Universidad Internacional de la Rioja (UNIR)

Escuela Superior de Ingeniería y Tecnología

Máster en Computación Cuántica

Una plataforma de inteligencia artificial cuántica: QuantumSolver AI

Trabajo Fin de Estudios

Presentado por: José Daniel Escánez Expósito

Dirigido por: Rodrigo Gil-Merino y Rubio

Ciudad: Santa Cruz de Tenerife

Fecha: 20 de julio de 2023

Índice de contenidos

Índice de figuras

Índice de tablas

Resumen

En este trabajo se expone el desarrollo de un potente módulo de inteligencia artificial cuántica, implementado como una ampliación de la librería cuántica de código abierto QuantumSolver, cuyo desarrollo comenzó en la Universidad de La Laguna como un Trabajo Fin de Grado. El módulo permite aplicar diversos modelos de aprendizaje supervisado cuántico sobre cualquier conjunto de datos clásico. Se ponen a disposición del usuario una selección de modelos y datos predefinidos, así como un intuitivo tutorial para añadirlos a la plataforma. Se recogen configuraciones bien conocidas para lograr ejecuciones por defecto, así como la opción de ajuste de diferentes parámetros, brindando una herramienta completa en materia de investigación. De esta manera, se describen los principales detalles de la implementación realizada, así como las representaciones diseñadas de los resultados y las conclusiones obtenidas del estudio practicado sobre los modelos añadidos.

Palabras clave: inteligencia artificial cuántica, aprendizaje automático cuántico, aprendizaje supervisado, QuantumSolver.

Abstract

This project presents the development of a powerful quantum artificial intelligence module, implemented as an extension of the QuantumSolver open source quantum library, whose development began at the University of La Laguna as my Bacherlor's Thesis. The module allows the application of several quantum supervised learning models on any classical dataset. A selection of predefined models and data are made available to the user, as well as an intuitive tutorial to add them to the platform. Well-known configurations are collected to achieve default runs, as well as the option to adjust different parameters, providing a complete tool for research. In this way, the main details of the implementation are described, as well as the designed representations of the results and the conclusions obtained from the study practiced on the added models.

Keywords: quantum artificial intelligence, quantum machine learning, supervised learning, QuantumSolver.

1. Introducción

La computación cuántica es un punto de convergencia entre varias disciplinas como las matemáticas, la física y las ciencias de la computación. Utiliza las leyes fundamentales, que rigen la naturaleza en la pequeña escala, para realizar cálculos automáticos (Feynman, 1982). Esto le confiere una inherente gran complejidad, haciendo que los programadores de propósito general y las personas que no están vinculadas a la informática, se muestren reticentes a explorar esta materia (IBM, 2018). Existe una gran barrera entre la programación tradicional, que ya de por sí no es sencilla, y la programación cuántica (C. Singh, 2007). Todo el marco teórico, físico y matemático subyacente a este paradigma limita el acercamiento de potenciales consumidores y desarrolladores de estas tecnologías (Carroll, 2019).

Un sencillo ejemplo de cómo escala la complejidad del modelo de computación cuántica respecto a la clásica, se puede apreciar simplemente en la comparación de la unidad mínima de información en ambos paradigmas, así como sus interpretaciones. La primera gran diferencia entre ambos modelos se encuentra en lo más pequeño: los átomos de la computación. En el modelo clásico se tiene una entidad de información que puede estar establecida unívocamente en uno de dos estados, muchas veces representado como 0 o 1: el bit (Shannon, 1948). Su interpretación en la divulgación o en la literatura científica es bastante variada, existiendo tantos ejemplos como sistemas binarios: una luz encendida o apagada, el resultado de lanzar una moneda, el color de una carta de póquer, etc. La computación cuántica cuenta con una unidad mucho más potente y compleja: el cúbit. Esta puede encontrarse en dos estados cuánticos análogos a los anteriores ($|0\rangle$ o $|1\rangle$) o en una superposición de ellos (Nielsen y Chuang, 2000). En la Ecuación 1.1, se puede observar la fórmula que define el estado de un sistema cuántico de un cúbit, con $\alpha, \beta \in \mathbb{C}$ y $\alpha^2 + \beta^2 = 1$.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1.1}$$

El bit cuántico puede encontrarse en este estado de superposición hasta que sea medido (Schrödinger, 1935), en ese momento se obtendrá el estado $|0\rangle$ con probabilidad $||\alpha||^2$ o el estado $|1\rangle$ con probabilidad $||\beta||^2$. Es gracias a este hecho, y a otras propiedades como el entrelazamiento, que impone una correlación no intuitiva desde el paradigma clásico (Einstein, Podolsky y Rosen, 1935), por lo que se revisa al completo la teoría clásica de

la computación y se obtienen análogos cuánticos más potentes que prometen mejorar las tecnologías y comunicaciones del futuro (Wiesner, 1983). Sin embargo, esto añade una gran complejidad al modelo que puede resultar disuasoria debido a su alta curva de aprendizaje (IBM, 2018).

Por otro lado, la inteligencia artificial (IA) es más difícil de definir, desde que no tiene una única interpretación. Históricamente, no ha habido consenso hacia la dirección en la que este concepto debe avanzar. Russell y Norvig (2009) presentan cuatro posibles términos para expresar diferentes enfoques y características de la inteligencia artificial, recogidos en la Tabla 1.1.

		¿Cómo?	
		Como las personas	Racionalmente
¿Qué?	Pensar	Pensar como las personas	Pensar racionalmente
	Actuar	Actuar como las personas	Actuar racionalmente

Tabla 1.1: Enfoques sobre la definición de inteligencia artificial. Fuente: elaboración propia, inspirada en Russell y Norvig (2009).

La descripción de cada uno de los siguientes conceptos se lista a continuación:

- Pensar como las personas: se refiere a la capacidad de una máquina de simular el pensamiento humano, imitando los procesos cognitivos y la lógica utilizada por los seres humanos (Haugeland, 1989).
- Pensar racionalmente: se refiere a la capacidad de una máquina de pensar y razonar lógicamente, siguiendo principios y reglas formales sin necesariamente imitar el pensamiento humano (Winston, 1992).
- Actuar como las personas: se refiere a la capacidad de una máquina de realizar acciones de manera similar a como lo harían los seres humanos, ya sea a través de la percepción, la interacción social u otros aspectos relacionados con el comportamiento humano (Kurzweil, Richter, Kurzweil y Schneider, 1990).
- Actuar racionalmente: se refiere a la capacidad de una máquina de realizar acciones que maximicen la consecución de objetivos o resultados deseados, basándose en la información disponible y en el razonamiento lógico (Poole, Mackworth y Goebel, 1998).

En base a estas definiciones anteriores, y de manera más general, la inteligencia artificial se puede describir como un campo de estudio y desarrollo en la creación de sistemas capaces de realizar tareas que suelen requerir de la inteligencia humana. Para ello, se necesita de un proceso denominado entrenamiento, en el que un modelo de IA es enseñado para realizar una tarea específica (Russell y Norvig, 2009). A menudo, esto implica presentar al modelo una gran cantidad de ejemplos o datos de entrada, junto con las correspondientes respuestas o resultados esperados. Según estos datos son procesados, el modelo ajusta los parámetros internos y aprende a reconocer patrones, pudiendo realizar predicciones o decisiones basadas en ellos (Samuel, 1959). El uso de técnicas de aprendizaje automático y algoritmos que son inherentemente intensivos a nivel computacional. La realización de cálculos matemáticos complejos para llevar a cabo este aprendizaje necesita de un gran potencial de cómputo (Burkov, 2019). Además, depende de los datos escogidos, es posible que no se llegue a aprender el comportamiento deseado (Poole y Mackworth, 2010). La computación cuántica, puede ser una solución para estos problemas.

El uso de las tecnologías cuánticas para la implementación de sistemas de inteligencia artificial, permite el desarrollo de la inteligencia artificial cuántica. Esta se encuentra en sus etapas iniciales de desarrollo, por lo que aún no ha alcanzado su pleno potencial (Wittek, 2014). Aún así, existen ventajas potenciales en comparación con la IA clásica. La inteligencia artificial cuántica puede aprovechar las propiedades únicas de la computación cuántica, como la superposición y el entrelazamiento, para realizar cálculos en múltiples estados a la vez. Esto permite la resolución más eficiente de problemas complejos y optimización a gran escala (Biamonte et al., 2017). Se concretarán más detalles sobre esta comparación entre paradigmas en la Sección 2.2.

Una vez se han definido los elementos esenciales para el desarrollo del presente trabajo, se procede a exponer la motivación del proyecto.

1.1. Motivación

Como se ha visto en el Capítulo anterior, existen barreras en la transferencia relativa a la computación cuántica. La divulgación y creación de herramientas accesibles resultan cruciales para el crecimiento de las tecnologías relacionadas. La librería QuantumSolver fue implementada con este objetivo, recogiendo en su primera versión la implementación parametrizada de una selección de algoritmos cuánticos bien conocidos (Escanez-Exposito,

2022d), como se describe en la Sección 2.1. Sin embargo, muchas cuestiones escaparon de su alcance en el momento de su planteamiento inicial, entre ellas la inteligencia artificial cuántica.

Aunque existen algunos componentes de software con objetivos similares a los que plantea esta propuesta (Warke, Behera y Panigrahi, 2020; Tudorache, Manta y Caraiman, 2021; Poggel, Quetschlich, Burgholzer, Wille y Lorenz, 2022), ninguna recoge el planteamiento unificador modular de QuantumSolver. Se propone un núcleo de ejecución de algoritmos cuánticos, que permite la adición de módulos temáticos para atender a distintos sectores, entre otros: algoritmos básicos, criptografía, información cuántica, lógica combinacional e inteligencia artificial. Añadir este último factor es realmente importante, dado que sigue siendo un campo de estudio sobre el que se prevé obtener avances significativos en los próximos años (DeBenedictis, 2018; Deville y Deville, 2021; Duong et al., 2022). Ofrecer un software sencillo, cohesionado y accesible, que permita el entrenamiento de modelos de inteligencia artificial cuántica, supone un gran aporte para los usuarios que sufren esa barrera teórica, así como para aquellos investigadores que deseen realizar de manera cómoda los experimentos planteados.

La creación de esta herramienta facilita enormemente ejecutar las técnicas de inteligencia artificial basadas en ordenadores cuánticos sobre conjuntos de datos de cualquier sector, obteniendo conclusiones acerca de la identificación y aprovechamiento de las áreas en las que el potencial de la computación cuántica estará mejor aprovechado. Así mismo, también se podrá comparar con facilidad los modelos añadidos, identificando cuáles son mejores en función de las características de entrada del conjunto de datos. Por último, también permitirá la ejecución de manera sencilla de cualquiera de los modelos disponibles sobre cualquier conjunto de datos propuesto por el usuario.

1.2. Planteamiento

Se propone realizar una revisión bibliográfica general sobre el aprendizaje automático cuántico, que facilite las sucesivas labores de desarrollo e implementación. Se plantea la creación de un nuevo módulo para la librería QuantumSolver incluyendo varios modelos de aprendizaje automático cuántico (QML, por sus siglas en inglés), concretamente de aprendizaje cuántico supervisado.

Se deberá permitir ejecutar estos modelos sobre conjuntos de datos preestablecidos,

bien conocidos dentro de la literatura científica y conjuntos de datos propuestos por el usuario. También se integrará un modelo de aprendizaje automático clásico (ML, por sus siglas en inglés), para habilitar la posibilidad de realizar comparaciones que verifiquen la calidad de los resultados obtenidos. De esta manera, se podrá comparar los efectos de los diferentes modelos aplicados en los distintos conjuntos de datos, incluyendo los añadidos por el usuario.

1.3. Estructura

La presente memoria está compuesta por siete capítulos:

- El Capítulo 1 realiza la introducción al tema del proyecto y sintetiza los aspectos iniciales más importantes. De esta manera, queda patente la motivación, planteamiento y estructura del trabajo.
- El Capítulo 2 configura el entorno inicial, poniendo en contexto QuantumSolver y el estado actual del aprendizaje automático clásico y cuántico.
- El Capítulo 3 muestra el diseño de la plataforma. También se lista una serie de importantes objetivos y requisitos que deben ser alcanzados y cumplidos por la propuesta como, por ejemplo, la creación del módulo de QML dentro de QuantumSolver, la sencillez de la propuesta y el envío de artículos a congresos.
- El Capítulo 4 expone una detallada descripción de la implementación realizada. Se describe el desarrollo construido sobre el diseño anteriormente planteado, listando las nuevas funcionalidades accesibles mediante el uso de la plataforma.
- El Capítulo 5 realiza un análisis exhaustivo de los resultados obtenidos, mostrando un caso de uso práctico y didáctico, que pone en valor la sencillez y versatilidad de las funcionalidades disponibles gracias al código implementado. Además, se discuten algunos aspectos generales del módulo, así como sus limitaciones.
- El Capítulo 6 desarrolla las conclusiones del trabajo, destacando la consecución de los objetivos planteados, el impacto de la plataforma y su potencial.
- El Capítulo 7 sugiere algunas posibles líneas futuras de trabajo. A pesar de que estas queden fuera del alcance del mismo, se argumenta el interés científico e industrial de que sean llevadas a cabo.

Además, al final de este documento se anexan las revistas, los congresos nacionales e internacionales a los que se han enviado artículos a modo de seguimiento del desarrollo de la propuesta. Se diferencian aquellas publicaciones realizadas en las etapas iniciales, que avalaron el éxito de la plataforma; de las que han tenido lugar durante el desarrollo de este trabajo, destacando la importancia de su ejecución y planteamiento.

2. Contexto y estado de la técnica

2.1. Antecedentes: historia de la librería QuantumSolver

Durante el mes de diciembre de 2021 surgió la propuesta de implementación de la librería de código abierto QuantumSolver (Escanez-Exposito, 2022d), que inicialmente permitía la ejecución de algoritmos cuánticos predefinidos, tanto en hardware cuántico real, como en simuladores. Esta API[1] contiene una intuitiva estructura de clases (Escanez-Exposito, 2022c) que permite la definición de nuevos algoritmos de manera sencilla para programadores de propósito general (sin que estén especializados necesariamente en computación cuántica). Además, para brindar una mayor accesibilidad al proyecto, no solo cuenta con esta interfaz por línea de comandos (CLI, por sus siglas en inglés), sino que también tiene disponible una útil interfaz web[2]. Esta última puede visualizarse en la Figura 2.1.

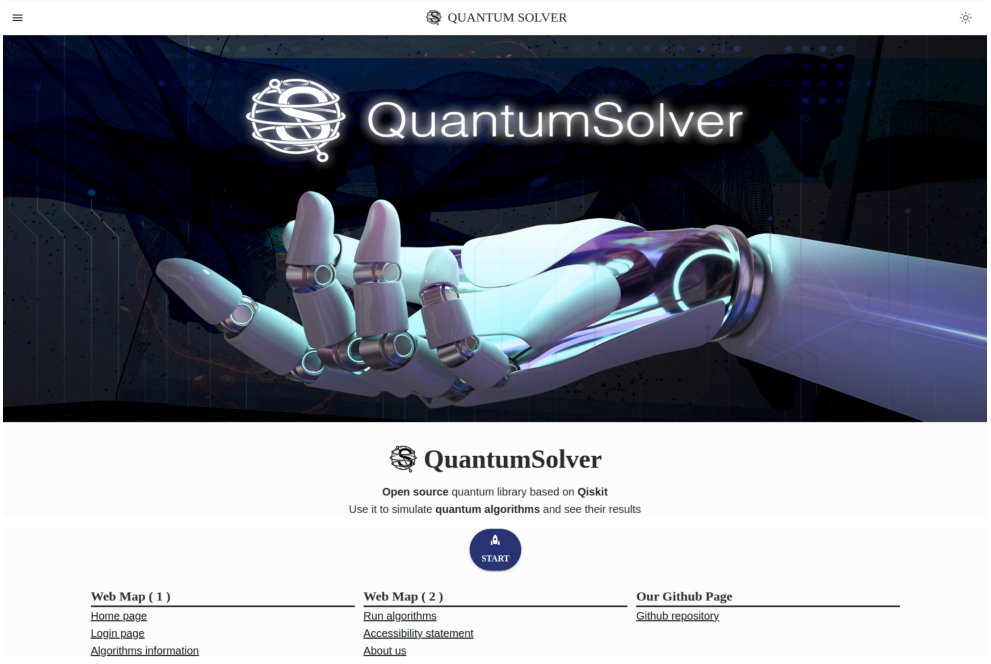

Figura 2.1: Página principal de la interfaz web de QuantumSolver. Fuente: elaboración propia.

[1] API es el acrónimo en inglés de "interfaz de programación de aplicaciones". Se trata de un software intermediario entre dos sistemas. Una API puede permitir, para una determinada aplicación, el acceso a los recursos disponibles en unos servidores externos.

[2] Esta interfaz web se puede visualizar siguiendo el tutorial de Escanez-Exposito (2022a).

A finales de 2022, el proyecto de QuantumSolver recibió el primer premio en el *I Concurso en Computación en la Nube e Inteligencia Artificial* (Cátedra Edosoft, 2022) y el tercer premio en el *Concurso TFG/TFM/Tesis en Ciberseguridad* (Cátedra de Ciberseguridad Binter, 2022), ambos por la Universidad de La Laguna.

Esta herramienta gestiona la ejecución de los algoritmos mediante un importante componente desarrollado denominado *QExecute* (Escanez-Exposito, 2022b). Este se encarga de la autenticación contra los servicios de *IBM Quantum* (2023b), logrando acceder al hardware proporcionado. El proceso de identificación se realiza gracias a una clave con formato *API token*[3] de *IBM Quantum Experience* (2023f). Sin embargo, para no requerir que obligatoriamente el usuario cree una cuenta en *IBM Quantum* para obtener este *token*, se desarrolló un modo invitado, que posibilita la ejecución anónima dentro de QuantumSolver. En este solo se permite ejecutar en el simulador *aer_simulator* (2023a), careciendo del hardware cuántico real de IBM.

Recientemente, los objetivos de la propuesta han crecido tanto en profundidad como en amplitud. Se están añadiendo frecuentemente algoritmos nuevos a la librería, por lo que ha sido necesario clasificarlos según su propósito. Para ello, QuantumSolver se ha dividido en diferentes módulos:

- QuantumSolver Basic: recoge los algoritmos básicos de la propuesta inicial, que son listados a continuación.
    - Generación de números aleatorios (QRand). El problema que resuelve este método es realmente importante por sus aplicaciones criptográficas. Desde el paradigma clásico, se hacen intentos por diseñar algoritmos que simulen esta aleatoriedad que es intrínseca al paradigma cuántico (L'Ecuyer, 2012).
    - Deutsch-Jozsa para la clasificación de funciones binarias en: constantes, que devuelven el mismo valor independientemente de la entrada; o balanceadas, que reparten uniformemente las salidas posibles entre todos los casos de entrada (Deutsch y Jozsa, 1992).
    - Bernstein-Vazirani: obtención de una clave binaria secreta arbitraria mediante una función oráculo, que clásicamente aporta información parcial sobre la contraseña (Bernstein y Vazirani, 1993).

[3]Un *API token* es una firma cifrada que permite identificar al usuario contra los servidores correspondientes.

  - Grover: búsqueda eficiente en bases de datos no estructuradas, es decir, en una secuencia no ordenada (Grover, 1996).
  - Teleportación cuántica: transmisión de un cúbit mediante dos bits de comunicación (Bennett et al., 1993).
  - Codificación superdensa: transmisión de dos bits por medio de un único cúbit de comunicación (Bennett y Wiesner, 1992).

- QuantumSolver Crypto: ofrece un catálogo de simuladores de protocolos criptográficos cuánticos. Se recogen algunos algoritmos para la distribución cuántica de claves (QKD, por sus siglas en inglés), que se apoyan en la auténtica aleatoriedad de su generación (Shannon, 1949), como son:

  - BB84 (Bennett y Brassard, 1984): primer protocolo de QKD, basado en el principio de incertidumbre (Heisenberg, 1927).
  - E91 (Ekert, 1991): método de QKD utilizando el entrelazamiento y la violación de las desigualdades de Bell (Bell, 1964; Einstein et al., 1935).
  - B92 (Bennett, 1992): algoritmo de QKD que se basa en propiedades de estados no ortogonales.

  Además, cuenta con implementaciones de versiones cuánticas de los criptosistemas de ElGamal (1985) y RSA (Rivest, Shamir y Adleman, 1978).

- QuantumSolver Subroutine: contiene una colección de métodos más complejos en los que la parte cuántica es solo una subrutina del procedimiento general. Algunos ejemplos ya implementados son:

  - Estimación cuántica de fase: estima la fase o autovalor del autovector de un operador unitario (Kitaev, 1995).
  - Algoritmo de Simon: clasificación de funciones binarias (Simon, 1997).
  - Algoritmo de Shor: factorización de números enteros (Shor, 1994).

- QuantumSolver Composer: permite simular circuitos clásicos mediante el uso subyacente de circuitos cuánticos. Al hacer uso del código desarrollado, el usuario tiene acceso a una API que permite definir algoritmos clásicos a nivel de circuito de manera clásica. Las estructuras de datos implementadas traducen estas instrucciones

clásicas a instrucciones propias de los circuitos cuánticos, que respetan los principios de reversibilidad necesarios (Nielsen y Chuang, 2000). De esta manera, se pueden definir complejos algoritmos cuyo diseño sería inviable desde el paradigma cuántico, a través de una conversión directa de la estructura clásica. Se plantea el desarrollo de versiones cuánticas reducidas (es decir, utilizando un menor número de cúbits) de los criptosistemas AES (Daemen y Rijmen, 1999) y ChaCha20 (Nir y Langley, 2018).

A medida que la propuesta ha ido creciendo, se han enviado periódicamente los avances conseguidos a diferentes congresos nacionales e internacionales que pueden ser encontrados en el Apéndice B(Escanez-Exposito, Caballero-Gil y Martin-Fernandez, 2022b, 2022c, 2022a, 2022e, 2022d, 2023; Escanez-Exposito y Caballero-Gil, 2023; Escanez-Exposito, Marchan-Sekulic y Caballero-Gil, 2023; Escanez-Exposito, Hernandez-Martin y Caballero-Gil, 2023).

En el presente trabajo, se añade un nuevo módulo a esta plataforma: QuantumSolver AI. Este brinda la posibilidad de ejecutar diferentes modelos de QML sobre varios conjuntos de datos conocidos, siguiendo la estructura general de la librería. Supone todo un reto unificar el software ya desarrollado con la novedosa propuesta de inteligencia artificial cuántica, que permita obtener útiles resultados aplicados a diferentes sectores interesados en el avance de estas tecnologías.

2.2. Estado actual del aprendizaje automático supervisado

2.2.1. Aprendizaje automático clásico

El potencial de la inteligencia artificial ha crecido a un ritmo exponencial en los últimos años (Kelleher, Namee y D'Arcy, 2020). La mejora constante de tecnologías y modelos permiten alcanzar nuevos hitos en varias disciplinas:

- Procesamiento del lenguaje natural: interacción entre la computación y el lenguaje humano (Ornstein, 1955; Khurana, Koli, Khatter y Singh, 2023).

- Aprendizaje automático: desarrollo de técnicas que permiten que los ordenadores adquieran, o parezcan adquirir, habilidades, conocimientos, conductas y valores (Samuel, 1959; Khan et al., 2022).

- Aprendizaje profundo: conjunto de algoritmos de aprendizaje automático basados en asimilar representaciones de datos (Amari, 1972; Chugh, Kumar y Singh, 2021).

- Visión por computador: conjunto de métodos para adquirir, procesar, analizar y comprender imágenes para la producción de información por parte de un ordenador (Papert, 1966; Fuoli, Huang, Paudel, Van Gool y Timofte, 2023).

Todo ello cuenta con innumerables aplicaciones en prácticamente todos los sectores de la sociedad, como la sanidad, la economía, la industria y el transporte (Podoletz, 2023; Klein, Depping, Wohlfahrt y Fassbender, 2023; Luusua, Ylipulli, Foth y Aurigi, 2023).

El aprendizaje automático puede dividirse en tres campos diferenciados: supervisado, donde el sistema aprende a clasificar mediante el uso de datos previamente etiquetados a modo de ejemplo (Geman, Bienenstock y Doursat, 1992); no supervisado, donde los modelos clasifican a partir de las variables de entrada, sin contar con etiquetas preestablecidas (Buhmann y Kuhnel, 1992); y por refuerzo, donde se establece un entorno en el que la IA puede explorar las acciones posibles, y encontrar las más satisfactorias, sujetas a un conjunto de premios y penalizaciones (Kaelbling, Littman y Moore, 1996).

Se procede a explicar en detalle el funcionamiento de la clasificación por aprendizaje supervisado clásico mediante redes neuronales, ya que será de gran utilidad para comprender el diseño e implementación descritos posteriormente (véase capítulos 3 y 4).

2.2.1.1. Clasificación por aprendizaje supervisado

Se trata de una técnica de aprendizaje automático que permite predecir la categoría a la que pertenecen los datos de entrada, en base a un entrenamiento previo utilizando datos etiquetados (Russell y Norvig, 2009). Para realizar el entrenamiento, se suele utilizar un subconjunto de todos los datos con los que se cuentan, reservando el resto para la verificar la calidad de los resultados del proceso. Se utiliza un algoritmo, como el que se describe en los siguientes apartados, para llevar a cabo la asignación pertinente de los datos de prueba en categorías específicas (véase Fig. 2.2). De esta manera, se reconocen relaciones específicas entre los atributos de las entidades del conjunto de datos, extrayendo conclusiones sobre cómo realizar la clasificación.

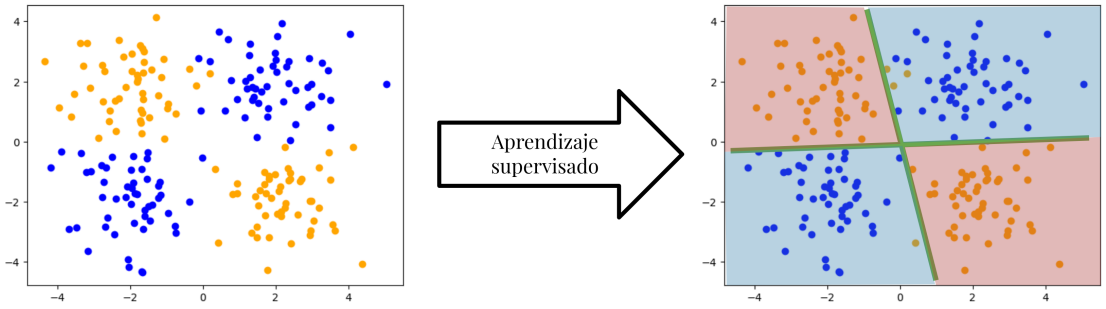

Figura 2.2: Imagen conceptual de la clasificación por aprendizaje supervisado. Fuente: elaboración propia.

2.2.1.2. Redes neuronales

Es la estructura utilizada principalmente en los algoritmos de aprendizaje profundo, que procesan los datos de entrenamiento imitando la interconectividad del cerebro humano a través de capas formadas por nodos o neuronas artificiales (McCulloch y Pitts, 1943). La neurona artificial más simple es el perceptrón, que está compuesto por una lista de entradas x, una lista de pesos w, una función de activación $g(a) = a - u > 0$ definida por el umbral u y una salida $f(x)$, como puede observarse en la Figura 2.3 (Rosenblatt, 1958).

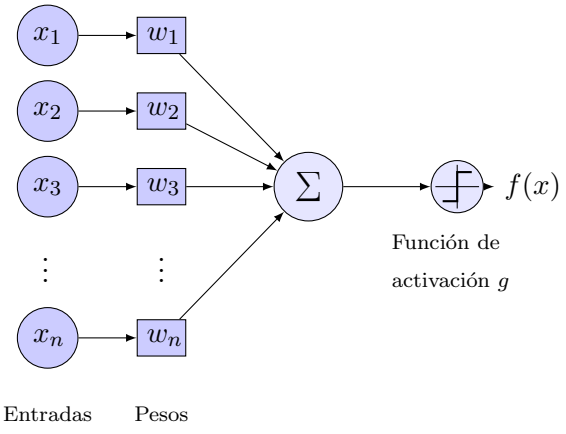

Figura 2.3: Neurona artificial del tipo perceptrón simple. Fuente: elaboración propia, inspirada en Bargen (2013).

La definición matemática de la operación realizada por el perceptrón, que define la salida $f(x)$, viene dada por la siguiente ecuación:

$$f(x) = \begin{cases} 1 & \text{si } \sum_{i=1}^{n} w_i \cdot x_i - u > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.1}$$

Además, las neuronas se organizan en capas, donde usualmente cada una de estas últimas se conectan con la siguiente, formando una estructura en cascada. En el caso de una capa densa, como se ilustra en la Figura 2.4, cada neurona artificial está conectada con todas las de la capa posterior.

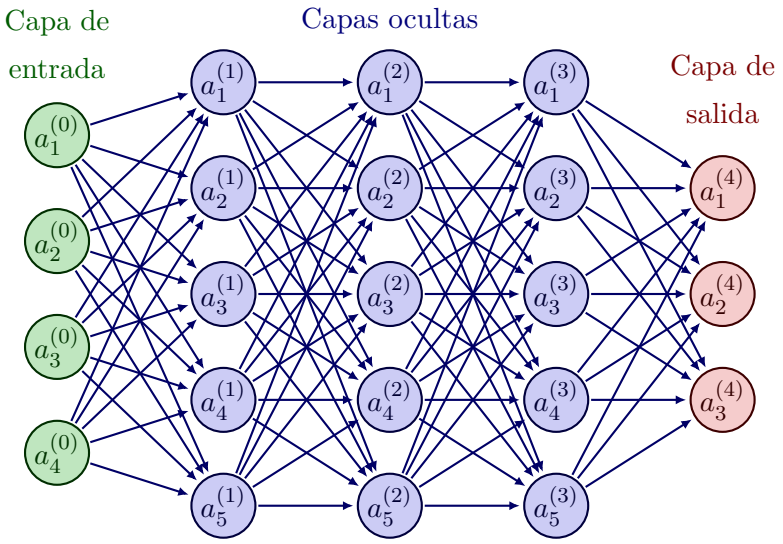

Figura 2.4: Ejemplo de red neuronal densa. Fuente: elaboración propia, inspirada en Neutelings (2023).

Durante el proceso de entrenamiento, las redes neuronales ajustan los pesos de los nodos mediante el algoritmo de descenso de gradiente (Curry, 1944). Este algoritmo utiliza la función de pérdida para medir la discrepancia entre las salidas predichas por el modelo y las salidas reales. Luego, aplica retropropagación, que es una técnica que permite que la información de la función de pérdida fluya hacia atrás a través de la red para calcular el gradiente, permitiendo ajustar los pesos y sesgos de manera que se minimice la función (Rosenblatt, 1961). Una vez que la red neuronal ha sido entrenada, se comprueba si el modelo ha alcanzado una buena precisión, gracias a los datos reservados para la verificación (Larose y Larose, 2014).

La capacidad de las redes neuronales para aprender y adaptarse a partir de los datos de entrenamiento, las convierte en una poderosa herramienta en el ámbito de la inteligencia artificial. El diseño y entrenamiento de redes neuronales sigue siendo un área de

investigación activa, con constantes avances y mejoras en los algoritmos y arquitecturas utilizadas.

La computación cuántica también ha estado en auge y se estima una gran proyección de las tecnologías relacionadas en los próximos años (IBM-IBV, 2021), surgiendo de manera natural el intento de aprovechar la potencia de este paradigma computacional en la realización de cálculos siguiendo estrategias inteligentes (DeBenedictis, 2018; Deville y Deville, 2021; Duong et al., 2022). En este trabajo se hará especial hincapié en el QML.

2.2.2. Aprendizaje automático cuántico

Dentro del QML existe otra forma de categorizar los modelos en función del tipo de datos con los que trabajan y de algoritmos que aplican, es decir, si estos son clásicos o cuánticos (Dunjko, Taylor y Briegel, 2016). Siguiendo la referencia anterior, se exponen cuatro posibles combinaciones:

- Tratar datos clásicos con algoritmos clásicos inspirados en algoritmos cuánticos (CC).
- Tratar datos clásicos con algoritmos cuánticos, planteamiento en el que se centrará este trabajo (CQ).
- Tratar datos cuánticos con algoritmos clásicos, donde juega un papel fundamental la caracterización, el control y la lectura de cúbits (QC).
- Tratar datos cuánticos con algoritmos cuánticos, que es un campo de estudio muy incipiente (QQ).

Por el momento, el planteamiento de utilizar la tecnología cuántica sobre datos clásicos (CQ) es la que más interés y resultados positivos ha generado recientemente (Cerezo, Verdon, Huang, Cincio y Coles, 2022). Por ello, se ha elegido este enfoque para la realización del presente trabajo. De esta manera, se pretenden utilizar tecnologías vanguardistas que ya han sido testadas, y que tienen ciertas garantías, facilitando el objetivo que se desea alcanzar y avalando el éxito de su desarrollo.

A continuación, se exponen los análogos cuánticos de las herramientas anteriormente explicadas del ML clásico, es decir, desde el paradigma de la computación cuántica.

2.2.2.1. Codificación del conjunto de datos

Escoger un método correcto para representar la información es crucial en los sistemas de QML (IBM, 2022e). Así como en los sistemas clásicos podría no tener tanta importancia, dado que la dificultad principal se encuentra principalmente en la conversión numérica de los parámetros (Bengio, Ducharme y Vincent, 2000); en los sistemas cuánticos influye directamente en la calidad de los resultados obtenidos (Schuld, Sweke y Meyer, 2021). Por ello, resulta fundamental abordar la búsqueda de mecanismos eficientes para representar la información para procesarla usando QML.

A continuación se exponen los métodos más conocidos y utilizados en la actualidad. Para ello, se parte de un conjunto $\mathscr{X}$ de M datos clásicos, cada uno de ellos con N características, que viene dado por la ecuación:

$$\mathscr{X} = \{x^{(1)}, \ldots, x^{(m)}, \ldots, x^{(M)}\}, \tag{2.2}$$

que ilustra la representación matemática del conjunto de datos, donde $x^{(m)}$ es un vector N-dimensional, $\forall m \in \{1, \ldots, M\}$. Los métodos son los siguientes:

- Codificación en base: asocia cadenas binarias de tamaño n con un estado de la base computacional para un sistema de n cúbits. Es decir, para una cadena de n bits $x = (b_1, b_2, \ldots, b_n)$ el estado correspondiente es $|x\rangle = |b_1, b_2, \ldots, b_n\rangle$, con $b_i \in \{0, 1\}$, $\forall i \in \{1, \ldots, n\}$ (IBM, 2022d).

  De esta manera, para codificar el conjunto de datos $\mathscr{X}$, cada elemento debe corresponderse con una cadena binaria de tamaño n, para realizar la asignación directa con el estado cuántico correspondiente. Por ello, el número de datos se ve acotado por el número de cúbits del sistema ($M \leqslant 2^n$). En la Ecuación 2.3 se muestra la representación del conjunto de datos completo como superposición de estados de la base computacional:

  $$|\mathscr{X}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^m\rangle. \tag{2.3}$$

  A pesar de que es una codificación bastante intuitiva, una de sus principales desventajas es que los esquemas actuales de implementación no son demasiado eficientes. En particular, con vectores de estado dispersos[4] crece la complejidad del circuito que

[4]Un vector disperso es aquel en la que la mayor parte de sus elementos son cero.

realiza la codificación.

- Codificación en amplitud (IBM, 2022a): asigna cada punto x normalizado N-dimensional con las amplitudes de un sistema cuántico $|\psi_x\rangle$ de n cúbits. En la Ecuación 2.4 se puede observar la descripción de este estado,

$$|\psi_x\rangle = \sum_{i=1}^{2^n} x_i \, |i\rangle \,, \tag{2.4}$$

donde x_i representa el i-ésimo elemento de x y el i-ésimo elemento de la base computacional para n cúbits viene dado por $|i\rangle$.

Para codificar el conjunto de datos $\mathscr{X}$, se concatenan los M puntos N-dimensionales en un único vector de amplitudes α de tamaño $N \times M$. Este vector viene descrito por la ecuación:

$$\alpha = A_{norm}(x_1^{(1)}, \ldots, x_N^{(1)}, \ldots, x_1^{(m)}, \ldots, x_N^{(m)}, \ldots, x_1^{(M)}, \ldots, x_N^{(M)}), \tag{2.5}$$

donde $A_{norm} = \frac{1}{\sqrt{\sum_{j=1}^{N}\sum_{k=1}^{M}(x_j^{(k)})^2}}$ es una constante de normalización que satisface $|\alpha|^2 = 1$. Esta codificación requiere, por tanto, de $n \geqslant \log_2(NM)$ cúbits.

Una vez obtenido el vector α, se pueden asignar sus componentes α_i como amplitudes a los estados correspondientes $|i\rangle$. La combinación lineal de estos estados en función de las amplitudes describe el estado del conjunto de datos $|\mathscr{X}\rangle$, como se muestra a continuación:

$$|\mathscr{X}\rangle = A_{norm} \sum_{j=1}^{N}\sum_{k=1}^{M} x_j^{(k)} \, |(k-1)N + j\rangle = \sum_{i=1}^{2^n} \alpha_i \, |i\rangle \,. \tag{2.6}$$

- Codificación en ángulo: vincula las N características de un punto a los ángulos de rotación de un sistema de n cúbits, siendo $N \leqslant n$ (IBM, 2022b). En la Ecuación 2.7 se puede observar la codificación de un punto arbitrario $x = (x_1, \ldots, x_N)$:

$$|\alpha_x\rangle = \bigotimes_{i=1}^{N} cos(x_i) \, |0\rangle + sin(x_i) \, |1\rangle \,. \tag{2.7}$$

También existe una versión generalizada de esta codificación, la codificación densa en ángulo, que permite representar dos características por cúbit, usando la fase relativa como ilustra la siguiente ecuación:

$$|\alpha_x\rangle = \bigotimes_{j=1}^{N/2} cos(x_{2j-1})\,|0\rangle + e^{ix_{2j}} sin(x_{2j-1})\,|1\rangle\,. \tag{2.8}$$

En cualquier caso, esta codificación se puede interpretar como una puerta cuántica unitaria S_{x_j} dada por las ecuaciones:

$$S_{x_j} = \bigotimes_{i=1}^{N} U(x_j^{(i)}), \tag{2.9}$$

$$U(x_j^{(i)}) = \begin{pmatrix} cos(x_j^{(i)}) & -sin(x_j^{(i)}) \\ sin(x_j^{(i)}) & cos(x_j^{(i)}). \end{pmatrix} \tag{2.10}$$

Además, por su parecido con la puerta de rotación sobre el eje Y (véase Ecuación 2.11), se puede expresar de manera realmente sencilla a partir de la misma. La única diferencia es que se debe duplicar el ángulo de entrada (por lo que $U(x_j^{(i)}) = RY(2x_j^{(i)})$).

$$RY(\theta) = exp(-i\frac{\theta}{2}Y) = \begin{pmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \tag{2.11}$$

Por ejemplo, con este tipo de codificación, para codificar un punto $x = (\phi, \theta, \lambda)$ se puede hacer uso del circuito que se ilustra en la Figura 2.5.

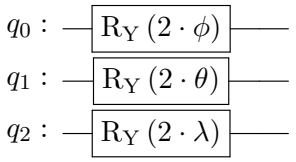

Figura 2.5: Codificación en ángulo para un punto tridimensional. Fuente: elaboración propia, inspirada en IBM (2022b).

- Codificacion arbitraria: La codificación arbitraria codifica N características como N rotaciones en puertas parametrizadas en n cúbits, donde $n \leqslant N$, brindando una profundidad constante al circuito para garantizar su ejecución en hardware cuántico real (IBM, 2022c). Al igual que la codificación en ángulo, sólo codifica un punto de los datos cada vez, en lugar de un conjunto de datos completo.

Como su nombre indica, estos circuitos son diseñados arbitrariamente satisfaciendo las condiciones anteriores. En la Figura 2.6 se puede observar el ejemplo de una instancia del circuito EfficientSU2 disponible en Qiskit (Qiskit Development Team, 2023a), que utiliza únicamente tres cúbits para la codificación de un punto $x = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2)$ de doce características.

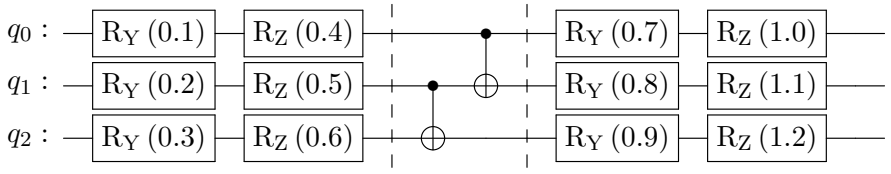

Figura 2.6: Ejemplo de codificación arbitraria mediante el circuito EfficientSU2 de Qiskit. Fuente: IBM (2022c).

La búsqueda de esquemas de codificación eficientes siguiendo este tipo de patrones es un campo de investigación en activo.

2.2.2.2. Ansatz

En el contexto de los circuitos variacionales, un ansatz suele describir una secuencia de puertas cuánticas que actúan a modo de "circuito parametrizado de acercamiento" (Nakaji y Yamamoto, 2021; Xanadu, 2023b). Este tiene el objetivo de ajustar sus parámetros para lograr obtener la salida esperada. Además, tiene una arquitectura muy similar a la de una red neuronal, en el sentido de que los parámetros son ajustados de manera parecida a como se ajustan los pesos de una red neuronal clásica.

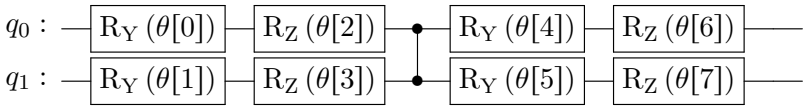

Figura 2.7: Ejemplo de circuito ansatz del tipo TwoLocal disponible en Qiskit. Fuente: IBM (2022e).

En la Figura 2.7 se puede observar una implementación del ansatz TwoLocal disponible en Qiskit (IBM, 2023e). Este hace uso de puertas de rotación Y y Z, además de puertas de fase controladas (Havlíček et al., 2019).

2.2.2.3. Clasificador cuántico variacional

El clasificador cuántico variacional combina los dos tipos de circuitos anteriores, codificador y ansatz, para lograr atribuir los pesos correctos al circuito variacional (Peruzzo et al., 2014), ajustando el modelo al conjunto de datos propuesto $\mathscr{X}$.

Matemáticamente se puede definir el estado cuántico deseado mediante la ecuación:

$$|\psi(\vec{x}_i;\vec{\theta})\rangle = U_{W(\vec{\theta})}U_{\phi(\vec{x}_i)}\,|0\rangle\,, \tag{2.12}$$

que expone la relación entre los vectores de entrada $\vec{x}_i$ (cada uno de ellos tiene atribuida su clase y_i) con las puertas cuánticas de ansatz $U_{W(\vec{\theta})}$ y de codificación $U_{\phi(\vec{x}_i)}$.

En la fase de entrenamiento, se buscan los valores de $\vec{\theta}$ que generan mejores predicciones de las clases correspondientes a los elementos de entrada. Tras una primera ejecución, se comparan las predicciones $\hat{y}_i$ con las etiquetas dadas y_i, calculando el éxito obtenido a través de una función de coste clásica. En base a la salida de la misma, se utiliza un algoritmo clásico de optimización para elegir un nuevo valor de $\vec{\theta}$. Este proceso es repetido hasta que la función de coste logra estabilizarse. Un diagrama ilustrativo de este proceso se puede observar en la Figura 2.8.

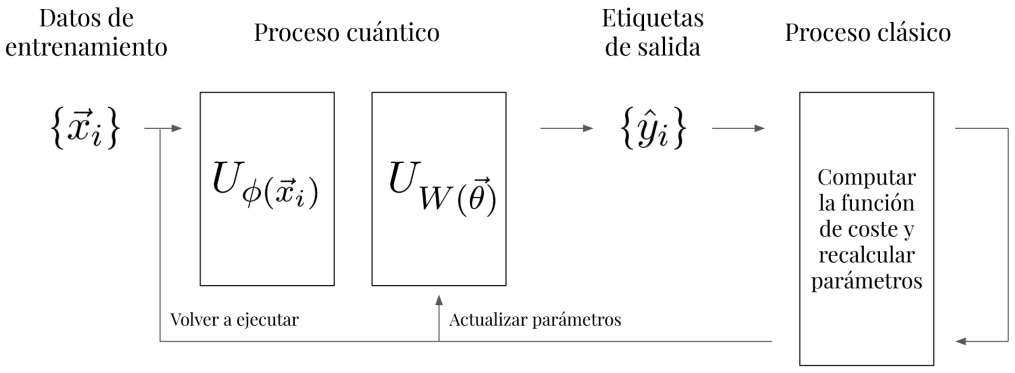

Figura 2.8: Esquema del clasificador cuántico variacional. Fuente: elaboración propia, inspirada en IBM (2022e).

2.2.3. Discusiones

Siguiendo el esquema anterior (Figura 2.8), si el proceso de codificación escogido no es correcto, dado que puede representar la información de manera que no sea separable, se puede llegar a la situación de que sea completamente imposible que el sistema clasifique correctamente, independientemente de la calidad del ansatz (Pérez-Salinas, Cervera-Lierta,

Gil-Fuster y Latorre, 2020). Precisamente, este punto puede provocar controversia al considerar los circuitos variacionales como redes neuronales cuánticas (Ciliberto et al., 2018). En la versión clásica de las redes neuronales, independientemente del modelo utilizado para representar la información, el éxito de la clasificación reside mayormente en el proceso del ajuste de los pesos de la red. Esta discrepancia, hace que surjan nuevas alternativas en los circuitos variacionales para lograr un efecto similar al de una red neuronal clásica. Una de ellas es la aplicación repetida del proceso cuántico formado por codificador y ansatz ($U_{W(\vec{\theta})}U_{\phi(\vec{x}_i)}$), de tal manera que la codificación se refuerza en diferentes etapas del circuito, como ocurre en una red neuronal recurrente (Bausch, 2020). También se estudian enfoques mixtos con redes neuronales híbridas clásicas y cuánticas (Chalumuri, Kune y Bs, 2021; Sebastianelli, Zaidenberg, Spiller, Saux y Ullo, 2022).

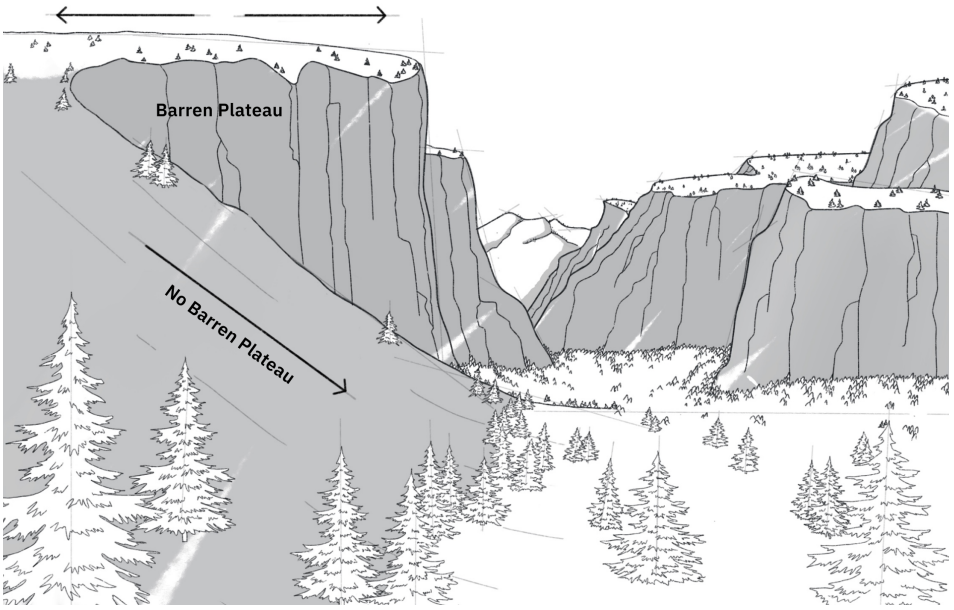

Figura 2.9: Imagen ilustrativa de *barren plateau*. Fuente: IBM (2022f).

Por último, se hace hincapié en uno de los problemas actuales de los circuitos variacionales. En las secciones 2.2.1.2 y 2.2.2.2, referentes a las redes neuronales clásicas y al ansatz, respectivamente; se expuso un objetivo común entre ambos paradigmas de computación: optimizar una función objetivo ajustando los pesos o parámetros. Sin embargo, en las redes neuronales cuánticas, la varianza del gradiente disminuye de forma exponencial con el número de cúbits (McClean, Boixo, Smelyanskiy, Babbush y Neven, 2018). Esto hace que la función objetivo describa un entorno plano o similar en todas las direcciones (véase Figura 2.9), por lo que el método de optimización puede encontrar dificultades al

determinar la dirección en la que buscar la solución. Esta situación es denominada *barren plateau*. Algunos estudios recientes buscan la solución a este problema en el ruido cuántico intrínseco en las máquinas cuánticas reales (Wang et al., 2021). Otros buscan la relación del *barren plateau* con factores como la expresividad del ansatz utilizado (Holmes, Sharma, Cerezo y Coles, 2022). A partir de estas investigaciones se sugieren soluciones al mismo, como la utilización de sombras clásicas (en inglés, *classical shadows*), un protocolo de predicción de funciones de estado cuántico utilizando únicamente un número logarítmico de medidas (Huang, Kueng y Preskill, 2020; Sack, Medina, Michailidis, Kueng y Serbyn, 2022). Estos estudios apuntan a que al realizar un menor número de mediciones sobre los cúbits utilizados, se reduce el efecto de esta indeseada condición.

En este Capítulo se ha contextualizado la plataforma QuantumSolver, poniendo de manifiesto el nuevo módulo de QML que se propone en el presente trabajo: QuantumSolver AI. También se ha realizado un estudio en profundidad sobre la teoría necesaria para desarrollar la posterior implementación que se encuentra descrita en el Capítulo 4. Estos conceptos han sido clave para comprender las diferencias principales entre los enfoques de los dos paradigmas de computación, clásico y cuántico, sobre las redes neuronales en el contexto de la clasificación por aprendizaje supervisado. Por un lado, las redes neuronales clásicas están formadas por estructuras de nodos que ajustan sus pesos para lograr la clasificación objetivo. Por el otro, sus homólogos cuánticos son circuitos variacionales, que ajustan los parámetros de las puertas cuánticas presentes en los mismos, logrando un resultado similar. Cabe destacar, que el modelo cuántico cuenta con la problemática de que una codificación desafortunada de la información puede llevar al inexorable fracaso del entrenamiento. También se ha hecho hincapié en la dificultad de ciertos optimizadores al utilizar un alto número de cúbits, que generan una malla plana en la que no es trivial realizar la búsqueda de una mejor solución: *barren plateau*. Por último, se han propuesto algunas soluciones recientes para estos problemas, que se encuentran presentes en la literatura científica.

En el siguiente Capítulo se hará un recorrido por las diferentes decisiones de diseño y sus justificaciones apoyadas en los fundamentos teóricos anteriormente desarrollados. Se establecerán de manera clara los objetivos que persigue la propuesta, los requisitos necesarios para el cumplimiento de los mismos, y cómo se traducen ambos en los esquemas de responsabilidad diseñados a favor de una correcta implementación (véase Capítulo 4).

3. Diseño de la plataforma

Es importante tener en cuenta todos los detalles del contexto que rodea a la plataforma inicial QuantumSolver (véase la Sección 2.1), para comprender el nuevo módulo que se describe en el presente trabajo. La plataforma QuantumSolver tiene un diseño jerárquico de clases (Escanez-Exposito, 2022c) que permite concebir su estructura y funcionalidad de manera sencilla. Sirve como núcleo de diferentes módulos, y cada uno de estos responde a diferentes funcionalidades. El nuevo módulo QuantumSolver AI es uno de ellos, que abre un abanico de posibilidades funcionales y científicas dentro del proyecto. A continuación, se muestran los pilares sobre los que se construye QuantumSolver AI.

3.1. Objetivos

Al comenzar la fase de diseño de la propuesta, se concretaron los objetivos que sirvieron de guía durante la etapa de desarrollo. Estos se encuentran, por orden de importancia, definidos en el listado posterior:

1. Creación de un módulo de QML dentro de la plataforma QuantumSolver. Las nuevas funcionalidades deben estar disponibles públicamente dentro del repositorio general de código abierto (Escanez-Exposito, 2022a).

2. Acceso al módulo por línea de comandos. El código desarrollado debe de contar, al menos, con una interfaz CLI que permita al usuario interactuar con el sistema, y visualizar los resultados, de manera cómoda.

3. Mantener una propuesta sencilla. Este aspecto se debe alcanzar en coherencia con la plataforma original, persiguiendo la realización de simplificaciones que ayuden al entendimiento del proyecto. Además, en cuanto a esta cuestión, se hace especial hincapié en los siguientes puntos:

   a) Facilidad de uso. Debe ser sencillo e intuitivo de utilizar, contando con documentación suficiente para poder acceder a todas las funcionalidades ofrecidas.

   b) Simplicidad en la contribución. También debe ser trivial añadir nuevas entidades como:

   - Conjuntos de datos.

- Modelos de inteligencia artificial.

c) Código sostenible. Este término, acuñado por Blé (2022) como homólogo del inglés *maintainable code*, hace alusión a aquel que se puede mantener fácilmente a lo largo del tiempo. El código desarrollado debe estar correctamente estructurado, manteniéndose simple y fácil de entender para que pueda adaptarse con facilidad a los cambios que surjan durante el desarrollo de la propuesta.

4. Publicación de artículos en congresos científicos. Con el fin de validar y reconocer el trabajo realizado, se presentan los resultados en distintas etapas del desarrollo a varios congresos nacionales y/o internacionales.

3.2. Requisitos

A partir de los objetivos, se identificaron los principales requisitos para lograr responder adecuadamente a los mismos. Los más relevantes, que determinaron la línea principal de desarrollo del nuevo módulo de la plataforma, se recogen en la siguiente lista:

- El módulo debe tener un diseño coherente con el resto de la aplicación. Esto significa mantener el siguiente modelo de responsabilidad, definiendo las tres siguientes estructuras de datos:
  - Entidad: tipo de dato sencillo con pequeñas funcionalidades añadidas que permite el acceso y control sobre la información que contiene.
  - Gestor: tipo de dato más complejo que administra las entidades de manera global, permitiendo su selección y parametrización.
  - Controlador global: administrador de gestores, que logra la completa coordinación de los mismos para ofrecer todas las funcionalidades implementadas.

  Este modelo se ha desarrollado siguiendo el patrón de diseño *composite* (Gamma, Helm, Johnson y Vlissides, 1995), que utiliza clases para agrupar otras más sencillas, que permiten componer objetos formando estructuras de árbol.

- Para facilitar la implementación del anterior modelo de responsabilidad, se trabajará dentro del marco del paradigma de programación orientada a objetos (Pierce, 2002; Blé Jurado, 2020). De esta manera, se desarrollan diferentes clases para cada caso de uso, categorizándolas en el tipo adecuado.

- El módulo debe permitir el entrenamiento de modelos de inteligencia artificial sobre los conjuntos de datos que se encuentren disponibles por defecto en la plataforma, así como los que pueda añadir el usuario al módulo. Además, debe existir un mecanismo único y automático de preprocesado de las características de entrada para lograr adaptar cada conjunto a cada modelo.
- Debe ser posible añadir y trabajar fácilmente con diferentes modelos de aprendizaje automático, ya sean clásicos o cuánticos.
- Es necesario definir o seleccionar un formato estándar, como ORC (Leverenz, 2018) o CSV (Shafranovich, 2005), para que los conjuntos de datos con los que entrenar los modelos se puedan añadir de manera sencilla al módulo. Esto conseguirá facilitar la labor de desarrollo, así como la accesibilidad por parte del usuario que desee añadir su propio conjunto de datos.
- Se realizará un estudio en profundidad de las tecnologías a utilizar para poder hacer uso de las mismas con rigor científico. Esto facilitará la redacción y publicación de artículos, mostrando los resultados de la propuesta y exponiendo de manera clara el marco teórico que los rodean.

3.3. Diseño

A continuación, se realiza un recorrido por los diferentes elementos que componen el diseño del módulo desarrollado, haciendo especial hincapié en sus dependencias y en cómo responden a los requisitos y objetivos anteriormente planteados.

3.3.1. Entidades

3.3.1.1. Conjunto de datos

Esta estructura debe contar con la información necesaria para representar un conjunto de datos compatible con el módulo. Para ello, cada entidad de este tipo se generará a partir de un fichero en el formato ampliamente utilizado CSV (Shafranovich, 2005). Este archivo deberá especificar, en la primera fila, los nombres de los diferentes atributos o características del conjunto de datos, separados por comas. El resto de filas indicarán, por cada una de ellas, los datos específicos separados por comas de cada instancia. El último atributo es reservado, con el nombre "class", para la etiqueta asignada a esa fila. De esta

manera, se configura una tabla en la que cada columna recoge los diferentes valores de una misma característica y cada fila representa una instancia etiquetada del conjunto de datos. Una vez se ha interpretado correctamente la información, el objeto estará disponible como elemento del sistema para poder efectuar todas las funcionalidades disponibles sobre esos datos. Se puede observar un ejemplo visual de esta estructura en la Tabla 3.1.

atr1	atr2	atr3	class
0.2	8	0.6	A
1.8	10	0.9	B
3.27	25	1.6	C
0.15	10	0.7	A
1.3	10.5	1.2	B

Tabla 3.1: Ejemplo de diseño de la estructura de datos de un conjunto de datos. Fuente: elaboración propia.

Además, se deberá permitir realizar cambios automáticos sobre estas entidades generadas, que permitan observar y trabajar únicamente con aquellas características que sean consideradas más importantes. Este proceso automático de simplificación del conjunto de datos recibirá el nombre de "vista". Las vistas facilitarán la adaptación automática de los conjuntos de datos para los modelos de ML que trabajarán con ellos.

Por último, se deberán añadir funcionalidades ligadas a estas entidades relacionadas con la representación de su información, así como de su normalización o preparación automática para la compatibilidad con los modelos.

3.3.1.2. Modelo de aprendizaje automático

Se trata de una entidad funcional que deberá ofrecer dos métodos bien diferenciados (Larose y Larose, 2014):

- Entrenamiento. En esta función se procede a ajustar el modelo en función de un subconjunto de los datos que queda reservado para el entrenamiento. De esta manera, el sistema intenta aprender a clasificar los elementos aportados del conjunto de datos.

- Evaluación. En este método se puntúa la calidad del entrenamiento previo, intentando aplicar el modelo entrenado sobre valores del conjunto de datos que han quedado

reservados para esta fase de comprobación. Este paso define si el entrenamiento ha sido correcto, al ajustarse correctamente a los datos de evaluación.

Cabe destacar que si el método de entrenamiento se excede, puede conseguir que el modelo quede muy acoplado a los datos que sirvieron para entrenarlo, resultando un fracaso en el método de evaluación. A este fenómeno se le denomina "sobreajuste" (Prechelt, 2012).

Las diferentes maneras de diseñar esos métodos de entrenamiento, tanto los clásicos como los cuánticos, definirán los distintos modelos disponibles en la plataforma.

3.3.2. Gestores

3.3.2.1. Gestor de conjuntos de datos

Este gestor deberá ser capaz de permitir la visualización y selección de los conjuntos de datos disponibles, así como de la creación y aplicación de vistas sobre los mismos.

3.3.2.2. Gestor de modelos de aprendizaje automático

De manera análoga, el gestor de modelos de aprendizaje automático permite la visualización, parametrización y selección de los modelos disponibles.

3.3.3. Controlador global

Por último, el controlador global es capaz de actuar como coordinador de gestores para lograr el entrenamiento del modelo seleccionado sobre el conjunto de datos actual, según los pertinentes gestores (véase Figura 3.1).

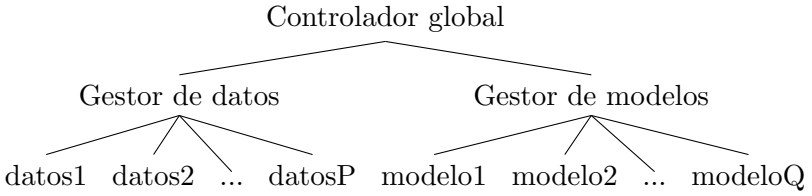

Figura 3.1: Diagrama de dependencia entre entidades, gestores y controlador global de QuantumSolver AI. Fuente: elaboración propia.

Este controlador debe contar con la visualización de un menú que permita la visualización y selección de datos y modelos, delegando esta tarea en sus respectivos gestores. Debe también permitir la creación y representación de la una vista aplicada sobre un conjunto

de datos. Una vez definido lo anterior, se debe permitir el entrenamiento, tras el que se debe obtener la evaluación del mismo.

En este Capítulo se han descrito los objetivos y requisitos que han guiado el diseño del nuevo módulo de inteligencia artificial añadido a la plataforma QuantumSolver. Los objetivos hacían alusión, de manera general, a la sencillez, coherencia e intuitividad de la propuesta. Esto se reflejó en el listado de requisitos, definiendo las estructuras de responsabilidad necesarias, así como las directrices de implementación a seguir de código sostenible. Por último, se extendió el planteamiento de desarrollo, entrando en detalles concretos sobre los elementos diseñados y sus funcionalidades recogidas. La relación entre los mismos (entidades, gestores y controlador) se puede visualizar en la representación elaborada, que conforma la Figura 3.1.

En el siguiente Capítulo se procederá a explicar en detalle la relación entre estas ideas de diseño y su transformación práctica en fragmentos funcionales de código. Se describirán las estructuras de datos desarrolladas y se visualizarán trazas de ejecución del programa principal desarrollado. Además, se discutirá en profundidad sobre la utilidad y alcance del nuevo módulo implementado.

4. Implementación realizada

4.1. Estructura de clases

Para atender al diseño expuesto en el Capítulo anterior, se han implementado las clases[1] descritas en este Capítulo, siguiendo el paradigma de orientación a objetos (Pierce, 2002). Se trata de un sencillo esquema que para cada entidad, gestor y controlador global (véase Figura 3.1), modela una clase que recoge las funcionalidades y datos que les corresponden. De esta manera, se logra distribuir coherentemente las diferentes responsabilidades de cada elemento. A continuación, se explican en profundidad las clases implementadas.

4.1.1. Entidades

Las entidades diseñadas y descritas en el Capítulo anterior, hacían referencia a un tipo de dato sencillo con pequeñas funcionalidades añadidas para el acceso y control sobre la información contenida. Estas son dos: conjunto de datos y modelo. En la descripción de sus implementaciones, se utilizarán las palabras del inglés Dataset y Model, haciendo alusión al nombre utilizado para sus clases.

4.1.1.1. Entidad Dataset

Se ha desarrollado una entidad Dataset, que representa la estructura y funcionalidad de un conjunto de datos en QuantumSolver AI. Para la construcción de la misma, se pasa como argumento el nombre del fichero que guarda los datos en formato CSV (Shafranovich, 2005).

La primera fila deben ser los nombres de los diferentes atributos separados por comas. Debajo de esta cabecera, cada fila corresponde con un punto del conjunto de datos. La última columna será la etiqueta *class*, que definirá la categoría a la que pertenece cada fila. La Figura 4.1 expone el fichero CSV análogo al ejemplo de conjunto de datos de la Tabla 3.1.

Cada columna o atributo del conjunto de datos se somete a un proceso de normalización siguiendo la expresión:

$$x' = \frac{x - min(x)}{max(x) - min(x)}, \tag{4.1}$$

[1] Una clase es una plantilla para objetos de datos y funcionalidades según un modelo predefinido.

```
atr1,atr2,atr3,class
0.2,8,0.6,A
1.8,10,0.9,B
3.27,25,1.6,C
0.15,10,0.7,A
1.3,10.5,1.2,B
```

Figura 4.1: Ejemplo de un fichero en formato CSV correspondiente a un objeto del tipo Dataset. Fuente: elaboración propia.

donde x denota la columna sin normalizar y x' el resultado. De esta manera, al valor más bajo de cada columna se le asigna el cero, y al más alto el uno. Para los valores intermedios se realiza un escalado lineal trivial. Esta normalización de las variables asegura que todas las características tengan el mismo rango, evitando que una variable con valores grandes domine sobre las demás y afecte negativamente el rendimiento del modelo. Además, se acelera la convergencia del algoritmo y mejora la interpretación de los coeficientes, facilitando así la toma de decisiones basadas en el modelo resultante (D. Singh y Singh, 2020). La implementación de la Ecuación 4.1 se puede encontrar en la Figura 4.2.

```
def __normalize(self, df):
  features = df.loc[:, df.columns != 'class']
  features = (features - features.min()) / (features.max() - features.min())
  df.loc[:, df.columns != 'class'] = features
  return df
```

Figura 4.2: Proceso de normalización en la clase Dataset. Fuente: elaboración propia.

Una vez se han normalizado los datos, se procede a dividirlos en dos conjuntos: uno de entrenamiento, con el que se ajustaran los pesos o parámetros de los diferentes modelos; y otro de prueba, que servirá para comprobar la exactitud de la clasificación. Como se muestra en la Figura 4.3 en el argumento *train_size*, se utilizará el 75 % de los datos de cada conjunto para el proceso de aprendizaje. Además, para lograr mejores resultados, por medio del argumento *stratify* se garantiza que los conjuntos de entrenamiento y prueba tienen la misma proporción de instancias de cada clase.

Para cada conjunto de datos, se calculan las variables importantes mediante un modelo

```
def __split_data(self, df):
  self.features = df.iloc[:, :-1]
  self.targets = df.iloc[:, -1:]
  self.train_data, self.test_data, self.train_targets, self.test_targets = \
      train_test_split(
        self.features,
        self.targets,
        train_size=0.75,
        stratify=self.targets
      )
```

Figura 4.3: División del conjunto de datos en la clase Dataset. Fuente: elaboración propia.

```
def __compute_important_vars(self):
  initial_decision_tree = tree.DecisionTreeClassifier(max_depth=4)
  initial_decision_tree.fit(self.train_data, self.train_targets)

  feature_importances = {}
  for i, col in enumerate(self.df.columns[:-1]):
    feature_importances[col] = initial_decision_tree.feature_importances_[i]

  important_vars = []
  for key, value in sorted(feature_importances.items(), key=itemgetter(1), reverse=True):
    if value != 0.0:
      important_vars.append(key)

  return important_vars
```

Figura 4.4: Cálculo de las variables importantes en la clase Dataset. Fuente: elaboración propia.

de árbol de decisión, como se ilustra en la Figura 4.4. Así, cada objeto del tipo Dataset tiene una lista ordenada, según la importancia de cada columna, con el nombre de las variables que parecen ser más determinantes en la clasificación, facilitando la creación de vistas. Una vista se ha implementado como la proyección, dando preferencia a aquellas variables más importantes, de un número de columnas sobre el conjunto de datos completo (véase Figura 4.5).

```
def set_view_size(self, view_size):
  self.view = self.df[self.important_vars[:view_size] + ['class']]
  self.__split_data(self.view)

def reset_view(self):
  self.view = self.df
  self.__split_data(self.view)

def get_view(self):
  return self.view
```

Figura 4.5: Métodos referentes a las vistas en la clase Dataset. Fuente: elaboración propia.

En la Figura 4.6 se muestra el constructor[2] de la clase Dataset, en el que se muestran los diferentes conceptos explicados anteriormente, como: la normalización, la división entre conjunto de datos de entrenamiento y prueba, la definición de las variables importantes y la creación de vistas (véanse Figuras 4.2, 4.3, 4.4, 4.5, respectivamente). Para observar los métodos importados que implementan algunas funcionalidades específicas se puede consultar el repositorio completo (Escanez-Exposito, 2022a). En la Sección 4.1.2.1 se expondrán las instancias generadas de esta clase.

```
class Dataset:
  def __init__(self, file_name):
    self.file_name = file_name
    self.name = file_name.split('/')[-1].split('.')[0]
    self.df = self.__normalize(pd.read_csv(file_name))

    self.__split_data(self.df)
    self.important_vars = self.__compute_important_vars()
    self.view = self.df
```

Figura 4.6: Constructor de la clase Dataset. Fuente: elaboración propia.

[2]Un constructor es el método que inicializa a una instancia (objeto) de una clase

4.1.1.2. Entidad Model

Se ha implementado la entidad Model, que establece la funcionalidad de entrenamiento y prueba para todas sus clases derivadas[3], guardando como datos: el nombre del modelo, una breve descripción, el objeto con el modelo importado de una librería externa y una lista para guardar los valores de la función objetivo. A diferencia de los objetos de la clase Dataset, que generaba un objeto (también llamado instancia de la clase) por cada conjunto de datos, en esta clase Model se hace estrictamente necesario crear una clase derivada por cada modelo a implementar. Es decir, para definir una forma adicional de entrenar un modelo, es necesario crear una nueva clase que lo desarrolle o lo importe de una librería externa. El código referente a esta clase puede ser visualizado en la Figura 4.7. Las clases derivadas desarrolladas serán expuestas en la Sección 4.1.2.2.

```
class Model(ABC):
  def __init__(self, name, desc):
    ## The name of the model
    self.name = name
    ## A short description of the model
    self.description = desc
    ## The model itself
    self.model = None
    ## Objective function values
    self.objective_func_vals = []

  ## A method to fit the model
  def fit(self, features, targets, args):
    self.set_model(args)
    self.objective_func_vals = []
    return self.model.fit(features, targets)

  ## A method to score the trained model
  def score(self, features, targets):
    return self.model.score(features, targets)
```

Figura 4.7: Implementación de la clase Model. Fuente: elaboración propia.

[3]Una clase derivada es aquella extensión de una clase principal, que además de contener los datos y funcionalidades de su superior, también implementa métodos específicos.

4.1.2. Gestores

Los gestores corresponden a un tipo de dato más complejo que administra las entidades de manera global, permitiendo su selección y parametrización. Existe uno para los conjuntos de datos y otro para los modelos. Los nombres de las clases son las de las entidades, incluyendo el sufijo Manager.

4.1.2.1. Gestor DatasetManager

Este gestor se encarga de inicializar todos los conjuntos de datos disponibles en el módulo y ponerlos a disposición del controlador global. Para ello, recorre todos los ficheros CSV que se encuentran en el directorio reservado para tal fin, y para cada uno de ellos crea una instancia de la clase Dataset. Esta funcionalidad puede observarse en la Figura 4.8.

```
1 DATASETS_PATH = 'src/ai/datasets/'
2
3 class DatasetManager:
4   def __init__(self):
5     self.datasets = [Dataset(DATASETS_PATH + file_name) \
6         for file_name in os.listdir(DATASETS_PATH)]
7     self.current_dataset = None
```

Figura 4.8: Constructor de la clase DatasetManager. Fuente: elaboración propia.

Además, contiene funcionalidades para mostrar los conjuntos de datos disponibles, seleccionarlos, obtener el conjunto actual, crear una vista y restablecerla. El código desarrollado de estas funcionalidades puede ser consultado en el repositorio (Escanez-Exposito, 2022a).

Se han escogido tres conjuntos de datos para incluirlos inicialmente en el módulo:

- Iris: es uno de los conjunto de datos más reconocidos en la literatura científica. Suele formar parte del "¡Hola mundo!"[4] de los proyectos y modelos de inteligencia artificial. Está conformado por las siguientes variables:
  - Longitud de pétalo.

[4]Se puede utilizar la expresión "¡Hola mundo!" para referirse a un primer ejercicio típico o fundamental desde el punto de vista didáctico.

  - Ancho de pétalo.
  - Longitud de sépalo.
  - Ancho de sépalo.

  Para cada flor de las 50 instancias, se indican estas cuatro características, acompañadas del atributo de clase (UCI Machine Learning, 2016b).

- Breast Cancer Detection: recoge las medias, errores estándares y mayores valores de las siguientes variables de núcleos celulares de masa mamaria:

  - Radio.
  - Textura.
  - Perímetro.
  - Área.
  - Suavidad.
  - Compacidad.
  - Concavidad.
  - Puntos cóncavos.
  - Simetría.
  - Dimensión fractal.

  Se pueden encontrar más detalles sobre estas variables en la página del conjunto de datos (UCI Machine Learning, 2016a). Además, cada una de las 569 instancias se encuentra clasificada en muestra benigna o maligna.

- Password Strength: se compone de características numéricas de contraseñas, algunas de ellas presentes en el conjunto de datos original (Bansal, 2021), que ha sido modificado exhaustivamente y reducido. Inicialmente, el conjunto de datos tenía las siguientes características de las diez mil contraseñas más utilizadas:

  - La propia contraseña.
  - La longitud total.
  - El número de letras.
  - El número de dígitos.

- El número de letras mayúsculas.
- El número de letras minúsculas.
- El número de caracteres especiales.
- El número de vocales.

Se ha realizado una reducción del mismo, ya que se trataba de un número de datos demasiado grande como para que las tecnologías de QML actuales pudieran trabajar con ellas. Se ha reducido a 250 instancias totales para este fin. Además, se han añadido otras 400 contraseñas generadas aleatoriamente con la herramienta *Dataset generation tool* (Correa-Marichal, 2023). Este conjunto de datos no se encontraba etiquetado, por lo que surge la necesidad de clasificar la fortaleza de estas 650 contraseñas totales, para lo que se utilizó el paquete *passwords* para el lenguaje de programación Rust (Magic Len, 2023). De esta manera, se añade el último atributo *class* al conjunto, que puede obtener cuatro valores distintos en función de la seguridad de cada contraseña.

Se hace especial hincapié en la facilidad de adición de un conjunto de datos al módulo, dado que requiere simplemente de añadir el fichero CSV con los datos al directorio reservado.

4.1.2.2. Gestor ModelManager

Este elemento inicializa, y pone a disposición del controlador global, todas las clases derivadas de VQC utilizando las combinaciones posibles entre:

- Circuitos de codificación:
  - ZFeatureMap, que es un circuito de evolución de puerta Z de Pauli de primer orden sin entrelazamiento (Qiskit Development Team, 2023d).
  - ZZFeatureMap, que es un circuito de evolución de puerta Z de Pauli de segundo orden con entrelazamiento (Qiskit Development Team, 2023e).
- Circuitos ansatz:
  - ExcitationPreserving, que es circuito heurístico de la función de onda que preserva la excitación (Qiskit Development Team, 2023b).

- EfficientSU2, que consiste en capas de operaciones de un solo cúbit formadas por SU(2) y entrelazamientos CX (Qiskit Development Team, 2023a).
- RealAmplitudes, que consiste en capas de operaciones de un solo cúbit formadas por rotaciones Y y entrelazamientos CX (Qiskit Development Team, 2023c).

En la Figura 4.9 se pueden observar las combinaciones implementadas y añadidas al módulo. A modo de ejemplo, se muestra implementación de la combinación entre el circuito de codificación ZZFeatureMap y el ansatz EfficientSU2 en la Figura 4.10. Cabe destacar que en la función *set_model* se establece el codificador y el ansatz correspondientes, que han sido importados previamente desde la librería de circuitos de Qiskit.

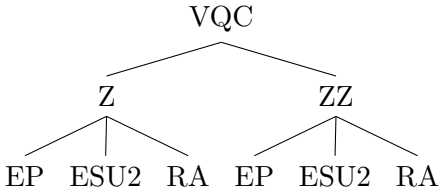

Figura 4.9: Diagrama de variantes de las clases modelo derivadas de VQC.
Fuente: elaboración propia.

Además, se ha añadido un método clásico de clasificación de máquinas de vectores de soporte, proporcionado por la librería de código libre para el aprendizaje automático Scikit-learn (learn Developers, 2023), con el fin de comparar y evaluar los resultados cuánticos.

4.1.3. Controlador global

El controlador global QuantumSolverAI es capaz de utilizar y componer los elementos anteriores para lograr una estructura coherente y accesible a todas las funcionalidades que son necesarias en el programa principal que se explicará en el siguiente apartado. La configuración de la composición del controlador global se muestra en la Figura 4.11, que es análoga a la Figura 3.1, mostrando la fidelidad entre el desarrollo realizado y el diseño inicial.

El controlador cuenta con un complejo sistema de menús, para visualizar y seleccionar los conjuntos de datos y modelos disponibles, establecer vistas, realizar visualizaciones de los conjuntos de datos con las vistas aplicadas, y realizar los entrenamientos (de manera individual o con dos modos experimentales que permiten ejecutar todos los posibles

```python
from ai.models.qs_vqc import QS_VQC
from qiskit.circuit.library import ZZFeatureMap, EfficientSU2


class QS_VQC_ZZ_ESU2(QS_VQC):
  def __init__(self):
    super().__init__('VQC_ZZ_ESU2', \
        'Variational Quantum Classifier - ZZFeatureMap - EfficientSU2')

  def set_model(self, args):
    n_qubits = args[0]
    feature_map = ZZFeatureMap(feature_dimension=n_qubits, reps=2)
    ansatz = EfficientSU2(num_qubits=n_qubits, reps=2)

    args.append(feature_map)
    args.append(ansatz)

    super().set_model(args)
```

Figura 4.10: Clase derivada de VQC con el codificador ZZFeatureMap y ansatz EfficientSU2. Fuente: elaboración propia.

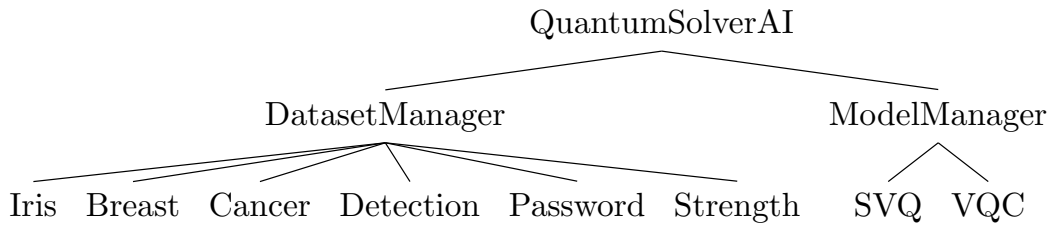

Figura 4.11: Diagrama de dependencia entre las clases que componen QuantumSolver AI. Fuente: elaboración propia.

modelos sobre un mismo conjunto de datos, o todos los modelos sobre todos los datos disponibles). El código de esta clase puede ser encontrado en el repositorio[5] (Escanez-Exposito, 2022a). En la siguiente Sección se observará una traza del programa principal desarrollado, haciendo uso de todas las funcionalidades anteriormente listadas.

4.2. Programa principal

Al ejecutar el programa mediante el comando "*python src/main_ai.py*", siguiendo las instrucciones del fichero *README* del repositorio (Escanez-Exposito, 2022a), se muestra un menú principal que permite la selección de diversas opciones, como puede apreciarse en la Figura 4.12. La visualización de los conjuntos de datos y modelos, así como se de sus descripciones (véase Figura 4.13). También existen opciones de selección, para establecer el modelo y conjunto de datos actual, que cuenta con control de errores en los parámetros introducidos por el usuario (véase Figura 4.14).

```
QuantumSolver AI
================

[1] See available Datasets
[2] See available Models
[3] Select Dataset
[4] Select Model
[9] Experimental Mode [ALL] (Train all models on all datasets)
[0] Exit

[&] Select an option:
```

Figura 4.12: Menú de QuantumSolver AI. Fuente: elaboración propia.

Una vez se ha elegido un conjunto de datos, se puede practicar una vista sobre el mismo, indicando el tamaño o número de características al que se desea restringir (véase Figura 4.15). También se puede realizar una visualización del conjunto de datos con la vista aplicada (véase Figura 4.16). En esta representación, se muestran los valores de las variables en gráficos bidimensionales, que compara las características expuestas en una relación "uno a uno". Es decir, para cada par posible de variables, muestra los diferentes puntos que conforman el conjunto de datos expresando las coordenadas correspondientes en gráficos de dos dimensiones.

[5]Enlace al repositorio: `https://github.com/jdanielescanez/quantum-solver`

```
[&] Select an option: 1

Avaliable datasets:
[1] passwords
[2] breast-cancer-detection
[3] iris

...

[&] Select an option: 2

Avaliable models:
[1]     Name: SVC
        Description: Support Vector Classifier
[2]     Name: VQC_Z_RA
        Description: Variational Quantum Classifier - ZFeatureMap - RealAmplitudes
[3]     Name: VQC_Z_ESU2
        Description: Variational Quantum Classifier - ZFeatureMap - EfficientSU2
[4]     Name: VQC_Z_EP
        Description: Variational Quantum Classifier - ZFeatureMap - ExcitationPreserving
[5]     Name: VQC_ZZ_RA
        Description: Variational Quantum Classifier - ZZFeatureMap - RealAmplitudes
[6]     Name: VQC_ZZ_ESU2
        Description: Variational Quantum Classifier - ZZFeatureMap - EfficientSU2
[7]     Name: VQC_ZZ_EP
        Description: Variational Quantum Classifier - ZZFeatureMap - ExcitationPreserving
```

Figura 4.13: Visualización de conjuntos de datos y modelos de QuantumSolver AI. Fuente: elaboración propia.

```
[&] Select an option: 3
[&] Select a dataset of the list [1 - 3]: 6
[!] The dataset must be one of the list [1 - 3]
[&] Select a dataset of the list [1 - 3]: 3
[$] iris selected

QuantumSolver AI
================

[1] See available Datasets
[2] See available Models
[3] Select Dataset
        Current Dataset: iris
[4] Select Model
[5] Set a View
[6] See current Dataset (applying the View)
[8] Experimental Mode [Dataset] (Train all models on the current dataset)
[9] Experimental Mode [ALL] (Train all models on all datasets)
[0] Exit

[&] Select an option:
```

Figura 4.14: Establecimiento del conjunto de datos en QuantumSolver AI.

Fuente: elaboración propia.

```
[&] Select an option: 5
[*] The important variables of passwords dataset are:
    ['length', 'num_upper', 'num_lower', 'num_digits', 'num_chars']
[&] Select a view size in range [1 - 5]: 5
[$] Created view:
       length  num_upper  num_lower  num_digits  num_chars      class
0    0.263158   0.000000   0.000000         0.6   0.000000  Dangerous
1    0.368421   0.000000   0.533333         0.0   0.470588       Weak
2    0.368421   0.000000   0.000000         0.8   0.000000       Weak
3    0.263158   0.000000   0.400000         0.0   0.352941  Dangerous
4    0.421053   0.000000   0.000000         0.9   0.000000       Weak
..        ...        ...        ...         ...        ...        ...
645  1.000000   0.428571   0.466667         0.1   0.764706     Strong
646  1.000000   0.428571   0.333333         0.2   0.647059     Strong
647  1.000000   0.428571   0.400000         0.2   0.705882     Strong
648  1.000000   0.500000   0.400000         0.3   0.764706     Strong
649  1.000000   0.428571   0.400000         0.2   0.705882     Strong
```

Figura 4.15: Ejemplo vista aplicada de tamaño 5 al conjunto de datos Password Strength. Fuente: elaboración propia.

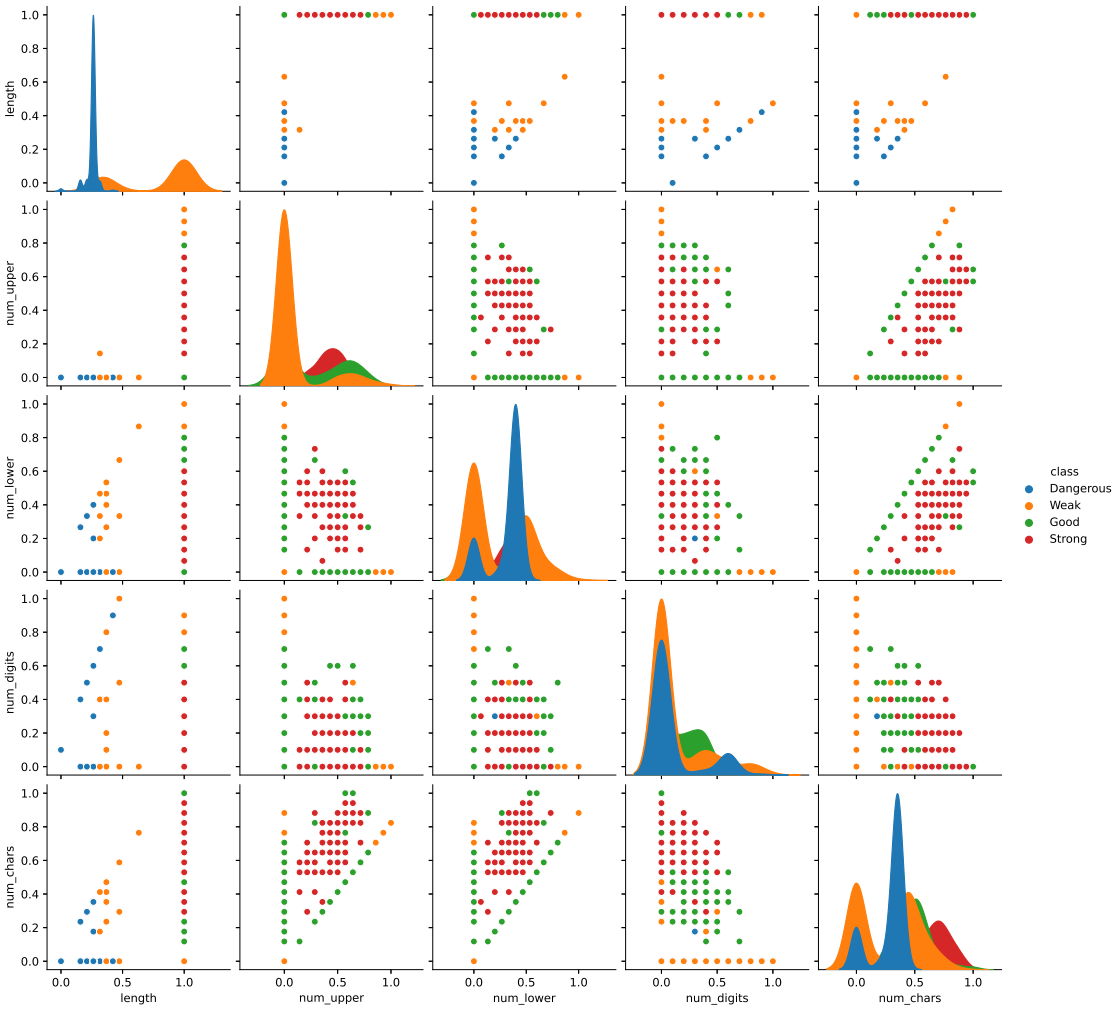

Figura 4.16: Visualización de una vista de tamaño 5 del conjunto de datos Password Strength en QuantumSolver AI. Fuente: elaboración propia.

Una vez han sido elegidos conjunto de datos y modelo, se puede realizar el entrenamiento, obteniendo los resultados como se muestra en la Figura 4.17. De manera similar, se pueden utilizar los modos experimentales para conseguir comparar todos los modelos disponibles. En el siguiente Capítulo se llevará a la práctica una ejecución exhaustiva con esta modalidad, demostrando la sencillez de uso gracias a la complejidad de la arquitectura de la propuesta. Además, se describirá en detalle el proceso a seguir para añadir un conjunto de datos propio, logrando resolver adecuadamente un problema propuesto por el usuario.

```
[&] Select an option: 7
* Executing VQC_ZZ_ESU2 - iris

[$] Experiment finished in 67.35921502113342 s!
Results (VQC_ZZ_ESU2 - iris): {'train': 0.6933333333333334, 'test': 0.68}
```

Figura 4.17: Ejemplo de entrenamiento del modelo VQC_ZZ_ESU2 sobre el conjunto de datos Iris. Fuente: elaboración propia.

5. Resultados y discusiones

Para comprobar el correcto funcionamiento del módulo implementado, se ha entrenado cada modelo disponible sobre los tres conjuntos de datos incluidos en la plataforma, obteniendo sus resultados de evaluación en los subconjuntos de entrenamiento y prueba. Por cada sección de este Capítulo, se expondrá un conjunto de datos y las relaciones entre las variables implicadas, discutiendo los resultados obtenidos de todos los modelos empleados. Además, se expone un caso de uso, mostrando el proceso de adición de un conjunto arbitrario a la plataforma, así como las discusiones pertinentes. Cabe destacar que, debido a las limitaciones en el uso del hardware cuántico real gratuito de IBM, las ejecuciones han sido realizadas en un simulador local. La posible extensión a realizar para añadir esta característica supondría un ligero cambio en la forma actual de ejecutar los algoritmos. Se debe realizar una actualización para hacer compatible el componente QExecute, encargado de las ejecuciones en QuantumSolver, con el servicio de Qiskit Runtime (Escanez-Exposito, 2022b; IBM, 2015).

5.1. Conjunto de datos Iris

El conjunto de datos Iris (UCI Machine Learning, 2016b) está conformado por cuatro variables continuas (las longitudes y anchos en centímetros de los pétalos y sépalos de flores); y una categórica (la clase o especie de cada flor: setosa, versicolour o virginica). En la Figura 5.1 se puede observar la representación generada mediante QuantumSolver AI de este conjunto de datos, donde destacan las agrupaciones formadas por los puntos de una misma clase de manera generalizada para todos los pares de variables posibles. Esto demuestra una fuerte correlación entre las variables y las respectivas clases de los puntos que conforman las agrupaciones. Por ello, parece un conjunto de datos adecuado para intentar realizar la clasificación de los mismos.

Se exponen los resultados de todos los modelos recogidos en la plataforma en la Tabla 5.1. Para cada modelo se muestra una puntuación entre 0 y 1, que determina el nivel de éxito del modelo frente a los datos de entrenamiento entrenamiento y prueba. El primer modelo (SVC) hace referencia al método clásico de clasificación de máquinas de vectores de soporte, el resto de modelos son clasificadores cuánticos varacionales (VQC). Estos últimos siguen la siguiente nomenclatura: VQC_X_A, donde X es un circuito codificador; y A,

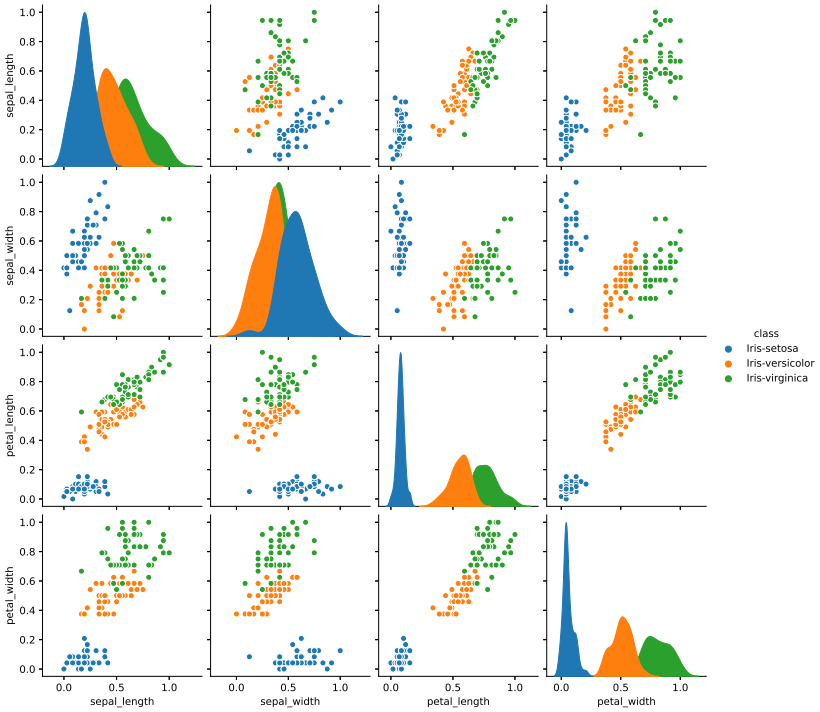

Figura 5.1: Representación del conjunto de datos Iris. Fuente: elaboración propia.

Modelo	Entrenamiento	Prueba
SVC	0.96	0.95
VQC_Z_RA	0.91	0.84
VQC_Z_ESU2	0.85	0.95
VQC_Z_EP	0.71	0.74
VQC_ZZ_RA	0.66	0.76
VQC_ZZ_ESU2	0.67	0.61
VQC_ZZ_EP	0.45	0.45

Tabla 5.1: Resultados del modo experimental de QuantumSolver AI sobre el conjunto de datos Iris. Fuente: elaboración propia.

un circuito ansatz. X puede ser Z o ZZ, según el tipo utilizado de codificador arbitrario proporcionado por Qiskit, refiriéndose a ZFeatureMap y ZZFeatureMap, respectivamente (Qiskit Development Team, 2023d, 2023e). Los ansatzs utilizados vienen definidos por A: RA, RealAmplitudes; ESU2, EfficientSU2; y EP, Excitation Preserving (Qiskit Development Team, 2023a, 2023b, 2023c).

Se observa una clara ventaja en el modelo clásico. Para el conjunto de prueba utilizado en la comparación de los modelos, obtiene una amplia mejoría respecto a la mayoría de modelos, consiguiendo una puntuación casi perfecta (0.95). Sin embargo, se puede observar como utilizando el codificador ZFeatureMap, carente de entrelazamiento cuántico, el modelo se ajusta mejor a este conjunto de datos, logrando un valor prácticamente igual al resultado clásico con el ansatz EfficientSU2 (0.95). Un resultado destacable es también el del ansatz RealAmplitudes, utilizando el mismo codificador (0.84). Si bien el resto de modelos muestran cierto aprendizaje (dado que al existir tres clases, una clasificación aleatoria obtendría aproximadamente una marca de 0.33), se puede observar que no consiguen puntuaciones demasiado altas.

Se hace hincapié la mejora obtenida, gracias a QuantumSolver AI, frente al tutorial de IBM para la aplicación de QML en conjuntos de datos reales que no sean hayan sido preparados específicamente para QML (IBM, 2023d). En esta referencia se expone que, como se esperaba, los modelos clásicos obtienen mejores resultados que sus homólogos cuánticos. La justificación dada es que el aprendizaje automático clásico tiene una larga trayectoria que el QML no ha recorrido todavía, por lo que aún tiene que alcanzar ese nivel de madurez. En estos experimentos realizados se reduce el número de características (lo que en QuantumSolver AI corresponde con una vista), exponiendo la bajada del rendimiento de todos los modelos. Otra observación es que un cambio de ansatz puede conducir a mejores o peores resultados, como se ha visto en la comparación realizada en la Tabla 5.1. Sin embargo, como se comentó en la Sección 2.2.3, el impacto del esquema de codificación elegido parece todavía mayor. Por ello, explorando el codificador ZFeatureMap, que no fue propuesto en el tutorial por IBM, se logró la tasa de acierto de 0.95. El mejor resultado cuántico expuesto en la referencia obtuvo solo un 0.87 de exactitud.

5.2. Conjunto de datos Breast Cancer Detection

El trabajo realizado con este conjunto de datos durante las fases iniciales del módulo de QuantumSolver AI, así como los resultados obtenidos, impulsó la publicación del artículo "Classical vs. Quantum Machine Learning for Breast Cancer Detection" en la *19th International Conference on the Design of Reliable Communication Networks* (DRCN 2023, véase Apéndice A).

En los datos se recogen las medias, errores estándares y mayores valores de las variables

de núcleos celulares de masa mamaria: radio, textura, perímetro, área, suavidad, compacidad, concavidad, puntos cóncavos, simetría y dimensión fractal (UCI Machine Learning, 2016a). Se trata de 569 puntos con 30 dimensiones cada uno, clasificados como muestras benignas (B) o malignas (M).

En la Figura 5.2, se expone la representación generada por QuantumSolver AI de una vista de este conjunto de datos de tamaño 5. Se ha practicado esta reducción, dado que las 30 dimensiones presentes inicialmente eran inabarcables por el simulador utilizado para el entrenamiento.

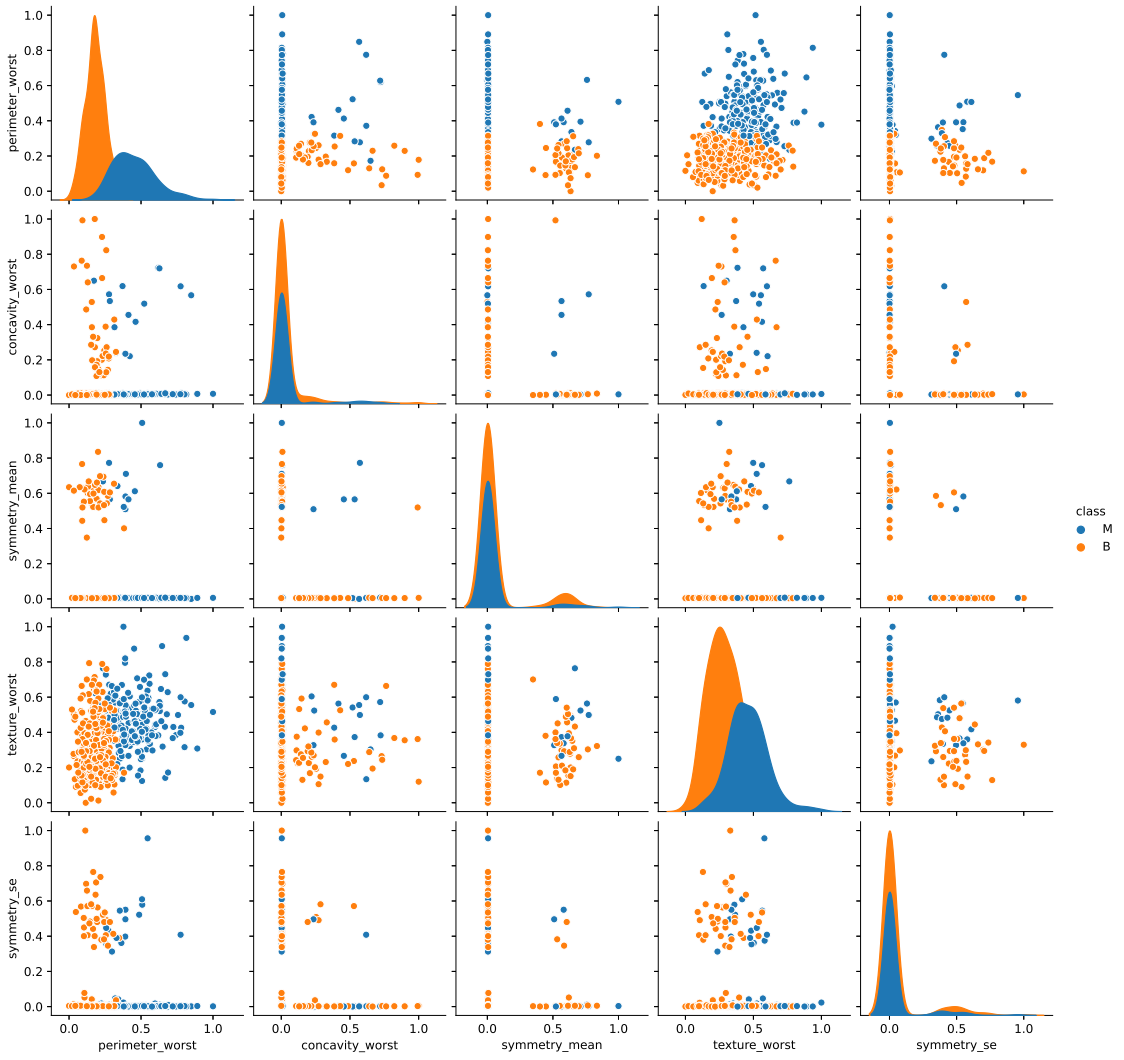

Figura 5.2: Representación de la vista de tamaño 5 sobre el conjunto de datos Breast Cancer Detection. Fuente: elaboración propia.

De manera similar al conjunto de datos anterior, aunque de manera menos clara, se observan también esa separación entre las clases en función de los valores que toman las

variables. En algunas parejas, se obtienen nubes de puntos más difusas que prestan a confusión. Sin embargo, en la columna correspondiente a la variable *perimeter_worst*, se puede apreciar cómo parece existir un umbral cercano a 0.35 que delimita la clase a la que pertenece cada punto. Este comportamiento también es adecuado para el planteamiento de la clasificación.

En la Tabla 5.2 se exponen los resultados obtenidos para estos datos, siguiendo la misma representación y nomenclatura anterior.

Modelo	Entrenamiento	Prueba
SVC	0.96	0.93
VQC_Z_RA	0.84	0.88
VQC_Z_ESU2	0.83	0.87
VQC_Z_EP	0.76	0.78
VQC_ZZ_RA	0.77	0.77
VQC_ZZ_ESU2	0.83	0.83
VQC_ZZ_EP	0.79	0.74

Tabla 5.2: Resultados del modo experimental de QuantumSolver AI sobre el conjunto de datos Breast Cancer Detection. Fuente: elaboración propia.

En este caso, se puede apreciar un comportamiento similar al previo con el conjunto de datos Iris. Sin embargo, cabe resaltar que en este solo se definen dos clases, por lo que una clasificación azarosa acertaría aproximadamente con una tasa del 50 %. De nuevo, el esquema clásico obtiene la mejor puntuación (0.93). El codificador ZFeatureMap produce un mejor ajuste a este conjunto de datos por parte del modelo, consiguiendo un valor de 0.88 con el ansatz RealAmplitudes y de 0.87 con el EfficientSU2. Este último circuito variacional, también obtiene esta buena marca con el codificador ZZFeatureMap (0.83). El resto de resultados, si bien es cierto que no son buenas aproximaciones, tampoco corresponden con una clasificación azarosa, demostrando cierto éxito en el proceso de entrenamiento.

5.3. Conjunto de datos Password Strength

La adición de este conjunto de datos al módulo, ha supuesto el envío del artículo "Password Strength Analysis through Supervised Quantum Machine Learning" al *Key Manage-*

ment and Key Recover Special Issue of Journal of Surveillance, Security and Safety (JSSS, véase Apéndice A), que en el momento en el que se redacta el presente documento se encuentra en fase de revisión. En este artículo se describe en profundidad la generación de los datos a partir de un conjunto público de características numéricas de las contraseñas más utilizadas (Bansal, 2021), que ha sido modificado exhaustivamente y reducido. Estas características, inicialmente eran: la propia contraseña, la longitud total y el número de letras, dígitos, letras mayúsculas y minúsculas, caracteres especiales y vocales. Se ha realizado una reducción a 250 instancias del mismo, para adaptarlo al simulador utilizado. Además, otras 400 contraseñas han sido generadas con la herramienta *Dataset generation tool* (Correa-Marichal, 2023). El conjunto de datos final de 650 contraseñas no se encontraba etiquetado, por lo que surge la necesidad de clasificar la fortaleza de estas, para lo que se utilizó el paquete *passwords* para Rust (Magic Len, 2023). Con ello se añade la última columna *class* al conjunto, que tiene cuatro valores posibles que determinan la seguridad de cada contraseña. Se muestra la representación del conjunto completo de datos en la Figura 5.3.

En este caso, los datos muestran una distribución y estructura distinta a los anteriores. En lugar de agruparse como nubes de puntos, lo hacen a lo largo de una malla discreta, debido a la naturaleza de las variables. Al observar las diferentes columnas, se pueden obtener algunas conclusiones. Por ejemplo, en la primera columna, los gráficos referentes a la relaciones de la variable de longitud de las contraseña muestran que si la contraseña es muy corta, se clasifica como peligrosa (azul). Si la contraseña es muy de longitud intermedia, se denomina débil (amarillo). En otro caso, en función del resto de variables, puede tomar los valores: débil, buena (verde) o fuerte (rojo). En otros gráficos como el que se genera al comparar el número de letras con el número de mayúsculas, se puede comprobar que si todos los caracteres son de ese tipo (mayúsculas) o no lo contienen en absoluto, la fortaleza de ese punto se ve penalizada. Por ello, que en los triángulos generados en estos casos se puede observar como el nivel de fortaleza corresponde con la parte interna del triángulo, y el perímetro está conformado por aquellas instancias más débiles.

En la Tabla 5.3 se exponen los resultados de este último modo experimental ejecutado, entrenando todos los modelos sobre el conjunto de datos Password Strength.

El método clásico muestra una puntuación de 0.88, siendo esta la peor puntuación obtenida de este modelo frente a la de los demás conjuntos de datos. Una peor marca podría

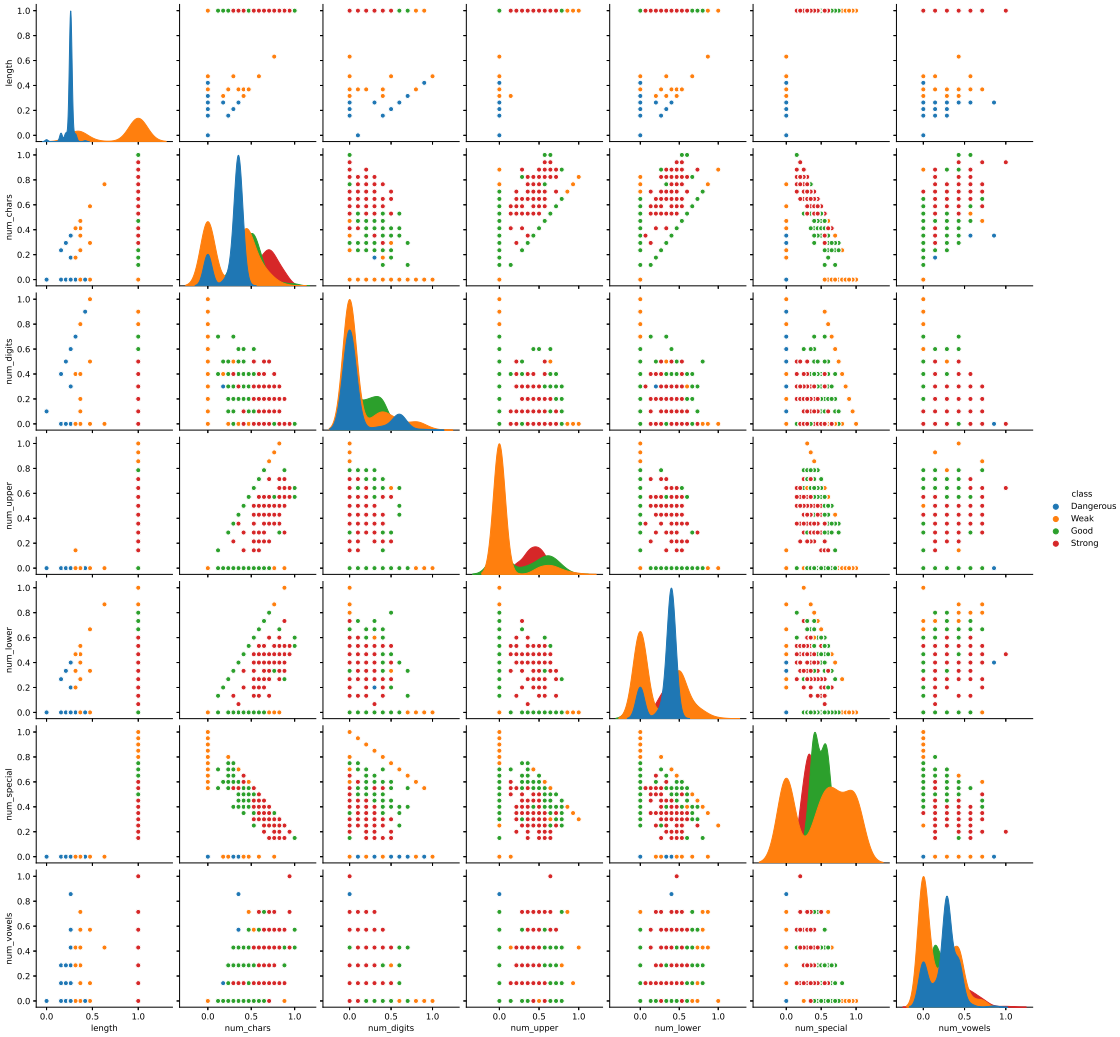

Figura 5.3: Representación del conjunto de datos Password Strength.

Fuente: elaboración propia.

Modelo	Entrenamiento	Prueba
SVC	0.89	0.88
VQC_Z_RA	0.64	0.63
VQC_Z_ESU2	0.54	0.49
VQC_Z_EP	0.27	0.28
VQC_ZZ_RA	0.56	0.58
VQC_ZZ_ESU2	0.53	0.50
VQC_ZZ_EP	0.52	0.45

Tabla 5.3: Resultados del modo experimental de QuantumSolver AI sobre el conjunto de datos Password Strength. Fuente: elaboración propia.

indicar una descompensación en el conjunto de datos generado, que estaría empeorando el trabajo de clasificación. En el caso cuántico, el mejor resultado viene dado de nuevo por el codificador ZFeatureMap y el ansatz RealAmplitudes, logrando una puntuación de 0.63. El siguiente mejor valor es conseguido por este mismo ansatz, pero con el codificador ZZFeatureMap (0.58). Cabe destacar que al existir cuatro posibles clases, la clasificación aleatorio obtendría una evaluación de 0.25.

Tras exponer y analizar la ejecución de los modelos actualmente disponibles en la plataforma sobre los conjuntos de datos predefinidos, se muestra una guía de uso del módulo para añadir datos y modelos por parte del usuario.

5.4. Caso de uso

En esta Sección se desarrolla un tutorial para añadir tanto modelos como conjuntos de datos, por parte del usuario, al módulo QuantumSolver AI. Es realmente importante mantener la simplicidad en este trámite, dado que uno de los objetivos principales es facilitar la realización de contribuciones a la plataforma. Se hará un recorrido por los pasos a seguir, mientras se destaca la sencillez del proceso.

5.4.1. Adición de un modelo

Para agregar un nuevo modelo a QuantumSolver AI, se debe definir una clase siguiendo las instrucciones de la Sección 3.3.1.2. En el caso de modelos cuánticos (véase Sección

4.1.2.2), esto requiere de la definición de dos componentes funcionales: la codificación y el ansatz. Estos dos métodos pueden ser desarrollados por el propio programador, o importados desde una librería externa. Qiskit tiene una gran cantidad de circuitos implementados que pueden servir para esta labor (Qiskit Development Team, 2023a, 2023b, 2023c, 2023d, 2023e).

En la Figura 4.10 se puede observar cómo se importan los circuitos ZZFeatureMap y EfficientSU2 para crear una clase derivada de QS_VQC, definiendo el modelo mediante el método *set_model*. Añadiendo el fichero que contiene a esta clase desarrollada al directorio "*src/ai/models*", e importándola en el ModelManager, creando una instancia en la lista de modelos disponibles, quedaría un modelo funcional de manera local para el usuario. Se muestra en la Figura 5.4 la adición en la última importación, y en el último elemento de la lista de modelos, de un supuesto modelo con codificador X y ansatz A.

```
from ai.models.qs_svc import QS_SVC
from ai.models.qs_vqc_z_ra import QS_VQC_Z_RA
from ai.models.qs_vqc_z_esu2 import QS_VQC_Z_ESU2
from ai.models.qs_vqc_z_ep import QS_VQC_Z_EP
from ai.models.qs_vqc_zz_ra import QS_VQC_ZZ_RA
from ai.models.qs_vqc_zz_esu2 import QS_VQC_ZZ_ESU2
from ai.models.qs_vqc_zz_ep import QS_VQC_ZZ_EP
from ai.models.qs_vqc_x_a import QS_VQC_X_A # Última importación

class ModelManager:
  def __init__(self):
    self.current_model = None
    self.models = [
      QS_SVC(),
      QS_VQC_Z_RA(),
      QS_VQC_Z_ESU2(),
      QS_VQC_Z_EP(),
      QS_VQC_ZZ_RA(),
      QS_VQC_ZZ_ESU2(),
      QS_VQC_ZZ_EP(),
      QS_VQC_X_A(), # Último elemento de la lista
    ]
```

Figura 5.4: Constructor de la clase ModelManager. Fuente: elaboración propia.

Para realizar la contribución a QuantumSolver AI y que el modelo añadido de manera local sea visible públicamente dentro de la plataforma, se debe seguir la metodología *Pull Request* expuesta en la página de referencia de GitHub (2023).

5.4.2. Adición de un conjunto de datos

Para añadir un conjunto de datos a la plataforma, se deberá preparar un fichero en formato CSV (Shafranovich, 2005) siguiendo las instrucciones de la Sección 3.3.1.1. Es decir, el archivo deberá contener una cabecera en la primera fila que contenga los nombres de las características de los datos, siendo el último atributo el reservado para la etiqueta. Este deberá tener el nombre "class". A continuación, en cada fila se debe describir un punto del conjunto de datos, separando los valores de los atributos por comas, como se muestra en la Figura 4.1. Una vez que se ha generado este fichero con el formato correcto, el único paso adicional es añadirlo al directorio reservado para alojar los archivos de datos: "*src/ai/datasets*".

A partir de este momento, el conjunto de datos pasa a formar parte de la plataforma de manera local. Es decir, el usuario podrá ejecutar los modelos disponibles sobre el conjunto de datos propuesto, haciendo uso de todas las funcionalidades brindadas (visualización de los datos y creación de vistas, entre otras).

Se procede a exponer un ejemplo práctico de los pasos llevados a cabo en un conjunto de datos arbitrario de prueba denominado Example. El primer paso es añadir el fichero CSV al directorio reservado "*src/ai/datasets*". En la Figura 5.5 se puede observar el fichero en el formato correcto.

```
atr1,atr2,atr3,class
97,89,79,A
0,5,4,B
97,93,95,A
10,15,2,B
91,81,71,A
79,97,100,A
30,15,11,B
24,7,12,B
```

Figura 5.5: Conjunto de datos Example en formato CSV. Fuente: elaboración propia.

A continuación, se ejecuta el programa principal con el comando "*python src/main_ai.py*". Indicando la primera opción del menú principal para listar los diferentes conjuntos de da-

tos disponibles, se debe observar los datos de Example; y con la tercera opción, se puede seleccionar (véase Figura 5.6).

```
$ python src/main_ai.py

QuantumSolver AI
================

[1] See available Datasets
[2] See available Models
[3] Select Dataset
[4] Select Model
[9] Experimental Mode [ALL] (Train all models on all datasets)
[0] Exit

[&] Select an option: 1

Avaliable datasets:
[1] passwords
[2] breast-cancer-detection
[3] example
[4] iris

QuantumSolver AI
================

[1] See available Datasets
[2] See available Models
[3] Select Dataset
[4] Select Model
[9] Experimental Mode [ALL] (Train all models on all datasets)
[0] Exit

[&] Select an option: 3
[&] Select a dataset of the list [1 - 4]: 3
[$] example selected
```

Figura 5.6: Listado de conjuntos de datos disponibles y selección. Fuente: elaboración propia.

Obtener una representación gráfica del conjunto es posible gracias a la sexta opción del menú principal, como se muestra en la Figura 5.7. Tras ejecutar el modo experimental para este conjunto de datos con la octava opción, se obtienen los resultados expuestos en la Figura 5.8.

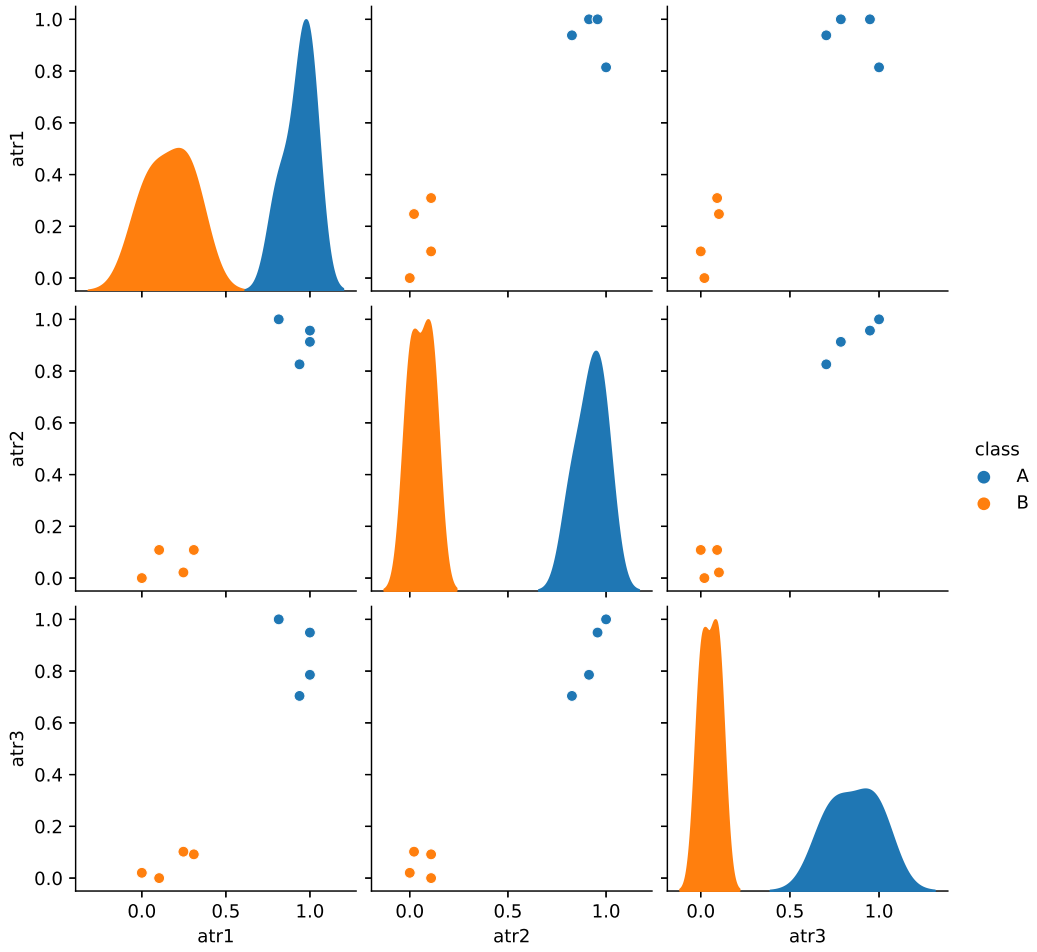

Figura 5.7: Representación del conjunto de datos Example. Fuente: elaboración propia.

```
[&] Select an option: 8

[*] Executing SVC - example

[$] Experiment finished in 0.29169631004333496 s!
Results (SVC - example): {'train': 1.0, 'test': 1.0}

[*] Executing VQC_Z_RA - example

[$] Experiment finished in 5.57772421836853 s!
Results (VQC_Z_RA - example): {'train': 1.0, 'test': 1.0}

[*] Executing VQC_Z_ESU2 - example

[$] Experiment finished in 6.082913398742676 s!
Results (VQC_Z_ESU2 - example): {'train': 1.0, 'test': 1.0}

[*] Executing VQC_Z_EP - example

[$] Experiment finished in 6.178684234619141 s!
Results (VQC_Z_EP - example): {'train': 0.5, 'test': 0.5}

[*] Executing VQC_ZZ_RA - example

[$] Experiment finished in 6.220736742019653 s!
Results (VQC_ZZ_RA - example): {'train': 1.0, 'test': 1.0}

[*] Executing VQC_ZZ_ESU2 - example

[$] Experiment finished in 7.013388395309448 s!
Results (VQC_ZZ_ESU2 - example): {'train': 1.0, 'test': 1.0}

[*] Executing VQC_ZZ_EP - example

[$] Experiment finished in 7.147686004638672 s!
Results (VQC_ZZ_EP - example): {'train': 0.16666666666666666, 'test': 0.5}

Dataset: example
---------------

                         Model | Train Score | Test Score
============================================================
                           SVC |        1.00 |       1.00
                        VQC_Z_RA |        1.00 |       1.00
                       VQC_Z_ESU2 |        1.00 |       1.00
                        VQC_Z_EP |        0.50 |       0.50
                       VQC_ZZ_RA |        1.00 |       1.00
                      VQC_ZZ_ESU2 |        1.00 |       1.00
                        VQC_ZZ_EP |        0.17 |       0.50
```

Figura 5.8: Resultados de ejecución sobre los datos Example. Fuente: elaboración propia.

De esta manera, se puede experimentar fácilmente con el conjunto de datos, realizando cambios para volver a cargarlo en la plataforma y ver cómo varían los resultados de las ejecuciones. Algunos de estas variaciones podrían ser pruebas con diferentes balanceos o reducciones el tamaño del número de puntos presente en el mismo. Si, por el contrario, se quiere conseguir la publicación del conjunto de datos propuesto de manera definitiva, igual que con los modelos, se deberá llevar a cabo mediante la metodología *Pull Request* explicada en la página de documentación oficial de GitHub (2023). De esta manera, el código desarrollado pasa a forma parte del repositorio de QuantumSolver.

5.5. Discusiones

En este Capítulo se han mostrado los resultados de ejecución del modo experimental del módulo QuantumSolver AI, ejecutando todos los modelos disponibles (véase Sección 4.1.2.2) sobre todos los conjuntos de datos (véase Sección 4.1.2.1). Se ha discutido la adaptación de todos los datos utilizados para viabilidad de practicar la clasificación, concretamente dentro de las limitaciones impuestas por las tecnologías actuales con las que se ha trabajado, es decir, un simulador local. Se han comparado los diferentes entrenamientos de los modelos, mostrando que, en general, el codificador arbitrario ZFeatureMap y el ansatz RealAmplitudes ofrecen buenos resultados.

Tras reflexionar sobre los resultados, se hace hincapié en la conveniencia de brindar al módulo desarrollado de la funcionalidad de personalización de hiperparámetros[1] para cada modelo. En la versión actual, los hiperparámetros tienen un valor constante predefinido de dos repeticiones por circuito de codificación y por ansatz. La elección del correcto establecimiento de estas importantes variables desempeña el mismo papel crítico en el QML que en el aprendizaje automático clásico. La búsqueda de hiperparámetros óptimos o más eficientes puede llevar mucho tiempo, por lo que sería conveniente aplicar las mismas técnicas que se utilizan en el aprendizaje automático clásico, como búsqueda de cuadrícula, búsqueda aleatoria o enfoques más sofisticados (Bergstra y Bengio, 2012).

El conjunto de datos referente a la fortaleza de las contraseñas puede ser mejorado reduciendo el número de instancias repetidas, aportando una mayor variedad, y utilizando diferentes métodos de generación de claves. Además, el método utilizado para evaluar la

[1] Los hiperparámetros son variables ajustables que permiten controlar el proceso de entrenamiento de un modelo. Por ejemplo, en un circuito cuántico variacional, un hiperparámetro puede ser el número de repeticiones del circuito de codificación o del ansatz.

seguridad de las contraseñas también podría ser estudiado en profundidad para lograr una distribución más acorde a los problemas de clasificación típicos. Variando estos aspectos intrínsecos a la información que se desea clasificar, así como los parámetros que determinan en gran parte el éxito o fracaso del entrenamiento, se estima que se puedan conseguir resultados más cercanos a los que aporta actualmente el aprendizaje automático clásico, e incluso mejorarlos.

Se puede observar como todas las funcionalidades descritas en el Capítulo son realmente intuitivas, tanto para el usuario que desea realizar un entrenamiento con Quantum-Solver AI, como para aquel programador que desee contribuir con la librería (por ejemplo, añadiendo su propio conjunto de datos o modelo). La estructura general es coherente con los módulos anteriores de la plataforma, haciendo hincapié en el uso de la programación orientada a objetos (Pierce, 2002). La visualización de gráficos y resultados es bastante sencilla y llamativa, pudiendo comparar fácilmente el éxito de los diferentes entrenamientos realizados.

6. Conclusiones

En el presente trabajo se ha expuesto un nuevo módulo de inteligencia artificial cuántica dentro de la plataforma de código abierto QuantumSolver. Se ha realizado un estudio en profundidad del estado actual del aprendizaje automático, haciendo especial hincapié en el marco del QML. Para ello se han mostrado conceptos iniciales como son las redes neuronales clásicas y cuánticas, concretando los diferentes modos de codificación de la información en estas últimas. También se ha estudiado el proceso de ajuste dentro del entrenamiento. Se han realizado las discusiones oportunas entre las similitudes y diferencias de ambos paradigmas, así como de sus limitaciones y posibles soluciones.

La novedosa aportación realizada incluye una interfaz por línea de comandos para interactuar con los elementos de código desarrollado. Para realizar esta implementación, ha sido necesario plantear un diseño de responsabilidades poniendo en práctica principios de código sostenible. Así, se listó de manera exhaustiva los objetivos, como la propia creación del módulo, el mantenimiento sencillo de la propuesta, o la publicación de artículos a congresos científicos; y los requisitos, como la coherencia del módulo dentro de la plataforma QuantumSolver, la facilidad en la contribución externa, o el preprocesamiento automatizado, para que cualquier conjunto de datos sea adaptado a los modelos de aprendizaje incluidos. En el diseño especificado se sugirieron diferentes elementos de responsabilidad: entidades de conjuntos de datos y modelos, para almacenar la información de los mismos, brindándoles cierta funcionalidad local; gestores, para permitir la visualización, personalización y selección de datos y modelos; y controlador global, capaz de coordinar los gestores, logrando poner a disposición del usuario todas las funcionalidades de la plataforma. Gracias a la exitosa puesta en práctica de este diseño, se lograron satisfacer todos los objetivos y requisitos propuestos.

La implementación realizada incluye una traducción directa entre los elementos de diseño y las clases desarrolladas. Esto garantiza la adhesión de la propuesta al marco de la programación orientada a objetos, acercándola al deseado entorno de código sostenible. Las funcionalidades de cada entidad se implementaron de manera ordenada, en relación a sus responsabilidades. Se ha documentado todo el proceso de desarrollo, haciendo especial hincapié en los detalles más importantes propuesta, como la adición automática de conjuntos de datos, la representación de los mismos, o el alcance de la interacción con la plataforma, brindando un útil tutorial de ejecución.

Además, para cada conjunto de datos contenidos inicialmente en la plataforma (Iris, Breast Cancer Detection y Password Strength), se han ejecutado todos los modelos disponibles. Se ha realizado una discusión en profundidad sobre los resultados obtenidos, así como de los análisis e interpretaciones de las visualizaciones de los conjuntos de datos. Gracias a la obtención de estos resultados, se ha podido realizar la publicación de artículos en congresos científicos (véase Apéndice A), logrando alcanzar el último objetivo. Un tutorial exhaustivo para la contribución con la librería ha sido desarrollado, mostrando los pasos individuales a realizar para la adición de un nuevo modelo o conjunto de datos, por parte de un usuario inexperto.

De esta manera, el módulo se ajusta a todo el abanico de público potencial. Si bien la plataforma en conjunto, gracias a la interfaz web, intenta atraer al público en general, el módulo desarrollado cuenta con usuarios más específicos. Por un lado, existe un perfil científico interesado en investigar en materia de aprendizaje supervisado cuántico, gracias a la automatización de experimentos propuestos por la herramienta. Desde esta perspectiva, tiene un gran potencial al poder ajustarse a cualquier conjunto de datos que pueda ser añadido local o globalmente. Esto permite un acercamiento al QML desde cualquier sector científico-técnico, obteniendo una sólida respuesta por parte de QuantumSolver AI, al permitir la generación de resultados de manera sencilla. Por otro lado, el módulo también está pensado para un perfil de desarrollador, abriéndose a contribuciones activas a la librería. Al tratarse de una plataforma de código abierto, es indispensable tener en cuenta a la posible comunidad que pueda formarse alrededor de la misma, y que podría llegar a mantenerla. Alternativamente, la herramienta está alcanzando, por profundidad y amplitud, un grado de madurez que la capacita para poder convertirse en un producto comercial. Por ejemplo, se podría considerar el desarrollo de dos versiones de QuantumSolver: una académica, centrada en investigación, que sería gratuita para universidades y otros centros educativos; y una empresarial, focalizada en la creación de soluciones cuánticas a determinados problemas de los sectores interesados (industria, transporte, farmacéutica, etc.), que sería de pago para las compañías interesadas. Conseguir que el proyecto siga creciendo adecuadamente, requiere de la reflexión sobre algunos aspectos, como la idea de la ingeniería del software cuántico (Serrano, Perez-Castillo y Piattini, 2022). En esta, se intentan aplicar los principios de ingeniería sobre el nuevo paradigma cuántico, consiguiendo nuevos planteamientos ante algunos problemas que no se encontraban en el caso clásico. Por ejemplo, algunas de estas cuestiones son:

- Fundamentos de informática cuántica: la comprensión de los principios básicos de la computación cuántica es primordial para aplicarlos desde el punto de vista de la ingeniería (Nielsen y Chuang, 2000).

- Lenguajes de programación cuánticos: existen varias librerías y lenguajes de programación diseñados específicamente para codificar algoritmos cuánticos, como Qiskit (IBM, 2023c), Braket (Amazon, 2018), Pennylane (Xanadu, 2023a), D-Wave (D-Wave Quantum Inc., 2023), Q Sharp (Microsoft, 2023), Cirq (Google, 2023) y Forest (Rigetti, 2019). Estos proporcionan abstracciones y constructos específicos para trabajar con sistemas cuánticos, permitiendo la simulación y ejecución de programas en ordenadores cuánticos reales o simulados. En el último Capítulo se hará mención de alguno de estos lenguajes y del papel que puede jugar en el futuro del proyecto.

- Diseño y optimización de algoritmos cuánticos: la ingeniería de software cuántico implica diseñar algoritmos útiles que aprovechen las ventajas de este tipo de sistemas (Serrano et al., 2022), como el algoritmo de factorización de números enteros de Shor (1994) o el algoritmo de búsqueda de Grover (1996). También conlleva la optimización de estos algoritmos para mejorar su eficiencia y rendimiento.

- Simulación y depuración: debido a la naturaleza experimental y en desarrollo de los ordenadores cuánticos, es esencial contar con herramientas de simulación y depuración para probar y verificar programas antes de ejecutarlos en hardware cuántico real (Li et al., 2020). Estas herramientas permiten simular el comportamiento de los algoritmos y verificar su corrección. La parte de testeo también es realmente importante, y marca algunas dificultades adicionales respecto al paradigma clásico. Por ejemplo, supone un nuevo reto ingeniar pruebas para comprobar la autenticidad de una generación aleatoria de números (Marangon et al., 2018; Lunghi et al., 2015).

- Interfaz con hardware cuántico: al interactuar con este hardware, ya sea en laboratorios o mediante el uso de servicios de computación cuántica en la nube, se deben enviar programas cuánticos a computadores cuánticos reales. Por tanto, aparece de manera natural la tarea de gestión de la ejecución de procesos y la optimización del uso de recursos cuánticos (Reilly, 2015).

Debido al planteamiento inicial de la propuesta, algunos aspectos han quedado fuera del alcance de la misma, como el aprendizaje automático cuántico no supervisado, la

resolución de problemas de optimización o el uso de otros lenguajes y tecnologías. Por ello, se realizará la discusión pertinente en el siguiente Capítulo, brindando una opinión crítica sobre las nuevas líneas de investigación posibles gracias al módulo de QuantumSolver AI.

7. Trabajo futuro

Existe una enorme cantidad de variadas posibles funcionalidades y desarrollos de la propuesta, por lo que surge la necesidad de acotarla, restringiéndola para el presente trabajo a los objetivos y requisitos listados anteriormente (véase Capítulo 3). Las posibilidades resultan prácticamente infinitas, tanto en profundidad como en amplitud, debido a la naturaleza de código abierto que caracteriza a la plataforma. Por ello, se plantean algunas ideas de trabajos futuros, con el ánimo de que sean realizadas fuera del alcance descrito. Los planteamientos de las líneas sugeridas están divididas en dos grupos: visualización y funcionalidad.

Por un lado, la representación de la información juega un papel fundamental en la accesibilidad de la propuesta. La interfaz por línea de comandos implementada, si bien responde a los objetivos y requisitos iniciales, no termina de ajustarse al público general, pudiendo ser considerada un impedimento a la hora de interactuar con los recursos y funcionalidades que ofrece la plataforma. Como solución a esta problemática, se propone el desarrollo de una interfaz web para este módulo, o la extensión de la ya existente para la plataforma. De esta manera, la propuesta sería mucho más visual, mejorando la intuitividad de QuantumSolver AI en el proceso de extracción de resultados para el público general. En este mismo sentido, la representación gráfica de los conjuntos de datos o de las vistas aplicadas, también pueden ser mejoradas o ampliadas, ofreciendo al usuario diferentes modelos de representación distintos, que mejoren la experiencia científica ofrecida.

Por otro lado, también es posible añadir más contenido y funcionalidad al módulo desarrollado. Algunos modelos de inteligencia artificial cuántica que quedaron fuera del alcance de la propuesta inicial pueden ser: regresiones lineales, aprendizaje no supervisado o redes generativa antagónicas cuánticas (Schuld, Sinayskiy y Petruccione, 2016; Aïmeur, Brassard y Gambs, 2013; Lloyd y Weedbrook, 2018). Otra funcionalidad interesante, desde el punto de vista científico, es permitir la adaptación por parte del usuario de hiperparámetros, como ya se comentó en al final del Capítulo 5. Otra opción podría ser la de ofrecer la posibilidad de ajustarlos automáticamente, ya que la selección adecuada del establecimiento de estas variables fundamentales desempeña un papel crucial tanto en el QML como en el aprendizaje automático convencional. La exploración de hiperparámetros más eficaces puede requerir una considerable cantidad de tiempo, por lo que se hace necesaria la existencia de herramientas que permitan interactuar con los mismos de manera sencilla.

Otros servicios menores que todavía no ofrece la plataforma, pero que podrían resultar también útiles, son:

- Posibilidad de guardado en vistas aplicadas a conjuntos de datos y modelos entrenados, para poder replicar las condiciones de los experimentos con mayor facilidad.
- Funcionalidad de evaluación de un punto no perteneciente al conjunto de datos, permitiendo al usuario introducir las características de un elemento para que el sistema evalué su clasificación con respecto a un modelo entrenado.
- Personalización de gráficos generados, pudiendo adaptar los colores y escalas de las representaciones provistas.
- Añadir validación cruzada en el proceso de entrenamiento. Actualmente, se realiza un entrenamiento simple sobre los datos reservados para el mismo y se testean con los de prueba, como se ha descrito en el Capítulo 4. Sin embargo, la validación cruzada supone un mejor planteamiento para el aprendizaje del modelo, especialmente útil en conjuntos de datos limitados para obtener una evaluación robusta del modelo (Stone, 1978). Se trata de una técnica para evaluar y validar modelos mediante la partición iterativa de los datos en conjuntos de entrenamiento y prueba, brindando una medida más confiable del rendimiento y ayudando a detectar problemas de sobreajuste (Prechelt, 2012). Además existen excelentes tecnologías para facilitar este desarrollo como la implementación presente en la librería scikit-learn (2023).

Al margen de las anteriores líneas futuras de ampliación directa de las funcionalidades desarrolladas, existen otras maneras de mejorar la plataforma mediante objetivos más lejanos, en lo que el alcance actual del proyecto se refiere. Estas metas pueden tener un enorme interés científico, al añadir todavía más posibilidades de uso al módulo, logrando causar más tracción en el sector industrial.

En este trabajo no se ha abordado, porque salía completamente de su alcance, otra de las aplicaciones más relevantes de la inteligencia artificial cuántica: la optimización. Por ello, debe ser el siguiente destino de QuantumSolver AI. En consonancia con el Capítulo anterior, se propone el uso de tecnologías como D-Wave (D-Wave Quantum Inc., 2023) o Pennylane (Xanadu, 2023a), que son más adecuadas para el tipo de problemas que se plantea resolver. Algunas potenciales aplicaciones de optimización de la herramienta podrían ser:

- En la ingeniería, para diseñar y optimizar sistemas, como la distribución de energía, la planificación de redes de comunicación, la programación de la producción y la logística.
- En el ámbito financiero, para la gestión de carteras de inversión, la planificación de rutas y los algoritmos de operaciones bursátiles.
- En la ciencia de datos, para ajustar modelos estadísticos, optimizar parámetros en algoritmos de aprendizaje automático y encontrar estructuras ocultas en conjuntos de datos.

Otros campos como la medicina, la robótica, la planificación urbana y la logística también se podrían ver beneficiados mediante el uso de la herramienta, que sería capaz de resolver problemas complejos y tomar decisiones eficientes y efectivas.

Actualmente, QuantumSolver se encuentra muy ligada al modelo de ejecución y definición de algoritmos de Qiskit. Por medio de un ligero cambio en la estructura de clases presente en la plataforma, que modifique la manera en la que se conciben los circuitos y su modo de ejecución, podría adaptarse para conseguir la utilización de otros lenguajes de programación, que se alternarían en función del tipo de problema que se desee resolver.

Sin duda, todas las posibilidades son realmente apasionantes. QuantumSolver AI propone un gran número de retos asequibles, logrando una herramienta funcional completa de aprendizaje supervisado cuántico que presenta interesantes líneas futuras para el constante desarrollo abierto de la plataforma.

Referencias

Aïmeur, E., Brassard, G. y Gambs, S. (2013). Quantum speed-up for unsupervised learning. *Machine Learning*, *90*, 261–287.

Amari, S. (1972). Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements. *IEEE Transactions on Computers*, *C-21*, 1197-1206.

Amazon. (2018). *Amazon Braket.* Vigmostad Bjørke. Recuperado el 24 de julio de 2023 de `https://aws.amazon.com/braket/`

Bansal, S. (2021). *10000 most common passwords.* Recuperado el 24 de julio de 2023 de `https://www.kaggle.com/datasets/shivamb/10000-most-common-passwords`

Bargen, D. (2013). *Tikz: Diagram of a Perceptron.* Stack Exchage. Recuperado el 24 de julio de 2023 de `https://tex.stackexchange.com/questions/104334/tikz-diagram-of-a-perceptron`

Bausch, J. (2020). Recurrent quantum neural networks. *Advances in neural information processing systems*, *33*, 1368–1379.

Bell, J. S. (1964). On the Einstein Podolsky Rosen paradox. *Physics Physique Fizika*, *1*, 195–200.

Bengio, Y., Ducharme, R. y Vincent, P. (2000). A neural probabilistic language model. *Advances in Neural Information Processing Systems*, *13*.

Bennett, C. H. (1992). Quantum cryptography using any two nonorthogonal states. *Physical Review Letters*, *68*, 3121–3124.

Bennett, C. H. y Brassard, G. (1984). Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, *560*, 7–11.

Bennett, C. H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A. y Wootters, W. K. (1993). Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, *70*, 1895–1899.

Bennett, C. H. y Wiesner, S. J. (1992). Communication via one- and two-particle operators on Einstein-Podolsky-Rosen States. *Physical Review Letters*, *69*(20), 2881–2884.

Bergstra, J. y Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(2).

Bernstein, E. y Vazirani, U. (1993). Quantum complexity theory. *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing - STOC '93*.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N. y Lloyd, S. (2017).

Quantum machine learning. *Nature*, *549*(7671), 195–202.

Blé Jurado, C. (2020). *Diseño Ágil con TDD.* Carlos Blé Jurado. (ISBN: 9781445264714)

Blé Jurado, C. (2022). *Código sostenible: Cómo Programar Código Fácil de mantener.* Savvily. (ISBN: 9788409361250)

Buhmann, J. y Kuhnel, H. (1992). Unsupervised and supervised data clustering with competitive neural networks. En *Proceedings 1992, IJCNN International Joint Conference on Neural Networks* (Vol. 4, pp. 796–801).

Burkov, A. (2019). *The hundred-page machine learning book* (Vol. 1). Andriy Burkov Quebec City, QC, Canada.

Carroll, S. (2019). *Even physicists don't understand Quantum Mechanics.* The New York Times. Recuperado el 24 de julio de 2023 de `https://www.nytimes.com/2019/09/07/opinion/sunday/quantum-physics.html`

Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L. y Coles, P. J. (2022). Challenges and opportunities in quantum machine learning. *Nature Computational Science*, *2*(9), 567–576.

Chalumuri, A., Kune, R. y Bs, M. (2021). A hybrid classical-quantum approach for multi-class classification. *Quantum Information Processing*, *20*.

Chugh, G., Kumar, S. y Singh, N. (2021). Survey on Machine Learning and Deep Learning Applications in Breast Cancer Diagnosis. *Cognitive Computation*, *13*(6), 1451–1470.

Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S. y Wossnig, L. (2018). Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *474*(2209), 20170551.

Correa-Marichal, J. (2023). *Dataset Generation Tool.* Recuperado el 24 de julio de 2023 de `https://github.com/zodi4cx/password-generator/`

Curry, H. B. (1944). The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, *2*(3), 258–261.

Cátedra de Ciberseguridad Binter. (2022). *Concurso TFG/TFM/Tesis de la Cátedra de Ciberseguridad Binter - ULL.* Universidad de La Laguna. Recuperado el 24 de julio de 2023 de `https://fg.ull.es/noticias/2022/10/05/concurso-tfg-tfm-tesis-de-la-catedra-de-ciberseguridad-binter-ull/`

Cátedra Edosoft. (2022). *Concursos.* Universidad de La Laguna. Recuperado el 24 de julio de 2023 de `https://www.ull.es/catedras/catedra-edosoft-computacion`

-nube/concursos/

D-Wave Quantum Inc. (2023). *D-wave.* Recuperado el 24 de julio de 2023 de https://www.dwavesys.com/

Daemen, J. y Rijmen, V. (1999). The Rijndael block cipher: AES proposal. En *First candidate conference (AeS1)* (pp. 343–348).

DeBenedictis, E. P. (2018). A Future with Quantum Machine Learning. *Computer*, *51*(2), 68-71.

Deutsch, D. y Jozsa, R. (1992). Rapid solution of problems by Quantum Computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, *439*(1907), 553–558.

Deville, Y. y Deville, A. (2021). New single-preparation methods for unsupervised quantum machine learning problems. *IEEE Transactions on Quantum Engineering*, *2*, 1–24.

Dunjko, V., Taylor, J. M. y Briegel, H. J. (2016). Quantum-enhanced machine learning. *Physical Review Letters*, *117*(13).

Duong, T. Q., Ansere, J. A., Narottama, B., Sharma, V., Dobre, O. A. y Shin, H. (2022). Quantum-Inspired Machine Learning for 6G: Fundamentals, Security, Resource Allocations, Challenges, and Future Research Directions. *IEEE Open Journal of Vehicular Technology*, *3*, 375–387.

Einstein, A., Podolsky, B. y Rosen, N. (1935). Can quantum-mechanical description of physical reality be considered complete? *Physical review*, *47*(10), 777.

Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. *Physical Review Letters*, *67*, 661–663.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. En G. R. Blakley y D. Chaum (Eds.), *Advances in Cryptology* (pp. 10–18). Berlin, Heidelberg: Springer Berlin Heidelberg.

Escanez-Exposito, D. (2022a). *GitHub QuantumSolver repository.* Recuperado el 24 de julio de 2023 de https://github.com/jdanielescanez/quantum-solver

Escanez-Exposito, D. (2022b). *QExecute.* Recuperado el 24 de julio de 2023 de https://github.com/jdanielescanez/quantum-solver/blob/main/src/execution/qexecute.py

Escanez-Exposito, D. (2022c). *QuantumSolver Documentation.* Recuperado el 24 de julio de 2023 de https://jdanielescanez.github.io/quantum-solver/

Escanez-Exposito, D. (2022d). *QuantumSolver: Librería para el Desarrollo Cuántico.*

Universidad de La Laguna. Recuperado el 24 de julio de 2023 de `http://riull.ull.es/xmlui/handle/915/28728`

Escanez-Exposito, D. y Caballero-Gil, P. (2023). Theoretical Analysis and Software Implementation of a Quantum Encryption Proposal. En *2023 19th International Conference on the Design of Reliable Communication Networks (DRCN)* (pp. 1–7).

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022a). Evolución de la librería QuantumSolver para el desarrollo cuántico. XVII Reunión Española sobre Criptología y Seguridad de la Información. RECSI 2022. *Editorial de la Universidad de Cantabria*, 94–99.

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022b). *Qiskit Quantum Hardware Testing via Implementations of QKD Algorithms.* The International Association for Cryptologic Research (IACR). Recuperado el 24 de julio de 2023 de `https://ches.iacr.org/2022/acceptedposters.php`

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022c). QuantumSolver: A quantum tool-set for developers. *The 2022 World Congress in Computer Science, Computer Engineering, and Applied Computing (CSCE 2022).*

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022d). QuantumSolver: Librería para el desarrollo cuántico. *Actas de las VII Jornadas Nacionales de Investigación en Ciberseguridad*, 107-110.

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022e). Study and implementation of an interactive simulation of quantum key distribution using the E91 cryptographic protocol. *Proceedings of the International Conference on Ubiquitous Computing & Ambient Intelligence*, 965–970.

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2023). Interactive simulation of quantum key distribution protocols and application in Wi-Fi networks. *Wireless Networks*.

Escanez-Exposito, D., Hernandez-Martin, A. y Caballero-Gil, P. (2023). Implementación del protocolo criptográfico Six-State. *Actas de las VIII Jornadas Nacionales de Investigación en Ciberseguridad*, 415-419.

Escanez-Exposito, D., Marchan-Sekulic, V. y Caballero-Gil, P. (2023). Implementación de los Algoritmos Cuánticos de Simon y de Shor. *Actas de las VIII Jornadas Nacionales de Investigación en Ciberseguridad*, 409-414.

Feynman, R. P. (1982). Simulating physics with computers. *International Journal of*

Theoretical Physics, 133–153.

Fuoli, D., Huang, Z., Paudel, D. P., Van Gool, L. y Timofte, R. (2023). An Efficient Recurrent Adversarial Framework for Unsupervised Real-Time Video Enhancement. *International Journal of Computer Vision*, *131*(4), 1042–1059.

Gamma, E., Helm, R., Johnson, R. y Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH.

Geman, S., Bienenstock, E. y Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, *4*(1), 1–58.

GitHub. (2023). *Creating a pull request.* Recuperado el 24 de julio de 2023 de `https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request`

Google. (2023). *CIRQ.* Recuperado el 24 de julio de 2023 de `https://quantumai.google/cirq`

Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search. En *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (p. 212–219). New York, NY, USA: Association for Computing Machinery.

Haugeland, J. (1989). *Artificial intelligence: The very idea*. MIT press.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M. y Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, *567*(7747), 209–212.

Heisenberg, W. (1927). Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, *43*(3), 172–198.

Holmes, Z., Sharma, K., Cerezo, M. y Coles, P. J. (2022). Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, *3*(1), 010313.

Huang, H.-Y., Kueng, R. y Preskill, J. (2020). Predicting many properties of a quantum system from very few measurements. *Nature Physics*, *16*(10), 1050–1057.

IBM. (2015). *Qiskit Runtime.* IBM Quantum. Recuperado el 24 de julio de 2023 de `https://www.ibm.com/quantum/qiskit-runtime`

IBM. (2018). *Quantum computing is coming to your business.* Recuperado el 24 de julio de 2023 de `https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/quantumstrategy`

IBM. (2022a). *Amplitude encoding.* Recuperado el 24 de julio de 2023 de `https://`

learn.qiskit.org/course/machine-learning/data-encoding#data-6-0

IBM. (2022b). *Angle encoding.* Recuperado el 24 de julio de 2023 de https://learn.qiskit.org/course/machine-learning/data-encoding#data-9-0

IBM. (2022c). *Arbitrary encoding.* Recuperado el 24 de julio de 2023 de https://learn.qiskit.org/course/machine-learning/data-encoding#data-12-0

IBM. (2022d). *Base encoding.* Recuperado el 24 de julio de 2023 de https://learn.qiskit.org/course/machine-learning/data-encoding#data-3-0

IBM. (2022e). *Data encoding - Introduction.* Recuperado el 24 de julio de 2023 de https://learn.qiskit.org/course/machine-learning/data-encoding#data-1-0

IBM. (2022f). *Variational classification.* Recuperado el 24 de julio de 2023 de https://learn.qiskit.org/course/machine-learning/variational-classification

IBM. (2023a). *Aer Simulator.* Recuperado el 24 de julio de 2023 de https://qiskit.org/documentation/tutorials/simulators/1_aer_provider.html#The-Aer-Simulator

IBM. (2023b). *IBM Quantum.* Recuperado el 24 de julio de 2023 de https://quantum-computing.ibm.com/

IBM. (2023c). *Qiskit.* Recuperado el 24 de julio de 2023 de https://qiskit.org/

IBM. (2023d). *Training a quantum model on a real dataset.* Qiskit. Recuperado el 24 de julio de 2023 de https://qiskit.org/ecosystem/machine-learning/tutorials/02a_training_a_quantum_model_on_a_real_dataset.html

IBM. (2023e). *TwoLocal.* Recuperado el 24 de julio de 2023 de https://qiskit.org/documentation/stubs/qiskit.circuit.library.TwoLocal.html

IBM. (2023f). *User account.* Recuperado el 24 de julio de 2023 de https://quantum-computing.ibm.com/composer/docs/iqx/manage/account/#account-overview

IBM-IBV. (2021). *The Quantum Decade: A Playbook for Achieving Awareness, Readiness, and Advantage*. IBM Institute for Business Value. Recuperado el 24 de julio de 2023 de https://books.google.es/books?id=MeN_zgEACAAJ

Kaelbling, L. P., Littman, M. L. y Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237–285.

Kelleher, J. D., Namee, M. B. y D'Arcy, A. (2020). *Fundamentals of machine learning for Predictive Data Analytics: Algorithms, worked examples, and case studies.* The MIT Press.

Khan, S. U., Islam, N., Jan, Z., Haseeb, K., Shah, S. I. A. y Hanif, M. (2022). A machine learning-based approach for the segmentation and classification of malignant cells in breast cytology images using gray level co-occurrence matrix (GLCM) and support vector machine (SVM). *Neural Computing and Applications*, *34*(11), 8365–8372.

Khurana, D., Koli, A., Khatter, K. y Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744.

Kitaev, A. Y. (1995). *Quantum measurements and the Abelian Stabilizer Problem.*

Klein, U., Depping, J., Wohlfahrt, L. y Fassbender, P. (2023). Application of artificial intelligence: risk perception and trust in the work context with different impact levels and task types. *AI & Society*.

Kurzweil, R., Richter, R., Kurzweil, R. y Schneider, M. L. (1990). *The age of intelligent machines* (Vol. 580). MIT press Cambridge.

Larose, D. T. y Larose, C. D. (2014). *Discovering knowledge in data: an introduction to data mining* (Vol. 4). John Wiley & Sons.

learn Developers, S. (2023). *Sklearn.svm.SVC.* Scikit. Recuperado el 24 de julio de 2023 de `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

Leverenz, L. (2018). *LanguageManual ORC.* Recuperado el 24 de julio de 2023 de `https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC`

Li, G., Zhou, L., Yu, N., Ding, Y., Ying, M. y Xie, Y. (2020). Projection-based runtime assertions for testing and debugging quantum programs. *Proceedings of the ACM on Programming Languages*, *4*(OOPSLA), 1–29.

Lloyd, S. y Weedbrook, C. (2018). Quantum generative adversarial learning. *Physical review letters*, *121*(4), 040502.

Lunghi, T., Brask, J. B., Lim, C. C. W., Lavigne, Q., Bowles, J., Martin, A., ... Brunner, N. (2015). Self-testing quantum random number generator. *Physical review letters*, *114*(15), 150501.

Luusua, A., Ylipulli, J., Foth, M. y Aurigi, A. (2023). Urban AI: understanding the emerging role of artificial intelligence in smart cities. *AI & Society*, *38*(3), 1039–1044.

L'Ecuyer, P. (2012). *Random number generation.* Springer.

Magic Len. (2023). *Passwords - rust implementation.* Lib.rs. Recuperado el 24 de julio

de 2023 de `https://lib.rs/crates/passwords`

Marangon, D. G., Plews, A., Lucamarini, M., Dynes, J. F., Sharpe, A. W., Yuan, Z. y Shields, A. J. (2018). Long-term test of a fast and compact quantum random number generator. *Journal of Lightwave Technology*, *36*(17), 3778–3784.

McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. y Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature communications*, *9*(1), 4812.

McCulloch, W. S. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

Microsoft. (2023). *Introduction to Q# & Quantum Development Kit - Azure Quantum.* Recuperado el 24 de julio de 2023 de `https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk`

Nakaji, K. y Yamamoto, N. (2021). Expressibility of the alternating layered ansatz for quantum computation. *Quantum*, *5*, 434.

Neutelings, I. (2023). *Neural networks.* TikZ. Recuperado el 24 de julio de 2023 de `https://tikz.net/neural_networks/`

Nielsen, M. A. y Chuang, I. L. (2000). *Quantum Computation and Quantum Information.* Cambridge University Press.

Nir, Y. y Langley, A. (2018). *ChaCha20 and Poly1305 for IETF Protocols* (n.º 8439). RFC 8439. RFC Editor.

Ornstein, J. (1955). Mechanical Translation: New Challenge to Communication. *Science*, *122*(3173), 745-748.

Papert, S. A. (1966). *The summer vision project.* DSpace@MIT. Recuperado el 24 de julio de 2023 de `https://dspace.mit.edu/handle/1721.1/6125`

Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E. y Latorre, J. I. (2020). Data re-uploading for a universal quantum classifier. *Quantum*, *4*, 226.

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., ... O'Brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, *5*(1), 4213.

Pierce, B. C. (2002). *Types and programming languages.* The MIT Press.

Podoletz, L. (2023). We have to talk about emotional AI and crime. *AI & Society*, *38*(3), 1067–1082.

Poggel, B., Quetschlich, N., Burgholzer, L., Wille, R. y Lorenz, J. M. (2022). Recommen-

ding Solution Paths for Solving Optimization Problems with Quantum Computing. *arXiv preprint arXiv:2212.11127*.

Poole, D. y Mackworth, A. (2010). *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

Poole, D., Mackworth, A. y Goebel, R. (1998). *Computational Intelligence: A Logical Approach*.

Prechelt, L. (2012). Early Stopping — But When? En G. Montavon, G. B. Orr y K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade: Second Edition* (pp. 53–67). Berlin, Heidelberg: Springer Berlin Heidelberg.

Qiskit Development Team. (2023a). *EfficientSU2.* IBM. Recuperado el 24 de julio de 2023 de `https://qiskit.org/documentation/stubs/qiskit.circuit.library.EfficientSU2.html`

Qiskit Development Team. (2023b). *ExcitationPreserving.* IBM. Recuperado el 24 de julio de 2023 de `https://qiskit.org/documentation/stubs/qiskit.circuit.library.ExcitationPreserving.html`

Qiskit Development Team. (2023c). *RealAmplitudes.* IBM. Recuperado el 24 de julio de 2023 de `https://qiskit.org/documentation/stubs/qiskit.circuit.library.RealAmplitudes.html`

Qiskit Development Team. (2023d). *ZFeatureMap.* IBM. Recuperado el 24 de julio de 2023 de `https://qiskit.org/documentation/stubs/qiskit.circuit.library.ZFeatureMap.html`

Qiskit Development Team. (2023e). *ZZFeatureMap.* IBM. Recuperado el 24 de julio de 2023 de `https://qiskit.org/documentation/stubs/qiskit.circuit.library.ZZFeatureMap.html`

Reilly, D. J. (2015). Engineering the quantum-classical interface of solid-state qubits. *npj Quantum Information*, *1*(1), 1–10.

Rigetti. (2019). *Welcome to the Docs for the Forest SDK!* Recuperado el 24 de julio de 2023 de `https://pyquil-docs.rigetti.com/en/v2.7.2/`

Rivest, R. L., Shamir, A. y Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126.

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

Rosenblatt, F. (1961). *Principles of neurodynamics: perceptrons and the theory of brain*

mechanisms (Inf. Téc.). New York: Cornell Aeronautical Lab Inc Buffalo NY.

Russell, S. J. y Norvig, P. (2009). *Artificial Intelligence: a modern approach* (3.ª ed.). Pearson.

Sack, S. H., Medina, R. A., Michailidis, A. A., Kueng, R. y Serbyn, M. (2022). Avoiding barren plateaus using classical shadows. *PRX Quantum*, *3*(2), 020365.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210-229.

Schrödinger, E. (1935). Die gegenwärtige Situation in der Quantenmechanik. *Naturwissenschaften*, *23*(50), 844–849.

Schuld, M., Sinayskiy, I. y Petruccione, F. (2016). Prediction by linear regression on a quantum computer. *Physical Review A*, *94*(2), 022342.

Schuld, M., Sweke, R. y Meyer, J. J. (2021). Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, *103*, 032430.

Scikit-learn developers. (2023). *Cross-validation: Evaluating estimator performance.* Recuperado el 24 de julio de 2023 de `https://scikit-learn.org/stable/modules/cross_validation.html`

Sebastianelli, A., Zaidenberg, D. A., Spiller, D., Saux, B. L. y Ullo, S. L. (2022). On Circuit-Based Hybrid Quantum Neural Networks for Remote Sensing Imagery Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *15*, 565-580.

Serrano, M. A., Perez-Castillo, R. y Piattini, M. (2022). *Quantum Software Engineering.* Springer Nature.

Shafranovich, Y. (2005). *Common Format and MIME Type for Comma-Separated Values (CSV) Files* (n.º 4180). RFC 4180. RFC Editor.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423.

Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell System Technical Journal*, *28*(4), 656-715.

Shor, P. (1994). Algorithms for quantum computation: discrete logarithms and factoring. En *Proceedings 35th Annual Symposium on Foundations of Computer Science* (p. 124-134).

Simon, D. R. (1997). On the Power of Quantum Computation. *SIAM Journal on Com-*

puting, *26*(5), 1474-1483.

Singh, C. (2007). Helping students learn quantum mechanics for quantum computing. *AIP Conference Proceedings*.

Singh, D. y Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, *97*, 105524.

Stone, M. (1978). Cross-validation: A review. *Statistics: A Journal of Theoretical and Applied Statistics*, *9*(1), 127–139.

Tudorache, A.-G., Manta, V.-I. y Caraiman, S. (2021). Implementation of the Bernstein-Vazirani Quantum Algorithm Using the Qiskit Framework. *Bulletin of the Polytechnic Institute of Iași. Electrical Engineering, Power Engineering, Electronics Section*, *67*(2), 31–40.

UCI Machine Learning. (2016a). *Breast cancer wisconsin (diagnostic) data set.* Kaggle Inc. Recuperado el 24 de julio de 2023 de `https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data`

UCI Machine Learning. (2016b). *Iris species.* Kaggle Inc. Recuperado el 24 de julio de 2023 de `https://www.kaggle.com/datasets/uciml/iris`

Wang, S., Fontana, E., Cerezo, M., Sharma, K., Sone, A., Cincio, L. y Coles, P. J. (2021). Noise-induced barren plateaus in variational quantum algorithms. *Nature communications*, *12*(1), 6961.

Warke, A., Behera, B. K. y Panigrahi, P. K. (2020). Experimental realization of three quantum key distribution protocols. *Quantum Information Processing*, *19*(11), 407.

Wiesner, S. (1983). Conjugate Coding. *SIGACT News*, *15*(1), 78–88.

Winston, P. H. (1992). *Artificial intelligence.* Addison-Wesley Longman Publishing Co., Inc.

Wittek, P. (2014). *Quantum machine learning: what quantum computing means to data mining.* Academic Press.

Xanadu. (2023a). *Pennylane.* Recuperado el 24 de julio de 2023 de `https://pennylane.ai/`

Xanadu. (2023b). *Variational circuits.* Recuperado el 24 de julio de 2023 de `https://pennylane.ai/qml/glossary/variational_circuit`

A. Publicaciones durante la realización de QuantumSolver AI

A.1. Classical vs. Quantum Machine Learning for Breast Cancer Detection

El cáncer de mama es una de las principales causas de mortalidad en las mujeres de todo el mundo, y su detección precoz es fundamental para el éxito del tratamiento. La precisión del diagnóstico del cáncer de mama ha mejorado gracias al aprendizaje automático. Esta investigación compara con gran detalle la eficacia de los sistemas de aprendizaje automático convencional y cuántico para detectar el cáncer de mama. Utilizando un conjunto de datos de acceso público, el proyecto examinará varios modelos de aprendizaje automático cuántico y los comparará con algoritmos de aprendizaje automático clásicos. Los resultados de este estudio podrían aportar información sobre las posibles ventajas del aprendizaje automático cuántico para la detección del cáncer de mama y, en definitiva, contribuir a mejorar la precisión del diagnóstico de esta enfermedad. El artículo fue publicado y se puede consultar en la siguiente referencia:

D. Escanez-Exposito and S. Díaz-Santos, "Classical vs. Quantum Machine Learning for Breast Cancer Detection" (2023). 19th International Conference on the Design of Reliable Communication Networks (DRCN), Vilanova i la Geltru, Spain, pp. 1-5. IEEE Xplore, ISBN: 978-1-6654-7598-3.

A.2. Password Strength Analysis through Supervised Quantum Machine Learning

Este artículo describe el uso de un módulo dedicado al aprendizaje automático cuántico, incluido en el módulo de desarrollo cuántico QuantumSolvier AI. En particular, incluye tanto un estudio en profundidad de la teoría necesaria, como una descripción de su implementación, que aquí se ha aplicado para analizar la fortaleza de las contraseñas. La implementación desarrollada permite concluir que el modelo cuántico presenta el problema de que una codificación desafortunada de la información puede llevar al fracaso del

entrenamiento. También confirma la dificultad de utilizar un número elevado de cúbits con ciertos casos de optimización, ya que en esos casos se genera una malla plana en la que no es trivial buscar una solución mejor. El artículo se encuentra, en la fecha en la que se realiza el deposito del presente trabajo, en fase de revisión.

D. Escanez-Exposito and P. Caballero-Gil (2023). "Password Strength Analysis through Supervised Quantum Machine Learning". Key Management and Key Recover Special Issue of Journal of Surveillance, Security and Safety (JSSS), `https://jsssjournal.com/`*, 2023, Online ISSN: 2694-1015.*

A.3. Development and testing of a quantum artificial intelligence module

En este trabajo se exponen detalles profundos sobre la implementación de Quantum-Solver AI. Se detalla la correlación entre las concepciones de su diseño y la conversión práctica en porciones operativas de código. Se exponen las estructuras de datos creadas y se exhiben resultados de ejecución del programa principal desarrollado. Además, se analiza exhaustivamente la utilidad y el alcance del módulo implementado. Por último, se plantean consejos y características a tener en cuenta para los conjuntos de datos candidatos a formar parte de la plataforma. En la fecha en la que se deposita el trabajo, el artículo se encuentra en fase de redacción.

Escanez-Exposito, D., Caballero-Gil, P. y Gil-Merino, R. (2024). "Development and testing of a quantum artificial intelligence module", The 39th ACM/SIGAPP Symposium On Applied Computing (SAC 2024).

B. Publicaciones no relativas al desarrollo de QuantumSolver AI

B.1. QuantumSolver: A quantum tool-set for developers

Este artículo presenta la propuesta inicial del conjunto de herramientas cuánticas de código abierto llamado QuantumSolver, basado en Qiskit, para ayudar a los desarrolladores sin conocimientos en computación cuántica. La librería desarrollada incluye un conjunto de algoritmos con diferentes características: generación de números aleatorios, algoritmo Bernstein-Vazirani y distribución cuántica de claves utilizando el protocolo BB84. Este documento describe los principales detalles sobre la implementación del conjunto de herramientas, centrándose en los retos encontrados. Finalmente, se analizan los resultados obtenidos con algunas conclusiones que comparan y discuten las funcionalidades incluidas.

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022). QuantumSolver: A quantum tool-set for developers. The 2022 World Congress in Computer Science, Computer Engineering, and Applied Computing (CSCE 2022).

B.2. Evolución de la librería QuantumSolver para el desarrollo cuántico

Se expone una primera evolución del la librería cuántica QuantumSolver, incluyendo la descripción de varios algoritmos con distintas funcionalidades, como el algoritmo de Grover, el teletransporte cuántico, el protocolo de codificación superdensa, la generación de números aleatorios, la resolución de los problemas de Deutsch-Jozsa y Bernstein-Vazirani, y el protocolo BB84 de distribución de claves cuánticas. Se exponen los principales detalles de la implementación, así como algunas conclusiones obtenidas de la investigación realizada.

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022). Evolución de la librería QuantumSolver para el desarrollo cuántico. XVII Reunión Española sobre Criptología y Seguridad de la Información. RECSI 2022. Editorial de la Universidad de Can-

tabria, 94–99.

B.3. Qiskit Quantum Hardware Testing via Implementations of QKD Algorithms

Este trabajo propone un estudio en profundidad sobre el rendimiento de muchos algoritmos cuánticos diferentes ejecutados en el hardware puesto a disposición por Qiskit de IBM. En particular, se describen los resultados de la librería cuántica QuantumSolver basada en el SDK de Qiskit, que permite operar con una amplia variedad de algoritmos cuánticos que pueden ejecutarse desde una línea de comandos o una interfaz web, sobre hardware y simuladores cuánticos reales. Se ilustran los resultados de las ejecuciones de prueba, comparando las ejecuciones en los diferentes equipos, mediante un póster visual.

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022). Qiskit Quantum Hardware Testing via Implementations of QKD Algorithms. The International Association for Cryptologic Research (IACR). Recuperado el 9 de julio de 2023 de `https://ches.iacr.org/2022/acceptedposters.php`

B.4. Study and implementation of an interactive simulation of quantum key distribution using the E91 cryptographic protocol

Este trabajo presenta una implementación del protocolo E91 para la distribución de claves cuánticas, proporcionando una estructura de clases intuitiva que representa las diferentes relaciones existentes entre las entidades colaboradoras en la comunicación simulada. Este desarrollo se incluye en un conjunto de herramientas, denominado QuantumSolver, desarrollado bajo licencia MIT de código abierto. Esta librería también incluye varios algoritmos con diferentes funcionalidades, como la generación de números aleatorios, la resolución de los problemas Deutsch-Jozsa y Bernstein-Vazirani, el algoritmo de Grover, el teletransporte cuántico, el protocolo de codificación superdensa y el protocolo de criptografía cuántica BB84. Aquí se describen los principales detalles de la implementación, así como algunas conclusiones obtenidas de la investigación realizada sobre sus funcionalidades, ilustradas en coloridos mapas de calor.

Escanez-Exposito, D., Caballero-Gil, P. y Martin-Fernandez, F. (2022). Study and implementation of an interactive simulation of quantum key distribution using the E91 cryptographic protocol. Proceedings of the International Conference on Ubiquitous Computing & Ambient Intelligence (UCAmI 2022), 965–970.

B.5. Implementación del protocolo criptográfico Six-State

Se presenta en este trabajo una breve descripción de la distribución de claves cuánticas mediante el protocolo *Six-State*, así como de los principales detalles de la implementación realizada de ese protocolo. Dicha implementación ha sido objeto de ampliación de la librería de código abierto *QuantumSolver*, cuyo objetivo es facilitar el desarrollo cuántico, permitiendo la ejecución de una colección creciente de algoritmos y protocolos criptográficos cuánticos. Además, aquí se hace especial hincapié en el análisis de los resultados obtenidos con la implementación del protocolo.

Escanez-Exposito, D., Hernandez-Martin, A. y Caballero-Gil, P. (2023). Implementación del protocolo criptográfico Six-State. Actas de las VIII Jornadas Nacionales de Investigación en Ciberseguridad, 415-419.

B.6. Implementación de los Algoritmos Cuánticos de Simon y de Shor

En este trabajo se expone la implementación realizada de los algoritmos de Simon y de Shor así como su inclusión a la librería de código abierto para el desarrollo de software cuántico *QuantumSolver*. Para ello se ha introducido un módulo llamado *QuantumSolver Subroutine*, que abre nuevas posibilidades a futuras incorporaciones de implementaciones en la librería. El principal objetivo de este trabajo es acercar la computación cuántica de una forma accesible y atractiva a todos los públicos, destacando además los avances más importantes que se están logrando en esta tecnología. Por ejemplo, *QuantumSolver* no solo permite poner el foco en cómo la computación cuántica puede resolver problemas más rápido que la computación clásica, sino también en cómo esto puede afectar a las formas actuales de proteger la información. En este sentido, entre los elementos cardinales

destacan los algoritmos cuánticos de Simon y Shor por su capacidad de resolver problemas particularmente difíciles para la computación clásica, que implican la ruptura de esquemas criptográficos de gran despliegue en las tecnologías actuales, como *RSA* o *Diffie–Hellman*.

Escanez-Exposito, D., Marchan-Sekulic, V. y Caballero-Gil, P. (2023). Implementación de los Algoritmos Cuánticos de Simon y de Shor. Actas de las VIII Jornadas Nacionales de Investigación en Ciberseguridad, 409-414.

B.7. Theoretical Analysis and Software Implementation of a Quantum Encryption Proposal

Este trabajo analiza una propuesta de sustituto cuántico para los actuales sistemas criptográficos de clave pública basados en factorización y logaritmo discreto. Se describe tanto su análisis teórico como su implementación práctica. La implementación se ha realizado desde cero utilizando Qiskit, reformulando todo el protocolo para compactarlo dentro de la librería QuantumSolver. Los resultados obtenidos son concluyentes sobre la inseguridad del esquema propuesto.

Escanez-Exposito, D. y Caballero-Gil, P. (2023). Theoretical Analysis and Software Implementation of a Quantum Encryption Proposal. En 2023 19th International Conference on the Design of Reliable Communication Networks (DRCN) (pp. 1–7).

B.8. Interactive simulation of Quantum Key Distribution protocols and application in Wi-Fi networks

La distribución de claves permite a dos partes producir y compartir una clave secreta aleatoria, que luego puede utilizarse para cifrar y descifrar mensajes con criptosistemas simétricos. Por ello, suele considerarse la primitiva criptográfica más fundamental de las comunicaciones secretas, especialmente en las redes inalámbricas. Mientras que el método tradicional se basa en la suposición sobre la fortaleza de algún problema matemático, la distribución cuántica de claves (QKD) implica componentes de mecánica cuántica y puede considerarse incondicionalmente segura. Este trabajo presenta una implementación de los dos protocolos QKD conocidos como E91 y B92, que incluye una estructura de clases

intuitiva para representar las diferentes relaciones entre las dos entidades colaboradoras en la comunicación simulada. Además, esta implementación ha permitido profundizar en el estudio y comparación de ambos algoritmos. Se describen los principales detalles del desarrollo, junto con algunas conclusiones obtenidas de la investigación realizada sobre sus funcionalidades, ilustradas en vistosos mapas de calor. Este trabajo también incluye un análisis preliminar del potencial de la aplicación de estos protocolos QKD para su aplicación híbrida en redes Wi-Fi basadas en el estándar IEEE 802.11.

Escanez-Exposito, D., Caballero-Gil, P y Martín-Fernández, F. (2023). Interactive simulation of Quantum Key Distribution protocols and application in Wi-Fi networks. Wireless Networks. Springer. (ISSN: 1572-8196).