Exploración y Evaluación de Técnicas de Reducción de Características en *Quantum Machine Learning*

Nuria Reyes-Dorta
Universidad de La Laguna
Tenerife, Spain
nuri.reyes578@gmail.com

Pino Caballero-Gil
Universidad de La Laguna
Tenerife, Spain
pcaballe@ull.edu.es

Carlos Rosa-Remedios
Universidad de La Laguna
Tenerife, Spain
crosarem@ull.edu.es

***Resumen*—En este trabajo se profundiza en la aplicación práctica de diferentes técnicas de reducción de características sobre datasets de gran tamaño y su impacto en los resultados a la hora de aplicar modelos de *Quantum Machine Learning*. Para dicho análisis se ha empleado un conjunto de datos que refleja el problema de detección de aplicaciones malware de Android. Considerando este problema específico y el uso de algoritmos cuánticos, con algunas de las técnicas analizadas se logra obtener un F1-score superior al 90 %. Estos prometedores resultados permiten seleccionar las técnicas más óptimas a la hora de trabajar con datasets con un gran volumen de características, facilitando así la posterior aplicación de algoritmos de *Quantum Machine Learning*.**

***Index Terms*—Quantum Machine Learning, Recursive Feature Elimination, PCA, MCA, Chi Square, Información Mutua, VQC, PQSVC, QSVC.**

Tipo de contribución: *Investigación en desarrollo*

I. Introducción

En la actualidad, la aplicación de computación cuántica está emergiendo con fuerza en diferentes ámbitos tecnológicos, aunque aún no ha alcanzado el mismo nivel de desarrollo que el *Machine Learning* (ML) clásico. En diversos trabajos [1], [2] se trata el tema. En el artículo [3] se demuestra la base teórica de la computación cuántica, mientras que en [4], se argumenta que un sistema clásico no podría representar adecuadamente un sistema mecánico cuántico. Dentro de este innovador campo, hay una subdisciplina conocida como *Quantum Machine Learning* (QML) que se encuentra aún en una fase incipiente de publicaciones. Un ejemplo interesante de su aplicación se encuentra en el trabajo [5], en el que se usan las aproximaciones cuánticas conocidas como *Variational Quantum Circuits* (VQC) y *Quantum Support Vector Classifier* (QSVC) para afrontar el desafío de la detección de *phishing* en la red de transacciones de Ethereum. En este caso, se observa una mejora en los resultados al utilizar QML en comparación con ML tradicional.

Al considerar diferentes modelos de QML, como el de los VQC, dependiendo del volumen del conjunto de datos y de la capacidad computacional disponible, los tiempos de ejecución suelen ser bastante elevados. Para optimizar estos tiempos, se tiende a reducir el número de variables con respecto a las iniciales e incluso se suele también reducir el número de observaciones. En este escenario, disponer de técnicas que permitan realizar estas acciones sin excesiva pérdida de información es fundamental. Es ahí donde cobran especial importancia las técnicas de reducción de características.

Otros de los problemas que encontramos cuando se trabaja con grandes volúmenes de datos es que los actuales sistemas cuánticos suelen tener bastantes limitaciones en términos de la cantidad y calidad de los de cúbits. A medida que aumenta la complejidad del conjunto de datos, puede volverse bastante difícil implementar y ejecutar algoritmos de QML en hardware cuántico por culpa de esas limitaciones. Reducir la dimensionalidad del conjunto de datos puede ayudar a mitigar ese problema, al reducir la cantidad de recursos necesarios para la ejecución del algoritmo, permitiendo así que sea más factible implementar en el hardware cuántico disponible.

Por otra parte, dentro de este campo se pueden encontrar modelos como el *Pegasos Quantum Support Vector Machine*, que está pensado para trabajar con grandes volúmenes de datos. Su tiempo de ejecución suele ser un poco más elevado con respecto a cualquier modelo de ML. Esto se debe principalmente a la implementación de un kernel cuántico, que marca la diferencia clave entre ambos enfoques.

El objetivo de esta investigación es analizar el impacto que provocan las diferentes técnicas de reducción de características en los modelos QML en combinación con los diferentes algoritmos disponibles en este ámbito, evaluando tanto los tiempos de ejecución como la precisión resultante.

Para llevar a cabo este estudio se ha seleccionado un conjunto de datos que aborda el problema de detectar aplicaciones con malware en el sistema Android.

Este trabajo se estructura como sigue. En la Sección II se introducen brevemente unos conceptos previos sobre computación cuántica y QML. En la sección III se aborda trabajos relacionados para contextualizar mejor el estudio en el panorama más amplio de la investigación en QML. En la sección IV se presentan las características generales del conjunto de datos y del preprocesado llevado a cabo. En la sección V se detallan los modelos utilizados, cuyos resultados se muestran en la sección VI. En la sección VII, se discuten los resultados obtenidos. Por último, el trabajo se cierra en la sección VI con algunas conclusiones y trabajos futuros.

II. Conceptos Previos

En esta sección se explican brevemente algunas nociones sobre computación cuántica, los principios de QML y las técnicas de selección de características para el análisis.

ISBN:978-84-09-62140-8

II-A. Computación Cuántica

La idea de los ordenadores cuánticos nació en la década de los 80 [6]. La computación cuántica permite realizar cálculos aprovechando el comportamiento de las partículas cuánticas.

Mientras que la unidad mínima de información en los ordenadores clásicos es el bit, en los ordenadores cuánticos son los cúbits, que son probabilísticos. Concretamente, los cúbits pueden estar en alguno de sus estados básicos $\mid 0\rangle, \mid 1\rangle$, pero también pueden estar en un estado adicional, que se denomina estado de superposición:

$$\mid \psi\rangle = \alpha \mid 0\rangle + \beta \mid 1\rangle \tag{1}$$

donde $|\alpha|^2 + |\beta|^2 = 1$, además, $\alpha, \beta \in \mathbb{C}$ y se denominan α, β amplitudes de estado.

Los cúbits pueden ser manipulados usando puertas cuánticas, que operan sobre los diferentes posibles estados del conjunto de cúbits sobre los que se aplican. Algunas de las puertas más significativas son:

- Puerta CNOT: Dados 2 cúbits, si el primer cúbit está en estado $\mid 1\rangle$ invierte el estado del segundo cúbit.
- Puerta de Hadamard: Coloca un cúbit en estado de superposición.
- Puerta de Toffoli: Extensión a 3 cúbits de la puerta CNOT.

II-B. Quantum Machine Learning

QML es un campo en desarrollo que fusiona la física cuántica, la computación cuántica y el ML para mejorar o acelerar las tareas de computación clásica. Estas tareas incluyen optimización, modelado generativo, inferencia, clasificación y regresión en el ámbito del aprendizaje automático. Este campo se puede clasificar en dos categorías principales: ML mejorado mediante técnicas cuánticas y QML [7].

II-C. Algoritmos de Quantum Machine Learning

- *Quantum Support Vector Classifier*
  Quantum Support Vector Classifier es la versión cuántica de la técnica de *Support Vector Classsifier*. Este modelo incorpora kernel cuánticos en el proceso de clasificación, y tiene como objetivo mejorar potencialmente el rendimiento de las tareas de clasificación de ML. Los *kernels* cuánticos pueden verse como una versión mejorada de los *kernels* clásicos, que se utilizan para proyectar datos en espacios de características de mayor dimensión, lo que facilita su separación y clasificación. Esto se debe a que el kernel cuántico aprovecha la computación cuántica para calcular eficientemente medidas de similitud entre estados cuánticos, que representan los datos en el espacio de características mejoradas cuánticamente [7].

- *Pegasos Quantum Support Vector Classifier*
  Pegasos Quantum Support Vector Classifier (PQSVC) es un algoritmo de QML que acelera el cálculo de SVM mediante el uso de principios de la mecánica cuántica. Es la versión cuántica del algoritmo Pegasos. El kernel cuántico de fidelidad en *Pegasos Quantum Support Vector Machine* facilita el cálculo de productos internos en espacios de características de alta dimensión, lo que permite la separación de datos complejos y no linealmente separables [15].

- *Variational Quantum Classifier*
  La técnica de Circuito Cuántico Variacional (VQC) emplea Circuitos Cuánticos Parametrizados que incorporan un mapa de características (feature map) para codificar datos en un estado cuántico. Utiliza un optimizador clásico para ajustar los parámetros del circuito, conocidos colectivamente como 'ansatz'. Estos circuitos son análogos a las redes neuronales clásicas en cuanto actúan como aproximadores universales de funciones, utilizando una evolución unitaria para mapear el estado cuántico que contiene las características del problema al estado que contiene las etiquetas [7]. Algunos ansatz más utilizados son:
  - *EfficientSU2*: El circuito generado por EfficientSU2 está compuesto de varias capas. Cada capa consta de dos partes:
    Rotaciones de un solo qubit: Se aplican rotaciones generales sobre cada qubit individual. Estas rotaciones son parametrizadas por tres ángulos, permitiendo una rotación arbitraria en el espacio de Bloch de cada qubit. Matemáticamente, estas rotaciones pueden describirse por operadores del tipo $U(\theta, \phi, \lambda)$ que corresponden a una rotación $SU(2)$.
    Entrelazamientos: Después de las rotaciones de un solo qubit, se aplican puertas de entrelazamiento entre pares de qubits. Normalmente se usan puertas CNOT que ayudan a generar el entrelazamiento necesario entre los qubits para representar una transformación unitaria general sobre el estado completo del sistema cuántico.[12].
  - *RealAmplitudes*: El circuito es una función de onda de prueba heurística que se utiliza como Ansatz en aplicaciones químicas o circuitos de clasificación en aprendizaje automático. El circuito consta de capas alternas de *RealAmplitudes* Y rotaciones y CX Enredos. El patrón de entrelazamiento puede ser definido por el usuario o seleccionado de un conjunto predefinido. Se llama así porque los estados cuánticos preparados solo tendrán amplitudes reales, la parte compleja siempre es 0 [13].
  - *ExcitationPreserving*: el circuito conserva la relación de $\mid 00\rangle, \mid 01\rangle + \mid 11\rangle\, y \mid 11\rangle$ estados. La matriz que representa la operación es:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta/2 & -\sin\theta/2 & 0 \\ 0 & \sin\theta/2 & \cos\theta/2 & 0 \\ 0 & 0 & 0 & e^{-I\psi} \end{pmatrix}$$

    Esta función de onda de prueba consta de capas de Z rotaciones con entrelazamientos de 2 qubits. El enredo se crea usando $XX + YY$ rotaciones y, opcionalmente, una puerta de fase controlada para el modo [14].

II-D. Técnicas de selección de características

- *Recursive Feature Elimination*
  La técnica de *Recursive Feature Elimination* (RFE) busca eliminar las características redundantes y seleccionar la más relevantes del conjunto de datos completo. El objetivo es encontrar un subconjunto óptimo de

características que mejore el rendimiento del modelo. Repetidamente crea modelos y reserva la mejor o peor característica en cada iteración. Al final, las características se eliminan o conservan según su impacto en el rendimiento del modelo, hasta que se agotan todas las características [16].

- *Principal Component Analysis*
  La técnica de *Principal Component Analysis* (PCA) es un método de selección de característica que funciona de la siguiente forma. Primero, para cada componente de característica resta su valor promedio. Segundo, se calcula la matriz de covarianza, el valor de la característica y el vector de características. Luego se ordena en orden descendente según los valores de la característica. Finalmente, se determina el número de componentes principales n (número de componentes de características seleccionadas) y se seleccionan los primeros n componentes de características como características seleccionadas finales [17].

- *Multiple Correspondence Analysis*
  Multiple Correspondence Analysis (MCA) es el sistema equivalente al PCA para datos nominales. Por otro lado, el MCA es una extensión del *Correspondence Analysis* (CA) estándar, que permite analizar más de dos variables a la vez. En el MCA, se crea una matriz de indicadores donde las filas representan las instancias y las columnas representan las categorías de las variables. Para aplicar el MCA, primero se discretizan las características en intervalos o valores nominales, y luego se combinan con la clase para formar la matriz de indicadores [18].

- *Chi Square*
  La prueba *chi-square* es un conocido test de hipótesis usado para determinar si existe una relación entre dos variables categóricas. Esta prueba aplicada a la selección de características se emplea para evaluar la independencia entre la característica y el objetivo. Básicamente, evalúa la relevancia de cada una de las características. Requiere que las características numéricas se adapten a una forma discreta antes de hacer los cálculos, de modo que la suma se realice sobre cada valor de la característica. Las características más relevantes se seleccionan en función de los valores más alto obtenidos en la prueba [19].

- Información Mutua
  La *Mutual Information* (MI) es un método basado en filtros que se utiliza con frecuencia. Este método mide la importancia de la información de las características al seleccionar criterios específicos con etiquetas de clase. Se asume que las características con una fuerte correlación con la inestabilidad mejorarán el rendimiento de la clasificación [20]. El método MI se utiliza para representar el grado de correlación entre la característica 't' y la categoría 'C', donde la categoría 'C' es la variable dependiente [21].

III. Investigación en QML y Detección de Malware en Android con ML

Esta sección examina el panorama de la investigación en QML y su relevancia en diversos contextos, incluida la detección de malware en dispositivos Android utilizando técnicas de Machine Learning (ML).

El primer ejemplo que vamos a ver es la detección de Phising. En el artículo [5] la problemática abordada es la detección de cuentas de phishing en la plataforma Ethereum. El estudio presenta resultados de un marco propuesto con datos codificados mediante QRAC y los compara con otros esquemas de codificación tanto en simuladores como en hardware cuántico real. Los resultados muestran que los modelos basados en QRAC superan a los modelos basados en otros esquemas de codificación, incluso en comparación con algoritmos de ML clásico como SVM y LGBM. Además, se discuten observaciones sobre el rendimiento de diferentes técnicas de codificación y modelos de QML en diferentes configuraciones. Se sugiere la aplicación de técnicas de mitigación de errores para mejorar el rendimiento en entornos de hardware real.

El artículo [8] destaca la investigación sobre el uso de métodos de QML para analizar grandes volúmenes de datos de seguridad en redes. Se identificaron la plataforma Qiskit y el marco Tensorflow Quantum como las más prometedoras. Se desarrolló un conjunto de datos de flujo para mejorar el procesamiento QML, y se implementó un método de codificación para transformar la representación de bits en qubits. Se encontró que la detección de intrusiones basada en QML supera en precisión y rendimiento a los enfoques tradicionales de ML. Los clasificadores QSVM y QCNN alcanzaron una precisión del 98 %, aunque el QCNN se considera más prometedor debido a su capacidad para seleccionar características significativas y variar la complejidad del método. El trabajo futuro se enfoca en la optimización de algoritmos y esquemas de paralelización para el entrenamiento rápido de modelos QML.

Otro ejemplo que podemos encontrar es el artículo [9]. El estudio aborda la interpretación de modelos cuánticos de ML, destacando la diferencia fundamental en la interpretabilidad entre modelos cuánticos y clásicos. Utiliza el conjunto de datos Iris para ilustrar un enfoque de interpretación basado en la región local de indecisión, que supera las limitaciones de técnicas clásicas como LIME. Además, propone un método para aproximar esta región, lo que facilita la interpretación de modelos cuánticos en términos de sus límites de decisión. Este enfoque tiene implicaciones importantes para mejorar la transparencia y la interpretabilidad en la era de la computación cuántica y el aprendizaje automático.

Dos artículos que abordan la detección de aplicaciones maliciosas en dispositivos Android mediante técnicas de *Machine Learning*.

El artículo [10] presenta un método propuesto para detectar malware ofuscado en dispositivos Android mediante un análisis dinámico con aprendizaje profundo explicativo. Se utiliza una Red Neuronal Convolucional (CNN) desarrollada por los autores, la cual procesa trazas de llamadas al sistema extraídas de aplicaciones en ejecución para generar imágenes. Estas imágenes se utilizan para construir un conjunto de datos

para entrenar la CNN. El método de detección de malware propuesto consta de tres pasos principales: recolectar trazas de llamadas al sistema, generar imágenes a partir de estas trazas y entrenar el modelo de aprendizaje profundo. La efectividad del modelo se evalúa en la detección de malware ofuscado utilizando dos conjuntos de datos obtenidos de repositorios.

El artículo [11] introduce H-LIME, un método de explicación para modelos de aprendizaje automático de caja negra enfocado en aplicaciones de seguridad y programación. H-LIME supera a otros métodos de explicabilidad en términos de precisión, completitud, dispersión, estabilidad y eficiencia. Se evalúa rigurosamente utilizando criterios específicos y se demuestra que proporciona explicaciones más precisas, dispersas y completas que LIME, especialmente en la detección de malware malicioso. Además, H-LIME puede identificar ubicaciones específicas de características maliciosas y patrones, lo que lo hace versátil para explicar decisiones en modelos de aprendizaje automático con estructuras de datos jerárquicas.

IV. Dataset

Para llevar a cabo esta investigación, se seleccionó el conjunto de datos disponible en: https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653/1?file=10391994. Este conjunto de datos aborda el problema de detección de *malware* en dispositivos Android y consta de 215 atributos extraídos de 15,036 aplicaciones. De estas aplicaciones, 5,560 son de malware del proyecto DREBIN, mientras que 9,476 son aplicaciones benignas. Es importante destacar que este conjunto de datos se encuentra desbalanceado dada la disparidad en el número de aplicaciones maliciosas y benignas [22].

El proyecto DREBIN es un sistema de detección de malware diseñado específicamente para dispositivos Android. DREBIN utiliza un análisis estático amplio para extraer conjuntos de características de diferentes fuentes, los cuales son mapeados a un espacio vectorial conjunto. A continuación, se emplean técnicas de aprendizaje automático, como Máquinas de Vectores de Soporte lineales, para identificar malware. En el último paso, se identifican las características que contribuyen a la detección de una aplicación maliciosa y se presentan al usuario para explicar el proceso de detección. Estas características incluyen permisos, llamadas a Application Programming Interface (API), métodos de cifrado y descifrado, entre otros [23].

Al aplicar los métodos de selección de características *Recursive Feature Elimination*, *Chi Square* y MI sobre este dataset, se identificaron las características más relevantes mostradas en la Figura 1.

Estas características contienen la siguiente información:

- *transact* Permite que un proceso envíe mensajes a otro proceso a través de la interfaz IBinder.
- *SEND_SMS* indica si una aplicación tiene o no permiso para enviar mensajes de texto a través de SMS.
- *onServiceConnected, bindService, attachInterface, ServiceConnection*: Estas características están relacionadas con la conexión y vinculación de servicios en Android.
- *android.os.Binder* esta característica podría está relacionada con el uso de la clase Binder en Android, que se utiliza para la comunicación entre procesos.

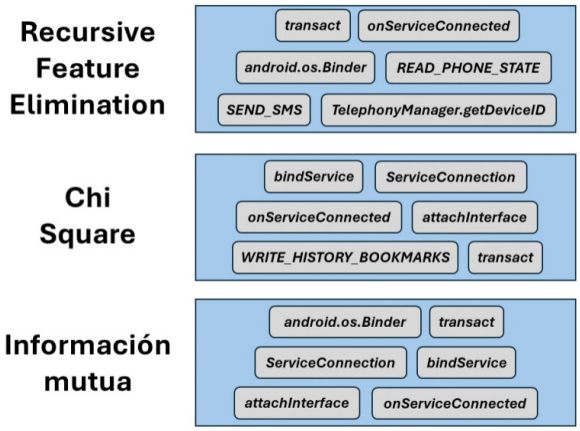

Figura 1. Criterio resultante de selección de características

- *WRITE_HISTORY_BOOKMARKS* indica si la aplicación tiene permisos para escribir marcadores en el historial del navegador.
- *READ_PHONE_STATE, TelephonyManager.getDeviceId* hacen referencia a la posibilidad de acceder al estado de la llamada y de obtener el identificador único del dispositivo o *International Mobile Equipment Identity* (IMEI).

IV-A. Preprocesado de datos

Todas las variables que conforman el dataset son variables categóricas que toman el valor 0 o 1. Además, al estudiar posibles datos duplicados se detectan 7.786 observaciones repetidas. En consecuencia se ha decidido eliminar esas observaciones duplicadas.

Dado que el objetivo es aplicar modelos cuánticos a este conjunto de datos, se necesita reducir tanto el número de observaciones como el número de variables. Para ello, se han aplicado los siguientes métodos:

- RFE
- PCA
- MCA
- *Chi Square*
- MI

Se ha decidido mantener un número de 6 variables independientes, ya que utilizar un número mayor de variables generaría tiempos de ejecución de los modelos superiores a 3 horas. Además, se han seleccionado 100 observaciones aleatorias de cada una de las clases. Finalmente hemos obtenido un conjunto de datos de 7 características contando con la variable dependiente y con 200 observaciones. Para el entrenamiento se ha decidido utilizar el 80 % de observaciones y para la prueba el 20 % de observaciones.

V. Propuesta para el Modelo

Para el conjunto de datos analizado se implementaron los siguientes modelos:

- VQC
- QSVC
- PQSVC

Para el VQC se eligió como *feature map* el ZZFeatureMap con dos repeticiones.

Como *ansatz* o circuito cuántico parametrizado se eligieron:

- RealAmplitudes (1)
- EfficientSU2 (2)
- ExcitationgPreserving (3)

Como algoritmos de optimización se eligieron:

- SLSQP
- COBYLA
- GradientDescent
- SPSA

Tanto el modelo PQSVC como el QSVC están basados en SVC pero se benefician de un núcleo cuántico. Para la generación de ese núcleo cuántico se utiliza el mapa de característica ZFeatureMap con una repetición y la fidelidad predeterminada instanciada en FidelityQuantumKernel. Además, tanto PQSVC como QSVC están diseñados para analizar grandes conjuntos de datos. Sin embargo, en este caso se ha utilizado el mismo conjunto de datos que el usado en VQC.

VI. Resultados

En esta sección se comparan los resultados obtenidos al aplicar cada una de las técnicas de reducción de características seleccionadas en combinación con las distintas parametrizaciones del modelo QML. Se han usado los parámetros: Precisión, *F1-score* y Tiempo de ejecución como referencia, salvo en el caso de los modelos de PQSVC y QSVC, donde se ha omitido el tiempo de ejecución ya que el entrenamiento se realiza en menos de 1 minuto.

Todos los modelos se implementaron en un PC con un procesador Intel i7-13700H y 16 GB de memoria RAM. Se ha utilizado como herramienta de simulación Qiskit sobre python. Qiskit es un marco de desarrollo de código abierto desarrollado por IBM para trabajar con computadoras cuánticas. Proporciona una variedad de herramientas y bibliotecas que permiten a los investigadores y desarrolladores simular y ejecutar algoritmos cuánticos en computadoras cuánticas reales o simuladas.

Tabla I
Recursive Feature Elimination (benignas)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	100,00	73,68	4245
SLSQP	(2)	90,00	81,81	15625
SLSQP	(3)	95,45	91,30	28733
COBYLA	(1)	90,00	81,82	171
COBYLA	(2)	73,91	72,34	214
COBYLA	(3)	100,00	85,71	352
GradiantDescent	(1)	100	66,67	1077
GradiantDescent	(2)	71,43	66,67	2422
GradiantDescent	(3)	100	50,00	13116
SPSA	(1)	94,74	83,72	436
SPSA	(2)	94,74	83,72	543
SPSA	(3)	94,74	83,72	889

VII. Discusión

Para evaluar el impacto de las diversas técnicas de reducción de características, se ha empleado la métrica F1-score. que combina la precisión y el *recall*, lo que proporciona una evaluación más equilibrada del rendimiento del modelo que si se evaluaran estas métricas por separado.

Tabla II
Recursive Feature Elimination (no benignas)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	61,54	76,19	4245
SLSQP	(2)	70,00	77,78	15625
SLSQP	(3)	83,33	88,24	28733
COBYLA	(1)	70,00	77,78	171
COBYLA	(2)	58,82	60,61	214
COBYLA	(3)	72,73	84,21	352
GradiantDescent	(1)	57,14	72,73	1077
GradiantDescent	(2)	52,63	57,14	2422
GradiantDescent	(3)	50,00	66,68	13116
SPSA	(1)	71,43	81,08	436
SPSA	(2)	71,43	81,08	543
SPSA	(3)	71,43	81,08	889

Tabla III
Recursive Feature Elimination (benignas)

Modelo	Precisión (%)	F1-score (%)
PQSVC	92,00	93,88
QSVC	92,00	93,88

Tabla IV
Recursive Feature Elimination (no benignas)

Modelo	Precisión (%)	F1-score (%)
PQSVC	93,33	90,32
QSVC	93,33	90,32

Tabla V
PCA (benignas)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	61,54	64,00	1293
SLSQP	(2)	68,42	60,47	6501
SLSQP	(3)	50,00	45,46	13235
COBYLA	(1)	72,73	66,67	545
COBYLA	(2)	55,00	50,00	591
COBYLA	(3)	50,00	47,83	734
GradiantDescent	(1)	60,00	54,55	1069
GradiantDescent	(2)	69,57	54,55	2525
GradiantDescent	(3)	54,55	52,17	13205
SPSA	(1)	63,16	55,81	1392
SPSA	(2)	54,17	54,17	1529
SPSA	(3)	68,42	60,47	1890

Tabla VI
PCA (no benignas)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	42,86	40,00	1293
SLSQP	(2)	47,62	54,05	6501
SLSQP	(3)	30,00	33,33	13235
COBYLA	(1)	55,56	58,82	545
COBYLA	(2)	35,00	38,89	591
COBYLA	(3)	27,78	47,83	734
GradiantDescent	(1)	40,00	44,44	1069
GradiantDescent	(2)	52,94	54,55	2525
GradiantDescent	(3)	33,33	35,29	13205
SPSA	(1)	42,86	48,65	1392
SPSA	(2)	31,25	31,25	1529
SPSA	(3)	47,62	54,05	1890

En el caso del método *Recursive Feature Elimination* en combinación con el modelo VQC, se observa que se obtie-

Tabla VII
PCA (BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	76,19	71,11
QSVC	80,00	72,73

Tabla VIII
PCA (NO BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	57,89	62,86
QSVC	60,00	66,67

Tabla IX
MCA (BENIGNAS)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	84,21	74,42	2680
SLSQP	(2)	90,45	84,44	2700
SLSQP	(3)	59,09	56,52	13648
COBYLA	(1)	68,75	55,00	553
COBYLA	(2)	75,00	75,00	596
COBYLA	(3)	60,00	46,15	740
GradiantDescent	(1)	68,42	60,47	1194
GradiantDescent	(2)	50,00	36,84	2601
GradiantDescent	(3)	46,15	32,43	13720
SPSA	(1)	71,43	66,67	1428
SPSA	(2)	85,00	77,27	1537
SPSA	(3)	70,59	56,41	1972

Tabla X
MCA (NO BENIGNAS)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	61,90	70,27	2680
SLSQP	(2)	73,68	80,00	2700
SLSQP	(3)	38,88	41,17	13648
COBYLA	(1)	45,83	55,00	553
COBYLA	(2)	62,50	62,50	596
COBYLA	(3)	40,00	48,15	740
GradiantDescent	(1)	47,62	54,05	1194
GradiantDescent	(2)	34,62	42,86	2601
GradiantDescent	(3)	33,33	41,86	13720
SPSA	(1)	52,63	57,14	1428
SPSA	(2)	65,00	72,22	1537
SPSA	(3)	47,83	56,41	1972

Tabla XI
MCA (BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	92,00	93,88
QSVC	100	93,33

Tabla XII
MCA (NO BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	93,33	90,32
QSVC	84,21	91,43

Tabla XIII
Chi Square (BENIGNAS)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	100,00	80,00	1108
SLSQP	(2)	100,00	76,92	2507
SLSQP	(3)	100,00	82,93	13578
COBYLA	(1)	94,74	83,72	178
COBYLA	(2)	100,00	76,92	222
COBYLA	(3)	78,26	76,60	365
GradiantDescent	(1)	20,00	0,07	1062
GradiantDescent	(2)	28, 57	26,67	2496
GradiantDescent	(3)	31,82	30,43	13588
SPSA	(1)	100,00	80,00	457
SPSA	(2)	100,00	76,92	570
SPSA	(3)	95,00	86,37	931

Tabla XIV
Chi Square (NO BENIGNAS)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	66,67	80,00	1108
SLSQP	(2)	64,00	78,05	2507
SLSQP	(3)	69,57	82,05	13578
COBYLA	(1)	71,43	81,08	178
COBYLA	(2)	64,00	78,05	222
COBYLA	(3)	64,71	66,67	365
GradiantDescent	(1)	34,29	47,06	1062
GradiantDescent	(2)	05,26	05,71	2496
GradiantDescent	(3)	05,56	05,88	13588
SPSA	(1)	66,67	80,00	457
SPSA	(2)	64,00	78,05	570
SPSA	(3)	75,00	83,33	931

Tabla XV
Chi Square (BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	66,67	80,00
QSVC	95,00	86,37

Tabla XVI
Chi Square (NO BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	100,00	40,00
QSVC	75,00	83,33

Tabla XVII
INFORMACIÓN MUTUA (BENIGNAS)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	100,00	80,00	1106
SLSQP	(2)	100,00	82,93	2487
SLSQP	(3)	94,74	83,72	13477
COBYLA	(1)	100,00	80,00	172
COBYLA	(2)	100,00	80,00	223
COBYLA	(3)	100,00	76,92	365
GradiantDescent	(1)	33,33	33,33	1059
GradiantDescent	(2)	30,43	29,79	2478
GradiantDescent	(3)	58,33	70,00	13573
SPSA	(1)	100,00	82,93	449
SPSA	(2)	100,00	80,00	570
SPSA	(3)	94,74	83,72	927

nen mejores resultados al utilizar los optimizadores SPSA y SLSQP. Destacan particularmente las combinaciones SLSQP con el *ansatz ExcitationgPreserving*, y COBYLA con el *ansatz ExcitationgPreserving*. Por otro lado, en los modelos PQSVC y QSVC se obtienen los mismos resultados.

Tabla XVIII
INFORMACIÓN MUTUA (NO BENIGNAS)

Optimizador	*Ansatz*	Precisión (%)	F1-score (%)	Tiempo (seg)
SLSQP	(1)	66,67	80,00	1106
SLSQP	(2)	69,57	82,05	2487
SLSQP	(3)	71,43	81,08	13477
COBYLA	(1)	66,67	80,00	172
COBYLA	(2)	76,67	80,00	223
COBYLA	(3)	64,00	78,05	365
GradiantDescent	(1)	00,00	00,00	1059
GradiantDescent	(2)	00,00	00,00	2478
GradiantDescent	(3)	25,00	10,00	13573
SPSA	(1)	69,57	82,50	449
SPSA	(2)	66,67	80,00	570
SPSA	(3)	71,43	81,08	927

Tabla XIX
INFORMACIÓN MUTUA (BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	60,00	75,00
QSVC	95,00	86,37

Tabla XX
INFORMACIÓN MUTUA (NO BENIGNAS)

Modelo	Precisión (%)	F1-score (%)
PQSVC	00,00	00,00
QSVC	75,00	83,33

En cuanto al método PCA, se observa que, en combinación con el modelo VQC, se obtienen mejores resultados al utilizar el optimizador COBYLA. Entre estas combinaciones, sobresale COBYLA con el *ansatz RealAmplitudes*. Además, en comparación con el VQC, se observan resultados más satisfactorios en los modelos PQSVC y QSVC.

Con el método de selección de características MCA, se aprecia que los mejores resultados se logran con los optimizadores SLSQP y SPSA usando el modelo VQC. Destacan las combinaciones SLSQP con el *ansatz EfficientSU2*, y SPSA con el *ansatz EfficientSU2*. Nuevamente, se observan resultados más satisfactorios en los modelos PQSVC y QSVC en comparación con el VQC.

Al emplear el método de selección de características *Chi Square*, se obtienen los mejores resultados con los optimizadores COBYLA y SPSA. Sobresalen las combinaciones COBYLA con el *ansatz RealAmplitudes*, y SPSA con el *ansatz ExcitationgPreserving*. Sin embargo, se observan resultados menos satisfactorios en los modelos PQSVC y QSVC en comparación con el VQC.

Finalmente, con el método MI de selección de características, los mejores resultados se logran con los optimizadores SLSQP y SPSA. Entre estas combinaciones, sobresalen SLSQP con el *ansatz EfficientSU2*, y SPSA con el *ansatz RealAmplitudes*. En comparación con los modelos PQSVC y QSVC, se obtienes resultados menos satisfactorios al aplicar PQSVC. Sin embargo, se obtiene mejores resultados con el modelo QSVC en comparación con el VQC.

En base a los resultados obtenidos, se puede concluir que el mejor método de selección de características es el *Recursive Feature Elimination* ya que muestra un equilibrio consistente al aplicarse en diferentes modelos de QML. Por otro lado, los resultados de los métodos PCA y MCA revelan un desempeño insatisfactorio en comparación con otros enfoques, especialmente en combinación con el VQC. No obstante, se observan mejoras significativas al aplicar los modelos PQSVC y QSVC en combinación con el método MCA, en comparación con los enfoques *Chi Square*, MI y PCA.

En lo que respecta a los tiempos de ejecución, es notable la diferencia obtenida al aplicar los optimizadores COBYLA y SPSA, que destacan por su eficiencia al no depender del uso del gradiente. No obstante, se observa que el SPSA sobresale en todos los métodos empleados para obtener los mejores resultados.

Respecto a los *ansatz*, los resultados son similares en la mayoría de los métodos, con la excepción de una notable diferencia en el tiempo de ejecución al utilizar el *ansatz ExcitationPreserving*, especialmente en combinación con el optimizador SLSQP. El rendimiento del optimizador *GradientDescent* es considerablemente inferior en comparación con otros optimizadores. En contraste, los optimizadores SLSQP y SPSA brindan los mejores resultados en todos los métodos empleados.

A pesar de que el VQC produce resultados satisfactorios, se observa que al utilizar PQSVC y QSVC se obtienen resultados similares o incluso superiores en un tiempo de ejecución menor en comparación con el VQC, exceptuando el caso en el que se emplea el método MI.

VIII. CONCLUSIONES

Esta investigación se basa en explorar las diferentes técnicas de selección de características y evaluarlas utilizando diferentes algoritmos de QML.

Tras el análisis de los diferentes métodos de selección de características empleados se concluye que los mejores resultados se obtienen utilizando *Recursive Feature Elimination* debido al equilibrio que mantiene al emplear los diferentes modelos de *Quantum Machine Learning*.

Al analizar los diferentes parámetros para VQC se concluye que los mejores optimizadores por norma general son el SLSQP y SPSA, destacando el SPSA, porque aunque no genera mejores resultados que el SLSQP, sí tiene un menor tiempo de ejecución, lo que se debe tener en cuenta en función de la prioridad que se deba dar a este parámetro.

Durante el estudio se ha detectado que el método de selección de características PCA produce unos resultados muy pobres, aunque su versión para los conjuntos de datos nominales MCA no produce resultados interesantes cuando se emplea el algoritmo VQC, produce resultados prometedores al aplicar el PQSVC y QSVC.

Otro resultado que revela el estudio realizado es que tanto el VQC como PQSVC y QSVC obtienen resultados prometedores en la mayoría de los casos. La única diferencia es que tanto el PQSVC y QSVC están pensados para analizar conjuntos de datos en menor tiempo de ejecución, lo cual queda patente en los valores obtenidos en las pruebas.

En resumen, la computación cuántica ofrece ventajas significativas en términos de procesamiento masivo paralelo, selección de características y eficiencia en la optimización de modelos en comparación con la computación clásica, lo que

la hace más útil y beneficiosa para el aprendizaje automático en ciertos escenarios y aplicaciones.

Puesto que se trata de un trabajo en desarrollo, los siguientes pasos se dirigen a ampliar el abanico de datasets utilizados, así como optimizar la metodología para trabajar con conjuntos de datos de mayor tamaño, minimizando los tiempos de ejecución.

Agradecimientos

Este trabajo ha sido posible gracias a las Cátedras de Ciberseguridad de la Universidad de La Laguna patrocinadas por Binter, y por INCIBE en el marco de los fondos del Plan de Recuperación, Transformación y Resiliencia, financiada por la Unión Europea (Next Generation). Además forma parte del proyecto PID2022-138933OB-I00 financiado por MCIN/AEI/ 10.13039/501100011033/FEDER, UE.

Referencias

[1] Jack D. Hidary: Quantum Computing: An Applied Approach, vol. 1, Springer, 2019.

[2] Paramita Basak Upama, Md Jobair Hossain Faruk, Mohammad Nazim, Mohammad Masum, Hossain Shahriar, Gias Uddin, Shabir Barzanjeh, Sheikh Iqbal Ahamed, Akond Rahman: Evolution of Quantum Computing: A Systematic Survey on the Use of Quantum Computing Tools, IEEE Annual Computers, Software and Applications Conference, 2022.

[3] Paul Benioff: The computer as a physical system: A microscopic quantum mechanical hamiltonian model ofcomputers as represented by turing machines, Journal of Statistical Physics, vol. 22, pp 563–591, 1980.

[4] Richard P. Feynman: Simulating physics with computers, International Journal of Theoretical Physics, vol. 21, pp. 467-488, 1982.

[5] Anupama Ray, Sai Sakunthala, Anil Prabhakar: Improving phishing detection in Ethereum transaction network using Quantum Machine Learning, IEEE International Conference on Quantum Computing and Engineering, 2023.

[6] Michael Nofer, Kevin Bauer, Oliver Hinz, Wil van der Aalst, Christof Weinhardt: Quantum Computing, Bussines & Information Systems Engineering, vol. 65, pp. 361-367, 2023.

[7] Hacene Belhadef, Hala Benchiheb, Lynda Lebdjiri: Exploring the Capabilities and Limitations of VQC and QSVC for Sentiment Analysis on Real-World and Synthetic Datasets, European Conference on Advances in Databases and Information Systems, vol. 1850, pp. 415-424, Springer, 2023.

[8] Maxim Kalinin, Vasiliy Krundyshev: Security intrusion detection using quantum machine learning techniques, Journal of Computer Virology and Hacking Techniques (2023) vol. 19, pp. 125–136.

[9] Lirandë Pira, Chris Ferrie: On the Interpretability of Quantum Neural Networks, 2023.

[10] Francesco Mercaldo, Giovanni Ciaramella, Antonella Santone, Fabio Martinelli: Obfuscated Mobile Malware Detection by Means of Dynamic Analysis and Explainable Deep Learning, ARES 2023.

[11] Jeff Mitchell, Niall McLaughlin, Jesus Martinez-del-Rincon: Generating sparse explanations for malicious Android opcode sequences using hierarchical LIME, 2024.

[12] https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.EfficientSU2.

[13] https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.RealAmplitudes.

[14] https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.circuit.library.ExcitationPreserving#excitationpreserving.

[15] R. Bhavsar, N. K. Jadav, U. Bodkhe, R. Gupta, S. Tanwar, P. N. Bokoro: Classification of Potentially Hazardous Asteroids Using Supervised Quantum Machine Learning, IEEE Access, vol.11, pp. 75829-75848, 2023.

[16] R.K.Jeyauthmigha, R.C.Suganthe: Recursive Feature Elimination and Clustering Technique for Network Anomaly Detection, International Conference on Current Trends towards Converging Technologies, pp. 1-6, 2018.

[17] X. Yu, Y. Ma, R. Jin, L. Xu, X. Duan: A Multi-Scale Feature Selection Method for Steganalytic Feature GFR, IEEE Access, vol.8, 2020.

[18] Qiusha Zhu, Lin Lin, Mei-Ling Shyu, Shu-Ching Chen: Feature Selection Using Correlation and Reliability Based Scoring Metric for Video Semantic Detection, IEEE Fourth International Conference on Semantic Computing, pp. 462-469, 2010.

[19] Sushil Kumar, V.B Singh, S.K Muttoo: Bug Report Classification by Selecting Relevant Features Using Chi Square, Information Gain and Latent Semantic Analysis, International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), pp. 1-5, 2021.

[20] Hastari Utama: Sentiment Analysis in Airline Tweets Using Mutual Information for Feature Selection, International Conference on Information Technology, Information Systems and Electrical Engineering, pp. 295-300, 2019.

[21] Liang Ting, Yu Qingsong: Spam Feature Selection Based on the Improved Mutual Information Algorithm, International Conference on Multimedia Information Networking and Security, pp. 67-70, 2012.

[22] Suleiman Y. Yerima, Sakir Sezer: DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection, IEEE Transactions on Cybernetics, vol. 49, No. 2, 2019.

[23] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck: DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket, Ndss, vol. 14, pp. 23-26, 2014.