Logotipo, nombre de la empresa

Descripción generada automáticamente

FACULTAD DE INGENIERÍA

INGENIERÍA EN CIBERSEGURDAD

titulo

Comparación de Tiny BERT, Redes Neuronales Profundas y Naive Bayes para la

Detección de Correos Electrónicos de Phishing: Análisis de Precisión y Exactitud

Tesis de pregrado para optar al título de Ingeniero en Ciberseguridad

Autor:

Walter Cristopher Rubilar Morales

Profesor Guía:

Marcelo Reyes Rogget

Santiago, Chile

2024

Imagen que contiene dibujo

Descripción generada automáticamente

**DECLARACIÓN DE ORIGINALIDAD**

Se declara por este medio que el trabajo de titulación presentado para su defensa y evaluación es original; las fuentes, herramientas y aplicaciones utilizadas que contribuyeron a la investigación realizada están debidamente citadas en el texto y acreditadas en el apartado de las referencias, conforme con los requisitos que establece el estilo bibliográfico APA y respetando los aspectos que conciernen a la propiedad intelectual.

Por lo tanto, ante cualquier falta de integridad académica encontrada y que atente contra la Ley N°17.336 de Propiedad Intelectual, se asume la responsabilidad que representa para tal efecto, dejando constancia de ello, con fecha 10 de septiembre del año 2024, en la ciudad de Santiago.

Facultad de Ingeniería

Departamento de Ciencias de la Ingeniería

Título del trabajo "Comparación de Tiny BERT, Redes Neuronales Profundas y Naive Bayes para la Detección de Correos Electrónicos de Phishing: Análisis de Precisión y Exactitud"

Un dibujo de una persona

Descripción generada automáticamente con confianza baja

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Walter Cristopher Rubilar Morales

Dedicatoria

Quiero dedicar esta tesis y mi etapa universitaria a mi familia compuesta por mis abuelitos de nombre Walter Hugo Morales Castro e Isabel del Rosario Rubilar Serra y a mi madre de nombre Marcela Antonieta Morales Rubilar, quienes me acompañaron de manera incondicional en este largo y arduo camino, tanto universitario como en el camino de la vida, para ellos, es la presente tesis.

Agradecimientos

En este apartado, quiero además agradecer a todas las personas que me acompañaron en este proceso, a mi familia que mencione precedentemente y amigos que me ayudaron y apoyaron en momentos difíciles y compañeros con los cuales sorteamos cada dificultad de la presente carrera, además agradecer al profesor guía de la tesis y a cada uno de los docentes de la Universidad Andrés Bello, quienes me guiaron en este camino.

Comparación de Tiny BERT, Redes Neuronales Profundas y Naive Bayes para la Detección de Correos Electrónicos de Phishing: Análisis de Precisión y Exactitud

Walter Rubilar Morales

Engineering Faculty, Universidad Andrés Bello, Quillota 980, Viña del Mar 2520000, Chile; walter.c.rubilar.morales@gmail.com

**Resumen:** Los ataques de phishing constituyen una amenaza persistente en la ciberseguridad, y la detección precisa de correos electrónicos maliciosos es esencial para proteger a los usuarios. La eficacia de los modelos de aprendizaje automático para esta tarea puede variar considerablemente. Este estudio presenta un análisis comparativo experimental de tres modelos prominentes para la detección de phishing: Bert Tiny Bert, redes neuronales profundas y el clasificador Naive Bayes. En función de lo anterior, los resultados del presente artículo se resumen en que Naive Bayes tuvo un buen desempeño con un 86% de precisión en validación, mientras que las redes neuronales profundas fueron las más efectivas con un 92% de precisión en la validación, en tanto Tiny Bert y Bert Fine-Tuned mostraron un rendimiento moderado debido a limitaciones de recursos, no obstante su rendimiento fue superior a un 60%, demostrando un desempeño superior a un 50%, lo que significa que su desempeño en precisión de validación es bueno y se aleja de valores de aleatoriedad.

**Abstract:** Phishing attacks constitute a persistent threat in cybersecurity, and the accurate detection of malicious emails is essential to protect users. The effectiveness of machine learning models for this task can vary significantly. This study presents an experimental comparative analysis of three prominent models for phishing detection: Bert, Tiny Bert, deep neural networks, and the Naive Bayes classifier. Based on the above, the results of this article can be summarized as Naive Bayes having performed well with 86% accuracy in validation, while deep neural networks were the most effective with 92% accuracy in validation. Meanwhile, Tiny Bert and Bert Fine-Tuned showed moderate performance due to resource limitations, although their performance exceeded 60%, demonstrating a performance well above 50%, indicating that their validation accuracy is good and far from random values.

**Keywords:** Phishing; Machine Learning; Tiny BERT; Redes Neuronales Profundas; Naive Bayes; Comparación Experimental.

1. Introducción

Los ataques de phishing representan una amenaza significativa y persistente en el ámbito de la ciberseguridad, poniendo en riesgo tanto a individuos como a organizaciones (Kaspersky, 2023). Estos ataques suelen involucrar correos electrónicos engañosos diseñados para que los destinatarios(víctimas) revelen información sensible o ingresen a links que les solicitaran usuarios y contraseñas o que instalen software malicioso (links o archivos que contienen algún malware que, al interactuar con el referido documento, este infecte el equipo que está ocupando la víctima). La creciente sofisticación de las técnicas de phishing requiere el desarrollo de mecanismos de detección efectivos para proteger contra tales amenazas (Kaspersky, 2023).

El uso creciente de correos electrónicos como vector de ciberataques enfocados en ingeniería social destaca la necesidad de desarrollar métodos efectivos para su detección y clasificación. Estos ataques, diseñados para engañar a las víctimas mediante comunicaciones persuasivas, requieren técnicas avanzadas de análisis de lenguaje natural (PLN) para identificar patrones y comportamientos maliciosos que pudiese tener el o los atacantes.

La aplicación de modelos de aprendizaje automático ha surgido como un enfoque prometedor e importante para mejorar la detección de phishing. Se han propuesto varios modelos, cada uno con características distintas y posibles ventajas. Tiny BERT, una versión compacta del modelo BERT, destaca en la comprensión de matices contextuales y semánticos en el contenido de los correos electrónicos. Las redes neuronales profundas, conocidas por su capacidad para aprender patrones complejos a partir de grandes volúmenes de datos, ofrecen una herramienta poderosa para detectar intentos de phishing y poder determinar patrones de ataques. Por otro lado, el clasificador Naive Bayes, con su enfoque probabilístico, sigue siendo un método ampliamente utilizado debido a su simplicidad de uso y su eficiencia.

El enfoque en la comparación de métodos de Machine Learning es esencial para determinar qué técnicas ofrecen el mejor rendimiento en este contexto. La elección de los tres métodos seleccionados Clasificador Naive Bayes, redes neuronales profundas y Tiny BERT se basa en su relevancia y aplicación en el análisis de texto. El Clasificador Naive Bayes es conocido por su simplicidad y eficacia en problemas de clasificación de texto, proporcionando una base sólida para comparación. Las redes neuronales profundas, con su capacidad para capturar complejas representaciones del lenguaje, ofrecen un contraste significativo con modelos más simples de identificación o clasificación. Por último, Tiny BERT, como una versión optimizada del modelo BERT, combina alta precisión con eficiencia computacional, siendo ideal para tareas que requieren un procesamiento rápido y eficaz.

El análisis cuantitativo de estos métodos permitirá evaluar su precisión en la detección de correos electrónicos maliciosos y su capacidad para identificar patrones de ingeniería social. Este enfoque comparativo no solo contribuirá al conocimiento académico en el campo de la ciberseguridad, sino que también proporcionará herramientas prácticas para mejorar la protección contra ataques basados en texto (correos electrónicos). La comprensión de cómo cada método maneja el preprocesamiento, la tokenización y la clasificación de datos es esencial para desarrollar soluciones más robustas y efectivas en la lucha contra las amenazas digitales.

A pesar de los avances, y de los distintos modelos de aprendizaje supervisado que existen, aún se mantiene el debate sobre qué modelo ofrece la solución más efectiva para la detección de ataques de ingeniería social como el ataque phishing. Mientras que Tiny BERT y las redes neuronales profundas pueden proporcionar un análisis más sofisticado, Naive Bayes ofrece una alternativa práctica con menores demandas computacionales, por lo que se hace más practico en algunos casos, trabajar con el clasificador ya señalado, no obstante al ser un clasificador, solo Naive Bayes podrá clasificar datos que estén en la base de datos con la que se entrenó, en tanto las redes neuronales profundas y las redes neuronales Tiny Bert, tienen la facultad de reentrenarse según el feedback de información que vayan recibiendo.

Este estudio tiene como objetivo realizar un análisis comparativo experimental de estos tres modelos Tiny BERT, redes neuronales profundas y Naive Bayes para evaluar su rendimiento en la identificación de correos electrónicos maliciosos.

La importancia de este trabajo radica en su potencial para mejorar las metodologías de detección de phishing al proporcionar una comparación exhaustiva de estos modelos. Los hallazgos de este estudio nos permitirán identificar insights (percepción o entendimiento) sobre cuál modelo funciona mejor en diferentes condiciones, contribuyendo así al desarrollo de estrategias anti-phishing más efectivas. Las conclusiones clave destacarán las fortalezas y limitaciones de cada enfoque, proporcionando orientación para futuras investigaciones y la implementación protocolos en las prácticas de ciberseguridad.

Para profundizar en las bases de este proyecto experimental, se seleccionaron tres métodos de Machine Learning los cuales son, un Clasificador Naive Bayes, redes neuronales profundas y redes neuronales Tiny BERT.

El objetivo es realizar una comparación cuantitativa de su rendimiento en el reconocimiento de correos electrónicos y su contenido, específicamente en el contexto de vectores de ciberataques enfocados en ingeniería social. Estos ataques suelen utilizar correos electrónicos diseñados para engañar a la víctima, por lo que es esencial comprender y aplicar métodos de análisis de lenguaje natural (NLP).

**2.- Materiales y Métodos**

**2.1.- Estado del Arte**

**2.1.1.- Procesamiento de Lenguaje Natural.**

El Procesamiento de Lenguaje Natural (PLN) es una rama de la Inteligencia Artificial (IA) centrada en el diseño de métodos y algoritmos que toman como entrada o producen como salida datos en la forma de lenguaje humano [1]. Esto puede venir en forma de texto o audio, y una vez que el audio es transcrito, ambos tipos de datos tienen un análisis común. ((Bravo-Márquez & Dunstan, 2021)

El procesamiento del lenguaje natural (PLN) combina la lingüística computacional (modelización del lenguaje humano basada en reglas) con modelos estadísticos y de machine learning para que los ordenadores y dispositivos digitales reconozcan, comprendan y generen texto y voz.

El PLN, es una rama de la inteligencia artificial (IA), se encuentra en el corazón de las aplicaciones y los dispositivos que pueden realizar las siguientes acciones. (ibm, 2024)

* traducir texto de un idioma a otro
* responder a órdenes escritas u orales
* reconocer o autenticar usuarios por voz
* resumir grandes volúmenes de texto
* evaluar la intención o el sentimiento de un texto o discurso
* generar texto o gráficos u otros contenidos a petición.

Hoy en día, la mayoría de la gente ha interactuado con el PLN (procesamiento del lenguaje natural) en forma de sistemas GPS operados por voz, asistentes digitales, softwares *speech to text*, chatbots de atención al cliente y otras comodidades para el consumidor. Pero el PLN también desempeña un papel cada vez más importante en las soluciones empresariales que ayudan a agilizar y automatizar las operaciones de negocio, aumentar la productividad de los empleados y simplificar los procesos empresariales de misión crítica. (ibm, 2024)

Tareas de PLN

El lenguaje humano está lleno de ambigüedades que hacen increíblemente difícil escribir programas informáticos que determinen con precisión el significado de un texto o de unos datos vocales. Homónimos, homófonos, sarcasmo, modismos, metáforas, excepciones gramaticales y de uso, variaciones en la estructura de las frases... Éstas son sólo algunas de las irregularidades del lenguaje humano que los seres humanos tardan años en aprender, pero que los programadores deben enseñar a las aplicaciones basadas en el lenguaje natural a reconocer y entender con precisión desde el principio, si quieren que dichas aplicaciones sean útiles. (ibm, 2024)

Varias tareas de PLN descomponen los datos de texto y voz humanos de forma que ayuden al ordenador a dar sentido a lo que está ingiriendo o aprendiendo. (ibm, 2024)

El Procesamiento de Lenguaje Natural, se ocupa de la capacidad de las máquinas para entender, interpretar y generar lenguaje humano de una manera que sea tanto significativa como útil. El objetivo del PLN no es solo entender palabras de forma aislada, sino también captar el contexto en el que se encuentran. Algunas tareas comunes en el PLN incluyen:

* **Clasificación de Texto:** Determinar el sentimiento en una reseña, identificar si un correo electrónico es spam, evaluar la corrección gramatical de una oración o establecer si dos oraciones están lógicamente relacionadas.
* **Etiquetado de Palabras:** Identificar componentes gramaticales de una oración, como sustantivos y verbos, o entidades nombradas como personas, ubicaciones y organizaciones.
* **Generación de Texto:** Completar un texto de manera automática, llenar espacios en blanco o traducir texto a otros idiomas.
* **Extracción de Respuestas:** Responder a preguntas basadas en un contexto dado.
* **Resumen de Texto:** Resumir un texto largo en una versión más corta que conserve la esencia del contenido.

El PLN también se extiende al reconocimiento de voz y la visión por computadora, como la transcripción de muestras de audio y la descripción de imágenes.

**Integración en Aplicaciones NLP:** La integración de modelos como TinyBERT en aplicaciones NLP se realiza a menudo mediante interfaces gráficas que facilitan la interacción del usuario con el sistema. Por ejemplo, en una aplicación desarrollada con Tkinter, un botón (tk.Button) puede desencadenar procesos como la carga y preprocesamiento de datos, la aplicación del modelo de clasificación y la visualización de resultados. Esta integración permite ejecutar tareas de NLP de manera eficiente y accesible para los usuarios finales**.** (Tkinter — Python Interface To Tcl/Tk, s. f.)

**2.1.2.- Hugging Face y TinyBERT**

Hugging Face es una empresa de tecnología de vanguardia especializada en el procesamiento del lenguaje natural (PLN). Su plataforma de código abierto ha transformado el campo del PLN al proporcionar herramientas y modelos preentrenados accesibles para desarrolladores e investigadores. Una de las principales contribuciones de Hugging Face es la biblioteca Transformers, que incluye modelos avanzados como BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), y T5 (Text-To-Text Transfer Transformer). (Huggingface, 2024).

TinyBERT es una variante optimizada del modelo BERT, diseñada para proporcionar una mayor eficiencia computacional mientras mantiene un alto rendimiento en tareas de PLN. A diferencia de BERT, cuyo tamaño y requisitos computacionales pueden ser una limitación en ciertos entornos, TinyBERT reduce la complejidad del modelo, permitiendo un procesamiento más rápido y menos costoso. Esta eficiencia se logra mediante la reducción del número de capas y parámetros del modelo, mientras se conserva una alta capacidad para entender el contexto del lenguaje.

En el artículo "TinyBERT: Distilling BERT for Natural Language Understanding", se describe el proceso de aprendizaje de TinyBERT en dos etapas: pre-entrenamiento y ajuste fino. Durante el pre-entrenamiento, TinyBERT aprende el conocimiento general de lenguaje de BERT. El ajuste fino, por otro lado, adapta el modelo a tareas específicas mediante una técnica conocida como destilación, que implica la transferencia del conocimiento del modelo grande (BERT) al modelo más compacto (TinyBERT). Esta metodología incluye la destilación general, que ayuda a TinyBERT a aprender el conocimiento general embebido en BERT, y la destilación específica de tareas, que ajusta el modelo a tareas concretas. ((Jiao et al., 2020)

Los modelos BERT y su variante Tiny BERT, utilizan ampliamente el Transformer. Es un mecanismo de atención capaz de aprender contextos entre las palabras de un texto e incluso las subpalabras a diferencia de los modelos direccionales que comprenden la entrada del texto en una secuencia (de derecha a izquierda o de izquierda a derecha), los codificadores del Transformador son enormemente diferentes. ¿Por qué? Porque pueden leer una secuencia de una sola vez, de ahí el término bidireccional. El uso de esta característica permite al modelo aprender el contexto de una palabra según su entorno. (Huggingface, 2024).

Referente al modelo BERT, podemos mencionar que este se basa en bloques de codificadores Transformers bidireccionales [19], que pueden pre entrenarse de manera independiente en cada gran conjunto de datos y luego ajustarse en una tarea específica para hacerlo adecuado para la tarea final. ((Lv et al., 2024)

Antes de entrenar, BERT realizará primero un procesamiento simple del texto o dataset, el cual se entrenará el modelo BERT o los derivados de esto, añadiendo embeddings de palabras y codificación de posiciones. El propósito del embedding de palabras es convertir la representación digital del texto objetivo en un vector, para capturar la relación entre palabras en un espacio de alta dimensión. Mientras tanto, dado que no hay procesamiento de la información de la posición de las palabras en el codificador de BERT, es posible que no se pueda realizar un análisis semántico preciso basado en la posición de las palabras. Por lo tanto, se necesita agregar codificación de posiciones después de la capa de embedding de palabras para compensar la falta de información de posición. Así, la entrada final del modelo BERT es Ei = Em(i) + Pos(i), donde Em(i) es el embedding de palabras, y Pos(i) es la codificación de posiciones. El dataset de entrada también se transforma de un vector a una matriz de vectores. ((Lv et al., 2024).

**2.1.3.- Implementación y Configuración del Tokenizador WordPiece**

La tokenización es un paso fundamental en el procesamiento de texto, transformando el texto en una secuencia de tokens que el modelo puede procesar. TinyBERT utiliza el algoritmo WordPiece para tokenizar el texto, que segmenta palabras en subpalabras para mejorar la eficiencia y manejar variaciones morfológicas del lenguaje.

**Preprocesamiento de Tokens:** Incluye la conversión del texto a minúsculas, la eliminación de acentos y la segmentación del texto en tokens. Estos pasos no son reversibles; por ejemplo, la reconstrucción exacta del texto original a partir de los tokens no es posible. (Huggingface, 2024).

**Configuración Personalizada:**

**Opciones de Pre-tokenización:** El algoritmo WordPiece puede configurarse para aplicar solo la tokenización sin pasos adicionales, lo que se realiza configurando lowercase=False, strip\_accents=False y split=False. Las entradas deben ser tensores de cadena pre-divididos o tensores irregulares.

**Opciones de Salida:** La salida puede ser un tensor irregular (para lotes de cadenas) o un tensor denso (para cadenas únicas) con longitud de secuencia especificada.

**Tipo de Salida:** El tipo de dato de salida se puede ajustar mediante el argumento dtype, que puede ser un tipo entero o de cadena.

Estas metodologías avanzadas y técnicas permiten a TinyBERT manejar grandes volúmenes de datos textuales de manera eficiente y con alta precisión en tareas específicas, como la detección de correos electrónicos maliciosos. La combinación de tokenización avanzada, arquitectura optimizada y técnicas de entrenamiento robustas asegura que TinyBERT sea una herramienta poderosa en el ámbito del procesamiento de lenguaje natural.

**2.1.4.- Phishing:**

la probabilidad de que una persona responda a un ataque de phishing— ha sido estudiada extensamente durante los últimos 15 años. El phishing es un ataque de oportunidad: los criminales utilizan técnicas de ingeniería social para persuadir a los empleados a que revelen información sensible inadvertidamente. Un artículo denominado, "Why phishing works", encontró que la mayoría de los participantes del estudio carecían del conocimiento técnico necesario para identificar sitios web fraudulentos utilizados en el phishing, así como correos electrónicos fraudulentos. La dificultad de detectar con éxito la comunicación engañosa ha sido confirmada repetidamente en investigaciones posteriores. (Ryan T. Wright, 2023)

A pesar de la significativa inversión en comprender y combatir los ataques de phishing, ha habido poco cambio en la susceptibilidad de los usuarios a los mensajes de phishing. En 2007, Gartner Group estimó que el 19% de sus empleados hizo clic en enlaces fraudulentos y el 3% reveló información personal. Ciertos tipos de phishing dirigido lograron engañar hasta al 72% de los destinatarios para que revelaran información. En 2011, Cisco reportó que entre el 10% y el 15% de los destinatarios respondieron a mensajes de phishing con información sensible (Cisco Systems, 2011).

En 2016, el Verizon Data Breach Report analizó más de 100,000 incidentes y reportó que aproximadamente el 13% de los destinatarios abrieron correos electrónicos de phishing e hicieron clic en el enlace. Ese mismo año, los investigadores estimaron una tasa de susceptibilidad del 20%. (Ryan T. Wright, 2023).

Seguidamente en sus inicios, los ataques de phishing eran relativamente simples, utilizando trucos como enlaces incorrectos o URL con errores tipográficos. Sin embargo, con el avance de la tecnología, los phishers han mejorado sus técnicas, utilizando certificados falsos para crear sitios web falsos que imitan a los legítimos, lo que dificulta su detección.

**Tipos de Phishing:**

1. Whaling: El whaling es un tipo de phishing más focalizado que tiene como objetivo hacerse pasar por una autoridad importante, como un ejecutivo de una empresa. Los atacantes envían mensajes que aparentan provenir de estas figuras de autoridad y solicitan acciones como transferencias de fondos.

2. Spear Phishing: El spear phishing consiste en ataques dirigidos a personas específicas, organizaciones o puestos dentro de una empresa. Los estafadores personalizan los correos electrónicos con información sobre el objetivo, como su nombre, cargo y empresa, para aumentar la probabilidad de éxito.

3. Pharming: El pharming implica redirigir el tráfico de navegación hacia un sitio web falso, haciéndolo parecer legítimo. Esto se hace para obtener credenciales o información confidencial de los usuarios.

4. Smishing: El smishing combina técnicas de ingeniería social con mensajes de texto SMS en lugar de correos electrónicos. Los estafadores intentan hacerse pasar por entidades de confianza, como bancos, para obtener información confidencial de los usuarios.

5. Whishing: Una variante del smishing, el whishing, utiliza la aplicación de mensajería WhatsApp para enviar mensajes rápidos que ofrecen promociones falsas con el objetivo de engañar a los usuarios.

Los ataques de phishing continúan evolucionando y adaptándose a las nuevas tecnologías y formas de comunicación. Es crucial que los usuarios estén al tanto de estas técnicas y tomen medidas para proteger su información personal y financiera contra estos ataques.

**2.1.4.- Redes Neuronales Profundas.**

Una red neuronal es un modelo de machine learning que emula los procesos cognitivos del cerebro humano, imitando el funcionamiento de las neuronas biológicas para identificar patrones, evaluar opciones y alcanzar conclusiones. Este tipo de modelo se estructura en capas de nodos o neuronas artificiales, que se organizan en una capa de entrada, una o varias capas ocultas y una capa de salida. Cada nodo está conectado a otros nodos en capas adyacentes y posee parámetros específicos, como ponderaciones y umbrales. La activación de un nodo y la transmisión de datos a la siguiente capa dependen de si la salida del nodo supera el umbral establecido. De lo contrario, el nodo no transmite información a la capa siguiente. (IBM,2023).

Las redes neuronales se entrenan utilizando conjuntos de datos que les permiten ajustar sus parámetros y mejorar su precisión con el tiempo. Una vez perfeccionadas, estas redes se convierten en herramientas poderosas en informática e inteligencia artificial, capaces de clasificar y agrupar datos con alta velocidad. Tareas complejas como el reconocimiento de voz o imágenes, que podrían tomar horas para ser realizadas manualmente por expertos humanos, pueden completarse en minutos mediante el uso de redes neuronales. Un ejemplo representativo de esta tecnología es el algoritmo de búsqueda de Google, que se basa en redes neuronales para proporcionar resultados relevantes y rápidos (IBM,2023).

En la literatura, las redes neuronales también se conocen como redes neuronales artificiales (ANN) o redes neuronales simuladas (SNN). Representan un subconjunto del machine learning y son fundamentales para los modelos de deep learning, que buscan replicar y ampliar la capacidad de aprendizaje y toma de decisiones del cerebro humano.

En función de lo anterior, debo mencionar un artículo científico, referente a Aplicación de Deep Learning al aprendizaje de modelos en robótica cognitiva, en el cual menciona lo siguiente Cada acción que se quiere predecir va a depender completamente del contexto actual en el que se encuentre el robot. Porque el ambiente es muy poco probable que sea idéntico a los anteriores, es por esto por lo que las predicciones son probabilidades ([0, 1] probabilidad de ejecutar o no una acción). La idea es que por cada acción (llámese acción a ejecuciones desde un movimiento que pueda hacer el robot como mover una mano, hasta moverse a un lado y agarrar un objeto) haya un mecanismo de aprendizaje automático que pueda decidir qué tan probable es que sea buena para lograr un objetivo. Podemos decir que un objetivo puede componerse de una o más acciones, es por esto por lo que las acciones futuras dependen de las anteriores, y se debe de aprender de estas, es decir: olvidar lo menos posible. (Ariel Rodríguez-Jiménez, 2020).

En función de lo ya descrito, podemos mencionar que al igual que la utilización de redes neuronales profundas en robótica, que se enfocan en probabilidades al ejecutar una acción y si esta ejecución es correcta o no, en el presente artículo, ocuparemos la misma hipótesis o postulado en relación a las redes neuronales profundas, enfocándonos al igual que el citado artículo, en probabilidades de reconocimiento e identificación de correos electrónicos, ya que las redes neuronales profundas en su entrenamiento, reconocen patrones y en función de esos patrones de aprendizaje, se puede desprender información referente a la precisión del aprendizaje y luego medir la precisión probabilística de sus resultados en pruebas de reconocimiento y/o acciones a ejecutar. (Ariel Rodríguez-Jiménez, 2020).

**2.1.5.- Clasificador Naive Bayes.**

En machine learning, los clasificadores Naive Bayes (o bayes ingenuo) son una familia de clasificadores probabilísticos basados en aplicar el teorema de bayes con fuertes suposiciones de independencia entre sus características. (Chaparro, 2019) Ha sido intensamente estudiado desde los 1950s y se mantiene como un popular método base para la categorización de texto, especialmente si es combinado con métodos más avanzados como SVM. El teorema de bayes, en teoría probabilística y estadísticas, describe la probabilidad de un evento, basado en conocimiento previo de condiciones que pueden estar relacionadas con dicho evento (Zhang, s. f.). Matemáticamente, el teorema de bayes se define de la siguiente manera

(1)

Dicha fórmula se lee de la siguiente manera: “La probabilidad de que ocurra A dado B es igual a la probabilidad de que ocurra B dado A, por la probabilidad de que ocurra A, dividido en la probabilidad de que ocurra B”, esta descripción se refleja en la ecuación anterior enumerada como (1). (Chaparro, 2019)

Naïve Bayes es parte de una familia de algoritmos de aprendizaje generativo, lo que significa que busca modelar la distribución de entradas de una clase o categoría determinada. A diferencia de los clasificadores discriminativos, como la regresión logística, no aprende qué características son más importantes para diferenciar entre clases. (IBM,2023).

Referente a lo anterior existe el Multinomial Naive Bayes el cual es una versión más de la familia de clasificadores bayes, aplicada a la distribución multinomial. Una distribución multinomial hace referencia a que si, por ejemplo, en una distribución binomial se refiere a experiencias con resultados dicotómicos (claro u oscuro, vivo o muerto, positivo o negativo, etc…), en una distribución multinomial se habla de experiencias con r posibles resultados, cada uno con una probabilidad asignada. (Chaparro, 2019)

**2.1.6.- ROC AUC SCORE**

El ROC AUC Score es una métrica fundamental en la evaluación de clasificadores binarios, ya que mide el área bajo la curva ROC (Receiver Operating Characteristic). Esta curva traza la tasa de verdaderos positivos (True Positive Rate, TPR) frente a la tasa de falsos positivos (False Positive Rate, FPR), permitiendo visualizar el desempeño del modelo en diferentes umbrales de clasificación. El valor del ROC AUC Score oscila entre 0 y 1, donde un valor de 1 refleja una clasificación perfecta, 0.5 indica un desempeño equivalente a una clasificación aleatoria, y 0 representa un modelo que clasifica incorrectamente de manera sistemática. Esta métrica es especialmente valiosa en la comparación de modelos, proporcionando una medida agregada de desempeño a lo largo de todos los umbrales posibles.

Junto al ROC AUC Score, otras métricas clave incluyen la precisión (Precision) y la exhaustividad (Recall). La precisión evalúa la exactitud del modelo en términos de la proporción de verdaderos positivos entre todas las predicciones positivas, siendo crucial en contextos donde los falsos positivos tienen un costo elevado. El recall, por su parte, mide la capacidad del modelo para identificar todas las instancias positivas, lo que es esencial en escenarios donde es crítico minimizar los falsos negativos. Estas métricas se complementan entre sí y ofrecen una perspectiva detallada del comportamiento del clasificador.

El F1-Score es una métrica derivada que combina la precisión y el recall en una media armónica, proporcionando un único valor que refleja el equilibrio entre ambas métricas. Es particularmente útil cuando se busca un compromiso entre la precisión y el recall, especialmente en situaciones donde el balance entre falsos positivos y falsos negativos es crucial para el desempeño global del modelo. El F1-Score, al ponderar igualmente la precisión y el recall, se convierte en una herramienta indispensable para la evaluación de clasificadores en entornos científicos.

**2.2.- Tablero Kanban.**

A continuación, se confeccionará un tablero Kanban, donde se definirá cada fase en la que se ha avanzado y se colocara si la fase está en prueba, ha sido finalizada entre otros detalles, definiendo los tiempos de trabajo para el presente proyecto.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Descripción | Tiempo para lograr funcionalidad (días hábiles) | Responsable(s) | Pendiente | Haciendo | Terminado | En Pruebas | Despliegue |
| F1 | Recopilar datos forenses detallados de incidentes de phishing anteriores, incluyendo registros y capturas. (recopilación de dataset) |  | investigador |  |  | X |  |  |
| F2 | Analizar correos y archivos adjuntos sospechosos utilizando herramientas de análisis, identificando frecuencia y patrones. |  | investigador |  |  | X |  |  |
| F3 | Identificar comportamientos anómalos y patrones de escrituras en los correos de ataques de phishing. |  | investigador |  |  | X |  |  |
| F4 | Recopilar indicadores de compromiso específicos para ataques de phishing, como IPs y URLs maliciosas. |  | investigador |  |  | X |  |  |
| F5 | Desarrollar algoritmos de machine learning óptimos para la identificación de ataques de phishing |  | investigador |  |  | X |  |  |
| F6 | Identificación de patrones de escritura e intención en correos electrónicos mediante procesamiento del lenguaje natural. |  | investigador |  |  | X |  |  |
| F7 | Migrar el dataset a una base de datos de MySQL Workbench |  | investigador |  |  | X |  |  |
| F8 | Confeccionar un algoritmo de preprocesado y tokenizado que desglose el texto, en palabras, oraciones, caracteres, subcaracteres, números. |  | investigador |  |  | X |  |  |
| F9 | Guardar los datos tokenizados del respectivo dataset en una base de datos de MySQL Workbench, para mantener la fidelidad de estos. |  | investigador |  |  | X |  |  |
| F10 | Confeccionar un algoritmo Clasificador Naive Bayes y entrenarlo con los datos tokenizados |  | investigador |  |  | X |  |  |
| F11 | Confeccionar un algoritmo de red neuronal profunda y entrenarlo con los tokenizados. |  | investigador |  |  | X |  |  |
| F12 | Confeccionar y usar un modelo pre entrenado de la red neuronal BERT, confeccionada por Hugging Face, denominado como Tiny BERT |  | investigador |  |  | X |  |  |
| F13 | Verificar el comportamiento durante el entrenamiento de cada algoritmo de machine learning |  | investigador |  |  |  | X |  |
| F14 | Verificar el comportamiento posterior al entrenamiento, analizando precisión y curva de ROC. |  | investigador |  |  |  | X |  |
| F15 | Verificar el porcentaje de precisión al predecir y clasificar correos electrónicos a fin de poner aprueba el entrenamiento de cada algoritmo |  | investigador |  |  |  | X |  |

(Tabla Nro. 01, correspondiente a tablero Kanban de etapas del experimento)

(Fuente de elaboración propia)

**2.2.1.- Matriz de riesgo**.

Seguidamente se confeccionará una tabla, denominada matriz de riesgo, en la cual mencionaremos todos los posibles riesgos que podemos enfrentar en el proceso del estudio de los diferentes comportamientos de los modelos de Machine Learning, propuestos en un principio.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Riesgo | Descripción | Probabilidad | Impacto | Nivel de riesgo | mitigación |
| Sobreajuste del Modelo (Overfitting) | TinyBERT o las redes neuronales pueden sobreajustarse a los datos de entrenamiento debido a la complejidad del modelo y a la cantidad de datos con los que se entrenaran | Media | Alta | Media-Alta | Implementar técnicas de regularización, cross-validation o validación cruzada, k folds entre otras y ajustar el tamaño del modelo. |
| Subajuste del Modelo (Underfitting) | Naive Bayes podría no capturar bien las complejidades del lenguaje natural, resultando en un modelo con baja precisión. | Media | Media | Media | Asegurar de que el preprocesamiento, tokenizado y las características sean adecuadas para el modelo. |
| Desbalance de Clases | La clasificación en las etiquetas de phishing y no phishing podría estar desbalanceada, afectando el rendimiento del modelo. | Alta | Alta | Alta | Aplicar técnicas de balanceo de clases como SMOTE o ADASYN. |
| Eficiencia Computacional | El procesamiento y entrenamiento del modelo puede ser lento o consumir muchos recursos, especialmente con modelos complejos como TinyBERT. | Alta | Alta | Alta | Optimizar el código, usar hardware adecuado (GPU), y ajustar los hiperparámetros para reducir el tiempo de procesamiento. |
| Seguridad de la Base de Datos | La base de datos MySQL podría ser vulnerable a ataques o pérdida de datos sensibles. | Media | Alta | Media-Alta | Implementar medidas de seguridad robustas, como cifrado de datos, y controles de acceso adecuados. |
| Exactitud en la Tokenización | La tokenización inexacta puede afectar la calidad del entrenamiento y las predicciones del modelo. | Alta | Media | Media | Revisar y ajustar la tokenización para asegurar que los datos se procesen correctamente. |
| Implementación de Padding | La implementación incorrecta de padding podría causar que el modelo ignore partes relevantes del texto o agregue ruido. | Baja | Media | Baja-Media | Verificar que el padding se aplique adecuadamente para mantener la longitud uniforme sin pérdida de información relevante. |
| Dependencia de Modelos Preentrenados | Dependencia de TinyBERT preentrenado puede limitar la adaptabilidad del modelo a datos específicos del dominio. | Media | Media | Media | Considerar el fine-tuning del modelo con datos específicos del dominio y evaluar otras arquitecturas si es necesario. |
| Mantenimiento y Actualización del Modelo | La necesidad de actualizar y mantener los modelos a lo largo del tiempo para seguir siendo efectivos. | Media | Media | Media | Planificar ciclos de revisión y actualización periódica del modelo basados en nuevos datos y cambios en el entorno. |

(Tabla Nro. 02, matriz de riesgo, con la descripción de los riesgos del experimento)

(Fuente elaboración propia)

**3.- Arquitectura de construcción de los algoritmos de Machine Learning**

En este experimento a fin de comparar modelos de machine learning se ha implementado el algoritmo de Tiny BERT, de redes neuronales profundas y un clasificador Naive Bayes en Python 3.10 y desarrollado en el entorno de Visual Studio Code, estos modelos han sido diseñados para trabajar con datos migrados desde un documento Excel a un motor de base de datos MySQL 8.0 Workbench. En virtud del motor de base de datos mencionado, se creó una base de datos, denominada "Phishing", la cual contiene datos estructurados en tablas con columnas esenciales y un ID de autoincremento, lo que asegura la integridad y fidelidad de la información durante el desarrollo y análisis del experimento.

Cabe hacer presente, que la base de datos antes mencionada fue creada con un Excel que contiene aproximadamente 8349 líneas de textos, separadas en columnas, que contienen datos relevantes de correos electrónicos de Phishing, como encabezado de correos electrónicos y el cuerpo de este, además de la respectiva clasificación, la cual está conformada por dos etiquetas claras que son phishing y no phishing, la que clasifica y especifica cada línea de texto del dataset de estudio, en función de lo anterior, dicha clasificación la utilizare para que cada uno de los modelos de machine learning aprendan desde la clasificación basal del dataset, a fin de , dicho dataset contenedor de los datos de estudio, fue obtenido y descargado desde la página web kaggle.com, la cual es una plataforma de competencia de ciencia de datos y una comunidad en línea para científicos de datos y profesionales del aprendizaje automático de Google LLC. (kaggle, s.f.)

**3.1.- Metodologías Importantes en el Algoritmo TinyBERT Enfocado en la Detección de Patrones de Procesamiento de Lenguaje Natural**

Para iniciar la explicación del referido algoritmo, se debe hacer mención que se exporto a la función de inicio (def\_init) del algoritmo Tiny bert el código TinyBert General 4L\_312D, el que corresponde a un modelo pre entrenado específico de TinyBERT desarrollado por Huawei Noah's Ark Lab. Este modelo, al igual que otros modelos preentrenados, se utiliza comúnmente para tareas de procesamiento de lenguaje natural (NLP) debido a su capacidad para capturar y representar el contexto del lenguaje en texto de manera eficiente, en función de esto, este modelo ya tiene una estructura de fabrica con métricas ya tomadas, el cual hace que sea una elección razonable para lo que se busca ejecutar y las tareas a resolver.

Cabe destacar que la página web de Hugging Face, ofrece una gran variedad de modelos pre entrenados, en diferentes tareas de procesamiento de lenguaje natural, para lo cual, al utilizar estos modelos, es necesario crear un algoritmo denominado Fine Tunning, el cual re entrena el modelo pre entrenado con nuestros datos que queremos usar para la tarea especifica en cuestión, en este caso la tarea se definió como un modelo de Bert o Tiny Bert que sea capaz de definir o clasificar una línea de texto, si corresponde o no a un correo electrónico malicioso.

En este caso y por la arquitectura original de BERT, la propuesta inicial de esta parte del proyecto es verificar como el modelo o su forma más reducida, es capaz de leer bidireccionalmente una línea de texto que contiene datos como correo electrónico, encabezado del correo y cuerpo de este y en función de su lectura y reconocimiento de patrones y caracteres definir a que corresponde el referido correo.

Como se mencionó en un principio, para lograr o tener una aproximación de que modelos nos podría servir para clasificar datos correspondientes a correos electrónicos maliciosos, se usara no solo un modelo pre entrenado, sino dos modelos distintos y se observara su comportamiento en relación con el dataset de kaggle que contiene información de correos maliciosos.

Por lo cual los modelos a utilizar son los siguientes **TinyBert** y el modelo **ver Bert finetuned phishing.** Cada modelo esta entrenado para realizar tareas, específicas de procesamiento del lenguaje natural.

1.- **TinyBert:** TinyBert es 7.5 veces más pequeño y 9.4 veces más rápido en inferencia que BERT-base, y logra un rendimiento competitivo en tareas de comprensión del lenguaje natural. Realiza una destilación novedosa de transformadores tanto en la etapa de pre-entrenamiento como en la de aprendizaje específico para tareas. En la destilación general, utilizamos el BERT-base original sin ajuste fino como maestro y un corpus de texto a gran escala como datos de aprendizaje. Al llevar a cabo la destilación de Transformer en el texto de dominio general, obtenemos un TinyBERT general que proporciona una buena inicialización para la destilación específica de tareas. Aquí proporcionamos el TinyBERT general para tus tareas (Huggingface, 2024)

**TinyBert** Este modelo es una versión compacta de TinyBert con 4 capas y una dimensión oculta de 312, diseñado para ofrecer un equilibrio entre tamaño y rendimiento. Mantiene una alta eficiencia en tareas de comprensión del lenguaje natural mientras reduce los requerimientos computacionales en comparación con modelos más grandes. Ideal para aplicaciones en dispositivos con recursos limitados. (Huggingface, 2024)

2.- **Bert finetuned phishing:** Este modelo es una versión afinada de bert large uncased en un conjunto de datos de phishing, capaz de detectar phishing en sus cuatro formas más comunes: URLs, correos electrónicos, mensajes SMS e incluso sitios web.(Huggingface, 2024).

Este modelo puede lograr los siguientes resultados en el conjunto de evaluación:

* **Pérdida (Loss): 0.1953**
* **Precisión (Accuracy): 0.9717**
* **Precisión (Precision): 0.9658**
* **Recuperación (Recall): 0.9670**
* **Tasa de Falsos Positivos (False Positive Rate): 0.0249**

**Descripción del Modelo**

BERT es un modelo de transformers preentrenados en un gran corpus de datos en inglés de manera auto supervisada o sea entrenado con un enorme dataset con información referente a la tarea a cumplir. Esto significa que fue preentrenado en textos crudos únicamente, sin que tuviera alguna alteración o fueran etiquetados de ninguna manera o forma (lo que le permite utilizar muchos datos disponibles públicamente) mediante un proceso automático para generar entradas y etiquetas a partir de esos textos. (Huggingface, 2024).

**Este modelo tiene la siguiente configuración:**

* 24 capas
* Dimensión oculta de 1024
* 16 cabezas de atención
* 336 millones de parámetros

**Motivación y Propósito**

El phishing es uno de los ataques cibernéticos más frecuentes y costosos según varios informes de seguridad. Este modelo tiene como objetivo prevenir de manera eficiente y precisa los ataques de phishing contra individuos y organizaciones. Para lograrlo, BERT fue entrenado en un conjunto de datos diverso y robusto que contiene: URLs, mensajes SMS, correos electrónicos y sitios web, lo que permite al modelo extender su capacidad de detección más allá de lo habitual y ser utilizado en diversos contextos o en diversos vectores de ataque. (Huggingface, 2024).

**Hiperparámetros de Entrenamiento**

**Los siguientes hiperparámetros se utilizaron durante el entrenamiento:**

* Tasa de aprendizaje (learning\_rate): 2e-05
* Tamaño del batch de entrenamiento (train\_batch\_size): 16
* Tamaño del batch de evaluación (eval\_batch\_size): 16
* Semilla (seed): 42
* Optimizador (optimizer): Adam con betas=(0.9, 0.999) y epsilon=1e-08
* Tipo de planificador de tasa de aprendizaje (lr\_scheduler\_type): lineal
* Número de épocas (num\_epochs): 4

(Huggingface, 2024).

**3.1.2. Tokenización y Codificación de Texto**

TinyBERT es un modelo optimizado para el procesamiento de lenguaje natural (NLP) que se basa en una arquitectura de red neuronal refinada para abordar las limitaciones de los modelos de gran tamaño como BERT.

Cabe hacer presente que, en el ámbito del procesamiento de lenguaje natural, la tokenización es un proceso crucial que descompone el texto en unidades más manejables conocidas como tokens, que pueden ser palabras, sub palabras o caracteres. TinyBERT, al igual que otros modelos basados en BERT (Bidirectional Encoder Representations from Transformers), utiliza una técnica avanzada de tokenización subword para convertir el texto en identificadores numéricos que el modelo puede procesar. En particular, TinyBERT emplea el tokenizador **WordPiece**, que segmenta las palabras en subpalabras, lo que facilita una representación más eficiente y precisa del lenguaje.

La codificación del texto es un paso que sigue a la tokenización y convierte estos tokens en vectores de **embeddings**. TinyBERT utiliza tanto embeddings de posición como embeddings de token para representar las entradas textuales en un espacio de alta dimensión, permitiendo que el modelo capture tanto el significado individual de cada token como las interacciones contextuales entre ellos. Este enfoque es crucial para que el modelo pueda comprender la relación entre diferentes partes del texto y realizar predicciones más precisas.

Para mejorar la exactitud del tokenizado y la calidad de las predicciones, se decidió seleccionar las columnas de entrada que contienen la información a analizar, así como la columna de clasificación. Este enfoque integral asegura que el modelo de TinyBert se entrene con una representación completa y detallada del texto, aprovechando la capacidad de WordPiece para manejar variaciones morfológicas y contextuales de manera más eficiente que otras técnicas de tokenización.

**Concatenación:**

En este proceso de tokenizacion, específicamente en la función tokenize\_data del algoritmo TINY BERT, ejecuta diversas tareas que ayudan a la precisión del entrenamiento del referido algoritmo, una de ellas es la concatenación de las columnas de entrada con la columna objetivo ( columna de clasificación) esta metodología de concatenar las columnas de entradas más la columna objetivo, hace que la línea de texto completa compuesta por las tres columnas seleccionadas se mantenga lo más integra posible, agregando su respectiva clasificación, lo que hace que al momento del tokenizado de la línea de texto completa, la red neuronal Tiny Bert, se entrene con mucha más precisión.

Ejemplo de la concatenación, expresado en la consola de Visual Studio Code.

Texto

Descripción generada automáticamente

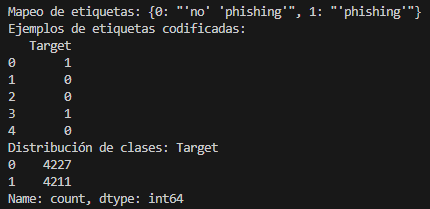
(Imagen Nro.01, imagen ilustrativa de los datos concatenados)

(Fuente elaboración propia)

En esta fase de concatenación de las columnas se observa cómo se concatenaron las columnas seleccionadas como columnas de entrada con su respectiva etiqueta de clasificación contenida en la columna objetivo.

**Mapeo de etiquetas:**

Este es otro algoritmo de la función tokenize\_data, en el cual se ejecuta un algoritmo de mapeo de etiquetas, el cual funciona etiquetando a las etiquetas de la columna objetivo con relación a clasificación binaria, (0,1), obteniendo una mejor identificación al momento de ser tokenizado y entrenado luego el modelo de red neuronal Tiny Bert.



(Imagen Nro.02, imagen ilustrativa mapeo de etiquetas)

(Fuente elaboracion propia)

Se observa una distribución de etiquetas casi similares para ambas clasificaciones contenidas en la columna objetivo.

Luego de ejecutados todas esas etapas importantes de concatenación y mapeo de etiquetas y se observa que los datos están en su totalidad y dispuestos a ser tokenizado, la función sigue su secuencia y comienza el tokenizado enfocado a modelos BERT, el cual tiene el siguiente formato:

Verificación de longitud de tokens

**Texto

Descripción generada automáticamente**

(ImagenNro.03, ejemplo grafico de tokenizacion en modelos Bert)

(Fuente elaboración propia)

1. **Longitud de input\_ids**: La longitud total es 512 tokens. Esto indica que el texto ha sido tokenizado y ajustado para coincidir con la longitud máxima aceptada por el modelo. En el caso de modelos como BERT, 512 es la longitud máxima estándar para los inputs\_ids, que representa la secuencia completa de tokens del texto.
2. **input\_ids (primeros 10 tokens)**: Los primeros 10 valores en input\_ids son [101, 1005, 8915, 27572, 7951, 2319, 5403, 2480, 1005, 1005]. Estos números son identificadores de tokens que corresponden a palabras o sub palabras específicas en el vocabulario del modelo. El token 101 suele representar el inicio de la secuencia (token [CLS] en BERT), mientras que los demás son tokens del texto original, y el valor 1005 se repite, indicando tokens que pueden ser palabras comunes en el vocabulario del modelo.
3. **attention\_mask (primeros 10 tokens)**: La máscara de atención muestra [1, 1, 1, 1, 1, 1, 1, 1, 1, 1] para los primeros 10 tokens. Los valores de 1 en la attention\_mask indican que estos tokens son relevantes para el modelo y deben ser considerados durante el proceso de atención. En este caso, todos los primeros 10 tokens son relevantes, sin tokens de padding en esta sección.

**3.1.3. Arquitectura de los fine tunning usado en los modelos elegidos de Bert y Tiny Bert**

En los diferentes fine-tunings, usados por modelos pre entrenados de BERT y Tiny BERT se usó una arquitectura similar para todos, donde solamente los cambios de cada uno se ven reflejados en los hiperparametros basales que estos mantienen, pero la estructura corresponde a la siguiente:

**Arquitectura del Modelo**

1. **Modelo Base**: Se utiliza los modelos **TinyBert General 4L\_312D** y el modelo **Bert finetuned phishing** que son versiones o más pequeñas que BERT o más grandes, en relación con las capas que contienen. La arquitectura básica del fine tunning que se ocupo es similar a en los tres modelos antes mencionados, correspondiente a la siguiente:

* **Número de Capas**: 24
* **Número de Cabezas de Atención**: 16
* **Dimensión Oculta**: 1024
* **Tamaño del Intermedio**: 4096
* **Tipo de Embeddings**: Absolute Position Embeddings
* **Tamaño del Vocabulario**: 30522

1. **Configuración Personalizada**: Se ajusta la configuración de TinyBert y el modelo bert finetuned phishing o para tareas de clasificación de texto. Esta configuración incluye:

**Dropout**: Se aplica un dropout del 10% en las atenciones y en las capas ocultas para evitar el sobreajuste.

**ID de Etiquetas**: Se definen dos etiquetas para la tarea de clasificación: "benign" (0) y "phishing" (1).

1. **Tokenizador**: El tokenizador BertTokenizer se usa para convertir el texto en secuencias de IDs que el modelo BERT puede procesar. Se configura para truncar o rellenar el texto a una longitud máxima de 512 tokens.

**Flujo de Trabajo del Fine-Tuning**

1. **Preprocesamiento de Datos**:

**Limpieza del Texto**: Se realiza una limpieza básica que incluye la eliminación de puntuación y la conversión a minúsculas.

**Preparación del Dataset**: Se carga un archivo Excel con datos de correos electrónicos y se concatenan las columnas relevantes en una sola columna de texto. Se asignan etiquetas binarias (phishing = 1, no phishing = 0).

1. **División de Datos**:

Los datos se dividen en conjuntos de entrenamiento y evaluación utilizando train\_test\_split para asegurar que el modelo se entrene en una parte de los datos y se evalúe en otra.

1. **Tokenización y Conversión a Tensores**:

Se tokenizan los textos usando el tokenizador de BERT y se convierten en tensores para ser usados en PyTorch. Esto incluye las secuencias de IDs, las máscaras de atención y las etiquetas.

1. **Configuración del Entrenamiento**:

**Argumentos de Entrenamiento**: Se configuran parámetros como el tamaño del batch, la tasa de aprendizaje, el número de épocas y el directorio de salida para el modelo entrenado.

**Optimización**: Se utiliza el optimizador AdamW con una tasa de aprendizaje de 2e-5 y un épsilon de 1e-6 para ajustar los pesos del modelo durante el entrenamiento.

1. **Entrenamiento y Evaluación**:

**Entrenamiento**: El modelo se entrena usando el Trainer de Hugging Face con el dataset de entrenamiento y los argumentos configurados.

**Evaluación**: Después del entrenamiento, el modelo se evalúa en el conjunto de evaluación para medir su rendimiento, obteniendo métricas como la pérdida, precisión y recall.

1. **Predicciones**:

Se obtienen las predicciones del modelo en el conjunto de evaluación, y se calcula la probabilidad de la clase positiva (phishing) para cada ejemplo.

Este flujo de trabajo o arquitectura que se utilizó en los dos modelos es un flujo simple, pero asegura que el modelo TinyBERT General 4L\_312D y el modelo Bert finetuned se adapten de manera efectiva a la tarea específica de detección de phishing, aprovechando su capacidad para manejar secuencias de texto complejas y proporcionar predicciones precisas.

**3.1.4. Entrenamiento y Optimización**

El proceso de entrenamiento de TinyBert incluye una fase de fine-tuning, que ajusta el modelo preentrenado a un conjunto de datos específico para mejorar su rendimiento en tareas concretas, como la clasificación de correos electrónicos. Durante esta fase, se emplea el optimizador AdamW, una variante del optimizador Adam que incorpora un término de decaimiento de peso, mejorando así la convergencia y el rendimiento del modelo. La función de pérdida utilizada es la entropía cruzada, que mide la discrepancia entre las predicciones del modelo y las etiquetas verdaderas, guiando el ajuste de los parámetros del modelo para minimizar esta discrepancia.

El fine-tuning permite que TinyBert se adapte a las características específicas del dataset de entrenamiento, optimizando así su capacidad para detectar patrones y realizar predicciones precisas en el contexto de la clasificación de correos electrónicos maliciosos.

**3.1.5. Evaluación y Métricas**

La evaluación del rendimiento de TinyBERT se realiza utilizando métricas estándar de clasificación, que incluyen precisión, recall y F1-score. Estas métricas proporcionan una visión integral de la capacidad del modelo para clasificar correctamente los datos en las categorías deseadas. Además, se utiliza la matriz de confusión para analizar las verdaderas y falsas predicciones del modelo, ofreciendo una perspectiva detallada de su desempeño.

La curva ROC (Receiver Operating Characteristic) y el área bajo la curva ROC (AUC) son herramientas clave para evaluar la capacidad del modelo para discriminar entre clases positivas y negativas en diferentes umbrales de decisión. La ecuación para la curva ROC se define como:

(2)

(3)

(4)

Las tres ecuaciones mencionadas están relacionadas con la evaluación del rendimiento de un clasificador mediante la curva ROC (Receiver Operating Characteristic). La ecuación (2) (ROC) describe cómo se construye esta curva comparando la tasa de verdaderos positivos (TPR) y la tasa **de** falsos positivos (FPR) en diferentes umbrales de clasificación. La ecuación (3) (TPR) mide la sensibilidad del modelo, es decir, la proporción de muestras positivas correctamente clasificadas. La ecuación (4) (FPR), por su parte, mide el nivel de falsos positivos, es decir, la proporción de muestras negativas que el modelo clasifica erróneamente como positivas. Juntas, estas métricas permiten evaluar el equilibrio entre sensibilidad y precisión en la detección de instancias positivas y negativas.

Estas métricas y herramientas permiten a TinyBERT manejar grandes volúmenes de datos textuales con alta eficiencia y precisión. La combinación de técnicas avanzadas de tokenización, una arquitectura optimizada, métodos efectivos de entrenamiento y evaluación robusta asegura que TinyBERT sea una herramienta efectiva y eficiente en el campo del procesamiento de lenguaje natural, particularmente en la detección de correos electrónicos maliciosos y otras aplicaciones similares.

En relación con la métrica ROC (Receiver Operating Characteristic), esta al momento de entrenar el modelo, debe de estar por sobre los 0.5, para evitar aleatoriedad en el modelo, si este modelo al momento en que se calcula la curva de ROC es de 0.5 o inferior, se encontrara en una situación que cada una de las predicciones que este haga, serán netamente aleatorias, perdiendo precisión en sus identificaciones y/ predicciones.

**3.2.- Metodologías Importantes en el Algoritmo del Clasificación Naive Bayes en Tkinter Enfocado en la Detección de Patrones de Procesamiento de Lenguaje Natural**

El proceso de entrenamiento del modelo Naive Bayes se realiza utilizando el clasificador MultinomialNB, que es adecuado para datos de conteo o frecuencia como los que se encuentran en tareas de clasificación de texto. En este método, tras seleccionar las columnas de entrada y la columna objetivo, se extraen los datos relevantes de la base de datos y se transforman en una forma adecuada para el modelo. La transformación se realiza a través de la vectorización del texto usando TF-IDF (Term Frequency-Inverse Document Frequency), un enfoque común en el procesamiento de lenguaje natural que convierte el texto en una matriz numérica donde cada elemento representa la importancia relativa de una palabra en un documento en comparación con su frecuencia en el corpus.

El modelo MultinomialNB se entrena con los datos vectorizados y la columna objetivo codificada. Posteriormente, se evalúa el modelo calculando y mostrando la matriz de confusión, el informe de clasificación y la precisión. La matriz de confusión proporciona una visión integral de la calidad de las predicciones al mostrar la cantidad de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. El informe de clasificación incluye métricas de precisión, recall y F1-score para cada clase, mientras que la precisión global mide el porcentaje total de clasificaciones correctas, las cuales las clasificaciones pueden ser como phishing o no phishing, esto según la respectiva etiqueta de clasificación de la respectiva columna objetivo.

El objetivo principal es entrenar el clasificador Naive Bayes con los datos proporcionados y evaluar su rendimiento mediante métricas de desempeño específicas. Este análisis es crucial para entender la capacidad del modelo para generalizar y hacer predicciones precisas en datos tipo object o líneas de texto, a fin de identificar que tan preciso es el clasificador en tareas de procesamiento de lenguaje natural.

En relación con lo mencionado anteriormente el clasificador, funciona mediante diversas ecuaciones simples de identificación de verdaderos y falsos positivos y negativos, que reconoce el algoritmo, mediante se entrena, siendo la siguiente ecuación.

Tabla de ecuaciones que se generan al momento de entrenar el clasificador Naive Bayes, se especificaran las métricas que ocupa el respectivo clasificador, en función de reconocer patrones de lenguaje de correos de ataque de phishing

**Tabla Naive Bayes.**

|  |  |  |
| --- | --- | --- |
| **Nombre de métricas** | **descripción** | **Formula calculo** |
| **Índice de Éxito de Identificación de Patrones de Ataque precisión** | La ecuación (5) calcula el porcentaje de éxito en la identificación de patrones de ataque específicos de phishing en comparación con el total de ataques analizados, esta métrica se enfoca en la precisión del sistema de aprendizaje supervisado para identificar correos maliciosos | Precisión=  (5) |
| **índice de éxito de identificación de patrones de ataques Recall** | La ecuación (6) calcula el porcentaje de éxito en la identificación de patrones de ataque específicos de phishing en comparación con el total de ataques analizados, el recall mide la proporción de los casos positivos reales que fueron correctamente identificados por el modelo | Recall=  (6) |
| **Índice de precisión de identificación de verdaderos y falsos positivos y negativos en correos electrónicos** | Para lograr esto, podemos utilizar el enfoque probabilístico de Naive Bayes.  La ecuación (7) de Naive Bayes se adapta bien a este escenario ya que nos permite calcular la probabilidad de que los datos recopilados y verificar que pertenezcan a una en clase particular dadas las características observadas | )=  (7) |
| **Índice de reconocimiento del patrón más repetitivo y menos repetitivo** | En este punto definiremos dentro de cada dataset, los datos que más se repiten, ecuación (8) y que menos se repiten, ecuación (9) en los diversos ataques de Phishing esto en virtud de tener un enfoque de quienes o que modus operandi es el más común y así identificar patrones y posibles atacantes. | **Frecuencia relativa al patrón que más se repite:**  **Frecuencia Máxima=**  (8)  **Frecuencia Mínima=**    (9) |

(Tabla Nro. 03, contenedora de las ecuaciones que se ejecutan en clasificador Naive Bayes)

(Fuente elaboración propia)

En relación de lo anterior, una vez que se han calculado cada una de las métricas señaladas y se obtienen los valores reales del entrenamiento, es posible obtener un detalle claro del referido entrenamiento y de su funcionamiento, lo cual debiese de tener relación con los datos de predicción.

Una vez entrenado el modelo, la funcio predecir\_texto, la que se encuentra dentro del mencionado algoritmo, la que permite utilizar el clasificador para realizar predicciones sobre nuevos textos. Los textos ingresados por el usuario se vectorizan utilizando el mismo TF-IDF que se aplicó durante el entrenamiento. Esto garantiza que los datos de entrada estén en la misma representación que los datos de entrenamiento a fin de asegurar mayor fidelidad y precisión al momento de efectuar las predicciones. El modelo realiza las predicciones sobre estos datos vectorizados, y los resultados se decodifican para retornar las etiquetas originales.

**3.4.- Redes Neuronales Profunda.**

En primera instancia en el referido algoritmo, se creó y ejecuto una función de preprocesado, tokenizado y guardado de datos en una base de datos, de nombre “phishing”, la que se encuentra dentro de la aplicación de MySQL Workbench 8.0, donde la mencionada función ejecuta dos acciones claves, primero, traspasa el dataset completo la base de datos, manteniendo la fidelidad de los datos originales, luego se ejecuta la respectiva tokenizacion, creando otra tabla donde se guarden los datos tokenizados, este algoritmo ejecuta la acción, tokenizando, palabras completas, luego correos electrónicos que pudiese tener el dataset de Kaggle, luego tokeniza números, caracteres y sub caracteres, obteniendo la mayor información, lo más detallada posible, esto a fin de que el procesamiento del lenguaje natural, sea lo más exacto posible, generando una primera etapa de filtrado de información para generar mayor precisión.

Luego en el proceso de tokenización del algoritmo de redes neuronales profundas, se buscan transformar los datos originales y tokenizados que se encuentran en tablas separadas de la base de datos a un formato adecuado para el modelado y el análisis de la red neuronal en comento. El proceso se inicia en la función **cargar\_y\_preprocesar\_datos**, que carga los datos desde una base de datos y los preprocesa para su posterior análisis. Primero, se seleccionan y cargan las columnas relevantes desde una tabla especificada, se verifica la presencia de valores faltantes (NaN) o espacios vacíos que pudiesen afectar la fidelidad de los datos de las columnas seleccionadas como entrante, luego en la columna objetivo, y se elimina cualquier fila que contenga valores faltantes en esta columna. La columna objetivo se asegura de ser de tipo cadena (string). Luego, se concatena el texto de las columnas seleccionadas y la columna objetivo en una nueva columna denominada 'text', que se utiliza para el análisis de texto.

Seguidamente el método **tokenize\_data** se encarga de la tokenización propiamente dicha. Primero, verifica si los datos han sido cargados. Si no se han cargado, la tokenización no puede proceder. Una vez confirmada la carga de los datos, se realiza la tokenización. Este proceso se lleva a cabo utilizando el vectorizador TF-IDF (Term Frequency-Inverse Document Frequency), que convierte el texto en una matriz numérica donde cada fila representa un documento y cada columna representa una palabra en el vocabulario, ponderada por su importancia relativa en el conjunto de datos.

Antes de la tokenización, se realiza un mapeo de las etiquetas de texto en la columna objetivo a valores numéricos utilizando LabelEncoder. Esto es necesario porque los algoritmos de aprendizaje automático requieren etiquetas numéricas para su entrenamiento. Después del mapeo, se aplica TF-IDF a los textos concatenados para obtener las representaciones vectoriales de los documentos.

Ejemplo de mapeo de datos utilizado en redes neuronales profundas.

Etiquetas mapeadas:

["'no' 'phishing'" "'phishing'"]

El conjunto de datos se divide en conjuntos de entrenamiento y prueba, utilizando **train\_test\_split** con un tamaño de prueba del 20% y una división estratificada para mantener la proporción de clases. Posteriormente, se utiliza SMOTE (Synthetic Minority Over-sampling Technique) para balancear las clases en el conjunto de entrenamiento, generando muestras sintéticas para las clases minoritarias.

Finalmente, los datos balanceados se guardan en archivos. Npy para su uso futuro y se procede a llamar al método **entrenar\_modelo** para comenzar el entrenamiento del modelo con los datos tokenizados.

**3.4.1.- Modelo de red neuronal profunda.**

La función **entrenar\_modelo** se encarga de iniciar el proceso de entrenamiento del modelo de red neuronal una vez que se ha completado la tokenización de los datos. En primer lugar, la función verifica si el proceso de tokenización ha sido completado. Si no se ha realizado la tokenización, la función imprime un mensaje de error y termina la ejecución, ya que el entrenamiento del modelo requiere datos tokenizados.

Una vez confirmada la finalización de la tokenización, la función procede a cargar los datos balanceados previamente guardados en archivos. npy. Estos datos incluyen **X\_train\_balanced**, que son las características del conjunto de entrenamiento balanceado, **y y\_train\_balanced**, que son las etiquetas correspondientes. Estos archivos son cargados utilizando np.load, que permite la recuperación de datos almacenados en formato de archivo NumPy.

Con los datos listos, la función inicia la configuración y entrenamiento del modelo. Se define un modelo de red neuronal secuencial (Sequential) utilizando la biblioteca Keras de TensorFlow. Este modelo consta de varias capas:

* **Capa densa de entrada:** Una capa Dense con 128 unidades y una función de activación ReLU. Esta capa recibe las entradas del modelo, cuya forma se basa en el número de características en X\_train\_balanced.
* **Capa de dropout:** Una capa Dropout con una tasa de deserción del 50% (0.5). Esta capa ayuda a prevenir el sobreajuste (overfitting) durante el entrenamiento al aleatoriamente desactivar el 50% de las neuronas en cada paso de entrenamiento.
* **Capa densa oculta:** Otra capa Dense con 64 unidades y activación ReLU, que agrega complejidad al modelo y permite aprender representaciones más abstractas de los datos.
* **Capa de dropout:** Otra capa Dropout con una tasa de deserción del 50% para seguir ayudando a evitar el sobreajuste.
* **Capa de salida:** Una capa Dense con una unidad y función de activación sigmoidea (sigmoid). Esta capa produce una salida en el rango de 0 a 1, adecuada para problemas de clasificación binaria.

El modelo es compilado utilizando el optimizador Adam y la función de pérdida binary\_crossentropy, que es adecuada para problemas de clasificación binaria. Además, se especifica la métrica de precisión (accuracy) para evaluar el rendimiento del modelo durante el entrenamiento.

Para mejorar la robustez del entrenamiento y evitar el sobreajuste, se configura un mecanismo de EarlyStopping. Esta técnica monitoriza la pérdida de validación (val\_loss) y detiene el entrenamiento si la pérdida no mejora durante un número específico de épocas (en este caso, 3 épocas).

Finalmente, el modelo es entrenado utilizando el método fit, que ajusta los parámetros del modelo con los datos de entrenamiento (X\_train\_balanced y y\_train\_balanced). Se especifica un número de épocas (10) y un tamaño de lote (32), y se utiliza un 20% de los datos para validación durante el entrenamiento. El entrenamiento también utiliza el callback early\_stopping para interrumpir el entrenamiento si es necesario.

Una vez completado el entrenamiento, se imprime un mensaje indicando que el proceso de entrenamiento ha finalizado. Luego, se llama al método evaluar\_modelo para evaluar el rendimiento del modelo en datos de prueba o de validación, asegurando que el modelo ha sido correctamente entrenado y puede generalizar bien a nuevos datos.

**3.4.2.- Modelo de predicción.**

Este proceso se desglosa en varias fases, comenzando con la configuración de una ventana emergente destinada a la entrada del texto para la predicción. La ventana emergente, creada mediante el método abrir\_ventana\_prediccion, incluye un campo para ingresar el texto, un control deslizante para ajustar el umbral de decisión (que determina la sensibilidad de la clasificación), y un botón para activar la predicción. El umbral es un parámetro crucial que define el límite a partir del cual una probabilidad predicha se considera una clase positiva (por ejemplo, "Phishing").

El método realizar\_prediccion es responsable de manejar la lógica de predicción tras la entrada del usuario. Primero, valida que el campo de texto no esté vacío. Luego, obtiene el umbral de decisión y llama a predecir\_texto para obtener el resultado. En predecir\_texto, el texto ingresado es preprocesado y transformado usando las técnicas aplicadas durante el entrenamiento del modelo, como la vectorización TF-IDF. Posteriormente, se realiza una predicción basada en el modelo entrenado, que devuelve una probabilidad de que el texto pertenezca a la clase positiva. Esta probabilidad se compara con el umbral ajustado para determinar la clase final, y el resultado se muestra en la interfaz gráfica.

El flujo de trabajo también incluye métodos auxiliares como preprocess y transformar\_texto, que están diseñados para realizar el preprocesamiento y la transformación del texto de entrada, respectivamente. Aunque en el código actual están implementados como funciones placeholder, en una implementación completa, estas funciones se encargarían de limpiar y preparar el texto, aplicando técnicas como la normalización y la vectorización TF-IDF. La capacidad de ajustar el umbral permite personalizar la sensibilidad del modelo, proporcionando flexibilidad para equilibrar entre la precisión y la cobertura en la clasificación binaria de phishing frente a no phishing.

**4.- Resultados de las mediciones.**

**4.1.- Tiny Bert.**

Para iniciar las mediciones, vamos a mencionar lo siguiente, que para crear algoritmos de machine learning que sean capaces de clasificar líneas de textos o en este caso correos electrónicos con su respecto correo, encabezado y cuerpo del correo, es necesario tener algoritmos de hagan la labor de procesar el lenguaje natural, como se ha mencionado anteriormente, por ende Bert entrega modelos de algoritmo de redes neuronales ya entrenados en diferentes labores, pero el enfoque principal de estos modelos son el procesamiento del lenguaje natural, para esto y como se mencionó anteriormente, se usó dos modelos de redes neuronales, creadas por Hugging Face, enfocada en procesamiento del lenguaje natural y en el presente experimento, se estandarizo un algoritmo de fine tunning para los dos modelos, donde solamente cambian las métricas o hiperparametros de entrenamiento, en los cuales se observa el respectivo desempeño, a continuación, se mostraran las métricas de los dos modelos de BERT y Tiny BERT escogidos, cabe hacer presente que para definir el comportamiento de estos modelos, se tuvo que reducir demasiado el dataset, debido a la utilización de recursos que necesitada cada fine tunning y cada modelo pre entrenado, a lo cual se usaron 700 líneas de datos con etiquetas de phishing y no phishing, las que clasifican cada línea, además para mejorar un poco la toma de muestra y métricas estos algoritmos se ejecutaron en Google Colab, por la cantidad de recursos de memoria ram que ofrece.

**1.- TYNY BERT**

Para este modelo correspondiente a un modelo Tiny Bert, se debe crear el fine tunning, cumpliendo las métricas establecidas y medidas, expuestas por la empresa Hugging Face. Correspondientes a las que aparecen especificadas en el modelo **Json**, descargable de la página de Hugging Face. El cual define estos parámetros, de creación del fine tuning.

***“model\_name": "huawei-noah/TinyBERT\_General\_4L\_312D",***

***"num\_labels": 2,***

***"max\_length": 512,***

***"num\_train\_epochs": 3,***

***"train\_batch\_size": 8,***

***"eval\_batch\_size": 8,***

***"warmup\_steps": 500,***

***"weight\_decay": 0.01,***

***"logging\_steps": 10,***

***"evaluation\_strategy": "steps",***

***"save\_total\_limit": 1,***

***"learning\_rate": 2e-5***

En consecuencia, de esto se ocupó un modelo de fine tuning, donde la configuración de los hiperparametros fue referente al siguiente criterio, que en función de las épocas y su comportamiento, se decidió no usar tres épocas, se trató de estandarizar a 16 épocas, ya que el batch size se aumentó a un número multiplicado por 2, dando como resultado 36, ya que se multiplico dos veces, en relación de esto, se mantuvo el learning rate en 2e-5, donde al entrenar el referido fine tuning, para un espectro de 764 datos, correspondientes a correos electrónicos de Phishing y no Phishing se obtuvieron los siguientes resultados.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Validation loss** | **Accuracy** | **F1** | **Precisión** | **Recall** | **AUC** |
| **16** | **0.65** | **0.65** | |  | | --- | | **0.69** | | **0.60** | **0.80** | **0.74** |

(Tabla Nro.04, reporte de clasificación Tiny Bert)

(Fuente elaboración propia)

En este reporte de clasificación observamos la creación de un fine tuning para **TinyBert,** con buenos datos de clasificación, destacando que la creación de este fue realizado en la plataforma Google Colab y ocupando los recursos de memoria de esta, los cuales no son muy amplios, cabe hacer presente que por la composición del computador de trabajo en donde se efectuaron los experimentos, no fue capaz de realizar un entrenamiento completo en Visual Studio Code, en el cual se intentó entrenar con 8000 líneas de texto no dando fruto alguno, y al reducir, tampoco fue posible, por ende los resultados que se expresan en este apartado, son los mejores obtenidos hasta el momento, durante el entrenamiento de este modelo de Machine Learning, seguidamente y continuando con el respectivo análisis, en función del reporte de clasificación, se observan valores de alrededor del 60% de precisión en aprendizaje o entrenamiento, lo cual en primera instancia podemos mencionar que si bien no es un modelo robusto como tal, ya que posee diversas limitaciones para levantarlo en un entorno local, se logró obtener datos de entrenamiento que supera la media o el 50%, y saliendo del margen de aleatoriedad en el entrenamiento y reconocimiento, en función de lo anterior, evidenciaremos estos, con la muestra de la respectiva matriz de confusión.

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

(grafico Nro.01, matriz de confusión Tiny Bert)

(Fuente de elaboración propia)

En este apartado observamos lo siguiente de la respective matriz

|  |  |
| --- | --- |
| **Verdaderos Negativos (TN)** | **166** |
| **Falsos Positivos (FP)** | **162** |
| **Falsos Negativos (FN)** | **61** |
| **Verdaderos Positivos (TP)** | **244** |

(Tabla Nro.05, matriz de confusión Tiny Bert)

(Fuente de elaboración propia)

En relación al porcentaje de precisión del modelo que corresponde a un valor de aproximadamente 61% por ciento, Podemos decir que el respective modelo al entrenarse, puede detectar 166 verdaderos positivos y 162 falsos positivos, no obstante tiene la capacidad de detectar 244 falsos negativos y 61 falsos positivos, si bien los números nos muestra un entrenamiento que desbalanceado, en algunos aspectos, podemos divisar que es capaz de reconocer ambas clases con cierto grado de precisión, por ende es un modelo que hasta el momento es posible de mejorar.

En relación de esto, se procedió a ejecutar la curva de ROC, a fin de identificar el valor bajo la curva y que tanto Podemos mejorar el modelo o el margen de mejora o afinamiento y obtuvimos el siguiente resultado.

Gráfico, Gráfico de líneas

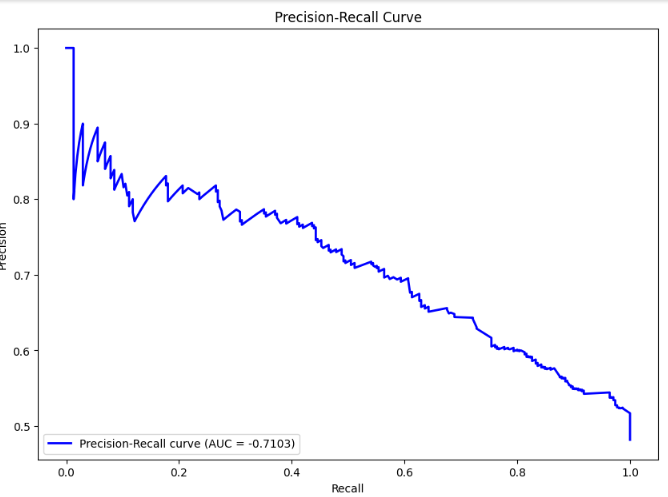
Descripción generada automáticamente

(grafico Nro.02, curva de ROC Tiny Bert)

(Fuente elaboración propia)

En esta curva, se logra observar que aún existe un margen de afinamiento del modelo de aproximadamente un 26%, el cual se puede mejorar en torno al entrenamiento a fin de tener valores precisos en clasificación ya que el valor de ROC es de 0.73681 o 74% aproximado, por ende, el modelo muestra que tiene aún parámetros a mejorar en su desarrollo, donde se pueden implementar diferentes valores en los hiperparametros de entrenamiento, como mejorar la etapa de preprocesamiento y tokenizacion, a fin de obtener valores más precisos, ademas implementar otro método de balanceo de datos, podria ayudar a mejorar la obtención de resultados de entrenamiento.

**Gráfico Precision Recall**



(grafico Nro. 03, grafico precisión recall)

(Fuente elaboración propia)

**2.- FINE TUNING DE PHISHING.**

Continuando con la fase experimental, se procedió a realizar un ajuste fino (finetuning) en el modelo preentrenado bert finetuned-phishing. Durante este proceso, se observaron resultados de entrenamiento que indican un desempeño moderado del modelo en cuanto a la clasificación de las clases "Phishing" y "No Phishing", utilizando un conjunto de datos obtenido de Kaggle. A continuación, se presentan los valores de evaluación obtenidos durante el entrenamiento del modelo:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Epoch** | **Accuracy** | **F1** | **Precisión** | **Recall** | **AUC** |
| **4** | **0.62** | |  | | --- | | **0.57** | | **0.64** | **0.52** | **0.69** |

(Tabla Nro.06, reporte de clasificación de fine tuned phishing)

(Fuente de elaboracion propia)

Estos valores indican que el modelo tiene una capacidad razonable para identificar las clases de interés, con una precisión superior al 63%, lo que sugiere que el modelo es relativamente bueno para predecir casos de "Phishing". El área bajo la curva ROC (0.6910) también refuerza esta conclusión, mostrando que el modelo tiene un desempeño adecuado en la discriminación entre las clases.

No obstante, el valor de **Recall** (0.5180) y el **F1 Score** (0.5704) sugieren que el modelo podría beneficiarse de ajustes adicionales o mejorar en la arquitectura del modelo afinado. Un **Recall** bajo indica que el modelo no está capturando todos los casos positivos, es decir, algunos correos electrónicos de "Phishing" no están siendo identificados, mientras que el **F1 Score** revela un equilibrio entre precisión y recall que podría mejorarse.

El **Accuracy** del 62.4% y el **ROC AUC** de 0.6910 son muestras de un modelo que ha logrado cierto grado de ajuste en identificación y clasificación, pero que aún podría mejorar su capacidad predictiva. Estos resultados, aunque prometedores, sugieren que el modelo está en una fase intermedia de desarrollo, y con un ajuste fino de los hiperparámetros, como la tasa de aprendizaje, la regularización, o incluso el uso de técnicas de balanceo de clases más sofisticadas, podría alcanzar un rendimiento superior, en el cual se podría obtener valores de entrenamiento, similares a los documentados por el repositorio Hugging Face.

En conclusión, el modelo **bert finetuned**-phishing muestra un desempeño inicial prometedor. Con los valores obtenidos, se puede considerar que el modelo tiene un buen punto de partida y es susceptible de mejoras adicionales para lograr una mayor precisión y recall, lo que lo convertiría en una herramienta más robusta para la detección de correos electrónicos de phishing.

**Valores del modelo creado por Hugging face**

***his model has the following performance in email and SMS phishing detection:***

**Accuracy: 0.990318, Precision: 0.990170, Recall: 0.984365, AUC: 0.999146**

**Los hiperparametros ideales de entrenamiento para el modelo, según Hugging Face, es dable mencionar que estos hiperparametros están en la arquitectura original del modelo en cuestión.**

* **24-layer**
* **1024 hidden dimension**
* **16 attention heads**
* **336M parameters**

El modelo **bert phishing text** tiene resultados impresionantes probados por Hugging Face con un **Accuracy** de 0.990318, **Precision** de 0.990170, **Recall** de 0.984365, y un **AUC** de 0.999146, lo que indica un rendimiento casi perfecto en la detección de phishing en correos electrónicos. (Huggingface), los valores anteriores, fueron el reporte de clasificación con el cual se entrenó el respectivo modelo BERT de manera original, demostrando con esto la alta capacidad que tiene para realizar clasificaciones y poder identificar correos electrónicos maliciosos, es por dicha razón la cual se decidió utilizar este modelo para intentar re entrenarlo con los datos en cuestión, logrando los valores descritos en mi reporte de clasificación, los cuales son valores moderadamente buenos e indican un entrenamiento inicial conforme, con vistas en el futuro de ser mejorado en su totalidad.

**Matriz de Confusión.**

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

(grafico Nro.04, matriz de confusión fine tunig bert)

(Fuente elaboración propia)

**Datos Matriz de confusión:**

|  |  |
| --- | --- |
| **Verdaderos Negativos (TN)** | **237** |
| **Falsos Positivos (FP)** | **91** |
| **Falsos Negativos (FN)** | **147** |
| **Verdaderos Positivos (TP)** | **158** |

(Tabla Nro. 07, matriz de confusión fine tuned phishing)

(Fuente de elaboración propia)

**Análisis de resultados.**

**Verdaderos Negativos (TN) = 237:** Referente a esto el modelo fue capaz de identificar correctamente 237 correos que efectivamente no eran phishing o clasificados como “No Phishing” lo cual es un buen indicador de su capacidad para evitar falsos positivos.

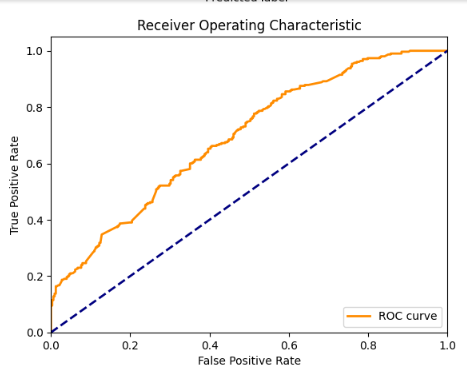
**Falsos Positivos (FP) = 91:** Este valor nos indica que, en 91 ocasiones, el modelo clasificó erróneamente un correo seguro como phishing.

**Falsos Negativos (FN) = 147:** este parámetro indica que 147 correos de phishing pasaron desapercibidos para el modelo, lo que sugiere que existe un riesgo de en las clasificaciones, ya que estos correos no fueron detectados correctamente.

**Verdaderos Positivos (TP) = 158:** Señala el número de correos que fueron correctamente identificados como "Phishing". En 158 casos, el modelo fue efectivo en detectar correos clasificados como “Phishing”.

El alto número de **TN** sugiere que el modelo es bueno para identificar correos seguros, mientras que el número de **TP** muestra que el modelo tiene cierta capacidad para detectar amenazas. Sin embargo, la presencia de **FN** y **FP** destaca áreas donde el modelo podría mejorar, especialmente para reducir la cantidad de correos de phishing que son mal clasificados o no detectados por el modelo (**FN**), así como para minimizar la clasificación errónea de correos maliciosos (**FP**). Esto indica que, aunque el modelo tiene una base sólida y que aún se tiene espacio para optimizaciones que mejoren su capacidad de detección.

**Curva de ROC**



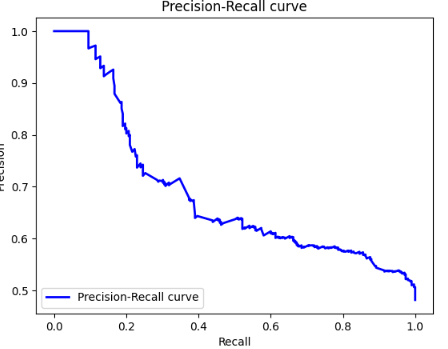
(grafico Nro.05, grafico curva de ROC fine tuning bert)

(Fuente elaboración propia)

El valor de **0.6910** bajo la curva ROC (Receiver Operating Characteristic) es un indicador del rendimiento general del modelo al diferenciar entre las clases "Phishing" y "No Phishing". La curva ROC muestra la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) a través de diferentes umbrales de decisión.

Por ende, un valor de **0.6910** implica que el modelo tiene un rendimiento aceptable en función de clasificación, pero aún podría beneficiarse de ajustes adicionales para mejorar su capacidad de discriminación entre correos de phishing y no phishing, un ejemplo de esto es mejorando o usando otra configuración de hiperparametros de entrenamiento o afinando el modelo de tokenizado y preprocesado.

**Gráfico Precision Recall**



(grafico Nro.06, grafico precisión recall, fine tuning bert)

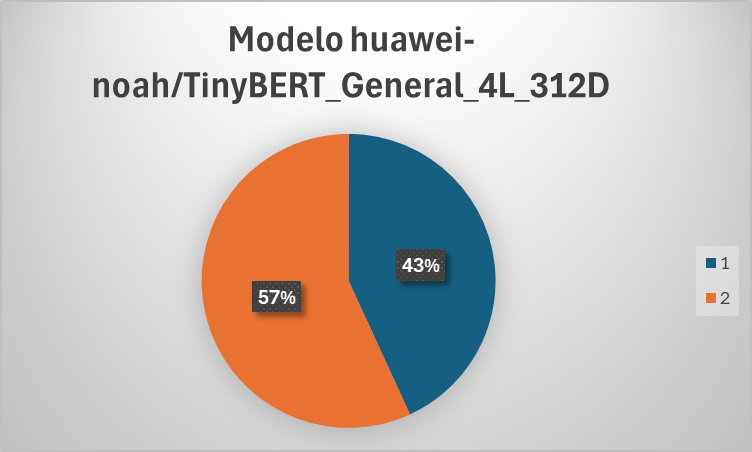
(Fuente elaboración propia)

**Cálculos obtenidos en las clasificaciones.**

En esta fase es importante señalar que se usó un dataset de aproximadamente 51 líneas de texto, que no fueron incluidas en el entrenamiento de cada modelo, con el fin de observar su comportamiento en reconocimiento.

Continuando con la fase experimental de los modelos BERT y Tiny BERT, se procedió a realizar las respectivas mediciones, a finde verificar y cuantificar la capacidad de clasificación de ambos modelos BERT y Tiny Bert, en tareas de reconocimiento de correos electrónicos maliciosos, para lo siguiente se tomaron diferentes muestras de un dataset, de datos, distintos del que se entrenaron ambos modelos a fin de cuantificar su respectiva capacidad de clasificación, en razón de esto, se comenzó cuantificando el modelo **TinyBert,** respecto a su capacidad predictora, obteniendo los siguientes resultados.

**Tiny Bert**



(grafico Nro.07, grafico desempeño Tiny Bert)

(Fuente elaboración propia)

Acá podemos inferir lo siguiente, que el modelo antes mencionado predijo correctamente solo el 43% de las líneas de textos a predecir, y con un 57% de predicciones erróneas, en este caso, podemos mencionar que las predicciones a diferencia de los datos de entrenamiento si se efectuaron con un nivel de aleatoriedad alto, por ende los datos obtenidos no reflejan el reporte de clasificación del entrenamiento del respectivo modelo, ya que los valores correctos están muy por debajo de los valores de precisión y accuracy del entrenamiento.

**Reporte de clasificación TinyBert, del modelo entrenado.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Epoch | Validation loss | Accuracy | F1 | Precission | Recall | AUC |
| 16 | 0.65 | 0.65 | |  | | --- | | 0.69 | | 0.60 | 0.80 | 0.74 |

(Tabla Nro.08, reporte de clasificación Tiny Bert)

(Fuente elaboración propia)

**Datos obtenidos y cuantificados con relación a predicciones, los cuales se presentarán mediante tabla dinámica.**

Al observar la respectiva tabla dinámica y comparar los valores de precisión y accuracy vs los valores positivos de predicción, se observa que en los valores de predicción son mucho menor que los de entrenamiento, lo cual demuestra que el modelo predictivo no se ajustó correctamente, además es dable mencionar que la cantidad de datos con la que se entrenó, no alcanza hacer el 10% del dataset original, ya que el modelo o en general los modelos BERT requieren, demasiados recursos, de los cuales no se tenían en el presente experimento.

**Tabla dinámica.**

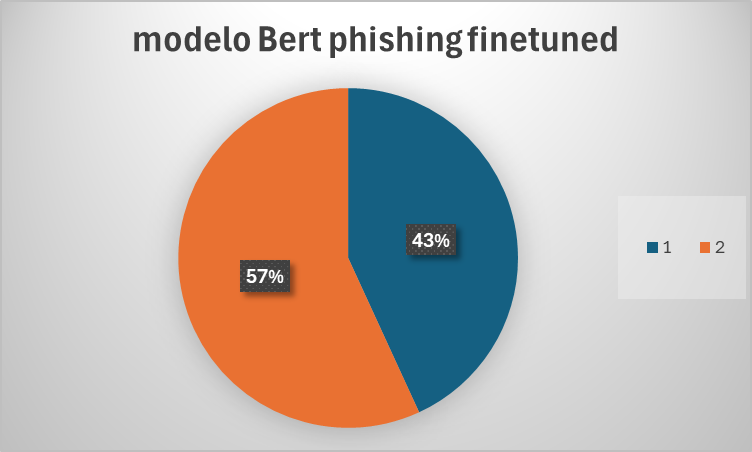
**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

(tabla Nro.09, tabla dinámica Tiny Bert)

(Fuente elaboración propia)

**Fine tuning de Phishing**

****

(grafico Nro.08, grafico de rendimiento fine tuning bert)

(Fuente elaboración propia)

Acá podemos inferir lo siguiente, que el modelo antes mencionado predijo correctamente solo el 43% de las líneas de textos a predecir, y con un 57% de predicciones erróneas, en este caso, podemos mencionar que las predicciones a diferencia de los datos de entrenamiento si se efectuaron con un nivel de aleatoriedad alto, por ende los datos obtenidos no reflejan el reporte de clasificación del entrenamiento del respectivo modelo, ya que los valores correctos están muy por debajo de los valores de precisión y accuracy del entrenamiento.

**Reporte de clasificación Fine tuned phishing, del modelo entrenado**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Epoch** | **Accuracy** | **F1** | **Precisión** | **Recall** | **AUC** |
| **4** | **0.62** | |  | | --- | | **0.57** | | **0.64** | **0.52** | **0.69** |

(tabla Nro.10, reporte de clasificación Fine tuned phishing)

(Fuente elaboración propia)

**Datos obtenidos y cuantificados con relación a predicciones, los cuales se presentarán mediante tabla dinámica.**

Al observar la respectiva tabla dinámica y comparar los valores de precisión y accuracy vs los valores positivos de predicción, se observa que en los valores de predicción son mucho menor que los de entrenamiento, lo cual demuestra que el modelo predictivo no se ajustó correctamente, ademas es dable mencionar que la cantidad de datos con la que se entrenó, no alcanza hacer el 10% del dataset original, ya que el modelo o en general los modelos BERT requieren, demasiados recursos, de los cuales no se tenían en el presente experimento.

**Tabla dinámica**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

(Tabla Nro.11, tabla dinámica Fine tuned phishing)

(Fuente elaboración propia)

**Comparación de precisión en entrenamiento de los modelos estudiados**

|  |  |  |
| --- | --- | --- |
|  | **Finetuning de Phishing** | **TinyBERT** |
| **Precision** | **0.64** | **0.60** |
| **Acuraccy** | **0.62** | **0.65** |

(Tabla Nro.12, comparación de modelos estudiados)

(Fuente elaboración propia)

Debido a la tabla anterior y a los resultados tanto del entrenamiento, como de las respectivas predicciones y según el artículo científico “**Un Método para Detectar Sitios Web de Phishing Basado en Tiny-BERT Stacking”,** específicamente en el apartado decomparaciones de otros modelos de machine learning utilizado en el reconocimiento de correos electrónicos maliciosos, URLs, entro otros, mencionan lo siguiente.

Para superar tales desafíos de correos electrónicos maliciosos, hemos propuesto un método basado en Tiny-BERT Stacking para detectar sitios web de phishing. El concepto central de este método es primero usar tiny-BERT para convertir las URL en vectores de características, y luego usar CatBoost, XGBoost, LightGBM y GBDT como aprendices básicos, formando un modelo Stacking a través del aprendizaje en ensamblaje, y usar vectores de características de URL para extraer profundamente la información de características para detectar sitios web de phishing. Los resultados experimentales han mostrado que nuestro modelo propuesto no solo evita el tedioso proceso de extracción manual de características, sino que también **logra una alta precisión en la tarea de detección de sitios web de phishing.** ((Lv et al., 2024)

podemos ver que nuestro modelo propuesto también funciona bien en conjuntos de datos grandes con una precisión del 99.14%, que es un 1.16% y un 1.24% superior al uso de solo los modelos BERT y tiny-BERT, respectivamente. Y nuestro modelo es más estable, con un F1\_score del 99.01%. Referimos el número de iteraciones del experimento como Epoch. ((Lv et al., 2024)

en razón de lo anterior, podemos concluir que la comparación de ambos modelos, en nuestro caso experimental tanto como Tiny Bert y el modelo Bert, respectivamente, para la misma cantidad de datos de entrenamiento para ambos, tuvieron un comportamiento similar en datos de entrenamiento, no obstante, a lo propuesto por el articulo científico antes mencionado, denominado “**Un Método para Detectar Sitios Web de Phishing Basado en Tiny-BERT Stacking”, el cual señala que el modelo Tiny BERT,** usado para detectar correos electrónicos maliciosos combinado con modelos de **CatBoost, XGBoost, LightGBM y GBDT** como aprendices básicos, generan mayor impacto y tienen una mejor precisión en la detección de estos, no obstante al comparar el modelo hibrido del artículo, con otros modelos de machine learning, el modelo hibrido, tenía un comportamiento mejor ante reconocimiento de correos maliciosos, de aproximadamente de un 1.16% y un 1.24% superior al uso de solo los modelos BERT y tiny-BERT ((Lv et al., 2024), no obstante a los números que muestra, en relación al comportamiento de los modelos Bert y Tiny Bert para la misma tarea, se observa un mejor comportamiento para modelos Bert, dado por su complejidad y la cantidad de redes neuronales que aloja este.

Dicho esto, el articulo menciona que los modelos Tiny Bert, por si solo si son efectivos en reconocimiento de correos electrónicos, no obstante, tiene una mejor eficiencia con modelos Bert para una tarea similar, esto en el presente experimento lo podemos corroborar, en función a los valores de entrenamiento obtenidos y tabulados en las respectivas tablas de reporte de clasificación de ambos modelos, se observa que el modelo Bert tiene una precisión de 2% aproximadamente mayor que el modelo Tiny Bert, para tareas de clasificación, esto usando solamente los valores de entrenamiento, ya que por razones de recursos, no se pudo medir con exactitud los valores de clasificación, estos datos acreditan, lo señalado en el artículo anterior, en el cual señala que los modelos Bert tienen 1% o 2% más de precisión, que los modelos Tiny Bert, para una tarea similar, no obstante a esto, si al modelo Tiny Bert, se crea de manera hibrida, tal como lo propone dicho artículo, es posible que tenga resultados más exactos, no obstante, esta última conclusión debe ser comprobada aun, lo cual no fue materia del presente artículo, lo que se presenta una arista a futuras investigaciones.

**4.2.- Clasificador Naive Bayes:**

Continuando con el ciclo experimental a continuación, se probará el desempeño de un clasificador para la tarea de identificación y clasificación de correos electrónicos maliciosos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **support** |
| **No Phishing (0)** | **0.84** | **0.88** | **0.86** | **4227** |
| **Phishing (1)** | **0.87** | **0.83** | **0.85** | **4211** |

(Tabla Nro.13, reporte de clasificación Naive Bayes)

(Fuente elaboración propia)

**Precisión: 0.8545863948803034.**

**Porcentaje de precisión: 86%**

En este apartado, en específico, se observa las métricas de entrenamiento del modelo de clasificación Naive Bayes, en el cual de manera preliminar se obtienen buenos resultados de su entrenamiento, en el cual podemos mencionar o generar la hipótesis de que el modelo creado, se entrenó de manera efectiva, con un alto porcentaje de precisión, lo que lo hace una herramienta de machine learning robusta, en función del reconocimiento y predicciones de correos electrónicos que pudiesen pertenecer a delitos de phishing.

La matriz de confusión muestra la cantidad de cada tipo de clasificación realizada por el modelo. Los verdaderos negativos (TN) y verdaderos positivos (TP) representan las instancias correctamente clasificadas, mientras que los falsos positivos (FP) y falsos negativos (FN) indican las instancias mal clasificadas.

**Matriz de Confusión:**

|  |  |
| --- | --- |
| **Verdaderos Negativos (TN)** | **3717** |
| **Falsos Positivos (FP)** | **510** |
| **Falsos Negativos (FN)** | **717** |
| **Verdaderos Positivos (TP)** | **3494** |

(Tabla Nro.14, matriz de confusión)

(Fuente de elaboración propia)

Seguidamente y a fin de explicar un poco el enfoque de las métricas de mediciones que ocupa el modelo de clasificación podemos mencionar lo siguiente, debido a los valores obtenidos y cuantificados.

Precisión (Precision): La precisión indica la proporción de verdaderos positivos entre todas las instancias clasificadas como positivas. En el contexto del modelo Naive Bayes, una precisión global del 86% sugiere que, de todas las instancias que el modelo clasificó como positivas, el 86% eran verdaderamente positivas. Esta métrica es crucial para evaluar la exactitud del modelo en términos de evitar falsas alarmas.

Recall: El recall mide la proporción de verdaderos positivos detectados entre todas las instancias verdaderamente positivas. El modelo muestra un recall del 88% para la clase 0 y del 83% para la clase 1. Estos valores indican que el modelo es más efectivo en identificar la clase 0 que la clase 1. Un recall alto sugiere que el modelo es eficiente en la detección de las instancias de cada clase, aunque se observa una ligera disminución en la capacidad para identificar la clase 1, no obstante, en este punto, podemos mencionar que esa diferencia de 5% entre la identificación de ambas clases puede ser insignificante en los resultados globales.

F1-Score: El F1-score es la media armónica de precisión y recall, proporcionando un equilibrio entre estas dos métricas. Un F1-score promedio ponderado de 0.86 indica que el modelo mantiene un buen equilibrio entre precisión y recall, ofreciendo una medida integral de su rendimiento en la clasificación.

Matriz de Confusión: La matriz de confusión revela que el modelo tiene un buen rendimiento general, con una mayor cantidad de verdaderos positivos y verdaderos negativos en comparación con los falsos positivos y falsos negativos. Sin embargo, se observa que el número de falsos negativos es significativo. Esto puede sugerir que el modelo tiene dificultades para identificar ciertos casos positivos, lo que podría mejorarse ajustando el modelo o utilizando técnicas adicionales de preprocesamiento y balanceo de datos.

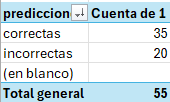
Se puede generar la hipótesis con los datos obtenidos en el entrenamiento que el modelo Naive Bayes demuestra una capacidad sólida para la clasificación e identificación de correos electrónicos maliciosos.

**Porcentaje de precisión.**

A fin de determinar el porcentaje de precisión en las predicciones y/o reconocimiento de correos electrónicos maliciosos, se efectuó una prueba, seleccionando cada línea de texto, que esta compuesta por tres columnas que son From(SMTP), From (Header) y Subject, la cual estructuran el correo electrónico completo, donde podemos obtener información del correo remitente, del encabezado y cuerpo del mismo, a fin de poder determinar y clasificar que correo electrónico pertenece o se clasifica como correo de ataque de phishing y cual no es un correo de phishing.

Para este caso, se marcó con 0 si es una clasificación errónea y con 1, si es una clasificación positiva, lo cual para el caso de predicciones del modelo Naive Bayes, obtuvimos un porcentaje promedio, del total de las líneas de texto correspondientes a correos electrónicos, las cuales conforman un espectro total de 55 líneas para hacer pruebas, donde se obtuvo una precisión del **64.29% de precisión**, obteniendo un valor optimo de predicción, en el cual se puede concluir que el entrenamiento del modelo fue consistente.

**Tabla dinámica de datos.**

****

(Tabla Nro. 15, tabla dinámica Naive Bayes)

(Fuente elaboración propia)

En la tabla dinámica se observa a simple vista que predomina las predicciones erróneas en el modelo de clasificación Naive Bayes con 34 predicciones de esta categoría, no obstante, obtenemos solo 20 predicciones correctas, las cuales contemplan en estas 20 predicciones correctas, las clasificaciones establecidas en el mapeo de etiquetas correspondientes a 0= no phishing y 1= phishing.

Gráfico, Gráfico circular

Descripción generada automáticamenteNomenclatura de colores

Naranjo: resultados predichos de manera incorrecta: **35.76%**

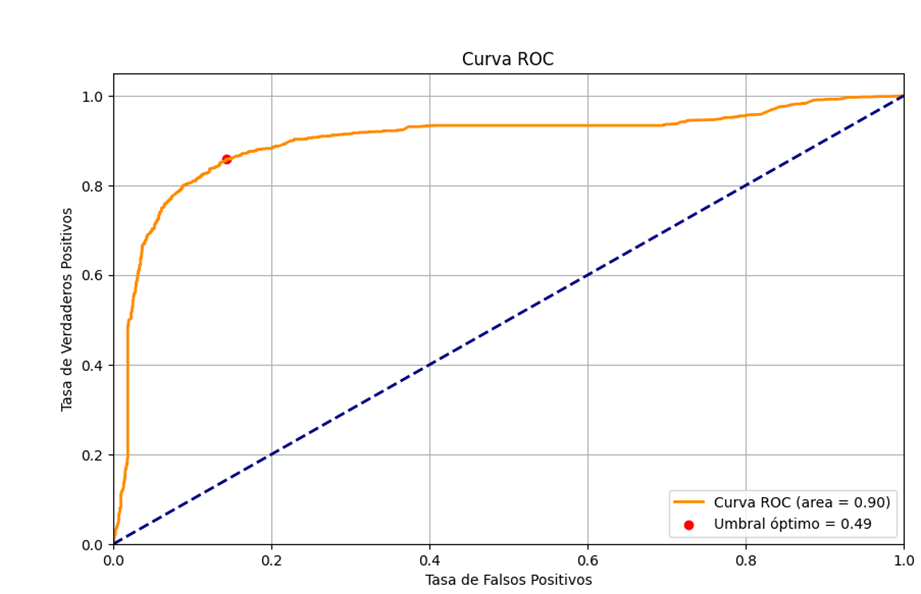
Azul: resultados predichos de manera correcta: **64.29**

(grafico Nro.09, grafico rendimiento Naive Bayes)

(Fuente elaboración propia)

**Curva de ROC.**

Los resultados obtenidos en las predicciones se pueden avalar en función de la curva de ROC, debido a esto tenemos en el cálculo del área bajo la curva un porcentaje de aproximadamente 0.90, en el cual nos muestra que el modelo de machine learning diseñado, tiene muy poco margen para clasificar de manera aleatoria, siendo un modelo de clasificación robusto para el cumplimiento de la tarea propuesta.

****

(grafico Nro.10, curva de ROC Naive Bayes)

(Fuente elaboración propia)

**4.3.-Red neuronal profunda:**

En esta etapa experimental que corresponde a las mediciones de la red neuronal profunda, es importante señalar que se harán dos experimentos referente al comportamiento de estas redes en cuestión, se medirá primero con la totalidad del número de neuronas con el que se creó el modelo inicialmente, para lo cual se procederá a tomar las respectivas muestras y observar cómo se comportan en relación a las clasificaciones que deben realizar, luego se realizara las mismas pruebas, pero disminuyendo la cantidad de neuronas a la mitas y observando su comportamiento tanto en entrenamiento como en predicciones.

En función de lo anterior y posterior a las mediciones efectuadas al Clasificador Naive Bayes, se ejecutaron las pruebas de la red neuronal profunda, para lo cual se obtuvieron de manera preliminar los siguientes resultados del reporte de clasificación, cabe hacer mención que, en esta primera prueba, se hicieron las mediciones con tres capaz de neuronas, conformadas por la primera capa con 128, la segunda con 64 y la última con uno, con un dropout de 0.5.

**Imagen ilustrativa.**

Texto

Descripción generada automáticamente

(Imagen Nro. 04, imagen ilustrativa de la arquitectura de la red neuronal)

(Fuente elaboración propia)

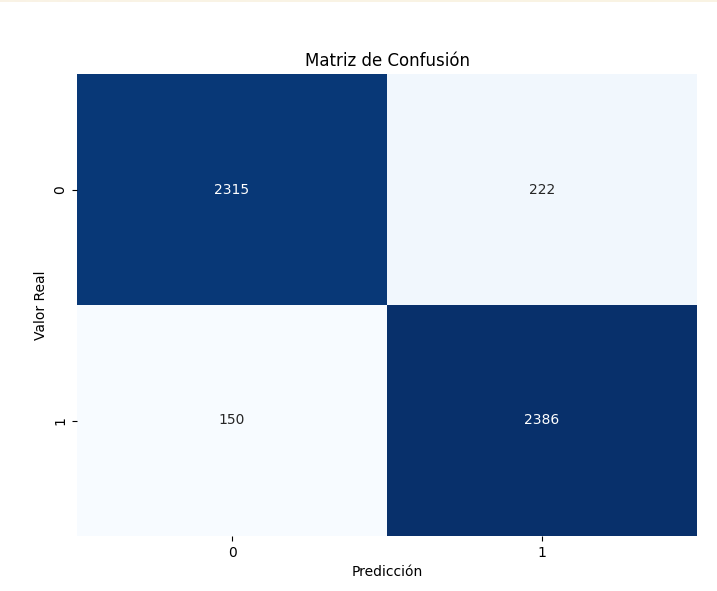
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F-1 score** | **Support** |
| phishing | 0.94 | 0.91 | 0.93 | 2537 |
| No phishing | 0.91 | 0.94 | 0.93 | 2536 |

(Tabla Nro.16, reporte de clasificación red neuronal profunda)

(Fuente de elaboración propia)

En primera instancia se puede observar con claridad una similitud en los valores de cada etiqueta esto lo observamos en este reporte de clasificación en el cual la diferencia de precisión, recall F-1 score y support es mínima, siendo de aproximadamente un porciento en cada ítem, lo que nos permite inferir a simple vista que tal diferencia casi insignificantica en ambas etiquetas se da debido a un balanceo correcto de los datos del dataset utilizado para el referido entrenamiento, obteniendo niveles de muestras similares para cada etiqueta, logrando con esto cualquier problemas de sobre ajuste o sub ajuste que se puede suscitar con los datos desbalanceados.

En función de esto, podemos observar los resultados de la matriz de confusión en el cual nos evidencia la cantidad de verdaderos y falsos positivos y negativos identificados por el modelo de red neuronal profunda, obteniendo los siguientes resultados:

****

(grafico Nro.11, matriz de confusión red neuronal profunda)

(Fuente elaboración propia)

**Matriz de confusión:**

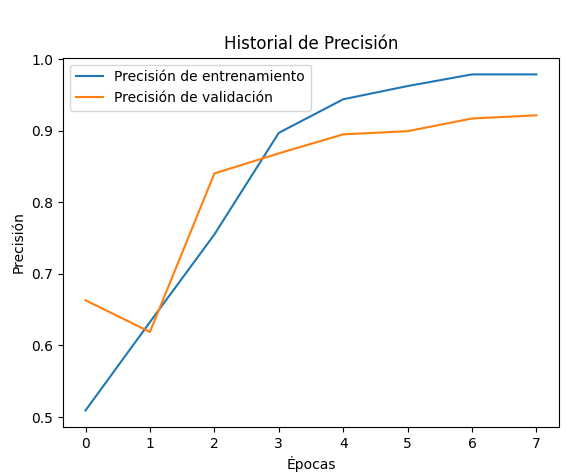
|  |  |
| --- | --- |
| **Verdaderos Negativos (TN)** | **2315** |
| **Falsos Positivos (FP)** | **222** |
| **Falsos Negativos (FN)** | **150** |
| **Verdaderos Positivos (TP)** | **2386** |

(Tabla Nro.17, matriz de confusión red neuronal profunda)

(Fuente elaboración propia)

Debido a esta matriz, podemos concluir preliminarmente que el modelo de red neuronal muestra un buen desempeño en la clasificación de correos electrónicos. Los valores indican que el modelo ha identificado correctamente son 2315 correos electrónicos como verdaderos negativos (TN) y 2386 correos electrónicos como verdaderos positivos (TP), reflejando una buena capacidad para reconocer correctamente los casos de "no phishing" y "phishing", respectivamente. Sin embargo, también se observan 222 falsos positivos (FP) y 150 falsos negativos (FN). Estos números sugieren que, aunque el modelo tiene una alta precisión en general, aún hay margen para mejorar en la reducción de errores tanto en la identificación de correos electrónicos no phishing como en la detección de correos phishing, no obstante, a los datos obtenidos precedentemente el margen de mejora en el entrenamiento no es mucho, puede ser una mejora de 5% más en precisión de detección de clases, lo que no tendría un impacto significativamente enorme en el modelo.

**Gráfico de entrenamiento del modelo de red neuronal profunda.**

**Gráfico, Gráfico de líneas

Descripción generada automáticamente**

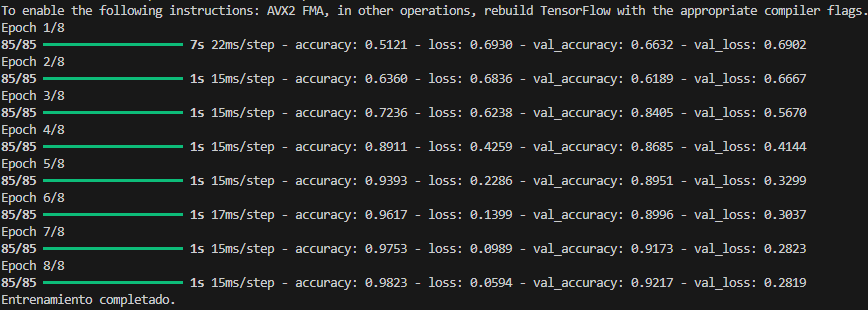
(grafico Nro.12, grafico de desempeño en entrenamiento)

(Fuente elaboración propia)

En el gráfico de rendimiento de los valores de entrenamiento, validación y perdida, podemos observar cuatro fenómenos distintos que se dan al momento de entrenar un modelo de red neuronal, primero observaremos la perdida durante el entrenamiento, la cual mide el error entre las predicciones del modelo y los valores reales. Es una función matemática que se minimiza a través de algoritmos de optimización, como la gradiente descendente, para ajustar los pesos del modelo. Una pérdida baja indica un buen ajuste del modelo a los datos, mientras que una pérdida alta sugiere que el modelo está realizando predicciones incorrectas. Monitorear la pérdida ayuda a evaluar el rendimiento del modelo y su capacidad para generalizar a nuevos datos.

En relación con esto, a continuación, expongo el ejemplo del entrenamiento del modelo y el comportamiento inversamente proporcional que tienen tanto la perdida como el entrenamiento y la validación.

Imagen demostrativa del entrenamiento del modelo y sus respectivas épocas.



(Imagen Nro.05, ilustración del entrenamiento)

(Fuente elaboración propia)

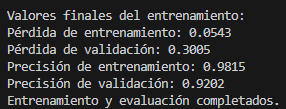
Seguidamente y en función de los valores obtenidos en la gráfica de desempeño de la red neuronal profunda, se observa que durante el entrenamiento, en la primera época de este, correspondiente al inicio, se observa una precisión o accuracy de 0.5121 (51%), lo cual indica que en un principio, la efectividad del modelo para entender y clasificar datos es igual 50% del total, el cual es un valor inicial bajo, para interpretar y determinar patrones, no obstante, y como lo mencione anteriormente, ya que estos valores son inversamente proporcionales el uno del otro, la tasa de perdida que se obtuvo en la primera época de entrenamiento corresponde a 0.6930 (69%), la cual es una tasa alta, por ende en un principio el modelo no estaba del todo ajustado, no obstante al finalizar el entrenamiento, específicamente en la época 8, se observa un accuracy de 0.9823 (98%) lo cual indica que el modelo esta ajustado, haciendo predicciones correctas, con alta precisión, de igual manera se observa una baja perdida de alrededor de 0.0594 (0.06%), lo cual indica que el modelo de red neuronal esta adecuadamente ajustado para realizar predicciones y clasificaciones en razón de su entrenamiento, de manera precisa.

Los valores de validación en el entrenamiento de un modelo de red neuronal indican cómo se desempeña el modelo con datos que no ha visto durante el entrenamiento. Estos valores proporcionan una medida de la capacidad del modelo para generalizar a nuevos datos y evitar el sobreajuste.

Un buen rendimiento en la validación significa que el modelo no solo ha aprendido de los datos de entrenamiento, sino que también puede hacer predicciones precisas sobre datos no vistos. Por lo general, se monitorean métricas como la precisión. Si el rendimiento en validación es significativamente peor que en entrenamiento, puede ser un indicio de sobreajuste y estaríamos en frente de un modelo que generaría predicciones erróneas o aleatorios.

Un val\_accuracy de 0.9217 indica que el modelo tiene una precisión del 92.17% en el conjunto de validación, sugiriendo un buen desempeño en datos no vistos. Una val\_loss de 0.2819 muestra que el modelo tiene una pérdida relativamente baja en el conjunto de validación, indicando que está haciendo predicciones bastante precisas y con errores moderados.

Finalmente, y luego de entrenarse completamente el modelo se obtienen estos datos:



(Imagen Nro.06, imagen de resultado de entrenamiento)

(Fuente elaboracion propia)

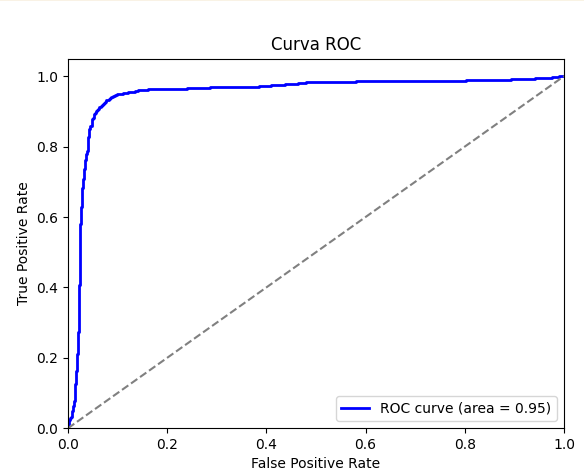
|  |  |
| --- | --- |
| **Pérdida de entrenamiento** | **0.543** |
| **Pérdida de validación** | **0.3005** |
| **Precisión de entrenamiento** | **0.9815** |
| **Precisión de validación** | **0.9202** |

(tabla Nro.18, tabla valores de entrenamiento)

(Fuente elaboracion propia)

Observando que el modelo, mantiene una precision del 98% de entrenamiento y validacion del 92%, haciendolo un modelo robusto ante la clasificacion correos electronicos, obteniendo datos resultados que acreditan que las predicciones se hara en funcion de la precision del entrenamiento y no es un modelo predictivo aleatoreo.

**Curva de ROC**

****

(grafico Nro.13, curva ROC red neuronal profunda)

(Fuente elaboración propia)

La curva ROC muestra la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) del modelo. TPR describe la tasa a la que el clasificador predice como “positivo” a las observaciones que son “positivas”. FPR describe la tasa a la que el clasificador predice como “positivo” a las observaciones que en realidad son “negativas”. Los clasificadores perfectos tienen una TPR de 1 y una FPR de 0. (deland, s.f.)

Cabe hacer presente, que el valor del área bajo la curva, de la curva de ROC, representa los valores de identificación y predicción, donde un valor de 0.5 estaríamos en presencia de un modelo aleatorio, en tanto, un modelo que tenga un área bajo la curva de 1, es un modelo de predicciones precisas, en este caso, tenemos un valor de 0.95 (95%), el cual nos permite mencionar que nuestro modelo es robusto en relación a la detección de verdaderos y falsos positivos y negativos, por ende tiene un porcentaje de error de aproximadamente de un 0.05 (5%), lo cual no generara un impacto significativo en el modelo, lo que lo hace un modelo altamente eficaz de generar predicciones exactas.

**Gráfico Precision Recall.**

**Gráfico

Descripción generada automáticamente**

(grafico Nro.14, grafico precisión recall red neuronal profunda)

(Fuente elaboración propia)

En el gráfico de precisión recall, nos muestra la capacidad del modelo que tiene para ser mejorado, tanto en el entrenamiento como en su precisión, por lo cual, en el presente gráfico, podemos observar el siguiente resultado del grafico precisión recall, siendo de un 0.97 o 97%, por ende, con este valor lo que muestra que el modelo necesita poco o nada de un refinamiento en su arquitectura.

**Predicciones de las redes neuronales profundas.**

En este apartado del presente estudio, se efectuarán cuatro mediciones en total, del comportamiento real del modelo, en el cual mediremos su capacidad de predictora y clasificatoria antes líneas de textos que componen un correo electrónico, las que están conformadas por la dirección de correo electrónico, el encabezado del correo y el cuerpo de este, esto con la finalidad de cuantificar la capacidad de la maquina en tareas de clasificación.

**Resultados de predicciones con umbral de predicción de 0.4**

(grafico Nro.15, grafico de rendimiento en predicción)

(Fuente elaboración propia)

Para comenzar se realizaron, las pruebas de predicción con un umbral de precisión del 0.4%, el cual significa de cuán seguro debe estar el modelo antes de clasificar un texto como "Phishing".

Para este caso obtuvimos aproximadamente un 59% de precisión en los resultados, lo cual se contrasta con los datos de entrenamiento y validación, siendo un porcentaje menor, pero demostrando que el modelo, hace predicciones correctas en un 59%, lo cual en su función de predicción estaríamos ante un modelo de buen desempeño, el cual podría mejorar o poder mejorar en relación con predicciones.

Cabe hacer presente que la muestra a testear, fueron aproximadamente 51 líneas de texto, correspondientes a correos electrónicos con diferentes categorías.

**Tabla dinámica de datos.**

Tabla

Descripción generada automáticamente

(tabla Nro. 19, tabla dinámica red neuronal profunda)

(Fuente elaboración propia)

**Resultados de predicción con umbral de predicción del 0.3**

(grafico Nro.16, grafico de rendimiento en predicción)

(Fuente elaboración propia)

En este punto en particular se obtuvo valores mejores que el umbral anterior, entendiendo que se disminuyó el umbral de predicción, en este caso en particular se disminuyó en un punto del umbral, bajando de 0.4 a 0.3, pero se observa un comportamiento más eficaz en clasificaciones correctas, aumentando caso en 5% las respectivas clasificaciones de líneas de textos, por lo que el comportamiento del modelo estudiado da como resultado que a valores inferiores al 0.5, se obtienen clasificaciones más exactas.

**Tabla dinámica**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

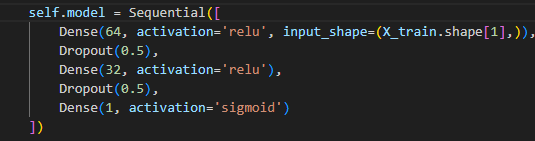
(tabla Nro.20, tabla dinámica, red neuronal profunda)

(Fuente elaboración propia)

**Red neuronal profunda, disminuyendo su cantidad de neuronas por capaz, división por 2**

En este caso, se probó el mismo algoritmo de redes neuronales profundas, no obstante, como se mencionó en el anunciado de este párrafo, las capaz se dividirán por 2, que es lo que quiere decir que la capa que contiene 128 redes neuronales se reducirá a 64 y la capa que tenía 64 redes neuronales se redujo a 34, donde al observar las métricas de entrenamiento y validación, se obtiene un modelo bastante preciso, al igual que el modelo mencionado precedentemente, el que ya fue medido en su desempeño, por lo tanto se medirá el desempeño del modelo propuesto con la reducción de sus redes neuronales a la mitad..

**Imagen ilustrativa**



(Imagen Nro. 07, imagen ilustrativa de la arquitectura de la red neuronal profunda reducida)

(Fuente elaboración propia)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F-1 score** | **Support** |
| phishing | 0.94 | 0.88 | 0.91 | 2537 |
| No phishing | 0.89 | 0.95 | 0.92 | 2536 |

(Tabla Nro. 21, reporte de clasificacio, red neuronal profunda disminuida a la mitad)

(Fuente elaboracion propia)

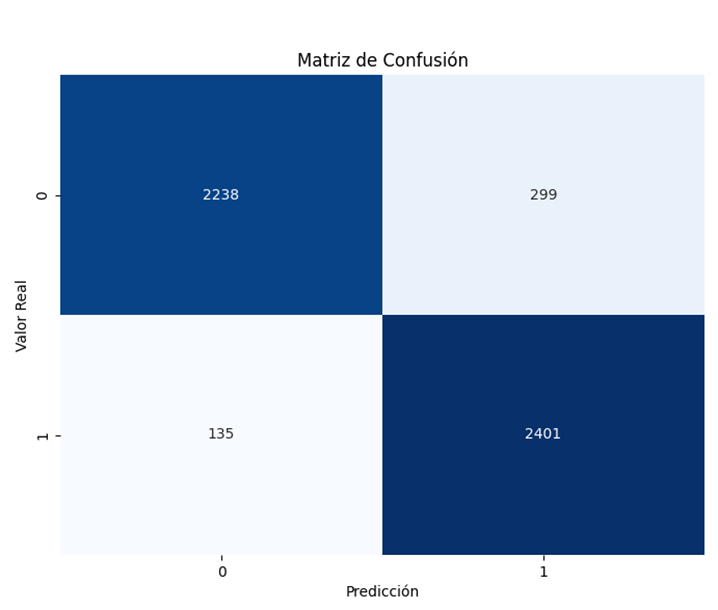
**Precisión de entrenamiento: 0.9760**

**Precisión de validación: 0.8981**

En este punto, se observa una precisión de entrenamiento de un 98% aproximado y una precisión de validación del 90% aproximado, lo que lo hace un modelo, robusto en su entrenamiento y en la identificación de datos y su respectiva clasificación, que, en este caso, debe clasificar, datos de correos electrónicos maliciosos, observando a simple vista una mínima disminución en los valores de precisión y validación versus los obtenidos en el primer modelo de red neuronal profunda, los cuales alcanzaron valores de precisión y validación de 98% en entrenamiento y 92% en validación, observando una diferencia en milésimas entre ambos modelos, siendo diferencias no significativas en función del entrenamiento del modelo de red neuronal, compuestos por 64 neuronas en su capa inicial, con 32 en la capa subsiguiente y 1 en la capa final.

Como ya se señaló, se observa una mínima diferencia en los valores de entrenamiento, de aproximadamente 0.01 punto porcentual y los valores de precisión similares, lo que realmente muestra que ambas arquitecturas de redes neuronales, hasta el momento ofrecen un gran desempeño para la tarea que se está investigando.

**Matriz de confusión.**

****

(grafico Nro.17, matriz de confusión)

(Fuente de elaboración propia)

|  |  |
| --- | --- |
| **Verdaderos Negativos (TN)** | **2238** |
| **Falsos Positivos (FP)** | **299** |
| **Falsos Negativos (FN)** | **135** |
| **Verdaderos Positivos (TP)** | **2401** |

(Tabla Nro. 22, matriz de confusión red neuronal profunda disminuida a la mitad)

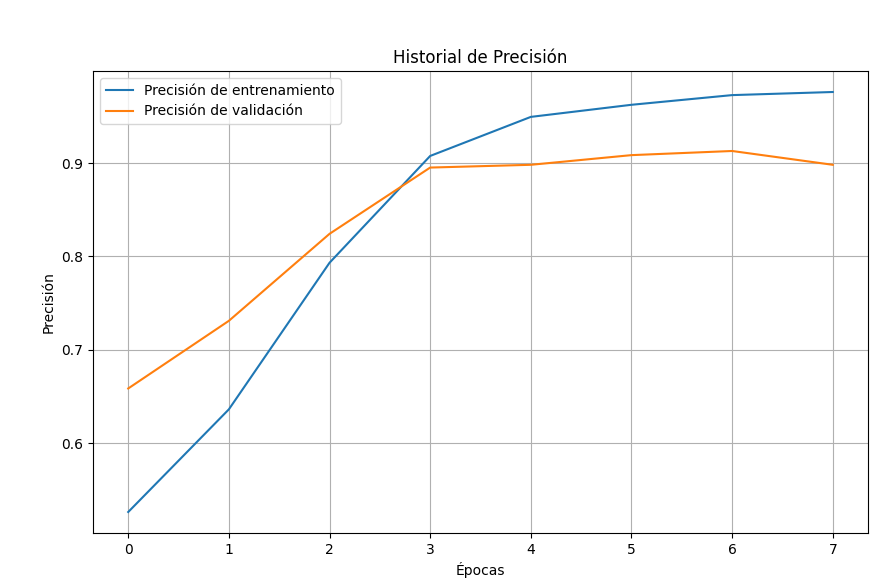
(Fuente elaboración propia)

Seguidamente y en relación de los datos de la matriz de confusión del modelo estudiado, podemos observar valores que muestran que el modelo tiene un buen desempeño en la clasificación de correos electrónicos. Los valores indican que el modelo ha identificado correctamente son 2238 correos electrónicos como verdaderos negativos (TN) y 2401 correos electrónicos como verdaderos positivos (TP), reflejando una buena capacidad para reconocer correctamente los casos de "no phishing" y "phishing", respectivamente. Sin embargo, también se observan 299 falsos positivos (FP) y 135 falsos negativos (FN). Mostrando una pequeña mejora en detección de verdaderos positivos y una disminución en la detección de verdaderos negativos, no obstante estas oscilaciones siguen siendo muy pequeñas, lo que genera una comparación casi por ponderaciones de cada modelo, donde en una visión macro del comportamiento, ambos modelos tienen un comportamiento similar en la detección, no obstante se diferencian en puntos específicos, como detección de verdaderos y falsos positivos, siendo en este caso, el presente modelo más preciso en este punto especifico, donde tiene mejor clasificación de verdaderos positivos, no obstante el modelo de 124 neuronas iniciales, clasifica o detecta de mejor manera los falsos positivos, por ende las diferencias de ambos modelos, se basan netamente en aspectos de precisión en comportamientos puntuales de un modelo vs otro.

|  |  |  |
| --- | --- | --- |
| **Verdaderos y falsos negativos** | **Modelo 124 neuronas iniciales** | **Modelo 64 neuronas iniciales** |
| **Verdaderos positivos** | **2386** | **2401** |
| **Falsos positivos** | **222** | **299** |

(Tabla Nro.24, tabla de comparación de rendimiento)

(Fuente elaboración propia)

**Gráfico de datos de entrenamiento redes neuronales con neuronas reducidas a la mitad**

**Gráfico, Gráfico de líneas

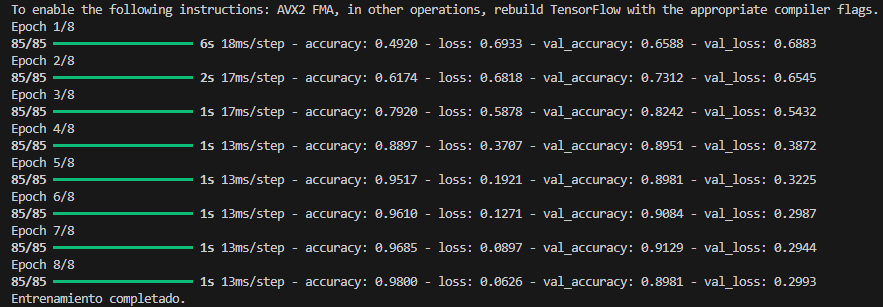
Descripción generada automáticamente**

(grafico Nro.18, grafico de desempeño en entrenamiento red neuronal profunda reducida)

(Fuente elaboración propia)

Continuando con los análisis de este modelo de red neuronal, en este apartado, se observa un comportamiento similar en los gráficos de entrenamiento de esta red, divisando el mismo comportamiento que en la red neuronal con 124 neuronales iniciales, correspondiente a que a mayor valores validación y entrenamiento, menor perdida se obtiene durante el entrenamiento.

Imagen demostrativa del entrenamiento

****

(Imagen Nro.08, imagen ilstrativa del entremaniento de la red neuronal profunda reducida)

(Fuente elaboracion propia)

En la imagen anterior, es posible divisar el respectivo comportamiento de la red neuronal, y su forma de entrenamiento y validacion en ocho epocas, divisando con claridad el comportamiento inversamente proporcional entre los datos de entrenamiento y validacion y los datos de perdida de cada epoca.

Finalmente al terminar su entrenamiento tenemos estos valores.

Texto

Descripción generada automáticamente

(Imagen Nro. 09, resultados de los datos de entrenamiento)

(Fuente elaboracion propia)

**Curva ROC**

**Gráfico, Gráfico de líneas

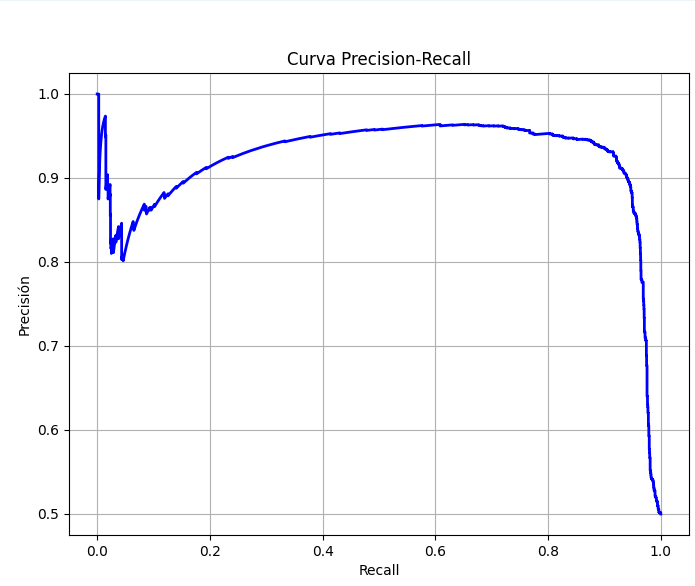
Descripción generada automáticamente**

(grafico Nro.19, curva ROC redes neuronal profunda reducida)

(Fuente elaboración propia)

Tal como se observó en el modelo d 124 redes neuronales en su capa inicial, tenemos un área bajo la curva de ROC de aproximadamente 95% lo que indica que el modelo tiene una excelente capacidad para distinguir entre clases positivas y negativas, clasificando correctamente los casos en el 95% de las veces. Esto refleja un rendimiento sobresaliente, con un buen equilibrio entre sensibilidad y especificidad, lo que lo hace muy confiable para la tarea de clasificación en cuestión.

**Curva Precision Recall.**

****

(grafico Nro.21, curva precisión recall, red neuronal profunda reducida)

(Fuente elaboración propia)

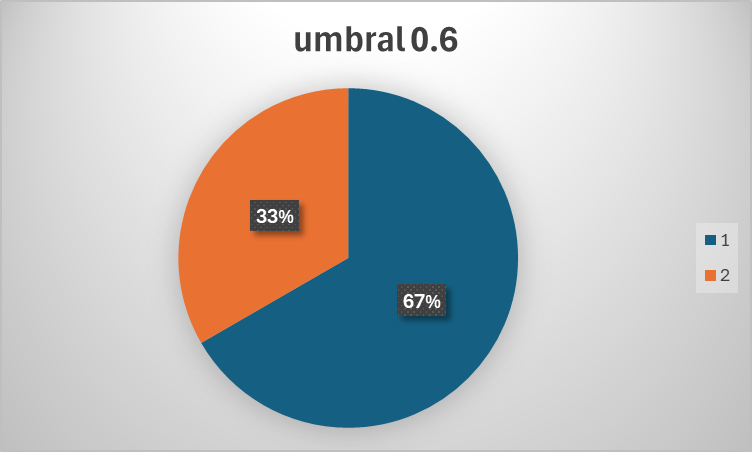
En el gráfico de precisión recall, nos muestra la capacidad del modelo que tiene para ser mejorado, tanto en el entrenamiento como en su precisión, por lo cual, en el presente gráfico, podemos observar el siguiente resultado del grafico precisión recall, siendo de un 0.95 o 95%, por ende, con este valor lo que muestra que el modelo necesita poco o nada de un refinamiento en su arquitectura, se puede intentar afinar a lo más a cinco por ciento que ese es el porcentaje faltante, no obstante muestra que el modelo actual no necesita mucho refinamiento en cuestión.

Según lo observado y como conclusión, cabe destacar que como se ha mencionado durante este apartado las diferencias de valores del comportamiento del modelo de red neuronal, son mínimos, se puede mencionar a simple vista o en función a la observación del desarrollo de entrenamiento y validación, que el modelo o que las redes neuronales profundas, son un modelo de machine learning potente para entrenar procesamiento de lenguaje natural y que son capaces de encontrar diferencias y categorizar correctamente las líneas de texto que mantiene una etiqueta que las califica como tal, además mencionar que al aumentar las redes neuronales en función de un exponente 2, la mejora en el funcionamiento se va observando de 0.1 punto porcentual aproximadamente, ya que el modelo funciona de manera robusta en ambas formas de arquitectura, por ende una mejora como tal, o un afinamiento, en este caso, no podría hacer una gran diferencia.

**Predicciones de las redes neuronales profundas con 64 neuronas iniciales.**

Seguidamente, se efectuarán dos mediciones en total, del comportamiento real del modelo, en el cual mediremos su capacidad predictora y clasificatoria antes líneas de textos que componen un correo electrónico, las que están conformadas por la dirección de correo electrónico, el encabezado del correo y el cuerpo de este, esto con la finalidad de cuantificar la capacidad de la maquina en tareas de clasificación.

**Resultados de predicciones con umbral de predicción de 0.4**



(grafico Nro.22, grafico de rendimiento en predicción)

(Fuente elaboración propia)

Para comenzar se realizaron, las pruebas de predicción con un umbral de precisión del 0.6%, el cual significa de cuán seguro debe estar el modelo antes de clasificar un texto como "Phishing".

Para este caso obtuvimos aproximadamente un 67% de precisión en los resultados, lo cual se contrasta con los datos de entrenamiento y validación, siendo un porcentaje menor, pero demostrando que el modelo, hace predicciones correctas en un 67%, lo cual en su función de predicción estaríamos ante un modelo de buen desempeño, el cual podría mejorar o poder mejorar en relación con predicciones.

Cabe hacer presente que la muestra a testear, fueron aproximadamente 51 líneas de texto, correspondientes a correos electrónicos con diferentes categorías.

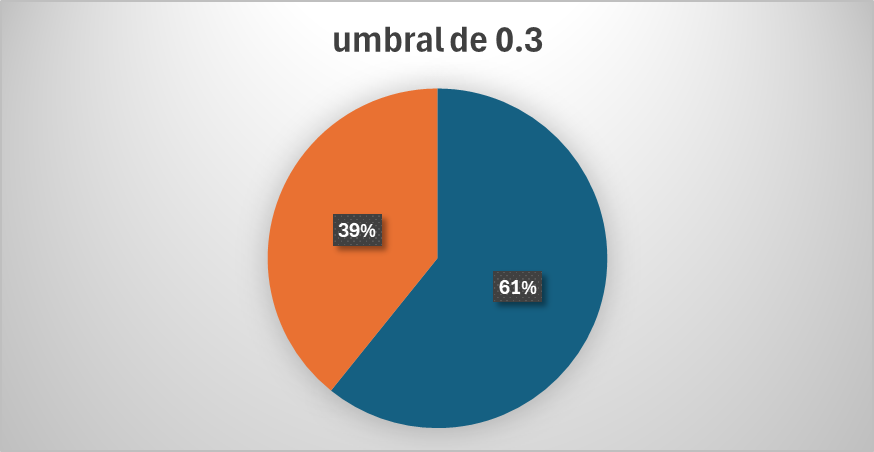
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

(tabla Nro.25, tabla dinámica red neuronal profunda, disminuida a la mitad)

(Fuente elaboración propia)

**Resultados de predicción con umbral de predicción del 0.3**

****

(grafico Nro.23, grafico de desempeño de predicciones)

(Fuente elaboración propia)

En este punto en particular se obtuvo valores menores que el umbral anterior, entendiendo que se disminuyó el umbral de predicción, en este caso en particular se disminuyó en tres puntos del umbral, bajando de 0.6 a 0.3, pero se observa un comportamiento menos eficaz en clasificaciones correctas, disminuyendo casi en 6% las respectivas clasificaciones de líneas de textos, por lo que el comportamiento del modelo estudiado da como resultado que a valores inferiores al 0.5, se obtienen clasificaciones menos exactas y a valores superiors, se obtienen datos más exactos.

**Tabla dinámica**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

(tabla Nro.26, tabla red neuronal profunda, disminuida a la mitad)

(Fuente elaboración propia)

**Conclusiones**.

Para comenzar con las conclusiones del presente estudio, podemos mencionar que en primera instancia al realizar las comparaciones de ambos modelos propuestos por Hugging Face referente a redes neuronales Bert, siendo estos los modelos **TinyBert** y **bert finetuned phishing**, los cuales presentan un desempeño moderado en la clasificación de correos electrónicos como phishing o no phishing, es dable resaltar este punto referente al desempeño moderado, debido a que ambos modelos necesitan, demasiados recursos de memoria RAM, por lo cual en el computador en que se hicieron las pruebas no estaba diseñado o preparado en cantidad de memoria para hacer pruebas optimas, además en relación a esta problemática durante el proceso, se tuvo que disminuir completamente el dataset original con el que se iba a entrenar los modelos, que en un inicio era un dataset de 8000 líneas de texto, compuesto con líneas de textos que conforman correos electrónicos, clasificados como “Phishing” y “No Phishing”, en razón a e esto se tuvo que reducir a 700 líneas de textos, correspondiente a un espectro de aproximadamente 8% del total, no obstante, las pruebas se realizaron, utilizando la plataforma de Google Colab, de la cual se obtuvieron datos de entrenamiento reales, que muestran el comportamiento de ambos modelos, quizás de una manera reducida, pero se puede concluir que, si bien ambos modelos se pudieron entrenar correctamente, tuvieron una precisión de **0.6345381526104418 (65% aproximado)**, para el modelo **Bert finetuned phishing** y una precisión **0.6010 (60% aproximado)**, para el modelo **TinyBert**, concluyendo en esta etapa, que el modelo **Bert finetuned phishing,** correspondiente a un fine tuning entrenado con el modelo **bert phishing custom**, tuvo mayor precisión en el entrenamiento, que el modelo de **tiny bert**, dado que el modelo mencionado precedentemente mide la proporción de verdaderos positivos entre todos los ejemplos que el modelo clasificó como positivos, por ende el modelo **bert fine tuned phishing** tiene una precisión ligeramente superior (0.6345) en comparación con **TinyBert** (0.6010). Esto indica que el modelo **bert finetuned phishing** es un poco más confiable al clasificar correctamente las instancias positivas o clasificar correctamente correos electrónicos de Phishing, no obstante, y en relación al accuracy (exactitud) el modelo **TinyBert** tiene una exactitud ligeramente superior (0.6477) en comparación con **bert finetuned phishing** (0.6240), lo que indica que, en general el modelo **TinyBert** clasifica más instancias correctamente en comparación con **bert finetuned** **phishing**, que quiere decir esto, que el modelo correspondiente a Tiny Bert, es más exacto clasificando ambas etiquetas respectivamente, que el modelo pre entrenado ya mencionado, cabe destacar que este desempeño es ligeramente mayor, siendo una diferencia de uno o dos puntos porcentuales.

Continuando el respectivo análisis, el modelo **TinyBERT** muestra un AUC de 0.7368, en tanto el modelo **bert finetuned phishing** tiene un AUC de 0.6910. Estos resultados sugieren que, aunque los modelos son efectivos en la clasificación, existe un margen significativo para mejorar, especialmente en la reducción de falsos negativos y el aumento de la precisión general, para esto sería una buena medida implementar ajustes en los hiper parámetros y analizar el comportamiento de entrenamiento de ambos modelos, mejorar el preprocesamiento y la tokenización, y utilizar técnicas de balanceo de clases más sofisticadas podrían contribuir a mejorar los resultados obtenidos, entendiendo esto, si se requieren realizar experimentos o entrenar el modelo con recursos limitados.

No obstante, a esto, y en función, de la primera etapa experimental de ambos modelos de Bert, podemos mencionar que, si bien no se pudo obtener valores de clasificación, ya que las clasificaciones se realizaron con parámetros aleatorios, lo que indica, que se debe hacer un afinamiento al proceso de clasificación, en razón al tokenizado de las líneas de texto a analizar y que el modelo entrenado se guarde y cargue correctamente en la función de clasificación, a pesar de este inconveniente, fue posible realizar el análisis antes mencionado y observar el comportamiento de ambos modelos, donde se puede concluir, que para la tarea de clasificación e identificación ambos modelos serian una buena opción para implementar no obstante esto variara, en razón de nuestro, enfoque, ya que si queremos usar un modelo preciso en clasificación de correos maliciosos, el modelo que se podría utilizar seria el modelo **bert finetuned phishing,** de lo contrario si se requiere mayor exactitud en la clasificación de correos electrónicos discriminando si son maliciosos o no, podríamos usar el modelo **Tiny Bert,** debido a su exactitud, en la diferenciación de correos usando ambas etiquetas.

Continuando con la segunda fase experimental, se procedió a realizar medidas de clasificación de correos electrónicos utilizando un clasificador Naive Bayes para la identificación y clasificación de correos electrónicos maliciosos, el cual ha mostrado resultados prometedores, aunque también revela áreas de mejora y afinamiento.

En primer lugar, las métricas de evaluación como precisión, recall y F1-Score indican un buen desempeño general del modelo. Con una precisión global del 86%, el clasificador Naive Bayes ha demostrado ser efectivo en la clasificación de correos electrónicos. La métrica de precisión sugiere que, de todas las instancias clasificadas como positivas (correos de phishing), el 86% eran verdaderamente positivas, lo que subraya la capacidad del modelo para evitar falsas alarmas o detectar falsos positivos.

Sin embargo, al observar el recall, notamos una diferencia en la capacidad del modelo para identificar las dos clases. El modelo muestra un recall del 88% para la clase "No Phishing" y un 83% para la clase "Phishing". Aunque esta diferencia del 5% puede parecer pequeña, evidencia que el modelo es ligeramente más efectivo en identificar correos no maliciosos que en detectar correos de phishing. Esto podría llevar a una mayor tasa de falsos negativos, lo que significa que algunos correos de phishing no fueron identificados correctamente, lo cual podría generar un desbalance en el análisis, no siendo capaz de recopilar o clasificar correctamente los correos maliciosos.

La matriz de confusión revela un rendimiento equilibrado, con una cantidad considerable de verdaderos positivos y verdaderos negativos, pero también con un número significativo de falsos negativos. Esto indica que el modelo tiene margen para mejorar en la identificación de correos electrónicos de phishing. Para lo cual se podría ajustar el modelo, debido al preprocesado o tokenizado de lenguaje natural en el modelo, a fin de que pudiese identificar más caracteres que puedan hacer clasificaciones más certeras.

La prueba adicional realizada con 55 líneas de texto para evaluar el porcentaje de precisión mostró un 64.29% de acierto en las predicciones. Aunque este porcentaje es menor que el de las métricas generales del modelo, sigue siendo un resultado aceptable. No obstante, el hecho de que el modelo haya cometido un mayor número de errores en esta prueba específica resalta la importancia de seguir ajustando y optimizando el modelo para mejorar su precisión en contextos reales.

Finalmente, la curva ROC y el área bajo la curva (AUC) de aproximadamente 0.90 indican que el modelo es robusto y tiene un alto grado de discriminación, lo que significa que tiene una baja probabilidad de clasificar correos de manera aleatoria. Este resultado refuerza la confianza en el modelo como una herramienta efectiva para la tarea propuesta.

En resumen, el clasificador Naive Bayes demuestra una capacidad sólida para la clasificación de correos electrónicos maliciosos, aunque hay áreas que pueden ser mejoradas. Con ajustes adicionales y optimización, este modelo podría convertirse en una herramienta aún más efectiva para detectar correos de phishing.

Finalmente, el experimento realizado sobre la red neuronal profunda muestra que, tanto en su configuración original como al reducir a la mitad la cantidad de neuronas en cada capa, el modelo demuestra un desempeño robusto y confiable en la clasificación de correos electrónicos como phishing o no phishing. La red neuronal profunda, en su configuración inicial, logró una alta precisión y recall, reflejando su capacidad para clasificar de manera efectiva las instancias en el dataset. Los valores de precisión cercanos al 98% durante el entrenamiento y 92% en la validación, junto con un área bajo la curva ROC de 0.95, indican que el modelo es altamente eficaz en la identificación de casos positivos y negativos, lo que lo hace adecuado para su aplicación en la detección de phishing y/o clasificación de correos para ambas etiquetas.

La reducción en la cantidad de neuronas no tuvo un impacto negativo significativo en el rendimiento del modelo, lo que sugiere que la red es lo suficientemente robusta para mantener una alta precisión y recall incluso con una arquitectura más sencilla o minimizando sus redes neuronales. Esto también implica que el modelo tiene una buena capacidad de generalización y no depende críticamente de una alta complejidad para realizar predicciones precisas, por ende, el modelo funciona tanto con composiciones complejas, como con composiciones sencillas, sin tanto impacto en sus resultados de clasificación.

En cuanto a las pruebas de predicción con diferentes umbrales de decisión, se observó que un umbral más bajo (0.3) mejora ligeramente la precisión en las clasificaciones, lo que podría ser útil en escenarios donde es más importante reducir los falsos negativos que los falsos positivos, este resultado, se dio con la composición completa de las redes o con el modelo inicial, no obstante que al modelo reducido a la mitad en sus redes neuronales, tuvo mayor precisión en tareas de predicción en un umbral más alto (0.6), lo que indica que en este modelo se evita obtener falsos positivos o se comporta más exacto, aumentando el umbral de detección de falsos positivos. Sin embargo, la diferencia en rendimiento entre los distintos umbrales fue moderada, lo que resalta la estabilidad del modelo frente a variaciones en los parámetros de decisión.

En resumen, el modelo de red neuronal profunda implementado es altamente efectivo para la clasificación de correos electrónicos y tiene el potencial de ser implementado en entornos reales para la detección de phishing, manteniendo un buen equilibrio entre precisión y capacidad de generalización. La reducción de la complejidad del modelo no afecta significativamente su rendimiento, lo que sugiere que puede ser optimizado aún más para mejorar la eficiencia sin sacrificar la calidad de las predicciones.

**Referencias.**

Dunstan, J. D., Hogan, A., & Sipiran Mendoza, I. (2023). Light and fast language models for Spanish through compression techniques, Tesis, Universidad de Chile, paginas 24-50, URL https://repositorio.uchile.cl/bitstream/handle/2250/196742/Light-and-fast-language-models-for-spanish-through-compression-techniques.pdf?sequence=1

Jiao, X. J., Yin, Y. Y., Shang, L. S., Jiang, X. J., Chen, X. C., Li, L. L., Wang, F. W., & Liu, Q. L. (2020). *TinyBERT: Distilling BERT for Natural language understanding*, Key Laboratory of Information Storage System, Huazhong University of Science and Technology, paginas 4-7, URL. https://C:/Users/walte/Downloads/PAPER%20TINY%20BERT%20IMPORTANTE.pdf

Lv, X. L., Zhu, S. Z., Chan, S. C., He, D. H., Kwang, K. K., & Choo, R. C. (2024). *A method for detecting phishing websites based on Tiny-Bert stacking,*, IEEE INTERNET OF THINGS JOURNAL, pagina 2-6, URL  https://ieeexplore-ieee-org.recursosbiblioteca.unab.cl/stamp/stamp.jsp?tp=&arnumber=10172235&tag=1

Bravo-Márquez, F. B.-M., & Dunstan, J. D. (2021). Procesamiento de lenguaje natural: Dónde estamos y qué estamos haciendo. *revistasdex.udechile.cl*, <http://revistasdex.udechile.cl>., 45-46.

Ryan T. Wright, Steven L. Johnson, and Brent Kitchens, (2023), Phishing susceptibility in context:amultilevel information processing perspective on deception detection, pagina804-806, URL <https://research-ebsco-com.recursosbiblioteca.unab.cl/c/blztel/viewer/pdf/hbvhmgdtqb>,

Zhang, H. Z. (s. f.). The optimality of naive bayes. *American Association For Artificial Intelligence*, pagina 1-2, URL <https://cdn.aaai.org/FLAIRS/2004/Flairs04>097.pdf. <https://cdn.aaai.org/FLAIRS/2004/Flairs04-097.pdf>

Huawei-Noah/TinyBERT\_General\_4L\_312D. (s. f.). <https://huggingface.com>, obtenido en mayo 2024, de https://huggingface.co/huawei-noah/TinyBERT\_General\_4L\_312D

Ariel Rodríguez-Jiménez, E. A.-M.-B.-P, 07 de abril de 2020, revistas.tec.ac.cr, URL https://revistas.tec.ac.cr/index.php/tec\_marcha/article/view/5171/4823

deland, s. (s.f.). Obtenido de https://la.mathworks.com/discovery/roc-curve.html#:~:text=Las%20curvas%20ROC%20(caracter%C3%ADstica%20operativa,dos%20clases%20de%20salidas%20distintas.

¿Qué es el procesamiento del Lenguaje Natural (NLP)? (2d. C.), <https://www.ibm.com>, Recuperado 1 de mayo de 2024, URL https://www.ibm.com/es-es/topics/natural-language-processing

Clasificacion de ataques de phishing. (s. f.). Kaggle. Recuperado 5 de abril de 2024, de <https://www.kaggle.com/>.

Chaparro, T. I.. Análisis de sentimientos en redes sociales orientado a la percepcion de la calidad de servicios de internet, redes moviles, tv cable y electricidad. *Análisis de sentimientos en redes sociales orientado a la percepcion de la calidad de servicios de internet, redes moviles, tv cable y electricidad*, mayo de 2019, Santiago, Chile, URL <https://repositorio.unab.cl/server/api/core/bitstreams/d86c47e4-c953-4b9e-9c47-2b8d51f26c04/content>