Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Computación

Optimización de la Precisión en la Detección de Noticias Falsas de política en español mediante la Aplicación de Algoritmos de Optimización en la Regresión Logística

Optimization of the Accuracy in the Detection of Fake News Politics in Spanish through Application of Optimization Algorithms in LogisticRegression

Trabajo de Integración Curricular previa a la obtención del título de Ingeniero en Ciencias de la Computación

AUTOR:

Santiago Emanuel Tene Castillo

DIRECTOR:

Ing. Luis Antonio Chamba Eras, PhD

Loja – Ecuador

2025

Certificación

Ing. Luis Antonio Chamba Eras, PhD
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominado: **Optimización de la Precisión en la Detección de Noticias Falsas de política en español mediante la Aplicación de Algoritmos de Optimización en la Regresión Logística**, previo a la obtención del título de **Ingeniero en Ciencias de la Computación**, de la autoría del estudiante **Santiago Emanuel Tene Castillo**, con cédula de identidad **1105174989**, una vez que se determina que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

LUIS ANTONIO
CHAMBA ERAS

……..

Ing. Luis Antonio Chamba Eras, PhD
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Autoría

Yo, **Santiago Emanuel Tene Castillo**, declaro ser autor del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular, en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma:

Cédula de identidad: 1105174989

Fecha: 17 de marzo de 2025

Correo electrónico: santiago.tene@unl.edu.ec

Teléfono: 0996543184

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular

Yo, **Santiago Emanuel Tene Castillo**, declaro ser el autor del Trabajo de Integración Curricular denominado: **Optimización de la Precisión en la Detección de Noticias Falsas de política en español mediante la Aplicación de Algoritmos de Optimización en la Regresión Logística**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los diecisiete días del mes de marzo de dos mil veinticinco.

Firma:

Autor: Santiago Emanuel Tene Castillo
Cédula de identidad: 1105174989
Dirección: Loja, Aristóteles y Aurelio Valdiviezo
Correo electrónico: santiago.tene@unl.edu.ec
Teléfono: 0996543184

DATOS COMPLEMENTARIOS:

Director del Trabajo de Integración Curricular: Ing. Luis Antonio Chamba Eras, PhD

Dedicatoria

Con especial cariño, dedico este logro a mi familia, el pilar fundamental de mi camino. A mi esposa e hijo, quienes fueron mi mayor fuente de inspiración y motivación para seguir adelante y culminar este trabajo de integración curricular. A mis queridos padres, Juan y Angélica, por su apoyo incondicional y por estar siempre a mi lado en cada desafío. A mis hermanos, Juan y Catherine, ejemplos de esfuerzo y superación constante. A mis profesores, quienes compartieron conmigo su conocimiento y guiaron mi aprendizaje a lo largo de mi carrera. Y, por supuesto, a mis compañeros y amigos, con quienes compartí esta etapa universitaria. A todos ustedes, mi sincero agradecimiento.

Santiago Emanuel Tene Castillo

Agradecimiento

Agradezco a Dios, quien guía siempre nuestros pasos y fortalece nuestro camino. A los docentes que han sido parte de mi formación, y en especial a mi director, Ing. Luis Chamba Eras, por su invaluable apoyo, su sabiduría compartida, sus consejos y sus enriquecedoras charlas. Al Ing. Roy León, por su ayuda en los momentos de incertidumbre, cuando mis ideas aún no estaban claras. A mi familia, por su apoyo incondicional y su confianza en mí en cada paso de este recorrido. A la Universidad Nacional de Loja, por brindarme la oportunidad de formarme académicamente, y a mi querida carrera de Ingeniería en Computación, por todo el conocimiento y las experiencias que han marcado mi crecimiento profesional y personal.

Santiago Emanuel Tene Castillo

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Índice de Anexos

1. Título

Optimización de la Precisión en la Detección de Noticias Falsas de política en español mediante la Aplicación de Algoritmos de Optimización en la Regresión Logística.

Optimization of the Accuracy in the Detection of Fake News Politics in Spanish through Application of Optimization Algorithms in LogisticRegression.

2. Resumen

La regresión logística, ampliamente utilizada en clasificación de textos para detectar noticias falsas, presenta limitaciones en su optimización dentro de este campo específico. La escasa exploración sistemática de algoritmos de optimización Gradiente Descendente (GD), Gradiente Descendente Estocástico (SGD), Gradiente Descendente por Mini-Lotes (MBGD), AdaGrad, Adam y RMSProp dificulta determinar su impacto real en la mejora de métricas de clasificación. Este trabajo de integración curricular abordó dicho problema aplicando estos seis algoritmos a un modelo de regresión logística para la detección de noticias políticas falsas en español, bajo la metodología CRISP-ML. El proceso incluyó: 1) Ingeniería de datos para generar un conjunto personalizado, 2) Optimización mediante ajuste de hiperparámetros de los algoritmos, y 3) Evaluación con matriz de confusión y métricas (Sensibilidad, Especificidad, Precisión, Exactitud y F1-Score). Los resultados demostraron que la variante SGD-LR (Gradiente Descendente Estocástico) superó significativamente al modelo base no optimizado (73.7% vs. 80.3% en precisión), así como a las demás técnicas evaluadas. Este incremento del 6.6% evidencia que la selección estratégica de algoritmos de optimización impacta directamente en el rendimiento de modelos de clasificación. El estudio no solo valida la eficacia del SGD para esta tarea específica, sino que establece un precedente metodológico al integrar CRISP-ML en el proceso de optimización de modelos. Estos hallazgos resaltan la necesidad de incluir fases sistemáticas de experimentación con optimizadores como paso crítico en el desarrollo de sistemas de detección de desinformación, particularmente para contenidos en español donde los estudios técnicos siguen siendo escasos.

Palabras clave: Machine learning, Clasificación de texto, Ajustar hiperparámetros, CRISP-ML, Algoritmos de clasificación, Métodos de mejora.

Abstract

Logistic regression, while widely employed in text classification for fake news detection, shows suboptimal optimization practices in this specific domain. The limited systematic exploration of optimization algorithms—Gradient Descent (GD), Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent (MBGD), AdaGrad, Adam, and RMSProp—hinders the accurate assessment of their impact on classification metrics. This Curricular Integration Project (CIP) addressed this gap by applying these six algorithms to a logistic regression model for detecting Spanish-language political fake news, following the CRISP-ML methodology. The workflow included: 1) Data engineering to create a custom dataset, 2) Model optimization through hyperparameter tuning of the algorithms, and 3) Evaluation using confusion matrices and performance metrics (Sensitivity, Specificity, Precision, Accuracy, and F1-Score). Results revealed that the SGD-LR variant (Stochastic Gradient Descent) outperformed both the baseline non-optimized logistic regression model (73.7% vs. 80.3% precision) and other evaluated optimizers. This 6.6% improvement highlights how strategic algorithm selection directly enhances classification performance. The study not only validates SGD's efficacy for this task but also sets a methodological precedent by integrating CRISP-ML into optimization workflows. These findings underscore the necessity of systematic experimentation with optimizers as a critical phase in developing misinformation detection systems, particularly for Spanish-language content where technical studies remain scarce.

Keywords: Machine learning, Text classification, Hyperparameter tuning, CRISP-ML, Classification algorithms, Improvement methods.

3. Introducción

La regresión logística cuenta con un alto potencial para detectar noticias políticas falsas en español. Sin embargo, no se optimiza lo suficiente en este campo. La falta de experimentación con algoritmos de optimización como Gradiente Descendente (GD), Gradiente Descendente Estocástico (SGD), Gradiente Descendente por Mini-Lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam) y Propagación de la Raíz del Promedio Cuadrático (RMSProp) impide analizar su efecto en la precisión del clasificador. La falta de experimentación con el modelo de regresión logística genera un problema de optimización que limita su rendimiento. Este hecho se evidencia en las preguntas del Anexo 2, donde se aplica una variante del método Delphi a un investigador en el área de inteligencia artificial y se refuerza la problemática de optimización para detectar noticias falsas de política en español. En la pregunta 3 de esa investigación, se profundiza en el modelo de regresión logística y se concluye que sus características y metodología de optimización restringen su precisión. La escasa experimentación con algoritmos de optimización como AdaGrad, Adam y RMSProp reduce la posibilidad de mejora, de modo que se considera fundamental explorar estos métodos para impulsar los resultados.

Las técnicas de optimización resultan esenciales para elevar la precisión de los modelos de clasificación en machine learning. Sin una optimización adecuada, estos modelos enfrentan problemas de convergencia y sobreajuste, lo que debilita la confianza en sus resultados. Se recomienda utilizar algoritmos de optimización basados en gradiente como gradiente descendente estocástico (SGD), tanto por su eficiencia en el procesamiento de grandes volúmenes de datos como por su capacidad de escapar de mínimos locales, lo que favorece el incremento de la precisión en tareas de clasificación [1][2].

Al aplicar algoritmos de optimización (GD, SGD, MBGD, AdaGrad, Adam y RMSProp) en la regresión logística para clasificar noticias falsas de política en español, surge la pregunta: "¿Qué porcentaje de precisión se puede alcanzar en la identificación de noticias falsas de política en español mediante el modelo de clasificación Regresión Logística al aplicar los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp?".

Para responder a esta interrogante, el objetivo principal consiste en aplicar los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp en el modelo regresión logística para la detección de noticias falsas de política en español, Asimismo, se definieron dos objetivos específicos: (1) entrenar el modelo regresión logística con los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp mediante CRISP-ML y (2) evaluar la mejora en la precisión del modelo en comparación con la precisión inicial sin optimización utilizando la matriz de confusión.

El cumplimiento de estos objetivos deriva en la generación de un dataset unificado que integra dos colecciones de noticias políticas en español más datos adicionales de noticias de política de Ecuador. La contribución principal del Trabajo de Integración Curricular consiste en crear un modelo de regresión logística optimizado. Entre las limitaciones, destacan la limitación de recursos computacionales por falta de un equipo de altas prestaciones y la dificultad para obtener datos confiables en español, factores que impactan directamente en el proceso de entrenamiento y evaluación del modelo.

Este trabajo aborda su desarrollo a través del marco teórico, metodología, resultados, discusión y conclusiones. En el marco teórico se establecen los fundamentos conceptuales y metodológicos, incluyendo antecedentes, teorías y algoritmos de optimización como GD, SGD, MBGD, AdaGrad, Adam y RMSProp. La metodología, basada en el enfoque CRISP-ML, describe las etapas de ingeniería de datos, ingeniera de modelos y evaluación, detallando los recursos técnicos y computacionales empleados. Los resultados evidencian que el Gradiente Descendente Estocástico (SGD) alcanzó la mayor precisión, superando al modelo base sin optimización. En la discusión se analizan las mejoras y limitaciones del estudio, mientras que las conclusiones resaltan la efectividad de los algoritmos de optimización y la importancia de contar con un dataset diverso. Finalmente, las recomendaciones sugieren explorar modelos más avanzados, aplicar técnicas de aumento de datos y evaluar su implementación en entornos reales para futuras investigaciones.

4. Marco teórico

El marco teórico proporcionó el sustento teórico, conceptual y metodológico para fundamentar el trabajo de integración curricular, estructurándose en cinco componentes articulados. Su objetivo principal consistió en integrar antecedentes, teorías, conceptos técnicos y enfoques metodológicos que respaldaron la investigación, mediante: 1) una contextualización crítica de los antecedentes, donde se identificó la escasa experimentación con regresión logística y sus técnicas de optimización basadas en gradiente descendente (GD), limitación que obstaculizaba la evaluación rigurosa del impacto de estos algoritmos en la precisión del modelo; 2) la fundamentación teórica, centrada en los principios estadísticos del machine learning y la formulación matemática de la regresión logística; 3) la base conceptual, que describió las variantes de optimización aplicadas (GD, SGD, MBGD, AdaGrad, Adam, RMSProp) y su rol en entrenamiento de modelos; 4) el sustento metodológico, que definió herramientas tecnológicas (librerías de programación, entornos de desarrollo) y marcos de trabajo (CRISP-ML) para la implementación práctica; y 5) la revisión sistemática de trabajos relacionados, lo que permitió contrastar hallazgos previos y consolidar un marco de referencia sólido para el análisis. Esta estructura no solo priorizó la claridad conceptual, sino que también estableció conexiones explícitas entre la selección de optimizadores, las metodologías de implementación y las métricas de evaluación, con el fin de garantizar validez interna al proceso investigativo.

4.1. Antecedentes

La regresión logística se considera un algoritmo de Machine Learning (ML) con un potencial relevante en la detección de noticias falsas de política en español; sin embargo, su aplicación en esta área no ha recibido la optimización necesaria. La ausencia de experimentación con métodos de optimización como Gradiente Descendente (GD), Gradiente Descendente Estocástico (SGD), Gradiente Descendente por Mini-Lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam) y Propagación de la Raíz del Promedio Cuadrático (RMSProp) dificulta cuantificar su impacto en la precisión del clasificador. Esta carencia de pruebas culmina en un problema de optimización que limita el rendimiento del modelo.

En este contexto, el algoritmo de Gradiente Descendente (GD) resulta esencial para el entrenamiento de los modelos de ML. Dichos métodos tienen por objetivo minimizar la función de pérdida mediante el ajuste iterativo de los parámetros, orientando el proceso en la dirección del gradiente negativo de la función con respecto a dichos parámetros [3].

Los algoritmos de optimización influyen directamente en la precisión del modelo de regresión logística, porque han demostrado un incremento de la precisión, pasando de una precisión inicial sin optimización de un 50% hasta aproximadamente un 90% optimizado, en

el contexto de la detección de perfiles falsos en redes sociales, lo que demuestra una exitosa aplicación del método de optimización gradiente descendente [4].

El objetivo principal es optimizar el modelo regresión logística usando los algoritmos de optimización Gradiente Descendente (GD), Gradiente Descendente Estocástico (SGD), Gradiente descendente por Mini-Lote (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam) y Propagación de la Raíz del Promedio Cuadrática (RMSProp) en el descubrimiento de noticias falsas de política en español y comparar la precisión de este modelo optimizado con la precisión inicial, facilitando la validación de la autenticidad de los contenidos divulgados en medios de comunicación digital y redes sociales relacionados con política en idioma español.

4.2. Fundamentación Teórica

4.2.1. Machine learning (ML)

Machine learning es una técnica que se sustenta en el aprendizaje a partir de la experiencia, lo que significa que el sistema aprende de su “propia experiencia” interactuando con los datos. En lugar de depender de instrucciones explícitas, el sistema genera respuestas a partir del análisis de los datos de entrada. Se volverá cada vez más inteligente y preciso después de cada ciclo de aprendizaje, sin la necesidad de intervención humana. Machine learning se encarga de hacer que las computadoras aprendan a partir de un dataset y del desarrollo de algoritmos que puedan extraer patrones de distintos datos. De esta manera, se cambia el enfoque de la programación convencional, en la que se programa paso a paso la solución a una necesidad planteada [5][6].

La Figura 1 presenta un esquema general de las categorías y algoritmos utilizados en las diferentes técnicas de machine learning. Cada técnica está diseñada orientado a un tipo particular de problema y se clasifica en cuatro categorías: (aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semi-supervisado, aprendizaje por refuerzo) [7].

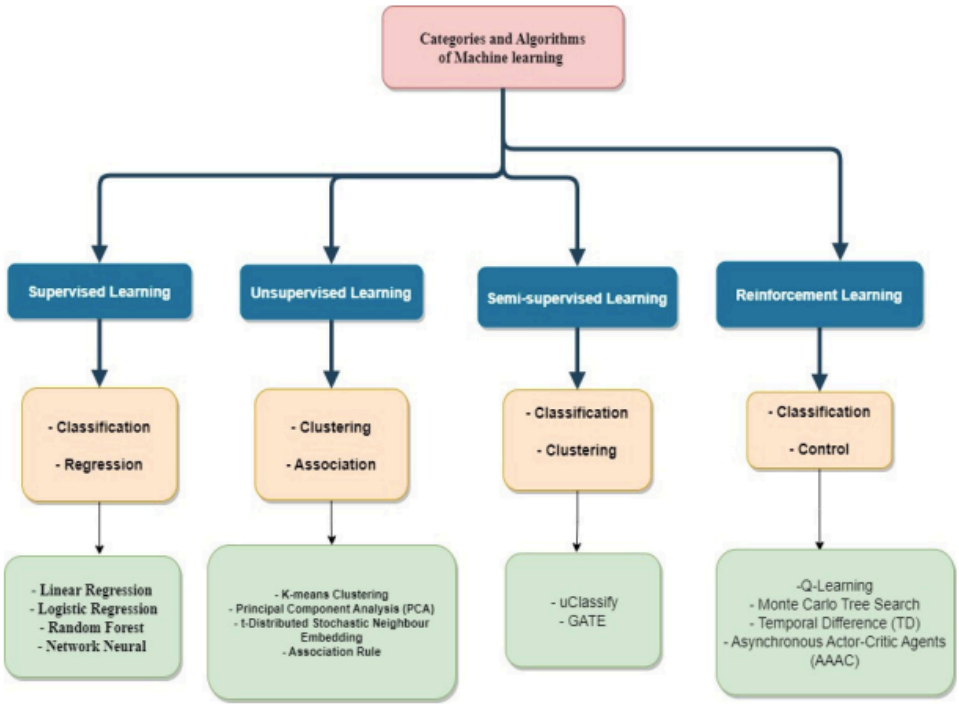


Figura 1. Categorías y algoritmos de machine learning [7].

4.2.2. Técnicas de Machine learning

4.2.2.1. Aprendizaje supervisado

En aprendizaje supervisado, el algoritmo es entrenado con un conjunto de datos etiquetado, es decir, las respuestas correctas ya están incluidas en el conjunto de entrenamiento. La tarea principal se basa en mapear las entradas a las salidas que correspondan. En el conjunto de datos existe una entrada y una salida dadas como ejemplo. Este tipo de aprendizaje es exitoso en la tarea de clasificación en diversas áreas [8].

4.2.2.2. Clasificación

La clasificación se concebía como un método de aprendizaje supervisado en el que se anticipaba una etiqueta de clase para un caso específico. Desde el punto de vista matemático, se define como una función (f) que relaciona las variables de entrada (X) con las variables de salida (Y), asignando así la etiqueta, el objetivo o la categorización correspondiente. Se lleva a cabo en datos estructurados y no estructurados. Un ejemplo, en la detección de spam como “spam” y “no spam” en los proveedores de servicios de correos electrónicos, lo que se considera un problema de clasificación. En la Figura 2 se representa un ejemplo de clasificación con modelos de regresión, donde la línea de puntos representa el límite lineal que separa dos clases [9].

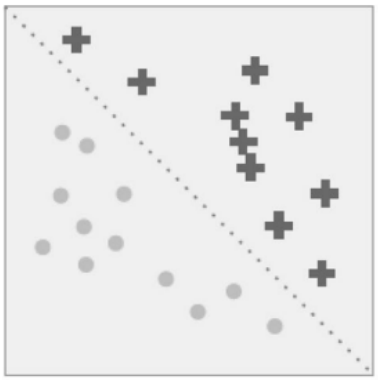


Figura 2. Representación del método de clasificación [8].

4.2.2.2.1. Clasificación binaria

Los algoritmos de clasificación binaria son similares a enseñar a un niño a distinguir entre un perro y un gato. Podemos presentar al niño varias imágenes de perros y gatos, donde se le enseñará a diferenciar entre los animales dependiendo de sus características. Después de ese “entrenamiento”, cuando el niño vea un perro o un gato en la calle, será capaz de distinguirlos y decir el nombre del animal. En el ejemplo, el nombre del animal correspondiente en las fotos es lo que se llama “etiqueta” de los datos de entrada. Por simple y sencillo que parezca el ejemplo anterior, existe mucha matemática, álgebra lineal y estadística detrás de los algoritmos que permiten a la computadora realizar este tipo de clasificación [10].

4.2.2.3. Aprendizaje no supervisado

El algoritmo trabaja con datos no etiquetados, los cuales no conoce, por lo que debe encontrar patrones o estructuras ocultas sin ninguna indicación ni interferencia humana. El entrenamiento se define de manera similar al supervisado, pero la diferencia radica en la comprensión de nuevos patrones, que se buscan en grupos de datos con similitudes, identificando tendencias, agrupaciones y estructuras significativas. Ese es el reto del aprendizaje no supervisado. Se reconoce que las tareas más comunes en aprendizaje no supervisado abarcaban la agrupación, la estimación de densidad, el aprendizaje de características, la reducción de dimensionalidad, la búsqueda de reglas de asociación y la detección de anomalías, entre otras [9][11].

4.2.3. Noticias falsas

Las "noticias falsas" o "fake news" se refieren a la desinformación o información falsa presentada como si fuera una noticia real. Pueden ser comprendidas como una exageración o modificación de una noticia o hecho, con el único fin de generar desinformación o confusión en la audiencia. Se presentan en formatos que imitan al de una noticia usual. Existen muchos propósitos al que van destinados las noticias falsas, tales como favorecer intereses políticos, influir en debates públicos o simplemente conseguir ganancias económicas al direccionar el tráfico de visitas a estos medios de comunicación digital, que en su mayoría son alojados en sitios web o en redes sociales. Uno de sus distintivos es que no existe una revisión editorial

ni algún ente que asegure el acatamiento de estándares mínimos, como la constatación de fuente, la valoración del impacto o la precisión de la información [12].

Algunos rasgos comunes de las noticias falsas incluyen [13]:

- **Titulares sensacionalistas:** Suelen estar exagerados, con el objetivo de ser alarmistas, provocando curiosidad o morbo.
- **Falta de fuentes creíbles:** Las noticias falsas generalmente no citan fuentes confiables o verificables.
- **Apelación emocional:** Están diseñadas para explotar emociones; deben generar un gran impacto emocionar en el posible lector.
- **Transmisión de mensajes sencillos:** Se estructura de manera similar a cuentos infantiles, definiendo rápidamente a los buenos y a los malos. De esta manera, se limita la interpretabilidad por parte del lector.
- **Imágenes que impactan:** Siempre se encuentran imágenes que provocan conmoción, haciendo referencia directa al título.
- **Imágenes manipuladas con Inteligencia Artificial:** Cada vez es más difícil corroborar imágenes debido a la tecnológica de la inteligencia artificial (IA) que permite retocarlas, con el objetivo de transmitir una falsa legitimidad de la noticia.

En la Figura 3 se representa un ejemplo de una noticia real frente a una noticia falsa: la noticia real se presenta en la parte superior de la ilustración y en la parte inferior se muestra la noticia falsa [14].

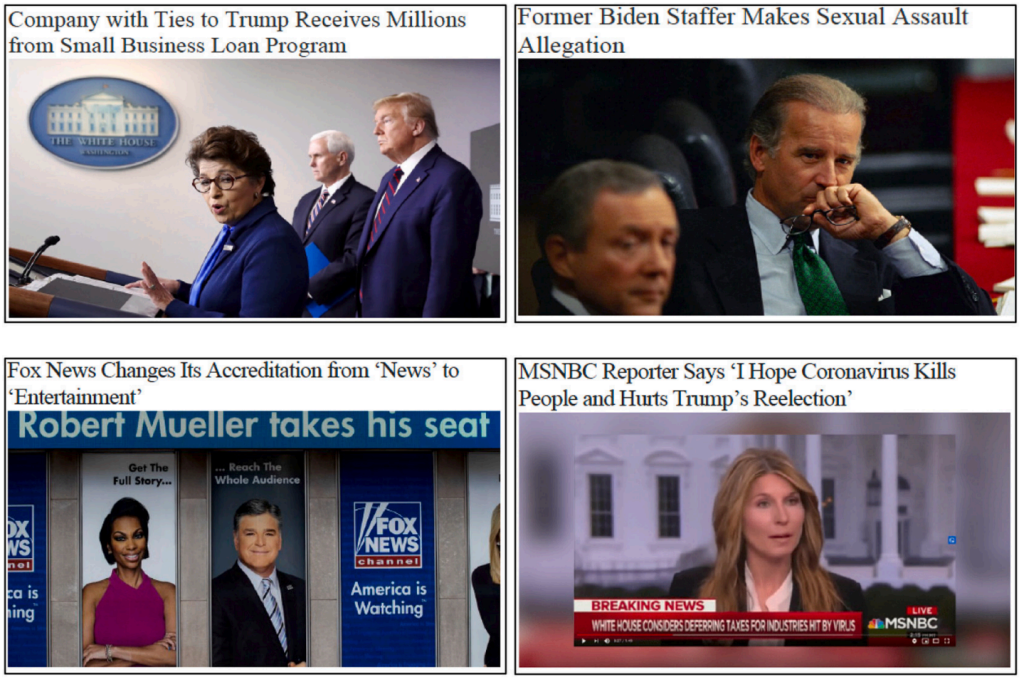


Figura 3. Ejemplos de titulares de noticias reales (arriba) y falsos (abajo) [13].

Los algoritmos que recopilan datos de los usuarios pueden convertirse en el objetivo de propagandas políticas personalizada. Las personas igualmente pueden estar expuestas a

noticias falsas, compartidas por personas en redes sociales o difundidas por medios digitales no confiables. Estos comportamientos ayudan a la difusión de noticias falsas de forma rápida. La exposición repetida a noticias falsas es una amenaza para la sociedad [15].

4.2.4. Algoritmo de clasificación

4.2.4.1. Regresión Logística

Esta técnica estadística se utiliza para modelar variables dependientes que solo pueden asumir dos categorías, como binario 1/0, sí/no, éxito/fracaso, entre otros. La regresión logística es una opción adecuada para evitar posibles problemas de ineficiencia o sesgo que podrían surgir en un modelo lineal. Su objetivo es modelar la relación entre variables de respuesta categóricas y las covariables. Existe una combinación lineal de variables independientes con probabilidades logarítmicas de un evento en un modelo logístico. En la ecuación mostrada en la Figura 4, "Y" representa la variable dependiente, "X" las variables independientes, "B" es el coeficiente de regresión y "e" denota el término de error [16][17].

$$\text{logit}(Y) = \alpha + X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + X_4\beta_4 + X_5\beta_5 + X_6\beta_6 + X_7\beta_7 + X_8\beta_8 + X_9\beta_9 + X_{10}\beta_{10} + X_{11}\beta_{11} + \varepsilon$$

Figura 4. Fórmula básica del modelo regresión logística [17].

En la Figura 5 se representa la función sigmoide, también llamada función logística, forma una curva en forma de S que puede tomar valores entre 1 y 0. Se dice que, si la curva viaja hacia un infinito positivo, se predice 1, y si viaja al infinito negativo, predice 0. En la siguiente figura se ilustra una función logística [18].

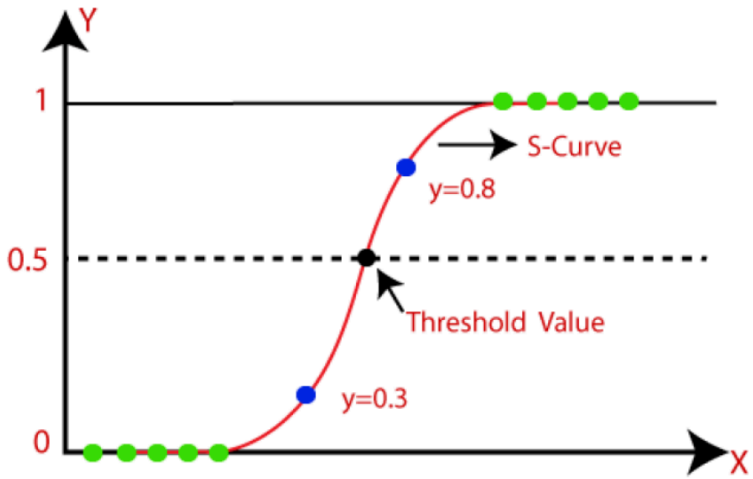


Figura 5. Función Logística [19].

El eje X representa el conjunto de datos y el eje Y muestra la probablidad de que la función logística asigne una entrada a X**.** Para los valores de X, la probabilidad es de 0, y cuando los valores aumentan en X, se acerca a 1. Los valores se interpretan como los clasificados como verdaderos positivos (en la parte superior, cerca de $Y = 1$) y verdaderos negativos (en la parte inferior, cerca de $Y = 0$) [19].

4.3. Fundamentación Conceptual

4.3.1. Algoritmos de optimización

4.3.1.1. Gradiente descendente (GD)

Gradiente descendente es un método que busca valores de los parámetros que ayuden a la minimización de la función de pérdida del modelo. Realiza pasos repetidos, llamados iteraciones, en la dirección opuesta al gradiente de la función. Las variantes del algoritmo de optimización gradiente descendente difieren según la cantidad de los datos usados en la función objetivo y la forma en que se actualizan los parámetros. Se debe encontrar un equilibrio entre el tiempo de ejecución de cada actualización y la actualización de los parámetros. En la Figura 6 se observa el gradiente junto con el mínimo de la función, que es lo que se desea conseguir [20].

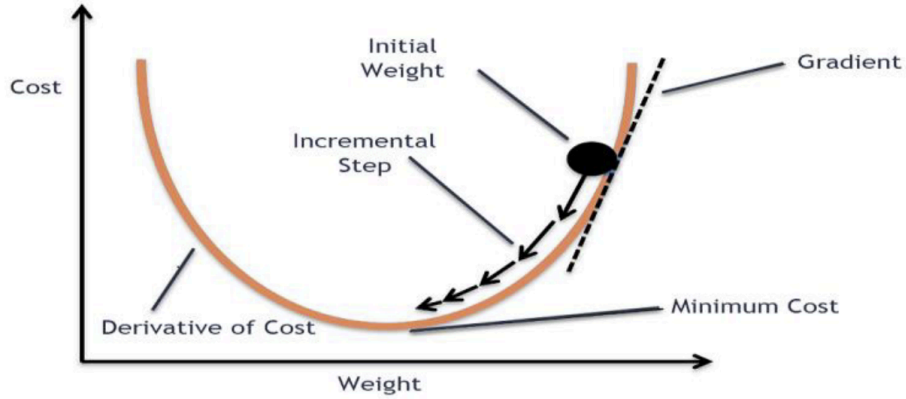


Figura 6. Modelo de gradiente descendente [20].

En la Figura 7 se define la fórmula básica del gradiente descendente donde [21]:

- $\theta^{(t)}$ es el vector de parámetros en la iteración, los parámetros θ son los que se intenta ajustar para minimizar la función objetivo $f(\theta)$.
- $f(\theta)$ es la función objetivo o llama función de costo.
- $\nabla_\theta f(\theta^{(t)})$ es el gradiente de la función $f(\theta)$ respecto a los parametros θ, calculado en la iteración t.
- η es la tasa de aprendizaje.
- $\theta^{(t+1)}$es el nuevo valor de los parámetros en la iteración siguiente $t+1$.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta f\left(\theta^{(t)}\right)$$

Figura 7. Fórmula básica del gradiente descendente [20].

4.3.1.2. Gradiente descendente estocástico (SGD)

Este algoritmo de optimización se usa para reducir la función de pérdida en los modelos de machine learning. Se aplica de manera repetitiva, siempre calculando el gradiente. Los parámetros se actualizan con un subconjunto aleatorio de datos, conocido como mini batch, haciéndolo más rápido y eficiente en términos de memoria, lo que es ideal

para laborar con grandes volúmenes de datos. El gradiente descendente estocástico acorta el tiempo de actualización cuando se trabaja con múltiples muestras y elimina la duplicación computacional, acelerando el cálculo. Su principal ventaja es que puede alcanzar una velocidad rápida de convergencia en problemas con convexidad fuerte. El gradiente estocástico, en lugar de calcular el gradiente de forma exacta, solo toma una muestra aleatoria para realizar estimaciones del gradiente en cada iteración o ciclo, es decir, realiza un solo cálculo por cada iteración [3][22].

En la Figura 8, las flechas rojas representan los pasos del gradiente estocástico, que busca el mínimo de la función, representado con el símbolo de "+" [23].

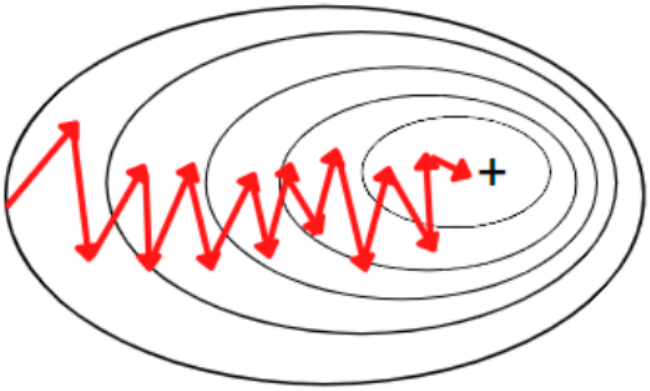


Figura 8. Representación de gradiente descendente estocástico [23].

En la Figura 9 se presenta la fórmula del gradiente descendente estocástico, donde la función de pérdida $\mathcal{L}$ se calcula sobre mini-lotes de datos en cada iteración, usando retropropagación hasta que el algoritmo converge [24].

$$\theta_{t+1} = \theta_t - \eta \nabla_i \mathcal{L}(\theta_t)$$

Figura 9. Fórmula básica del gradiente descendente estocástico [24].

La fórmula se conforma de lo siguiente [24].

- θ_t, representa los parámetros de la red en el instante t.
- i se refiere a la muestra del mini-lote.
- η es la tasa de aprendizaje.

4.3.1.3. Gradiente descendente por mini lotes (MBGD)

El gradiente descendente por mini lotes considera una pequeña cantidad de ejemplos en vez de seleccionar todos los ejemplos del entrenamiento. Esto ayuda al cálculo de un gradiente para actualizar los pesos en cada iteración de forma más eficiente. El descenso de gradiente por mini lotes tiene como objetivo reducir la varianza de las actualizaciones de los pesos, lo que resulta en una convergencia más estable y en actualizaciones más frecuentes, dando lugar a un aprendizaje más rápido. Es la implementación más común del gradiente

descendente en el campo de machine learning. Aunque el algoritmo de optimización por mini lotes es muy efectivo gracias a su sencillez, su principal limitación radica en la sensibilidad a valores atípicos, o outliers, especialmente cuando se utiliza con funciones de pérdida cuadrática [25][26][27].

En la Figura 10 se representa el comportamiento del gradiente descendente por mini lotes. Las curvas de nivel plasman la función de costo, es decir, muestran cómo cambia la función de costo a medida que cambian los parámetros, mientras que las líneas rojas muestras el recorrido que siguen los parámetros del modelo [28].

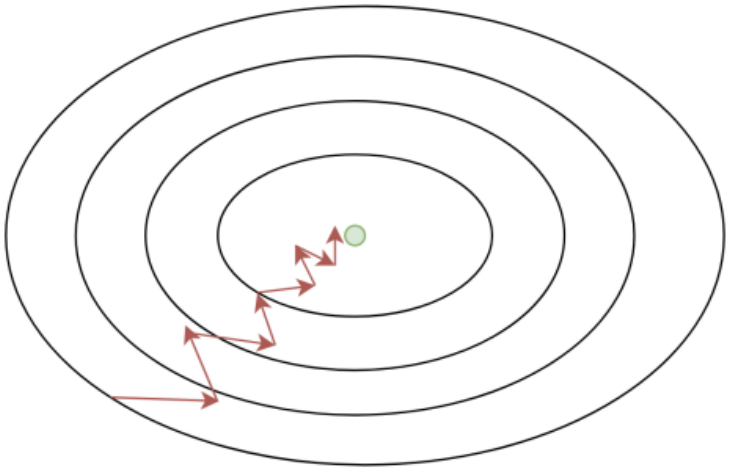


Figura 10. Representación de curvas de función de costo MBGD [28].

La Figura 11 se presenta la ecuación matemática del algoritmo MBGD, donde [29]:

- η representa la tasa de aprendizaje.
- z_t es un min-lote elegido al azar.
- θ_t es el conjunto de parámetros o pesos en la t de la iteración.
- $\nabla_\theta MSE(z_t, \theta_t)$ es el gradiente de la función de costo (error cuadrático medio).

$$\theta_{t+1} = \theta_t - \eta \times \nabla_\theta MSE(z_t, \theta_t)$$

Figura 11. Fórmula básica de MBGD [29].

4.3.1.4. Algoritmo de gradiente adaptativo (AdaGrad)

AdaGrad, o Adaptive Gradient Algorithm, ajusta la tasa de aprendizaje de los parámetros mediante actualizaciones pequeñas para aquellos que tienen características comunes y tasas superiores para lo que tienen menos frecuencia, resultando efectivo en datos dispersos. Se ha demostrado que tiene éxito en muchas tareas que se relacionan a machine learning. AdaGrad es un método adaptativo donde la tasa de aprendizaje se ajusta individualmente para cada parámetro durante el entrenamiento. Sin embargo, AdaGrad tiene un inconveniente: la tasa de aprendizaje se ralentiza durante el entrenamiento, lo que provoca que el rendimiento del aprendizaje sea insatisfactorio en múltiples épocas [30][31].

En la Figura 12 se presenta la ecuación matemática de AdaGrad, donde [32]:

- η representa la tasa de aprendizaje.
- θ_t son los parámetros del modelo en la iteración t.
- G_t es la matriz de escala, que ajusta la magnitud del paso para cada parámetro.
- ∇_t es el gradiente de la función de pérdida con respecto a los parámetros.

$$\theta_{t+1} = \arg\min_{\theta \in \mathcal{C}} \|\theta - (\theta_t - \eta G_t^{-1} \nabla_t)\|_{G_t}^2,$$

Figura 12. Fórmula básica de AdaGrad [32].

4.3.1.5. Estimación de momento adaptativo (Adam)

Estimación de momento adaptativo (Adam) es un método de optimización basado en la optimización estocástica, donde se necesita el gradiente de primer orden, lo que resulta en una utilización de memoria muy reducida. El método selecciona lo mejor del modelo AdaGrad, pero con el beneficio de mejorar la efectividad para gradientes dispersos, almacenando los promedios de los gradientes y los cuadrados de los gradientes de las iteraciones previas. Adam es un algoritmo de optimización adaptativo altamente eficiente, que se utiliza frecuentemente como reemplazo del método tradicional de reducción del gradiente estocástico. Adam emplea un método de actualización de parámetros similar al gradiente descendente con RMSProp utilizando la media móvil exponencial del gradiente al cuadrado (v_t) y media móvil exponencial del gradiente (m_t). Por lo tanto, existe una combinación de RMSProp y Momentum, se utiliza la siguiente ecuación [33][34].

En la Figura 13 se representa la ecuación matemática de Adam, donde [35]:

- θ_t son los parámetros del modelo en el paso t.
- η representa la tasa de aprendizaje.
- m_t es el valor de la primera estimación del momentum.
- v_t es la estimación de la segunda estadística de momentum.
- $\in$ es un valor pequeño añadido para evitar divisiones por cero.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t^{\wedge} + \in}} m_t^{\wedge}$$

Figura 13. Fórmula básica de Adam [35].

4.3.1.6. Propagación de la raíz del promedio cuadrático (RMSProp)

Este método está diseñado para reducir la cantidad de evaluaciones de la función, mejorando el rendimiento del algoritmo. Está basado principalmente en el cálculo del promedio ponderado de los cuadrados de los gradientes, muy similar al proceso que realiza el gradiente con momentum. Su diferencia radica en la actualización de los parámetros, usando una tasa de aprendizaje adaptada a partir del promedio ponderado de los cuadrados del gradiente, en donde se tiene en cuenta tanto el peso como el sesgo. RMSProp tiene una

característica especial: restringe el balanceo en la dirección vertical, ayudando a que la tasa de aprendizaje sea más rápida en la dirección horizontal, lo que acelera la convergencia. Esto lo hace muy eficiente y requiera poca memoria. Además, RMSProp acelera el proceso de optimización al reducir las evaluaciones de la función. Este optimizador se formula al tomar el promedio de los pesos al cuadrado de los gradientes y dividirlo por la raíz cuadrada del cuadrado medio [36][37].

En la Figura 14 se representa la ecuación de actualización del algoritmo RMSProp es la siguiente, donde [34]:

- g_t es la primera derivada con respecto a la función de pérdida.
- β_2 es la tasa de decaimiento exponencial.
- η representa la tasa de aprendizaje.
- θ_t son los parámetros del modelo en el paso t.
- $\in$ es un valor pequeño añadido para evitar divisiones por cero.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \in}} g_t$$

Figura 14. Fórmula básica de RMSProp [34].

En la Figura 15 se representa el comportamiento de la propagación de la raíz del promedio cuadrático (RMSProp). Las curvas de contorno plasman la función de costo, es decir, muestran cómo cambia la función de costo a medida que se modifican los parámetros. Las líneas rojas indican los pasos del algoritmo en diferentes iteraciones, donde se puede observar que los pases iniciales son más largos. RMSProp permite al optimizador tomar pasos más controlados hacia el mínimo de la función, evitando saltos grandes o inestables[1].

[1] Proceso general de búsqueda de la solución óptima RMSProp

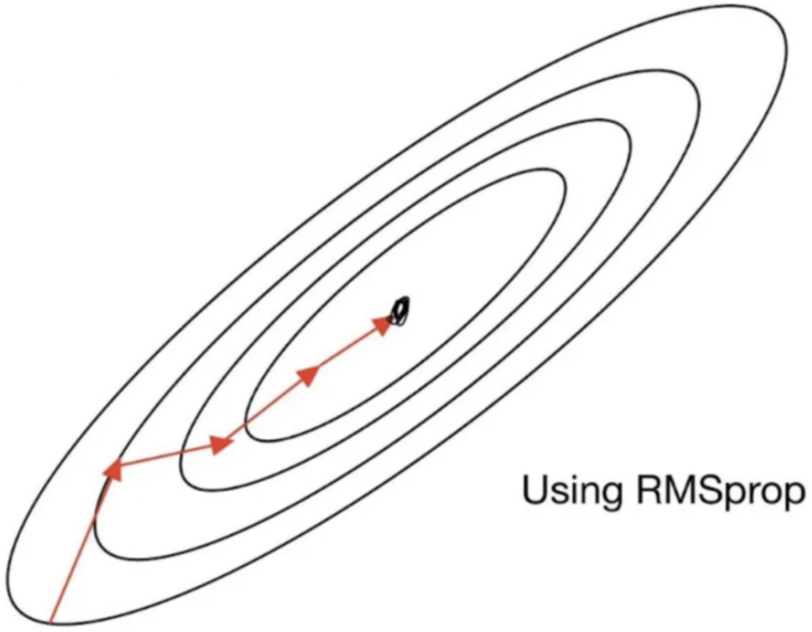


Figura 15. Representación de curvas de función de costo RMSProp[1].

4.3.2. Resumen de algoritmos de optimización

La Tabla 1 resume algunas de los algoritmos de optimización más utilizadas, detalla las propiedades, ventajas y desventajas. Entre ellas se encuentran lo algoritmos mencionados anteriormente desde el descenso de gradiente hasta las variantes más modernas como RMSProp y Adam, que han demostrado ser eficaces en aplicaciones de gran escala.

Tabla 1. Resumen algoritmos de optimización.

Algoritmo	Propiedades	Ventajas	Desventajas
Gradiente descendente (GD).	Con cada actualización, el algoritmo encuentra el objetivo y gradualmente converge a los valores óptimos.	La función objetivo es convexa, facilitando la tarea de encontrar la mejor solución posible.	El costo de cálculo es alto, ya que opera ajustando elementos en dirección inversa al gradiente.
Gradiente descendente estocástico (SGD).	Cada iteración está basada en una muestra aleatoria que actualiza el gradiente en lugar de calcular directamente el gradiente.	El costo de cálculo se reduce, ya que el tiempo que toma calcular cada actualización es independiente del número total de muestras de entrenamiento.	Determinar el tamaño adecuado de la tasa de aprendizaje es difícil.
Gradiente descendente por mini lotes (MBGD).	Se basa en mini-lotes (pequeñas porciones) de los datos para actualizar el gradiente en lugar de calcular directamente el gradiente.	Reduce la varianza en las actualizaciones y mejora la estabilidad. Acelera el entrenamiento frente a grandes conjuntos de datos.	Elegir el tamaño adecuado del lote es un desafío. Puede no ser tan preciso como el gradiente descendente ni tan rápido como el gradiente estocástico.
Algoritmo de gradiente adaptativo (AdaGrad).	La tasa de aprendizaje será baja siempre y cuando el gradiente sea alto, y viceversa.	Se puede usar para resolver problemas con gradiente disperso. La tasa de aprendizaje de cada parámetro se ajusta adaptativamente.	Para abordar problemas no convexos no es adecuado.

Algoritmo	Propiedades	Ventajas	Desventajas
Estimación de momento adaptativo (Adam).	Combina el algoritmo del momentum con métodos adaptativos.	Con grandes cantidades de datos y un espacio de características más grande, es eficaz para resolver la mayoría de los problemas de optimización no convexos.	En algunos casos, el enfoque puede no converger.
Propagación de la raíz del promedio cuadrado (RMSProp).	Cambia el camino de la acumulación del gradiente total a un promedio móvil exponencial.	Se utiliza para resolver problemas no estacionarios y no convexos. Adecuado para espacios grandes y multidimensionales.	La tasa de actualización se vuelve a calcular alrededor del mínimo local en el último período de entrenamiento.

4.3.3. Matriz de confusión

E s una herramienta vital para evaluar cualquier modelo optimizado permitiendo verificar el desempeño del modelo regresión logística en la predicción de noticias falsas de política en español facilitando de gran manera la interpretación del rendimiento. En la matriz las clases predichas se visualizan en las filas y en las columnas serán representadas las clases reales, tal cual se muestra en la Figura 16. Donde, se identifican los elementos de la matriz: verdaderos positivos (TP), falsos positivos (FP), falsos negativos (FN) y verdaderos negativos (TN). La matriz de confusión permite una identificación de patrones mal clasificados, así como una visión del comportamiento del modelo tanto de aciertos como de errores [38].

Figura 16. Modelo de matriz de confusión binaria [38].

Se presentan las diferentes métricas que conforman la matriz de confusión en la siguiente lista [39][40]:

- **Verdaderos positivos (TP) -** Son las instancias en las que el modelo predice correctamente la clase positiva. Es decir, es la cantidad de datos positivos que el sistema clasifica correctamente.

- **Falsos positivos (FP) –** El valor de falso positivo es la cantidad de datos positivos pero que en realidad el sistema clasificó incorrectamente.
- **Falsos negativos (FN) -** Se da cuando el modelo predice que una instancia es negativa, cuando en realidad es positiva. El valor falso negativo es la cantidad de datos negativos que es sistema clasifica incorrectamente.
- **Verdaderos negativos (TN) -** Sucede cuando el modelo predice correctamente que una instancia es negativa, es la cantidad de datos negativos que el sistema clasifica de manera correcta.
- **Precisión (Precision) –** Es una medida utilizada para evaluar las instancias positivas entre las instancias recuperadas, se calcula mediante la proporción entre las predicciones verdaderas positivas y el conjunto de todos los valores positivos.

$$Precision = \frac{TP}{TP + FP}$$

- **Especificidad (Specificity) -** Es la proporción de verdaderos negativos con respecto al total de casos realmente negativos. Se traduce como la capacidad del modelo en predecir casos negativos.

$$Specificity = \frac{TN}{TN + FP}$$

- **Exactitud (Accuracy) -** Es el porcentaje de predicciones correctas (positivas y negativas) hechas por el modelo sobre el total de predicciones realizadas.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Sensibilidad (Sensitivity) –** Marca la capacidad del modelo para predecir casos negativos. Es la proporción entre verdaderos positivos entre verdaderos positivos y falsos negativos.

$$Sensitivity = \frac{TP}{TP + FN}$$

- **F1 Score -** Es la media armónica entre la precisión y la sensibilidad (recall). Es una métrica que toma en cuenta la precisión y la sensibilidad de manera conjunta.

$$F_1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

4.3.4. Web Scraping

Es una técnica que permite extraer datos estructurados a partir de documentos como HTML. Es altamente útil en escenarios donde los datos no están disponibles en formatos fácilmente legibles o descargables, ya que permite obtener información más rica y compleja que la entrada manual. Esta herramienta resulta esencial en la era de la información, siendo ampliamente utilizada en campos como la inteligencia empresarial, inteligencia artificial, ciencia de datos, big data y ciberseguridad. Además, se integra de manera eficiente con

lenguajes de programación como Python, lo que la hace una alternativa flexible y popular en estos contextos [41].

El proceso de Web Scraping se representa en Figura 17 y está dividido en tres etapas fundamentales.

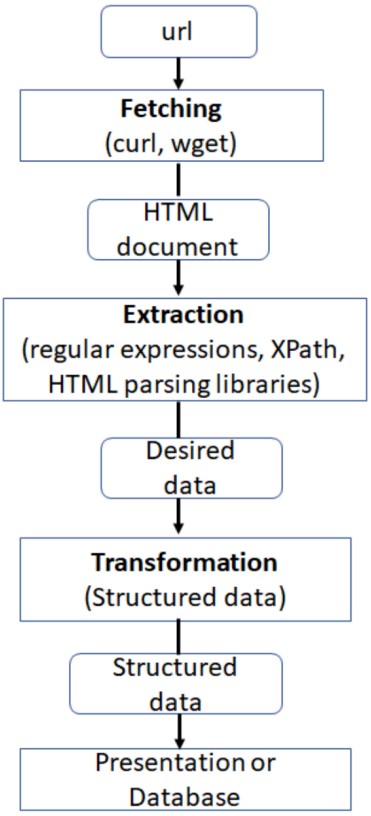


Figura 17. Proceso de extracción de datos (Web Scraping).

Etapas de web scraping [41]:

- **Etapa de obtención:** Consiste en acceder al sitio web que contiene la información deseada. Esto se realiza principalmente utilizando protocolos HTTP, los cuales permiten enviar solicitudes al servidor y recibir respuestas con los datos necesarios.
- **Etapa de extracción:** Una vez descargada la página HTML, se procede a identificar y extraer los datos relevantes. Para ello, se emplean herramientas como expresiones regulares, bibliotecas de análisis de HTML (como BeautifulSoup o lxml) o consultas XPath.
- **Etapa de transformación:** Los datos extraídos se procesan y convierten a un formato estructurado, como JSON, CSV o bases de datos, facilitando su almacenamiento, análisis o presentación.

4.4. Fundamentación Metodológica

4.4.1. Metodología

4.4.1.1. CRISP-ML

CRISP-ML se basa en la revisión y actualización de CRISP-DM incorporando los elementos necesarios para trabajar con diversas técnicas de machine learning y crear el nivel adecuado de interpretabilidad a través de todo el proceso de creación de la solución de machine learning [42].

En la Figura 18 se presenta el proceso estándar para la construcción de aplicaciones de machine learning utilizando la metodología CRISP-ML, que es una versión mejorada de CRISP-DM. Esta metodología garantiza la calidad en cada una de las fases del proceso.

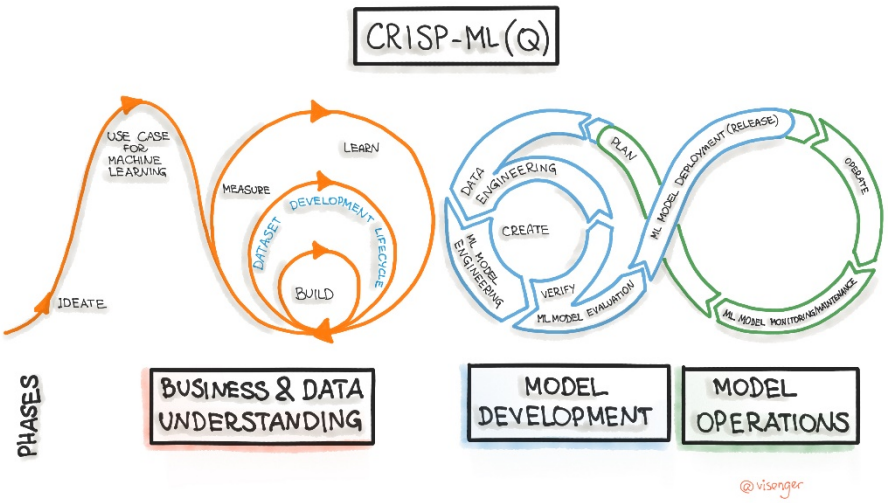


Figura 18. Proceso para el desarrollo de aplicaciones ML con metodología CRISP-ML[2].

Algunas ventajas de utilizar CRISP-ML se mencionan a continuación [43].

- **Expansión de CRISP-DM. -** Expansión de CRISP-DM. — CRISP-ML se expande a partir de CRISP-DM, abordando las necesidades del Machine Learning en cuanto a la adaptación de entornos cambiantes, incluyendo la monitorización y el mantenimiento constante de los modelos desplegados previamente.
- **Fases del proceso.** - CRISP-ML abarca varias fases de una aplicación de ML, incluyendo la comprensión del negocio y de los datos, ingeniería de datos, ingeniería de modelos de machine learning, la evaluación, el despliegue y el mantenimiento o monitoreo.
- **Iteración y mejora continua. -** CRISP-ML es iterativo, lo que permite retroceder a fases anteriores para ajustar el modelo según sea necesario. Esta flexibilidad y mejora continua permite mitigar riesgos durante el desarrollo del modelo, asegurando así la calidad y eficacia del en cada paso.

[2] Proceso de metodología CRISP-ML.

- **Enfoque industria y aplicación neutral. -** Es un modelo de procesos con un enfoque neutro y adecuado para una variedad de desafíos en ML, considerando su aplicabilidad y estabilidad como una metodología segura y sólida.

La utilización de CRISP-ML asegura que el modelo optimizado a través algoritmo de optimización GD y sus variantes se realice de una manera completa y planificada, generando resultados precisos. Además, la metodología CRISP-ML garantiza facilidad en el estudio y la aplicación en el proyecto gracias a su estructura y práctica. Esta estructura detallada ayudará a mejorar significativamente la eficiencia y eficacia del proyecto, asegurando resultados más confiables y aplicables en un entorno real.

4.4.2. Librerías

4.4.2.1. Pandas

Es una biblioteca de código abierto para análisis y gestión de datos, rápido, potente y fácil de usar, creada para el lenguaje de programación Python y utilizada para la preparación y análisis de datos. Si se trabaja con una gran cantidad de datos, Pandas facilita el trabajo con los mismo. Las principales características de Pandas se pueden clasificar de la siguiente manera:

- Explorar datos rápidamente.
- Leer varios formatos de archivos.
- Limpiar datos.
- Manipular datos

Pandas trabaja con objetos Data Frame, que son estructuras de datos bidimensionales en las que los objetos se almacenan de manera tabular en forma de filas y columnas. Pandas se utiliza en diversos ámbitos, como la atención sanitaria, para evaluar el riesgo de enfermedades crónicas y en el sector energético, para mejorar el rendimiento y reducir costes de mantenimiento mediante predicciones de fallos en dispositivos [44][45].

4.4.2.2. Pytorch

PyTorch es un marco de machine learning de código abierto desarrollado por el equipo de inteligencia artificial de Facebook. Destaca por su flexibilidad para crear y entrenar modelos de machine learning, y su capacidad para trabajar tanto con CPU como con GPU, lo que lo hace ideal para diversas aplicaciones. Una de sus principales ventajas es la posibilidad de modificar modelos en tiempo real, lo que facilita la experimentación con arquitecturas complejas. Además, PyTorch ha ganado popularidad gracias a su facilidad de uso, versatilidad y el sólido respaldo de su comunidad [46].

4.4.2.3. Scikit-learn

Es una biblioteca de machine learning de propósito general en Python. Proporciona implementaciones de algoritmos de generación última para problemas supervisados y no supervisados. Las aplicaciones posibles de scikit-learn son muy amplias. Scikit-learn Prioriza

la facilidad de uso, el rendimiento, una documentación clara y la coherencia de la API. Su diseño con dependencias mínimas favorece su adopción tanto en entornos académicos como comerciales [47].

4.4.2.4. Natural Language Toolkit (NLTK)

Es una biblioteca que se usa específicamente para la lingüística, cubriendo el procesamiento del lenguaje natural simbólico y estadístico. El procesamiento del lenguaje natural utiliza varias técnicas para la generación, manipulación y análisis del lenguaje natural o humano. Una de las ventajas de esta librería es que puede procesar texto en lenguaje natura de una forma rápida y sencilla. NLTK tiene una gran cantidad de corpus integrados, que incluyen varios tipos de textos como novelas, textos de chat, reseñas de películas, noticias, corpus de discursos, etc. Contiene un paquete de bibliotecas de procesamiento de texto para clasificar, tokenizar, derivar, etiquetar, analizar y razonar semánticamente [48].

En la Figura 19 se muestra los pasos seguidos para resumir el texto de una noticia de fútbol utilizando la biblioteca NLTK [49].

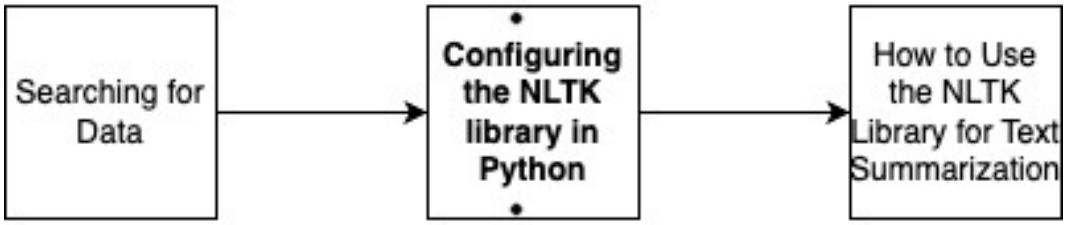


Figura 19. Etapas para resumir texto con NLTK [49].

4.4.3. Herramientas y tecnologías

4.4.3.1. Dataset

Se coleccionan conjuntos de datos sobre noticias falsas de política en español, ya que un dataset diverso es fundamental para entrenar y evaluar los modelos de ML. “Un dataset es un conjunto de muestras que son recopiladas y disponibles para desarrollar modelos en el ámbito de ML [50]“.

4.4.3.2. Google colab

Es una plataforma en la nube que se basa en cuadernos Jupyter. Google Colab brinda la oportunidad de acceder de forma gratuita a GPU y TPU, y se conecta con Google Drive. Esto posibilita disfrutar de todas las bondades de los cuadernos Jupyter junto con la sólida infraestructura de Google. El único requisito para utilizar este servicio, es necesario disponer de conexión a internet y una cuenta de Gmail [51].

4.4.3.3. Python

Python Es valorado por su sencillez y eficacia, destacándose por su sólido ecosistema de bibliotecas centradas en el aprendizaje automático. Estas cualidades lo convierten en una opción ideal para aplicaciones de clasificación de texto debido a su eficaz gestión de datos y modelos de aprendizaje automático. Por este motivo, Python será el principal lenguaje de

programación considerado para la implementación de los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp en el modelo Regresión Logística [52].

4.4.3.4. Anaconda

Anaconda es un paquete de software gratuito que incluye una serie de herramientas diseñadas para la ciencia de datos, permitiendo el acceso a entornos con Python o R. Anaconda facilita la actualización de bibliotecas de manera sencilla y permite instalar y usar Python fácilmente mediante una interfaz gráfica compatible con las bibliotecas más importantes. Dentro de Anaconda, tenemos Jupyter, un programa que se basa en la web que se ejecuta en el navegador de forma local. Cada bloque de código se ejecuta de forma independiente, lo que lo convierte en un programa flexible y fácil de usar. Permite el uso de diferentes tipos de texto en un mismo notebook, lo que facilita acceder a la salida del código, visualizar ecuaciones y textos sin formato, generando facilidad en la creación y distribución de documentos [53].

4.5. Trabajos Relacionados

La revisión sistemática de literatura emprendida durante la investigación permitió reunir trabajos que sustentaron el presente trabajo. Se analizaron diversos enfoques metodológicos, lo que orientó la estrategia de investigación y la selección de técnicas de procesamiento de la información. En la Tabla 2 se presentan los principales textos que resultaron determinantes para el desarrollo del TIC, proporcionando su código, título y una breve descripción de su aporte e importancia en la elaboración de este proyecto. Cabe destacar que, de los 31 trabajos contemplados en la SLR, se han seleccionado 8 trabajos relacionados para incluirlos en esta tabla. Estos documentos brindaron bases teóricas sólidas y lineamientos para el entrenamiento y validación de resultados **(véase Anexo 7)**.

Tabla 2. Trabajos relacionados.

Código	Titulo	Descripción
	Detección de Perfiles Falsos en Sociales	
TR01	Fake Profile Detection Using Logistic Regression and Gradient Descent Algorithm on Online Social Networks	Este estudio muestra que el uso de Regresión Logística mejorada mediante el algoritmo de GD es eficaz para identificar perfiles falsos en plataformas de redes sociales. Como resultado, la precisión ha experimentado un notable aumento del 52% sin optimización al 85% con optimización [4].
	Detección de Noticias falsas	
TR02	Optimizing Hyperparameters and Performance Analysis of LSTM Model in Detecting Fake News on social media.	En este estudio se utilizaron una variedad de técnicas de optimización para mejorar la precisión en la detección de noticias falsas. Las técnicas empleadas fueron: SGD, AdaGrad, RMSProp y Adam. Estos métodos fueron esenciales para hacer un ajuste fino de los parámetros del modelo, logrando mejorar el rendimiento. Cada uno de estos algoritmos de optimización tuvo beneficios en términos de ajuste fino y tasa de aprendizaje [54].

Código	Titulo	Descripción
TR03	Fake News Detection Using Machine Learning Ensemble Methods.	Se utilizó la Regresión Logística y se aplicaron técnicas de optimización como SGD y Adam para mejorar la eficacia del modelo en la detección de noticias falsas. Estas técnicas de optimización ayudaron a refinar los parámetros del modelo Regresión Logística. Los resultados mostraron que el algoritmo de Regresión Logística sin optimizar alcanzaba una precisión del 87%. Al aplicar los algoritmos de optimización Adam y SGD, la precisión mejoró significativamente, aumentando a un rango de 91-93%, dependiendo del conjunto de datos utilizado [55].
TR04	Fake News Detection Using N-Gram Analysis and Machine Learning Algorithms.	el estudio de Asha J y Meenakowshalya A. (2021) explora el análisis del uso de N-gramas y varios algoritmos de ML. Los algoritmos evaluados fueron: Máquinas de Vectores de Soporte (SVM), SVM Lineal, K-Nearest Neighbor (KNN), SGD, Árboles de Decisión (DT) y Regresión Logística (LR). Se determinó que el algoritmo de optimización SGD, combinado con TF-IDF y unigramas, ofreció los mejores resultados para la detección de noticias falsas, alcanzando una precisión máxima del 94.2% después del ajuste [56].
TR05	Stance classification for fake news detection with machine learning	En el estudio “Stance Classification for Fake News Detection with Machine Learning” se analiza la detección de noticias falsas y se aplican algoritmos de optimización como SGD, mejorando la eficacia de los modelos tanto en la clasificación como en la detección de noticias falsas. Se utiliza el conjunto de datos AFND (Arabic Fake News Detection). El rendimiento después de la optimización muestra que la precisión de detección es del 87%. Los modelos de detección de noticias falsas pueden aplicarse o integrarse en plataformas de medios sociales para ayudar a la mitigación en la propagación de noticias falsas y, de esta manera, mejorar la calidad de la información que se ve día a día en línea [57].
TR06	Fake news detection in spanish using deep learning techniques	Este estudio explora diversos enfoques basados en machine learning para detectar noticias falsas. Martínez-Gallego et al. (2022) emplearon modelos preentrenados como BERT y ELMo para clasificar noticias falsas en español, logrando una precisión máxima del 80%, explorando diferentes estrategias y arquitecturas de entrenamiento para establecer una línea base para futuras investigaciones en esta área, incluido el uso de aprendizaje por transferencia y traducción automática para aprovechar modelos entrenados con datos en inglés [58].
TR07	Predicting and mitigating the effect of skewness on credibility assessment of social media content using machine learning: A twitter case study	En este estudio aplicaron SGD en redes neuronales recurrentes (RNN), mejorando la precisión de clasificación del 97% al 98% mediante procesos de optimización. Estos resultados destacan la eficacia de las técnicas de optimización incluso en escenarios complejos [59].

Código	Titulo	Descripción
TR08	Recent advances in stochastic gradient descent in deep learning	Este estudio analiza los avances recientes en el uso del método de descenso de gradiente estocástico (SGD) en el aprendizaje profundo, cubriendo sus aplicaciones en procesamiento de lenguaje natural, visión por computadora y procesamiento de audio, así como variantes de SGD como Adam, RMSprop y AdaGrad. Se evalúan condiciones teóricas y desafíos prácticos, destacando una brecha entre teoría y aplicaciones reales. También se proponen métodos adaptativos y de segunda orden para mejorar la convergencia y estabilidad en problemas no convexos [3].

TR: Trabajo relacionado.

5. Metodología

La metodología empleada en este trabajo se organizó en tres componentes principales: área de estudio, procedimientos y recursos. En el área de estudio, se detalla la ubicación donde se realizó y desarrollo el trabajo. Los procedimientos siguieron el marco CRISP-ML, dividiéndose en tres etapas clave: ingeniería de datos (selección, preparación, equilibrio de clases, limpieza, aumento, normalización y división de datos), ingeniería de modelos (ajuste de hiperparámetros, selección de optimización, elección del modelo y entrenamiento) y evaluación del modelo (evaluación y pruebas para validar su rendimiento). A continuación, se detallan los procedimientos realizados en cada fase, siguiendo el orden de las tareas propuestas. En cuanto a los recursos, se utilizaron recursos científicos (método Delphi, método experimental y estudio de caso), recursos técnicos (VSC, PyTorch, Jupyter Anaconda) y recursos computacionales (hardware especializado como GPUs), asegurando un enfoque robusto y reproducible en todas las etapas del trabajo.

5.1. Área de estudio

El trabajo de integración curricular (TIC) se desarrolló en el periodo de septiembre 2024 a febrero 2025 en la Universidad Nacional de Loja, Facultad de Energía y los Recursos Naturales no Renovables, específicamente en la carrera de Ingeniería en Computación, ubicada en la Avenida Reinaldo Espinosa.

En la Figura 20 se representan las coordenadas -4.030184, -79.199314 en Google Maps.

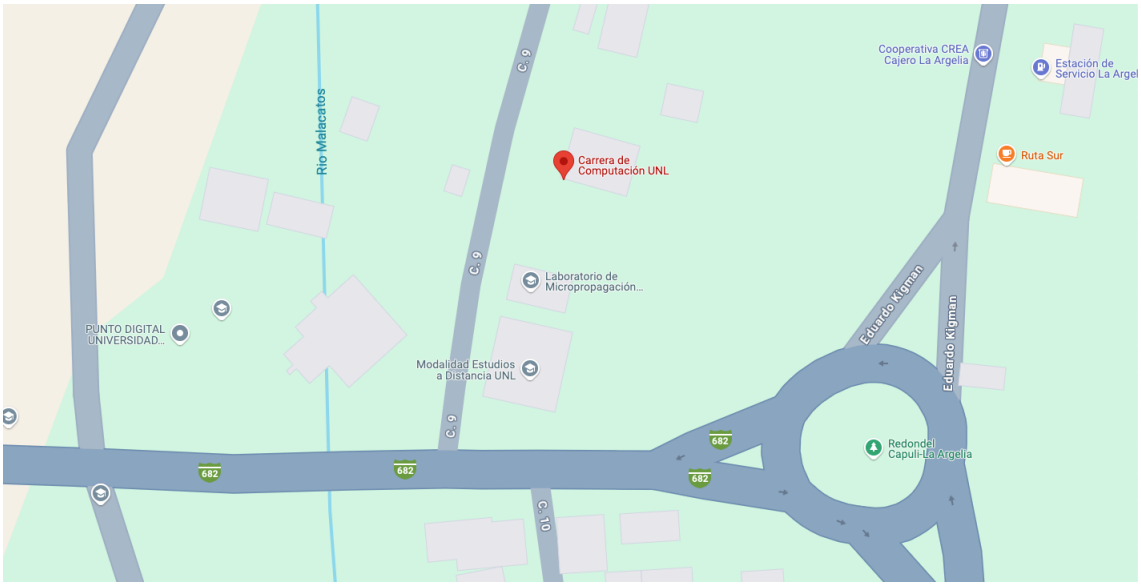


Figura 20. Ubicación carrera de Ingeniería en computación.

5.2. Procedimiento

Para desarrollar los objetivos planteado en el trabajo de integración curricular, se empleó un enfoque experimental que adaptó las fases de Ingeniería de datos, ingeniería de

modelos y evaluación, adaptando la metodología CRISP-ML. Dicha metodología, garantiza la facilidad del estudio y su aplicación en el proyecto gracias a su estructura y practicidad.

5.2.1. Objetivo 1: Entrenar el modelo regresión logística con los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp mediante CRISP-ML.

Para lograr el primer objetivo, se aplicó la metodología CRISP-ML, una adaptación de CRISP-DM orientada al desarrollo de soluciones de machine learning. Esta metodología, organizada en fases, guiando el proceso desde la comprensión del problema hasta la implementación del modelo. Se eligió CRISP-ML debido a su capacidad para adaptarse con flexibilidad a las necesidades del proceso. A continuación, se presentan las fases y tareas aplicadas en este objetivo:

Ingeniería de datos

- Selección de datos
- Preparación de los datos
- Equilibrio de clases
- Limpieza de datos
- Aumento de datos
- Normalización de datos
- División de datos

Ingeniería de Modelos

- Ajuste de hiperparámetros
- Selección de optimización
- Selección del modelo
- Entrenamiento de modelo

A continuación, se detallan los procedimientos realizados para cada fase y etapa, siguiendo el orden de las tareas propuestas **(véase Anexo 3)**.

Fase: Ingeniería de datos.

Tarea 1: Selección de datos (Personalización dataset)

Para la selección de datos, se procedió a personalizar y generar un dataset partiendo de periódicos nacionales (Ecuador), utilizando dos datasets disponibles en Kaggle: Spanish Political Fake News[3] y Spanish Political News Dataset[4]. Estos se combinaron con noticias políticas provenientes de distintos periódicos ecuatorianos. Las noticias recopiladas abarcan el periodo 2018 - 2024 **(véase la Sección 6.1 para los resultados de la Tarea 1).**

[3] Dataset (Spanish Political Fake News).
[4] Dataset (Spanish Political News Dataset).

En la Figura 21 se muestran los principales medios de comunicación empleados para la recolección de información: El comercio[5], La Hora[6], El Telégrafo[7], Diario Crónica[8], Diario Extra[9], La República EC[10].

Figura 21. Periódicos utilizados para extraer noticias de política ecuatoriana.

Para la extracción de datos, se empleó la técnica de web scraping, garantizando el acceso libre a la información y evitando el tratamiento de datos sensibles. De este modo, el dataset resultante se utiliza para fines educativos.

En la Figura 22 se muestra el código utilizado para el proceso de extracción, cuyas salidas se almacenaron en formato (CSV).

[5] El Comercio
[6] La Hora
[7] El Telégrafo
[8] Diario Crónica
[9] Diario Extra
[10] La República EC

```python
results = []
output_file = 'elcomercio_1.csv'
post_id = 57232
with open(output_file, mode='w', newline='', encoding='utf-8') as csv_file:
    csv_writer = csv.writer(csv_file)

    csv_writer.writerow(['ID', 'Label', 'Titulo', 'Descripcion', 'Fecha'])

    for page in range(1, 600):
        response = requests.get(base_url + str(page))
        if response.status_code == 200:
            data = response.json()
            for post in data:
                date_formatted = post["date"][:10]
                if date_formatted:
                    date_formatted = date_formatted[8:10] + "/" + date_formatted[5:7] + "/" + date_formatted[:4]
                title = post["title"]["rendered"]
                description = post.get("excerpt", {}).get("rendered", "")
                description_formatted=description.replace('<p>', '').replace('</p>', '')

                csv_writer.writerow([post_id, 1, title, description_formatted, date_formatted])

                formatted_post = {
                    "id": post_id,
                    "date": date_formatted,
                    "title": title,
                    "description": description_formatted
                }
                results.append(formatted_post)

                post_id += 1

data_elcomercio = pd.read_csv(output_file)

data_elcomercio
```

Figura 22. Código para extracción de noticias mediante web scraping.

Se integraron y fusionaron todos los datasets para obtener una base más sólida y con una cantidad suficiente de información.

En la Figura 23 se muestra el código para la integración de los datasets.

```python
import pandas as pd

dataframes = [data_fakenews, data_elcomercio, data_lahora, data_eltelegrafo, data_cronica, data_extra, data_republica1, data_republica3]

for i, df in enumerate(dataframes):
    dataframes[i]['ID'] = df['ID'].astype(str)
    dataframes[i]['Label'] = df['Label'].astype(int)

dataset_completo = pd.concat(dataframes, ignore_index=True, sort=False)

dataset_completo

output_file = 'dataset_completo.csv'
dataset_completo.to_csv(output_file, index=False, encoding='utf-8')

print(f"El dataset ha sido guardado en {output_file}")
```

Figura 23. Código para concatenar los datasets.

Tarea 2: Limpieza de datos

Con el propósito de asegurar la calidad del contenido en las noticias de política en español, tanto falsas como verdaderas, se aplicaron criterios de inclusión y exclusión **(véase los resultados de la Tarea 2 en la Sección 6.1).**

Criterios de inclusión:

- Noticias de política en español.
- Texto de buena calidad.
- Noticias falsas y verdaderas de política en español.
- Noticias del periodo 2018 - 2024.

Criterios de exclusión:

- Registros duplicados.
- Noticias no relacionadas con política.
- Registros con valores faltantes.

En la Figura 24 se evidenció la presencia de duplicados, los cuales se eliminaron automáticamente mediante la función duplicated de la biblioteca pandas.

```
# Verificar valores duplicados
duplicated_rows = data.duplicated()
print(f"Número de filas duplicadas: {duplicated_rows.sum()}")

if duplicated_rows.sum() > 0:
    print("Filas duplicadas:")
    print(data[duplicated_rows])
```

Figura 24. Código para verificar registros duplicados.

Posteriormente, en la Figura 25 se detalla el proceso para detectar y suprimir registros con valores faltantes empleando la función isnull informando el número de datos afectados por columna y eliminándolos según los criterios establecidos.

Durante la limpieza, se filtraron variables con información irrelevante (por ejemplo, “Sin descripción”, “Sin título”, “Sin fecha”), tal como se muestra en la Figura 26.

```
# Verificar valores faltantes
missing_values = data.isnull().sum()
print("Valores faltantes por columna:")
print(missing_values)

rows_with_missing = data.isnull().any(axis=1).sum()
print(f"Número de filas con valores faltantes: {rows_with_missing}")

data = data.dropna()
```

Figura 25. Verificar y eliminar valores de registro faltantes.

```
data = data[~data['Descripcion'].isin(['Sin descripción']) &
            ~data['Titulo'].isin(['Sin título']) &
            ~data['Fecha'].isin(['Sin fecha'])]

data
```

Figura 26. Código para limpiar registros sin data relevante.

Además, se convirtió la variable “Fecha” de tipo String a Datetime para aplicar el criterio de inclusión que abarcaba noticias entre 2018 y 2024, proceso ilustrado en la Figura 27.

Se eliminó código HTML incrustado en algunos textos de las noticias ecuatorianas utilizando la biblioteca Beautiful Soup. Este procedimiento, presentado en la Figura 28, se aplicó a las variables “Descripcion” y “Titulo”, garantizando así información más limpia y coherente.

```
# Convertir la columna 'Fecha' a formato datetime
data['Fecha'] = pd.to_datetime(data['Fecha'], format='%d/%m/%Y', errors='coerce')

data = data[data['Fecha'].dt.year >= 2019]

data
```

Figura 27. Código para aplicar criterio de inclusión.

```python
from bs4 import BeautifulSoup

def limpiar_html(texto_html):
    soup = BeautifulSoup(texto_html, 'html.parser')
    texto_limpio = soup.get_text()
    return texto_limpio

data['Descripcion'] = data['Descripcion'].apply(limpiar_html)
data['Titulo'] = data['Titulo'].apply(limpiar_html)

data
```

Figura 28. Código para limpiar código HTML del dataset.

Como último paso, se eliminaron símbolos y signos que dificultaban la lectura de las noticias. Dichos caracteres se reemplazaron por espacios en blanco, asegurando una interpretación adecuada del texto por parte del modelo. La Figura 29 muestra el código empleado para esta tarea. Una vez concluida la limpieza, el dataset quedó listo para las fases posteriores.

```python
data['Titulo'] = data['Titulo'].str.replace('»', '', regex=False).str.replace('«', '', regex=False)
data['Descripcion'] = data['Descripcion'].str.replace('»', '', regex=False).str.replace('«', '', regex=False)
data['Descripcion'] = data['Descripcion'].str.replace('[…]', '', regex=False)

data
```

Figura 29. Código para eliminar símbolos del dataset.

Tarea 3: Equilibrio de clases

Se inspeccionó la distribución de las clases y se detectó la necesidad de balancearlas. Se aplicó submuestreo a la clase mayoritaria, reduciendo su tamaño al eliminar aleatoriamente algunas instancias, sin afectar de forma significativa la representatividad. Esta técnica se implementó sobre la variable “Label”, donde 0 indicaba noticias falsas y 1 verdaderas, siendo la clase con valor 1 la mayoritaria **(véase los resultados de la Tarea 3 en la Sección 6.1).**

La Figura 30 presenta el proceso de submuestreo realizado en Python.

```python
from sklearn.utils import resample

print("Distribución de clases antes del submuestreo:")
print(data['Label'].value_counts())

mayoritaria = data[data['Label'] == 1]
minoritaria = data[data['Label'] == 0]

mayoritaria_submuestreada = resample(mayoritaria,
                                     replace=False,
                                     n_samples=len(minoritaria),
                                     random_state=42)

data_balanceada = pd.concat([mayoritaria_submuestreada, minoritaria])

data_balanceada = data_balanceada.sample(frac=1, random_state=42).reset_index(drop=True)

print("Distribución de clases después del submuestreo:")
print(data_balanceada['Label'].value_counts())

data = data_balanceada
```

Figura 30. Código para submuestreo de la clase Label.

Tarea 4: Aumento de datos

En cuanto al aumento de datos (data augmentation), inicialmente se consideró la técnica de Back Translation, que conserva el significado y contexto de las noticias mediante la traducción del texto a otro idioma y su posterior re-traducción al original, generando variaciones gramaticales sin alterar el sentido. Se planteó utilizar la API de Google (googletrans) para las traducciones en Python.

No obstante, esta técnica no se implementó por las siguientes limitaciones. El API de Google presentaba restricciones en el volumen de peticiones gratuitas, lo que dificultaba la ampliación del dataset sin incurrir en costos adicionales. Además, podía introducir errores de traducción que comprometieran la calidad de los datos, un aspecto crítico al requerir precisión. Finalmente, la falta de recursos computacionales para el procesamiento intensivo de traducciones también influyó en la decisión de omitir esta técnica.

Tarea 5: Normalización de datos

Durante la normalización, se llevaron a cabo las tareas de conversión del texto a minúsculas, eliminación de paréntesis, eliminación de caracteres especiales y números, eliminación de stop words para estandarizar el texto de las noticias y facilitar su análisis. Con el apoyo de la biblioteca Natural Language Toolkit (NLTK), se convirtió el texto a minúsculas y se eliminaron caracteres especiales y números, al no agregar valor al análisis. Además, se aplicó la tokenización para dividir el texto en palabras o tokens individuales **(véase los resultados de la Tarea 5 en la Sección 6.1).**

En la Figura 31 se representa el código empleado para este proceso, lo que permitió una mejor representación del texto ante el modelo de machine learning.

```python
import pandas as pd
import nltk
import spacy
import re
from nltk.corpus import stopwords

nltk.download('stopwords')

# Cargar el modelo en español
nlp = spacy.load('es_core_news_sm')

def normalizar_texto(texto):
    # Convertir a minúsculas
    texto = texto.lower()

    # Eliminar todos los paréntesis
    texto = re.sub(r'[()]', '', texto)

    # Eliminar caracteres especiales y números
    texto = re.sub(r'[^a-záéíóúüñ\s]', '', texto)

    # Tokenización
    tokens = nltk.word_tokenize(texto)

    # Eliminación de stop words
    spanish_stopwords = set(stopwords.words('spanish'))
    tokens = [word for word in tokens if word not in spanish_stopwords]
    #print(f"Tokens sin stopwords: {tokens}")

    # Unir tokens en una sola cadena
    texto_normalizado = ' '.join(tokens)

    return texto_normalizado

df = pd.DataFrame(data)

df['Titulo'] = df['Titulo'].apply(normalizar_texto)
df['Descripcion'] = df['Descripcion'].apply(normalizar_texto)

data
```

Figura 31. Normalización de los datos.

Tarea 6: División de datos.

Una vez finalizada la normalización, se procedió a la división del conjunto de datos con el fin de entrenar el modelo de regresión logística sin optimización y con algoritmos de optimización (GD, SGD, MBGD, AdaGrad, Adam, RMSProp) **(véase los resultados de la Tarea 6 en la Sección 6.1).**

Modelo sin optimización:

Se asignó el 70% de los datos para entrenamiento, 15% para validación y 15% para evaluación, siguiendo los criterios fueron establecidos mediante lo aprendido en la materia de data mining, la cantidad del dataset, y los trabajos relacionados. El proceso se realizó mediante la mezcla y división de un dataset en tres subconjuntos: entrenamiento, validación y prueba. Primero, el dataset completo se carga, se realiza un muestreo aleatorio completo para mezclar las filas, utilizando random_state con el objetivo de mantener la reproducibilidad

y, tras la mezcla, se reinician los índices, se definen los tamaños relativos de cada subconjunto. Finalmente, cada uno de estos subconjuntos se guarda en archivos CSV separados.

Se aplicó el mismo criterio de división al entrenamiento con optimización, asegurando así una comparación justa. Esta estrategia permitió atribuir las variaciones en la precisión exclusivamente a los procesos de optimización, en lugar de a la estructura de los datos.

En la Figura 32 se muestra el código para la división de datos para el modelo con y sin optimización.

```
# Cargar el dataset
dataset = pd.read_csv("dataset_preOptimization.csv", sep=";")

# Mezclar datos
shuffled_dataset = dataset.sample(frac=1, random_state=42).reset_index(drop=True)

# Divisiones
train_size = 0.7
val_test_size = 0.15

# Calcular índices para la división
train_end = int(len(shuffled_dataset) * train_size)
val_end = train_end + int(len(shuffled_dataset) * val_test_size)

# Dividir el dataset en conjuntos de entrenamiento, validación y prueba
train_data = shuffled_dataset[:train_end]
val_data = shuffled_dataset[train_end:val_end]
test_data = shuffled_dataset[val_end:]

# Guardar en archivos CSV
train_data.to_csv("train_data.csv", index=False, sep=";")
val_data.to_csv("val_data.csv", index=False, sep=";")
test_data.to_csv("test_data.csv", index=False, sep=";")

print("Conjuntos de datos guardados como CSV:")
print(" - train_data.csv")
print(" - val_data.csv")
print(" - test_data.csv")
```

Figura 32. Código de la división de datos en el proceso de optimización.

Fase: Ingeniería de Modelos.

Tarea 7: Ajuste de hiperparámetros.

Se implementó el ajuste de hiperparámetros para determinar los valores óptimos que minimizaran la función de costo. Durante este proceso, los parámetros se actualizaron en la dirección opuesta al gradiente, controlando la magnitud del ajuste mediante la tasa de aprendizaje. Además, la regularización se implementó en algunos optimizadores aplicada en los gradientes durante las actualizaciones de los parámetros, ayudando a prevenir el sobreajuste y la estabilización del entrenamiento del modelo. A continuación, se presentan las configuraciones de cada algoritmo de optimización (GD, SGD, MBGD, AdaGrad, Adam y RMSProp) **(véase los resultados de la Tarea 7 en la Sección 6.1).**

Gradiente descendente (GD):

Para este algoritmo, se definieron los parámetros alpha (tasa de aprendizaje) y num_iters (iteraciones) con un enfoque experimental. Se ajustaron hasta encontrar valores

que permitieran la convergencia del modelo, reduciendo gradualmente la variación en el costo de la función.

Gradiente descendente estocástico (SGD):

En el SGD se configuraron parámetros clave que controlaron la frecuencia, rapidez y estilo del aprendizaje, así como la regularización:

- **Alpha:** Tasa de aprendizaje que controlar el tamaño de los pasos de actualización.
- **Num_epochs:** Número de ciclos completos en los que el modelo recorre una vez por todos los datos.
- **Batch_size:** Los mini lotes dividen los datos en pequeños subconjuntos para calcular el gradiente por lote en lugar de hacerlo por muestra individual.
- **Lambda:** Parámetro de regularización que contribuyó a prevenir el sobreajuste.

Gradiente descendente por mini lotes (MBGD):

En el gradiente descendente por mini lotes (MBGD), se establecieron alpha, num_iters y batch_size para asegurar una convergencia adecuada. Estos valores se ajustaron mediante ensayo y error, controlando la rapidez y la frecuencia del aprendizaje, así como el tamaño de los mini lotes, con el fin de optimizar el rendimiento del modelo.

Algoritmo de gradiente adaptativo (AdaGrad):

En AdaGrad, la tasa de aprendizaje se ajustó de forma adaptativa para cada parámetro. Se ejecutaron *n* actualizaciones de los parámetros, acumulando el cuadrado de gradientes previos. Esta estrategia facilitó la convergencia en el mínimo de la función de costo al considerar el historial de actualizaciones.

Estimación de momento adaptativo (Adam):

Se definieron dos parámetros clave para optimizar algoritmo Adam, utilizado para ajustar los parámetros del modelo de regresión logística y minimizar su función de costo. Este enfoque facilitó un entrenamiento más efectivo, reduciendo la necesidad de ajustar manualmente la tasa de aprendizaje para cada problema.

- **Alpha (tasa de aprendizaje):** Representó la tasa de aprendizaje, un factor de escalado que determinó el tamaño de los pasos en la dirección del gradiente para actualizar los parámetros theta.
- **Num_iters:** Indicó la cantidad de iteraciones o pasos totales que el algoritmo ejecutaría para actualizar los parámetros theta.

Propagación de la raíz del promedio cuadrático (RMSProp):

Este algoritmo ajusta la tasa de aprendizaje de forma dinámica, adaptándola a cada parámetro del modelo según el historial reciente de gradientes. Para ello, se definen y ajustan los siguientes parámetros:

- **Alpha (tasa de aprendizaje):** Controla el tamaño de los pasos en la dirección del gradiente. Un valor muy alto puede provocar inestabilidad en la convergencia, mientras que uno demasiado bajo ralentiza el proceso de aprendizaje.
- **Rho (factor de decaimiento):** Determina la proporción en la que los gradientes previos influyen en la actualización de los parámetros. Un valor cercano a 1 otorga mayor peso a la historia más lejana, mientras que valores menores destacan gradientes recientes.
- **Epsilon (valor pequeño):** Evita divisiones por cero al momento de normalizar los gradientes acumulados, contribuyendo a la estabilidad numérica del algoritmo.
- **Num_iters (iteraciones):** Indica el número total de pasos que el algoritmo realiza para actualizar los parámetros del modelo. Este valor se fija con base en el tamaño del conjunto de datos, la complejidad del problema y la disponibilidad de recursos computacionales.

Estos parámetros se ajustaron mediante prueba y error, buscando un balance entre estabilidad y rapidez de convergencia. Con RMSProp, el modelo actualiza los parámetros teniendo en cuenta no solo la dirección y magnitud del gradiente en cada iteración, sino también el historial de sus valores, lo que facilita converger hacia el mínimo de la función de costo

Consideraciones:

- Un valor de *alpha* excesivamente alto podía hacer que el algoritmo saltara sobre el mínimo en lugar de acercarse gradualmente, provocando una convergencia inestable o incluso la falta de convergencia.
- Un valor de *alpha* demasiado bajo podría ralentizar el aprendizaje, incrementando el tiempo necesario para alcanzar el mínimo de la función de costo.

Tarea 8: Selección del modelo.

Se seleccionó el modelo de regresión logística por sus condiciones favorables en la clasificación de texto, especialmente en el análisis de noticias de política en español. Aunque poseía un alto potencial para detectar información falsa, su aplicación en este ámbito no se había optimizado suficientemente. La limitada experimentación con algoritmos de optimización como Gradiente Descendente (GD), Gradiente Descendente Estocástico (SGD), Gradiente Descendente por Mini-Lotes (MBGD), Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam) y Propagación de la Raíz del Promedio Cuadrático (RMSProp) dificultaba precisar su impacto en la precisión del clasificador **(véase los resultados de la Tarea 8 en la Sección 6.1).**

Tarea 9: Selección de Optimización.

Gradiente descendente (GD):

Se implementó el algoritmo de Gradiente Descendente (GD) con el objetivo de minimizar la función de costo y ajustar los parámetros del modelo.

En el método se entrenaron los parámetros a lo largo de múltiples épocas. En cada iteración, se calculó la predicción sobre los datos de entrenamiento (empleando los valores de *x*, los pesos y el sesgo), y se determinó la pérdida utilizando la función de entropía cruzada. Posteriormente, se evaluó la precisión del modelo comparando las predicciones binarias con las etiquetas verdaderas. A continuación, se calcularon los gradientes de los pesos y del sesgo, actualizando dichos parámetros mediante descenso de gradiente con momento y una tasa de aprendizaje que disminuyó gradualmente con el paso de las épocas. Durante este proceso, se registraron los valores de pérdida y precisión en listas para su análisis posterior **(véase los resultados de la Tarea 9 en la Sección 6.1).**

En la Figura 33 se muestra la función utilizada para aplicar este procedimiento en el entrenamiento de la regresión logística.

```
def fit(self, X_train, y_train, X_val, y_val):
    n_samples, n_features = X_train.shape
    self.weights = np.zeros(n_features)
    self.bias = 0
    self.velocity_weights = np.zeros(n_features)

    for epoch in range(self.epochs):
        # Predicciones en entrenamiento
        linear_model_train = np.dot(X_train, self.weights) + self.bias
        y_pred_train = self.sigmoid(linear_model_train)

        # Pérdida en entrenamiento
        loss_train = -np.mean(y_train * np.log(y_pred_train + 1e-9) +
                              (1 - y_train) * np.log(1 - y_pred_train + 1e-9))
        self.losses_train.append(loss_train)

        # Precisión en entrenamiento
        y_pred_train_class = [1 if i > 0.5 else 0 for i in y_pred_train]
        accuracy_train = np.mean(y_pred_train_class == y_train)
        self.accuracies_train.append(accuracy_train)

        # Gradientes
        dw = (1 / n_samples) * np.dot(X_train.T, (y_pred_train - y_train))
        db = (1 / n_samples) * np.sum(y_pred_train - y_train)

        # Actualización con momento
        self.velocity_weights = self.momentum * self.velocity_weights - self.learning_rate * dw
        self.velocity_bias = self.momentum * self.velocity_bias - self.learning_rate * db
        self.weights += self.velocity_weights
        self.bias += self.velocity_bias

        # Decaída del learning rate
        self.learning_rate *= (1 / (1 + self.decay_rate * epoch))
```

Figura 33. Función del algoritmo de optimización regresión logística, con parámetros del modelo a lo largo de múltiples épocas.

Gradiente descendente estocástico (SGD):

En el optimizador SGD, la función recibía el modelo, una tasa de aprendizaje y el valor del momento. Además, se recorría cada parámetro y su correspondiente velocidad. Si el parámetro poseía un gradiente calculado, la velocidad se actualizaba incorporando el gradiente y el factor de momento, para luego sumarse al parámetro y ejecutar así un paso de

descenso de gradiente con momento. Durante el bucle principal, el modelo entraba en modo entrenamiento, se calculaban las salidas, la pérdida y se realizaba la retropropagación. Finalmente, se llamaba a la función principal para actualizar los parámetros con las nuevas velocidades, implementando un descenso de gradiente con momento más estable y rápido que el gradiente descendente simple.

En la Figura 34 se describe la función del algoritmo de optimización gradiente descendente estocástico, utilizado para optimizar el modelo.

```
def mbgd_momentum_step(model, velocities, lr=0.005, momentum=0.9, wd=1e-4):
    with torch.no_grad():
        for param, v in zip(model.parameters(), velocities):
            if param.grad is not None:
                param.grad.add_(param, alpha=wd)  # L2 regularization
                v.mul_(momentum).add_(param.grad, alpha=-lr)
                param.add_(v)

velocities = [torch.zeros_like(param, device=device) for param in model.parameters()]

n_samples = X_train_tensor.shape[0]

# Entrenar el número total de épocas, sin early stopping
for epoch in range(epochs):
    indices = torch.randperm(n_samples, device=device)
    X_train_shuffled = X_train_tensor[indices]
    y_train_shuffled = y_train_tensor[indices]

    epoch_loss = 0.0
    model.train()

    for i in range(0, n_samples, batch_size):
        X_batch = X_train_shuffled[i:i+batch_size]
        y_batch = y_train_shuffled[i:i+batch_size]

        model.zero_grad()
        outputs = model(X_batch).squeeze()
        loss = criterion(outputs, y_batch)
        loss.backward()
        mbgd_momentum_step(model, velocities, lr=learning_rate, momentum=momentum, wd=weight_decay)
        epoch_loss += loss.item()
```

Figura 34. Función del algoritmo de optimización gradiente descendente estocástico (SGD).

Gradiente descendente por mini lotes (MBGD):

Se implementó el algoritmo gradiente descendente por mini lotes (MBGD) con momento y regularización L2 para ajustar los parámetros del modelo. El modelo recibía una lista de velocidades, la tasa de aprendizaje, el factor de momento y un valor de decaimiento de peso para la regularización L2. Por cada parámetro con gradientes se añadía el término de regularización L2 al gradiente y luego se utilizaba este gradiente ajustado, junto con la velocidad, para actualizar cada parámetro con momento y descenso de gradiente.

Antes del bucle principal de entrenamiento, las velocidades se inicializaban en cero. Durante la ejecución, se creaban índices aleatorios para barajar los datos de entrada y las etiquetas, formándose mini-lotes que se procesaban uno a uno. En cada mini-lote se realizaba el forward pass, el cálculo de la pérdida, la retropropagación de gradientes y, finalmente, se llamaba a la función de gradiente para ajustar los parámetros. De este modo, el entrenamiento combinaba mini-lotes, momento y regularización L2, optimizando el modelo a lo largo de varias épocas.

En la Figura 35 se describe la función del algoritmo de optimización gradiente descendente por mini lotes.

```python
def mbgd_momentum_step(model, velocities, lr=0.005, momentum=0.9, wd=1e-4):
    with torch.no_grad():
        for param, v in zip(model.parameters(), velocities):
            if param.grad is not None:
                param.grad.add_(param, alpha=wd)  # L2 regularization
                v.mul_(momentum).add_(param.grad, alpha=-lr)
                param.add_(v)

velocities = [torch.zeros_like(param, device=device) for param in model.parameters()]

n_samples = X_train_tensor.shape[0]

# Entrenar el número total de épocas, sin early stopping
for epoch in range(epochs):
    indices = torch.randperm(n_samples, device=device)
    X_train_shuffled = X_train_tensor[indices]
    y_train_shuffled = y_train_tensor[indices]

    epoch_loss = 0.0
    model.train()

    for i in range(0, n_samples, batch_size):
        X_batch = X_train_shuffled[i:i+batch_size]
        y_batch = y_train_shuffled[i:i+batch_size]

        model.zero_grad()
        outputs = model(X_batch).squeeze()
        loss = criterion(outputs, y_batch)
        loss.backward()
        mbgd_momentum_step(model, velocities, lr=learning_rate, momentum=momentum, wd=weight_decay)
        epoch_loss += loss.item()
```

Figura 35. Esquema de entrenamiento utilizando gradiente descendente por mini lotes con momento y regularización L2.

Algoritmo de gradiente adaptativo (AdaGrad):

El optimizador AdaGrad recibía los parámetros del modelo, una tasa de aprendizaje, un decaimiento de peso para L2 y un valor ϵ *(épsilon)* para evitar divisiones por cero. Al inicializar el optimizador, se creaba un diccionario de acumuladores para almacenar la suma de los cuadrados de gradientes pasados de cada parámetro, base del funcionamiento de Adagrad para ajustar la tasa de aprendizaje según el historial de gradientes.

En el método step, por cada parámetro con gradiente, se aplicaba la regularización L2, se actualizaba el acumulador con el cuadrado del gradiente actual y se calculaba el denominador para la actualización, dividiendo por la raíz del acumulador más ϵ *(épsilon)*. Finalmente, se actualizaba el valor del parámetro teniendo en cuenta su historial de gradientes. El método zero_grad establecía en cero los gradientes de cada parámetro antes del siguiente paso de entrenamiento.

En la Figura 36 se representa la función del algoritmo de gradiente adaptativo,

```python
class ManualAdagrad:
    def __init__(self, model_parameters, lr=1e-4, weight_decay=1e-2, eps=1e-8):
        self.params = list(model_parameters)  # lista de parámetros del modelo
        self.lr = lr
        self.weight_decay = weight_decay
        self.eps = eps

        # Diccionario para almacenar la suma de cuadrados de gradientes de cada parámetro
        self.accumulators = {}
        for p in self.params:
            # Igual shape que p.data, inicializado a cero
            self.accumulators[p] = torch.zeros_like(p.data, device=p.data.device)

    def step(self):
        """Actualiza los parámetros según la regla de Adagrad."""
        for p in self.params:
            if p.grad is None:
                continue

            # Aplicar regularización L2 directamente al gradiente
            p.grad.data.add_(p.data, alpha=self.weight_decay)

            # Sumar el cuadrado del gradiente al acumulador
            self.accumulators[p] += p.grad.data ** 2

            # Denominador para la actualización
            denom = torch.sqrt(self.accumulators[p]) + self.eps

            # Regla de Adagrad
            p.data = p.data - self.lr * (p.grad.data / denom)

    def zero_grad(self):
        """Pone a cero los gradientes."""
        for p in self.params:
            if p.grad is not None:
                p.grad.zero_()
```

Figura 36. Código de la función de AdaGrad.

Estimación de momento adaptativo (Adam):

El optimizador Adam, se inicializaba con una lista de parámetros del modelo, una tasa de aprendizaje, coeficientes beta para el cálculo de momentos (por defecto 0.9 y 0.999) y un valor pequeño ϵ *(épsilon)* para evitar divisiones por cero.

En el constructor se almacenaban los parámetros y se creaban dos listas, m y v, con la misma forma que los parámetros del modelo, destinadas a guardar las estimaciones del primer y segundo momento del gradiente. También se definía un contador de pasos (self.t), incrementado en cada actualización, necesario para corregir el sesgo en los momentos.

El método zero_grad reiniciaba los gradientes a cero antes de cada actualización. En step, por cada parámetro con gradiente, se actualizaban las estimaciones de primer y segundo momento, se corregían los valores de dichos momentos con base en el número de pasos (self.t) y finalmente se actualizaba el parámetro. Así, se aplicaba el optimizador Adam, que suele presentar una convergencia más rápida y estable que el gradiente descendente estándar.

En la Figura 37 se presenta la función del algoritmo de estimación de momento adaptativo, conocido como Adam, utilizado para la optimización eficiente de parámetros en modelo de regresión logística.

```python
class ManualAdam:
    def __init__(self, params, lr=1e-3, betas=(0.9, 0.999), eps=1e-8):
        self.params = list(params)  # parámetros del modelo
        self.lr = lr
        self.betas = betas
        self.eps = eps

        # Listas para almacenar las estimaciones de primer y segundo momento
        self.m = [torch.zeros_like(p.data) for p in self.params]
        self.v = [torch.zeros_like(p.data) for p in self.params]

        # Contador de pasos (epochs * iteraciones)
        self.t = 0

    def zero_grad(self):
        """
        Pone en cero todos los gradientes de los parámetros.
        """
        for p in self.params:
            if p.grad is not None:
                p.grad.zero_()

    def step(self):
        """
        Actualiza cada parámetro según la regla de ADAM.
        """
        self.t += 1
        b1, b2 = self.betas

        for i, param in enumerate(self.params):
            if param.grad is None:
                continue

            g = param.grad.data  # gradiente

            # Actualizar estimaciones de primer y segundo momento
            self.m[i] = b1 * self.m[i] + (1 - b1) * g
            self.v[i] = b2 * self.v[i] + (1 - b2) * (g * g)

            # Corrección por sesgo
            m_hat = self.m[i] / (1 - b1 ** self.t)
            v_hat = self.v[i] / (1 - b2 ** self.t)

            # Actualización del parámetro
            param.data = param.data - self.lr * (m_hat / (torch.sqrt(v_hat) + self.eps))
```

Figura 37. Código con la implementación manual del optimizador basado en el algoritmo Adagrad utilizando PyTorch.

Propagación de la raíz del promedio cuadrático (RMSProp):

La función RMSProp del optimizador recibe como parámetros la lista de parámetros del modelo, la tasa de aprendizaje, un factor de decaimiento para el promedio móvil exponencial de las magnitudes de los gradientes, y un valor ϵ *(épsilon)*, para evitar divisiones por cero.

En el constructor de la clase, se inicia un diccionario self.state para cada parámetro, donde se almacenará el promedio móvil de los gradientes al cuadrado. El método zero_grad pone a cero los gradientes de todos los parámetros antes de realizar la siguiente actualización.

El método step recorre cada parámetro que cuente con un gradiente calculado. Primero, se actualiza el promedio móvil exponencial, multiplicando el valor almacenado por alpha y añadiendo el cuadrado del gradiente actual ponderado por (1 - alpha). A continuación, se calcula el denominador tomando la raíz cuadrada del promedio móvil más ϵ *(épsilon)*, garantizando la estabilidad numérica.

Finalmente, cada parámetro se actualiza restando el gradiente escalado por la inversa de dicho denominador y la tasa de aprendizaje. Esta estrategia permite que la magnitud del paso de actualización se adapte dinámicamente a la magnitud promedio de los gradientes, promoviendo un entrenamiento más estable y una convergencia más suave. En la Figura 38 se muestra el código de la función del método de optimización en detalle.

```python
class ManualRMSProp:
    """
    Implementación manual de RMSProp sin usar librerías de optimización.
    """
    def __init__(self, params, lr=1e-4, alpha=0.99, eps=1e-8):
        self.params = list(params)
        self.lr = lr
        self.alpha = alpha
        self.eps = eps
        self.state = {}
        for p in self.params:
            # Almacena el promedio móvil de los gradientes al cuadrado
            self.state[p] = torch.zeros_like(p, memory_format=torch.preserve_format)

    def zero_grad(self):
        for p in self.params:
            if p.grad is not None:
                p.grad.zero_()

    def step(self):
        for p in self.params:
            if p.grad is None:
                continue
            grad = p.grad.data
            # Actualiza el promedio móvil de gradientes^2
            self.state[p].mul_(self.alpha).addcmul_(grad, grad, value=1 - self.alpha)
            # Actualiza el parámetro
            denom = self.state[p].sqrt().add_(self.eps)
            p.data.addcdiv_(grad, denom, value=-self.lr)

# Usar nuestro optimizador manual RMSProp en lugar de optim.RMSprop
optimizer = ManualRMSProp(model.parameters(), lr=0.00005, alpha=0.99, eps=1e-8)
```

Figura 38. Código de la implementación manual del optimizador RMSProp en PyTorch, sin emplear librerías de optimización predefinida.

Tarea 10: Entrenamiento del modelo.

Se emplearon Google Colab y el notebook de Jupyter, disponible en Anaconda, para entrenar el modelo. Sin embargo, se optó por continuar el trabajo exclusivamente en Jupyter de Anaconda debido a problemas surgidos en Google Colab, donde el kernel fallaba y la página dejaba de responder. Se sospechó que el elevado número de épocas pudo haber provocado dichos inconvenientes **(véase los resultados de la Tarea 10 en la Sección 6.1).**

Se llevaron a cabo dos entrenamientos: uno con optimización y otro sin ella, con el propósito de comparar ambos modelos. En ambos casos, se utilizaron distintos parámetros sobre el mismo conjunto de datos. A continuación, se presentan los códigos empleados en este proceso **(véase los resultados de la Tarea 10 en la Sección 6.1).**

Sin optimización:

En la Figura 39 se muestra el código utilizado para realizar el entrenamiento, empleando la biblioteca *scikit-learn*. Luego se entrenó con los datos de entrenamiento.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_vec, y_train)
```

Figura 39. Código para entrenar el modelo regresión logística.

Con optimización:

En la optimización con gradiente descendente y sus variaciones la regresión logística utiliza la función sigmoide para modelar la probabilidad de pertenecía a una clase dada. Mientras que el ajuste de los parámetros (pesos) se realizó con gradiente descendente basándose en minimizar la función de costo de la regresión logística.

El modelo regresión logística de PyTorch crea una capa lineal que transforma el espacio de entrada, en una salida unidimensional, lo cual es típico en la regresión logística binaria. En el método forward, el tensor de entrada x se procesa a través de la capa lineal, que combina linealmente las características utilizando pesos y un sesgo aprendibles. Posteriormente, se aplica la función sigmoide que mapea la salida a un valor entre 0 y 1, interpretado como una probabilidad. De esta forma, el parámetro clave es input_dim, el cual determina el tamaño de la capa lineal y, por ende, de los pesos y el sesgo internos del modelo.

El código de la función sigmoide y el modelo de regresión logística definido mediante PyTorch la cual se representa en la Figura 40.

```
def sigmoid(self, z):
    return 1 / (1 + np.exp(-z))


class LogisticRegressionTorch(nn.Module):
    def __init__(self, input_dim):
        super(LogisticRegressionTorch, self).__init__()
        self.linear = nn.Linear(input_dim, 1)

    def forward(self, x):
        return torch.sigmoid(self.linear(x))
```

Figura 40. Implementación de la función sigmoide y modelo de regresión logística definido mediante PyTorch.

5.2.2. Objetivo 2: Evaluar la mejora en la precisión del modelo en comparación con la precisión inicial sin optimización utilizando la matriz de confusión.

En este objetivo se evaluó la precisión alcanzada por los algoritmos de optimización aplicados al modelo de regresión logística, haciendo énfasis en la métrica de precisión. Siguiendo la metodología CRISP-ML y mediante la matriz de confusión, se preparó la evaluación con el dataset de noticias falsas resultante de la ingeniería de datos. Luego, se entrenó el modelo junto con los optimizadores de gradiente descendente y sus variantes. A

continuación, se compararon los resultados obtenidos por cada optimizador en el conjunto de evaluación, y se identificó el mejor modelo optimizado. Finalmente, se contrastó dicho modelo con la regresión logística sin optimización. A continuación, se describen los pasos seguidos para cumplir este objetivo.

Fase: Evaluación del modelo.

La evaluación se realizó aplicando métricas enfocadas en la precisión, con el fin de identificar la opción más adecuada tanto en la fase de entrenamiento (validación) como en la de evaluación, basándose en los resultados de la Tabla 13 de la Tarea 10. Dicho análisis permitió seleccionar el modelo con mejor rendimiento para pasar a la fase de pruebas, en la cual se comparó la evolución de la métrica de precisión respecto al modelo de regresión logística sin optimización **(véase los resultados en la Sección 6.2).**

Fase: Pruebas del modelo.

Para las pruebas del modelo, se utilizó la matriz de confusión y se calculó la métrica principal de precisión, considerando además las métricas de sensibilidad, especificidad, exactitud y F1 Score). Se aplicaron análisis comparativos mediante la consolidación de información a través de tablas y gráficos estadísticos. Esta evaluación permitió analizar la capacidad de los modelos optimizados para clasificar correctamente las noticias falsas y verdaderas de política en español (véase en Resultados en la fase Pruebas del modelo).

5.2.2.1. Flujo de evaluación de los modelos

Para evaluar el modelo de regresión logística junto con el mejor método de optimización determinado tras la validación, se siguió el esquema representado en la Figura 41. Además de la preparación y división de los datos, el modelo fue entrenado, se ajustaron los hiperparámetros del optimizador y se realizó la evaluación con el conjunto de pruebas, haciendo uso de la matriz de confusión. Posteriormente, dichos resultados se compararon con la línea base (regresión logística sin optimización) para constatar la efectividad de la optimización y el incremento en términos de precisión **(véase Anexo 3)**.

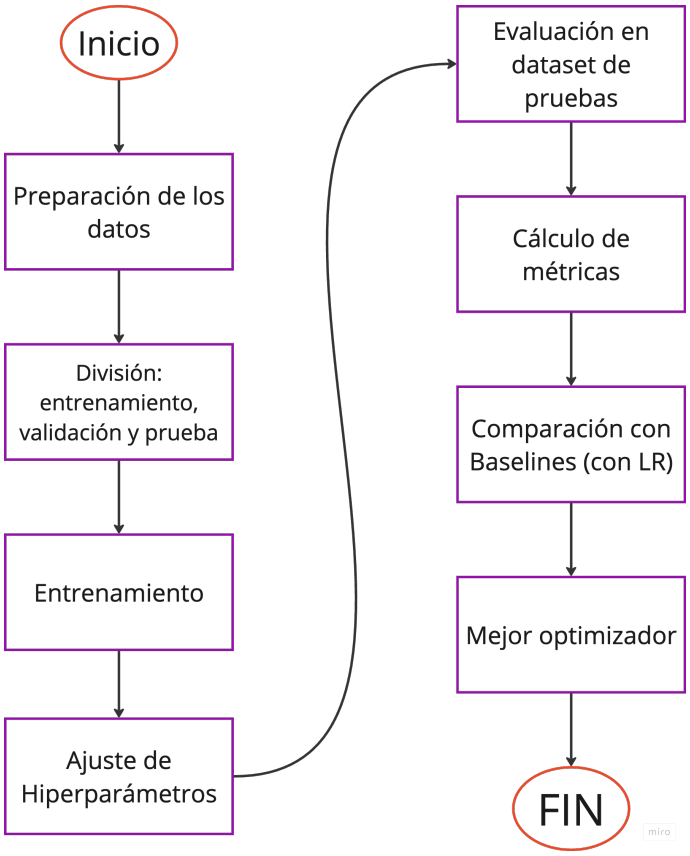


Figura 41. Diagrama de flujo del proceso durante la evaluación del modelo regresión logística junto a los optimizadores.

5.3. Recursos

A continuación, se describieron los recursos necesarios para la elaboración de este trabajo de integración curricular, clasificados en recursos científicos, técnicos y computacionales.

5.3.1. Recursos Científicos

- **Método Delphi:** Se realizó una variación de este método con la evaluación del Ing. Luis Chamba Eras, PhD., lo cual resultó fundamental para justificar el problema **(véase en Anexo 1 y Anexo 2).**
- **Método Experimental:** Se aplicó junto con la metodología CRISP-ML a lo largo de las fases y tareas relacionadas con la generación del dataset, la ingeniería de datos y el entrenamiento y validación del modelo con los algoritmos de optimización.
- **Estudio de caso:** Se implementó esta técnica para localizar y analizar trabajos relacionados, aportando información valiosa que permitió un entendimiento más profundo del objeto de estudio **(véase en Tabla 2. Trabajos relacionados.**

5.3.2. Recursos Técnicos

- **Visual Studio Code:** Se empleó como editor principal de código para la implementación de scripts en Python, enfocados en la normalización, estandarización y limpieza de datos.
- **PyTorch:** Se utilizó para aprovechar la GPU en el entrenamiento de los modelos, tanto sin optimización como con gradiente descendente y sus variantes.
- **Jupyter Anaconda:** Se recurrió a este entorno para realizar pruebas de ejecución y procesamiento computacional, como alternativa a Visual Studio Code.

5.3.3. Recursos Computacionales

Se contó con recursos de hardware específicos para el entrenamiento de los modelos, buscando optimizar el ajuste de hiperparámetros y permitir un uso intensivo de la capacidad disponible (detallados en la Tabla 3). Dichos recursos se emplearon durante todo el proceso, desde el inicio hasta la conclusión de este trabajo de integración curricular.

Tabla 3. Hardware y características relevantes para el entrenamiento del modelo de regresión logística optimizado a través de gradiente descendente y sus variantes.

Componente	Modelo	Especificaciones
Procesador (CPU)	AMD Ryzen 5 5600	6 núcleos y 12 hilos, ideal para procesamiento de datos.
Memoria RAM	16 GB DDR4 a 3200 MHz	Ideal para el manejo de datos moderados.
Tarjeta gráfica (GPU)	NVIDIA RTX 4060, 8 GB VRAM	Excelente para acelerar el tiempo de entrenamiento gracias a CUDA.
Almacenamiento	SSD de 512GB	Alta velocidad en lectura y escritura mejorando la carga de datos.
Sistema Operativo	Windows 10	Compatible con frameworks como PyTorch y Scikit-Learn.

Unidad central de procesamiento (CPU); Memoria de acceso aleatorio (RAM); Cuarta generación de memoria (DDR4); Frecuencia de operación Megahertz (MHZ); Unidad de procesamiento Gráfico (GPU); Memoria de acceso aleatorio (VRAM); Unidad de estado sólido (SSD); Unidad de medida de almacenamiento (GB).

6. Resultados

6.1. Objetivo 1: Entrenar el modelo regresión logística con los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp mediante CRISP-ML.

Fase: Ingeniería de datos.

Tarea 1: Selección de datos (Personalización dataset)

Se llevó a cabo la fusión de tres datasets: los dos primeros contenían noticias relacionadas con la política española, mientras que el tercero fue generado mediante técnicas de web scraping con noticias de política de Ecuador. La Tabla 4 presenta los registros correspondientes a las versiones de los distintos datasets.

Tabla 4. Tamaño y la composición de cada dataset previo a su integración.

	Primer Dataset	Segundo Dataset	Tercer Dataset
Clase	Datos	Datos	Datos
1 (noticias verdaderas)	33.351	6.811	4.883
0 (noticias falsas)	23.880	9.498	3.932

Spanish Political Fake News (Primer Dataset); Spanish Political News Dataset (Segundo Dataset); Dataset generado por Web Scraping (Tercer Dataset).

El dataset de datos final quedó conformado por dos clases: la clase 1 representó noticias verdaderas, mientras que la clase 0 correspondió a las noticias falsas. La cantidad de noticias para cada clase se presenta en la Tabla 5. Asimismo, en la Figura 42 se muestra la distribución y cantidad de datos en el dataset final. La visualización muestra un pequeño predominio de las noticias verdaderas (clase 1) sobre las noticias falsas (clase 0) en la composición general del dataset.

Tabla 5. Distribución total de datos en el dataset final, compuesto por 45045 noticias verdaderas y 37310 noticias falsas.

Clase	Datos
1 (noticias verdaderas)	45.045
0 (noticias falsas)	37.310

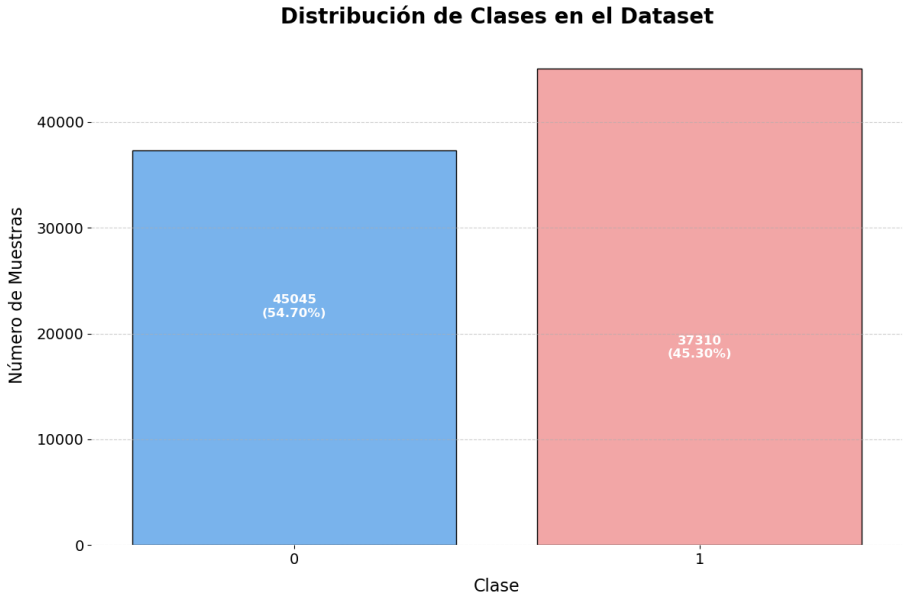


Figura 42. Distribución de clases dentro del conjunto de datos final.

Tarea 2: Limpieza de datos

El número inicial de noticias en ambas clases, verdaderas y falsas, sumaba un total de 82355 registros. Tras el proceso de limpieza de datos, que implicó la eliminación de registros vacíos, incompletos y duplicados, el total se redujo a 82041. La

Tabla 6 presenta el número de datos resultante después de aplicar la limpieza de datos.

Tabla 6. Número de datos tras aplicar técnicas de limpieza de datos.

Clase	Datos
1 (noticias verdaderas)	44903
0 (noticias falsas)	37138

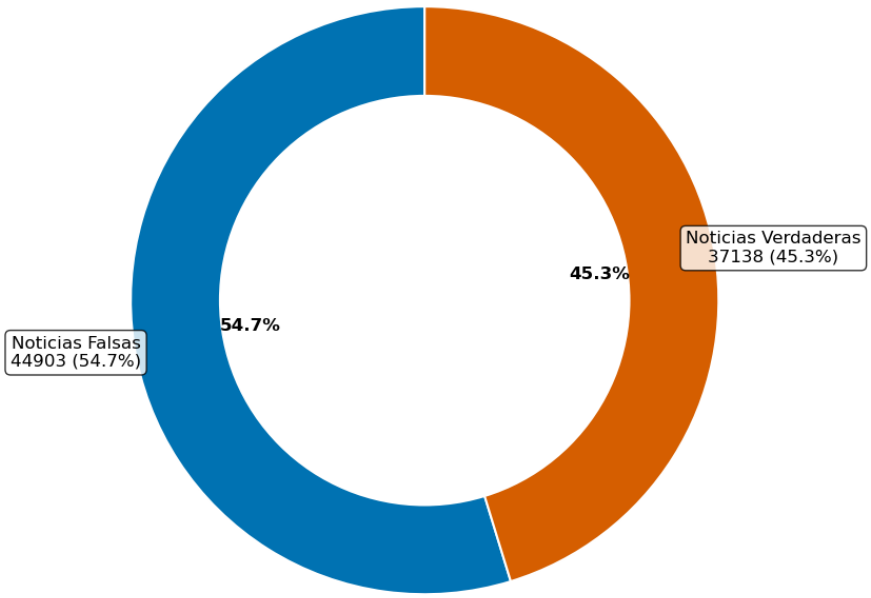


Figura 43. Distribución del atributo Label en el dataset después de aplicar la limpieza de datos.

Tarea 3: Equilibrio de clases

Se limitó el número de registros a la cantidad correspondiente a la clase con menos datos, lo que permitió un aprendizaje equitativo de las características de ambas clases. El resultado este procedimiento es que cada clase quedó con *37138* registros, sumando un total de *74276* registros utilizados para el entrenamiento, validación y la evaluación de los modelos. La Tabla 7 presenta el número de registros resultantes después de aplicar el equilibrio de clases. Además, en la Figura 44 se presenta las dos clases equilibradas y preparadas para la siguiente etapa del proceso. Conformada por 37,138 muestras (50%) son Noticias Falsas (0), y 37,138 muestras (50%) son Noticias Verdaderas (1)

Tabla 7. Número de datos resultantes después de aplicar el equilibrio de clases.

Clase	Datos
1 (noticias verdaderas)	37138
0 (noticias falsas)	37138

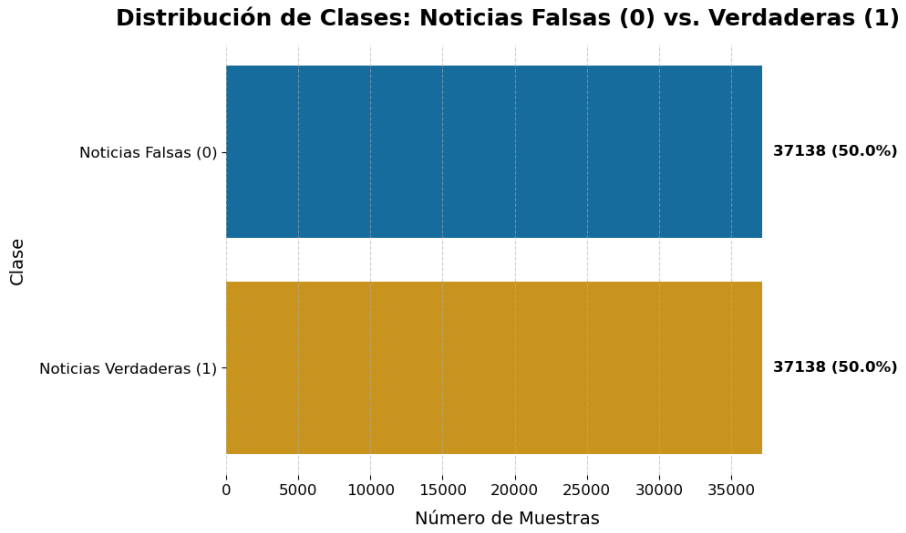


Figura 44. Distribución equilibrada de clases en un dataset de noticias.

Tarea 5: Normalización de datos

Los resultados de la normalización de datos ayudaron en el entrenamiento del modelo, puesto que proporcionó un conjunto de información coherente y comprensible que permitió identificar características y patrones relevantes para alcanzar la convergencia **(para profundizar en esta tarea véase Anexo 5).**

Convertir minúsculas a los registros

Se llevó a cabo la conversión a minúsculas de los registros del dataset. En la Tabla 8 se muestran los datos antes de la normalización, donde se aprecian mayúsculas en algunos campos; posteriormente, se exhiben los mismos registros tras la normalización, evidenciando los cambios realizados.

Tabla 8. Proceso de transformación del texto a minúscula en los registros del dataset.

Registro Original	Registro en minúsculas
Inma Nieto pierde las primarias de PNC frente	inma nieto pierde las primarias de pnc frente
Lasso irá a Lima por los 25 años del acuerdo	lasso irá a lima por los 25 años del acuerdo
Carmen Calvo, sobre la entrada de miembros de	carmen calvo, sobre la entrada de miembros de
Yolanda Díaz respalda la decisión de Colau y	yolanda díaz respalda la decisión de colau y
Los obispos advierten que juzgarán el pacto de	los obispos advierten que juzgarán el pacto de

Eliminar caracteres especiales

Se realizó la eliminación de caracteres especiales y números en los registros del dataset, lo que permitió estandarizar la información para un entrenamiento más eficaz y maximizar el potencial del modelo. La Tabla 9 se muestran los datos originales, donde se identificaron caracteres especiales que fueron removidos para evitar ruido e interferencia en el conjunto. Seguidamente se mostró el resultado final, ya sin dichos caracteres, logrando un dataset más limpio y ordenado para continuar con el proceso.

Tabla 9. Eliminación de caracteres especiales en los registros del dataset.

Caracteres especiales	Registro sin caracteres especiales
Santiago Abascal emula a EAJ-PNV al cargar	Santiago Abascal emula a EAJ PNV al cargar
Empresa Vinazin S.A desiste de plan inmobiliario	Empresa Vinazin desiste de plan inmobiliario
Carmen Calvo, sobre la “entrada” de miembros de	carmen calvo, sobre la entrada de miembros de
Yolanda Díaz respalda la decisión de (Colau) y	yolanda díaz respalda la decisión de colau y
Villarejo: '*´Por atacar a un comisario se han	villarejo: por atacar a un comisario se han

Tokenización

Se llevó a cabo la división del texto en partes más pequeñas, denominada tokens. Este paso resultó fundamental en el procesamiento del lenguaje natural, ya que permitió al modelo operar con unidades de texto significativas y facilitó el análisis. A continuación, se presenta un ejemplo para ilustrar el proceso.

Ejemplo:

- Texto original: “Hola, ¿cómo estás?”
- Después de la tokenización: [“hola”, “cómo”, “estás”]

En la Tabla 10 se presenta el texto antes y después del proceso de tokenización. Esta visualización permitió apreciar la estructura tal como fue recolectada y observar su transformación en unidades más pequeñas, lo que resultó fundamental para el posterior procesamiento y análisis durante el entrenamiento del modelo.

Tabla 10. Proceso de tokenización aplicado a los textos del dataset, el texto original se descompone en unidades individuales.

Texto original	Texto tokenizado
candidato presidencia guillermo lasso alianza	[‘candidato’, ‘presidencial’, ‘guillermo’, ‘lasso’, ’alianza’]
moncloa destaca concepto estratégico aprobado	[‘moncloa’,’destaca’,’concepto’,’estratégico’,’aprobado’]
parte miembros adelante andalucía donarán	[‘parte’,’miembros’, ‘adelante’,’andalucía’, ‘donarán’]
vivienda movilidad turismo temas clave	[‘vivienda’, ‘movilidad’, ‘turismo’, ‘temas’, ‘clave’]
yeremi vargas dice gobierno puede plantear	[‘Yeremi’, ‘vargas’, ‘dice’, ‘gobierno’, ‘puede’, ‘plantear’]

Unir tokens en una sola cadena

Se efectuó la unión de los tokens en una sola cadena al eliminar todas las palabras y caracteres irrelevantes, para lo cual se empleó el método *join* de las cadenas de Python. Este proceso generó un texto más legible y comprensible, aspecto fundamental para los modelos de clasificación de texto en machine learning que requieren entradas en formato de cadena. En la Tabla 11 se presenta el proceso de unión de las cadenas tokenizadas en la tarea de normalización de datos.

Tabla 11. Tokens convertidos a una sola cadena.

Texto tokenizado
presidente daniel noboa previstas serie actividades bilaterales asamblea general naciones unidas
alcalde jaen arremete vicepresidenta calvo mientras psoe cordoba defiende
presidente ecuador guillermo lasso jueves cedera cargo presidente electo daniel noboa mostro martes satisfecho haber pactado correismo pese
presidente lenin moreno reune lunes abril mandatario electo guillermo lasso oficializar intercambio información
evento universidad tecnica babahoyo presidente ecuatoriano daniel noboa entrego mil becas dolares estudiantes

Tarea 6: División de datos.

Durante la tarea, los registros se dividieron en tres conjuntos: entrenamiento, validación y pruebas, con el propósito de realizar entrenamientos con y sin optimización. Las noticias se seleccionaron aleatoriamente, y se empleó un código en Python descrito en la sección de metodología **(véase en Tarea 6: División de datos)** para generar dichos conjuntos. En la Figura 45 se detalla el porcentaje de datos asignado a cada conjunto.

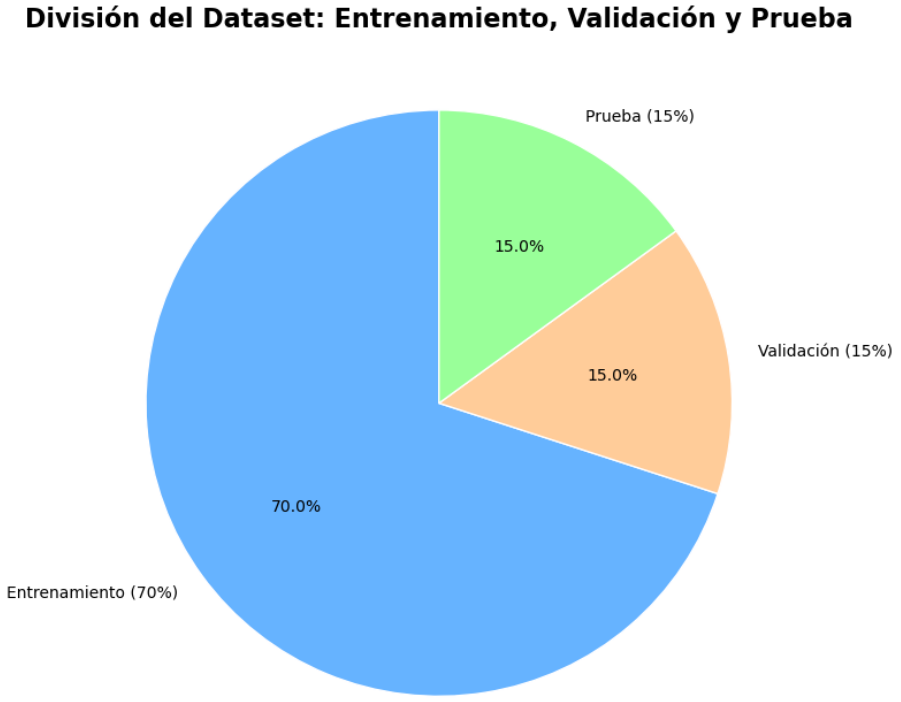


Figura 45. División del dataset: 70% para entrenamiento, 15% para validación y 15% para prueba.

Fase: Ingeniería de Modelos.

Tarea 7: Ajuste de hiperparámetros.

En la Tabla 12, se presentan las configuraciones utilizadas de hiperparámetros para los distintos algoritmos de optimización estos parámetros fueron utilizados para realizar cada uno de los entrenamientos.

Tabla 12. Configuración de los hiperparámetros de los algoritmos de optimización.

Hiperparámetro	GD	SGD	MBGD	AdaGrad	Adam	RMSprop
Learning Rate	10^{-2}	10^{-4}	$5x10^{-3}$	10^{-4}	10^{-4}	$5x10^{-5}$
Batch Size	1	1	64	128	256	256
Momentum	0.9	0.9	0.9	-	-	-
Beta 1	-	-	-	-	0.9	-
Beta 2	-	-	-	-	0.999	-
Epsilon	-	-	-	10^{-8}	10^{-8}	10^{-8}
Decay Rate	10^{-12}	10^{-12}	-	-	-	0.99
Regularization	-	-	10^{-4}	10^{-2}	-	-
Epochs	5.000	100.000	3.000	1.210	602	1.165
Stopping Tolerance	10^{-5}	-	-	10^{-4}	10^{-4}	10^{-4}

Número de épocas (Epochs); Tasa de aprendizaje (Learning rate); Tamaño del lote (Batch size); Gradiente descendente (GD), Gradiente descendente estocástico (SGD), Gradiente descendente por mini lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam), Propagación de la Raíz del Promedio Cuadrática (RMSProp).

Tarea 8: Selección del modelo.

El modelo de regresión logística evidenció un desempeño estable en la fase de validación, obteniendo una exactitud de 0.745, una precisión de 0.751, una especificidad de 0.737, una sensibilidad de 0.752 y un F1 Score de 0.752. Dichos valores confirmaron la capacidad del modelo para distinguir adecuadamente entre clases positivas y negativas; en consecuencia, se llevaron a cabo diversas optimizaciones mediante Gradiente Descendente (GD), Gradiente Descendente Estocástico (SGD), Gradiente Descendente por Mini Lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam) y Propagación de la Raíz del Promedio Cuadrático (RMSProp). En la tabla correspondiente se presentaron las precisiones obtenidas con cada método, evidenciando la utilidad del modelo de regresión logística en tareas de predicción.

Tarea 9: Selección de optimización.

De los algoritmos de optimización evaluados, el Gradiente Descendente Estocástico (SGD) alcanzó la mejor precisión en validación con un 81.30%, superando ligeramente al Gradiente Descendente (GD), que obtuvo un 80.0%. En contraste, algoritmos como el Gradiente Descendente por Mini Lotes (MBGD), Adam y RMSProp lograron precisiones inferiores, de 76.11%, 79.02% y 79.24%, respectivamente, mientras que AdaGrad presentó el desempeño más bajo con un 71.51%. Por lo tanto, SGD se destacó como el mejor algoritmo de optimización en términos de precisión. Considerando los resultados de la Tarea 10 y la

Tabla 13, se seleccionó el optimizador SGD para realizar pruebas con datos no conocidos por el modelo.

Tarea 10: Entrenamiento del modelo.

En la Tabla 13 se detallan las precisiones alcanzadas por los diferentes algoritmos de optimización, entre ellos Gradiente descendente (GD), Gradiente descendente estocástico (SGD), Gradiente descendente por mini lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam), Propagación de la Raíz del Promedio Cuadrática (RMSProp). Dichos resultados evidenciaron cómo los ajustes de hiperparámetros contribuyeron a mejorar el rendimiento y prevenir el sobreajuste al verificar la precisión de cada modelo.

Tabla 13. Resultados de la precisión de los algoritmos de optimización con distintos ajustes de hiperparámetros.

Algoritmo de optimización	Epochs	Learning Rate	Batch Size	Momentum	Partition	Precision (%)
GD	5.000	10^{-2}	1	0.9	70:15:15	80.0%
SGD	100.000	10^{-4}	1	0.9	70:15:15	81.3%
MBGD	3.000	$5x10^{-3}$	64	0.9	70:15:15	76.5%
AdaGrad	1.210	10^{-4}	128	-	70:15:15	71.5%
Adam	602	10^{-4}	256	-	70:15:15	79.0%
RMSProp	1.165	$5x10^{-5}$	256	-	70:15:15	79.2%

Número de épocas (Epochs); Tasa de aprendizaje (Learning rate); Tamaño del lote (Batch size); División de datos (Partition); Precisión (Precision); Gradiente descendente (GD), Gradiente descendente estocástico (SGD), Gradiente descendente por mini lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam), Propagación de la Raíz del Promedio Cuadrática (RMSProp).

En la siguiente gráfica se representaron las comparaciones de las precisiones para cada uno de los optimizadores, tomando como referencia los resultados de entrenamiento y validación de la Tabla 13 de la Tarea 10. De esta forma, se visualizó de manera clara el desempeño de cada modelo junto con su nivel de precisión.

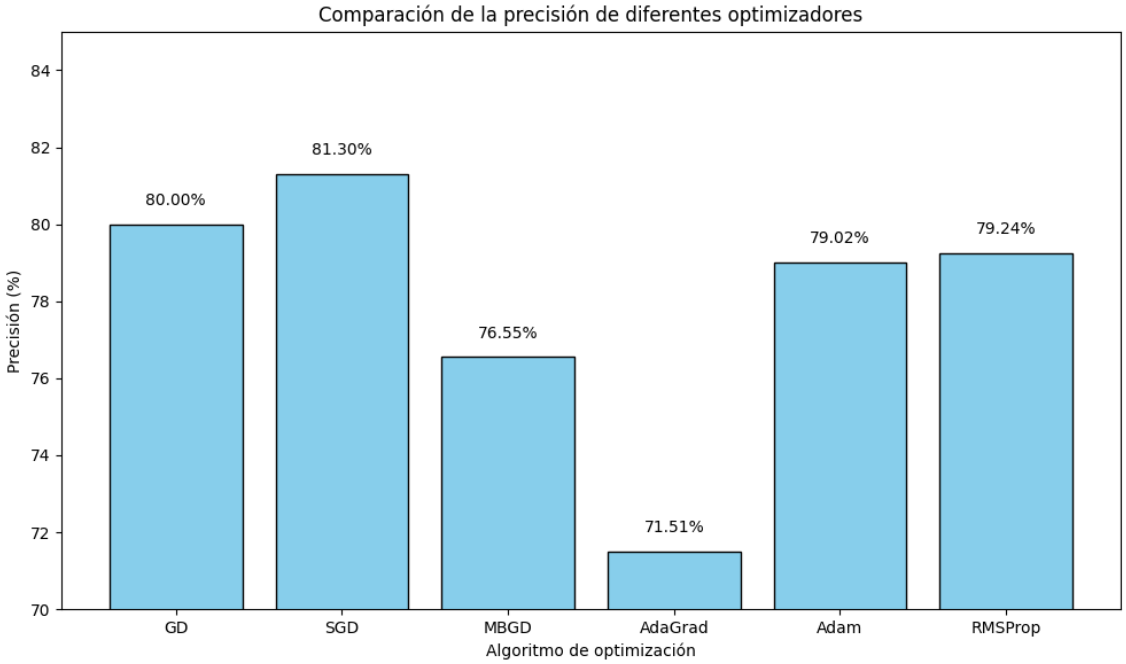


Figura 46. Comparativa de la métrica precisión con los optimizadores Gradiente descendente (GD), Gradiente descendente estocástico (SGD), Gradiente descendente por mini lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam), Propagación de la Raíz del Promedio Cuadrática (RMSProp).

6.2. Objetivo 2: Evaluar la mejora en la precisión del modelo en comparación con la precisión inicial sin optimización utilizando la matriz de confusión.

Fase: Evaluación del modelo

La evaluación del modelo base de regresión logística sin optimización estableció un punto de referencia en términos de precisión, con el propósito de comparar el impacto de la optimización sobre el rendimiento. Para asegurar equidad en dicha comparación, ambos modelos (con y sin optimización) se evaluaron con la misma cantidad de datos **(véase Tarea 10),** donde se evidenció que el gradiente descendente estocástico (SGD) obtuvo la mayor precisión de 81.3% en la fase de validación. Asimismo, se aclara que el modelo de regresión logística se entrenó en su configuración predeterminada de la biblioteca sklearn, sin introducir modificaciones en sus parámetros.

Fase: Pruebas del modelo.

Las métricas obtenidas para cada modelo se analizaron a través de sus respectivas matrices de confusión, como se describe a continuación. Se calcularon las métricas precisión (precisión), especificidad (specificity), exactitud (accuracy), sensibilidad (sensitivity), f1-score, utilizando la variable Label con las clases 0 para noticias falsas y 1 para noticias verdaderas. En la Tabla 14 se presentan los valores de TP, TN, FP, FN obtenidos tanto en el modelo base como en el modelo optimizado.

A continuación, se compararon tanto la precisión como el resto de las métricas de rendimiento entre el modelo base de regresión logística sin optimización **(véase en Anexo 4**

la Figura 55) y el mejor modelo, que correspondió al algoritmo de gradiente descendente estocástico (SGD) **(véase en Anexo 4 la Figura 60)**, dado que obtuvo la mayor precisión durante la fase de validación.

Tabla 14. Resultados de la cantidad de predicciones correctas e incorrectas mediante la matriz de confusión del modelo regresión logística sin optimización.

Clase	Medidas		
	Predicción Negativa	**Predicción Positiva**	**Total**
Clase:0 (Negativo)	TN = 4141	FP = 1480	**5621**
Clase: 1 (Positivo)	FN = 1367	TP = 4154	**5521**
Total	**5508**	**5634**	**11142**

TP (Verdaderos Positivos); TN (Verdaderos Negativos); FP (Falsos Positivos); FN (Falsos Negativos).

Tabla 15. Resultados de la cantidad de predicciones correctas e incorrectas mediante la matriz de confusión del modelo optimizado con el algoritmo de optimización gradiente descendente estocástico (SGD).

Clase	Medidas		
	Predicción Negativa	**Predicción Positiva**	**Total**
Clase:0 (Negativo)	TN = 4612	FP = 1009	**5621**
Clase: 1 (Positivo)	FN = 1396	TP = 4125	**5521**
Total	**6008**	**5134**	**11142**

TP (Verdaderos Positivos); TN (Verdaderos Negativos); FP (Falsos Positivos); FN (Falsos Negativos).

En la Tabla 16 se muestran los resultados entre el modelo sin optimización y el mejor modelo optimizado (SGD), en donde se comparan las métricas generadas a partir de la matriz de confusión.

Tabla 16. Comparación de los resultados obtenidos de las matrices de confusión del modelo regresión logística sin optimización y el modelo optimizado mediante gradiente descendente estocástica (SGD).

Modelo regresión logística sin optimización	Modelo regresión logística optimizado mediante SGD
• **Precisión (Precision)** $precision = \frac{TP}{TP + FP} = \frac{\Sigma\, TP}{\Sigma\, TP + \Sigma\, FP}$ $precision = \frac{4154}{4154 + 1480} = \frac{4154}{5634}$ $precision = 0,737 = 73,7\%$	• **Precisión (Precision)** $precision = \frac{TP}{TP + FP} = \frac{\Sigma\, TP}{\Sigma\, TP + \Sigma\, FP}$ $precision = \frac{4125}{4125 + 1009} = \frac{4125}{5134}$ $precision = 0,803 = 80,3\%$
• **Especificidad (Specificity)** $specificity = \frac{TN}{TN + FP} = \frac{\Sigma\, TN}{\Sigma\, TN + \Sigma\, FP}$ $specificity = \frac{4141}{4141 + 1480} = \frac{4141}{5621}$ $specificity = 0,736 = 73,6\%$	• **Especificidad (Specificity)** $specificity = \frac{TN}{TN + FP} = \frac{\Sigma\, TN}{\Sigma\, TN + \Sigma\, FP}$ $specificity = \frac{4612}{4612 + 1009} = \frac{4612}{5621}$ $specificity = 0,820 = 82,0\%$
• **Sensibilidad (Sensitivity)**	• **Sensibilidad (Sensitivity)**

$$sensitivity = \frac{TP}{TP + FN} = \frac{\Sigma\, TP}{\Sigma\, TP + \Sigma\, FN}$$
$$sensitivity = \frac{4154}{4154 + 1367} = \frac{4154}{5521}$$
$$sensitivity = 0,752 = 75,2\%$$

$$sensitivity = \frac{TP}{TP + FN} = \frac{\Sigma\, TP}{\Sigma\, TP + \Sigma\, FN}$$
$$sensitivity = \frac{4125}{4125 + 1396} = \frac{4125}{5521}$$
$$sensitivity = 0,747 = 74,7\%$$

Modelo regresión logística sin optimización	**Modelo regresión logística optimizado mediante SGD**
• **Exactitud (Accuracy)** $accuracy = \frac{TP + TN}{TP + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$ $accuracy = \frac{\Sigma\, TP + \Sigma\, TN}{\Sigma TP + \Sigma \mathrm{TN} + \Sigma \mathrm{FP} + \Sigma \mathrm{FN}}$ $accuracy = \frac{4154 + 4141}{4154 + 4141 + 1480 + 1367}$ $accuracy = \frac{8295}{11142}$ $accuracy = 0,744 = 74,4\%$	• **Exactitud (Accuracy)** $accuracy = \frac{TP + TN}{TP + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$ $accuracy = \frac{\Sigma\, TP + \Sigma\, TN}{\Sigma TP + \Sigma \mathrm{TN} + \Sigma \mathrm{FP} + \Sigma \mathrm{FN}}$ $accuracy = \frac{4125 + 4612}{4125 + 4612 + 1009 + 1396}$ $accuracy = \frac{8737}{11142}$ $accuracy = 0,784 = 78,4\%$
• **F1-Score** $f1 = 2\frac{precision * sensitivity}{precision + sensitivity}$ $f1 = 2\,\frac{0,737 * 0,752}{0,737 + 0,752}$ $f1 = 2\,\frac{0,554}{1,489} = 0,744 = 74,5\%$	• **F1-Score** $f1 = 2\frac{precision * sensitivity}{precision + sensitivity}$ $f1 = 2\,\frac{0,803 * 0,747}{0,803 + 0,747}$ $f1 = 2\,\frac{0,5998}{1.55} = 0,7739 = 77,4\%$

Precisión (precisión); especificidad (specificity); exactitud (accuracy); sensibilidad (sensitivity); f1-score.

Durante la fase de evaluación del modelo, se usaron idénticos hiperparámetros (épocas, tasa de aprendizaje, tamaño de lote, momento) y la misma partición del conjunto de datos, conforme a lo señalado en la Tabla 13 de la Tarea 10. En esta etapa, el conjunto de prueba (test_data) se empleó para verificar el desempeño de los distintos métodos, aplicando la matriz de confusión **(véase Anexo 4)**. Los resultados corroboraron lo obtenido en la fase de validación: el modelo con gradiente descendente estocástico (SGD) alcanzó la mayor precisión, tal como se aprecia en la Figura 47 de la comparativa de los diferentes optimizadores.

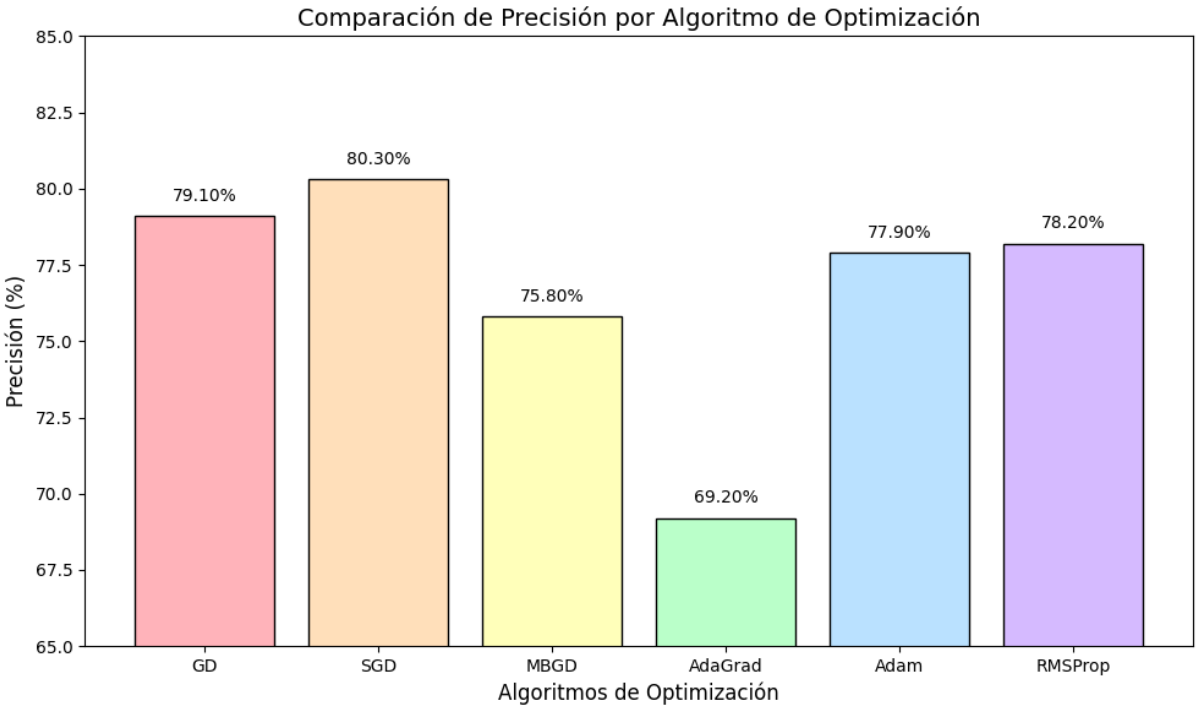


Figura 47. Comparación de la precisión de los optimizadores Gradiente descendente (GD), Gradiente descendente estocástico (SGD), Gradiente descendente por mini lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam), Propagación de la Raíz del Promedio Cuadrática (RMSProp) en el dataset de prueba.

El desempeño del rendimiento de los algoritmos de optimización Gradiente descendente (GD), Gradiente descendente estocástico (SGD), Estimación de Momento Adaptativo (Adam), Propagación de la Raíz del Promedio Cuadrática (RMSProp) los cuales dieron las mejores métricas tal cual nos demuestra sus matrices de confusión (ver Figura 56, Figura 65, Figura 66 del Anexo 4).

Por otro lado, el algoritmo Gradiente descendente por mini lotes (MBGD), mostró un rendimiento aceptable superando siempre al modelo base sin optimización regresión logística se puede observar mediante la matriz de confusión de este modelo (ver Figura 63 del Anexo 4). Sin embargo, el optimizador AdaGrad fue el que tuvo el peor rendimiento en comparación con la regresión logística (ver Figura 64 del Anexo 4).

En la Figura 48 se representa una comparación entre las métricas generadas a partir de la matriz de confusión en donde se ve la superioridad entre el modelo optimizado con gradiente descendente estocástico (SGD) en contra del modelo sin optimización regresión logística.

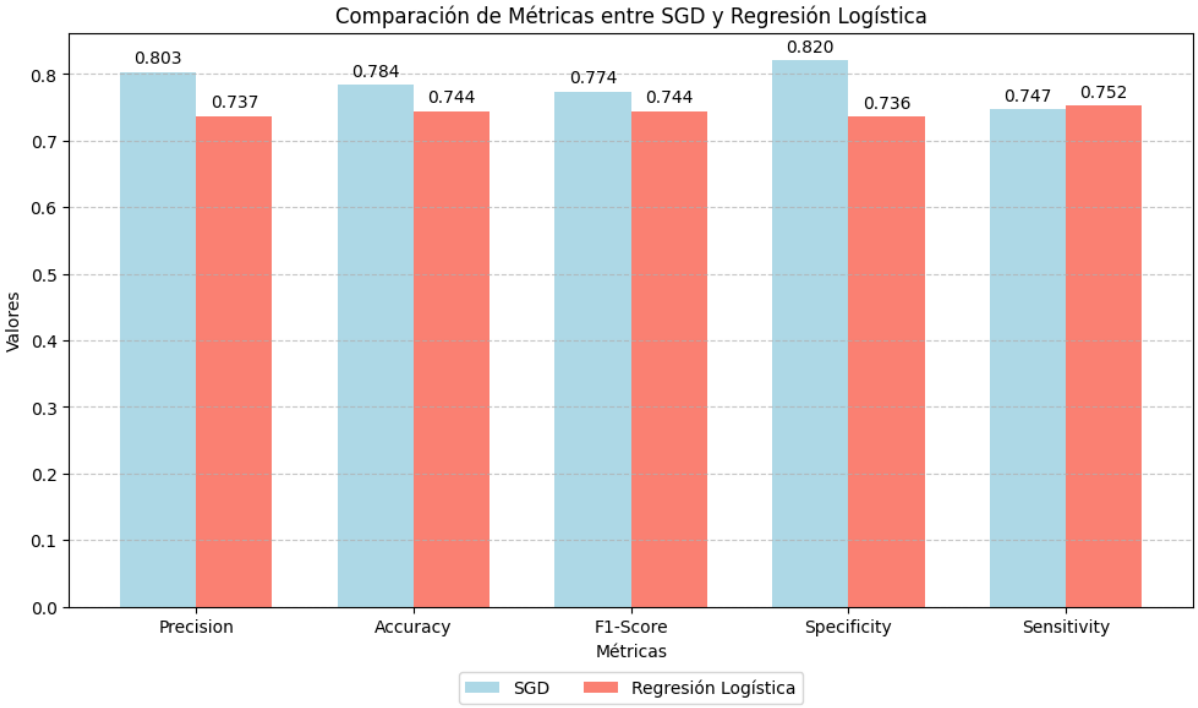


Figura 48. Comparación de las métricas de la matriz de confusión del modelo regresión logística y el modelo optimizado mediante gradiente descendente estocástico.

Por lo tanto, mediante la se puede seleccionar el mejor modelo tanto en validación como en pruebas el cual es el gradiente descendente estocástico (SGD) debido a que presenta las mejores métricas en comparación con el modelo base. Se genero un archivo .pt[11] en donde estará empaquetado el modelo resultante llamado SGD-LR (gradiente descendente estocástico – regresión logística).

Desarrollo de un prototipo de interfaz gráfica

Para mejorar la evaluación, se desarrolló un prototipo de interfaz web[12] basado en Flask que permite a los usuarios analizar la veracidad de noticias de política en español mediante la inserción de texto. Este prototipo utiliza el modelo en formato pt generado después de la evaluación del modelo y facilita la generación de informes con un número configurable de predicciones (entre 5 y 20), los cuales pueden descargarse en formato PDF para su análisis posterior. La interfaz, diseñada con un enfoque minimalista, prioriza la usabilidad, presentando únicamente la información básica y esencial para el usuario. No obstante, aún no se ha implementado una metodología formal para evaluar la experiencia de uso, aunque se busca en un futuro que esta sea intuitiva y eficiente.

En la Figura 49 se presenta el prototipo de la interfaz gráfica web en su versión local. Actualmente la interfaz solo se encuentra disponible para uso en un entorno loca, limitando la accesibilidad desde otros dispositivos. Se considera el funcionamiento posterior en la nube considerando opciones como AWS (Amazon Web Services), GCP (Google Cloud Platform) o

[11] Modelo empaquetado optimizado (SGD-LR)
[12] Prototipo Interfaz Gráfica

Heroku, que ofrecen servicios de alojamiento de aplicaciones web. Por otro lado, tenemos alternativas como Docker para contenerizar la aplicación facilitando el despliegue en servidores remotos. Dado que el modelo se encuentra en formato .pt no es pesado y la interfaz no requeriría una infraestructura completa.

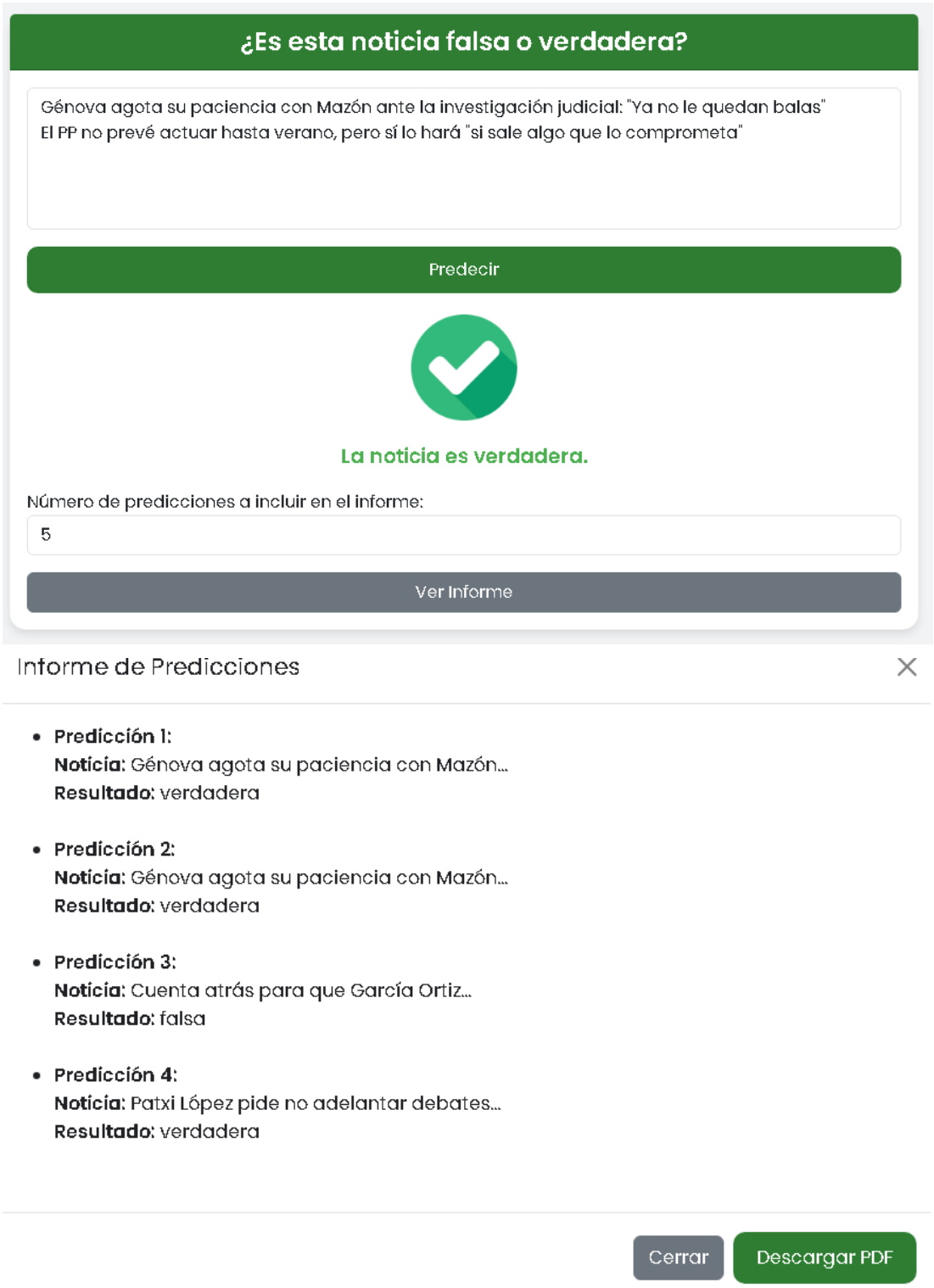


Figura 49. Desarrollo de prototipo de interfaz gráfica web basada en Flask.

7. Discusión

Se establecieron dos objetivos específicos y un objetivo general. Por lo tanto, en este apartado se discuten principalmente los resultados obtenidos para cada objetivo.

7.1. Primer objetivo: Entrenar el modelo regresión logística con los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp mediante CRISP-ML.

Se logró aplicar y adaptar las fases de la metodología CRISP-ML, lo que permitió, a través de la fase de ingeniería de datos, la generación de un dataset de noticias en español enriquecido con contenido de periódicos ecuatorianos. De esta manera, se obtuvo un nuevo conjunto de datos utilizado para la optimización, diferenciándose de la mayoría de trabajos relacionados, que emplean en su mayoría datasets en inglés **(véase en TR02, TR03, TR04 y TR05)**.

En relación con este objetivo, se consiguió realizar un entrenamiento adecuado mediante los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp, en donde se evidenció una mejora en el porcentaje de precisión durante la fase de validación en comparación con el 75,1% inicial del modelo de regresión logística. En particular, el algoritmo de gradiente descendente estocástico (SGD) presentó el mejor desempeño, alcanzando una precisión del 81,3%. **(véase Figura 46 & Tabla 13)** Sin embargo, estos optimizadores aún poseen un amplio potencial de convergencia; por ejemplo, en el trabajo **TR08** se profundiza en variantes mejoradas de SGD, tales como S2GD, SDCA, MISO o SARAH+.

En este trabajo, a diferencia de lo planteado en el alcance del proyecto de investigación de integración curricular (**Anexo 6**), se consideró inicialmente el uso de aumento de datos mediante Back Translation para enriquecer el dataset y mejorar la capacidad de generalización del modelo, pero esta técnica fue descartada debido a limitaciones como las restricciones del API de Google, los costos asociados, posibles errores de traducción y la falta de recursos computacionales; esto pudo haber impactado los resultados al depender exclusivamente de la calidad del dataset original.

La capacidad de procesamiento disponible constituyó una limitación durante el entrenamiento, impidiendo aprovechar el potencial de los métodos de gradiente descendente y sus variantes, en especial los adaptativos. Las restricciones de hardware (memoria RAM, GPU y CPU) limitaron una evaluación más extensa del desempeño de estos algoritmos. Dichas dificultades podrían mitigarse mediante la paralelización de hardware con múltiples núcleos o GPU, lo que resultaría especialmente beneficioso para optimizadores como SGD al manejar grandes volúmenes de datos, tal como se describe Tian et al. En **TR08**.

Además, es importante señalar que la regularización no se trató como una tarea independiente como lo mencionado en el alcance del proyecto de investigación de integración curricular (**Anexo 6**), sino como parte del ajuste de hiperparámetros, integrándose en el

proceso de optimización para encontrar los valores más adecuados en el rendimiento del modelo, lo que permitió un enfoque más consistente en la configuración del modelo.

7.2. Segundo objetivo: Evaluar la mejora en la precisión del modelo en comparación con la precisión inicial sin optimización utilizando la matriz de confusión.

La evaluación de los modelos mediante la matriz de confusión evidenció que el modelo optimizado con gradiente descendente estocástico alcanzó la mejor precisión, superando en 6.6% al modelo sin optimización. Este resultado coincide con la mayoría de los trabajos relacionados, confirmando que la aplicación de métodos de optimización (Gradiente Descendente, Gradiente Descendente Estocástico, Gradiente Descendente por Mini Lotes, Algoritmo de Gradiente Adaptativo, Estimación de Momento Adaptativo y Propagación de la Raíz del Promedio Cuadrático) mejora significativamente el rendimiento de las métricas, sobre todo en términos de precisión **(véase en TR01, TR02 y TR03)**.

Al comparar nuestros resultados, la combinación SGD-LR ofreció una mejora de 6.6% frente al trabajo **TR04**, donde el uso de SGD optimizado con Random Search CV logró un incremento de 0.7%. De igual forma, el método Adam-LR alcanzó una precisión de 77.95%, superando el 74.8% obtenido por Adam en la detección de noticias falsas en árabe (**TR05**). Por otra parte, en **TR02** se reporta que Adam mejora la precisión en un 9.54% para el dataset DS1 y RMSProp en un 5.87% para el dataset DS4; en contraste, en nuestro estudio, Adam logró un incremento de 4.25% y RMSProp de 4.5%.

Estos resultados contrastaron con los hallazgos del trabajo **TR01**, donde el modelo de regresión logística en conjunto con gradiente descendente (LR-GD) evidenció un incremento del 15% en sus métricas, especialmente en la precisión. En cambio, en este estudio, el uso de gradiente descendente (GD) logró una mejora del 5.4%. Se logró comprender las ventajas y limitaciones de los diversos métodos de optimización, respondiendo así la pregunta de investigación. En particular, los porcentajes de precisión que alcanzó cada algoritmo fueron: GD (79.1 %), SGD (80.3 %), MBGD (75.8 %), AdaGrad (69.20 %), Adam (77.9 %) y RMSProp (78.2 %) **(véase Figura 47)**.

Además, se identificaron limitaciones en cuanto a la calidad y disponibilidad de datos libres sobre noticias políticas en español. Para lo cual, se fusionaron distintos datasets con información obtenida mediante web scraping contando con un dataset de 82,355 registros, a diferencia de otros estudios, por ejemplo, **TR03** (44,898 registros), **TR06** (51,233 registros) y **TR02** (12,800 registros). Esta personalización del dataset con las limitaciones propias del modelo sumado a sesgos y ruido de los datos podría haber limitado la capacidad de generalización ante nuevas instancias. Con el fin de experimentar con la capacidad y la detección de noticias falsas, se contempla el uso de algoritmos de clasificación más avanzados, como SVM, Random Forest o redes neuronales con distintas configuraciones, siguiendo las recomendaciones de Mafla Checa. Et al. [60].

El cronograma planteado en el proyecto de investigación de integración curricular (**Anexo 6**) se ejecutó en su mayoría según lo planeado. Sin embargo, la etapa de entrenamiento de modelos requirió una extensión de dos semanas, derivada del tiempo dedicado al ajuste de hiperparámetros y al proceso iterativo de pruebas. Estas modificaciones optimizaron la calidad y los resultados finales de la fase de entrenamiento.

Además, el presupuesto del proyecto de integración curricular sufrió un ajuste menor al reemplazar el uso planificado de Google Colab Pro por la adquisición de una GPU **(véase Tabla 3)**. Este cambio generó costos adicionales no contemplados en la propuesta inicial del proyecto de investigación de integración curricular (**Anexo 6**).

8. Conclusiones

- La aplicación de algoritmos de optimización (GD, SGD, MBGD, AdaGrad, Adam y RMSProp) en un modelo de regresión logística para la detección de noticias falsas de política en español demostró que el gradiente descendente estocástico (SGD) alcanza una precisión del 80.3%, superando en un 6.6% al modelo sin optimización (73.7%). Además, el gradiente descendente (GD) obtuvo un 79.10% de precisión, mientras que RMSProp alcanzó un 78.20%. Este resultado evidencia la importancia de ajustar la regresión logística para lograr una clasificación más confiable en este ámbito.
- Para el entrenamiento, se creó un nuevo dataset de 82,355 registros al fusionar dos datasets y añadir noticias de política ecuatoriana, garantizando diversidad de fuentes, volumen de datos y riqueza informativa. Junto con el ajuste de hiperparámetros, este proceso optimizó la precisión del modelo, destacando que SGD logró el mejor rendimiento entre los métodos evaluados. Asimismo, la aplicación de la metodología CRISM-ML facilitó un entrenamiento estructurado, asegurando la implementación y comparación de los algoritmos.
- La evaluación del modelo en la fase de validación, mediante matrices de confusión, ofreció un análisis detallado de su rendimiento. Las métricas de sensibilidad, especificidad, precisión, exactitud y F1 confirmaron que los métodos tradicionales de descenso de gradiente, como el gradiente descendente estocástico (SGD) y el gradiente descendente (GD), obtuvieron los mejores resultados. Sin embargo, los algoritmos de optimización adaptativos, como RMSProp, también demostraron un desempeño competitivo. Además, se identificaron limitaciones en la detección de noticias falsas debido a la complejidad semántica del lenguaje y la posible existencia de sesgos en los datos, lo que resalta la necesidad de seguir mejorando los enfoques de optimización y la calidad del conjunto de datos utilizados.

9. Recomendaciones

- Explorar modelos de machine learning más avanzados o potentes como basados en redes neuronales con el fin de realizar una comparación en las métricas con el modelo optimizado (SGD-LR) propuesto en este trabajo.
- Recopilar nuevos datos con distintos contextos, más allá del ámbito político, para mejorar la precisión y la capacidad de generalización del modelo.
- Reproducir la optimización del modelo de regresión logística combinado con los distintos métodos (GD, SGD, MBGD, AdaGrad, Adam, RMSProp) con mayor poder computacional, a fin de evaluar de forma más completa diversas configuraciones de hiperparámetros. Del mismo modo, se recomienda explorar enfoques de conjunto (ensemble), como LR + SGD + TF-IDF + BERT, que pueden incrementar la precisión a costa de requerir recursos de procesamiento más elevados.
- Aplicar técnicas de aumento de datos, como Back Translation, que en este trabajo se descartaron por limitaciones de la API de Google y recursos computacionales.
- Investigar la aplicación de estos modelos optimizados en entornos reales donde la mitigación de desinformación sea prioritaria.
- Integrar métodos de optimización más sofisticados o combinarlos para comparar las mejoras que aportan respecto de los optimizadores ya utilizados, permitiendo identificar fortalezas y debilidades en diferentes escenarios de clasificación de texto.
- Evaluación continua a lo largo del tiempo para detectar posibles carencias y perfeccionar el modelo optimizado conforme evolucionen las necesidades y los datos disponibles.

10. Bibliografía

[1] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[2] F. Mian, “Enhancing Integrated Logistic Support through Machine Learning-Driven Optimization : Predicting component category based on part number,” 2024.

[3] Y. Tian, Y. Zhang, and H. Zhang, “Recent Advances in Stochastic Gradient Descent in Deep Learning,” Feb. 01, 2023, *MDPI*. doi: 10.3390/math11030682.

[4] E. V. S. Raja, B. L. V. S. Aditya, and S. N. Mohanty, “Fake Profile Detection Using Logistic Regression and Gradient Descent Algorithm on Online Social Networks,” *EAI Endorsed Transactions on Scalable Information Systems*, vol. 11, no. 1, 2024, doi: 10.4108/eetsis.4342.

[5] N. Sharma, R. Sharma, and N. Jindal, “Machine Learning and Deep Learning Applications-A Vision,” *Global Transitions Proceedings*, vol. 2, no. 1, pp. 24–28, Jun. 2021, doi: 10.1016/j.gltp.2021.01.004.

[6] Bobadilla Jesús, “Machine Learning y Deep Learning Usando Python, Scikit y Keras Informática,” 2020.

[7] M. M. Taye, “Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions,” May 01, 2023, *MDPI*. doi: 10.3390/computers12050091.

[8] S. A. Grillo *et al.*, “Adjacent Inputs with Different Labels and Hardness in Supervised Learning,” *IEEE Access*, vol. 9, pp. 162487–162498, 2021, doi: 10.1109/ACCESS.2021.3131150.

[9] I. H. Sarker, “Machine Learning: Algorithms, Real-World Applications and Research Directions,” May 01, 2021, *Springer*. doi: 10.1007/s42979-021-00592-x.

[10] D. Lopez-Bernal, D. Balderas, P. Ponce, and A. Molina, “Education 4.0: Teaching the basics of knn, lda and simple perceptron algorithms for binary classification problems,” *Future Internet*, vol. 13, no. 8, Aug. 2021, doi: 10.3390/fi13080193.

[11] E. M. Rojas, “Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo,” Apr. 2020.

[12] V. Castillo-Riquelme, P. Hermosilla-Urrea, J. P. Poblete-Tiznado, and C. Durán-Anabalón, “Fake news and unfunded beliefs in the post-truth age,” *Universitas (Stuttg)*, no. 34, pp. 87–108, Feb. 2021, doi: 10.17163/uni.n34.2021.04.

[13] C. Lamoneda Reyes, “Fake news at high school: propuesta de intervención educativa en competencias básicas para educación en detección de noticias falsas (fake news) y manipulación a través de la red,” 2021. Accessed: Feb. 03, 2025. [Online]. Available: https://hdl.handle.net/10953.1/13551

[14] D. P. Calvillo, B. J. Ross, R. J. B. Garcia, T. J. Smelter, and A. M. Rutchick, “Political Ideology Predicts Perceptions of the Threat of COVID-19 (and Susceptibility to Fake News About It),” *Soc Psychol Personal Sci*, vol. 11, no. 8, pp. 1119–1128, Nov. 2020, doi: 10.1177/1948550620940539.

[15] R. P. Bringula, A. E. Catacutan-Bangit, M. B. Garcia, J. P. S. Gonzales, and A. M. C. Valderama, “‘Who is gullible to political disinformation?’ : predicting susceptibility of university students to fake news,” *Journal of Information Technology and Politics*, vol. 19, no. 2, pp. 165–179, 2022, doi: 10.1080/19331681.2021.1945988.

[16] R. D. Joshi and C. K. Dhakal, “Predicting type 2 diabetes using logistic regression and machine learning approaches,” *Int J Environ Res Public Health*, vol. 18, no. 14, Jul. 2021, doi: 10.3390/ijerph18147346.

[17] A. Alves *et al.*, “Leia este artigo se você quiser aprender regressão logística”, doi: 10.1590/1678-987320287406.

[18] A. Arista, “Comparison Decision Tree and Logistic Regression Machine Learning Classification Algorithms to determine Covid-19,” *Sinkron*, vol. 7, no. 1, pp. 59–65, Jan. 2022, doi: 10.33395/sinkron.v7i1.11243.

[19] K. Kuntoro, “Comparing accuracy of logistic regression, k-nearest neighbor, support vector machine, and naïve bayes models using tracking ensemble machine learning,” *JP J Biostat*, vol. 24, no. 1, pp. 1–13, Oct. 2023, doi: 10.17654/0973514324001.
[20] Haji, Saad Hikmat, Adbulazeez, and Adnan Mohsin, “Comparison of optimization techniques based on gradient descent algorithm: a review,” Feb. 2021.
[21] L. Chen and J. Bruna, “Beyond the Edge of Stability via Two-step Gradient Updates,” 2023.
[22] A. Beznosikov, E. Gorbunov, H. Berard, and M. Diro, “Stochastic Gradient Descent-Ascent: Unified Theory and New Efficient Methods,” 2023.
[23] Moreno Cabañas and Nelson bruno Andrés, “Estrategias de selección de mini batches utilizando procesos puntuales determinantales para el entrenamiento de redes neuronales mediante descenso de gradiente estocástico,” *Universidad de chile*, 2023, doi: 10.58011/wd30-ad68.
[24] N. Hyder and G. Friedland, “Learning Rate Estimation for Stochastic Gradient Descent,” 2022. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-155.html
[25] H. Wang, H. Luo, and Y. Wang, “MBGDT:Robust Mini-Batch Gradient Descent.”
[26] F. R. J. Simanungkalit, H. Hanifah, G. Ardaneswari, N. Hariadi, and B. D. Handari, “Prediction of students’ academic performance using ANN with mini-batch gradient descent and Levenberg-Marquardt optimization algorithms,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Nov. 2021. doi: 10.1088/1742-6596/2106/1/012018.
[27] M. Nishchal, J. Mr, and N. Bhandari, “Computational Complexity of Gradient Descent Algorithm.”
[28] A. Rambidis, “Algorithmic Behaviours of Adagrad in Underdetermined Linear Regression,” 2023.
[29] S. Surono, A. Thobirin, Z. A. R. Hsm, A. Y. Astuti, B. R. Kp, and M. Oktavia, “Optimization of Fuzzy System Inference Model on Mini Batch Gradient Descent,” in *Frontiers in Artificial Intelligence and Applications*, IOS Press BV, Oct. 2022, pp. 224–232. doi: 10.3233/FAIA220387.
[30] A. Alblwi, “Improving the Adaptive Moment Estimation Optimization Methods for Modern Machine Learning,” ProQuest Dissertations & Theses, 27994120, University of Delaware, 2020.
[31] R. Selvaraj, T. Satheesh, V. Suresh, and V. Yathavaraj, “Optimized Machine Learning for CHD Detection using 3D CNN-based Segmentation, Transfer Learning and Adagrad Optimization,” *SSRG International Journal of Electrical and Electronics Engineering*, vol. 10, no. 3, pp. 20–34, Mar. 2023, doi: 10.14445/23488379/IJEEE-V10I3P103.
[32] P. Kairouz, K. Rush, A. Thakurta, M. Belkin, and S. Kpotufe, “(Nearly) Dimension Independent Private ERM with AdaGrad Rates via Publicly Estimated Subspaces Google Research,” 2021.
[33] I. H. Kartowisastro and J. Latupapua, “A Comparison of Adaptive Moment Estimation (Adam) and RMSProp Optimisation Techniques for Wildlife Animal Classification Using Convolutional Neural Networks,” *Revue d’Intelligence Artificielle*, vol. 37, no. 4, pp. 1023–1030, Aug. 2023, doi: 10.18280/ria.370424.
[34] M. Reyad, A. M. Sarhan, and M. Arafa, “A modified Adam algorithm for deep neural network optimization,” *Neural Comput Appl*, vol. 35, no. 23, pp. 17095–17112, Aug. 2023, doi: 10.1007/s00521-023-08568-z.
[35] J. Kang, X. Zhu, L. Shen, and M. Li, “Fault diagnosis of a wave energy converter gearbox based on an Adam optimized CNN-LSTM algorithm,” *Renew Energy*, vol. 231, Sep. 2024, doi: 10.1016/j.renene.2024.121022.
[36] R. P. Kiran, A. Professor, and N. N. C, “Op-RMSprop (Optimized-Root Mean Square Propagation) Classification for Prediction of Polycystic Ovary Syndrome (PCOS) using Hybrid Machine Learning Technique.” [Online]. Available: www.ijacsa.thesai.org
[37] P. Arafin, A. M. Billah, and A. Issa, “Deep learning-based concrete defects classification and detection using semantic segmentation,” *Struct Health Monit*, vol. 23, no. 1, pp. 383–409, Jan. 2024, doi: 10.1177/14759217231168212.

[38] H. Yun, “Prediction model of algal blooms using logistic regression and confusion matrix,” *International Journal of Electrical and Computer Engineering*, vol. 11, no. 3, pp. 2407–2413, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2407-2413.

[39] D. Krstinić, M. Braović, L. Šerić, and D. Božić-Štulić, “Multi-label Classifier Performance Evaluation with Confusion Matrix,” Academy and Industry Research Collaboration Center (AIRCC), Jun. 2020, pp. 01–14. doi: 10.5121/csit.2020.100801.

[40] D. Valero-Carreras, J. Alcaraz, and M. Landete, “Comparing two SVM models through different metrics based on the confusion matrix,” *Comput Oper Res*, vol. 152, Apr. 2023, doi: 10.1016/j.cor.2022.106131.

[41] M. A. Khder, “Web scraping or web crawling: State of art, techniques, approaches and application,” *International Journal of Advances in Soft Computing and its Applications*, vol. 13, no. 3, pp. 144–168, 2021, doi: 10.15849/ijasca.211128.11.

[42] I. Kolyshkina and S. Simoff, “Interpretability of Machine Learning Solutions in Public Healthcare: The CRISP-ML Approach,” *Front Big Data*, vol. 4, May 2021, doi: 10.3389/fdata.2021.660206.

[43] S. Studer *et al.*, “Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,” *Mach Learn Knowl Extr*, vol. 3, no. 2, pp. 392–413, Jun. 2021, doi: 10.3390/make3020020.

[44] S. Gholizadeh, “Top Popular Python Libraries in Research,” *Journal of Robotics and Automation Research*, vol. 3, no. 2, pp. 142–145, 2022, Accessed: Feb. 03, 2025. [Online]. Available: https://doi.org/10.22541/au.164580055.55493761/v1

[45] A. C. . Müller, Sarah. Guido, and Kristian. Rother, *Einführung in Machine Learning mit Python : Praxiswissen Data Science*. Heidelberg, Germany: dpunkt.verlag GmbH, 2017.

[46] M. N. Rao, “A Comparative Analysis of Deep Learning Frameworks and Libraries,” *International Journal of Intelligent Systems and Applications in Engineering IJISAE*, vol. 2023, no. 2s, pp. 337–342, 2023, Accessed: Feb. 03, 2025. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/2707

[47] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” Jan. 2012, [Online]. Available: http://arxiv.org/abs/1201.0490

[48] M. Wang and F. Hu, “The application of nltk library for python natural language processing in corpus research,” *Theory and Practice in Language Studies*, vol. 11, no. 9, pp. 1041–1049, Sep. 2021, doi: 10.17507/tpls.1109.09.

[49] E. J. Rifano, Abd. C. Fauzan, A. Makhi, E. Nadya, Z. Nasikin, and F. N. Putra, “Text Summarization Menggunakan Library Natural Language Toolkit (NLTK) Berbasis Pemrograman Python,” *ILKOMNIKA: Journal of Computer Science and Applied Informatics*, vol. 2, no. 1, pp. 8–17, Apr. 2020, doi: 10.28926/ilkomnika.v2i1.32.

[50] C. and C. C. and G. L. and J. E. and L. B. and P. S. and S. T. Cappi, “Dataset Definition Standard (DDS),” *arXiv preprint arXiv:2101.03020*, Jun. 2021, doi: 10.48550/arXiv.2101.03020.

[51] W. Vallejo, C. Díaz-Uribe, and C. Fajardo, “Google Colab and Virtual Simulations: Practical e-Learning Tools to Support the Teaching of Thermodynamics and to Introduce Coding to Students,” *ACS Omega*, vol. 7, no. 8, pp. 7421–7429, Mar. 2022, doi: 10.1021/acsomega.2c00362.

[52] S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,” Apr. 01, 2020, *MDPI AG*. doi: 10.3390/info11040193.

[53] P. Kode Program Dan Simulasi, “Pembuatan Kode Program Dan Simulasi Skema Masakan ACD Menggunakan Jupyter Notebook Pada Pemrograman Python Anaconda,” Politeknik LPP Yogyakarta, 2023. Accessed: Feb. 03, 2025. [Online]. Available: https://repository.polteklpp.ac.id/id/eprint/3812

[54] R. R. Rajalaxmi, L. V. Narasimha Prasad, B. Janakiramaiah, C. S. Pavankumar, N. Neelima, and V. E. Sathishkumar, “Optimizing Hyperparameters and Performance Analysis of LSTM Model in

Detecting Fake News on Social media," *ACM Transactions on Asian and Low-Resource Language Information Processing*, Mar. 2022, doi: 10.1145/3511897.

[55] I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, "Fake News Detection Using Machine Learning Ensemble Methods," *Complexity*, vol. 2020, 2020, doi: 10.1155/2020/8885861.

[56] A. John and S. Journals, "Fake News Detection Using N-Gram Analysis and Machine Learning Algorithms," 2022, doi: 10.37591/JoMCCMN.

[57] M. Alsafadi, "Stance Classification for Fake News Detection with Machine Learning," *Technology, Engineering & Mathematics (EPSTEM)*, vol. 22, 2023, [Online]. Available: www.isres.org

[58] K. Martínez-Gallego, A. M. Álvarez-Ortiz, and J. D. Arias-Londoño, "Fake News Detection in Spanish Using Deep Learning Techniques," Oct. 2021, [Online]. Available: http://arxiv.org/abs/2110.06461

[59] S. Basharat, S. Afzal, A. M. Bamhdi, S. Khurshid, and M. Chachoo, "Predicting and Mitigating the Effect of Skewness on Credibility Assessment of Social Media Content Using Machine Learning: A Twitter Case Study," *International Journal of Computer Theory and Engineering*, vol. 15, no. 3, pp. 101–110, Aug. 2023, doi: 10.7763/IJCTE.2023.V15.1338.

[60] N. J. Mafla Checa, "Identificación automática de noticias falsas en español utilizando técnicas de minería de datos y procesamiento del lenguaje natural.," Escuela Politécnica Nacional, Quito, 2021. Accessed: Feb. 03, 2025. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/21606

11. Anexos

Anexo 1. Correo electrónico enviado al experto para la realización de la variante de la técnica de investigación Delphi.

Figura 50. Email enviado al experto para realización de la técnica Delphi con variación.

Estimado,

Santiago Emanuel Tene Castillo

Fecha: 03-07-2024

Asunto: validación de preguntas

Por informar que he revisado detalladamente el conjunto de preguntas formuladas a través de la variación del método Delphi. Tras una cuidadosa evaluación, puedo confirmar que las preguntas cumplen con los estándares de rigor y relevancia exigidos para el proyecto en cuestión. Por lo tanto, me dispongo a firmar la validación de estas.

Agradezco la oportunidad de colaborar en este importante proceso y quedo a su disposición para cualquier otra consulta o requerimiento adicional.

Firma. Ing. Luis Antonio Chamba Eras

Figura 51. Evaluación de las preguntas a través de la variación de Delphi.

Anexo 2. Entrevista y validación mediante una variante de la técnica de investigación Delphi.

Elaborado por: Santiago Emanuel Tene Castillo

email: santiago.tene@unl.edu.ec

Revisado y validado por: Ing. Luis Antonio Chamba Eras

email: lachamba@unl.edu.ec

Técnica de investigación: Variación método Delphi.

El uso de esta variación del método Delphi, con la validación de las respuestas por parte de un experto, contribuye a la validez y robustez de la problemática del proyecto. En esta variación, las preguntas son formuladas y validadas por un experto en el campo, en este caso, el Ing. Luis Chamba Eras. El proceso comienza con la recopilación de preguntadas respaldadas por una revisión exhaustiva de literatura, asegurando que sean relevantes y bien fundamentadas. Posteriormente, esas preguntas son validadas por el experto para garantizar su pertinencia y adecuación al tema de investigación.

Preguntas:

¿Qué problemas pueden surgir si no se utilizan técnicas de optimización en modelos de machine learning?

La falta de optimización en modelos de machine learning puede resultar en varios problemas considerables. El uso ineficiente de recursos computacionales, así como los largos tiempos de entrenamiento con un uso excesivo de memoria y poder de procesamiento, pueden afectar gravemente al rendimiento del modelo, específicamente en términos de precisión y eficiencia. Como resultado, pueden existir predicciones inexactas y modelos nada robustos [2].

¿Cuáles son los desafíos asociados con el uso del descenso de gradiente en la detección de noticias falsas?

Los desafíos incluyen la elección del tamaño de paso (learning rate), la posibilidad de quedarse atrapado en mínimos locales y la necesidad de un número significativo de iteraciones para converger a la solución óptima [1].

¿Qué limita actualmente la precisión alcanzable por los modelos de regresión logística en la detección de noticias falsas al aplicar algoritmos de optimización?

La precisión que puede alcanzar la regresión logística en la detección de noticias falsas está condicionada tanto por las características intrínsecas del modelo como por la metodología de optimización empleada. Aunque el uso de algoritmos como el Gradiente Descendente Estocástico (SGD) ha mostrado mejoras, según el estudio de Faizan Mian (2024), la falta de experimentación con otros métodos como AdaGrad, Adam, y RMSProp puede estar limitando el potencial de mejora en la precisión y eficiencia del modelo. Por lo tanto, es crucial investigar más a fondo la influencia de diversos algoritmos de optimización para determinar si pueden contribuir a un aumento considerable de la precisión [6].

¿Por qué es crucial utilizar técnicas de optimización basadas en el algoritmo de gradiente descendente en modelos de machine learning?

Usar estas técnicas de optimización es de vital importancia en modelos de machine learning porque ayudan a encontrar los valores óptimos de los parámetros del modelo de una manera eficiente, trayendo consigo una mejora general del modelo y una reducción en el error de predicción. No es práctico manejar grandes conjuntos de datos o modelos complejos sin una adecuada optimización [2].

¿Qué limitaciones existen en el manejo de grandes volúmenes de datos sin Gradiente Descendente?

El gradiente descendente es especialmente útil para el manejo de grandes volúmenes de datos de manera eficiente. No implementar este método de optimización puede limitar la escalabilidad de algoritmos de clasificación, teniendo como resultado un procesamiento no eficiente de grandes conjuntos de datos y así afectar el rendimiento general del modelo [5].

¿Los algoritmos de optimización ayudan a aumentar la precisión de los modelos de machine learning?

La relación que existe entre la precisión y la optimización de un modelo de machine learning es fundamental. Sin una adecuada optimización, los modelos, especialmente los de clasificación, pueden presentar dificultades para alcanzar una precisión alta. Debido a consecuencia de diversos problemas como una convergencia lenta o el sobreajuste del modelo [3].

¿Por qué la falta de un método de optimización efectivo puede llevar a un entrenamiento ineficiente en modelos de machine learning?
La falta de un método de optimización puede llevar a un entrenamiento ineficiente en modelos de machine learning debido a las funciones de costo asociado a los modelos, estos suelen ser no convexas, es decir, que contienen múltiples mínimos locales. Esto resultará en un entrenamiento que no converge adecuadamente o que lo hace muy lentamente [4].

¿Qué impacto tiene la falta de experimentación con algoritmos de optimización en la precisión de algoritmos de clasificación?

Según Sigrist, F. (2021) en la página 4-5 discute la falta de experimentación con los algoritmos de optimización, lo que resulta en una precisión limitada del modelo. Es difícil determinar el impacto y la mejora de la precisión del clasificador si es que no existe una aplicación y pruebas de estos métodos. Esto puede deberse a que diferentes algoritmos de optimización pueden tener distintos efectos en la convergencia y la capacidad del modelo para clasificar correctamente [7].

[1] E. V. Sai Raja, B. L V S Aditya, and S. N. Mohanty, "Fake Profile Detection Using Logistic Regression and Gradient Descent Algorithm on Online Social Networks", *EAI Endorsed Scal Inf Syst*, vol. 11, no. 1, Nov. 2023.

[2] Li Yang and Abdallah Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295-316, 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.07.061.

[3] D. Soydaner, "A comparison of optimization algorithms for deep learning," International Journal of Pattern Recognition and Artificial Intelligence, vol. 34, no. 13, p. 2052013, 2020.

[4] D. Yi, J. Ahn, and S. Ji, "An effective optimization method for machine learning based on ADAM," *Applied Sciences*, vol. 10, no. 3, p. 1073, 2020.

[5] A. S. Assiri, S. Nazir, y S. A. Velastin, "Breast tumor classification using an ensemble machine learning method," *Journal of Imaging*, vol. 6, no. 6, p. 39, 2020.

[6] F. Mian, "Enhancing Integrated Logistic Support through Machine Learning-Driven Optimization: Predicting component category based on part number," 2024.

[7] F. Sigrist, "Gradient and Newton boosting for classification and regression," *Expert Systems With Applications*, vol. 167, p. 114080, 2021.

Anexo 3. Diagrama de la solución planteada para el trabajo de integración curricular (TIC).

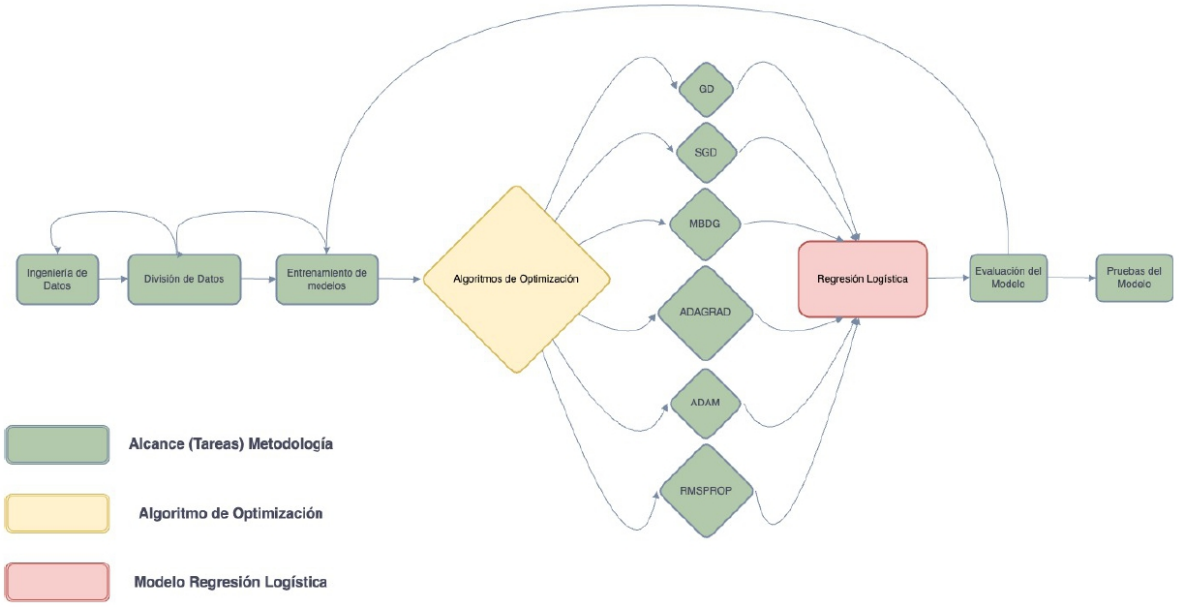


Figura 52. Bosquejo solución al trabajo de integración curricular.

Anexo 4. Evaluación del desempeño del modelo regresión logística y los algoritmos de optimización Gradiente descendente (GD), Gradiente descendente estocástico (SGD), Gradiente descendente por mini lotes (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam), Propagación de la Raíz del Promedio Cuadrática (RMSProp).

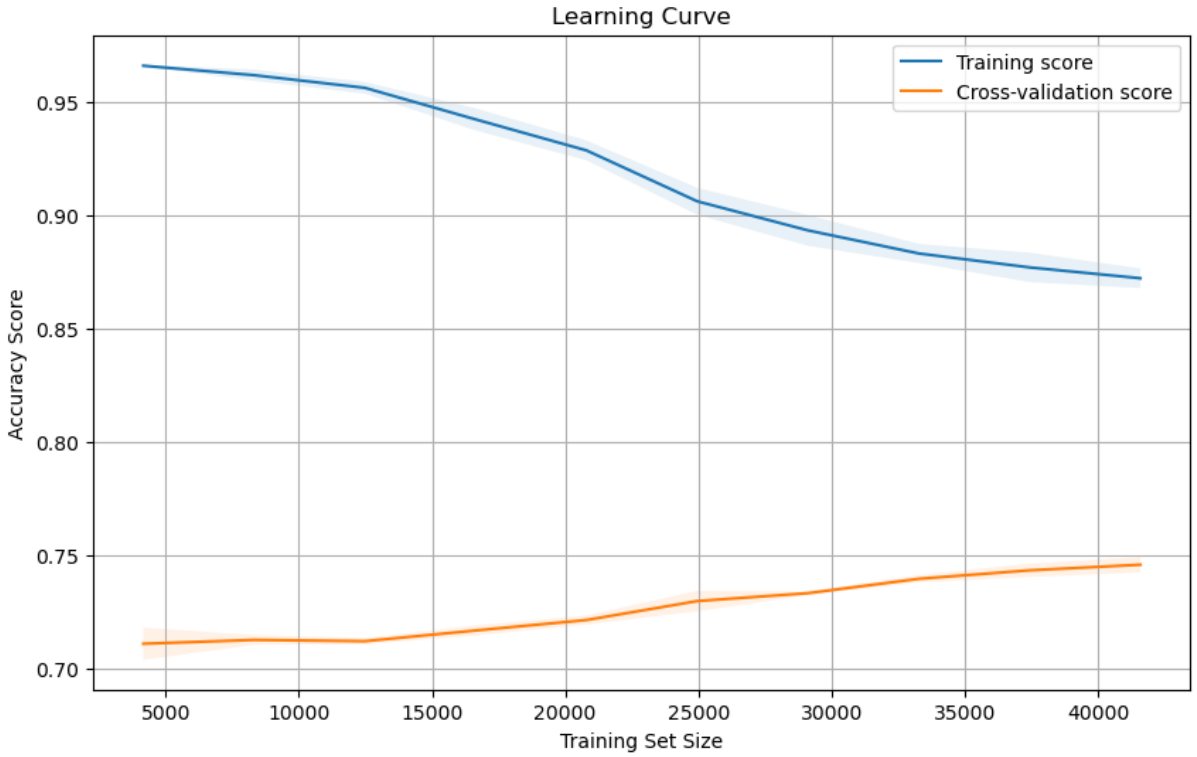


Figura 53. Curva de entrenamiento del modelo regresión logística para entrenamiento y validación.

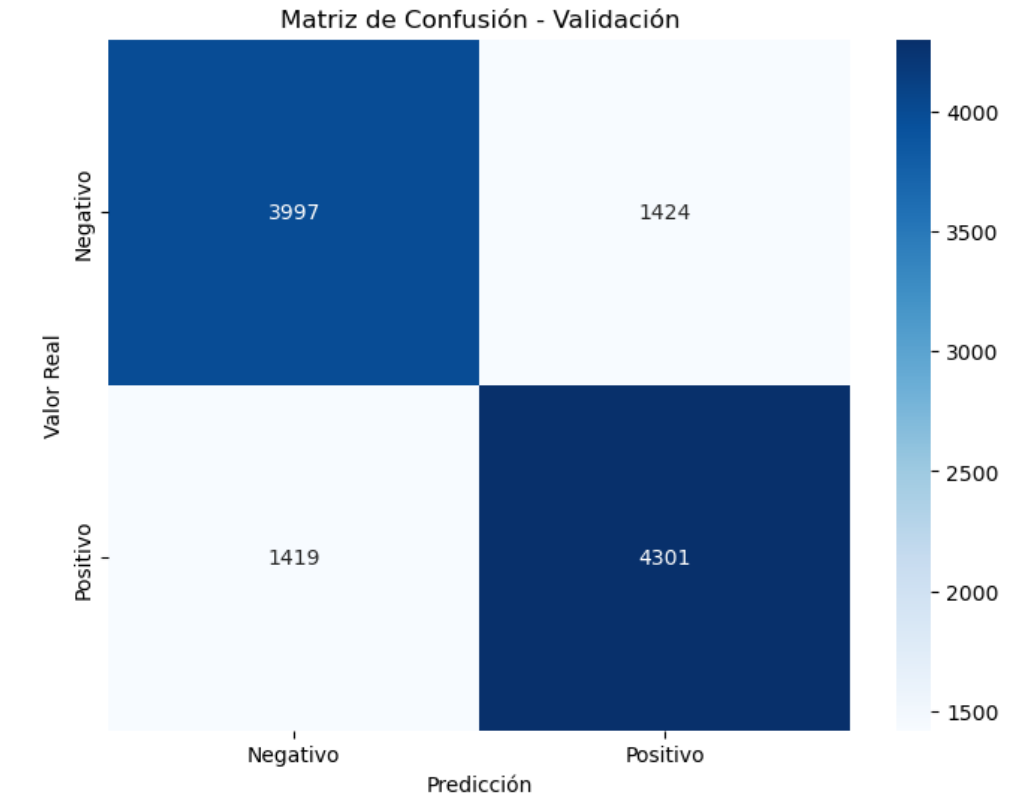


Figura 54. Matriz de confusión del modelo regresión logística durante la validación del modelo.

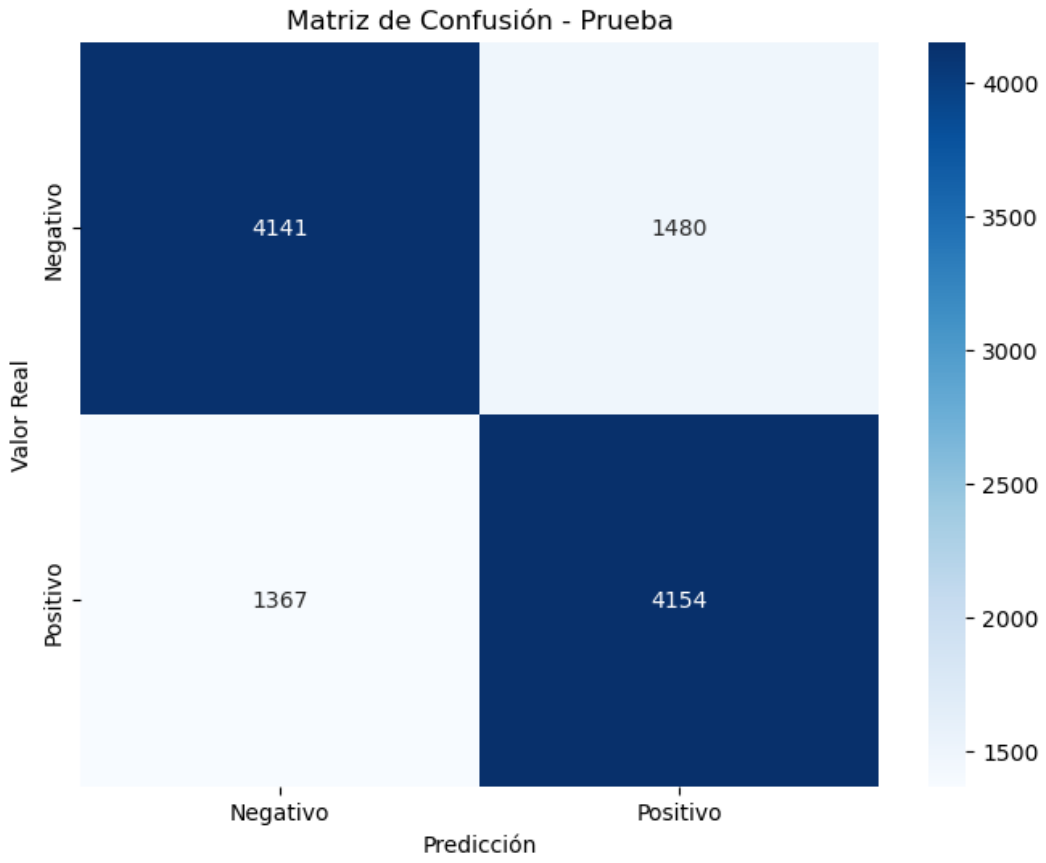


Figura 55. Matriz de confusión del modelo regresión logística durante la evaluación del modelo.

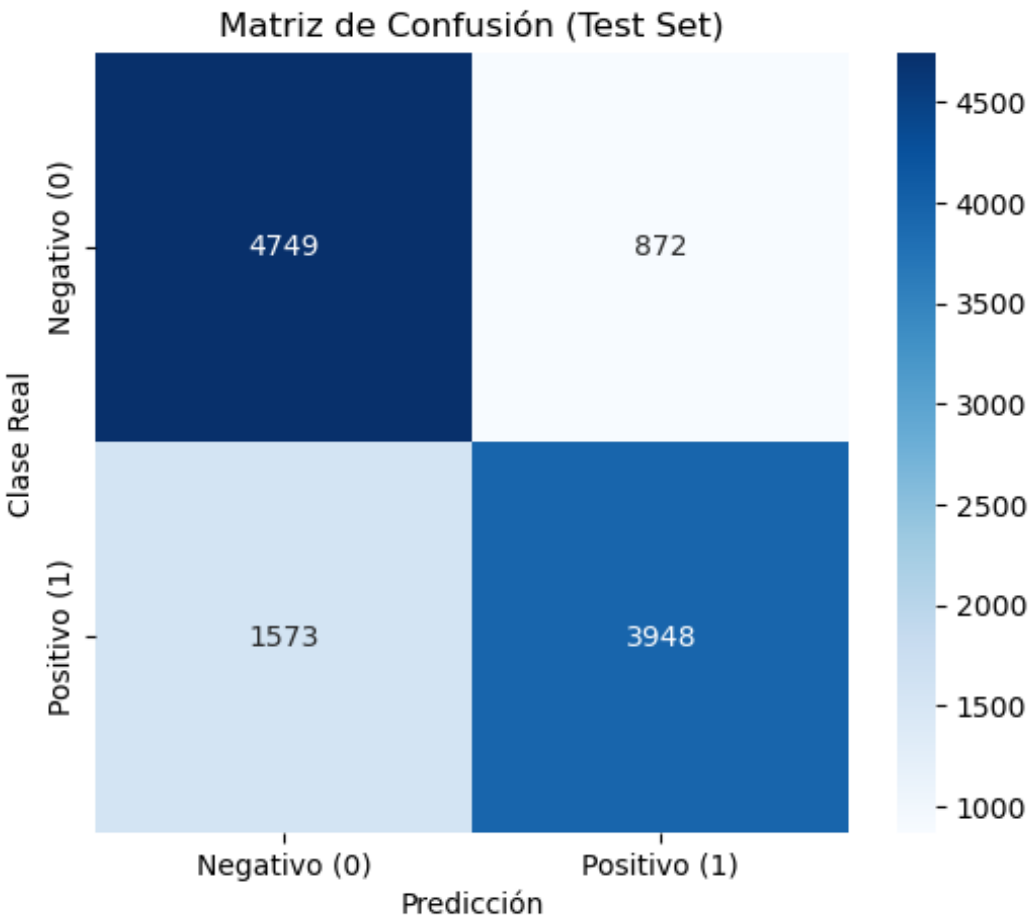


Figura 56. Matriz de confusión del optimizado gradiente descendente (GD)

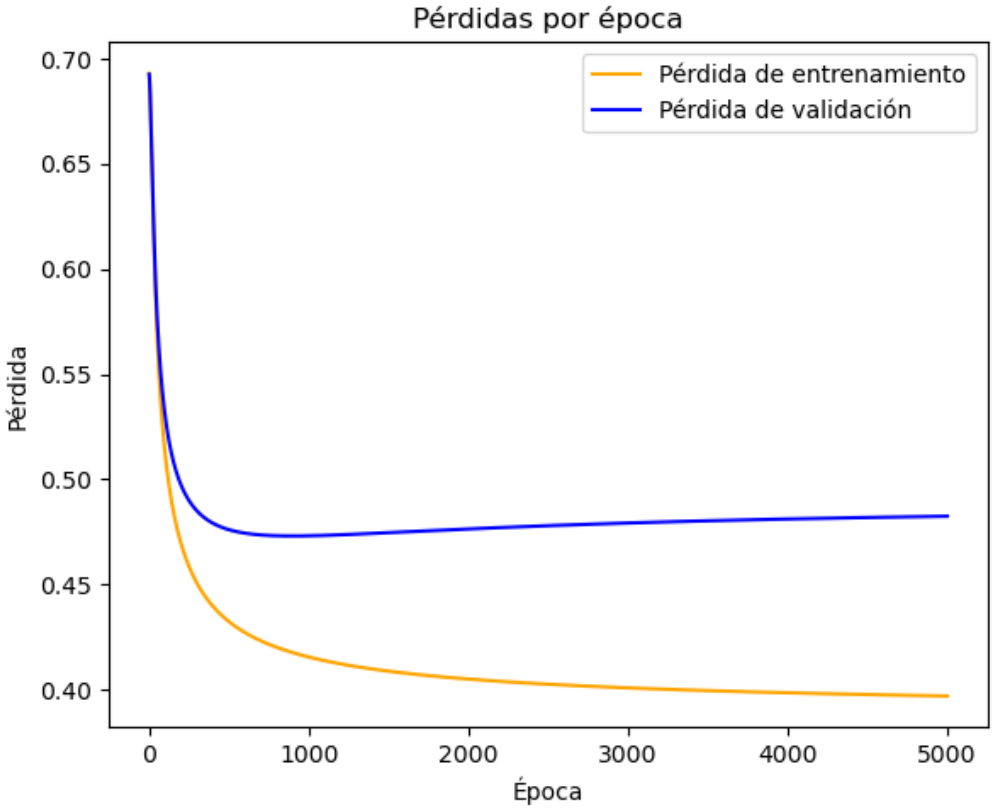


Figura 57. Gráfico de pérdida por épocas del optimizador gradiente descendente (GD).

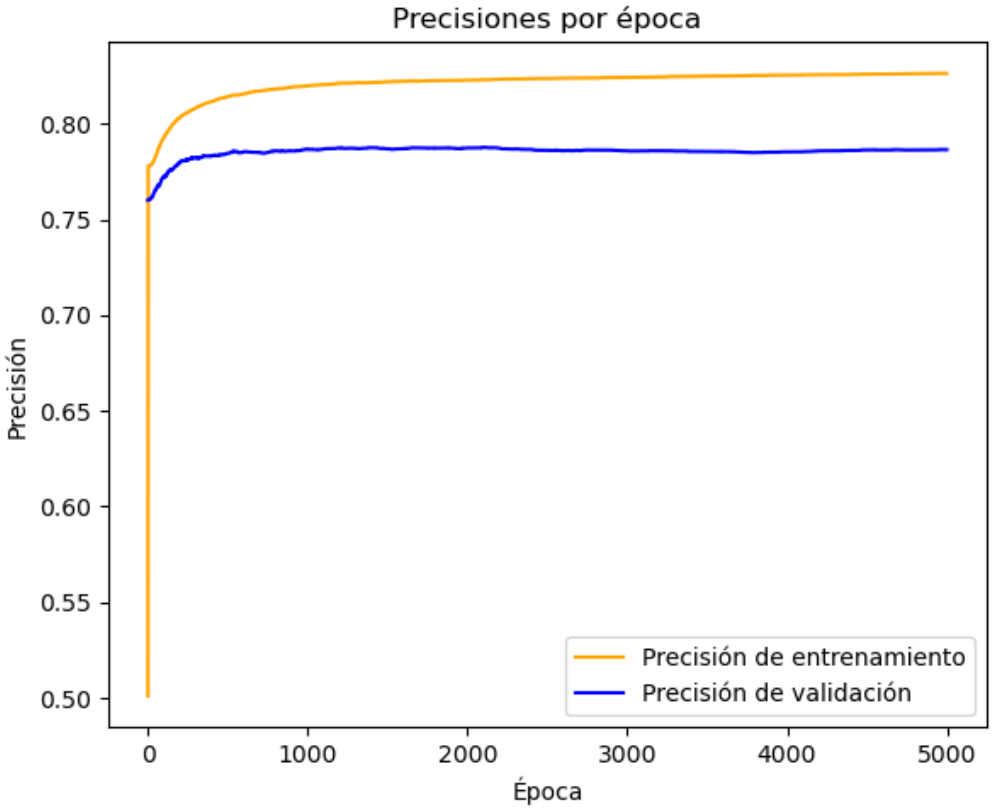


Figura 58. Gráfico de precisión por épocas del optimizador gradiente descendente (GD).

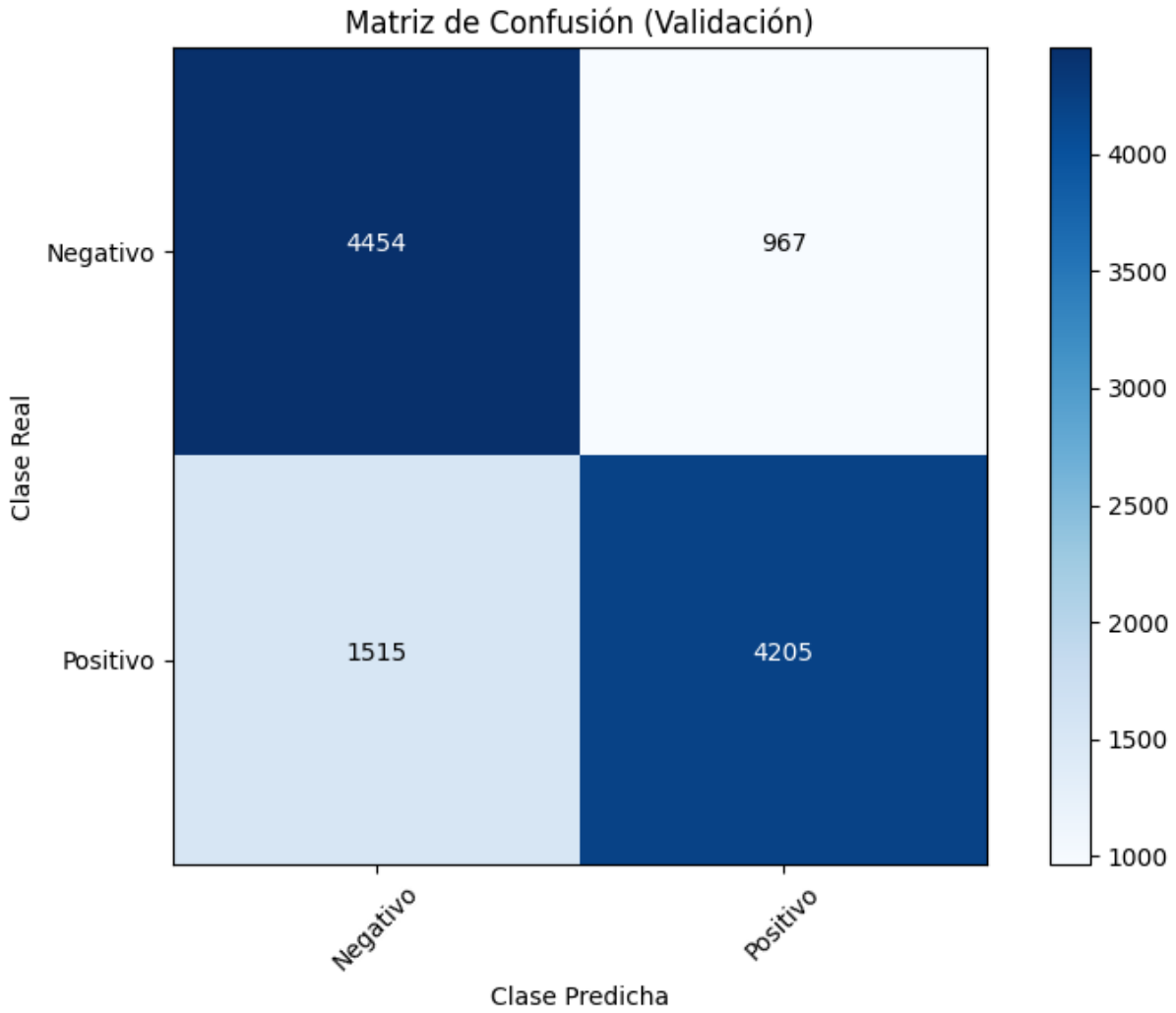


Figura 59. Matriz de confusión del modelo LR-SGD durante la validación.

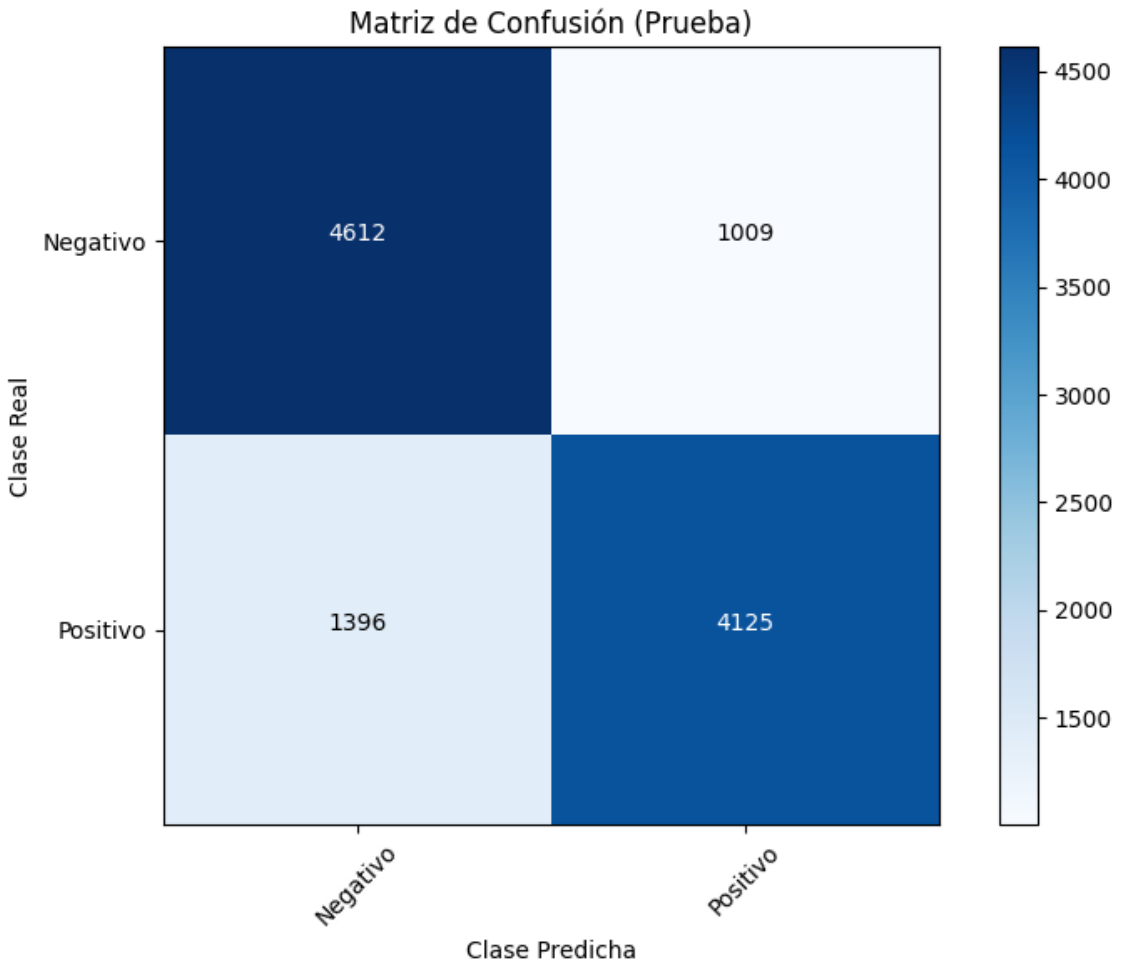


Figura 60. Matriz de confusión del modelo LR-SGD durante la evaluación.

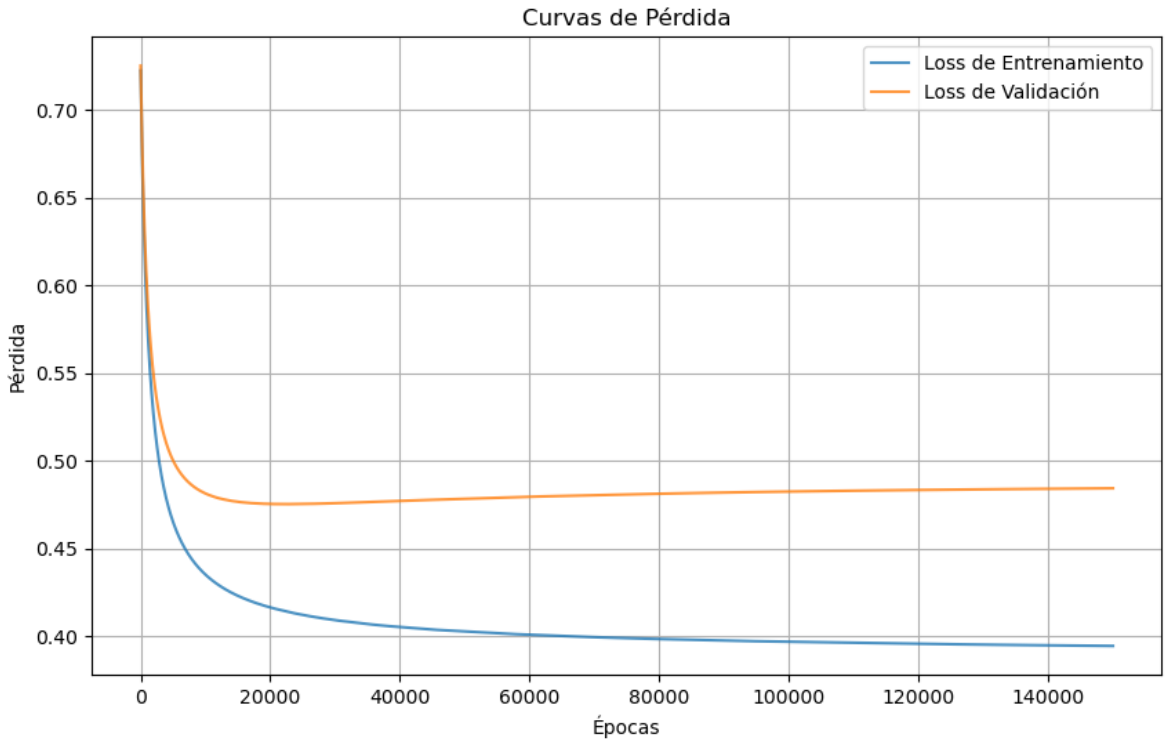


Figura 61. Gráfico de curvas de pérdida del optimizador gradiente descendente estocástico (SGD) en entrenamiento y validación.

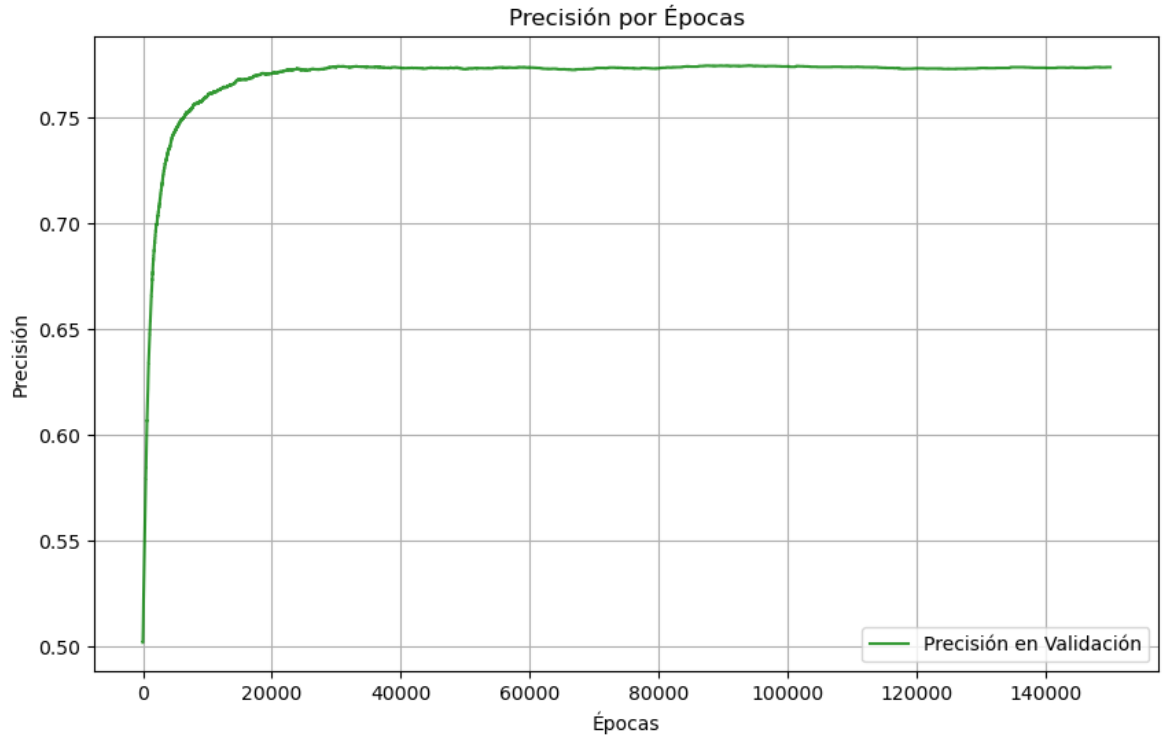


Figura 62. Gráfico de precisión por épocas del optimizador gradiente descendente estocástico (SGD) en validación.

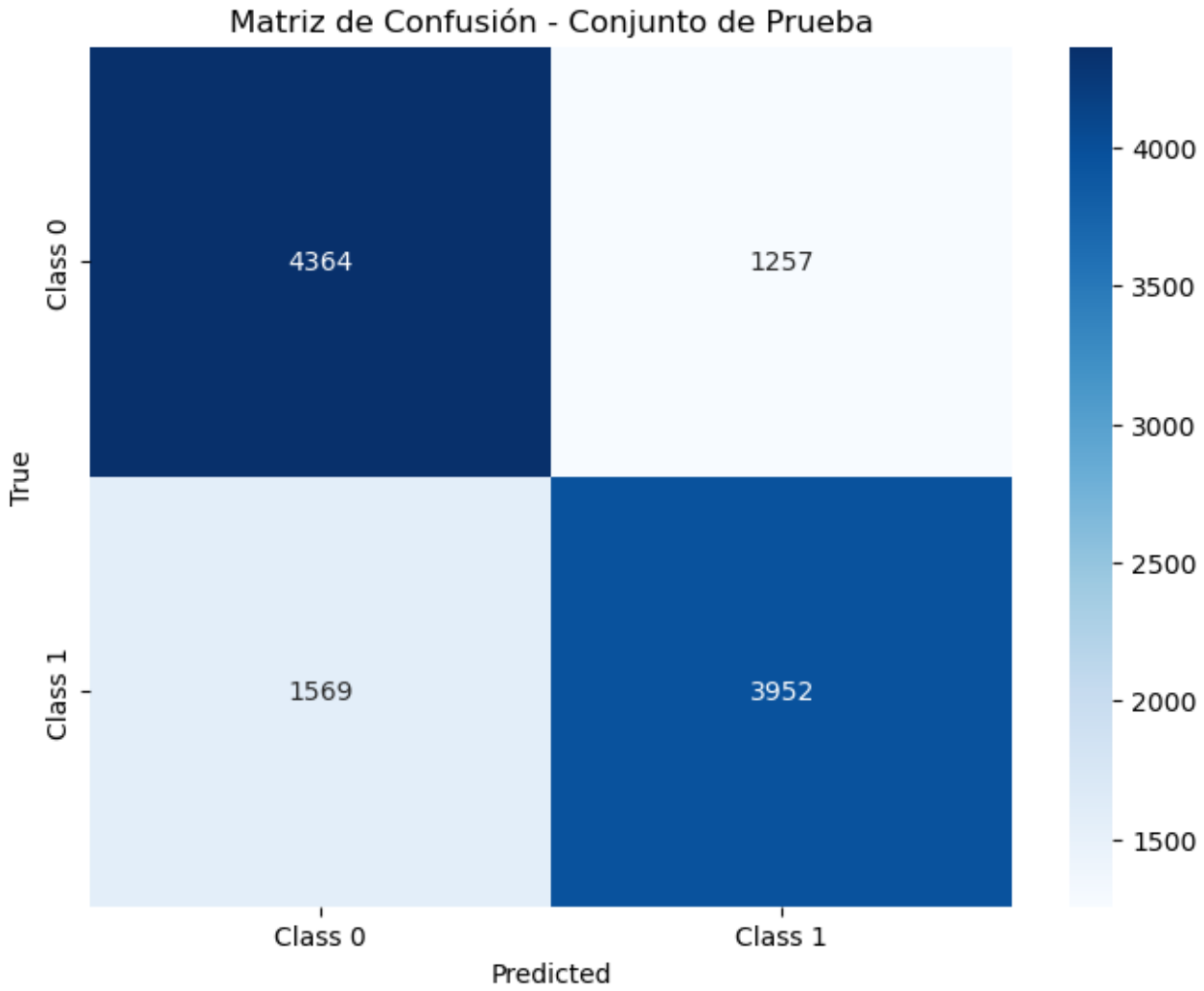


Figura 63. Matriz de confusión del optimizador gradiente descendente por mini lotes (MBGD) sobre el dataset de prueba.

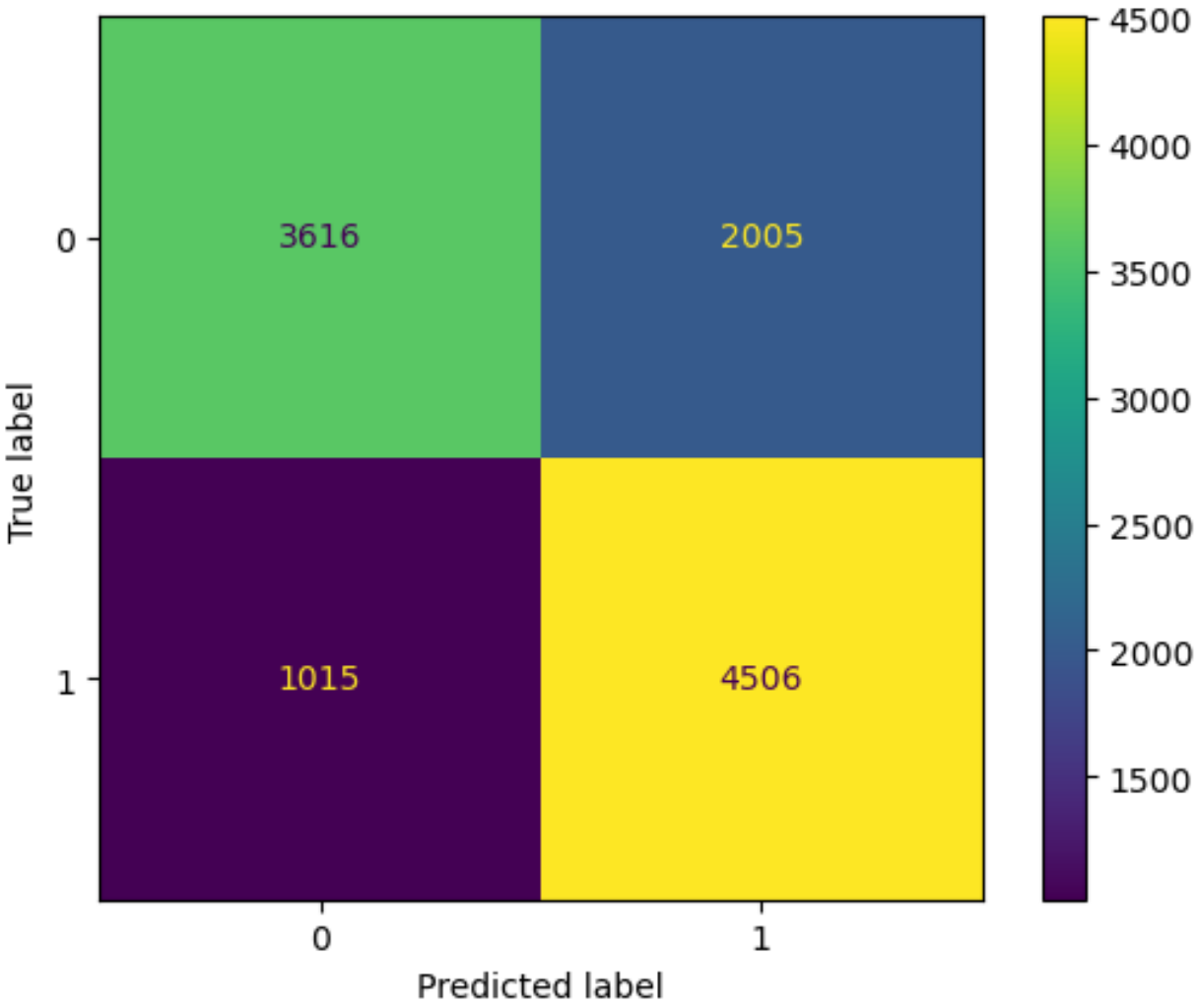


Figura 64. Matriz de confusión del optimizador adaptativo AdaGrad sobre el dataset de prueba.

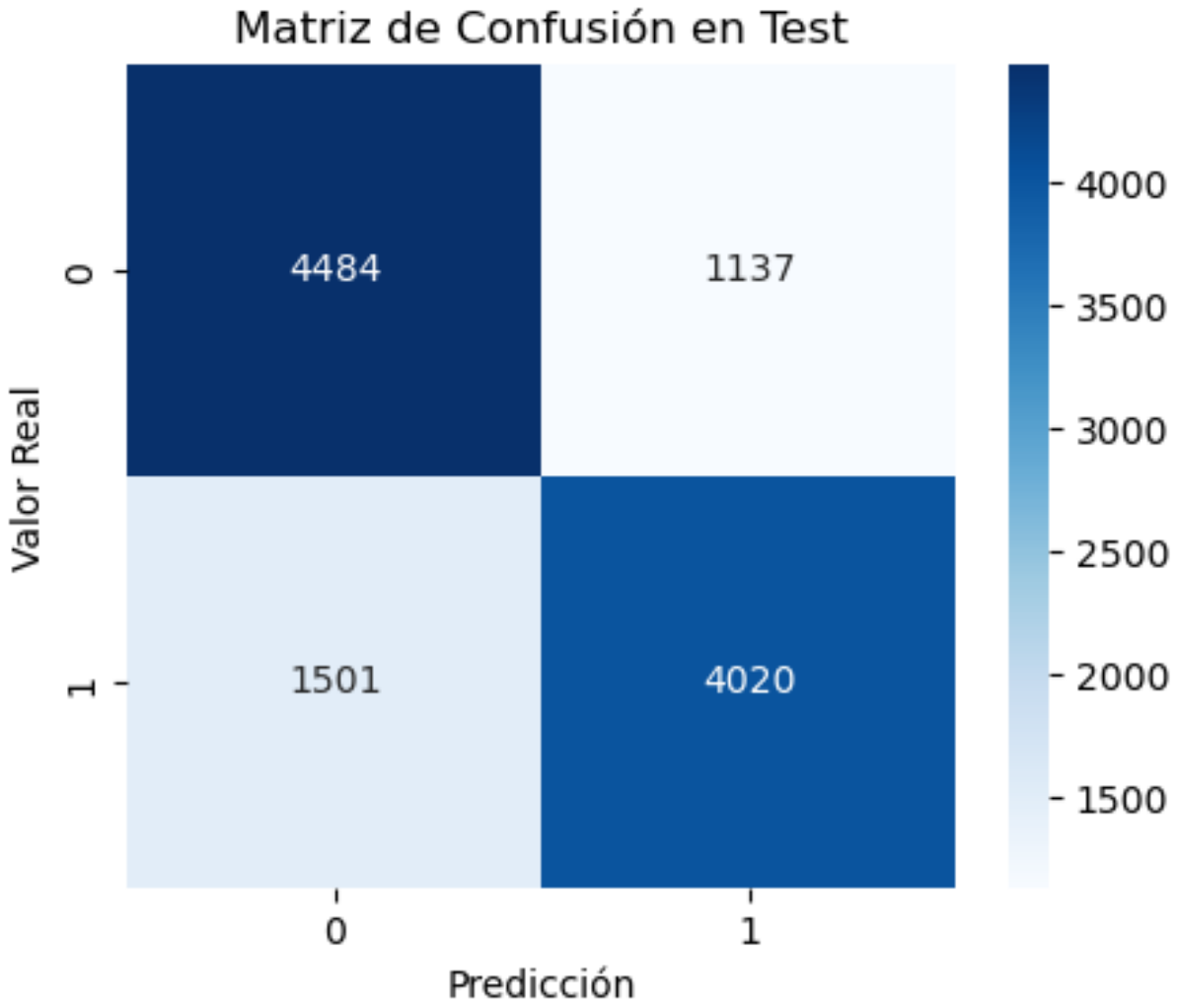


Figura 65. Matriz de confusión del optimizador adaptativo Adam sobre el dataset de prueba.

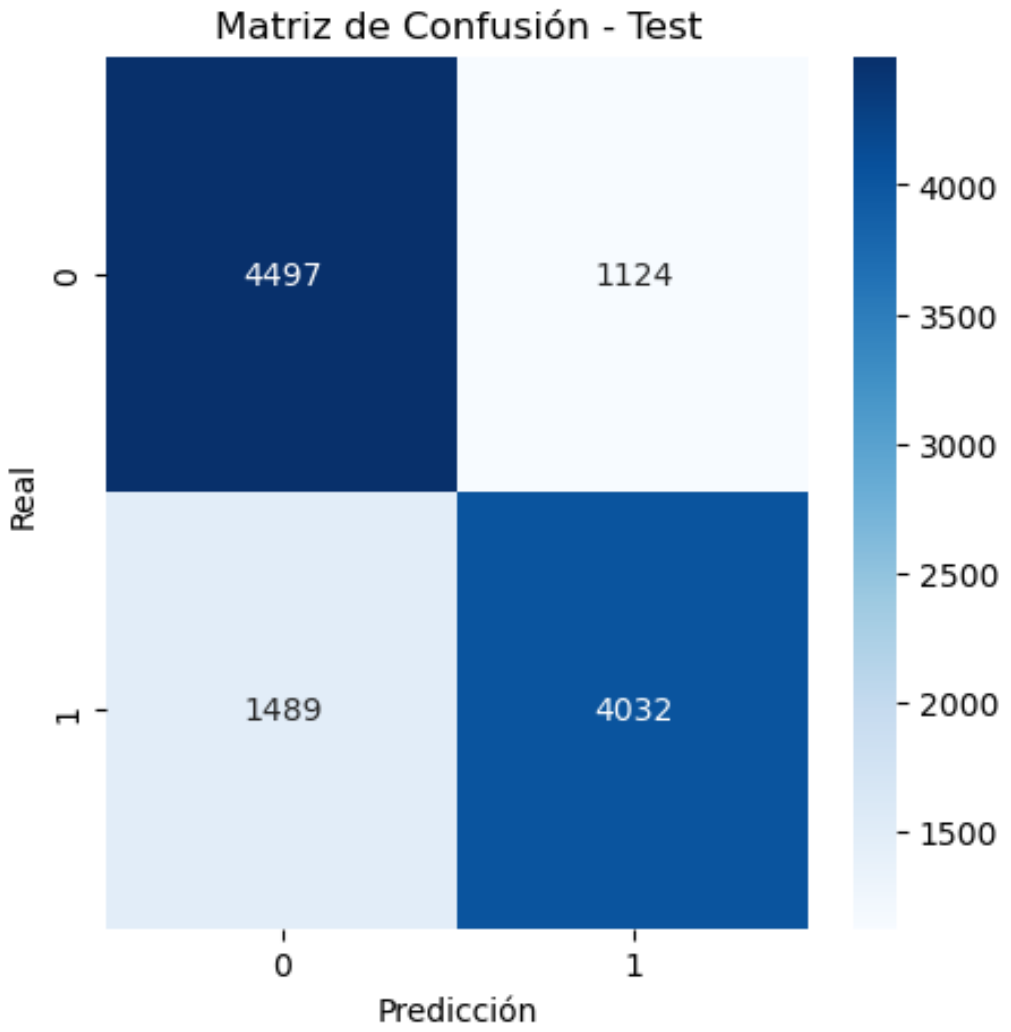


Figura 66. Matriz de confusión del optimizador RMSProp sobre el dataset pruebas.

Anexo 5. Repositorio del proyecto

Se creó un repositorio con la finalidad de permitir a la comunidad interesada reproducir los experimentos de optimización en modelos básicos, como la regresión logística. De este modo, se puede verificar la forma en que la elección y ajuste de hiperparámetros contribuyen a mejorar la precisión y demás métricas de la matriz de confusión.

A continuación, se presenta la URL del repositorio:

Link: Repositorio: Optimización de la Precisión en la Detección de Noticias Falsas de Política

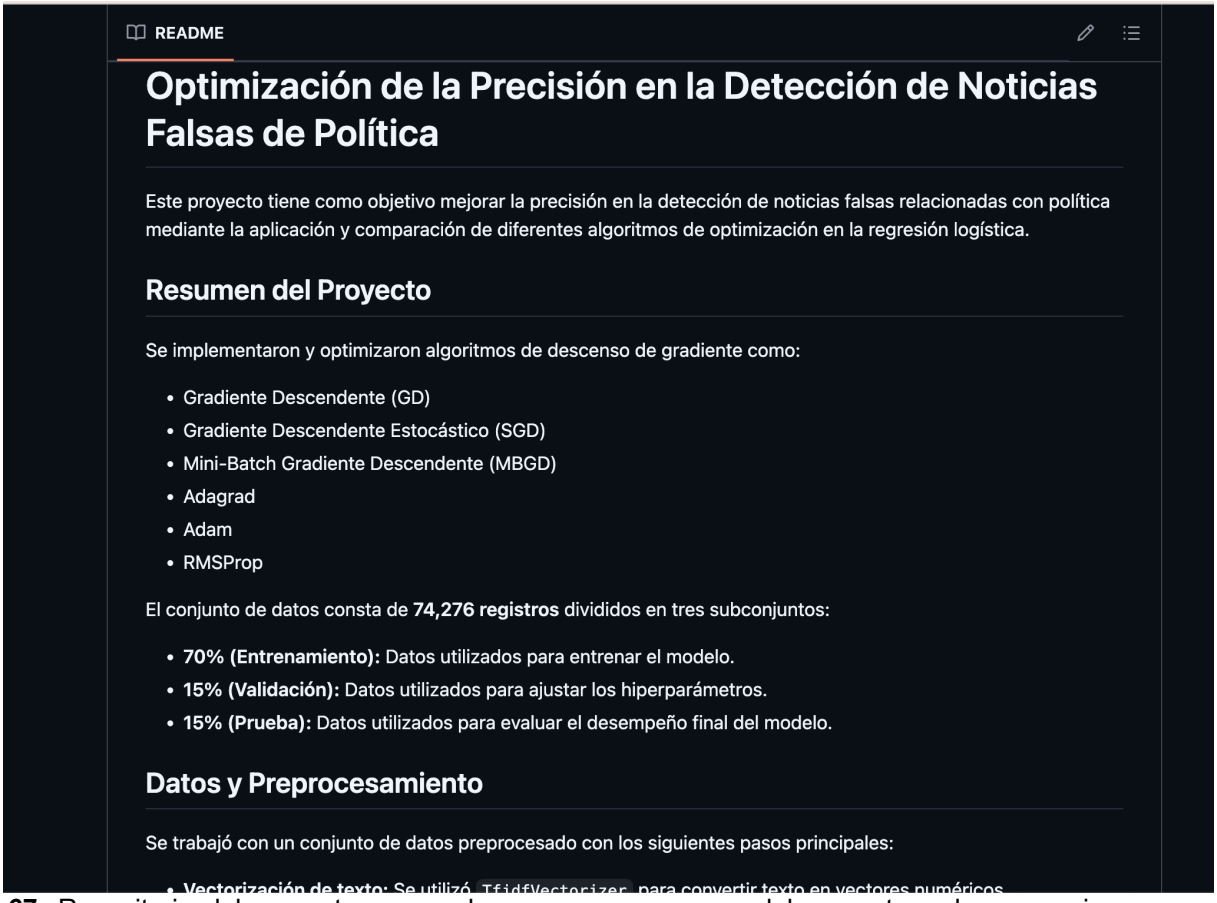


Figura 67. Repositorio del proyecto con un breve resumen acerca del proyecto, y lo necesario para realizar la reproducibilidad del proyecto.

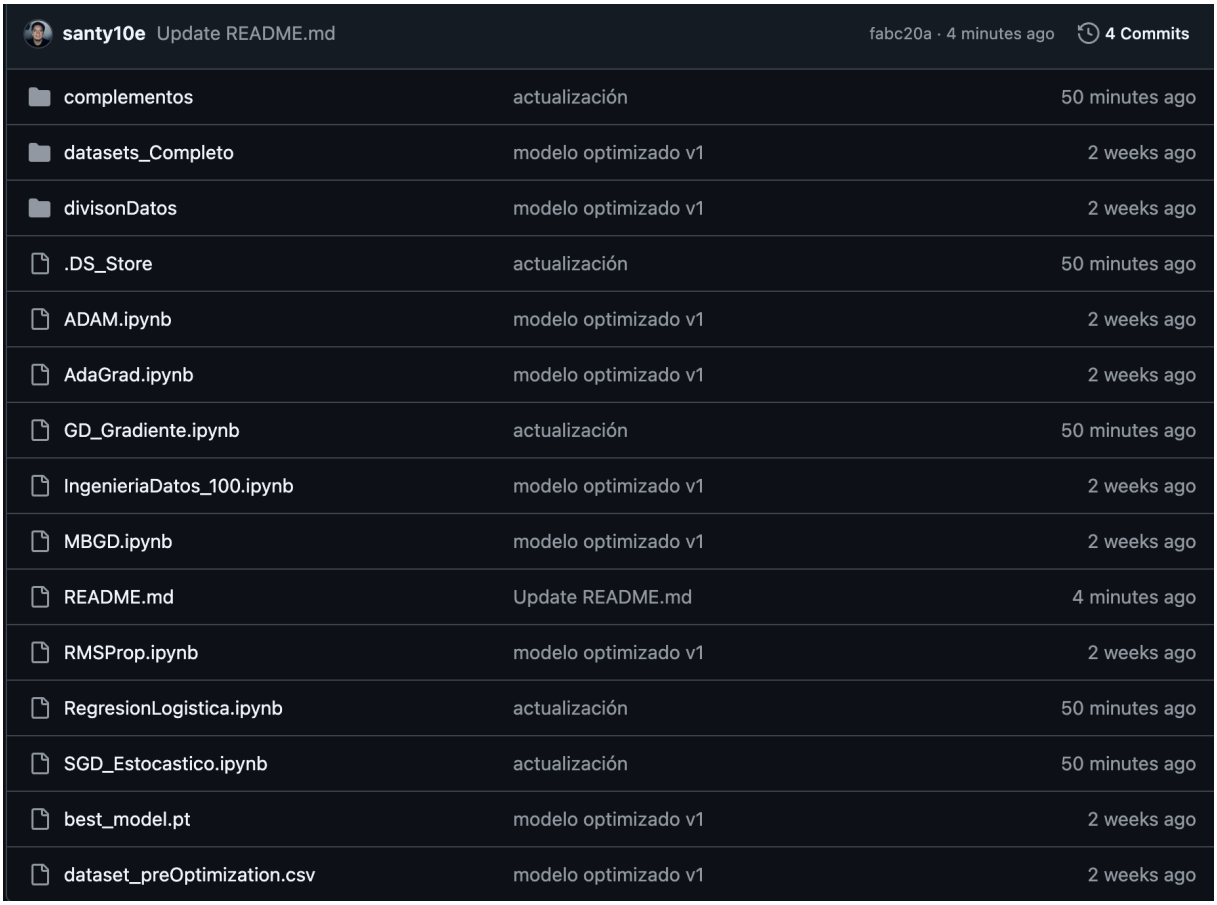


Figura 68. Estructura general del proyecto.

Anexo 6. Proyecto de investigación de integración curricular.

Universidad Nacional de Loja

Facultad de Energía, las Industrias y los Recursos Naturales no Renovables

CARRERA DE COMPUTACIÓN

Optimización de la Precisión en la Detección de Noticias Falsas de política en español mediante la Aplicación de Algoritmos de Optimización en la Regresión Logística

Itinerario: Sistemas Inteligentes

PROYECTO DE INVESTIGACIÓN DE INTEGRACIÓN CURRICULAR.

Autor:

◇ ORCID, Santiago E. Tene-Castillo, santiago.tene@unl.edu.ec

Tutor:

- Pablo F. Ordoñez-Ordoñez, Mg.Sc.

Cotutor especialista:

- Luis Antonio Chamba Eras, Ph.D.

Carrera de Ingeniería en Sistemas / Computación

LOJA - ECUADOR

2020

Índice de figuras

Índice de tablas

Optimización de la Precisión en la Detección de Noticias Falsas de política en español mediante la Aplicación de Algoritmos de Optimización en la Regresión Logística

Itinerario: Sistemas Inteligentes

---

1. Problema de investigación

1.1. Situación Problemática

La regresión logística es un algoritmo de Machine Learning (ML) con gran potencial para la detección de noticias falsas de política en español. Sin embargo, su aplicación en este campo aún no ha sido suficientemente optimizada. La falta de experimentación con algoritmos de optimización Gradiente Descendente (GD), Gradiente Descendente Estocástico (SGD), Gradiente descendente por Mini-Lote (MBGD), Algoritmo de Gradiente Adaptativo (AdaGrad), Estimación de Momento Adaptativo (Adam) y Propagación de la Raíz del Promedio Cuadrática (RMSProp) dificulta determinar el impacto en la precisión del clasificador. Este problema de optimización, debido a la escasez de pruebas, resulta en una precisión limitada del modelo.

En este contexto, la importancia del algoritmo de optimización GD es esencial para el entrenamiento de modelos de ML. El objetivo de estos métodos es minimizar la función de pérdida realizando un ajuste iterativo en los parámetros del modelo, trasladándose hacia la dirección del gradiente negativo de la función de perdida con respecto a los parámetros. Se conocen variantes del algoritmo de optimización, GD los cuales son SGD, MBGD, y otros métodos adaptativos como Adam, AdaGrad y RMSProp. [1]

El estudio Sai-Raja et al. (2023) abarca la problemática en la detección de spam en las redes sociales en línea, para lo cual se evalúa la precisión del modelo Regresión Logística en múltiples conjuntos de datos, los resultados que se pueden apreciar son los siguientes. El modelo Regresión Logística sin GD obtuvo una precisión de 80% mientras que el modelo con GD obtiene una precisión del 95% esto en la detección de perfiles falsos. Entonces, gracias a este estudio podemos darnos la idea de que el modelo de Regresión Logística por sí solo no arroja buenos resultados a la hora de realizar la predicción porque su precisión es relativamente baja, mientras que, el modelo junto con GD es eficiente, ya que tenemos una reducción en las funciones de costo. [2].

Además, RR Rajalaxmi (2022) en su estudio refiere a que existen trabajos que tienen limitaciones presentando un modelo ineficiente y con poca precisión, por lo cual se propone un modelo optimizado para la detección de noticias falsas usando memoria a largo plazo (LSTM) para realizar la clasificación. Los resultados coinciden en que el modelo sin optimización tiene menor precisión, se realizó una evaluación en múltiples conjuntos de datos y como resultado tenemos que para el conjunto DS-1 la precisión inicial fue de 47.19%, mientras que optimizado la precisión aumentó al 54.48%. Los algoritmos de optimización utilizados para el ajuste de los hiperparámetros fueron: SGD, AdaGrad, RMSProp, Adadelta, Adamax, Adam, Nadam. Para este caso el método óptimo fue

Adam [3].

Cabe destacar la investigación de Aruna,D.O. et al (2024), donde se utiliza el modelo de Regresión Logística para identificación y clasificación de artículos o declaraciones de noticias falsas como genuinas o falsas, se realiza un entrenamiento mediante la técnica de optimización GD para poner minimizar la pérdida; se demuestra la necesidad de optimización del modelo Regresión Logística porque ayuda a la disminución en el error de predicción [4].

Por otro lado, el estudio realizado por Asha J & Meenakowshalya A.(2021) en la detección de noticias falsas mediante técnicas de extracción de características N-Gram podemos observar que en estos escenarios es donde los algoritmos de optimización toman protagonismo porque se aplican para ajustar el rendimiento de los modelos aumentando la precisión. Se utilizan seis algoritmos de ML los cuales son: Máquinas de Vectores de Soporte (SVM), SVM Lineal, K-Nearest Neighbor (KNN), SGD, Árboles de Decisión (DT) y Regresión Logística (LR) y como resultado podemos ver que el método de optimización SGD es el que se encuentra el mejor parámetro permitiendo una mejora en la precisión mediante la utilización de búsqueda de cuadrícula y búsqueda aleatoria; donde la precisión se vio mejorada en un aproximado de 0.8 en puntos porcentuales, teniendo un 94,2% de precisión después de la optimización [5].

De manera similar, Iftikhar Ahmad et al, estudia la desafiante tarea de clasificar automáticamente artículos textuales como noticias “falsas” o no. El estudio evalúa el modelo utilizando cuatro conjuntos de datos. El modelo de regresión logística obtiene una puntuación de precisión mínima del 88% y un promedio del 90% en las otras tres mediciones de conjuntos de datos. Después de que se aplicaron al modelo métodos de optimización que utilizan el algoritmo SGD para ajustar sus hiperparámetros, la precisión del modelo aumentó al 97% [6]. El presente autor, refiriéndose a Ahmed et al. Destaca que “el modelo de regresión logística siempre requiere un método de optimización como SGD para lograr la máxima precisión”.

En consonancia con estos hallazgos, Mafla-Checa (2021) realiza un estudio orientado a la detección de noticias falsas específicamente sobre temas de política ecuatoriana para lo cual realiza una evaluación de algoritmos de clasificación de ML en donde se tiene resultados alentadores, ya que logra detectar con un poco más del 90% la falsedad en las noticias [7]. Sin embargo, en una clasificación binaria, los modelos de ML en especial la Regresión Logística correctamente optimizados podría dar mejores resultados.

Para profundizar en el tema, se realizó una serie de preguntas a un experto, en este caso, dirigidas al ingeniero Luis Chamba Eras, quien afianzó la problemática. Las preguntas realizadas se encuentran en el **”anexo I”**. En los siguientes párrafos se describen cada una de las preguntas.

En relación con esto, es palpable como la escasez en la optimización resulta en recursos computaciones desperdiciados e ineficientes esto trae consigo tiempos de entrenamientos prolongados, uso de memoria y procesador exagerados resultando en una disminución en la precisión y lleva a tener predicciones no exactas, por lo tanto, los

modelos tienen poca robustez.

Asimismo, el modelo Regresión Logística y su precisión en la detección de noticias falsas de política en español se ve limitado de acuerdo a sus características propias y a su metodología de optimización. la limitada experimentación del modelo junto con técnicas de optimización como AdaGrad, Adam y RMSProp acotan la mejora de la precisión, por lo que existe la necesidad en la exploración de estos métodos para impulsar los resultados del modelo.

Además, aplicar métodos de optimización sustentada mediante algoritmos de optimización GD asegura la eficiencia del modelo en términos de ajuste de parámetros, teniendo como meta la mejora del rendimiento. Es esencialmente útil la implementación de estos algoritmos de optimización en el manejo de modelos complejos y volúmenes de datos mayores, el fin principal es minorar errores de predicción y como resultado optimizar los modelos de clasificación y en especial el modelo de Regresión Logística.

Siguiendo la línea de pensamiento; el ausentismo de una técnica de optimización tal como GD limita de alguna manera la escalabilidad de los modelos de clasificación, afectando directamente su eficiencia en procesar grandes volúmenes de datos. Es así que las técnicas de optimización mencionadas en este trabajo resultan de vital importancia para manejar todas las cantidades importantes de información, asegurando un óptimo rendimiento del modelo.

A continuación, se considera de forma crucial que los algoritmos, técnicas o métodos de optimización son importantes en el incremento de la precisión de los modelos de clasificación de ML por tal caso sin una adecuada optimización estos modelos pueden confrontar problemas en la convergencia y sobreajuste dificultando alcanzar una precisión alta y de esta manera se compromete la fiabilidad en los resultados obtenidos.

Del mismo modo, la insuficiencia de una optimización correcta y efectiva acarrea a un entrenamiento del modelo poco efectivo conforme a funciones de costos altos de los modelos de ML los cuales tienen presencia de múltiples mínimos locales como respuesta hay una convergencia lenta e inadecuada en el entrenamiento teniendo un impacto negativo en la eficiencia del modelo.

Por otra parte, el impacto negativo en la precisión de los modelos de clasificación se ve directamente afectado por la nula o escasa experimentación con algoritmos de optimización donde se restringe la mejora en términos de eficiencia y precisión. Se enfatiza la importancia de ejecutar pruebas y ajustes de forma continua para lograr una optimización del rendimiento del modelo en la correcta capacidad de clasificación.

Por lo tanto, el clasificador de Regresión Logística y su precisión en la detección de noticias falsas de política en español es muy limitada debido a la baja precisión del modelo. Sin embargo, la situación se puede solventar de una manera eficaz mediante la incorporación y experimentación con los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp los cuales permitirán mejorar y evaluar de manera correcta la precisión. Al realizar los ajustes, el incremento en la efectividad del modelo será evidente, lo que proporcionará resultados fiables y precisos en la identificación de información en las noticias falsas de política en español.

Finalmente, si el problema no se aborda, las consecuencias negativas en la capacidad del modelo de Regresión Logística a la hora de clasificar noticias falsas de política en español de manera correcta se mantendrá en segundo plano, lo que puede resultar en la expansión de la desinformación. La limitante falta de precisión en la detección de las noticias falsas de política en español puede impactar de forma negativa la confianza del público en las fuentes de información. Por lo tanto, el implemento estratégico de la optimización es crucial para mejorar la precisión del modelo y clasificar de manera efectiva, asegurando la integridad total de la información.

1.2. Pregunta de Investigación

¿Qué porcentaje de precisión se puede alcanzar en la identificación de noticias falsas de política en español mediante el modelo de clasificación Regresión Logística al aplicar los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp?

2. Justificación

Este trabajo se justifica teóricamente debido a la necesidad de implementar y evaluar los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp a través del modelo Regresión logística para la detección o clasificación de noticias falsas de política en español utilizando machine learning. Se pretende utilizar y ampliar el conocimiento existente sobre cómo la optimización influye en la precisión de los modelos de clasificación, permitiendo de esta manera una precisa identificación de noticias falsas de política en español. Trabajos previos han demostrado que la aplicación de los algoritmos de optimización pueden influir en la precisión del modelo de Regresión logística. Por ejemplo, como lo sugiere el estudio de Sai-Raja et al. (2023) donde la optimización incrementó la precisión del modelo del 52% al 92.70%. La exitosa aplicación del método de optimización GD y sus variantes aseguran muchas aplicaciones ayudando a combatir la reducción de la propagación de noticias falsas, en especial en temas de política.

La justificación práctica de este proyecto se basa en la optimización del modelo de Regresión logística para agilizar la clasificación de noticias falsas en español orientadas a política. Se ambiciona conseguir un modelo optimizado que sea competente y funcional. Para tener como resultado esta intención se utilizarán los conocimientos adquiridos en las clases de Machine learning, Data Mining, Human-Computer Interaction, Human Perception in Computer Vision, incluyendo habilidades propias adquiridas de forma autodidacta a lo largo de la carrera.

El desarrollo del trabajo se puede emplear como recurso de apoyo científico, académico y tecnológico a futuras investigaciones que se vinculen a la misma línea de investigación o se extrapolen hacia un tema de estudio en particular. De la misma manera, permitirá a las autoridades correspondientes elaborar y ejecutar estrategias con el propósito de mejorar la detección y mitigación de noticias falsas en especial a la categoría de política, factores que afectan la calidad de la información pública y la confianza en los medios de comunicación. Además, la aplicación práctica se da por la posibilidad que ofrece a investigadores, periodistas y reguladores un vínculo con técnicas avanzadas de machine learning y optimización, fortaleciendo así la capacidad de identificar y combatir la desinformación en diversos contextos.

3. Objetivos

3.1. General

Aplicar los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp en el modelo Regresión Logística para la detección de noticias falsas de política en español.

3.2. Específicos

- Entrenar el modelo regresión logística con los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp mediante CRISP-ML.
- Evaluar la mejora en la precisión del modelo en comparación con la precisión inicial sin optimización utilizando la matriz de confusión.

4. Marco Teórico

4.0.1. Antecedentes del problema

El modelo de ML Regresión Logística se emplea en la identificación y clasificación de Noticias Falsas pero en especial en noticias de política en español. Aportando en la verificación y validez de la autenticidad de los contenidos divulgados en medios de comunicación digital y redes sociales. Sin embargo, un problema generalizado en este campo es la limitación en la experimentación con los algoritmos GD, SGD, MBGD, AdaGrad, Adam y RMSProp. El hecho de no realizar varios experimentos restringe la evaluación de su impacto en la precisión del clasificador de regresión logística. Por lo tanto, una detección tan limitada de los elementos de la noticia es el resultado de una precisión deficiente del modelo. Esto implica que es necesario seguir explorando la investigación en esta área para abordar esta limitación y mejorar la precisión del clasificador en la lucha contra las noticias falsas en español y de manera especial en el tema de política. La capacidad de optimizar a través del descenso de gradiente, y con mayor énfasis en su variante de SGD, es importante para el algoritmo de clasificación. Por lo tanto, es vital en la detección de noticias falsas de política en español, donde la eficiencia de la actividad de procesamiento es crítica para el buen funcionamiento del sistema. Para mejorar la precisión del modelo, es importante ajustar los hiperparámetros y elegir las funciones características adecuadas. Lo encontramos a través de la optimización de la función de costo por Gradiente GD [8]. La clasificación de noticias falsas en español y en especial de política ha demostrado ser un problema crítico, ya que se vuelve más difícil garantizar su autenticidad en los medios de comunicación digital. Es una cuestión que tiene profundas implicancias para la sociedad. Por lo tanto, el desafío se basa en optimizar un clasificador basado en la regresión logística, una técnica predominante en la inteligencia artificial y el ML [9]. El propósito de este trabajo es optimizar un modelo confiable utilizando GD, SGD, MBGD, AdaGrad, Adam y RMSProp algoritmos de optimización en la clasificación de falsas noticias, así como para comparar la mejora en la precisión de este modelo con la precisión final. La razón de esto es el uso de regresión logística para la clasificación de noticias y la razón principal se la detalla en el último párrafo de la sección de **Trabajos Relacionados**.

4.0.2. Tecnologías

4.0.2.1. Materiales

El proyecto se fundamentará en la utilización de herramientas y tecnologías que permiten la implementación eficaz de algoritmos de optimización en el modelo Regresión Logística para la clasificación de noticias falsas en español de política. A continuación, se nombran y describen los materiales y tecnologías empleadas.

1. **Python.** — Es valorado por su sencillez y eficacia, destacándose por su sólido ecosistema de bibliotecas centradas en el aprendizaje automático. Estas cualidades lo convierten en una opción ideal para aplicaciones de clasificación de texto debido a su eficaz gestión de datos y modelos de aprendizaje automático. Por este motivo, Python será el principal lenguaje de programación considerado para la implementación de los algoritmos de optimización GD, SGD, MBGD, AdaGrad, Adam y RMSProp en el modelo Regresión Logística [10].

2. **Google Colab.** — Es una plataforma en la nube que se basa en cuadernos Jupyter. Google Colab brinda la oportunidad de acceder de forma gratuita a GPU y TPU, y se conecta con Google Drive. Esto posibilita disfrutar de todas las bondades de los cuadernos Jupyter junto con la sólida infraestructura de Google. El único requisito para hacer uso de este servicio es contar con conexión a internet y una cuenta de Gmail [11].

3. **Dataset.** — Se coleccionan conjuntos de datos sobre noticias falsas de política en español, ya que un dataset diverso es fundamental para entrenar y evaluar los modelos de ML. "Un dataset es un conjunto de muestras que son recopiladas y disponibles para desarrollar modelos en el ámbito de ML [12]."

4. **Scikit-Learn.** — Es una biblioteca eficiente y sencilla para ML en Python1. Destaca especialmente en el modelo de datos, reconocida por su facilidad de uso, rendimiento y documentación exhaustiva. Proporciona herramientas competentes para la selección y evaluación de modelos [13].

5. **PyTorch.** — PyTorch es una de las principales plataformas para ejecutar y desarrollar aplicaciones de ML . Su popularidad y accesibilidad entre investigadores se deben a su excelente integración con Python1. es Conocido por su rapidez y flexibilidad, lo que lo hace ideal para proyectos de investigación donde la experimentación es más frecuente que la producción a gran escala [13].

4.0.2.2. Metodología

Debido a sus capacidades, relevancia, estructura detallada y aseguramiento de calidad, se ha seleccionado la metodología CRISP-ML para la realización del proyecto."Es un modelo de proceso para ML y mejora el modelo CRIPS-DM, adaptándolo para

aplicación de ML [14]". Algunas de las ventajas de utilizar CRISP-ML se mencionan a continuación.

1. **Expansión de CRISP-DM.** — CRISP-ML se expande a partir de CRISP-DM, abordando las necesidades del Machine Learning en cuanto a la adaptación de entornos cambiantes, incluyendo la monitorización y el mantenimiento constante de los modelos desplegados previamente [14].

2. **Fases del proceso.** — CRISP-ML abarca varias fases de una aplicación de ML, incluyendo la comprensión del negocio y de los datos, la preparación de datos, el modelado, la evaluación, el despliegue y el mantenimiento o monitoreo [14].

3. **Iteración y Mejora Continua.** — CRISP-ML es iterativo, lo que permite retroceder a fases anteriores para ajustar el modelo según sea necesario. Esta flexibilidad y mejora continua permite mitigar riesgos durante el desarrollo del modelo, asegurando así la calidad y eficacia del en cada paso [14].

4. **Enfoque industria y aplicación neutral.** — Es un modelo de procesos con un enfoque neutro y adecuado para una variedad de desafíos en ML, considerando su aplicabilidad y estabilidad como una metodología segura y sólida [14]

La utilización de CRISP-ML asegura que el modelo optimizado a través del algoritmo de optimización GD y sus variantes se realice de una manera completa y planificada, generando resultados precisos. Además, la metodología CRISP-ML garantiza facilidad en el estudio y la aplicación en el proyecto gracias a su estructura y práctica. Esta estructura detallada ayudará a mejorar significativamente la eficiencia y eficacia del proyecto, asegurando resultados más confiables y aplicables en un entorno real.

4.0.3. Base teórica

GD es un algoritmo de optimización que busca valores de los parámetros que minimicen la función de pérdida del modelo. Funciona dando pasos repetidos en la dirección opuesta al gradiente de la función en el punto actual. Existen variantes del GD que varían según la cantidad de datos utilizados en la función objetivo. Basándonos en la cantidad de datos, se busca un equilibrio entre la exactitud de la actualización de los parámetros y el tiempo necesario para ejecutar cada actualización [15]. En la siguiente figura se muestra la representación gráfica del algoritmo de optimización GD.

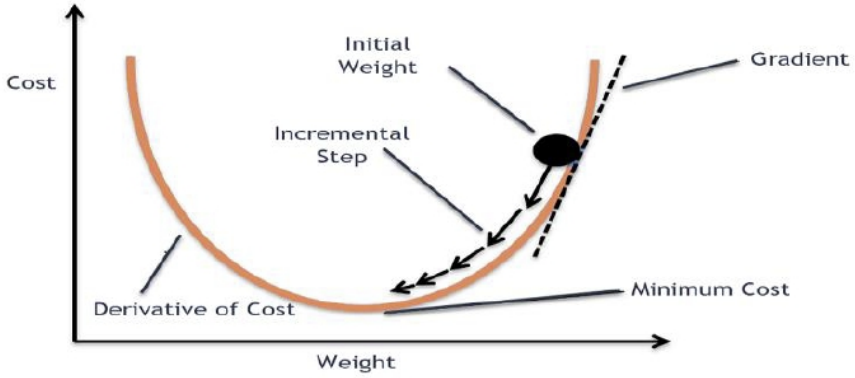


Figura 4.1: Algoritmo Gradiente Descendente

El SGD es una técnica usada en el campo de la optimización para reducir la función de pérdida en modelos de ML. Este método se aplica de manera iterativa, calculando el gradiente utilizando todo el conjunto de datos en cada paso. El SGD actualiza los parámetros del modelo con un subconjunto aleatorio de datos, conocido como mini-batch, lo que lo hace más rápido y eficiente en términos de memoria, especialmente al trabajar con grandes volúmenes de información [16].

Una variante del algoritmo GD es el Descenso del Gradiente por Minibatches MBGD. Este modelo es sencillo pero efectivo, utilizado tanto para regresión lineal como polinómica, teniendo un impacto mayor que el descenso por lotes. Sin embargo, su principal limitación radica en su sensibilidad a valores atípicos u outliers al utilizar pérdidas cuadráticas [17].

Adaptive Gradient Algorithm AdaGrad ajusta la tasa de aprendizaje para los parámetros, realizando actualizaciones pequeñas en aquellos asociados con características comunes y tasas más altas para aquellos menos frecuentes. AdaGrad resulta efectivo cuando los datos son dispersos y ha demostrado ser exitoso en diversas tareas relacionadas con ML [18].

Por otra parte, Adam se fundamenta en la optimización estocástica, necesitando solo el gradiente de primer orden y haciendo uso de una cantidad reducida de memoria. Este método combina todos los beneficios de AdaGrad, pero demuestra su efectividad para gradientes dispersos al almacenar los promedios de los gradientes y los cuadrados de los gradientes de todas las iteraciones previas [19].

Una extensión del descenso de gradiente diseñada para acelerar el proceso de optimización de los modelos es el método RMSProp. Este enfoque reduce la cantidad de evaluaciones de la función, lo que mejora el rendimiento del algoritmo. RMSProp se basa en calcular el promedio ponderado de los cuadrados de los gradientes, similar al descenso de gradiente con momentum, pero se diferencia en la forma en que actualiza los parámetros. Durante la retropropagación en el algoritmo RMSProp, se lleva a cabo la actualización de parámetros del modelo, como los pesos (W) o sesgos (B), utilizando una tasa de aprendizaje adaptada a partir del promedio ponderado de los cuadrados

del gradiente [20].

Después de aplicar cualquier algoritmo optimizador, es fundamental realizar una evaluación utilizando la matriz de confusión. Esta herramienta permite evaluar el desempeño del modelo de Regresión Logística en la predicción de noticias falsas en español de política y facilita la interpretación del rendimiento clasificatorio. En esta matriz, las clases predichas se disponen en las filas, mientras que las columnas representan las clases reales. En ella se pueden identificar elementos como verdaderos positivos (TP), falsos positivos (FP), falsos negativos (FN) y verdaderos negativos (TN). Además, se examinan y se calculan medidas de evaluación que abarcan la exactitud, la sensibilidad y la especificidad, lo cual contribuye a evaluar el desempeño del modelo. La exactitud se determina al dividir el número de predicciones acertadas entre el total de muestras. . La matriz de confusión facilita la identificación de patrones mal clasificados y proporciona una visión de cómo se comportará el modelo en términos de aciertos y errores [21].

Confusion matrix		True class (Actual)	
		P	N
Hypothesized class (Predicted)	Y	True Positives	False Positives
	N	False Negatives	True Negatives

Figura 4.2: Matriz de Confusión

Finalmente, mencionaremos el enfoque principal de este estudio, que es la Regresión Logística. Esta técnica estadística se utiliza para modelar variables dependientes que solo pueden asumir dos categorías, como binario 1/0, sí/no, éxito/fracaso, entre otros. La Regresión Logística es una opción adecuada para evitar posibles problemas de ineficiencia o sesgo que podrían surgir en un modelo lineal. En la ecuación mostrada en la Figura 4.3, "Y" representa la variable dependiente, "X" las variables independientes, "B" es el coeficiente de regresión y "e" denota el término de error [22].

$$\text{logit}(Y) = \alpha + X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + X_4\beta_4 + X_5\beta_5 + X_6\beta_6 + X_7\beta_7 + X_8\beta_8 + X_9\beta_9 + X_{10}\beta_{10} + X_{11}\beta_{11} + \varepsilon$$

Figura 4.3: Fórmula básica del modelo Regresión Logística

4.0.3.1. Trabajos Relacionados

Un estudio realizado por Martínez-Gallego y colaboradores (2022) sobre la detección de noticias falsas en español utilizando técnicas de aprendizaje automático con modelos preentrenados como BERT y ELMo logró una precisión máxima del 80% en la clasificación de noticias falsas. Los algoritmos empleados incluyeron Regresión Logística, Máquinas de Soporte Vectorial (SVM), Bosques Aleatorios, Clasificador Naive Bayes, Boosting y Redes Neuronales Artificiales [23]. Estos resultados señalan ciertas limita-

ciones en la tarea de clasificación en español, lo que sugiere la necesidad de optimizar a través de algoritmos como GD, SGD, MBGD, AdaGrad, Adam y RMSProp en el modelo de Regresión Logística.

Por lo tanto, este estudio muestra que el uso de Regresión Logística mejorada mediante el algoritmo de GD es eficaz para identificar perfiles falsos en plataformas de redes sociales. Como resultado, la precisión ha experimentado un notable aumento del 52% sin optimización al 85% con optimización, y el AUC ha alcanzado un valor de 0.89, lo que indica que la mejora del modelo puede resultar efectiva en la detección de perfiles fraudulentos. [2].

Por su parte, el estudio de Asha J y Meenakowshalya A. (2021) explora el análisis del uso de N-gramas y varios algoritmos de ML. Los algoritmos evaluados fueron: Máquinas de Vectores de Soporte (SVM), SVM Lineal, K-Nearest Neighbor (KNN), SGD, Árboles de Decisión (DT) y Regresión Logística (LR). Se determinó que el algoritmo de optimización SGD, combinado con TF-IDF y unigramas, ofreció los mejores resultados para la detección de noticias falsas, alcanzando una precisión máxima del 94.2% después del ajuste [5].

Se utilizó la Regresión Logística y se aplicaron técnicas de optimización como SGD y Adam para mejorar la eficacia del modelo en la detección de noticias falsas. Estas técnicas de optimización ayudaron a refinar los parámetros del modelo Regresión Logística. Los resultados mostraron que el algoritmo de Regresión Logística sin optimizar alcanzaba una precisión del 87%. Al aplicar los algoritmos de optimización Adam y SGD, la precisión mejoró significativamente, aumentando a un rango de 91-93%, dependiendo del conjunto de datos utilizado [6].

Por lo tanto, en la detección de noticias falsas en español, se emplea técnicas de procesamiento de texto y ML. Ajustar los modelos con algoritmos de optimización Adam demuestra un aumento de la precisión, lo que resalta la importancia de la optimización en los modelos de clasificación de ML [7].

En otro estudio, se utilizaron una variedad de técnicas de optimización para mejorar la precisión en la detección de noticias falsas. Las técnicas empleadas fueron: SGD, AdaGrad, RMSProp y Adam. Estos métodos fueron esenciales para hacer un ajuste fino de los parámetros del modelo, logrando mejorar el rendimiento. Cada uno de estos algoritmos de optimización tuvo beneficios en términos de ajuste fino y tasa de aprendizaje [3].

Siguiendo con este enfoque, en este estudio se analizaron varias técnicas de optimización tales como SGD, MBGD, AdaGrad, RMSProp, y otros. Estos métodos son cruciales para resolver problemas de optimización en modelos de ML, aumentando significativamente la tasa de convergencia y la estabilidad de los algoritmos de optimización. Además, el estudio proporciona pautas importantes y detalladas sobre la incorporación de parámetros [24].

En el estudio "Stance Classification for Fake News Detection with Machine Learning" se analiza la detección de noticias falsas y se aplican algoritmos de optimización como SGD, mejorando la eficacia de los modelos tanto en la clasificación como en la detección

de noticias falsas. Se utiliza el conjunto de datos AFND (Arabic Fake News Detection). El rendimiento después de la optimización muestra que la precisión de detección es del 87%. Los modelos de detección de noticias falsas pueden aplicarse o integrarse en plataformas de medios sociales para ayudar a la mitigación en la propagación de noticias falsas y, de esta manera, mejorar la calidad de la información que se ve día a día en línea [25].

En el estudio se aborda el paradigma de precisión, es decir, se busca la optimización del modelo. En un escenario real, la solución es integrar modelos de ML. Para ello, se utilizan varios modelos como Naive Bayes, Regresión Líneal, Regresión Logística y Máquinas de soporte vectorial (SVM). Los algoritmos de optimización implementados para este estudio incluyen una variante del GD llamado Gradiente Descendente Gaussiano. La predicción se realizó usando un conjunto de datos de Twitter extraídos de varias partes del mundo. Finalmente, se compararon los modelos optimizados en términos de precisión, demostrando una alta efectividad, con una precisión alrededor del 94% al 97% [26].

En la siguiente tesis, presentada para obtener el grado de Máster en Ciencias Computacionales en la Universidad Laurentian de Sudbty, se analiza el método de SGD aplicado a modelos de Redes Neuronales Recurrentes (RNN) para la clasificación de noticias falsas. Antes de la optimización, la precisión del modelo era de 97%, mientras que, tras la optimización, que incluyó ajustes de función de perdida y número de épocas o iteraciones en la configuración del modelo, la precisión aumentó al 98% con una reducción de pérdida de hasta 0.137 [27]. Este incremento en modelos de RNN para la clasificación correcta de noticias como falsas o verdades nos brinda una idea de como se vería la optimización del algoritmo de clasificación Regresión Logística, ya que ayudará a la predicción de noticias falsas en español de política con una precisión más confiable.

En este artículo se aborda el desarrollo de un sistema para predecir y clasificar noticias falsas utilizando técnicas de ML. En la sección de metodología, se explica cómo el algoritmo de Regresión Logística se aplica para la predicción de noticias falsas. El GD se utiliza para optimizar los parámetros del modelo de Regresión Logística, encargándose de la inicialización de los pesos y la actualización de los parámetros para minimizar la función de pérdida usando los algoritmos de optimización GD [4].

Se presenta un enfoque para la detección de noticias falsas utilizando una técnica de ponderación de dos capaces. Para ello, se utiliza el algoritmo de optimización SGD para optimizar la función de pérdida mediante actualizaciones incrementales. Esto es válido y crucial en escenarios donde se reciben los datos de manera continua. Por lo tanto, el método de SGD también se utiliza para manejar de manera eficiente grandes cantidades de datos, como es el caso de la detección de noticias falsas, donde los datos pueden ser extensos y variados. Además, se emplea el modelo de Regresión Logística para clasificación binaria, en este caso, la detección de noticias falsas [28]. La Regresión Logística es tomada en cuenta en problemas de ML por su simplicidad, eficiencia. Por lo tanto, permite mejorar la capacidad de cualquier sistema para identificar y clasificar con precisión las noticias falsas en español en comparación con otros métodos.

Este artículo explora las técnicas diversas para la detección de noticias falsas. Su investigación se enfoca en combatir las noticias falsas debido al impacto negativo en la sociedad. A través de una revisión sistemática de la literatura, el estudio presenta varios enfoques de ML aplicados a la detección de noticias falsas, haciendo mención del algoritmo de optimización SGD utilizado para predecir parámetros de modelos que sean compatibles con la mejor correspondencia entre salidas reales y predichas [29].

Tabla 4.1: Trabajos Relacionados

Trabajo Relacionado	**Algoritmos de Clasificación**	**Precisión**	**Algoritmos de Optimización**
Detección de Perfiles Falsos en Redes Sociales			
Fake Profile Detection Using Logistic Regression and Gradient Descent Algorithm on Online Social Networks	Regresión Logística	52% Sin optimizar 92.70% Optimizado	Gradiente Descendente
Detección de Noticias Falsas			
Optimizing Hyperparameters and Performance Analysis of LSTM Model in Detecting Fake News on Social media	LSTM	98.74% Sin optimizar 99.52% Optimizado	SGD, AdaGrad, RMSProp, Adam
Fake News Detection Using Machine Learning Techniques	Regresión Logística	77.28%	No especificado
Fake News Detection Using N-Gram Analysis and Machine Learning Algorithms	Máquinas de Vectores de Soporte (SVM), Árboles de Decisión (DT) y Regresión Logística (LR).	94.2%	SGD combinado con TF-IDF
Fake news detection using machine learning ensemble methods	Regresión Logística	87% Sin optimizar 91-93% Optimizado	Adam SGD
Stance classification for fake news detection with machine learning	Redes Neuronales Concurrentes (CNN)	87.7% Optimizado con SGD y & 74.8% con Adam	Adam SGD

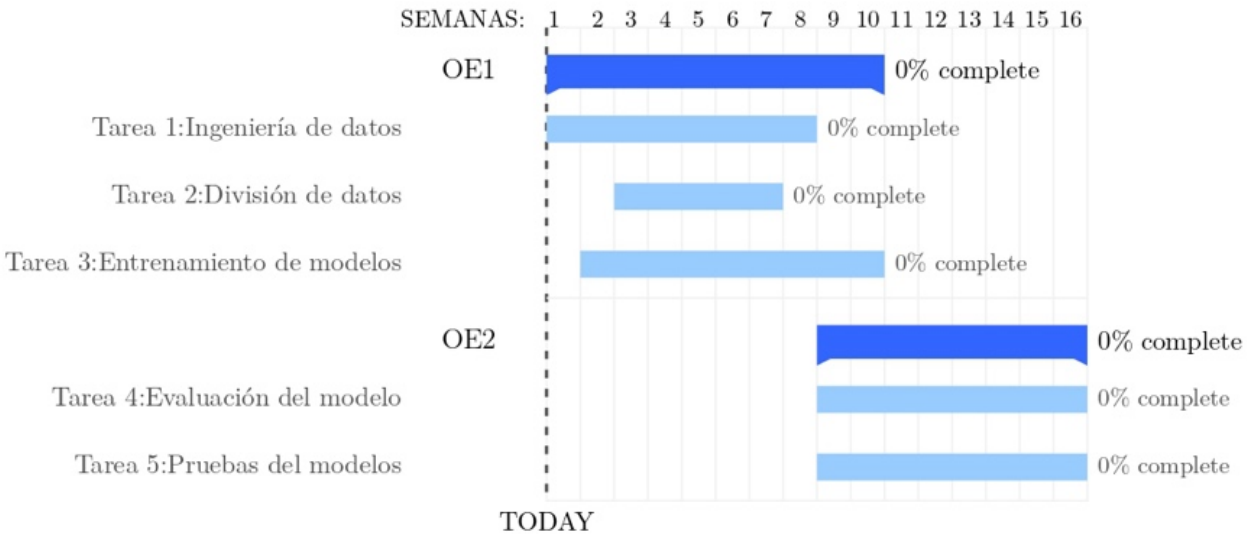
Finalmente, la elección de los algoritmos de optimización y el modelo de Regresión Logística se basa en la tabla 4.1. Los algoritmos de optimización como GD, SGD, MBGD, AdaGrad, Adam y RMSProp se escogieron porque son métodos fundamentales para optimizar funciones de costo, como la función de pérdida logística en la Regresión Logística. Estos algoritmos ayudan a optimizar la Regresión Logística de manera eficiente, asegurando al modelo, precisión y practicidad en las implementaciones. La Regresión Logística es un modelo especializado en la clasificación binaria, por lo que puede ser empleado en la identificación y clasificación de noticias falsas en español orientado a temas de política. Sin embargo, los trabajos relacionados muestran una precisión baja del modelo, lo que lo convierte en un gran candidato para la aplicación de técnicas de optimización.

5. Metodología

Tabla 5.1: Metodología del PTT

OBG	OBE	Métodos/Tec	Alcance	Materiales	Lugar	Responsable	Producto
OB	OE1	CRISP-ML,	Ingeniería de datos - Selección de datos - Preparación de los datos - Equilibrio de clases - Limpieza de datos - Aumento de datos - Normalización de datos División de datos Entrenamiento de modelos - Selección de Optimización - Selección del modelo - Entrenamiento del modelo - Ajuste de hiperparámetros - Regularización	Noticias política en español, Dataset, Dataset de entrenamiento, Python, Scikit-Learn, Pytorch , Google Colab	Laboratorio	Santiago Tene	Modelo entrenado
	OE2	CRISP-ML, Matriz de confusión	Evaluación del modelo Pruebas del modelo	Dataset de pruebas, Dataset de validación	Laboratorio	Santiago Tene	Modelo aceptado Modelo optimizado

6. Cronograma



7. Presupuesto y Financiamiento

7.0.1. Talento Humano

En la tabla 7.1 se consideran los salarios de todos los participantes del desarrollo de este proyecto. Los detalles se verán a continuación

Rol	Persona	Horas (Tiempo)	Valor U.	Valor Total
Desarrollador	Santiago Tene Castillo	640	$ 6,75	$4 320,00
Tutor	Ing. Luis Chamba Eras	400	$1 0,45	$ 4 180,00
Total				$ 8 500,00

Tabla 7.1: Tabla de costos para talento humano

7.0.2. Servicios

Será necesario contratar los servicios básicos para el desarrollo del proyecto. En este caso, se han tomado en cuenta la electricidad e internet. Los costos previstos o estimados se pueden ver detalladamente en la tabla 7.2.

Servicios	Meses (Tiempo)	Valor Unitario	Valor Total
Electricidad	4	$ 8,69	$ 34,76
Internet	4	$ 25,00	$ 100,00
Total			$ 134,75

Tabla 7.2: Tabla costos de servicios

7.0.3. Hardware

El hardware necesitado para el desarrollo del proyecto se pueden observar de una manera más clara en la tabla 7.3. A continuación.

Recurso	Cantidad	Valor Unitario	Valor Total
Laptop	1	$ 850,00	$ 850,00
Monitor	1	$ 120,00	$ 120,00
Total			$ 970,00

Tabla 7.3: Tabla de costos de hardware

7.0.4. Software

Los recursos de software requeridos para este proyecto se lo detalla en la tabla 7.4 presentada a continuación.

Recursos	Meses (Tiempo)	Valor Unitario	Valor Total
Microsoft Office	4	$ 16,93	$ 67,72
Google Colab - GPU	4	$ 50,00	$ 200,00
Total			$ 267,72

Tabla 7.4: Tabla Recursos de Software

7.0.5. Materiales de oficina

Los materiales de oficina necesarios para el desarrollo del proyecto son básicos y no necesitan una evaluación minuciosa. Por lo tanto, se describen de manera general en la tabla 7.5. Aunque podrían incluirse más elementos como materiales de oficina, en este caso no se ha determinado que se necesiten más materiales específicos para el desarrollo del proyecto.

Materiales	Cantidad	Valor Unitario	Valor Total
Papelería en general	1	$ 85,00	$ 85,00
Total			$ 85,00

Tabla 7.5: Tabla Materiales oficina

7.0.6. Resumen del Presupuesto

De acuerdo con los recursos económicos previamente detallados, hemos creado la tabla 7.6. Proporcionando una visión integral de los gastos necesarios para llevar a cabo el proyecto. El costo total del proyecto, que incluye todos los gastos mencionados, es de $ 9 957,48. La administración adecuada de los recursos es fundamental para cumplir con todos los objetivos propuestos en el plazo establecido. A continuación se detalla todo el presupuesto requerido.

Material	Valor Total
Talento Humano	$ 8 500,00
Servicios	$ 134,76
Hardware	$ 970,00
Software	$ 267,72
Materiales de Oficina	$ 85,00
Total	$ 9 957,48

Tabla 7.6: Total del Presupuesto

Bibliografía

[1] Y. Tian, Y. Zhang, and H. Zhang, "Recent advances in stochastic gradient descent in deep learning," *Mathematics*, vol. 11, no. 3, p. 682, 2023.

[2] E. V. S. Raja, B. L. Aditya, and S. N. Mohanty, "Fake profile detection using logistic regression and gradient descent algorithm on online social networks," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 11, no. 1, 2024.

[3] R. Rajalaxmi, L. Narasimha Prasad, B. Janakiramaiah, C. Pavankumar, N. Neelima, and V. Sathishkumar, "Optimizing hyperparameters and performance analysis of lstm model in detecting fake news on social media," *Transactions on Asian and Low-Resource Language Information Processing*, 2022.

[4] D. O. Aruna, K. Manoj, L. V. Krishna, M. Yasaswini, and K. Eswar, "Fake news detection using machine learning techniques," 2024.

[5] J. Asha and A. Meenakowshalya, "Fake news detection using n-gram analysis and machine learning algorithms," *Journal of Mobile Computing, Communications & Mobile Networks*, vol. 8, no. 1, pp. 33–43p, 2021.

[6] I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, "Fake news detection using machine learning ensemble methods," *Complexity*, vol. 2020, no. 1, p. 8885861, 2020.

[7] N. J. Mafla Checa, "Identificación automática de noticias falsas en español utilizando técnicas de minería de datos y procesamiento del lenguaje natural.," B.S. thesis, Quito, 2021., 2021.

[8] A. Srivastava, "Real time fake news detection using machine learning and nlp," *Int. Res. J. Eng. Technol.(IRJET)*, vol. 7, no. 06, 2020.

[9] A. Singh, A. Ugale, N. Shah, and A. Sankhe, "Fake news detection using logistic regression & multinomial naive bayes," 2021.

[10] S. Raschka, J. Patterson, and C. Nolet, "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence," *Information*, vol. 11, no. 4, p. 193, 2020.

[11] W. Vallejo, C. Díaz-Uribe, and C. Fajardo, "Google colab and virtual simulations: practical e-learning tools to support the teaching of thermodynamics and to introduce coding to students," *ACS omega*, vol. 7, no. 8, pp. 7421–7429, 2022.

[12] C. Cappi, C. Chapdelaine, L. Gardes, E. Jenn, B. Lefevre, S. Picard, and T. Soumarmon, "Dataset definition standard (dds)," *arXiv preprint arXiv:2101.03020*, 2021.

[13] A. Pajankar and A. Joshi, *Hands-on Machine Learning with Python: Implement Neural Network Solutions with Scikit-learn and PyTorch*. Springer, 2022.

[14] S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K.-R. Müller, "Towards crisp-ml (q): a machine learning process model with quality assurance methodology," *Machine learning and knowledge extraction*, vol. 3, no. 2, pp. 392–413, 2021.

[15] S. H. Haji and A. M. Abdulazeez, "Comparison of optimization techniques based on gradient descent algorithm: A review," *PalArch's Journal of Archaeology of Egypt/Egyptology*, vol. 18, no. 4, pp. 2715–2743, 2021.

[16] A. Beznosikov, E. Gorbunov, H. Berard, and N. Loizou, "Stochastic gradient descent-ascent: Unified theory and new efficient methods," in *International conference on artificial intelligence and statistics*, pp. 172–235, PMLR, 2023.

[17] H. Wang, H. Luo, and Y. Wang, "Mbgdt: robust mini-batch gradient descent," *arXiv preprint arXiv:2206.07139*, 2022.

[18] A. Alblwi, *Improving the adaptive moment estimation optimization methods for modern machine learning*. University of Delaware, 2020.

[19] I. H. Kartowisastro and J. Latupapua, "A comparison of adaptive moment estimation (adam) and rmsprop optimisation techniques for wildlife animal classification using convolutional neural networks.," *Revue d'Intelligence Artificielle*, vol. 37, no. 4, 2023.

[20] K. P. Rakshitha and N. Naveen, "Op-rmsprop (optimized-root mean square propagation) classification for prediction of polycystic ovary syndrome (pcos) using hybrid machine learning technique," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 6, 2022.

[21] H. Yun, "Prediction model of algal blooms using logistic regression and confusion matrix," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 3, pp. 2407–2413, 2021.

[22] A. A. T. Fernandes, D. B. Figueiredo Filho, E. C. d. Rocha, and W. d. S. Nascimento, "Read this paper if you want to learn logistic regression," *Revista de Sociologia e Política*, vol. 28, p. 006, 2021.

[23] K. Martínez-Gallego, A. M. Álvarez-Ortiz, and J. D. Arias-Londoño, "Fake news detection in spanish using deep learning techniques," *arXiv preprint arXiv:2110.06461*, 2021.

[24] G. V. Krivovichev and V. Y. Sergeeva, "Analysis of a two-step gradient method with two momentum parameters for strongly convex unconstrained optimization," *Algorithms*, vol. 17, no. 3, p. 126, 2024.

[25] M. Alsafadı, "Stance classification for fake news detection with machine learning," *The Eurasia Proceedings of Science Technology Engineering and Mathematics*, vol. 22, p. 191–198, 2023.

[26] S. Basharat, S. Afzal, A. Bamhdi, S. Khurshid, and M. Chachoo, "Predicting and mitigating the effect of skewness on credibility assessment of social media content using machine learning: A twitter case study," *International Journal of Computer Theory and Engineering*, vol. 15, p. 101, 11 2023.

[27] A. Shah, *Distinguishing fake and real news of twitter data with the help of machine learning techniques*. PhD thesis, Laurentian University of Sudbury, 2021.

[28] A. H. E. Wynne and K. T. Swe, "A soft two-layers voting model for fake news detection,"

[29] H. Garg, A. Goyal, and M. A. Joshi, "Techniques of fake news detection," *International Journal of Civil, Mechanical and Energy Science*, vol. 6, no. 2, pp. 6–9, 2020.

Revisión Sistemática de Literatura: Impacto de los algoritmos de optimización en la regresión logística para mejorar la detección de noticias falsas

Santiago Tene-Castillo
Carrera de Ingeniería en Sistemas
Universidad Nacional de Loja, Loja-Ecuador
Santiago.tene@unl.edu.ec

***Abstract*-- Through a systematic literature review, three key research questions are explored: (1) the limitations of logistic regression in detecting fake news when optimization algorithms are applied, (2) the issues arising from the lack of optimization techniques in machine learning models, and (3) the impact of optimization algorithms on the accuracy of the models. A total of 50 relevant articles were identified from 85 found, after applying inclusion and exclusion criteria, and the review analyzed how different optimization techniques, such as gradient descent, AdaGrad, Adam, and RMSProp, can influence the improvement of fake news detection.**

The results indicate that while logistic regression is effective, it has limitations in terms of its accuracy without proper optimization. The lack of optimization can lead to issues such as inefficient use of computational resources, prolonged training times, and lower accuracy. On the other hand, optimization algorithms are crucial for improving accuracy and avoiding issues of convergence and overfitting, leading to more accurate and robust detection of fake news.

***Resumen*—** A través de una revisión sistemática de la literatura, se exploran tres preguntas clave de investigación: (1) las limitaciones de la regresión logística en la detección de noticias falsas cuando se aplican algoritmos de optimización, (2) los problemas derivados de la falta de técnicas de optimización en los modelos de machine learning, y (3) el impacto de los algoritmos de optimización en la precisión de los modelos. Se identificaron 50 artículos relevantes a partir de un total de 85 encontrados, tras aplicar criterios de inclusión y exclusión, y se analizó cómo diferentes técnicas de optimización, como el gradiente descendente, AdaGrad, Adam y RMSProp, pueden influir en la mejora de la detección de noticias falsas.

Los resultados indican que la regresión logística, aunque efectiva, tiene limitaciones en cuanto a su precisión sin una adecuada optimización. La falta de optimización puede generar problemas como un uso ineficiente de recursos computacionales, tiempos de entrenamiento prolongados y una menor precisión. Por otro lado, los algoritmos de optimización son cruciales para mejorar la precisión y evitar problemas de convergencia y sobreajuste, lo que lleva a una detección más precisa y robusta de noticias falsas.

***Keywords*-** machine learning, logistic regression, optimization, hyperparametric, gradient descendent.

Introducción

La proliferación de noticias falsas en las plataformas digitales ha emergido como uno de los desafíos más significativos de la era moderna, afectando tanto a la sociedad como a la credibilidad de las fuentes de información. En este contexto, la inteligencia artificial (IA) y, en particular, los modelos de aprendizaje automático, han mostrado un gran potencial para abordar este problema mediante la automatización de la detección de noticias falsas. Sin embargo, a pesar de los avances en la investigación, la precisión y la eficiencia de estos modelos siguen siendo un área de preocupación, especialmente cuando se trata de técnicas de optimización que mejoran su rendimiento.

La regresión logística, uno de los modelos más utilizados en la clasificación de datos, ha sido empleada con éxito en la detección de noticias falsas. Sin embargo, la precisión de este modelo puede verse afectada por limitaciones inherentes a su estructura y por la falta de optimización adecuada. Los algoritmos de optimización, como el gradiente descendente, AdaGrad, Adam y RMSProp, son esenciales para mejorar la convergencia y precisión de estos modelos, permitiendo una detección más precisa y eficiente de noticias falsas.

Este estudio busca abordar los desafíos existentes en la utilización de la regresión logística y otros modelos de aprendizaje automático para la detección de noticias falsas, centrándose especialmente en la importancia de las técnicas de optimización para mejorar su rendimiento [61].

Metodología

El alcance de esta revisión sistemática se centra en examinar el impacto de los algoritmos de optimización en los modelos de aprendizaje automático, específicamente en el contexto de la detección de noticias falsas. A través de una recopilación exhaustiva de artículos relevantes publicados desde 2020, se busca evaluar cómo la aplicación de diferentes técnicas de optimización afecta la precisión, eficiencia y capacidad de convergencia de los modelos utilizados para detectar noticias falsas. A través de esta revisión, se pretende ofrecer una comprensión más profunda de los avances en la detección de noticias falsas mediante el uso de técnicas de optimización y proporcionar una base sólida para futuras investigaciones en este campo [62].

Resultados

En esta sección se indican los resultados obtenidos en cada etapa, se trata de explicar que las búsquedas produjeron 50 artículos, de los cuales tras aplicar palabras clave, título y resumen, aplicando criterios de inclusión y excusión, quedaron un total de 31 artículos para poder dar contestación a las preguntas de investigación.

Planificación de la revisión

Identificación de la necesidad de una revisión

Mediante una revisión sistemática de la literatura, se busca descubrir el conocimiento existente en torno a un tema de investigación específico. En este caso se buscó y seleccionó metodologías, herraminetas, algoritmos de optimización, y modelos de machine learning de clasificación. Para cumplir con el alcance propuesto, se formularon tres preguntas de investigación clave.

Preguntas de investigación

El auge de las noticias falsas ha generado un gran interés entre los investigadores especializados en Inteligencia Artificial (IA), lo que se manifiesta en diversas preguntas como: "¿Cuáles son los enfoques y modelos de machine learning más efectivos para la detección de noticias falsas?" Para abordar esta pregunta, se formulan tres preguntas de investigación como estrategia de análisis.

A continuación, en la TABLA XVII se presentan las tres preguntas específicas de investigación formuladas para responder la pregunta general y guiar el desarrollo de la investigación:

TABLA XVII
PREGUNTAS DE INVESTIGACIÓN

Preguntas de investigación	
P1	¿Qué limitaciones presenta la regresión logística en la detección de noticias falsas al aplicar algoritmos de optimización?
P2	¿Qué problemas pueden surgir si no se utilizan técnicas de optimización en modelos de machine learning?
P3	¿Los algoritmos de optimización ayudan a aumentar la precisión de los modelos de machine learning?

El desarrollo de un protocolo de revisión

Diseño del protocolo de búsqueda

Estrategias de búsqueda

Se aplicó una adaptación de Estrategia de Búsqueda Estructurada implica definir claramente preguntas de investigación, seleccionar bases de datos relevantes, identificar palabras clave y operadores booleanos, y aplicar filtros de búsqueda para limitar los resultados. También utiliza referencias cruzadas y alertas de búsqueda para asegurar que la recopilación de literatura sea exhaustiva y relevante.

TABLA XVIII
MATERIALES UTILIZADOS

Editor de texto	Microsoft Word
Gestor bibliográfico	Mendeley
Traductor	DeepL

Fuentes bibliográficas

Como fuentes bibliográficas, se ha seleccionado algunas bibliotecas virtuales, como son:

- IEEE Digital Library (https://ieeexplore.ieee.org/)
- Science@Direct (https://www.sciencedirect.com/)
- ACM Digital Library (https://dl.acm.org/)
- Google Scholar (https://scholar.google.com/)

Se seleccionó el buscador académico Google Scholar, debido a su popularidad y confiabilidad. Esta herramienta integra diversas bases de datos en un único motor de búsqueda, lo cual facilitará responder las preguntas de investigación.

Definir palabras claves para el problema de estudio

Definir palabras clave es fundamental para enfocar la búsqueda en una Revisión Sistemática de la Literatura (SLR). Las palabras clave deben estar estrechamente relacionadas con el problema de estudio y abarcar sinónimos o términos alternativos para incluir la mayor cantidad de investigaciones relevantes. Las palabras clave son: machine learning, logistic regression, optimization, hyperparametric, gradient descendent.

Cadenas de búsqueda

Una vez delimitadas las bases de datos y las palabras claves, se analizaron las posibles combinaciones con ayuda de los operadores lógicos "AND / OR", dando como resultado los scripts o cadenas de búsqueda representados en la TABLA XIX. Los scripts de búsqueda generados fueron modificados dependiente a las distintas bases de datos y traducidos al inglés, para poder ampliar el campo de búsqueda y encontrar información actualizada.

Criterios de selección

Para definir cuáles son los artículos en los que basaremos la revisión de literatura, debemos seleccionar los documentos a partir de las preguntas de investigación y con base a ciertos puntos: criterio de inclusión y exclusión. Por otra parte, únicamente se consideran artículos de las bases de datos escogidas.

Criterios de inclusión

Durante la búsqueda, se consideraron los siguientes criterios de inclusión:

- El documento contiene elementos novedosos o
- específico referente a noticias falsas, algoritmos de optimización y modelos de clasificación.
- Artículos científicos publicados a partir de 2020.
- El artículo deberá contener la cadena de búsqueda en su título o resumen.
- El artículo deberá contener las palabras claves.

- El artículo muestra explícitamente el aporte realizado.

Criterios de exclusión

En los criterios de exclusión se seleccionaron aspectos que se consideran no aportarán a la investigación. Por ello, los artículos que no serán tomados en cuenta presentan las siguientes características:

- En el artículo se discuten temas afines que no son propiamente del tema en discusión.
- El artículo replíca un estudio o metodología sin ningún desarrollo conceptual o práctico.
- Documentos que no sean de los motores de búsqueda seleccionada

Realizar la revisión

Identificación de la investigación

El objetivo de la presente revisión sistemática de literatura, es dar respuesta a las preguntas de investigación, a través de la búsqueda de estudios primarios que contribuyan con información fiable entorno a los temas de interés.

TABLA XIX
CADENAS DE BÚSQUEDA

ID	Cadenas de búsqueda
CB1	((("fake news") AND ("optimization algorithms") AND ("bias")) OR (("metrics") AND ("classification models") AND ("limitations"))) [Publication Date: (01/01/2020 TO 31/12/2024)]
CB2	((("fake news") AND ("gradient descent") AND ("logistic regression")) OR (("metrics") AND ("classification accuracy") AND ("limitations"))) [Publication Date: (01/01/2020 TO 31/12/2024)]
CB3	((("logistic regression") AND ("optimization") AND ("classification models")) OR (("fake news") AND ("gradient descent") AND ("bias"))) [Publication Date: (01/01/2020 TO 31/12/2024)]
CB4	((("optimization algorithms") AND ("classification models") AND ("accuracy")) OR (("fake news") AND ("logistic regression") AND ("limitations"))) [Publication Date: (01/01/2020 TO 31/12/2024)]
CB5	((("classification models") AND ("fake news") AND ("optimization techniques")) OR (("metrics") AND ("logistic regression") AND ("bias"))) [Publication Date: (01/01/2020 TO 31/12/2024)]
CB6	((("fake news detection") AND ("classification accuracy") AND ("optimization techniques")) OR (("logistic regression") AND ("bias") AND ("gradient descent"))) [Publication Date: (01/01/2020 TO 31/12/2024)]
CB7	((("optimization algorithms") AND ("fake news classification") AND ("model performance")) OR (("classification metrics") AND ("logistic regression") AND ("accuracy"))) [Publication Date: (01/01/2020 TO 31/12/2024)]
CB8	((("gradient descent") AND ("classification models") AND ("false news")) OR (("logistic regression") AND ("optimization") AND ("classification techniques"))) [Publication Date: (01/01/2020 TO 31/12/2024)]

CB (CADENA DE BUSQUEDA)

En la TABLA XX se presenta un resumen de los trabajos o estudios relacionados que fueron encontrados junto con el número de estudios seleccionados según la fuente bibliográfica.

TABLA XX
RESUMEN DE ESTUDIOS SELECCIONADOS

Fuente Bibliográfica	Encontrados	Seleccionados
IEEE Digital Library	30	16
Science@Direct	10	7
ACM Digital Library	25	15
Google Scholar	20	12
TOTAL	**85**	**50**

Se obtuvieron 43 estudios relacionados, estos se detallan a continuación, en la TABLA XXI.

TABLA XXI
ESTUDIOS SELECCIONADOS

N°	Estudios seleccionados	Ref.	Resumen
ES01	Fake News Detection Using Machine Learning Ensemble Methods	62	El artículo propone un enfoque de conjunto de aprendizaje automático para la clasificación automatizada de artículos de noticias como falsos o reales, utilizando varias características textuales extraídas de los artículos.
ES02	Stochastic Gradient Descent-Ascent: Unified Theory and New Efficient Methods	22	El artículo propone un análisis de convergencia unificado que cubre una gran variedad de métodos de ascenso-descenso de gradiente estocástico (SGDA) y desarrolla varias variantes nuevas de SGDA, incluido un nuevo método de varianza reducida, nuevos métodos distribuidos con compresión y un nuevo método con aleatorización de coordenadas.
ES03	Distinguishing Fake and Real News of Twitter Data with the help of Machine Learning Techniques by	63	El artículo presenta una metodología para detectar noticias falsas utilizando técnicas de aprendizaje automático y procesamiento del lenguaje natural, demostrando una precisión mejorada en comparación con trabajos anteriores.
ES04	Predicting and Mitigating the Effect of Skewness on Credibility Assessment of Social Media Content Using Machine Learning: A Twitter Case Study	64	El artículo propone un modelo de aprendizaje automático integrado que combina el método Bayes naive gaussiano y la regresión logística para superar el problema de la paradoja de la precisión en conjuntos de datos de redes sociales altamente sesgados, y evalúa su desempeño frente a otros modelos de aprendizaje automático populares.
ES05	A Soft Two-Layers Voting Model for Fake News Detection A Soft Two-Layers Voting Model for Fake News Detection	65	El artículo presenta un nuevo enfoque de clasificador de votación ponderada de dos capas para la detección de noticias falsas que tiene como objetivo mejorar la precisión y la eficacia de los sistemas de detección de noticias falsas.
ES06	Analysis of a Two-Step Gradient Method with Two Momentum Parameters for Strongly Convex Unconstrained Optimization	66	El artículo presenta un análisis teórico y numérico de un método de gradiente de dos pasos con dos parámetros de momento para una optimización sin restricciones fuertemente convexa, demostrando que el parámetro de momento adicional puede proporcionar amortiguación de las oscilaciones y una convergencia más rápida en comparación con el método de bola pesada estándar, especialmente para funciones no cuadráticas y no convexas.

ES07	Stance Classification for Fake News Detection with Machine Learning	67	El artículo presenta un enfoque de aprendizaje profundo que utiliza un modelo CNN-BiLSTM con optimización SGD para detectar noticias falsas en árabe, logrando una precisión del 87,7% en el conjunto de datos AFND.
ES08	Techniques of Fake News Detection	68	El artículo proporciona una descripción general de las técnicas para detectar noticias falsas, incluidos enfoques de aprendizaje automático como bosques aleatorios, descenso de gradiente y máquinas de vectores de soporte, y analiza la importancia y los desafíos de esta tarea.
ES09	Fake News Detection Using Machine Learning Techniques	69	El artículo presenta un enfoque basado en el aprendizaje automático para detectar noticias falsas, que implica la recopilación de datos, preprocesamiento, extracción de características, entrenamiento y evaluación de modelos e implementación de una aplicación web para la predicción de noticias falsas en tiempo real.
ES10	Recent Advances in Stochastic Gradient Descent in Deep Learning	3	El artículo proporciona un análisis detallado de las aplicaciones contemporáneas de aprendizaje profundo, presenta varias versiones del descenso de gradiente estocástico y sus variantes, y analiza la brecha entre las condiciones teóricas y las aplicaciones prácticas.
ES11	Optimizing Hyperparameters and Performance Analysis of LSTM Model in Detecting Fake News on Social media	70	El artículo propone un modelo de aprendizaje profundo optimizado basado en LSTM para detectar noticias falsas en las redes sociales y evalúa su desempeño en los conjuntos de datos ISOT y LIAR después del ajuste de hiperparámetros utilizando métodos de búsqueda en cuadrícula y búsqueda aleatoria.
ES12	IDENTIFICACIÓN AUTOMÁTICA DE NOTICIAS FALSAS EN ESPAÑOL	71	El artículo presenta una metodología basada en el aprendizaje estadístico que detecta de forma precisa y automática las noticias falsas en función de cómo están redactadas.
ES13	Leia este artigo se você quiser aprender regressão logística	72	El artículo ofrece una introducción intuitiva a la regresión logística y demuestra su aplicación para analizar el efecto de los escándalos de corrupción en la probabilidad de reelección de los candidatos que se postulan para la Cámara de Representantes de Brasil.
ES14	Op-RMSprop (Optimized-Root Mean Square Propagation) Classification for Prediction of Polycystic Ovary Syndrome (PCOS) using Hybrid Machine Learning Technique	36	El artículo propone un nuevo modelo híbrido de aprendizaje automático denominado Optimized-SVLR (máquina de vectores de soporte optimizada y regresión logística) que combina la funcionalidad de SVM y la regresión logística y la optimiza utilizando el algoritmo RMSprop para predecir con precisión el síndrome de ovario poliquístico (SOP) con un 89,03 % de precisión, lo que supera a otros algoritmos de aprendizaje automático existentes.
ES15	A Comparison of Adaptive Moment Estimation (Adam) and RMSProp Optimisation Techniques for Wildlife Animal Classification Using Convolutional Neural Networks	33	Este artículo evalúa el rendimiento de tres arquitecturas CNN (DenseNet-121, ResNet-50 y AlexNet) utilizando dos técnicas de optimización (Adam y RMSProp) para la clasificación de animales salvajes, con un enfoque en el impacto de la tasa de aprendizaje en la precisión de los modelos.
ES16	MBGDT:Robust Mini-Batch Gradient Descent	25	El artículo presenta un nuevo método llamado Mini-Batch Gradient Descent with Trimming (MBGDT) que tiene como objetivo mejorar la robustez de los modelos de aprendizaje automático, especialmente en

			presencia de valores atípicos en datos de alta dimensión.
ES17	Prediction model of algal blooms using logistic regression and confusion matrix	73	El artículo describe un método para predecir floraciones de algas utilizando regresión logística y análisis de matriz de confusión, donde los pasos clave son encontrar factores ambientales marinos influyentes, realizar regresión logística, obtener una matriz de confusión y mejorar la sensibilidad y precisión ajustando el límite de decisión.
ES18	Desarrollo de una herramienta web de detección de Fake News en Ecuador, empleando algoritmos de Machine Learning	74	El artículo presenta el desarrollo de una herramienta web para la detección de noticias falsas en Ecuador utilizando algoritmos de aprendizaje automático y aprendizaje profundo.
ES19	Fake News Detection Model Basing on Machine Learning Algorithms	75	El artículo presenta un modelo de detección de noticias falsas que utiliza algoritmos de aprendizaje automático para clasificar las noticias como reales o falsas, y el clasificador Gradient Boosting logra la mayor precisión del 99%.
ES20	Fake News Detection Using a Logistic Regression Model and Natural Language Processing Techniques	76	El artículo presenta un sistema de detección de noticias falsas que utiliza técnicas de regresión logística y procesamiento del lenguaje natural para clasificar artículos de noticias como reales o falsos, siendo el modelo de regresión logística superior a otros clasificadores.
ES21	Performance Comparison of Machine Learning Classifiers for Fake News Detection	77	En este artículo se propone un modelo para detectar noticias falsas utilizando diferentes técnicas de vectorización de texto y clasificadores de aprendizaje automático, encontrando que una red neuronal y un SVM lineal tuvieron el mejor desempeño con un 94% de precisión.
ES22	Artificial Intelligence as an alternative in the detection of false news	78	El artículo ofrece una descripción general de cómo se puede utilizar la inteligencia artificial para detectar noticias falsas, incluido un análisis del problema de las noticias falsas, el papel de la IA y las técnicas de aprendizaje automático como el procesamiento del lenguaje natural y el aprendizaje profundo para abordar este problema, y el estado actual de la investigación y el desarrollo en esta área.
ES23	Machine Learning Technique Based Fake News Detection	79	El artículo presenta un enfoque basado en aprendizaje automático para detectar noticias falsas, donde los autores entrenaron y evaluaron varios modelos de aprendizaje automático y aprendizaje profundo en un conjunto de datos de artículos de noticias etiquetados como verdaderos, falsos, parcialmente falsos u otros, siendo el clasificador Naive Bayes el que tuvo el mejor desempeño entre los modelos probados.
ES24	Comparison Fake New Spanish	80	El artículo propone un modelo para detectar noticias falsas utilizando técnicas de aprendizaje automático y procesamiento del lenguaje natural, como bag-of-words, TF-IDF e incrustaciones de palabras, y compara el rendimiento de diferentes algoritmos de clasificación, incluidos SVM, regresión logística y XGBoost.
ES25	Detection of fake news in a new corpus for the Spanish language	81	El artículo presenta un nuevo corpus en español de noticias reales y falsas, proporciona estadísticas sobre la superposición de vocabulario entre las dos clases y categorías y evalúa varios enfoques de aprendizaje

			automático para detectar automáticamente noticias falsas en el corpus.
ES26	PREDICCIÓN DE NOTICIAS FALSAS CON MACHINE LEARNING	82	El artículo presenta un proyecto que tiene como objetivo utilizar técnicas de aprendizaje automático para predecir si un artículo de noticias es falso o no, con el fin de ayudar a los usuarios a consumir solo noticias veraces y evitar las consecuencias negativas de la rápida propagación de información errónea en línea.
ES27	SEK Clasificacion Tweets	83	El artículo describe el desarrollo de una herramienta para la clasificación y análisis de tweets utilizando técnicas de Machine Learning para identificar la influencia de cuentas automatizadas (bots) en trending topics en Ecuador.
ES28	Recent Advances in Stochastic Gradient Descent in Deep Learning	84	El artículo proporciona un análisis detallado de las aplicaciones contemporáneas de aprendizaje profundo, presenta varias versiones del descenso de gradiente estocástico y sus variantes, y analiza la brecha entre las condiciones teóricas y las aplicaciones prácticas.
ES29	NATURAL LANGUAGE PROCESSING COMO HERRAMIENTA PARA DETECTAR FAKE NEWS: APLICADO A LAS ELECCIONES DE ALCALDÍA	85	El artículo analiza el uso de técnicas de procesamiento del lenguaje natural (PLN) para detectar noticias falsas, con un enfoque en la aplicación de estas técnicas a los tuits de los candidatos en una elección a la alcaldía de Cali, Colombia.
ES30	FAKE NEWS DETECTION IN SPANISH USING DEEP LEARNING TECHNIQUES	86	Este artículo aborda el problema de la detección de noticias falsas en español utilizando técnicas de aprendizaje automático, explorando diferentes estrategias y arquitecturas de entrenamiento para establecer una línea base para futuras investigaciones en esta área, incluido el uso de aprendizaje por transferencia y traducción automática para aprovechar modelos entrenados con datos en inglés.
ES31	FAKE NEWS DETECTION SYSTEM USING LOGISTIC REGRESSION, DECISION TREE AND RANDOM FOREST	87	El artículo presenta un análisis comparativo de tres modelos de aprendizaje automático para diseñar un sistema de detección de noticias falsas, donde el modelo de árbol de decisión logra la mayor precisión del 99,64%.

Síntesis de los datos

¿Qué limitaciones presenta la regresión logística en la detección de noticias falsas al aplicar algoritmos de optimización?

La precisión que puede alcanzar la regresión logística en la detección de noticias falsas está condicionada tanto por las características intrínsecas del modelo como por la metodología de optimización empleada. Aunque el uso de algoritmos el Gradiente Descendente Estocástico (SGD) ha mostrado mejoras, según el estudio de Faizan Mian (2024), la falta de experimentación con otros métodos como AdaGrad, Adam, y RMSProp puede estar limitando el potencial de mejora en la precisión y eficiencia del modelo. Por lo tanto, es investigar más a fondo la influencia de diversos algoritmos de optimización para determinar si pueden contribuir a un aumento considerable de la precisión.

Del mismo modo, la insuficiencia de una optimización correcta y efectiva acarrea a un entrenamiento del modelo poco efectivo conforme a funciones de costos altos de los modelos de ML los cuales tienen presencia de múltiples

mínimos locales como respuesta hay una convergencia lenta e inadecuada en el entrenamiento teniendo un impacto negativo en la eficiencia del modelo.

¿Qué problemas pueden surgir si no se utilizan técnicas de optimización en modelos de machine learning?

La falta de optimización en modelos de machine learning puede resultar en varios problemas considerables. El uso ineficiente de recursos computacionales, así como los largos tiempos de entrenamiento con un uso excesivo de memoria y poder de procesamiento, pueden afectar gravemente al rendimiento del modelo, específicamente en términos de precisión y eficiencia. Como resultado, pueden existir predicciones inexactas y modelos nada robustos.

Usar estas técnicas de optimización es de vital importancia en modelos de machine learning porque ayudan a encontrar los valores óptimos de los parámetros del modelo de una manera eficiente, trayendo consigo una mejora general del modelo y una reducción en el error de predicción. No es práctico manejar grandes conjuntos de datos o modelos complejos sin una adecuada optimización.

En relación con esto, es palpable como la escasez en la optimización resulta en recursos computaciones desperdiciados e ineficientes esto trae consigo tiempos de entrenamientos prolongados, uso de memoria y procesador exagerados resultando en una disminución en la precisión y lleva a tener predicciones no exactas.

A continuación, en la TABLA XXII se presentan técnicas de optimización.

TABLA XXII
ALGORITMOS DE OPTIMIZACIÓN

Lenguaje de programación	• Python
Algoritmos	• Gradiente descendente • Gradiente descendente estocástico • Gradiente descendente por mini lotes • AdaGrad • Adam • RMSProp
Herramientas de software	• Jupyter Notebook • Google Colab
Bibliotecas	• Pandas • NLTK • Scikit-Learn • GoogleTrans

¿Los algoritmos de optimización ayudan a aumentar la precisión de los modelos de machine learning?

La relación que existe entre la precisión y la optimización de un modelo de machine learning es fundamental. Sin una adecuada optimización, los modelos, especialmente los de clasificación, pueden presentar dificultades para alcanzar una precisión alta. Debido a consecuencia de diversos problemas como una convergencia lenta o el sobreajuste del modelo. A continuación, se considera de forma crucial que los algoritmos, técnicas o métodos de optimización son importantes en el incremento de la precisión de los modelos de clasificación de ML por tal caso sin una adecuada optimización estos modelos pueden confrontar problemas en la convergencia y sobreajuste dificultando alcanzar una precisión alta y de esta manera se compromete la fiabilidad en los resultados obtenidos.

Revisión de informes

Dar formato al informe principal

Esta tarea hace referencia a la creación del informe con el proceso realizado para la obtención de resultados, es decir, el presente trabajo de RSL es el informe a considerar.

Conclusiones

De acuerdo a la RSL realizada se han obtenido las siguientes conclusiones:

- La regresión logística y otros modelos de machine learning requieren optimización para mejorar la precisión y eficiencia. El uso limitado de algoritmos de optimización, como AdaGrad, Adam y RMSProp, podría estar restringiendo el potencial de precisión en la detección de noticias falsas.
- La ausencia de técnicas de optimización puede llevar a un uso ineficiente de recursos computacionales, tiempos de entrenamiento prolongados y un consumo excesivo de memoria y poder de procesamiento, lo que finalmente afecta la precisión y robustez de los modelos.
- La correcta elección y aplicación de algoritmos de optimización es fundamental para evitar problemas de convergencia lenta y lograr un entrenamiento eficiente. Los problemas de múltiples mínimos locales y la falta de convergencia adecuada pueden llevar a un bajo rendimiento del modelo.
- Los algoritmos de optimización como el gradiente descendente estocástico y Adam son esenciales para alcanzar una alta precisión en modelos de clasificación, ya que ayudan a mitigar problemas como el sobreajuste y la convergencia inadecuada, garantizando así resultados más confiables y precisos en tareas de detección de noticias falsas.

Referencias:

(1) L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," Neurocomputing, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

(2) F. Mian, "Enhancing Integrated Logistic Support through Machine Learning-Driven Optimization: Predicting component category based on part number," 2024.

(3) Y. Tian, Y. Zhang, and H. Zhang, "Recent Advances in Stochastic Gradient Descent in Deep Learning," Feb. 01, 2023, MDPI. doi: 10.3390/math11030682.

(4) E. V. S. Raja, B. L. V. S. Aditya, and S. N. Mohanty, "Fake Profile Detection Using Logistic Regression and Gradient Descent Algorithm on Online Social Networks," EAI Endorsed Transactions on Scalable Information Systems, vol. 11, no. 1, 2024, doi: 10.4108/eetsis.4342.

(5) N. Sharma, R. Sharma, and N. Jindal, "Machine Learning and Deep Learning Applications-A Vision," Global Transitions Proceedings, vol. 2, no. 1, pp. 24–28, Jun. 2021, doi: 10.1016/j.gltp.2021.01.004.

(6) Bobadilla Jesús, "Machine Learning y Deep Learning Usando Python, Scikit y Keras Informática," 2020.

(7) M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," May 01, 2023, MDPI. doi: 10.3390/computers12050091.

(8) S. A. Grillo et al., "Adjacent Inputs with Different Labels and Hardness in Supervised Learning," IEEE Access, vol. 9, pp. 162487–162498, 2021, doi: 10.1109/ACCESS.2021.3131150.

(9) I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," May 01, 2021, Springer. doi: 10.1007/s42979-021-00592-x.

(10) D. Lopez-Bernal, D. Balderas, P. Ponce, and A. Molina, "Education 4.0: Teaching the basics of knn, lda and simple perceptron algorithms for binary classification problems," Future Internet, vol. 13, no. 8, Aug. 2021, doi: 10.3390/fi13080193.

(11) E. M. Rojas, "Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo," Apr. 2020.

(12) V. Castillo-Riquelme, P. Hermosilla-Urrea, J. P. Poblete-Tiznado, and C. Durán-Anabalón, "Fake news and unfunded beliefs in the post-truth age," Universitas (Stuttg), no. 34, pp. 87–108, Feb. 2021, doi: 10.17163/uni.n34.2021.04.

(13) U. DE Jaén, R. Lamoneda, C. Tutor, and D. Horno López, "FAKE NEWS AT HIGH SCHOOL: PROPUESTA DE INTERVENCIÓN EDUCATIVA EN COMPETENCIAS BÁSICAS PARA EDUCACIÓN EN DETECCIÓN DE NOTICIAS FALSAS (FAKE NEWS) Y MANIPULACIÓN A TRAVÉS DE LA RED," 2021. Accessed: Feb. 03, 2025. (Online). Available: https://hdl.handle.net/10953.1/13551

(14) D. P. Calvillo, B. J. Ross, R. J. B. Garcia, T. J. Smelter, and A. M. Rutchick, "Political Ideology Predicts Perceptions of the Threat of COVID-19 (and Susceptibility to Fake News About It)," Soc Psychol Personal Sci, vol. 11, no. 8, pp. 1119–1128, Nov. 2020, doi: 10.1177/1948550620940539.

(15) R. P. Bringula, A. E. Catacutan-Bangit, M. B. Garcia, J. P. S. Gonzales, and A. M. C. Valderama, "'Who is gullible to political disinformation?': predicting susceptibility of university students to fake news," Journal of Information Technology and Politics, vol. 19, no. 2, pp. 165–179, 2022, doi: 10.1080/19331681.2021.1945988.

(16) R. D. Joshi and C. K. Dhakal, "Predicting type 2 diabetes using logistic regression and machine learning approaches," Int J Environ Res Public Health, vol. 18, no. 14, Jul. 2021, doi: 10.3390/ijerph18147346.

(17) A. Alves et al., “Leia este artigo se você quiser aprender regressão logística”, doi: 10.1590/1678-987320287406.
(18) A. Arista, “Comparison Decision Tree and Logistic Regression Machine Learning Classification Algorithms to determine Covid-19,” Sinkron, vol. 7, no. 1, pp. 59–65, Jan. 2022, doi: 10.33395/sinkron.v7i1.11243.
(19) K. Kuntoro, “COMPARING ACCURACY OF LOGISTIC REGRESSION, K-NEAREST NEIGHBOR, SUPPORT VECTOR MACHINE, AND NAÏVE BAYES MODELS USING TRACKING ENSEMBLE MACHINE LEARNING,” JP J Biostat, vol. 24, no. 1, pp. 1–13, Oct. 2023, doi: 10.17654/0973514324001.
(20) Haji, Saad Hikmat, Adbulazeez, and Adnan Mohsin, “COMPARISON OF OPTIMIZATION TECHNIQUES BASED ON GRADIENT DESCENT ALGORITHM: A REVIEW PJAEE,” Feb. 2021.
(21) L. Chen and J. Bruna, “Beyond the Edge of Stability via Two-step Gradient Updates,” 2023.
(22) A. Beznosikov, E. Gorbunov, H. Berard, and M. Diro, “Stochastic Gradient Descent-Ascent: Unified Theory and New Efficient Methods,” 2023.
(23) Moreno Cabañas and Nelson bruno Andrés, “Estrategias de-seleccion-de-mini-batches-utilizando-procesos-puntuales-determinantales-para-el-entrenamiento-de-redes-neuronales-mediante-descenso-de-gradiente-estocastico,” Universidad de chile, 2023, doi: 10.58011/wd30-ad68.
(24) N. Hyder and G. Friedland, “Learning Rate Estimation for Stochastic Gradient Descent,” 2022. (Online). Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-155.html
(25) H. Wang, H. Luo, and Y. Wang, “MBGDT:Robust Mini-Batch Gradient Descent.”
(26) F. R. J. Simanungkalit, H. Hanifah, G. Ardaneswari, N. Hariadi, and B. D. Handari, “Prediction of students’ academic performance using ANN with mini-batch gradient descent and Levenberg-Marquardt optimization algorithms,” in Journal of Physics: Conference Series, IOP Publishing Ltd, Nov. 2021. doi: 10.1088/1742-6596/2106/1/012018.
(27) M. Nishchal, J. Mr, and N. Bhandari, “Computational Complexity of Gradient Descent Algorithm.”
(28) A. Rambidis, “Algorithmic Behaviours of Adagrad in Underdetermined Linear Regression,” 2023.
(29) S. Surono, A. Thobirin, Z. A. R. Hsm, A. Y. Astuti, B. R. Kp, and M. Oktavia, “Optimization of Fuzzy System Inference Model on Mini Batch Gradient Descent,” in Frontiers in Artificial Intelligence and Applications, IOS Press BV, Oct. 2022, pp. 224–232. doi: 10.3233/FAIA220387.
(30) A. Alblwi, “Improving the Adaptive Moment Estimation Optimization Methods for Modern Machine Learning,” ProQuest Dissertations & Theses, 27994120, University of Delaware, 2020.
(31) R. Selvaraj, T. Satheesh, V. Suresh, and V. Yathavaraj, “Optimized Machine Learning for CHD Detection using 3D CNN-based Segmentation, Transfer Learning and Adagrad Optimization,” SSRG International Journal of Electrical and Electronics Engineering, vol. 10, no. 3, pp. 20–34, Mar. 2023, doi: 10.14445/23488379/IJEEE-V10I3P103.
(32) P. Kairouz, K. Rush, A. Thakurta, M. Belkin, and S. Kpotufe, “(Nearly) Dimension Independent Private ERM with AdaGrad Rates via Publicly Estimated Subspaces Google Research,” 2021.
(33) I. H. Kartowisastro and J. Latupapua, “A Comparison of Adaptive Moment Estimation (Adam) and RMSProp Optimisation Techniques for Wildlife Animal Classification Using Convolutional Neural Networks,” Revue d’Intelligence Artificielle, vol. 37, no. 4, pp. 1023–1030, Aug. 2023, doi: 10.18280/ria.370424.
(34) M. Reyad, A. M. Sarhan, and M. Arafa, “A modified Adam algorithm for deep neural network optimization,” Neural Comput Appl, vol. 35, no. 23, pp. 17095–17112, Aug. 2023, doi: 10.1007/s00521-023-08568-z.
(35) J. Kang, X. Zhu, L. Shen, and M. Li, “Fault diagnosis of a wave energy converter gearbox based on an Adam optimized CNN-LSTM algorithm,” Renew Energy, vol. 231, Sep. 2024, doi: 10.1016/j.renene.2024.121022.
(36) R. P. Kiran, A. Professor, and N. N. C, “Op-RMSprop (Optimized-Root Mean Square Propagation) Classification for Prediction of Polycystic Ovary Syndrome (PCOS) using Hybrid Machine Learning Technique.” (Online). Available: www.ijacsa.thesai.org
(37) P. Arafin, A. M. Billah, and A. Issa, “Deep learning-based concrete defects classification and detection using semantic segmentation,” Struct Health Monit, vol. 23, no. 1, pp. 383–409, Jan. 2024, doi: 10.1177/14759217231168212.

(38) H. Yun, “Prediction model of algal blooms using logistic regression and confusion matrix,” International Journal of Electrical and Computer Engineering, vol. 11, no. 3, pp. 2407–2413, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2407-2413.
(39) D. Krstinić, M. Braović, L. Šerić, and D. Božić-Štulić, “Multi-label Classifier Performance Evaluation with Confusion Matrix,” Academy and Industry Research Collaboration Center (AIRCC), Jun. 2020, pp. 01–14. doi: 10.5121/csit.2020.100801.
(40) D. Valero-Carreras, J. Alcaraz, and M. Landete, “Comparing two SVM models through different metrics based on the confusion matrix,” Comput Oper Res, vol. 152, Apr. 2023, doi: 10.1016/j.cor.2022.106131.
(41) M. A. Khder, “Web scraping or web crawling: State of art, techniques, approaches and application,” International Journal of Advances in Soft Computing and its Applications, vol. 13, no. 3, pp. 144–168, 2021, doi: 10.15849/ijasca.211128.11.
(42) I. Kolyshkina and S. Simoff, “Interpretability of Machine Learning Solutions in Public Healthcare: The CRISP-ML Approach,” Front Big Data, vol. 4, May 2021, doi: 10.3389/fdata.2021.660206.
(43) S. Studer et al., “Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,” Mach Learn Knowl Extr, vol. 3, no. 2, pp. 392–413, Jun. 2021, doi: 10.3390/make3020020.
(44) S. Gholizadeh, “Top Popular Python Libraries in Research,” Journal of Robotics and Automation Research, vol. 3, no. 2, pp. 142–145, 2022, Accessed: Feb. 03, 2025. (Online). Available: https://doi.org/10.22541/au.164580055.55493761/v1
(45) A. C. . Müller, Sarah. Guido, and Kristian. Rother, Einführung in Machine Learning mit Python: Praxiswissen Data Science. Heidelberg, Germany: dpunkt.verlag GmbH, 2017.
(46) M. N. Rao, “A Comparative Analysis of Deep Learning Frameworks and Libraries,” International Journal of Intelligent Systems and Applications in Engineering IJISAE, vol. 2023, no. 2s, pp. 337–342, 2023, Accessed: Feb. 03, 2025. (Online). Available: https://ijisae.org/index.php/IJISAE/article/view/2707
(47) F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Jan. 2012, (Online). Available: http://arxiv.org/abs/1201.0490
(48) M. Wang and F. Hu, “The application of nltk library for python natural language processing in corpus research,” Theory and Practice in Language Studies, vol. 11, no. 9, pp. 1041–1049, Sep. 2021, doi: 10.17507/tpls.1109.09.
(49) E. J. Rifano, Abd. C. Fauzan, A. Makhi, E. Nadya, Z. Nasikin, and F. N. Putra, “Text Summarization Menggunakan Library Natural Language Toolkit (NLTK) Berbasis Pemrograman Python,” ILKOMNIKA: Journal of Computer Science and Applied Informatics, vol. 2, no. 1, pp. 8–17, Apr. 2020, doi: 10.28926/ilkomnika.v2i1.32.
(50) C. and C. C. and G. L. and J. E. and L. B. and P. S. and S. T. Cappi, “Dataset Definition Standard (DDS),” arXiv preprint arXiv:2101.03020, Jun. 2021, doi: 10.48550/arXiv.2101.03020.
(51) W. Vallejo, C. Díaz-Uribe, and C. Fajardo, “Google Colab and Virtual Simulations: Practical e-Learning Tools to Support the Teaching of Thermodynamics and to Introduce Coding to Students,” ACS Omega, vol. 7, no. 8, pp. 7421–7429, Mar. 2022, doi: 10.1021/acsomega.2c00362.
(52) S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,” Apr. 01, 2020, MDPI AG. doi: 10.3390/info11040193.
(53) P. Kode Program Dan Simulasi, “Pembuatan Kode Program Dan Simulasi Skema Masakan ACD Menggunakan Jupyter Notebook Pada Pemrograman Python Anaconda,” Politeknik LPP Yogyakarta, 2023. Accessed: Feb. 03, 2025. (Online). Available: https://repository.polteklpp.ac.id/id/eprint/3812
(54) R. R. Rajalaxmi, L. V. Narasimha Prasad, B. Janakiramaiah, C. S. Pavankumar, N. Neelima, and V. E. Sathishkumar, “Optimizing Hyperparameters and Performance Analysis of LSTM Model in Detecting Fake News on Social media,” ACM Transactions on Asian and Low-Resource Language Information Processing, Mar. 2022, doi: 10.1145/3511897.
(55) I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, “Fake News Detection Using Machine Learning Ensemble Methods,” Complexity, vol. 2020, 2020, doi: 10.1155/2020/8885861.
(56) A. John and S. Journals, “Fake News Detection Using N-Gram Analysis and Machine Learning Algorithms,” 2022, doi: 10.37591/JoMCCMN.
(57) M. Alsafadi, “Stance Classification for Fake News Detection with Machine Learning,” Technology, Engineering & Mathematics (EPSTEM), vol. 22, 2023, (Online). Available: www.isres.org

(58) K. Martínez-Gallego, A. M. Álvarez-Ortiz, and J. D. Arias-Londoño, “Fake News Detection in Spanish Using Deep Learning Techniques,” Oct. 2021, (Online). Available: http://arxiv.org/abs/2110.06461
(59) S. Basharat, S. Afzal, A. M. Bamhdi, S. Khurshid, and M. Chachoo, “Predicting and Mitigating the Effect of Skewness on Credibility Assessment of Social Media Content Using Machine Learning: A Twitter Case Study,” International Journal of Computer Theory and Engineering, vol. 15, no. 3, pp. 101–110, Aug. 2023, doi: 10.7763/IJCTE.2023.V15.1338.
(60) N. J. Mafla Checa, “Identificación automática de noticias falsas en español utilizando técnicas de minería de datos y procesamiento del lenguaje natural.,” Escuela Politécnica Nacional, Quito, 2021. Accessed: Feb. 03, 2025. (Online). Available: http://bibdigital.epn.edu.ec/handle/15000/21606
(61) O. L. Londoño, P. Luis, F. Maldonado, G. Liccy, and C. Calderón Villafáñez, “GUÍA PARA CONSTRUIR ESTADOS DEL ARTE.”
(62) “Una guía corta para escribir Revisiones Sistemáticas de Literatura Parte 2”, (Online). Available: http://orcid.org/0000-0003-3043-3037
(63) I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, “Fake News Detection Using Machine Learning Ensemble Methods,” Complexity, vol. 2020, 2020, doi: 10.1155/2020/8885861.
(64) A. Shah, “Distinguishing Fake and Real News of Twitter Data with the help of Machine Learning Techniques.”
(65) S. Basharat, S. Afzal, A. M. Bamhdi, S. Khurshid, and M. Chachoo, “Predicting and Mitigating the Effect of Skewness on Credibility Assessment of Social Media Content Using Machine Learning: A Twitter Case Study,” International Journal of Computer Theory and Engineering, vol. 15, no. 3, pp. 101–110, Aug. 2023, doi: 10.7763/IJCTE.2023.V15.1338.
(66) H. Ei Wynne, K. Thanda Swe, H. Ei, K. Thanda, and A. Hnin Ei Wynne, “A Soft Two-Layers Voting Model for Fake News Detection,” Journal of Engineering Research, vol. 8, p. 2024, 2024.
(67) G. V. Krivovichev and V. Y. Sergeeva, “Analysis of a Two-Step Gradient Method with Two Momentum Parameters for Strongly Convex Unconstrained Optimization,” Algorithms, vol. 17, no. 3, Mar. 2024, doi: 10.3390/a17030126.
(68) M. Alsafadi, “Stance Classification for Fake News Detection with Machine Learning,” Technology, Engineering & Mathematics (EPSTEM), vol. 22, 2023, (Online). Available: www.isres.org
(69) H. Garg and A. Goyal, “Techniques of Fake News Detection,” International Journal of Civil, Mechanical and Energy Science, vol. 6, no. 2, pp. 6–9, 2020, doi: 10.22161/ijcmes.622.
(70) L. Venkata Krishna, “Fake News Detection Using Machine Learning Techniques,” International Journal of Research and Analytical Reviews, 2024, (Online). Available: https://www.researchgate.net/publication/379644758
(71) R. R. Rajalaxmi, L. V. Narasimha Prasad, B. Janakiramaiah, C. S. Pavankumar, N. Neelima, and V. E. Sathishkumar, “Optimizing Hyperparameters and Performance Analysis of LSTM Model in Detecting Fake News on Social media,” ACM Transactions on Asian and Low-Resource Language Information Processing, Mar. 2022, doi: 10.1145/3511897.
(72) “ESPE_herramienta web de detección de Fake News en Ecuador”.
(73) A. Alves et al., “Leia este artigo se você quiser aprender regressão logística”, doi: 10.1590/1678-987320287406.
(74) H. Yun, “Prediction model of algal blooms using logistic regression and confusion matrix,” International Journal of Electrical and Computer Engineering, vol. 11, no. 3, pp. 2407–2413, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2407-2413.
(75) “ESPE_herramienta web de detección de Fake News en Ecuador”.
(76) M. A. Taha, H. D. A. Jabar, and W. K. Mohammed, “Fake News Detection Model Basing on Machine Learning Algorithms,” Baghdad Science Journal, vol. 21, no. 8, pp. 2771–2781, 2024, doi: 10.21123/bsj.2024.8710.
(77) J. A. Adeyiga, P. G. Toriola, T. E. Abioye(Ogunbiyi), A. E. Oluwatosin, and oluwasefunmi ’Tale Arogundade, “Fake News Detection Using a Logistic Regression Model and Natural Language Processing Techniques,” Jul. 14, 2023. doi: 10.21203/rs.3.rs-3156168/v1.
(78) Proceedings of the 2nd International Conference on Inventive Research in Computing Applications (ICIRCA 2020): 15-17 July, 2020. IEEE, 2020.
(79) A.-R. Avanzada -Rita, “Artículo de investigación. REVISTA TIA”.

(80) B. Kumar Sutradhar, M. Zonaid, N. Jahan Ria, S. Rashed Haider Noori, and A. Affiliations, “Machine Learning Technique Based Fake News Detection.”
(81) Comparación de rendimiento del aprendizaje automático Clasificadores para la detección de noticias falsas.
(82) J. P. Posadas-Durán, H. Gomez-Adorno, G. Sidorov, and J. J. M. Escobar, “Detection of fake news in a new corpus for the Spanish language,” Journal of Intelligent and Fuzzy Systems, vol. 36, no. 5, pp. 4868–4876, 2019, doi: 10.3233/JIFS-179034.
(83) I. Krasimirova Hristova, “PREDICCIÓN DE NOTICIAS FALSAS CON MACHINE LEARNING.”
(84) M. Emiliano and G. Pico, “DECLARACIÓN JURAMENTADA.”
(85) Y. Tian, Y. Zhang, and H. Zhang, “Recent Advances in Stochastic Gradient Descent in Deep Learning,” Feb. 01, 2023, MDPI. doi: 10.3390/math11030682.
(86) J. David, P. Directora, D. E. Proyecto, F. De, C. Administrativas, and Y. E. Economía, “NATURAL LANGUAGE PROCESSING COMO HERRAMIENTA PARA DETECTAR FAKE NEWS: APLICADO A LAS ELECCIONES DE ALCALDÍA DE CALI PARA EL PERIODO 2020-2023,” 2019.
(87) K. Martínez-Gallego, A. M. Álvarez-Ortiz, and J. D. Arias-Londoño, “Fake News Detection in Spanish Using Deep Learning Techniques,” Oct. 2021, (Online). Available: http://arxiv.org/abs/2110.06461
(88) O. O. A., O. I. R., and A. B. A., “Fake News Detection System Using Logistic Regression, Decision Tree and Random Forest,” British Journal of Computer, Networking and Information Technology, vol. 7, no. 1, pp. 115–121, May 2024, doi: 10.52589/bjcnit-ioyrpy7g.

Anexo 8. Certificado traducción resumen del TIC

CERTIFICACIÓN DE TRADUCCIÓN

Loja, 19 de marzo de 2025

Lic. Viviana Valdivieso Loyola Mg. Sc.
DOCENTE DE INGLÉS

A petición verbal de la parte interesada:

CERTIFICA:

Que, desde mi legal saber y entender, como profesional en el área del idioma inglés, he procedido a realizar la traducción del resumen, correspondiente al Trabajo de Integración Curricular titulado **Optimización de la Precisión en la Detección de Noticias Falsas de política en español mediante la Aplicación de Algoritmos de Optimización en la Regresión Logística.** de la autoría de: **Santiago Emanuel Tene Castillo,** portador de la cédula de identidad número **1105174989**

Para efectos de traducción se han considerado los lineamientos que corresponden a un nivel de inglés técnico, como amerita el caso.

Es todo cuanto puedo certificar en honor a la verdad, facultando al portador del presente documento, hacer uso del mismo, en lo que a bien tenga.

Atentamente. -

Firmado electrónicamente por:
VIVIANA DEL CISNE
VALDIVIESO LOYOLA

Lic. Viviana Valdivieso Loyola Mg. Sc.
1103682991

N° Registro Senescyt 4to nivel **1031-2021-2296049**
N° Registro Senescyt 3er nivel **1008-16-1454771**