Universidad de Lima

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

COMPARACIÓN DE ALGORITMOS DE MACHINE LEARNING PARA CLASIFICAR OPINIONES DE USUARIOS ACERCA DE PUBLICACIONES HECHAS POR BANCOS DE PERÚ SOBRE PHISHING

Tesis para optar el Título Profesional de Ingeniero de Sistemas

Adrian Alonso Temoche Salinas

Código 20181875

Asesor

Oscar Efrain Ramos Ponce

Lima – Perú

Junio de 2024

Comparación de algoritmos de machine learning para clasificar opiniones de usuarios acerca de publicaciones hechas por bancos de Perú sobre phishing

Adrian Alonso Temoche Salinas
20181875@aloe.ulima.edu.pe
Universidad de Lima

Resumen: A nivel mundial el *phishing* en los últimos años se ha incrementado y el Perú no fue una excepción. Debido a la pandemia del Covid-19 se registró un aumento considerable de este tipo de ciberataques llamado *phishing*; entre los motivos de su crecimiento se encuentra la cuarentena. Este tipo de ataques suele afectar de manera negativa a los usuarios de internet y mucho más a las empresas, siendo uno de los ataques más frecuentes el *phishing*. Por ejemplo, en las noticias como en redes sociales se puede observar cómo los bancos o plataformas de pago advierten a sus usuarios sobre estos ataques para que estén atentos, al igual que se puede observar el malestar de los usuarios cuando publican o comentan en las redes sociales que han sido víctima de dichos ataques. Por este motivo, este trabajo busca identificar de manera correcta las opiniones de las personas acerca de publicaciones hechas por bancos de Perú sobre *phishing*, clasificando su opinión como positiva o negativa. Se utiliza procesamiento de lenguaje natural (*PLN*) y análisis de sentimiento para procesar el texto, con los algoritmos de *Naive Bayes*, *Random Forest* y *Support Vector Machine* (*SVM*) en el caso de *machine learning* y con el transformador *BERT* en el caso de *deep learning*, con el fin de conocer cuál de los cuatro algoritmos tiene mejores resultados. La metodología usada consiste en 5 pasos desde la búsqueda y creación del *dataset*, pasando por la etapa de preprocesamiento, entrenamiento, pruebas y finalizando en la evaluación de los resultados. De los cuatro algoritmos, *BERT* obtuvo el mejor resultado con una exactitud de 86.90% seguido de *Naive Bayes* que fue el que obtuvo un mejor resultado en la validación cruzada con 79.62% de exactitud y en la etapa de prueba con 86% de exactitud entre los tres algoritmos de *machine learning*. Además, los resultados obtenidos muestran que los tres algoritmos de *machine learning* tienen un desempeño similar.

Palabras Clave: aprendizaje automático, *PLN*, análisis de sentimiento, clasificación, *phishing*.

Abstract: Phishing in recent years has increased and this year was no exception. Due to the pandemic, there was a considerable increase in this type of cyber-attacks; among the reasons for its growth is the quarantine. This type of attack usually has a negative impact on Internet users and even more so on companies, with phishing being one of the most frequent attacks. For example, in the news and on social networks it can be observed how banks or payment platforms warn their users about these attacks so that they can be alert, as can be observed the discomfort of users when they post or comment on social media that they have been victims of such attacks. For this reason, this work seeks to correctly identify people's opinions about publications made by banks in Peru about phishing, classifying their opinion as positive or negative. Natural language processing (PLN) and sentiment analysis are used to process the text, with the Naive Bayes, Random Forest and Support Vector Machine (SVM) algorithms in the case of machine learning and with the BERT transformer in the case of deep learning, in order to know which of the four algorithms has better results. The methodology used consists of 5 steps from the search and creation of the dataset, through the preprocessing stage, training, testing, and ending with the evaluation of the results. Of the four algorithms, BERT obtained the best result with an accuracy of 86.90% followed by Naive Bayes which obtained the best result in the cross validation with 79.62% accuracy and in the testing stage with 86% accuracy among the three machine learning algorithms. In addition, the results obtained show that the three machine learning algorithms have similar performance.

Keywords: machine learning, PLN, sentiment analysis, classification, phishing.

1. INTRODUCCIÓN

En la actualidad las personas usan el internet como parte de su vida diaria, ya sea para realizar compras, pagos, entretenimiento, entre otras actividades. Según GWI (*GlobalWebIndex*), la población promedio mundial pasa 6 horas y 40 minutos navegando por internet (Thompson A, 2024). Esto da a entender, que la mayoría de las personas está una parte del día conectado a la red, expuesto a todo lo que hay dentro de ella, incluyendo los peligros que hay dentro del internet. Uno de esos peligros es el *phishing*, que constituye una amenaza la cual ha venido creciendo en los últimos años a mayor velocidad, entre los motivos la pandemia donde la cantidad de horas que una persona pasa frente a la computadora o su celular aumentó considerablemente a causa de la cuarentena. De acuerdo con Ipsos (2021), en el 2021 alrededor de 6 millones de personas eran usuarios de la banca digital en el Perú y la mayoría usaba más la banca móvil. Otro dato, es que de acuerdo con Ipsos (2022) revelo que 7 de cada diez personas que están bancarizadas utilizan la banca por internet, billeteras digitales o banca móvil.

El *phishing* tiene como objetivo el robo de información de datos privados con el fin de hacer uso de ellos para tener algún beneficio. Dentro de los problemas relacionados con el *phishing*, acorde a Shahrivari et al., (2020), se tiene la creación de sitios web simulados que fingen ser la página original, para robar información como el ID, nombre de usuario, contraseña de individuos y organizaciones. También, el envío de correos o spam masivo como menciona Baykara y Gürel (2018), con el objetivo que el usuario de clic en algún enlace o publicidad dentro del correo electrónico a fin de realizar varias funciones, incluida la captura de las credenciales de inicio de sesión o la información de la cuenta de la víctima. Estos correos electrónicos dañan a las víctimas debido a la pérdida de dinero y el robo de identidad. Además, en la literatura podemos notar un crecimiento mayor a medida con el pasar de los años. Por ejemplo, según Shahrivari et al., (2020) de acuerdo con el *Anti-Phishing Working Group (APWG)*, los ataques relacionados con *phishing* se han incrementado considerablemente en años recientes, de los 162 155 ataques detectados en el último cuarto de 2019, pasamos a la cantidad de 165 772 casos relacionados a *phishing* en el primer cuarto de 2020. Al buscar datos más actuales se encontró con el reporte de *APWG* del primer cuarto del 2022, los datos muestran que, durante el primer trimestre de 2022, el APWG observó un total de 1 025 968 ataques de phishing. Este trimestre fue el peor trimestre de phishing de la historia de APWG, y la primera vez que el total del trimestre superó el millón. El récord anterior fue de 888 585 ataques, observado en el cuarto trimestre de 2021. La cantidad de ataques de phishing se ha más que triplicado desde principios de 2020, cuando el APWG observó entre 68 000 y 94 000 ataques por mes. Este trabajo busca realizar la clasificación de opiniones de usuarios acerca de publicaciones de bancos de Perú sobre *phishing* utilizando procesamiento de lenguaje natural con el objetivo de hallar qué tan efectiva es está técnica para la clasificación de este tipo.

Otros trabajos relacionados al *phishing* abordan el tema desde el punto de vista de detección de este tipo de ciberataque. Por ejemplo, Abedin et al., (2020) detectan páginas web de *phishing* en base a las características que identifiquen a la página web, como los símbolos raros en la URL, la longitud de la URL, si tiene una IP en vez de un nombre de dominio, etc, utilizando algoritmos de *machine learning* como *KNN*, regresión logística y *Random Forest*. En el trabajo de Alkawaz et al., (2020) se detecta a las páginas web *phishing* en base a una lista negra que almacenará las URLs que sean maliciosas y cuando el usuario intente ingresar a una de ellas le llegará un correo indicando el intentó de acceso a una web *phishing*. En el caso de análisis de sentimiento Baj-Rogowska (2017) analiza el sentimiento de opinión, es decir, clasifica oraciones con tonos de humor neutros, positivos y negativos, utilizando las opiniones de los usuarios de Facebook sobre Uber. El trabajo Baid et al., (2017) analiza las reseñas de películas utilizando diferentes métodos, como *Naive Bayes*, *K-Nearest Neighbors* y *Random Forests* para saber su polaridad si son reseñas positivas o no. Para trabajos relacionados con *BERT* se tiene el de Sousa et al., (2019) proponen el uso de *BERT* para el análisis de sentimiento en artículos de noticias financieras e informar las decisiones del mercado de valores con el objetivo de indicar la tendencia del índice Dow Jones, si es positiva, neutral o negativa.

Sin embargo, no existen trabajos relacionados sobre las opiniones de los usuarios acerca del phishing en los bancos del Perú, por lo que este trabajo utiliza el tema de phishing y lo combina con el análisis de sentimiento con el objetivo de clasificar las opiniones de los usuarios acerca de las publicaciones hechas por los bancos del Perú sobre este tipo de ciberataque. A diferencia de los trabajos anteriores, los datos recopilados provienen de las publicaciones hechas por los bancos del Perú y están en español, algo que no es muy común ya que la mayoría de este tipo de trabajos sobre el análisis de sentimiento y el phishing suelen ser en inglés. Con lo mencionado anteriormente, este trabajo ayudaría a que el banco conozca la opinión de las personas ante este tipo de publicaciones, es decir, si la mayoría de los comentarios en este tipo de publicaciones que advierten sobre el phishing son tomados manera positiva por los usuarios o no. Esto ayudaría al banco a tomar con mayor consideración el sentir de los usuarios sobre este tema, lo cual podrían usar para futuras publicaciones sobre este tipo de robos y advertir de mejor manera a sus usuarios en base a las opiniones de los comentarios clasificados, logrando así una mejor recepción en sus canales de comunicación.

El artículo se divide en siete secciones, en la sección 1 se presenta la introducción dónde se da un contexto de lo que es el phishing, sus peligros y el motivo del artículo; en la sección 2, el estado del arte se cubrirán conceptos más relacionados con el tema propuesto y ejemplos de trabajos usando *PLN*, análisis de sentimiento, el transformador *BERT*; en la sección 3, se encuentra los antecedentes que cuenta con conceptos aún relacionados con el tema de investigación pero de manera más general; en la sección 4 se encuentra la metodología a utilizar y como se planteará la implementación; en la sección 5 se encontrarán los resultados obtenidos luego de haber realizado la implementación de la idea propuesta; en la sección 6 se discutirá los resultados obtenidos y se analizaran para saber si se obtuvo los resultados esperados o no; al final se encuentran las conclusiones y trabajos futuros.

2. ESTADO DEL ARTE

2.1 Procesamiento de lenguaje natural con análisis de sentimiento en seguridad

Esta sección presenta algunos usos que se le ha dado al procesamiento de lenguaje natural con análisis de sentimiento y cómo ha sido aprovechada. Gupta et al., (2016) proponen un estudio exploratorio que examine las actitudes de

ciberseguridad y el comportamiento a lo largo del tiempo en la plataforma de Twitter. Para ello, utilizaron análisis de sentimiento para conocer el comportamiento y las actitudes. Planean mostrar la relación entre actitudes y comportamientos en materia de ciberseguridad y cómo los comportamientos pueden estar condicionados por las actitudes. Se planteó la recolección de la data en dos periodos de tiempo. La primera fecha fue del 1 de marzo del 2013 hasta el 31 de agosto del 2013, y la segunda fecha fue del 1 de marzo del 2015 hasta el 31 de agosto del 2015.

Hao y Dai, (2016) mencionan las brechas de seguridad que varias compañías a lo largo del tiempo han tenido y cómo estas con el pasar de los años han aumentado. Por ejemplo: Target tuvo su brecha de seguridad en 2014 con 40 millones de tarjetas de débito y crédito robadas, además de 70 millones de registros personales tiempo después; Rose con su brecha de seguridad del registro de pago de sus clientes con 127 casos en 2009 a 217 en 2013 entre otras tiendas como Home Depot, Michaels, etc. El conocimiento que se tiene de las personas sobre este tipo de incidentes es muy poco o desconocido y mucho más cuando se trata de saber la opinión de los clientes, pues varias veces no son reportados estos incidentes para no causar algún escándalo mayor. La propuesta de investigación de los autores Hao y Dai es llenar ese vacío creando un marco para el estudio y vigilancia de los medios sociales sobre las violaciones de seguridad, para poder así obtener una imagen más clara acerca de las preocupaciones y actitudes del público ante este tipo de incidentes.

Sharma y Jain, (2020) realizaron una recopilación de varios enfoques y técnicas del análisis de sentimiento para seguridad y las redes sociales. Para la seguridad se menciona cómo el análisis de sentimiento se ha utilizado en este ámbito desde la detección de engaños, anomalías, gestión de riesgo, etc. También, se menciona cómo se ha usado esta técnica para los problemas de seguridad relacionados a la procedencia de los datos, la desconfianza, la seguridad del comercio electrónico, violaciones de seguridad de los consumidores, etc. Con el incremento del uso y popularidad de las redes sociales han aparecido otros temas a considerar, como la gran cantidad de volúmenes de datos y la preocupación de esos datos que las redes sociales proporcionan a través de sus usuarios. Por lo que el análisis de sentimiento desempeña un papel fundamental en la seguridad y el análisis de las redes sociales, por ejemplo: en la detección de spam, individuos maliciosos, *spammers*, estafadores en línea, etc.

2.2 Clasificación de opiniones usando análisis de sentimiento

En esta sección se busca mostrar trabajos relacionados a análisis de sentimiento y cómo se ha utilizado para la clasificación de comentarios de diferente manera. Othman et al., (2015) indican que la información expresada en el texto es un hecho o una opinión, y siempre que tenemos que tomar una decisión, tendemos a conocer la opinión de los demás, que es uno de los factores más influyentes en nuestra toma de decisiones. Tradicionalmente, las personas reciben información de amigos y familiares, mientras que las organizaciones utilizan encuestas, grupos focales, encuestas y consultores. En la actualidad gracias a la web hay varios medios por donde las personas son capaces de dar su opinión y expresar su sentir, con esto en mente se ve en la necesidad de contar con un sistema capaz de extraer la información y saber de manera automática que es lo que los usuarios expresan en sus comentarios. En este trabajo, proponen un método para clasificar las opiniones que consideran un documento o conjunto de documentos para un objeto. El sistema encontró diferentes tipos de declaraciones significativas, incluidas actitudes, superlativos y no actitudes. El sistema encuentra diferentes tipos de oraciones de opinión, incluidas opiniones, comparativos, superlativos y no opiniones. La tarea principal de este sistema es procesar el documento de entrada para etiquetar cada palabra del documento. Luego, las palabras en la declaración se resaltan en cada oración y se ponderan las palabras resaltadas. Finalmente, el sistema determina el tipo de significado para cada oración o grupo de oraciones según el valor de opinión de las palabras resaltadas.

Otro caso es el propuesto por Bakliwal et al., (2011) el análisis de emociones y las calificaciones juegan un papel importante en la predicción de lo que las personas piensan sobre productos, lugares y más. En este artículo, utilizando técnicas básicas de PLN como NGram, NGram etiquetado con POS, clasificamos las reseñas de películas y productos en general en dos extremos: positivas y negativas. Proponemos un modelo de enfoque para determinar si una revisión es positiva o negativa, probamos y usamos varios algoritmos de aprendizaje automático: *Naive Baye*s (*NB*), *Multilayer Perceptron* (MLP), *Support Vector Machine* (*SVM*) para la evaluación comparativa. Los investigadores desarrollaron un nuevo método de puntuación probando con dos métodos: el primero es el emparejamiento simple de NGram (N=1/2/3): se utilizan los unigramas, bigramas y trigramas de retroalimentación se utilizan para asignar una puntuación a la retroalimentación, clasificándose, así como positiva o negativa. Correspondencia de Ngrams con etiqueta posterior: los Ngram se construyen utilizando información de comentarios, trigramas, bigramas y unigramas etiquetados con POS que combinan solo adjetivos (JJ) y adverbios (RB). En función de la nota final de la revisión, se clasifica en positiva o negativa. También utilizamos algoritmos de aprendizaje automático *Naive Bayes* (*NB*) *Multilayer Perceptron* (*MLP*) y *Support Vector Machine* (*SVM*) para estudiar el rendimiento de nuestro método. El método se aplica a dos conjuntos de datos de reseñas de productos y películas.

Behdenna et al., (2018) mencionan que el análisis de sentimientos ha tenido mayor protagonismo en el campo de investigación de minería de textos. Su finalidad es extraer de los textos las opiniones, los sentimientos y la subjetividad de los usuarios. Debido a la aparición de la Web 2.0, los usuarios pueden compartir sus opiniones y sentimientos sobre

varios temas a través de nuevos métodos interactivos, y los usuarios ya no son sólo receptores pasivos de información. Dado que esta información es importante en diferentes ámbitos (político, comercial o personal), sería interesante procesar estas notificaciones de forma automática. Mencionan que el análisis de sentimientos tiene una variedad de aplicaciones, desde determinar las opiniones de los clientes sobre productos y servicios hasta las reacciones de los votantes a los anuncios políticos. Otras áreas de aplicación donde el análisis de sentimientos puede ser muy útil son Inteligencia empresarial: donde juega un papel muy importante en muchas aplicaciones, como la calificación crediticia o la reputación de la empresa. Sería útil categorizar cada reseña de acuerdo con un aspecto del negocio o transacción que describe, como la calidad del producto, el pedido o la integridad; Sistema de recomendaciones: ayuda a comprender cómo se sienten las personas acerca de un producto o tema. En base a esto, el cliente toma una decisión y predice el impacto del tema en otras áreas; Resumen de las reseñas: permite obtener opiniones sobre entidades específicas. Esto dará la calificación general del tema ya que es posible que los clientes no puedan decidir sobre un producto después de leer todas las reseñas; Inteligencia gubernamental: permite obtener opiniones sobre las políticas y decisiones del gobierno para inferir la probable reacción del público a ciertas aplicaciones de políticas.

Baj-Rogowska (2017) analiza el sentimiento de opinión, es decir, las clasifica en oraciones con tonos de humor neutros, positivos y negativos. Los datos utilizados para el análisis fueron las opiniones de los usuarios de Facebook sobre Uber recopiladas entre julio de 2016 y julio de 2017. El objetivo principal de la encuesta fue obtener información sobre la percepción de Uber durante un período de 13 meses, se utilizó Facebook debido a que se debe considerar como una enorme base de datos de opiniones que se puede aprovechar. Menciona que el término minería de opinión, también adopta el nombre de análisis de sentimientos utiliza las soluciones desarrolladas en el Procesamiento del Lenguaje Natural (PLN) donde ambos términos se usan indistintamente. Se indica que el análisis de sentimiento es un proceso complejo el cual consiste en 5 etapas: la recolección de datos: incluye la obtención de datos apropiados de redes sociales seleccionadas (por ejemplo, Facebook); preparación de texto: en esta etapa se elimina la información irrelevante y se eliminan los datos no textuales; detección de sentimiento: se analiza el texto refinado en función de las características de las palabras. Oraciones que contienen opiniones y opiniones subjetivas Opiniones y opiniones subjetivas del usuario, mientras que las objetivas son rechazadas; clasificación de sentimiento: las opiniones y opiniones subjetivas de los usuarios se clasifican como positivas, negativas, buenas, malas o utilizando otros sistemas de clasificación de opiniones; presentación del resultado: se presente en forma de gráfico de acuerdo a como se ajuste mejor a la información obtenida y que mejor sirva para explicar los datos representados. Si se tiene varios datos de diferentes fechas se puede usar un gráfico de tiempo que muestre el sentimiento de los usuarios por fecha o en otras medidas.

Existen otras formas en que se ha usado el análisis de sentimiento, por ejemplo, Nasrin et al., (2020) la tecnología ha facilitado la vida en casi todos los sentidos gracias a que las compras en línea son una de las bendiciones de la tecnología. Pero el fraude en línea dificulta las cosas debido a que está presente en este tipo de medios como en las redes sociales. Por ejemplo, aunque Facebook es una red social, muchas personas la usan como una plataforma de negocios en línea, pero debido a que hay estafadores en esta red social otros buenos empresarios están sufriendo porque la gente está perdiendo la fe en el mercado en línea. Por lo tanto, es importante averiguar qué páginas de Facebook son buenas y cuáles son una estafa. Se menciona que en el 2017 alrededor de 16.7 millones de personas se volvieron víctimas de fraude en línea. La metodología utilizada fue la siguiente: primero, se recopiló los datos de un grupo de Facebook para determinar si se trataba de una página comercial fraudulenta. Los datos recabados son publicaciones públicas y sus respectivas reseñas. En la parte de preprocesamiento, los datos se limpian para eliminar todo el texto innecesario, como enlaces, comentarios de una sola palabra, etc. Los datos preprocesados están listos para el análisis de sentimiento. Esto se puede hacer usando diferentes algoritmos o incluso usando un análisis de vocabulario llamado bolsa de palabras. Finalmente, luego del proceso de análisis de sentimiento, el sistema decidirá si enviarlo para detección de fraude. El resultado final se presentará en base al informe de detección de fraude. Este trabajo presenta un modelo para la detección de fraude utilizando análisis de sentimientos. Este proceso se realizará mediante análisis basado en diccionario. Si hay algún tipo de fraude en la página de Facebook de una empresa, puede ser detectado por las reseñas de los clientes. Dado que el bienestar del cliente depende de la satisfacción, el cliente dará una buena opinión si el producto es bueno. Por lo tanto, este sentimiento se puede utilizar para detectar fraudes en las páginas comerciales de Facebook. El análisis de opinión se realiza solo cuando los datos recopilados de Facebook están listos para analizar. En este proyecto, el sentimiento puede determinarse utilizando el algoritmo *Naive Bayes* y usarse para pruebas de sentimiento de dos estados. Ambos algoritmos generan gráficos circulares polares para emociones positivas y negativas. Se considera una buena página si y sólo si ambos gráficos tienen al menos un 65% de sentimiento positivo. El análisis de opinión que utiliza el algoritmo *Naive Bayes* utilizará su propia biblioteca y modelo analítico, y el resultado final se mostrará en forma de gráfico circular. Sin embargo, si el análisis se basa en un léxico, se debe declarar un algoritmo definido por el usuario para el análisis.

2.3 Aplicaciones de Procesamiento de Lenguaje Natural

Wongkar y Angdresey, (2019) mencionan que hay varios procesos que se llevan a cabo en este procesamiento de texto, el estudio se realizó sobre la opinión de las personas acerca de los candidatos presidenciales de indonesia del

año 2019. En primer lugar, se recopiló datos, en este estudio utilizamos tweets de datos que se recopilan de las redes sociales de Twitter mediante el uso de un rastreador. Además, analizan los tweets que se obtienen describiéndolos textualmente. A continuación, realizan el proceso de tokenización que consiste en limpiar el tuit y seleccionar las palabras significativas. Los algoritmos usados en esta investigación fueron: *Naïve Bayes*, *SVM* y *KNN*. Respecto al *dataset* no proporciona el dataset original, pero se menciona cómo se obtuvo. En el proceso de recopilación de datos, utilizaron tweets de datos que se obtienen utilizando datos del rastreador de Twitter tomados de enero a mayo de 2019. Consiste en un conjunto de comentarios etiquetados con positivo o negativo según corresponda al texto. Los resultados fueron: Naïve Bayes obtuvo 80,9 % ≈ 80,1 %, SVM obtuvo 63,99 %, KNN obtuvo 75,58 %.

Wenando et al., (2020) empieza con el *dataset*, luego sigue con el preprocesamiento de los datos el cual tiene el objetivo de eliminar los datos erróneos de los tuits y seleccionar las características en forma de palabra o términos; se procede a la clasificación que se realiza en base al peso de la palabra que definirá si es a favor o en contra de las elecciones presidenciales y con ello se finaliza con el resultado. Los algoritmos utilizados en la investigación fueron: *Naive Bayes*, *SMO*, *Logistic Regression*, *KNN*, *Decision Tree*. Respecto al *dataset* usado en el trabajo, se utilizó un conjunto de datos de las elecciones presidenciales de Indonesia del año 2019. Para la creación del *dataset*, recolectaron los datos a utilizar, en este caso tuits.

Respecto a los resultados obtenidos en la investigación, se divide en tres apartados, exactitud, precisión y *recall* (verdaderos positivos). Los resultados obtenidos en la investigación se presentan en tres categorías: exactitud, precisión y recall. En el caso del algoritmo Naive Bayes, se obtuvo una exactitud, precisión y recall del 80.20%. Por otro lado, el algoritmo SMO mostró un rendimiento ligeramente superior, con una exactitud, precisión y recall del 82.70%. En cuanto a la Regresión Logística, los resultados fueron de 77.70% para la exactitud, 78.60% para la precisión y 77.10% para el recall. El algoritmo KNN obtuvo una exactitud del 78.80%, una precisión del 78.60% y un recall del 79.10%. Finalmente, el Árbol de Decisión alcanzó una exactitud del 77.30%, una precisión del 77.00% y un recall del 78.30%.

Los cuales el algoritmo ganador en cada una de esas características fue el *SMO*. Sin embargo, el algoritmo de *Naive Bayes* en la columna de exactitud, precisión y recall solo está detrás de *SMO* por 2.5 % lo cual indica que también es un algoritmo prometedor. Respecto a por qué ganó el *SMO*, los autores no dan una razón en particular. Se podría decir que puede ser el tipo de datos recolectados como también la cantidad de ellos utilizados o el método utilizado, en la investigación menciona que usaron el *K-Fold Cross* con una validación cruzada de 10 veces.

Baid et al., (2017) las redes sociales y otras plataformas en línea contienen grandes cantidades de datos en forma de tweets, blogs, actualizaciones de estado, noticias y más. En este artículo, se analiza las reseñas de películas utilizando diferentes métodos, como *Naive Bayes*, *K-Nearest Neighbors* y *Random Forests* para saber su polaridad si son reseñas positivas o no. El experimento empezó con el análisis y resumen de cada artículo para seleccionar los algoritmos más usados encontrados, también se recolectó alrededor de 2000 reseñas de del portal de IMDb, la información fue almacenada en un archivo tipo TXT que se tenía dos carpetas una para los comentarios negativos y otra con los comentarios positivos donde luego se importó al software WEKA, es un sistema desarrollado en la Universidad de Waikato en Nueva Zelanda, que proveía una implementación de algoritmos de machine learning y procesamiento de texto. Los algoritmos escogidos fueron los 3 algoritmos mencionados al inicio. Luego de ser procesados por el software WEKA se obtuvieron los siguientes resultados en precisión: *Naive Bayes* con 81.4%, *KNN* con 55.30% y *Random Forest* con 78.65%. Determinar la polaridad de opiniones puede ayudar en varias áreas. Los sistemas inteligentes se pueden diseñar para proporcionar a los usuarios reseñas completas de películas, productos, servicios y más. Los usuarios no necesitan revisar individualmente y pueden tomar decisiones directamente en función de los resultados del sistema inteligente. Entre sus aplicaciones están la de compra de mercancía o servicio, donde las personas pueden comparar las opiniones de los usuarios acerca de un producto o servicio; mejorar la calidad del producto o servicio, donde los productores pueden recolectar las opiniones de sus consumidores para observar donde están sus falencias; en un sistema de recomendación, analizando las opiniones de las personas acerca de sus preferencia es intereses; detección de llama, donde los grupos de noticias, los blogs y las redes sociales se pueden monitorear fácilmente mediante el análisis de sentimientos. Esta tecnología detecta palabras soberbias, prepotentes y exageradas que se utilizan en tuits, noticias o foros y blogs en Internet.

Nayak y Natarajan, (2016) el análisis de sentimiento también conocido por *PLN* por sus siglas en inglés se utiliza para varios problemas, como determinar el punto de vista del autor, determinar la sensación general de un documento, etc. El objetivo básico es clasificar la polaridad de un documento, oración o función dada. Realizamos un análisis de opinión en un conjunto de datos de *feeds* de Twitter de revisión de películas estándar para explorar tres clasificadores de aprendizaje supervisado bien conocidos: *Naive Bayes*, *Support Vector Machines* (*SVM*) y *Random Forests* para determinar el más preciso que tenga positivos y negativos. Se realizó un procesamiento de los datos incluyendo la consistencia de mayúsculas y minúsculas, la obtención de datos utilizando la derivación de Porter, la eliminación de símbolos especiales que se encuentran comúnmente en los tweets como "@, #", etc., la eliminación de espacios, espacios adicionales y palabras redundantes, y mucho más. También usamos el término estadística de frecuencia de documento inversa (tf-idf) porque es un indicador poderoso de la importancia de una palabra determinada en el

conjunto de datos. También ayuda a eliminar palabras finales (palabras que no afectan la polaridad del texto dado) del conjunto de datos. Una vez preprocesado el conjunto de datos, se utilizará para entrenar el clasificador. Se obtuvo un *dataset* de 2000 datos divido en dos partes para el entrenamiento y prueba una vez que el preprocesamiento fue realizado cada clasificador fue entrenado y luego evaluado, las métricas usadas fueron precisión, *recall* y *f1-score* para evaluar el desempeño de los algoritmos. Los resultados obtenidos en precisión son: *Naive Bayes* obtuvo un 89%, *SVM* obtuvo un 88% y *Random Forest* obtuvo 85%.

Hussainalsaid et al., (2015) mencionan detectar de manera automática el sentimiento de los textos que contienen dichos documentos cobra mayor importancia, para ello se utiliza el procesamiento de lenguaje natural (*PLN*) que servirá para clasificar de manera automática el contenido emocional de los documentos URL. Respecto al *dataset* utilizado, mencionan que como cualquier sistema de machine learning es necesario preparar la base de datos a utilizar, debido a lo novedoso de su propuesta mencionan que no se pudo encontrar ningún *dataset* que vaya acorde a lo que necesitaban por lo que decidieron crear el suyo. El *dataset* creado consiste en seis columnas: a) La URL donde se encontró la página web original; b) La fecha en la que se obtuvo la URL; c) El texto que ha sido extraído de la página web en la misma fecha que aparece en el punto b; d) Etiquetar de forma manual los textos obtenidos para poder utilizarlo. Su metodología usada es el bigrama, es un caso especial de modelo de N-gramas con N=2 que puede calcularse contando la frecuencia de palabras seguidas en un corpus (Conjunto cerrado de textos o de datos destinado a la investigación científica), luego el Bi-grama puede ser calculado por normalización o por la estimación de máxima probabilidad que divide el recuento de la palabra T o por el número de tokens de palabras M. Para evaluar los modelos de lenguaje versus los de grupo, como Bigrams, se divide el cuerpo en grupos de entrenamiento y de prueba; se entrena los parámetros estadísticos del modelo en el conjunto de entrenamiento y luego son usados para calcular las probabilidades en el conjunto de prueba. El algoritmo de aprendizaje automático analiza el contenido de cada URL, independientemente de si la URL tiene una dirección violenta o no violenta, y detectará los patrones repetidos en cada categoría para que las futuras URL se clasifiquen como violentas o no violentas. Los resultados no fueron los esperados, pues el clasificador etiquetar todo como no violento (es decir 100% falsos negativos y cero falsos positivos), se menciona que el principal problema de este resultado podría ser el tamaño de la base de datos, así como la precisión.

El trabajo de Peng et al., (2018) se basa en el análisis del texto de los mensajes que se encuentra dentro de los emails de *phishing* para verificar la intención con la que se mandó el mensaje. La sentencia se considera perjudicial si solicita información confidencial o dirige una acción que pueda revelar información personal. Se aplican técnicas de procesamiento de lenguaje natural (*PLN*) para analizar cada oración y determinar el papel semántico de las palabras importantes en la oración en relación con el predicado. Por el papel que juega cada palabra en la oración, nuestro enfoque determina si la oración es una pregunta o una necesidad. Los posibles argumentos para la pregunta y el comando se extraen encontrando pares (objeto - verbo directo). Luego, cada aplicación de igual a igual se evaluará en función de si está incluida en la lista negra por temas maliciosos de igual a igual. Utilizamos el aprendizaje automático supervisado para incluir en la lista negra los pares maliciosos (acción directa del objeto) en función de los pares en nuestros conjuntos de capacitación de correo electrónico de phishing y no phishing. El sistema creado para esta propuesta se llama SEAHound, procesa para cada documento una frase a la vez y si el documento contiene algún ataque de ingeniería social. El algoritmo evalúa cada oración para determinar cuatro características: 1) Pregunta/comando maliciosa; 2) Tono urgente; 3) Saludo genérico; 4) URL maliciosa. Para saber si la URL incluida en el correo es maliciosa se utilizará la herramienta Netcraft Anti-Phishing que para efectos prácticos será de utilidad. Para el *training set* se utilizó 1000 *phishing emails* del Nazario *phishing email* y 1000 no *phishing emails* de Enron Corpus para el *dataset test* se utilizó 5014 phishing emails de Nazario y 5000 no phishing email de Enron Corpus. Después del entrenamiento se consideró que un par es malicioso si supera el umbral de confianza de 0.9 o superior. Utilizando este umbral se obtuvieron 636 pares (verbo-objeto directo) que se usaron para la lista negra. Los resultados obtenidos mostraron que usar solo la herramienta Netcraft no es del todo efectivo pues sólo obtuvieron 3625 verdaderos positivos mientras que SEAHound obtuvo 4545 verdaderos positivos, por el otro lado en precisión Netcraft gana con un 98% a comparación de SEAHound con un 95%.

2.4 Análisis de sentimientos con *BERT*

Sousa et al., (2019) explica que cuando se producen cambios de último momento, los precios de las acciones pueden cambiar drásticamente. Analizar dichas noticias puede tomar tiempo y hacer un análisis rápido sin cuidado puede inducir al error. En estas circunstancias difíciles de manejar, se necesita una forma más rápida de ayudar a los inversores. Por dicho motivo, los autores proponen el uso de Representaciones Codificadoras Bidireccionales de Transformadores por sus acrónimos en inglés llamado *BERT*, para el análisis de sentimiento en artículos de noticias e informar las decisiones del mercado de valores. Este modelo se entrenaría previamente en una gran cantidad de documentos de dominio general mediante una tarea de auto entrenamiento. Para poder refinar este modelo, se etiquetó manualmente los artículos de noticias bursátiles como positivos, neutrales o negativos. Se menciona que el conjunto de datos utilizados está disponible de forma gratuita y contiene 582 documentos de varias fuentes con las que puede trabajar el modelo. La propuesta en esta investigación es evaluar *BERT* en el problema de análisis de sentimiento de

noticias financieras para mejorar la previsión del mercado de valores. Este es un breve artículo donde presentamos los resultados preliminares. *BERT* es actualmente uno de los modelos lingüísticos más exitosos. El modelo se basa en un codificador de transformador, el transformador es una arquitectura de secuencia a secuencia que se basa completamente en el mecanismo de atención del codificador como decodificador. La arquitectura *BERT* ignora la red del decodificador y usa solo un codificador de transformación porque no es un modelo de secuencia a secuencia (aunque se puede usar para tales tareas).

La mayoría de los modelos de lenguaje se basan en una arquitectura unidireccional, es decir la salida está determinada sólo por las palabras anteriores (contexto izquierdo). El uso de estos modelos en ejercicios posteriores también restringe los modelos refinados a solamente la posición izquierda. Esto es una restricción de las tareas en donde todo el texto está disponible durante la predicción. Para explorar este conocimiento, *BERT* implementa una arquitectura de modelo de lenguaje bidireccional. El análisis de sentimiento se modela como clasificación de texto y, por lo tanto, se puede utilizar. El esquema de su trabajo se divide en tres partes: primero, la adquisición y preprocesamiento de artículos de noticias bursátiles; segundo, el modelo de análisis de sentimiento basado en *BERT* y tercero, la aplicación del modelo desarrollado para mejorar la toma de decisiones de pronóstico del mercado de valores. En la adquisición de los datos, luego de ser recolectada cada documento es transformado en un token de secuencia la tokenización fue hecha usando *WordPiece* (el cual es un algoritmo de tokenización). Luego, pasa a la etapa del análisis de sentimiento donde menciona que los propios creadores de *BERT* recomiendan dos modelos con diferentes valores uno llamado *BERT BASE* y otro llamado *BERT LARGE*, ambos con diferentes parámetros, pero debido a las limitantes de hardware se optó por el *BERT BASE*, se realizó también una validación cruzada de 10 veces y una vez que el modelo estaba en funcionamiento se utilizó el modelo entrenado con los datos etiquetados. En los resultados, se obtuvo los siguiente resultados en el apartado de exactitud comparando *BERT* con los algoritmos de *Naive Bayes* y *SVM*. La exactitud en BERT fue de 82.50%, mientras que en *Naive Bayes* el valor más alto fue de 61% y el valor más alto en *SVM* fue de 62.4%.

Alaparthi y Mishra, (2021) investigan la eficacia relativa de cuatro métodos diferentes de análisis de sentimientos: un modelo no supervisado basado en un léxico que utiliza *Sent WordNet*; un modelo tradicional de aprendizaje automático supervisado que usa regresión logística; utilizando el Modelo de Aprendizaje Profundo Supervisado para la memoria larga a corto plazo (*LSTM*); el Modelo de Aprendizaje Profundo Supervisado mejorado utilizando la Transformación de Representación del Codificador Bidireccional (*BERT*). Para su trabajo se utilizó un *dataset* etiquetado disponible públicamente de 50,000 reseñas de películas publicadas originalmente en *Internet Movie Database* (*IMDB*), para su posterior análisis con *Sent WordNet*, la regresión logística, *LSTM* y *BERT*. Para el preprocesado del texto se retiraron los signos de puntuación "raros" como el símbolo mayor, menor, hashtags, arrobas, comas, etc. Se lematizo las palabras, se reemplazó las mayúsculas por minúsculas, se quitó las palabras que se repetían menos de 1%, se removieron *stopwords*. Asimismo, se explicó cada uno de los métodos, por ejemplo: *Sent WordNet*, este grupo de métodos no está supervisado y se basa en léxicos o léxicos creados y diseñados específicamente para el análisis de sentimientos. El léxico contiene información sobre palabras, emociones positivas o negativas asociadas, polaridad caracterizada por la cantidad de positividad o negatividad de una palabra, etiquetas verbales (POS). El análisis de sentimiento que utiliza esta técnica implica el asociar las palabras en un corpus con la información correspondiente en un diccionario, así como información contextual para obtener una puntuación de sentimiento para los documentos que componen el corpus. Regresión Logística, los algoritmos tradicionales de aprendizaje automático predictivo supervisado también se pueden usar para entrenar modelos de clasificación de sentimientos. Dependiendo del contenido del comentario, el sentimiento es positivo o negativo. Sigue la validación del modelo. Uno de los métodos más populares para realizar esta clasificación binaria es la regresión logística. Los modelos *logit* se basan en la teoría de maximización de la utilidad que incluye factores aleatorios. El análisis de opinión mediante regresión logística se basa en un modelo de bolsa de palabras, en el que cada documento del corpus se trata como una bolsa de palabras no estructurada, independientemente de su contexto, orden o gramática.

Los algoritmos de redes neuronales más utilizados son los basados en mecanismos *feedforward* y propagación inversa de errores, que facilitan el análisis de entradas, salidas y múltiples capas ocultas. Una variante de las redes neuronales profundas es la memoria larga a corto plazo (*LSTM*), que tiene una arquitectura de red neuronal recurrente (*RNN*) que se puede usar para el análisis de sentimientos de datos textuales. Una celda *LSTM* generalmente consta de una celda, una puerta de entrada, una puerta de salida y una puerta de olvido. Los *LSTM* son redes bidireccionales que han demostrado ser más precisas que los perceptrones multicapa convencionales y los *RNN* estándar. *BERT*, tiene como objetivo entrenar previamente representaciones bidireccionales profundas de texto sin etiquetar al condicionar conjuntamente el contexto izquierdo y derecho en todas las capas. El *BERT* de la última generación está pre-entrenado para realizar dos tareas no supervisadas: modelado de lenguaje de código de enmascaramiento y la predicción de la siguiente oración, lo que la convierte en una técnica eficiente para la clasificación de sentimientos. *BERT* es una técnica avanzada y más realista porque acepta el hecho de que un documento puede pertenecer a varias clases al mismo tiempo. Se sabe que *BERT* logra excelentes resultados en 11 tareas de comprensión del lenguaje natural. La confiabilidad de *BERT* se puede inferir del hecho de que Google lo usa en su algoritmo de búsqueda y actualmente está disponible en más de 70 idiomas diferentes. De los 50,000 reseñas del *dataset* para los modelos se dividió en

60% para el entrenamiento y 40% para la validación, excepto para el modelo *BERT* que se le dio 35% entrenamiento, 15% validación y 50% prueba esto con el fin de evitar *overfitting* y que entrenar grandes cantidades de datos con *BERT* requiere de mucha capacidad computacional por lo que es mejor correr el *dataset* en pequeñas partes. Los resultados arrojaron en exactitud los siguientes valores: *Sent WordNet* obtuvo 63.08%, Regresión logística obtuvo 89.41%, *LSTM* obtuvo 86.67% y *BERT* obtuvo 92.31% siendo este último el que mejor resultado tuvo.

2.5 Conjunto de datos utilizados en trabajos de *phishing*

El conjunto de datos o también llamado dataset utilizado en los trabajos leídos para este documento suelen tener cierta estructura y cierto tipo de valores. También, el formato en el que se encuentra este conjunto de datos varía dependiendo de los investigadores y las preferencias o necesidades que tengan, pero los formatos más populares son el CSV o TXT. Este tipo de datasets se puede encontrar en diferentes medios, hay tres principales fuentes que en los últimos años han sido los más utilizados para encontrar este tipo de archivos. Dataset como Search, Kaggle, Mendeley, son páginas que almacenan varios tipos de dataset de diferentes temas, entre ellos el phishing. Por ejemplo, en el artículo de Abedin et al., (2020), mencionan que su dataset fue sacado de Kaggle, como también en el artículo de Korkmaz et al., (2020), ellos mencionan otra página, también conocida de datasets exclusivamente de phishing llamada Phistank lo que demuestra que hay varias fuentes para conseguir el conjunto de datos adecuado a tu investigación. Abdillah et al., (2022) mencionan que el *dataset* más usado por los investigadores es PhishTank con un total de 34 autores, Alexa con 15 autores y UCI *Machine learning* con 12 autores de acuerdo con su investigación realizada. Otro punto de vista es propuesto por los autores Aleroud y Zhou (2017), donde los usuarios, especialmente los usuarios experimentados, también pueden participar en la creación de un conjunto de datos de phishing, que también contribuye al descubrimiento basado en las calificaciones de los usuarios, ya sea por autenticación manual o por votación del usuario.

Además, no es raro encontrar artículos donde han usado *datasets* proporcionados por su propia institución o universidad, por lo que son de carácter privado y no disponibles. Así mismo, al no haber sido posible encontrar un *dataset* que se adecue a sus necesidades, los investigadores crean el suyo. Por ejemplo, Aydin y Baykal (2015) crearon su propio programa para obtener *datasets* en base a sus necesidades, como ellos describieron para poder tener las características de un texto, ellos crearon su código en lenguajes C# para extraer dichas características. La forma en que se describen los *dataset* en estos trabajos suele ser en forma de lista numerada, donde cada punto es una característica (columna), que tiene el archivo en formato CSV o TXT, se escribe el nombre de dicha característica y a qué se refiere. Por otra parte , la mayoría de los trabajos, pero no todos, suelen describir o explicar cómo fue que se obtuvo el *dataset* utilizado para su investigación, ya sea que ellos mismos lo fabricaron o lo sacaron de alguna página de internet como los mencionados anteriormente. Para aclarar otro punto, el *dataset* que se suele encontrar para este tipo de clasificación es de dos columnas, en la primera columna está el texto y en la segunda está la etiqueta de si es positivo, negativo, neutral, etc. Mientras que, el *dataset* en machine learning suele tener más columnas dónde cada columna tiene un valor numérico el cuál será utilizado.

3. ANTECEDENTES

3.1 Procesamiento de Lenguaje Natural

El procesamiento del lenguaje natural (*PLN*) es una rama de la inteligencia artificial que ayuda a las computadoras a comprender, interpretar y manipular el lenguaje humano. El procesamiento del lenguaje natural se basa en muchos campos, incluidas la informática y la lingüística informática, en su búsqueda por cerrar la brecha entre la comunicación humana y la comprensión informática.

Hirschberg y Manning (2015) lo definen como la lingüística computacional, también conocida como procesamiento del lenguaje natural (*PLN*), es un subcampo de la informática que se ocupa del uso de técnicas computacionales para aprender, comprender y producir contenido en lenguaje humano. Los sistemas de lenguaje informático pueden tener muchos objetivos: el objetivo puede ser ayudar en la comunicación entre humanos, como en la traducción automática (MT); ayudar en la comunicación entre humanos y máquinas, como en el caso de los agentes conversacionales; beneficia tanto a humanos como a máquinas al analizar y aprender de la gran cantidad de contenido de lenguaje humano disponible en línea. Joseph et al., (2016) lo resumen como procesamiento del lenguaje natural (*PLN*), un método para analizar texto por medios informáticos. El PLN se preocupa por obtener conocimiento sobre cómo los humanos entienden y usan el lenguaje. Esto se hace para desarrollar herramientas y técnicas apropiadas que puedan hacer que los sistemas informáticos comprendan y manipulen para realizar varias tareas requeridas. Entre sus aplicaciones, se encuentran el análisis de sentimientos, resumen, extracción de información, relleno de espacios, análisis del discurso, vinculación textual.

3.1.1 Métodos principales de preprocesamiento

El primer paso para crear un modelo *PLN* es preprocesar los datos. Los datos de texto que se tienen son sin procesar y pueden contener muchos errores junto con muchos textos inesperados, por lo que los resultados no darán un efecto preciso. Por tanto, para obtener mejores resultados, es necesario procesar los datos y facilitar su comprensión y análisis.

- *LowerCase*: es uno de los pasos de preprocesamiento más comunes, donde el texto se convierte en el mismo caso, preferiblemente en minúsculas. Existe un enfoque común para hacer que todo sea pequeño en aras de la simplicidad. Ayuda a mantener un flujo constante en el procesamiento del lenguaje natural y las operaciones de minería de texto. Aunque a menudo se pasa por alto, poner todos los datos de texto en minúsculas es una de las formas más simples y poderosas de preprocesamiento de texto. Se puede aplicar a la mayoría de los problemas de procesamiento de lenguaje natural y minería de texto y puede ayudar en los casos en que el conjunto de datos no es muy grande y ayuda mucho con la consistencia de los resultados esperados.
- *Tokenization*: es el proceso de dividir un objeto de texto en unidades más pequeñas llamadas tokens. Ejemplos de símbolos pueden ser palabras, letras, números, símbolos o n-gramas. El conjunto de datos generalmente consta de párrafos largos que constan de muchas líneas, y las líneas se componen de palabras. Es muy difícil analizar párrafos largos, por lo que primero se divide los párrafos en líneas separadas y luego los dividimos en palabras. Es normal centrarse en el análisis puro o la generación, dando por sentadas las unidades básicas, es decir, las palabras. Sin embargo, es claro que sin estas unidades base bien separadas, no se puede realizar ningún análisis o generación. Pero hasta ahora se ha prestado poca atención al proceso, que es un preprocesador en el sentido de definir las unidades base a procesar (Webster y Kit, 1992).
- *Normalization*: la mayoría de los conjuntos de datos que contienen varias palabras se crean a partir de una sola palabra agregando un sufijo o prefijo. Estos casos pueden causar redundancia en nuestro conjunto de datos y no darán mejores resultados. Por ejemplo, tenemos las palabras anormal y buenísimo donde el prefijo es el a en anormal siendo su raíz la palabra normal y el sufijo es ísimo en buenísimo siendo la raíz la palabra bueno. Por lo tanto, es importante convertir estas palabras a su forma original, lo que también reduce la cantidad de palabras únicas en nuestro conjunto de datos y mejora nuestros resultados. En *PLN* existen dos métodos usados para realizar la normalización de un texto. La lematización y *stemming* se usan comúnmente en el procesamiento del lenguaje natural. Estas dos técnicas a veces se consideran iguales, pero hay una diferencia importante. La lematización es generalmente más fácil de usar, porque produce la forma básica de una palabra que se requiere en muchas áreas de aplicación (por ejemplo, procesamiento entre idiomas y traducción automática). Sin embargo, la lematización es una tarea difícil, especialmente en el caso de lenguajes naturales muy variables que contienen muchas palabras de la misma forma estándar (Toman et al., 2006).
    - Stemming: se utiliza para eliminar cualquier tipo de sufijo de una palabra y devolver la palabra a su forma original, pero a veces la raíz de la palabra resultante es una palabra insignificante o no léxica.
    - Lematización: la lematización es similar al *stemming*, pero más eficiente. En la lematización, la palabra generada después del corte del sufijo siempre tiene un significado y pertenece al diccionario, es decir, no genera palabras falsas. Las palabras creadas después de la derivación también se llaman lemas.
- Eliminar las *Stopwords*: las *stop words* son palabras en cualquier idioma que ayudan a consolidar la oración y hacerla significativa. Por ejemplo, en inglés hay diferentes palabras como “I, am, are, is, to, etc. Estas palabras se conocen como palabras vacías. Pero estas palabras no son útiles para el modelo, por lo que debemos eliminarlas del conjunto de datos para que podamos concentrarnos solo en las palabras importantes en lugar de estas palabras de apoyo.

3.2 *Machine learning*

Según Naqa y Murphy, (2015), se conoce a *machine learning* como una rama en desarrollo de los algoritmos computacionales diseñados para imitar la inteligencia humana aprendiendo constantemente, dependiendo de la entrada de datos que se le asigne. La cantidad de aplicaciones es amplia, desde el reconocimiento de patrones hasta aplicaciones biomédicas y médicas. Además, mencionan que existen cuatro tipos de técnicas de *machine learning*: a) las supervisadas: necesita un conjunto de datos etiquetados, b) las no supervisado: son aquellas que no tiene etiquetas en el conjunto de datos de entrada, pero como salida generan una para diferenciar los datos, c) las semi-supervisadas: es como una combinación de los dos tipos mencionados anteriormente, donde se etiqueta un grupo pequeño de datos como entrenamiento para que luego el propio programa lo pueda hacer, d) por refuerzo: se entiende como un algoritmo

que aprenda a partir de la propia experiencia, se suele utilizar con un conjunto de reglas donde hay "recompensas" si las cumple o "castigo" sino, es decir se le enseña a la máquina a través del ensayo y error (Kunju et al., 2019).

Otra definición proporcionada por Alkawaz et al., (2020), señala que el *machine learning*, es la ciencia de hacer que las computadoras actúen sin estar expresamente programadas. Entre sus usos se encuentra en la clasificación de textos en función de alguna característica extraída de estos.

3.2.1 Algoritmos de machine learning

De acuerdo con Shahrivari et al., (2020) el *machine learning* provee de diversos métodos simples y eficientes para analizar datos. Entre las técnicas más comunes usadas en *machine learning* se encuentran las siguientes:

- *Support Vector Machine (SVM)*: La idea de *SVM* es de acercarse al punto más cercano entre dos clases a través de la máxima distancia entre clases. Está técnica es una técnica de *machine learning* supervisado.
- *K Near Neighbors (KNN)*: La técnica se utiliza para agrupar y reincidir. Es uno de los más sencillos algoritmos usados en el aprendizaje automático para clasificación y regresión lineal.

Otras técnicas también populares son (Kunju et al., 2019):

- *Random Forest*: Enfocado en la metodología de clasificación y regresión apropiada para manejar problemas que involucran agrupaciones en categorías. Como su nombre indica contiene una gran cantidad de árboles de decisiones individuales que cada uno genera una salida, cada árbol se encuentra en un bosque aleatorio donde se especifica la predicción de la clase y la salida es aquella que más predicciones tuvo de los árboles.
- *AdaBoost*: Es sabido que se puede utilizar para la ejecución de cualquier cálculo relacionado a IA (inteligencia artificial). Se parece a *Random forest* en su clasificación, donde parten de una clasificación débil hasta una forma de clasificación fuerte.
- *Naïve Bayes*: Es un algoritmo de *machine learning* supervisado, dónde es basado de acuerdo con la teoría de Bayes y usado para resolver problemas de clasificación. Se puede usar para una clasificación binaría o para multi clases.
- *Maximum Entropy* (*MaxEnt*): Es un algoritmo basado en características. La mayor diferencia con *Naïve Bayes* es que no asume ninguna independencia y puede manejar mejor los problemas superpuestos (Phand y Phand, 2017).

3.2.2 *Naive Bayes*

Es un algoritmo simple que utiliza la regla de Bayes junto con una fuerte creencia de que los atributos son condicionalmente independientes de acuerdo con su clase. Aunque dicha independencia es rota a menudo en la práctica. Entre las característica que destaca de este algoritmo es su precisión de clasificación alta, además de su eficiencia en cómputo y otras características que hacen de esta técnica una de las más populares al momento de elegir un algoritmo de clasificación y sea ampliamente implementado en la práctica. El teorema de Bayes trata sobre calcular la probabilidad que tiene que ocurra un suceso cuando se cuenta con información de antemano sobre dicho suceso. Naive Bayes proporciona un mecanismo para utilizar la información de los datos de la muestra para estimar la probabilidad posterior P(A | B), es decir saber las probabilidades que suceda A sabiendo que ha sucedido B. Una vez que tengamos dichas estimaciones se pueden utilizar para clasificar u otras aplicaciones de apoyo de decisiones. La expresión de Naive Bayes es la siguiente:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \qquad (1)$$

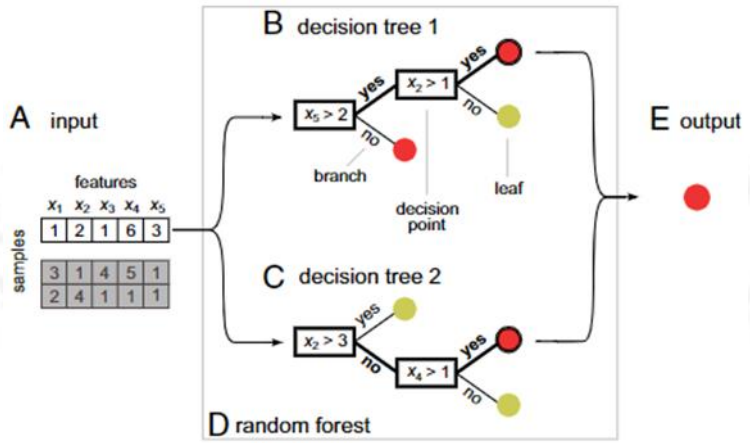
En este caso como se aprecia en la ecuación 1 se puede definir las probabilidades de la siguiente manera: P (A | B) = Probabilidad condicional de A dado B, P (B | A) = Probabilidad condicional de B dado A, P (A) = Probabilidad del evento A, P (B) = Probabilidad del evento B. La fórmula se puede dividir en tres secciones: P(A | B) se llama posteriori, P (B | A) P(A) se llama priori y P(B) se llama marginal. La A de P(A | B) es la hipótesis y la B es la evidencia, lo que significa que se debe encontrar la probabilidad de A en base a los datos dados de B. Lo mismo pasa si es P(B | A), es decir que se debe encontrar la probabilidad de B en base a los datos dados de A. Se ha utilizado para muchos propósitos, pero funciona especialmente bien para tareas de procesamiento de lenguaje natural (PLN). Naive Bayes es una familia de algoritmos probabilísticos que utilizan la teoría de la probabilidad y el teorema de Bayes para predecir etiquetas de texto (noticias, reseñas de clientes, etc.). Estos son estocásticos, es decir calcula la probabilidad de cada etiqueta en un texto en particular y luego genera la etiqueta con el valor más alto. Usan el teorema de Bayes para derivar estas probabilidades. El teorema de Bayes describe la probabilidad de una característica basada en el conocimiento previo de las condiciones que pueden estar asociadas con esa característica. Además, dentro de sus características se encuentra que tiene, baja varianza, aprendizaje incremental, predicción directa de las probabilidades posteriores, etc. (Webb et al., 2010).

3.2.3 *Random Forest*

Es un algoritmo mixto de aprendizaje automático. Se combinan con un conjunto de clasificadores de árboles, donde cada árbol emite un voto para la clase más popular y la combinación de estos resultados produce el resultado final de la clasificación. RF tiene una alta precisión de clasificación, tolera bien los valores atípicos y el ruido, y nunca se sobre ajusta. RF ha sido uno de los métodos de investigación más populares en la minería de datos e información en el campo biológico (Liu et al., 2012). Se explica cómo es que funciona la lógica detrás de random forest al momento de decidir, como se puede apreciar en la Figura 1.

Figura 1

Demostración gráfica de random forest



Nota. La imagen representa el proceso de *random forest* al momento de votar por la clase más popular. La primera parte, marcada con la letra A, corresponde a las muestras de entrada con cinco características. Las letras B y C representan las ramas que se bifurcan en puntos de decisión. La letra D engloba todo el proceso, describiéndolo como un bosque aleatorio. Finalmente, la letra E indica la salida, es decir, el consenso elegido entre los árboles de decisión. Tomado de *ResearchGate* (https://acortar.link/rPOTUJ)

En la imagen se puede apreciar en dónde dice la letra (A) está un conjunto de datos de entrada que queremos analizar, en este caso son tres muestras cada muestra es una fila. En la sección (B) se puede ver el árbol de decisión el cual está formado por sus dos ramas que se bifurcan en puntos de decisión, cada uno de estos puntos tiene una regla que asigna una muestra en una rama u otra de acuerdo con el valor de la característica y las ramas al final se dividen en dos hojas que una pertenece a la clase roja o a la clase amarilla. En la parte (C) se puede ver a otro árbol de decisión que está también clasificando a la muestra 1 con diferentes reglas en cada punto de decisión. Como se puede ver en el árbol de decisión 1 y 2 el resultado de cada uno devolvió que se clasificaba la muestra 1 como clase roja. En la parte (D) se ve a todo ese conjunto de árboles como un bosque aleatorio, en donde se combina los votos de los árboles que lo componen dando como resultado la predicción de a qué clase final pertenece. En la parte final (E) se puede ver como la conclusión del modelo después de sumar los votos de los árboles, fue que pertenece a la clase roja (Denisko y Hoffman, 2018).

3.2.4 *Support Vector Machine*

Existen varios algoritmos de aprendizaje automático que realizan la tarea de clasificación asignando etiquetas de categoría a los datos de entrada en función de los datos anteriores. ejemplo. Uno de estos algoritmos es *Support Vector Machine* (SVM), que se caracteriza por su capacidad para aprender modelos de clasificación de forma precisa y reproducible en una variedad de escenarios. SVM busca el mejor hiperplano que separa las observaciones de datos pertenecientes a una clase de las observaciones de datos de otra clase, utilizando como criterio la distancia máxima entre los puntos más cercanos a los límites de cada clase. Los hiperplanos pueden ser lineales o no lineales y pueden usarse para determinar las etiquetas más probables para datos invisibles. Para encontrar el hiperplano, SVM utiliza características de datos, que son valores numéricos que representan aspectos relevantes de los datos originales. Estas características no son necesariamente datos sin procesar, pero pueden ser datos obtenidos de transformaciones o selecciones anteriores. El objetivo de SVM es maximizar el porcentaje de etiquetas correctas asignadas a materiales no etiquetados después de la clasificación, garantizando al mismo tiempo que el modelo pueda generalizarse a nuevos materiales. Este objetivo lo comparten otros métodos de aprendizaje supervisado, pero las ventajas de SVM son la simplicidad, la eficiencia y la versatilidad (Pisner y Schnyer, 2020).

3.2.5 *BERT*

BERT acrónimo de Representaciones de Codificación Bidireccional de Transformadores en su traducción al español, se basa en una red neuronal de gran profundidad y bidireccional. Su metodología consiste en aprender a partir de grandes volúmenes de texto no etiquetado, sometiéndose a dos tareas no supervisadas: el modelo de lenguaje enmascarado y la predicción de la siguiente oración. Sin embargo, esto no significa que BERT no pueda trabajar con la metodología supervisada. De hecho, BERT se puede afinar con datos etiquetados para adaptarse a diferentes tareas específicas de procesamiento del lenguaje natural, como el análisis de sentimientos, la respuesta a preguntas o el reconocimiento de entidades nombradas. Este ajuste se realiza añadiendo una capa de clasificación y realizando un ajuste fino en todos los parámetros del modelo. BERT ha demostrado obtener resultados de vanguardia en once tareas de procesamiento del lenguaje natural, superando a modelos anteriores en términos de precisión y generalización. La versatilidad de BERT radica en su capacidad para adaptarse a diversas tareas con mínimos cambios en la arquitectura específica de cada tarea, simplemente mediante la incorporación de capas de salida adicionales. (Devlin et al., 2018)

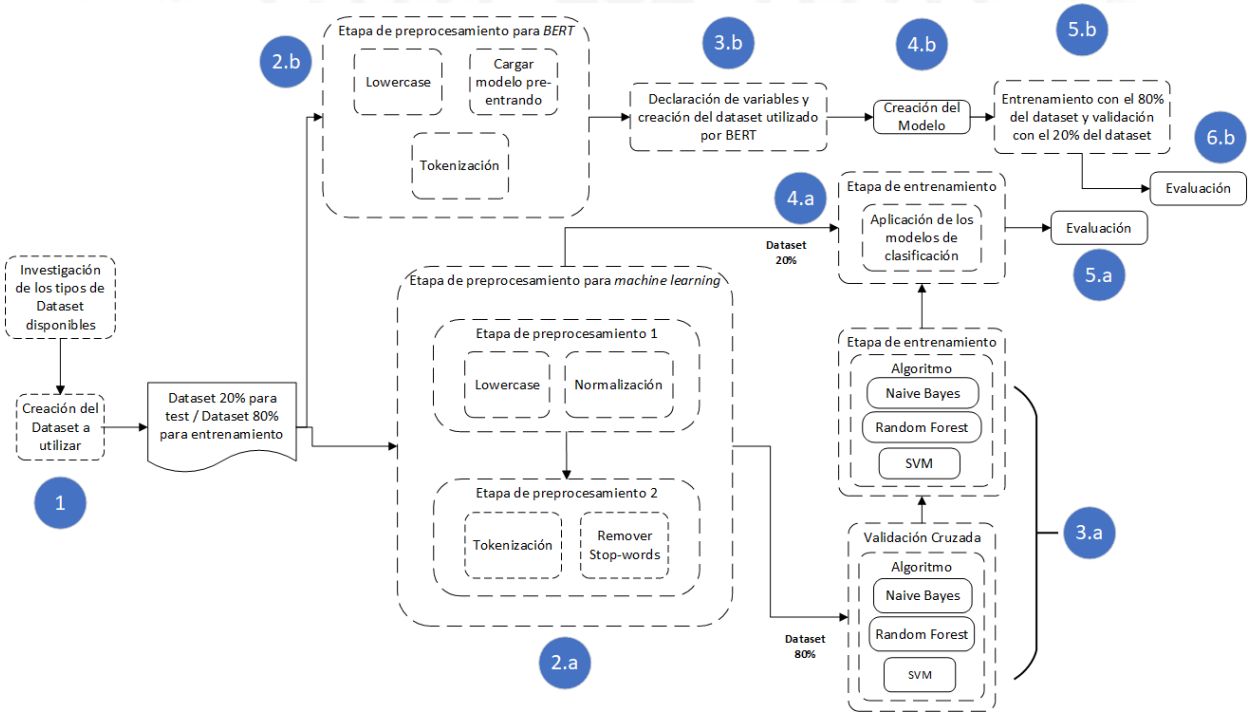
BERT se capacita en dos fases: pre-entrenamiento y ajuste fino. En la fase de pre-entrenamiento, BERT se educa utilizando grandes volúmenes de texto no etiquetado. Este proceso de aprendizaje se lleva a cabo mediante dos tareas no supervisadas: el modelo de lenguaje enmascarado (MLM) y la predicción de la siguiente oración (NSP). En el MLM, se ocultan ciertas palabras en una oración y luego BERT intenta predecir esas palabras ocultas basándose en el contexto proporcionado por las palabras visibles. Este procedimiento permite a BERT aprender representaciones de palabras que consideran tanto el contexto anterior como el posterior. En la NSP, BERT recibe pares de oraciones como entrada y aprende a predecir si la segunda oración en el par es la oración subsiguiente en el texto original. Este procedimiento ayuda a BERT a aprender a comprender las relaciones entre las oraciones. Después del pre-entrenamiento, BERT puede ser personalizado para una tarea específica de PLN. Durante el ajuste fino, se añade una capa de salida al modelo BERT pre-entrenado y se ajustan todos los parámetros del modelo en los datos de la tarea específica.

4. METODOLOGÍA

Se puede apreciar la secuencia de trabajo utilizada en la Figura 2.

Figura 2

Metodología implementada



Nota. La imagen representa los pasos que se siguieron para el desarrollo de la investigación. Fuente: elaboración propia

4.1 Generación del *dataset*

Esta etapa corresponde a la "etapa 1" de la figura 1. No se logró encontrar un *dataset* que cumpliera con las necesidades requeridas para la investigación por lo que se decidió crear uno. El *dataset* creado está compuesto de la

siguiente manera: tiene dos columnas, la primera es la lista de comentarios y la segunda es una lista de etiquetas que identifica a los comentarios como positivo o negativo como se puede apreciar en la Tabla 1. El criterio para determinar si un comentario era positivo o negativo se basó en las palabras utilizadas o en el mensaje que se quería transmitir con los comentarios. Por ejemplo, si se usaban palabras como "gracias", "amable", "tengan cuidado", etc., se podía suponer que se trataba de un comentario positivo. En el caso de los comentarios negativos, se utilizó la misma lógica basada en las palabras utilizadas o la intención que transmitían sus mensajes. Por ejemplo, cuando mencionan que no podían dejar su queja en el libro de reclamaciones, que no alertaban muy seguido sobre este tipo de robo o que el banco no tomaba medidas contra este tipo de mensajes, etc.

Tabla 1

Muestra del dataset que contiene los comentarios y su estado

Comentarios	Tipo de comentario
"Y que hacen con los números reportados? Van a publicarlos para que la gente esté alerta y los bloquee? Van a denunciar los números para que les den de baja? Cual es la finalidad?"	Negativo
"Gracias por avisarnos con esta publicacion hay que estar atentos"	Positivo
"Esa denuncia de esos mensajes hise hace un par de meses al bcp me volvieron a enviar ese mensaje obviamente con otro nro pero veo k la pagina sigue siendo la misma al parecer bcp no hace nada con esa pagina para k le den de baja"	Negativo
"A mi también me han llegado ese tipo de mensajes y me mandan a crear una nueva cuenta, tengan mucho cuidado."	Positivo
"He recibido mensajes personales de sus mismos colaboradores... y no se puede cargar nada en su libro de reclamaciones"	Negativo
"que bueno que ahora ya estan avisando de este tipo de casos"	Positivo

Nota. La tabla muestra tres ejemplos de comentarios positivos y tres ejemplos de comentarios negativos. Fuente elaboración propia.

Para su creación se utilizó la red social Facebook para recolectar los comentarios de las personas y para poder recolectarlos se usó la extensión *Instant Data Scrapper* que sirvió para recolectar las opiniones de usuarios acerca de publicaciones de bancos de Perú sobre *phishing*. Una vez que se creó el *dataset* y se verificó que cumplía con los requisitos, es decir, que se pudo crear dos columnas (la primera siendo la lista de comentarios y la segunda una lista de etiquetas que identifica a los comentarios como positivos o negativos), se procedió a hacer una limpieza de los comentarios, eliminando emojis o palabras sin sentido que no aportan al modelo. Luego, se procedió a etiquetar cada uno de los comentarios como positivo o negativo. El *dataset* escogido cuenta con alrededor de 2215 registros por lo que pasará a ser dividido en dos partes: el 80% del *dataset* será utilizado para el entrenamiento y el 20% será usado para la prueba, el motivo por el cual se usa dichos porcentajes es porque en los trabajos revisados solían usar dichos porcentajes o valores similares para el *dataset* de entrenamiento y el de prueba.

Al finalizar el paso 1 del gráfico que representa la metodología implementada, se observa como el gráfico se divide en dos líneas. La línea de abajo hace referencia al método de machine learning que utiliza los algoritmos Naive Bayes, Random Forest y SVM, mientras que la línea de arriba se refiere al método del *transformer* BERT. Ambos métodos se usaron para la clasificación de opiniones de usuarios acerca de publicaciones de bancos en Perú sobre phishing.

4.2 Método de *Machine learning*

4.2.1 Preprocesamiento de *machine learning*

Esta es la etapa de preprocesamiento corresponde a la 2.a de la figura 1, se dividió en dos subsecciones llamadas etapa de preprocesamiento 1 y preprocesamiento 2. Para está etapa de la metodología se utilizó la librería de Python llamada *Natural Language Toolkit* por su acrónimo en inglés *NLTK*, la cual es un conjunto de bibliotecas y programas simbólicos y estadísticos de procesamiento de lenguaje natural (PLN) y para la parte de lematización cómo se está trabajando con el idioma español se decidió usar la librería Spacy debido a que *NLTK* no soporta ese proceso en el idioma español. En la etapa de preprocesamiento 1 se agrupó *lowercase* y lematización. Estas etapas de preprocesamiento 1 se detallan a continuación:

- El *lowercase* o minúsculas en su traducción, se refiere a minimizar todas las letras que se encuentren en los comentarios con el objetivo que no se malinterprete las letras en mayúsculas con alguna otra intención y que todo el texto sea tratado como igual. Por ejemplo, si tuviéramos un texto que diga "¿HOLA cómo Estás?" Lo que hace el paso de *lowercase* es poner en minúscula toda la frase dando como resultado "¿hola cómo estás?".
- Para la normalización existen dos formas de convertir un token a su forma base, una forma es por *stemming* la cual es una forma de eliminar los sufijos de una palabra para que esté vuelva a su forma raíz, pero no suele ser muy recomendado su uso para pruebas finales debido a que algunas veces puede producir palabras que no tienen significado y que no están en el diccionario. La otra forma de normalización es por lematización, cual es un proceso sistemático para remover la forma flexiva de un elemento y transformarlo en un lema que puede ser definido como la forma canónica de un conjunto de palabras. El tipo de normalización escogido para esta investigación ha sido la de lematización debido a que suele ser la más completa y fiable al momento de regresar la palabra a su forma base. Por ejemplo, para la lematización la palabra "decir" es el lema o forma base de las palabras "dije", "diré", "dijéramos", etc.

En la etapa de preprocesamiento 2 se juntó la tokenización y la remoción de las *stop-words*. Las fases del preprocesamiento 2 se detallan a continuación:

- Para el caso de tokenización lo que se realiza es el proceso de separar un texto en nuestro caso un comentario en unidades más pequeñas llamadas token, estos pueden ser palabras, caracteres o subpalabras. Por ejemplo, una oración que diga "Hola, mucho gusto" al tokenizar sería separada de la siguiente manera ["Hola", ",", "mucho", "gusto"].
- Las *stop-words* hacen referencia a las palabras más usadas en un lenguaje, la razón por la cual las palabras vacías son importantes es que muchas veces si se eliminan las palabras que se usan con mucha frecuencia en un idioma determinado podemos centrarnos en las palabras que realmente importan. En este caso las *stopr-words* utilizadas incluye una lista de 40 *stop-words* en este caso del idioma español, dentro de la lista se puede encontrar palabras comunes como: 'habida', 'tuvo', 'estuviese', 'fueses', 'o', 'muchos', 'porque', etc.

4.2.2 Entrenamiento y Evaluación

Esta etapa corresponde a la 3.a figura 1, en este caso se utilizó la librería de sklearn importando cross_val_score, en este caso para la validación se está realizando con 5 iteraciones. Lo normal que se suele usar en este tipo de validaciones son 5 o 10 iteraciones y como resultado se obtiene la exactitud de cada una de esas iteraciones, en nuestro caso se usa el 80% del *dataset* para ver si difiere mucho los resultados obtenidos y su promedio de cuando se realiza la prueba con el 20% del *dataset* restante.

Se utilizaron tres algoritmos de aprendizaje máquina para entrenar el modelo propuesto para la clasificación de opiniones de usuarios acerca de publicaciones de bancos de Perú sobre *phishing*. Se utilizó el 80% para la etapa de entrenamiento con el algoritmo de aprendizaje máquina escogido. En este caso el primer algoritmo es *Naive Bayes* con los parámetros por defecto lo cuales son: un suavizado aditivo de Laplace o Lidstone igual a 1 para evitar el problema de probabilidad cero de *Naive bayes* en caso una palabra en revisión no haya estado presente en el conjunto de entrenamiento, sí se tomará en cuenta las probabilidades previas de la clase y las probabilidades previas de las clases se ajustan de acuerdo con los datos.

El segundo algoritmo *Random Forest* con los parámetros por defecto los cuales son el uso de cien árboles para el algoritmo, con un criterio de Gini para medir la calidad de una división. No se usa ningún estado aleatorio de base para el arranque de las muestras utilizadas. Se planea agregar más algoritmos a futuro si se logra culminar el entrenamiento con el algoritmo escogido actual o si se opta por otro algoritmo en vista de qué tan complejo sería implementarlo en la metodología.

El tercer algoritmo *Support Vector Machine (SVM)* con los parámetros por defecto los cuales son C que se refiere a un parámetro de penalización del término de error, donde es considerado como el grado de clasificación correcta que debe cumplir el algoritmo donde existe dos términos el primero se refiere a la regularización que beneficia a los pesos más simples y el otro término se refiere a que se clasifique correctamente los puntos de datos de entrenamiento. El parámetro gamma define hasta dónde llega la influencia de un solo ejemplo de entrenamiento, donde los valores bajos significan "lejos" y los valores altos significan "cerca" si gamma es demasiado grande, el radio del área de influencia de los vectores de soporte solo incluye el propio vector de soporte y cuando gamma es muy pequeño, el modelo está demasiado restringido y no puede capturar la complejidad o la "forma" de los datos. Se utiliza "*Kernel*" porque se utiliza un conjunto de funciones matemáticas en la *SVM*, que proporciona ventanas para manipular los datos. Por lo tanto, las funciones del *kernel* a menudo transforman los conjuntos de datos de entrenamiento para que las superficies de decisión no lineales pueden transformarse en ecuaciones lineales en espacios de mayor dimensión.

Esta etapa corresponde a la 4.a figura 1, después de la etapa de entrenamiento se pasará a la fase de prueba del modelo, dónde se usará el 20% restante del *dataset* escogido. En este paso se usa el 20% restante del *dataset* para evaluar la efectividad del modelo que pueda existir al usar datos diferentes que en el entrenamiento.

Esta etapa corresponde a la 5.a figura 1, se realiza la evaluación y comparación de los resultados obtenidos. Con los datos obtenidos se podrá llegar a la conclusión de si el modelo cumplió con lo previsto o no, además de ver que valores obtuvo en las métricas de exactitud, precisión y *recall*. También, servirá como base para la comparación de los resultados con otros algoritmos de aprendizaje máquina que se planteen implementar en futuras investigaciones.

4.3 Método BERT

4.3.1 Preprocesamiento de *BERT*

En esta etapa se realiza el preprocesamiento de los datos para su uso en el modelo *BERT* correspondiente a la 2.b de la figura 1. Se realizan tres pasos en esta etapa:

- Primero: Se realiza un *lowercase*, el cual consiste en poner en minúsculas todo el texto del dataset, esto con el objetivo de que no se malinterprete cualquiera de los comentarios del dataset en base a si tiene mayúsculas o no. Con esto se busca generar una estandarización de todos los comentarios agregados en el dataset. En este caso, el modelo pre-entrenado es en español de tipo *uncased* lo que indica que fue entrenado con todas las palabras en minúsculas, por lo que para que siga manteniendo ese orden se procede a realizar *lowercase* a todos los comentarios del *dataset* creado.
- Segundo: Se procede con el cambio de la columna que tiene la etiqueta de si el comentario es negativo o positivo a un valor número. En este caso se asigna como positivo el valor numérico de 1 y negativo el valor numérico de 0, esto con el objetivo de que tengamos una representación numérica de dichas etiquetas al modelo que será entrenado.
- Tercero: Se realiza el proceso de tokenización, al modelo base de *BERT* sin entrenamiento previo para que detecte sentimiento, requiere que el texto este presentado como dos frases. En el formato que utiliza *BERT* la primera frase va a ser el comentario original, al comienzo de la primera frase va a colocar un token de clasificación llamado *CLS*. Este único token servirá como un identificador, pues al momento de clasificar el comentario que es procesado por *BERT* dicho token contiene toda la información condesada del texto. Luego, cada palabra del texto será codificada con un número para que sea entendido por *BERT*. Después de que cada palabra del comentario se codifique en un número, se incluye un token especial llamado *SEP* que sirve como separador en caso se tenga otra frase. En nuestro caso no se tiene otra frase por lo que después del token *SEP* se añade otro token llamado *pad* lo que indica que no tenemos ningún tipo de texto.

  Para realizar la tokenización se debe de cargar el modelo. En este caso como se hace uso del idioma español, se carga un modelo en español llamado "dccuchile/bert-base-spanish-wwm-uncased" de tipo *uncased*. Se procede a definir la función *tokenizer*, la cual sirve para realizar el proceso de tokenización del texto que se le proporcione. En este caso, se le entrega el modelo pre-entrenado en español. Esto da como resultado un tokenizador BERT pre-entrenado que está optimizado para español y puede usarse para tareas de procesamiento de lenguaje natural en español.

4.3.2 Variables, modelo, entrenamiento y evaluación

Esta etapa corresponde a la 3.b figura 1, en esta parte se definió los valores de los parámetros que queramos asignar al modelo. Se declaran cuatro parámetros:

- El primer parámetro sirve como una semilla llamada *random seed*. Permite que se pueda reproducir los mismos resultados cada vez que se ejecute el código. Además, al configurar una semilla aleatoria, proporcionará una inicialización aleatoria al modelo, como la inicialización de los pesos de las iteraciones, la selección de tokens para el entrenamiento. De manera arbitraria se asignó el valor 63760142 a este parámetro.
- El segundo parámetro representa la máxima longitud. Como en este caso se está usando comentarios de personas, estos pueden llegar a ser muy largos algunas veces, por lo cual para evitar sobrecargar el modelo y consumir recursos computacionales y de memoria se define un límite. El valor de esta variable es de 100 palabras como máximo.
- El tercer parámetro sirve para definir el tamaño del lote, también llamado *batch size*. Nos permite definir cuántos ejemplos se procesan simultáneamente en una iteración en la etapa de entrenamiento, esto también para que la memoria no se sature. El valor de este parámetro es de 16 comentarios.

- El cuarto parámetro sirve para definir el número de clases. En este caso el objetivo es saber si un comentario es positivo o negativo, por lo cual el número de clases serán dos.

Luego de declarar los parámetros a usar, se procedió a crear un *dataset* para que pueda ser utilizado por *BERT*. Se empieza inicializando los comentarios, la etiqueta si fueron positivos o negativos, el *tokenizer* que permite convertir el texto al formato que se requiere en *BERT* y la longitud máxima que es cien. Se crean los bloques en base al *batch size* los cuales son seleccionados de forma aleatoria en bloques de 16 comentarios representados en el modelo. El comentario se convierte en una representación numérica que es entendida por el modelo *BERT* utilizando un diccionario que contiene la versión codificada del comentario, el cual contiene también el identificador de los tokens, la máscara de atención y con eso se devuelve los tensores de *Pytorch*. Estos tensores serán utilizados posteriormente para alimentar el modelo *BERT* durante la etapa de entrenamiento.

Esta etapa corresponde a la 4.b figura 1, en este punto se crea el clasificador se análisis de sentimiento basado en *BERT*. Esto toma como entrada los ids de los tokens y la máscara de atención para producir una salida que tenga las probabilidades de representar a un sentimiento positivo o negativo. Primero se configuran las iteraciones y parámetros necesarios para el modelo enfocado al análisis de sentimiento. Luego se crea una instancia del modelo *BERT* pre-entrenado usando el nombre del modelo pre-entrenado seleccionado previamente, en nuestro caso es el modelo *BERT* en español llamado *bert-base-spanish-wwm-uncased*, se prosigue con crear una capa de *dropout* con el objetivo que no haya sobreajuste (*overfitting*) durante el entrenamiento.

Después se define el flujo de datos del modelo para realizar la clasificación de sentimientos. Primero toma la representación numérica de los tokens el cuál es el valor del id del texto de entrada y su mascara de atención (el cual indica a la red neuronal qué partes de la entrada son relevantes y cuáles no deben ser tenidas en cuenta durante el procesamiento), se pasan los datos por el modelo *BERT*. En este punto se obtiene dos valores, la primera contiene todos los datos que corresponde a la codificación del comentario de entrada, es decir que me entrega todos las palabras codificadas del comentario y como segundo valor arroja la codificación del token de clasificación es decir el *CLS*. En nuestro caso el segundo valor es el que nos interesa, pues nosotros queremos clasificar los comentarios. Finalmente, se realiza el *dropout* con el segundo valor que obtuvimos del modelo y se apagan algunas conexiones durante el entrenamiento para prevenir el sobreajuste, luego se pasa por una capa lineal para reducir la dimensionalidad al número de clases que queremos predecir y se obtiene las predicciones del modelo para cada clase.

Esta etapa corresponde a la 5.b figura 1, en esta etapa se realiza el entrenamiento para que el modelo pueda enfocarse en texto de análisis de sentimiento y no texto en general. Primero se establece un total de 5 de iteraciones a través de todo el modelo llamadas *Epochs*, Los pesos del modelo son modificados durante el entrenamiento por un optimizador. Los parámetros del modelo se actualizan en este caso utilizando el optimizador AdamW, una variación del optimizador Adam. Se incluye el número total de pasos del entrenamiento, calcula cuántos pasos de entrenamiento se realizarán en total, considerando todos los lotes por todos los *Epochs*. Luego se procede a procesar los datos divididos en datos de entrenamiento y datos de prueba, los datos de entrenamiento se reciben como entrada al modelo, a medida que se itera el modelo se calcula las predicciones del modelo del *batch* actual, calcula el número de predicciones correctas, realiza una retropropagación del error para ajustar los pesos del modelo, se aplica la técnica de clip de gradiente para evitar problemas con gradientes muy grandes, se actualiza los pesos del modelo utilizando el optimizador y se reinicia los gradientes para el siguiente *batch*. Al final de todo este proceso se imprime la cantidad de predicciones correctas y el número total de ejemplos del entrenamiento, además que devuelve la exactitud con la cual se clasificó los comentarios como positivo o negativo. Para los datos de testeo se recibe el conjunto de datos que serán procesados a medida que se itera por cada *batch*, se calcula las predicciones del modelo para el lote actual, se procede con la pérdida entre las predicciones y las etiquetas reales, se calcula la cantidad de predicciones correctas para al final imprimir la cantidad de predicciones correctas y el número total de datos que hubo en el texto. También, devuelve la exactitud que se obtuvo durante esta etapa.

Esta etapa corresponde a la 6.b figura 1, se realiza la evaluación y comparación de los resultados obtenidos. Con los datos obtenidos se podrá llegar a la conclusión de si el modelo cumplió con lo previsto o no, además de ver que valores obtuvo en las métricas de exactitud. También, servirá como base para la comparación de los resultados con otros algoritmos de aprendizaje máquina que se planteen implementar en futuras investigaciones

5. RESULTADOS

5.1 Resultados machine learning

Los resultados obtenidos en la validación cruzada descrita en la sección 4.3 etapa 3 se pueden apreciar en la Tabla 2.

Tabla 2

Resultados obtenidos de la validación cruzada con k=5 iteraciones en el conjunto de entrenamiento

Capa	*Naive Bayes*	*Random Forest*	*Support Vector Machine*
1	80.00%	76.90%	76.90%
2	77.18%	77.46%	76.05%
3	79.66%	79.37%	78.24%
4	83.05%	81.07%	80.79%
5	78.24%	77.96%	78.53%
Promedio Exactitud	79.62%	78.55%	78.10%

Nota. La tabla muestra los resultados obtenidos en la validación cruzada con 5 iteraciones usando *Naive Bayes*, *Random Forest y SVM*. Se puede apreciar el porcentaje de exactitud en cada iteración y el promedio obtenido de las iteraciones. En este caso para *Naive Bayes*, *Random Forest* y *SVM* se está usando los parámetros por defecto.

Analizando los resultados de la validación cruzada se encontró que al utilizar *Naive Bayes* se tiene una exactitud promedio de 79.62% utilizando el conjunto de entrenamiento del *dataset* lo cual es un 6.38% menor que el porcentaje obtenido en la exactitud de los resultados obtenidos usando el conjunto de evaluación del *dataset* el cual fue de un 86% como se ve en al Tabla 3. En la tabla la palabra "negativo" hace referencia a los comentarios negativos y la palabra "positivo" hace referencia a los comentarios positivos.

Tabla 3

Resultados obtenidos del algoritmo de Naive Bayes usando el conjunto de evaluación

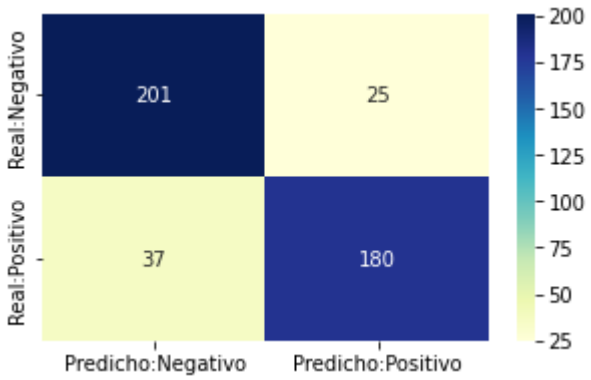
Categoría	Exactitud	Precisión	*Recall*
Positivo y Negativo	86%	86%	86%
Negativo	-	84%	89%
Positivo	-	88%	83%

Nota. La tabla muestra los resultados obtenidos de *Naive Bayes*. Se puede apreciar el porcentaje de exactitud, precisión y *recall*. Fuente: elaboración propia.

Respecto a los resultados obtenidos en la investigación con el algoritmo de *Naive Bayes*, se puede decir lo siguiente: la tabla cuenta con tres apartados, exactitud, precisión y *recall*. Exactitud hace referencia a la división del número total de predicciones correctas / número total de predicciones. En este caso dio un resultado de 86% por lo que se puede decir que se obtuvo un resultado adecuado. Para las columnas de precisión y *recall* que están en la fila de *Naive Bayes* se utilizó el *macro average* que es la media aritmética de todas las puntuaciones de precisión y *recall* de diferentes clases; para la precisión se basa en el porcentaje de resultados relevantes, es decir en el número de verdaderos positivos / (verdaderos positivos + falsos positivos). En el caso de los comentarios negativos se obtuvo una precisión de 84% y para los comentarios positivos se obtuvo una precisión de 88% y de manera general se obtuvo un 86%; en el *recall* se calcula de la siguiente manera el número de verdaderos positivos / (verdaderos positivos + falsos negativos) en esta ocasión con 89% en comentarios negativos y 83% en comentarios positivos y de manera general se obtuvo un 86%. El número de Falsos positivos (FP), Falsos negativos (FN), Verdaderos positivos (VP), Verdaderos negativos (VN) se puede apreciar mejor en la Figura 3.

Figura 3

Matriz de confusión usando Naive Bayes



Nota. La imagen muestra el número de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos del algoritmo *Naive Bayes*.

Analizando los resultados de la validación cruzada se encontró que al utilizar *Random Forest* se tiene una exactitud promedio de 78.55% utilizando el conjunto de datos de entrenamiento lo cual es un 3.16% menor que el porcentaje obtenido en la exactitud de los resultados obtenidos usando el conjunto de evaluación el cual fue de un 81.71% como se ve la Tabla 4.

Tabla 4

Resultados obtenidos del algoritmo de Random Forest usando el conjunto de evaluación

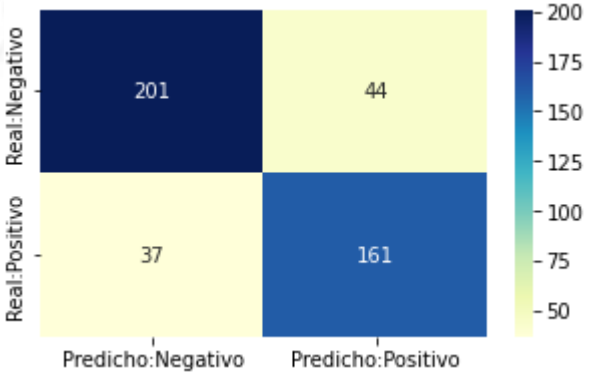
Categoría	Exactitud	Precisión	*Recall*
Positivo y negativo	81.71%	81%	82%
Negativo	-	84%	82%
Positivo	-	79%	81%

Nota. La tabla muestra los resultados obtenidos de *Random Forest*. Se puede apreciar el porcentaje de exactitud, precisión y *recall*. Fuente: elaboración propia.

En base a los resultados obtenidos en la investigación con el algoritmo de *Random Forest*, se puede decir lo siguiente: la tabla cuenta con tres apartados, exactitud, precisión y *recall*. En exactitud se dio un resultado de 81.71% por lo que se puede decir que se obtuvo un resultado adecuado. En el caso de los comentarios negativos se obtuvo una precisión de 84% y para los comentarios positivos se obtuvo una precisión de 79% y de manera general se obtuvo un 81%; Para *recall* se obtuvo un 82% en comentarios negativos y 81% en comentarios positivos y de manera general se obtuvo un 82%. El número de Falsos positivos (FP), Falsos negativos (FN), Verdaderos positivos (VP), Verdaderos negativos (VN) se puede apreciar mejor en la Figura 4.

Figura 4

Matriz de confusión usando Random Forest



Nota. La imagen muestra el número de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos del algoritmo *Random Forest*.

Analizando los resultados de la validación cruzada se encontró que al utilizar *Support Vector Machine* se tiene una exactitud promedio de 78.10% utilizando el conjunto de entrenamiento lo cual es un 2.9% menor que el porcentaje

obtenido en la exactitud de los resultados usando el conjunto de evaluación el cual fue de un 81% como se ve en al Tabla 5.

Tabla 5

Resultados obtenidos del algoritmo de Support Vector Machine usando el conjunto de evaluación

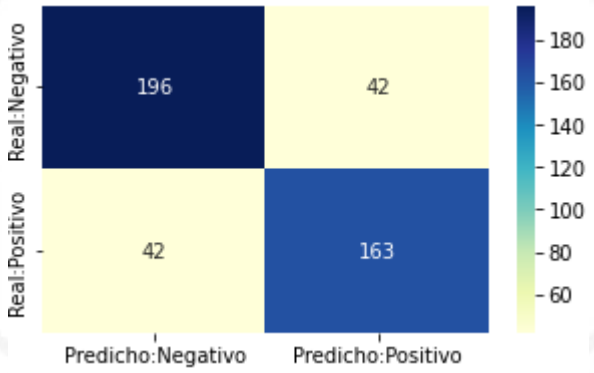
Categoría	Exactitud	Precisión	*Recall*
Positivo y negativo	81%	81 %	81%
Negativo	-	82%	82%
Positivo	-	80%	80%

Nota. La tabla muestra los resultados obtenidos de *Support Vector Machine*. Se puede apreciar el porcentaje de exactitud, precisión y recall. Fuente: elaboración propia.

De los resultados obtenidos en la investigación con el algoritmo de *Support Vector Machine*, se puede decir lo siguiente: la tabla cuenta con tres apartados, exactitud, precisión y *recall*. La exactitud dio un resultado de 81% por lo que se puede decir que se obtuvo un resultado adecuado. En el caso de los comentarios negativos se obtuvo una precisión de 82% y para los comentarios positivos se obtuvo una precisión de 80% y de manera general se obtuvo un 81%; Para *recall* en esta ocasión se obtuvo un 82% en comentarios negativos y 80% en comentarios positivos y de manera general se obtuvo un 81%. El número de Falsos positivos (FP), Falsos negativos (FN), Verdaderos positivos (VP), Verdaderos negativos (VN) se puede apreciar mejor en la Figura 5.

Figura 5

Matriz de confusión usando Support Vector Machine



Nota. La imagen muestra el número de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos del algoritmo *Support Vector Machine*.

Luego de haber obtenido los resultados de exactitud mostrados anteriormente de los tres algoritmos escogidos, se utilizó dos técnicas para buscar obtener mejores valores junto con los mejores parámetros para cada uno de los algoritmos utilizados. Se utilizaron dos técnicas para realizar esta los cuales fueron *Grid Search* y *Random Search*. La primera técnica llamada *Grid Search* se puede definir como un método para buscar valores de parámetros óptimos de un conjunto dado de parámetros de cuadrícula que utiliza a su vez validación cruzada. Se requiere entrada de modelo y parámetro, se extrae los mejores valores de parámetros y luego realiza predicciones. La segunda técnica llamada *Random Search* implementa los métodos de "ajuste" y "predicción" como cualquier clasificador, excepto que los parámetros del clasificador que se usa para la predicción se optimizan mediante la validación cruzada. A continuación, se puede apreciar en la Tabla 6 y 7 los resultados obtenidos de dichas técnicas.

Tabla 6

Resultados usando Grid Search para la identificación de los mejores hiperparámetros

GridSearchCV	Los mejores parámetros	El Score obtenido del conjunto de entrenamiento de la comparación de la predicción de trainX contra trainY	Score más alto en el conjunto de evaluación
Naive Bayes	alfa = 0.9; class_prior = ninguno; fit_prior = verdadero	80.19%	85.55%

Random Forest	max_depth = ninguno;
	min_samples_leaf = 1;
	min_samples_split = 4;
	n_estimators = 100;
	random_state = 63760142
		78.21%	82.16%

Support Vector Machine	C = 10;
	gamma = 0.01;
	kernel = rbf;
	random_state = 63760142
		78.10%	83.29%

Nota. La tabla muestra los resultados obtenidos en base a la técnica de *GridSearchCV* utilizado en la investigación para los tres algoritmos de *machine learning*.

Tabla 7

Resultados obtenidos Randomized Search para la identificación de los mejores hiperparámetros

RandomizedSearchCV	Los mejores parámetros	El Score obtenido del conjunto de entrenamiento de la comparación de la predicción de trainX contra trainY	Score más alto en el conjunto de evaluación

Naive Bayes	fit_prior = verdadero;
	class_prior = ninguno;
	alfa = 0.5
		79.68%	84.87%

Random Forest	random_state = 63760142; n_estimators = 150, min_samples_split = 2, min_samples_leaf = 3, max_depth = None	76.63%	81.03%

Support Vector Machine	random_state = 63760142;
	kernel= sigmoid;
	gamma = 0.1;
	C = 1
		77.25%	80.81%

Nota. La tabla muestra los resultados obtenidos en base a la técnica de *RandomizedSearchCV* utilizado en la investigación para los tres algoritmos de *machine learning*.

En la tabla 6 y 7 los parámetros para *Naive Bayes* significan lo siguiente. Fit_prior = verdadero/falso, si es verdadero aprenderá las probabilidades previas de la clase, caso contrario si es falso se refiere a que se usará un prior uniforme, class_prior = ninguno, significa que el prior se ajusta a los datos. Para *Random Forest*, random_state se refiere la semilla utilizada para para el generador aleatorio para que podamos asegurarnos de que los resultados que obtengamos se puedan reproducir; min_samples_plit representa la cantidad mínima de muestras necesarias para dividir un nodo interno; min_samples_leaf especifica el número mínimo de muestras requeridas para estar en un nodo hoja, es decir que no tenga hijos y le max_depth es el número de divisiones que cada árbol de decisión puede realizar. Si el número de divisiones es demasiado bajo, el modelo se ajusta por debajo de los datos y si es demasiado alto, el modelo se ajusta por encima y al tener *None* significa que los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos del min_samples_split establecido. Para *Support Vector Machine*, kernel hace referencia a que se utiliza debido a un conjunto de funciones matemáticas utilizadas en la máquina de vectores de apoyo que proporciona la ventana para manipular los datos; gamma básicamente controla la distancia de influencia de un punto de entrenamiento. Un valor gamma bajo indica un radio de similitud más grande, lo que da como resultado más grupos de puntos. Para valores altos de gamma, los puntos deben estar muy cerca uno del otro para ser considerados en el mismo grupo y C es el parámetro de penalización del término de error. Se puede considerar como el grado de corrección de la clasificación que debe cumplir el algoritmo, o el grado de optimización que debe cumplir el algoritmo de *SVM*.

5.2 Resultado *BERT*

Los resultados obtenidos de las iteraciones realizadas descrito en la sección 4.3.2 etapa 5 se pueden apreciar en la Tabla 8.

Tabla 8

Resultados obtenidos de las 5 iteraciones en el conjunto de evaluación

Etapa	*BERT*
1	85.10%
2	85.10%
3	87.58%
4	85.32%
5	86.90%
Promedio Exactitud	86.00%

Nota. La tabla muestra los resultados obtenidos en las 5 iteraciones en el conjunto de evaluación de *BERT*. Se puede apreciar el porcentaje de exactitud en cada iteración y el promedio obtenido de las iteraciones.

5.2Resultados *BERT*

Analizando los resultados del entrenamiento se encontró que al utilizar *BERT* se tiene una exactitud de 99.71% utilizando el conjunto de entrenamiento del *dataset* lo cual es un 12.81% mayor que el porcentaje obtenido en la exactitud de los resultados obtenidos usando el conjunto de evaluación del *dataset* el cual fue de un 86.90% como se ve en al Tabla 9. En la tabla la palabra “negativo” hace referencia a los comentarios negativos y la palabra “positivo” hace referencia a los comentarios positivos.

Tabla 9

Resultados obtenidos del modelo BERT usando el conjunto de evaluación

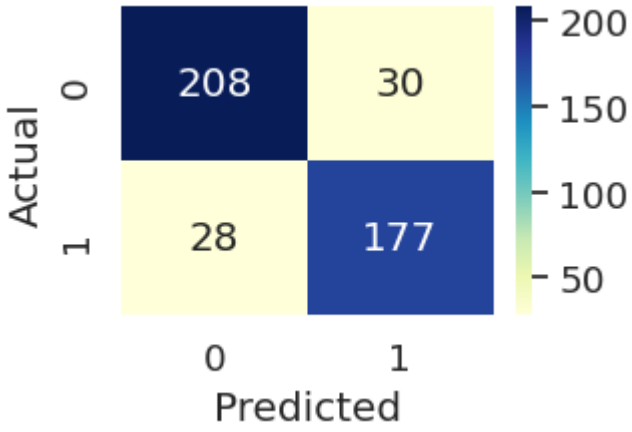
Categoría	Exactitud	Precisión	*Recall*
Positivo y negativo	86.90%	87%	87%
Negativo	-	88%	87%
Positivo	-	86%	86%

Nota. La tabla muestra los resultados obtenidos de *BERT*. Se puede apreciar el porcentaje de exactitud, precisión y *recall*. Fuente: elaboración propia.

De los resultados obtenidos en la investigación con el algoritmo de *BERT*, se puede decir lo siguiente: la tabla cuenta con tres apartados, exactitud, precisión y *recall*. La exactitud dio un resultado de 86.90% por lo que se puede decir que se obtuvo un resultado adecuado. En el caso de los comentarios negativos se obtuvo una precisión de 88% y para los comentarios positivos se obtuvo una precisión de 86% y de manera general se obtuvo un 87%; Para *recall* en esta ocasión se obtuvo un 87% en comentarios negativos y 86% en comentarios positivos y de manera general se obtuvo un 87%. El número de Falsos positivos (FP), Falsos negativos (FN), Verdaderos positivos (VP), Verdaderos negativos (VN) se puede apreciar mejor en la Figura 6.

Figura 6

Matriz de confusión usando BERT



Nota. La imagen muestra el número de falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos del algoritmo *BERT* siendo 1 los positivos y 0 los negativos.

6. DISCUSIÓN

En este apartado se discutirá los resultados obtenidos de los tres algoritmos de *machine learning* implementados, los cuales son *Naive Bayes*, *Random Forest* y *Support Vector Machine* y los resultados del transformador *BERT*. Se discutirá los resultados obtenidos, además de mencionar los parámetros que más influencia tuvieron en los algoritmos, como también se comparará los resultados obtenidos con trabajos previos recopilados en el estado del arte.

Respecto a los resultados obtenidos de la validación cruzada de cada uno de los algoritmos de *machine learning* implementados se puede apreciar que de las 5 iteraciones que se hicieron el valor más alto obtenido en *Naive Bayes* fue de 83.05% mientras que en *Random Forest* fue de 81.07% y en *SVM* fue de 80.79%. El promedio obtenido del total de las 5 iteraciones muestra que en teoría *Naive Bayes* tiene un porcentaje de 79.62% de promedio, *Random Forest* cuenta con 78.55% y *SVM* cuenta con 78.10%, pero si esos valores se comparan con el obtenido fuera de la validación cruzada en la etapa de evaluación se aprecia que los valores cambian con una diferencia notable, en el primero se obtuvo un porcentaje de 86%, el segundo obtuvo un porcentaje de 81.71% y el tercero obtuvo 81%. Esta notable diferencia podría ser el resultado de la manera en que los algoritmos tratan la información del dataset utilizado, pues cada algoritmo lo trabaja de manera diferente.

En el caso de *Naive Bayes* calcula el resultado en base a la probabilidad de un evento previo donde se le asigna a ese texto una etiqueta en nuestro caso sería si ese texto (palabra) pertenece al grupo de negativo o al grupo de positivos, por el otro lado *Random Forest* utiliza una series de árboles de clasificación, donde a cada árbol se le da el derecho de votar de a qué clase pertenecería dicho dato y en base a la combinación de estos resultados se produce la respuesta de a qué grupo pertenece y en el caso de *SVM* utiliza los datos de entrenamiento etiquetados y genera un hiperplano óptimo en base a los datos recibidos durante el entrenamiento que clasifica los nuevos datos que ingresen, donde el hiperplano es una línea que divide un plano en dos partes donde cada clase se encuentra en cada lado. Los parámetros que más afectaron a cada uno de los modelos en *Grid Search* son: *Naive Bayes* es el parámetro de alfa hace que los resultados varíen si es que se toma un valor u otro aparte de ser el valor que cambió en comparación con sus valores por defecto donde alfa es 1.0; para *Random Forest* el parámetro como el min_Samples_Split igual a 4 es el que ha cambiado en comparación con su valor por defecto que era 2 respectivamente y para *SVM* los parámetros que cambiaron fueron C con un valor de 10 y gamma igual a 0.01 en comparación con sus valores por defecto que son 1 y *scale* respectivamente, *scale* usaba una función para calcular el valor de gamma, respecto al *kernel* en todas las iteraciones siempre dio como valor escogido el rbf.

A diferencia de *Naive Bayes*, en *Random Forest* se puede cambiar el estado del random, min_samples_split y la cantidad de árboles con los que trabajar, en *SVM* se puede moficar el C, el kernel y gamma. Además, tanto *Random Forest* como *SVM* cuentan con más parámetros para modificar que *Naive Bayes* así que estos factores también pudieron haber influido en el resultado obtenido tanto en la validación cruzada que usó el conjunto de entrenamiento del *dataset* como en la prueba que usó el conjunto de evaluación del *dataset* restante. En los resultados de la prueba se pudo observar que ambos algoritmos obtuvieron valores cercanos por lo que se esperaría que en la matriz de confusión no difiera mucho los valores obtenidos o que sigan el mismo patrón, pero en este caso es diferente debido a que si observamos la matriz podemos ver que respecto a los datos que son negativos y fueron predichos como positivos es decir los falsos positivos y los datos que son positivos, pero fueron predichos como negativos o sea los falsos negativos, hay menos falsos positivos que falsos negativos en *Naive Bayes*, pero en *Random Forest* ocurre lo

opuesto donde hay menos falsos negativos que falsos positivos y en *SVM* ocurre un empate donde son iguales dichos valores.

Además, se compararon los resultados obtenidos de los algoritmos probados en esta investigación contra los resultados de algunos artículos del estado del arte. Se aprecia, por ejemplo, en el trabajo de Wongkar y Angdresey, (2019) realizó un análisis de sentimiento sobre las opiniones de los usuarios de la plataforma de Twitter sobre los candidatos a la presidencia de indonesia en 2019, se pudo observar que obtuvieron con el algoritmo de *Naive Bayes* entre 80,9% y 80,1%; con *SVM* obtuvieron un 63,99% ambos algoritmos en exactitud de los resultados. En las diferencias aparte del tema de investigación, se puede notar que el *dataset* usado por ellos consistía en comentarios de los usuarios de la plataforma de Twitter entre enero y mayo del año 2019 la cantidad de registros de este *dataset* es de 443 comentarios clasificados entre positivos y negativos, en este trabajo además usaron el algoritmo de *KNN* el cual tuvo un resultado de 75.58% de exactitud. Además, en su gráfico de metodología los autores muestran que usan *text mining* el cuál difiere en algunos conceptos de *PLN*, por ejemplo, *PLN* es usado para comprender el lenguaje humano mediante el análisis del texto, el habla o la sintaxis gramatical por la cual la máquina es capaz de "comprender" la intención del mensaje, mientras que, el *text mining* se utiliza para extraer información de contenidos estructurados y no estructurados. Se centra en la estructura más que en el significado del contenido, por lo que identifica patrones y atributos de la data, utiliza indicadores estadísticos como la frecuencia de las palabras, los patrones de palabras y la correlación dentro de las palabras para explicar el texto.

En el trabajo de Baid et al., (2017) trató sobre análisis de reseñas de películas para saber su polaridad si eran reseñas positivas o no, los resultados obtenidos en exactitud fueron los siguientes: *Naive* Bayes con 81.4% y *Random Forests* con 78.65%. Aparte de que su tema de investigación, una de las diferencias notorias con este trabajo son el *dataset* utilizado el cual contiene 2000 registro de críticas creadas por los usuarios de la plataforma de IMDb (*Internet Movie Database*) está dividido en 1000 reseñas positivas como 1000 reseñas negativas. Otra diferencia es la metodología usada por ellos, pues hicieron uso de un software llamado WEKA, el cual es un sistema desarrollado en la Universidad de Waikato en Nueva Zelanda, que proveía una implementación de algoritmos de machine learning y procesamiento de texto. Asu vez, en la investigación su tercer algoritmo utilizado por ellos fue *KNN* el cual obtuvo un resultado de 55.30% de exactitud. En el trabajo de Nayak y Natarajan, (2016) que trata sobre la clasificación de las opiniones de los usuarios de la plataforma de Twitter sobre películas se obtuvo en precisión que *Naive Bayes* tuvo 89%, *SVM* 88% y *Random Forest* 85%. Aparte del tema de su investigación, otra diferencia que notamos fue que el dataset usado es una recopilación de tuits de en total 2000 registros divido en positivo y negativo (no se menciona si el dataset cuenta con la misma cantidad de opiniones positivas que negativas), la métrica usada por ellos fue la precisión y no la exactitud. Además, los autores usaron el algoritmo de Porter que se usa principalmente en el idioma inglés para realizar *stemming* y no usaron lemmatización. Si observamos los valores obtenidos de los trabajos anteriores se puede observar que el rango de valores no difiere mucho de los obtenido en la investigación ya que en nuestro caso se obtuvo en *Naive bayes* un 86%, en *Random Forest* un 81.71% y en *SVM* un 81% por lo que los resultados mostrados en esta investigación son similares a los mostrados en el estado del arte.

Viendo más allá de *machine learning* nos encontramos en el terreno de *deep learning* donde existen los transformadores. El transformador es una arquitectura para transformar una secuencia en otra con la ayuda de dos partes (codificador y decodificador), usa el mecanismo de autoatención, ponderando diferencialmente la importancia de cada parte de los datos de entrada. Es utilizado en los campos del procesamiento del lenguaje natural (*PLN*).

En el caso del transformador *BERT* se realizaron 5 iteraciones con los datos de evaluación. Se observó que de la primera iteración donde se obtuvo un 85.10% de exactitud se pudo elevar dicho valor hasta un 86.90% en la 5 iteración. Esta incrementó en el porcentaje de exactitud de cada iteración puede deberse al modo en que *BERT* trabaja, dónde a diferencia de los modelos direccionales que primero leen el inicio para llegar al final de un texto, en los transformadores no se requiere que el texto sea procesado en orden. Esa capacidad y el incremento en la mejora de la exactitud refleja como *BERT* se vuelve más competente en comprender el lenguaje a medida que se entrena repetidamente, gracias a que *BERT* es capaz de procesar todas las palabras de un texto al mismo tiempo. También, las variables declaradas influenciaron en la forma que el modelo trabaja, al definir un límite de palabras que puede aceptar de un comentario para evitar sobrecargar los recursos computacionales como también en cuantos comentarios va a estar conformado un lote para evitar sobrecargar la memoria. Con respecto a la matriz de confusión se pudo observar que *BERT* obtuvo mejores resultado que los algoritmos de *machine learning*, pues se puede apreciar cómo obtuvo mejores predicciones de los verdaderos negativos en comparación con *Naive bayes, Random Forest SVM*; y para los verdaderos positivos obtuvo mejores resultados contra *Random Forest SVM*, pero no contra *Naive Bayes*.

En el trabajo de Sousa et al., (2019) mencionan su uso de *BERT* en el problema de análisis de sentimiento de noticias financieras para mejorar la previsión del mercado de valores donde se explica de manera general la lógica que usaron y cómo fue el proceso de investigación con el modelo *BERT*. Al final de la investigación, los autores lo comparan con dos algoritmos de *machines learning* los cuales son *Naive Bayes* y *SVM*. Los resultados obtenidos de la investigación

fueron los siguientes: *Naive Bayes* obtuvo 61%, *SVM* obtuvo 62.4% y *BERT* 82.50% todos en el parámetro de exactitud, mostrando que el modelo de *deep learning* tenía mejor desempeño. El otro trabajo revisado fue de Alaparthi y Mishra, (2021) donde querían descubrir que método tenía mejor desempeño en el análisis de sentimiento usando como muestra las reseñas de películas de la base de datos de *IMDB*, luego de explicar el preprocesamiento de los datos y una descripción resumida de cada método los autores muestran los resultados obtenidos de su investigación, donde de los 4 métodos usados *BERT* es el que tuvo mejor resultado con un 92.31% en exactitud.

7. CONCLUSIONES

Este trabajo presentó un método para clasificar las opiniones de usuarios acerca de publicaciones de bancos de Perú sobre *phishing*. Se utilizó el procesamiento de lenguaje natural con el objetivo de hallar qué tan efectiva es está técnica para la clasificación de este tipo de comentarios. El *dataset* se obtuvo utilizando técnicas de *web scraping* para la recolección de comentarios de las publicaciones de Facebook. Para procesar los comentarios del dataset creado, se emplearon técnicas de preprocesamiento como *lowercase*, tokenización, lematización y *stopwords.* Los algoritmos utilizados fueron *Naive Bayes* (*NB*), *Random Forest* (*RF*)y *SVM* de *machine learning*, junto con el procesamiento de lenguaje natural (PLN) para la parte del preprocesamiento. En el lado de *deep learning*, se usó el transformador *BERT*. Se experimentó con un dataset creado debido a que no se encontró uno que cumpliera con los requisitos necesarios para la investigación. Se concluyó en base a los resultados obtenidos de la validación cruzada que el algoritmo con mejor resultado en la exactitud en promedio fue el de *Naive Bayes*, superando a Random Forest por 1.07% y a *SVM* por 1.52%. Sin embargo, tanto Random Forest como SVM podrían ser utilizados, ya que la diferencia es pequeña. En el caso de BERT, se concluyó que, de sus cinco iteraciones con el conjunto de evaluación, la quinta iteración obtuvo el mejor resultado con una exactitud del 86.90%, superior a los resultados obtenidos con el conjunto de evaluación de los algoritmos *Naive Bayes* con 86%, *Random Forest* con 81.71% y *SVM* con 81%

También se utilizaron dos técnicas para probar diferentes conjunto de parámetros en los algoritmos con el fin de encontrar los parámetros más adecuados que entreguen un mejor resultado a los ya obtenidos. Se mostró que hubo una pequeña mejora en los resultados después de volver a correr los algoritmos de *machine learning* con los nuevos parámetros. Al analizar las matrices de confusión, se encontró que Naive Bayes tuvo menos datos incorrectamente clasificados como falsos positivos. Por lo tanto, si el objetivo es clasificar los comentarios para tener el menor número de falsos positivos, se recomendaría utilizar *Naive Bayes*. En cambio, si se busca minimizar el número de falsos negativos, se utilizaría Random Forest en el caso de machine learning. Sin embargo, si se opta por deep learning, se recomendaría utilizar BERT, ya que este algoritmo obtuvo un menor número de falsos negativos en comparación con los tres algoritmos de machine learning.

Se recomienda el uso del algoritmo de *Naive Bayes* para la clasificación de comentarios mediante análisis de sentimiento. Entre los tres algoritmos evaluados, este fue el que obtuvo los mejores resultados en el caso de optar por algoritmos de machine learning. Sin embargo, si se busca el mejor resultado posible, la opción ganadora sería el transformador BERT. Además, en el contexto de machine learning, se sugiere el uso de la validación cruzada para evaluar y comparar los resultados de los algoritmos utilizados. También se recomienda la aplicación de técnicas como Grid Search o Random Search, ya que ambos son útiles para determinar los hiperparámetros del modelo. Estas técnicas indican qué parámetros serían los más adecuados para obtener el mejor resultado según la métrica seleccionada. Como trabajo futuro, se plantea la posibilidad de agregar más clases de sentimiento para la clasificación, así como aumentar el tamaño del conjunto de datos para verificar si se observa una mejora en los valores obtenidos.

REFERENCIAS

(N.d.). Apwg.org. Recuperado el 1 de mayo, de 2022, de https://acortar.link/9u6bBu.

(N.d.). Ipsos.com. Recuperado el 9 de octubre de 2023, de https://www.ipsos.com/es-pe/banca-digital-2021

(N.d.). Ipsos.com. Recuperado el 14 de abril de 2024, de https://www.ipsos.com/es-pe/y-si-ya-no-voy-al-banco-las-ventajas-de-la-banca-digital

Abdillah, R., Shukur, Z., Mohd, M., & Murah, T. M. Z. (2022). Phishing classification techniques: A systematic literature review. *IEEE access: practical innovations, open solutions*, *10*, 41574-41591. https://doi.org/10.1109/access.2022.3166474

Abedin, N. F., Bawm, R., Sarwar, T., Saifuddin, M., Rahman, M. A., & Hossain, S. (2020). Phishing attack detection using machine learning classification techniques. 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 1125-1130. https://doi.org/10.1109/ICISS49785.2020.9315895

Alaparthi, S., & Mishra, M. (2021). BERT: A sentiment analysis odyssey. *Journal of Marketing Analytics*, *9*(2), 118-126. https://doi.org/10.1057/s41270-021-00109-8

Aleroud, A., & Zhou, L. (2017). Phishing environments, techniques, and countermeasures: A survey. *Computers & Security*, *68*, 160-196. https://doi.org/10.1016/j.cose.2017.04.006

Alkawaz, M. H., Steven, S. J., & Hajamydeen, A. I. (2020). Detecting phishing website using machine learning. 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), 111–114. https://doi.org/10.1109/cspa48992.2020.9068728

Aydin, M., & Baykal, N. (2015). Feature extraction and classification phishing websites based on URL. 2015 IEEE Conference on Communications and Network Security (CNS), 769-770. https://doi.org/10.1109/CNS.2015.7346927

Baid, P., Gupta, A., & Chaplot, N. (2017). Sentiment analysis of movie reviews using machine learning techniques. *International Journal of Computer Applications*, *179*(7), 45-49.

Baj-Rogowska, A. (2017, December). Sentiment analysis of Facebook posts: The Uber case. In *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)* (pp. 391-395). IEEE. https://doi.org/10.1109/INTELCIS.2017.8260068

Bakliwal, A., Arora, P., Patil, A., & Varma, V. (2011, November). Towards Enhanced Opinion Classification using PLN Techniques. In Proceedings of the workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011) (pp. 101-107).

Behdenna, S., Barigou, F., & Belalem, G. (2018). Document level sentiment analysis: a survey. *EAI Endorsed Transactions on Context-aware Systems and Applications*, *4*(13). http://dx.doi.org/10.4108/eai.14-3-2018.154339

Clasificador Bayes ingenuo en programación R – Acervo Lima. (s. f.). Acervolima.com. Recuperado 12 de junio de 2022, de https://es.acervolima.com/clasificador-bayes-ingenuo-en-programacion-r/

Denisko, D., & Hoffman, M. M. (2018). Classification and interaction in random forests. Proceedings of the National Academy of Sciences, 115(8), 1690-1692. https://doi.org/10.1073/pnas.1800256115

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Gupta, B., Sharma, S., & Chennamaneni, A. (s. f.). Twitter sentiment analysis: An examination of cybersecurity attitudes and behavior research-in-progress. Aisnet.org. Recuperado 25 de abril de 2022, de https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1016&context=sigdsa2016

Hao, J., & Dai, H. (2016). Social media content and sentiment analysis on consumer security breaches. *Journal of Financial Crime*, *23*(4), 855-869. https://doi.org/10.1108/jfc-01-2016-0001

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. Science, 349(6245), 261-266. https://doi.org/10.1126/science.aaa8685

Hussainalsaid, A., Azami, B. Z., & Abhari, A. (2015). Automatic classification of the emotional content of URL documents using PLN algorithms. *Proceedings of the 18th Symposium on Communications & Networking*, 56-59.

Joseph, S. R., Hlomani, H., Letsholo, K., Kaniwa, F., & Sedimo, K. (2016). Natural language processing: A review. International Journal of Research in Engineering and Applied Sciences, 6(3), 207-210.

Korkmaz, M., Sahingoz, O. K., & Diri, B. (2020). Detection of phishing websites by using machine learning-based URL analysis. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-7. https://doi.org/10.1109/ICCCNT49239.2020.9225561

Kunju, M. V., Dainel, E., Anthony, H. C., & Bhelwa, S. (2019). Evaluation of phishing techniques based on machine learning. 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 963-968. https://doi.org/10.1109/iccs45141.2019.9065639

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, *5*(1), 1-167. https://doi.org/10.2200/s00416ed1v01y201204hlt016

Liu, Y., Wang, Y., & Zhang, J. (2012, September). New machine learning algorithm: Random forest. In International Conference on Information Computing and Applications (pp. 246-252). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-34062-8_32

Maynard, D., & Funk, A. (2012). Automatic detection of political opinions in tweets. En *Lecture Notes in Computer Science* (pp. 88-99). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-25953-1_8

Naqa, I., & Murphy, M. J. (2015). What is machine learning?. In machine learning in radiation oncology (pp. 3-11). Springer, Cham. https://doi.org/10.1007/978-3-319-18305-3_1

Nasrin, S., Ghosh, P., Chowdhury, S. M., Abujar, S., & Hossain, S. A. (2020). Fraud detection of Facebook business page based on sentiment analysis. In *Proceedings of International Joint Conference on Computational Intelligence* (pp. 279-287). Springer, Singapore. https://doi.org/10.1007/978-981-13-7564-4_25

Nayak, A., & Natarajan, D. (2016). Comparative study of naive Bayes, support vector machine and random forest classifiers in sentiment analysis of twitter feeds. *International Journal of Advance Studies in Computer Science and Engineering (IJASCSE)*, *5*(1), 16.

Othman, M., Hassan, H., Moawad, R., & Idrees, A. M. (2015). Using PLN approach for opinion types classifier.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, *2*(1-2), 1-135. https://doi.org/10.1561/1500000011

Peng, T., Harris, I., & Sawa, Y. (2018). Detecting phishing attacks using natural language processing and machine learning. *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, 300-301. https://doi.org/10.1109/ICSC.2018.00056

Phand, S. A., & Phand, J. A. (2017). Twitter sentiment classification using stanford PLN. 2017 1st International Conference on Intelligent Systems and Information Management (ICISIM), 1-5. https://doi.org/10.1109/ICISIM.2017.8122138

Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. In Machine learning (pp. 101-121). Academic Press.

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. Encyclopedia of database systems, 5, 532-538. https://doi.org/10.1007/978-1-4899-7993-3_565-2

Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. arXiv preprint arXiv:1701.07179.

Shahrivari, V., Darabi, M. M., & Izadi, M. (2020). Phishing detection using Machine Learning techniques. En arXiv [cs.CR]. http://arxiv.org/abs/2009.11116

Sharma, S., & Jain, A. (2020). Role of sentiment analysis in social media security and analytics. *Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery*, *10*(5). https://doi.org/10.1002/widm.1366

Sousa, M. G., Sakiyama, K., de Souza Rodrigues, L., Moraes, P. H., Fernandes, E. R., & Matsubara, E. T. (2019, November). BERT for stock market sentiment analysis. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 1597-1601). IEEE. https://doi.org/10.1109/ICTAI.2019.00231

Thelwall, M., Buckley, K., & Paltoglou, G. (2011). Sentiment in twitter events. *Journal of the American Society for Information Science and Technology*, *62*(2), 406-418. https://doi.org/10.1002/asi.21462

Thompson A. (2024, 5 febrero). *Digital 2024: 5 billion social media users - We Are Social Japan*. We Are Social Japan. https://acortar.link/58rIsA.

Toman, M., Tesar, R., & Jezek, K. (2006). Influence of word normalization on text classification. Proceedings of InSciT, 4, 354-358.

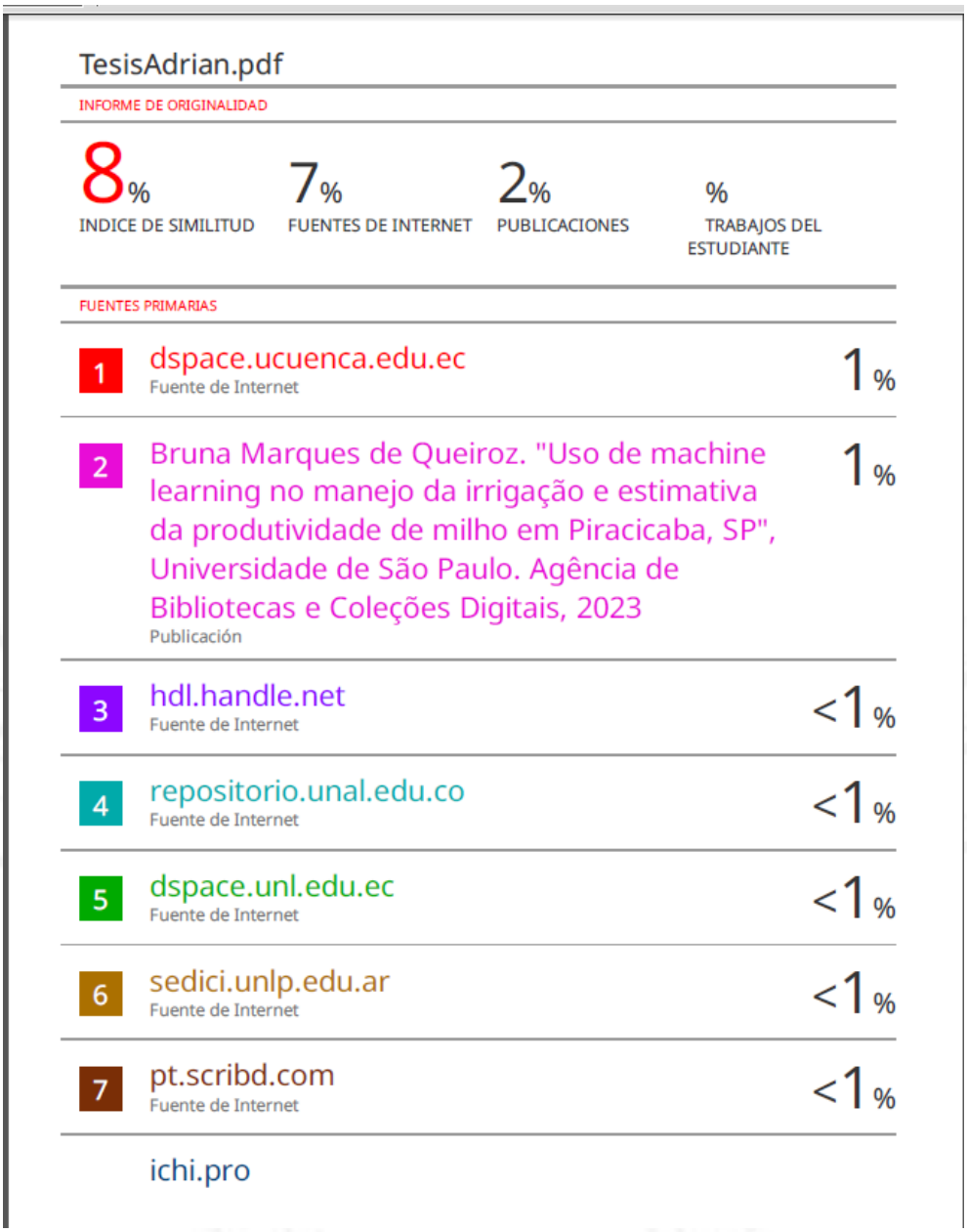
Tsytsarau, M., & Palpanas, T. (2012). Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, *24*(3), 478-514. https://doi.org/10.1007/s10618-011-0238-6

Webb, G. I., Keogh, E., & Miikkulainen, R. (2010). Naïve Bayes. *Encyclopedia of machine learning*, *15*, 713-714.

Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in PLN. In COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics.

Wenando, F. A., Hayami, R., Bakaruddin, & Novermahakim, A. Y. (2020). Tweet sentiment analysis for 2019 Indonesia presidential election results using various classification algorithms. 2020 1st International Conference on Information Technology, Advanced Mechanical and Electrical Engineering (ICITAMEE), 279-282. https://doi.org/10.1109/ICITAMEE50454.2020.9398513

Wongkar, M., & Angdresey, A. (2019, October). Sentiment analysis using Naive Bayes Algorithm of the data crawler: Twitter. In 2019 Fourth International Conference on Informatics and Computing (ICIC) (pp. 1-5). IEEE. https://doi.org/10.1109/ICIC47613.2019.8985884



TesisAdrian.pdf

INFORME DE ORIGINALIDAD

8%	7%	2%	%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	dspace.ucuenca.edu.ec Fuente de Internet	1%
2	Bruna Marques de Queiroz. "Uso de machine learning no manejo da irrigação e estimativa da produtividade de milho em Piracicaba, SP", Universidade de São Paulo. Agência de Bibliotecas e Coleções Digitais, 2023 Publicación	1%
3	hdl.handle.net Fuente de Internet	<1%
4	repositorio.unal.edu.co Fuente de Internet	<1%
5	dspace.unl.edu.ec Fuente de Internet	<1%
6	sedici.unlp.edu.ar Fuente de Internet	<1%
7	pt.scribd.com Fuente de Internet	<1%
	ichi.pro	