e ideologías en entornos digitales complejos como las redes sociales.

Palabras clave: análisis de sentimiento, Twitter, aprendizaje automático, Zero-Shot, emociones, ideología, redes sociales.

Abstract

In an age of information overload and widespread use of social networks, this study evaluates the effectiveness of various machine learning and deep learning models to interpret emotions and classify ideological content in Twitter messages. The methodology combines two approaches: an experimental phase based on training and comparing supervised models using a labeled dataset (Kaggle), and an applied case where a Zero-Shot model is implemented on a real-world corpus collected during the 2024 electoral campaign via web scraping. Both classical algorithms (SVM, Logistic Regression, Naive Bayes, Random Forest, Gradient Boosting, Ridge Classifier, and Passive-Aggressive) and advanced language-based models (BETO, SBERT, Zero-Shot with BART) are analyzed. Results provide insight into the accuracy, scalability, and real-world applicability of each technique, confirming their utility for automating the analysis of sentiment, emotions, and ideology in complex digital environments such as social media.

Keywords: sentiment analysis, Twitter, machine learning, Zero-Shot, emotions, ideology, social networks.

Índice

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

En la era digital, las redes sociales se han convertido en una de las principales fuentes de expresión, interacción y difusión de opiniones sobre temas de interés público. Plataformas como Twitter concentran millones de publicaciones diarias, muchas de ellas relacionadas con acontecimientos sociales, culturales, económicos y, especialmente, políticos. Este volumen masivo de contenido textual espontáneo representa una oportunidad única para conocer en tiempo real el estado de ánimo, las actitudes y las percepciones de distintos sectores de la población.

El análisis de sentimiento, en este contexto, ha emergido como una herramienta fundamental para extraer conocimiento a partir de los datos generados en redes sociales (Sharma, Ali y Kabir 2025). Esta disciplina combina técnicas de procesamiento del lenguaje natural (PLN), estadística y aprendizaje automático para detectar y clasificar las emociones u opiniones expresadas en los textos (Suryawanshi, 2024). Sus aplicaciones son múltiples y abarcan desde el marketing hasta la predicción electoral, pasando por el monitoreo de políticas públicas, la gestión de crisis o la detección de corrientes ideológicas.

En palabras de Patel y Goswami (2024), Twitter presenta características que lo convierten en una fuente privilegiada para este tipo de estudios: su carácter abierto, su naturaleza textual breve y su uso frecuente por actores políticos, medios de comunicación y ciudadanos, lo hacen ideal para explorar patrones discursivos y dinámicas de interacción. Sin embargo, este mismo entorno plantea desafíos importantes, como la ambigüedad del lenguaje, la ironía, la velocidad de propagación del contenido y la necesidad de herramientas que permitan procesar datos en tiempo real (Suryawanshi, 2024).

Este trabajo se desarrolla en ese marco, y tiene como objetivo evaluar y comparar el rendimiento de diferentes modelos de *"Machine Learning"* en tareas de clasificación de sentimiento e ideología en mensajes publicados en "*Twitter*". A través de un enfoque metodológico estructurado, que incluye la recolección, limpieza, representación y modelización de datos textuales, se busca no solo determinar qué modelos ofrecen mejores resultados, sino también aportar evidencia sobre su aplicabilidad en contextos sociales reales.

Para ello, el trabajo se estructura en dos partes complementarias. En la primera, se entrenan y evalúan diversos modelos de aprendizaje automático y profundo utilizando un conjunto de datos experimental (Kaggle), con el fin de identificar las técnicas más eficaces en la clasificación ideológica de tweets. En la segunda parte, se desarrolla un caso práctico aplicado basado en técnicas de *"Zero-Shot"*, utilizando un corpus original de mensajes políticos recolectados en tiempo real mediante *"web scraping"* durante la campaña electoral de 2024. Esta doble vía permite contrastar el rendimiento técnico de los modelos con su aplicabilidad en un entorno político actual.

1.1 Contextualización

1.1.1 Machine Learning

El aprendizaje automático, o *"Machine Learning"* (ML), constituye una rama esencial de la inteligencia artificial (IA) enfocada en el diseño de algoritmos capaces de aprender a partir de datos y mejorar su desempeño sin intervención humana directa. Según Taye (2023), el principio fundamental del ML es que los sistemas pueden adaptarse y optimizarse automáticamente a medida que reciben más información, lo que permite una toma de decisiones más eficaz y precisa en tareas específicas. Esta capacidad de aprender de la experiencia lo ha convertido en una herramienta estratégica en ámbitos como la medicina, el transporte, la agricultura o el análisis de datos y textos en redes sociales y portales de internet (Taye, 2023).

El concepto de que una máquina pueda adquirir conocimiento no es nuevo. Ghosh (2024) traza el origen del aprendizaje automático en los trabajos pioneros de Carl Friedrich Gauss y Adrien-Marie Legendre sobre el método de mínimos cuadrados que es considerado una de las bases del modelado estadístico moderno. Posteriormente, en la década de 1950, Frank Rosenblatt desarrolló el perceptrón, un modelo computacional inspirado en el cerebro humano, capaz de aprender tareas simples de clasificación. Este avance representó un punto de inflexión en la evolución de las redes neuronales, como indica también el autor, marcando la transición de modelos estadísticos a enfoques conexionistas más dinámicos. Años más tarde, la introducción del algoritmo de retropropagación impulsó de manera significativa el desarrollo del aprendizaje profundo (*"Deep Learning"*), permitiendo el entrenamiento efectivo de redes neuronales de múltiples capas (Ghosh, 2024).

En cuanto a su clasificación, Madhushani (2022) distingue claramente entre distintos tipos de aprendizaje, cada uno con sus particularidades metodológicas. El aprendizaje supervisado, el más común en la práctica, se basa en el uso de datos etiquetados. En este enfoque, el algoritmo recibe entradas junto con sus correspondientes salidas deseadas, lo que le permite establecer relaciones predictivas. Como señala la autora, este tipo de aprendizaje se utiliza ampliamente en tareas de clasificación, como la detección de spam, y de regresión, como la predicción de precios.

En contraposición, el aprendizaje no supervisado opera sobre datos sin etiquetar. Aquí, el modelo busca patrones o estructuras ocultas sin tener una referencia explícita del resultado esperado. Madhushani (2022) explica que esta técnica es útil, por ejemplo, en la agrupación de clientes según su comportamiento de compra o en la detección automática de anomalías. Además, destaca que el aprendizaje no supervisado suele ser más eficiente en términos de tiempo y coste, ya que no requiere una fase previa de etiquetado manual.

Taye (2023) introduce una categoría intermedia el aprendizaje semi-supervisado, que combina ambos enfoques. Este tipo de modelo se entrena con una pequeña cantidad de datos

etiquetados y una gran proporción de datos sin etiquetar, lo cual permite mejorar la precisión de los sistemas en contextos donde etiquetar los datos sería demasiado costoso o inviable.

También existe el aprendizaje por refuerzo, una técnica en la que un agente aprende mediante la interacción con su entorno, recibiendo recompensas o penalizaciones según las decisiones que toma. Aunque no se utiliza directamente en este trabajo, ha sido fundamental en avances recientes en robótica, sistemas de recomendación y juegos complejos como el ajedrez o "*Go*" (Taye, 2023).

En términos de arquitectura, el aprendizaje profundo *"Deep Learning"* (DL) ha transformado el panorama del ML moderno. Este enfoque se basa en redes neuronales profundas capaces de aprender representaciones jerárquicas de los datos. Ghosh (2024) subraya que estas arquitecturas permiten una extracción automática de características relevantes, lo que reduce la necesidad de intervención humana en el diseño del modelo. Taye (2023) complementa esta visión al destacar que el *"Deep Learning"* ha superado las capacidades humanas en tareas complejas como el reconocimiento de imágenes y el procesamiento del lenguaje natural.

Ahora bien, el desarrollo de modelos de *"Machine Learning"* no está exento de desafíos. Uno de los aspectos más críticos es la ética del uso de estos sistemas. Taye (2023) advierte sobre el riesgo de sesgos en los datos de entrenamiento, lo que puede derivar en decisiones discriminatorias o poco transparentes. Además, el coste computacional de los modelos más avanzados, así como la necesidad de grandes volúmenes de datos de calidad, plantea barreras importantes para su implementación en entornos con recursos limitados. En la misma línea, Madhushani (2022) destaca la importancia de abordar la privacidad de los datos y la explicabilidad de los modelos como aspectos clave para una adopción responsable de estas tecnologías.

En conclusión, el *"Machine Learning"* ha evolucionado desde sus raíces matemáticas y estadísticas hasta convertirse en un campo multidisciplinar con aplicaciones de alto impacto en diversos sectores. Su capacidad de adaptación, mejora continua y tratamiento de grandes volúmenes de datos lo convierten en una herramienta indispensable en la ciencia de datos moderna. No obstante, es fundamental seguir desarrollando enfoques éticos, sostenibles y explicables para garantizar un uso justo y eficaz de estas tecnologías.

1.1.2 Análisis de sentimientos

El análisis de sentimiento, también denominado *"Opinion Mining"* (OM), es una disciplina del procesamiento de lenguaje natural que se dedica a identificar y clasificar automáticamente opiniones expresadas en texto. Sharma, Ali y Kabir (2025) definen este proceso como la tarea de determinar la polaridad emocional positiva, negativa o neutra de una opinión en relación con una entidad o tema específico. Este campo ha cobrado relevancia en la era digital, especialmente ante la proliferación de contenido generado por usuarios en redes sociales, reseñas y foros, donde millones de textos reflejan percepciones y valoraciones sobre productos, servicios o eventos (Sharma, Ali y Kabir, 2025).

A lo largo de los últimos años, el análisis de sentimiento ha evolucionado considerablemente gracias al desarrollo de técnicas avanzadas de aprendizaje automático y profundo. En palabras de Suryawanshi (2024), el paso de enfoques basados en palabras clave a modelos capaces de capturar contexto y semántica ha transformado la precisión con la que los sistemas comprenden el lenguaje humano. Este avance ha sido posible por la adopción de arquitecturas como redes neuronales recurrentes y modelos basados en atención, que permiten procesar la estructura y el significado del texto más allá de la simple frecuencia de términos (Suryawanshi, 2024).

Existen diversos enfoques metodológicos para abordar el análisis de sentimiento. Uno de los más extendidos es el aprendizaje supervisado, que requiere conjuntos de datos etiquetados para entrenar modelos predictivos. Este método ha demostrado ser eficaz en tareas de clasificación, y suele emplear algoritmos como SVM, regresión logística, "*Naive Bayes*", "*Random Forest*" y "*Gradient Boosting*" (Patel & Goswami, 2024). Como destacan los autores, su principal fortaleza reside en la capacidad de generalizar a partir de ejemplos, siempre que la calidad y representatividad de los datos de entrenamiento sea adecuada.

En contraste, los métodos no supervisados no dependen de datos etiquetados. Patel y Goswami (2024) explican que estos se apoyan en técnicas como el agrupamiento ("*Clustering*"), el análisis de temas o el uso de lexicones para inferir sentimientos. Aunque generalmente ofrecen una menor precisión, resultan útiles cuando el etiquetado manual es inviable por razones de coste o volumen de datos. También permiten explorar estructuras emergentes sin imponer una clasificación previa, lo que los hace valiosos en etapas exploratorias.

Suryawanshi (2024) añade una tercera vía los enfoques híbridos, que combinan técnicas supervisadas y no supervisadas para aprovechar lo mejor de ambos mundos. En estos sistemas, una parte del conjunto se etiqueta manualmente para guiar el aprendizaje, mientras que otra se analiza de forma automática o se refina con lexicones y reglas. Este tipo de solución es cada vez más común en escenarios reales, donde los datos son abundantes, pero no siempre estructurados.

Las aplicaciones del análisis de sentimiento abarcan una amplia variedad de sectores. Según Sharma et al. (2025), estas herramientas se emplean en áreas como el comercio electrónico, la sanidad, la educación, la política y los medios de comunicación. Por ejemplo, analizar las opiniones sobre una marca puede ayudar a predecir comportamientos de compra o ajustar campañas de marketing. Asimismo, en contextos políticos, el análisis de sentimiento permite rastrear la percepción pública sobre figuras o políticas, ofreciendo a los actores institucionales datos en tiempo real sobre su impacto social.

No obstante, como advierte Suryawanshi (2024), este campo enfrenta retos significativos. Uno de los más persistentes es la ambigüedad inherente al lenguaje natural. Palabras con múltiples significados, frases sarcásticas o irónicas, y expresiones regionales dificultan la correcta interpretación del sentimiento. Sharma et al. (2025) enfatizan que incluso los modelos más sofisticados pueden verse superados por construcciones lingüísticas complejas que requieren un entendimiento contextual profundo. A esto se suman desafíos como los

errores ortográficos, el uso de emoticones, o la mezcla de idiomas, fenómenos comunes en plataformas sociales que complican el análisis computacional.

El vínculo entre análisis de sentimiento y aprendizaje automático es inseparable. Como explican Patel y Goswami (2024), sin técnicas de clasificación automática, sería inviable procesar el volumen y la complejidad de los datos textuales generados diariamente. El aprendizaje automático permite extraer patrones implícitos en los textos y construir modelos que se adaptan dinámicamente a nuevos contextos. En este sentido, los modelos de aprendizaje profundo han superado los enfoques clásicos en muchas tareas específicas, al capturar dependencias más largas y matices semánticos con mayor precisión (Suryawanshi, 2024).

En resumen, el análisis de sentimiento es una herramienta crucial para interpretar las emociones humanas reflejadas en lenguaje escrito. Su desarrollo ha sido impulsado por la sinergia entre técnicas lingüísticas y modelos de aprendizaje automático, permitiendo avances sustanciales en áreas como la atención al cliente, la inteligencia de negocios o la evaluación política. Sin embargo, como reconocen Sharma et al. (2025), los retos técnicos y lingüísticos aún persisten, lo que convierte este campo en un área activa de investigación y desarrollo.

1.2 Estado del arte

Durante la última década, el análisis de sentimiento ha emergido como una herramienta clave para entender dinámicas sociales complejas, particularmente en el ámbito político y en entornos digitales como "*Twitter*" ahora "*X*". Esta red social se ha consolidado como una fuente masiva de datos textuales donde los usuarios expresan opiniones, emociones y posturas ideológicas en tiempo real. Según Rodríguez Alcántara et al (2022), esta plataforma se ha convertido en un espacio privilegiado para el discurso político ciudadano, permitiendo monitorear tendencias y puntos de inflexión en la opinión pública a partir del contenido publicado por los usuarios (Rodríguez Alcántara, Falck Durán y Suazo Barahona, 2022).

Uno de los principales intereses de la comunidad científica ha sido estudiar el potencial del aprendizaje automático para automatizar este tipo de análisis. Tun y Khaing (2023) destacan que, frente al volumen creciente de datos políticos generados en redes sociales, se ha vuelto indispensable aplicar técnicas de clasificación automática capaces de identificar la polaridad ideológica o emocional de los mensajes. Su investigación propone un sistema de análisis multicategoría basado en grandes volúmenes de tuits, utilizando un enfoque supervisado con datos previamente etiquetados mediante un léxico político, diseñado por los autores, para identificar opiniones extremas, el sistema logró una precisión del 98% con algoritmos SVM de tipo lineal, lo que evidencia el poder de estas técnicas en contextos electorales complejos.

Por otro lado, los desafíos del análisis político en "*Twitter*" no solo tienen que ver con el volumen, sino también con el contexto social. Como señalan Valle-Cruz et al. (2024), fenómenos como la polarización política y las cámaras de eco han distorsionado la deliberación democrática en entornos digitales. En su estudio sobre las elecciones presidenciales en México de 2018, los autores aplicaron modelos de aprendizaje automático para detectar los niveles

de positividad, negatividad y neutralidad en los tuits dirigidos a cada candidato. Sus hallazgos revelaron que el candidato vencedor mostró los niveles más altos de polarización, lo que sugiere una estrecha relación entre emoción, viralidad y éxito electoral.

En el ámbito académico hispanoamericano, Madurga y Payo también han contribuido con un enfoque estructurado al problema. Estos autores proponen una arquitectura completa para el análisis de emociones y sentimientos en redes sociales, desde la recolección de datos hasta la clasificación utilizando técnicas como "*Bag of Words*", "*Word2Vec*" y modelos supervisados como SVM, "*Random Forest*" y Regresión Logística (Madurga & Payo, s.f., 2020). Lo interesante de su propuesta es la combinación de diferentes técnicas para optimizar la precisión del sistema y adaptarse a las peculiaridades lingüísticas del español (Madurga & Payo, s.f., 2020).

Más allá del ámbito electoral, Saura, Reyes-Menéndez y Palos-Sánchez (2018) exploraron el potencial del análisis de sentimiento en eventos comerciales como el "*Black Friday*". Su estudio demostró cómo los modelos de aprendizaje automático pueden capturar la percepción de los consumidores hacia campañas específicas, confirmando que el análisis de "*Twitter*" no se limita a la política, sino que también tiene aplicaciones en marketing, consumo y gestión de la reputación digital.

Ahora bien, uno de los puntos más relevantes de estos estudios es la identificación de patrones repetitivos en el diseño metodológico. La mayoría de las investigaciones analizadas aplican un flujo que comienza con la recolección de tuits aplicando filtros por "*Hashtags*" o cuentas, seguido por preprocesamiento textual, vectorización de características y aplicación de clasificadores supervisados. Entre los algoritmos más comunes se encuentran "*Naive Bayes*", Regresión Logística y SVM (Tun & Khaing, 2023).

No obstante, persisten importantes desafíos metodológicos y éticos. La calidad del análisis depende en gran medida de la representatividad del corpus, la gestión del ruido lingüístico y la capacidad del modelo para interpretar fenómenos como el sarcasmo o la ironía. Además, Valle-Cruz et al. (2024) advierten sobre el impacto potencial del uso de estas tecnologías en la profundización de la polarización ideológica, lo que plantea preguntas sobre la responsabilidad ética de quienes desarrollan e implementan estos sistemas.

En este contexto, comienza a ganar relevancia el uso de enfoques de aprendizaje "*Zero-Shot*", los cuales permiten clasificar textos sin necesidad de datos etiquetados previamente para cada categoría (Clarke et al, 2023). Esta capacidad resulta especialmente valiosa en dominios donde la anotación manual es costosa o inviable, como ocurre en entornos políticos dinámicos o multilingües (Shu et al, 2022). Clarke et al. (2023) proponen estrategias de preentrenamiento que mejoran la generalización de modelos ante etiquetas no vistas, permitiendo una clasificación más flexible y escalable.

En resumen, el estado del arte del análisis de sentimiento en redes sociales, y particularmente en contextos políticos, demuestra un creciente interés por aplicar técnicas de "*Machine Learning*" para comprender mejor la opinión pública. Estos trabajos no solo han revelado el potencial de "*Twitter*" como barómetro social, sino también los límites y riesgos de

automatizar la interpretación del discurso político. A medida que se perfeccionan los modelos y se amplía la disponibilidad de datos, es probable que el análisis de sentimiento se convierta en una herramienta aún más influyente en el estudio de las ciencias sociales y la toma de decisiones públicas y comerciales.

Capítulo 2. Fuentes de datos

El conjunto de datos utilizado en este estudio proviene de la plataforma de ciencia de datos Kaggle, específicamente del dataset titulado "Tweets Política España" (Moya, 2022), disponible públicamente en enlace: https://www.kaggle.com/datasets/ricardomoya/tweets-poltica-espaa. Este corpus contiene más de 200,000 tuits publicados por las cuentas oficiales de partidos políticos españoles, entre ellos el Partido Socialista Obrero Español (PSOE), Podemos, Partido Popular (PP) y Vox. Todos los mensajes están etiquetados por partido de origen, lo cual facilita su análisis comparativo.

Para la construcción del conjunto de entrenamiento, se adopta la premisa de que los contenidos generados por las cuentas oficiales reflejan fielmente la orientación ideológica institucional de los respectivos partidos. Esta aproximación metodológica ha sido validada en investigaciones previas sobre análisis de sentimiento y detección de afinidades políticas a través de redes sociales (Khan y Khan, 2024). Como señalan Khan y Khan (2024), el contenido publicado en plataformas como "*Twitter*" puede ser utilizado de forma efectiva para identificar patrones de afiliación política, ya que los mensajes reflejan una línea argumentativa alineada con la posición ideológica del emisor institucional.

Siguiendo esta lógica, se ha realizado una agrupación binaria del espectro político español para simplificar el problema de clasificación. En concreto, se han considerado las siguientes asignaciones:

- Izquierda: PSOE y Podemos
- Derecha: Partido Popular y Vox

Esta categorización es coherente con la representación ideológica comúnmente aceptada en la literatura sobre política española, y ha sido aplicada también en estudios internacionales. Por ejemplo, Boulianne (2015) destaca que las cuentas institucionales de partidos políticos, al ser gestionadas por sus propios equipos de comunicación, actúan como portavoces autorizados de la narrativa ideológica de dichas formaciones. Esta asunción proporciona una base sólida para emplear las siglas partidistas como etiquetas supervisadas en modelos de clasificación política.

Además, tal como subraya Khan et al. (2024), el uso de técnicas de aprendizaje automático para inferir afinidades políticas a partir de contenido textual no solo es factible, sino también altamente preciso, siempre que se cuente con ejemplos representativos y bien etiquetados. En ese sentido, el uso de cuentas verificadas y oficiales otorga mayor legitimidad y consistencia a la etiqueta ideológica asignada a cada tuit.

En este proyecto, por tanto, se parte de una estructura de clasificación binaria ("izquierda" vs. "derecha"), que reduce la complejidad del problema y permite una evaluación comparativa más precisa entre los modelos analizados. Esta decisión metodológica se alinea con estudios recientes que priorizan la claridad interpretativa en tareas de análisis de sentimiento ideológico, particularmente en el contexto de redes sociales políticas polarizadas.

Por otro lado, el proyecto incorpora un segundo conjunto de datos para la aplicación práctica del modelo "*Zero-Shot*". Este corpus fue recolectado en tiempo real durante la campaña electoral para las elecciones europeas de 2024, utilizando técnicas de "*web scraping*" a través de la plataforma "*Web Data Research Assistant*", compatible con la API de la red social X (anteriormente Twitter). Los mensajes fueron capturados entre el 24 de mayo y el 10 de junio de 2024, coincidiendo con el periodo oficial de campaña, Los mensajes provienen exclusivamente de cuentas verificadas de partidos políticos con representación parlamentaria, así como sus principales líderes.

Se aplicaron criterios estrictos de selección y limpieza: se excluyeron duplicados, se filtraron solo publicaciones originales (sin retweets ni respuestas), y se normalizó el contenido textual. El resultado final fue un conjunto curado de 2,218 mensajes únicos, que sirvió como base para la evaluación del modelo "*Zero-Shot*" en un contexto real y actual. Este segundo corpus aporta una dimensión aplicada al estudio, permitiendo contrastar el rendimiento observado en el entorno experimental con los desafíos propios de un escenario electoral real.

Capítulo 3. Objetivos

El objetivo de este proyecto es analizar y comparar el rendimiento de diferentes modelos de aprendizaje automático y profundo para la clasificación ideológica de mensajes en "*Twitter*". Para ello, el trabajo se articula en dos enfoques complementarios:

- Un enfoque experimental, que utiliza un conjunto de datos públicos (Kaggle) para entrenar y evaluar modelos supervisados en un entorno controlado.

- Un caso práctico aplicado, en el que se implementa un modelo de tipo "*Zero-Shot*" sobre un corpus original de mensajes políticos recolectados en tiempo real durante la campaña electoral de 2024 mediante técnicas de "*web scraping*". Para la clasificación de estos textos sin necesidad de entrenamiento supervisado, se utilizó un modelo pre-entrenado accesible a través de la plataforma "*Hugging Face*", ampliamente reconocida en el ámbito del procesamiento del lenguaje natural por ofrecer una colección de modelos de última generación disponibles para inferencia directa.

3.1 Objetivos Específicos

- Realizar la depuración y preprocesamiento del texto de los mensajes recopilados desde "*Twitter*", incluyendo limpieza, normalización, eliminación de ruido y representación vectorial mediante técnicas como TF-IDF y "*Embeddings*".

- Llevar a cabo un análisis exploratorio del corpus, identificando patrones de publicación, frecuencia de mensajes por autor o grupo político, y distribución de contenido por etiquetas ideológicas.

- Implementar y comparar distintos modelos de clasificación binaria incluyendo modelos basados en algoritmos clásicos como SVM, Regresión Logística, "*Naive Bayes*", "*Random Forest*", "*Gradient Boosting*", "*Ridge Classifier*" y "*Passive-Aggressive*" complementado con modelos avanzados como "*Zero-Shot*" con BART, SBERT y BETO para predecir la orientación política de los mensajes.

- Evaluar cuantitativamente el desempeño de los modelos utilizando métricas como precisión, "*Recall*", "*F1-score*" y matriz de confusión, y discutir sus implicaciones metodológicas.

- Aplicar los resultados obtenidos a un caso práctico, centrado en el análisis de polarización ideológica durante la campaña electoral de 2024, utilizando un corpus recolectado mediante técnicas de "*web scraping*".

Capítulo 4. Metodología

Para este estudio se optó por un enfoque de clasificación binaria, considerando únicamente dos orientaciones ideológicas: izquierda y derecha. Esta decisión responde a la necesidad de reducir la complejidad del problema y mantener un mayor control analítico sobre el comportamiento de los modelos. Al limitar la tarea a una clasificación binaria, se facilita tanto la interpretación de los resultados como la comparación directa entre distintos algoritmos de aprendizaje automático.

Con este objetivo, se seleccionaron diez enfoques de modelado diferentes que representan una diversidad metodológica entre técnicas clásicas y modelos recientes basados en arquitecturas Transformer. Los modelos implementados incluyen "*Support Vector Machine*" (SVM) (Cunha et al.,2025), Regresión Logística (Zhan, 2025), "*Naive Bayes*" (Jannah & Kusnawi, 2024), "*Random Forest*" (Bahrawi, 2019), "*Gradient Boosting*" (Moore et al., 2022), "*Ridge Classifier*" (Olivares Lopez et al., 2024) y "*Passive-Aggressive*" (Wang & Li, 2024 , todos combinados con representación TF-IDF, así como métodos modernos como BETO (BERT en español) (López Condori & Gonzales Saji, 2021), embeddings multilingües con SBERT (Deode et al., 2023) y clasificación "*Zero-Shot*" con BART (facebook/bart-large-mnli) (Clarke et al., 2023). Todos los modelos fueron evaluados bajo condiciones experimentales

equivalentes, utilizando representaciones vectoriales acordes a cada enfoque y un conjunto de datos equilibrado, lo que garantiza una base justa para la comparación de resultados.

4.1 SVM (Support Vector Machine) con TF-IDF

El modelo de "*Support Vector Machine*" (SVM) es uno de los algoritmos más reconocidos en el campo del aprendizaje supervisado, particularmente eficaz en tareas de clasificación de texto. Este modelo se fundamenta en la búsqueda de un hiperplano que separe las clases de manera óptima, maximizando la distancia entre los datos más cercanos de cada clase, conocidos como vectores de soporte. Cunha et al. (2025) destacan que, a pesar del auge de los modelos de lenguaje profundo, SVM sigue siendo competitivo en aplicaciones donde los recursos computacionales son limitados o donde se requiere una solución eficiente sin necesidad de entrenamiento intensivo.

Una de las características que potencian el rendimiento de SVM en clasificación de texto es su combinación con técnicas de extracción de características como el TF-IDF ("*Term Frequency - Inverse Document Frequency"*). Este método pondera la importancia de las palabras no solo por su frecuencia en el documento, sino también considerando su relevancia en el conjunto total de documentos. Como explican Al-Habib et al. (2023), esta técnica permite reducir el ruido textual y enfocar la clasificación en los términos que realmente diferencian una clase de otra. Al utilizar TF-IDF como paso previo al entrenamiento de SVM, se obtiene un modelo capaz de manejar de forma eficaz grandes volúmenes de datos textuales y extraer patrones informativos con alta precisión.

Desde un punto de vista funcional, el SVM con TF-IDF es particularmente útil en contextos donde se requiere una categorización precisa de textos en múltiples clases. Por ejemplo, en el trabajo de Vinod et al. (2023), se utilizó SVM para clasificar artículos científicos en diferentes áreas temáticas, logrando resultados competitivos en comparación con modelos más complejos. Esta capacidad de adaptación a tareas multiclase lo convierte en una opción robusta para escenarios como el análisis de sentimiento, la detección de spam o la clasificación temática de documentos.

Entre sus principales ventajas, SVM ofrece un alto rendimiento en términos de precisión, especialmente cuando se cuenta con un conjunto de datos limpio y bien representado mediante vectores TF-IDF. Además, su versatilidad permite utilizar diferentes funciones núcleo ("*Kernels*"), lo que facilita la adaptación a problemas lineales o no lineales. Según Al-Habib et al. (2023), la selección adecuada del "*Kernel*" lineal, polinomial o radial puede optimizar la separación de clases incluso cuando los datos no son perfectamente separables en el espacio original.

No obstante, el modelo también presenta limitaciones. Una de las principales es su sensibilidad ante conjuntos de datos desbalanceados, donde la clase mayoritaria puede dominar el proceso de aprendizaje. Este problema ha sido abordado en estudios recientes mediante técnicas de re-muestreo o asignación de pesos diferenciados, como fue aplicado por Al-Habib y su equipo (2023) para mejorar la clasificación de artículos científicos con categorías

poco representadas. Otro reto relevante es la interpretación del modelo: a diferencia de los árboles de decisión o modelos lineales simples, SVM no ofrece una representación intuitiva de las decisiones que toma, lo que puede dificultar su uso en contextos donde la explicabilidad es crítica.

Desde una perspectiva práctica, el uso de SVM con TF-IDF sigue siendo una opción válida y eficaz en múltiples dominios. Cunha et al. (2025) concluyen que, si bien los modelos de lenguaje grande (LLMS) superan a los enfoques tradicionales en tareas específicas, su costo computacional los hace menos viables en entornos donde la eficiencia es prioritaria. En este contexto, SVM ofrece un equilibrio entre rendimiento, simplicidad y velocidad de entrenamiento que lo mantiene vigente como herramienta de referencia en clasificación automática de texto.

En síntesis, la combinación de SVM con TF-IDF representa una solución sólida para tareas de procesamiento de lenguaje natural. Su eficiencia, adaptabilidad y buen rendimiento frente a datos estructurados correctamente lo consolidan como una alternativa competitiva frente a modelos más sofisticados, especialmente cuando se prioriza la eficiencia y la claridad metodológica en proyectos de clasificación textual.

4.2 Regresión Logística con representación TF-IDF

La regresión logística es un algoritmo de aprendizaje supervisado ampliamente utilizado para tareas de clasificación binaria, como el análisis de sentimiento. A pesar de su simplicidad, se ha mantenido como una herramienta eficaz y confiable en el procesamiento de texto, especialmente cuando se combina con métodos de representación como TF-IDF ("*Term Frequency-Inverse Document Frequency*"). Esta técnica transforma el texto en una matriz numérica que pondera la relevancia de las palabras según su frecuencia relativa en un documento y su escasez en el corpus total, permitiendo capturar información discriminativa esencial para la clasificación (Zhan, 2025).

En palabras de Wardana et al. (2024), la combinación de regresión logística con TF-IDF resulta especialmente útil para tareas en las que se requiere una alta eficiencia computacional sin sacrificar precisión. En su estudio con reseñas de usuarios en "*LinkedIn*", los autores demostraron que esta configuración puede alcanzar métricas sobresalientes, como un F1-Score superior al 93%, con tiempos de entrenamiento significativamente menores que modelos más complejos. Esta ventaja hace que la regresión logística sea ideal para aplicaciones donde se procesan grandes volúmenes de texto con recursos limitados.

El funcionamiento de este algoritmo se basa en la estimación de probabilidades mediante la función sigmoide, que convierte la salida de una combinación lineal de características (en este caso, las ponderaciones TF-IDF) en un valor entre 0 y 1. Esta probabilidad se compara contra un umbral (típicamente 0.5) para determinar la clase final. Como explica Hagi y Rarasati (2024), esta metodología permite clasificar textos de manera robusta y reproducible, facilitando su aplicación práctica en sistemas de análisis automatizado de reseñas o comentarios.

Entre sus principales ventajas, destaca su capacidad para entregar resultados interpretables La regresión logística permite conocer la contribución específica de cada palabra representada mediante TF-IDF al resultado final, algo valorado en entornos donde la explicabilidad es importante, como el análisis de opinión pública o la toma de decisiones corporativas (Popoola et al., 2024).

Además, el modelo se adapta bien a situaciones donde los datos están moderadamente balanceados y se han aplicado técnicas adecuadas de preprocesamiento, como normalización, eliminación de ruido textual y selección de características. Wardana et al. (2024) también señalan que, al incorporar técnicas complementarias como "*FastText*" para expandir el vocabulario semántico, la regresión logística puede aumentar su capacidad de generalización sin volverse excesivamente compleja.

Sin embargo, este enfoque no está exento de limitaciones. Una de las principales debilidades es su rendimiento decreciente cuando los datos presentan una alta no linealidad o relaciones complejas entre características, algo que ocurre frecuentemente en textos con ambigüedades semánticas o ironía. Zhan (2025) advierte también sobre el riesgo de sobreajuste al usar TF-IDF con "*datasets*" muy específicos, pues el modelo tiende a memorizar términos dominantes del conjunto de entrenamiento, perdiendo capacidad de generalización en los datos de prueba.

Otro reto habitual es la gestión del desbalanceo en los conjuntos de datos. Hagi y Rarasati (2024) encontraron que la proporción desigual entre clases (por ejemplo, opiniones mayoritariamente negativas) puede sesgar la predicción del modelo hacia la clase dominante. Para mitigar este problema, se recomienda aplicar técnicas como la reponderación de clases o el sobremuestreo de la clase minoritaria.

En conclusión, la regresión logística combinada con representación TF-IDF continúa siendo una alternativa sólida en el análisis de sentimiento textual. Su equilibrio entre simplicidad, eficiencia y capacidad interpretativa la convierte en una herramienta particularmente útil para proyectos con restricciones computacionales o que requieran transparencia en los resultados. No obstante, su uso óptimo exige una cuidadosa preparación de los datos y una evaluación continua del rendimiento ante nuevos contextos.

4.3 Naive Bayes

El algoritmo "*Naive Bayes*" es uno de los métodos de clasificación probabilística más utilizados en tareas de análisis de sentimiento, debido a su simplicidad, rapidez y capacidad para manejar grandes volúmenes de datos textuales. Se basa en el teorema de Bayes y asume la independencia condicional entre las características del texto, lo cual, aunque simplista, ha demostrado ser eficaz en una amplia variedad de dominios (Jannah & Kusnawi, 2024).

Este modelo calcula la probabilidad de que una determinada muestra pertenezca a una clase dada, en función de las características observadas. Por ejemplo, en el contexto del análisis de sentimiento, estima la probabilidad de que un mensaje pertenezca a una categoría positiva o

negativa, basándose en la frecuencia de aparición de determinadas palabras. Esta aproximación ha sido especialmente útil en entornos como "*Twitter*", donde los textos son breves y directos, lo que reduce el impacto de la dependencia entre palabras (Mantika et al., 2024).

Una de las principales funciones del algoritmo es su capacidad de clasificación eficiente, incluso con conjuntos de datos relativamente grandes. Amini y Setiawan (2024), por ejemplo, aplicaron Naive Bayes para analizar tuits relacionados con la candidatura de Gibran Rakabuming en Indonesia y obtuvieron una precisión del 82.19%, lo cual destaca su capacidad para identificar patrones en contextos altamente polarizados. En este estudio, el modelo fue alimentado con más de 3.000 tuits y mostró buenos resultados incluso en un escenario con desequilibrio entre clases.

Entre las ventajas más destacadas de "*Naive Bayes*" se encuentra su rapidez de entrenamiento y predicción, lo que lo convierte en una excelente opción cuando se requiere velocidad y eficiencia computacional. Asimismo, no necesita un gran volumen de datos para comenzar a ofrecer resultados aceptables, lo que lo hace idóneo para tareas de prototipado o escenarios donde el etiquetado manual es costoso o limitado (Amini & Setiawan, 2024). Además, su interpretabilidad es superior a la de otros modelos más complejos, ya que las predicciones pueden explicarse directamente a partir de las probabilidades aprendidas.

Sin embargo, el modelo también presenta limitaciones importantes. La más conocida es su suposición de independencia entre características, que rara vez se cumple en el lenguaje natural, donde las palabras suelen estar semánticamente relacionadas. Esta limitación puede afectar su rendimiento cuando se trata de interpretar estructuras lingüísticas complejas o expresiones idiomáticas. Otro desafío es su sensibilidad a la representación de los datos, en muchos casos, el uso de técnicas como TF-IDF mejora considerablemente su precisión, al atenuar el efecto de palabras frecuentes poco informativas (Jannah & Kusnawi, 2024).

Además "*Naive Bayes*" tiende a tener dificultades para manejar clases con alta superposición léxica, como se ha observado en análisis de sentimientos sobre productos con valoraciones ambiguas. En estos casos, su rendimiento puede verse superado por modelos más sofisticados como SVM o "*Random Forest*", aunque con un costo computacional mayor (Mantika et al., 2024).

A pesar de sus limitaciones, este modelo sigue siendo una herramienta fundamental en tareas de clasificación de texto. Su eficacia ha sido validada en numerosos estudios, desde análisis políticos hasta reseñas de productos y opiniones sobre servicios públicos. En particular, en el estudio de Amini y Setiawan (2024), la combinación de "*Naive Bayes*" con técnicas de preprocesamiento y vectorización como TF-IDF permitió capturar de manera efectiva la polaridad de los mensajes en redes sociales, confirmando su vigencia como modelo de referencia para tareas de análisis de sentimiento automatizado.

4.4 Random Forest

"*Random Forest*" es un algoritmo de aprendizaje supervisado basado en el ensamblado de árboles de decisión. Su principio central consiste en construir múltiples árboles durante el entrenamiento y devolver la clase que recibe la mayoría de los votos entre ellos. Este enfoque robusto y versátil ha demostrado ser altamente efectivo en tareas de clasificación, incluyendo el análisis de sentimiento. Como explica Bahrawi (2019), este modelo sobresale por su capacidad para manejar grandes volúmenes de datos con múltiples variables, reduciendo el riesgo de sobreajuste gracias a su mecanismo de agregación por votación.

En el contexto del análisis de sentimiento ha sido aplicado con éxito en la clasificación de opiniones expresadas en redes sociales, reseñas de productos y textos científicos. Ahmed Khan et al. (2024) utilizaron este modelo para clasificar opiniones políticas en redes sociales, obteniendo métricas competitivas frente a algoritmos como SVM. Según sus resultados, este alcanzó una precisión del 78.7% y un F1-score de 78.5%, lo que evidencia su capacidad para manejar textos con estructuras lingüísticas informales, típicas de plataformas como "*Twitter*".

Una de las principales funciones de "*Random Forest*" es su capacidad para realizar clasificación multiclase y binaria sin requerir ajustes complejos. Cada árbol de decisión dentro del conjunto es entrenado con una muestra aleatoria del conjunto de datos y selecciona subconjuntos aleatorios de características, lo que introduce diversidad en el aprendizaje. Esta propiedad mejora la generalización del modelo y lo hace menos sensible a ruidos o valores atípicos en los datos, como destacan Sanchez-Medina (2024) en su estudio sobre textos científicos generados por humanos y LLMS.

Entre sus ventajas ofrece una notable resistencia al sobreajuste en comparación con un árbol de decisión individual. Además, su estructura permite estimar la importancia relativa de cada variable, lo cual es útil en análisis de texto para identificar qué palabras o n-gramas tienen mayor peso en la predicción del sentimiento. Bahrawi (2019) señala también que su rendimiento es estable incluso con conjuntos de datos parcialmente incompletos o desbalanceados, lo que lo convierte en una opción sólida para entornos donde la calidad de los datos puede variar.

No obstante, este algoritmo no está exento de desventajas. Uno de los principales retos es su complejidad computacional, al generar cientos o miles de árboles, su entrenamiento puede volverse lento y demandante en términos de memoria. Ahmed Khan et al. (2024) advierten que esta limitación puede ser relevante en aplicaciones en tiempo real o cuando se trabaja con dispositivos de bajo rendimiento. Además, sigue siendo menos transparente que algoritmos lineales como la regresión logística, lo que puede dificultar su uso en contextos que requieren explicabilidad total.

Otra desventaja potencial es que su desempeño puede verse afectado si los datos contienen muchas variables irrelevantes. En estos casos, la inclusión de demasiadas características sin una etapa previa de selección puede introducir ruido y reducir la eficiencia del modelo. Por ello, es habitual combinarlo con técnicas de preprocesamiento y reducción de

dimensionalidad, como la representación TF-IDF, que mejora la calidad de los vectores de entrada (Sanchez-Medina, 2024).

En conclusión, "*Random Forest*" constituye una herramienta poderosa y flexible para el análisis automático de sentimientos. Su capacidad para manejar datos textuales de alta dimensión, su tolerancia al ruido y su rendimiento estable lo posicionan como una opción viable para tareas de clasificación ideológica en redes sociales. Aunque presenta desafíos en términos de coste computacional e interpretabilidad, su uso sigue siendo recomendado cuando se busca un equilibrio entre precisión, robustez y adaptabilidad.

4.5 Gradient Boosting

"*Gradient Boosting*" es una técnica de aprendizaje de conjunto que ha ganado gran popularidad en tareas de clasificación y regresión, especialmente en contextos donde se requiere un alto grado de precisión. Su fundamento se basa en construir modelos secuenciales, donde cada nuevo modelo intenta corregir los errores cometidos por los anteriores. Esta capacidad de aprendizaje iterativo permite que el algoritmo se adapte progresivamente a los patrones subyacentes en los datos, haciéndolo especialmente eficaz en problemas complejos como el análisis de sentimiento (Moore et al., 2022).

En el análisis de sentimiento, ha demostrado ser altamente competitivo frente a otros modelos clásicos. Por ejemplo, Ramavath, Subash y Kadainti (2025) aplicaron un enfoque combinado con selección de características para mejorar la clasificación de sentimientos en textos. Este método no solo optimizó la precisión del modelo, sino que también redujo el ruido de datos eliminando características irrelevantes, lo cual es crucial en tareas de procesamiento de lenguaje natural donde los textos pueden ser altamente redundantes.

Una de las principales funciones del modelo es su habilidad para manejar relaciones no lineales entre características y clases objetivo. Gracias a su estructura basada en árboles, "*Gradient Boosting*" puede capturar interacciones complejas entre palabras y contextos lingüísticos, lo que es fundamental para entender matices emocionales en el lenguaje humano. Arslan et al. (2024) demostraron que, al combinar técnicas de extracción profunda de características visuales con este modelo, se lograban mejoras sustanciales en la clasificación de imágenes con contenido emocional, validando su eficacia tanto en modalidades textuales como visuales.

Entre sus ventajas más destacadas, se encuentra su capacidad para lograr altos niveles de precisión y F1-score, incluso con conjuntos de datos desequilibrados o ruidosos. Moore et al. (2022) reportaron un rendimiento constante del 82% en precisión, "*Recall*" y F1-score al aplicar el modelo sobre reseñas de productos de Amazon, superando modelos como "*Naive Bayes*" o Regresión Logística recurrentes en ese contexto. Asimismo, el modelo permite un ajuste fino de hiperparámetros, lo cual es clave para adaptar el algoritmo a distintos dominios y lenguajes.

No obstante, este modelo también presenta desventajas relevantes. Una de ellas es su alto coste computacional, especialmente cuando se utilizan muchas iteraciones o árboles profundos. Esto puede dificultar su aplicación en tiempo real o en dispositivos con recursos limitados. Además, su interpretabilidad es limitada en comparación con modelos más simples como la regresión logística, lo que representa un reto en contextos donde la transparencia es crítica (Ramavath et al., 2025).

Otro reto importante es la tendencia al sobreajuste si no se realiza un control adecuado del aprendizaje. Debido a su alta capacidad de modelado, "*Gradient Boosting*" puede memorizar patrones espurios en los datos de entrenamiento, especialmente si estos no han sido depurados correctamente. Por ello, es común utilizar técnicas como validación cruzada y regularización para evitar este problema (Arslan et al., 2024).

En conclusión, "*Gradient Boosting*" es una herramienta poderosa y flexible para el análisis de sentimientos. Su capacidad para modelar relaciones complejas, combinada con técnicas de selección de características y ajustes finos, lo convierte en una de las opciones más robustas para tareas de clasificación emocional. Aunque enfrenta desafíos en cuanto a coste computacional y explicabilidad, sus resultados empíricos en múltiples estudios respaldan su utilidad para aplicaciones reales donde la precisión es prioritaria.

4.6 Ridge Classifier

El "*Ridge Classifier*", derivado de la regresión de "*Ridge*", es un modelo lineal que incorpora un término de regularización para evitar el sobreajuste. A diferencia de la regresión lineal tradicional, este enfoque penaliza los coeficientes grandes, lo que conduce a soluciones más estables y generalizables, especialmente útiles en contextos de alta dimensionalidad como el análisis de sentimientos. En palabras de Olivares López et al. (2024), este tipo de regularización ayuda a controlar la complejidad del modelo, mejorando su capacidad de generalización frente a nuevos datos.

En su funcionamiento, el modelo transforma el problema de clasificación en uno de regresión multiclase, asignando etiquetas numéricas a cada categoría y estimando un conjunto de funciones lineales, una por clase. La clase final se determina por la función que produce el mayor valor. Este enfoque resulta particularmente eficaz cuando se utiliza junto con representaciones dispersas de texto, como TF-IDF, tal como se evidenció en los experimentos realizados con datasets extraídos de "*Twitter*" (Suryawanshi, 2024).

Una de las ventajas más destacadas de este modelo es su eficiencia computacional. A diferencia de otros modelos como los "*Random Forest*" o "*Gradient Boosting*", este algoritmo requiere menos recursos de entrenamiento, lo que lo convierte en una opción ideal para entornos con limitaciones computacionales. Además, su naturaleza lineal facilita la interpretación de los coeficientes asociados a cada término del texto, permitiendo identificar cuáles palabras contribuyen más a una determinada predicción.

Sin embargo, esta simplicidad también conlleva limitaciones. Su estructura lineal lo hace menos efectivo frente a relaciones no lineales complejas entre variables. A diferencia de métodos como "*Random Forest*" o SVM, este puede no capturar interacciones sutiles entre términos que son frecuentes en el lenguaje natural, particularmente en textos informales como los de redes sociales. Esto fue evidenciado en los experimentos realizados por Olivares López et al. (2024), donde a pesar de obtener buenos resultados generales, se observó una mayor dificultad en la clasificación de tweets con tono neutral.

En cuanto a sus aplicaciones, este modelo ha demostrado un rendimiento sólido en tareas de análisis de sentimiento multiclase. En un estudio reciente, se logró una precisión del 82.5% y un F1-score equivalente, superando a otros modelos como SVM y "*Random Forest*" bajo las mismas condiciones experimentales (Olivares López et al., 2024). Esta capacidad para mantener un balance entre precisión y "*Recall*" lo hace adecuado para análisis donde es importante minimizar tanto los falsos positivos como los falsos negativos.

En contextos reales ha sido empleado en la clasificación de opiniones sobre inteligencia artificial en "*Twitter*", donde logró identificar correctamente la orientación sentimental de publicaciones relacionadas con términos como "*Machine Learning*", "AI" o "*Data*". No obstante, uno de los desafíos encontrados fue que estas palabras, aunque neutrales en muchos contextos, fueron frecuentemente clasificadas como negativas debido a su co-ocurrencia con opiniones críticas hacia la tecnología, lo cual resalta la importancia del contexto semántico en estos modelos (Suryawanshi, 2024).

En conclusión, el "*Ridge Classifier*" ofrece una alternativa eficaz y computacionalmente eficiente para el análisis de sentimiento, especialmente en tareas donde la interpretabilidad y la rapidez son factores clave. Aunque su desempeño puede verse limitado por su linealidad, su capacidad de generalización y facilidad de implementación lo convierten en una herramienta valiosa dentro del repertorio de técnicas de aprendizaje automático aplicadas al procesamiento del lenguaje natural.

4.7 Passive-Aggressive

El algoritmo "*Passive-Aggressive Classifier*" (PAC) representa una alternativa eficiente para tareas de clasificación en tiempo real y a gran escala, particularmente en el ámbito del análisis de sentimiento. Este modelo se enmarca dentro de los algoritmos de aprendizaje online, caracterizándose por su capacidad de actualizarse continuamente con nuevas muestras de datos sin necesidad de reentrenar desde cero. Tal como señalan Wang y Li (2024), esta propiedad lo convierte en una opción idónea para escenarios dinámicos como las redes sociales, donde los datos se generan de forma constante y en grandes volúmenes.

El funcionamiento del PAC se basa en dos comportamientos: "pasivo" cuando clasifica correctamente una instancia, y "agresivo" cuando se produce un error de clasificación. En este último caso, el modelo ajusta sus parámetros inmediatamente, lo que le permite adaptarse rápidamente a nuevas tendencias o patrones en los datos (Wang & Li, 2024).

Una de sus principales ventajas es su eficiencia computacional. A diferencia de algoritmos más complejos como modelos basados en árboles, el PAC requiere un uso reducido de memoria y ofrece tiempos de entrenamiento significativamente más rápidos. Esto lo hace especialmente útil para aplicaciones en dispositivos con recursos limitados o en sistemas que requieren respuestas inmediatas, como los "*Chatbots*" o los monitores de opinión en tiempo real (Wang & Li, 2024).

Otra fortaleza del modelo radica en su robustez frente a datos ruidosos o no balanceados. Como se documenta en el trabajo de Vladić et al. (2024), el PAC mantuvo un rendimiento competitivo incluso frente a conjuntos de datos de Twitter con alto desequilibrio entre clases, logrando una precisión aceptable sin requerir técnicas complejas de balanceo. Además, el uso conjunto con representaciones vectoriales como TF-IDF refuerza su capacidad de capturar la relevancia semántica de los términos en un corpus textual.

Sin embargo, esta simplicidad también conlleva limitaciones. Por ejemplo, su capacidad de modelar relaciones no lineales o interacciones complejas entre características es limitada. Esto puede perjudicar su rendimiento frente a textos con alta ambigüedad semántica o donde el significado depende fuertemente del contexto, como ocurre con el sarcasmo o la ironía frecuente en redes sociales. En el estudio de Wang y Li (2024), aunque el modelo mostró buena precisión con sentimientos negativos, presentó dificultades al clasificar correctamente los neutros, evidenciando este tipo de limitaciones.

Además, el PAC puede ser sensible al orden de los datos debido a su naturaleza secuencial. Esto implica que la secuencia en la que se le presentan los ejemplos de entrenamiento puede influir en su rendimiento, lo cual es relevante en entornos donde los datos no son distribuidos aleatoriamente.

En términos de rendimiento, Vladić et al. (2024) reportaron que el PAC alcanzó una precisión de 87% en el conjunto de entrenamiento y 74% en pruebas con un dataset de "*Twitter*", lo que evidencia su capacidad de generalización con ajustes adecuados. No obstante, también resaltaron que, en comparación con modelos como SVM o regresión logística, su precisión global puede ser ligeramente inferior si no se ajustan correctamente los hiperparámetros.

En conclusión, el "*Passive-Aggressive Classifier*" es un modelo eficaz, veloz y altamente adaptable para tareas de análisis de sentimiento, especialmente cuando se trabaja con datos en flujo o en tiempo real. Aunque presenta desafíos como su limitada capacidad de modelado no lineal y sensibilidad al orden de entrada, su balance entre rendimiento y eficiencia lo convierte en una herramienta útil para aplicaciones prácticas en contextos sociales y empresariales.

4.8 Clasificación Zero-Shot con BART (facebook/bart-large-mnli)

La clasificación de texto "*Zero-Shot*" con modelos como facebook/bart-large-mnli representa un avance significativo en el procesamiento del lenguaje natural, al permitir asignar etiquetas a textos sin requerir datos etiquetados específicos para cada tarea. En esencia, el enfoque

"*Zero-Shot*" redefine el paradigma tradicional del aprendizaje supervisado, al utilizar modelos preentrenados con inferencia textual para realizar tareas de clasificación directamente, sin una fase explícita de ajuste con datos específicos. Como destacan Clarke et al. (2023), este tipo de clasificación resulta especialmente valioso en contextos donde los esquemas de etiquetado evolucionan constantemente, como en redes sociales o sistemas de atención al cliente.

BART ("*Bidirectional and Auto-Regressive Transformers*") es un modelo de lenguaje preentrenado que combina técnicas de codificación y decodificación. En su variante entrenada con MNLI ("*Multi-Genre Natural Language Inference*"), se adapta a tareas de inferencia lógica, permitiendo reformular problemas de clasificación como preguntas de inferencia semántica. Es decir, en lugar de entrenar un clasificador para una etiqueta específica, se construye una hipótesis en lenguaje natural ("Este comentario es positivo") y el modelo evalúa si el texto original implica dicha hipótesis. Este enfoque fue refinado por Shu et al. (2022), quienes propusieron convertir la clasificación basada en aspectos en problemas de inferencia textual, logrando generalizar a nuevos dominios sin necesidad de anotaciones manuales.

Una de las principales funciones del modelo es su capacidad de evaluar relaciones semánticas entre texto y etiquetas expresadas como frases. Por ejemplo, en análisis de sentimiento, se compara la oración original con múltiples hipótesis del tipo "El tono de este mensaje es positivo", "El tono de este mensaje es negativo", etc. Clarke y su equipo demostraron que este método, reforzado mediante estrategias de preentrenamiento explícito e implícito, permite mejorar la capacidad del modelo para capturar el contexto de la tarea y adaptar su razonamiento a diferentes dominios
.
Entre las ventajas de esta metodología destaca su flexibilidad. Al no requerir entrenamiento adicional para cada nueva etiqueta, permite escalar fácilmente a nuevos escenarios, idiomas o temas. Además, la dependencia de hipótesis formuladas en lenguaje natural facilita la personalización del análisis y su interpretación por parte de usuarios no técnicos (Tesfagergish et al., 2022). Otro beneficio importante es la reducción de costes en la recolección y anotación de datos, lo cual es especialmente útil en tareas multilingües o de bajo recurso.

No obstante, también enfrenta retos considerables. Uno de los más evidentes es la disminución de precisión en comparación con modelos supervisados entrenados específicamente para una tarea. Clarke et al. (2023) observaron que los modelos "*Zero-Shot*" tienden a degradarse en tareas con clases altamente especializadas o etiquetas semánticamente ambiguas. Además, su rendimiento depende en gran medida de la formulación de las hipótesis: una redacción poco clara o mal adaptada al dominio puede deteriorar significativamente los resultados.

Por otro lado, Tesfagergish et al. (2022) advierten que estos modelos, aunque eficientes en contextos multiclase, presentan dificultades para distinguir emociones matizadas si no se ajusta cuidadosamente el set de hipótesis. Para solventar estas limitaciones, los autores proponen combinaciones híbridas con modelos supervisados en una arquitectura de aprendizaje semisupervisado, lo que mejora tanto la cobertura como la precisión del análisis de sentimientos y emociones.

En conclusión, la clasificación "*Zero-Shot*" con BART representa una alternativa poderosa y versátil frente a los enfoques tradicionales. Si bien aún existen desafíos técnicos y de rendimiento, su aplicabilidad sin necesidad de entrenamiento específico lo posiciona como una herramienta valiosa para contextos dinámicos y multilingües. Su implementación en tareas como análisis de sentimiento, detección de emociones o categorización temática abre nuevas oportunidades para explorar el lenguaje humano con modelos que razonan más allá de sus datos de entrenamiento.

4.9 Clasificación por Similitud con Embeddings Multilingües (SBERT)

La clasificación por similitud semántica con modelos como SBERT (Sentence-BERT) multilingüe se ha consolidado como una alternativa eficaz para tareas de análisis de sentimiento en entornos multilingües. A diferencia de los métodos de clasificación supervisada tradicional, este enfoque se basa en la comparación de representaciones vectoriales de frases, facilitando la identificación de categorías o sentimientos sin necesidad de entrenamiento específico para cada conjunto de datos. Como afirma Deode et al. (2023), SBERT logra representar oraciones completas en un espacio semántico común, permitiendo medir similitudes entre textos con alta precisión incluso en diferentes idiomas.

SBERT es una modificación del modelo BERT original que incorpora una arquitectura tipo siamés, donde dos oraciones se codifican simultáneamente para maximizar o minimizar su similitud según su relación semántica. Esta estructura ha demostrado ser particularmente útil en tareas como recuperación de información, emparejamiento de preguntas y análisis de sentimientos. En su adaptación multilingüe, SBERT permite comparar frases en distintos idiomas generando "*Embeddings*" en un espacio compartido. Esto posibilita, por ejemplo, clasificar un tweet en español comparándolo con ejemplos anotados en inglés, sin necesidad de traducción previa o ajuste adicional del modelo (Deode et al., 2023)

Uno de los principales aportes de este enfoque radica en su utilidad para lenguas con escasos recursos digitales. Mabokela et al. (2023) destacan que los modelos multilingües como SBERT permiten extender análisis sentimentales a idiomas subrepresentados, aprovechando modelos pre-entrenados con múltiples lenguas y evitando así los costosos procesos de anotación manual. Además, su robustez frente a textos con mezcla de lenguas o código alternado lo posiciona como una herramienta valiosa en contextos sociolingüísticos complejos, como los que se encuentran en África o en comunidades bilingües de América Latina.

Desde el punto de vista técnico, una ventaja notable es la eficiencia computacional del modelo una vez pre-entrenado. A diferencia de enfoques como los modelos autoregresivos de generación de texto, SBERT puede realizar inferencias rápidas sin necesidad de ajustes finos, lo que lo hace adecuado para sistemas en producción. Mao et al. (2024) subrayan que los métodos más recientes de "*Embeddings*" multilingües han logrado reducir significativamente el costo computacional de entrenamiento manteniendo un alto rendimiento en tareas multilingües, incluidos los análisis de sentimiento.

Sin embargo, el enfoque no está exento de desafíos. Uno de los más recurrentes es la pérdida de precisión cuando las frases objetivo son muy cortas, ambiguas o dependen fuertemente del contexto cultural. Además, Deode et al. (2023) señalan que, aunque el modelo funciona bien en tareas de similitud, su rendimiento puede degradarse si las clases semánticas no están bien diferenciadas. En otras palabras, la estrategia basada en similitud necesita que las categorías estén claramente representadas mediante ejemplos prototípicos para evitar ambigüedades.

Otro reto relevante es el sesgo lingüístico inherente a los datos de entrenamiento. Muchos modelos multilingües se entrenan principalmente con datos en inglés y en otros idiomas de alta disponibilidad, lo que puede llevar a un rendimiento inferior en lenguas menos representadas. Para mitigar esta limitación, Mao et al. proponen objetivos de entrenamiento alternativos como la reconstrucción a nivel de "*tokens*" y el aprendizaje contrastivo, diseñados para mejorar la alineación semántica entre idiomas sin necesidad de grandes corpus paralelos (Mao et al.,2024).

En resumen, la clasificación por similitud mediante "*Embeddings*" multilingües representa una solución eficiente y flexible para tareas multilingües de análisis de sentimiento. Su capacidad para operar sin entrenamiento específico y su adaptabilidad a distintos idiomas lo convierten en una opción ideal para aplicaciones en entornos heterogéneos. Sin embargo, para maximizar su efectividad, es esencial considerar la calidad de las frases de referencia, el equilibrio entre las lenguas representadas y la sensibilidad cultural de los textos analizados.

4.10 Clasificación BETO (BERT en español)

BETO es una adaptación del modelo BERT específicamente entrenada para el idioma español. A diferencia de otros modelos multilingües, este fue pre-entrenado exclusivamente en grandes corpus en español, como Wikipedia y OPUS, lo que le otorga una mayor capacidad de comprensión semántica del idioma. Este modelo fue desarrollado siguiendo la arquitectura base de BERT, pero con un enfoque particular en la riqueza lingüística y las variantes del español, lo cual representa una ventaja sustancial frente a otros modelos de PLN que no han sido adaptados a lenguas específicas (López Condori & Gonzales Saji, 2021).

El funcionamiento del modelo se basa en el uso de "*Transformers*", una arquitectura que permite procesar texto de forma bidireccional, lo que significa que el modelo tiene en cuenta el contexto de las palabras tanto a la izquierda como a la derecha. Esta característica mejora la precisión en tareas como el análisis de sentimiento, clasificación de texto y extracción de entidades nombradas. En palabras de Miranda et al. (2023), el uso de BERT y sus variantes como BETO ha permitido avances considerables en el análisis de opiniones en redes sociales, particularmente en español.

Una de las principales ventajas es su rendimiento en tareas de clasificación de texto con datos en español. Al estar pre-entrenado en un corpus monolingüe, su comprensión semántica es más profunda que la de modelos multilingües. Por ejemplo, en estudios como el de García-Peñalvo y Vázquez-Ingelmo (2023), se demostró que el "*Fine-tuning*" de BETO ofrecía un

mejor desempeño en términos de "*F1-score*" y exactitud frente a otras alternativas en competiciones de análisis de sentimientos.

Sin embargo, también presenta ciertos retos. Como menciona López Condori (2021), uno de los inconvenientes más relevantes es el alto costo computacional asociado al entrenamiento y "*Fine*-tuning" del modelo, especialmente cuando se trabaja con grandes volúmenes de texto. El proceso requiere hardware especializado y tiempos de cómputo prolongados, lo que limita su accesibilidad para investigadores o instituciones con recursos limitados
.
Otra limitación destacada es la longitud de entrada al igual que otros modelos BERT, tiene una longitud máxima de 512 tokens por entrada, lo que obliga a truncar textos extensos y podría derivar en pérdida de información valiosa durante el preprocesamiento (García-Peñalvo & Vázquez-Ingelmo, 2023). Además, aunque está entrenado en corpus en español, su rendimiento puede variar dependiendo del tipo de texto, el dominio temático o la variante dialectal empleada en los datos de entrada.

Entre las funciones más destacadas se encuentra su uso en clasificación de sentimiento, detección de discurso de odio y análisis de emociones. Como afirman Miranda et al. (2023), su capacidad para extraer representaciones contextuales precisas lo hace ideal para tareas de análisis de polaridad en redes sociales, donde el lenguaje tiende a ser informal, irónico o emocionalmente cargado. En pruebas realizadas por López Condori y Gonzales Saji (2021), BETO alcanzó una precisión promedio del 81% en la clasificación de comentarios en Google Play Store, superando a otros enfoques clásicos como "*Naïve Bayes*" o SVM.

En resumen, BETO representa una solución robusta y altamente especializada para tareas de procesamiento del lenguaje natural en español. Su precisión, combinada con su capacidad para adaptarse a tareas específicas mediante "*Fine-tuning*", lo posiciona como una herramienta de gran valor en proyectos de análisis de sentimiento. No obstante, es fundamental tener en cuenta sus requerimientos computacionales y las limitaciones inherentes a la arquitectura BERT, especialmente al aplicar el modelo en contextos reales y variados.

Capítulo 5 Limpieza de datos

La limpieza de los datos constituyó una etapa crucial para garantizar la validez y la calidad de los resultados obtenidos en los análisis posteriores. A continuación, se describen las principales operaciones realizadas sobre el conjunto de datos:

- Eliminación de registros vacíos, se procedió a eliminar aquellas observaciones cuyo campo de texto (tweet) se encontraba vacío o contenía valores nulos. Este filtrado preliminar aseguró que todas las instancias utilizadas contuvieran información significativa para el análisis textual.

- Normalización del contenido textual, con el objetivo de homogeneizar el formato de los datos y facilitar su tratamiento, se aplicó una función de preprocesamiento que incluyó diversas transformaciones, tales como: conversión de todos los caracteres a minúsculas, eliminación de URLs, menciones (@usuario) y hashtags, supresión de signos de puntuación, símbolos especiales y espacios redundantes, sustitución o eliminación de caracteres no alfabéticos y errores comunes.

- Depuración de duplicados, se identificaron y eliminaron los registros duplicados, basándose en el contenido exacto del texto. Esta depuración evitó la repetición de información que pudiera sesgar los modelos o inflar la representatividad de ciertas expresiones.

- Filtrado por longitud mínima, los tweets demasiado breves fueron descartados, ya que se consideró que mensajes con una longitud muy limitada no aportaban contexto suficiente para una clasificación efectiva ni para el análisis de sentimiento.

- Equilibrado del conjunto de datos, finalmente, se llevó a cabo un muestreo estratificado para asegurar un equilibrio entre las diferentes clases presentes en el conjunto de datos. Esta práctica resulta fundamental para evitar el sesgo de los modelos hacia la clase mayoritaria, especialmente en tareas de clasificación multiclase.

Estas acciones permitieron disponer de un conjunto de datos más limpio, coherente y representativo, sentando las bases para un análisis más robusto y confiable.

Capítulo 6 Análisis exploratorio del corpus

Con el objetivo de obtener una visión general del comportamiento del corpus en términos temporales y políticos, se llevó a cabo un análisis exploratorio centrado en los datos correspondientes al periodo comprendido entre los años 2019 - 2023. Esta selección responde al hecho de que los años anteriores (2013-2018) presentaban un volumen muy reducido de actividad. En contraste, el periodo analizado concentra la mayor parte del corpus con 191,401 tweets publicados, lo que permite captar con mayor claridad las dinámicas recientes en torno a los temas de interés político en la red social.

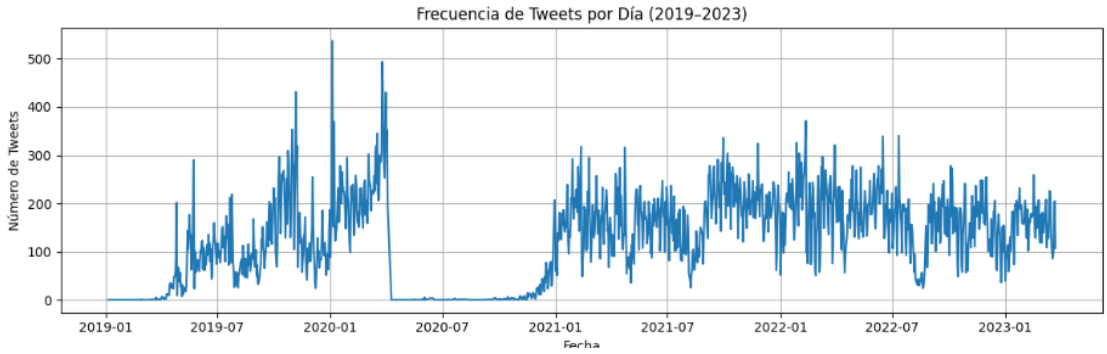

Figura 1 Frecuencia de Tweets por día

La evolución diaria del número de publicaciones muestra un crecimiento significativo a partir del segundo semestre de 2019, con picos de actividad especialmente concentrados durante el primer y segundo trimestre del 2020. Este aumento coincide con el estadillo de la Pandemia COVID-19, un contexto en el que la población permanecía en confinamiento domiciliario, con gran atención a las medidas gubernamentales y elevada actividad en redes sociales. A partir de 2021, se observa una estabilización en el volumen de tweets, manteniéndose en niveles relativamente altos hasta principios de 2023. Esta tendencia sugiere una consolidación del uso de "*Twitter*" como canal de comunicación política durante este periodo.

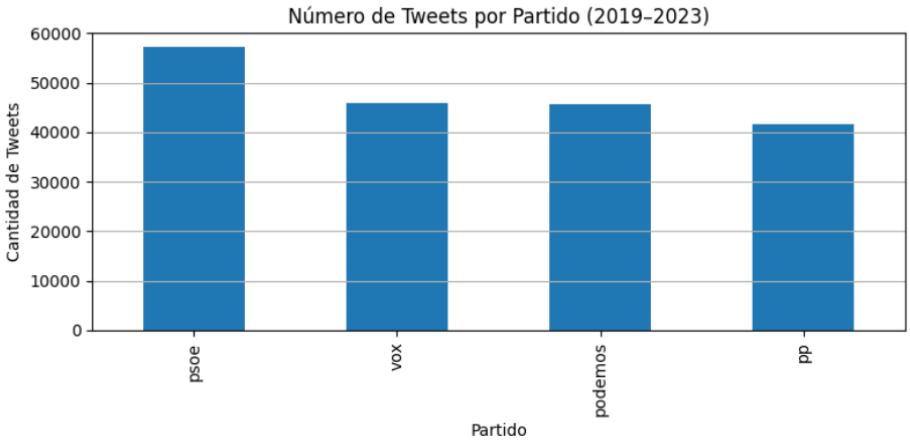

Figura 2 Número de Tweets por partido

El análisis del volumen de publicaciones por partido revela que el PSOE es el grupo más activo en Twitter durante este lapso, seguido por VOX, Podemos y el PP. Esta distribución podría reflejar distintas estrategias de comunicación digital adoptadas por cada partido, así como su nivel de presencia en redes sociales.

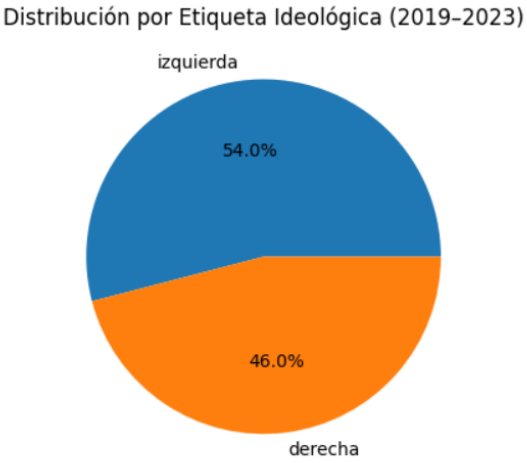

Figura 3 Distribución por etiqueta ideológica

En cuanto a la clasificación ideológica binaria, el corpus muestra una ligera mayoría de tweets asociados a formaciones de izquierda (54%) frente a los de derecha (46%). Esta diferencia,

aunque no extremadamente marcada, es relevante para interpretar el tono general y los posibles sesgos presentes en el conjunto de datos.

Capítulo 7 Modelización

En esta sección se presentan los procesos de entrenamiento, evaluación y comparación de distintos modelos de clasificación aplicados al análisis de ideología política en Twitter. Se ha optado por una estrategia mixta que combina modelos basados en algoritmos clásicos de "*Machine Learning*" con técnicas avanzadas basadas en modelos de lenguaje, a fin de valorar tanto el rendimiento como la viabilidad práctica de cada enfoque. En total, se exploran 10 modelos de clasificación distribuidos en dos grandes grupos:

Modelos basados en algoritmos clásicos:

- Support Vector Machine (SVM)
- Regresión Logística
- Naive Bayes
- Random Forest
- Gradient Boosting
- Ridge Classifier
- Passive-Aggressive

Modelos Avanzados:

- Zero-Shot Classification
- Embeddings + Similitud (SBERT)
- BETO (BERT entrenado en español)

Para asegurar una comparación justa entre modelos y evitar el sesgo inducido por clases mayoritarias, se construyó una muestra balanceada de 5000 tweets (2500 por cada etiqueta ideológica). Esto permite evaluar las métricas de clasificación de forma equitativa, especialmente aquellas como F1-Score y macro avg, que se ven muy afectadas por el desbalance.

- Los modelos basados en algoritmos clásicos fueron entrenados con la base completa de más de 190,000 tweets etiquetados. Posteriormente, se evaluaron sobre la muestra balanceada de 5,000 tweets.

- En los modelos "*Zero-Shot*" y "*Embeddings*", no se requiere una fase de entrenamiento. Por tanto, se aplicaron directamente sobre los 5,000 tweets para medir su rendimiento.

- El modelo BETO (basado en BERT pre-entrenado para español) requiere entrenamiento supervisado. Para ello, se construyó una base especial de 10,000 tweets balanceados (2,500 por partido) que sirvió como conjunto de entrenamiento. Luego, al igual que los demás modelos, fue evaluado sobre la base de 5,000 tweets.

El rendimiento de cada modelo fue medido mediante cuatro indicadores estándar en problemas de clasificación binaria:

- Precision: La precisión se refiere a la proporción de predicciones positivas correctas entre todas las que fueron clasificadas como positivas por el modelo. En el estudio de Khan y Khan (2024), esta métrica se utiliza para evaluar cuán acertadamente el modelo identifica publicaciones que apoyan o rechazan al gobierno, midiendo así su fiabilidad en escenarios sensibles de análisis político automatizado.

- Recall (Sensibilidad o Exhaustividad): La exhaustividad mide la proporción de verdaderos positivos detectados sobre el total real de elementos positivos en los datos. Es particularmente importante cuando se desea garantizar que la mayoría de los casos relevantes, como publicaciones antigubernamentales, sean correctamente identificados. Khan y Khan (2024) destacan esta métrica como clave para la detección efectiva de posturas políticas divergentes.

- F1-Score: Esta métrica es la media armónica entre precisión y exhaustividad. En contextos con desequilibrios en las clases (como ocurre en análisis de opiniones políticas), el F1-score proporciona una visión más balanceada del rendimiento general del modelo. En el experimento de Khan y Khan, esta métrica permitió comparar de forma justa los distintos algoritmos aplicados al conjunto de datos multilingüe.

- Support (Soporte): El soporte indica cuántas instancias de cada clase (por ejemplo, derecha, neutral, izquierda) están presentes en el conjunto de validación. Esta información ayuda a interpretar otras métricas en función de la representación de cada categoría. Aunque es una métrica descriptiva, en el estudio fue crucial para entender la distribución del corpus utilizado y contextualizar los resultados.

- Accuracy (Exactitud): La exactitud refleja el porcentaje de predicciones correctas (tanto positivas como negativas) respecto al total de predicciones realizadas. En su investigación, Khan y Khan (2024) la utilizan como medida general de desempeño, alcanzando un 69% de exactitud en inglés y 94% en urdu, lo que demuestra la eficacia del modelo BERT para la detección de afinidad política en redes sociales.

Estas métricas fueron calculadas tanto por clase (izquierda y derecha), como en promedios globales (macro y ponderado). Además, se utilizaron matrices de confusión para visualizar el rendimiento de clasificación por clase y facilitar la interpretación de los errores.

7.1 Modelos basados en algoritmos clásicos

7.1.1 SVM Support Vector Machine) con TF-IDF

Para este modelo se utilizó la clase "*LinearSVC*" del módulo "*Scikit-Learn*", una implementación optimizada del algoritmo SVM basada en el solucionador "*Liblinear*", que permite resolver el problema de optimización en su forma primal de manera eficiente. Este

clasificador es especialmente adecuado para conjuntos de datos grandes y de alta dimensionalidad, como ocurre en tareas de clasificación de texto con representaciones vectoriales TF-IDF.

A diferencia de otros enfoques SVM que emplean funciones núcleo ("*kernels*") no lineales, "*LinearSVC*" está basado exclusivamente en un "*Kernel*" lineal, lo que reduce la complejidad computacional sin comprometer la capacidad predictiva del modelo. Se exploraron las siguientes combinaciones de hiperparámetros:

-

C: [0.001, 0.01, 0.1, 1, 10, 100]

Se empleó "*GridSearchCV*" con validación cruzada estratificada de 5 particiones. Se utilizaron las métricas "*F1_weighted*" y "*Accuracy*", priorizando la primera como criterio de selección, ya que esta métrica permite evaluar el rendimiento global del modelo teniendo en cuenta el desequilibrio entre clases. El código completo se puede ver en los anexos. La mejor configuración obtenida fue:

C = 0.1

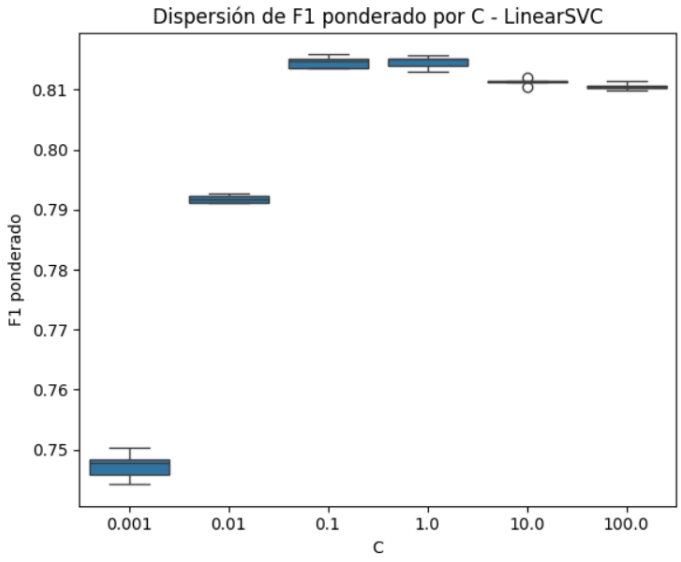

Figura 4 Boxplot SVM

Para evaluar la estabilidad del modelo, se graficaron los "*Boxplots*" de los scores de F1 ponderado por valor de C, mostrando que el modelo es relativamente estable en los distintos valores evaluados. El mejor rendimiento se obtuvo con C = 0.1, aunque C = 1 presentó resultados muy similares. En este contexto, se optó por C = 0.1 al ofrecer una mayor regularización, lo que contribuye a reducir el riesgo de sobreajuste, especialmente en entornos de alta dimensionalidad como los generados por representaciones TF-IDF. Este valor permite al modelo mantener un buen desempeño siendo, a su vez, más robusto frente a ruido o variaciones en datos no vistos.

Finalmente, el modelo se evaluó sobre el conjunto externo compuesto por 5000 tweets. Los resultados fueron los siguientes:

Tabla 1 Resultados SVM

Clase	Precision	Recall	F1- Score	Support
Derecha	0.85	0.83	0.84	2500
Izquierda	0.83	0.85	0.84	2500
Accuracy			0.84	5000
Macro avg	0.84	0.84	0.84	5000
F1_weighted	0.84	0.84	0.84	5000

En términos generales, el modelo alcanzó una "*Accuracy*" del 84%, lo que indica un desempeño sólido en la clasificación ideológica del contenido textual. El análisis por clase revela que el modelo obtuvo una precisión de 0.85 y un "*Recall*" de 0.83 para la clase derecha, lo cual implica una buena capacidad para identificar correctamente los casos verdaderos de esta categoría, aunque con una ligera propensión a confundir instancias de izquierda como derecha. Por su parte, para la clase izquierda, el modelo logró una precisión de 0.83 y un "*Recall*" de 0.85, lo que refleja un comportamiento equilibrado.

El "*F1-score*" general de 0.84, tanto en promedio macro como ponderado, confirma que el modelo mantiene un rendimiento consistente entre ambas clases. Este equilibrio es especialmente relevante dado que se trabajó con un conjunto de prueba balanceado, lo cual permite una evaluación más justa del comportamiento del clasificador.

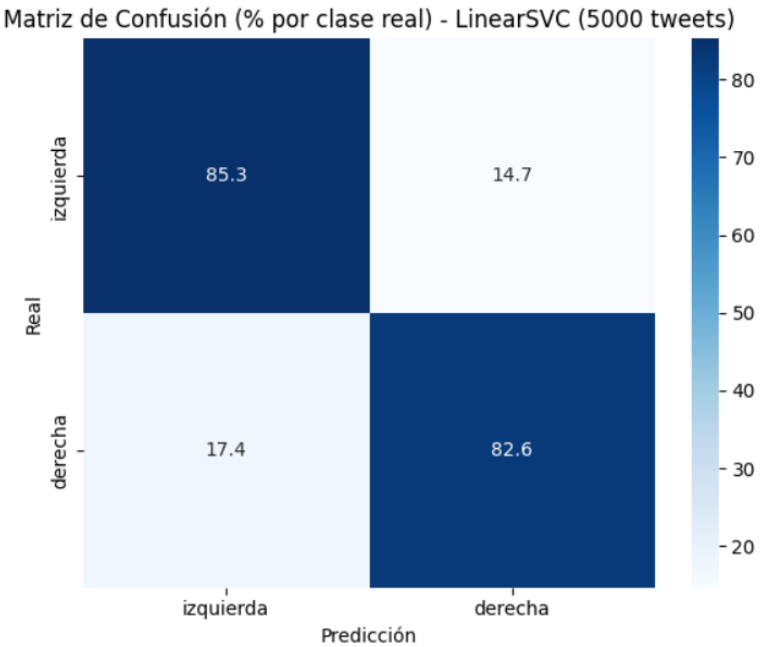

Figura 5 Matriz de confusión SVM

Desde el análisis de la matriz de confusión se puede observar que el modelo clasifica correctamente el 83.6% de los tweets etiquetados como izquierda y el 87.0% de los tweets etiquetados como derecha. Los errores de clasificación se distribuyen en un 16.4% de instancias de izquierda que fueron clasificadas incorrectamente como derecha, y un 13.0% de casos de derecha mal clasificados como izquierda.

Estos resultados confirman que, si bien el modelo presenta un buen desempeño general, tiende ligeramente a subestimar los textos de ideología izquierda, cometiendo más errores al identificarlos en comparación con los textos de derecha. Aun así, las diferencias no son drásticas y reflejan una clasificación razonablemente balanceada.

7.1.2 Regresión Logística con representación TF-IDF

Para el modelo de Regresión Logística se llevó a cabo una búsqueda de hiperparámetros mediante validación cruzada (CV=5), utilizando "*GridSearchCV*". Se exploraron las siguientes combinaciones de parámetros:

C: [0.001, 0.01, 0.1, 1, 10, 100]
"*Class_weight*": [None, '*balanced*']
"*Solver*": '*liblinear*'

Durante el ajuste, se utilizaron dos métricas de evaluación: "*F1_weighted*" y "*Accuracy*", siendo "*F1_weighted*" la seleccionada como métrica principal de optimización. El código completo de entrenamiento y evaluación del modelo se encuentra incluido en los anexos. El mejor conjunto de hiperparámetros obtenido fue:

C = 10
"*Class_weight*" = None
"*Solver*" = liblinear

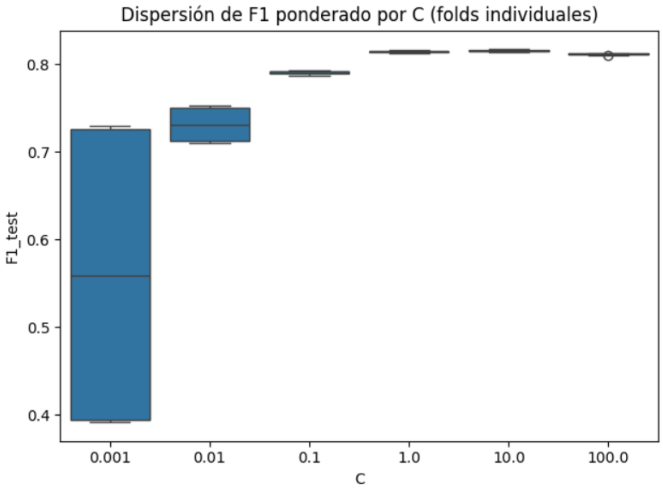

Figura 6 "Boxplots" Regresión Logística

El análisis del "*Boxplot*" individual confirmó la estabilidad del modelo con C=10, al mostrar una baja dispersión entre los índices de validación.

Los resultados mostraron un rendimiento muy similar entre C = 1 y C = 10, siendo este último ligeramente superior en F1 ponderado. Si bien un valor más alto de C implica una regularización menor y potencialmente mayor riesgo de sobreajuste, los resultados fueron estables en todos los "*folds*", como se muestra en la dispersión del "*boxplot*". Además, valores superiores a C = 10 no mejoraron el rendimiento, lo que refuerza la elección de C = 10 como valor óptimo dentro del rango evaluado.

Como se mencionó previamente, el modelo fue evaluado sobre una muestra balanceada de 5000 tweets. Los resultados se detallan a continuación:

Tabla 2 Resultados regresión logística

Clase	Precision	Recall	F1- Score	Support
Derecha	0.86	0.84	0.85	2500
Izquierda	0.85	0.86	0.85	2500
Accuracy			0.85	5000
Macro avg	0.85	0.85	0.85	5000
F1_weighted	0.85	0.85	0.85	5000

El rendimiento global del modelo alcanzó una precisión ("*Accuracy*") del 85%, lo cual refleja un desempeño competitivo en términos generales. A nivel de clases, los tweets de orientación derecha fueron clasificados correctamente con una precisión del 86% y un "*Recall*" del 84%, mientras que los tweets de orientación izquierda obtuvieron valores similares (precisión del 85% y "*Recall*" del 86%). Esto sugiere que el modelo logra un equilibrio adecuado en la clasificación de ambas categorías, sin mostrar un sesgo evidente hacia alguna clase particular. El valor de "*F1-score*" promedio (tanto macro como ponderado) también indica una buena consistencia general en la clasificación.

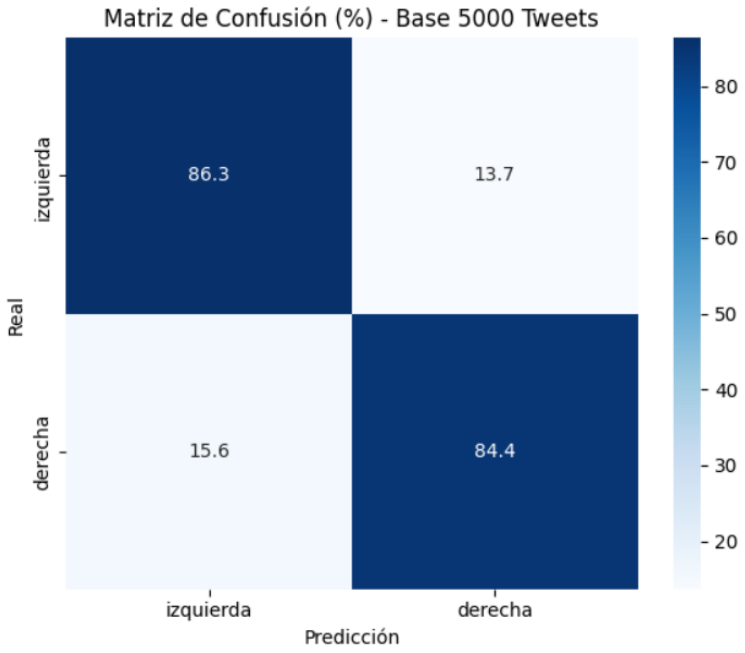

Figura 7 Matriz de Confusión Regresión Logística

En la matriz de confusión, los valores se presentan como porcentajes normalizados por fila, es decir, dentro de cada clase real. Esto permite evaluar qué proporción de ejemplos de cada clase fueron correctamente clasificados o confundidos con la clase opuesta.

La matriz de confusión respalda esta observación: el modelo clasificó correctamente el 86.3% de los tweets etiquetados como izquierda y el 84.4% de los tweets etiquetados como derecha. Los errores se distribuyen en un 13.7% de casos en los que tweets de izquierda fueron mal clasificados como derecha, y un 15.6% de casos en los que tweets de derecha fueron mal clasificados como izquierda.

7.1.3 Naive Bayes

Para el modelo de "*Naive Bayes*", se realizó una búsqueda sobre el hiperparámetro "*Alpha*", que controla el nivel de suavizado de "*Laplace*". La parrilla de valores evaluada fue:

"*Alpha*": [0.001, 0.01, 0.1, 1, 10, 100]

El ajuste se llevó a cabo utilizando "*GridSearchCV*" con validación cruzada (CV=5) y dos métricas de evaluación: "*F1_weighted*" y "*Accuracy*", siendo "*F1_weighted*" la métrica seleccionada como criterio principal. El código completo de entrenamiento y evaluación se encuentra incluido en los anexos. El valor óptimo encontrado fue:

"*Alpha*" = 0.01

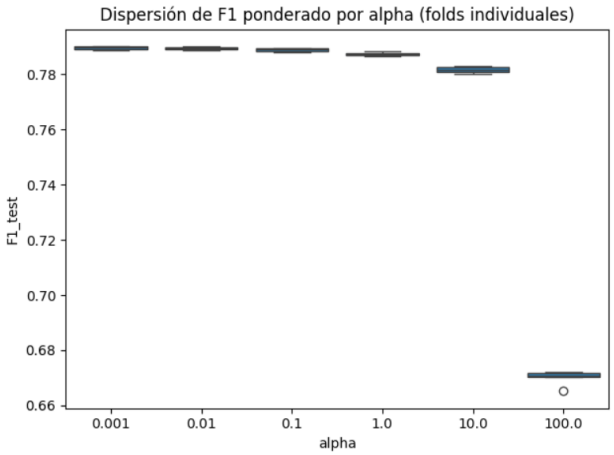

Figura 8 Boxplot "Naive Bayes"

El "*Boxplot*" correspondiente muestra que "*Alpha*" = 0.01 no solo obtuvo el mejor rendimiento promedio, sino también la menor variabilidad entre particiones. Esto respalda la robustez del modelo en distintos subconjuntos de datos.

Aunque valores cercanos como "*Alpha*" = 0.001 fueron considerados, no se observaron mejoras en estabilidad ni en rendimiento, lo que refuerza la elección de "*Alpha*" = 0.01 como valor óptimo dentro del rango analizado.

Tal como se mencionó anteriormente, la evaluación del modelo se realizó sobre una muestra balanceada de 5000 tweets. Los resultados obtenidos se resumen a continuación:

Tabla 3 Resultados Naive Bayes

Clase	Precision	Recall	F1- Score	Support
Derecha	0.82	0.77	0.79	2500
Izquierda	0.79	0.83	0.80	2500
Accuracy			0.80	5000
Macro avg	0.80	0.80	0.80	5000
F1_weighted	0.80	0.80	0.80	5000

El modelo alcanzó una precisión global del 80%, lo cual representa un rendimiento aceptable, aunque inferior al observado en los modelos SVM y de Regresión Logística.

En términos de desempeño por clase, se observa que los tweets etiquetados como "derecha" presentan una precisión del 82% y un "*Recall*" del 77%, mientras que los tweets de "izquierda" logran una precisión del 79% y un "*Recall*" del 83%. Esta diferencia sugiere que el modelo tiene una ligera tendencia a clasificar los mensajes hacia la clase "izquierda", lo cual se ve reforzado por un mayor número de errores al clasificar correctamente los tweets de "derecha".

El "*F1-score*" promedio fue de 0.80, lo que indica un rendimiento estable y razonablemente equilibrado, aunque por debajo del desempeño observado en los otros modelos.

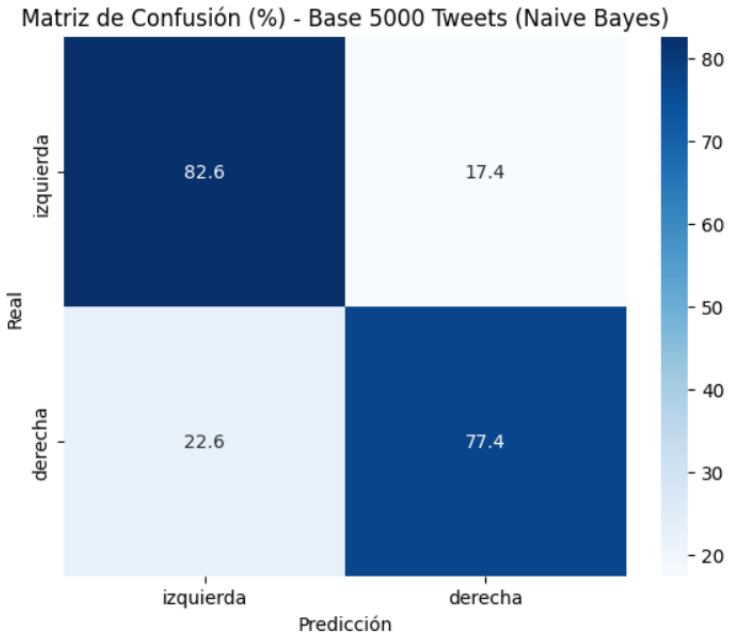

Figura 9 Matriz de Confusión Naive Bayes

La matriz de confusión complementa estas observaciones. El modelo identificó correctamente el 82.6% de los tweets de izquierda, pero solo el 77.4% de los tweets de derecha. Los errores se concentran principalmente en la clase derecha, donde el 22.6% de los mensajes fueron mal clasificados como izquierda, reforzando así la idea de un sesgo leve hacia esta categoría.

7.1.4 Random Forest

Para el modelo "*Random Forest*", se llevó a cabo una búsqueda de hiperparámetros sobre las combinaciones de "*N_estimators*" (número de árboles) y "*Max_depth*" (profundidad máxima del árbol), utilizando "*Class_weight*" = "*Balanced*" para ajustar los pesos de clase en función de su frecuencia. La parrilla de valores evaluada fue:

"*N_estimators*": [100, 200, 500]
"*Max_depth*": [None, 10, 20]
"*Class_weight*" = "*Balanced*"

El ajuste se realizó mediante "*GridSearchCV*" con validación cruzada (CV=5), evaluando dos métricas: "*F1_weighted*" y "*Accuracy*", siendo la primera utilizada como criterio principal para la selección del modelo óptimo. El código completo de entrenamiento y evaluación se encuentra incluido en los anexos. El mejor conjunto de hiperparámetros fue:

"*N_estimators*" = 500
"*Max_depth*" = "*None*"
"*Class_weight*" = "*Balanced*"

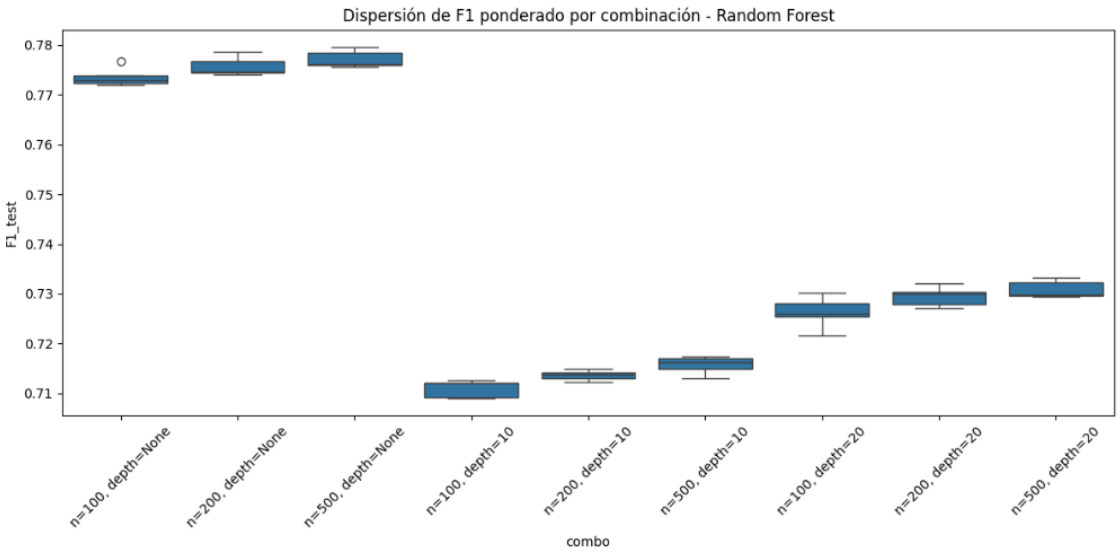

Figura 10 Boxplot "Random Forest"

El "*Boxplot*" por combinación mostró que el mejor rendimiento se obtuvo con la configuración que no limitó la profundidad máxima ("*Max_depth*" = "*None*"), lo que sugiere que permitir a los árboles crecer libremente contribuye a capturar mejor la complejidad de los datos. En particular, esta combinación presentó el mayor valor medio de F1 ponderado y una baja dispersión entre pliegues.

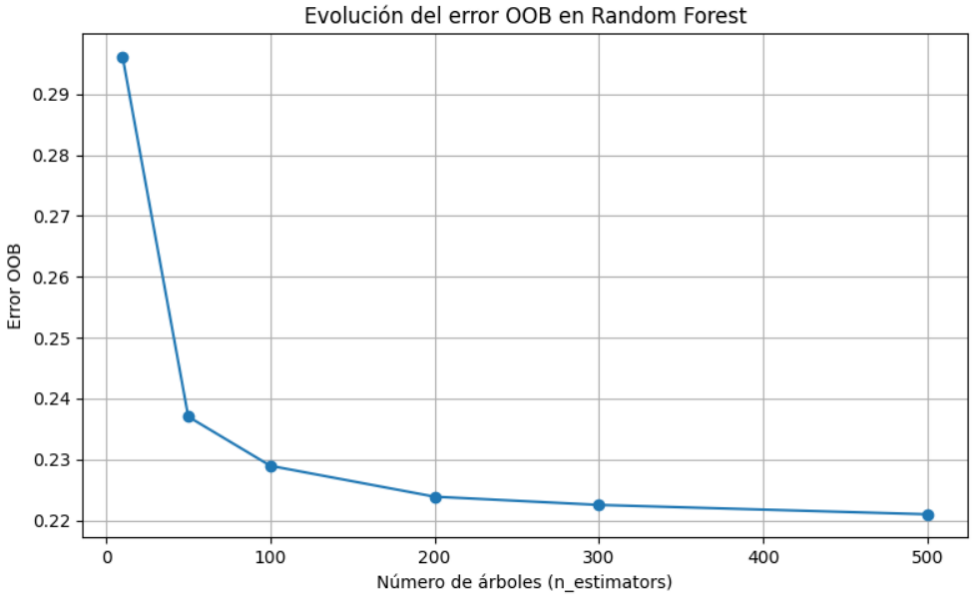

Figura 11 Evolución del Error OOB

Se observa una reducción progresiva del error "*Out-of-bag*" (OOB) a medida que se incrementa el número de estimadores, estabilizándose a partir de los 200 árboles. Esta tendencia sugiere que un mayor número de árboles contribuye a mejorar la capacidad de generalización del modelo, sin introducir sobreajuste.

En el proceso de búsqueda de hiperparámetros con "*Random Forest*", el modelo candidato inicial se encontraba en el extremo superior de la parrilla, con "*N_estimators*" = 500. A fin de verificar si existía margen de mejora, se amplió la grilla incluyendo valores superiores (1000, 1500, 2000). Sin embargo, debido a las limitaciones de memoria del entorno de ejecución, no fue posible completar todas las combinaciones. Aun así, el mejor resultado obtenido correspondió a un modelo con 500 árboles y "*Max_depth=None*", lo cual sugiere que un mayor número de estimadores podría seguir mejorando el rendimiento, aunque de forma marginal. Esta ampliación permitió validar que el modelo candidato se encuentra cerca de un óptimo y respalda su elección como modelo final.

Como se indicó anteriormente, el modelo fue evaluado sobre una muestra balanceada de 5000 tweets. Los resultados se presentan a continuación:

Tabla 4 Resultados Random Forest

Clase	Precision	Recall	F1- Score	Support
Derecha	0.97	1.00	0.98	2500
Izquierda	1.00	0.97	0.98	2500
Accuracy			0.98	5000
Macro avg	0.98	0.98	0.98	5000
F1_weighted	0.98	0.98	0.98	5000

Los resultados obtenidos fueron notablemente altos, con una "*Accuracy*" del 98%, el valor más elevado entre todos los modelos tradicionales evaluados. La clase "derecha" presentó una precisión de 0.97 y un "*Recall*" perfecto de 1.00, mientras que la clase "izquierda" obtuvo una precisión de 0.98 y un "*Recall*" de 0.97. Estas métricas reflejan una alta capacidad del modelo para clasificar correctamente la gran mayoría de los ejemplos, con apenas unos pocos errores.

El "*F1-score*" promedio (macro y ponderado) fue de 0.98, lo que sugiere un rendimiento extremadamente equilibrado y excepcional en ambas clases. No obstante, este resultado tan elevado puede ser indicativo de sobreajuste, especialmente si se considera que los datos provienen de una representación TF-IDF con alta dimensionalidad.

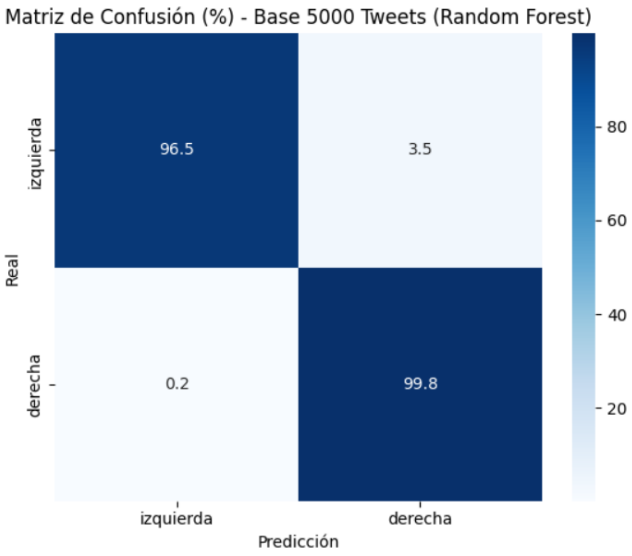

Figura 12 Matriz de Confusión "Random Forest"

La matriz de confusión lo confirma: el modelo clasificó correctamente el 96.5% de los tweets etiquetados como izquierda y el 98.6% de los de derecha. Los errores fueron mínimos, con apenas un 3.5% de ejemplos de izquierda mal clasificados como derecha, y tan solo un 0.2% de errores en la clase derecha.

7.1.5 Gradient Boosting

Para el modelo "*Gradient Boosting*", se realizó una búsqueda de hiperparámetros sobre las combinaciones de "*Learning_rate*", "*N_estimators*" (número de árboles) y "*Max_depth*" (profundidad máxima del árbol), considerando la sensibilidad del algoritmo a estos parámetros. La parrilla evaluada fue:

"*Learning Rate*": [0.01, 0.1]
"*N_estimators*": [100, 200, 500]
"*Max_depth*": [2, 3, 5]
"*Min_samples_leaf*": [1, 5]

La búsqueda se llevó a cabo utilizando "*GridSearchCV*" con validación cruzada (CV=5), y como métricas de evaluación se consideraron tanto "*Accuracy*" como "*F1_weighted*", siendo esta última la métrica principal para la selección del modelo óptimo. El código completo de entrenamiento y evaluación se encuentra incluido en los anexos. El mejor conjunto encontrado fue:

"*Learning Rate*" = 0.1
"*N_estimators*" = 500
"*Max_depth*" = 5
"*Min_samples_leaf*" = 5

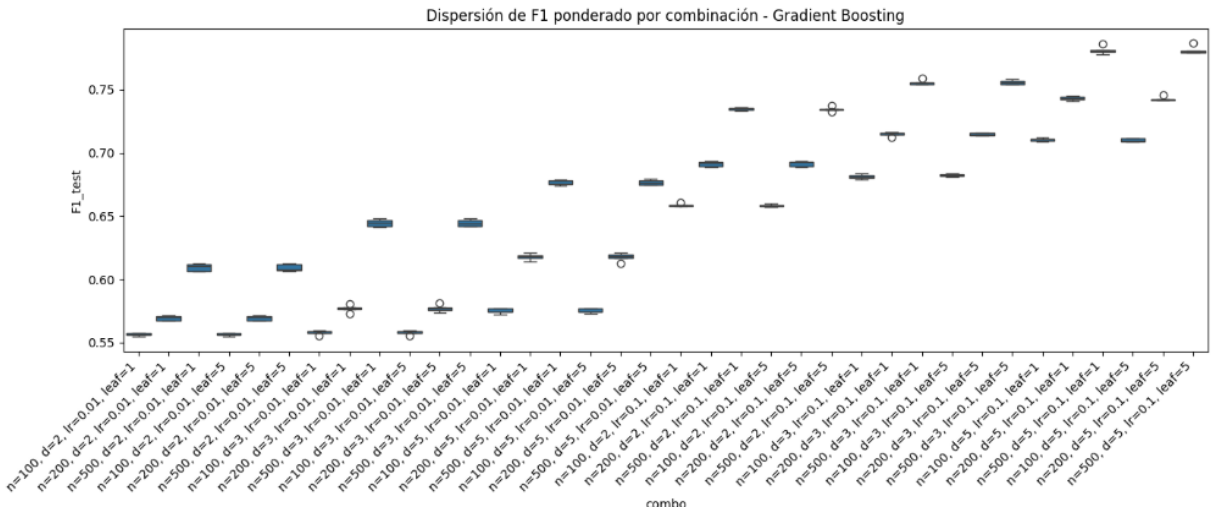

Figura 13 Boxplot "Gradient Boosting"

El "*Boxplot*" por combinación muestra que esta configuración no solo alcanzó el valor más alto de F1 ponderado, sino que también presentó una baja variabilidad entre los "*folds*", lo que respalda su estabilidad en distintos subconjuntos de datos.

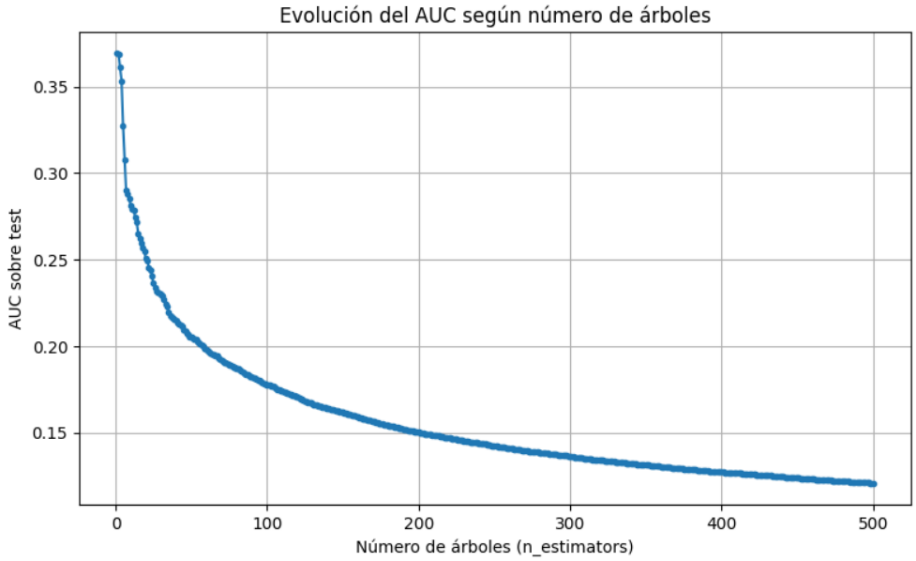

Figura 14 Evolución del AUC

Se aprecia un aumento sostenido en el rendimiento del modelo conforme se incrementa el número de estimadores, con una mejora progresiva del AUC que tiende a estabilizarse hacia los 500 árboles. Esta evolución confirma que el modelo continúa beneficiándose de una mayor capacidad de aprendizaje sin evidencias de sobreajuste.

A partir del análisis realizado y en línea con las recomendaciones metodológicas, se consideró ampliar la búsqueda de hiperparámetros del modelo de *"Gradient Boosting"* incluyendo configuraciones con un mayor número de estimadores (*"N_estimators"*) y tamaños de hoja más grandes (*"Min_samples_leaf"*), ya que estos ajustes suelen favorecer el rendimiento y la estabilidad del modelo al permitir un aprendizaje más progresivo y menos propenso al

sobreajuste. Sin embargo, debido a las limitaciones de memoria del sistema, no fue posible completar dichas ejecuciones. Aun así, el modelo actual, que utiliza 500 estimadores y "*Min_samples_leaf*" = 5, mostró un rendimiento sólido.

Los resultados de las métricas de evaluación sobre una muestra balanceada de 5000 tweets se resumen en la siguiente tabla:

Tabla 5 Resultados Gradient Boosting

Clase	Precision	Recall	F1- Score	Support
Derecha	0.85	0.73	0.79	2500
Izquierda	0.76	0.87	0.81	2500
Accuracy			0.80	5000
Macro avg	0.81	0.80	0.80	5000
F1_weighted	0.81	0.80	0.80	5000

El modelo alcanzó una precisión promedio del 80%, ubicándose en la media respecto a los modelos evaluados. Si bien la clase "izquierda" fue clasificada con una precisión del 76% y un "*Recall*" del 87%, la clase "derecha" obtuvo una precisión del 85% y un "*Recall*" del 73%, reflejando una clara disparidad en el desempeño por clase.

Estas diferencias se ven también en los valores de "*F1-score*", donde la clase izquierda alcanza 0.81 y la derecha 0.79. Esto sugiere una fuerte tendencia del modelo a clasificar los mensajes como izquierda, afectando su capacidad de generalización en contextos balanceados.

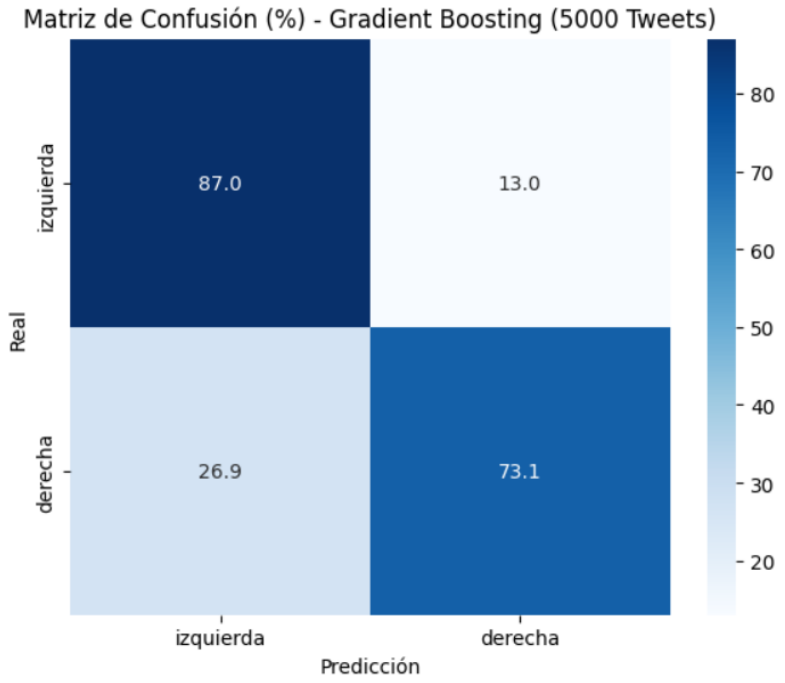

Figura 15 Matriz de confusión "Gradient Boosting"

La matriz de confusión refuerza esta observación: el modelo clasificó correctamente el 87% de los tweets de izquierda, pero solo el 73.1% de los tweets de derecha. Además, el 26.9% de los mensajes de derecha fueron clasificados incorrectamente como izquierda, lo que refleja un sesgo significativo.

Estos resultados sugieren que "*Gradient Boosting*" no es la opción más adecuada para este problema, al menos bajo esta configuración y representación textual. Probablemente, su bajo rendimiento se deba a la alta dimensionalidad del espacio TF-IDF y a la sensibilidad del modelo al ruido inherente a este tipo de representaciones.

7.1.6 Ridge Classifier

Para el modelo "*Ridge Classifier*", se llevó a cabo una búsqueda sobre el parámetro "*Alpha*", el cual regula la magnitud de la regularización. La parrilla de valores evaluada fue:

"*Alpha*": [0.001, 0.01, 0.1, 1, 10, 100]

Se utilizó "*GridSearchCV*" con validación cruzada (CV=5), evaluando las métricas "*F1_weighted*" y "*Accuracy*", con "*F1_weighted*" como criterio principal para la selección del modelo. El código completo de entrenamiento y evaluación se encuentra incluido en los anexos. El mejor resultado se obtuvo con:

"*Alpha*" = 1

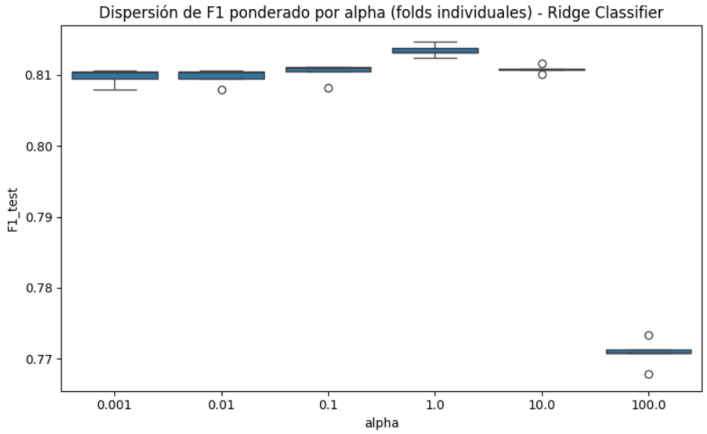

Figura 16 Boxplot "Ridge Classifier"

El "Boxplot" por partición de validación cruzada evidencia que el valor *"Alpha"* = 1 no solo alcanzó el mayor rendimiento promedio, sino que también presentó una estabilidad considerable entre los distintos "*folds*", sin valores atípicos extremos.

Como en los casos anteriores, la evaluación se realizó sobre una muestra balanceada de 5000 tweets. Los resultados se muestran a continuación:

Tabla 6 Resultados Ridge Classifier

Clase	Precision	Recall	F1- Score	Support
Derecha	0.86	0.83	0.85	2500
Izquierda	0.84	0.87	0.85	2500
Accuracy			0.85	5000
Macro avg	0.85	0.85	0.85	5000
F1_weighted	0.85	0.85	0.85	5000

El rendimiento global fue sólido, con un "*Accuracy*" del 85%, ubicándose entre los resultados más destacados del grupo. La precisión para la clase derecha fue de 0.84, con un "*Recall*" de 0.83, mientras que la clase izquierda obtuvo un desempeño ligeramente superior (precisión del 86% y "*Recall*" del 87%), lo cual refleja una mayor capacidad de recuperación ("*Recall*") de esta clase sin comprometer la precisión.

El "*F1-score*" promedio (macro y ponderado) fue de 0.85, confirmando la consistencia del modelo, especialmente en contextos balanceados. En comparación con otros clasificadores lineales como SVM y Regresión Logística, "*Ridge Classifier*" logró resultados muy competitivos, manteniendo una baja variabilidad en los "*folds*" durante la validación cruzada.

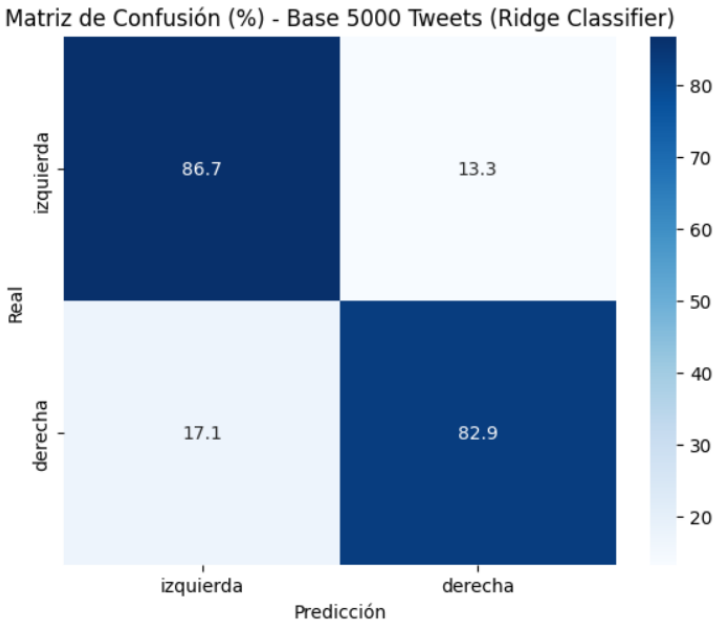

Figura 17 Matriz de Confusión "Ridge Classifier"

La matriz de confusión refuerza estas observaciones: el modelo clasificó correctamente el 86.7% de los mensajes de izquierda y el 82.9% de los mensajes de derecha. Los errores de clasificación fueron moderados, distribuyéndose en un 13.3% de mensajes de izquierda mal clasificados como derecha, y un 17.1% de errores en la dirección opuesta.

En conjunto, "*Ridge Classifier*" se posiciona como una opción robusta y fiable, ofreciendo un rendimiento competitivo aun sin la complejidad de modelos más avanzados.

7.1.7 Passive Aggressive

Para el modelo "*Passive-Aggressive*", se llevó a cabo una búsqueda de hiperparámetros sobre los valores de C, "*Loss*" y "*Average*", siendo estos los parámetros más influyentes en el ajuste del modelo. La parrilla evaluada fue:

C: [0.001, 0.01, 0.1, 1, 10, 100]
"*Loss*": ['*Hinge*', '*Squared_Hinge*']
"*Average*": ["*False*", "*True*"]

El ajuste se realizó mediante "*GridSearchCV*" con validación cruzada (CV=5) y dos métricas de evaluación: "*F1_weighted*" y "*Accuracy*", siendo "*F1_weighted*" el criterio principal de selección. El código correspondiente se encuentra incluido en los anexos. El mejor conjunto identificado fue:

C = 0.01
"*Loss*" = '*Squared_Hinge*'
"*Average*" = "True"

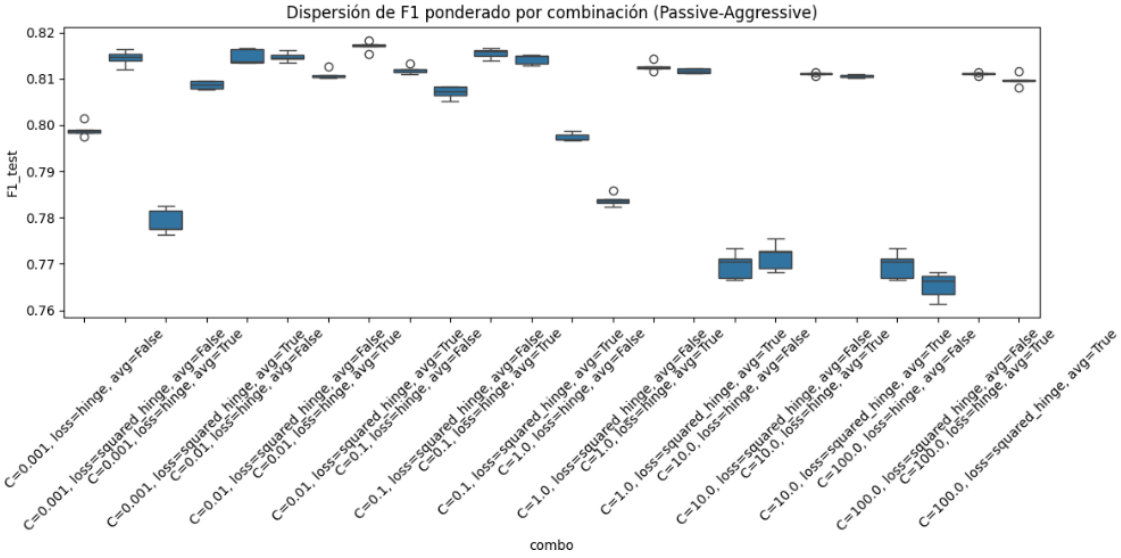

Figura 18 Boxplot "Passive-Aggressive"

El "*Boxplot*" por combinación mostró un rendimiento más estable y elevado en configuraciones con valores bajos de C. En particular, la combinación óptima destacó por su baja dispersión en F1 ponderado entre los diferentes "*folds*", lo que sugiere robustez del modelo en diferentes subconjuntos de datos.

Como se mencionó al inicio del capítulo el modelo se evalúa sobre una muestra balanceada de 5000 tweets.

Tabla 7 Resultados Passive Aggressive

Clase	Precision	Recall	F1- Score	Support
Derecha	0.86	0.83	0.85	2500
Izquierda	0.84	0.86	0.85	2500
Accuracy			0.85	5000
Macro avg	0.85	0.85	0.85	5000
F1_weighted	0.85	0.85	0.85	5000

En términos generales, el modelo alcanzó un "*Accuracy*" del 85%, mostrando un desempeño competitivo. Para la clase derecha, obtuvo una precisión del 84% y un "*Recall*" del 83%, mientras que la clase izquierda logró una precisión del 86% y un "*Recall*" del 87%. Esto indica un rendimiento equilibrado, con buena capacidad de generalización en ambas clases.

El "*F1-score*" promedio fue de 0.85, tanto en su forma macro como ponderada, lo que sugiere una consistencia adecuada del modelo entre clases. Aunque se encuentra ligeramente por debajo de los resultados de modelos más complejos, sus métricas son estables y fiables, particularmente en escenarios donde se requiere eficiencia computacional.

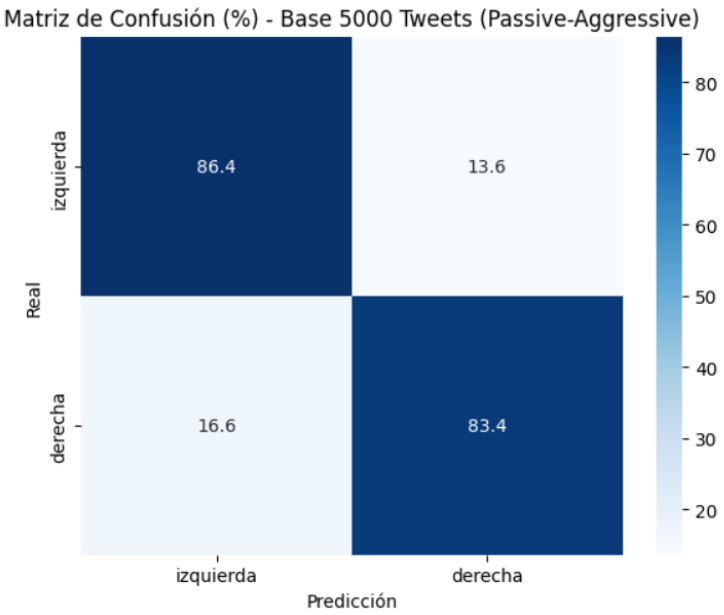

Figura 19 Matriz de Confusión "Passive-Aggressive"

La matriz de confusión refuerza esta lectura: el modelo identificó correctamente el 86.4% de los mensajes de izquierda y el 83.4% de los de derecha. Los errores se concentraron en un 13.6% de mensajes de izquierda mal clasificados como derecha, y un 16.6% en sentido inverso.

En resumen, "*Passive-Aggressive*" se presenta como una opción razonable para tareas de clasificación binaria de texto, ofreciendo un buen equilibrio entre rendimiento y eficiencia computacional, sin requerir una estructura de entrenamiento compleja.

7.2 Métodos avanzados

7.2.1 Clasificación Zero-Shot con BART (facebook/bart-large-mnli)

Para la clasificación "*Zero-Shot*" se utilizó el modelo preentrenado facebook/bart-large-mnli mediante la interfaz pipeline de la librería "*Transformers*" de "*Hugging Face*". Este modelo está basado en la arquitectura BART y ha sido ajustado para tareas de inferencia textual natural (NLI), lo que le permite realizar clasificación sin necesidad de reentrenamiento con ejemplos específicos. En este caso, se aplicó sobre el corpus balanceado, formulando la tarea como una hipótesis del tipo: "Este tweet es de {}", con las etiquetas objetivo "izquierda" y "derecha". El modelo evaluó cada ejemplo en función de su compatibilidad con dichas hipótesis y asignó la clase con mayor puntuación. No se realizaron ajustes finos sobre el modelo ni modificaciones internas, utilizando directamente sus pesos preentrenados. El código completo del procedimiento se encuentra disponible en los anexos.

Tabla 8 Resultados Zero-Shot

Clase	Precision	Recall	F1- Score	Support
Derecha	0.51	0.75	0.61	2500
Izquierda	0.53	0.29	0.38	2500
Accuracy			0.52	5000
Macro avg	0.52	0.52	0.49	5000
F1_weighted	0.52	0.52	0.49	5000

A pesar de ser un modelo avanzado, el rendimiento fue limitado, con una "*accuracy*" global de apenas 52%, lo que indica que el modelo clasifica correctamente solo un poco más de la mitad de los datos. Las métricas por clase revelan una fuerte asimetría en el comportamiento, para la clase derecha la precisión fue de 0.51 y el "*Recall*" alcanzó un 0.75, lo que indica que el modelo tiene una alta tendencia a predecir esta clase, aunque con errores significativos. En contraste, para la izquierda, la precisión fue levemente superior (0.53), pero el "*Recall*" cayó a 0.29, lo que refleja una gran incapacidad para recuperar correctamente los ejemplos reales de esta categoría.

El F1-score promedio fue de solo 0.49, quedando por debajo del umbral aceptable para tareas de clasificación binaria. Esto indica que el modelo no logra mantener un buen equilibrio entre precisión y exhaustividad.

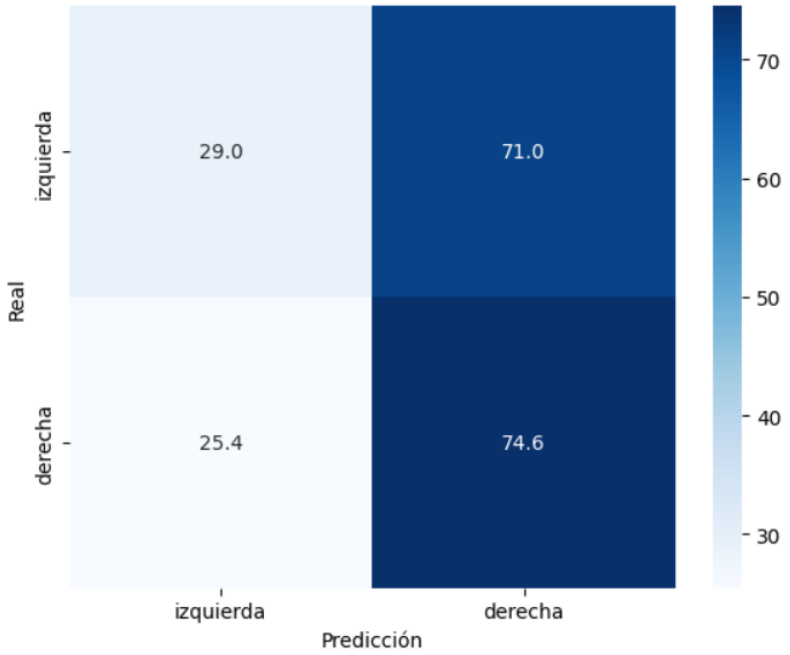

Figura 20 Matriz de Confusión "Zero-Shot"

La matriz de confusión ilustra este comportamiento, el modelo clasificó correctamente el 74.6% de los tweets de derecha, pero tan solo el 29.0% de los de izquierda. De hecho, el 71.0% de los mensajes de izquierda fueron clasificados erróneamente como derecha, lo que sugiere un sesgo estructural en la interpretación ideológica de los textos por parte del modelo.

En resumen, aunque el enfoque Zero-Shot resulta atractivo por su capacidad de generalización sin entrenamiento, su rendimiento práctico en esta tarea específica es insuficiente, especialmente frente a los modelos entrenados sobre el dominio del corpus.

7.2.2 Clasificación por Similitud con Embeddings Multilingües (SBERT)

Para este enfoque se utilizó el modelo preentrenado "*distiluse-base-multilingual-cased-v2*" de la biblioteca "*sentence-transformers*", una variante multilingüe de SBERT ("*Sentence-BERT*") optimizada para generar representaciones semánticas densas de oraciones. Este modelo no fue ajustado ni reentrenado; se utilizó directamente para codificar los textos en vectores de "*Embeddings*". La estrategia consistió en calcular un vector promedio por clase ideológica ("izquierda" y "derecha") a partir de los tweets etiquetados, y luego asignar la etiqueta a cada nuevo ejemplo en función de la similitud de coseno entre su "*Embedding*" y los vectores promedio de cada clase. Este procedimiento no requiere un clasificador tradicional, sino que basa la decisión en la cercanía semántica dentro del espacio vectorial generado por el modelo. El código completo del proceso se encuentra disponible en los anexos.

Aunque SBERT no está diseñado como un modelo de clasificación "*Zero-Shot*" en sentido estricto, en este trabajo se adaptó a dicho formato mediante una estrategia basada en similitud de "*Embeddings*". Para ello, se calcularon representaciones promedio por clase (izquierda y derecha) y se comparó cada tweet con estos vectores centroide utilizando la

similitud de coseno, permitiendo asignar una etiqueta ideológica sin necesidad de reentrenamiento supervisado.

Tabla 9 Resultados SBERT

Clase	Precision	Recall	F1- Score	Support
Derecha	0.65	0.66	0.66	2500
Izquierda	0.66	0.65	0.65	2500
Accuracy			0.66	5000
Macro avg	0.66	0.66	0.66	5000
F1_weighted	0.66	0.66	0.66	5000

El modelo fue evaluado sobre un conjunto balanceado de 5,000 tweets. El rendimiento obtenido fue modesto, alcanzando una "*Accuracy*" del 66%, claramente superior al enfoque "*Zero-Shot*", pero aún por debajo de los clasificadores tradicionales.

En términos de métricas por clase, se observó un desempeño muy equilibrado, para la clase derecha, el modelo alcanzó una precisión de 0.65 y un "*Recall"* de 0.66 y para la izquierda, la precisión fue de 0.66 y el "*Recall*" de 0.65.

Estas cifras se reflejan en un "*F1-score*" de 0.66 en ambas clases y en los promedios macro y ponderado. Si bien el rendimiento no es alto, sí muestra una simetría casi perfecta entre ambas categorías, lo que es destacable dado que el modelo no fue entrenado explícitamente para esta tarea, sino que se basa exclusivamente en distancias semánticas.

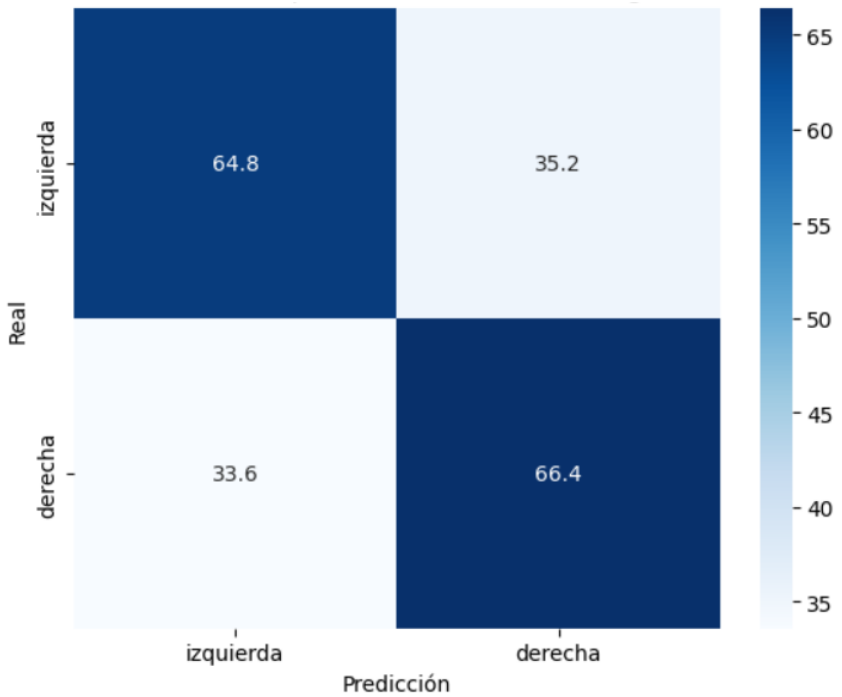

Figura 21 Matriz de Confusión "SBERT"

La matriz de confusión refuerza esta impresión, el modelo clasificó correctamente el 64.8% de los tweets de izquierda y el 66.4% de los de derecha. Aunque las tasas de error aún son considerables (alrededor del 33–35%), los errores no están concentrados en una sola clase, lo que sugiere que el modelo no presenta un sesgo ideológico marcado, a diferencia del enfoque "*Zero-Shot*".

En resumen, el enfoque basado en "*Embeddings*" + similitud logra un equilibrio razonable sin necesidad de entrenamiento supervisado, pero su desempeño general sigue estando por debajo del de los modelos tradicionales ajustados específicamente a la tarea.

7.2.3 Clasificación BETO (BERT en español)

Para este modelo se utilizó BETO, una variante del modelo BERT entrenada específicamente para el idioma español, a través de la versión "*dccuchile/bert-base-spanish-wwm-uncased*" disponible en "*Hugging Face*". A diferencia de otros modelos preentrenados utilizados en este trabajo, BETO fue ajustado ("*fine-tuned*") para la tarea de clasificación binaria sobre el conjunto de datos etiquetado. Se empleó la clase "*AutoModelForSequenceClassification*" con dos etiquetas ("*num_labels*"=2) y se tokenizaron los textos usando el tokenizador correspondiente ("*AutoTokenizer*"). El entrenamiento se llevó a cabo durante 3 épocas con un tamaño de lote de 16 ejemplos por dispositivo, aplicando una tasa de decaimiento de pesos ("*weight_decay*"=0.01). La evaluación se realizó sobre un subconjunto separado (20%) utilizando "*Trainer*" y "*TrainingArguments*", componentes estándar del ecosistema de "*Transformers*". Todos los detalles del proceso de entrenamiento, tokenización y evaluación se encuentran documentados en los anexos.

A diferencia de SBERT y BART, el modelo BETO fue entrenado específicamente sobre el corpus etiquetado mediante un procedimiento supervisado. Si bien no se llevó a cabo un "*fine-tuning*" extenso, sí se ajustaron los pesos del modelo para la tarea de clasificación binaria (izquierda vs. derecha) en el conjunto experimental. Por tanto, BETO no se empleó como modelo "*Zero-Shot*", sino como un clasificador entrenado con etiquetas manuales.

Tabla 10 Resultados BETO

Clase	Precision	Recall	F1- Score	Support
Derecha	0.80	0.84	0.82	2500
Izquierda	0.83	0.80	0.81	2500
Accuracy			0.82	5000
Macro avg	0.82	0.82	0.82	5000
F1_weighted	0.82	0.82	0.82	5000

El modelo alcanzó una "*accuracy*" del 82%, mostrando un rendimiento balanceado entre ambas clases. El F1-score macro y "*F1_weighted*" se mantiene también en 0.82, indicando un desempeño robusto y sin sesgos notables hacia alguna clase.

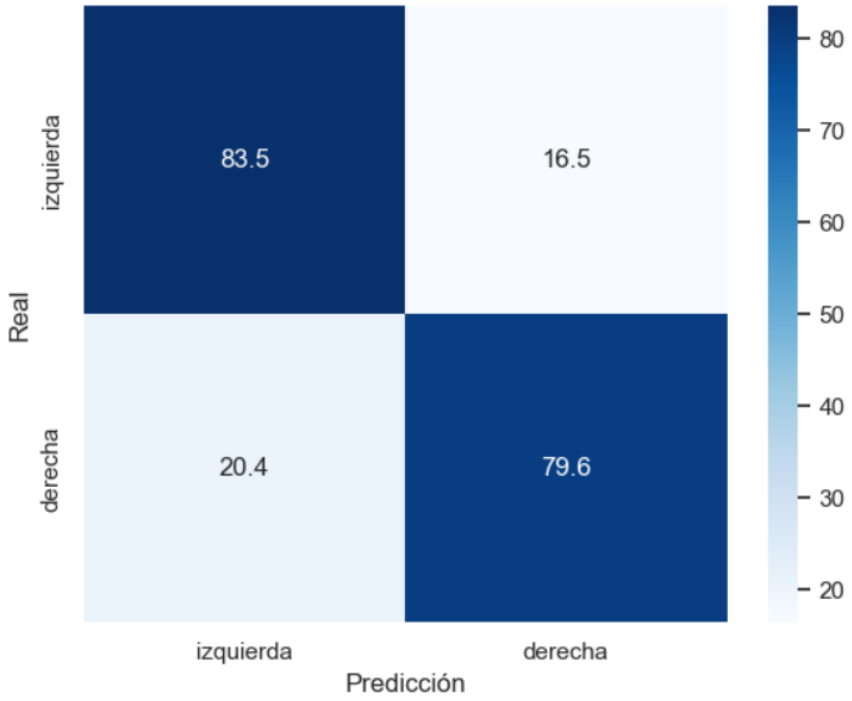

Figura 22 Matriz de Confusión BETO

El modelo acierta el 83.5% de los tweets de izquierda, aunque comete un error del 16.5% clasificándolos como derecha. Para la clase derecha, el 79.6% son clasificados correctamente, mientras que el 20.4% son confundidos con izquierda.

7.3 Comparación de modelos

7.3.1 Modelos basados en algoritmos clásicos

Tabla 11 Comparación resultados modelos basados en algoritmos clásicos

Modelo	Accuracy	Precision	Recall	F1 Score
SVM	0.84	0.84	0.84	0.84
Regresión Logística	0.85	0.85	0.85	0.85
Naive Bayes	0.80	0.80	0.80	0.80
Random Forest	0.98	0.98	0.98	0.98
Gradient Boosting	0.80	0.81	0.80	0.75
Ridge Classifier	0.85	0.85	0.85	0.85
Passive-Aggressive	0.85	0.85	0.85	0.85

Tabla 12 Resultados validación cruzada F1-Score ponderado – 5 fold

Modelo	Media	Desviación
SVM	0.814	0.001
Regresión Logística	0.815	0.001
Naive Bayes	0.789	0.001
Random Forest	0.775	0.001
Gradient Boosting	0.743	0.001
Ridge Classifier	0.813	0.001
Passive-Aggressive	0.817	0.001

"*Random Forest*" obtuvo las mejores métricas globales con un "*Accuracy*", "*Precisión*", "*Recall*" y "*F1-score*" de 0.98. Sin embargo, su desempeño en validación cruzada fue más bajo (0.775), lo que sugiere un posible sobreajuste al conjunto de entrenamiento.

Regresión Logística, "*Ridge Classifier*" y "*Passive-Aggressive*" mostraron un rendimiento muy competitivo, con valores constantes de 0.85 en todas las métricas. También fueron de los más estables en la validación cruzada (media 0.815, desviación muy baja), lo que refleja consistencia y robustez.

SVM ofreció resultados similares a Regresión Logística, "*Rigde Classifier*" y "*Passive-Aggressive*", tanto en rendimiento final (0.84 en todas las métricas) como en estabilidad (media = 0.814), consolidándose como una opción sólida con menor coste computacional.
"*Naive Bayes*" se ubicó en un rango medio con 80.0 en todas las métricas, con una desviación de 0.789 en validación cruzada, lo que refuerza su estabilidad, aunque con menor capacidad predictiva que otros modelos anteriormente mencionados.

"*Gradient Boosting*" fue el modelo con peor rendimiento global (F1 = 0.75), destacando un fuerte desbalance entre clases en su matriz de confusión, y también con baja media de F1 en validación cruzada. Esto sugiere dificultades de generalización en este contexto.

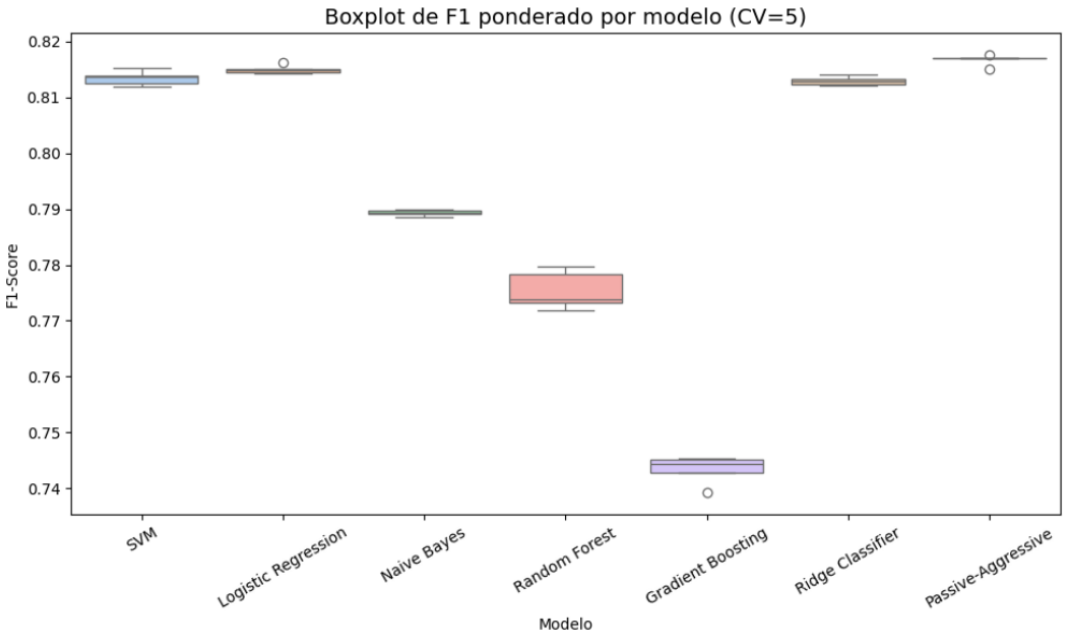

Figura 23 Boxplot Validación Cruzada Modelos Clásicos

El "*Boxplot*" de validación cruzada muestra claramente que Regresión Logística y "*Ridge Classifier*" y "*Passive-Aggressive*" son los modelos más estables y consistentes, con desviaciones mínimas entre los pliegues.

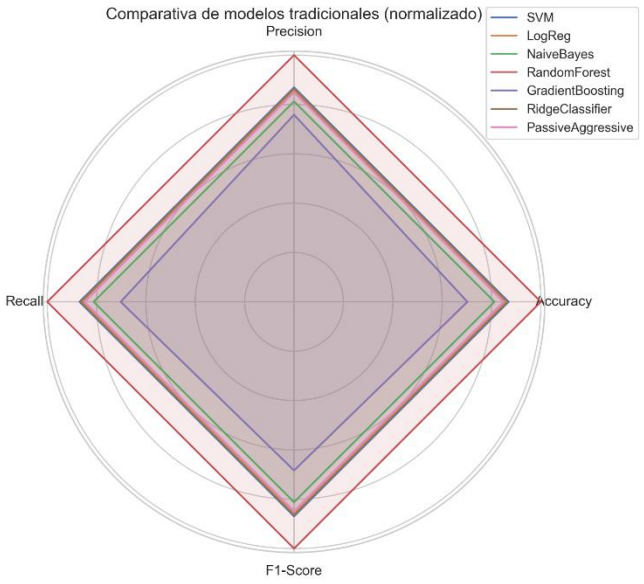

Figura 24 Radar Chart Modelos Clásicos

El "*Radar Chart*" evidencia que "*Random Forest*" domina en todas las métricas puntuales, pero este desempeño no se sostiene en validación cruzada, alertando sobre su posible dependencia al conjunto de entrenamiento.

La comparación sugiere que, si se prioriza el rendimiento absoluto, "*Random Forest*" es el modelo con mejores métricas, aunque con reservas por sobreajuste. Si en cambio se valora más la estabilidad y confiabilidad del modelo en nuevos datos, Regresión logística, "*Ridge Classifier*" y "*Passive-Aggressive*" resultan opciones preferibles para el presente proyecto.

7.3.2 Métodos avanzados

Tabla 13 Comparación resultados métodos avanzados

Modelo	Accuracy	Precision	Recall	F1 Score
Zero-Shot	0.52	0.52	0.52	0.49
SBERT	0.66	0.66	0.66	0.66
BETO	0.82	0.82	0.82	0.82

"*Zero-Shot*" mostró el rendimiento más bajo, aunque su capacidad de clasificación sin entrenamiento es notable, su F1-score de 0.49 refleja que no es competitivo frente a modelos entrenados específicamente en el dominio de las ideologías (izquierda y derecha). Además, su matriz de confusión indicó un sesgo importante hacia una de las clases, lo que limita su aplicabilidad en tareas balanceadas.

SBERT ("*Embeddings*" + Similitud) mejoró sensiblemente los resultados respecto a "*Zero-Shot*", alcanzando una precisión y "*Recall*" de 0.66. Esto demuestra que el uso de representaciones semánticas con comparación por similitud puede ser una alternativa simple y efectiva, aunque sigue lejos del rendimiento de los modelos entrenados.

BETO, por su parte, demostró una capacidad predictiva sobresaliente entre los modelos avanzados, con métricas consistentes de 0.82 en todas las dimensiones. Este resultado lo posiciona al nivel de los mejores modelos tradicionales (como SVM y Ridge), validando el potencial de los modelos "*transformer*" especializados en español para tareas de análisis político.

La comparativa evidencia que los modelos pre-entrenados sin "*fine-tuning*" ("*Zero-Shot*" y SBERT) pueden ofrecer una línea base razonable, especialmente cuando no se dispone de datos etiquetados. Sin embargo, cuando se cuenta con datos específicos y se entrena adecuadamente (como fue el caso de BETO), los resultados mejoran sustancialmente.

Capítulo 8 Estudio caso Zero-Shot

En el marco del presente Trabajo de Fin de Máster, se desarrolló un estudio de caso centrado en el análisis del discurso político en redes sociales, con el objetivo de identificar el tono emocional predominante en las publicaciones de diferentes partidos políticos españoles. Para ello, se aplicó un enfoque de análisis de sentimiento basado en modelos de inferencia "*Zero-*

Shot", una técnica de procesamiento del lenguaje natural (NLP) que permite clasificar textos sin necesidad de entrenamiento previo sobre una categoría específica.

El uso de "*Zero-Shot*" resulta especialmente valioso en contextos como el discurso político, donde las categorías de análisis son complejas, variables y no siempre se ajustan fácilmente a esquemas tradicionales de sentimiento positivo, negativo o neutro. En lugar de eso, el modelo BART se empleó para detectar la presencia de ideas, actitudes o narrativas a través de hipótesis lingüísticas previamente definidas.

Este procedimiento se aplicó sobre una base de datos compuesta por cinco mil de tweets emitidos por cuentas oficiales de partidos, sus principales representantes y usuarios de la red social. Los mensajes fueron evaluados por el modelo para estimar la probabilidad de que estuvieran vinculados a temas específicos como discurso anti elitista, voluntad del pueblo, pueblo como identidad o adversarios ideológicos (enemigo de derecha e izquierda)

Este capítulo presenta los resultados del análisis, tanto en términos cuantitativos como visuales, y constituye la base empírica para la elaboración del artículo científico "*Voices of Discontent: Unpacking Populist Rhetoric in Spain and the Rise of Anti-European Sentiment*", enviado para su evaluación por pares y posible publicación a la revista de impacto (Q1) "*Media and Communication".*

8.1 Metodología

La base de datos utilizada en este estudio se compone de más de cinco mil tweets, recopilados de cuentas oficiales de partidos políticos y figuras emblemáticas durante la campaña electoral para las elecciones europeas de 2024. Dado que desde 2023 el acceso gratuito a la API de Twitter ha sido restringido, la recolección de datos se realizó utilizando la plataforma "*Web Data Research Assistant*", compatible con la API de la red social X (anteriormente Twitter).

Los datos fueron recolectados entre el 24 de mayo y el 10 de junio de 2024, coincidiendo con el periodo oficial de campaña. Se aplicaron filtros basados en dos criterios: temporal (solo tweets publicados durante la campaña) y autoría (solo mensajes emitidos por las cuentas oficiales de partidos políticos con representación parlamentaria y sus líderes). Tras un proceso de limpieza, se eliminaron duplicados y se obtuvo un conjunto final de 2,218 mensajes únicos.

Con el objetivo de analizar el discurso político en Twitter desde una perspectiva semántica y emocional, se definieron cuatro grandes bloques temáticos que reflejan narrativas comunes en contextos políticos y populistas:

- Discurso anti elitista
- Pueblo uno
- Voluntad pueblo
- Enemigo izquierda
- Enemigo derecha

Cada uno de estos bloques está compuesto por subtemas específicos, operacionalizados en función de las características del discurso político observado. A su vez, cada subtema fue definido a través de un conjunto de hipótesis lingüísticas (etiquetas) que fueron empleadas en el modelo de clasificación "*Zero-Shot*" para detectar su presencia en los tweets.

El modelo "*Zero-Shot*", basado en una arquitectura de inferencia textual (como BART-MNLI), permite clasificar cada texto según la probabilidad de que este "hable sobre" alguna de las hipótesis definidas por el investigador. Para ello, es necesario estructurar previamente un conjunto de temas y subtemas que orienten la clasificación. A continuación, se muestra un ejemplo de cómo se desglosa esta estructura:

- Bloque: Discurso antielitista
    - Subtema: Contra élites políticas
        - Hipótesis asociadas:
            - la casta
            - los de arriba
            - paguitas
            - chiringuitos
            - corruptos
            - Soros
            - élites globalistas

En este caso, el modelo evalúa si un tuit expresa alguno de estos contenidos, comparando el texto con enunciados construidos a partir de la plantilla: "Este tweet es sobre {}". Si la hipótesis coincide semánticamente con el contenido del texto, el modelo asigna la categoría correspondiente

Este procedimiento se aplicó de forma sistemática en 11 subtemas, utilizando listas de hipótesis cuidadosamente diseñadas para capturar los elementos léxicos y simbólicos más representativos de cada categoría discursiva.

Tabla 14 Hipótesis discurso político

Tema	Subtema	hipótesis (Este texto habla sobre)
Discurso anti elitista	Discurso contra elites políticas	la casta, los de arriba, paguitas, chiringuitos, corruptos, Soros, elites globalistas.
	Discurso contra elites económicas	IBEX 35, poderes económicos, poderes oligárquicos, grandes fortunas, elite económica, neoliberalismo, grandes tecnológicas, empresas energéticas.
	Discurso contra elites mediáticas	poderes mediáticos, izquierda mediática, medios subvencionados, medios del bipartidismo.

	Discurso contra elites culturales	lobby LGTB, chiringuitos de género, ideología de género, propaganda progre, dictadura climática.
Pueblo uno	Representación positiva del pueblo	pueblo, patria
	Pueblo soberano	soberanía española, soberanía nacional
	Pueblo como clase social	los de abajo, españoles de a pie, España que madruga, trabajadores del campo, España real, gente trabajadora
	Pueblo como nación	la España viva, nuestro pueblo, la nación
Voluntad pueblo	Defensa de la voluntad popular	voluntad de los españoles, voluntad del pueblo, voluntad nacional
Enemigo izquierda	Enemigo de izquierda	fascismo, extrema derecha, ultraderecha, franquistas, negacionistas
Enemigo derecha	Enemigo de derecha	separatistas, comunistas, proetarra, chavismo, Venezuela, partidos golpistas, progres, menas, okupas, magrebíes, separatismo totalitario, república islámica, ley trans, inmigración ilegal

Cada subtema fue evaluado individualmente por el modelo, y solo se conservaron aquellos tweets que presentaban una probabilidad superior a 0.8 en al menos una de las hipótesis asociadas.

Para simplificar el análisis y mejorar la interpretación política de los resultados, se agruparon las cuentas oficiales de los partidos junto con las de sus principales representantes bajo una única etiqueta por partido. Esta decisión metodológica permite observar de manera más clara las estrategias discursivas de cada formación política en relación con los temas analizados.

Tabla 15 Relación partido político y figura

Partido	Figura
Partido Popular	Dolors Monserrath
PSOE	Teresa Ribera
Sumar	Estrella Galán
Podemos	Irene Montero
VOX	Jorge Buxade

8.2 Resultados análisis de sentimiento de discurso político

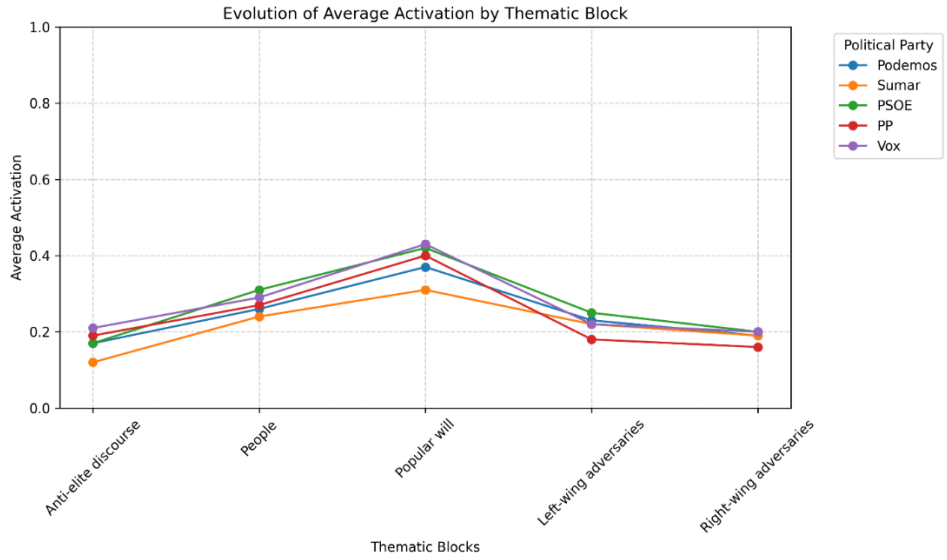

Figura 25 Evolución de la activación por bloque político

En el siguiente grafico se observa la evolución de la activación media por bloque temático y partido político. El análisis muestra que todos los partidos concentran su discurso con mayor intensidad en el bloque de la "Voluntad Popular", seguido de temáticas relacionadas con "el pueblo" y el discurso anti elitista.

Vox y PSOE destacan por su alta activación en "Voluntad Popular" (0.43 y 0.42 respectivamente), mientras que Sumar mantiene valores consistentemente más bajos en todos los bloques. Esto podría indicar una menor carga emocional en comparación con el resto de los partidos.

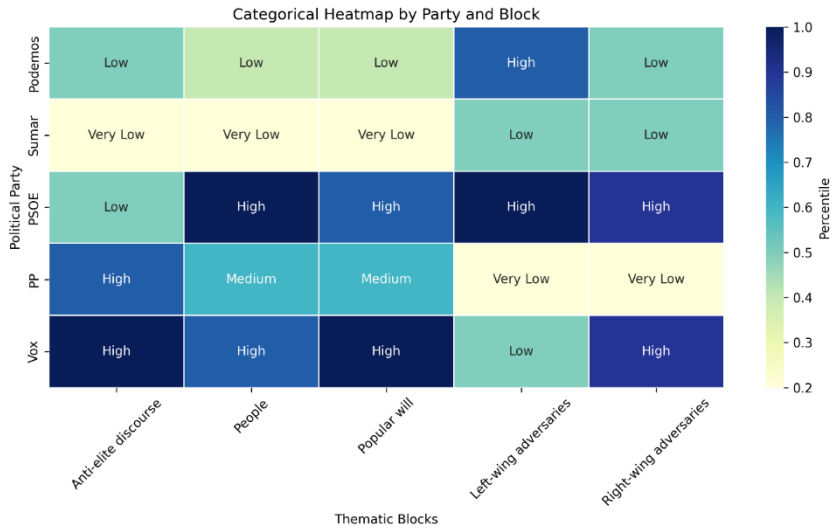

Figura 26 Mapa de calor categórico por partido y bloque político

El mapa de calor categórico clasifica las activaciones por percentiles (de "*Very Low*" a "*High*") para cada bloque y partido. Esta visualización facilita la comparación cualitativa del nivel de intensidad temática empleada por cada formación política.

Vox obtiene clasificación "*High*" en todos los bloques, salvo en "adversarios de izquierda", mientras que PP muestra "*High*" en discurso anti elitista, pero cae a "*Very Low*" en "Voluntad Popular" y "adversarios", por su parte PSOE es el único partido que obtiene la etiqueta "*High*" en todos los bloques salvo el anti elitista y finalmente Sumar se mantiene entre "*Very Low*" y "*Low*" en todos los bloques, lo que sugiere un tono discursivo menos marcado emocional o ideológicamente o bien más neutral.

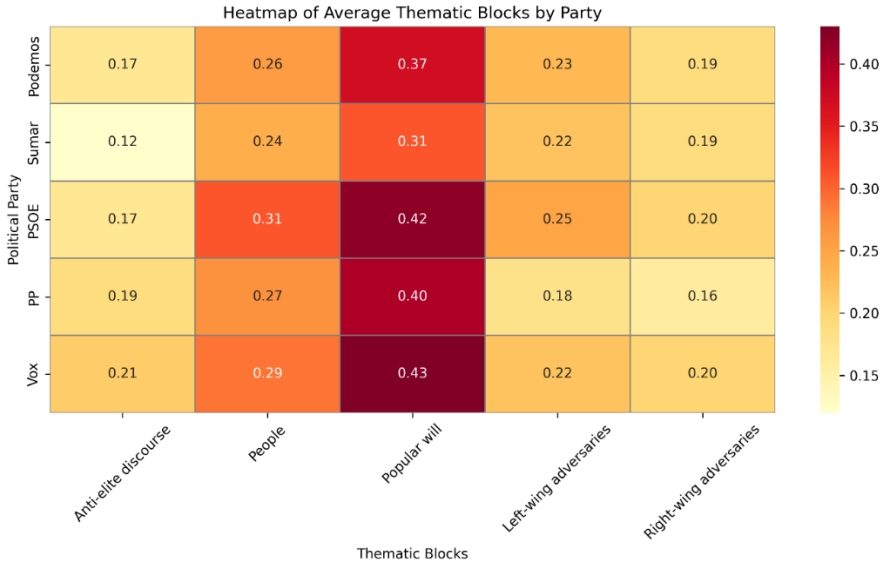

Figura 27 Mapa de calor cuantitativo por partido y bloque

A continuación, se muestran las activaciones promedio exactas por bloque temático y partido. Este "*heatmap*" cuantitativo complementa la visión categórica anterior al mostrar los valores reales de activación.

Voluntad Popular" lidera en todos los partidos, con valores entre 0.31 (Sumar) y 0.43 (Vox). El bloque "Pueblo" tiene una activación media consistente en todos los partidos, oscilando entre 0.24 y 0.31, mientras que el discurso anti elitista es más fuerte en Vox (0.21) y PP (0.19), y menos pronunciado en Sumar (0.12).

8.2.1 Discurso Anti-elitista

El bloque temático de discurso anti elitista agrupa expresiones que critican o desacreditan a las élites políticas, económicas, mediáticas y culturales. Este tipo de retórica es característico del discurso populista y suele tener una alta carga emocional, al contraponer a el pueblo frente a la elite.

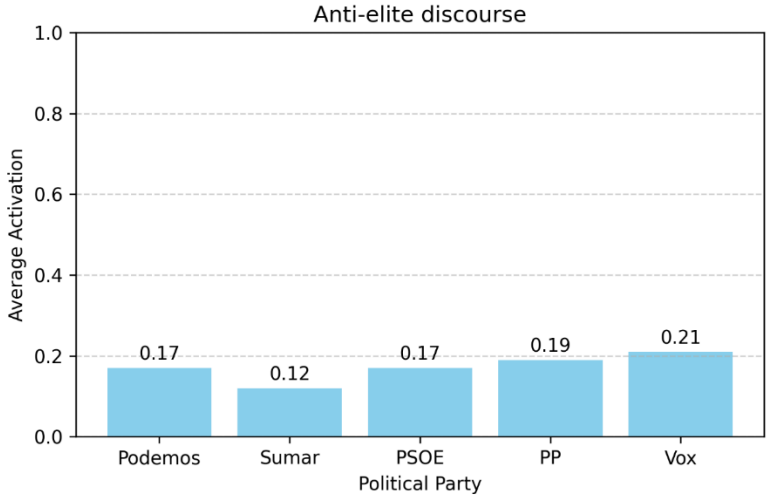

Figura 28 Discurso Anti-elitista

Se observa que Vox es el partido que muestra mayor activación media en este bloque (0.21), seguido por el Partido Popular (0.19). Estas formaciones, alineadas ideológicamente en la derecha, hacen uso de esta narrativa con mayor intensidad que los partidos de izquierda.

Estos resultados refuerzan la idea de que el discurso anti elitista no es exclusivo de una sola ideología, pero sí es más frecuente en los partidos con orientación crítica hacia las estructuras institucionales, especialmente cuando buscan conectar emocionalmente con sectores que perciben desigualdad o descontento con las élites.

8.2.2 Discurso Voluntad del Pueblo

El bloque Voluntad del Pueblo engloba expresiones que hacen referencia al pueblo como identidad colectiva, como sujeto político soberano, clase social trabajadora o nación. Este marco discursivo es común en narrativas populistas, donde se enfatiza la unidad y legitimidad moral del pueblo frente a otros actores.

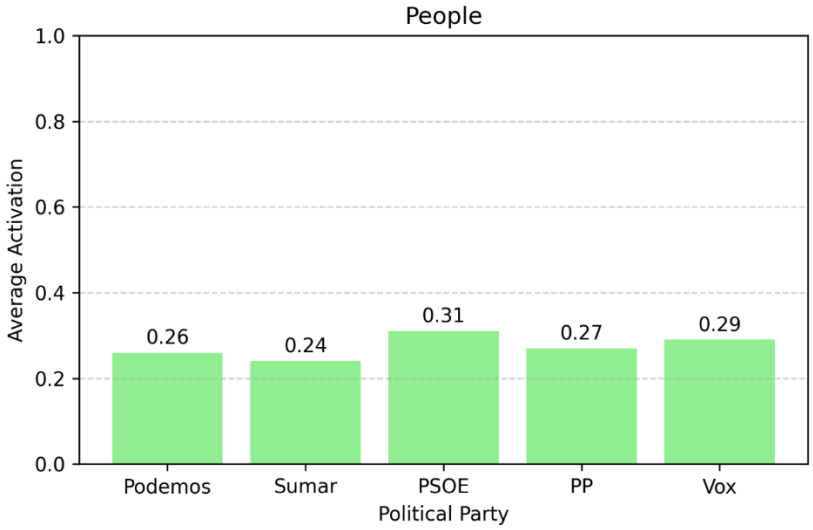

Figura 29 Discurso Voluntad del Pueblo

Se muestra las activaciones medias de este bloque por partido político. Se observa una distribución relativamente equilibrada entre formaciones, aunque con diferencias significativas: PSOE lidera este bloque con una activación de 0.31, seguido por Vox (0.29) y PP (0.27). Podemos (0.26) y Sumar (0.24) muestran una activación algo menor, aunque aún significativa.

Estos resultados indican que el uso del concepto de pueblo está generalizado entre los distintos partidos, aunque con variaciones sutiles en intensidad. El PSOE, en particular, parece utilizarlo con mayor frecuencia, posiblemente en un intento por conectar con un electorado amplio y apelando a la identidad nacional o social.

8.2.3 Discurso Voluntad Popular

El bloque voluntad popular agrupa expresiones que apelan directamente a la soberanía de la ciudadanía y a la legitimidad de su voluntad como fundamento de autoridad política. Este tipo de narrativa es común tanto en discursos populistas como en momentos de alta confrontación política.

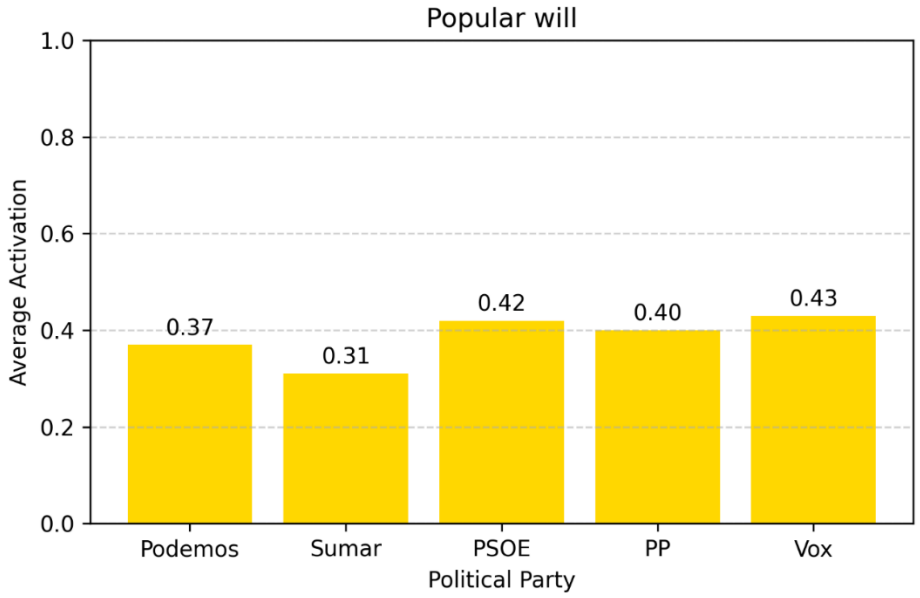

Figura 30 Discurso Voluntad Popular

Este bloque es el de mayor activación promedio en todos los partidos, lo que evidencia su uso transversal en el discurso político actual. Vox lidera este bloque con una activación media de 0.43, seguido por PSOE (0.42) y PP (0.40). Podemos mantiene un nivel relativamente alto (0.37), mientras que Sumar, aunque con el valor más bajo, sigue mostrando una activación significativa (0.31).

Este resultado sugiere que el discurso voluntad del pueblo es central en la ideología político contemporáneo en España, independientemente de la orientación (izquierda o derecha). Sin embargo, su uso puede responder a estrategias distintas, en partidos como Vox o PP puede vincularse con retóricas soberanistas, mientras que en el PSOE puede reflejar una apelación a la legitimidad institucional del voto y el mandato democrático.

8.2.4 Discurso Enemigo de Derecha

Este bloque agrupa expresiones que identifican o confrontan a sectores ideológicos de derecha como enemigos discursivos. Las hipótesis utilizadas incluyen referencias a conceptos como fascismo, ultraderecha, franquismo, neofascismo, entre otros.

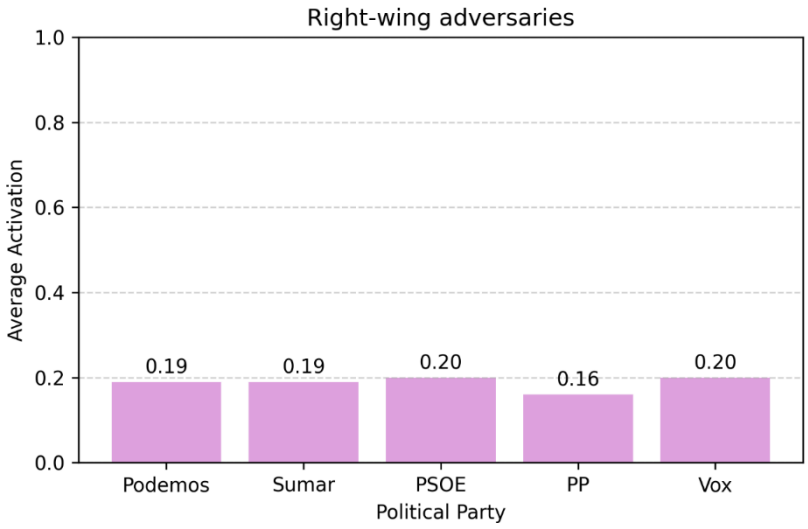

Figura 31 Discurso Enemigo de Derecha

En general, este bloque tiene una activación más baja y homogénea en comparación con otros. Los valores de activación media por partido se sitúan en un rango muy estrecho, entre 0.16 y 0.20 donde PSOE y Vox lideran con 0.20 mientras que Podemos y Sumar tienen activaciones similares (0.19) y por último PP presenta la activación más baja del bloque (0.16), lo cual es consistente con su posición ideológica, al ser menos probable que dirija su discurso contra sectores afines.

Este patrón sugiere que, si bien la crítica a la derecha está presente en el discurso de partidos como PSOE, Sumar y Podemos, no constituye un eje central de su narrativa en redes sociales, al menos en términos de activación medida mediante el modelo "*Zero-Shot*".

Además, el hecho de que Vox también obtenga una activación de 0.20 podría explicarse por menciones indirectas o referencias en terceros tuits que la técnica "*Zero-Shot*" captó como concordantes con las hipótesis.

8.2.5 Discurso enemigo de izquierda

El bloque incluye menciones negativas hacia actores ideológicos progresistas, comunistas, o vinculados al espectro de la izquierda política. Las hipótesis utilizadas incluyeron términos como progres, comunistas, socialistas, okupas, entre otros.

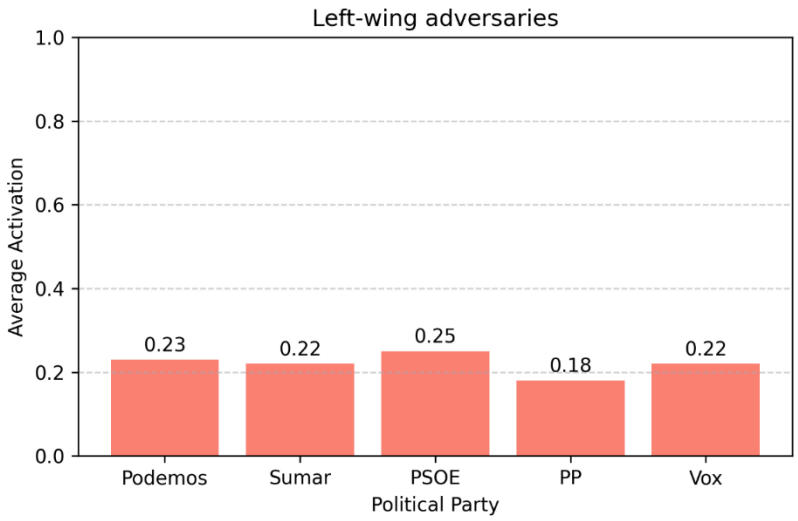

Figura 32 Discurso Enemigo de Izquierda

Se observa una activación promedio levemente superior respecto al bloque "Adversarios de derecha", aunque aún por debajo de los principales bloques temáticos. PSOE lidera el bloque con 0.25, seguido por Podemos (0.23) y Sumar junto con Vox (0.22). El Partido Popular muestra la activación más baja con 0.18, lo que resulta interesante dado que, por ideología, podría esperarse un discurso más crítico hacia la izquierda.

Este resultado sugiere que la crítica hacia la izquierda no es un eje dominante en la narrativa de los partidos de derecha en redes sociales, al menos desde el punto de vista léxico. Es posible que utilicen marcos simbólicos diferentes para expresar oposición ideológica sin recurrir directamente a etiquetas como las modeladas.

Conclusiones

Este trabajo tuvo como objetivo analizar y comparar distintos modelos de aprendizaje automático y profundo para la clasificación de la orientación ideológica en mensajes publicados en Twitter. Para ello, se planteó una metodología estructurada en dos fases: una experimental y otra aplicada.

En la fase experimental, se empleó un conjunto de datos etiquetado (Kaggle) para entrenar y evaluar distintos modelos supervisados. Se compararon modelos basados en algoritmos clásicos junto con modelos avanzados basados en lenguaje como BETO, SBERT y un enfoque "*Zero-Shot*" con BART. Tras aplicar un riguroso proceso de preprocesamiento y vectorización textual, los resultados mostraron que Regresión Logística, "*Ridge Classifier*" y "*Passive Aggressive*" fueron los modelos con mejor rendimiento global, superando incluso a los modelos avanzados en métricas como precisión y F1-score. Este hallazgo confirma que, en contextos donde se cuenta con un corpus etiquetado y bien definido, los modelos clásicos pueden ser más eficientes y precisos que los enfoques más complejos y costosos computacionalmente.

En la fase aplicada, se utilizó un modelo "*Zero-Shot*" (BART-MNLI) sobre un corpus original de mensajes políticos recolectados en tiempo real durante la campaña electoral de 2024 mediante técnicas de "*web scraping*". Esta etapa permitió evaluar el comportamiento de un modelo no supervisado en un entorno real, dinámico y sin etiquetado previo. Aunque su rendimiento fue inferior al de los modelos entrenados, demostró gran versatilidad para tareas de clasificación rápida en contextos no estructurados.

Bibliografía

Ahmed Khan, T., Sadiq, R., Shahid, Z., Alam, M. M., & Mohd Su'ud, M. (2024). Sentiment Analysis using Support Vector Machine and Random Forest. *Journal of Informatics and Web Engineering*, *3*(1), 67–75. https://doi.org/10.33093/jiwe.2024.3.1.5

Al-Habib, H., Imah, E. M., Puspitasari, R. D. I., & Prahani, B. K. (2023). Text Processing Using Support Vector Machine for Scientific Research Paper Content Classification. In I. H. Agustin (Ed.), *Proceedings of the 1st International Conference on Neural Networks and Machine Learning 2022 (ICONNSMAL 2022)* (Vol. 177, pp. 273–282). Atlantis Press International BV. https://doi.org/10.2991/978-94-6463-174-6_20

Alisya Mutia Mantika, Agung Triayudi, & Rima Tamara Aldisa. (2024). Sentiment Analysis on Twitter Using Naïve Bayes and Logistic Regression for the 2024 Presidential Election. *SaNa: Journal of Blockchain, NFTs and Metaverse Technology*, *2*(1), 44–55. https://doi.org/10.58905/sana.v2i1.267

Amini, T. A., & Setiawan, K. (2024). Application of the Naive Bayes Algorithm in Twitter Sentiment Analysis of 2024 Vice Presidential Candidate Gibran Rakabuming Raka using Rapidminer. *International Journal Software Engineering and Computer Science (IJSECS)*, *4*(1), 234–246. https://doi.org/10.35870/ijsecs.v4i1.2236

Arslan, M., Mubeen, M., Akram, A., Abbasi, S. F., Ali, M. S., & Tariq, M. U. (2024). A Deep Features Based Approach Using Modified ResNet50 and Gradient Boosting for Visual Sentiments Classification. *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 239–242. https://doi.org/10.1109/MIPR62202.2024.00045

Awanthika Madhushani. (2022). *Supervised and Unsupervised Learning (Machine Learning)| AI and Machine Learning*. https://doi.org/10.13140/RG.2.2.23820.49288

Bahrawi, N. (2019). Sentiment Analysis Using Random Forest Algorithm-Online Social Media Based. *Journal of Information Technology and Its Utilization*, *2*(2), 29. https://doi.org/10.30818/jitu.2.2.2695

Clarke, C., Heng, Y., Kang, Y., Flautner, K., Tang, L., & Mars, J. (2023). Label Agnostic Pre-training for Zero-shot Text Classification. *Findings of the Association for Computational Linguistics: ACL 2023*, 1009–1021. https://doi.org/10.18653/v1/2023.findings-acl.64

Cunha, W., Rocha, L., & Gonçalves, M. A. (2025). *A thorough benchmark of automatic text classification: From traditional approaches to large language models* (No. arXiv:2504.01930). arXiv. https://doi.org/10.48550/arXiv.2504.01930

Deode, S., Gadre, J., Kajale, A., Joshi, A., & Joshi, R. (2023). *L3Cube-IndicSBERT: A simple approach for learning cross-lingual sentence representations using multilingual BERT* (No. arXiv:2304.11434). arXiv. https://doi.org/10.48550/arXiv.2304.11434

Ferrer, L. (2023). *Analysis and Comparison of Classification Metrics* (No. arXiv:2209.05355). arXiv. https://doi.org/10.48550/arXiv.2209.05355

Ghosh, S. (2024). Study of Machine Learning Algorithms: A Historical Perspective. *Ci-STEM Journal of Digital Technologies and Expert Systems*, *01*(02), 86–97. https://doi.org/10.55306/CJDTES.2024.1204

Hagi, A., & Rarasati, D. B. (2024). Sentiment Analysis of Sirekap Application Review Using Logistic Regression Algorithm. *Jurnal Informatika*, *11*(2), 55–64. https://doi.org/10.31294/inf.v11i2.22066

Hugging Face. (2024). Hugging Face – The AI community building the future. https://huggingface.co

Jannah, N. Z. B., & Kusnawi, K. (2024). Comparison of Naïve Bayes and SVM in Sentiment Analysis of Product Reviews on Marketplaces. *Sinkron*, *8*(2), 727–733. https://doi.org/10.33395/sinkron.v8i2.13559

Khan, I. U., & Khan, M. U. S. (2024). Social Media Profiling for Political Affiliation Detection. *Human-Centric Intelligent Systems*, *4*(3), 437–446. https://doi.org/10.1007/s44230-024-00078-y

Kim, T. W., & Duhachek, A. (2020). Artificial Intelligence and Persuasion: A Construal-Level Account. *PSYCHOLOGICAL SCIENCE*, *31*(4), 363–380. https://doi.org/10.1177/0956797620904985

López Condori, J. J., & Gonzales Saji, F. O. (2021). Análisis de sentimiento de comentarios en español en Google Play Store usando BERT. *Ingeniare. Revista chilena de ingeniería*, *29*(3), 557–563. https://doi.org/10.4067/S0718-33052021000300557

Mabokela, K. R., Celik, T., & Raborife, M. (2023). Multilingual Sentiment Analysis for Under-Resourced Languages: A Systematic Review of the Landscape. *IEEE Access*, *11*, 15996–16020. https://doi.org/10.1109/ACCESS.2022.3224136

Madurga, A. C., & Payo, V. C. (n.d.). *Ana´lisis de sentimientos y emociones en redes sociales usando ML*.

Mao, Z., Chu, C., & Kurohashi, S. (2024). EMS: Efficient and Effective Massively Multilingual Sentence Embedding Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *32*, 2841–2856. https://doi.org/10.1109/TASLP.2024.3402064

Miranda, C. H., Sanchez-Torres, G., & Salcedo, D. (2023). Exploring the Evolution of Sentiment in Spanish Pandemic Tweets: A Data Analysis Based on a Fine-Tuned BERT Architecture. *Data*, *8*(6), 96. https://doi.org/10.3390/data8060096

Moore, C. S., Chinta, P. C. R., Katnapally, N., Ja, K., Routhu, K. K., & Velaga, V. (2022). Leveraging Big Data Analytics and Machine Learning Techniques for Sentiment Analysis of Amazon Product Reviews in Business Insights. *Universal Library of Engineering Technology*, 10–17. https://doi.org/10.70315/uloap.ulete.2022.002

Nikhil Sanjay Suryawanshi. (2024). Sentiment analysis with machine learning and deep learning: A survey of techniques and applications. *International Journal of Science and Research Archive*, *12*(2), 005–015. https://doi.org/10.30574/ijsra.2024.12.2.1205

Olivares Lopez, J., Sánchez López, A., González Velázquez, R., Santiago Díaz, M. D. C., & Zenteno Vázquez, A. C. (2024). Machine learning techniques for sentiment analysis. *International Journal of Combinatorial Optimization Problems and Informatics*, *15*(5), 6–16. https://doi.org/10.61467/2007.1558.2024.v15i5.554

Popoola, G., Abdullah, K.-K., Fuhnwi, G. S., & Agbaje, J. (2024). Sentiment Analysis of Financial News Data using TF-IDF and Machine Learning Algorithms. *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*, 1–6. https://doi.org/10.1109/ICAIC60265.2024.10433843

Ramavath, B., Subash, N., & Kadainti, S. (2025). Sentiment Analysis Using Light Weight—Gradient Boosting Machine based Feature Selection. *Journal of Computer Science*.

Rodríguez Alcántara, N., Falck Durán, A., & Suazo Barahona, S. A. (2022). Análisis automático de sentimiento en tuits de política de Honduras. *Innovare: Revista de ciencia y tecnología*, *11*(3), 158–165. https://doi.org/10.5377/innovare.v11i3.15349

Sanchez-Medina, J. J. (2024). *Sentiment analysis and random forest to classify LLM versus human source applied to Scientific Texts* (No. arXiv:2404.08673). arXiv. https://doi.org/10.48550/arXiv.2404.08673

Saura, J. R., Reyes-Menendez, A., & Palos-Sanchez, P. (n.d.). *A feeling analysis in Twitter with machine learning: Capturing sentiment from #BlackFriday offers*.

Sharma, N. A., Ali, A. B. M. S., & Kabir, M. A. (2025). A review of sentiment analysis: Tasks, applications, and deep learning techniques. *International Journal of Data Science and Analytics*, *19*(3), 351–388. https://doi.org/10.1007/s41060-024-00594-x

Shu, L., Xu, H., Liu, B., & Chen, J. (2022). *Zero-Shot Aspect-Based Sentiment Analysis* (No. arXiv:2202.01924). arXiv. https://doi.org/10.48550/arXiv.2202.01924

Sitarz, M. (2023). Extending F1 metric, probabilistic approach. *Advances in Artificial Intelligence and Machine Learning*, *03*(02), 1025–1038. https://doi.org/10.54364/AAIML.2023.1161

Sonal J. Patel, & Dr. Sachin A. Goswami. (2024). *A Comparative Study of Supervised and Unsupervised Classification Techniques for Sentiment Analysis in IMDB*. https://doi.org/10.5281/ZENODO.11044986

Taye, M. M. (2023). Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Computers*, *12*(5), 91. https://doi.org/10.3390/computers12050091

Tesfagergish, S. G., Kapočiūtė-Dzikienė, J., & Damaševičius, R. (2022). Zero-Shot Emotion Detection for Semi-Supervised Sentiment Analysis Using Sentence Transformers and Ensemble Learning. *Applied Sciences*, *12*(17), 8662. https://doi.org/10.3390/app12178662

Tun, Y. M., & Khaing, M. (2023). A large-scale sentiment analysis using political tweets. *International Journal of Electrical and Computer Engineering (IJECE)*, *13*(6), 6913. https://doi.org/10.11591/ijece.v13i6.pp6913-6925

Valle-Cruz, D., Sandoval-Almazán, R., López-Chau, A., & Criado, J. I. (2024). Unveiling political polarization on Twitter: Machine learning and sentiment analysis in presidential elections. *JeDEM - eJournal of eDemocracy and Open Government*, *16*(1), 186–212. https://doi.org/10.29379/jedem.v16i1.846

Vasquez, J., Gomez-Adorno, H., & Bel-Enguix, G. (n.d.). *Bert-based Approach for Sentiment Analysis of Spanish Reviews from TripAdvisor*.

Vinod, S. M., Bouh, M. M., Hossain, F., Paul, P., & Ahmed, A. (2023). Advancements in Text Classification, A Comprehensive Review. *2023 IEEE 11th Region 10 Humanitarian*

Technology Conference (R10-HTC), 679–684. https://doi.org/10.1109/R10-HTC57504.2023.10461820

Vladić, E., Mehanović, B., Novalić, M., Kečo, D., & Mehanović, D. (2024). *Applying NLP and Machine Learning for Sentiment Classification Across Multiple Twitter Datasets*. *12*.

Wang, Z., & Li, R. (2024). Advancements in social media sentiment analysis: A study utilizing the PassiveAggressiveClassifier model. *Applied and Computational Engineering*, *71*(1), 150–156. https://doi.org/10.54254/2755-2721/71/20241392

Wardana, N. S., Aditiawan, F. P., & Sari, A. P. (2024). Logistic Regression Classification with TF-IDF and FastText for Sentiment Analysis of LinkedIn Reviews. *VISA: Journal of Vision and Ideas*, *4*(3). https://doi.org/10.47467/visa.v4i3.2835

Zhan, Z. (2025). Comparative Analysis of TF-IDF and Word2Vec in Sentiment Analysis: A Case of Food Reviews. *ITM Web of Conferences*, *70*, 02013. https://doi.org/10.1051/itmconf/20257002013

Anexos

Código limpieza de datos

```
# Eliminar filas donde el tweet esté vacío o sea Na
df = df[df["tweet"].notna()]
df = df[df["tweet"].str.strip() != ""]
```

```
def clean_text(text):
    if isinstance(text, str):
        text = re.sub(r"(?:^|\s)([@#]\S+)", "", text)
        text = text.strip()
        text = text.lower()
        text = text.replace("\n", " ")
        text = " ".join(text.split())
        text = re.sub(r"http\S+|www\S+", "", text)
        text = re.sub(r"[^\w\s.,!?¿¡]", "", text)
        text = unicodedata.normalize('NFKD', text).encode('ASCII', 'ignore').decode('utf-8')
        text = re.sub(r"([!?¿¡])\1+", r"\1", text)
    return text

df["tweet_limpio"] = df["tweet"].apply(clean_text)

# Columna 'tweet_limpio'
df = df[df["tweet_limpio"].str.contains(r"[a-zA-Z]", na=False)]
df = df[df["tweet_limpio"].str.count(r"\s+") > 0]
df = df[df["tweet_limpio"].str.contains(r"[aeiouAEIOU]", na=False)]
```

```
# Eliminar duplicados exactos
df = df.drop_duplicates(subset="tweet_limpio")
```

```
# Filtrar tweets cortos
df = df[df["tweet_limpio"].str.len() > 20]

# Eliminar respuestas tipo de plantillas comunes
plantillas = [
    "gracias por contactarnos",
    "nos pondremos en contacto",
    "mensaje recibido",
    "puede consultar en la web",
    "más información en nuestro sitio",
    "puede llamarnos al"
]

# Convertimos cada plantilla en filtro
for frase in plantillas:
    df = df[~df["tweet_limpio"].str.contains(frase, case=False)]
```

```python
# Mapeo ideológico binario
mapeo_binario = {
    "psoe": "izquierda",
    "podemos": "izquierda",
    "pp": "derecha",
    "vox": "derecha"
}

# Crear columna con etiquetas
df["etiqueta"] = df["partido"].map(mapeo_binario)
```

Código análisis exploratorio

```python
#Analisis Exploratorio

import matplotlib.pyplot as plt
import pandas as pd

df["fecha"] = pd.to_datetime(df["timestamp"], unit="s")
df["año"] = df["fecha"].dt.year
df["mes"] = df["fecha"].dt.month
df["dia"] = df["fecha"].dt.date
df["hora"] = df["fecha"].dt.hour

# Filtrar desde 2019
df_filtrado = df[df["año"] >= 2019]

# Gráfico de tweets por día
plt.figure(figsize=(12, 4))
df_filtrado.groupby("dia").size().plot(kind="line", title="Frecuencia de Tweets por Día (2019-2023)")
plt.xlabel("Fecha")
plt.ylabel("Número de Tweets")
plt.grid(True)
plt.tight_layout()
plt.show()

# Tweets por partido político
plt.figure(figsize=(8, 4))
df_filtrado["partido"].value_counts().plot(kind="bar", title="Número de Tweets por Partido (2019-2023)")
plt.xlabel("Partido")
plt.ylabel("Cantidad de Tweets")
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

```python
# Distribución por etiqueta ideológica
plt.figure(figsize=(6, 4))
df_filtrado["etiqueta"].value_counts().plot(kind="pie", autopct="%1.1f%%", title="Distribución por Etiqueta Ide
plt.ylabel("")
plt.tight_layout()
plt.show()
```

Código modelo SVM

Entrenamiento

```
from sklearn.svm import LinearSVC
from sklearn.calibration import CalibratedClassifierCV
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

#SVM

# Parrilla de hiperparámetros
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100]}

# Modelo base
svc = LinearSVC(class_weight='balanced', max_iter=10000, dual=False)

# Busqueda de hiperparámetros
grid = GridSearchCV(
    estimator=svc,
    param_grid=param_grid,
    scoring='f1_weighted',
    refit=True,
    cv=5,
    verbose=1,
    n_jobs=-1
)

# Entrenamiento
grid.fit(X_train_tfidf, y_train)
print("Mejores hiperparámetros:", grid.best_params_)
# Boxplot con scores por fold
f1_scores = []
for C in param_grid['C']:
    model = LinearSVC(C=C, class_weight='balanced', max_iter=10000, dual=False)
    scores = cross_val_score(model, X_train_tfidf, y_train, cv=5, scoring='f1_weighted')
    for score in scores:
        f1_scores.append({'C': C, 'F1_test': score})

df_box = pd.DataFrame(f1_scores)

plt.figure(figsize=(6, 5))
sns.boxplot(x='C', y='F1_test', data=df_box)
plt.title("Dispersión de F1 ponderado por C - LinearSVC")
plt.ylabel("F1 ponderado")
plt.xlabel("C")
plt.tight_layout()
plt.show()
```

Evaluación

```
# Prediccion en base de 5000 tweets
y_pred_beto = svc_best.predict(X_beto_tfidf)

# Reporte de clasificación
print("Reporte de clasificación - LinearSVC (5000 tweets):\n")
print(classification_report(y_beto, y_pred_beto))

# Matriz de confusión (valores)
cm = confusion_matrix(y_beto, y_pred_beto, labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión - LinearSVC (5000 tweets)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz de confusión (porcentajes)
cm_percent = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión (% por clase real) - LinearSVC (5000 tweets)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código modelo Regresión Logística

Entrenamiento

```
#Regresion Logistica
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import time

# === Parrilla de hiperparámetros ===
param_grid_log = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'class_weight': ['balanced', None],
    'solver': ['liblinear']
}

print("\n--- Entrenamiento: Regresión Logística ---\n")
start = time.time()

# === Grid Search ===
grid_log = GridSearchCV(
    estimator=LogisticRegression(max_iter=1000),
    param_grid=param_grid_log,
    scoring=['f1_weighted', 'accuracy'],
    refit='f1_weighted',
    cv=5,
    n_jobs=-1,
    verbose=2,
    return_train_score=True
)
grid_log.fit(X_train_tfidf, y_train)
end = time.time()

print(f"\nTiempo total: {(end - start)/60:.2f} minutos")
print("Mejores hiperparámetros:", grid_log.best_params_)
# Boxplot: F1 ponderado por C
results = pd.DataFrame(grid_log.cv_results_)
results['C'] = results['param_C'].astype(str)

sns.boxplot(x='C', y='mean_test_f1_weighted', data=results)
plt.title("Boxplot de F1 ponderado por C (Regresión Logística)")
plt.xlabel("Valor de C")
plt.ylabel("F1 ponderado promedio (CV=5)")
plt.tight_layout()
plt.show()
```

Evaluación

```
# Predicción en base de 5000 tweets
y_pred_beto = best_model.predict(X_beto_tfidf)

print("\n Reporte - Base de 5000 Tweets:")
print(classification_report(y_beto, y_pred_beto))

# Matriz de Confusión
cm = confusion_matrix(y_beto, y_pred_beto, labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión - Base 5000 Tweets")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz de Confusión (%)
cm_percent = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión (%) - Base 5000 Tweets")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código modelo Naive Bayes

Entrenamiento

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import time

# === Parrilla de hiperparámetros ===
param_grid_nb = {
    'alpha': [0.001, 0.01, 0.1, 1, 10, 100]
}

print("\n--- Entrenamiento: Naive Bayes ---\n")
start = time.time()

# === GridSearchCV ===
grid_nb = GridSearchCV(
    estimator=MultinomialNB(),
    param_grid=param_grid_nb,
    scoring=['f1_weighted', 'accuracy'],
    refit='f1_weighted',
    cv=5,
    n_jobs=-1,
    verbose=2,
    return_train_score=True
)

grid_nb.fit(X_train_tfidf, y_train)
end = time.time()

print(f"\nTiempo total: {(end - start)/60:.2f} minutos")
print("Mejores hiperparámetros:", grid_nb.best_params_)
```

```
# Boxplot de F1 ponderado por alpha
results_nb = pd.DataFrame(grid_nb.cv_results_)
results_nb['alpha'] = results_nb['param_alpha'].astype(str)

sns.boxplot(x='alpha', y='mean_test_f1_weighted', data=results_nb)
plt.title("Boxplot de F1 ponderado por alpha (Naive Bayes)")
plt.xlabel("Alpha")
plt.ylabel("F1 ponderado promedio (CV=5)")
plt.tight_layout()
plt.show()
```

Evaluación

```
# Predicción en base de 5000 tweets
y_pred_beto = best_model.predict(X_beto_tfidf)

print("\n Reporte - Base de 5000 Tweets:")
print(classification_report(y_beto, y_pred_beto))

# Matriz de Confusión (valores)
cm = confusion_matrix(y_beto, y_pred_beto, labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión - Base 5000 Tweets (Naive Bayes)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz de Confusión (%)
cm_percent = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión (%) - Base 5000 Tweets (Naive Bayes)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código modelo Random Forest

Entrenamiento

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import time

# Parrilla de hiperparámetros actualizada
param_grid = {
    'n_estimators': [100, 200, 500]
    'max_depth': [None, 10, 20],
    'class_weight': ['balanced']
}

# GridSearchCV
start = time.time()
grid = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    scoring=['f1_weighted', 'accuracy'],
    refit='f1_weighted',
    cv=5,
    n_jobs=-1,
    verbose=2,
    return_train_score=True
)
grid.fit(X_train_tfidf, y_train)
end = time.time()
print(f"Tiempo total: {(end - start)/60:.2f} minutos")
print("Mejores hiperparámetros:", grid.best_params_)
# Boxplot de F1 por combinación
results = pd.DataFrame(grid.cv_results_)
results['combo'] = results.apply(
    lambda row: f"n={row['param_n_estimators']}, depth={row['param_max_depth']}", axis=1)

fold_cols = [c for c in results.columns if 'split' in c and 'test_f1_weighted' in c]
df_long = results.melt(id_vars='combo', value_vars=fold_cols,
                       var_name='Fold', value_name='F1_test')

plt.figure(figsize=(12, 6))
sns.boxplot(x='combo', y='F1_test', data=df_long)
plt.xticks(rotation=45)
plt.title("Dispersión de F1 ponderado por combinación - Random Forest")
plt.tight_layout()
plt.show()
```

```python
# gráfico OOB

from sklearn.ensemble import RandomForestClassifier

oob_errors = []
n_values = [10, 50, 100, 200, 300, 500]

for n in n_values:
    rf = RandomForestClassifier(n_estimators=n, oob_score=True, bootstrap=True, random_state=42)
    rf.fit(X_train_tfidf, y_train)
    oob_error = 1 - rf.oob_score_
    oob_errors.append(oob_error)

plt.figure(figsize=(8, 5))
plt.plot(n_values, oob_errors, marker='o')
plt.title("Evolución del error OOB en Random Forest")
plt.xlabel("Número de árboles (n_estimators)")
plt.ylabel("Error OOB")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Evaluación

```python
# Prediccion base 5000 tweets
y_pred_rf_beto = best_rf.predict(X_beto_tfidf)

print("Reporte - Base 5000 Tweets (Random Forest):")
print(classification_report(y_beto, y_pred_rf_beto))

# Matriz de Confusión
cm_beto = confusion_matrix(y_beto, y_pred_rf_beto, labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm_beto, annot=True, fmt="d", cmap="Blues", xticklabels=["izquierda", "derecha"], yticklabels=["izq
plt.title("Matriz de Confusión - Base 5000 Tweets (Random Forest)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz %
cm_beto_percent = cm_beto.astype("float") / cm_beto.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_beto_percent, annot=True, fmt=".1f", cmap="Blues", xticklabels=["izquierda", "derecha"], ytickla
plt.title("Matriz de Confusión (%) - Base 5000 Tweets (Random Forest)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código modelo Gradient Boosting

Entrenamiento

```
# Gradient Boosting

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import time

# Parrilla de hiperparámetros ajustada
param_grid = {
    'n_estimators': [100, 200, 500],
    'max_depth': [2, 3, 5],
    'learning_rate': [0.01, 0.1],
    'min_samples_leaf': [1, 5]
}

# GridSearch
start = time.time()
gb = GradientBoostingClassifier(random_state=42)
grid = GridSearchCV(
    estimator=gb,
    param_grid=param_grid,
    scoring=['f1_weighted', 'accuracy'],
    refit='f1_weighted',
    cv=5,
    return_train_score=False,
    n_jobs=-1,
    verbose=2
)
grid.fit(X_train_tfidf, y_train)
end = time.time()
print(f"Tiempo total: {(end - start)/60:.2f} minutos")
print("Mejores hiperparámetros:", grid.best_params_)
# Boxplot de combinaciones
results = pd.DataFrame(grid.cv_results_)
results['combo'] = results.apply(lambda row: f"n={row['param_n_estimators']}, d={row['param_max_depth']}, lr={r
split_cols = [col for col in results.columns if "split" in col and "test_f1_weighted" in col]

df_box = results[split_cols]
df_box['combo'] = results['combo']
df_melt = df_box.melt(id_vars='combo', var_name='Fold', value_name='F1_test')

plt.figure(figsize=(14, 6))
sns.boxplot(data=df_melt, x='combo', y='F1_test')
plt.xticks(rotation=45, ha='right')
plt.title('Dispersión de F1 ponderado por combinación - Gradient Boosting')
plt.tight_layout()
plt.show()
```

```python
from sklearn.metrics import roc_auc_score

staged_aucs = []
for y_prob in best_model.staged_predict_proba(X_test_tfidf):
    y_score = y_prob[:, 1]
    auc_score = roc_auc_score((y_test == "derecha").astype(int), y_score)
    staged_aucs.append(auc_score)

plt.figure(figsize=(8, 5))
plt.plot(range(1, len(staged_aucs)+1), staged_aucs, marker='.')
plt.title("Evolución del AUC según número de árboles")
plt.xlabel("Número de árboles (n_estimators)")
plt.ylabel("AUC sobre test")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Evaluación

```python
# Prediccion base 5000 tweets
# Predecir con el mejor modelo
y_pred_gb_beto = best_model.predict(X_beto_tfidf)

# Reporte
print("Reporte de clasificación - Gradient Boosting (Prueba):")
print(classification_report(y_beto, y_pred_gb_beto))

# Matriz de confusión (valores absolutos)
cm = confusion_matrix(y_beto, y_pred_gb_beto, labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión - Gradient Boosting (5000 Tweets)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz de confusión (% por clase real)
cm_percent = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión (%) - Gradient Boosting (5000 Tweets)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código modelo Ridge Classifier

Entrenamiento

```
#Ridge Classiffier

from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import time

# === Parrilla de hiperparámetros ===
param_grid_ridge = {
    'alpha': [0.001, 0.01, 0.1, 1, 10, 100]
}

print("\n--- Entrenamiento: Ridge Classifier ---\n")
start = time.time()

# === GridSearchCV ===
grid_ridge = GridSearchCV(
    estimator=RidgeClassifier(),
    param_grid=param_grid_ridge,
    scoring=['f1_weighted', 'accuracy'],
    refit='f1_weighted',
    cv=5,
    n_jobs=-1,
    verbose=2,
    return_train_score=True
)
grid_ridge.fit(X_train_tfidf, y_train)
end = time.time()

print(f"\nTiempo total: {(end - start)/60:.2f} minutos")
print("Mejores hiperparámetros:", grid_ridge.best_params_)
# Boxplot por fold
results_ridge = pd.DataFrame(grid_ridge.cv_results_)
results_ridge['alpha'] = results_ridge['param_alpha'].astype(str)

split_cols = [col for col in results_ridge.columns if 'split' in col and 'test_f1_weighted' in col]
melted = results_ridge.melt(id_vars=['alpha'], value_vars=split_cols,
                            var_name='Fold', value_name='F1_test')

plt.figure(figsize=(8, 5))
sns.boxplot(x='alpha', y='F1_test', data=melted)
plt.title("Dispersión de F1 ponderado por alpha (folds individuales) - Ridge Classifier")
plt.tight_layout()
plt.show()
```

Evaluación

```
# Predicción en base de 5000 tweets
y_pred_ridge_beto = best_model.predict(X_beto_tfidf)

print("\n Reporte - Base de 5000 Tweets:")
print(classification_report(y_beto, y_pred_ridge_beto))

# Matriz de Confusión (valores)
cm = confusion_matrix(y_beto, y_pred_ridge_beto, labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión - Base 5000 Tweets (Ridge Classifier)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz de Confusión (%)
cm_percent = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión (%) - Base 5000 Tweets (Ridge Classifier)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código modelo Passive-Aggresive

Entrenamiento

```
#Passive-Aggressive

from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import time

# Parrilla de hiperparámetros
param_grid_pac = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'loss': ['hinge', 'squared_hinge'],
    'average': [False, True]
}

print("\n--- Entrenamiento: Passive-Aggressive ---\n")
start = time.time()

# === GridSearchCV ===
grid_pac = GridSearchCV(
    estimator=PassiveAggressiveClassifier(max_iter=1000, random_state=42),
    param_grid=param_grid_pac,
    scoring=['f1_weighted', 'accuracy'],
    refit='f1_weighted',
    cv=5,
    n_jobs=-1,
    verbose=2,
    return_train_score=True
)
grid_pac.fit(X_train_tfidf, y_train)
end = time.time()

print(f"\nTiempo total: {(end - start)/60:.2f} minutos")
print("Mejores hiperparámetros:", grid_pac.best_params_)
```

```
# Boxplot por combinación
results_pac = pd.DataFrame(grid_pac.cv_results_)
results_pac['combo'] = results_pac.apply(lambda row: f"C={row['param_C']}, loss={row['param_loss']}, avg={row['

split_cols = [col for col in results_pac.columns if 'split' in col and 'test_f1_weighted' in col]
melted = results_pac.melt(id_vars=['combo'], value_vars=split_cols,
                          var_name='Fold', value_name='F1_test')

plt.figure(figsize=(12, 6))
sns.boxplot(x='combo', y='F1_test', data=melted)
plt.xticks(rotation=45)
plt.title("Dispersión de F1 ponderado por combinación (Passive-Aggressive)")
plt.tight_layout()
plt.show()
```

Evaluación

```
# Prediccion en base de 5000 tweets
y_pred_pac_beto = best_model.predict(X_beto_tfidf)

print("\n Reporte - Base de 5000 Tweets:")
print(classification_report(y_beto, y_pred_pac_beto))

# Matriz de Confusión (valores)
cm = confusion_matrix(y_beto, y_pred_pac_beto, labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión - Base 5000 Tweets (Passive-Aggressive)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz de Confusión (%)
cm_percent = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=["izquierda", "derecha"], yticklabels=["izquierda", "derecha"])
plt.title("Matriz de Confusión (%) - Base 5000 Tweets (Passive-Aggressive)")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código modelo Clasificación Zero-Shot con BART (facebook/bart-large-mnli)

```
from transformers import pipeline
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import numpy as np
from tqdm import tqdm

# Inicializar modelo
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

df_test = df_beto_balanceado

# Lista para guardar predicciones
predicciones = []

# Clasificar cada tweet
for tweet in tqdm(df_test["tweet_limpio"], desc="Clasificando"):
    resultado = classifier(tweet, ["izquierda", "derecha"], hypothesis_template="Este tweet es de {}")
    predicciones.append(resultado["labels"][0])  # Toma la predicción con mayor score

# Agregar predicciones
df_test["pred_zero_shot"] = predicciones

# Evaluar
print("\n Reporte de clasificación - Zero-Shot:\n")
print(classification_report(df_test["etiqueta"], df_test["pred_zero_shot"]))

print("\n Matriz de confusión:\n")
print(confusion_matrix(df_test["etiqueta"], df_test["pred_zero_shot"]))
```

Matriz de confusión

```
# Calcular matriz de confusión
cm = confusion_matrix(df_test["etiqueta"], df_test["pred_zero_shot"], labels=["izquierda", "derecha"])

# Matriz de confusión
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["izquierda", "derecha"],
            yticklabels=["izquierda", "derecha"])
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.title("Matriz de Confusión - Zero-Shot")
plt.tight_layout()
plt.show()

# Matriz de confusión %
cm_percent = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] * 100

plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues",
            xticklabels=["izquierda", "derecha"],
            yticklabels=["izquierda", "derecha"])
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.title("Matriz de Confusión (% por clase real) - Zero-Shot")
plt.tight_layout()
plt.show()
```

Código modelo clasificación por similitud con Embeddings multilingües (SBERT)

```
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import classification_report, confusion_matrix

# Modelo SBERT multilingüe
modelo_sbert = SentenceTransformer("distiluse-base-multilingual-cased-v2")

# Dataset balanceado
df = df_beto_balanceado.copy()

# Crear vectores promedio por clase
emb_izquierda = modelo_sbert.encode(df[df["etiqueta"] == "izquierda"]["tweet_limpio"].tolist()).mean(axis=0)
emb_derecha = modelo_sbert.encode(df[df["etiqueta"] == "derecha"]["tweet_limpio"].tolist()).mean(axis=0)

# Codificar todos los tweets del dataset
emb_test = modelo_sbert.encode(df["tweet_limpio"].tolist())

# Comparar cada tweet
pred_sbert = []
for tweet_vec in tqdm(emb_test, desc="Comparando similitudes"):
    sim_izq = cosine_similarity([tweet_vec], [emb_izquierda])[0][0]
    sim_der = cosine_similarity([tweet_vec], [emb_derecha])[0][0]
    pred_sbert.append("izquierda" if sim_izq > sim_der else "derecha")

# Guardar predicciones
df["pred_sbert"] = pred_sbert

# Reporte de clasificación
print("\n📋 Reporte de clasificación - Embeddings + Similitud:\n")
print(classification_report(df["etiqueta"], df["pred_sbert"]))
```

Matriz de confusión

```
# Matriz de confusión
cm = confusion_matrix(df["etiqueta"], df["pred_sbert"], labels=["izquierda", "derecha"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["izquierda", "derecha"], yticklabels=["izquierd
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.title("Matriz de Confusión - Embeddings + Similitud")
plt.tight_layout()
plt.show()

# Matriz de confusión (%)
cm_percent = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues", xticklabels=["izquierda", "derecha"], yticklabels=
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.title("Matriz de Confusión (% por clase real) - Embeddings + Similitud")
plt.tight_layout()
plt.show()
```

Código modelo Clasificación BETO (BERT en español)

Entrenamiento

```
from transformers import AutoModelForSequenceClassification, Trainer, TrainingArguments
from transformers import AutoTokenizer
import torch
from sklearn.model_selection import train_test_split
from datasets import Dataset

# Tokenizador y modelo BETO
tokenizer = AutoTokenizer.from_pretrained("dccuchile/bert-base-spanish-wwm-uncased")
model = AutoModelForSequenceClassification.from_pretrained(
    "dccuchile/bert-base-spanish-wwm-uncased", num_labels=2
)

# Asignar etiquetas numéricas
df_entreno_beto["label"] = df_entreno_beto["etiqueta"].map({"izquierda": 0, "derecha": 1})

# Separar en train/test
train_texts, test_texts, train_labels, test_labels = train_test_split(
    df_entreno_beto["tweet_limpio"],
    df_entreno_beto["label"],
    test_size=0.2,
    random_state=42,
    stratify=df_entreno_beto["label"]
)

# Tokenización
def tokenize(batch):
    return tokenizer(batch["tweet_limpio"], truncation=True, padding="max_length", max_length=128)

train_dataset = Dataset.from_dict({"tweet_limpio": train_texts.tolist(), "label": train_labels.tolist()}).map(t
test_dataset = Dataset.from_dict({"tweet_limpio": test_texts.tolist(), "label": test_labels.tolist()}).map(toke
```

```
# Argumentos del entrenamiento
training_args = TrainingArguments(
    output_dir="./beto_model",
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10
)

# Entrenador
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset
)

# Entrenar
trainer.train()

# Evaluar
eval_results = trainer.evaluate()
print(eval_results)
```

Evaluación

```
# Preparar etiquetas para test
df_beto_balanceado["label"] = df_beto_balanceado["etiqueta"].map({"izquierda": 0, "derecha": 1})

# Tokenizar base de test externa
test_dataset = Dataset.from_dict({
    "tweet_limpio": df_beto_balanceado["tweet_limpio"].tolist(),
    "label": df_beto_balanceado["label"].tolist()
}).map(tokenize, batched=True)

# Predecir
predictions = trainer.predict(test_dataset)
y_pred = np.argmax(predictions.predictions, axis=1)
y_true = test_dataset["label"]

# Reporte de clasificación
print("\n📊 Reporte de clasificación - BETO:\n")
print(classification_report(y_true, y_pred, target_names=["izquierda", "derecha"]))
```

```
# Matriz de confusión
cm = confusion_matrix(y_true, y_pred, labels=[0, 1])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["izquierda", "derecha"], yticklabels=["izquierd
plt.title("Matriz de Confusión - BETO")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()

# Matriz de confusión %
cm_percent = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] * 100
plt.figure(figsize=(6, 5))
sns.heatmap(cm_percent, annot=True, fmt=".1f", cmap="Blues", xticklabels=["izquierda", "derecha"], yticklabels=
plt.title("Matriz de Confusión (% por clase real) - BETO")
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.tight_layout()
plt.show()
```

Código comparación de modelos

```
modelos = [
    ("SVM (LinearSVC)", svc_best),
    ("Logistic Regression", best_model_log),
    ("Naive Bayes", best_model_nb),
    ("Random Forest", best_rf),
    ("Gradient Boosting", best_model_gb),
    ("Ridge Classifier", best_model_ridge),
    ("Passive-Aggressive", best_model_pac)
]

# Consolidar resultados F1 ponderado con validación cruzada
from sklearn.model_selection import cross_val_score
import pandas as pd

resultados = []

for nombre, modelo in modelos:
    f1 = cross_val_score(modelo, X_train_tfidf, y_train, cv=5, scoring='f1_weighted')
    resultados.append({
        "Modelo": nombre,
        "Media": round(f1.mean(), 3),
        "Desviación": round(f1.std(), 3)
    })

df_resultados = pd.DataFrame(resultados)
df_resultados.sort_values(by="Media", ascending=False, inplace=True)

import ace_tools as tools; tools.display_dataframe_to_user(name="Resultados F1 ponderado - Validación cruzada",

# Lista de mejores modelos
modelos = [
    ("SVM", svc_best),
    ("Logistic Regression", best_model_log),
    ("Naive Bayes", best_model_nb),
    ("Random Forest", best_rf),
    ("Gradient Boosting", best_model_gb),
    ("Ridge Classifier", best_model_ridge),
    ("Passive-Aggressive", best_model_pac)
]

# Calcular F1 ponderado por fold
folds_data = []
for nombre, modelo in modelos:
    f1_scores = cross_val_score(modelo, X_train_tfidf, y_train, cv=5, scoring='f1_weighted')
    for score in f1_scores:
        folds_data.append({
            "Modelo": nombre,
            "F1_test": score
        })

# Crear DataFrame
df_folds = pd.DataFrame(folds_data)
```

```python
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.boxplot(data=df_folds, x="Modelo", y="F1_test", palette="pastel")

plt.title("Boxplot de F1 ponderado por modelo (CV=5)", fontsize=14)
plt.xlabel("Modelo")
plt.ylabel("F1-Score")
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```