Juan Luis Castro Peña
Catedrático de Ciencias de la Computación
e Inteligencia Artificial.
Universidad de Granada

INTELIGENCIA ARTIFICIAL APLICADA A LA SEGURIDAD

INTELIGENCIA ARTIFICIAL APLICADA A LA SEGURIDAD

Resumen: En este artículo se realiza una revisión del estado del arte de las técnicas de Inteligencia Artificial más útiles para el desarrollo de aplicaciones en el ámbito de la seguridad. Nos centraremos en los aspectos que consideramos esenciales para entender las posibilidades, los problemas, las dificultades y las cosas a tener en cuenta para valorar los posibles proyectos concretos, y para abordarlos de la forma más eficaz.

Abstract: This paper aims to report the state-of-the-art of most useful Artificial Intelligence techniques for the development of applications in the field of security. We focused on those aspects that we consider essential to understand the possibilities, issues, difficulties and factors to take into account when assessing a specific project, in order to approach it in the most effective way.

Palabras clave: Inteligencia Artificial, Aprendizaje Automático, Aprendizaje Profundo, Aplicaciones de la IA, Seguridad

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Artificial Intelligence Applications, Security.

1.- INTRODUCCIÓN

En el campo de la *Inteligencia Artificial* (IA) estamos viviendo una tremenda explosión de interés científico y social que ha transcendido los mundos académico, científico o tecnológico, para llegar incluso a nivel de las conversaciones habituales diarias. Esto ha supuesto un importante incremento de los recursos dedicados a la aplicación de la IA. Son numerosas las empresas privadas y entidades públicas que están invirtiendo capital humano y financiero a fin de tratar de incorporar la IA en sus sistemas. Esto está suponiendo un muy positivo incremento de los desarrollos de aplicaciones de la IA en prácticamente todos los ámbitos. No obstante, en muchos casos no se está siguiendo de manera escrupulosa el rigor necesario para la correcta incorporación de los avances científicos y tecnológicos, lo que puede llegar a conllevar un sinfín de problemas.

Muchos de estos desarrollos los están llevando a cabo por equipos que no han estudiado con suficiente profundidad esta materia. La proliferación de gran cantidad cursos introductorios que no entran en los aspectos más profundos, está fomentando la aparición de lo que podríamos denominar "*expertos en IA a nivel de usuario",* que se encargan de la aplicación de esta técnica, sin la pertinente formación que se requiere para cualquier aplicación ingenieril. Algunos cursos son impartidos por empresas que ofertan servicios de herramientas para la aplicación de estas técnicas, y cuyo interés natural es que se contraten sus servicios. Otros cursos se ofertan como enseñanzas *no oficiales* de centros académicos que han encontrado en ese tema un buen mercado. En cualquier caso, por su duración y objetivos muchos de estos cursos no abordan aspectos relevantes para la correcta aplicación de la IA. De esta forma, se están utilizando herramientas avanzadas de IA para aplicarlas en problemas concretos por equipos que desconocen algunos puntos clave para una correcta aplicación de estas técnicas. Un caso típico es el uso de algoritmos de aprendizaje automático, utilizando las librerías que facilitan un fácil y rápido desarrollo, como las librerías de *Deep Learning* de Python o R, o las herramientas que ofrecen las grandes empresas de servicios como Google, Microsoft, Amazon o IBM. El problema es que se están aplicando estas técnicas y herramientas desconociendo aspectos esenciales del funcionamiento de las mismas para una correcta aplicación, como pueden ser significado y relevancia de los parámetros o alternativas para poder elegir los valores más adecuados en cada caso concreto, o los aspectos fundamentales a tener en cuenta para que el sistema aprenda correctamente, como el análisis de los conjuntos de datos (*datasets*) utilizado, y de los procedimientos de preprocesamiento, ajuste o validación. Se limitan a seguir manuales de uso que por su propia concepción y extensión no entran en estos temas. En esas circunstancias el resultado puede ser engañoso, como por ejemplo obtener modelos con un acierto muy elevado durante las pruebas de validación (incluso superior al 95%), pero que, cuando se aplican en el mundo real distan mucho de ni tan siquiera acercarse a esa efectividad.

Con esto se corre un doble peligro. Por un lado, cuando se aprecian los errores que el sistema tiene en el uso real (muchos más de lo que la validación de "laboratorio" pronosticaba), puede llevar a la decepción y la frustración, incluso al fracaso del producto, y el consiguiente bajo rendimiento de la inversión realizada. Un ejemplo ilustrativo es el caso de las numerosas herramientas que se desarrollaron para diagnosticar y pronosticar la evolución del COVID-19, para lo que se destinó una considerable cantidad de recursos durante 2019 y 2020. Los análisis que se han realizado de las mismas son bastante esclarecedores; ese es el caso del informe emitido por el prestigioso instituto de investigación *The Alan Turing Institue* (von Borzyskowski, 2020) relativo al impacto de

la respuesta de la comunidad de Ciencia de Datos e Inteligencia Artificial del Reino Unido a la lucha contra el COVID-19. Su conclusión es que, aunque hubo una amplia respuesta desarrollando numerosos proyectos, los resultados no habían sido realmente útiles, sugiriendo distintos aspectos a mejorar para que en una futura pandemia se pudiera obtener un resultado distinto. En la misma línea, en un artículo (Wynantsl, 2020) publicado en la revista *British Medical Journal* se han analizado 731 herramientas predictivas para el diagnóstico y el pronóstico del COVID-19, llegando a la conclusión de que ninguna es adecuada para el uso clínico. Tan solo dos de las propuestas analizadas se indican como suficientemente prometedoras como para ser consideradas en posteriores análisis. Un tercer trabajo exploratorio, (Roberts, 2021) publicado en la revista *Nature Machine Intelligence*, incide en este mismo sentido. Dicho artículo analiza 415 herramientas basadas en *Machine Learning* para predecir la evolución del paciente a partir de radiografías y TACs de tórax. La conclusión de nuevo es desalentadora: Ninguna se ha considerado adecuada para el uso clínico.

Pero puede ocurrir algo incluso peor. Que se acepte como correcto lo que el modelo aprendido diga, deformando nuestra visión de la realidad, puesto que se aceptarían como correctos los errores que el algoritmo comete. La controversia suscitada por el uso del sistema COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) puede ilustrar este punto. COMPAS es un sistema desarrollado por la empresa *Northpointe* para predecir la probabilidad de que alguien que ha cometido un delito vuelva a reincidir. El sistema se está utilizando en varios estados de los Estados Unidos, como Florida, California, o Nueva York entre otros, como sistema de ayuda para la toma de decisiones sobre cuestiones relacionadas con los acusados y los presos, tales como la prisión preventiva o la libertad condicional. El sistema está diseñado para predecir en concreto el riesgo de reincidencia en los siguientes dos años a partir de un amplio conjunto de factores, utilizando modelos de aprendizaje automático estadístico. Aceptada como una herramienta muy útil y fiable inicialmente, un estudio (Angwin, 2016) planteó algunas dudas sobre su funcionamiento. Este estudio analizó cómo había funcionado en un condado de Florida donde se utilizaba como ayuda para decidir sobre la prisión preventiva hasta la celebración de juicio. Se hizo un seguimiento del caso de 7214 acusados durante los años 2013 y 2014. El 39% de los acusados a los que el sistema había calificado de alta probabilidad de reincidencia en los siguientes dos años (se consideraba alta probabilidad cuando el sistema puntuaba por encima de 7 sobre 10) no habían reincidido durante ese periodo de tiempo. En prácticamente 4 de cada 10 casos, los positivos del algoritmo habían resultado ser falsos positivos, algo alejado de lo previsto según los estudios aportados por los desarrolladores. No obstante, lo realmente impactante del estudio era el sesgo que el algoritmo había mostrado sobre las personas de color. El 45% de los acusados de color que no reincidieron durante los siguientes dos años fueron calificados con una alta probabilidad de reincidencia, mientras que en el caso de los acusados blancos esto solo ocurrió en un 23% de los casos. El algoritmo utilizaba la raza como un factor determinante, algo que puede ocurrir pues los algoritmos de aprendizaje solo buscan un modelo que haga mínima una cierta función de error sobre los ejemplos con los que se entrena, y no entienden de razones éticas

Todo esto muestra la necesidad de realizar un planteamiento serio cuando se analizan las posibilidades de la aplicación de la IA a un problema concreto. La IA tiene un enorme potencial, y no podemos permitirnos el lujo de renunciar a él, al contrario, debemos tratar de aprovecharlo al máximo. Sin embargo, para ello hay que ser realistas en cuanto a lo que puede proporcionar, y bajo qué condiciones. Frente a las exageradas

expectativas que se están generado en la sociedad, es necesario que quienes tienen que decidir sobre las políticas y estrategias a seguir, o sobre si merece la pena abordar un proyecto concreto, sean conscientes de las excelentes posibilidades que hay, pero de forma realista, sopesando las diferentes cuestiones a tener en cuenta para lograr el éxito. Además, quienes desarrollen los sistemas deben ser rigurosos a la hora de aplicar las técnicas de la IA para obtener unos resultados adecuados. Difícilmente se puede ajustar un modelo de forma óptima si no se conoce cómo y por qué funciona, y por ende sus puntos clave, sus límites, las dificultades al aplicarlo y las posibilidades reales. Si no se tiene esto en cuenta, y no se aplican correctamente, se puede llegar a situaciones como la de los ejemplos comentados anteriormente, y así, desaprovechar en parte la inversión realizada. Esto es especialmente relevante en el caso de la aplicación de la IA al ámbito de la seguridad, en el que la fiabilidad de los resultados, y la justificación coherente de las decisiones tomadas son condiciones necesarias en muchas situaciones. Esto último obliga a incluir como requisito en algunos casos que estas decisiones deban ser explicadas y justificadas, lo que, a su vez, impone un análisis más profundo sobre las técnicas a aplicar. En este artículo vamos a revisar el estado del arte sobre las aplicaciones prácticas de la IA en seguridad, centrándonos en aspectos que deberían ser tenidos en cuenta por quienes la apliquen.

2.- ¿QUÉ ES LA IA?

El término Inteligencia Artificial surge en 1956 durante la conferencia *Dartmouth Summer Research Project on Artificial Intelligence*, donde fue definida como "la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cálculo inteligente". Los primeros años de esta ciencia se centraron en hacer que las máquinas pensaran como los humanos; de hecho, en 1985 Haugeland considera la IA como *"El nuevo y excitante esfuerzo de hacer que los ordenadores piensen ...máquinas con mentes en el más amplio sentido literal".* No obstante, si comparamos los ordenadores con el cerebro humano, ambos poseen unas características distintas en cuanto a la capacidad de cómputo. En algunos aspectos, esas características son ventajosas para los ordenadores, como en la velocidad de procesamiento y en la capacidad de utilizar y procesar una enorme cantidad de información. Sin embargo, en otros aspectos la capacidad de cómputo del cerebro humano aventaja a los ordenadores, como en el enorme número de procesadores que trabajan en paralelo (hasta 100 billones de neuronas), y el enorme número de parámetros que se pueden ajustar para aprender y adaptarse (cada neurona está conectada a unas 10.000 neuronas mediante una conexión sináptica que se van ajustando continuamente mediante el aprendizaje y la adaptación). Esto hizo que se planteara que la IA incorporase sistemas que piensan racionalmente, pero no necesariamente como lo hacen los humanos; así, Charniak y McDermott definen la IA como *"El estudio de las facultades mentales mediante el uso de los modelos computacionales".* Por otro lado algunos autores incidieron en que la IA debería de intentar que las máquinas realizaran cada una de las actividades que realiza el ser humano, considerando la IA como el desarrollo de sistemas que actúen como humanos: Rich y Knight, en 1991 definen la IA como "*El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor".* Se plantea que la IA debe abordar problemas complejos de forma inteligente, pudiendo actuar incluso mejor que los humanos, es decir como sistemas que actúen racionalmente para resolver un problema. Así, en 1998 Poole define la IA sin relacionarla con la inteligencia humana: *"La inteligencia computacional es el estudio del diseño de agentes inteligentes"*

3.- SISTEMAS DEDUCTIVOS

Inicialmente los modelos de IA que se plantearon eran sistemas deductivos, esto es, sistemas que deducen o razonan para tomar decisiones o resolver problemas considerados especialmente complejos. Se enfocó la investigación en el *Razonamiento Automático* (*Automatic Reasoning*) y en los modelos de *Representación del Conocimiento (Knowledge Representation)* necesario para ello. Se trata de modelos que permiten: a) definir formalmente un problema, b) representar el conocimiento necesario para resolverlo, y c) que incorporan *motores de inferencia* (*Inference Engine*) para implementan un razonamiento automático que busque la solución. Incluso se planteó la posibilidad de crear un sistema genérico que pudiera resolver cualquier problema (*General Problem Solver)*, pero el análisis de la realidad demostró que esto no era posible, y que había sido un planteamiento muy inocente, incluso iluso. Posiblemente en la actualidad estemos repitiendo esa actitud un tanto ilusa con el avance de los nuevos modelos de IA. En aquel momento se comprendió que había que centrarse en abordar problemas concretos, desarrollando *Sistemas Expertos (Expert Systems)* o Sistemas Basados en el Conocimiento (*Knowledge-Based Systems*) específicos para estos problemas.

Este tipo de sistemas tienen dos problemas: 1) que el conocimiento y razonamiento humano son en buena medida imprecisos; y 2) la enorme explosión combinatoria que el sistema tiene que explorar para encontrar la solución. Así, se han propuesto numerosos modelos para representar y razonar con *Incertidumbre (Uncertainty Models)*, como los modelos de *Razonamiento no Monótono*, (Non-Monotonic Reasoning), los de *Razonamiento Probabilístico (Probabilistic Reasoning)*, los de *Lógica Difusa (Fuzzy Logic)*, entre otros. Por otro lado, se han desarrollado distintas *Heurísticas* (Heuristic) y *Metaheurísticas (Metaheurísticas)* para intentar resolver el problema de encontrar una solución en un enorme espacio de búsqueda. Las principales dificultades que encontramos para aplicar este tipo de sistemas a un problema real son: a) la necesidad de representar una gran cantidad de conocimiento, con el consiguiente esfuerzo del equipo de desarrollo para adquirir ese conocimiento y representarlo y b) que el sistema deductivo sea capaz de encontrar de manera eficiente una solución en un espacio de búsqueda tan enorme. Podemos encontrar diversas técnicas para representar y reutilizar el conocimiento. Los *Sistemas Basados en Reglas (Rule-Based Systems)* del tipo si-entonces permiten representar de manera sencilla y entendible el conocimiento, pero son muy simples y necesitan una elevada cantidad de reglas para representar un conocimiento complejo. En el lado opuesto, las *Ontologías* (*Ontologies*) son modelos complejos, que requieren de un trabajo especializado para diseñarlas, pero tienen la ventaja de representar el conocimiento de manera reutilizable y comprensible tanto por máquinas como por personas. Ambos sistemas cuentan con algoritmos de razonamiento estándares con implementaciones bastante optimizadas.

Respecto a la otra dificultad, se han desarrollado diversas técnicas para tratar de encontrar una solución eficiente al problema de búsqueda en el enorme espacio de opciones, siendo habitual combinarlas para obtener sistemas más efectivos. Una técnica consiste en el diseño de *Sistemas Multi-agente (Multi-agent Systems)*; esto es, sistemas con varios agentes inteligentes, donde cada agente se encarga de una tarea específica menos compleja, y que interactúan y colaboran entre ellos de forma que ese trabajo colaborativo conjunto consiga resolver adecuadamente el problema. Estos sistemas reducen la complejidad computacional al dividir el problema en subproblemas y subtareas

de una menor complejidad. Además, permiten elegir para cada agente la técnica de IA más adecuada de acuerdo a la tarea que debe realizar, optimizando así el funcionamiento del sistema.

4.- SOLUCIONES APROXIMADAS

La explosión combinatoria hace imposible conseguir un algoritmo infalible de búsqueda, tanto por el tiempo como por el espacio necesario para hacer una búsqueda exhaustiva. Un problema tan formalmente simple como 3-SAT, que plantea decidir sobre la consistencia de un conjunto de cláusulas, incluso reducido a que cada cláusula solo contenga 3 literales, resultó ser el primer problema demostrado como NP-completo (Cook, 1971), y por tanto asumido como sin que pueda existir una solución algorítmica eficiente.

Por tanto, la IA no se puede plantear encontrar algoritmos que siempre acierten de manera infalible, puesto que, en cuanto el problema abordado sea realmente complejo no van a existir tales algoritmos; la IA se dedica a buscar algoritmos aproximados que acierten en el mayor número posible de casos o que proporciones soluciones cercanas a la óptima. Éste es un tema que es importante resaltar. No se va a poder encontrar soluciones perfectas o infalibles, porque en la inmensa mayoría de los casos los problemas abordados son NP-duros (Garey, 1979). Cuando un problema es NP duro, está demostrado que o bien no tiene solución eficiente (la opción asumida por la comunidad científica), o bien tendrían solución eficiente todos los problemas NP-completos (opción que se descarta porque a pesar de que estos problemas han sido y están siendo muy estudiados por toda la comunidad científica, nadie ha encontrado una solución eficiente para ninguno de ellos).

El hecho de que la IA solo pueda proporcionar soluciones aproximadas trae consigo que con cada propuesta sea fundamental incorporar estudios y análisis rigurosos de validación y evaluación, que midan el grado de bondad de la solución propuesta, y qué resultados se pueden esperar al aplicar esa solución. De hecho, con cada problema que aborda la IA se han ido estableciendo medidas estándares que son las que se usan para evaluar las soluciones propuestas. Cuando se van a aplicar técnicas de IA, resulta fundamental conocer esas medidas y ser conscientes de los resultados que los distintos algoritmos de IA opcionales que se pueden usar están obteniendo para ese tipo de problema. De esta forma, tendremos una estimación de lo que se puede esperar del desarrollo, y de cuáles serían las técnicas más adecuadas. Para obtener información rápida sobre algún aspecto relacionado con las técnicas de IA, como publicaciones, revistas, tecnologías, aplicaciones, investigadores, etc., la página web https://aitopics.org/ resulta una fuente de información y datos excelente.

5.- SISTEMAS INDUCTIVOS. APRENDIZAJE AUTOMÁTICO.

Además de intentar modelar un razonamiento o comportamiento racional, la IA también ha abordado la incorporación de la principal capacidad del cerebro humano con respecto a los sistemas computacionales, el aprendizaje y la adaptación. Se trata aquí de modelar el aprendizaje a partir de la experiencia. Lo habitual es desarrollar modelos a partir de un conjunto de datos o ejemplos, que habitualmente tendrá que ser muy grande para obtener buenos resultados.

Algunos sistemas utilizan la propia base de ejemplos como una componente del sistema predictivo. Podemos hablar así de *Modelos Basados en Ejemplos*. El algoritmo *k-NN* de los *k vecinos más cercanos (K-Nearest neighbor)* es bastante representativo de este tipo de sistemas. En este algoritmo se define una medida de distancia y se toma la decisión en función de la respuesta de los k ejemplos de la base de ejemplos más cercanos. Otro ejemplo son los sistemas de *Razonamiento Basado en Casos (Case-Based Reasoning)*, donde se obtiene la solución adaptando la del ejemplo de la base de ejemplos más cercano. Para obtener buenos resultados, este tipo de modelos requieren de una base de ejemplos que sea bastante representativa de los posibles casos de entrada, y en caso del K-NN y sus variantes, que además estén bien distribuidos. La elección de una adecuada función de distancia es fundamental, y requieren de una indexación de la base de ejemplos que reduzca la complejidad computacional de tener que revisar toda la base de ejemplos para resolver cada nuevo caso. Cuando la base de ejemplos es muy grande, el coste computacional del predictor es muy elevado, y esto obliga o bien a reducir el número de ejemplos que se use (por ejemplo mediante una selección de prototipos), o bien a estructurarlos para que la búsqueda de los ejemplos más cercanos se limite a un subconjunto de ejemplos. Una de las ventajas de este tipo de modelos es su capacidad de aprendizaje, pues el sistema iría mejorando al añadir nuevos ejemplos, aunque hay que tener en cuenta el aumento de la complejidad computacional que esto conlleva. Por ejemplo, si evaluamos el comportamiento e incorporamos los errores corregidos como nuevos ejemplos, el sistema irá aprendiendo de sus propios errores y mejorando con el uso. Por otro lado, el algoritmo es sensible a los errores de la base de ejemplos, pues un error puede afectar a todos los casos que estén cercanos por la función de distancia. Con todo K-NN sigue siendo un algoritmo que puede dar muy buenos resultados si se dispone de un conjunto de ejemplos adecuado y se realiza una adecuada indexación de la base de ejemplos (Nagarka, 2021); puesto que está incluido en prácticamente todas las librerías de aprendizaje automático se puede aplicar de manera rápida.

Otros sistemas no utilizan directamente la base de ejemplos en el sistema, sino que la usan para ajustar o aprender un modelo. Son los algoritmos de *Aprendizaje Automático (Machine Learning)*. El esquema general de estos algoritmos es el de definir modelos complejos, bien porque tienen muchos parámetros, bien porque tienen una estructura que hay que instanciar, o bien por ambas cosas a la vez. Estos modelos son ajustados mediante un algoritmo de aprendizaje que aprende el valor de los parámetros, y fija cada uno de los componentes de la estructura para hacer que el modelo responda lo más adecuadamente posible para ese conjunto de ejemplos.

6.- IMPORTANCIA DE LA BASE DE EJEMPLOS.

Teniendo en cuenta el esquema general del funcionamiento de estos algoritmos de aprendizaje podemos concluir varias consideraciones a tener en cuenta a la hora de aplicarlos correctamente (Dorfman, 2022):

a) Si queremos que el algoritmo sea capaz de aprender a resolver problemas complejos, necesitaremos que el modelo sea complejo, es decir que tenga un elevado número de parámetros o una estructura con bastantes componentes.

b) Si el modelo tiene un alto número de parámetros o componentes de la estructura, necesitaremos una enorme cantidad de ejemplos para poder ajustarlos correctamente. En cualquier caso, el número de parámetros y la cantidad de ejemplos tendrán que ir en consonancia para conseguir un correcto entrenamiento.

c) La calidad y distribución de los ejemplos es fundamental, pues el modelo se ajusta para responder bien a esos ejemplos. Si los ejemplos están sesgados o hay situaciones en las que se va a aplicar el modelo sin un suficiente número de ejemplos, el sistema aprendido incorporará ese sesgo y no funcionará correctamente en esas situaciones.

d) Se corre el peligro de sobre-aprendizaje, es decir que el sistema se especialice en responder bien a esos ejemplos con los que se ha entrenado, pero cuando se aplique a casos distintos no responda correctamente.

El análisis de una posible aplicación nos puede ilustrar lo mencionado. Supongamos que deseamos diseñar un sistema para detectar si un correo electrónico es *phishing,* y para ello queremos utilizar un modelo de IA mediante aprendizaje automático. Necesitaremos de un conjunto de ejemplos, tanto de casos positivos como negativos; es decir, un *conjunto de ejemplos* consistente en un conjunto de correos etiquetados cada uno como "no_phishing" o "phishing". Si solo contamos con 10 ejemplos de phishing y 100 de no_phising, y el algoritmo va a ajustar, por ejemplo, 200 parámetros, la probabilidad de que el modelo aprendido acierte al aplicarlo a otros casos es muy baja, pues se está ajustando el modelo con muy poca información en relación al número de parámetros. Seguramente, ajustando un modelo con 10 parámetros se obtendría bastante mejor resultado, pero la cantidad de datos con la que se está entrenando es muy pequeña y el modelo es demasiado simple, por lo que no cabe esperar unos buenos resultados. Por otro lado, aunque se tenga una gran cantidad de datos, si en el dataset el número de ejemplos de no_phishing es muy superior al número de ejemplos de phishing, es decir, si el conjunto no está balanceado, sino que hay muchos más casos negativos que positivos, para minimizar el error, el sistema se ajustará para no equivocarse en los casos más abundantes; tenderá a reducir los falsos positivos, pero se permitirán más errores en los casos de phishing, es decir, tenderá a tener más falsos negativos. Es una consecuencia de que, por defecto, los algoritmos de aprendizaje buscan reducir el error en el conjunto de ejemplos de entrenamiento, y no tienen en cuenta la distribución del mismo. Pero lo que más interesaría en este problema es no tener falsos negativos. Sería conveniente por tanto aplicar las técnicas de preprocesamiento de datos para casos no balanceados, y ponderar de forma distinta cada tipo de error. Supongamos a continuación que, para generar muchos ejemplos positivos utilizamos una IA del estilo de Chat GPT, generando de forma automática correos de phishing. Podríamos conseguir una enorme base de ejemplos, y con muchos positivos, pero la gran mayoría del mismo estilo, el de la IA utilizada para generarlos. El algoritmo de aprendizaje podría aprender un modelo bastante complejo, pues tendría una buena cantidad de ejemplos, pero estaría especializado en reconocer solo el phishing generado por esa IA, reconociendo la forma en que ésta genera los correos de phishing, pero fallaría bastante cuando tuviese que decidir sobre correos de phishing que no sean los generados de esa forma. Seguramente, esto pueden explica algunos de los resultados obtenidos por las herramientas estudiadas en (Roberts, 2021) para predecir la evolución del COVID-19 a partir de las radiografías y los TACs de tórax, y en cierta medida el sesgo encontrado en (Angwin, 2016) al analizar el funcionamiento del sistema COMPAS. En resumen, conseguir una adecuada base de ejemplos, tanto en número como en calidad, diversidad y distribución, y realizar un adecuado análisis previo, incluyendo una depuración y un correcto pre-pocesamiento (Baheti, 2023), va a ser fundamental para el éxito del modelo aprendido.

7.- PROCESO DE VALIDACIÓN.

Para evaluar cómo de bueno es un algoritmo de aprendizaje tenemos en cuenta distintas características y medidas que sirven para estimar cómo de bien se comportará el modelo aprendido (Breck et al., 2019). Esta estimación se realiza mediante el *proceso de validación* que prueba el algoritmo sobre casos con los que no se ha entrenado. Se suele entrenar con una parte de los ejemplos de los que se dispone, *ejemplos de entrenamiento*, y se dejan los restantes como *ejemplos de validación*, para ver cómo se comportará en los casos en los que no se ha entrenado. La división de los ejemplos de que se disponen entre entrenamiento y validación se suele hacer de manera aleatoria. Para que ese sorteo aleatorio no pueda distorsionar los resultados se suele hacer *validación cruzada*, dividiendo el conjunto de ejemplos en *n* trozos y repitiendo el proceso de ajuste del modelo *n* veces, utilizando cada vez uno de los trozos para validar. Se considera entonces el valor promedio obtenido en esos n procesos de aprendizaje realizados durante la validación cruzada.

En cuanto a las medidas, se calculan en función de la matriz de confusión. Lo ilustraremos para el caso de un problema de clasificación binario. En caso de más de dos clases utilizaríamos los conceptos de positivos y negativos por clase. Esta matriz recoge los valores VP de verdaderos positivos (ejemplos indicados como positivos por el algoritmo y etiquetados como positivos en la base de ejemplos), VN de verdaderos negativos, FP de falsos positivos (indicados como positivos por el algoritmo pero etiquetados como negativos en la base de ejemplos), y FN de falsos negativos. El número Total de ejemplos será por tanto Total = VP+FP+VN+FN, mientras que el número de ejemplos indicados como positivos por el algoritmo será iP = VP+FP, el de indicados como negativos por el algoritmo será iN = VN+FN, el número ejemplos realmente positivos será rP=VP+FN, y el de ejemplos realmente negativos será rN=VN+FP. Las medidas más comúnmente utilizadas son:

- Exactitud (*Accuracy*): Exactitud = (VP+VN) / Total = (VP+VN) / (VP+FP+VN+FN)
- Precisión (*Precision*): Precisión = VP / iP = VP / (VP+FP)
- Sensibilidad (*Recall*): Sensibilidad = VP / rP = VP / (VP+FN)
- Especificidad (*Specificity*): Especificidad = VN / rN = VN / (VN+FP)

Si en el conjunto de ejemplos hay un 90% de positivos y un 10% de negativos, el clasificador $Naive_o$ que no tiene en cuenta las características y se limita a clasificar siempre como la clase mayoritaria de la base de ejemplos, obtendría: Exactitud=0.9, Precisión=0.9, Sensibilidad=1 y Especificidad=0. Expresados en porcentaje, un 90% de aciertos totales, un 90% de los positivos detectados son correctos, un 100% de los positivos son detectados, y un 0% de los negativos son detectados. Para hacer una correcta evaluación de lo aprendido, conviene tener en cuenta los datos que obtendrían los algoritmos más simples (Veljanovska, 2017), como el algoritmo *Naive* Bayes, para el problema en cuestión.

8.- GENERALIZACIÓN, SOBRE-ENTRENAMIENTO Y ESPECIALIZACIÓN.

Un modelo de aprendizaje tendrá una buena *capacidad de generalización* cuando obtenga unas buenas medidas sobre los ejemplos de validación, con los que no ha sido entrenado. En un correcto entrenamiento estas deberían ser cercanas a las obtenidas con los ejemplos

de entrenamiento, y en ambos casos elevadas. Se considerará que el sistema ha aprendido cuando esas medidas estén significativamente por encima del algoritmo *Naive Bayes*.

Si el modelo tiene unas excelentes medidas sobre el conjunto de entrenamiento pero las medidas son pobres sobre el de validación, el sistema se habrá sobreentrenado (especializándose en acertar los ejemplos con los que se ha entrenado), pero no habrá aprendido a resolver el problema, pues no obtiene buenos resultados con los ejemplos con los que no se ha entrenado. Este es uno de los principales errores al aplicar los algoritmos de aprendizaje automático (Pothuganti, 2018). Se entrena el modelo sin tener en cuenta cómo se va comportando con los ejemplos de validación. El modelo cada vez obtiene mejores resultados con los ejemplos de entrenamiento, así que se deja entrenando durante mucho tiempo, pensando que así cada vez va aprendiendo más; pero al terminar el proceso de aprendizaje, cuando se aplica a los ejemplos con los que no se ha entrenado, el resultado es decepcionante, pues el sistema se ha sobre-entrenado: Se ha entrenado tanto para responder cada vez mejor con los ejemplos concretos de entrenamiento que falla con los restantes. Para evitarlo, hay que tener siempre en cuenta cómo se va comportando el modelo aprendido con los ejemplos de validación, y parar el aprendizaje cuando empiece a empeorar con estos ejemplos.

Una variante más sutil y, por tanto, más difícil de detectar es la *especialización* al conjunto global de ejemplos, no solo a los ejemplos de entrenamiento. Sería el caso que se produciría en el ejemplo del detector de phishing entrenado con ejemplos generados por una IA concreta. Como tanto los ejemplos de entrenamiento como los de validación tienen como característica común la forma en que esa IA genera el texto, el sistema podría utilizar esa característica para optimizar el modelo y no sería detectado por el proceso estándar de validación. Esto siempre puede ocurrir, pues los ejemplos se obtienen durante un periodo concreto, con un contexto y circunstancias concretas y, por tanto, tendrán características correspondientes a ese periodo. Sin embargo, cuando se aplica el modelo aprendido, se hace sobre un periodo posterior, donde algunas de esas características pueden haber cambiado. Se trata de un sesgo que siempre estará presente en algún grado. Para reconocer y medir en qué medida se ha producido esta especialización, se pueden diseñar procesos de validación posteriores, preferiblemente en un entorno real, o al menos recopilando nuevos ejemplos de diferentes contextos. Cuando, para un problema complejo, obtenemos un modelo con un porcentaje muy elevado de acierto (tanto en entrenamiento como en validación), por ejemplo por encima del 95%, deberíamos sospechar de una posible especialización a la base de ejemplos utilizada para entrenar, y plantear una validación posterior en un contexto real, o al menos diferente. Como este sesgo de especialización siempre ocurre en cierto grado, en general es conveniente incluir procesos de adaptación y ajuste del modelo que irían aprendiendo a corregir el mismo.

Hagamos el análisis de otra posible aplicación para ilustrar este tema. Supongamos que vamos a diseñar un sistema para detectar perfiles violentos en las redes sociales utilizando modelos de aprendizaje automático. Necesitaríamos disponer de un conjunto grande de ejemplos de perfiles etiquetados como violentos o no violentos. Éstos recogerían los contenidos de esos perfiles hasta el momento actual. Por tanto, en ellos se tratarán los temas que han estado de actualidad durante ese periodo y se usarán los términos y la jerga de ese periodo. Cuando apliquemos el modelo aprendido, seguramente habrán surgido nuevos temas, otros ya no serán de actualidad, y posiblemente se usarán algunos términos nuevos. Como el modelo ha aprendido a clasificar los perfiles de acuerdo a los ejemplos originales, no incorporará esa evolución temática y del lenguaje,

incluso puede que utilice algunos de los aspectos que ya no están tan presentes. Si se analizan las métricas sobre una base de casos actual, seguramente los resultados serán peores que las obtenidas en el estudio realizado durante el desarrollo. Ante un problema de este tipo, que evoluciona rápidamente, se hace muy necesario una evaluación posterior en un contexto real, y muy conveniente incorporar en nuestro sistema procesos de adaptación del modelo a la evolución del problema, por ejemplo mediante aprendizaje por refuerzo.

9.- MODELOS CLÁSICOS DE APRENDIZAJE AUTOMÁTICO

Hay muchos modelos que se han utilizado para el *Aprendizaje Automático (*Veljanovska, 2017). Como modelos clásicos podemos nombrar los *Árboles de decisión (Decision Tree)*, los bosques de decisión aleatorios (*Random Forest)*, o las *Máquinas de Vectores Soporte (Support Vector Machine o SVM)*

Los algoritmos de aprendizaje de árboles de decisión son algoritmos de aprendizaje no paramétrico, que aprenden una estructura de árbol para clasificar los ejemplos en función de los valores de las características. El árbol va dividendo el conjunto de ejemplos hasta que en el último nivel del árbol, las hojas, todos los ejemplos sean de la misma clase, o al menos haya una clase preponderante. Son algoritmos simples y de rápido aprendizaje, que obtendrían un gran resultado si se contase como ejemplos con todos los casos posibles. Además, tienen la ventaja de que son fáciles de interpretar, pues es equivalente a un sistema basado en reglas. La decisión que se realiza en cada hoja del árbol es equivalente a una regla, que resulta perfectamente interpretable siempre que tenga un número razonable de antecedentes. Un problema de los árboles de decisión es que son muy sensibles a los ejemplos, pues pequeñas variaciones en el conjunto de ejemplos producen árboles muy distintos. Con todo, el principal inconveniente es que su capacidad de generalización es muy reducida.

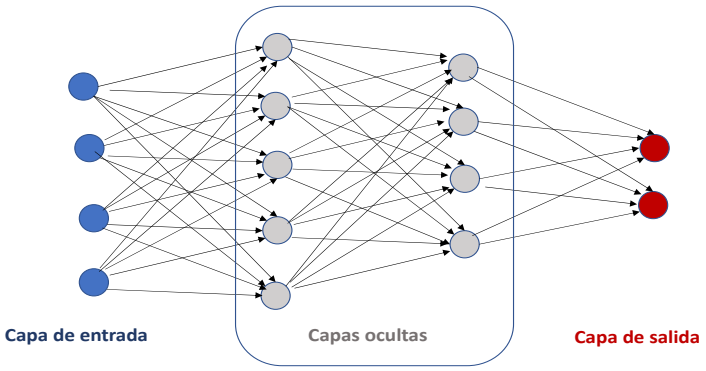
Los bosques de decisión aleatorios combinan la respuesta de múltiples árboles de decisión para obtener la decisión final. De esta forma se reduce el problema de la sensibilidad respecto de los ejemplos y se mejora la capacidad de generalización, a costa de obtener modelos más complejos, reduciendo la interpretabilidad del modelo.

Las máquinas de vectores soporte (SVM) desarrollan la idea de que cuando un espacio tiene una dimensión muy grande, los ejemplos van a estar muy separados y los clasificadores lineales obtienen muy buenos resultados. Lo que hacen es calcular el clasificador lineal de margen máximo que se obtendría al transformar el espacio de entrada en un espacio de altísima dimensión. En lugar de construir la transformación, que tendría un coste computacional inabordable, se utilizan unas *funciones núcleo (kernel*) para calcular la distancia entre los ejemplos en ese hipotético espacio, y así poder ajustar el clasificador. Como los clasificadores lineales solo dependen de los puntos más cercanos al hiperplano de regresión, solo depende de esos ejemplos, los llamados *vectores soporte*. Esto hace que el sistema resulte poco sensible a variaciones del conjunto de ejemplos, pues, en muchos casos, los vectores soporte no cambiarán al añadir o modificar ejemplos. Su principal ventaja es que pueden obtener buenos resultados en problemas de muy alta dimensionalidad, incluso aunque se disponga de muy pocos ejemplos (Soms, 2015). Aunque, en general, los modelos obtenidos con las SVMs no son interpretables, para ciertos núcleos, si se obtienen modelos interpretables (Castro, 2007). Por otro lado, pueden resultar muy sensibles a errores, si algún vector soporte es erróneo afectará mucho

al modelo. Además, su eficacia depende de la función *kernel* que se escoja y de la escala de los datos, por lo que conviene hacer un estudio con distintas funciones *kernel*, para así obtener los mejores resultados. Desgraciadamente, es habitual observar estudios donde, para comparar una propuesta se confronta con solamente la SVM que usa los parámetros por defecto de la librería que se esté usando, incluido el *kernel* por defecto.

10.- REDES NEURONALES

El modelo de aprendizaje automático más utilizado en las aplicaciones prácticas son las *Redes Neuronales Artificiales*, sobre todo por la capacidad de generalización respecto a los otros modelos. Uno de los primeros modelos de red neuronal fue el *Perceptron (*Rosemblat, 1956), que simula una red neuronal bastante simple, basada en el funcionamiento de una neurona biológica. Esta neurona artificial se conecta a las entradas del problema, que serían los estímulos de la neurona, y los parámetros a ajustar son los pesos de esas conexiones entre las entradas del sistema y la neurona. Estos pesos se usan para ponderar el estímulo transmitido por la correspondiente entrada. Un algoritmo de aprendizaje supervisado permite ajustar los pesos a partir del conjunto de ejemplos. El *Perceptron* despertó un gran interés sobre la posibilidad de desarrollar cerebros artificiales mediante modelos conexionistas, dando lugar al estudio como subárea de la IA de las redes neuronales artificiales. Pero un estudio sobre las capacidades y limitaciones del perceptrón (Minsky, 1969) demostró que este modelo solo era capaz de resolver problemas extremadamente simples, los problemas linealmente separables. Esto hizo que la investigación sobre las redes neuronales artificiales casi desapareciera. El siguiente impulso fue la aparición del algoritmo de *backpropagation* o retropropagación *de errores* (Rumelhart, 1986), que permitía entrenar sistemas con varias capas de perceptrones, conocido como *perceptrón multicapa, o red feedforward*, y que a la postre ha sido el modelo de red neuronal más usado. En la figura de abajo puede observarse la estructura de un perceptrón multicapa con 4 entradas y 2 salidas. Se ha diseñado con dos capas ocultas, la primera de 5 nodos y la segunda de 3 nodos. En el perceptrón multicapa los nodos de una capa están conectados con todos los nodos de la capa siguiente, de forma que si la capa t tiene n_t nodos y la capa t+1 tiene n_{t+1} nodos, habrá $n_t * n_{t+1}$ conexiones entre ambas capas, cada una con un peso que habrá que ajustar mediante el algoritmo de aprendizaje. Está demostrado que los *perceptrones multicapa* con solamente una capa oculta son aproximadores universales, tanto si no se utiliza una función de activación en la capa de salida (Hornik, 1989), como si, tal como es habitual, se utiliza esa función de activación (Castro, 2000). Esto supone que para cualquier problema cuya solución sea una función continua y con límites bien definidos, y para cualquier nivel de aproximación que se desee, siempre existe un *perceptrón multicapa* con solo una capa oculta que resuelve ese problema con ese nivel de aproximación. Este hecho, junto con que el algoritmo del *backpropagation* bajo una adecuada elección de parámetros de entrenamiento converge hacia un óptimo local, hizo del *perceptrón multicapa* un modelo muy usado en las aplicaciones de la IA, con una gran cantidad de aplicaciones a una gran cantidad de ámbitos. De hecho, cuando se indica que se aplica una red neuronal, sin especificar la estructura, se sobreentiende que se está utilizando un *perceptrón multicapa.*



Perceptrón multicapa con 2 capas ocultas de 5 y 4 nodos respectivamente

Las redes neuronales tienen una muy buena capacidad de generalización cuando se entrenan con un gran número de ejemplos. Hay que tener en cuenta el elevado número de parámetros que hay que ajustar, por lo que si no se cuenta con una gran cantidad de ejemplos difícilmente aprenderá correctamente. Tienen el inconveniente de que requieren una ingeniería de aplicación un tanto manual y por tanto costosa. No hay criterios claros para elegir el número de capas ocultas, ni el número de neuronas de cada capa, ni el valor de los parámetros del algoritmo de aprendizaje más adecuados. Además, al converger hacía un óptimo local hace que haya que ejecutarlo varias veces y quedarse con la mejor opción para tratar de obtener el óptimo global. De esta forma, para una correcta aplicación, hay que realizar muchas pruebas con distintos valores de las opciones, y elegir el modelo que obtenga el mejor resultado. Otros de los inconvenientes que inicialmente se resaltaron es que lo aprendido por las redes neuronales no era interpretable, aprendían a responder correctamente, pero no se podía justificar de una forma entendible para un ser humano el porqué de esa respuesta. En 1997 demostramos que los perceptrones multicapa con solo una capa oculta se podían traducir a un sistema basado en reglas difusas (Castro, 1997), aunque la equivalencia era como función matemática, y las reglas podían quedar fuera del dominio del problema. En 2002 encontramos una equivalencia donde las reglas quedaban dentro del dominio (Castro, 2002), lo que las hacía interpretables para el ser humano.

11.- APRENDIZAJE PROFUNDO

Tanto las SVMs como las redes neuronales pueden verse como sistemas que realizan una transformación compleja del espacio de entrada y, después, una clasificación lineal sobre ese espacio transformado. En el caso de las redes neuronales, todo el proceso hasta la última capa oculta se encargaría de la transformación, y la capa de salida de la clasificación lineal. En el caso de las SVMs la elección del núcleo se corresponde con la elección de una transformación compleja. En ambos casos, una parte de la transformación no es automática, sino que la elige el diseñador del sistema. Aunque en las redes neuronales los parámetros concretos de la transformación, es decir, los pesos de las conexiones que llegan a neuronas de las capas ocultas, también se ajustan mediante el algoritmo de aprendizaje, hay que realizar de forma manual la decisión del número de capas ocultas y del número de nodos de cada capa oculta.

La idea inicial del aprendizaje profundo es diseñar modelos que aprendan transformaciones para representar el espacio de entrada. Pero, en lugar de ser transformaciones especializadas en intentar resolver un problema concreto, se centran en intentar recoger las características más relevantes del espacio de entrada, en codificar el espacio de entrada. Se habla de aprendizaje profundo porque se diseña mediante múltiples capas, haciendo que la red final resulte muy compleja. Esta idea se ha utilizado para mejorar los resultados de aprendizaje y ha supuesto un salto espectacular en cuanto a las cosas que se pueden hacer mediante sistemas de IA, hasta el punto de que cuando en la actualidad se habla de un modelo de IA se sobreentiende que es un modelo desarrollado mediante aprendizaje automático y, más concretamente, mediante aprendizaje profundo.

En general, estos modelos utilizan la estructura codificador-descodificador (encoder-decoder). Se diseñan modelos paramétricos para codificar la entrada como un vector (encoder), junto con modelos para decodificar (decoder) ese vector y obtener los valores de entrada. Para entrenar esos modelos, no necesitamos tener datos etiquetados, por lo que podemos disponer de todos los datos que deseemos sin tener que realizar el esfuerzo de etiquetarlos, y así, hacer un entrenamiento masivo.

El caso más simple es el de los *autoencoders*, donde se entrena una red neuronal con una sola capa oculta para predecir la propia entrada (Dong, 2018). Los valores de las neuronas de la capa oculta es la codificación de la entrada, el codificador es el cálculo de estos valores utilizando las conexiones de la entrada a la capa oculta, y el decodificador consiste en el cálculo de los valores de salida a partir de los valores de la capa oculta. Un ejemplo de aplicación directa de este modelo es el caso de problemas para los que contamos con pocos ejemplos etiquetados en relación a la dimensionalidad de la entrada, pero podemos recopilar un gran número de casos sin etiquetar. Por ejemplo, se ha utilizado para detectar fraude en el uso de tarjetas de crédito (Lin, 2021). Para que la red pueda aprender bien necesitaremos que el problema tenga pocos parámetros que ajustar, de acuerdo al número de ejemplos etiquetados de que dispongamos. Podemos utilizar el procedimiento de autoencoder para ir reduciendo capa a capa la dimensión del espacio. Utilizamos autoencoder para codificar el espacio de la capa anterior a un espacio de dimensión menor. De esta forma, vamos generando capas ocultas que se entrenan paso a paso con un conjunto de posibles casos que no necesitan estar etiquetados, hasta obtener una última capa oculta de una dimensión acorde al número de ejemplos etiquetados de que dispongamos. Solo entrenaremos con los ejemplos etiquetados esa última parte de la red. La mejora de los resultados con respecto a ajustar todos los parámetros de la red completa con esos los ejemplos etiquetados suele ser muy significativa.

12.- PROCESAMIENTO DEL LENGUAJE NATURAL

Un caso donde el aprendizaje profundo ha supuesto un avance espectacular es el de los modelos del lenguaje natural. En este caso el espacio de entrada es un texto y el objetivo es entrenar un codificador que transforma una frase en un vector, o matriz numérica, que representa el significado de esa frase, para lo que se usan sistemas de codificación-decodificación. Como en el mundo se dispone de millones de textos, y se ha incrementado notablemente la capacidad de los ordenadores, se han podido entrenar diversos sistemas que han aprendido el significado, y que se han mostrado tremendamente efectivos, revolucionando en pocos años el procesamiento inteligente del lenguaje natural. El codificador *BERT* (Devlin, 2019), diseñado por Google, y que está disponible para ser

descargado y usado libremente, se ha convertido en un modelo estándar generalmente usado.

Inicialmente se introdujeron modelos de embedding, para codificar cada palabra como un vector, de forma que palabras cercanas en el espacio vectorial tenían una semántica similar. Por ejemplo, el modelo *Word2vec* (Mikolov 2013) se obtuvo entrenando una red con una capa oculta para predecir cuáles eran las palabras anteriores y posteriores a la palabra de entrada. El vector de las neuronas de la capa oculta era el vector que codificaba la palabra de entrada. Además de que las palabras con significado similar eran codificadas como vectores próximos, las operaciones con los vectores que codificaban las palabras tenían un significado semántico. Por ejemplo, mediante el cálculo (vector(mujer) – vector(hombre)) + vector(rey) se obtiene un vector cuyo punto con significado más cercano es vector(reina).

Posteriormente, se abordó la codificación de frases, secuencias de palabras, con redes neuronales recurrentes. En general, se tratan como secuencia de tolens, pues algunas palabras se descomponen en tokens mediante un proceso de "*tokenización*" (por ejemplo descomponiendo en raíz más terminación). La salida de la codificación del trozo de secuencia ya procesado se añade como entrada del codificador, que lo usa para obtener el código del siguiente token. Por eso, se habla de modelos con memoria, porque guardan en la memoria lo que han codificado hasta el momento como dato a tener en cuenta para realizar la siguiente codificación. De forma parecida se modela el proceso de descodificación. En la descodificación, además se incorporó el aprendizaje de la atención (attention model). La atención decide para cada token cuáles de los restantes tokens de la secuencia son relevantes para descodificar el mismo. De esta forma, se va aprendiendo simultáneamente como utilizar el contexto.

El siguiente paso adelante fue el uso de *Transformers* (Vaswani 2018). Los Transformers son modelos que toman como entrada un texto y genera un texto de salida mediante un proceso de codificación-descodificación. Su diseño permite de forma natural utilizarlos para traducción automática, introduciendo un texto de entrada y obteniendo la correspondiente traducción como salida. En los Transformers se puede realizar una codificación y descodificación en paralelo de los tokens, lo que ha resultado fundamental para poder entrenarlos con una enorme cantidad de datos. Para la codificación de cada token, se tiene en cuenta: a) la posición del token en la secuencia y, b) el resto de tokens de la secuencia, mediante un modelo de auto-atención (self-attention model). Los valores finales de la codificación se calculan mediante una red neuronal. Con el modelo de auto-atención se aprende a utilizar el contexto para codificar mejor. Para la descodificación se utiliza un proceso secuencial donde se va generando el siguiente token de salida a cada paso. Para generar el siguiente token se tienen en cuenta: a) la posición del token a generar, b) una atención sobre los tokens de entrada, y c) una auto-atención sobre los tokens ya generados. Para la descodificación final se utiliza de nuevo una red neuronal al que se añade una capa softmax. El resultado es que los transformers son modelos muy complejos, donde hay que ajustar muchos parámetros: los de la red neuronal de codificación, los del modelo de autoatención del codificador, los de los modelos de atención y auto-atención del descodificador, y los de la red neuronal de descodificación. El avance en el desarrollo de hardware con alta capacidad de procesamiento en paralelo, sobre todo de los procesadores de las tarjetas de video, ha permitido entrenar modelos de *transformers* cada vez más complejos. Además, muchas compañías ofrecen servicios de computación en la nube con herramientas para poder entrenar este tipo de modelos.

Cuando los modelos son muy complejos, la potencia de cómputo que se requiere es tan elevada que solo se puede realizar en los equipos de computación de altas prestaciones de compañías muy potentes, o en centros de supercomputación.

Como indicamos antes, el codificador que se ha convertido en estándar es *BERT,* desarrollado por Google*,* que se obtuvo entrenando un modelo de transformers. Se entrenó inicialmente utilizando una enorme cantidad de ejemplos obtenidos de la Wikipedia y Google books. Primero, se entrenó para aprender a generar la frase completa cuando se eliminaba una palabra; y, después, para aprender a decidir si una segunda frase era realmente la siguiente frase del texto. Se obtuvo un modelo con unos resultados espectaculares, tanto que actualmente es muy utilizado como base en cualquier problema con datos de lenguaje natural. Se parte del modelo general ya pre-entrenado, y se ajusta utilizando lo ejemplos para resolver el problema concreto, lo que se conoce como ajuste fino (*fine-tuning*). Incluso partiendo de un modelo pre-entrenado, si se diseña un sistema complejo para el *fine tuning*, también haría falta una alta capacidad computacional. El modelo final puede utilizarse en equipos sin tantos requerimientos, pero el aprendizaje del modelo habrá requerido de un gran esfuerzo computacional. Existen modelos pre-entrenados en más de 100 idiomas, y se pueden encontrar librerías de código abierto de Bert ya ajustados para muchos problemas de procesamiento del lenguaje natural.

Para ilustrar el salto en el desempeño que ha supuesto Bert, fijémonos en los resultados con el *dataset* SWAG. Éste es un *dataset* con 113.000 cuestiones para las que se ofrecen 4 alternativas. Cada cuestión es una descripción de una escena de video, las 4 opciones son descripciones de 4 posibles continuaciones de la escena, y la correcta es la que describe lo que realmente ocurre en el video. Es una tarea que requiere comprender el lenguaje y deducir por sentido común cuál de las 4 opciones sería la más razonable. Los mejores modelos previos a Bert no superaban el 60% de acierto, ni en entrenamiento ni en validación. Cuando se entrenó Bert, consiguió más de un 86%, tanto en entrenamiento como en validación. El acierto de un humano, al que se le pasaron 100 cuestiones fue del 85%.

13.- GENERACIÓN AUTOMÁTICA DE LENGUAJE

De forma similar a como se han desarrollado modelos generales que codifican el lenguaje natural, centrándose en la parte de codificación de los *transformers*, se han desarrollado generadores de lenguaje aprovechando la parte de descodificación. El modelo más conocido es el modelo GPT (*Generative Pre-Training,* Radford, 2018) de la empresa OpenAI , que en la actualidad ha desarrollado la versión GPT-4. La versión GPT-2 está disponible como código abierto, mientras que la versión GPT-3 (Brown 2020) sólo está disponible en forma de API.

El modelo se ha obtenido entrenando una estructura de *transformer*s para generar texto. El modelo de auto-atención del descodificador del transformer hace que el sistema se fije en los tokens generados que sean más relevantes para generar el siguiente token. Con esto se consigue la corrección sintáctica y la concordancia del texto. La potencia del modelo se basa en la complejidad de cada uno de los componentes del *transformer*, y en el tremendo entrenamiento llevado a cabo: OpenAi indica que GPT-3 es un modelo de 175 millones de parámetros y que para entrenarlo se han utilizado 700GB de textos. Es el modelo que se usa como base en Chat-GPT.

Chat-GPT incorpora algunos componentes sobre GPT-3. En primer lugar, se ha incorporado un *fine tuning* mediante aprendizaje supervisado para adaptar el modelo para distintas tareas en respuesta a una entrada de texto: hacer resúmenes, generar documentos, responder a cuestiones, etc. Esto ha dado lugar al modelo que OpenAI llama GPT-3.5. Además, incorpora una serie de estrategias alternativas para generar distintas respuestas para una misma entrada, de forma que se pueda ir adaptando la política a seguir para generar la respuesta. Chat GPT incorpora un sistema de aprendizaje por refuerzo para ir optimizando esa política y así generar cada vez mejores respuestas. Para que este aprendizaje se realice de forma automática se ha entrenado un modelo que calcula la recompensa de una respuesta ante una entrada. Para hacer esto se ha generado un conjunto de ejemplos cogiendo entradas, generando respuestas distintas según distintas estrategias, y evaluando manualmente cada respuesta. Una vez generado un conjunto grande de ejemplos de evaluación, se ha entrenado el modelo que se encarga de calcular automáticamente la recompensa. De esta forma el sistema se autoevalúa y optimiza la política a seguir para generar cada vez mejores respuestas.

14.- PROCESAMIENTO DE IMÁGENES

Otro problema al que se ha aplicado la IA y que ha experimentado un gran avance en los últimos años es el procesamiento de imágenes. Para ello se recurre a las *redes neuronales convolucionales (RNC),* las cuales han supuesto una revolución en el sector del reconocimiento de imágenes (Radzi, 2015). Las redes convolucionales usan copias de un mismo detector de características para distintas zonas de la imagen., mediante convolución con un filtro o kernel. Con esto se reduce el número de pesos que deben ajustarse, pues solo hay que ajustar los pesos del detector. Además, como se pueden utilizar distintos detectores se aprenden distintas características. Como contrapartida el número de características se eleva dando lugar a espacios de alta dimensionalidad. Para reducir esa dimensionalidad se suele combinar cada capa de convolución con una capa de *pooling*, que reduce la dimensión. Reiterando el proceso de una capa de convolución, seguida de una de *pooling,* se va generando una red cada vez más profunda que va utilizando características a más alto nivel. En la última capa una red neuronal se encarga de responder utilizando las características extraídas por las capas anteriores.

La red LeNet (Lecun, 1998) se puede considerar como la precursora de las redes convolucionales usadas actualmente. Fue diseñada para clasificar dígitos escritos a mano. La red tiene dos capas convolucionales con pooling, seguida de una red neuronal de 3 capas. Fue el primer gran éxito del uso de RNC. Posteriormente surgieron modelos cada vez más complejos. Alexnet (*Krizhevsky,* 2017) es una red con una estructura similar a Letnet pero con muchos más parámetros, 61 millones frente a los 62000 de Letnet. VGGNet (Simonyan 2015) usa bloques en lugar de simples capas de convolución, esto es, combinaciones de capas convolucionales que se aplican de forma repetitiva. Supuso una revolución pues los bloques permitían aprender características de alto nivel. Además, cuando se usan bloques, utilizando convoluciones basados en filtros pequeños se obtiene mejor resultado.

Una buena forma para medir el avance de las RNC en la clasificación de imágenes es la tasa de error que obtienen en el reto ImageNet. El objetivo es clasificar imágenes en una de 10.000 posibles categorías, usando 1.2 millones de imágenes para entrenamiento y 50.000 para validación. Se considera error solo si la clasificación correcta no está entre las 5 primeras indicadas por la red. En 2012 el menor error obtenido era del 26%, AlexNet

lo redujo a un 17% y VGGNet en sus sucesivas evoluciones a un 8%. El error del ser humano para este reto es del 5%.

Para abordar problemas de procesamiento de imágenes se suele utilizar como modelo pre-entenado la parte convolucional de un modelo tipo VGGNet, después de haberlo entrenado con los ejemplos de Imagenet. Ejemplos de estos modelos son VGG-16 y VGG-19. El 16 y 19 se refiere al número de capas de la red que se han pre-ajustado. El modelo final se obtiene añadiendo una red neuronal de clasificación y realizando un *fine tuning* mediante entrenamiento supervisado.

Un ejemplo típico de aplicación de estos modelos en el campo de la seguridad son los sistemas de reconocimiento facial o el reconocimiento de situaciones de alerta, que pueden ser utilizados para la identificación de sospechosos, la detección de alertas terroristas o el control de accesos entre otros.

15.- GENERACIÓN AUTOMÁTICA DE IMÁGENES

Las Redes Neuronales Generativas Adversarias (GAN, Generative Adversarial Networks, Goodfellow, 20014) son los modelos que se utilizan en la actualidad para generar imágenes artificiales. Se pueden utilizar para generar imágenes para sistemas de simulación, o para generar imágenes falsas para tratar de engañar. Las GAN confrontan dos redes neuronales, una "generadora" y otra "discriminadora". La red generadora produce muestras de aquello que queremos crear (imágenes, textos, sonidos…), mientras que la red discriminadora decide si lo creado por la red generadora se ajusta a lo que se está buscando. Es un proceso donde cada una de las redes va mejorando y aprende de su oponente. En la propuesta original, la red generadora toma como entrada ruido generado aleatoriamente, y genera como salida imágenes a partir de ese ruido. Para entrenarla, se considera la respuesta del discriminador para esas imágenes generadas. La medida en que el discriminador indica que las imágenes no son correctas es el error que se usa para corregir los pesos del generador. Para entrenar el discriminador, necesitamos de un conjunto de imágenes reales. Los ejemplos de entrenamiento del discriminador serán tanto las imágenes de ese conjunto de imágenes reales, que se etiquetan como correctas, como las imágenes del generador, que se etiquetan como incorrectas. Durante el entrenamiento, la mejora de una red sirve para mejorar la otra. Cuanto mejor sea el discriminador mejores imágenes tendrá que generar el generador para conseguir que éste no las considere incorrectas; y, cuanto mejor sea el generador mejor tendrá que ser el discriminador para distinguir las imágenes generadas de las reales.

Si en lugar de utilizar un generador de imágenes a partir de ruido aleatorio, se utiliza una entrada de un cierto tipo (una imagen, un texto, etc.), tendremos que el sistema aprende a traducir ese tipo de entrada en una imagen del tipo definido por el conjunto de imágenes reales. Por ejemplo, si utilizamos imágenes de personas como entrada del generador, e imágenes de animales en el conjunto de imágenes reales, una vez entrenado nuestro generador traducirá la imagen de una persona en la imagen de un animal asociado. Se han desarrollado algunas arquitecturas específicas para traducir imágenes de un tipo en imágenes de otro tipo, como por ejemplo DCLGan (Dual Contrastive Learning for Unsupervised Image-to-Image Translation, Han 2021).

16.- EL PROBLEMA DE LA INTERPRETABILIDAD

Como hemos indicado, el uso del aprendizaje profundo ha supuesto todo un salto hacia adelante en las distintas áreas de la IA en las que se ha aplicado, con mejoras notables en muy poco tiempo. Esto ha hecho que la sociedad, que ha percibido esa rápida mejora, crea que ese crecimiento se va a mantener hasta llegar a unos resultados que resultan un poco inverosímiles. Hemos de ser realistas. Este salto ha sido motivado por la aparición de esta técnica, la del aprendizaje profundo, y la mejora que aporte esa técnica tendrá un límite. Hemos de aprovechar esa mejora para sacarle el máximo provecho y obtener sistemas que nos ayuden en distintos problemas, es un reto realmente apasionante, pero debemos de ser conscientes de las dificultades y los límites que nos encontraremos. Por ejemplo, resulta inviable pensar que se puedan construir modelos del tamaño de los desarrollados para el lenguaje natural en otros ámbitos. Simplemente porque no se dispone, y no parece que se vaya a disponer, de un conjunto tan enorme de ejemplos como la cantidad de texto de que se ha dispuesto para para entrenar los modelos del lenguaje. Ante problemas concretos, resulta inimaginable que podamos disponer de una cantidad tan enorme de ejemplos. Esto no quiere decir que no se puedan abordar y obtener excelentes resultados, pero requieren de un esfuerzo y un conocimiento profundo sobre cómo funcionan los modelos para conseguir sacarle el máximo provecho a los ejemplos de que podamos disponer.

Por otro lado, los modelos de aprendizaje profundo, que han supuesto una mejora tan grande en efectividad, han tenido como coste el retroceso en cuanto a la interpretabilidad o explicabilidad del modelo. Los árboles de decisión, cuando la profundidad y el número de nodos no es muy elevado, se pueden traducir en reglas entendibles por un ser humano. Por tanto, se puede saber lo que realmente ha aprendido el modelo y en base a qué criterio decide. En el caso de las SVM, ya demostramos que para ciertas funciones de núcleo también (Castro, 2007), pero para la mayoría de los núcleos no se sabe cómo obtener una interpretación del modelo. En el caso de las redes neuronales, demostramos (Castro, 1997 y 2002) que los perceptrones multicapa con una capa oculta se podían interpretar como un conjunto de reglas difusas, en concreto una regla por cada neurona de la capa oculta. Pero en los modelos de aprendizaje profundo, con una gran cantidad de capas y cada capa de distinto tipo, se ha perdido esta interpretabilidad. Parece muy complicado conseguir una interpretación de las características obtenidas tras tantas transformaciones de distinto tipo del espacio. En definitiva, con el aprendizaje profundo aprendemos modelos muy complejos, capaces de predecir o clasificar bastante bien, pero no sabemos describir, de forma entendible por el ser humano, qué es lo que se ha aprendido, ni por tanto con qué criterio se predice o clasifica. En ese sentido, se dice que son modelos de caja negra. Actualmente, este es uno de los temas que más se está abordando por la comunidad científica, tratando de interpretar los modelos. Están surgiendo diferentes aproximaciones y formas de abordar el tema, pero, por ahora, se está lejos de obtener interpretaciones o explicaciones que puedan ser consideradas entendibles por un ser humano. Esto hace que, si se va a aplicar la IA en un problema donde se requiera que el modelo explique las decisiones, nos veamos limitados a utilizar modelos más simples (como los árboles de decisión o las redes neuronales con una sola capa oculta) y con un número de parámetros reducidos, con lo que se pierde bastante en cuanto a la efectividad que podremos esperar del modelo.

17.- APLICACIONES DE LA IA EN EL ÁMBITO DE LA SEGURIDAD

Algunos de los campos de aplicación que se pueden abordar con técnicas de IA en el campo de la seguridad son:

17.1.- Análisis predictivos de delitos

A partir de los registros de datos, se pueden utilizar las técnicas de aprendizaje automático para realizar análisis predictivos. Se pueden diseñar modelos para detectar dónde es más plausible que se pueda cometer un delito, e incluso los tipos de amenazas que se pueden dar. Aunque el sistema cometerá errores, puede ser de mucha ayuda para prevenir delitos. En este artículo (Cinelli 2019) se realiza una comparativa europea del análisis predictivo en la inteligencia policial. Se indica que hay un interés creciente, y que en 2019 ya se estaban usando este tipo de herramientas en 10 países europeos, con alrededor de unos 20 programas distintos. Los modelos predictivos son una de las aplicaciones típicas de la IA y el Big Data. En el campo de la economía, donde es habitual usar modelos predictivos, se han incorporado modelos basados en IA en la mayoría de las herramientas, así que resulta previsible que lo mismo ocurra en el campo de la seguridad.

17.2.- Detección automática de amenazas o delitos

Se pueden diseñar sistemas de IA que detecten personas o situaciones sospechosas. El campo de aplicación es enorme, pues seguramente habría que diseñar sistemas especializados según el tipo de amenaza. Los sistemas de reconocimiento facial han conseguido un nivel de desarrollo bastante elevado y son herramientas directamente aplicables. Interpol ya usa sistemas de reconocimiento facial para identificar personas de interés para una investigación (https://www.interpol.int/es/Como-trabajamos/Policia-cientifica/Reconocimiento-facial).

La identificación de situaciones sospechosas a partir de imágenes es un campo más complejo, donde se pueden aplicar las técnicas y modelos avanzados de clasificación de imágenes. Dado el avance en el desempeño de los modelos del procesamiento de imágenes mediante aprendizaje profundo, resulta bastante prometedor abordar problemas concretos de este tipo. Tendrían un coste en trabajo y tiempo de desarrollo a considerar, y requieren de un adecuado planteamiento, pero seguramente empezarán a surgir proyectos y herramientas en esta línea.

Por otro lado, se pueden desarrollar sistemas más concretos, diseñados para detectar diversas situaciones sospechosas específicas. Por ejemplo, la sospecha de denuncias falsas, como el sistema VeriPol que usa la Policía Nacional de España (Quijano-Sánchez 2018), o de posible uso fraudulento de tarjetas de crédito (Strelcenia 2022), o de posible fraude fiscal (Rodríguez-Pérez 2018). Las técnicas de aprendizaje automático bien aplicadas pueden dar buenos resultados en estos casos. En general estos son problemas donde el tipo de error debe ser considerados de forma distinta, preponderando evitar los falsos negativos, y donde el conjunto de ejemplos estará altamente desbalanceado. La aplicación directa de los modelos de aprendizaje y validación estándares puede ser bastante mejorada mediante un estudio más profundo, y sobre todo con la incorporación de modelos de aprendizaje por refuerzo para que el sistema se vaya adaptando a la evolución del problema.

17.3.- Prevención de posibles delitos

Otro apartado prometedor para el desarrollo de herramientas y proyectos es la prevención de posibles delitos. Se pueden aplicar sistemas de IA que detecten anomalías o comportamientos que convendría investigar para evitar un posible delito. Ya se está investigando en algunos problemas de este tipo. Por ejemplo, en la detección de perfiles falsos en redes sociales (Breuer 2020), o la detección de webs fraudulentas, o las suplantaciones de identidad (Monaro 2021). En estos casos, no se trata de detectar un delito, sino comportamientos que hagan sospechar que, detrás de un "sitio", "perfil" o "identidad", se pueden esconder actividades delictivas. El problema es más complejo que la detección directa del delito, pero el beneficio del sistema es mayor pues se trata de prevenir. Por otro lado, si se desea investigar de manera preventiva, parece razonable que el sistema indique argumentos que lo justifique. Esto implica el uso de modelos de IA que sean interpretables o explicables, algo que como hemos indicado reduce las posibilidades a modelos más clásicos de aprendizaje automático, y con un tamaño reducido. En muchos casos se dispone de sitios públicos donde se reportan casos positivos que pueden servir como fuente de ejemplos. Por contra, este tipo de problemas tienen una muy rápida evolución, pues se producen constantes cambios de perfiles, webs, o identidades para evitar ser descubiertos, lo que hacen que esos ejemplos queden obsoletos muy rápidamente. Si entrenamos modelos de aprendizaje automático a partir de las características directas, para cuando vayan a ser aplicados pueden haber cambiado lo suficiente como para reducir considerablemente la tasa de acierto. Por ello, se hace necesario el uso de otro tipo de características que sean menos sensibles a esos cambios, y con un mayor poder semántico para que el modelo resulte más explicable. En (Francisco, 2022b) abordamos el problema de detectar perfiles radicales yihadistas en redes sociales. Como apenas se disponía de ejemplos, se desarrolló una herramienta, Nutcracker (Francisco 2023), para generar bases de datos etiquetadas con un mínimo esfuerzo, utilizando la dinámica del comportamiento, las relaciones profundas entre usuarios, en lugar de aspectos más directos, que se podían modificar más fácilmente (Francisco 2022a). Además, utilizamos técnicas de análisis del discurso para extraer características menos camuflables, y que tuvieran mayor poder semántico, como la intencionalidad o la emoción de los mensajes. En la actualidad Nutcracker está siendo utilizado en un nuevo proyecto para detectar noticias falsas.

18.- CONCLUSIONES

En los últimos años el aprendizaje profundo ha supuesto una revolución para el avance de las aplicaciones de la IA. Se han obtenido complejos modelos del lenguaje tanto para reconocer, BERT, como para generar lenguaje, GPT, que pueden ser entrenados mediante un ajuste fino para abordar problemas de clasificación a partir de texto. Asimismo, se han obtenido complejos modelos convolucionales pre-entrenados, VGG-16 y VGG-19, que también pueden ser entrenados mediante un ajuste fino para abordar problemas de clasificación de imágenes. Estos avances suponen que haya un amplio espectro de problemas del ámbito de la seguridad done se pueden desarrollar herramientas basadas en IA. La detección automática de amenazas o delitos a partir del análisis de imágenes de video es un campo donde ya se han desarrollado herramientas efectivas, y donde es previsible que surjan nuevas. Además, parece prometedor utilizar aprendizaje profundo para abordar la detección de situaciones específicas de interés para la seguridad, como ya se está haciendo para la detección de denuncias falsas. Un caso de aplicación con muy buenas perspectivas es la detección de tipos concretos de fraude, como puede ser el fiscal,

de tarjetas de crédito, con criptomonedas, etc. Otro campo donde las técnicas de IA parecen prometedoras es en el análisis predictivo de delitos. Hay diversas herramientas que se están utilizando, y la tendencia es que haya un progresivo incremento, siendo muy previsible que se incorporen modelos basados en IA a estas herramientas. En cuanto a los sistemas preventivos, suponen un campo de aplicación de indudable valor añadido que incorpora dificultades adicionales la rápida evolución del problema y la necesidad en algunos casos de utilizar modelos interpretables. Se observa un gran interés en ellos, con diversos proyectos a los que también conviene estar atentos. En conclusión, hay una gran cantidad de posibilidades para las aplicaciones de la IA en seguridad, que investigadores, tecnólogos y entidades tenemos la oportunidad, casi obligación, de abordar con la máxima efectividad posible.

19.- BIBLIOGRAFÍA

Angwin, J. Larson, J. (2016). "Machine Bias". *ProPublica*. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Extraído el 26 de Febero de 2023.

Aznarte, JL; Benitez, JM; Castro, JL (2007). Smooth transition autoregressive models and fuzzy rule-based systems: Functional equivalence and consequences. Fuzzy Sets and Systems, 158-24, p. 2734-2745.

Baheti, P. (2023) A Simple Guide to Data Preprocessing in Machine Learning. Extraido el 13 de abril de 2023, https://www.v7labs.com/blog/data-preprocessing-guide

Breuer, A.; Eilat, R.; Weinsberg, U. (2020). Graph-Based Early Detection of Fake Accounts on Social Networks. Proceedings of The Web Conference 2020, p. 1287–1297

Lin, T; Jiang, J (2021) Credit Card Fraud Detection with Autoencoder and Probabilistic Random Forest. *Mathematics* 2021, *9*(21)

Mikolov, T., Chen, K., Carrado, G. and Dean, J., (2013). Efficient Estimation of Word Representations in Vector Space. 1st ed. [ebook]. Extraído el 20 de noviembre de 2015. <http://arxiv.org/pdf/1301.3781.pdf>

Minsky, M.; Papert, S. (1969). *Perceptrons.* M.I.T. Press.

Monaro M, Zampieri I, Sartori G, Pietrini P, Orrù G (2021). The detection of faked identity using unexpected questions and choice reaction times. Psychol Res. Sep;85(6):2474-2482.

Nagarka, P. Bhattacharya, A.; Jafari, O (2021). Exploring State-of-the-Art Nearest Neighbor (NN) Search Techniques. CODS-COMAD '21: Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD), 443–446

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D., Sutskever, I. (2018) Language Models are Unsupervised Multitask Learners. Extraído el 18 de enero de 2020. https://paperswithcode.com/paper/language-models-are-unsupervised-multitask

Radzi, F.; Khalil-Hani, M.; Imran, H.; …; Yan C. (2015). Generalizing convolutional neural networks for pattern recognition tasks. ARPN Journal of Engineering and Applied Sciences. 10. 5298-5308.

Rodriguez-Perez, E (2018). Análisis de detección de fraude fiscal mediante técnicas de Aprendizaje Automático. Tesis de Máster, Universidad Politécnicas de Madrid. Extraído el 25 de marzo de 2023.

https://oa.upm.es/55006/1/TFM_EDUARDO_RODRIGUEZ_PEREZ.pdf

Strelcenia E; Prakoonwit, S. Generating Synthetic Data for Credit Card Fraud Detection Using GANs, 2022 International Conference on Computers and Artificial Intelligence Technologies (CAIT), Quzhou, China, 2022, p. 42-47

Pothuganti, S. (2018). Review on over-fitting and under-fitting problems in Machine Learning and solutions. International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering. 7.

Quijano-Sánchez, L.; Liberatore, F.; Camacho-Collados, J.; Camacho-Collados, M. (2018) Applying automatic text-based detection of deceptive language to police reports: Extracting behavioral patterns from a multi-step classification model to understand how we lie to the police. Knowledge-Based Systems 149.

Roberts, M.: Driggs, D.; Thorpe, M.; et al. (2021) Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans. Nat Mach Intell 3, 199–217.

Rumelhart, D., Hinton, G. & Williams, R. (1986). Learning representations by back-propagating errors. Nature 323, 533–536.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386–408.

Simonyan K.; Zisserman, A.; (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations, 2015. Extraído el 8 de de diciembre de 2018. https://arxiv.org/abs/1409.1556

Soms, N. (2015). A Review on Support Vector Machine: A Classification Method. National Conference on Recent Advances in Design, Automation and Intelligent Systems 2015

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J. Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, Illia (2017-06-12). "Attention Is All You Need". Extraído el 10 de febrero de 2018. https://arxiv.org/abs/1706.03762

Veljanovska, k. (2017). Machine Learning Algorithms Comparison. International Journal of Engineering}, vol 7-11, p. 60-64

von Borzyskowsky, I.; Mazumder, A.; Mateen, B.; Wooldridge, M. (2020). Reflections on the response of the UK's data science and AI community to the COVID-19 pandemic.

Extraído el 20 de febrero del 2023 de https://www.turing.ac.uk/news/publications/data-science-and-ai-age-covid-19-report.

Wynants, L.; Van Calster, B.; Collins, G.S.; et al. (2020). Prediction models for diagnosis and prognosis of covid-19: systematic review and critical appraisal. British Medical Journal, BMJ 2020;369:m1328.