Sistema autónomo de detección de noticias falsas en medios digitales utilizando inteligencia artificial

Ramón Felipe Pérez Osorio

Director

Jaime Rubiano Llorente

Universidad Nacional Abierta y a Distancia UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería ECBTI

Programa de Ingeniería de Sistemas (Resolución 011685)

2026

Contenido

Resumen 5

Abstract 6

Lista de Tablas 7

Lista de Ilustraciones 8

Introducción 12

Planteamiento del Problema 14

Justificación 16

Objetivos 18

Objetivo General 18

Objetivos Específicos 18

Delimitación del Proyecto 19

Marco Conceptual 20

Marco Teórico 22

Estado del Arte 22

Marco Jurídico 23

Nacionales 23

Internacionales 24

Marco Tecnológico 25

Metodología de la Investigación 26

Enfoque Metodológico 26

Muestra y Población del Proyecto 26

Construcción e Implementación del Instrumento de Medición y Recolección de los Datos 26

Análisis y Diagnostico del Proceso Investigativo 27

Metodología de Desarrollo Tecnológico 28

Metodología Ágil 28

Análisis de Requerimientos 28

Funcionales 28

Diseño del Prototipo 29

Fases del Desarrollo Tecnológico 31

ODS y TRL 33

ODS 33

TRL 33

Cronograma 34

Recursos Necesarios 35

Resultados o Productos Esperados 36

Desarrollo del Proyecto 37

Recopilación de Datos 37

Preprocesamiento de Datos 41

Diseño y Entrenamiento de Modelos 45

Modelos Tradicionales 45

Modelos de Deep Learning 70

Evaluación de Resultados 95

Implementación con Transformers 105

MBERT 105

BETO 109

Desarrollo de API 121

Desarrollo de Interfaz Web 129

Pruebas y Validación del Sistema 131

Discusión 143

Conclusiones 146

Conclusiones Específicas Según los Objetivos Planteados 146

Conclusiones Generales 147

Recomendaciones 150

Impacto, Implicaciones Éticas y Consideraciones Sociales del Sistema 153

Limitaciones del Estudio 155

Referencias 158

Anexos 164

Anexo A – Repositorio del Proyecto 164

Anexo B – Manual de Usuario del Sistema 164

Resumen

Las noticias falsas presentan un problema creciente a nivel global en el ámbito digital, esto debido a que su información escandalizante permite una rápida viralización que supera a la de las verdaderas, influyendo en la opinión pública, la estabilidad social y la confianza en la información. A partir de esto, se plantea el desarrollo de un sistema autónomo para la detección de noticias falsas en español utilizando inteligencia artificial y técnicas de procesamiento de lenguaje natural y *machine learning*. El proyecto está pensado para utilizar *datasets* públicos. Así mismo, inicialmente se evaluarán modelos tradicionales, seguidos de modelos de aprendizaje profundo y, por último, modelos basados en *Transformers*. Los modelos serán evaluados con métricas adecuadas para identificar y seleccionar el de mejor desempeño. Finalmente, este modelo será implementado en una aplicación web mediante la integración con una *API*, permitiendo que el usuario interactúe ingresando su noticia y obteniendo un porcentaje de veracidad de esta.

Palabras clave: Noticias falsas, procesamiento de lenguaje natural, aprendizaje automático, aprendizaje profundo, transformers.

Abstract

Fake news is a growing global problem in the digital world. This is due to the fact that shocking information spreads rapidly, surpassing that of real news, influencing public opinion, social stability, and trust in information. Based on this, the development of an autonomous system for detecting fake news in Spanish is proposed, using artificial intelligence, natural language processing, and machine learning techniques. The project is designed to use public datasets. Likewise, traditional models will initially be evaluated, followed by deep learning models, and finally, Transformer-based models. The models will be evaluated with appropriate metrics to identify and select the best performer. Finally, this model will be implemented in a web application through integration with an API, allowing the user to interact by entering their news and obtaining a percentage of its veracity.

Keywords: Fake news, natural language processing, machine learning, deep learning, transformers.

Lista de Tablas

Tabla 1 Cronograma de actividades del proyecto 34

Tabla 2 Recursos necesarios para la ejecución del proyecto..... 35

Tabla 3 Resultados o productos esperados del proyecto 36

Tabla 4 Información General del Dataset..... 37

Tabla 5 Estructura y Calidad de Datos 37

Tabla 6 Balance, Idioma y Licencia 38

Tabla 7 Resumen comparativo de matrices de confusión — Validación y Prueba 96

Tabla 8 Métricas Principales de Rendimiento..... 98

Tabla 9 Métricas de Diagnóstico y Error Específico 99

Tabla 10 Métricas Avanzadas de Robustez 101

Tabla 11 Métricas de Eficiencia 102

Tabla 12 Modelos recomendados según propósito y tipo de desempeño 104

Tabla 13 Resumen comparativo de matrices de confusión — Transformers Validación y Prueba 114

Tabla 14 Métricas Principales de Rendimiento — Transformers 115

Tabla 15 Métricas de Diagnóstico y Error Específico — Transformers 116

Tabla 16 Métricas Avanzadas de Robustez — Transformers..... 117

Tabla 17 Métricas de Eficiencia - Transformers 118

Tabla 18 Métricas de modelos reentrenados 119

Tabla 19 Parámetros del mejor modelo de CNN 131

Tabla 20 Validación del Sistema con Múltiples Noticias..... 135

Lista de Figuras

Figura 1 Árbol causa-efecto del problema 15

Figura 2 Prototipo de la aplicación web - Home 29

Figura 3 Prototipo de la aplicación web - Resultado 30

Figura 4 Prototipo de la aplicación web - Home - Modo oscuro 30

Figura 5 Prototipo de la aplicación web - Resultado - Modo oscuro 31

Figura 6 Distribución porcentual de clases en el dataset (Pandas) 39

Figura 7 Conteo de valores nulos en el dataset (Pandas) 39

Figura 8 Gráfico circular de distribución de clases (Power BI) 40

Figura 9 Feature Statistics del Dataset (Orange Data Minning) 40

Figura 10 Flujo del Preprocesamiento de los datos 42

Figura 11 Distribución de Carpetas 44

Figura 12 Matriz de confusión - Validación - Regresión Logística 49

Figura 13 Matriz de confusión - Prueba - Regresión Logística 50

Figura 14 Comparación de métricas entre Validación y Prueba - Regresión Logística 51

Figura 15 Rendimiento por parámetros - Regresión Logística 52

Figura 16 Matriz de confusión - Validación - Máquinas de Vectores de Soporte 54

Figura 17 Matriz de confusión - Prueba - Máquinas de Vectores de Soporte 54

Figura 18 Comparación de métricas entre Validación y Prueba - Máquinas de Vectores de Soporte 55

Figura 19 Rendimiento por parámetros - Maquinas de Vectores de Soporte 56

Figura 20 Matriz de confusión - Validación - Naive Bayes 58

Figura 21 Matriz de confusión - Prueba – Naive Bayes 58

Figura 22 Comparación de métricas entre Validación y Prueba - Naive Bayes 59

Figura 23 Rendimiento por parámetros - Naive Bayes..... 60

Figura 24 Matriz de confusión - Validación - Arboles de Decisión..... 62

Figura 25 Matriz de confusión - Prueba – Arboles de Decisión..... 63

Figura 26 Comparación de métricas entre Validación y Prueba - Arboles de Decisión 64

Figura 27 Rendimiento por parámetros - Arboles de Decisión 65

Figura 28 Matriz de confusión - Validación - Random Forest 67

Figura 29 Matriz de confusión - Prueba – Random Forest..... 68

Figura 30 Comparación de métricas entre Validación y Prueba - Random Forest..... 69

Figura 31 Rendimiento por parámetros - Random Forest 70

Figura 32 Matriz de confusión - Validación - LSTM 74

Figura 33 Matriz de confusión - Prueba – LSTM..... 74

Figura 34 Comparación de métricas entre Validación y Prueba - LSTM..... 75

Figura 35 Evolución de métricas por épocas – LSTM 76

Figura 36 Matriz de confusión - Validación - BiLSTM..... 78

Figura 37 Matriz de confusión - Prueba – BiLSTM..... 78

Figura 38 Comparación de métricas entre Validación y Prueba - BiLSTM 79

Figura 39 Evolución de métricas por épocas – BiLSTM..... 80

Figura 40 Matriz de confusión - Validación - GRU..... 82

Figura 41 Matriz de confusión - Prueba – GRU..... 83

Figura 42 Comparación de métricas entre Validación y Prueba - GRU 84

Figura 43 Evolución de métricas por épocas – GRU..... 85

Figura 44 Matriz de confusión - Validación - CNN..... 87

Figura 45 Matriz de confusión - Prueba – CNN 88

Figura 46 Comparación de métricas entre Validación y Prueba - CNN 89

Figura 47 Evolución de métricas por épocas – CNN..... 90

Figura 48 Matriz de confusión - Validación - CNN + BiLSTM + GRU + Atención Simple 92

Figura 49 Matriz de confusión - Prueba - CNN + BiLSTM + GRU + Atención Simple 93

Figura 50 Comparación de métricas entre Validación y Prueba - CNN + BiLSTM + GRU + Atención Simple..... 93

Figura 51 Evolución de métricas por épocas – CNN + BiLSTM + GRU + Atención Simple ... 94

Figura 52 Matriz de confusión – Validación – mBERT..... 106

Figura 53 Matriz de confusión – Prueba – mBERT..... 107

Figura 54 Comparación de métricas entre Validación y Prueba – mBERT..... 108

Figura 55 Evolución de métricas por épocas – mBERT..... 109

Figura 56 Matriz de confusión – Validación – BETO 110

Figura 57 Matriz de confusión – Prueba – BETO 111

Figura 58 Comparación de métricas entre Validación y Prueba – BETO.....112

Figura 59 Evolución de métricas por épocas – BETO.....113

Figura 60 Pantalla de home de la Interfaz 129

Figura 61 Pantalla de resultado de la Interfaz..... 130

Figura 62 Validación de la Interfaz Web..... 132

Figura 63 Endpoints de la API..... 132

Figura 64 Validación del Endpoint de Predicción 133

Figura 65 Validación del Endpoint de Lista de Modelos..... 133

Figura 66 Validación del Endpoint de Métricas..... 134

Figura 67 Validación del Endpoint de Logs.. 134

Introducción

Las noticias falsas se han convertido en un problema frecuente en internet y en las redes sociales. Estas publicaciones engañosas tienen el potencial de modificar la manera en que las personas comprenden lo que ocurre y generar impactos negativos en diversas áreas, como desconfianza injustificada en alguna marca o producto, reconocimiento por logros o atribuciones que no corresponden, uso de medicamentos o tratamientos no diagnosticados, y malas tomas de decisiones por desinformación, incluyendo decisiones financieras que pueden terminar en estafas, pérdidas o incluso problemas de depresión. Un ejemplo de esto fueron las personas que, por creencias erróneas, tomaron malas decisiones e incluso llegaron a quitarse la vida en años como el 2012, cuando se afirmaba que sería el fin del mundo.

A esto se suma que las noticias falsas son difíciles de identificar y que hoy en día cualquiera puede publicar información asegurando que es verídica, sin que exista una forma sencilla de confirmarlo. En redes sociales no se necesita un título o estudios para afirmar que algo es cierto. Por esta razón, este proyecto propone la creación de una herramienta que ayude en esa tarea, ya que no siempre es fácil identificarlas.

El proyecto consiste en probar y comparar diferentes modelos de aprendizaje automático, desde los más básicos de *machine learning* hasta modelos más avanzados de *deep learning*, además de algunos *transformers*, para identificar cuál ofrece mejores resultados en la clasificación de noticias en español. Todos los modelos se entrenarán y se compararán entre sí en la versión que haya dado mejores resultados, para luego seleccionar uno. Con el modelo elegido se construirá una *API* que será consumida por una aplicación web sencilla, en la que el usuario podrá escribir una noticia y recibir un porcentaje que indique la probabilidad de si es real o falsa.

Con esto se busca ofrecer una solución práctica y también un aporte académico frente al problema de la desinformación digital.

Planteamiento del Problema

Según la UNAD, "*un estudio realizado por el Instituto Tecnológico de Massachusetts (MIT) reveló que las noticias falsas se propagan seis veces más rápido que las verdaderas en redes sociales*" (Marketing Investigación, s. f.). Esto ocurre por el aprovechamiento del algoritmo que manejan las redes, en donde lo escandaloso genera más visitas, ya que este tipo de contenido apela a emociones humanas como el miedo, la indignación, la esperanza o la rabia.

"*A partir de los 2000, con la popularización de las redes sociales, millones de personas comenzaron a compartir y consumir contenido sin un control editorial adecuado*" (Marketing Investigación, s. f.). Existen varias razones por las cuales hay interés en que se propaguen las noticias falsas: la manipulación intencional por parte de actores políticos, económicos o ideológicos; los algoritmos de las plataformas; intereses económicos; y la polarización política y la desconfianza en los medios tradicionales.

A esto se suma lo influyente que puede llegar a ser la información falsa por lo escandalosa que suena, y que muchas personas no confían en los medios oficiales. Se piensa colectivamente que estos buscan mantener al ciudadano común desinformado, por lo que cuando alguien cualquiera publica información asegurando que el gobierno intentó ocultarla, llama la atención de todos. Esto puede usarse para beneficiar o perjudicar a alguien en concreto, por ejemplo, aumentando o disminuyendo el precio de acciones de una empresa, o manipulando la bolsa según conveniencia.

Según Córdoba Cano (2025), "*un informe de la BBC, en Estados Unidos, indica que las noticias falsas pueden generar hasta 10.000 dólares solo por una publicación viral*" (Ministerio

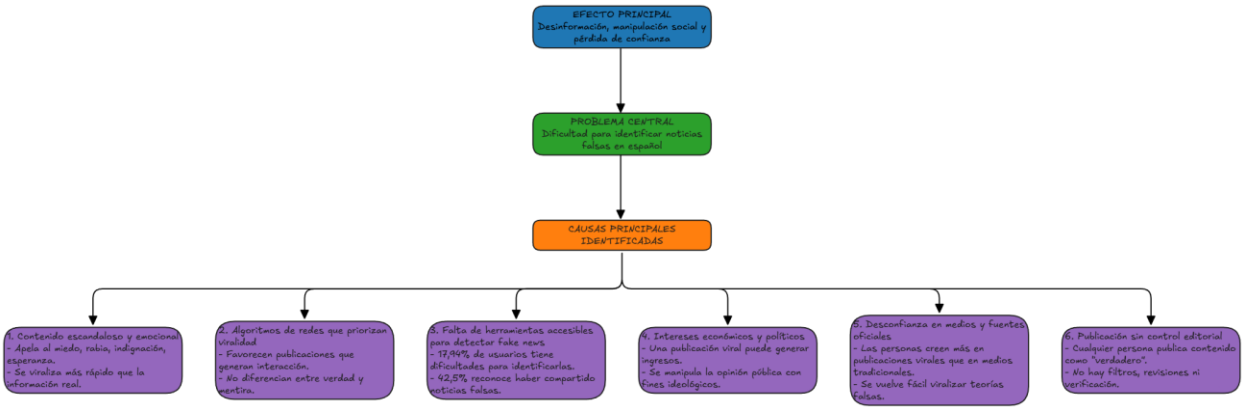
de Tecnologías de la Información y las Comunicaciones, citado en Córdoba Cano, 2025), lo cual hace atractiva su producción y distribución.

Además, de acuerdo con la encuesta realizada por Córdoba Cano (2025), el 17,94 % de los encuestados afirma tener dificultades para identificar información falsa; sin embargo, el 42,5 % reconoce haber compartido información falsa alguna vez. Esto evidencia la necesidad de desarrollar un sistema autónomo que, de forma automatizada, analice el contenido textual y permita identificar noticias falsas con un nivel de precisión aceptable, reduciendo así el impacto negativo que generan en la sociedad.

En consecuencia, surge la siguiente pregunta investigativa que orienta el desarrollo del presente proyecto:

¿Cómo desarrollar un sistema autónomo capaz de identificar noticias falsas en español, utilizando técnicas de procesamiento de lenguaje natural y modelos de *machine learning*, que alcance un nivel de precisión aceptable y permita su uso práctico mediante una *interfaz web*?

Figura 1
Árbol causa-efecto del problema



Nota. Hecho con Excalidraw. *Fuente.* Autoría propia.

Justificación

Las noticias falsas se han convertido en un problema digital a nivel global que cada vez toma más presencia. La rápida viralización de estas permite cambiar el juicio de masas en poco tiempo; esto se puede llegar a usar para desestabilizar a cualquiera, como una guerra digital, disminuyendo las ventas y generando desconfianza. Por lo tanto, se requieren soluciones tecnológicas desde la ingeniería de sistemas.

Este proyecto se compone de áreas clave como el procesamiento de lenguaje natural, el aprendizaje automático, la generación de una *API* para consumir servicios, el desarrollo web, las metodologías ágiles y las bases de datos, para el desarrollo de una herramienta digital que ayude al usuario a identificar noticias falsas. Además, la propuesta es viable al usar bases de datos y *datasets* públicos, junto con bibliotecas de *software* de código abierto.

Este trabajo pertenece a la línea de investigación de Ciencia de Datos y Sistemas Complejos, contribuyendo a la formación del ingeniero mediante un proyecto aplicado con análisis y solución de problemas a través de tecnologías digitales.

Este proyecto también podría generar impacto social al ofrecer a las personas una opción para verificar si las noticias que leen son verdaderas o falsas, lo que permitiría que no caigan tan fácilmente en información engañosa. Esto podría mitigar ligeramente la viralización de noticias falsas y contribuir a que las personas tomen mejores decisiones, tanto en cuestiones personales como financieras y políticas.

El proyecto se relaciona directamente con el Objetivo de Desarrollo Sostenible 16, que busca fortalecer la paz, la justicia y el funcionamiento de las instituciones. Contar con un sistema que ayude a reconocer información falsa contribuye a que el entorno digital sea más transparente

y menos propenso a la manipulación. Con ello se favorece la verificación ciudadana y se disminuye la circulación de contenidos engañosos que pueden generar efectos negativos a nivel social, político o económico.

Dado que la propuesta será desarrollada y evaluada en un entorno que imita condiciones reales, aunque aún no se ejecute en un ambiente productivo completo, se ubica dentro del nivel de madurez tecnológica TRL 5. En esta etapa, los componentes principales ya se integran y se prueban en un escenario representativo, lo que permite confirmar que el sistema funciona y que puede seguir avanzando hacia fases de implementación más avanzadas.

El trabajo también guarda coherencia con la misión de la UNAD, al utilizar tecnologías de la información para crear soluciones que aportan a la formación académica y al desarrollo social. La propuesta promueve el aprendizaje autónomo y el uso responsable de los recursos digitales, además de aportar una herramienta que responde a un problema actual como la desinformación en línea. Con esto se refuerza la labor educativa y de proyección social que caracteriza a la universidad.

Finalmente, al contar con un documento completo que recopila todo el proceso del proyecto, este podría servir como guía para ingenieros y estudiantes, permitiéndoles avanzar más rápido en futuros trabajos similares. También deja evidencia de cuál es el mejor algoritmo de *machine learning*, *deep learning* o *transformers* para el procesamiento de texto, y las tablas, gráficos y comparativas facilitarán a la comunidad académica la interpretación y análisis de los datos y métricas registradas.

Objetivos

Objetivo General

Desarrollar un sistema autónomo para la identificación de noticias falsas en español con una precisión aceptable, empleando técnicas de procesamiento de lenguaje natural y *machine learning*, con una *interfaz web* para su uso por parte del usuario.

Objetivos Específicos

Recolectar *datasets* públicos en español compuestos por noticias falsas y verdaderas previamente etiquetadas, aplicando procesos de preprocesamiento de texto.

Diseñar el modelo de detección de *fake news,* estableciendo la arquitectura necesaria para su entrenamiento y validación con métricas aceptables.

Implementar la *API* encargada de servir las predicciones del modelo para su consumo desde la *interfaz web*.

Construir la *interfaz web* que permita al usuario interactuar con el sistema de detección de noticias falsas.

Delimitación del Proyecto

El sistema autónomo de detección de noticias falsas en medios digitales utilizando inteligencia artificial es un proyecto en el que el sistema se delimita a detectar texto en español de noticias falsas, no manejará imágenes ni videos o audios, así como tampoco manejará texto en otros idiomas aparte del español. Este proyecto podría llegar a tener despliegue a producción en la plataforma Render, que lo permite de manera gratuita, aunque por tiempo limitado. El sistema contará con el modelo de *machine learning* entrenado para detección de noticias falsas en español, comparativa de los modelos de *machine learning*, *deep learning* y *Transformers*, *API* desarrollada con FastAPI, interfaz de usuario en React básica para la interacción del usuario, documentación técnica y académica del proyecto, y validación del sistema con noticias de prueba.

El sistema no contará con una interfaz de usuario muy trabajada y/o profesional; la API no estará desplegada tampoco, será solo para el uso de interacción con la interfaz. El repositorio estará público y se podrá usar todo lo anterior dicho. Este proyecto se realizará en un tiempo de 4 meses como máximo, en un entorno educativo, y se aplicará en un nivel académico.

Marco Conceptual

Las noticias falsas o *fake news*, según la UNICEF (*¿Cómo detectar «fake news» en Colombia?, 2022*), son anuncios sensacionalistas que aparentan tener veracidad periodística. Estas utilizan la saturación de información, fragmentos sacados de contexto, imágenes o logos de instituciones para dar veracidad, así como estadísticas, datos o publicaciones alteradas, generalmente con un tono escandalizante para llamar la atención y asegurar la viralidad de la información.

Para abordar este problema es necesario apoyarse en el Procesamiento de Lenguaje Natural (PLN), concepto que, según IBM (*Stryker & Holdsworth, 2025*), es un subcampo de la informática y la inteligencia artificial que, a partir del uso de técnicas como el *machine learning*, permite que las computadoras comprendan el lenguaje humano y se comuniquen a través de él. Para esto, se debe transformar el lenguaje natural en datos que la computadora pueda entender; una de las formas es mediante la *tokenización* del texto.

Dentro de estas técnicas se encuentra el *machine learning*, el cual es una rama de la inteligencia artificial. Es decir, el *machine learning* es inteligencia artificial, pero no toda la inteligencia artificial es *machine learning*. En esta rama, lo que se hace es entrenar un modelo para predecir un resultado esperado a partir de unos datos y sus etiquetas, para luego evaluarlo pasándole nuevos datos. La manera en la que se sabe si un modelo es adecuado es si sus predicciones tienen un puntaje bueno en métricas como precisión (*precision*) y *recall*. El *machine learning* se divide principalmente en modelos supervisados, no supervisados y por refuerzo.

Las métricas de evaluación permiten saber qué tan preciso es un modelo al hacer predicciones. Debido a los diferentes algoritmos de regresión y clasificación, así como a los

distintos fines por los que podría ser creado un modelo de *machine learning*, no siempre se puede confiar en una sola métrica. Por ejemplo, *accuracy* es una métrica que generalmente se usa como acompañante de las demás, ya que en modelos con datos desequilibrados puede dar un puntaje alto, pero al mismo tiempo cometer fallos del 100 % en las clases minoritarias. Para elegir la métrica correcta se suele utilizar una herramienta llamada *matriz de confusión*, que muestra los falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos. Esto permite identificar si el modelo falla más prediciendo positivos que eran negativos, o viceversa. A partir de esto, se debe pensar qué error sería más costoso; por ejemplo, en un caso de predicción de cáncer es más grave predecir un falso negativo, puesto que podría costarle la vida a un paciente.

Para entrenar uno de estos modelos se necesitan conjuntos de datos o *datasets*. Un *dataset* es un archivo que contiene los datos que se usarán para entrenar el modelo de *machine learning*. Estos archivos pueden ser de varios tipos, como *CSV*, *JSON*, *XML*, o también pueden provenir de bases de datos mediante el consumo de *APIs*. Los datos deben estar etiquetados, lo que quiere decir que deben tener una categoría. Por ejemplo, un *dataset* de celulares debe tener al menos la columna o etiqueta de "marca"; de este modo, se puede trabajar con ellos. Algunos *datasets* pueden contener datos corruptos, duplicados, sin formato fijo o nulos, por lo que se deben preprocesar los datos para evitar que el modelo haga malas predicciones.

Finalmente, se necesita una *API* para poder utilizar los datos procesados que generan los modelos. Una *API*, como lo indica su nombre, es una interfaz que facilita la conexión entre dos componentes de software a través de *endpoints*. La interfaz expone entradas a datos específicos para hacer más fácil la consulta, y los *endpoints* son esas entradas expuestas.

Marco Teórico

Estado del Arte

A partir de la investigación realizada, en la cual se revisaron artículos científicos (More, Jadhav & Yadav, 2021; M, S, Vn & S, 2025; Gálvez, Albores, Álvarez-González, Conde & Gálvez, 2024), así como repositorios como GitHub (nishitpatel01, 2023; alcorpas10, 2023) y repositorios de *datasets* como Kaggle o Hugging Face, sobre *datasets*, modelos o proyectos de detección de *fake news*, se encontró que se emplean algunos algoritmos tradicionales como SVM, Naive Bayes y Regresión Logística; modelos profundos como LSTM, BiLSTM, CNN y GRU; y *Transformers* como BETO y mBERT, los cuales alcanzaron niveles altos de precisión, especialmente en los modelos en inglés. Aunque existen cientos de conjuntos de datos en español con miles de registros cada uno, estos suelen tener limitaciones de clases equilibradas, sesgo o actualización, lo cual podría solucionarse uniendo uno o dos, aunque requiera normalización de esquemas, etiquetas y calidad. La mayoría de los trabajos o proyectos encontrados mantienen un enfoque experimental y no avanzan más allá de *notebooks*, al menos en español, ya que en inglés se encontraron proyectos profesionales muy completos, incluyendo APIs o interfaces. Por último, en español, aunque se ofrece la posibilidad de reutilizar los modelos en aplicaciones web, no se identificaron herramientas abiertas y accesibles al público.

Marco Jurídico

El desarrollo del proyecto está fundamentado en las normas nacionales e internacionales, esto enfocado en el tratamiento de datos, la gestión de la información digital, el uso de tecnologías emergentes y el uso ético de la IA:

Nacionales

- Ley 527 de 1999: "Por medio de la cual se define y reglamenta el acceso y uso de los mensajes de datos, del comercio electrónico y de las firmas digitales, y se establecen las entidades de certificación y se dictan otras disposiciones." (*Ley 527 de 1999 - Gestor Normativo*, s. f.)
- Ley 1341 de 2009: "Por la cual se definen principios y conceptos sobre la sociedad de la información y la organización de las Tecnologías de la Información y las Comunicaciones ?TIC?, se crea la Agencia Nacional de Espectro y se dictan otras disposiciones." (*Ley 1341 de 2009 - Gestor Normativo*, s. f.)
- Ley 1266 de 2008: "Ley estudiada por la Corte Constitucional mediante Sentencia C-1011 de 2008. Por la cual se dictan las disposiciones generales del hábeas data y se regula el manejo de la información contenida en bases de datos personales, en especial la financiera, crediticia, comercial, de servicios y la proveniente de terceros países y se dictan otras disposiciones." (*Ley 1266 de 2008 - Gestor Normativo*, s. f.)
- Ley 1581 de 2012: "Por la cual se dictan disposiciones generales para la protección de datos personales." (*Ley 1581 de 2012 - Gestor Normativo*, s. f.)
- Decreto 1377 de 2013: "Que mediante la Ley 1581 de 2012 se expidió el Régimen General de Protección de Datos Personales, el cual, de conformidad con su artículo 1°, tiene por objeto *"(...) desarrollar el derecho constitucional que tienen todas las personas*

a conocer, actualizar y rectificar las informaciones que se hayan recogido sobre ellas en bases de datos o archivos, y los demás derechos, libertades y garantías constitucionales a que se refiere el artículo 15 de la Constitución Política; así como el derecho a la información consagrado en el artículo 20 de la misma"." (*Decreto 1377 de 2013 - Gestor Normativo*, s. f.)

- Ley 1712 de 2014: "Por medio de la cual se crea la Ley de Transparencia y del Derecho de Acceso a la Información Pública Nacional y se dictan otras disposiciones." (*Ley 1712 de 2014 - Gestor Normativo*, s. f.)
- Conpes 3975 de 2019: "POLÍTICA NACIONAL PARA LA TRANSFORMACIÓN DIGITAL E INTELIGENCIA ARTIFICIAL" («Documento CONPES 3975 de 2019», s. f.)
- Resoluciones de MinTIC (2021 en adelante).

Internacionales

- GDPR – Unión Europea, 2018.
- Directrices de la UNESCO sobre Ética de la Inteligencia Artificial, 2021.
- Recomendación de la OCDE sobre IA, 2019.

Marco Tecnológico

- **Python:** El lenguaje de programación ampliamente utilizado para trabajar con datos e inteligencia artificial.
- **FastAPI:** Framework de Python que facilita la creación de *endpoints* para una API, con alta eficiencia.
- **Scikit-learn:** Librería de Python para el uso de *machine learning*; aunque ofrece algoritmos básicos, cuenta con buena documentación y una API intuitiva.
- **TensorFlow:** Biblioteca de código abierto desarrollada por Google para *machine learning* y *deep learning*, que permite crear y entrenar redes neuronales a gran escala.
- **Keras:** Biblioteca de código abierto que se ejecuta sobre TensorFlow, manejando una API más intuitiva y pensada para redes neuronales.
- **React:** Biblioteca de JavaScript para el desarrollo de interfaces de usuario, mantenida por Meta.

Metodología de la Investigación

Enfoque Metodológico

El proyecto sigue un enfoque cuantitativo, porque todo el análisis se basa en datos numéricos, métricas de evaluación y resultados obtenidos directamente del comportamiento de los modelos. No se usa percepción humana ni análisis subjetivo, sino medidas estadísticas para comparar el rendimiento de cada modelo.

Muestra y Población del Proyecto

En este caso, la población está formada por las noticias digitales en español que circulan en internet. Para el proyecto se tomó como muestra varios *datasets* públicos encontrados en *Kaggle* y *Hugging Face*, que ya venían clasificados como noticias verdaderas o falsas.

Después de limpiarlos y unirlos, se obtuvo un conjunto de datos de aproximadamente **50.159** registros, con un balance cercano al **59%** de noticias reales y **41%** de noticias falsas, lo cual es suficiente para entrenar y evaluar los modelos.

Construcción e Implementación del Instrumento de Medición y Recolección de los Datos

Los mismos *dataset* son en este caso el instrumento, pero deben ser los *dataset* ya procesados. Para dejarlos listos se hicieron estos pasos:

1. Se unieron cuatro *datasets* públicos compatibles.
2. Se *normalizaron* las *etiquetas* para trabajar con dos valores: **0** (falso) y **1** (verdadero).
3. Se limpiaron los textos: se pasaron a minúsculas, se eliminaron caracteres especiales, se normalizaron tildes, y también se quitaron duplicados, URLs y emojis.

4. Por último, se dividieron en tres conjuntos: entrenamiento (**70%**), validación (**15%**) y prueba (**15%**), cuidando que ambas clases quedaran equilibradas en cada subconjunto.

Con esto, los datos quedaron listos para ser usados como instrumento de medición en la parte de los modelos.

Análisis y Diagnostico del Proceso Investigativo

En la revisión de los datos se encontraron algunos problemas:

- Algunos *datasets* no tenían licencias de uso claras.
- Había duplicados y registros incompletos, que se eliminaron.
- Existía un ligero desbalance entre clases (más noticias reales que falsas).
- Cada *dataset* tenía estructuras distintas, por lo que tocó *normalizar* columnas y etiquetas.
- También se notó que hay un sesgo temporal, porque las noticias falsas cambian mucho con el tiempo.

Aun con esas limitaciones, los datos resultan adecuados para entrenar los modelos, siempre y cuando no se dependa solo de la métrica *accuracy*, sino también de otras como *recall* y *F1-score*.

Metodología de Desarrollo Tecnológico

Metodología Ágil

Para el desarrollo se tomaron elementos de ***Scrum***, organizando las actividades en entregas cortas y revisables. Se trabajó por *sprints*, priorizando primero la limpieza de datos y entrenamiento de modelos, luego la creación de la *API* y finalmente la interfaz web. Cada etapa se revisó al finalizar para ajustar lo necesario antes de continuar.

Análisis de Requerimientos

Funcionales

- El sistema debe permitir que un usuario escriba una noticia y reciba un porcentaje de veracidad.
- Debe integrar un modelo de IA previamente entrenado.
- Tiene que ofrecer una API en FastAPI para exponer los servicios del modelo.
- La interfaz web en React debe consumir esa API y mostrar los resultados.

No funcionales:

- El desarrollo se hará con librerías de código abierto como Scikit-learn, TensorFlow, Keras, FastAPI y React.
- El tiempo de respuesta no debería superar los 5 segundos para un texto promedio.
- El sistema debe poder desplegarse en plataformas gratuitas como Render o Hugging Face Spaces.
- Se tendrán en cuenta medidas básicas de seguridad, como validar entradas y sanitizar texto.

Diseño del Prototipo

El prototipo fue diseñado en *Excalidraw* e incluye dos vistas principales: la pantalla de inicio (home) y la pantalla de resultados. Se pensó tanto en modo oscuro como en modo claro para la interfaz.

En la pantalla de inicio se encuentra un campo de texto donde el usuario puede ingresar la noticia, seguido de un botón que permite analizarla. Al hacer clic, el sistema redirige a la pantalla de resultados.

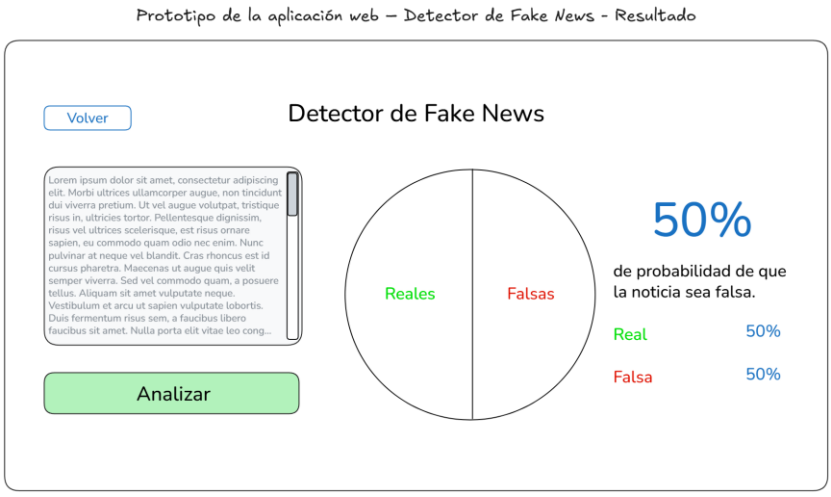
En esta vista, el usuario dispone de un botón para regresar al home, un campo de texto más pequeño con la noticia (editable para realizar cambios) y nuevamente el botón de análisis para evaluar una nueva entrada. Además, se presenta un gráfico circular que muestra el porcentaje de veracidad o falsedad, acompañado de un indicador numérico destacado con el porcentaje correspondiente y una breve explicación que interpreta el resultado.

Figura 2
Prototipo de la aplicación web - Home

Nota. Hecho en Excalidraw. *Fuente.* Autoría propia.

Figura 3

Prototipo de la aplicación web - Resultado



Nota. Hecho en Excalidraw. *Fuente.* Autoría propia.

Figura 4

Prototipo de la aplicación web - Home - Modo oscuro

Nota. Hecho en Excalidraw. *Fuente.* Autoría propia.

Figura 5
Prototipo de la aplicación web - Resultado - Modo oscuro

Nota. Hecho en Excalidraw. *Fuente.* Autoría propia.

Fases del Desarrollo Tecnológico

El proyecto se realizará siguiendo una metodología aplicada a 5 fases:

- **Recopilación y preprocesamiento de los datos:** Se seleccionará uno o varios conjuntos de datos de plataformas como Kaggle o Hugging Face, para posteriormente ser preprocesados, incluyendo normalización y limpieza de datos, y por último ser divididos en 3 conjuntos: entrenamiento, validación y prueba.
- **Diseño y entrenamiento de modelos:** Se comenzará entrenando modelos básicos como SVM o Regresión Logística, y así establecer una línea de comparativa. Luego se usarán modelos de aprendizaje profundo como LSTM o GRU; por último, se usarán Transformers preentrenados en español como BETO o mBERT para hacer una evaluación como alternativas avanzadas.

- **Evaluación de resultados:** Se realizará una medición del rendimiento de cada uno de los modelos con métricas como *Accuracy*, *Precisión*, *Recall* y matriz de confusión, para luego realizar una comparativa de todos los modelos y seleccionar el de mejor desempeño.
- **Desarrollo de la aplicación web:** Se creará una API con FastAPI para exponer los *endpoints* del modelo entrenado. Se desarrollará la interfaz interactiva en React que consumirá la API y permitirá que el usuario ingrese una noticia y reciba un porcentaje de veracidad de esta.
- **Pruebas y validación del sistema:** Se realizará una validación interna con datos de prueba, verificando que la página funcione correctamente y se realizará documentación de resultados y limitaciones del sistema.

ODS y TRL

ODS

Puesto que el proyecto busca fortalecer la confianza en la información disponible en internet, reduciendo la viralización de noticias falsas, evitando la desinformación que puede llevar a situaciones como pánico, competencia desleal y manipulación política, el proyecto se articula en el ODS 16 – Paz, Justicia e Instituciones Sólidas.

TRL

Considerando que el proyecto será desarrollado y validado en condiciones que simulan el entorno, aunque aún no estará desplegado para probarse en el entorno productivo, el proyecto estará focalizado en el TRL 5 – Validación de los sistemas, subsistemas o componentes en un entorno relevante.

Cronograma

Tabla 1

Cronograma de actividades del proyecto

Consecutivo	Actividad	Detalle	Mes de inicio	Mes final	% del proyecto
1	Recopilación de datos	Búsqueda, selección y descarga de *datasets* en Kaggle y Hugging Face.	Agosto 2025	Agosto 2025	10 %
2	Preprocesamiento de datos	Limpieza, normalización, eliminación de duplicados y división en entrenamiento/validación/prueba.	Agosto 2025	Septiembre 2025	15 %
3	Diseño y entrenamiento de modelos	Entrenamiento de modelos tradicionales (SVM, Regresión Logística) y de aprendizaje profundo (LSTM, GRU).	Septiembre 2025	Octubre 2025	20 %
4	Evaluación de resultados	Evaluación de métricas (Accuracy, Precisión, Recall, F1, matriz de confusión) y selección del mejor modelo.	Octubre 2025	Octubre 2025	15 %
5	Implementación con Transformers	Ajuste y pruebas con BETO y mBERT para comparar resultados avanzados.	Octubre 2025	Noviembre 2025	10 %
6	Desarrollo de API	Creación de endpoints con FastAPI para exponer el modelo entrenado.	Noviembre 2025	Noviembre 2025	10 %
7	Desarrollo de interfaz web	Creación de interfaz en React para interacción con el usuario.	Noviembre 2025	Diciembre 2025	10 %
8	Pruebas y validación del sistema	Verificación de funcionamiento, documentación de resultados y limitaciones.	Diciembre 2025	Diciembre 2025	10 %

Nota. La tabla presenta el cronograma de actividades del proyecto, detallando las fases principales del desarrollo del sistema de detección de noticias falsas, el periodo de ejecución de cada actividad y el porcentaje de avance estimado dentro del proyecto total. *Fuente.* Autoría propia.

Recursos Necesarios

Tabla 2

Recursos necesarios para la ejecución del proyecto

Consecutivo	Tipo de Recurso	Descripción	Presupuesto
1	Bibliografía	Artículos científicos, papers indexados (Scopus, IEEE, SSRN), repositorios académicos (UNAD, MIT, etc.), documentación oficial de librerías (TensorFlow, FastAPI, React).	$ 0.00
2	Equipo Humano	1 estudiante (ejecutor del proyecto), asesor asignado por la UNAD.	$ 0.00
3	Equipos y Software	Computador portátil personal. Software: Python 3.x, FastAPI, React, Scikit-learn, TensorFlow, Keras, pandas, GitHub, VS Code.	$ 0.00
4	Materiales y suministros	Conexión estable a internet, libreta de apuntes, servicios en la nube gratuitos (Google Colab, Hugging Face Spaces, GitHub Pages).	$ 0.00

Nota. La tabla describe los recursos requeridos para la ejecución del proyecto, incluyendo bibliografía, equipo humano, herramientas de software, equipos tecnológicos y materiales de apoyo, junto con el presupuesto estimado para cada uno. *Fuente.* Autoría propia.

Resultados o Productos Esperados

Tabla 3
Resultados o productos esperados del proyecto

Consecutivo	Resultado o producto esperado	Indicador	Beneficiario
1	Modelo de machine learning entrenado para detección de fake news en español	Precisión y recall superiores al 90% en el conjunto de prueba	Comunidad académica, usuarios interesados en IA
2	Comparativa de modelos (tradicionales, profundos y Transformers)	Documento con tabla de métricas (accuracy, precisión, recall, F1) por modelo	Investigadores, estudiantes de ingeniería
3	API desarrollada con FastAPI para exponer el modelo	API funcional con endpoints disponibles para consulta	Desarrolladores y usuarios finales
4	Interfaz web en React para la interacción del usuario	Página web que permita ingresar una noticia y obtener el porcentaje de veracidad	Público general, usuarios de internet
5	Documentación técnica y académica del proyecto	Informe final con metodología, resultados, limitaciones y conclusiones	Universidad, docentes, futuros investigadores
6	Validación del sistema con noticias de prueba	Registro de pruebas y resultados de validación interna	Estudiante (autor) y comunidad académica

Nota. La tabla presenta los resultados y productos esperados del proyecto, los indicadores asociados para su verificación y los beneficiarios directos de cada entregable, evidenciando el impacto académico, tecnológico y social del sistema propuesto. *Fuente.* Autoría propia.

Desarrollo del Proyecto

Recopilación de Datos

Se realizó la búsqueda de *datasets* en las plataformas **Hugging Face** y **Kaggle**, usando palabras clave como *"fake news spanish"*, *"noticias falsas español"* y *"spanish news classification"*.

A partir de la búsqueda se encontraron cinco *datasets*. De ellos se eligieron cuatro para ser unidos y normalizados. El *dataset* de **Arsenii Tretiakov** fue descartado debido a que no tiene licencia clara.

A continuación, se presentan las tablas con la información recolectada:

Tabla 4
Información General del Dataset

Nombre	Autor/Organización	Fuente	Cantidad de Registros
FakeNewsSpanish_Kaggle2	Sebastián Ayala Ruano	Hugging Face	598
fake_news_corpus_spanish	María Grandury	Hugging Face	572
Spanish Political Fake News	Javier Otero Vizoso	Kaggle	57,231
Spanish Fake and Real News	Fabricio A. Zules	Kaggle	598
noticias falsas en español	Arsenii Tretiakov	Kaggle	2,000

Nota. La tabla resume la información general de los conjuntos de datos utilizados en el proyecto, incluyendo el nombre del *dataset*, autor u organización responsable, fuente de obtención y cantidad de registros disponibles para el entrenamiento y evaluación de los modelos. *Fuente.* Autoría propia.

Tabla 5
Estructura y Calidad de Datos

Nombre	Columnas	Duplicados	Nulos
FakeNewsSpanish_Kaggle2	texto, clase	0	0
fake_news_corpus_spanish	ID, CATEGORY, TOPICS, SOURCE, HEADLINE, TEXT…	0	0
Spanish Political Fake News	ID, Label, Titulo, Descripcion, Fecha	462	0
Spanish Fake and Real News	texto, clase	0	0
noticias falsas en español	class, Text	42	0

Nota. La tabla muestra la estructura y calidad de los *datasets* analizados, detallando las columnas disponibles, la presencia de registros duplicados y valores nulos, aspectos clave para el preprocesamiento y la confiabilidad de los datos utilizados. *Fuente.* Autoría propia.

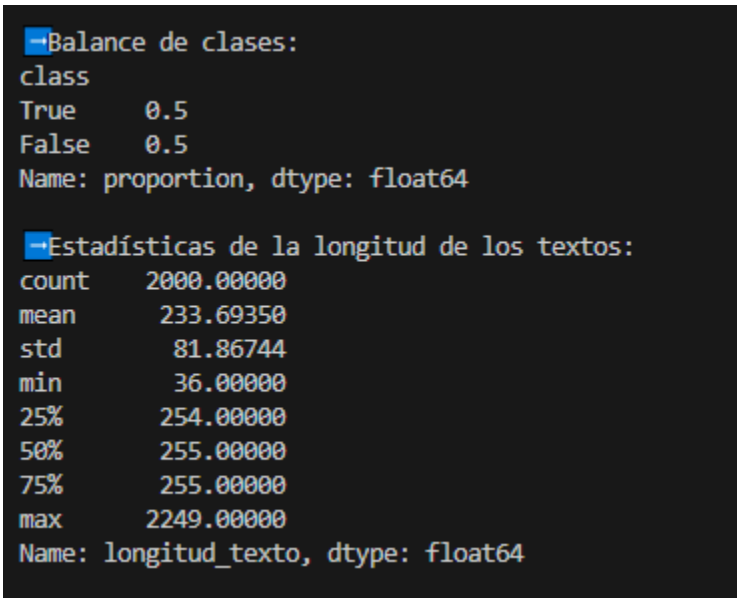
Tabla 6
Balance, Idioma y Licencia

Nombre	Balance	Idioma	Formato	Clases	Última actualización	Licencia
FakeNewsSpanish_Kaggle2	56,7% / 43,3%	Español	CSV	2	22/03/2022	Creative Commons Attribution Non Commercial Share Alike 4.0
fake_news_corpus_spanish	50% / 50%	Español	CSV	2	15/09/2024	Creative Commons Attribution 4.0
Spanish Political Fake News	58% / 42%	Español	CSV	2	00/00/2023	Attribution 4.0 International (CC BY 4.0)
Spanish Fake and Real News	56,7% / 43,3%	Español	CSV	2	00/00/2018	CC BY-SA 4.0
noticias falsas en español	50% / 50%	Mixto	CSV	2	00/00/2020	Unknown

Nota. La tabla presenta información relacionada con el balance de clases, idioma, formato, número de clases, fecha de última actualización y tipo de licencia de los *datasets* empleados, permitiendo evaluar su idoneidad y condiciones de uso dentro del proyecto. *Fuente.* Autoría propia.

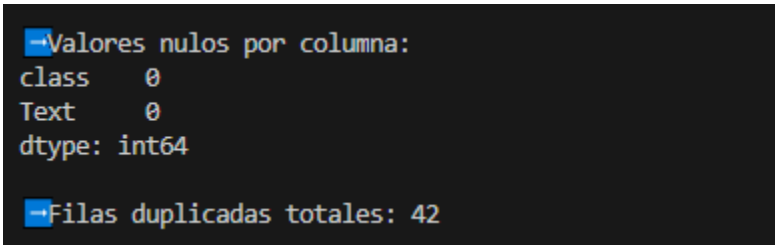
También se utilizaron las herramientas **Power BI**, **Orange Data Mining** y **pandas** para revisar los *datasets*:

Figura 6
Distribución porcentual de clases en el dataset (Pandas)



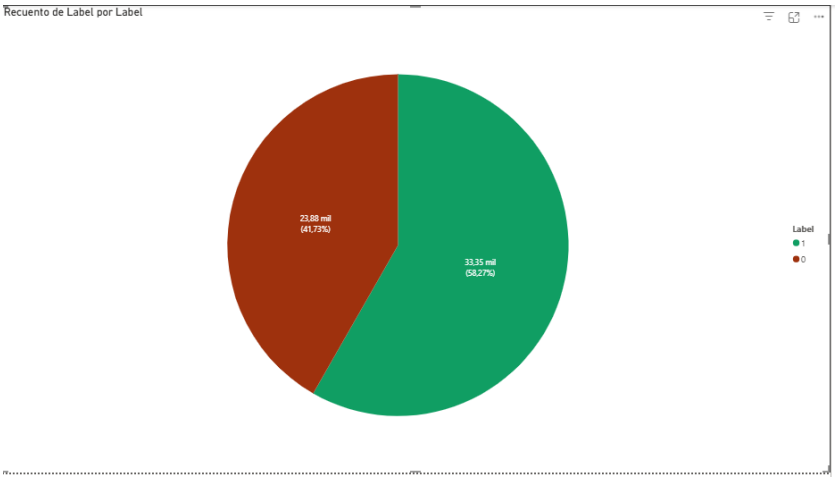
Nota. Hecho con Pandas en Visual Studio Code. *Fuente.* Autoría propia.

Figura 7
Conteo de valores nulos en el dataset (Pandas)



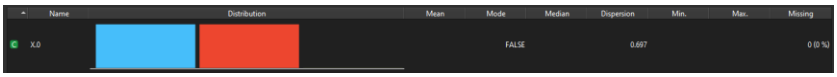
Nota. Hecho con Pandas en Visual Studio Code. *Fuente.* Autoría propia.

Figura 8
Gráfico circular de distribución de clases (Power BI)



Nota. Hecho con Power BI. *Fuente.* Autoría propia.

Figura 9
Feature Statistics del Dataset (Orange Data Minning)



Nota. Hecho con Power BI. *Fuente.* Autoría propia.

Enlaces a los Datasets revisados:

- FakeNewsSpanish_Kaggle2 – Hugging Face
- fake_news_corpus_spanish – Hugging Face
- Spanish Political Fake News – Kaggle
- Spanish Fake and Real News – Kaggle
- noticias falsas en español – Kaggle

Preprocesamiento de Datos

Se planeó el proceso de preprocesamiento de datos con 9 pasos en total. Este paso es de suma importancia, ya que, aunque suene solo a técnico o por convención, se ha demostrado que influye en la precisión del modelo. Según *Camacho-Collados y Pilehvar (2018)*, en su análisis de técnicas de preprocesamiento de texto en el ámbito de NLP antes de entrenar un modelo de Deep Learning, este paso impacta directamente en el rendimiento de modelos neuronales como CNN y LSTM. En sus resultados obtuvieron una diferencia de un 2,4 % en *accuracy*, lo cual prueba que sí influye. Aun así, no todas las técnicas dieron resultados positivos en todos los modelos; algunos solo aumentaban la complejidad sin demostrar una mejora significativa, por lo que se debe planear un proceso acorde al contexto del modelo.

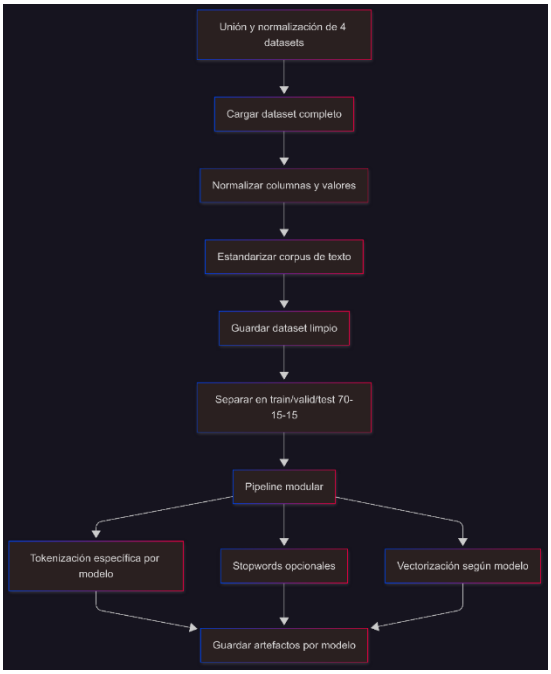
Estos son los pasos que se utilizarán para el preprocesamiento de los datos:

1. Al haberse elegido no usar un solo dataset, sino unir 4 de ellos para disminuir el sesgo de una sola categoría de las noticias, se debe realizar la unión y normalización de los 4 datasets. Esto se hará dejando únicamente las columnas *texto* y *clase*. Dentro de *clase* se cambiarán sus valores para que únicamente tengan 0 = *fake* y 1 = *real*. Este dataset se guardará en formato Parquet, ya que es más veloz y liviano que CSV.
2. Se cargará el dataset completo desde Parquet para aprovechar su velocidad.
3. Se normalizarán columnas y valores básicos: se eliminarán duplicados y registros vacíos, se estandarizará la codificación de texto a UTF-8 y se homogeneizarán las clases 0 y 1.
4. Estandarización del texto a nivel corpus: se pasará a minúsculas, se eliminarán caracteres extraños, espacios adicionales, se normalizarán las tildes y se eliminarán etiquetas HTML.
5. Se guardará el dataset con el corpus limpio para permitir una tokenización más flexible, ya que se usará como base para varios modelos.

6. Se separarán los datos en conjuntos de *train*, *validation* y *test* usando la división 70/15/15, aprovechando los más de 50 000 registros.
7. Se realizará un *pipeline* modular. De esta manera se podrá configurar cada modelo como mejor convenga, aprovechando las distintas formas de tokenizar, quitar *stopwords* y vectorizar. Esto es importante, ya que para algunos modelos es beneficioso hacerlo, mientras que para otros no. Como se usará Scikit-learn, TensorFlow y Transformers, cada uno tendrá su mejor manera de hacerlo.
8. Se guardará cada objeto preprocesado asociado a cada modelo, para usarlos fácilmente en la fase posterior de entrenamiento.

La siguiente imagen muestra el flujo del preprocesamiento de datos:

Figura 10
Flujo del Preprocesamiento de los datos



Nota. Hecho en Mermaid. *Fuente.* Autoría propia.

Durante cada paso del preprocesamiento, se guardaron los datos en un archivo *Parquet* para tener claro, en caso de fallos, en qué paso ocurría. En el proceso de unificar los *datasets* se guardó el archivo *"fake_news_unificado.parquet"* en la carpeta *"raw"*, y se consiguió un *dataset* con **58.427 registros**. Luego, al realizar la normalización y estandarización, se generó el archivo *"fake_news_estandarizado.parquet"* en la carpeta *"processed"*.

Principalmente, se realizó una eliminación de valores nulos y duplicados, eliminación de textos vacíos y estandarización en UTF-8. Después de estas acciones, los registros se redujeron de 58.427 a **50.159**. En la estandarización, básicamente se aplicó una conversión a minúsculas, eliminación de caracteres especiales, reducción de espacios múltiples a solo uno, normalización de tildes, eliminación de etiquetas HTML, sustitución de *URLs*, menciones, *hashtags*, emojis y números por *tokens* especiales, además de un filtrado de textos con menos de 3 palabras. Luego de este proceso, se mantuvieron los 50.159 registros.

El *dataset* presenta un desbalance, quedando **59% de los registros como reales y 41% como falsos**, lo que indica que deberán usarse técnicas como la estratificación. Sin embargo, este desbalance no es un inconveniente crítico, ya que, para considerarse grave, la clase minoritaria debería representar un 20% o menos con respecto a la mayoritaria *(Abdelhamid & Desai, 2024)*.

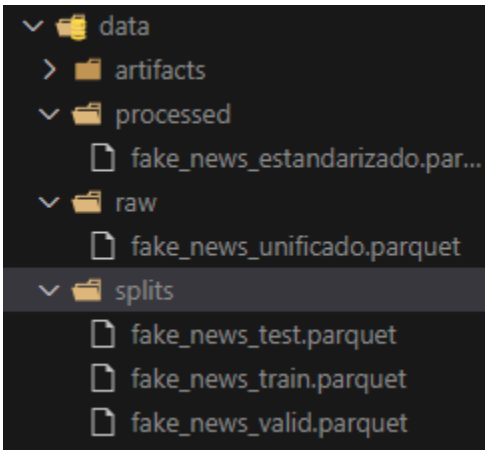
Posteriormente, se dividió el *dataset* estandarizado usando la función *train_test_split* de *scikit-learn*. Se dividió primero en *X_train***,** *y_train* con un 70% de los datos y *X_temp***,** *y_temp* con un 30%. Esto se hizo para luego dividir los *temp* en *test* y *valid*, cada uno con el 50%. Todas las divisiones usaron estratificación. Luego, se unieron las *X* con sus ***y*** respectivas para guardarlos en 3 *datasets*.

Por último, se guardaron en la carpeta *splits* con los nombres *fake_news_train.parquet*, *fake_news_valid.parquet* y *fake_news_test.parquet*. Cada archivo cuenta con la siguiente

cantidad de registros: *train* con **35.511 registros**, *valid* y *test* con **7.611** cada uno. El equilibrio de clases se mantiene cercano en los 3 *datasets*, con un **59% de datos reales** y un **41% de datos falsos**.

Se decidió que el paso 7 se realizara durante la fase de **Diseño y Entrenamiento de Modelos**, debido a que, de esta manera, se hace más modular la *tokenización* y, dado que ***Keras*** permite crear *redes neuronales* con una capa de *tokenización*, es mejor hacerlo así. Esta fue la distribución de carpetas y los archivos creados durante esta fase:

Figura 11
Distribución de Carpetas



Nota. Tomada en el entorno de desarrollo del proyecto en Visual Studio Code. *Fuente.* Autoría propia.

Diseño y Entrenamiento de Modelos

Para los modelos tradicionales que servirán como base, porque son rápidos de entrenar y comparar, se decidió usar *Scikit-Learn* por su sencillez e interfaz intuitiva para implementarlos. Los modelos elegidos son: *Regresión Logística*, *Máquinas de Vectores de Soporte (SVC)*, *Naive Bayes*, *Árboles de Decisión* y *Random Forest*. Para los modelos de *Deep Learning*, en los cuales se entra en terreno de *redes neuronales recurrentes* y *convolucionales* que suelen ser más pesados que los modelos tradicionales, pero capturan mejor la semántica, se usará *TensorFlow* con la *API* de *Keras*. Los modelos elegidos son: *LSTM*, *BiLSTM*, *GRU*, *CNN* y modelos híbridos como *CNN + LSTM* o *CNN + BiLSTM*. En esta fase no se entrenarán aún *Transformers*, debido a que este tipo de modelos debe dársele una prioridad mayor a los demás, ya que están preparados específicamente para el manejo de texto. Aunque la evaluación de resultados se realizará en una fase posterior, en esta se hará una evaluación de cada modelo sin compararlo con los demás, con el objetivo de configurar y usar los parámetros más adecuados para cada uno.

Modelos Tradicionales

Regresión Logística. Para el modelo de regresión logística, lo primero que se hizo fue cargar los *parquets* de los *datasets*, guardando su información dentro de variables. Posteriormente, se utilizó un *pipeline* para la vectorización y creación del modelo, ya que este facilita la integración y optimización en conjunto de la vectorización y el modelo. Esto permite evitar fugas de datos y asegura la reproducibilidad.

Para la vectorización se usó *TF-IDF*, lo que en lugar de simples conteos penaliza palabras demasiado frecuentes, resalta términos más relevantes para cada clase y permite capturar combinaciones de palabras. También se usó una técnica llamada *stopwords*, la cual consiste en omitir palabras muy frecuentes en un idioma que no aportan información semántica relevante

para los modelos, tales como *el*, *la*, *de*, *por*. Al eliminarlas, el modelo reduce el ruido y mejora su discriminación entre noticias falsas y reales. Se utilizaron las *stopwords* pertenecientes a la librería *stopwordsiso*.

Para la creación del modelo se configuraron parámetros como:

- ***max_iter = 5000***, lo que asegura la convergencia en un *dataset* con tantos registros.
- ***tol = 1e-4***, nivel de tolerancia de convergencia; un valor bajo exige mayor precisión antes de detener el entrenamiento.
- ***class_weight = "balanced"***, que al igual que *stratify* en *train_test_split*, ayuda a mitigar el desequilibrio en las clases ajustando automáticamente los pesos para evitar favorecer a la clase mayoritaria.
- ***n_jobs = -1***, permite que el entrenamiento use todos los núcleos disponibles del procesador, mejorando el rendimiento.
- ***random_state***, que se repetirá a lo largo de este documento, ya que permite establecer una semilla para garantizar la reproducibilidad.
- ***memory = "cache"***, que permite almacenar en caché los pasos intermedios del *pipeline*, acelerando entrenamientos repetidos.

Posteriormente, para obtener los mejores resultados del modelo, se usó una técnica llamada ***búsqueda en cuadrícula*** o *Grid Search*, en la cual se definen parámetros a probar y se generan todas las combinaciones posibles para encontrar las que den mejor rendimiento. Sin embargo, como debe probar demasiadas combinaciones, su complejidad es muy alta y toma demasiado tiempo.

Por lo tanto, en este proyecto se optó por usar *RandomizedSearch*, que en lugar de probar todas las combinaciones posibles selecciona aleatoriamente un número limitado de ellas. Esto

explora un espacio más amplio de valores sin necesidad de probarlos todos, reduciendo el tiempo de ejecución. En este caso se dejó en **75** la cantidad de combinaciones.

Para el *scoring* se asignó *f1_weighted*, indicándole a *RandomizedSearch* que evaluara cada combinación con *F1-score*. Más adelante se explicará la importancia de esta métrica.

Junto con *RandomizedSearch* se usó otra técnica llamada ***Validación Cruzada*** o *Cross Validation*. Esta se utiliza para confirmar que no haya fuga de información y que el modelo sea bueno prediciendo datos desconocidos, y no solo los que ve en el entrenamiento. De *Cross Validation* existen varios métodos; en este caso se usó ***K-Fold***, que divide los datos de entrenamiento en *K* conjuntos más pequeños para validar el modelo. Aquí se asignó un ***3-fold***, por lo que cada combinación de *hiperparámetros* de *RandomizedSearch* se validó con 3 conjuntos de datos que, aunque es un poco bajo ya que normalmente se recomienda un valor entre 5 y 10, se hizo de esta manera para mantener un tiempo adecuado de entrenamiento.

Posteriormente, en la evaluación de todos los modelos creados por *RandomizedSearch*, se guarda para su posterior uso en un archivo llamado *logistic_regression_best_model.pkl*. Para evitar sobrescribir el archivo, también se verifica si ya existe. En caso de que exista, se compara si el que ya está guardado tiene mejor puntaje que el encontrado con *RandomizedSearch*. Si es así, el archivo se deja tal cual está; en caso contrario, se reemplaza por el del nuevo modelo. De esta forma se asegura que, aunque se aumente la cantidad de combinaciones de parámetros que puede probar *RandomizedSearch*, siempre se mantenga el mejor de todos los intentos.

Por último, se evalúa el mejor modelo, comenzando con una herramienta muy útil llamada ***matriz de confusión***, la cual permite ver en 4 cuadrantes el rendimiento del modelo en las predicciones. Cada uno de los cuadrantes constituye el porcentaje de los datos que se ubica en esa posición. Como se observa en la **Figura 3**, los dos cuadrantes superiores representan el

porcentaje total de las noticias reales, mientras que los dos cuadrantes inferiores representan el porcentaje total de las noticias falsas.

Cada dato recibe uno de los siguientes nombres dependiendo de en qué cuadrante caiga:

- **Verdaderos Positivos (VP):** datos reales que el modelo predijo como reales (predicción correcta).
- **Verdaderos Negativos (VN):** datos falsos que el modelo predijo como falsos (predicción correcta).
- **Falsos Positivos (FP):** datos falsos que el modelo predijo como reales (error).
- **Falsos Negativos (FN):** datos reales que el modelo predijo como falsos (error).

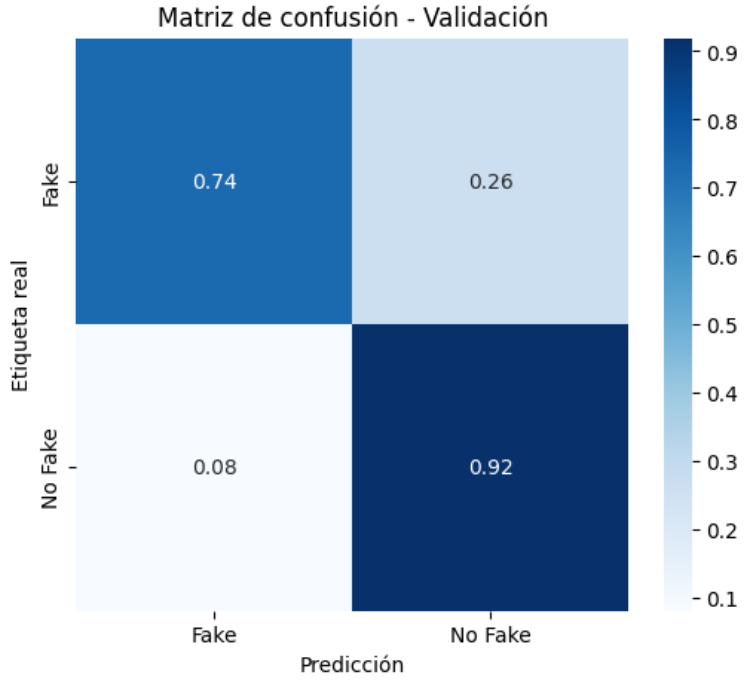
Como se observa en la **Figura 12**, el mejor modelo de *Regresión Logística* obtuvo un **74% de acierto** detectando las noticias falsas y un **92% de acierto** detectando las noticias reales, en el conjunto de validación. Esto equivale a un **26% de error** en noticias falsas y un **8% de error** en noticias reales.

En la **Figura 13**, el mismo modelo obtuvo un **72% de acierto** en noticias falsas y un **92% de acierto** en noticias reales, en el conjunto de prueba. Esto da un **28% de error** en noticias falsas y un **8% de error** en noticias reales, lo que indica que el modelo no es del todo malo generalizando para identificar noticias falsas, aunque comparando su rendimiento con la capacidad de detección de noticias reales que tiene, la detección de noticias falsas se vuelve mediocre.

Otra de las ventajas de la matriz de confusión es que, a partir de los cuadrantes de error, se puede detectar qué clase le cuesta más identificar. Con ello, se podría tomar una decisión: mejorar los parámetros para que generalice mejor ambas clases, o bien hacer que generalice mejor una clase para reducir el porcentaje de error más crítico. En este caso, es más perjudicial

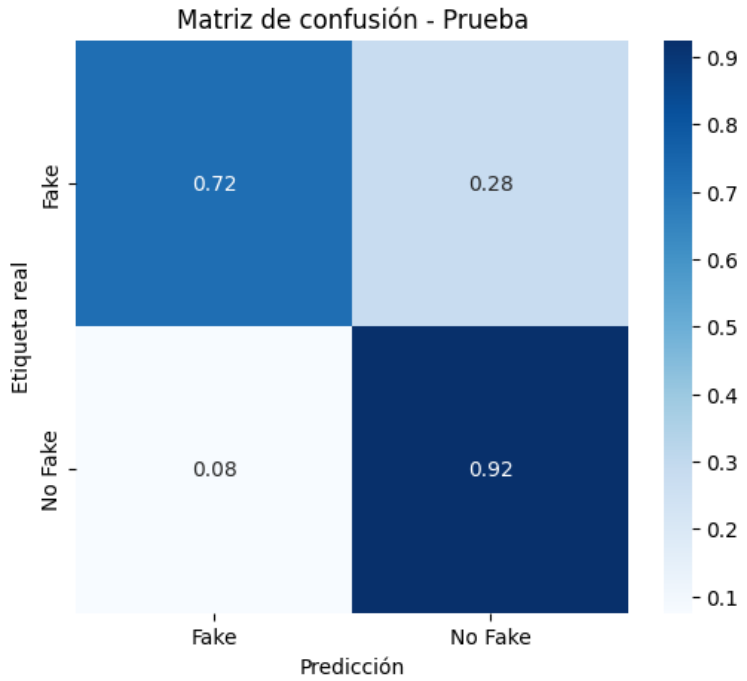
que el modelo prediga como verdadera una noticia que en realidad es falsa, que marcar como falsa una noticia verdadera. Esto se debe a que el objetivo principal es identificar las noticias falsas; por lo tanto, que una noticia verdadera sea clasificada como falsa no trae mayores inconvenientes, mientras que una noticia falsa con datos peligrosos que puedan desinformar sí representa un riesgo. Por ello, en este caso, es mejor **priorizar la disminución de los falsos positivos**.

Figura 12
Matriz de confusión - Validación - Regresión Logística



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Figura 13
Matriz de confusión - Prueba - Regresión Logística



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

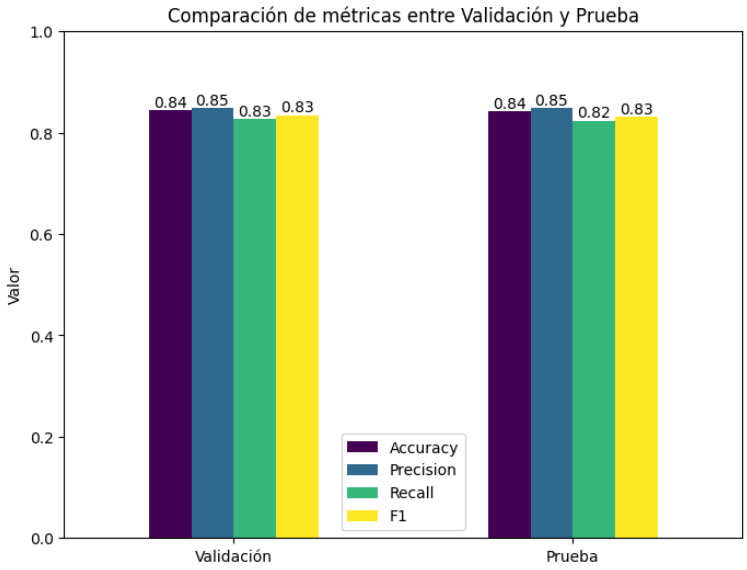
Es por esta misma razón que se seleccionó ***F1-Score (f1_weighted)*** como métrica en el *scoring* de *RandomizedSearch*, ya que promedia el desempeño de cada clase según su proporción en el *dataset*. De esta manera, aunque da mayor peso a la clase mayoritaria (noticias reales), también tiene en cuenta los errores en la clase minoritaria, lo que lo hace más equilibrado que usar métricas como la *exactitud*.

También se usaron otras métricas para evaluar el modelo. Como puede verse en la **Figura 14**, tanto en el conjunto de datos de validación como en el de prueba, los puntajes son similares.

- **Accuracy**: aunque es la métrica más sencilla de interpretar, ya que da el porcentaje de predicciones correctas sobre el total de datos evaluados, tiene un problema en conjuntos de datos desbalanceados, pues puede ocultar la mala generalización en las clases minoritarias.

- **Precision**: mide el porcentaje de predicciones positivas que fueron correctas. Es decir, si el modelo predijo 20 noticias como falsas y solo 15 lo eran, el porcentaje sería 75%. Esta métrica impide que el modelo aumente su precisión ignorando las clases minoritarias.
- **Recall**: es la contraparte de *Precision*, ya que mide el porcentaje de elementos correctos detectados sobre el total real. Por ejemplo, si había 25 noticias falsas y el modelo detectó 15, el porcentaje sería 60%.
- **F1-Score**: combina *Precision* y *Recall* en un balance armónico, lo que lo convierte en una métrica más robusta cuando existe desbalance en las clases.

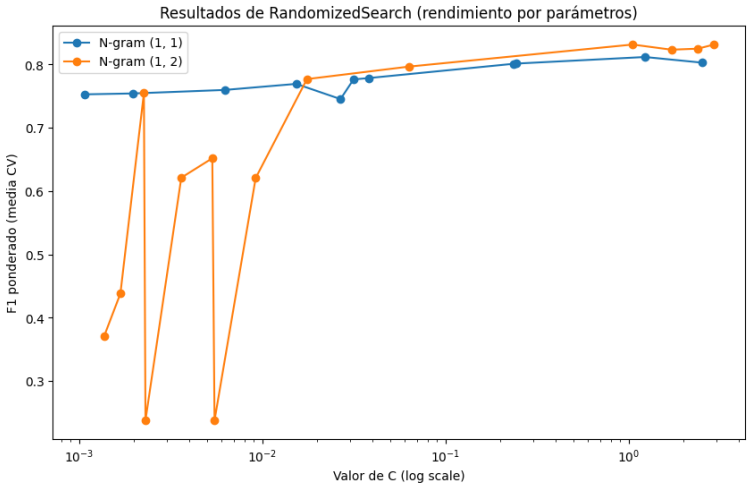
Figura 14
Comparación de métricas entre Validación y Prueba - Regresión Logística



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Por último, se puede observar en la **Figura 15** cómo iba modificándose el *score* del modelo en cada combinación de parámetros. Aunque en algunos puntos cayó en picado, es posible que esto se deba a que no contaba con suficientes iteraciones para converger. El mejor modelo obtuvo un *score* de **0.83143**, lo que indica que no es un mal modelo; sin embargo, no cumple con las expectativas que se requieren para el modelo final.

Figura 15
Rendimiento por parámetros - Regresión Logística



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Máquinas de Vectores de Soporte. Para el modelo de *Máquinas de Vectores de Soporte* se siguió un flujo similar al del modelo de *Regresión Logística*; sin embargo, hubo algunas modificaciones.

Al principio se planteó usar *SVC* con *kernel linear* y *rbf*, pero al hacer pruebas se obtuvo que la duración de todo el entrenamiento podría superar los tres días. Por lo tanto, se modificó por *LinearSVC*, ya que este está optimizado y escala mucho mejor en datos con alta dimensionalidad, como es el texto en este caso. Efectivamente, el tiempo de entrenamiento se redujo a menos de cinco minutos.

Posteriormente, se usó un *pipeline* tal como en el modelo de Regresión Logística, empleando *TF-IDF* como *vectorizador* junto con la clase para crear el modelo. Los parámetros fueron similares: *max_iter=5000*, *class_weight="balanced"*, *random_state=42* y *memory="__cache__"*. La única diferencia fue que al *pipeline* se le agregó el *escalador MaxAbsScaler*, debido a que *SVC* es sensible a la escala de los datos en las *representaciones vectoriales*.

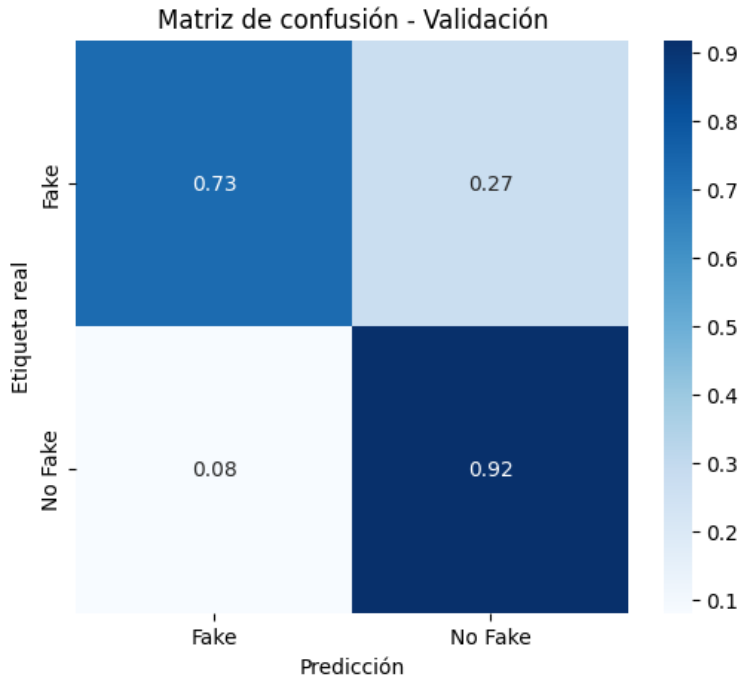
Luego, para obtener mejores resultados, también se usó la técnica de búsqueda en cuadrícula y validación cruzada con tres pares de parámetros. Además, para no recorrer todas las combinaciones, se utilizó *RandomizedSearchCV* con **75** combinaciones y validación cruzada de 3-folds. El *scoring* se estableció en *"f1_weighted"*. El sistema revisa si existe un modelo previo y compara cuál de los dos es mejor, guardando únicamente el óptimo.

Por último, este modelo también fue evaluado con las mismas métricas que la Regresión Logística, empezando por la *matriz de confusión*. Como se observa en la **Figura 16**, el mejor modelo de *Máquinas de Vectores de Soporte* obtuvo un **73%** de acierto en noticias falsas y un **92%** de acierto en noticias reales, en el conjunto de validación. Esto equivale a un **27%** de error en noticias falsas y un **8%** en noticias reales.

En la **Figura 17**, el mismo modelo obtuvo un **71%** de acierto en noticias falsas y un **92%** en noticias reales, en el conjunto de prueba. Esto da un **29%** de error en noticias falsas y un **8%** en noticias reales. Estos resultados indican que, al igual que el modelo de Regresión Logística, no es del todo malo generalizando para identificar noticias falsas, aunque de nuevo si se compara con su rendimiento en la detección de noticias reales este se vuelve mediocre y, a simple vista, parece ligeramente peor que el de Regresión Logística.

Figura 16

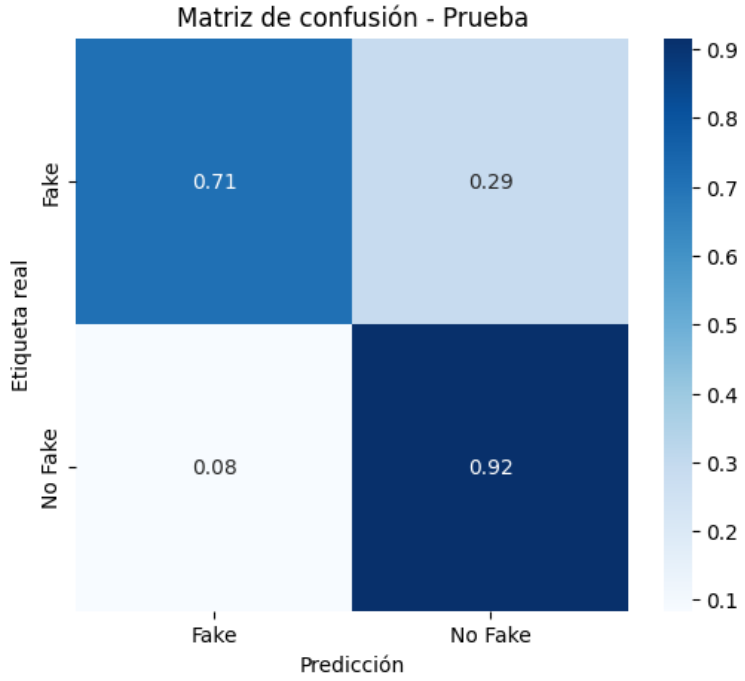
Matriz de confusión - Validación - Máquinas de Vectores de Soporte



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Figura 17

Matriz de confusión - Prueba - Máquinas de Vectores de Soporte

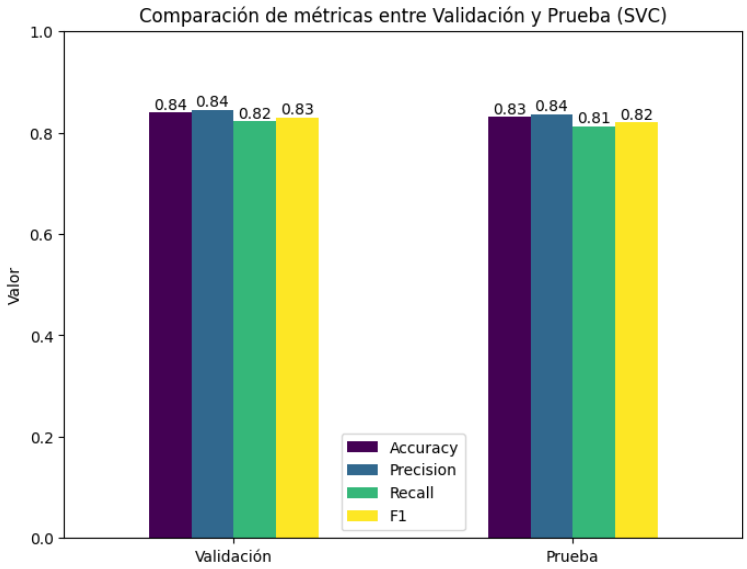


Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 18**, son similares, aunque inferiores a las del modelo de Regresión Logística, presentando una mayor diferencia en el conjunto de prueba.

Figura 18

Comparación de métricas entre Validación y Prueba - Máquinas de Vectores de Soporte



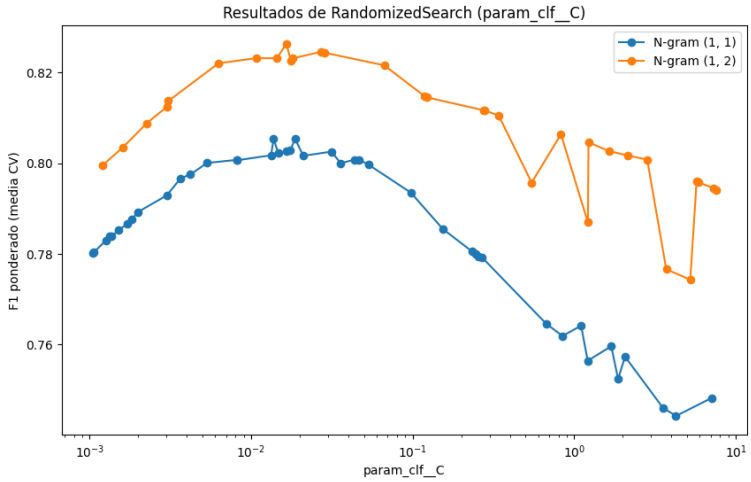
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Por último, en la **Figura 19** se puede observar cómo el *score* se modificaba con cada combinación de parámetros. A diferencia del gráfico de Regresión Logística, este modelo es más estable, no presenta caídas repentinas y parece requerir menos para *converger*. Sin embargo, parece ser más eficiente con conjuntos de datos pequeños y revisando palabras sueltas (no en parejas), por lo que no es del todo óptimo para este caso de uso.

El mejor modelo de Máquinas de Vectores de Soporte obtuvo un *score* de **0.82634**, lo que indica que no es un mal modelo, pero queda por debajo del modelo de Regresión Logística.

Figura 19

Rendimiento por parámetros - Maquinas de Vectores de Soporte



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Naive Bayes. Para el modelo de *Naive Bayes* se siguió el mismo flujo que en los modelos anteriores. En este caso se usó *MultinomialNB*, ya que es un algoritmo muy empleado en la clasificación de texto, debido a que trabaja bien con conteos y frecuencias. El tiempo de entrenamiento de este modelo fue menor que el de los dos anteriores: es más rápido tanto en entrenamiento como en predicción.

Posteriormente, se utilizó el *pipeline* que se ha usado desde el primer modelo. Se empleó *TF-IDF* como *vectorizador*, junto con la clase *(MultinomialNB)* para crear el modelo. A diferencia de los otros, a este no se le pasaron parámetros directamente. Al pipeline se le pasó *memory="__cache__"*. Tampoco fue necesario agregarle un *escalador*, como con *SVC*, ya que este modelo no es tan sensible a la escala de los datos en las *representaciones vectoriales*.

Después, como en los demás modelos, para obtener mejores resultados se usó la técnica de *búsqueda en cuadrícula* y *validación cruzada* con tres pares de parámetros. Este modelo, a diferencia de los anteriores, utiliza el parámetro *alpha* para adaptar sus pesos. Para este valor se aplicó una *distribución log-uniforme* entre **0.001** y **10**. También, para no probar todas las

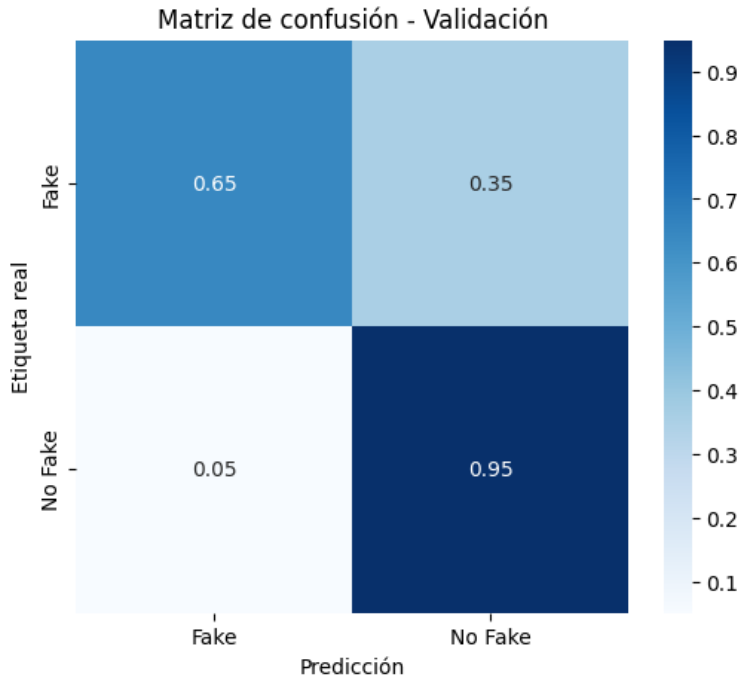
combinaciones, se utilizó *RandomizedSearchCV* con **75** combinaciones y *validación cruzada* de 3-folds. El *scoring* se estableció en *"f1_weighted"*. El sistema revisa si existe un modelo previo y compara cuál de los dos es mejor, guardando únicamente el más óptimo.

Este modelo fue evaluado de la misma forma y con las mismas *métricas* que los dos anteriores, empezando por la *matriz de confusión*. Como se observa en la **Figura 20**, el mejor modelo de *Naive Bayes* obtuvo un **65%** de acierto en noticias falsas y un **95%** en noticias reales, en el conjunto de validación. Esto equivale a un **35%** de error en noticias falsas y un **5%** en noticias reales.

En la **Figura 21**, el mismo modelo obtuvo un **64%** de acierto en noticias falsas y un **95%** en noticias reales, en el conjunto de prueba. Esto da un **36%** de error en noticias falsas y un **5%** en noticias reales. Estos resultados indican que, al igual que los modelos anteriores, no es del todo malo generalizando para identificar noticias falsas, pero incluso queda por debajo en un margen del **8%**. Sin embargo, su rendimiento en la detección de noticias falsas si se compara con su rendimiento en la detección de noticias reales se vuelve mediocre y también inferior al de los anteriores en la detección de noticias verdaderas, ya que el rendimiento de las noticias reales supera a los anteriores con un margen cercano al **3%**. A simple vista, parece tanto mejor detectando noticias reales como peor detectando noticias falsas.

Figura 20

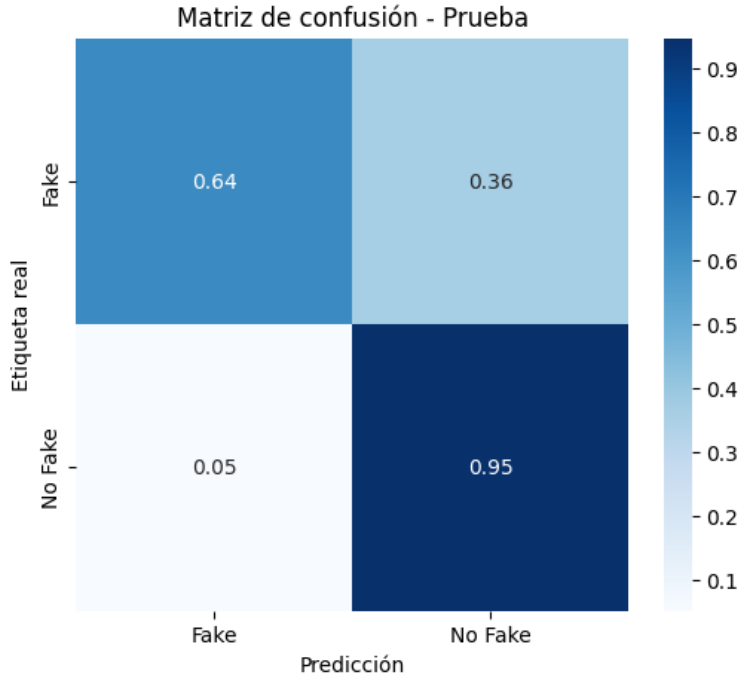
Matriz de confusión - Validación - Naive Bayes



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Figura 21

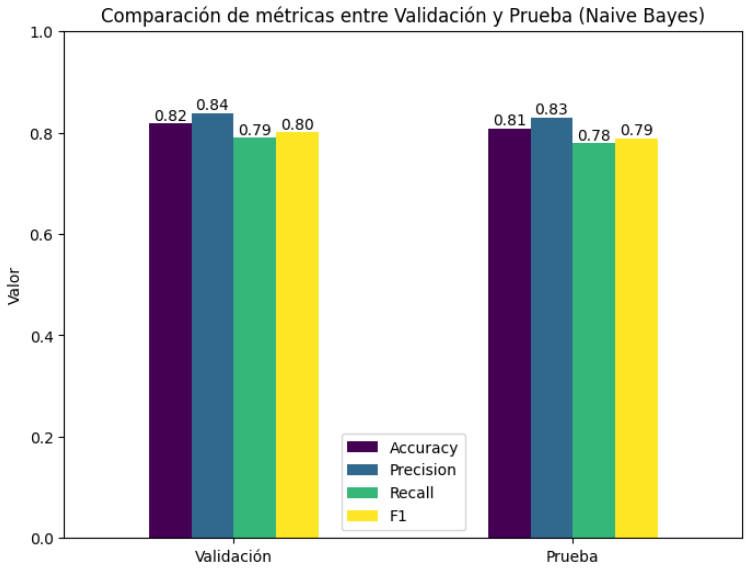
Matriz de confusión - Prueba – Naive Bayes



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás *métricas* usadas para evaluar el modelo, como se observa en la **Figura 22**, son similares a las del modelo de *SVC*, aunque inferiores. La mayor diferencia es de un **3%**, presente en *Recall* y *F1* tanto en el conjunto de validación como en el de prueba.

Figura 22
Comparación de métricas entre Validación y Prueba - Naive Bayes



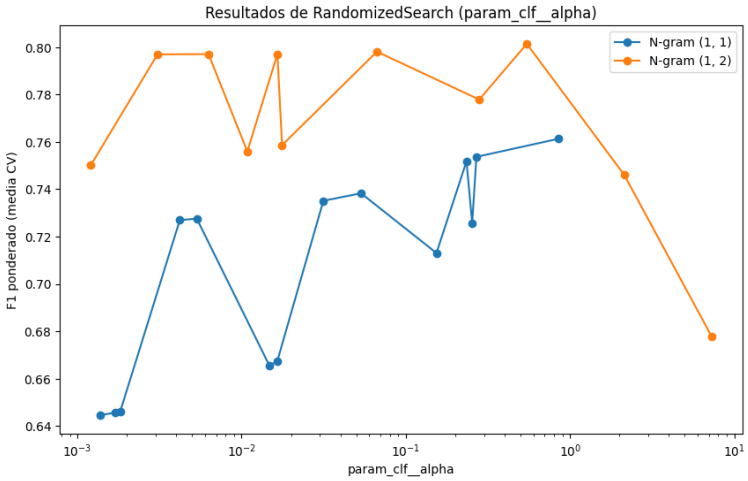
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Por último, en la **Figura 23** se puede observar cómo el *score* se modificaba con cada combinación de parámetros y cambios en los valores de *alpha*. Este modelo parece ser más estable que el de *Regresión Logística*, pero menos estable que el de *SVC*, ya que presenta caídas más pequeñas que las de *Regresión Logística*, pero más frecuentes que las de *SVC*. Por lo tanto, parece requerir menos que *Regresión Logística* y más que *SVC* para converger.

Sin embargo, este modelo resulta más eficiente con conjuntos de datos grandes que *SVC*, y también revisando palabras sueltas (al menos al inicio). Al final, la línea de *bigramas* parece crecer mientras que la de *unigramas* decae, lo que sugiere que con un poco más de entrenamiento podría mejorar sus resultados. El mejor modelo de *Naive Bayes* obtuvo un *score* de **0.81046**, lo que lo ubica por debajo de los modelos anteriores. No obstante, este es el puntaje

general, pero como ya se mencionó antes, para este caso de uso nos beneficia más obtener una mayor precisión prediciendo noticias falsas que una buena generalización en ambas clases, aunque esto implique una disminución en el *score*. Por esta razón, este modelo no está del todo descartado.

Figura 23
Rendimiento por parámetros - Naive Bayes



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Arboles de Decisión. Para el modelo de *Árboles de Decisión* se siguió el mismo flujo que en los modelos anteriores. En este caso se usó *DecisionTreeClassifier*, ya que este es un problema de clasificación más que de regresión. El tiempo de entrenamiento de este modelo fue intermedio: no fue superior al del modelo de *Máquinas de Vectores de Soporte*, pero tampoco fue menor que el del modelo de *Naive Bayes*.

Después se utilizó el *pipeline* que se ha venido usando desde el primer modelo. Se empleó *TF-IDF* como *vectorizador* y las *stop words* de la librería *stopwordsiso*, junto con la clase (*DecisionTreeClassifier*) para crear el modelo. A este modelo se le pasaron parámetros básicos directamente, como *class_weight="balanced"*, la semilla con *random_state=42*, y

finalmente *memory="__cache__"*. No fue necesario agregarle un *escalador*, ya que este modelo no es tan sensible a la escala de los datos en las *representaciones vectoriales*, como lo sería *SVC*.

Después, al igual que en los demás modelos y con la intención de obtener los mejores resultados, se usó la técnica de *búsqueda en cuadrícula* y *validación cruzada*, aunque esta vez con seis pares de parámetros para optimizar el modelo. Los parámetros nuevos fueron:

- ***clf__criterion***, que define cómo se mide la impureza de los *nodos*.
- ***clf__max_depth***, crucial para controlar el *sobreajuste*.
- Valores mínimos de muestras por división y por hoja, que ayudan a regular el crecimiento del árbol.

Para la profundidad máxima se usó *randint*, que define una *distribución discreta uniforme de números enteros*, y se hizo para que tomara números entre **5** y **50**. Este modelo, a diferencia de los demás, no tiene un parámetro que ajuste directamente sus pesos como *C* en *Regresión Logística* o *alpha* en *Naive Bayes*; más bien se rige por el nivel de profundidad que alcanza.

Para no probar todas las combinaciones de parámetros, se utilizó *RandomizedSearchCV* con **75** combinaciones y *validación cruzada* de *3-folds*. El *scoring*, igual que en las ocasiones anteriores, se estableció en *"f1_weighted"*. El sistema revisa si ya existe un modelo previo y compara cuál de los modelos de *Árboles de Decisión* es mejor, guardando únicamente el más óptimo.

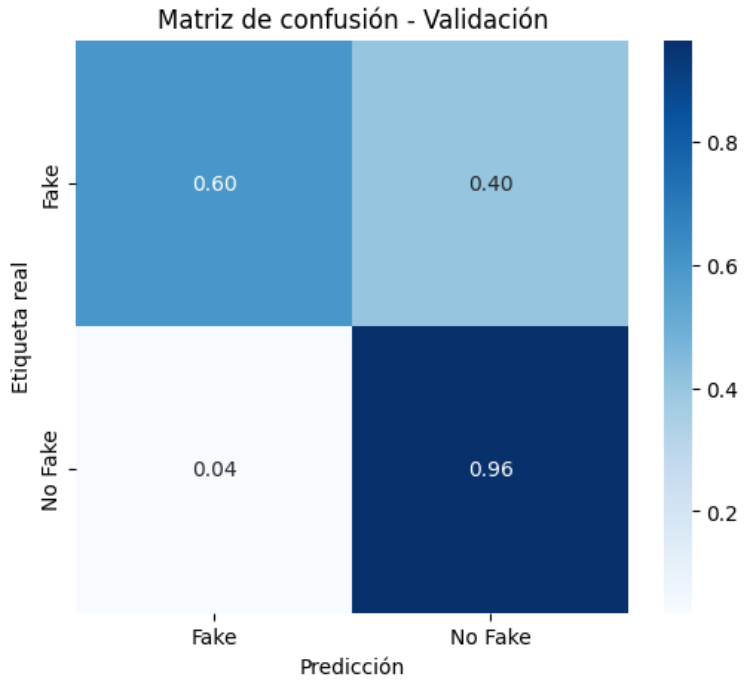
Este modelo fue evaluado individualmente de la misma forma y con las mismas métricas que los modelos anteriores, empezando por la *matriz de confusión*. Como se observa en la **Figura 24**, el mejor modelo de *Árboles de Decisión* obtuvo un **60%** de acierto en noticias falsas

y un **96%** en noticias reales en el conjunto de validación. Esto equivale a un **40%** de error en noticias falsas y un **4%** en noticias reales.

En la **Figura 25**, el mismo modelo obtuvo un **60%** de acierto en noticias falsas y un **96%** en noticias reales en el conjunto de prueba. Esto da, al igual que en validación, un **40%** de error en noticias falsas y un **4%** en noticias reales. Estos resultados indican que, al igual que los modelos anteriores, no es del todo malo generalizando para identificar noticias falsas.

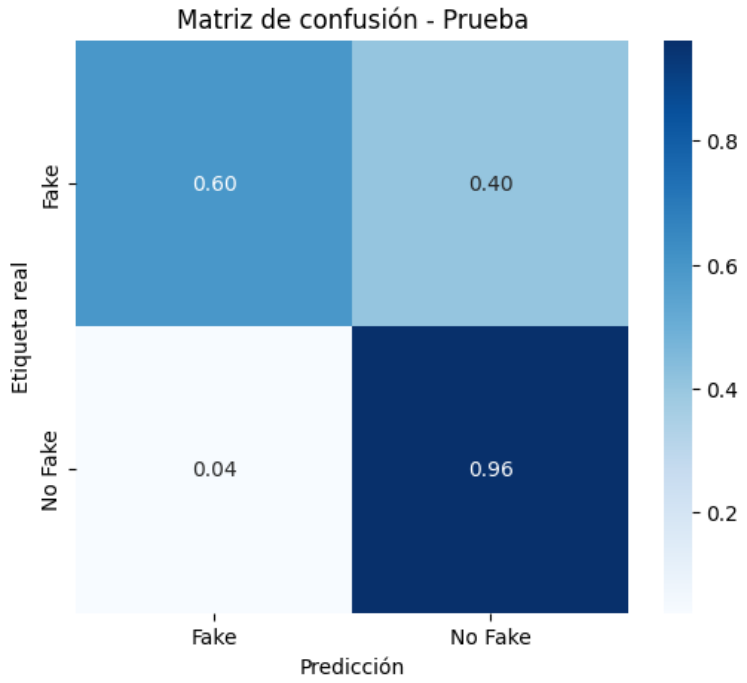
Presenta resultados similares a los de *Naive Bayes*, aunque incluso lo supera en detección de noticias reales, poniéndose en primer lugar como el modelo que mejor generaliza y predice este tipo de noticias. Sin embargo, su rendimiento en la detección de noticias falsas es mediocre e inferior al de los anteriores, convirtiéndose en el peor modelo en la predicción de noticias falsas. A simple vista, parece tanto el mejor detectando noticias reales como el peor detectando noticias falsas.

Figura 24
Matriz de confusión - Validación - Arboles de Decisión



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Figura 25
Matriz de confusión - Prueba – Arboles de Decisión

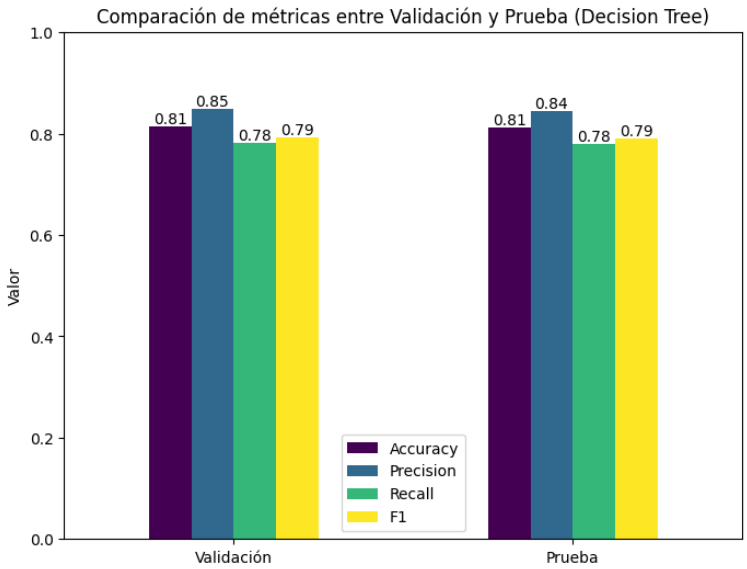


Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 26**, son similares a las del modelo de *Naive Bayes*. Ambos mantienen un equilibrio en el puntaje de las métricas: algunas coinciden, en otras un modelo es superior y en otras inferior, pero la mayor diferencia no supera el **1%**.

Figura 26

Comparación de métricas entre Validación y Prueba - Arboles de Decisión



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

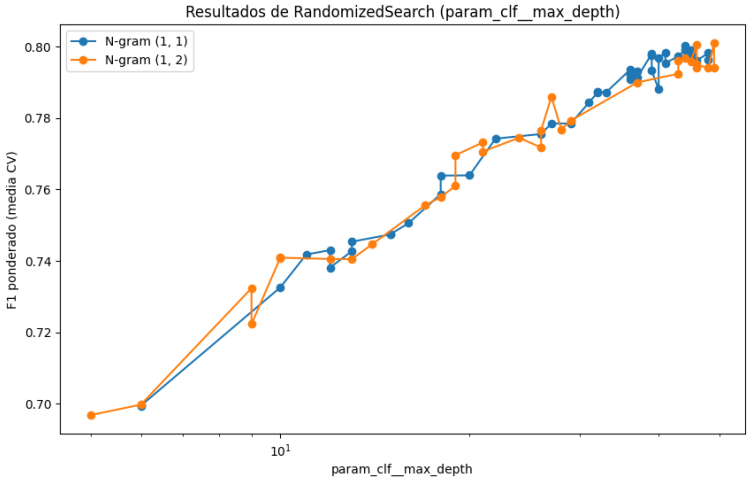
Por último, en la **Figura 27** se puede observar cómo el *score* se modificaba con cada combinación de parámetros y cambios en los valores de profundidad. Este modelo parece ser el más estable hasta el momento, ya que presenta caídas pequeñas y poco frecuentes, y su tendencia es ascendente. Con un mayor tiempo de entrenamiento podría obtener mejores resultados. Por lo tanto, parece requerir incluso menos que *SVC* para *converger*.

Este modelo no parece tener problemas con conjuntos de datos pequeños ni grandes. También maneja sin inconvenientes y casi en igualdad de condiciones tanto los *unigramas* como los *bigramas*. El mejor modelo de *Árboles de Decisión* obtuvo un score de **0.80097**, lo que lo ubica al final, por debajo de los modelos anteriores.

No obstante, este es el puntaje general, y al igual que con *Naive Bayes* debería analizarse más adelante si este modelo vale la pena usarlo sobreponiendo su rendimiento en detección de noticias falsas por encima del *score* general. Aunque tiene peores resultados que *Naive Bayes* en

ese aspecto, podría estar cayendo en prácticas incorrectas como priorizar la predicción de noticias reales. Por esta razón, este modelo no está del todo descartado.

Figura 27
Rendimiento por parámetros - Arboles de Decisión



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Random Forest. Como es costumbre, para el modelo de *Random Forest* se siguió el mismo flujo que con los modelos anteriores. En este caso se usó *RandomForestClassifier*. *Random Forest* es un algoritmo que no es más que un conjunto de *Árboles de Decisión* unidos entre ellos para no caer en el sobreajuste de un único *Árbol de Decisión*. El tiempo de entrenamiento de este modelo fue alto, poniéndose como el más demorado hasta el momento.

Posteriormente se utilizó el *pipeline*, usando *TF-IDF* como *vectorizador*, junto con las *stop words* y la clase *RandomForestClassifier* para crear el modelo. A este modelo se le pasaron los mismos parámetros básicos directamente que a los del modelo de *Árboles de Decisión*. Al igual que aquel, tampoco fue necesario agregarle un *escalador*.

Con la intención de obtener los mejores resultados, a este modelo también se le aplicó la técnica de *búsqueda en cuadrícula* y *validación cruzada*. Sin embargo, a diferencia del modelo

de *Árboles de Decisión*, en lugar de seis pares de parámetros, esta vez se usaron siete. Los parámetros nuevos fueron:

- ***clf__n_estimators*** = *randint(100, 500)*, que permite establecer el número de árboles que se unirán.
- **clf__max_features**, que controla cuántas características se consideran en cada división.

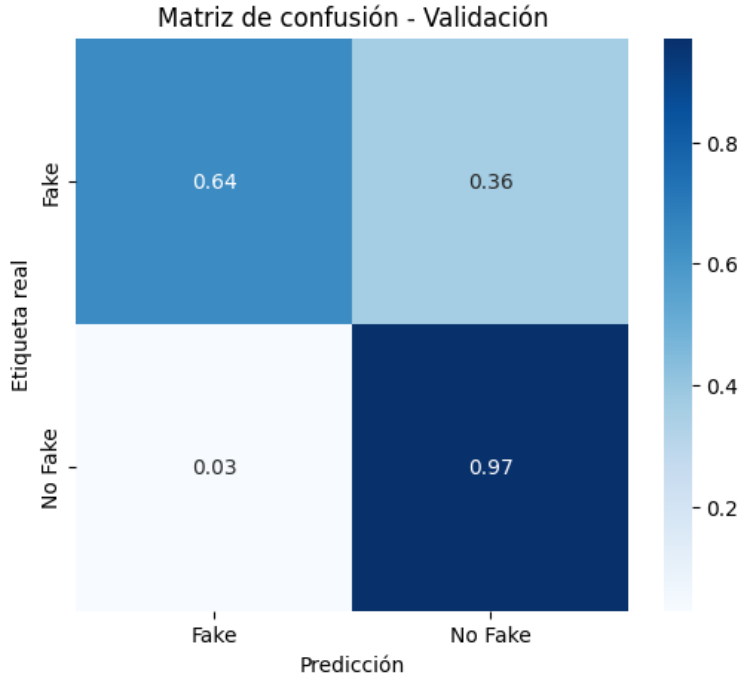
Este modelo, al igual que *Árboles de Decisión*, no tiene un parámetro que ajuste directamente sus pesos. Más bien, se rige por el número de árboles que lo componen. Y para no probar todas las combinaciones de parámetros, se utilizó *RandomizedSearchCV* con **75** combinaciones y *validación cruzada* de *3-folds*. El *scoring* también se configuró en *"f1_weighted"*. El sistema revisa si ya existe un modelo previo y compara cuál de los modelos de *Random Forest* es mejor, guardando únicamente el más óptimo.

Este modelo fue evaluado individualmente de la misma forma y con las mismas métricas que los anteriores, empezando por la *matriz de confusión*. Como se observa en la **Figura 28**, el mejor modelo de *Random Forest* obtuvo un **64%** de acierto en noticias falsas y un **97%** en noticias reales en el conjunto de validación. Esto equivale a un **36%** de error en noticias falsas y un **3%** en noticias reales.

En la **Figura 29**, el mismo modelo obtuvo un **64%** de acierto en noticias falsas y un **97%** en noticias reales en el conjunto de prueba. Esto da, al igual que en validación, un **36%** de error en noticias falsas y un **3%** en noticias reales. Estos resultados indican que, al igual que los modelos anteriores, no es del todo malo generalizando para identificar noticias falsas. Incluso, aunque mejoró detectando noticias reales, no empeoró detectando noticias falsas, como se veía en el patrón de los modelos anteriores, que al mejorar en reales empeoraban en falsas.

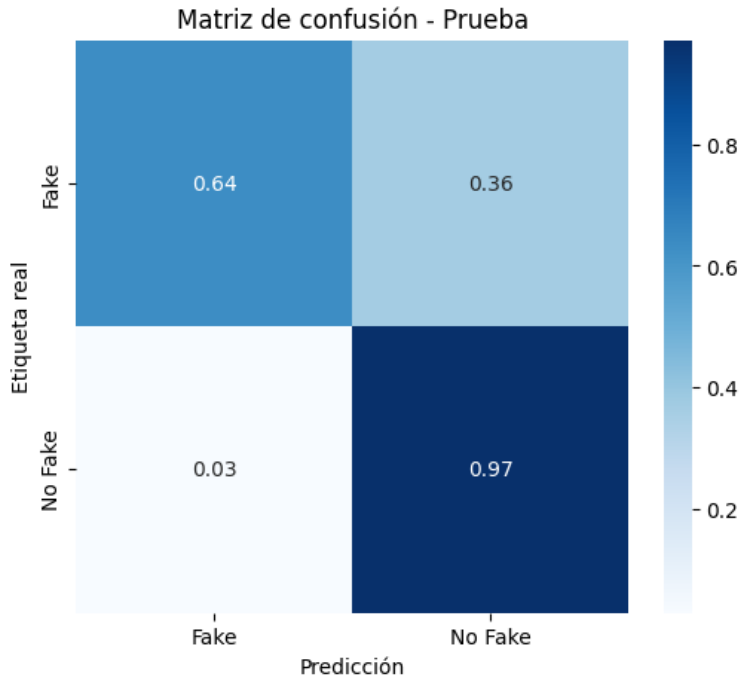
Presenta resultados similares a los de *Naive Bayes*, aunque incluso lo supera en detección de noticias reales, poniéndose en primer lugar como el modelo que mejor generaliza y predice este tipo de noticias, superando a *Árboles de Decisión*. Sin embargo, su rendimiento en la detección de noticias falsas es mediocre e inferior al de algunos de los modelos anteriores, pero iguala a *Naive Bayes*. A simple vista, parece el mejor detectando noticias reales y no es el peor detectando noticias falsas.

Figura 28
Matriz de confusión - Validación - Random Forest



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Figura 29
Matriz de confusión - Prueba – Random Forest

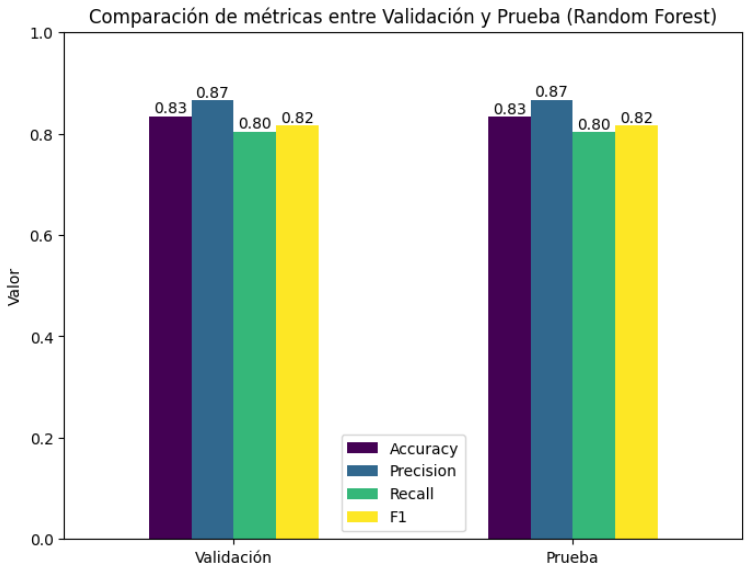


Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 30**, son superiores en todo a las del modelo de *Árboles de Decisión*, y también superan a las de *Naive Bayes*. El modelo de *Random Forest* presenta una mejora superior de alrededor de **3%** en la métrica con mayor diferencia.

Figura 30

Comparación de métricas entre Validación y Prueba - Random Forest



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

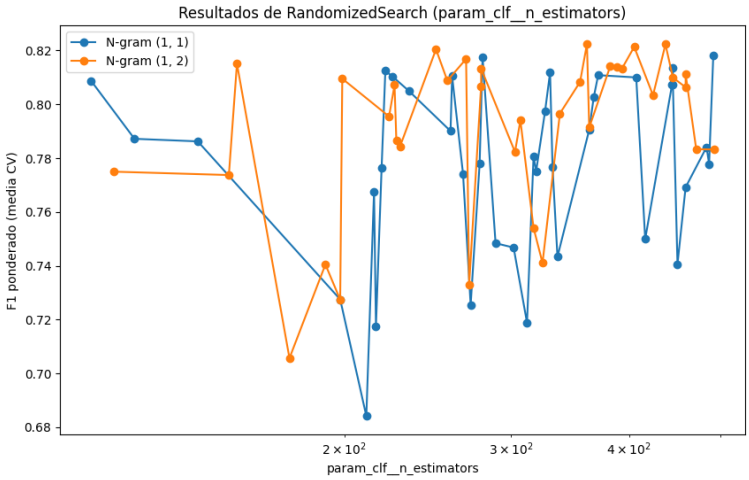
Por último, en la **Figura 31** se puede observar cómo el *score* se modificaba con cada combinación de parámetros y cambios en los valores de la cantidad de árboles. Este modelo parece ser el más inestable hasta el momento, ya que presenta caídas más grandes que las de *Regresión Logística* y con más frecuencia que las de *SVC*. Por lo tanto, parece requerir más que todos para converger. Pero, a pesar de esto, su tendencia también es ascendente, por lo que con un mayor tiempo de entrenamiento podría dar mejores resultados.

Este modelo, al igual que *Árboles de Decisión*, no parece tener problemas con conjuntos de datos pequeños ni grandes. También maneja sin inconvenientes y casi en igualdad de condiciones tanto los unigramas como los bigramas. El mejor modelo de *Random Forest* obtuvo un *score* de *0.82238*, lo que lo ubica en el tercer puesto, por debajo del modelo de *Máquinas de Vectores de Soporte*.

No obstante, este es el puntaje general y, al igual que con *Naive Bayes* y *Árboles de Decisión*, debería analizarse más adelante si este modelo vale la pena usarlo, sobreponiendo su

rendimiento en detección de noticias falsas por encima del *score* general. Aunque tiene mejores resultados que *Naive Bayes* en ese aspecto, y mejores resultados detectando noticias falsas que *Árboles de Decisión*, aún podría estar cayendo en prácticas incorrectas como priorizar la predicción de noticias reales. Se debería analizar si intentar aumentar su precisión en noticias falsas, o si tal como está ya podría ser de utilidad, ofreciéndose al usuario como modelo alterno de detección de noticias reales. Por esta razón, este modelo no está del todo descartado.

Figura 31
Rendimiento por parámetros - Random Forest



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Modelos de Deep Learning

LSTM. Para los modelos de *Deep Learning* se usó otro flujo distinto, ya que estos se implementaron con *TensorFlow* y *Keras*. Al ser modelos más pesados que los tradicionales, se configuró el código para que detectara si había alguna *GPU* disponible. En caso afirmativo, se estableció el crecimiento dinámico de la memoria de la *GPU*; en caso contrario, se trabajó desde el *procesador*.

También se configuraron *tareas multihilo* para aprovechar todos los hilos del *procesador*, y se habilitó un *batch* dinámico dependiendo del *hardware* del equipo. Un *batch* es un subconjunto del conjunto de entrenamiento, lo cual permite no cargar todo el dataset en memoria. Posteriormente, se configuró la semilla en **42** tanto para *TensorFlow* como para *NumPy*.

Luego se *vectorizó* el texto con la capa *TextVectorization* de *Keras*, la cual transforma texto en *vectores de enteros*. Esta capa es muy útil porque puede añadirse a un flujo (*pipeline*) y el modelo usar directamente su salida. El *tokenizador* se configuró con un máximo de **3000 tokens**, una secuencia de salida de **200**, y los parámetros:

- ***standardize****="lower_and_strip_punctuation"*
- ***split****="whitespace"*

Posteriormente se construyó el modelo usando *Sequential*, lo que permite apilar varias capas. En este caso se empleó una arquitectura *LSTM (Long Short-Term Memory)*, un tipo de *Red Neuronal Recurrente* que mantiene memoria y ayuda a capturar dependencias temporales, como el orden de las palabras.

El modelo se diseñó con varias capas:

- ***Embedding:*** Recibe como entrada el tamaño del vocabulario (extraído por *TextVectorization*). Se configuró con vector de salida de **128**, longitud de secuencia **200**, y *mask_zero=True*, que indica a las siguientes capas que el índice 0 corresponde a *padding* y no debe procesarse. Esta capa genera una matriz donde cada *token* se transforma en un vector denso que captura relaciones semánticas.

- ***SpatialDropout1D:*** Apaga aleatoriamente dimensiones enteras del *embedding* (**20%**), lo que evita depender siempre de las mismas y mejora la generalización. Este apagado se realiza aleatoriamente en cada *batch*.
- ***LSTM:*** Con **128** neuronas, *dropout=0.4* para descartar entradas y *recurrent_dropout=0.2* para descartar conexiones recurrentes.
- ***BatchNormalization:*** Normaliza la salida de la capa anterior con media **0** y varianza **1**. Esto acelera la convergencia. Aunque en *RNN* debe usarse con precaución, después de una capa *LSTM* es menos riesgoso y puede estabilizar el entrenamiento.
- ***Dense (64, ReLU):*** Capa totalmente conectada de **64** neuronas con activación *ReLU*, para modelar relaciones no lineales y combinar la información extraída por la *LSTM*.
- ***Dropout (0.3):*** Apaga aleatoriamente el **30%** de las neuronas para reducir *overfitting*.
- ***Dense (1, sigmoid):*** Capa final con una neurona, encargada de la clasificación binaria. Se usa activación *sigmoid*, que permite una transición suave entre **0** y **1**.

El modelo se compiló con el optimizador *Adam*, que ajusta pesos en cada iteración. Como función de pérdida se usó *binary_crossentropy*, apropiada para clasificación binaria.

Para optimizar los datos, se utilizó la función *to_tf_dataset*, que convierte los *arreglos* en *tf.data.Dataset*, un formato optimizado para *TensorFlow*.

En cuanto a *callbacks*:

- ***EarlyStopping:*** Detuvo el entrenamiento si una métrica no mejoraba después de **14** épocas.
- ***ReduceLROnPlateau:*** Redujo la tasa de aprendizaje si una métrica no mejoraba después de **3** épocas.

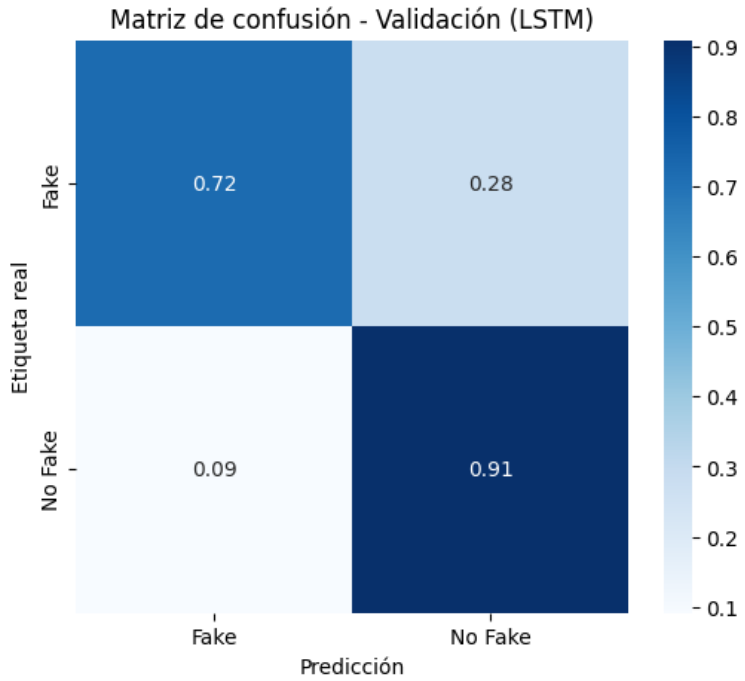
Finalmente, se entrenó el modelo por **15 épocas**, siendo este el modelo más demorado en cuestión de tiempo de entrenamiento, hasta el momento.

Los modelos de *Deep Learning* serán evaluados de la misma forma y con las mismas *métricas* que los modelos tradicionales. La única diferencia es que no se evaluó el cambio del *score* conforme variaban los parámetros, ya que en los modelos de *Deep Learning* no puede realizarse la *búsqueda en cuadrícula* (*GridSearch*).

Empezando con este modelo por la *matriz de confusión*: como se observa en la **Figura 32**, el mejor modelo de *LSTM* obtuvo un **72% de acierto en noticias falsas** y un **91% en noticias reales** en el conjunto de validación. Esto equivale a un **28% de error en noticias falsas** y un **9% en noticias reales**.

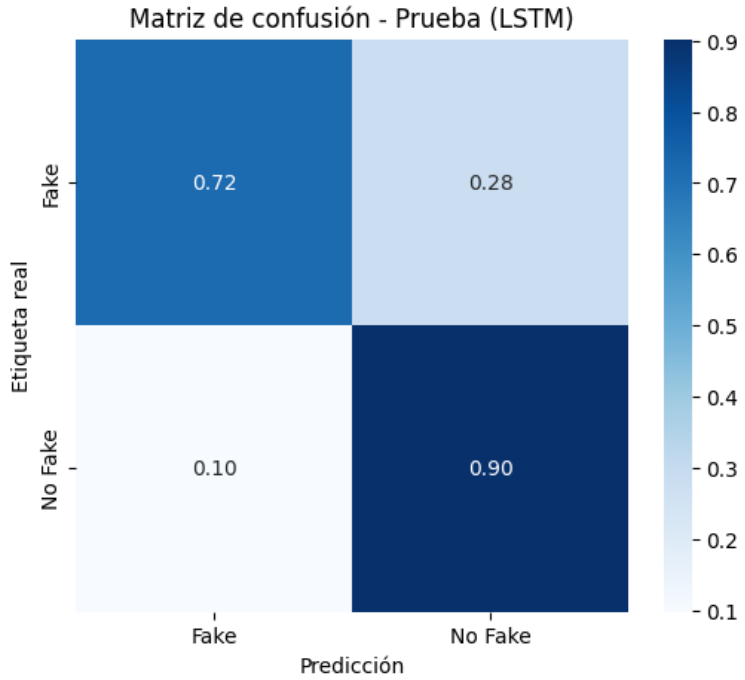
En la **Figura 33**, el mismo modelo obtuvo un **72% de acierto en noticias falsas** y un **90% en noticias reales** en el conjunto de prueba. Esto da, en validación, un **28% de error en noticias falsas** y un **10% en noticias reales**. Estos resultados demuestran que el modelo es bueno prediciendo noticias reales y falsas, aunque es ligeramente inferior en la detección de noticias falsas. Además, considerando el tiempo y la complejidad de entrenamiento, no parece valer la pena frente a otros modelos tradicionales.

Figura 32
Matriz de confusión - Validación - LSTM



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

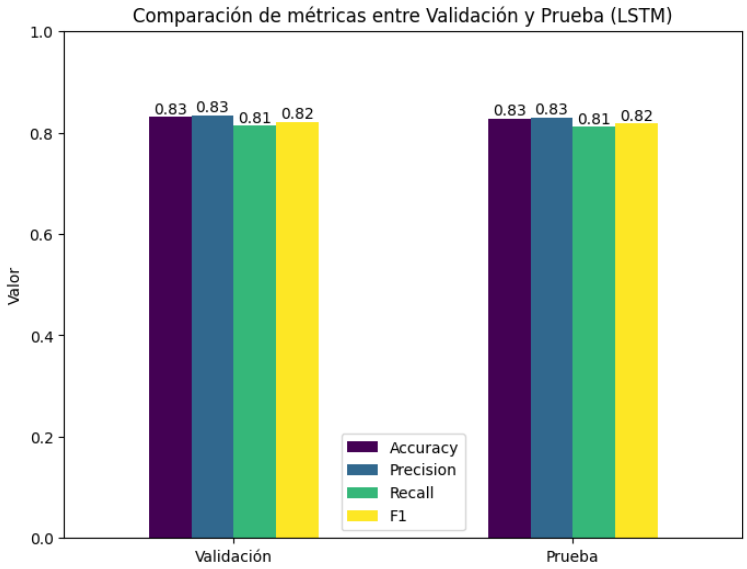
Figura 33
Matriz de confusión - Prueba – LSTM



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás *métricas* usadas para evaluar el modelo, como se observa en la **Figura 34**, son bastante equilibradas. Ninguna *métrica* supera a la otra por más de **2%**, y todas se encuentran en el rango de **81% a 83%**.

Figura 34
Comparación de métricas entre Validación y Prueba - LSTM



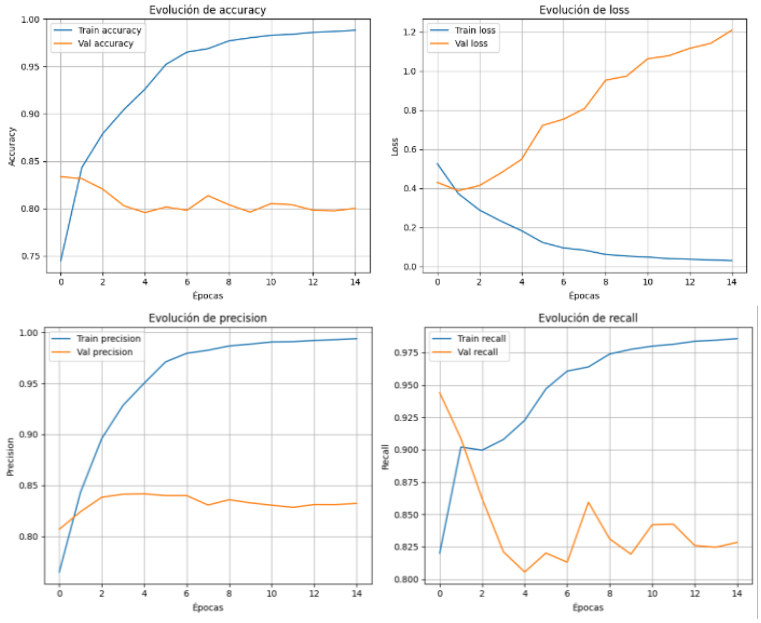
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Por último, en la **Figura 35** se observa cómo cada una de las *métricas* se modificaba con cada *época*. Este modelo no es tan inestable como los modelos tradicionales y parece tener una tendencia a mejorar; sin embargo, con cada *época* esa tendencia disminuía, por lo que entrenarlo durante más *épocas* podría no valer la pena.

Este modelo, sin embargo, presenta muestras evidentes de *overfitting*. En las *métricas* donde tanto la línea de *Train* como la de *Val* deberían crecer y mantenerse similares, ocurre que *Train* crece mientras que *Val* decrece. Esto indica que sería necesario simplificar el modelo o aumentar los valores de *Dropout* y *SpatialDropout1D* para controlarlo. El mejor modelo de *LSTM* obtuvo un *score* de **0.82112**, lo que hace que no quede al final del ranking.

Sin embargo, no es el mejor en ningún aspecto y presenta *overfitting*. Se tendrían que realizar más pruebas para ajustarlo y obtener un mejor rendimiento, lo cual sería demorado. Por lo tanto, deberá analizarse más adelante si este modelo puede llegar a ser útil o si será descartado completamente.

Figura 35
Evolución de métricas por épocas – LSTM



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

BiLSTM. Para el modelo de *BiLSTM* se usó un flujo igual al del modelo de *LSTM*, ya que este modelo es como *LSTM*, solo que bidireccional; es decir, mientras *LSTM* analiza la secuencia en una sola dirección, *BiLSTM* lo hace en ambas, lo que permite capturar el contexto tanto hacia adelante como hacia atrás en la secuencia. En este caso se usaron las mismas configuraciones del modelo anterior, incluyendo la estructura de las capas y el *pipeline*, así como la capa de *vectorización* con *TextVectorization* y los mismos parámetros. Se construyó el modelo usando *Sequential*, y también se empleó la arquitectura *LSTM*; la única diferencia es que la capa

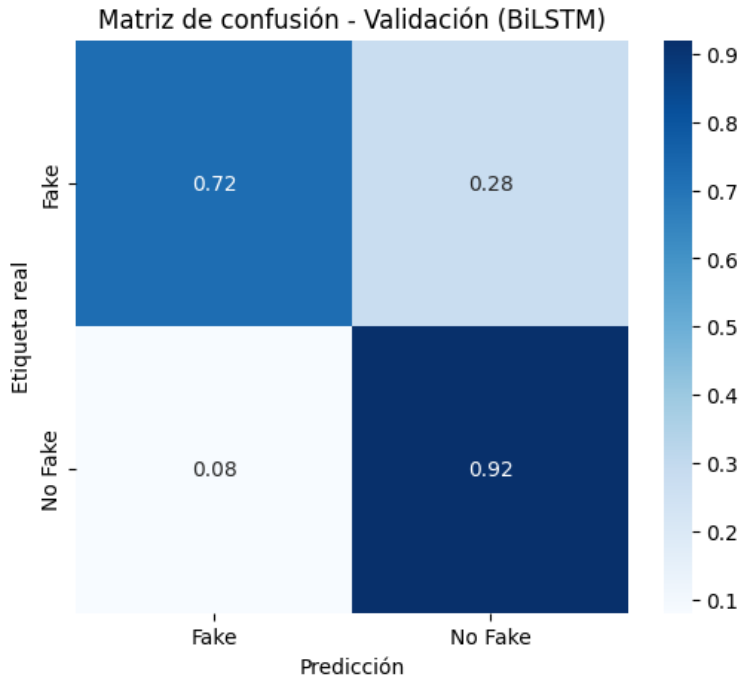
de *LSTM* se envuelve con una capa *Bidirectional*. Con este pequeño cambio la arquitectura *LSTM* se convierte en *BiLSTM*.

Se siguió usando *Adam* como optimizador y *binary_crossentropy* como *función de pérdida*; los *callbacks* permanecen igual: *EarlyStopping* con **14** épocas de paciencia y *ReduceLROnPlateau* con **3** épocas de paciencia. Se establecieron **15** épocas de entrenamiento; este modelo fue mucho más demorado que *LSTM*.

Este modelo también fue evaluado individualmente de la misma forma y con las mismas métricas que el modelo *LSTM*, comenzando por la *matriz de confusión*. Como se observa en la **Figura 36**, el mejor modelo de *BiLSTM* obtuvo un **72%** de acierto en noticias falsas y un **92%** en noticias reales en el conjunto de validación. Esto equivale a un **28%** de error en noticias falsas y un **8%** en noticias reales.

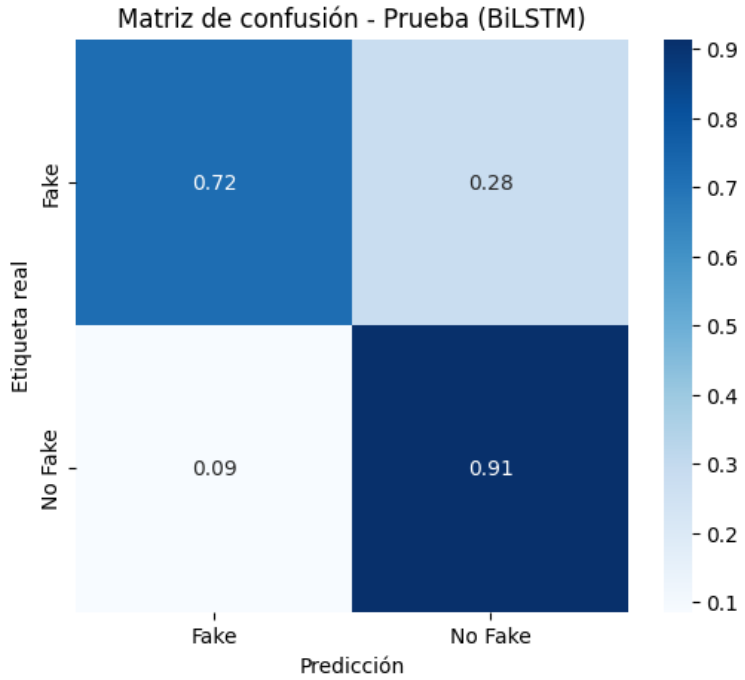
En la **Figura 37**, el mismo modelo obtuvo un **72%** de acierto en noticias falsas y un **91%** en noticias reales en el conjunto de prueba. Esto equivale, en prueba, a un **29%** de error en noticias falsas y un **10%** en noticias reales. Estos resultados indican que apenas hay un cambio entre usar *BiLSTM* o *LSTM*; aunque no es malo generalizando, los resultados son mediocres en relación con la diferencia de tiempo que llevó entrenarlo comparado con *LSTM*. Sus resultados impiden hacer una suposición concluyente, ya que, aunque en validación supera por **1%** a *LSTM*, en prueba vuelve a perderlo.

Figura 36
Matriz de confusión - Validación - BiLSTM



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

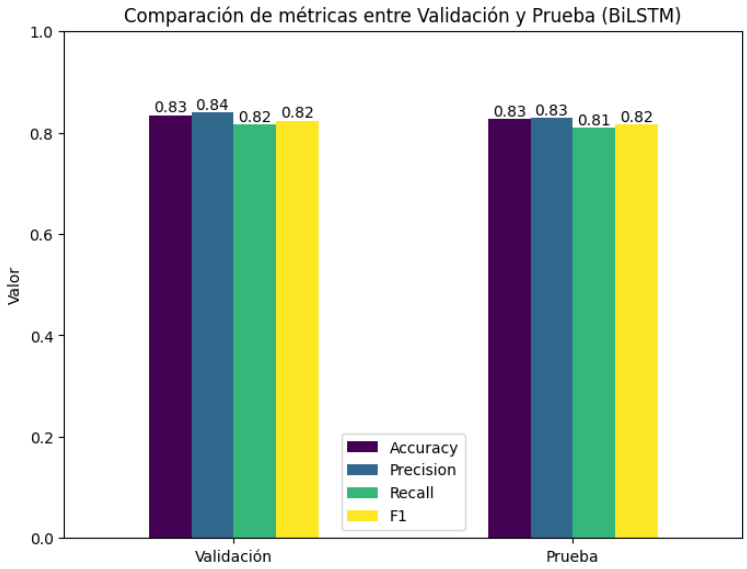
Figura 37
Matriz de confusión - Prueba – BiLSTM



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 38**, son bastante equilibradas e igualan a las de *LSTM* en prueba; en validación apenas supera alguna métrica por un **1%**.

Figura 38
Comparación de métricas entre Validación y Prueba - BiLSTM



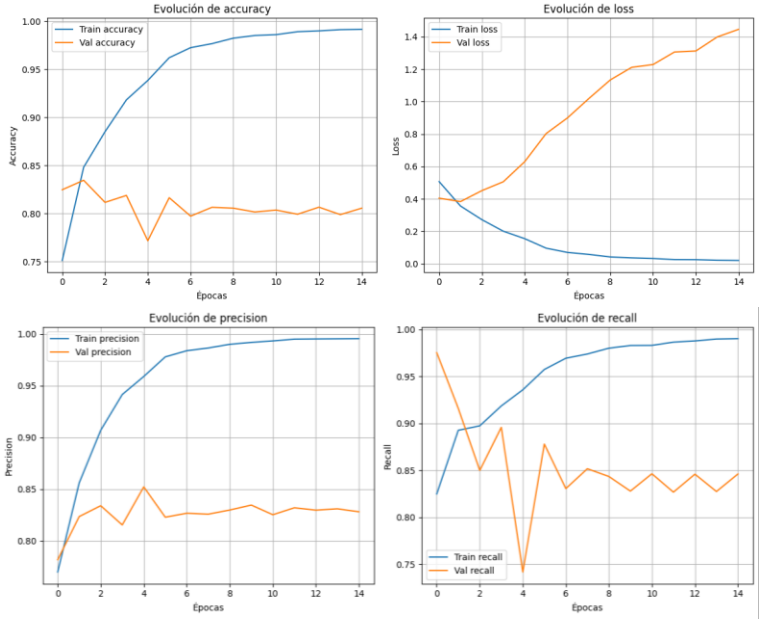
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Por último, en la **Figura 39** se observa cómo cada una de las métricas se modificaba con cada época. Este modelo es más inestable que el de *LSTM* y muestra una tendencia a mejorar; no obstante, esa tendencia disminuye con cada época.

El modelo presenta tanto momentos de mayor *overfitting* como momentos en que lo es menos que *LSTM*; aun así, el *overfitting* es evidente. Los abismos entre la línea de *Train* y la de *Val* son más irregulares que en *LSTM*. Esto indica que sería necesario simplificar el modelo —por ejemplo, volver a *LSTM*— o aumentar los valores de *Dropout* y *SpatialDropout1D* de manera más drástica que en *LSTM* para controlarlo. El mejor modelo de *BiLSTM* obtuvo un *score* de **0.82792**, lo que hace que supere a *LSTM*, pero la diferencia no es tan evidente.

Al igual que *LSTM*, tampoco es el mejor en ningún aspecto decisivo; al aumentar la complejidad del modelo presenta *overfitting* más pronunciado que el de *LSTM*. Para considerar el uso de este modelo habría que realizar más modificaciones que en *LSTM* para ajustarlo; además, al ser más demorado de entrenar, da la impresión de que podría descartarse fácilmente si no se justifican mejoras claras en rendimiento.

Figura 39
Evolución de métricas por épocas – BiLSTM



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

GRU. Para el modelo de *GRU* se usó un flujo igual al de los modelos anteriores. Este modelo es un tipo de *Red Neuronal Recurrente*, optimizada para trabajar con secuencias como texto o audio. Es una simplificación del modelo de *LSTM*, ya que *GRU* utiliza solo dos puertas en lugar de las tres que maneja *LSTM*. Además, como *GRU* no tiene una celda de memoria, entrena mucho más rápido y de forma más eficiente. De hecho, este modelo hasta el momento es el que menos se ha demorado por paso en cada época.

Nuevamente se usaron las mismas configuraciones aplicadas desde el modelo de *LSTM*, incluyendo la estructura de las capas, el *pipeline* y la capa de *vectorización* con *TextVectorization*, junto con los mismos parámetros. El cambio realizado fue a partir de lo observado con los dos modelos anteriores: para evitar el *overfitting* se aumentaron los valores de *dropout_rate*, *recurrent_dropout*, *SpatialDropout1D* y *Dropout*, provocando que más conexiones, *neuronas* y dimensiones fueran desactivadas aleatoriamente durante el entrenamiento.

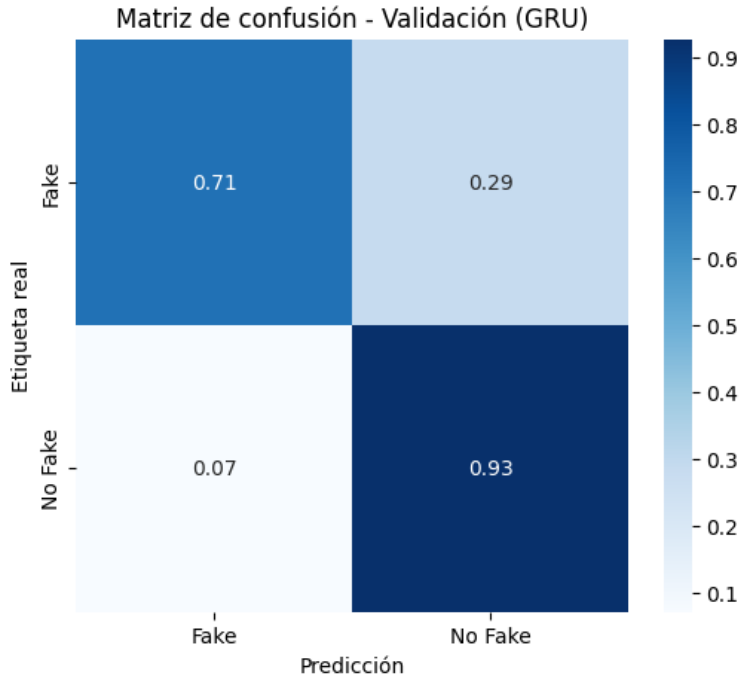
El modelo se construyó usando *Sequential* y se empleó la arquitectura *GRU* con **128** neuronas. Se siguió utilizando *Adam* como *optimizador* y *binary_crossentropy* como *función de pérdida*. Los *callbacks* permanecieron iguales: *EarlyStopping* con **14** épocas de paciencia y *ReduceLROnPlateau* con **3** épocas de paciencia. Se establecieron las mismas **15** épocas de entrenamiento.

Este modelo también fue evaluado individualmente de la misma forma y con las mismas *métricas* que el modelo *LSTM*, comenzando por la *matriz de confusión*. Como se observa en la **Figura 40**, el mejor modelo de *GRU* obtuvo un **71%** de acierto en noticias falsas y un **93%** en noticias reales en el conjunto de validación. Esto equivale a un **29%** de error en noticias falsas y un **7%** en noticias reales.

En la **Figura 41**, el mismo modelo obtuvo un **71%** de acierto en noticias falsas y un **92%** en noticias reales en el conjunto de prueba. Esto equivale, en prueba, a un **29%** de error en noticias falsas y un **8%** en noticias reales. Estos resultados indican que apenas hay diferencia frente a los obtenidos con *LSTM* y *BiLSTM*, a pesar de que el modelo *GRU* tomó menos de la mitad del tiempo en entrenarse. Aunque sus resultados no son malos generalizando, tampoco son

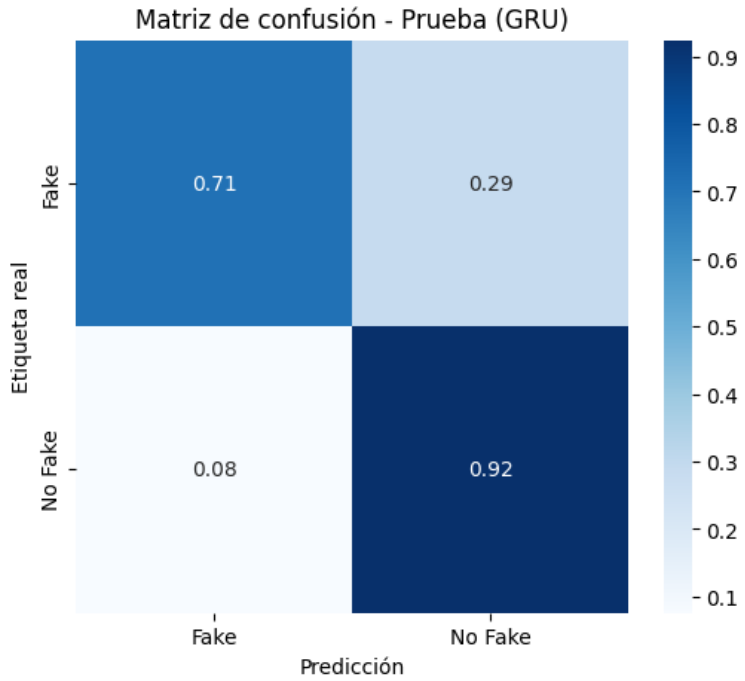
del todo satisfactorios: es bastante equilibrado, pero no destaca en ningún aspecto, salvo en el tiempo de entrenamiento.

Figura 40
Matriz de confusión - Validación - GRU



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

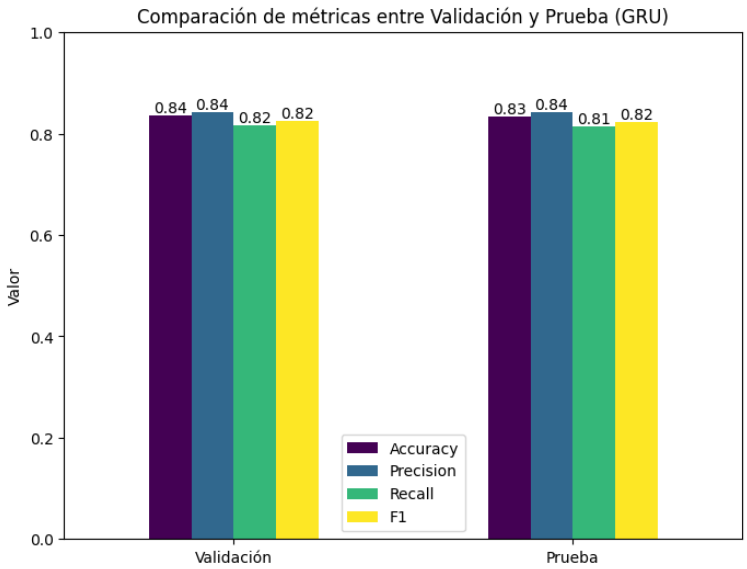
Figura 41
Matriz de confusión - Prueba – GRU



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 42**, son bastante equilibradas y se igualan a las de *BiLSTM* en ambos conjuntos; incluso logra superarlo en varias métricas por un **1%**.

Figura 42
Comparación de métricas entre Validación y Prueba - GRU



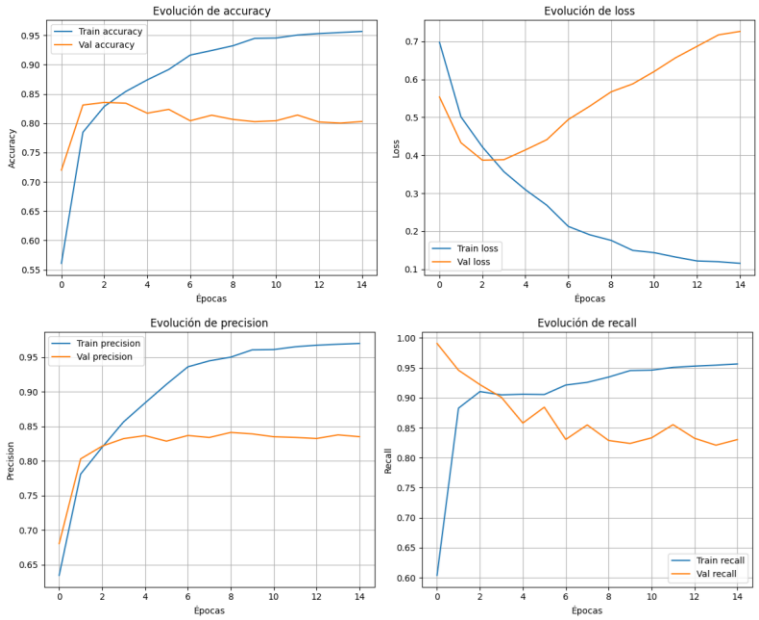
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Por último, en la **Figura 43** se observa cómo cada una de las métricas se modificaba con cada *época*. Este modelo es menos inestable que los anteriores y muestra una tendencia a mejorar; no obstante, como en los otros casos, esa tendencia disminuye con cada época.

Gracias a los cambios realizados, este modelo muestra mucha menos evidencia de *overfitting*; sin embargo, sigue presentándose, aunque ya no tan pronunciado ni exagerado. Para próximos intentos con este modelo será necesario buscar más técnicas para evitar el *overfitting*. El mejor modelo de *GRU* obtuvo un score de **0.82785**, lo que lo hace superar a *LSTM*, aunque no alcanza a *BiLSTM*.

Al igual que en los modelos anteriores, tampoco es el mejor en ningún aspecto decisivo. Sin embargo, al disminuir la complejidad, el modelo presenta mayor disposición a generalizar correctamente que los otros dos. Para pensar en usar este modelo se necesitarían realizar más modificaciones para mejorar su rendimiento y reducir el *overfitting*. Aun así, por su velocidad de entrenamiento, sería una opción más plausible que los anteriores.

Figura 43
Evolución de métricas por épocas – GRU



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

CNN. Para el modelo de *CNN* se mantuvo el flujo de los modelos anteriores. Este modelo es una *Red Neuronal Convolucional*; este tipo de red está diseñada para procesar datos con forma de rejilla, por esta razón se hicieron famosas para procesar imágenes. Aun así, también pueden usarse para procesar texto.

A diferencia de una *red densa*, la *CNN* no aprende conexiones directas entre todas las *neuronas*; en su lugar, aprende *filtros* que se mueven a lo largo de los datos, extrayendo *patrones locales*. Este modelo es muy rápido para entrenar y supera por bastante a *GRU* en este aspecto.

Se usaron las mismas configuraciones que en los modelos anteriores, tanto en la estructura de las capas, como en el *pipeline* y la capa de *vectorización* con *TextVectorization*, junto con los mismos parámetros del modelo *GRU* para limitar el *overfitting*.

El modelo se creó con *Sequential*. Se usó:

- Una capa de ***Embedding***, con *mask_zero* desactivado ya que la *red convolucional* no aprovecha esta característica.
- Una capa de ***SpatialDropout1D***.
- Una capa ***Conv1D***, la cual implementa la red convolucional, con **128** *filtros*, *kernel* de **5**, *activation="relu"* y *padding="same"*.
- Una capa de ***BatchNormalization***.
- Una capa de ***GlobalMaxPooling1D***.
- Una capa ***Dense*** de **64** *neuronas* con *activation="relu"*.
- Una capa de ***Dropout***.
- Una capa ***Dense*** de **1** *neurona* con *activation="sigmoid"*.

Se utilizó *Adam* como *optimizador* y *binary_crossentropy* como *función de pérdida*. Además, se implementaron *EarlyStopping* y *ReduceLROnPlateau*, con **14** y **3** *épocas de paciencia*, respectivamente, y un total de **15** *épocas* de entrenamiento.

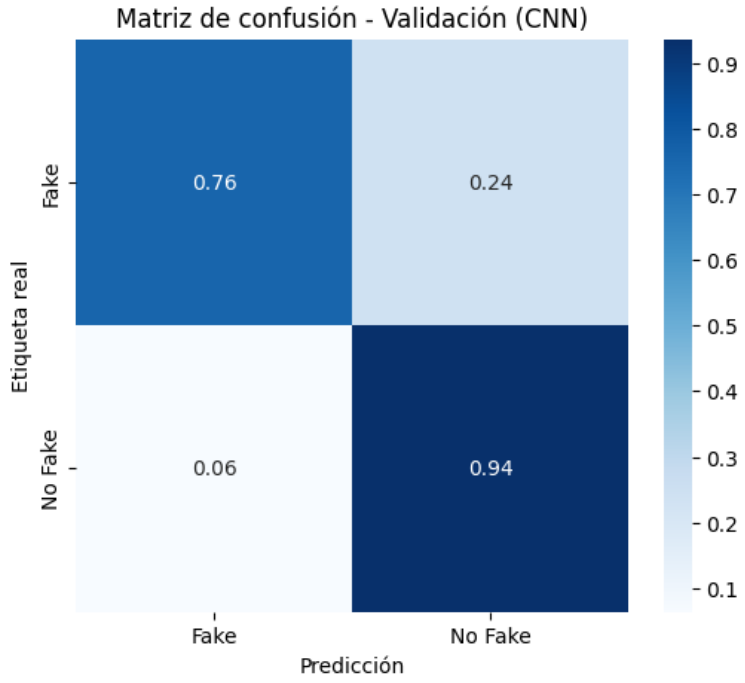
Este modelo fue evaluado de manera individual, de la misma forma y con las mismas métricas que el modelo *LSTM*, comenzando por la *matriz de confusión*.

- En la **Figura 44**, el mejor modelo de *CNN* obtuvo un **76%** de acierto en noticias falsas y un **94%** en noticias reales en el conjunto de validación (**24%** de error en noticias falsas y **6%** en noticias reales).
- En la **Figura 45**, los resultados fueron iguales en el conjunto de prueba: **76%** de acierto en noticias falsas y **94%** en noticias reales.

Estos resultados indican que es un modelo bastante bueno prediciendo tanto noticias reales como falsas. De hecho, es el mejor modelo hasta el momento en la detección de **noticias**

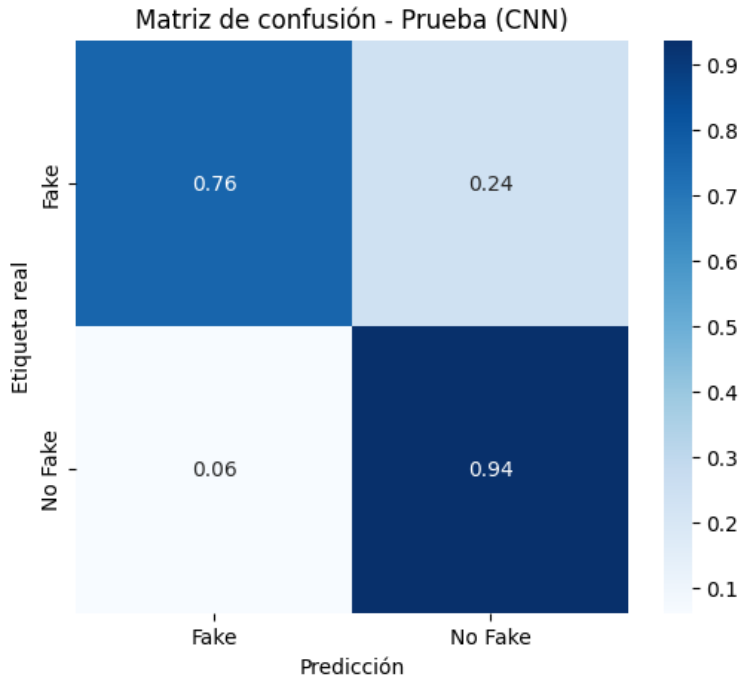
falsas. En la detección de **noticias reales** no es el mejor, pero se acerca bastante. Además, fue el más rápido en entrenamiento.

Figura 44
Matriz de confusión - Validación - CNN



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Figura 45
Matriz de confusión - Prueba – CNN

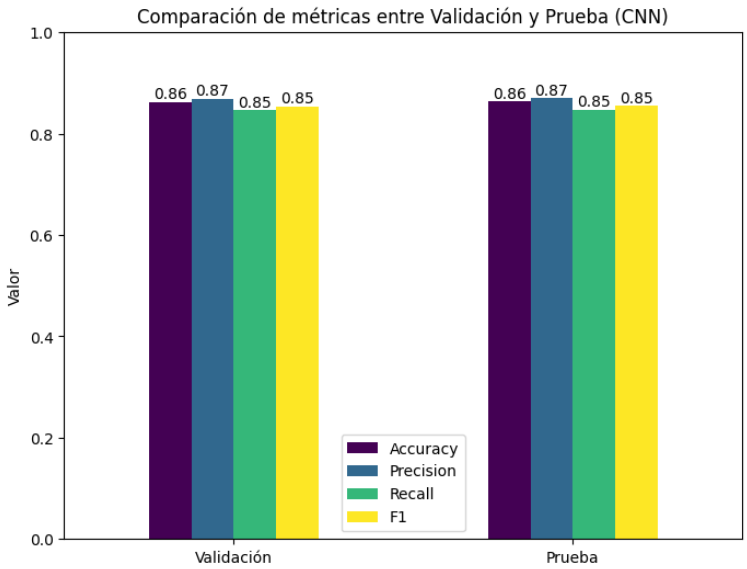


Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 46**, son bastante equilibradas y altas. Es el modelo que más se acerca a cumplir los indicadores del resultado esperado.

Figura 46

Comparación de métricas entre Validación y Prueba - CNN



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

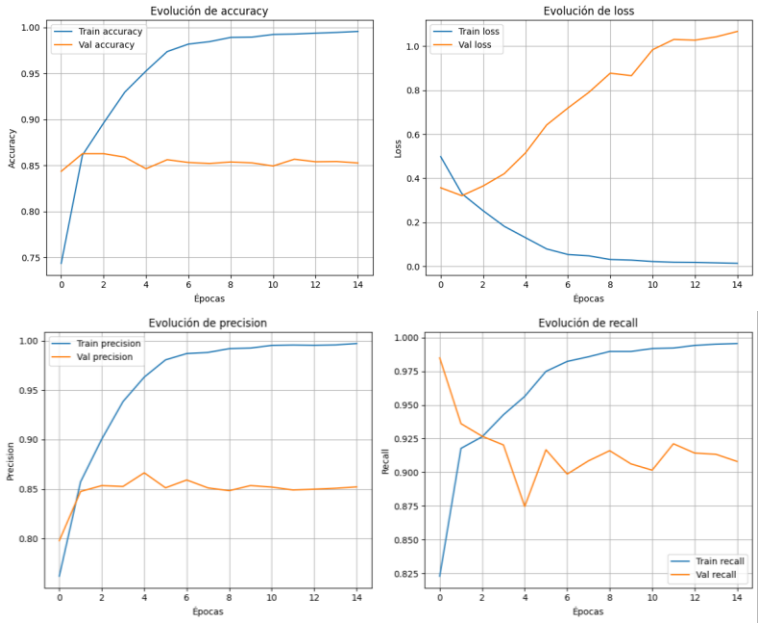
En la **Figura 47** se observa cómo cada una de las métricas se modificaba con cada época. Este modelo presenta gráficas similares al modelo *GRU*: es un poco inestable, pero su tendencia a mejorar es la más baja de todas.

Los cambios realizados para evitar el *overfitting* obtuvieron resultados similares a los de *GRU*. Al igual que en este, siguen presentándose evidencias de *overfitting*, por lo que será necesario buscar otras estrategias para disminuirlo.

El mejor modelo de *CNN* obtuvo un *score* de **0.85423**, lo que evidencia que sí es superior a los demás en términos de generalización.

En conclusión, este modelo, a diferencia de los anteriores, demuestra ser el mejor en varios aspectos y puntúa como el más óptimo hasta el momento para este caso de uso. Por lo tanto, se deberá analizar si será el modelo elegido para continuar con el desarrollo y reentrenarlo con la intención de mejorar su rendimiento.

Figura 47
Evolución de métricas por épocas – CNN



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

CNN + BiLSTM + GRU + Atención Simple. Para el modelo híbrido de *CNN + BiLSTM + GRU + atención simple* se mantuvo, en su mayoría, el mismo flujo que en los modelos anteriores. Un modelo *híbrido* es aquel que combina varios enfoques para obtener lo mejor de cada uno; en este caso, se usaron los modelos ya vistos y se le adicionó una capa de *atención simple*. En un principio se planteó hacerlo con *Attention*, pero debido a salidas que no podía manejar dicha capa, se reemplazó por *MultiHeadAttention*.

Este modelo, al unir varios enfoques, tiene un tiempo de entrenamiento más alto que los anteriores. En su mayoría se usaron las mismas configuraciones, pero hubo algunos cambios. Para empezar, en este modelo no se usó *Sequential* para crear la arquitectura, sino que se guardó la salida de cada capa en una variable y después se pasó a la siguiente. La razón de no usar *Sequential* es que de este modo se logra mayor personalización.

El modelo tiene:

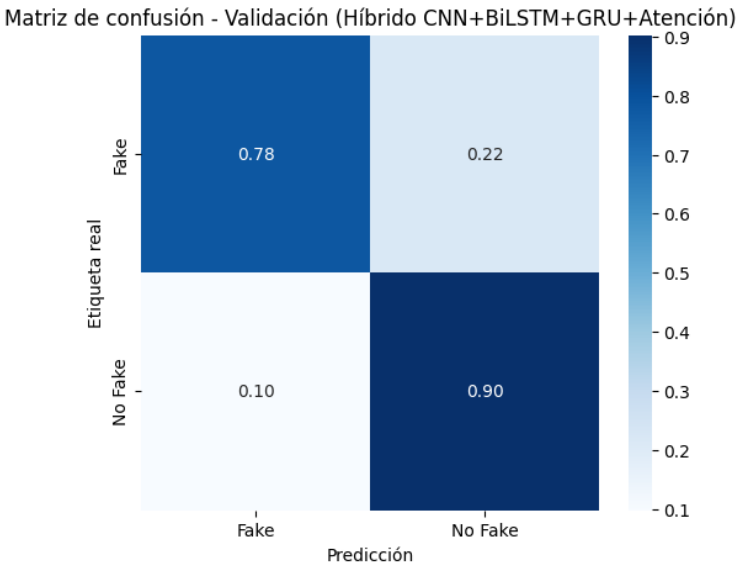
- Una capa de *vectorización* con ***TextVectorization***.
- Una capa ***Input***, que instancia el *tensor* de *Keras* para recibir la entrada.
- Una capa de ***Embedding*** con *mask_zero* desactivado, ya que aunque las capas *BiLSTM* la aprovechan, las demás no.
- Una capa de ***SpatialDropout***.
- Una capa ***Conv1D*** con **128** *filtros*, activación *ReLU* y *padding valid*.
- Una capa de ***BatchNormalization***.
- Una capa de ***GlobalMaxPooling1D*** y otra de ***GlobalAveragePooling1D***.
- Una capa ***BiLSTM*** con **64** unidades.
- Otra capa de ***BatchNormalization***.
- Una capa ***GRU*** con **64** unidades.
- Otra capa de ***BatchNormalization***.
- Una capa de ***MultiHeadAttention*** con **4** cabezas, que permite observar la secuencia desde varios ángulos para capturar diferentes tipos de relaciones entre *tokens*.
- Una capa de ***GlobalAveragePooling1D***, además de otras dos capas para representar el *pooling* global de la *GRU*.
- Concatenación de todas las características para pasarlas a una red densa, con capas ***Dense*** de **256**, **128**, **64** y **1** neurona. Después de cada una se aplicó ***BatchNormalization*** y ***Dropout***.

Se siguió usando *Adam* como optimizador y *binary_crossentropy* como *función de pérdida*, y se agregó *gradient clipping*. Los *callbacks* se mantuvieron igual: *EarlyStopping* y *ReduceLROnPlateau*, con **14** y **3** épocas de paciencia, respectivamente, y un total de **15** épocas de entrenamiento.

Este modelo también fue evaluado individualmente de la misma forma y con las mismas métricas que el modelo *LSTM*, comenzando por la *matriz de confusión*. Como se observa en la **Figura 48**, el mejor modelo *híbrido* obtuvo un **78%** de acierto en noticias falsas y un **90%** en noticias reales en validación, lo que equivale a un **22%** de error en noticias falsas y un **10%** en noticias reales.

En la **Figura 49**, el mismo modelo obtuvo un **79%** de acierto en noticias falsas y un **91%** en noticias reales en prueba, lo que equivale a un **21%** de error en noticias falsas y un **9%** en noticias reales. Estos resultados indican que, superando incluso a *CNN*, predice mejor las noticias falsas, aunque no lo supera detectando noticias reales. Sin embargo, la diferencia en la duración del entrenamiento no justifica del todo las mejoras obtenidas.

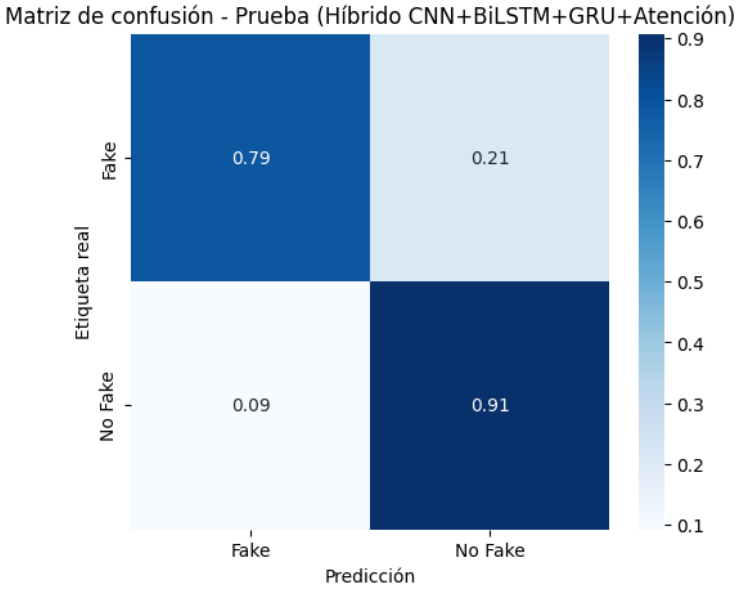
Figura 48
Matriz de confusión - Validación - CNN + BiLSTM + GRU + Atención Simple



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Figura 49

Matriz de confusión - Prueba - CNN + BiLSTM + GRU + Atención Simple

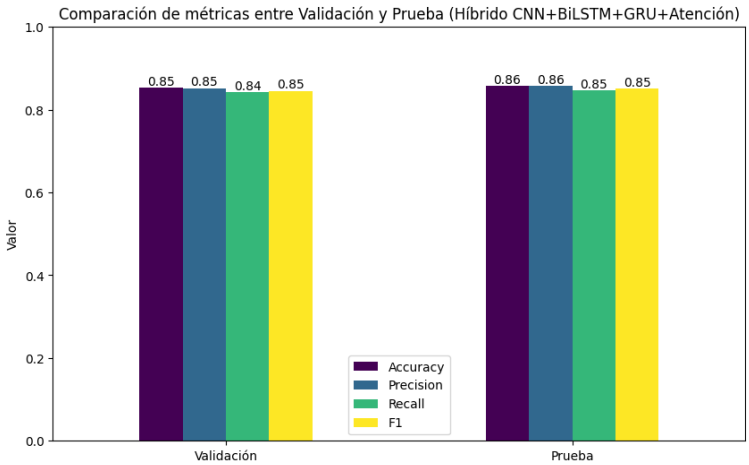


Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 50**, son bastante equilibradas, aunque no superan a las de *CNN*.

Figura 50

Comparación de métricas entre Validación y Prueba - CNN + BiLSTM + GRU + Atención Simple



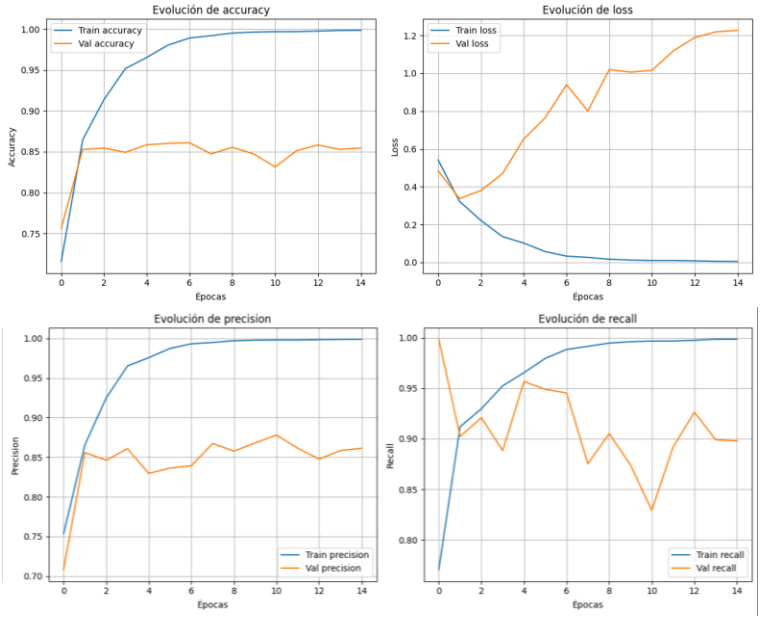
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

En la **Figura 51** se observa cómo cada métrica se modificaba con cada época. Este modelo presenta gráficas similares al modelo de CNN anterior: aunque es un poco más inestable, su tendencia a mejorar se mantiene.

A pesar de que este modelo es más complejo que *CNN*, demostró que presenta menos evidencias de *overfitting* que aquel, aunque aún se observan rastros de este problema. El mejor modelo híbrido obtuvo un *score* de **0.84571**, lo que confirma que no supera al modelo *CNN*.

En conclusión, este modelo demuestra ser el mejor en la predicción de noticias falsas. Sin embargo, no se posiciona como el mejor en otros aspectos y su entrenamiento es el más demorado de todos. Por lo tanto, si a futuro se planea hacer uso de este modelo, deberá tenerse en cuenta la dificultad y el tiempo de su reconfiguración y reentrenamiento. Aun así, no está descartado del todo.

Figura 51
Evolución de métricas por épocas – CNN + BiLSTM + GRU + Atención Simple



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Evaluación de Resultados

En este paso se presentarán y analizarán los resultados obtenidos de los modelos de *Machine Learning* y *Deep Learning* entrenados y validados en el paso anterior.

Usando un *script*, se cargaron los 10 modelos y se evaluaron en igualdad de condiciones, obteniendo métricas como *Accuracy*, *Precision*, *Recall*, *F1*, *ROC AUC*, *Loss*, *MCC*, *Cohen's Kappa* y el tiempo de predicción; además de otras más específicas, como la tasa de falsos positivos y falsos negativos.

También se analizó la eficiencia, la variabilidad, el *overfitting*, el nivel de confianza del modelo, y a partir de estos resultados se clasificó el modelo para identificar posibles problemas de sobreajuste o subajuste.

Con base en todo esto, se clasificaron los modelos en 10 propósitos distintos, permitiendo conocer cuál fue el mejor en cada caso y en qué circunstancias es más adecuado usar cada uno.

Esto ofrece la posibilidad de que, en lugar de presentar un único modelo, se puedan ofrecer varios dependiendo de la necesidad del usuario.

Comencemos con los resultados de la *matriz de confusión* de cada modelo.
Cada resultado, como se observa en la **Tabla 7**, muestra primero el valor de *validación* y luego el de *prueba*.

Como podemos observar, en la detección de noticias reales, los valores entre los diez modelos no difieren mucho entre sí; son bastante similares. Se evidencia que, en la mayoría de los casos, cuando los modelos son mejores detectando noticias reales, también son peores detectando noticias falsas, lo que indica que podrían estar incurriendo en malas prácticas, como priorizar las noticias reales sobre las falsas.

En este caso, como se observa, el mejor modelo en la detección de noticias reales es *Random Forest*, con apenas un **3%** de error, mientras que el mejor modelo en la detección de noticias falsas es el modelo *Híbrido*, con un **21,5%** de error.

Así quedaría el ranking de los tres mejores modelos en la detección de noticias reales:

1. *Random Forest.*
2. *Decision Tree.*
3. *Naive Bayes.*

Esto demuestra que, en este caso, los modelos tradicionales parecen más aptos para la detección de noticias reales.

Y así quedaría el ranking de los tres mejores modelos en la detección de noticias falsas:

1. *Hibrido.*
2. *CNN.*
3. *Logistic Regression.*

Esto no demuestra necesariamente que los modelos de *deep learning* sean siempre mejores, pero en un análisis general del rendimiento de los diez modelos, es evidente que los modelos de *deep learning* superan a los modelos tradicionales en la detección de noticias falsas.

Tabla 7
Resumen comparativo de matrices de confusión — Validación y Prueba

Modelo	Noticias Reales		Noticias Falsas	
Logistic Regression	92%	92%	74%	72%
SVC	92%	92%	73%	71%
Naive Bayes	95%	95%	65%	64%
Decisión Tree	96%	96%	60%	60%
Random Forest	97%	97%	64%	64%
LSTM	91%	90%	72%	72%
BiLSTM	92%	91%	72%	72%
GRU	93%	92%	71%	71%
CNN	94%	94%	76%	76%
Hibrido	90%	91%	78%	79%

Nota. La tabla resume los resultados obtenidos a partir de las matrices de confusión de los diferentes modelos evaluados, mostrando el porcentaje de clasificación correcta de noticias reales y falsas en las fases de validación y prueba del sistema. *Fuente.* Autoría propia.

Continuemos con las métricas de cada modelo. Cada métrica, como se muestra en la **Tabla 8**, presenta primero el valor de validación y luego el de prueba.

En *Accuracy*, el mejor modelo tanto en validación como en prueba fue el *CNN*, con valores de **0.86281** y **0.86346**, lo que lo convierte en el modelo que mejor clasifica en general.

En *Precisión*, el mejor modelo es el *Híbrido*, con **0.85599** y **0.85922**. Esto indica que es el modelo más acertado al predecir positivos cuando realmente lo son; es decir, el mejor para reducir falsos positivos (noticias falsas marcadas como reales).

En *Recall*, el mejor modelo es el *Random Forest*, con **0.97106** y **0.97172**, lo que demuestra que es el más eficaz detectando la mayor cantidad de noticias que realmente son reales, coincidiendo con lo observado en las matrices de confusión.

En *F1-Score*, el mejor modelo es nuevamente el *CNN*, con **0.88959** y **0.89018**. Esto indica que es el modelo más equilibrado entre *Precisión* y *Recall*, es decir, un modelo balanceado que no sacrifica detección por exactitud.

En *ROC AUC*, el mejor modelo también es el *CNN*, con **0.92681** y **0.92152**, lo que significa que tiene la mejor capacidad global para distinguir entre noticias falsas y reales.

Por último, en *Log Loss*, el mejor modelo vuelve a ser el *CNN*, con **0.32048** y **0.32491**, lo que indica que tiene las predicciones mejor calibradas, es decir, las probabilidades más confiables y de mayor calidad.

Esto indica, a simple vista, que el modelo *CNN*, aunque no se destaca como el mejor en la detección de *fake news*, sí sobresale en varios aspectos importantes. Esto lo lleva a tener el mejor

rendimiento general y una mayor confiabilidad, convirtiéndolo en el candidato más adecuado para implementarse en el sistema de detección de *fake news*.

Sin embargo, se sigue analizando la posibilidad de ofrecer más de un modelo al usuario, brindando así una opción para cada circunstancia específica dentro de lo que sea posible.

Tabla 8
Métricas Principales de Rendimiento

Modelo	Accuracy		Precision		Recall	
Logistic Regression	0.84375	0.84152	0.83346	0.82729	0.91898	0.92453
SVC	0.83994	0.83140	0.82888	0.81968	0.91853	0.91585
Naive Bayes	0.82509	0.81997	0.79474	0.78950	0.94880	0.94768
Decisión Tree	0.81445	0.81195	0.77563	0.77433	0.96483	0.96170
Random Forest	0.83416	0.83390	0.79399	0.79334	0.97106	0.97172
LSTM	0.83153	0.82798	0.82414	0.82354	0.90852	0.90182
BiLSTM	0.83850	0.83337	0.82613	0.82330	0.92009	0.91384
GRU	0.83915	0.83679	0.82257	0.82165	0.92766	0.92408
CNN	0.86281	0.86346	0.84747	0.84745	0.93612	0.93744
Hibrido	0.85269	0.85768	0.85599	0.85922	0.90229	0.90761

Modelo	F1-Score		ROC AUC		Log Loss	
Logistic Regression	0.87413	0.87321	0.90648	0.89655	0.38822	0.39638
SVC	0.87141	0.86510	0.82260	0.81280	5.76887	6.07674
Naive Bayes	0.86496	0.86139	0.88022	0.86685	0.40114	0.41499
Decisión Tree	0.85994	0.85790	0.83884	0.83716	1.11784	1.15277
Random Forest	0.87364	0.87352	0.89446	0.88721	0.51104	0.51343
LSTM	0.86428	0.86090	0.88882	0.88718	0.38795	0.38841
BiLSTM	0.87059	0.86621	0.89190	0.88722	0.38185	0.38761
GRU	0.87196	0.86986	0.88131	0.87599	0.39076	0.40010
CNN	0.88959	0.89018	0.92681	0.92152	0.32048	0.32491
Hibrido	0.87853	0.88275	0.92480	0.92267	0.33726	0.33940

Nota. La tabla presenta las principales métricas de rendimiento de los modelos evaluados, incluyendo accuracy, precision, recall, F1-score, ROC AUC y log loss, utilizadas para comparar el desempeño de modelos tradicionales, de aprendizaje profundo y enfoques híbridos en la detección de noticias falsas. *Fuente.* Autoría propia.

Continuamos con las métricas de diagnóstico, y al igual que en la tabla anterior, en la **Tabla 9** se presentan primero los valores de validación y, en la columna a su derecha, los de prueba.

En *Specificity*, la cual es una métrica complementaria de *Recall*, el mejor modelo es el *Híbrido*, con **0.78119** y **0.78576**. Esto indica que el modelo *Híbrido* es el mejor detectando noticias falsas cuando realmente lo son, por lo que nuevamente se posiciona como el mejor para reducir *falsos positivos*.

En *False Positive Rate*, el mejor modelo también es el *Híbrido*, con **0.21880** y **0.21423**, lo que indica que es el modelo que menos clasifica noticias falsas como reales.

En *False Negative Rate*, el mejor modelo es el de *Random Forest*, con **0.02893** y **0.02827**, lo que indica que es el modelo que menos clasifica noticias reales como falsas.

En *Balanced Accuracy*, el mejor modelo es el de *CNN*, con **0.84663** y **0.84716**, lo que indica que es el modelo que mantiene mejor equilibrio entre *Recall (detección de noticias reales)* **y** *Specificity (detección de noticias falsas)*.

En *Negative Predictive Value*, el mejor modelo es el de *Random Forest*, con **0.93853** y **0.93975**, lo que indica que, de todas las noticias que predijo como falsas, fue el que más acertó, convirtiéndose en el más confiable para la detección de noticias falsas.

Estos resultados demuestran que mientras los modelos *Hibrido*, *Random Forest* y *CNN* presentan los comportamientos mas equilibrados y confiables. Mientras que el modelo *Hibrido* destaca por su capacidad de reducir *falsos positivos*, el modelo *Random Forest* destaca por su precisión al identificar correctamente las noticias falsas, y el modelo *CNN* destaca por mantener un buen balance general entre la detección de noticias reales y falsas.

Tabla 9
Métricas de Diagnóstico y Error Específico

Modelo	Specificity		FPR (False Positive Rate)		FNR (False Negative Rate)	
Logistic Regression	0.73532	0.78119	0.26467	0.27806	0.08101	0.07546
SVC	0.72666	0.70974	0.27333	0.29025	0.08146	0.08414
Naive Bayes	0.64677	0.63598	0.35322	0.36401	0.05119	0.05231
Decisión Tree	0.59769	0.59621	0.40230	0.40378	0.03516	0.03829
Random Forest	0.63683	0.63534	0.36316	0.36465	0.02893	0.02827

LSTM	0.72056	0.72161	0.27943	0.27838	0.09147	0.09817
BiLSTM	0.72088	0.71744	0.27911	0.28255	0.07990	0.08615
GRU	0.71158	0.71103	0.28841	0.28896	0.07233	0.07591
CNN	0.75713	0.75689	0.24286	0.24310	0.06387	0.06255
Hibrido	0.78119	0.78576	0.21880	0.21423	0.09770	0.09238

Modelo	Balanced Accuracy		NPV (Negative Predictive Value)	
Logistic Regression	0.82715	0.82323	0.86295	0.86911
SVC	0.82260	0.81280	0.86088	0.85411
Naive Bayes	0.79779	0.79183	0.89759	0.89404
Decisión Tree	0.78126	0.77896	0.92182	0.91531
Random Forest	0.80394	0.80353	0.93853	0.93975
LSTM	0.81454	0.81172	0.84531	0.83612
BiLSTM	0.82049	0.81564	0.86224	0.85251
GRU	0.81962	0.81755	0.87219	0.86669
CNN	0.84663	0.84716	0.89157	0.89360
Hibrido	0.84174	0.84668	0.84725	0.85514

Nota. La tabla presenta métricas de diagnóstico orientadas al análisis de errores de clasificación de los modelos evaluados, incluyendo especificidad, tasa de falsos positivos *(FPR)*, tasa de falsos negativos *(FNR)*, exactitud balanceada y valor predictivo negativo *(NPV)*, permitiendo evaluar el comportamiento del sistema frente a errores críticos en la detección de noticias falsas y reales.

Fuente. Autoría propia.

Las siguientes métricas corresponden a las de robustez, tal como se observa en la **Tabla 10**. En este caso, las métricas que se dividen en validación y prueba son menos numerosas.

En *Matthews Correlation Coefficient*, el mejor modelo es el de *CNN*, con **0.71578** y **0.71731**. Esto indica que es el modelo que muestra la mayor correlación entre predicciones y etiquetas reales, considerando los cuatro tipos de resultados *(TP, TN, FP y FN)*.

En *Cohen's Kappa*, el mejor modelo es nuevamente el de *CNN*, con **0.70964** y **0.71097**, lo que indica que es el modelo con el mayor grado de acuerdo con las etiquetas verdaderas, es decir, que predice consistentemente más allá de la casualidad.

En *Variance*, el mejor modelo es el de *Naive Bayes*, con **0.00018**, lo que indica que presenta la menor variabilidad de rendimiento entre *folds* o ejecuciones, siendo así el modelo más estable y reproducible.

En *Overfitting Accuracy*, el mejor modelo es el de *Random Forest*, con **0.00548**, lo que indica que es el modelo con la menor diferencia entre entrenamiento y prueba, y por lo tanto, el que mejor generaliza y menos tiende a memorizar los datos.

Por último, en el *Tipo de Problema*, a partir del modelo *SVC* con **0.05459** en *Overfitting Accuracy*, todos los modelos que presentan un valor superior muestran signos de ligero *overfitting*, siendo el más pronunciado en el modelo *Híbrido*, como era de esperarse en un modelo de mayor complejidad.

Estos resultados evidencian que los modelos de *CNN*, *Naive Bayes* y *Random Forest* son quienes presentan los comportamientos más consistentes y robustos. Mientras que el *CNN* destaca por su solidez general y alta correlación entre predicciones y valores reales; el *Naive Bayes* destaca por su estabilidad y baja variabilidad; y el *Random Forest* por su excelente capacidad de generalización sin sobreajustar.

Tabla 10
Métricas Avanzadas de Robustez

Modelo	MCC (Matthews Correlation Coefficient)		Cohen's Kappa		Variance
Logistic Regression	0.67503	0.67097	0.66946	0.66365	0.00020
SVC	0.66711	0.64925	0.66098	0.64220	0.00078
Naive Bayes	0.64214	0.63163	0.62220	0.61062	0.00018
Decisión Tree	0.62636	0.62029	0.59453	0.58934	0.00027
Random Forest	0.66730	0.66711	0.63936	0.63871	7.03314
LSTM	0.64896	0.64129	0.64363	0.63677	0.00123
BiLSTM	0.66425	0.65317	0.65748	0.64697	0.00088
GRU	0.66643	0.66119	0.65779	0.65305	0.00056
CNN	0.71578	0.71731	0.70964	0.71097	0.00071
Hibrido	0.69329	0.70379	0.69175	0.70208	0.00163

Modelo	Overfitting Accuracy	Tipo de Problema
Logistic Regression	0.02895	Good Fit
SVC	0.05459	Slight Overfitting
Naive Bayes	0.02562	Good Fit
Decisión Tree	0.03359	Good Fit
Random Forest	0.00548	Good Fit
LSTM	0.07268	Slight Overfitting
BiLSTM	0.06031	Slight Overfitting
GRU	0.04935	Good Fit
CNN	0.05698	Slight Overfitting
Hibrido	0.08814	Slight Overfitting

Nota. La tabla muestra métricas avanzadas de robustez de los modelos, como el *coeficiente de correlación de Matthews (MCC)*, el *índice Kappa de Cohen*, la varianza y el nivel de sobreajuste, las cuales permiten analizar la estabilidad, consistencia y capacidad de generalización de los modelos evaluados. *Fuente.* Autoría propia.

Las últimas métricas utilizadas para evaluar los modelos corresponden a las métricas de eficiencia, tal como se observa en la **Tabla 11**. Los datos están expresados en segundos.

En el *Tiempo de Prueba*, el mejor modelo es el de *Naive Bayes*, lo que significa que es el más rápido al momento de realizar las predicciones. El peor es el modelo *Híbrido*, con un tiempo de **35.71858 segundos**, como era de esperarse debido a su mayor complejidad.

En el *Tiempo de Entrenamiento*, el mejor modelo vuelve a ser el de *Naive Bayes*, con un tiempo aproximado de **990 segundos**, lo que indica que es el más eficiente durante el proceso de aprendizaje. El peor desempeño lo presenta nuevamente el modelo *Híbrido*, con un tiempo de **16,650 segundos**.

Estos resultados evidencian que el modelo *Naive Bayes* se presenta como el más eficiente, tanto en entrenamiento como en prueba, destacándose por su bajo costo computacional y rapidez. En contraste, el modelo *Híbrido* es su contraparte, mostrándose como el menos eficiente al presentar los mayores tiempos tanto en entrenamiento como en prueba.

Tabla 11
Métricas de Eficiencia

Modelo	Tiempo de Prueba	Tiempo de Entrenamiento
Logistic Regression	0.45751	1957.5
SVC	0.46809	1305
Naive Bayes	0.41668	990
Decisión Tree	0.44414	2722.5
Random Forest	0.65658	11632.5
LSTM	20.69643	7492.5
BiLSTM	18.67044	16150.5
GRU	7.54011	6493.5
CNN	4.19540	1598.4

Hibrido	35.71858	16650

Nota. La tabla presenta los tiempos de entrenamiento y prueba de cada modelo, permitiendo comparar la eficiencia computacional y los costos de procesamiento asociados a los diferentes enfoques de aprendizaje automático y aprendizaje profundo implementados en el sistema.

Fuente. Autoría propia.

Y para finalizar este análisis, se denota que en todas las métricas los modelos más recurrentes como los mejores son tres: *Random Forest, CNN y el modelo Híbrido*. En las métricas de eficiencia también hizo aparición el modelo *Naive Bayes*, por lo que se evidencia que con **cuatro modelos** se podría ofrecer al usuario un modelo distinto según el propósito que tenga en mente.

El propósito asignado a cada modelo se puede observar en la **Tabla 12**, en la cual, aunque solo se incluyen cuatro modelos, se presentan **ocho propósitos**. Esto se debe a que, al diferenciar dichos propósitos, se facilita la comprensión por parte del usuario. Sin embargo, los propósitos que comparten un mismo modelo podrían agruparse, aunque esto los haría más ambiguos para los usuarios.

Es evidente que estos modelos, para ser utilizados, deberán **reentrenarse** con el fin de mejorar su rendimiento individual y así cumplir con los indicadores definidos al inicio del proyecto.

Por último, el uso de estos cuatro modelos como definitivos **aún no es seguro**, ya que dependerá de los resultados obtenidos en la siguiente sección, donde se probará el uso de **Transformers**. En caso de que los Transformers presenten un rendimiento superior, los modelos actuales serán **descartados o comparados**, con el objetivo de determinar si pueden incluirse asignándoles propósitos específicos en los que superen a los modelos basados en Transformers.

Tabla 12

Modelos recomendados según propósito y tipo de desempeño

Propósito	Modelo Recomendado
Mejor modelo general.	CNN
Mejor en detección de noticias reales.	Random Forest
Mejor en detección de noticias falsas.	Hibrido
Mejor en detección equilibrada de ambos tipos de noticias.	CNN
Mejor en rapidez y eficiencia.	Naive Bayes
Mejor en estabilidad y reproducibilidad.	Naive Bayes
Modelo con menor sobreajuste.	Random Forest
Mejor en consistencia general.	CNN

Nota. La tabla resume los modelos recomendados de acuerdo con diferentes propósitos de uso, tales como desempeño general, detección específica de noticias reales o falsas, eficiencia, estabilidad y control del sobreajuste, facilitando la selección del modelo más adecuado según el contexto de aplicación. *Fuente.* Autoría propia.

Implementación con Transformers

Para los modelos de *Transformers* se utilizó *PyTorch*, ya que la librería de *Transformers* ya no es compatible con *TensorFlow*. *"Los transformadores son un tipo de arquitectura de red neuronal que transforma o cambia una secuencia de entrada en una secuencia de salida"* (¿Qué son los transformadores? Explicación de los transformadores en inteligencia artificial, AWS, s. f.).

Estos modelos son óptimos para el procesamiento de datos en forma de secuencias, como texto, imágenes o audio, ya que usan un mecanismo llamado *autoatención*, el cual les permite dar más peso a las partes más relevantes de la secuencia y así entender el contexto y las relaciones entre palabras a largo plazo.

Para su implementación se utilizó la librería *Transformers*, que permite acceder a los modelos de *Hugging Face*.

MBERT

Para el modelo *mBERT* se siguió un flujo similar al de los modelos anteriores. Se cargaron los datos, se *tokenizaron* con *AutoTokenizer*, se construyó el modelo con *"bert-base-multilingual-cased"*, y se entrenó durante **15** épocas.

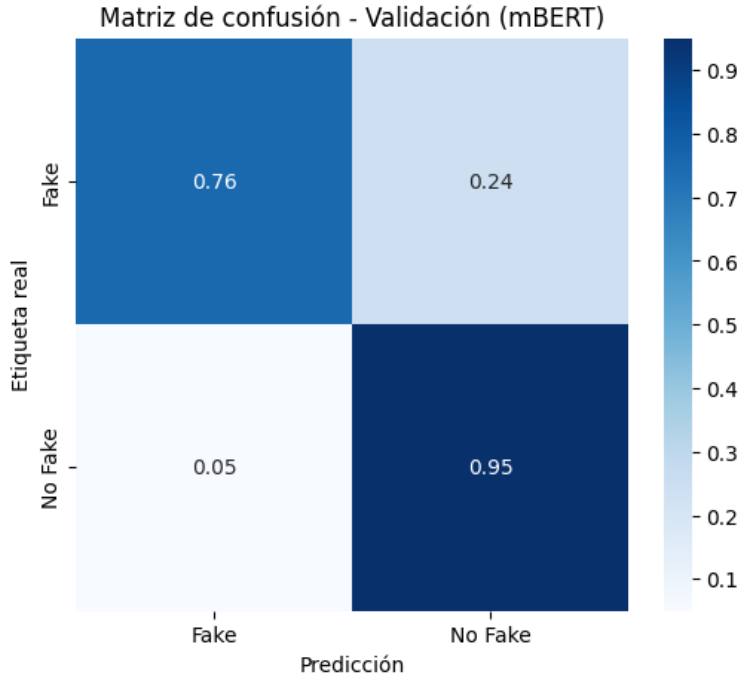
Este modelo fue evaluado individualmente, utilizando las mismas métricas que los modelos anteriores, comenzando por la *matriz de confusión*.

Como se observa en la **Figura 52**, el mejor modelo de *mBERT* obtuvo un **76%** de acierto en noticias falsas y un **95%** en noticias reales en el conjunto de validación, lo que indica un **24%** de error en la detección de noticias falsas y un **5%** de error en la detección de noticias reales.

En la **Figura 53** se puede observar que el mejor modelo de *mBERT* obtuvo los mismos valores: un **76%** de acierto en noticias falsas y un **95%** en noticias reales, manteniendo los mismos porcentajes de error en el conjunto de prueba.

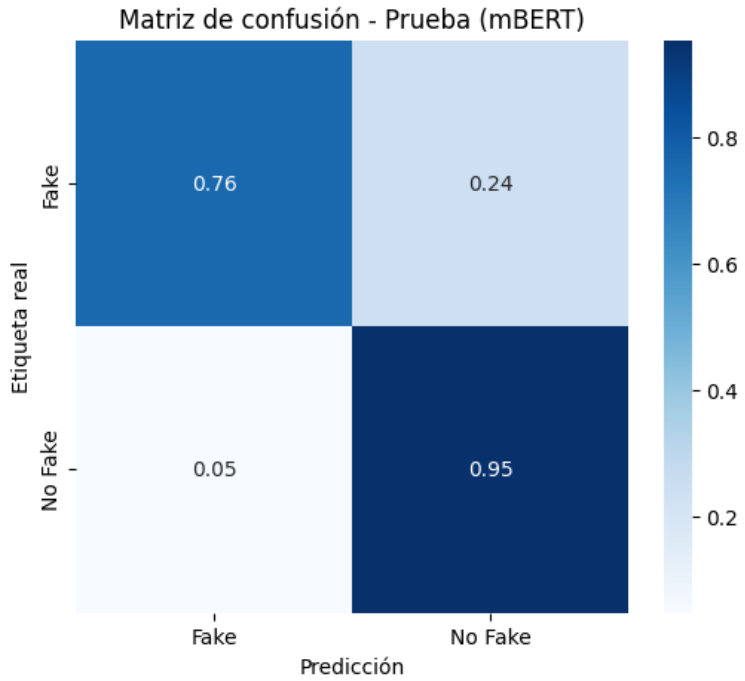
Esto indica que el modelo es bueno prediciendo tanto noticias reales como falsas, sin embargo, no supera al modelo *CNN* y su tiempo de entrenamiento es mucho mayor, incluso superando a modelos más pesados como *BiLSTM*.

Figura 52
Matriz de confusión – Validación – mBERT



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

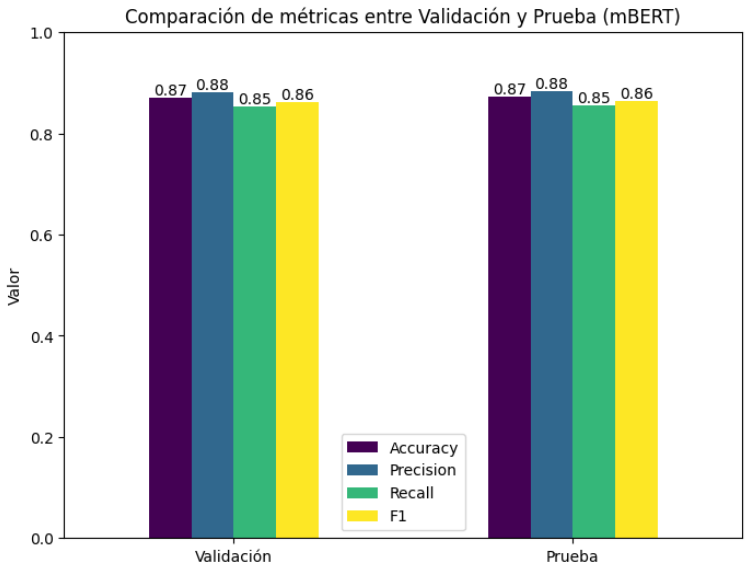
Figura 53
Matriz de confusión – Prueba – mBERT



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 54**, son bastante equilibradas, e incluso en algunas supera a *CNN*, lo cual podría ser positivo para su posible reutilización.

Figura 54
Comparación de métricas entre Validación y Prueba – mBERT



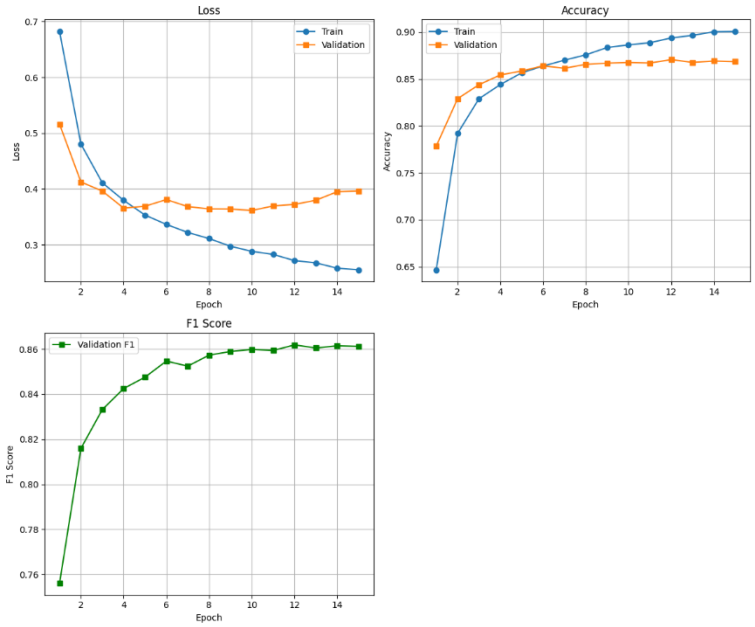
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

En la **Figura 55** se observa cómo cada una de las métricas se modificaba con cada *época*. Este modelo presenta gráficas bastante estables, tanto en validación como en prueba, con una distancia mínima entre ambas curvas, lo que sugiere que no hay problemas de sobreajuste *(overfitting)*.

El mejor modelo de *mBERT* obtuvo un score de **0.86185**, lo que lo posiciona como el mejor modelo en términos de generalización.

En conclusión, este modelo demuestra superioridad en varios aspectos, tal como se esperaría de un *Transformer*; sin embargo, es necesario analizar si las mejoras valen la pena considerando su alto tiempo de entrenamiento y el hecho de que es el modelo más pesado hasta el momento, siendo en términos de almacenamiento más de 10 veces más grande que el modelo más pesado de *Deep Learning*.

Figura 55
Evolución de métricas por épocas – mBERT



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

BETO

Al igual que con *mBERT*, para el modelo de *BETO* se siguió el mismo flujo: se utilizó *AutoTokenizer* y se construyó el modelo con *"dccuchile/bert-base-spanish-wwm-cased"*, entrenándolo durante **15** épocas.

Este tipo de *Transformer* tiene una particularidad que lo hace superior a *mBERT* al menos para este caso, y es que mientras *mBERT* maneja un diccionario multilingüe, *BETO* está entrenado exclusivamente en español.

Sin embargo, no se pudo aprovechar al máximo el potencial de *BETO* debido a incompatibilidades con la versión de PyTorch utilizada, y dado que otras librerías dependían de esa misma versión, no fue posible actualizarla. Esto provocó que se perdieran algunos datos en el proceso, afectando parcialmente el entrenamiento.

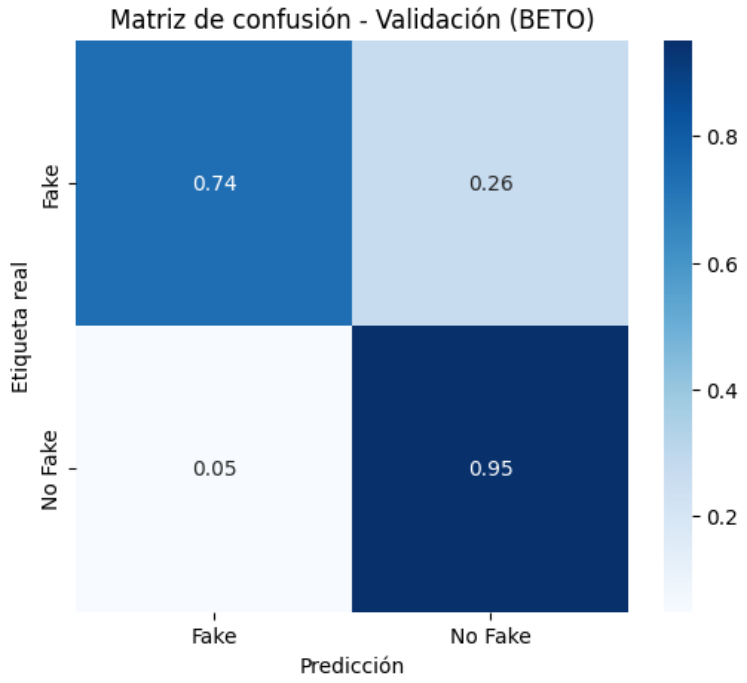
Este modelo fue evaluado individualmente, utilizando las mismas métricas que los modelos anteriores, comenzando por la *matriz de confusión*.

Como se observa en la **Figura 56**, el mejor modelo de *BETO* obtuvo un **74%** de acierto en noticias falsas y un **95%** en noticias reales en el conjunto de validación, lo que indica un **26%** de error en la detección de noticias falsas y un **5%** de error en la detección de noticias reales.

En la **Figura 57** se puede observar que el mejor modelo de *BETO* obtuvo valores similares: un **73%** de acierto en noticias falsas y un **95%** en noticias reales, con un **27%** de error en la detección de noticias falsas y un **5%** de error en las noticias reales.

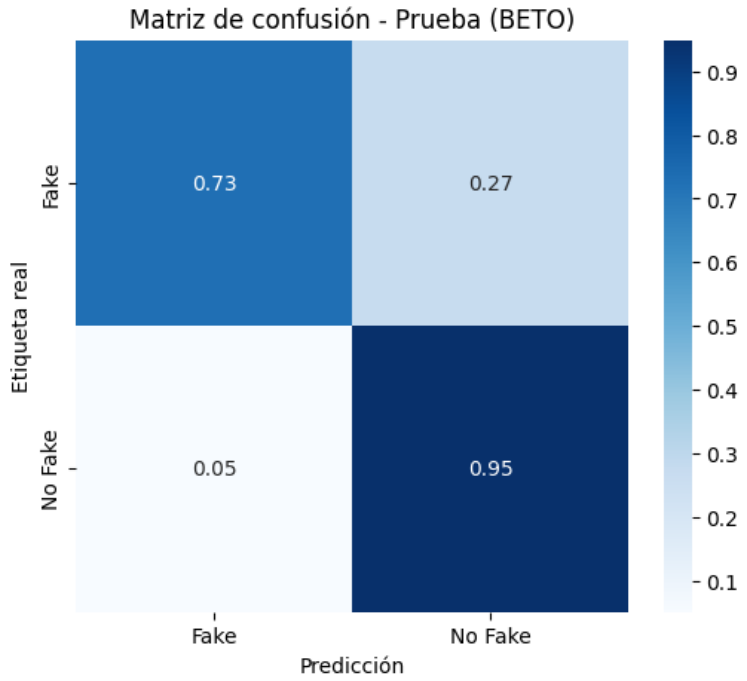
Esto indica que el modelo es bueno prediciendo tanto noticias reales como falsas, sin embargo, es inferior al modelo *mBERT*, lo que demuestra que la configuración y las incompatibilidades técnicas sí afectaron su rendimiento. Además, su tiempo de entrenamiento fue mayor que el de *mBERT*.

Figura 56
Matriz de confusión – Validación – BETO



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

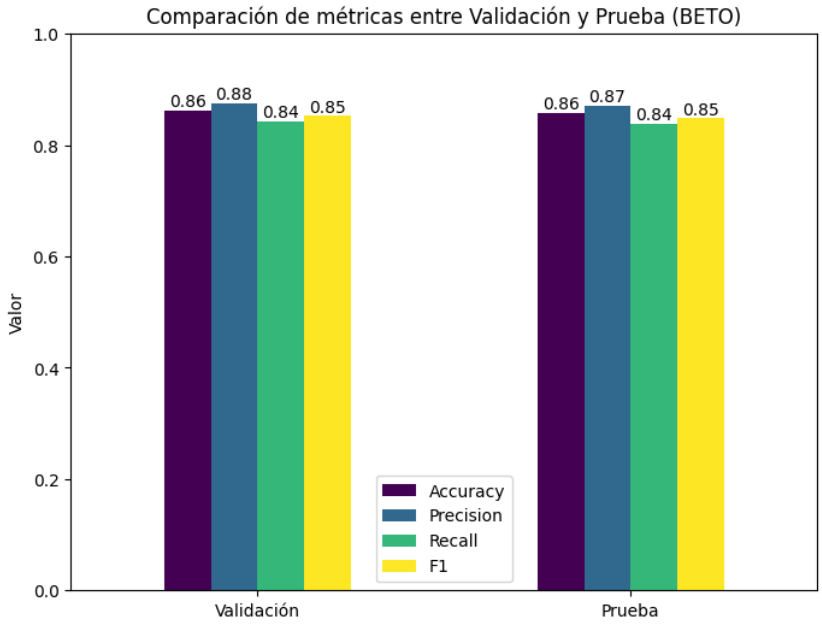
Figura 57
Matriz de confusión – Prueba – BETO



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Las demás métricas usadas para evaluar el modelo, como se observa en la **Figura 58**, son bastante equilibradas, aunque ligeramente inferiores a las obtenidas por *mBERT*.

Figura 58
Comparación de métricas entre Validación y Prueba – BETO



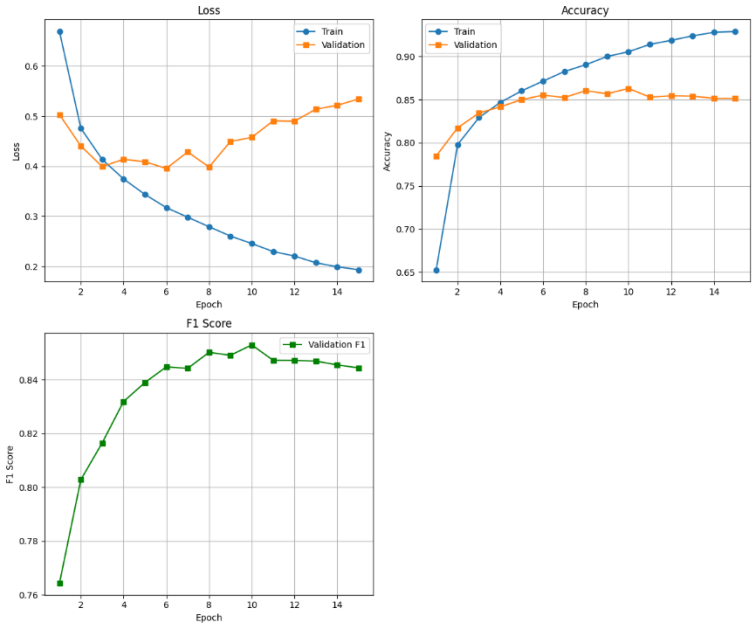
Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

En la **Figura 59** se observa cómo cada una de las métricas se modificaba con cada época. Este modelo presenta gráficas bastante estables, tanto en validación como en prueba, con una distancia mínima entre ambas curvas, aunque muestra signos más evidentes de *overfitting* que *mBERT*.

El mejor modelo de *BETO* obtuvo un score de **0.85295**, lo que lo posiciona como un buen modelo, aunque no supera a *CNN* ni a *mBERT*.

En conclusión, este modelo demuestra un buen desempeño en la detección de noticias reales y falsas, pero el entorno de desarrollo no le favoreció, afectando su rendimiento. Además, es un modelo pesado en términos de recursos computacionales, por lo que se deberá analizar en las próximas métricas si vale la pena reentrenarlo, considerando alternativas que reduzcan la influencia negativa del entorno de ejecución.

Figura 59
Evolución de métricas por épocas – BETO



Nota. Hecho con Seaborn. *Fuente.* Autoría propia.

Ahora, a partir de la **Tabla 13**, en la cual se observan las métricas del rendimiento según las matrices de confusión de *mBERT* y *BETO*, se procederá a compararlos con los mejores modelos del ranking: los tres mejores modelos en la detección de noticias reales y los tres mejores en la detección de noticias falsas, analizados anteriormente en la sección **4 de Evaluación de Resultados**.

Como se observa, no son malos modelos, pero tampoco los mejores. Aunque igualan el desempeño de algunos modelos, pierden relevancia debido al alto costo de reentrenamiento, ya que los demás modelos son mucho más rápidos y sencillos de reentrenar.

Así quedan las posiciones actuales:

Ranking de los mejores modelos en la detección de noticias reales:

1. Random Forest
2. Decision Tree
3. Naive Bayes / mBERT / BETO

Ranking de los mejores modelos en la detección de noticias falsas:

1. Híbrido
2. CNN / mBERT
3. BETO
4. Logistic Regression

Tabla 13

Resumen comparativo de matrices de confusión — Transformers Validación y Prueba

Modelo	Noticias Reales		Noticias Falsas	
mBERT	95%	95%	76%	76%
BETO	95%	95%	74%	73%

Nota. La tabla presenta un resumen comparativo de las matrices de confusión obtenidas durante las fases de validación y prueba de los modelos basados en *Transformers*, mostrando el porcentaje de clasificación correcta para noticias reales y falsas. *Fuente.* Autoría propia.

Siguiendo con las métricas principales de rendimiento, como se observa en la **Tabla 14**, se compararán los modelos de *mBERT* y *BETO* con el mejor modelo anterior de cada métrica, para así descubrir rápidamente si pueden ser utilizados por ser los mejores en algún tipo de propósito.

Comenzando con *Accuracy*, el modelo de *mBERT* supera a *CNN*, posicionándolo como el mejor en clasificación general, mientras que *BETO* lo supera en validación, pero es inferior en prueba. En *Precision*, ninguno de los dos modelos supera al modelo *Híbrido*. En *Recall*, tampoco superan al modelo *Random Forest*. En *F1-Score*, nuevamente *mBERT* supera a *CNN*, posicionándolo como el modelo más equilibrado, que no sacrifica detección por exactitud; *BETO* lo supera en validación, pero es inferior en prueba.

En *ROC AUC*, *mBERT* vuelve a superar a *CNN*, posicionándolo como el modelo con la mejor capacidad global para distinguir entre noticias reales y falsas, mientras que *BETO* es

inferior a *CNN*. Por último, en *Log Loss*, ninguno de los dos modelos supera a *CNN*, lo que significa que son menos confiables y de menor calidad que este.

Estos resultados demuestran que *mBERT* y *BETO* sí son mejores en algunas circunstancias; sin embargo, su mejora no es determinante.

Tabla 14

Métricas Principales de Rendimiento — Transformers

Modelo	Accuracy		Precision		Recall	
mBERT	0.87056	0.87227	0.84884	0.84934	0.94992	0.95258
BETO	0.86294	0.85873	0.83823	0.83454	0.95148	0.94879

Modelo	F1-Score		ROC AUC		Log Loss	
mBERT	0.89654	0.89800	0.93454	0.93338	0.34811	0.35101
BETO	0.89127	0.88800	0.91876	0.92093	0.43513	0.42654

Nota. La tabla muestra las principales métricas de rendimiento de los modelos *Transformers* evaluados, incluyendo *accuracy, precision, recall, F1-score, ROC AUC* y *log loss*, utilizadas para comparar su desempeño en la detección de noticias falsas en español. *Fuente.* Autoría propia.

Siguiendo con las *métricas de diagnóstico*, en *Specificity* ninguno de los dos modelos supera al modelo *Híbrido*, lo que significa que no son tan buenos para reducir falsos positivos. En *FPR*, nuevamente ninguno de los dos modelos supera al modelo *Híbrido*. En *FNR*, tampoco superan al modelo *Random Forest*.

En *Balanced Accuracy*, *mBERT* supera al modelo *CNN*, lo que lo posiciona como el mejor en equilibrio entre la detección de noticias reales y falsas, mientras que *BETO* no supera a *CNN*. En *NPV*, ninguno de los dos modelos supera a *Random Forest*.

Estos resultados demuestran que los modelos *Transformers* son similares a *CNN*, ya que en cada una de las métricas en las que se han posicionado como los mejores, reemplazaron directamente a *CNN*. Sin embargo, la mejora es tan pequeña que se debe analizar si vale la pena elegir *mBERT* o *BETO* por sobre *CNN*.

Tabla 15

Métricas de Diagnóstico y Error Específico — Transformers

Modelo	Specificity		FPR (False Positive Rate)		FNR (False Negative Rate)	
mBERT	0.75617	0.75657	0.24382	0.24342	0.05007	0.04741
BETO	0.73532	0.72899	0.26467	0.27100	0.04851	0.05120

Modelo	Balanced Accuracy		NPV (Negative Predictive Value)	
mBERT	0.85304	0.85457	0.91285	0.91718
BETO	0.84340	0.83889	0.91314	0.90811

Nota. La tabla presenta métricas de diagnóstico específicas para los modelos *Transformers*, tales como especificidad, tasas de falsos positivos y falsos negativos, exactitud balanceada y valor predictivo negativo, permitiendo evaluar la calidad del modelo en escenarios de clasificación crítica. *Fuente.* Autoría propia.

Continuando, se analizan las métricas de robustez, tal como se observa en la **Tabla 16**. En *MCC*, el modelo de *mBERT* vuelve a superar a *CNN*, lo que lo posiciona como el modelo que muestra la mayor correlación entre las predicciones y las etiquetas reales, mientras que *BETO* supera a *CNN* en validación, pero es inferior en prueba.

En *Cohen's Kappa*, *mBERT* nuevamente supera a *CNN*, lo que lo posiciona como el modelo que más predice consistentemente más allá de la casualidad, mientras que *BETO* es inferior. En *Overfitting Accuracy*, *mBERT* supera a *Random Forest*, posicionándolo como el modelo que mejor generaliza y menos tiende a memorizar los datos, mientras que *BETO* es inferior a *Random Forest*.

Por último, en el *Tipo de Problema*, *mBERT* presenta un buen ajuste (*good fit*), mientras que *BETO* muestra ligeros signos de *overfitting*.

Estos resultados confirman las similitudes entre *mBERT* y *CNN*, lo que podría consolidarlo como un modelo robusto. Sin embargo, deberán analizarse maneras más eficientes de entrenar y configurar el modelo.

Tabla 16
Métricas Avanzadas de Robustez — Transformers

Modelo	MCC (Matthews Correlation Coefficient)		Cohen's Kappa	
mBERT	0.73337	0.73728	0.72515	0.72867
BETO	0.71836	0.70948	0.70791	0.69887

Modelo	Overfitting Accuracy	Tipo de Problema
mBERT	0.0473	Good_Fit
BETO	0.0727	Slight_Overfitting

Nota. La tabla muestra métricas avanzadas de robustez para los modelos *Transformers*, incluyendo el coeficiente de *correlación de Matthews*, el *índice Kappa de Cohen* y el análisis de sobreajuste, con el fin de evaluar la estabilidad y confiabilidad del modelo seleccionado. *Fuente.* Autoría propia.

Finalmente, se observan las métricas de eficiencia, tal como se muestra en la **Tabla 17**. Los datos están expresados en segundos.

En el *Tiempo de Prueba*, el modelo *BETO* es más rápido que *mBERT*; sin embargo, ambos casi duplican el tiempo del modelo más demorado entre los modelos analizados anteriormente. Esto significa que están muy lejos de siquiera acercarse a *Naive Bayes*, que sigue siendo el modelo más rápido en prueba.

En el *Tiempo de Entrenamiento*, *BETO* vuelve a superar a *mBERT*; no obstante, es evidente la gran diferencia que existe entre estos dos modelos y los anteriores. El modelo anterior más demorado fue el *Híbrido*, con **16,650 segundos**, mientras que *BETO* tardó **287,647.2 segundos**. Es decir, aunque *BETO* se demoró menos que *mBERT*, la diferencia con el modelo *Híbrido* es tan grande que, en ese mismo tiempo, se podría entrenar el *Híbrido* casi **18 veces**.

Por lo tanto, con base en estos resultados, es evidente que, si se busca eficiencia y velocidad, los modelos *Transformers* no son los más óptimos, ni de cerca.

Tabla 17
Métricas de Eficiencia - Transformers

Modelo	Tiempo de Prueba	Tiempo de Entrenamiento
mBERT	57.43612	297634.95
BETO	54.41707	287647.2

Nota. La tabla presenta los tiempos de entrenamiento y prueba de los modelos *Transformers*, evidenciando el costo computacional asociado a este tipo de arquitecturas y permitiendo compararlas con modelos tradicionales y de aprendizaje profundo. *Fuente.* Autoría propia.

Luego de la evaluación realizada a todos los modelos, se puede tomar la decisión definitiva sobre cuál o cuáles se usarán en producción. Analizando los resultados, se identifica que, efectivamente, el modelo *mBERT* podría ser una buena opción para reemplazar el modelo *CNN*, ya que es similar, pero lo supera en varias métricas.

Sin embargo, el aspecto de la **eficiencia** hace que no sea sencillo tomar la decisión de implementarlo, por lo que, a pesar de ser un buen modelo, para este caso **no será utilizado**, al igual que *BETO*. Esto se debe a que su uso podría representar un riesgo de extender los plazos de desarrollo, poniendo en peligro la fecha de finalización del proyecto.

Por lo tanto, la decisión final es utilizar los modelos con su propósito específico, tal como se mencionó en la **Tabla 12**. Estos modelos serán reentrenados, esta vez empleando una arquitectura más profesional, con el fin de mejorar su rendimiento en los propósitos en los que destacaron individualmente, y no la que se utilizó para la evaluación inicial, la cual buscaba equilibrio entre rendimiento y velocidad.

Para los modelos de *Deep Learning*, se emplearán todas las técnicas, mientras que para los modelos de *Machine Learning* solo algunas, las técnicas que se usarán como *GloVe (un diccionario de embeddings preentrenados)*, *regularización L2* y *lematización*, además de aplicar estrategias para corregir el *overfitting* presente en los modelos *CNN* e *Híbrido*.

Como se puede observar en la **Tabla 18**, estas son las métricas de los modelos finales que se usaran en producción luego del reentrenamiento. Como se aprecia, después del reentrenamiento los modelos tuvieron una ligera mejora en la detección de noticias falsas y algunos empeoraron en la detección de noticias reales. Muchas métricas también empeoraron, pero esto se debe a que se priorizó la detección de noticias falsas por encima del rendimiento general.

También se nota que se corrigió el ligero *overfitting* que tenían los modelos de *CNN* e *Híbrido*. Además, se observa que los modelos clásicos, tanto *Naive Bayes* como *Random Forest*, aumentaron su complejidad, ya que su *Overfitting Accuracy* incrementó, y también lo hizo su tiempo de prueba.

Esto evidencia que el modelo de *CNN* sigue siendo el más apto para usarse, ya que, a pesar de que el modelo *Híbrido* es el mejor en la detección de noticias falsas, el modelo de *CNN* es el que obtiene las mejores puntuaciones en la mayoría de las métricas. Aun así, como ya se había planeado, los cuatro modelos se usarán para los distintos objetivos planteados anteriormente, únicamente modificando el rol del modelo *CNN* para que tome el lugar de *Naive Bayes* en el propósito de "Mejor en rapidez y eficiencia".

Tabla 18
Métricas de modelos reentrenados

Modelo	Naive Bayes		Random Forest		CNN		Hibrido	
Noticias Reales	92%	92%	98%	98%	93%	92%	89%	88%
Noticias Falsas	67%	65%	64%	64%	78%	78%	80%	80%
Accuracy	0.81852	0.81419	0.83758	0.83968	0.86925	0.86504	0.84901	0.84586
Precision	0.79969	0.79426	0.79485	0.79605	0.85928	0.85788	0.82529	0.82204
Recall	0.92410	0.92475	0.97707	0.97929	0.93100	0.92453	0.94413	0.94300
F1-Score	0.85740	0.85455	0.87659	0.87821	0.89370	0.88996	0.88072	0.87838
ROC AUC	0.87361	0.86309	0.89833	0.89169	0.93401	0.92992	0.91690	0.91088
Log Loss	0.47754	0.48440	0.52062	0.52279	0.29804	0.30660	0.32937	0.34116
Specificity	0.66634	0.65490	0.63650	0.63855	0.78023	0.77934	0.71190	0.70590
FPR	0.33365	0.34509	0.36349	0.36144	0.21976	0.22065	0.28809	0.29409
FNR	0.07589	0.07524	0.02292	0.02070	0.06899	0.07546	0.05586	0.05699
Balanced Accuracy	0.79522	0.78983	0.80679	0.80892	0.85562	0.85193	0.82801	0.82445

NPV	0.85897	0.85798	0.95064	0.95537	0.88694	0.87757	0.89838	0.89580
MCC	0.62362	0.61488	0.67633	0.68137	0.72852	0.71949	0.68902	0.68251
Cohen's Kappa	0.61144	0.60143	0.64628	0.65085	0.72458	0.71613	0.67757	0.67065
Overfitting Accuracy	0.02704		0.01761		0.04012		0.03121	
Tipo de Problema	Good_Fit		Good_Fit		Good_Fit		Good_Fit	
Tiempo de Prueba	19.17734		19.58762		3.51525		9.37657	

Nota. La tabla presenta las métricas obtenidas tras el proceso de reentrenamiento de los modelos seleccionados, permitiendo analizar la mejora en el desempeño, estabilidad y reducción del sobreajuste en comparación con las versiones iniciales evaluadas. *Fuente.* Autoría propia.

Desarrollo de API

El desarrollo de la API se realizó con el framework ***FastAPI***. Al analizar los requerimientos, se encontró que la interfaz web solo necesitaría acceso a un *endpoint* para permitir que los usuarios puedan usar los modelos de detección de noticias falsas. Este *endpoint* sería **/predict**.

Sin embargo, considerando que otros desarrolladores podrían usar esta API, se crearon otros seis *endpoints*. El prefijo que comparten todos es **/api/v1**.

Los siete *endpoints* y su funcionamiento son los siguientes:

1. **/predict:**

   Este *endpoint* permite hacer una petición **POST** al servidor. Recibe como parámetros el texto de la noticia que será evaluada y el modelo que se desea usar, los cuales deben enviarse con las etiquetas **text** y **model**.

   Si la petición se procesa correctamente, la respuesta del servidor devolverá un **JSON** con la siguiente estructura:


```
{
  “timestamp”: “2025-11-07 12:45:12”,
  “status_code”: 200,
  “response”: {
    “prediction”: “real”,
    “confidence”: 0.6942,
    “model_used”: “CNN”,
    “inference_time_ms”: 59324.275,
    “tokens_count”: 10,
```

"prediction_id": "pred_1762537511955"
}
}

2. **/confidence/{prediction_id}:**

Este *endpoint* permite hacer una petición **GET** al servidor. Al incluir el ID de la predicción, el servidor devuelve la confianza que el modelo tiene sobre su predicción.

Si la petición se procesa correctamente, la respuesta tendrá la siguiente estructura:

"/confidence/pred_1762536423631": {
"timestamp": "2025-11-07 12:27:03",
"status_code": 200,
"response": {
"prediction_id": "pred_1762536423631",
"prediction": "real",
"confidence": 0.7789,
"model_used": "CNN",
"timestamp": "2025-11-07T12:27:03.631473"
}
}

3. **/models:**

Este *endpoint* permite realizar una petición **GET**. Si se procesa correctamente, la respuesta del servidor será un listado de los modelos disponibles con información básica, con una estructura similar a la siguiente:

{

“timestamp”: “2025-11-07 12:45:58”,

“status_code”: 200,

“response”: {

“models”: [

{

“name”: “Naive Bayes”,

“description”: “Clasificador probabilístico basado en el teorema de Bayes con TF-IDF.”,

“type”: “Machine Learning”,

“status”: true,

“size_mb”: 22.32,

“last_used”: “N/A”

},

{

“name”: “Random Forest”,

“description”: “Modelo de ensamble de árboles de decisión con TF-IDF.”,

“type”: “Machine Learning”,

“status”: true,

“size_mb”: 35.36,

“last_used”: “N/A”

},

{

“name”: “CNN”,

"description": "Red neuronal convolucional para clasificación de texto.",

"type": "Deep Learning",

"status": true,

"size_mb": 110.52,

"last_used": "2025-11-07 12:45:57"

},

{

"name": "Híbrido",

"description": "Modelo híbrido CNN + BiLSTM + GRU con mecanismo de atención.",

"type": "Deep Learning",

"status": true,

"size_mb": 52.02,

"last_used": "N/A"

}

]

}

}

4. **/reload:**

Este *endpoint* permite hacer una petición **POST**, la cual recarga un modelo. Es útil para actualizar modelos sin reiniciar el servidor.

Debe recibir el nombre del modelo en la etiqueta **model**.

La respuesta tendrá una estructura como la siguiente:

{
“timestamp”: “2025-11-07 12:46:20”,
“status_code”: 200,
“response”: {
“status”: “success”,
“message”: “Modelo 'CNN' recargado correctamente.”,
“reload_time_ms”: 20237.489
}
}

5. **/metrics?model={model}:**

Este *endpoint* permite realizar una petición **GET** al servidor. Al agregar el nombre del modelo, el servidor devuelve un JSON con sus métricas y la fecha de su última actualización.

{
“timestamp”: “2025-11-07 12:46:21”,
“status_code”: 200,
“response”: {
“metrics”: {
“model”: “CNN”,
“accuracy”: 0.8634691195795007,
“precision”: 0.8474542161400684,
“recall”: 0.9374443455031166,
“f1_score”: 0.8901807419934468,

“auc”: 0.9215264811164543,

“updated_at”: “2025-10-17 07:51:55”

}

}

}

6. /health:

Este *endpoint* permite realizar una petición **GET**, que identifica el estado de salud de la API. Devuelve información relacionada con su funcionamiento.

{

“timestamp”: “2025-11-07 12:46:21”,

“status_code”: 200,

“response”: {

“status”: “ok”,

“uptime”: “0h 2m 10s”,

“framework”: “FastAPI”,

“tensorflow_version”: “2.20.0”,

“gpu_available”: false,

“loaded_models”: [

“Naive Bayes”,

“Random Forest”,

“CNN”,

“Híbrido”

]

```
    }
}
```


7. **/logs:**

Este *endpoint* permite realizar una petición **GET**, la cual devuelve un listado de los 100 *logs* más recientes del servidor.


```
{
  “timestamp”: “2025-11-07 12:46:21”,
  “status_code”: 200,
  “response”: {
    “logs”: [
      {
        “timestamp”: “2025-11-07 12:45:57”,
        “model”: “CNN”,
        “text”: “Test para confianza”,
        “prediction”: “real”,
        “confidence”: 0.7789
      },
      {
        “timestamp”: “2025-11-07 12:45:11”,
        “model”: “CNN”,
        “text”: “Los científicos descubren una nueva forma de producir energía limpia.”,
        “prediction”: “real”,
        “confidence”: 0.6942
```

```
            }
        ]
    }
}
```

Desarrollo de Interfaz Web

Habiendo finalizado la creación e implementación de la *API* con *FastAPI*, se procedió a desarrollar la interfaz web con la cual el usuario final interactuará. Esta se realizó utilizando ***React***, con ***Vite.js*** como herramienta principal de desarrollo, complementada con otras librerías como ***Bootstrap***, ***react-router-dom*** y ***mui-x-charts***.

Tal como se observaba en los prototipos de las **Ilustraciones 1 a 4**, solo se requerían dos pantallas; por ello, el proyecto se dividió en dos páginas principales: **index** y **resultado**.

Como se aprecia en la **Figura 60**, la pantalla de inicio (home) cuenta con una sección tipo ***textarea*** para ingresar el texto de la noticia que se desea evaluar. Esta entrada requiere un mínimo de **40 caracteres** para poder procesarse. También incluye un ***select*** que permite al usuario elegir el modelo que prefiera según sus criterios. Finalmente, un botón envía la noticia y el modelo seleccionado al *backend* a través de la *API*.

Posteriormente, la aplicación redirige a la pantalla mostrada en la **Figura 61**, donde se visualiza la respuesta del *backend* transformada en un **gráfico**, acompañado de una breve sección que explica los resultados obtenidos. De igual manera, dentro de esta pantalla el usuario tiene la posibilidad de realizar una nueva consulta sin necesidad de regresar a la pantalla de inicio.

Figura 60
Pantalla de home de la Interfaz

Nota. Interfaz vista desde Brave. *Fuente.* Autoría propia.

Figura 61
Pantalla de resultado de la Interfaz

Nota. Interfaz vista desde Brave. *Fuente.* Autoría propia.

Pruebas y Validación del Sistema

Para comenzar con las pruebas, validación, análisis de resultados y limitaciones del proyecto, es importante mencionar que el modelo de ***CNN*** **superó los 100 MB**, por lo que *GitHub* no permite su subida. Por ello, se especifican en la **Tabla 19** los parámetros que dieron como resultado el mejor modelo de *CNN*:

Tabla 19
Parámetros del mejor modelo de CNN

num_filters	160
kernel_sizes	[3, 4, 5]
dropout_rate	0.5
l2_reg	0.001
embedding_dim	300

Nota. La tabla describe los *hiperparámetros* utilizados en la configuración del mejor modelo basado en redes neuronales convolucionales *(CNN)*, los cuales fueron ajustados durante el proceso de optimización para maximizar el desempeño del sistema. *Fuente.* Autoría propia.

Empezando por las pruebas y validación, se levantaron la *API* y la interfaz web localmente. Ambos iniciaron correctamente, indicando que, de momento, no hay inconvenientes. Posteriormente, se realizaron pruebas pegando noticias y verificando que la interfaz web estaba consumiendo correctamente la *API*, tal como se observa en la **Figura 62**.

Se comprobó el uso con cada una de las opciones de modelos y ninguna presentó fallos a simple vista. Revisando las consolas, tampoco se evidenció ningún error.

Tal como se había planeado, la interfaz **no permite enviar peticiones al *backend* si el texto no supera los 40 caracteres**. Además, si el usuario no selecciona un modelo específico, no habrá fallo, ya que se estableció uno como predeterminado.

La interfaz tampoco permite acceder a la página **/resultado** si no se ha realizado antes una predicción, redireccionando al **/home**.

Adicionalmente, y como un plus, la interfaz funciona correctamente en cuanto al flujo de interacción únicamente con teclado. También mantiene visible el foco todo el tiempo y, al tener buen contraste, permite un uso adecuado para la mayoría de las personas con discapacidades.

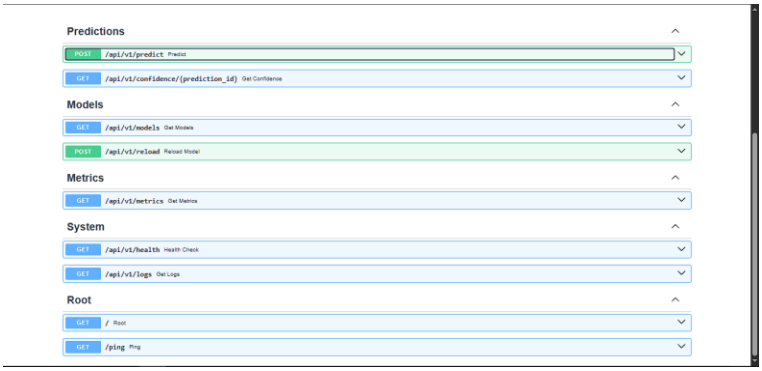
Figura 62
Validación de la Interfaz Web

Nota. Interfaz vista desde Brave. *Fuente.* Autoría propia.

Ahora validamos el funcionamiento de la *API*. Esto podría hacerse con programas como *Postman*, pero la documentación automática de *FastAPI* también lo permite, por lo que utilizamos esta.

Tal como se observa en la **Figura 63**, la *API* se divide en cinco categorías y cada una, a excepción de *Metrics*, tiene dos *endpoints*, para un total de nueve *endpoints*.

Figura 63
Endpoints de la API



Nota. Endpoints por Swagger. *Fuente.* Autoría propia.

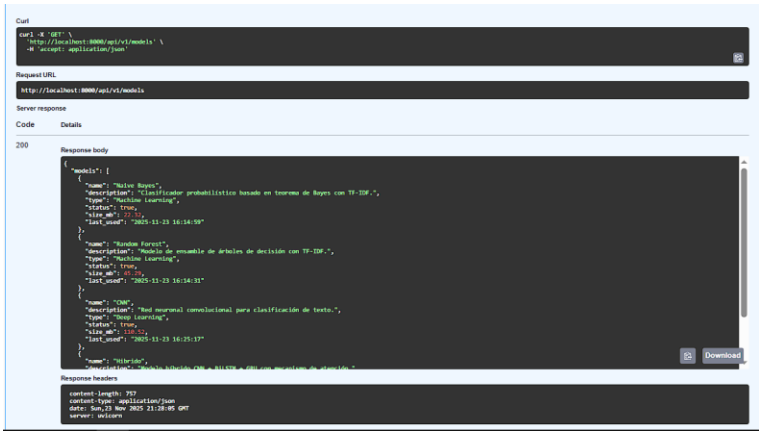
Se probó el funcionamiento del *endpoint* de predicción y, como se observa en la **Figura 64**, la *API* respondió correctamente devolviendo un *JSON* con la predicción, la confianza, el modelo usado, el tiempo de inferencia en milisegundos, la cantidad de tokens y el ID de predicción.

Figura 64
Validación del Endpoint de Predicción

Nota. Endpoints por Swagger. *Fuente.* Autoría propia.

También se probó el *endpoint* que lista los modelos disponibles y, como se puede ver en la **Figura 65**, la *API* continúa sin presentar fallos ni errores. Devuelve un *JSON* con el nombre del modelo, una descripción, el tipo, el tamaño del archivo y su estatus.

Figura 65
Validación del Endpoint de Lista de Modelos



Nota. Endpoints por Swagger. *Fuente.* Autoría propia.

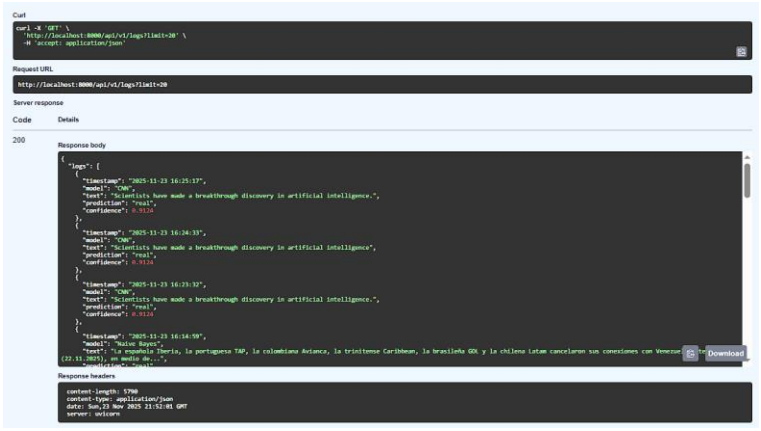
Asimismo, se probó el *endpoint* para obtener las métricas de cada modelo. Se solicitaron las métricas del modelo *CNN* y la *API* respondió correctamente, como se observa en la **Figura 66**.

Figura 66
Validación del Endpoint de Métricas

Nota. Endpoints por Swagger. *Fuente.* Autoría propia.

Por último, se probó el *endpoint* de *logs* y, al igual que los anteriores, no presentó ningún tipo de fallo, tal como se muestra en la **Figura 67**.

Figura 67
Validación del Endpoint de Logs



Nota. Documentación de API por Swagger. *Fuente.* Autoría propia.

Estos fueron los *endpoints* más importantes y los que podrían provocar algún fallo relevante, concluyendo que la *API* funciona correctamente.

Por último, en la validación y prueba del sistema completo se usaron noticias cortas, largas, ambiguas, con errores ortográficos, de diversos temas, en otros idiomas y algunas generadas por *IAs*. Tal como se observa en la **Tabla 20**, las predicciones son bastante consistentes entre todos los modelos; sin embargo, también se evidencia que la diferencia del *dataset*, con un **59%** de noticias reales y un **41%** de noticias falsas, provocó que los modelos terminaran priorizando las noticias reales, en especial los modelos clásicos, ya que estos fallan al detectar las noticias falsas.

También parece que, aunque el modelo *Híbrido* carecía de rendimiento en muchas métricas, en esta validación práctica fue el que mejor se desempeñó. Detectó noticias falsas y reales correctamente, incluso en otros idiomas y con errores ortográficos. No obstante, aunque reconoce las noticias falsas, presenta una **baja confianza** en sus predicciones.

El siguiente fue el modelo *CNN*, que demuestra ser el más equilibrado. Por esto, recomendaría hacer una misma predicción usando el modelo *Híbrido*, *Random Forest* y *CNN* para verificar cuál es la predicción más probable de ser correcta.

En cuanto a las noticias en otros idiomas, los modelos clásicos parecen más sensibles a fallar, mientras que *CNN* e *Híbrido* se muestran más flexibles, adaptándose correctamente a pesar de no haber sido entrenados en otros idiomas aparte del español.
También se evidencia que el peor modelo es *Naive Bayes*.

Tabla 20
Validación del Sistema con Múltiples Noticias

Modelo	Noticia Resumida	Predicción	Observación
Hibrido	¿Quién es el general Huertas? El oficial reincorporado por Petro con presuntos nexos con disidencias La información encontrada en el …	Real	88% de Confianza
Random Forest	¿Quién es el general Huertas? El oficial reincorporado por Petro con presuntos nexos con disidencias La información encontrada en el …	Real	55.5% de Confianza
CNN	¿Quién es el general Huertas? El oficial reincorporado por Petro con presuntos nexos con disidencias	Real	88% de Confianza

	La información encontrada en el …		
Naive Bayes	¿Quién es el general Huertas? El oficial reincorporado por Petro con presuntos nexos con disidencias La información encontrada en el …	Real	55.3% de Confianza
Hibrido	Fuerzas Militares investigan la probable filtración de información a disidencias de alias “Calarcá” Además de las Fuerzas Militares, la Procuraduría y…	Real	89.1% de Confianza
Random Forest	Fuerzas Militares investigan la probable filtración de información a disidencias de alias “Calarcá” Además de las Fuerzas Militares, la Procuraduría y…	Real	57.3% de Confianza
CNN	Fuerzas Militares investigan la probable filtración de información a disidencias de alias “Calarcá” Además de las Fuerzas Militares, la Procuraduría y…	Real	88.6% de Confianza
Naive Bayes	Fuerzas Militares investigan la probable filtración de información a disidencias de alias “Calarcá” Además de las Fuerzas Militares, la Procuraduría y…	Real	60.7% de Confianza
Hibrido	“Nuestra hermana murió por las teorías conspirativas de nuestra madre sobre el cáncer” Gabriel y Sebastian Shemirani observaban con preocupación cómo…	Falsa	41% de Confianza
Random Forest	“Nuestra hermana murió por las teorías conspirativas de nuestra madre sobre el cáncer” Gabriel y Sebastian Shemirani observaban con preocupación cómo…	Real	53.9% de Confianza
CNN	“Nuestra hermana murió por las teorías conspirativas de nuestra madre sobre el cáncer” Gabriel y Sebastian Shemirani observaban con preocupación cómo…	Falsa	14.6% de Confianza
Naive Bayes	“Nuestra hermana murió por las teorías conspirativas de nuestra madre sobre el cáncer” Gabriel y Sebastian Shemirani observaban con preocupación cómo…	Real	78% de Confianza
Hibrido	La princesa Kate pide disculpas por la “confusión” que causó su foto retirada por las agencias de noticias por inconsistencias Catherine…	Falsa	41.4% de Confianza
Random Forest	La princesa Kate pide disculpas por la “confusión” que causó su foto retirada por las agencias de noticias por inconsistencias Catherine…	Real	56.7% de Confianza
CNN	La princesa Kate pide disculpas por la “confusión” que causó su foto retirada por las agencias de noticias por inconsistencias Catherine…	Real	71.2% de Confianza
Naive Bayes	La princesa Kate pide disculpas por la “confusión” que causó su foto retirada por las agencias de noticias por inconsistencias Catherine…	Real	68.6% de Confianza
Hibrido	Emisarios de un oligarca próximo a Putin agitan a la ultraderecha en Madrid: “Estamos ganando la batalla de la propaganda” Bajo…	Falsa	5.8% de Confianza
Random Forest	Emisarios de un oligarca próximo a Putin agitan a la ultraderecha en Madrid: “Estamos ganando la batalla de la propaganda” Bajo…	Real	56.2% de Confianza
CNN	Emisarios de un oligarca próximo a Putin agitan a la ultraderecha en Madrid: “Estamos ganando la batalla de la propaganda” Bajo…	Falsa	30.3% de Confianza

Naive Bayes	Emisarios de un oligarca próximo a Putin agitan a la ultraderecha en Madrid: “Estamos ganando la batalla de la propaganda” Bajo…	Real	67.8% de Confianza
Hibrido	Este vídeo de un aterrizaje forzoso de un helicóptero estadounidense está hecho con IA aunque Grok lo de por cierto Circula…	Real	57.2% de Confianza
Random Forest	Este vídeo de un aterrizaje forzoso de un helicóptero estadounidense está hecho con IA aunque Grok lo de por cierto Circula…	Falsa	47.5% de Confianza
CNN	Este vídeo de un aterrizaje forzoso de un helicóptero estadounidense está hecho con IA aunque Grok lo de por cierto Circula…	Falsa	47.7% de Confianza
Naive Bayes	Este vídeo de un aterrizaje forzoso de un helicóptero estadounidense está hecho con IA aunque Grok lo de por cierto Circula…	Real	67.8% de Confianza
Hibrido	La limitasión de la UE a la pesca a 27 días al año es solo para la pesca de arrastre que…	Falsa	33.8% de Confianza
Random Forest	La limitasión de la UE a la pesca a 27 días al año es solo para la pesca de arrastre que…	Real	60.5% de Confianza
CNN	La limitasión de la UE a la pesca a 27 días al año es solo para la pesca de arrastre que…	Real	83.2% de Confianza
Naive Bayes	La limitasión de la UE a la pesca a 27 días al año es solo para la pesca de arrastre que…	Real	85.7% de Confianza
Hibrido	Canacol Energy se acoge ahora a ley de quiebras en EE. UU., días después de declararse en insolvencia en Canadá Canacol…	Falsa	3% de Confianza
Random Forest	Canacol Energy se acoge ahora a ley de quiebras en EE. UU., días después de declararse en insolvencia en Canadá Canacol…	Real	59.1% de Confianza
CNN	Canacol Energy se acoge ahora a ley de quiebras en EE. UU., días después de declararse en insolvencia en Canadá Canacol…	Real	68.3% de Confianza
Naive Bayes	Canacol Energy se acoge ahora a ley de quiebras en EE. UU., días después de declararse en insolvencia en Canadá Canacol…	Real	80.6% de Confianza
Hibrido	Things to know about the growing pressures facing Zelenskyy during a crucial week of diplomacy KYIV, Ukraine (AP) — Ukrainian President Volodymyr…	Real	76.8% de Confianza
Random Forest	Things to know about the growing pressures facing Zelenskyy during a crucial week of diplomacy KYIV, Ukraine (AP) — Ukrainian President Volodymyr…	Falsa	49.9% de Confianza
CNN	Things to know about the growing pressures facing Zelenskyy during a crucial week of diplomacy KYIV, Ukraine (AP) — Ukrainian President Volodymyr…	Real	78.9% de Confianza
Naive Bayes	Things to know about the growing pressures facing Zelenskyy during a crucial week of diplomacy KYIV, Ukraine (AP) — Ukrainian President Volodymyr…	Falsa	39% de Confianza
Hibrido	There are more questions than answers after NFL games Sunday The Kansas City Chiefs saved their season. The Philadelphia Eagles gave…	Real	73.3% de Confianza
Random Forest	There are more questions than answers after NFL games Sunday The Kansas City Chiefs saved their season. The Philadelphia Eagles gave…	Real	59% de Confianza
CNN	There are more questions than answers after NFL games Sunday	Real	58.9% de Confianza

Naive Bayes	The Kansas City Chiefs saved their season. The Philadelphia Eagles gave… There are more questions than answers after NFL games Sunday The Kansas City Chiefs saved their season. The Philadelphia Eagles gave…	Falsa	48.5% de Confianza
Hibrido	Ministerio de Salud anuncia ampliación del programa de vacunación para zonas rurales El Ministerio de Salud confirmó este lunes la ampliación…	Real	86.8% de Confianza
Random Forest	Ministerio de Salud anuncia ampliación del programa de vacunación para zonas rurales El Ministerio de Salud confirmó este lunes la ampliación…	Real	53.4% de Confianza
CNN	Ministerio de Salud anuncia ampliación del programa de vacunación para zonas rurales El Ministerio de Salud confirmó este lunes la ampliación…	Real	72.2% de Confianza
Naive Bayes	Ministerio de Salud anuncia ampliación del programa de vacunación para zonas rurales El Ministerio de Salud confirmó este lunes la ampliación…	Real	70.9% de Confianza
Hibrido	Un informe “filtrado” asegura que Colombia implementará un impuesto único del 1% a todas las compras digitales desde septiembre En redes…	Falsa	37.9% de Confianza
Random Forest	Un informe “filtrado” asegura que Colombia implementará un impuesto único del 1% a todas las compras digitales desde septiembre En redes…	Real	52.9% de Confianza
CNN	Un informe “filtrado” asegura que Colombia implementará un impuesto único del 1% a todas las compras digitales desde septiembre En redes…	Real	84% de Confianza
Naive Bayes	Un informe “filtrado” asegura que Colombia implementará un impuesto único del 1% a todas las compras digitales desde septiembre En redes…	Real	76.6% de Confianza

Nota. La tabla presenta los resultados de la validación del sistema utilizando múltiples noticias reales y falsas, mostrando las predicciones generadas por distintos modelos y el nivel de confianza asociado, con el fin de evaluar el comportamiento del sistema en escenarios reales de uso. *Fuente.* Autoría propia.

Continuando con los resultados, al principio del proyecto se establecieron **6 resultados esperados**. Ahora, al estar en la fase final del proyecto, se verifica si dichos resultados se cumplieron o no.

Empecemos con el **resultado esperado número 1**, el cual menciona que se espera obtener un modelo de *machine learning* entrenado para detección de *fake news* en español. El

indicador para cumplirlo es obtener en *precisión* y *recall* porcentajes superiores al **90%** en el conjunto de prueba.

Entonces, el modelo de *machine learning* entrenado para detección de *fake news* sí se obtuvo. En *recall* se logró un porcentaje superior al **97%**; sin embargo, en *precisión* no se consiguió el objetivo, ya que el mejor modelo obtuvo solo un porcentaje ligeramente superior al **85%**. Aun así, por esta razón, en lugar de un solo modelo se dispusieron **cuatro**, para mitigar esta pequeña deficiencia. Por lo tanto, podría concluir que este resultado esperado se cumplió, y la comunidad académica y los usuarios interesados en *IA* podrán beneficiarse.

Continuando con el **resultado esperado número 2**, este consiste en una comparativa entre los modelos entrenados, y su indicador es un documento con tablas de métricas. Dentro de este documento se encuentran registradas múltiples tablas y gráficas comparativas; citando algunas, van desde la **Tabla 7 hasta la Tabla 18**, entre las cuales me atrevería a mencionar que la **Tabla 18** es la más importante, ya que con una sola mirada permite saber rápidamente de qué carece y en qué es mejor cada uno de los **4** modelos elegidos al final.

Es por esta razón que considero que este resultado esperado también se cumple satisfactoriamente, por lo que investigadores y estudiantes de ingeniería pueden beneficiarse de este documento.

Ahora, el **resultado esperado número 3** consiste en el desarrollo de una *API* con *FastAPI* para exponer el modelo, y su indicador es contar con una *API* funcional con *endpoints* disponibles para consulta. Tal como se observa en la **actividad 6** de este proyecto, la *API* fue desarrollada completamente con *FastAPI*, incluyendo documentación automática con ejemplos y descripciones. Incluso, dentro de los *tests* unitarios realizados se puede ver el funcionamiento de cada *endpoint*.

El proyecto incluye el archivo *requirements.txt* para levantar fácilmente la *API*, y también un *script* de *PowerShell* para hacerlo más sencillo. En el *Readme* está explicado, por lo que considero que este resultado esperado también está satisfactoriamente completo. Así, tanto los desarrolladores como los usuarios finales podrán beneficiarse.

El siguiente es el **resultado esperado número 4**. Este consiste en el desarrollo de la *interfaz web* en *React* para la interacción del usuario, y su indicador es una página web que permita ingresar la noticia y obtener el porcentaje de veracidad. Tal como se observa en la **actividad 7** del proyecto, la *interfaz web* se desarrolló correctamente en *React*, incluyendo como plus algunos elementos de diseño accesible. La interfaz es sencilla, pero no impráctica ni inútil. Dentro del *Readme* del repositorio también está especificado cómo levantar el *frontend*, y el enlace al repositorio se encuentra en el **Anexo A**. Por lo tanto, considero que este resultado esperado también se cumplió correctamente, y el público general y los usuarios de internet se beneficiarían de esto.

El siguiente es el **resultado esperado número 5**. Este consiste en la documentación técnica y académica del proyecto, y su indicador es un informe final con metodología, resultados, limitaciones y conclusiones. Dentro de este documento ya se encuentra la mayoría de esos elementos, y más adelante se incluirán las limitaciones y conclusiones, además de que se cuenta con la documentación del código y el *Readme*. Por lo tanto, considero que este resultado esperado también se completa con éxito, por lo que la universidad, los docentes y futuros investigadores se verán beneficiados por esto.

Por último, el **resultado esperado número 6** consiste en la validación del sistema con noticias de prueba, y su indicador es el registro de pruebas y resultados de validación interna. Este punto se realizó en esta misma **actividad** y se registró en la **Tabla 20**, dando un análisis de

los resultados obtenidos. Por lo tanto, considero que este punto también se completa exitosamente, y tanto el autor como la comunidad académica nos beneficiamos de esto.

Ahora, con las limitaciones del proyecto, empecemos por lo más evidente: estos modelos no identifican las noticias falsas por ser falsas, sino que las detectan **porque están escritas como falsas**. Es decir, los modelos no cuentan con el contexto suficiente para determinar si un hecho o afirmación dentro de una noticia es verdadera o falsa; lo que hacen es aprender patrones de escritura y palabras que, solas o combinadas, aumentan la probabilidad de que la noticia sea falsa. Esto implica que, si una noticia real se escribe con características propias de una falsa, será detectada como falsa, y si una noticia falsa se escribe con estilo de noticia real, será detectada como real.

Con el acceso a asistentes de *IA* como *ChatGPT*, es posible crear noticias falsas que suenen completamente reales. Por esta razón, los modelos necesitarían un reentrenamiento con más contexto, *datasets* más variados y técnicas más avanzadas. Para que realmente pudieran identificar si un evento es verdadero o falso, deberían tener acceso a plataformas externas que permitan verificar hechos o buscar indicios de veracidad, lo cual está fuera del alcance de este proyecto, ya que solo se planteó trabajar únicamente con *PLN*.

Además, las noticias cambian rápidamente con el tiempo. Como vimos al principio, los medios buscan adaptarse para captar la mayor atención posible, incluso si eso implica alterar las noticias para hacerlas más llamativas. Por esto, el formato de las noticias cambia y el límite entre noticias reales y falsas se vuelve más difuso, lo que puede hacer que los modelos “envejezcan” rápidamente.

Otra limitación importante fue el *dataset*. Aunque contaba con un gran número de registros y la diferencia entre noticias reales y falsas era mínima, al final sí afectó el

entrenamiento: todos los modelos terminaron priorizando las noticias reales, incluso usando técnicas para equilibrar la importancia de las noticias falsas. Además, aunque el *dataset* era amplio, sigue existiendo el riesgo de que las noticias analizadas en producción tengan estructuras, vocabulario o enfoques muy diferentes a las del entrenamiento. Por ello, como recomendación futura, se sugiere incluir un módulo de *web scraping* para crear un *dataset* más amplio y diverso.

Debido a que los modelos aprenden patrones, presentan limitaciones al interpretar elementos lingüísticos como ironía o sarcasmo. El contexto cultural también sería importante de considerar. Otra limitación es que, al analizar únicamente texto, el sistema pierde información clave presente en enlaces, imágenes o videos, elementos que muchas noticias utilizan para reforzar su credibilidad o persuadir al lector.

Los modelos tipo *Transformers* requieren recursos elevados de *hardware* para entrenamiento e inferencia, lo que dificulta probar, ajustar y entrenar continuamente. Esto también ocurre con modelos clásicos y de *deep learning* cuando sus arquitecturas crecen. Al añadir técnicas avanzadas, el tiempo de entrenamiento aumenta considerablemente, lo que dificulta realizar múltiples pruebas. Finalmente, el desempeño de los modelos puede mejorar o empeorar según técnicas como *lematización*, *tokenización* o *normalización*: pequeños cambios en el preprocesamiento pueden generar diferencias notables. Esto se observó con los modelos *CNN* e *Híbrido*, en los cuales la *lematización* mejoraba el rendimiento del modelo de *CNN*, pero empeoraba el del modelo *Híbrido*.

Discusión

Al revisar los resultados obtenidos en todas las pruebas, se puede notar que cada modelo tuvo un comportamiento distinto, y esto ayudó a entender mejor qué tan difícil es clasificar noticias falsas solo a partir del texto. Lo primero que se observa es que los modelos tradicionales, como *Naive Bayes*, *Decision Tree* y *Random Forest*, funcionaron bastante bien cuando las noticias eran reales. Esto tiene sentido porque estos modelos suelen responder mejor cuando el texto sigue estructuras claras o patrones más marcados. Sin embargo, también se vio que estos mismos modelos se quedaron cortos al momento de detectar las noticias falsas. El pequeño desbalance del *dataset* terminó influyendo más de lo esperado, y por eso la mayoría priorizó las noticias reales.

En contraste, los modelos de *deep learning*, especialmente *CNN* y el modelo *Híbrido*, sí lograron captar detalles más finos del lenguaje. Esto se notó bastante cuando se revisaron las métricas de detección de noticias falsas. El modelo *Híbrido* fue el que más sobresalió en esa parte, mientras que *CNN* fue el que mantuvo un rendimiento más estable en la mayoría de las métricas. Cuando se revisan en conjunto todas las tablas del proyecto, se repite la misma tendencia: *CNN* no siempre gana en todo, pero sí mantiene un equilibrio muy difícil de igualar.

Con respecto a los modelos *Transformers*, aunque en algunos valores superaron a *CNN*, su tiempo de entrenamiento fue demasiado alto. Para este proyecto, que tenía un tiempo limitado, esto fue un problema grande. Fue evidente que *mBERT* y *BETO* tienen potencial, pero también quedó claro que requieren hardware mucho más potente y tiempo adicional para ajustarlos. Por esta razón, aunque en ciertos números parecían una buena opción, en la práctica no resultó conveniente usarlos.

Otro punto importante es la relación de estos resultados con los objetivos. El objetivo general, que consistía en construir un sistema capaz de detectar noticias falsas en español, sí se cumplió. También se lograron los objetivos específicos, porque se entrenaron los modelos, se compararon, se creó la *API* y se construyó la interfaz web. Sin embargo, el comportamiento de los modelos durante las pruebas dejó varias cosas claras: los modelos no detectan si una noticia es cierta o no según los hechos, sino según la forma en la que está escrita. Esto explica por qué algunas noticias reales, pero redactadas de forma alarmante, fueron clasificadas como falsas, y también por qué algunas falsas pasaron como reales si imitaban el estilo de un artículo legítimo.

Durante la validación final, usando noticias cortas, largas, ambiguas, en otros idiomas y con errores ortográficos, se vio que *CNN* y el modelo *Híbrido* se adaptaban mejor. En cambio, *Naive Bayes* fue el que peor funcionó, incluso en casos simples. También se notó que, aunque el modelo Híbrido detectó noticias falsas en varios casos difíciles, lo hacía con baja confianza, mientras que *CNN* fue más estable. Estos comportamientos coinciden con lo que se esperaba al ver sus métricas.

En lo relacionado con el impacto y lo ético, es importante aclarar que este tipo de sistema no debe reemplazar el criterio humano. Los modelos pueden ayudar, pero no pueden verificar hechos ni interpretar ironía, intenciones o contexto. Si en algún momento se usa un sistema como este en medios reales, tendría que ser supervisado, y no depender solo de la "probabilidad" de veracidad que arroja el modelo. Las noticias falsas afectan a muchas personas, pero también sería peligroso marcar como falsa una noticia que sí es real solo porque está escrita de una manera poco común.

Por último, algo que queda muy evidente es que no existe un modelo perfecto. Cada uno resolvió mejor una parte del problema. Por eso, en lugar de insistir en encontrar un único

modelo, se optó por mantener varios y asignarles un propósito distinto según sus fortalezas. Esta decisión surgió directamente del análisis y de la práctica, no solo de las métricas. En general, el proyecto permitió entender mejor las ventajas y desventajas de cada enfoque y, al final, se pudo construir un sistema funcional que cumple con lo planteado y que también deja espacio para mejoras futuras.

Conclusiones

Conclusiones Específicas Según los Objetivos Planteados

Con relación al objetivo específico de recolectar y preprocesar conjuntos de datos públicos en español, durante el desarrollo del proyecto se usaron varios conjuntos de datos disponibles públicamente. Fue necesario analizar cada uno de estos conjuntos y preparar los datos antes de que los modelos los usaran. Este proceso incluyó limpiar el texto, eliminar información irrelevante y transformar los datos a un formato adecuado para su procesamiento.

Gracias a este trabajo, se obtuvo un conjunto de datos consistente, lo que permitió entrenar y evaluar los modelos de detección de noticias falsas de manera correcta.

Respecto al objetivo específico de diseñar el modelo de detección de *fake news* y definir su arquitectura, se diseñaron y evaluaron diferentes enfoques, incluyendo modelos de *machine learning*, *deep learning* y *Transformers*, para identificar el método que mejor se ajustara al problema. Se usaron métricas como *accuracy*, *precision*, *recall* y *F1-score* para comparar los resultados.

Los resultados mostraron que no había un único modelo perfecto, ya que cada enfoque tuvo sus propias fortalezas y debilidades. Por esta razón, se decidió trabajar con cuatro modelos finales. Entre ellos, el modelo *CNN* tuvo el mejor desempeño general, alcanzando aproximadamente un **86%** de *accuracy*, por lo que se consideró el más adecuado para el proyecto.

En cuanto al objetivo específico de implementar la *API* para servir las predicciones del modelo, se desarrolló una *API* usando *FastAPI*, lo que permitió hacer consultas al modelo y obtener las predicciones automáticamente. Durante las pruebas, la *API* funcionó correctamente y

de manera estable, lo que demostró que los modelos entrenados pueden integrarse sin problemas en una aplicación real y responder adecuadamente a las solicitudes desde la interfaz web.

Finalmente, en relación con el objetivo específico de construir la interfaz web, se desarrolló una interfaz en *React* que permite al usuario ingresar el texto de una noticia y recibir una predicción de forma clara y sencilla. Esta interfaz facilitó la interacción con el sistema y mostró que el proyecto no se limita solo a un desarrollo técnico, sino que puede ser usado por personas sin conocimientos avanzados en programación.

Conclusiones Generales

El desarrollo de este proyecto permitió demostrar que era posible realizar un modelo de detección de noticias falsas en español, incluyendo una *API* y una *interfaz web*, evitando dejarlo como un proyecto técnico al que solo pueden acceder personas con conocimientos avanzados en código y permitiendo su utilización por personas con menos experiencia. Durante el desarrollo se analizaron y evaluaron múltiples modelos de *machine learning*, *deep learning* y *Transformers* para identificar cuál era el mejor. Al final, se determinó que cada uno tenía fortalezas y debilidades, por lo que se descartó continuar buscando un único modelo perfecto. En su lugar, se optó por mejorar las fortalezas de cuatro modelos seleccionados, dando más opciones a los usuarios, lo cual se identificó como la mejor solución para abordar este problema.

Los modelos de *machine learning* ofrecieron rapidez y buen desempeño en textos estructurados, además de no presentar problemas de *underfitting* ni *overfitting* debido a la simpleza de su arquitectura. Los modelos de *deep learning* mejoraron la detección de noticias falsas al identificar patrones lingüísticos más complejos, aunque su entrenamiento fue más demorado y mostraron más problemas de *overfitting* que los modelos de *machine learning*.

Los modelos de *Transformers* no estuvieron en su mejor forma debido a incompatibilidades de *PyTorch* con otras dependencias. Apenas pudieron demostrar rendimientos similares a los otros dos tipos y su tiempo de entrenamiento fue mucho mayor, requiriendo *hardware* más potente. Por esta razón tuvieron que ser descartados, ya que podían poner en riesgo la fecha de finalización del proyecto.

El modelo de *CNN* demostró ser el más adecuado para este problema. Se adapta correctamente a estructuras lingüísticas más complejas que los modelos de *machine learning* y, aunque su tiempo de entrenamiento aumenta y puede presentar *overfitting*, este es mucho menor que en los modelos *híbridos* o en los *Transformers*, lo que lo convierte en una opción atractiva.

A pesar de haber mencionado en las limitaciones que los modelos no detectan si una noticia es verdadera o falsa como tal, sino si la estructura en la que está escrita parece falsa o no, considero que cumple con lo establecido para el proyecto, ya que esto es precisamente a lo que se refiere el *PLN*: a partir del análisis del texto, los modelos aprendieron patrones y pudieron identificar estructuras típicas de noticias falsas o verdaderas.

Es importante señalar que, aunque uno de los indicadores definidos inicialmente para el primer resultado esperado establecía alcanzar valores superiores al **90%** tanto en *precisión* como en *recall*, los resultados obtenidos evidencian un cumplimiento parcial de dicho indicador. En particular, se logró superar ampliamente el umbral establecido en la métrica de *recall*, alcanzando valores superiores al **97%**, mientras que la precisión se situó alrededor del **86%** en el mejor modelo evaluado. Esta diferencia refleja una situación común en problemas reales de clasificación de texto, donde existe un compromiso natural entre ambas métricas y donde priorizar la detección efectiva de noticias falsas puede implicar sacrificar ligeramente la *precisión*, efecto que se ve acentuado por el desequilibrio presente en el *dataset* utilizado.

La construcción de la *API* con *FastAPI* y de la *interfaz web* con *React* validó que el sistema puede integrarse con éxito en aplicaciones reales. En general, el proyecto cumplió con los objetivos planteados dentro del tiempo establecido, desarrolló un sistema funcional, evaluado y desplegable, y permitió al futuro ingeniero de sistemas adquirir experiencia práctica en habilidades avanzadas y muy demandadas actualmente en el mercado laboral, como el *procesamiento del lenguaje natural*, análisis y preprocesamiento de datos, *machine learning*, *deep learning*, *Transformers*, y herramientas de software como el desarrollo de *APIs* e *interfaces web*. Y aunque podría haberse mejorado reentrenando los modelos o buscando mejores parámetros con *hardware* más potente, el proyecto cumplió con lo indicado de la mejor forma posible.

Recomendaciones

A partir de los resultados, las pruebas prácticas y las limitaciones encontradas durante el desarrollo, se pueden plantear varias recomendaciones para mejorar el sistema y para trabajos futuros. La primera, y quizá la más importante, es ampliar y actualizar el *dataset*. Aunque se usó un conjunto grande de noticias, la proporción entre noticias reales y falsas terminó influyendo en el aprendizaje de los modelos, y en las pruebas finales quedó claro que todos priorizan las noticias reales. Por eso, sería conveniente incluir más fuentes, temas distintos y noticias falsas más variadas, o incluso generar un módulo de *web scraping* que permita crear un *dataset* propio y actualizado con regularidad.

También sería recomendable probar nuevas técnicas de preprocesamiento. En este proyecto se vio que pequeñas decisiones, como usar o no *lematización*, cambiaban el rendimiento de algunos modelos, especialmente *CNN* e *Híbrido*. En un trabajo futuro podría evaluarse un preprocesamiento más especializado para cada modelo, en lugar de usar un *pipeline* general para todos.

Otra recomendación es integrar algún tipo de verificación externa. Por ahora, el sistema analiza únicamente el texto, lo cual funciona bien para clasificar patrones de escritura, pero no para verificar hechos. Esto significa que una noticia falsa redactada como si fuera real puede pasar sin problema. Una forma de mejorar esto sería agregar una capa adicional que consulte bases de datos externas, buscadores o fuentes oficiales, aunque esto ya implicaría un sistema mixto entre *PLN* y verificación factual.

En el caso de los modelos *Transformers*, aunque *mBERT* demostró ser muy bueno en varias métricas, su tiempo de entrenamiento y el peso del modelo hacen que no sea práctico

incluirlo en producción en el estado actual. Si se cuenta con mejores recursos en el futuro, podría reentrenarse con más épocas, un mejor entorno o técnicas de *fine-tuning* más ligeras (como *LoRA* o modelos cuantizados). Esto permitiría aprovechar sus fortalezas sin comprometer el tiempo de desarrollo.

Además, sería útil implementar un sistema que combine varias predicciones, especialmente porque *CNN* e *Híbrido* funcionaron muy bien en pruebas reales, pero cada uno destacó en casos distintos. En lugar de usar un único modelo, se podría crear un sistema de votación o de ponderación para obtener una predicción final más estable. Esto también ayudaría en escenarios donde un modelo duda o entrega una confianza muy baja.

Otra recomendación es documentar aún más los casos especiales. Durante las pruebas aparecieron comportamientos interesantes, por ejemplo, la sensibilidad de los modelos clásicos cuando el texto está en otro idioma. Registrar más ejemplos de este tipo ayudaría a afinar mejor las reglas del sistema o a decidir cuándo recomendarle al usuario volver a escribir la noticia o completarla.

También sería conveniente pensar en una arquitectura más modular para el despliegue final. La *API* y la interfaz funcionaron bien en local, pero si se quiere llevar a producción real, habría que considerar contenedores, orquestación, tolerancia a fallos y un sistema de logs más avanzado. Esto también facilitaría el reentrenamiento continuo para evitar que los modelos "envejezcan" rápidamente.

Por último, se recomienda mantener el enfoque ético. Este tipo de sistemas pueden ser útiles, pero deben verse como herramientas de apoyo, no como autoridad absoluta en la

detección de noticias falsas. Sería conveniente incluir advertencias en la interfaz, explicar los límites del sistema y aclarar que una predicción no reemplaza la verificación humana.

Impacto, Implicaciones Éticas y Consideraciones Sociales del Sistema

El sistema desarrollado en este proyecto puede ser útil para que las personas tengan una herramienta de apoyo cuando quieran revisar una noticia, sobre todo quienes no están familiarizados con cómo funciona la desinformación. Aunque no es un verificador de hechos, sí puede orientar al usuario cuando un texto parece sospechoso. Esto, en cierto modo, puede ayudar a que la gente tome decisiones un poco más informadas frente a lo que lee en redes o en medios digitales.

Aun así, este tipo de sistemas también tiene riesgos y es importante mencionarlos. El modelo no "sabe" si algo es verdad o mentira; solo reconoce ciertas formas de escribir que suelen aparecer en noticias falsas. Esto hace que exista la posibilidad de que una noticia real termine marcada como falsa, o al revés, sobre todo si está escrita de una forma muy neutral. Por eso, el uso del sistema siempre debería ir acompañado del criterio del usuario y no reemplazar la verificación real.

También hay un tema de sesgos. Aunque el *dataset* usado era amplio, sigue siendo una parte del entorno real, y eso hace que algunos estilos o temas puedan quedar subrepresentados. Esto puede influir en la manera en la que el sistema responde. Además, el lenguaje cambia con el tiempo, las noticias cambian sus formatos, y la desinformación también evoluciona, así que lo que hoy funciona bien, mañana podría no ser tan preciso si no se reentrena el modelo periódicamente.

Otro punto que se debe tener en cuenta es que un sistema de este tipo podría malinterpretarse como una herramienta de censura. Ese no es el propósito del proyecto. Aquí lo que se busca es dar un apoyo inicial, no tomar decisiones por encima del usuario. Por eso,

cualquier uso en un contexto real debería dejar claro que el sistema tiene limitaciones y que no se debe confiar al 100% en una sola predicción.

A nivel social, si se usa adecuadamente, este tipo de herramientas puede aportar a mejorar la alfabetización digital, porque obliga al usuario a cuestionar lo que lee. Sin embargo, también requiere responsabilidad por parte de quien lo implemente. Ser transparente sobre lo que el sistema puede y no puede hacer es fundamental para evitar malentendidos o usos indebidos.

Limitaciones del Estudio

Durante el desarrollo del proyecto aparecieron varias limitaciones que es importante dejar claras, ya que influyen tanto en el alcance del sistema como en la interpretación de los resultados. La primera tiene que ver con el tipo de información que el sistema es capaz de analizar. Este proyecto se enfocó únicamente en texto escrito en español, por lo que no se manejan imágenes, videos, audio ni ningún otro formato que hoy en día también se usa para difundir noticias falsas. Tampoco se procesan textos en otros idiomas, así que cualquier noticia fuera del español queda por fuera del sistema.

Otra limitación es que los modelos no determinan si un hecho es verdadero o no. Lo que realmente hacen es detectar patrones de escritura, vocabulario y estructuras que suelen aparecer en noticias falsas. Esto implica que, si una noticia real se redacta con un estilo sensacionalista, puede clasificarse como falsa, y una noticia falsa bien redactada puede clasificarse como real. Para que un sistema identifique hechos como tal, tendría que consultar bases de datos externas, hacer búsquedas en línea o integrar módulos de verificación factual, lo cual queda por fuera del alcance del proyecto.

También está el problema del *dataset*. Aunque era grande y variado, tenía una pequeña diferencia entre noticias reales y falsas (59% frente a 41%), y esto terminó afectando el entrenamiento: todos los modelos favorecieron las noticias reales, especialmente los modelos clásicos. Además, aunque el *dataset* incluía noticias de diferentes medios, sigue existiendo el riesgo de que en producción aparezcan textos muy distintos a los del entrenamiento. Esto puede reducir la capacidad del sistema para generalizar, sobre todo en temas recientes o redactados con estilos poco comunes.

En cuanto a los modelos, existe una limitación importante relacionada con los recursos técnicos. Los modelos de *Transformers* como *mBERT* y *BETO* son muy pesados y requieren mucho tiempo de entrenamiento e inferencia. En este proyecto se trabajó con un entorno limitado, lo que obligó a reducir configuraciones y evitar experimentos más complejos. De hecho, *BETO* se vio afectado directamente por incompatibilidades en el entorno, lo cual impactó su rendimiento final. Esta limitación también afectó la posibilidad de realizar entrenamientos repetidos o ajustes avanzados.

Otro punto es que los modelos no interpretan aspectos como ironía, sarcasmo, referencias culturales o dobles sentidos. Esto es común en las noticias y en la comunicación digital actual. Al analizar únicamente el texto plano, el sistema pierde información clave que podría cambiar completamente el significado de una frase.

También hay limitaciones relacionadas con el tiempo del proyecto. Toda la investigación, desarrollo, implementación y pruebas se realizaron en un periodo de cuatro meses dentro de un entorno académico. Esto impidió realizar pruebas más profundas en producción, integrar módulos adicionales o refinar la interfaz. Por lo tanto, el sistema funciona correctamente en local, pero aún no se ha probado en un entorno real con usuarios finales.

Otra limitación es que la interfaz en *React* es sencilla y, aunque cumple con su objetivo, no es una interfaz profesional. Tampoco se consideró un despliegue permanente porque la plataforma gratuita (*Render*) solo permite mantener el servicio activo por tiempos limitados. Esto restringe la posibilidad de validar el sistema en escenarios completamente reales.

Por último, los modelos pueden “envejecer” rápido. El estilo de las noticias cambia con el tiempo, los medios modifican sus formas de titular y las estrategias de desinformación también

evolucionan. Si los modelos no se reentrenan periódicamente, su efectividad puede disminuir. Este proyecto se enfocó en un entrenamiento único y completo, pero no incluye un sistema automático de actualización o mejora continua.

Referencias

Abdelhamid, M., & Desai, A. (2024). *Balancing the scales: A comprehensive study on tackling class imbalance in binary classification*. arXiv. https://arxiv.org/html/2409.19751v1

Acosta-Sánchez, C. A. (2019). *Cuestiones prácticas para el planteamiento del problema en un proyecto de investigación*. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/31824

Alcorpas. (s. f.). *FakesStorage: Spanish dataset for fake news detection*. GitHub. https://github.com/alcorpas10/FakesStorage

Bucheli, H. A. (2019). *CDIO (Concebir, Diseñar, Implementar y Operar): Iniciativa para la resolución de problemas en ingeniería*. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/33800

Cadenas, J. F. (2025, noviembre 24). *La limitación de la UE a la pesca a 27 días al año es solo para la pesca de arrastre que no cumpla medidas de sostenibilidad*. Newtral. https://www.newtral.es/limitacion-pesca-arrastre-ue/20251124/

Capdevila, G. I. (2025, noviembre 24). *Emisarios de un oligarca próximo a Putin agitan a la ultraderecha en Madrid: "Estamos ganando la batalla de la propaganda"*. Newtral. https://www.newtral.es/emisarios-putin-asisten-cumbre-ultraderecha-falange-madrid/20251124/

Classification: Accuracy, recall, precision, and related metrics. (s. f.). *Google for Developers*. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Clavijo, L. J. R. (2025, noviembre 24). *Canacol Energy se acoge ahora a ley de quiebras en EE. UU., días después de declararse en insolvencia en Canadá*. Portafolio. https://www.portafolio.co/energia/canacol-energy-se-acogio-a-la-ley-de-quiebras-de-estados-unidos-ante-dificil-situacion-financiera-483593

UNICEF. (2022, 11 de mayo). *¿Cómo detectar «fake news» en Colombia?* UNICEF. https://www.unicef.org/colombia/casicaigo

Córdoba Cano, D. (2025). *La desinformación en línea y su influencia en la opinión pública*. Repositorio Institucional UNAD. https://repository.unad.edu.co/bitstream/handle/10596/67127/dcordobacan.pdf

Datenschutz-Grundverordnung. (2024, noviembre 15). *Finaler Text der DSGVO*. https://dsgvo-gesetz.de/

Decreto 1377 de 2013. (s. f.). *Gestor normativo*. Función Pública. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=53646

Find open datasets and machine learning projects. (s. f.). *Kaggle*. https://www.kaggle.com/datasets

Gálvez, A., Albores, F., Álvarez-González, R., Conde, S., & Gálvez, S. (2024). *A study on the detection of fake news in Spanish*. International Journal of Combinatorial Optimization Problems and Informatics, 15, 85–94. https://doi.org/10.61467/2007.1558.2024.v15i2.467

IBM. (2025, junio 27). *Aprendizaje automático*. https://www.ibm.com/mx-es/think/topics/machine-learning

Jose Camacho-Collados, & Mohammad Taher Pilehvar. (2018). *On the role of text preprocessing in neural network architectures*. Association for Computational Linguistics. https://aclanthology.org/W18-5406/

Judicial, R. (2025a, noviembre 24). *Fuerzas Militares investigan la probable filtración de información a disidencias de alias "Calarcá"*. El Espectador. https://www.elespectador.com/judicial/fuerzas-militares-investigan-la-probable-filtracion-de-informacion-a-disidencias-de-alias-calarca/

Judicial, R. (2025b, noviembre 24). *¿Quién es el general Huertas? El oficial reincorporado por Petro con presuntos nexos con disidencias*. El Espectador. https://www.elespectador.com/judicial/quien-es-el-general-juan-miguel-huertas-el-oficial-reincorporado-por-petro-con-presuntos-nexos-con-disidencias/

Koshiw, I. (2025, noviembre 24). *Trump presses on Ukraine as Zelenskyy and Europe push back*. AP News. https://apnews.com/article/ukraine-russia-war-zelenskyy-peace-plan-europe-65b83ecc28b58bfbcf48b6f9e7dbd564

Ley 527 de 1999. (s. f.). *Gestor normativo*. Función Pública. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=4276

Ley 1266 de 2008. (s. f.). *Gestor normativo*. Función Pública. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=34488

Ley 1341 de 2009. (s. f.). *Gestor normativo*. Función Pública. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=36913

Ley 1581 de 2012. (s. f.). *Gestor normativo*. Función Pública. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

Ley 1712 de 2014. (s. f.). *Gestor normativo*. Función Pública. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=56882

Lledó, M. (2025, noviembre 24). *Este vídeo de un aterrizaje forzoso de un helicóptero estadounidense está hecho con IA aunque Grok lo de por cierto*. Newtral. https://www.newtral.es/helicoptero-eeuu-aterrizaje/20251124/

Maaddi, R. (2025, noviembre 24). *There are more questions than answers after NFL games Sunday*. AP News. https://apnews.com/article/mahomes-stafford-mayfield-hurts-brown-4ba973324854d0f3311d166b555c1763

Machine learning: Cross validation. (s. f.). *W3Schools*. https://www.w3schools.com/python/python_ml_cross_validation.asp

Machine learning: Grid search. (s. f.). *W3Schools*. https://www.w3schools.com/python/python_ml_grid_search.asp

mariagrandury. (2024, septiembre 15). *Fake news corpus Spanish*. Hugging Face. https://huggingface.co/datasets/mariagrandury/fake_news_corpus_spanish

Marketing Investigación. (s. f.). *Fake news: La amenaza silenciosa que se propaga más rápido que la verdad*. Noticias UNAD. https://noticias.unad.edu.co/index.php/investigacion-noticias/7411-fake-news-la-amenaza-silenciosa-que-se-propaga-mas-rapido-que-la-verdad

MINTIC Colombia. (s. f.). *Resoluciones*. https://www.mintic.gov.co/portal/inicio/Normatividad/Resoluciones/

M., N., S., A., Vn., S., & S., S. (2025). *Fake news detection using deep learning*. https://doi.org/10.2139/ssrn.5089165

More, P., Jadhav, A., & Yadav, S. (2021). *Detection of fake news using machine learning*. International Journal of Advanced Research in Computer and Communication Engineering. https://doi.org/10.17148/ijarcce.2021.10484

MultinomialNB. (s. f.). *Scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

Nishitpatel. (s. f.). *Fake news detection in Python*. GitHub. https://github.com/nishitpatel01/Fake_News_Detection

¿Qué es una API? (s. f.). *Amazon Web Services*. https://aws.amazon.com/es/what-is/api/

¿Qué son los transformadores en inteligencia artificial? (s. f.). *Amazon Web Services*. https://aws.amazon.com/es/what-is/transformers-in-artificial-intelligence/

Recomendación sobre la ética de la inteligencia artificial. (2021). *UNESCO*. https://unesdoc.unesco.org/ark:/48223/pf0000381137_spa

Recomendación sobre la inteligencia artificial. (s. f.). *OECD Legal Instruments*. https://legalinstruments.oecd.org/

Redacción. (2024, marzo 11). *Princesa Kate: La princesa de Gales pide disculpas por la confusión que causó su foto retirada por las agencias de noticias por inconsistencias*. BBC News Mundo. https://www.bbc.com/mundo/articles/c88xjp0zlwqo

sayalaruano. (s. f.). *FakeNewsSpanish Kaggle2*. Hugging Face. https://huggingface.co/datasets/sayalaruano/FakeNewsSpanish_Kaggle2

Spanish fake and real news. (2018, diciembre 28). *Kaggle*. https://www.kaggle.com/datasets/zulanac/fake-and-real-news

Spanish political fake news. (2023, diciembre 18). *Kaggle*. https://www.kaggle.com/datasets/javieroterovizoso/spanish-political-fake-news

Spring, M. (2025, junio 30). *Nuestra hermana murió por las teorías conspirativas de nuestra madre sobre el cáncer*. BBC News Mundo. https://www.bbc.com/mundo/articles/cjrl80j4x7no

Stryker, C., & Holdsworth, J. (2025, abril 23). *Procesamiento de lenguaje natural*. IBM. https://www.ibm.com/mx-es/think/topics/natural-language-processing

z/OS basic skills. (s. f.). *IBM Documentation*. https://www.ibm.com/docs/en/zos-basic-skills?topic=more-what-is-data-set

Anexos

Anexo A – Repositorio del Proyecto

Código fuente de la API en FastAPI, modelo de IA entrenado, scripts de preprocesamiento y aplicación web en React.

Enlace: https://github.com/LLFELIPEVV/sistema-fake-news-ia

Anexo B – Manual de Usuario del Sistema

Documento que describe el uso del sistema de detección de noticias falsas, incluyendo instrucciones para ejecutar la API, iniciar el frontend, realizar análisis de texto y resolver problemas comunes.

Enlace: https://drive.google.com/file/d/1iQR-tgbjnR9NFIrfBc7SWZToK8IMa1fN/view?usp=sharing