UNIVERSITÀ DEGLI STUDI DI SALERNO
Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

TESI DI LAUREA

UNA PIATTAFORMA WEB PER LA PRE-ELABORAZIONE DI DATASET NEL QUANTUM MACHINE LEARNING

Relatori:
Prof.ssa Filomena Ferrucci
Prof. Giovanni Acampora

Candidato:
Gennaro Alessio Robertazzi
Matricola: 0512103792

ANNO ACCADEMICO 2019/2020

ABSTRACT

Questa tesi introduce un'applicazione web che funge da strumento per la pre-elaborazione e la classificazione dei dati in ambienti di quantum machine learning. In particolare, lo strumento proposto utilizza la versione evolutiva del prototype selection e la feature extraction basata su PCA in modo da cercare di ridurre al minimo la dimensione dei dati e consentire una classificazione più accurata, eseguita da un algoritmo quantistico di SVM eseguito su processori appartenenti alla famiglia IBM Q.

This thesis introduces a web-based application acting as a tool for data preprocessing and classification in quantum machine learning environments. In particular, the proposed tool uses evolutionary prototype selection and PCA-based feature extraction so as to try to minimize the size of data and enable a more accurate classification performed by a quantum SVM algorithm run on processors belonging to the IBM Q family.

INDICE

INTRODUZIONE

Siamo immersi in un mondo dominato dalla tecnologia. Ogni anno i computer diventano sempre più veloci e i chip che permettono loro di funzionare si fanno sempre più piccoli e potenti. L'industria segue il suo corso per portare avanti questa tendenza di miniaturizzazione affinché il consumatore finale possa usufruire di dispositivi sempre più sottili e con maggiori prestazioni. Tuttavia, la miniaturizzazione ha un limite: infatti le tecniche di fabbricazione di chip si avvicinano al regno dell'estremamente piccolo e diventa sempre più difficile progredire. Il motivo di tale crescente difficoltà è che, se le dimensioni sono molto piccole, entrano in gioco le leggi della meccanica quantistica. Il computer quantistico promette una grande accelerata della capacità di elaborazione digitale. I suoi microscopici componenti faranno le veci dei transistor, con un vantaggio in più, e cioè che possono occupare diversi stati contemporaneamente [1], un fenomeno chiamato *sovrapposizione.* I computer quantistici prospettano una rivoluzione nel nostro modo di comunicare ovvero l'uso del *teletrasporto quantistico,* una tecnica per trasmettere informazioni quasi istantaneamente fra due dispositivi che si trovano in qualsiasi punto dello spazio. Queste e tante altre sono le applicazioni della computazione quantistica che promettono un salto tecnologico immenso che inevitabilmente impatterà sulla nostra vita.

Il Machine Learning (tradotto *apprendimento automatico*) si riferisce a un'area dell'informatica in grado di allenare modelli matematici a partire da dati storici, con l'obiettivo di effettuare operazioni di classificazione/regressione/clusterizzazione [2]. Come parte dell'intelligenza artificiale e della statistica, gli algoritmi di Machine Learning elaborano grandi quantità di informazioni al fine di svolgere compiti che sono naturalmente associati al cervello umano, come il riconoscimento di immagini, parole o identificazione di schemi. Tali algoritmi sono il motore dei moderni assistenti vocali e dei sistemi che ci suggeriscono nuovi prodotti da acquistare o video da guardare, nell'individuare e diagnosticare malattie, e tanto ancora. In breve, il Machine Learning entra in gioco quando abbiamo bisogno che i computer interpretino i dati in base

all'esperienza. Ciò di solito comporta la raccolta di enormi quantità di dati input-output, gli algoritmi di Machine Learning devono essere molto efficienti per gestire i cosiddetti big data. Nel Quantum Machine Learning, l'intersezione fra computazione quantistica e apprendimento automatico, vengono sviluppati algoritmi quantistici per poter risolvere queste problematiche utilizzando l'efficienza del quantum computing. Un algoritmo di Machine Learning, in genere, necessita di un dataset ovvero una serie di informazioni separate trattate come una singola unità da un computer. Spesso questi insieme di dati possono essere molto grandi, dato il limite attuale dei *qubits* messi a disposizione per operare su un computer quantistico, necessitiamo di dover ridurre la dimensione dei dataset: sfruttando delle tecniche di *preprocessing* otteniamo gli stessi risultati aspettati dal dataset originale. Questa tesi si focalizza sullo sviluppo di una piattaforma capace di effettuare un preprocessing di dataset tale da fornire ad un algoritmo di Quantum Machine Learning il dataset ridotto inoltre introduce la computazione quantistica ed il Machine learning.

RESEARCH QUESTION

Le tecniche di preprocessing usate per il machine learning classico possono essere usate per supportare lo sviluppo di tecniche efficienti di machine learning su computer quantistici?

CAPITOLO 1

QUANTUM COMPUTING

Per capire cos'è che rende il computer quantistico tanto più potente di quello classico, per prima cosa bisogna spiegare il funzionamento di entrambi. Nella computazione classica, viene sfruttata la notazione binaria, l'unità minima di informazione è il bit rappresentato con 1 e 0. Il computer quantistico, come quello classico, usa un'unità minima fondamentale di informazione, il *qubit*. La differenza sta nel fatto che i qubit sono entità più complicate, uno dei principi di funzionamento è la cosiddetta *sovrapposizione* una proprietà della meccanica quantistica che, applicata alla computazione, può moltiplicare in modo incredibile le capacità di calcolo delle macchine.

1.1 IL QUBIT

Nella fisica classica, quando una particella gira su stessa, non si hanno dubbi su come lo faccia: basta guardarla per sapere che, rispetto ad un dato asse ruota in senso orario o antiorario. Nel caso di una particella quantistica, la realtà è decisamente più complessa, perché questa si può trovare in due o più stati contemporaneamente, un fenomeno chiamato sovrapposizione, dove la particella si comporta come se fosse in vari stati allo stesso tempo. Per implementare un qualcosa di simile in un computer quantistico, può essere utilizzato l'elettrone considerando una sua proprietà quantistica nota come *spin*, correlata con il loro momento angolare, può assumere due valori: "up" e "down", ma che può avere allo stesso

tempo un continuo dei due valori, dato che l'elettrone è in grado di occupare una sovrapposizione dei due stati prima che si effettui la misurazione. Ossia, con un computer quantistico si supera il problema di non poter calcolare tutte le possibili combinazioni, perché lo stesso computer è in una sovrapposizione di tutti gli stati possibili. Un bit classico può essere solo in due stati: 0 o 1. Nel computer quantistico, bisogna tenere conto che i qubit possono trovarsi in una sovrapposizione di stati (o combinazione lineare) di due stati fondamentali (o stati della base computazionale) e quindi avere contemporaneamente i valori 0 e 1. Denotiamo il qubit come:

$$|0\rangle + |1\rangle$$

La precedente notazione ci dice che il sistema si trova in una sovrapposizione di due stati: si può trovare contemporaneamente con il primo bit a 0 e il secondo bit a 1 e viceversa [3]. Quindi, lo stato del primo bit condiziona quello del secondo: se misuriamo il primo bit, lo stato del secondo viene immediatamente determinato (se il primo viene misurato 0, l'altro è 1 e viceversa). Questa correlazione è definita come *entanglement* (intreccio quantistico). Un altro fenomeno che appartiene esclusivamente ai computer quantistici è la cosiddetta *interferenza quantistica:* quando due onde si trovano in uno stesso punto, interferiscono l'una con l'altra, dando luogo a un'onda più grande oppure a un annullamento reciproco. In modo simile, anche le particelle quantistiche sono in grado di interferire tra di loro, di modo che o si annullano o si sommano. In particolare, per le particelle quantistiche, ciò che si somma è l'*ampiezza di probabilità* di ottenere una data misura. L'ampiezza di probabilità elevata al quadrato dà la probabilità che qualcosa accada. Prendiamo il precedente caso di due qubit in sovrapposizione, $|0\rangle + |1\rangle$. Ciascuna delle due possibilità ha un'ampiezza associata il cui quadrato fornisce la probabilità di ottenere quel risultato quando si compie la misurazione. Una possibilità è la seguente:

$$\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$$

Il numero che moltiplica ogni stato è la sua ampiezza di probabilità. Quindi, per esempio, la probabilità di ottenere il risultato $|1\rangle$ è:

$$\left(\frac{1}{\sqrt{2}}\right)^2 = 0{,}5$$

Ovvero la probabilità di ottenere questa misura è del 50%. Dato che il quadrato dell'ampiezza dà la probabilità, le ampiezze possono essere negative. Di fatto si possono usare numeri complessi, cosa che rende possibile l'interferenza.

Un qubit sarà in uno stato generico:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Dove α e β sono numeri complessi che descrivono la probabilità di ottenere gli stati $|0\rangle$ e $|1\rangle$.

Per essere più precisi, quando misuriamo lo stato di un qubit, otteniamo $|0\rangle$ con probabilità $|\alpha|^2$ oppure $|1\rangle$ con probabilità $|\beta|^2$.

Visto che vale sempre $|\alpha|^2+|\beta|^2= 1$, possiamo riscrivere lo stato di un qubit nel modo seguente:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$$

$$\text{con } 0 \leq \theta \leq \pi,\ 0 \leq \varphi < 2$$

I numeri θ e φ individuano un punto sulla sfera tridimensionale di raggio unitario sulla s*fera di Bloch* (vedi Figura 1.1), che risulta un utile mezzo per rappresentare lo stato di un qubit singolo.

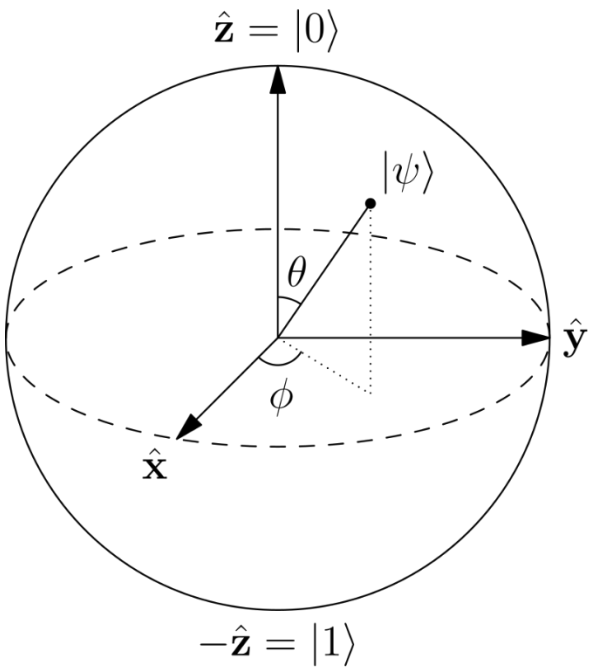


figura 1.1: Stato $|\psi\rangle$ generico di un qubit sulla sfera di Bloch.

Sebbene lo stato di un qubit possa essere un qualsiasi punto della superficie della sfera, è bene ricordare che una misura dello stato darà sempre e solo 0 o 1, facendo collassare la sovrapposizione degli stati nel risultato ottenuto. Ovvero se la misura dà risultato 0, il

qubit si troverà nello stato $|0\rangle$. Se invece la misura dà risultato 1, il qubit si troverà nello stato $|1\rangle$.

Se in generale consideriamo un sistema di n qubit, avremo 2^n ampiezze per gli stati fondamentali. Per n= 500 questo numero è più grande del numero stimato di atomi nell'universo, quindi nessun computer classico potrebbe contenere così tante informazioni. La cosa impressionante è che, sebbene sia un sistema relativamente piccolo di qubit (basti pensare a quanti pochi sono 500 bit), il numero di variabili in gioco è enorme, e la Natura effettua i suoi calcoli su tutte queste variabili mentre il sistema evolve. Uno degli obiettivi della computazione quantistica è sfruttare questa incredibile potenza di calcolo a nostro vantaggio.

1.2 PORTE QUANTISTICHE

Nei computer classici abbiamo delle porte logiche, che combinandole, è possibile effettuare qualsiasi operazione matematica. Tutti i valori che effettuiamo d'abitudine con i nostri computer, dalla stesura di un testo all'elaborazione di immagini, non sono altro che l'esecuzione di operazioni matematiche attraverso istruzioni che possono essere ridotte alla manipolazione i 1 e 0 mediante porte. Per un bit singolo, abbiamo solo due porte possibili: l'identità, che mantiene lo stato del bit, e la porta NOT, che inverte lo stato del bit (se prima era 0 diventa 1 e viceversa). Come si potrebbe realizzare una porta quantistica NOT che agisca sui qubit?

Per esempio, potrebbe scambiare lo stato $|0\rangle$ con lo stato $|1\rangle$. Però cosa succederebbe alla sovrapposizione degli stati? A causa della proprietà della meccanica quantistica, tutte le porte quantistiche devono essere lineari. Ciò significa che se avessimo lo stato iniziale $\alpha|0\rangle + \beta|1\rangle$, lo stato ottenuto dalla porta NOT sarà $\alpha|1\rangle + \beta|0\rangle$.

Un modo comodo di rappresentare le porte quantistiche è dato dalle matrici. Se per esempio definiamo la porta NOT come la matrice

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

E scriviamo lo stato quantistico $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ come il vettore colonna

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Allora l'azione della porta NOT può essere scritta come

$$|\psi'\rangle = X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Se pensiamo alle porte quantistiche su un singolo qubit come a matrici 2x2, viene spontaneo chiedersi se qualsiasi matrice 2x2 possa essere una porta quantistica. La risposta è che solo le matrici unitarie, a causa della condizione di normalizzazione, possono essere porte quantistiche le cui azioni su un qubit possono essere visualizzate come rotazioni o inversioni della sfera di Bloch. Tuttavia, non ci sono altre restrizioni. Questo significa che, al contrario del caso classico in cui c'è una sola porta a bit singolo non banale (ovvero la porta NOT), nel caso quantistico ci sono molte porte a qubit singolo non banali. Tra le porte quantistiche più usate si possono trovare le porte Hadamard, NOT, NOT controllata (o CNOT), Toffoli e Fredkin, (vedi figura 1.2).

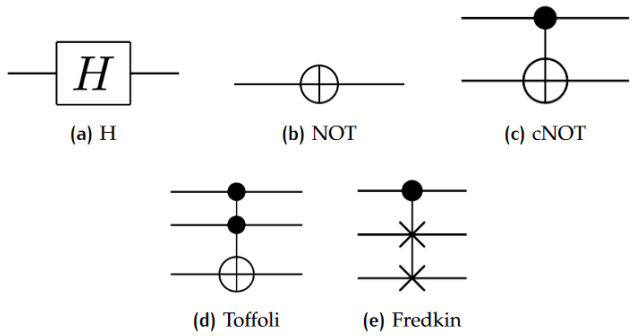


figura 1.2: Alcune porte quantistiche

Di notevole importanza sono la porta di *Hadamard* :

$$H \equiv \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

che come si vede in Figura 1.3, è rappresentata da una rotazione attorno all'asse y di 90° seguita da una rotazione attorno all'asse x di 180° e le tre matrici di Pauli

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -1 \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Le matrici X, Y e Z sono modi di girare la sfera di Bloch di π attorno agli assi x, y e z rispettivamente. La porta Z lascia invariato lo stato $|0\rangle$ e inverte il segno dello stato $|1\rangle$ mentre la porta X non è altro che la porta NOT, e trasforma $|0\rangle$ in $|1\rangle$ e viceversa. Ma per di più porta tutto quello che si trova sopra l'equatore sotto di esso.

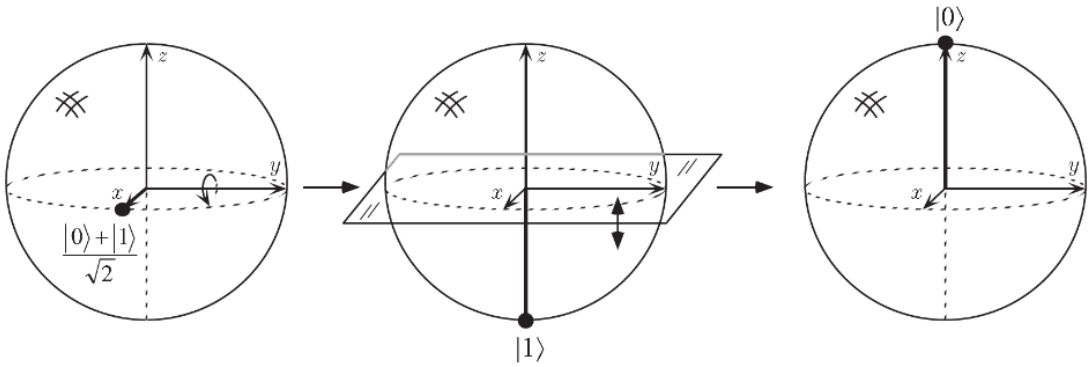


Figura 1.3 Azione della porta *Hadamard* sullo stato $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$

L'azione delle porte X, Z e H è riassunta nella Figura 1.4, dove è mostrata anche la porta NOT classica.

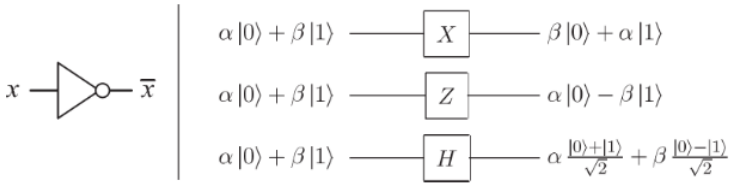


Figura 1.4 A sinistra la porta NOT per un bit singolo, a destra le porte X, Z e Hadamard per un qubit singolo.

1.3 IMPLEMENTAZIONI

1.3.1 Condizioni per la realizzazione

Un computer quantistico deve essere ben isolato per mantenere le sue proprietà quantistiche, ma allo stesso tempo i suoi qubit devono essere accessibili così che possano essere manipolati per eseguire dei calcoli e leggere i risultati. Un'implementazione realistica deve trovare il giusto equilibrio tra questi aspetti. Un concetto fondamentale per capire la validità di un particolare computer quantistico è il rumore quantistico, a volte chiamato decoerenza, che si ha quando un sistema quantistico interagisce con l'ambiente esterno, perdendo di conseguenza le proprietà quantistiche. Il *tempo di coerenza*, ovvero quanto impiega il sistema a perdere coerenza, diventando così inutile per la computazione quantistica; dall'altra il *tempo che si impiega a effettuare un'operazione.* In un computer quantistico ideale, il tempo di decoerenza sia infinito e quello di un'operazione sia pari a zero; nella pratica il primo il massimo possibile ed il secondo il minimo possibile. Le quattro condizioni fondamentali per utilizzare con successo la computazione quantistica sono:

- ***Rappresentazione digitale dell'informazione quantistica:*** L'insieme degli stati accessibili deve essere finito. Per esempio, una particella con spin 1/2 vive in uno spazio di Hilbert dato dallo span dei due stati $|\uparrow\rangle$ $|\downarrow\rangle$, e quindi è un qubit praticamente ideale quando ben isolata.

- ***Preparazione stato iniziale*****:** Oltre al tempo di coerenza, ci sono altre sfide da affrontare. Una di queste è la preparazione dello stato iniziale, che è l'equivalente dell'inserimento di dati in un computer classico. Siccome è possibile realizzare qualsiasi operazione con una appropriata combinazione di porte quantistiche, è sufficiente preparare lo stato iniziale affinché sia $|000...0\rangle$ e poi manipolarlo per ottenere lo stato desiderato. Il problema dello stato $|0000000...\rangle$ è che ha *entropia zero.* L'entropia è una grandezza correlata con l'informazione contenuta in un sistema fisico; ha a che vedere con il suo grado di disordine. Ogni sistema *isolato,* che non scambi né materia né energia con l'ambiente esterno, tende ad accrescere la propria entropia, in generale uno stato ordinato tenderà a evolvere in uno stato disordinato, fino a raggiungere il cosiddetto *stato di equilibrio termico,* che corrisponde alla distribuzione statistica più probabile dei qubit. Si ha bisogno di uno stato iniziale a entropia zero, ma per le leggi della fisica l'entropia non può che aumentare. Lo scopo è dunque, preparare lo stato iniziale in modo che la sua entropia sia minima, e mantenerlo così, questo equivale a mantenere il sistema alla temperatura più bassa possibile; il computer quantistico non potrà operare in condizioni ambientali.

- **Esecuzione di trasformazioni unitarie:** Una volta preparato lo stato iniziale, è necessario poterlo modificare. Come questo sarà effettuato, dipende dal tipo di computer se, per esempio, si usano fotoni come qubit, le porte logiche saranno una combinazione di specchi e cristalli di diverso tipo; se, in alternativa, si usano nuclei atomici, saranno necessari laser con diverse lunghezze d'onda per cambiare il loro livello energetico. La sfida, qui, consiste nel realizzare la modifica desiderata senza distruggere la coerenza, cosa difficile giacché, per definizione, si sta perturbando il sistema.

- **Misurazione degli stati di uscita:** Infine, bisogna poter leggere il risultato della computazione. Come si è detto in precedenza, si tratta di un procedimento

complesso, perché l'operazione di misura distrugge lo stato e, se non è stato preparato in modo accorto, si perderanno i risultati del calcolo. A questo punto, si può decidere di realizzare misurazioni in un insieme di sistemi equivalenti o utilizzare tecniche come la trasformata di Fourier quantistica per preparare lo stato, di modo che sia sufficiente una sola misura.

1.3.2 Divincenzo's Criteria

I criteri di Divincenzo sono condizioni [4] necessarie per costruire un computer quantistico, sette condizioni proposte nel 2000 dal fisico teorico David P. Divincenzo, cinque appartengono alla computazione quantistica, gli ultimi due alla comunicazione quantistica. In sintesi, un possibile computer quantistico di successo deve riunire una serie di caratteristiche:

1. Il sistema deve essere *scalabile*, con qubit ben caratterizzati;
2. Deve essere possibile preparare uno *stato iniziale generico*, ad esempio $|0000..\rangle$. Diversamente, sarà impossibile introdurre dati nel computer;
3. I *tempi di decoerenza* devono essere abbastanza lunghi, per poter realizzare un numero sufficiente di operazioni sfruttando la correlazione quantistica;
4. Occorre un *insieme universale di porte quantistiche*, ovverosia si deve poter costruire una varietà sufficiente di porte quantistiche per permettere qualsiasi operazione logica;
5. Si deve disporre di un modo per *misurare lo stato dei qubit*, senza il quale sarebbe impossibile estrarre l'informazione processata dal computer;

Per la comunicazione quantistica abbiamo:

6. Deve esserci un sistema per *convertire i qubit* immagazzinati in qubit messaggeri, ovverosia deve esistere un sistema per trasmettere informazioni;
7. La capacità di *trasmettere fedelmente qubit* tra le varie locazioni specificate, per la medesima ragione del punto precedente.

CAPITOLO 2

MACHINE LEARNING

I computer oggi possono imparare dall'esperienza e quindi migliorare automaticamente l'efficienza dei propri programmi durante l'esecuzione. Un semplice ma efficace strumento è il machine learning (trad. apprendimento automatico) sottoinsieme dell'intelligenza artificiale, si occupa di creare sistemi che apprendono o migliorano le performance in base ai dati che utilizzano. Si può dividere la materia in tre tipi principali:

- Apprendimento supervisionato: generalmente inizia con una serie stabilita di dati e una certa comprensione di come i dati siano classificati;

- Apprendimento non supervisionato: utilizzato quando il problema richiede una quantità enorme di dati non etichettati. Ad esempio, le applicazioni di social media, come Twitter, Instagram e Snapchat, hanno tutti grandi quantità di dati non etichettati. Capire il significato dietro a questi dati richiede algoritmi che classificano i dati in base ai modelli o ai cluster rilevati;

- Apprendimento per rinforzo: L'apprendimento per rinforzo è un modello di apprendimento comportamentale. L'algoritmo riceve un riscontro dall'analisi dei dati, guidando l'utente al miglior risultato. L'apprendimento per rinforzo differisce dagli altri tipi di apprendimento supervisionato, perché il sistema non è addestrato con il dataset di esempio. Piuttosto, il sistema impara attraverso la prova e l'errore. Quindi, una sequenza di decisioni di successo determinerà il rafforzamento del processo, perché riesce subito a risolvere il problema al meglio.

- Deep Learning: Esso è un metodo specifico di machine learning che incorpora reti neurali in strati successivi per imparare dai dati in modo iterativo. Il deep

learning è particolarmente utile quando si cerca di apprendere i modelli da dati non strutturati. Le reti neurali complesse di deep learning sono progettate per emulare il funzionamento del cervello umano, così i computer possono essere addestrati ad occuparsi di astrazioni e problemi mal definiti. reti neurali e il deep learning sono spesso utilizzati nelle applicazioni di riconoscimento delle immagini, di linguaggio e di visione artificiale.

Nel nostro caso, ci concentreremo sull' apprendimento supervisionato.

2.1 MACHINE LEARNING SUPERVISED

L'obiettivo dell'apprendimento supervisionato [5] è di imparare un modello da dati etichettati che ci permetta di fare previsioni su dati futuri sconosciuti. Con il termine supervisionato si intende l'esistenza di un insieme di campioni dove i risultati aspettati sono già conosciuti. In questo modo, quando la macchina si trova di fronte ad un problema, non dovrà fare altro che attingere alle esperienze inserite nel proprio sistema, analizzarle, e decidere quale risposta dare sulla base di esperienze già codificate. Ad esempio, è possibile creare un'applicazione di machine learning che distingua tra milioni di animali, sulla base di immagini e descrizioni scritte.

Gli algoritmi di regressione lineare e logistica, di classificazione multiclasse e le macchine a vettori di supporto sono alcuni esempi di machine learning supervisionato.

2.2 SUPPORT VECTOR MACHINE

Il Support Vector Machine (di seguito SVM) è un metodo di apprendimento supervisionato addestrato su un set di dati, esso predice se una particolare osservazione appartiene ad una determinata classe su cui è stata addestrata. In genere viene associato ad algoritmi per la classificazione, regressione e rilevamento di valori anomali. È inoltre popolare in applicazioni quali l'elaborazione del linguaggio naturale, il riconoscimento vocale e delle immagini e la computer vision.

Il SVM è basato sull'idea di trovare un iperpiano che divida al meglio un set di dati in due o più classi [6]. Sono essenziali i *vettori di supporto*, essi sono i data points più vicini all'iperpiano. Tali punti dipendono dal set di dati che si sta analizzando e se vengono rimossi o modificati alterano la posizione dell'iperpiano divisorio.

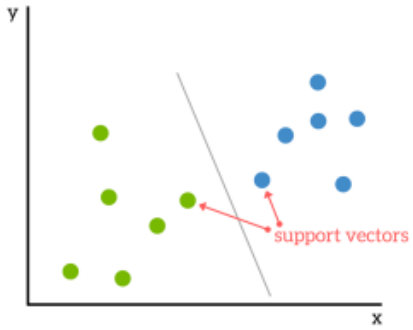


Figura 2.1: Punti di un piano cartesiano definiti come support vectors

Un' altro concetto chiave è il *margine* definito come la distanza tra i vettori di supporto di due classi differenti più vicini all'iperpiano. Alla metà di questa distanza viene tracciato l'iperpiano, o retta nel caso si stia lavorando a due dimensioni.

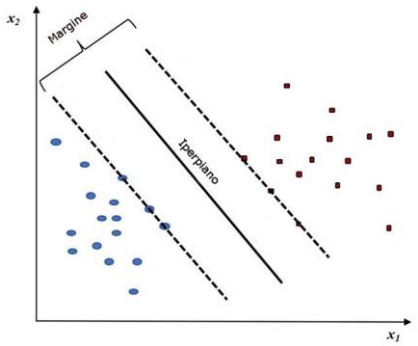


Figura 2.2: Margine o Iperpiano nel piano cartesiano

Il Support Vector Machine ha l'obiettivo di identificare l'iperpiano che meglio divide i vettori di supporto in classi. Per farlo esegue i seguenti step:

1. Cerca un *iperpiano linearmente separabile* o un limite di decisione che separa i valori di una classe dall'altro. Se ne esiste più di uno, cerca quello che ha margine più alto con i vettori di supporto, per migliorare l'accuratezza del modello.
2. Se tale iperpiano non esiste, SVM utilizza una *mappatura non lineare* per trasformare i dati di allenamento in una dimensione superiore (se siamo a due dimensioni, valuterà i dati in tre dimensioni). In questo modo, i dati di due classi possono sempre essere separati da un iperpiano, che sarà scelto per la suddivisione dei dati.

Iperpiano linearmente separabile: Nel caso dell'iperpiano linearmente separabile è possibile avere, teoricamente, un numero infinito di rette per separare i data points. Il problema è trovare quale tra le infinite rette risulti ottimale, ossia quella che generi il minimo errore di classificazione su una nuova osservazione. Infatti, più lontano dall'iperpiano si trovano i nostri punti dati, più siamo fiduciosi che essi siano stati classificati correttamente. Pertanto, desideriamo che i nostri data points siano il più lontano possibile dall'iperpiano, pur rimanendo nella parte corretta. Quindi, quando vengono aggiunti nuovi dati di test, il modello decide la classe che assegniamo ad essa. Questo è ciò che fa il Support Vector Machine nella pratica, quando trova un iperpiano linearmente separabile [7] .

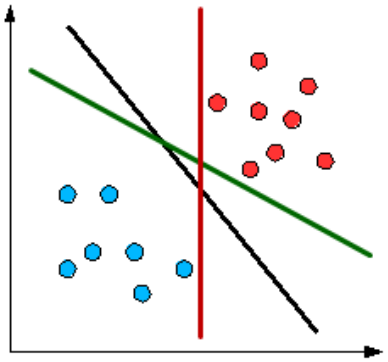


Figura 2.3: Tre rette che suddividono i data points in base al loro colore

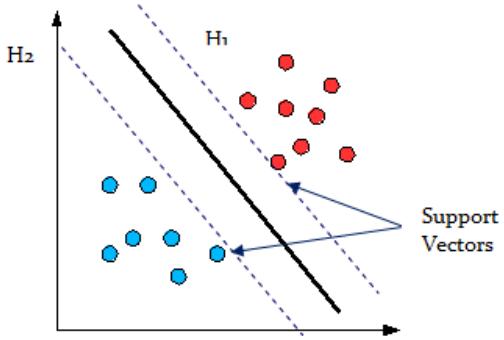


Figura 2.4: Retta ottimale per distinguere i data points dai blu ai rossi

Nel nostro esempio, secondo quanto affermato è la retta nera la linea che massimizza la distanza tra i vettori di supporto e l'iperpiano stesso.

In termini matematici, l'iperpiano ottimale può essere definito come un prodotto scalare multidimensionale in forma compatta:

$$\vec{w}\vec{x} + w_0 = 0$$

dove w è il *vettore di peso*, x è il *vettore di caratteristiche di input* e w_0 è il *bias*. Esso può essere scritto anche nel seguente modo più dettagliato:

$$w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n = 0$$

in altre parole, in n dimensioni un iperpiano di separazione è una combinazione lineare di tutte le dimensioni uguagliate a 0. Ragionando a due dimensioni per semplificare il problema avremo:

$$w_0 + w_1x_1 + w_2x_2 = 0$$

I punti che stanno sopra l'iperpiano, ovvero i data points rossi e che rappresentano una classe, soddisfano la prossima condizione:

$$w_0 + w_1x_1 + w_2x_2 > 0$$

e qualsiasi punto che si trova sotto l'iperpiano, appartiene all'altra classe, che è soddisfatta dalla successiva condizione:

$$w_0 + w_1x_1 + w_2x_2 < 0$$

Volendo includere in queste condizioni anche i limiti dei margini delle classi è possibile regolare i coefficienti o i pesi w_1 e w_2 nella seguente forma:

$$w_0 + w_1x_1 + w_2x_2 \geq 1 \text{ , per } y = +1$$

$$w_0 + w_1x_1 + w_2x_2 \leq -1 \text{ , per } y = -1$$

I due risultati sopra rappresentano i *confini del margin**e*** (sono i valori tratteggiati nell'immagine seguente e sono anch'essi due iperpiani). I dati di addestramento che ricadono esattamente sui confini del margine sono chiamati vettori di supporto.

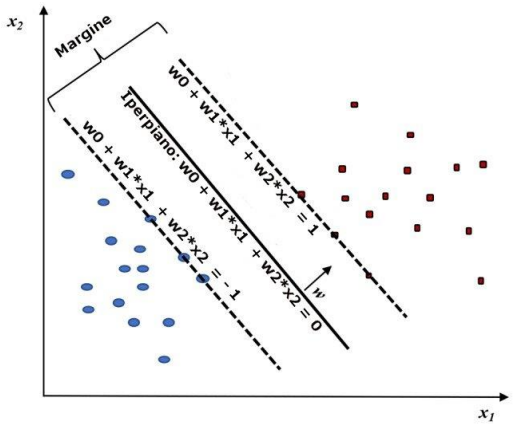


Figura 2.5: margine con vettori di supporto e iperpiano ottimale

2.2.1 METODO KERNEL

Se si ipotizza di dover classificare due classi dove non esiste un limite di decisione lineare, una singola linea retta che separa le classi (vedi figura 2.6)

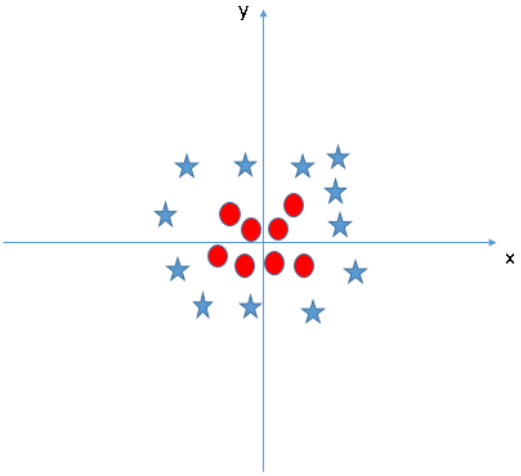


Figura 2.6: Piano cartesiano con due tipi di data points

è possibile aggiungere una terza dimensione, nel nostro caso la dimensione z stabiliamo che venga calcolata in un certo modo per noi conveniente: $z = x^2 + y^2$ (equazione di un cerchio).

La terza dimensione ci darà uno spazio tridimensionale. Una fetta di questo spazio può essere rappresentata dalla seguente figura:

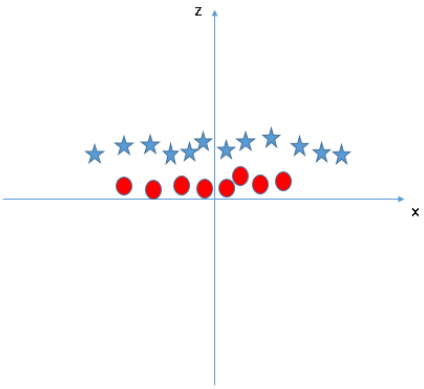


Figura 2.7: Visualizzazione sugli assi x, z

Si noti che poiché ora siamo in tre dimensioni, l'iperpiano è un piano parallelo all'asse x a una certa quota z. Quando guardiamo l'iperpiano nello spazio di input originale, esso sembra un cerchio (vedi figura 2.8).

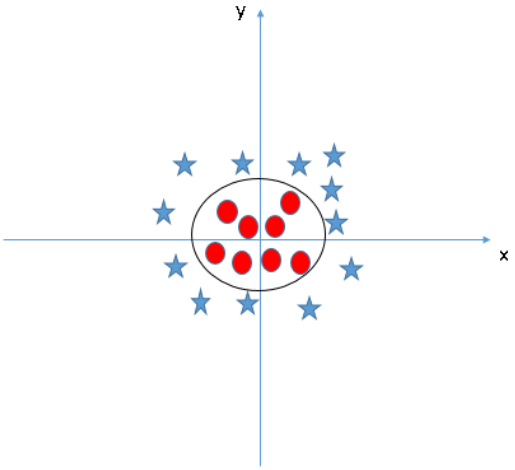


Figura 2.8

Nel nostro esempio abbiamo trovato un modo per classificare i dati non lineari mappando abilmente il nostro spazio a una dimensione tridimensionale, utilizzando quello che viene definito il *metodo Kernel.* Gli algoritmi SVM utilizzano un insieme di funzioni matematiche definite come kernel. Il suo scopo è di prendere i dati come input e trasformarli nella forma richiesta qualora non sia possibile determinare un iperpiano linearmente separabile, come avviene nella maggior parte dei casi.

Gli algoritmi SVM utilizzano un insieme di funzioni matematiche definite come kernel. Il suo scopo è di prendere i dati come input e trasformarli nella forma richiesta qualora non sia possibile determinare un iperpiano linearmente separabile, come avviene nella maggior parte dei casi.

In generale il Kernel può essere definito come:

$$K(x, y) = < f(x), f(y) >$$

dove K è la **funzione del kernel**, x, y sono vettori di input a dimensione n.

f è usato per mappare l'input dallo spazio n dimensionale a quello m dimensionale (di livello più alto del livello n). Mentre <x, y> indica il prodotto scalare.

Applicare la funzione Kernel all'esempio visto nel paragrafo precedente significa calcolare il prodotto scalare per la terza dimensione (z). Se il nuovo spazio che vogliamo è $z = x^2 + y^2$, il prodotto scalare in tale spazio risulterà come dalle seguenti espressioni:

$$x \cdot y = x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2$$

$$x \cdot y = x_1 \cdot x_2 + y_1 \cdot y_2 + (x_1^2 + y_1^2) \cdot (x_2^2 + y_2^2)$$

Normalmente, il kernel è lineare e otteniamo un classificatore lineare. Tuttavia, usando un kernel non lineare (come in questo caso) possiamo ottenere un classificatore non lineare senza trasformare completamente i dati: modifichiamo solo il **prodotto scalare** a quello dello spazio che vogliamo e SVM ci aiuterà a trovare il miglior iperpiano.

2.2.2 TIPOLOGIE DI KERNEL

Le funzioni Kernel [8] più diffuse sono le seguenti:

- **Kernel lineare**, che è definito come:

$$K(x_i, y_j) = x_i \cdot y_j$$

Questo è il tipo di kernel più semplice e che funziona bene per la classificazione del testo.

- **kernel polinomiale**, che è definito come:

$$K(x_i, y_j) = (x_i \cdot y_j + c)^d$$

Questo kernel contiene due parametri: una costante c e un grado di libertà d. Un valore d con 1 rappresenta il kernel lineare. Un valore maggiore di d renderà il limite decisionale più complesso e potrebbe comportare un *overfitting dei dati.*

- **kernel RBF,** che viene definito come:

$$K(x_i, y_j) = e^{(-\gamma \| x_i - y_j \|^2)}$$

Il kernel RBF (Radial Basis Function) è anche chiamato il kernel gaussiano. Risulterà in un limite decisionale più complesso. Il kernel RBF contiene un parametro γ: un piccolo valore di γ farà sì che il modello si comporti come un SVM lineare, mentre un grande valore di γ renderà il modello fortemente influenzato dagli esempi dei vettori di supporto.

2.2.3 VANTAGGI SVMs

I principali vantaggi di questo algoritmo derivano dal fatto che è *efficace in dimensioni spaziali elevate*, un altro punto è *l'efficienza della memoria*, poiché solo un sottoinsieme dei punti di training viene utilizzato nel processo decisionale effettivo di assegnazione di nuovi membri, solo questi punti devono essere considerati quando si prendono decisioni. Da considerare anche la *versatilità*: la separazione di classi è spesso altamente non lineare. La capacità di applicare nuovi kernel consente una sostanziale flessibilità per i limiti decisionali, portando a una maggiore performance di classificazione.

Da un altro punto di vista ci sono gli svantaggi del Support Vector Machine di fatti l'interpretazione *non è semplice*: Uno svantaggio comune delle tecniche non parametriche è la mancanza di trasparenza dei risultati. Ed infine abbiamo un *metodo non probabilistico:* Poiché il classificatore funziona posizionando gli oggetti sopra e sotto un iperpiano di classificazione, non esiste un'interpretazione probabilistica diretta per l'appartenenza al gruppo. Tuttavia, una potenziale metrica per determinare "l'efficacia" della classificazione è quanto lontano sia il confine della decisione rispetto al nuovo punto.

2.3 QUANTUM MACHINE LEARNING

Il quantum machine learning (di seguito QML) è l'intersezione tra quantum computing e intelligenza artificiale, essa si prospetta tale da cambiare l'aspetto del futuro. Il QML è un campo che mira a scrivere algoritmi quantistici per eseguire attività di Machine learning [9].

Come quando il numero delle features del SVM producono troppe dimensioni (n-features producono n-dimensioni), è difficile per i computer classici gestire calcoli così grandi. Anche se il computer classico fosse in grado di gestirlo, ci vorrebbe troppo tempo. Fortunatamente, i computer quantistici hanno la potenza di calcolo per gestire questi algoritmi gravosi. Utilizzano leggi della meccanica quantistica che trascendono dalla fisica macroscopica, come la sovrapposizione e l'entanglement per risolvere i problemi più velocemente delle loro controparti classiche. [10]

2.4 QUANTUM SUPPORT VECTOR MACHINE

In questo lavoro, mostriamo come la SVM, può essere implementata su un computer quantistico [10]. La Quantum Support Vector Machine è diversa in quanto utilizza una *feature map* per mappare i data points su un circuito quantistico.

La più grande differenza rispetto alla controparte classica risiede nell'algoritmo stesso: dato uno spazio delle caratteristiche dimensionali N e M campioni, un SVM classico, deve calcolare $\frac{M(M-1)}{2}$ prodotti scalari per costruire la matrice del kernel (ogni uno di loro richiede O $(N)^2$ operazioni), quindi richiede O (M^3) operazioni per risolvere il problema quadratico nello spazio duale. Alla fine, abbiamo $O(\log(\varepsilon^{-1}\text{poly}(N,M))$ operazioni per addestrare il nostro classico SVM.

D'altra parte, un qSVM può eseguire prodotti scalari "naturalmente" (valutando la matrice del kernel) in parallelo, terminando con un totale di O $\log_2$ (NxM).

CAPITOLO 3

DATASET PREPROCESSING FOR QUANTUM MACHINE LEARNING

Siccome ogni feature del QSVM viene associata ad un qubit ed il numero dei qubit attualmente a nostra disposizione è limitato (32 qubit), necessitiamo di un preprocessing sul dataset per poter agire efficacemente sul hardware quantistico. In questo lavoro mostriamo due tipi di approcci, Feature Extraction con PCA e Prototype Selection. Il primo consiste nel ridimensionare il dataset riducendo il numero di colonne, mentre il secondo ne riduce il numero di righe; combinando queste due tecniche siamo in grado di ottenere un dataset che può essere implementato efficacemente nel nostro algoritmo quantistico di apprendimento automatico, ottenendo vantaggi in termine di tempo computazionale e accuratezza dei risultati.

3.1 FEATURE EXTRACTION WITH PCA

Principal Component Analysis (di seguito PCA) è un metodo di Feature Extraction che rientra nei problemi di trasformazione lineare ortogonale [11]. Praticamente converte una matrice di *n* caratteristiche in un nuovo set di dati con meno di *n* caratteristiche. Cioè, riduce il numero di caratteristiche costruendo un nuovo numero più piccolo di variabili che catturano una parte significativa delle informazioni trovate nelle caratteristiche originali. Tecnicamente, la PCA trova gli autovettori di una matrice di covarianza con gli autovalori più alti e quindi li utilizza per proiettare i dati in un nuovo sottospazio di dimensioni uguali o inferiori. L'output di PCA sono proprio queste componenti principali, il cui numero è inferiore o uguale al numero di variabili originali.

Dato il solito set di dati con campioni $x_j^i \in X$, avente spazio dimensionale d per le caratteristiche, può essere rappresentato come:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & \cdots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_n^n \end{bmatrix}$$

e quindi il valore medio per ciascuna caratteristica è definito come:

$\mu \coloneqq E[x] = (< x_i > \cdots < x_k >)..,while\ \sigma_{i,j} = (\Sigma)_{i,j} \coloneqq E[(x - \mu)(x - \mu)^T]_{i,j}$

è definito come la covarianza tra due caratteristiche, ottenendo la sua migliore stima come $\sigma_{ij} = \frac{1}{n} \Sigma_{h=1}^{n} (x_i^h - \mu_i)(x_j^h - \mu_j)$. In una forma vettoriale è scritto come:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1k} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{k1} & \sigma_{k2} & \dots & \sigma_k^2 \end{bmatrix}$$

Gli autovettori della figura precedente individuano le nuove direzioni indipendenti all'interno dello spazio delle caratteristiche, mentre gli autovalori associati sono le varianze intorno quegli assi. Dal momento che più importante i vettori sono quelli con la varianza più alta, la dimensione k dello spazio delle caratteristiche è ridotta a d < k, prendendo come vettori di base quelli associati agli autovalori di grado maggiore.

3.2 PROTOTYPE SELECTION

Il Prototype Selection (di seguito PS) è una tecnica volta a ridurre la dimensione training set, senza deprecare, ma migliorare l'accuratezza della classificazione [12]. In particolare, mira a ridurre il training set originale ad un prototipo rappresentativo con dimensione inferiore ottenendo una precisione di classificazione simile o addirittura superiore. Essa è un utile tecnica poiché consente di diminuire il costo computazionale dei classificatori, migliorando le loro capacità di generalizzazione attraverso l'eliminazione del rumore.

Grazie a questa capacità, i metodi di Prototype Selection (PSM) possono fornire, nel nostro caso, un numero preciso di istanze, certamente minore di quello originale, una minore quantità di dati implica una riduzione del runtime di classificazione del QSVM, senza perdita di accuratezza, rispetto al dataset originale.

In questo lavoro, viene applicato il metodo proposto dagli algoritmi genetici, essi sono una ottima soluzione per risolvere problemi di ottimizzazione. L'aggettivo "genetico" è ispirato dal principio di selezione naturale ed evoluzione biologica teorizzato da Charles Darwin, deriva dal fatto che gli algoritmi genetici attuano meccanismi concettualmente simili a quelli dei processi biochimici scoperti da questa scienza. In

sintesi, gli algoritmi genetici consistono in algoritmi che permettono di valutare (*fitness*) diverse soluzioni di partenza (come se fossero diversi *individui* biologici) e che ricombinandole (analogamente alla riproduzione biologica sessuata) ed introducendo elementi di disordine (analogamente alle mutazioni genetiche casuali) producono nuove soluzioni (nuovi individui) che vengono valutate scegliendo le migliori (selezione ambientale) nel tentativo di convergere verso soluzioni "di ottimo". Ognuna di queste fasi di ricombinazione e selezione si può chiamare generazione come quelle degli esseri viventi.

Di seguito elenchiamo una serie di terminologie utili alla comprensione del funzionamento:

- *Cromosoma*: una delle soluzioni ad un problema considerato. Generalmente è codificata con un vettore di bit o di caratteri;
- *Popolazione*: insieme di soluzioni relative al problema considerato;
- *Gene*: parte di un cromosoma. Generalmente consiste in una o più parti del vettore di bit o caratteri che codificano il cromosoma;
- *Fitness*: grado di valutazione associato ad una soluzione. La valutazione avviene in base ad una funzione appositamente progettata detta *funzione di fitness*;
- *Crossover*: generazione di una nuova soluzione mescolando delle soluzioni esistenti.
- *Mutazione*: alterazione casuale di una soluzione.

Nel caso del nostro PS, si basa sull'idoneità di un individuo S sul valore *eval (S)* il cui pseudo-codice è mostrato nella *figura 3.1*. Quindi, la ricombinazione del materiale genetico è simulata attraverso due operatori: il crossover che scambia porzioni tra due cromosomi selezionati casualmente e una mutazione che causa alterazione casuale dei geni cromosomici. L'algoritmo termina la sua evoluzione quando vengono raggiunte le condizioni necessarie. Nel nostro algoritmo, l'evoluzione termina quando il numero massimo di valutazioni di idoneità (corrispondente al numero di soluzioni valutate) è stato raggiunto. L'output è rappresentato dal sottoinsieme di istanze della popolazione finale con i migliori valore di fitness. Una soluzione per la rappresentazione è vista come un vettore di n variabili (una per ogni istanza nel training set T R) dove ogni variabile può essere impostata su due possibili stati: 0 e 1. In dettaglio, se la variabile è impostata a 1, l'istanza di addestramento è inclusa nel sottoinsieme di T R, altrimenti, non è incluso. Tuttavia, questa tecnica porta ad avere grandi dimensioni cromosomiche. Nel nostro

caso, decidiamo di farlo considerando ogni istanza del set di dati da ridurre identificata da un indice i (con i = 0; 1; ….; *n*-1) dove *n* è il numero delle istanze del set di dati). Quindi, una soluzione viene codificata tramite un vettore di p valori interi *i,j* (con j = 0; 1;….; *p*-1) dove *i.j* è l'indice associato all'istanza. Il nostro algoritmo sfrutta una codifica formata da intera. La dimensione del cromosoma p è correlata al tasso di riduzione dato in input.

Formalmente, consideriamo T R il dataset da ridurre caratterizzato da n istanze e *r%* il tasso di riduzione; la dimensione del cromosoma p è la seguente:

$$p = n - n * \frac{r\%}{100}$$

A differenza di altri schemi di codifica di protoype selection, questa proposta permette di affrontare meglio grandi set di dati. In effetti, la dimensione del *cromosoma* non è uguale al numero di istanze del set di dati ma al numero di istanze del set di dati ridotto. Ad ogni modo, dataset con milioni di dati posso essere migliorati permettendo come ad esempio ad un classificatore quantistico di migliorare le prestazioni.

Input: the number of the instances n of the original dataset; the number of instances to be selected p; parameters of GA (size of the population pop_size, crossover rate p_c, mutation rate p_m, tournament size t_size), termination criteria (maximum number of evaluations max_evals).

Output: the $best_chromosome$ representing the reduced dataset

```
1: gen ← 0;
2: pop ← generatePopulation(pop_size);          // Generate randomly an initial population pop of pop_size chromosomes
3: evaluateFitness(pop);          // Evaluate the fitness value for each chromosome and increase the number of executed evaluations evals
4: best_chromosome ← getBestChromosome(pop);          // Select the best chromosome of the current population
5: while (evals ≤ max_evals) do
6:     offspring=executeTournament(pop, pop_size, t_size);          // Select pop_size chromosomes to compose the offspring
7:     executeSinglePointCrossover(offspring, pc);          // Recombine chromosomes of the offspring according to a crossover rate pc
8:     executeUniformIntegerMutation(offsping, pm);          // Mutate chromosomes of offspring with a mutation probability pm
9:     evaluateFitness(offspting);          // Evaluate the fitness value for the changed chromosomes and increase the number of executed evaluations evals
10:    pop ← offspring;          // The new population is replaced by the offspring
11:    best_chromosome ← getBestChromosome(pop);
12:    gen ← gen + 1;          // Increment number of iterations
13: end while
14: return best_chromosome;
```

Figura 3.1: Pseudo-codice dell'algoritmo genetico utilizzato dal nostro problema di PS.

CAPITOLO 4
UN SISTEMA DI DATASET PREPROCESSING PER QUANTUM SUPPORT VECTOR MACHINE

4.1 WEB-PLATAFORM

La piattaforma web nasce con lo scopo di aumentare l'efficacia e l'efficienza degli algoritmi quantistici di machine learning attraverso l'utilizzo di tecniche di preprocessing volti a ridurre dataset.

Attualmente gli algoritmi quantistici eseguiti sui computer quantistici, sono ancora in fase di prototipazione e le risorse fruibili alla comunità scientifica sono ancora limitate. In particolare, i qubit di un computer quantistico di IBM possono arrivare al più a 32. Tale limitazione, nel nostro caso, implica l'utilizzo di non oltre 32 features; in quanto il rapporto qubit-features è legato da una relazione 1:1. Pertanto, per venire in contro a questa esigenza, si adotta una tecnica di Feature Extraction nota come PCA per ridurre il numero di features, migliorando anche l'accuratezza del qSVM. Inoltre, per diminuire il costo temporale dell'algoritmo viene utilizzata un'altra tecnica di preprocessing nota come Prototype Selection per poter ridurre il numero delle istanze presenti nel dataset.

Combinando queste due tecniche siamo in grado di migliorare l'accuratezza e il costo temporale per eseguire il qSVM. La piattaforma viene presentata all'utente finale sottoforma di interfaccia web.

L’applicazione presenta un form per il caricamento del dataset; il dataset per poter essere caricato correttamente l’utente deve specificare se l’applicazione deve effettuare automaticamente la divisione del dataset in train e test oppure caricare il dataset suddiviso manualmente attraverso l’utilizzo di due form, il primo dedicato al train set ed il secondo il test set.

A seguito del caricamento del dataset, l’utente attraverso l’uso di checkbox, specifica rispettivamente se utilizzare la tecnica di Prototype Selection, la tecnica di Features Extraction tramite PCA oppure combinare le tecniche.

Una volta inizializzato il processo di caricamento del dataset, digitando sul pulsante di ‘upload dataset’ viene eseguito il programma principale che risiede nel back-end, effettuando il preprocessing del dataset. Successivamente per usare correttamente l’algoritmo quantistico, l’utente deve specificare attraverso la compilazione di un form il suo token fornito da IBM Q Experience, per potersi associare il processo di esecuzione sul computer quantistico di IBM. In ultimo l’utente inserisce la sua e-mail per poter ricevere il risultato della sua computazione.

4.2 BACK-END IN PYTHON

Il motore della nostra applicazione risiede all’ interno del nostro web-server (vedi 4.4.8), il codice operante al suo interno è stato sviluppato interamente in Python.

4.2.1 Il linguaggio Python

Python (di seguito Py) è un linguaggio multi-paradigma, che supporta sia la programmazione procedurale, sia la programmazione ad oggetti. Inoltre, supporta anche diversi elementi della programmazione funzionale (come iteratori e generatori). Nel nostro caso si è resa molto utile in quanto Py è senza dubbio uno dei linguaggi con il più alto livello di astrazione. Ciò significa che la sintassi è molto più semplice rispetto ad altri linguaggi, e in poche linee è possibile effettuare azioni molto complesse. In un contesto di lavoro o accademico, questo è un notevole vantaggio poiché permette di concentrarsi sugli algoritmi, tralasciando i dettagli. Un’altra particolarità di Python è che possiede una grande quantità di librerie molto sviluppate per il Machine Learning e il

Quantum Computing soprattutto qiskit (vedi 4.4.3) in quanto scritto principalmente scritto in Py.

4.2.2 SPLIT DEL DATASET

Per poter ottenere combinazioni ottimali sui dati in modo da generare un buon modello predittivo, e quindi generalizzare bene anche i nuovi dati, necessitiamo di un insieme di dati di allenamento chiamato training set. L'affidabilità di questo modello viene valutato tramite un altro insieme di dati chiamato test set.

Generalmente questi due insiemi si ricavano dal dataset di partenza, nel nostro caso l'utente può scegliere se caricare il dataset diversificato oppure che la piattaforma se ne occupi di suddividere il dataset.

Qualora l 'utente scelga di suddividere automaticamente il dataset, entra in funzione un processo di suddivisione del dataset che tramite la funzione fornita dal pacchetto *pandas*, prende come parametro il path del file, legge il nostro input, nel nostro caso un file CSV e ritorna due insiemi di file, il primo corrispondente all' 80% dell'insieme di dati ed il secondo al 20% dei dati. I due file denominati 'Data_training.csv' e 'Data_testing.csv' corrispondono al training set ed al test set rispettivamente.

```
def splitDataset(filename):
    df = pd.read_csv(filename, index_col=0)

    SEED = 1

    # 80/20 holdout split
    train, test = train_test_split(df, stratify=df['Species'], test_size=20, random_state=SEED)

    return train.to_csv('Data_training.csv'), test.to_csv('Data_testing.csv')
```

Screen del codice per effettuare lo split sul dataset in train e test set

Il processo viene richiamato tramite il codice "mainPreprocessing.py" attraverso la sua funzione dedicata "train_testSplit.py".

4.2.3 PREPROCESSING TRAMITE PROTOTYPE SELECTION

Per poter mettere a disposizione questa tecnica è stato implementato l'algoritmo di Prototype Selection (vedi 3.2)

All'interno del nostro algoritmo "mainPreprocessing.py", la variabile "prototypeSelection" quando assume valore True viene richiamata la funzione "callPS" contenuta in 'callPS.py'. Essa esegue il preprocessing sul file 'Data_training.py', restituendo un nuovo insieme di dati rappresentante il dataset contenente un numero definito di istanze.

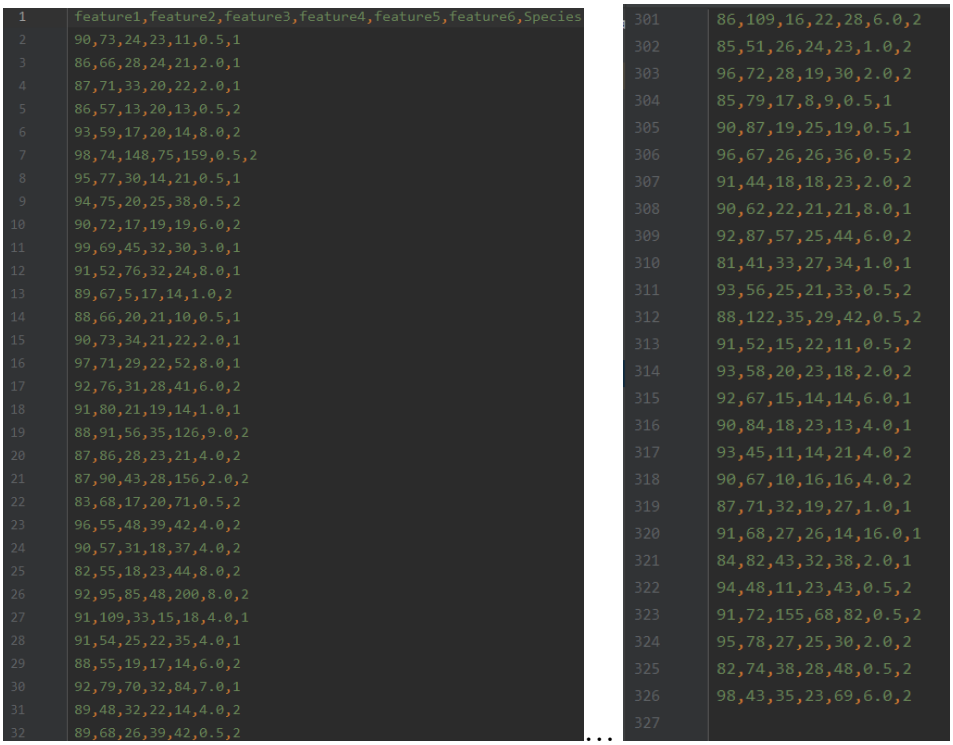
La funzione contiene una prima fase per standardizzare i dati attraverso la funzione 'prepareData(databasePath)' appartenente ad 'utils.py'. Successivamente vengono definite le variabili 'number_of_solution' e 'number_of_reduced_training_instances'. In particolare, la seconda definisce il numero di istanze che conterrà il nuovo dataset. Le variabili verranno fornite successivamente alla funzione 'runGeneticAlgorithXPS' contenuta in 'PrototypeSelectionProblem.py' essa è delegata all'esecuzione del problema in questione

La funzione ritornerà il nuovo insieme ridotto sottoforma di file csv denominato 'reducedTrainingPS.py'

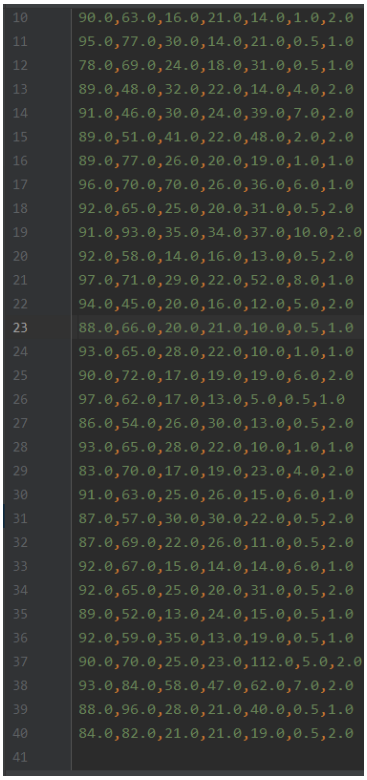
```
def callPS(databasePath):


    x_train, x_test, number_of_features, number_of_classes, number_of_total_instances=utils.prepareData(databasePath)

    number_of_solutions=500
    number_of_reduced_training_instances=40
    chromosomeToEvaluate,fitness=ps.runGeneticAlgorithXPS(number_of_solutions, x_train,number_of_reduced_training_instances )

    print(chromosomeToEvaluate)
    np.savetxt('C:/xampp/htdocs/quantumKNN/python/reducedTrainingPS.csv', x_train[chromosomeToEvaluate,:], delimiter=",", fmt='%s')
```

Screen della funzione 'callPS' che richiama all'esecuzione del problema prototype selection

Nella figura successiva verrà mostrata la differenza tra il dataset originale ed il dataset preprocessato con la tecnica prototype selection.



Screen del dataset originale ‘Bupa.csv’ contenente 325 righe di dati e 7 colonne



Screen del train set del dataset ‘bupa.csv’ ridotto con PS contente 40 righe e 7 colonne

4.2.4 PREPROCESSING CON FEATURE EXTRACTION TRAMITE PCA

La seconda tecnica impiegata nel preprocessing è la feature extraction che permette la riduzione definita di features sul dataset.

Questa tecnica adotta lo stesso principio della precedente. Si richiama tramite l'assegnazione con valore True della variabile '*featureExtraction'*. La funzione in questione è '*featureExtractionPCA2()'* appartenente al codice '*featureExtractionPCA1.py'*.

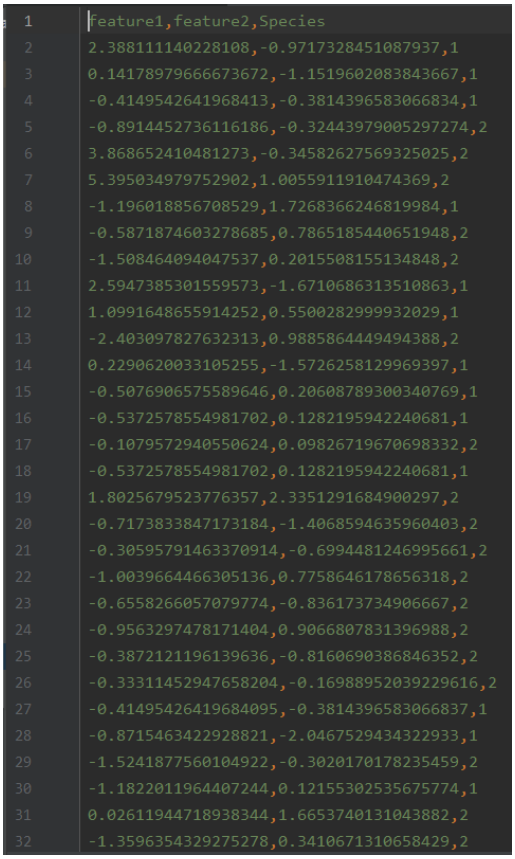
La funzione prende come parametro il file ed il numero di features da ridurre. Legge attraverso la funzione *'read_csv()*' di pandas il file in questione, rimuove le classi dal file ed esegue la PCA (vedi 3.1) sul nuovo insieme di dati, e restituisce in nuovo file denominato 'yourPCA.csv' il nuovo dataset. Siccome la feature extraction nel nostro caso viene eseguita sul train set e test set del dataset quando verrà eseguita sul test set, e siccome il file yourPCA.csv è stato già generato, in questo caso produrrà un nuovo file 'yourPCA1.csv' che lo contraddistingue dal primo.

```python
def featureExtractionPCA2(filename, features):
    f = pd.read_csv(filename)
    keep_col = features
    new_f = f[keep_col]
    new_f.to_csv("tempPCA.csv", index=False)

    dataset = pd.read_csv('tempPCA.csv')

    #X1 = dataset.drop('Id', 1)
    X = dataset.drop('Species', 1)

    y = dataset['Species']
    print(dataset.head())
    #print(y)

    # Splitting the dataset into the Training set and Test set
    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

    from sklearn.preprocessing import StandardScaler

    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    from sklearn.decomposition import PCA

    pca = PCA(n_components=2)
    X_train = pca.fit_transform(X_train)
    X_test = pca.transform(X_test)
```

```python
    explained_variance = pca.explained_variance_ratio_
    # print(explained_variance)

    z = np.concatenate((X_train, X_test))
    # write csv data
    if not os.path.exists('C:/xampp/htdocs/quantumKNN/python/yourPCA.csv'):
        np.savetxt('C:/xampp/htdocs/quantumKNN/python/yourPCA.csv', z, delimiter=",", fmt='%s')
    else:
        np.savetxt('C:/xampp/htdocs/quantumKNN/python/yourPCA1.csv', z, delimiter=",", fmt='%s')

    # print(z)

    return z
```

Screen del codice della funzione che esegue feature extraction by PCA

Nella figura successiva verrà mostrata la differenza tra il dataset originale (vedi figura 4.x) ed il dataset preprocessato con la tecnica di feature extraction.



```
feature1,feature2,Species
2.388111140228108,-0.9717328451087937,1
0.14178979666673672,-1.1519602083843667,1
-0.4149542641968413,-0.3814396583066834,1
-0.8914452736116186,-0.32443979005297274,2
3.868652410481273,-0.34582627569325025,2
5.395034979752902,1.0055911910474369,2
-1.196018856708529,1.7268366246819984,1
-0.5871874603278685,0.7865185440651948,2
-1.508464094047537,0.2015508155134848,2
2.5947385301559573,-1.6710686313510863,1
1.0991648655914252,0.5500282999932029,1
-2.403097827632313,0.9885864449494388,2
0.2290620033105255,-1.5726258129969397,1
-0.5076906575589646,0.20608789300340769,1
-0.5372578554981702,0.1282195942240681,1
-0.1079572940550624,0.09826719670698332,2
-0.5372578554981702,0.1282195942240681,1
1.8025679523776357,2.3351291684900297,2
-0.7173833847173184,-1.4068594635960403,2
-0.30595791463370914,-0.6994481246995661,2
-1.0039664466305136,0.7758646178656318,2
-0.6558266057079774,-0.836173734906667,2
-0.9563297478171404,0.9066807831396988,2
-0.3872121196139636,-0.8160690386846352,2
-0.33311452947658204,-0.16988952039229616,2
-0.41495426419684095,-0.3814396583066837,1
-0.8715463422928821,-2.0467529434322933,1
-1.5241877560104922,-0.3020170178235459,2
-1.1822011964407244,0.12155302535675774,1
0.02611944718938344,1.6653740131043882,2
-1.3596354329275278,0.3410671310658429,2
```

Screen del dataset ottenuto con tecnica di feature extraction by PCA avente 3 colonne e 345 righe

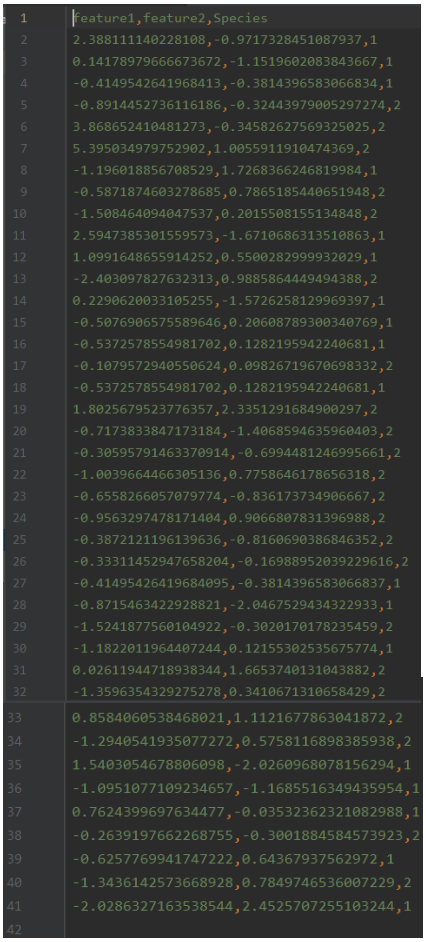
4.2.5 PREPROCESSING COMBINATO

La piattaforma permette, nel caso l'utente scelga di eseguire entrambe le tecniche di preprocessing, di eseguire una riduzione combinata.

In questo caso, le variabili 'prototypeSelection' e 'featureExtraction' saranno entrambe settate con il valore True nel nostro codice principale (*mainPreprorcessing.py*). Pertanto, verrà eseguito prima la tecnica di Prototype Selection sul dataset di training, successivamente la feature extraction sul risultato della PS, e di seguito applicata al test set. Il risultato consisterà in un nuovo dataset avente colonne e righe ridotte, più tecnicamente istanze e features minori rispetto al dataset originale.

L’impego di queste due tecniche comporterà, come analizzato successivamente, un vantaggio sia in ordine temporale, costo computazionale e accuratezza rispetto al non adottare nessuna di queste tecniche.

Nel nostro caso, questa tecnica è stata applicata al dataset di training *‘bupa.csv’* contenente 325 righe e 7 colonne, il risultato consisterà in un nuovo dataset avente 40 righe e 3 colonne (figura 4.x)



```
feature1,feature2,Species
2.388111140228108,-0.9717328451087937,1
0.14178979666673672,-1.1519602083843667,1
-0.4149542641968413,-0.3814396583066834,1
-0.8914452736116186,-0.3244397900529727 4,2
3.868652410481273,-0.34582627569325025,2
5.395034979752902,1.0055911910474369,2
-1.196018856708529,1.7268366246819984,1
-0.5871874603278685,0.7865185440651948,2
-1.508464094047537,0.2015508155134848,2
2.5947385301559573,-1.6710686313510863,1
1.0991648655914252,0.5500282999932029,1
-2.403097827632313,0.9885864449494388,2
0.2290620033105255,-1.5726258129969397,1
-0.5076906575589646,0.20608789300340769,1
-0.5372578554981702,0.1282195942240681,1
-0.1079572940550624,0.09826719670698332,2
-0.5372578554981702,0.1282195942240681,1
1.8025679523776357,2.3351291684900297,2
-0.7173833847173184,-1.4068594635960403,2
-0.30595791463370914,-0.6944811246995661,2
-1.0039664466305136,0.7758646178656318,2
-0.6558266057079774,-0.836173734906667,2
-0.9563297478171404,0.9066807831396988,2
-0.3872121196139636,-0.8160690386846352,2
-0.33311452947658204,-0.16988952039229616,2
-0.41495426419684095,-0.3814396583066837,1
-0.8715463422928821,-2.0467529434322933,1
-1.5241877560104922,-0.3020170178235459,2
-1.1822011964407244,0.12155302535675774,1
0.02611944718938344,1.6653740131043882,2
-1.3596354329275278,0.3410671310658429,2
0.8584060538468021,1.1121677863041872,2
-1.2940541935077272,0.5758116898385938,2
1.5403054678806098,-2.0260968078156294,1
-1.0951077109234657,-1.1685516349435954,1
0.7624399697634477,-0.03532362321082988,1
-0.2639197662268755,-0.3001884584573923,2
-0.6257769941747222,0.64367937562972,1
-1.3436142573668928,0.7849746536007229,2
-2.0286327163538544,2.452570725510 3244,1
```

Screen del dataset di training ‘bupa.csv’ ridotto con f.e. by PCA

4.2.6 myQSVM()

Per poter utilizzare al meglio il qSVM è stato programmato tramite il linguaggio python e per essere fruibile alla nostra applicazione prevede l’utilizzo di una funzione, chiamata myQSVM(train,, test, features, token, qubit=2);

La funzione ha come primo e secondo parametro, sottoforma di file CSV, il dataset di train e test rispettivamente; come terzo parametro un array di stringhe contenente il nome delle features del dataset; un quarto parametro in formato stringa associato al token di IBM Q Experience dell'utente e come ultimo parametro, in formato intero, il numero di qubit da utilizzare, di base impostato a due. La funzione ritornerà il risultato sottoforma di numero float indicando l'accuratezza della classificazione.

Nel cuore della nostra funzione risiede un'altra funzione, è la funzione che implementa il SVMs sul hardware Quantistico di IBM

Essa è definita come:
QSVM(*feature_map*, *training_dataset=None*, *test_dataset=None*, *datapoints=None*, *multiclass_extension=None*, *lambda2=0.001*, *quantum_instance=None*)

Parametri:

- **feature_map** (Union[QuantumCircuit, FeatureMap]) – Feature map module, used to transform data;
- **training_dataset** (Optional[Dict[str, ndarray]]) – Training dataset;
- **test_dataset** (Optional[Dict[str, ndarray]]) – Testing dataset.
- **datapoints** (Optional[ndarray]) – Prediction dataset.
- **multiclass_extension** (Optional[MulticlassExtension]) – Se il numero delle classi è maggiore di due quindi uno schema multiclasse deve essere fornito sotto forma di estensione multiclasse.
- **lambda2** (float) – Fattore di regolarizzazione della norma.
- **quantum_instance** (Union[QuantumInstance, Backend, BaseBackend, None]) – Quantum Instance o Backend.

Nel nostro caso questa funzione è stata adattata come mostrato in figura:

```
51      qsvm = QSVM(feature_map, training_input, test_input, multiclass_extension=AllPairs())
```

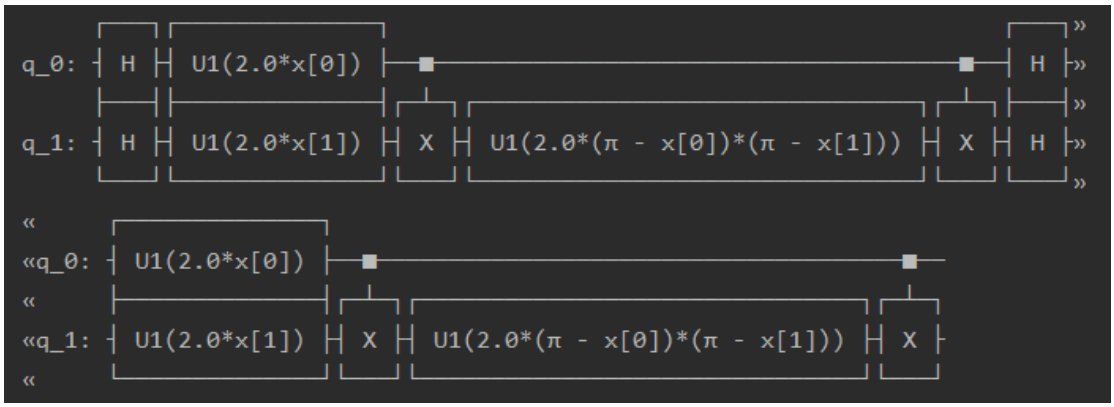
Screen della funzione utilizzata nel nostro qsvm

training_input e *testing_input* sono i nostri dataset di train e test. In particolare, il parametro *feature map* indica la mappa delle caratteristiche utilizzata per trasformare i dati, essa utilizza la funzione **ZZFeatureMap(),** un 'evoluzione della *Second-order Pauli-Z.*

```
49      feature_map = ZZFeatureMap(feature_dimension=feature_dim, reps=2, entanglement='linear')
```

Screen della funzione ZZFeatureMap()

feature_dim nel nostro esperimento assume valore 2, generando il seguente circuito quantistico:



Screen del circuito quantistico applicato al dataset bupa.csv con preprocessing combinato

Nel caso in cui abbiamo più di una classe nel nostro dataset, all'interno della funzione QSVM() viene specificata la ***multiclass_extension().*** L' estensione multiclasse utilizza tecniche diverse per eseguire la classificazione multiclasse, nel nostro algoritmo abbiamo utilizzato l'estensione AllPairs():

- **AllPairs()** : Nella riduzione di tutte le coppie, si addestra k (k − 1) / 2 classificatori binari per un problema multiclasse; ognuno riceve i campioni di un paio di classi dal set di addestramento originale e deve imparare a distinguere

queste due classi. Al momento della previsione, viene utilizzato uno schema di voto ponderato: tutti i classificatori k (k − 1) / 2 vengono applicati a un campione invisibile e ad ogni classe viene assegnata la somma di tutti i punteggi ottenuti dai vari classificatori. Il classificatore combinato restituisce come risultato la classe che ottiene il valore più alto.

In fine il valore viene restituito tramite la funzione **run()** applicata alla funzione **QSVM()**

```
56        result = qsvm.run(quantum_instance)
```

Screen della funzione run()

La funzione *run()* prende come parametro un'altra funzione, quale **QuantumInstance().** Nel nostro esperimento abbiamo settato la funzione sopracitata come:

```
53        quantum_instance = QuantumInstance(backend, shots=1024, seed_simulator=seed, seed_transpiler=seed)
```

Screen della funzione QuantumInstance()

Dove il primo parametro è il tipo di hardware quantistico utilizzato, il secondo 'shots' è il numero di ripetizioni su ogni singolo circuito per il campionamento. Di seguito, il codice completo del qSVM:

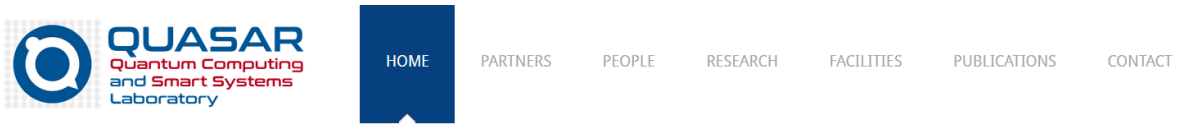
```
from qiskit import IBMQ
from qiskit.aqua.components.multiclass_extensions import AllPairs
from qiskit.circuit.library import ZZFeatureMap
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import QSVM
from datasetLoader import LoadDataset

token = 'e909c180c225ad0228d74d987df754589bf4a740d2902a1c20422f4a576e12a21edf64b17a2b9346ac60a6b47cd7a7446780fca9d96ea2ed5dae7ff29ed4275f'

def myQSVM(train,test,features,token,qubit=2):
    import time
    start_time = time.time()

    IBMQ.enable_account(token)
    provider = IBMQ.get_provider(hub='ibm-q')
    backend = provider.get_backend('ibmq_qasm_simulator')  # Specifying Quantum device
    seed = 8192
    aqua_globals.random_seed = seed
    feature_dim = qubit  # number of qubits

    training_input, test_input = LoadDataset(train, test, features, label='Species')  # creating dataset

    feature_map = ZZFeatureMap(feature_dimension=feature_dim, reps=2, entanglement='linear') #map quantum circuit
    print(feature_map)
    qsvm = QSVM(feature_map, training_input, test_input, multiclass_extension=AllPairs()) #generate qsvm function
    quantum_instance = QuantumInstance(backend, shots=1024, seed_simulator=seed, seed_transpiler=seed) #create instance of quantum hardware
    print('Running....\n')
    result = qsvm.run(quantum_instance) # run algorithm
    print(f'Testing success ratio: {result["testing_accuracy"]}') #get test accracy
    print("--- %s seconds ---" % (time.time() - start_time)) #get a time of execution
```

Screen funzione myQSVM()

4.3 FRONT-END

Per rendere l'applicazione fruibile, in modo rapido e diretto, alla comunità scientifica ed a chi non ha skills sulla programmazione, tramite l'interfaccia grafica di QUANTUM REDAMP si possono utilizzare sia le tecniche di preprocessing ed effettuare la propria classificazione quantistica sul dataset. La nostra home page della nostra piattaforma denominata "QuantumSVM.html", è composta da Header, Body e Footer.

L'header, o intestazione della pagina, è la parte superiore del layout di una pagina web che, solitamente, si ripete identica all'interno di tutte le pagine dello stesso sito. All'interno dell'header, troviamo posto al logo del sito e la navbar (cioè il menu di navigazione che contiene i link alle principali sezioni/pagine del sito web). Di seguito uno screenshot dell'header del nostro sito:



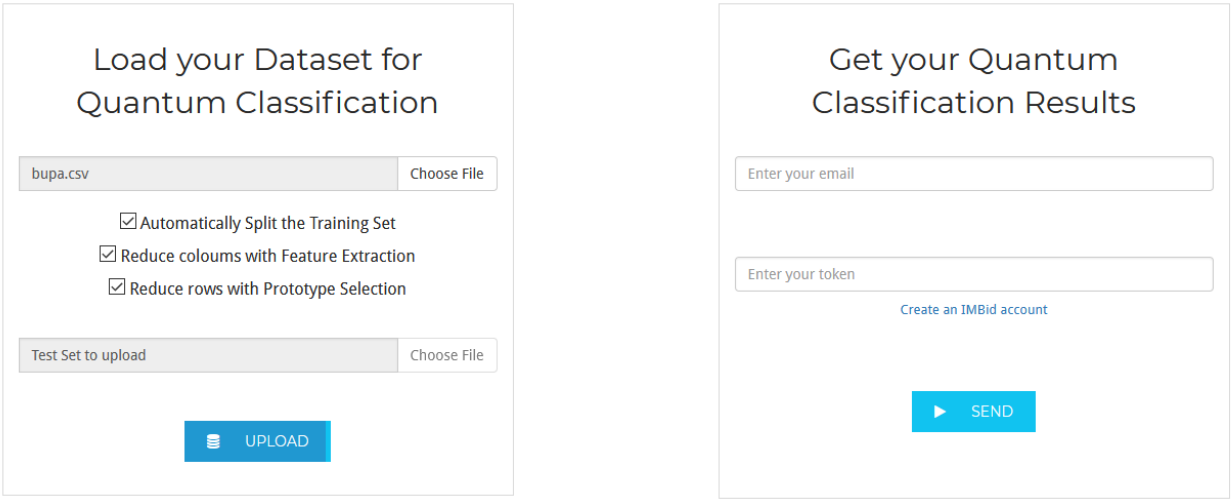
Screenshot header home page QUANTUM REDAMP dov'è visibile la navbar

Il linguaggio HTML contempla due tag che potrebbero venire alla mente quando si parla di header. Si tratta dei tag <head> e <header>. Il primo tag - <head> - è utilizzato per gestire le intestazioni del documento HTML, cioè per ospitare i meta-tag, gli script, i fogli di stile ed altri elementi strutturali e "invisibili" all'utente. Il secondo tag <header> (nel nostro caso ospitato nel tag <body>, di seguito analizzato) è stato introdotto da HTML5 per contenere le "sezioni introduttive" quindi viene essere utilizzato come contenitore per l'header della pagina web (all' interno del nostro sito si gestiscono l'header incapsulandolo all'interno di comuni tag opportunamente utilizzati per gestire le sezioni della *navbar*).

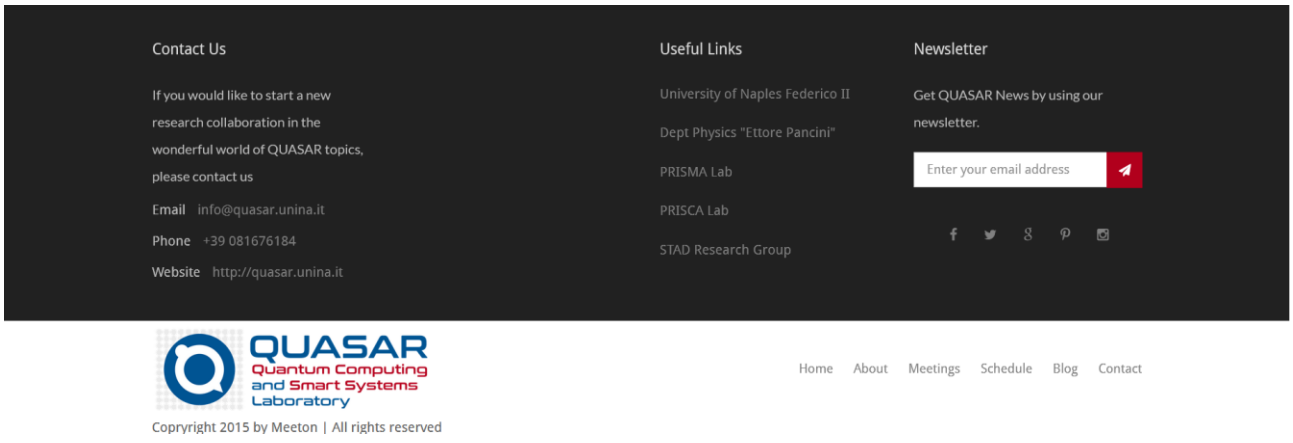
Il tag <body> definisce ed identifica il corpo della pagina, cioè la "porzione visibile" di un qualsiasi documento HTML. Avendo lo scopo di contenere tutti gli elementi visibili, è evidente che i tag <body> e </body> saranno rispettivamente il tag di apertura e di chiusura al cui interno verranno inseriti elementi quali paragrafi di testo, tabelle, immagini, link, ecc. Nel nostro caso ospita i form per effettuare il caricamento del dataset

per il preprocessing ed i form per l'inserimento delle informazioni utili all'ottenimento del risultato della classificazione quantistica.

Di seguito screenshot del body del sito, composto da due form:



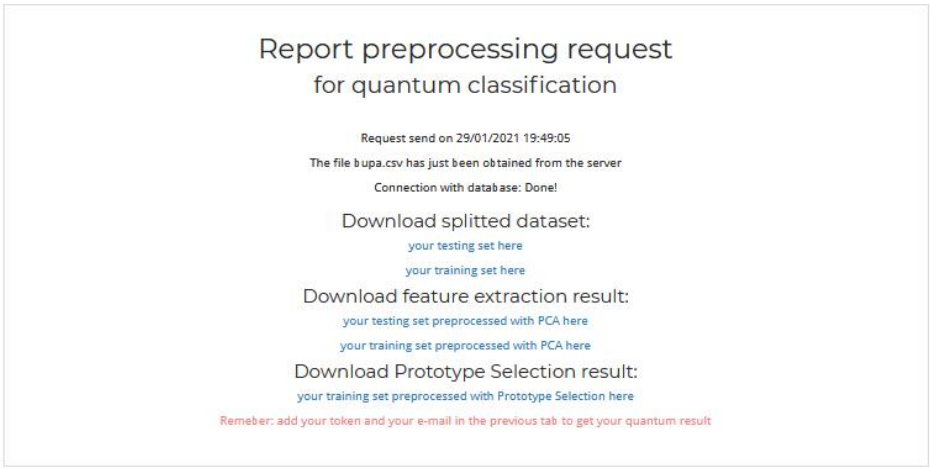
Il footer, anche detto pié di pagina, è la parte inferiore del layout di una pagina web che, solitamente, si ripete identica all'interno di tutte le pagine dello stesso sito. All'interno del footer, solitamente, vengono inserite una serie di informazioni tipiche come, ad esempio, una sezione contatti, link utili, normativa privacy, iscrizione alla newsletter, diritto d'autore ecc. Di seguito uno screenshot del footer di questo sito:



Screenshot header home page QUANTUM REDAMP dov'è visibile il footer

4.3.1 UPLOAD E DOWNLOAD DATI

Inserito l'upload del file tramite i due form dedicati, train set e test set presenti nella Home Page e selezionata la tecnica di riduzione da effettuare sul dataset; cliccando sul pulsante "upload", viene richiamato lo script PHP che a sua volta richiama il back-end python per effettuare il preprocessing sul dataset. La pagina PHP utilizzata nella nostra piattaforma è "upload_dataset.php". Una volta richiamato lo script ed eseguito, la pagina web appare graficamente come la home page; contiene lo stesso header e footer, con la differenza sul body. Il body della pagina restituisce la data e l'orario della trasmissione della richiesta di preprocessing, un messaggio di corretto inserimento del file nella cartella del server ed un terzo messaggio di conferma salvataggio del path all'interno del database. In fine riceveremo la possibilità di effettuare il salvataggio dei file preprocessati con le tecniche selezionate tramite la pagina 'download.php'.



Screen del front-end della pagina upload.php

4.4 STRUMENTI

Di seguito, verranno descritti gli strumenti usati per la corretta implementazione della piattaforma.

4.4.1 IBM Q Experience

L'IBM Q Experience è un servizio offerto gratuitamente da IBM che permette a chiunque di avere a che fare con un computer quantistico. Sono presenti risorse didattiche per imparare a scrivere il primo circuito, strumenti di comunità come una piattaforma di domande e risposte e soprattutto un sistema per creare i propri algoritmi. Si può accedere agli ordini di esecuzione inviati tramite Qiskit e recuperarne i risultati in un secondo momento.

4.4.2 Qiskit

Framework open-source per il quantum computing, scritto principalmente in python e fondato da IBM Research [13]; fornisce tools per la creazione e la manipolazione di programmi quantistici, eseguiti su prototipi di dispositivi quantistici tramite IBM Q Experience. Qiskit offre la capacità di sviluppare software quantistici sia a basso livello sia a livelli astratti. Queste funzionalità sono fornite dai seguenti componenti distinti per utilizzo: *Terra, Aqua, Aer e Ignis*. Nel nostro caso ci concentreremo sulla componente Aqua. Qiskit Aqua contiene una libreria di algoritmi quantistici cross-domain su cui è possibile costruire applicazioni per il calcolo quantistico a breve termine. Di fatti, il nostro codice qSVM e le funzioni presenti risiedono proprio in questa componente.

4.4.3 Pycharm

PyCharm è un ambiente di sviluppo integrato utilizzato nella programmazione, in particolare per il linguaggio Python

4.4.4 Anaconda

Anaconda è una distribuzione dei linguaggi di programmazione Python e R per il calcolo scientifico. Che mira a semplificare la gestione e la distribuzione dei pacchetti.
Le versioni dei pacchetti in Anaconda sono gestite dal sistema di gestione dei pacchetti conda. Esiste anche una piccola versione bootstrap di Anaconda, utilizzata nel nostro caso studio, chiamata Miniconda, che include solo conda, Python, i pacchetti da cui dipendono e un piccolo numero di altri pacchetti.

4.4.5 MySQL

MySQL è un sistema open source di gestione di database relazionali SQL, nel nostro caso abbiamo utlizzato una sua fork chiamata HeidiSQL per la gestione del nostro database

4.4.6 Php

Html da solo non sempre riesce a soddisfare le varie esigenze che una pagina web richiede. Si ricorre così a linguaggi di scripting lato server, uno di questi è proprio PHP, un linguaggio interpretato dal server che rende dinamiche le pagine web. Lo si inserisce all'interno del codice html come un normale linguaggio di scripting. Nel nostro caso è stato utilizzato per collegare lo script lato back-end per poter gestire il preprocessing del Dataset

4.4.7 Xampp

XAMPP, acronimo di Cross Apache MySQL PHP e Perl, è una piattaforma software che facilita l'installazione e la gestione degli strumenti più comuni per lo sviluppo di applicazioni web, raggruppandoli in un unico luogo. Infatti, attraverso un'unica installazione è possibile avere in una sola cartella Apache, cioè il web server ovvero il programma che gestisce le richieste che arrivano da un qualsiasi client attraverso il

protocollo *HTTP*, *MySQL* cioè il DBMS, *PHP* e *Perl*, cioè due linguaggi utili per lo sviluppo di applicazioni web.

4.5 EXPERIMENT

Una volta sviluppati tutti gli strumenti necessari per implementare il quantum support vector machine, passiamo alla fase sperimentale. Si è sperimentata la piattaforma sull'output prodotto dal dataset *Bupa (citazione).* Per esso si è calcolata l'accuratezza ottenuta su ogni tipo di preprocessing effettuato sul datatset. L'algoritmo è stato implementato con il pacchetto *qiskit 0.23.2 , qiskit-aqua 0.8.1* .

4.5.1 FORMATTAZIONE DEL DATASET

Il file dataset contiene solamente i dati, senza attributi o identificati. Pertanto, vengono aggiunti nuovi dati per formattare il dataset. In particolare, viene aggiunto un attributo che identifica ogni istanza denominato 'Id'; una nuova istanza, nella riga 0, contenente il nome delle features denominata 'addAttribute()' . Entrambe le funzioni prendono come parametro il file da standardizzare. Questo processo viene applicato in quanto la funzione qSVM() prende in input solamente dataset formati: dalla prima riga che identifica gli attributi, ovvero le features; un campo degli attributi deve essere dedicato all'identificativo dell'istanza ('Id') e l'attributo per distinguere le classi viene identificato con il campo 'Species'.

La funzione che aggiunge gli identificativi alle istanze è aggID() per la fase di training, essa prende come argomento il path del dataset di training in formato CSV e ritorna un nuovo file chiamato 'IdFeatureDataset_compatted.csv'

La funzione che aggiunge gli identificativi alle istanze è aggIdTesting() per la fase di testing, essa prende come argomento il path del dataset di training in formato CSV e ritorna un nuovo file chiamato 'IdData_Testing_compatted.csv

La funzione che aggiunge gli attributi al file di dati è addAttribute(), essa prende come argomento il path del dataset in formato CSV e ritorna un nuovo file 'featureDataset.csv'

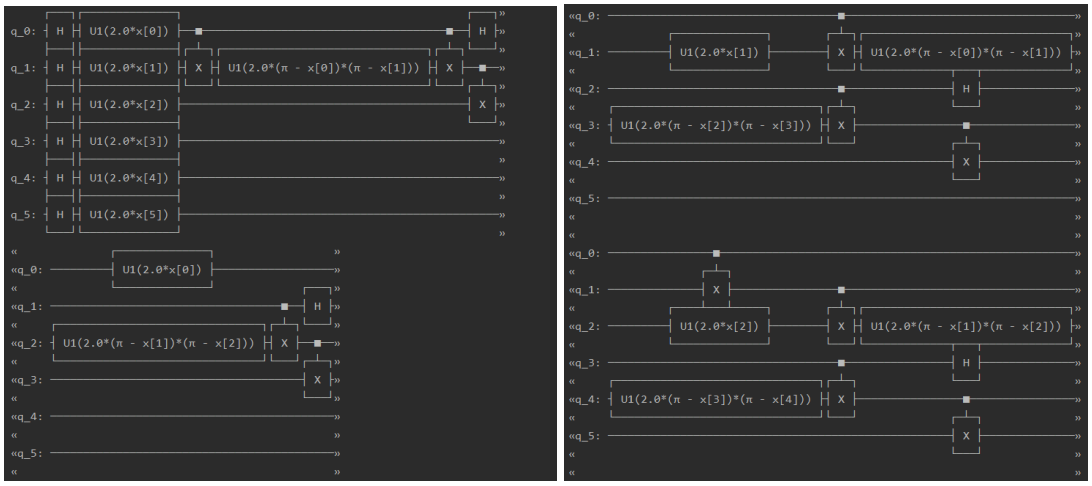
4.5.2 PREPROCESSING WITH PROTOTYPE SELECTION FOR QSVM

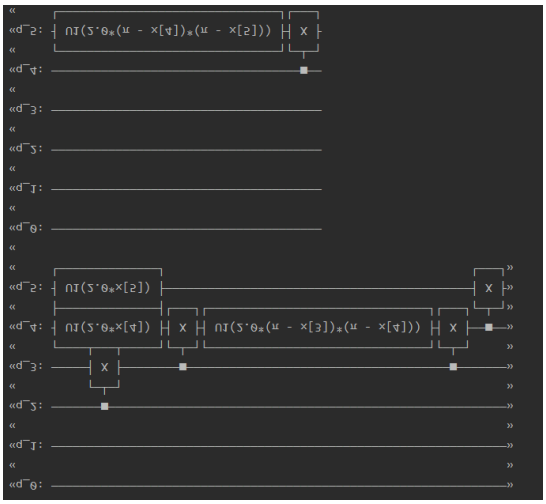
Quando la variabile ‘prototypeSelection’ assume valore True ed eseguito lo split del dataset, viene chiamato il prototype selection tramite la funzione ‘callPS.py’ prendendo come argomento il dataset di training. Nel nostro esperimento all’interno del codice ‘mainPreprocessing.py’, la funzione ‘callPS()’ viene settata con il parametro ‘Data_testing.csv’ eseguendo il problema in questione sul dataset preso come argomento.

La funzione ‘*callPS()*’ ritornerà il dataset preprocessato in un nuovo file, in formato CSV, denominato ‘*reducedTrainingPS.csv’* formato dai nuovi valori ed ha numero di istanze quante indicate nella funzione sopracitata (vedi par. 4.2.3). Standardizzato il dataset di training e testing (vedi par. 4.4.2) vengono dati gli output della fase di standardizzazione all’algoritmo di qSVM. Quindi la funzione in questione sarà:

```
qsvm.myQSVM('IdFeatureDataset_compatted.csv','IdData_Testing_compatted.csv',
features,token,6)
```

I primi due argomenti sono il dataset di testing, il secondo di training, il terzo le features dichiarate come variabile globale, quarto parametro il token dichiarato come variabile globale e il quinto equivale alla lunghezza dell’array delle features rappresentante il numero di qubit utilizzati (vedi par. 4.2.6). Eseguita la funzione, viene generato il circuito per la risoluzione del problema:





Screen del circuito quantistico utlizzato per il dataset preprocessato con il PS

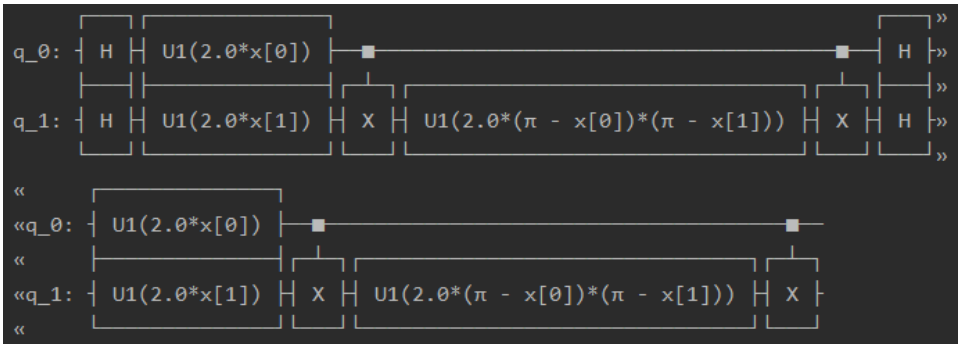
4.5.3 PREPROCESSING WITH FEATURE EXTRACTION PER QSVM

Quando la variabile '*featureExtraction'* del codice 'mainPreprocessing.py' assume valore True, ed eseguito lo split del dataset, viene chiamata la feature extraction tramite la funzione *'featureExtractionPCA2(filename, features)'* prendendo come argomento il dataset di training in prima fase e poi il dataset di testing in seconda fase. Nel nostro esperimento all'interno del codice 'mainPreprocessing.py', la funzione '*featureExtractionPCA2(filename, features)*' viene settata con il primo parametro '*Data_training.csv',* il secondo parametro con '*features1',* array di stringhe contente gli attributi del dataset. La stessa funzione viene applicata al dataset di testing con il parametro '*filename=Data_testing.csv'.* Il risultato della funzione (*vedi par. 4.2.4)*, applicato ai due dataset, produrrà due nuovi file 'yourPCA.csv', 'yourPCA1.csv' con i nuovi valori, avente due features, un attributo per la classe. Standardizzato il dataset di training e testing (vedi par. 4.4.2) vengono dati gli output della fase di formattazione all'algoritmo di qSVM. Quindi la funzione in questione risulterà:

```
qsvm.myQSVM('Data_PCA_testing.csv', 'Data_PCA_testing.csv', featuresPCA,
token, 2)
```

I primi due argomenti sono il dataset di testing, il secondo di training, il terzo le features dichiarate come variabile globale, quarto parametro il token dichiarato come variabile globale e il quinto equivale alla lunghezza dell'array delle features rappresentante il numero di qubit utilizzati (vedi par. 4.2.6).

Eseguita la funzione, viene generato il circuito per la risoluzione del problema.



4.5.4 PREPROCESSING COMBINATO CON PROTOTYPE SELECTION E FEATURE EXTRACTION PER QSVM

In questo caso per attuare la tecnica come in *par. 4.2.5* le variabili *prototypeSelection* e *featureExtraction* assumono entrambe valore True. Nello specifico, verrà applicata la tecnica di prototype selection al dataset di training e la feature extraction al risultato della computazione della prototype selection ed al dataset di testing.

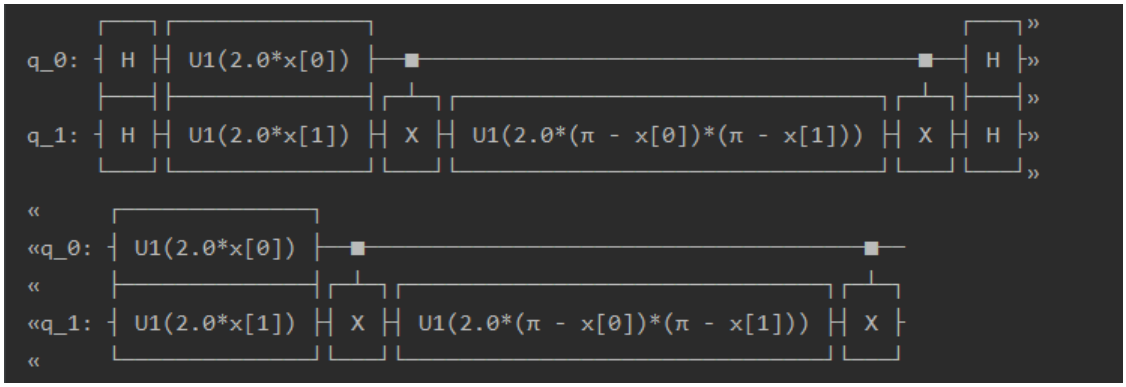
La procedura risulterà la stessa mostrata in *par. 4.4.3* per ridurre il numero di istanze sul dataset di testing tramite la PS. Mentre per ridurre il numero delle colonne del dataset tramite la feature extraction verrà applichiamo questa tecnica al risultato della PS quindi applicheremo come mostrato in *par 4.4.4* la funzione '*featureExtractionPCA2(filename,features)'* prima al dataset di training quindi '*reducedTrainingPS.csv'* e poi al file contente il dataset di testing *'Data_testing.csv'*.

I risultati della funzione di FE.verrano prima formattati per essere standardizzati alla classificazione tramite il qSVM *(vedi par. 4.4.2)* e successivamente forniti alla sua funzione.

La funzione risultate con due qubit per la classificazione quantistica sarà:

```
qsvm.myQSVM('Data_PCA_testing.csv', 'Data_PCA_testing.csv', featuresPCA,
token, 2)
```

Eseguita la funzione viene generato il seguente circuito quantistico per la classificazione:



Screen del circuito quantistico utlizzato per il dataset preprocessato con il PS + F.E.

CAPITOLO 5
RISULTATI SPERIMENTALI

5.1. DESCRIZIONE DATASET BUPA

Il dataset *Bupa* contiene un insieme di informazioni reali del settore medico sulla valutazione di pazienti con malattia epatica. Il file, in fomato CSV, è formato da 345 istanze e 7 attributi. Nel merito, le prime cinque variabili sono le analisi del sangue di pazienti ritenute sensibili a qualche malattia epatica che potrebbe derivare da un consumo eccessivo di alcol. Ogni riga del set di dati costituisce la registrazione di un singolo individuo maschio. Il settimo campo è stato creato dai ricercatori BUPA come selettore usato per la fase di train e test. In particolare, i sette attributi riguardano:

1. MCV (volume corpuscolare medio degli eritrociti);
2. Fosfotasi alcalina;
3. SGPT alanina aminotransferasi;
4. SGOT aspartato aminotransferasi;
5. gammagt gamma-glutamil transpeptidasi;
6. equivale al numero di bevande alcoliche da mezzo litro bevute al giorno;
7. selettore creato dai ricercatori BUPA per suddividere i dati in set di train / test.

5.2 RISULTATI E DISCUSSIONI

Il test è stato condotto sui dati contenuti nel dataset ‘bupa’. I risultati ottenuti sono visibili dalla tabella mostrata in seguito. (*Tab 5.2)*

Si evince dalla tabella che mediante la tecnica di Prototype Selection si è migliorato nettamente il tempo di esecuzione della classificazione quantistica ed anche l'accuratezza (*vedi tab. 5.2 technique 'P.S'*) rispetto al non applicare nessuna tecnica (*vedi technique 'none'*). In particolare, si è passati dal classificare 345 istanze e 7 features in oltre 1h senza nessun preprocessing a 102 secondi applicando la PS. Nel secondo caso, abbiamo applicato la F.E. migliorando notevolmente sia l'accuratezza che il tempo di esecuzione. In particolare, l'accuratezza è aumentata da 0.44 senza preprocessing a 0.9 attuando la F.E., possiamo inoltre notare come il tempo d'esecuzione sia migliorato ulteriormente anche rispetto alla P.S ovvero da 102.38s a 27.32s (vedi tec. F.E.). Il terzo caso esposto in tabella è l'attuazione di entrambe le tecniche, possiamo notare che l'accuratezza è rimasta la medesima della F.E. ma siamo riusciti ulteriormente a migliorare il tempo d'esecuzione da 27.32s a 23.83s. (*vedi tec. 'Combined PS+FE' in tab 5.2*) Il quarto caso mostra la classificazione quantistica applicata ad un dataset senza preprocessing in questo caso l'accuratezza è pari a 0.44 eseguita in 1h 3min 7s (vedi tec. None).

PREPROCESSING TECHNIQUE	**INSTANCES**	**FEATURES**	**ACCURACY (Testing Succes Ratio)**	**TIME EXECUTION (training + testing time espresso in secondi)**
Prototype Selection	60 (40 train + 20 test)	7	53%	102.38
Feature Extraction (PCA)	345 (325 training + 20 Testing)	2	90%	27.32
Combined (P.S + F.E.)	60 (40 train + 20 test)	2	90%	23.82
None	345 (325 training + 20 Testing)	7	45%	3760.216 (1h 3 min 7sec)

Tab 5.2: tabella di confronto sulle varie tecniche di preprocessing per il QSVM

CAPITOLO 6
CONCLUSIONI E SVILUPPI FUTURI

In questa tesi abbiamo presentato i concetti principali del machine learning supervisionato, il quantum computing e come queste due branche dell'informatica possono fondersi per produrre un nuovo concetto il Quantum Machine Learning. Oggi i computer quantistici forniscono ancora quantità misurate di risorse pertanto nasce l'esigenza di adattarsi a questi limiti di implementazione. Il mio lavoro si è concentrato proprio sullo sviluppo di un sistema che possa dimostrare come l'utilizzo di queste tecniche di preprocessing sui dataset sia ancora valido su algoritmi quantistici e che, tramite una user interface web dedicata, possa agevolare studenti, ricercatori ad approcciarsi a questi metodi senza difficoltà o necessità di conoscenze sulla programmazione. Si sono analizzate nel dettaglio le tecniche di preprocessing sui dataset, per la riduzione delle istanze e delle caratteristiche. Oggi è difficile mantenere stabile un sistema quantistico, dato che la più piccola perturbazione proveniente dall'esterno tende a interferire e dunque danneggiare il funzionamento del dispositivo. Anche se attualmente, i ricercatori di IBM dichiarano di aver sviluppato protocolli per ridurre questo errore [14] e controllare sistemi a più qubit. Nel nostro esperimento ci siamo concentrati su Ibm Quantum Experience, il primo computer quantistico in modalità cloud, con un processore fino a 32 qubit. E ancora, il processore appena rilasciato, Ibm Quantum Hummingbird è a 65 qubit sul quale sarebbe interessante attuare nuovi esperimenti al fine di risolvere problemi ancora più difficili. Nei prossimi anni la multinazionale IBM supererà la soglia dei 100 qubit con Ibm Quantum Eagle a 127 qubit, mentre nel 2023 è previsto l'arrivo del processore a 1.121 qubit [15], con cui gli scienziati sarebbero in grado di mantenere una manciata di qubit e farli interagire fra

loro, la base di partenza per la creazione di un computer quantistico a tutti gli effetti. Questi dispositivi potranno cambiare il mondo, a detta dei ricercatori Ibm.

Nel nostro caso siamo stati in grado di dimostrare tramite i risultati sperimentali ottenuti dalla nostra piattaforma, che tramite l'attuazione di tecniche di ottimizzazione delle informazioni fornite in input si è possibile incrementare l'efficacia della risoluzione di classificatori quantistici. Sappiamo che anche il quantum machine learning è un campo estremamente nuovo con una crescita molto forte, alcune delle aree che il QML nel quale irromperà sarà lo studio di nanoparticelle, la creazione di nuovi materiali attraverso mappe molecolari e atomiche, modellazione molecolare per scoprire nuovi farmaci e ricerca medica, comprendere la composizione più profonda del corpo umano, riconoscimento e classificazione migliorati dei modelli, promuovere l'esplorazione dello spazio, creazione di una sicurezza connessa completa tramite la fusione tra IoT e blockchain. Con sviluppi più sorprendenti, dovuto dalla crescita dei computer quantistici il QML risolverà più problemi di quanto avremmo mai potuto immaginare.

BIBLIOGRAFIA

[1] https://arxiv.org/abs/1203.5813
Quantum computing and the entanglement frontier, Jhon Preskill

[2] Ethem Alpaydin. Introduction to Machine Learning, Second Edition. The MIT Press, 2010.

[3] Benenti Giuliano, Giulio Casati, Giuliano Strini. Principles Of Quantum Computation And Information - Volume I: Basic Concepts. World Scientific Publishing, 2004.

[4] D.P. DiVincenzo, D. Loss, Quantum information is physical,Superlattices and Microstructures, https://scholar.smu.edu/datasciencereview/vol1/iss4/11

[5] Machine learning: supervised methods, Danilo Bzdok, Martin Krzywinski & Naomi Altman: https://www.nature.com/articles/nmeth.4551

[6] https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
A Practical Guide to Support Vector Classication, Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin

[7]https://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html
Support vector machines: The linearly separable case

[8] https://www.sciencedirect.com/science/article/abs/pii/S0925231203003734
Advanced support vector machines and kernel methods,
V.DavidSánchez

[9] https://doi.org/10.1038/nature23474
Biamonte, J., Wittek, P., Pancotti, N. et al.
Quantum machine learning. Nature 549, 195–202 (2017).

[10] https://arxiv.org/abs/1307.0471
Patrick Rebentrost, Masoud Mohseni, Seth Lloyd

[11] Ringnér, M. What is principal component analysis?.
Nat Biotechnol **26,** 303–304 (2008).
https://doi.org/10.1038/nbt0308-303

[12] G.Acampora and A. Vitiello, "TSSweb: a Web Tool for Training Set Selection," *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Glasgow, United Kingdom, 2020, pp. 1-7, doi: 10.1109/FUZZ48607.2020.9177677

[13] Qiskit API documentation, 2019 : https://qiskit.org/documentation/

[14]https://www.ibm.com/blogs/research/2014/06/dealing-with-errors-in-quantum-computing/
Dealing with errors in quantum computing, Jerry Chow

[15] https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/
IBM’s Roadmap For Scaling Quantum Technology, Jay Gambetta