UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Identificazione di requisiti di privacy tramite QNLP

RELATORE

Prof. Fabio Palomba

Dott. Francesco Casillo

Università degli Studi di Salerno

CANDIDATO

Anthony Zunico

Matricola: 0512110374

Anno Accademico 2022-2023

Questa tesi è stata realizzata nel

"A volte sono le persone che nessuno può immaginare a fare qualcosa, che fanno cose che nessuno può immaginare."

Alan Turing

Abstract

Nel campo del machine learning lo sviluppo di nuove tecnologie è stato cruciale nel miglioramento delle prestazioni e delle predizioni di un modello. L'apprendimento quantistico, infatti, rappresenta un nuovo modo di approcciarsi alle varie problematiche trattate dai modelli di machine learning che sfruttano meccanismi quantistici come il principi di sovrapposizione, che migliorano l'efficienza del nostro modello e la precisione delle sue predizioni.
Lo scopo di questa tesi é quello di utilizzare l'apprendimento quantistico per l'identificazione dei requisiti di privacy, analizzeremo i risultati confrontandoli con quelli dell'apprendimento classico facendo una serie di considerazioni, in modo da avere un punto di partenza su cui poter attuare sviluppi futuri di tale studio.
Gli studi effettuati hanno permesso di avere una corretta panoramica di come l'approccio quantistico si differenzia rispetto all'approccio classico nell'identificazione dei requisiti di privacy, considerando vantaggi e svantaggi osservando i risultati delle metriche di valutazione, i tempi di esecuzione e le risorse di calcolo impiegate.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto Applicativo

Nell'ingegneria dei requisiti avviene un processo che si occupa di raccogliere e specificare tutti i servizi che un sistema deve fornire. Gli obiettivi principali dell'ingegneria dei requisiti sono quello di comprendere il problema, definire i requisiti, analizzare i requisiti inseriti, gestire le relazioni tra requisiti e il loro aggiornamento, validare i requisiti in base alle esigenze dell'utente [1]. Tra i requisiti che andiamo a definire tra i più importanti ci sono i requisiti di sicurezza, che garantiscono al sistema di resistere alle minacce come il furto di dati o gli accessi non autorizzati.
Tra i requisiti di sicurezza possiamo definire ad esempio:

- Requisiti di Autenticazione;
- Requisiti di Autorizzazione;
- Requisiti di Conformità Normativa;
- Requisiti di Gestione degli Accessi;
- Requisiti di Privacy;

- Requisiti di Protezione dei Dati.

Un sottoinsieme specifico che abbiamo trattato sono i requisiti di privacy che si concentrano sulla protezione e sulla gestione dei dati personali degli utenti, nel rispetto soprattutto di normative rigorose come il Regolamento Generale sulla Protezione dei Dati (GDPR). È importante identificare i requisiti di privacy a causa di numerose problematiche legate alla comprensione delle normative o alle esigenze degli utenti [2].

1.2 Motivazioni e Obiettivi

Nel campo del machine learning lo sviluppo di nuove tecnologie è stato cruciale nel miglioramento delle prestazioni e delle predizioni di un modello. L'apprendimento quantistico, infatti, rappresenta un nuovo modo di approcciarsi alle varie problematiche trattate dai modelli di machine learning che sfruttano meccanismi quantistici come il principi di sovrapposizione, che migliorano l'efficienza del nostro modello e la precisione delle sue predizioni.
A supporto di questo tipo di approccio sono state create numerose librerie che utilizzano simulatori quantistici e altre funzionalitá che permettono di allenare i nostri modelli e validarli supportando i meccanismi precedentemente citati. Ad oggi é ancora difficile stabilire se l'approccio quantistico sia totalmente superiore all'approccio classico, in particolare a causa della difficoltá di implementazione di determinati algoritmi quantistici e a causa dell'utilizzo di hardware che non riescono a mantenere il carico di lavoro richiesto.
Queste difficoltá rendono necessario testare accuratamente su diversi casi di studio le pipeline quantistiche e studiarne le criticitá confrontandole con quelle delle pipeline classiche. Lo scopo di questa tesi é proprio quello di utilizzare queste pipeline per l'identificazione dei requisiti di privacy, una volta simulati analizzeremo i risultati facendo delle considerazioni finali in modo che questo possa essere un punto di partenza per sviluppi futuri di tale studio.

1.3 Risultati Ottenuti

Gli studi effettuati hanno portato alla simulazione di due classificatori per l'identificazione dei requisiti di privacy, uno che si basa sugli strumenti classici dell'apprendimento e uno che utilizza concetti quantistici per l'esecuzione della propria task.
La simulazione di questi classificatori è stata fatta sul dataset creato da uno studio di Riaz et al. [3] che è formato da circa 5980 istanze che sono state successivamente analizzate ed è stata applicata una pulizia dei dati che ha portato ad avere solamente 3737 requisiti di sicurezza di cui 238 di privacy.
Per far elaborare i testi ai modelli sono state applicate delle tecniche di NLP della libreria scelta per trasformare i requisiti di sicurezza in diagrammi di stringhe. Questo ha permesso ai modelli di machine learning di poter essere addestrati su questi dati e poterli valutare grazie all'esecuzione di una 5-fold cross validation che ha portato i seguenti risultati:

- **Approccio classico:** ha migliori risultati rispetto all'approccio quantistico per quanto riguarda la Recall e la F1-score.
- **Approccio quantistico:** ha migliori risultati rispetto all'approccio classico per quanto riguarda la Precision e l'Accuracy.

1.4 Struttura della Tesi

Nel capitolo 2, si trova una panoramica per quanto riguarda l'apprendimento classico e l'apprendimento quantistico con riferimento a lavori e ai framework effettuati nella ricerca di questi approcci. Nello stato dell'arte invece sono stati presentati degli studi pertinenti ai temi che abbiamo affrontato come ad esempio l'identificazione dei requisiti di privacy e al confronto di risultati ottenuto da pipeline quantistiche e pipeline classiche.
Nel capitolo 3, sono descritte l'obiettivo della sperimentazione, l'analisi della libreria scelta, l'analisi del dataset, l'elaborazione dei dati, la costruzione della pipeline quantistica e della pipeline classica, la scelta della tecnica di valutazione e delle metriche da valutare.

Nel capitolo 4, sono presenti tutti i risultati rappresentati come grafici divisi per fold che sono poi stati commentati e valutati riportando i migliori risultati ottenuti per ogni fold, infine sono state fatte delle considerazioni finali del miglior approccio valutando anche i tempi e i costi di esecuzione.
Nel capitolo 5, è stata fatta una descrizione sommaria del lavoro che è stato fatto partendo dal dataset preso in esame finendo con la discussione dei risultati finali, inoltre è stata dedicato uno spazio per discutere di eventuali sviluppi futuri della nostra sperimentazione.

CAPITOLO 2

Background Stato dell'arte

2.1 Background e Stato dell'arte

2.1.1 Computazione classica

Nella teoria classica della computazione, "computabile" è sinonimo di "computabile grazie ad una macchina di Turing". La macchina di Turing esibisce un comportamento classico, deterministico: ad ogni passo si può specificare con certezza la sua configurazione successiva [4]. Alan Turing enunciò tali principi nel 1936, questi principi sono rimasti immutati da allora, nonostante i progressi tecnologici che permettono oggi di produrre dispositivi piú potenti rispetto a quelli che si potevano realizzare in passato. L'assunzione alla base di tali principi é che una macchina di Turing idealizza un dispositivo meccanico di computazione che obbedisce alle leggi della fisica classica [5].

2.1.2 Apprendimento classico

In generale definiamo l'apprendimento classico o automatico come la capacitá di migliorare autonomamente le sue performance su una determinata attivitá attraverso l'esperienza. Tale processo coinvolge l'acquisizione di conoscenza dai dati e la sua

applicazione per prendere decisioni o compiere previsioni [6].
Esistono principalmente due tipi di apprendimento:

- **Apprendimento supervisionato:** all'algoritmo di apprendimento viene fornito un set di dati di addestramento che contiene esempi sia di input che di output desiderati.
- **Apprendimento non supervisionato:** all'algoritmo di apprendimento non vengono forniti esempi di output desiderati, é l'algoritmo che deve identificare schemi e relazioni nei dati.

Le task che il machine learning puó fare sono varie, tra cui:

- **Classificazione:** consiste nel raggruppare i dati in base a una categoria o a una classe (Esempio: Identificare se un'immagine contiene un gatto o un cane).
- **Regressione:** consiste nel prevedere un valore numerico a partire da un set di dati (Esempio: Prevedere il prezzo di un'azione in base a una serie di fattori economici).
- **Clustering:** consiste nel raggruppare i dati in base alla loro somiglianza (Esempio: Identificare gruppi di clienti con interessi simili).
- **Rilevamento di anomalie:** consiste nell'identificare i dati che si discostano dalla norma (Esempio: Identificare transazione flaudolente) [7].

L'apprendimento classico viene utilizzato per numerose applicazioni come il riconoscimento di immagini, la classificazione dei testi, la previsione dei prezzi delle azioni, classificazione dei requisiti non funzionali. Descrivendo i vantaggi possiamo dire che l'apprendimento é efficace nel risolvere problemi di machine learning, sono facilmente interpretabili nel loro funzionamento e possono essere scalati per lavorare con grandi quantitá di dati [8].

2.1.3 Computazione quantistica

La computazione quantistica pone le sue radici su quella che é la meccanica quantistica, cioé come si comporta la materia e l'energia a livello atomico. Grazie al

suo principio di sovrapposizione, le particelle quantistiche possono assumere piú stati allo stesso tempo, in informatica questo viene utilizzato dai cosidetti qubit, i corrispondenti dei bit classici, ma grazie alla loro proprietá sono molto piú efficienti. L'utilizzo di algoritmi quantistici ha portato infatti a cercare di risolvere problemi molto complicati da risolvere con i computer classici [9]. Come abbiamo visto ci sono molti vantaggi nella computazione quantistica, ma ci sono anche molte problematiche legate alla complessitá di progettazione e implementazione di computer quantistici, inoltre gli hardware quantistici hanno un costo elevato e sono di fatto ancora in fase di sviluppo [10].

2.1.4 Casi d'uso per l'apprendimento quantistico

L'apprendimento quantistico durante gli anni ha portato un notevole sviluppo nella creazione di nuove tecnologie efficienti e accurate per una vasta varietá di settori.

OQFL é un framework di apprendimento federato quantum-based che può essere utilizzato per la difesa contro gli attacchi di frode in ITS. OQFL utilizza un'architettura di rete neurale quantistica per migliorare l'accuratezza e la robustezza del modello di apprendimento federato. L'utilizzo di tale framework ha portato a migliorare l'accuratezza del modello di apprendimento federato del 10% e di ridurre la probabilitá di attacchi di frode del 20% [11].

Un altro utilizzo quantum-based é stato proposto nella crittografia di immagini con l'utilizzo di bit quantistici. Tale crittografia avviene in due fasi: la prima consiste nel convertire l'immagine in una sequenza di qubit, la seconda consiste nel codificare la sequenza di qubit utilizzando una chiave quantistica. Una chiave quantistica é un insieme di qubit che vengono utilizzati per codificare la sequenza di qubit dell'immagine con un algoritmo di codifica quantistica; questo rende la crittografia quantistica piú sicura di quella classica e soprattutto efficiente se eseguita su computer quantistici [12].

2.1.5 Identificazione dei requisiti di privacy

I sistemi software raccolgono ed utilizzano una quantitá sempre crescente di dati personali, quindi é importante che i requisiti di privacy di questi sistemi siano identificati e soddisfatti in modo efficace. Per identificare i requisiti di privacy si fa uso di diverse metodologie come:

1. **Privacy by Design:** é un approccio alla progettazione che tiene conto della privacy fin dall'inizio del processo di sviluppo.

2. **Goal-oriented requirements engineering:** é un approccio all'ingegneria dei requisiti che si concentra sugli obiettivi degli stakeholder.

3. **Risk-based approach:** é un approccio che si concentra sull'identificazione e sulla gestione dei rischi per la privacy [13].

Il PriS é un metodo dell'ingegneria dei requisiti di sicurezza che incorpora i requisiti di privacy nelle prime fasi del processo di sviluppo del sistema. Questo metodo facilita l'identificazione dell'architettura del sistema che meglio supporta i processi aziendali correlati alla privacy. Tale metodologia é composta da quattro fasi: **Elicitation** dove i requisiti vengono identificati e documentati, **Analysis** dove i requisiti di privacy vengono analizzati per identificare le minacce alla privacy e i controlli di privacy necessari per mitigare le minacce, **Design** dove i controlli di privacy vengono integrati nell'architettura del sistema, **Evaluation** dove il sistema viene valutato per la conformitá ai requisiti di privacy [14].
F. Casillo et al. [15] (2022) hanno presentato un approccio per ridurre i rischi per la privacy durante lo sviluppo agile del software, rilevando automaticamente le informazioni relative alla privacy nel contesto dei requisiti delle **User Story**. Tale approccio combina l'elaborazione del linguaggio naturale (NLP) e le risorse linguistiche con algoritmi di deep learning per identificare gli aspetti della privacy nelle User Story. Questo processo é stato valutato attraverso uno studio empirico con un dataset di 1680 user stories. I risultati ottenuti mostrano che gli algoritmi di deep learning consentono di ottenere previsioni migliori rispetto a quelle ottenute con metodi di machine learning convenzionali, in piú se si applica il Transfer Learning si puó aumentare l'Accuracy del 10%.

F. Ebrahimi et al. [16] (2022) espongono questo metodo per l'estrazione e la sintesi automatica delle preoccupazioni sulla privacy dalle recensioni di app mobili. Questo metodo é basato su un modello di apprendimento non supervisionato che utilizza un algoritmo di topic modeling per identificare i temi comuni nelle recensioni. I temi identificati vengono poi utilizzati per generare una sintesi delle preoccupazioni sulla privacy espresse dagli utenti. La valutazione di questa metodologia é stata eseguita su un dataset di 10000 recensioni di app mobile e i risultati ottenuti dimostrano che il metodo é in grado di estrarre efficacemente le preoccupazioni sulla privacy e generare sintesi accurate e coincise.

2.2 Stato dell'arte

2.2.1 Riconoscimento dei requisiti di privacy

Pattaraporn Sangaroonsilp et al. [17] (2023) hanno condotto una sperimentazione sull'uso dell'apprendimento automatico per classificare automaticamente i requisiti di privacy negli issue report. Questi ultimi sono documenti che vengono utilizzati per registrare i problemi e le richieste di modifica in un progetto software, garantire la privacy degli utenti finali é uno degli obiettivi della classificazione dei requisiti di privacy. Propongono un metodo legato all'apprendimento automatico per la classificazione che fa utilizzo di un set di caratteristiche linguistiche per identificare i requisiti di privacy negli issue report; un metodo del genere puó ottenere una precisione del 95% nella classificazione dei requisiti.
Chen et al. [18] (2020) propongono un framework unificato per la classificazione dei requisiti di privacy e sicurezza. Questo framework utilizza una combinazione di criteri tra cui:

- **L'oggetto dei requisiti:** i requisiti possono riguardare dati personali, dati sensibili o dati personali sensibili.
- **La natura dei requisiti:** i requisiti possono essere di tipo preventivo, di tipo reattivo o di tipo di sicurezza.
- **Il livello di dettaglio dei requisiti:** i requisiti possono essere generali o specifici.

- **La fonte dei requisiti:** i requisiti possono provenire da diverse fonti come leggi o regolamenti, esigenze di determinati utenti o di organizzazioni.

Il framework é basato su un modello a cinque dimensioni e ció consente di classificare i requisiti in modo completo e accurato, inoltre é stato testato su un sistema di e-commerce che ha provato la sua validitá.
Il National Institute of Standards and Technologogy (NIST) [19] (2019) ha sviluppato un framework per aiutare le organizzazioni a identificare, documentare e soddisfare i propri requisiti di privacy. Il framework utilizza una combinazione di criteri, tra cui:

- **Il livello di sensibilitá dei dati:** i dati possono essere classificati come pubblici, privati o sensibili.
- **Le modalitá di raccolta, utilizzo e condivisione dei dati:** i dati possono essere raccolti, utilizzati e condivisi in vari modi, ognuno dei quali puó presentare diversi rischi di privacy.
- **Le esigenze degli utenti:** i requisiti di privacy possono essere determinati dalle esigenze degli utenti.

Il framework é basato su un modello a tre dimensioni, che consente di essere completo e accurato nella sua classificazione, in piú é molto flessibile ed é disponibile pubblicamente per utilizzarlo su classificazioni personalizzate.

2.2.2 Approcci quantistici presenti in letteratura

Chen et al. [20] (2020) hanno presentato un classificatore che utilizza un algoritmo di machine learning quantistico chiamato quantum support vector machine (QSVM). Il QSVM é stato applicato a un dataset di due classi, e i risultati hanno dimostrato che é in grado di classificare i dati con un'accuratezza del 100%. I risultati ottenuti indicano che gli algoritmi di machine learning quantistici possono essere utilizzati per la classificazione di dati in modo efficiente, ma ci sono alcune limitazioni da dover specificare: il dataset utilizzato é relativamente piccolo, il QSVM é stato applicato a un problema di classificazione binaria e che il tutto é stato testo su un simulatore quantistico e non un computer quantistico reale.

Shang et al. [21] (2023) hanno riportato un caso di studio utilizzando la libreria quantistica Lambeq per implementare un classificatore quantum basato su una QSVM per la classificazione di testo. Stavolta il classificatore é stato testato su un dataset di testi composto da 100000 testi, tale testing é stato fatto anche su un classificatore classico. I risultati hanno dimostrato che il classificatore quantistico é in grado di classificare i testi con un'accuratezza del 98%, mentre un classificatore classico basato su SVM é in grado di classificare i testi con un'accuratezza del 95%.
Nielsen et al. [22] (2011) discutono di come si possono applicare i computer quantistici per la classificazione di immagini. Presentano, inoltre, diversi algoritmi di classificazione quantistica, tra cui:

1. **Classificatore a base di misura:** utilizza una misura quantistica per determinare la classe di un'immagine.

2. **Classificatore a base di simulazione:** simula il comportamento di un classificatore classico su un computer quantistico.

3. **Classificatore a base di apprendimento automatico:** utilizza algoritmi di apprendimento automatico per imparare a classificare le immagini.

Vantaggi che hanno riscontrato sono stati che i classificatori quantistici possono essere più efficienti dei classificatori classici nel calcolare la distanza tra le immagini e i centri dei cluster e che i classificatori quantistici possono anche essere più precisi dei classificatori classici nel classificare le immagini con rumore.

CAPITOLO 3

Metodologia

3.1 Obiettivo della sperimentazione

Lo scopo di questa sperimentazione è studiare l'apprendimento quantistico sviluppando un modello di machine learning quantistico per classificare i requisiti di sicurezza per determinare se sono legati alla privacy o meno, tutto questo rappresentandoli come diagrammi di stringhe. Scegliendo una classificazione del genere ci permetterá di studiare il comportamento di una pipeline quantistica confrontandola con una pipeline classica per valutarne le differenze di performance.
Inizieremo con la ricerca di modelli di machine learning quantistici esistenti insieme alla libreria associata puntando a quelle che possono permettere di utilizzare funzionalitá di simulazione quantistica, classificazione binaria e apprendimento classico; fin da subito la libreria Lambeq ci é sembrata la scelta migliore per la nostra sperimentazione. Quindi, sceglieremo un dataset di requisiti di sicurezza o privacy e elaboreremo i dati per prepararli alla classificazione quantistica e classica. Infine, simuleremo i classificatori di entrambi gli approcci sul dataset scelto.

Definiamo cosí le nostre domande di ricerca a cui verrá data risposta nei risultati della nostra sperimentazione:

RQ$_1$. *In che misura la pipeline classica riesce ad identificare i requisiti di privacy?*

Risponderemo a questa prima domanda eseguendo il training del modello classico sul nostro dataset di requisiti di sicurezza e privacy, valideremo poi modello eseguendo una 5-fold cross validation e infine estrarremo i risultati delle metriche per farne delle considerazioni valutando anche i tempi e le risorse occupate.

RQ$_2$. *In che misura la pipeline quantistica riesce ad identificare i requisiti di privacy?*

Risponderemo a questa seconda domanda eseguendo il training del modello quantistico sul nostro dataset di requisiti di sicurezza e privacy, valideremo poi modello eseguendo una 5-fold cross validation e infine estrarremo i risultati delle metriche per farne delle considerazioni valutando anche i tempi e le risorse occupate.

RQ$_3$. *In che misura la pipeline quantistica si differenzia dalla classica nell'identificazione di requisiti di privacy?*

Risponderemo a questa terza domanda prendendo i risultati che abbiamo estratto precedentemente e confrontarli tra loro in modo da considerare l'approccio migliore per l'identificazione dei requisiti di privacy.

3.1.1 Hardware utilizzato

Inizialmente durante la fase di testing della libreria e pre-processing dei dati la scheda tecnica del computer é stata di una scheda madre **Gigabyte GA-H110M**, un processore **Intel I7 6700K**, **16 GB** di RAM a 2100 MHz. Successivamente a causa di una qualitá tecnica delle componenti, la simulazione delle pipeline e la rappresentazione dei risultati sono state eseguite su **Google Colab** [23] utilizzando il piano Pro offerto dalla piattaforma.
Grazie alla libreria che abbiamo scelto si é evitato l'utilizzo di macchine quantistiche, siccome le funzionalitá offerte includono simulatori quantistici che permettono di eseguire algoritmi quantistici su un computer classico.

3.2 Struttura della sperimentazione

La sperimentazione sarà fatta su due classificatori binari:

1. Il primo sarà un classificatore binario **classico**, che rappresenta un qualsiasi classificatore basato sui principi della meccanica classica.

2. Il secondo sarà un classificatore binario **quantistico**, che utilizza i principi della meccanica quantistica.

Nel corso della nostra sperimentazione ci siamo concentrati nella costruzione di tali classificatori e li abbiamo simulati per studiarne le differenze e le criticità per poi fare delle considerazioni finali sui risultati ottenuti.

3.3 Analisi della libreria

Per questa sperimentazione, abbiamo voluto selezionare una libreria in Python che ci permettesse di utilizzare un modello di machine learning per la classificazione binaria, sia in ambito quantistico che classico, sul nostro dataset.
La scelta della libreria **Lambeq** è stata cruciale in quanto ci consente di convertire qualsiasi frase in un circuiti quantistici e semplifica l'addestramento per esperimenti di NLP sia nel dominio quantistico che in quello classico.

3.3.1 Materiale utilizzato

Durante l'intera durata della sperimentazione, abbiamo esaminato attentamente la documentazione [24] e il repository GitHub di Lambeq [25]. Queste risorse ci hanno dato la possibilità di esplorare tutte le funzionalità utili al nostro scopo, testare diversi esempi di applicazione, accedere a strumenti di debugging e applicare metodologie di valutazione dei risultati ottenuti.

3.3.2 Modalità di analisi

Come prima fase ci siamo concentrati sul leggere le informative della libreria, questo ci ha permesso di formulare delle considerazioni preliminari su come utilizzare i mezzi proposti da Lambeq.
Successivamente, abbiamo dato maggior peso alle guide fornite dalla documentazione, focalizzandoci sull'utilizzo delle funzionalità che sono piú utili per i nostri scopi, come la tokenizzazione, il parsing di stringhe in diagrammi di stringhe, il parsing di diagrammi di stringhe in circuiti e la scelta del modello di machine learning.

3.4 Analisi del Dataset

Per il nostro esempio di classificazione abbiamo utilizzato il dataset creato da Riaz et al. [3], composto da sei documenti riguardanti il settore dell'**assistenza sanitaria**. I ricercatori hanno estratto frasi in linguaggio naturale da ciascun documento, concentrandosi sui requisiti di sicurezza. Successivamente, sono stati categorizzati i requisiti di sicurezza in sette categorie:

1. **Confidentiality**
2. **Availability**
3. **Integrity**
4. **Accountability**
5. **Access Control Identity**
6. **Privacy**
7. **Operational**

Ogni entry nel dataset contiene una serie di informazioni, tra cui il requisito stesso, l'etichetta di sicurezza e categorie associate. Inoltre, è presente una sezione denominata '*Security Words*', in cui vengono elencate tutte le parole chiave che hanno determinato la classificazione di quel requisito come requisito di sicurezza.

Inoltre comprende anche una sezione '*File*', che indica a quale dei 6 dataset scelti dai ricercatori appartiene il requisito in questione.

Un'altra sezione chiamata '*Entities*' contiene una tokenizzazione predefinita del requisito. Questa rappresentazione in forma di token offre una visione dettagliata della struttura del requisito, utile per analisi successive.

Per quanto riguarda le relazioni tra i token all'interno del requisito, queste sono riportate nella sezione '*Dependencies*'. Questa sezione mette in luce le dipendenze tra i

token, facilitando la comprensione delle relazioni sintattiche e semantiche all'interno del requisito stesso.

Infine, la sezione '*Parts of Speech*' svolge un ruolo fondamentale nella definizione di ciascun token come elemento del periodo. Questo passo è cruciale per comprendere il significato e la funzione di ogni parola all'interno del contesto del requisito.

Il dataset completo contiene **5980** requisiti, ciascuno dei quali può appartenere a una o più delle categorie sopra elencate. Tuttavia, per la nostra analisi, ci siamo focalizzati esclusivamente sui requisiti di sicurezza, che ammontano a **4032**. Questa selezione ci permette di escludere i requisiti che sicuramente non fanno riferimento a categorie riguardanti la privacy.

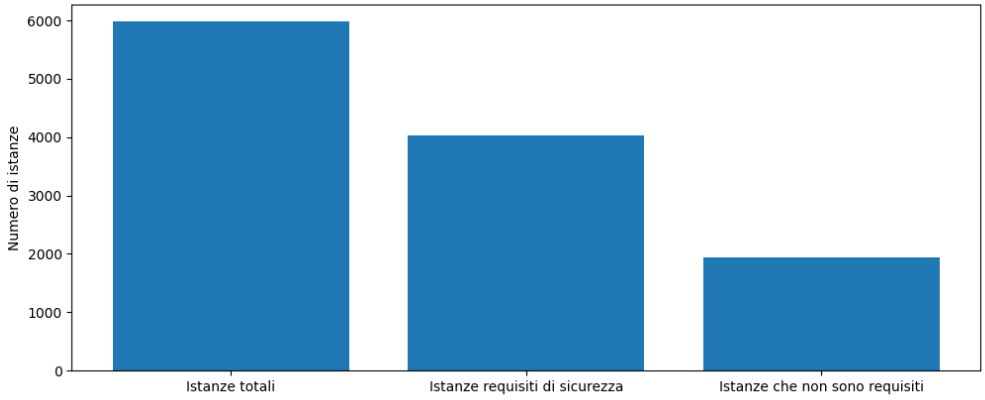

Figura 3.1: Dataset prima di effettuare una pulizia dei dati

É importante notare che i dati non sono ancora stati sottoposti a una fase di pulizia, solo dopo una serie di test applicativi abbiamo apportato significative modifiche o riformulazioni dei requisiti.

3.5 Elaborazione dei dati

3.5.1 Estrazione dei dati

Nel nostro dataset, come spiegato in precedenza, ogni entry ha numerose sezioni, ognuna delle quali fornisce informazioni aggiuntive sui requisiti. Poiché ci interessava sperimentare specificamente con i requisiti di sicurezza e privacy, abbiamo estratto solo queste informazioni dal dataset grazie all'utilizzo della libreria python **pandas** [26], che ci ha permesso di gestire agevolmente i dati strutturati e creare un nuovo dataset partendo da quello originale.

Abbiamo definito un processo di estrazione che ci ha permesso di isolare i requisiti di sicurezza all'interno del dataset. Per fare ciò, abbiamo analizzato ciascuna entry e individuato i requisiti che riguardavano la sicurezza del sistema. Successivamente, abbiamo aggiunto un'etichetta "0" a tutti i requisiti di sicurezza che non erano legati alla privacy e un'etichetta "1" a quelli legati alla privacy.

L'utilizzo delle etichette "0" e "1" ci ha consentito di creare una suddivisione chiara tra i requisiti di sicurezza generici e quelli di privacy, permettendoci di concentrarci sulla sperimentazione di quest'ultimi.

La flessibilità offerta da pandas ci ha permesso di gestire facilmente un gran numero di entry e di ottenere un dataset pronto per la nostra sperimentazione.

Con questa operazione di estrazione dei dati e l'etichettatura dei requisiti di sicurezza e privacy, siamo ora pronti per procedere ad analizzare i dati per valutare una possibile pulizia dei dati.

3.5.2 Data cleaning

Durante questa fase, abbiamo analizzato attentamente tutti i requisiti per valutarne la correttezza sintattica, semantica e le eventuali problematiche relative alla

compatibilità con gli strumenti utilizzati nel nostro classificatore.
Per queste motivazioni sono state attuate le seguenti operazioni sui nostri dati:

- **Eliminazione dei caratteri speciali:** Ogni requisito di sicurezza è stato opportunamente pulito da caratteri speciali non necessari alla comprensione della frase.

- **Eliminazione di alcuni caratteri della punteggiatura:** Abbiamo adeguatamente rimosso ogni segno di punteggiatura che poteva interferire con la comprensione delle frasi da parte di alcuni componenti della pipeline.

- **Riformulazione di alcuni requisiti:** Alcuni requisiti di sicurezza contenevano elementi che richiedevano la loro riscrittura per garantire una corretta comprensione della frase da parte della pipeline. Ad esempio, la frase 'Alerts and notifies nurses' è stata modificata in 'Alerts nurses and notifies nurses'.

- **Aggiunta di caratteri di escape:** Durante l'applicazione degli ansatz per l'ottenimento dei circuiti, è emerso che tale funzione di parsing ha difficoltá a riconoscere alcuni segni di punteggiatura. Per risolvere questa problematica, abbiamo aggiunto i caratteri di escape sui segni di punteggiatura problematici. L'introduzione dei caratteri di escape ha consentito di ignorare il significato di tali simboli, garantendo una corretta elaborazione del testo per l'ottenimento dei circuiti.

- **Operazioni di normalizzazione dei requisiti:** Sono state adottate alcune operazioni sui requisiti in modo che il dataset abbia una struttura uniforme:

  - Aggiunta di spazi tra etichetta e requisito;
  - Rimozione di caratteri non riconosciuti nella codifica testuale;
  - Aggiunta del punto alla fine di tutti i requisiti;
  - Rimozione di apici o parentesi che racchiudono l'intero requisito;
  - Modifica del primo carattere dei requisiti ponendolo maiuscolo;

- **Esclusione delle seguenti tipologie di istanze:**

  - Istanze contenenti un numero di parole inferiore a due;

- Istanze contenenti solo URL o email;
- Istanze espresse come didascalie di immagini;

Tale pulizia dei dati ha riportato diverse rimozioni nel nostro dataset cercando di preservare la quantità di requisiti di privacy presenti. Sono stati rimossi circa 306 requisiti, di conseguenza in totale abbiamo 3737 requisiti di cui 238 di privacy.

Le operazioni eseguite sopra sono state eseguite dopo diverse analisi testuali e diverse esecuzioni della pipeline che hanno riportato diverse problematiche nell'elaborazione dei dati, tali problematiche le spiegheremo in seguito, alcune di queste ci hanno costretto a cambiare alcune componenti della pipeline quantistica e classica.

3.6 Costruzione della pipeline di machine learning

Dopo aver eseguito le dovute pulizie sui dati ci siamo concentrati sul costruire la nostra pipeline per la nostro esempio di classificazione, di seguito verranno elencate le parti in comune tra la pipeline classica e pipeline quantistica, per poi suddividere i due approcci in base alle scelte strutturali adottate. La libreria Lambeq ci ha aiutati tanto nella costruzione delle pipeline, siccome le funzionalitá offerte ci ha permesso di fare un analisi accurata delle limitazioni e delle criticitá degli strumenti utilizzati.

Figura 3.2: Un esempio di costruzione di una pipeline su Lambeq

3.6.1 Tokenizzazione

La tokenizzazione consiste nella suddivisione di ogni nostro requisito in diverse parti dette token. Ci siamo limitati a suddividere parola per parola senza considerare i segni di punteggiatura presenti in ogni frase.

In questo esempio, la tokenizzazione è fornita attraverso la gerarchia di classi Tokeniser e specificamente utilizzando la classe SpacyTokeniser, basata sulla popolare libreria NLP *spaCy*. L'output che otteniamo da questo codice è la lista dei token

```python
from lambeq import SpacyTokeniser

tokeniser = SpacyTokeniser()
sentence = "This sentence is a privacy requirement."
tokens = tokeniser.tokenise_sentence(sentence)

#Output: ['This', 'sentence', 'is', 'a', 'privacy', 'requirement']
```

Figura 3.3: Esempio di codice Python sull'utilizzo di SpacyTokeniser.

ottenuti da ogni requisito.

3.6.2 Parsing in diagrammi di stringhe

Dopo la fase di tokenizzazione, successivamente tutte le frasi tokenizzate sono state trasformate in diagrammi di stringhe usando le funzionalità di Lambeq per il parsing di stringhe a diagrammi. Questa è una fase importante siccome costituisce una rappresentazione astratta ideale delle relazioni tra le parole in una frase. Durante la nostra sperimentazione abbiamo verificato come diverse tipologie di parser possano interagire in modo diverso con la nostra pipeline, per questa motivazione bisogna prima specificare due tipologie di parser:

1. **Syntax-based model:** un modello basato sulla sintassi cerca di catturare le relazioni strutturali e gerarchiche all'interno delle frasi, il modello utilizzato da Lambeq è quello di **BobcatParser**.

```python
from lambeq import BobcatParser

parser = BobcatParser(verbose='text')

train_diagrams = parser.sentences2diagrams(train_data)
val_diagrams = parser.sentences2diagrams(val_data)
```

Figura 3.4: Esempio di codice Python sull'utilizzo di BobcatParser.

2. **Word-sequence models:** un modello basato sulla sequenza di parole trattano il testo di una frase come una sequenza lineare di token, senza considerare la struttura sintattica o grammaticale, i modelli che Lambeq propone sono **CupsReader** e **StairsReader**, che si differenziano tra loro soltanto nella struttura logica dei diagrammi.

```
from lambeq import cups_reader
from lambeq import stairs_reader

cups_diagram = cups_reader.sentence2diagram(sentence)
stairs_diagram = stairs_reader.sentence2diagram(sentence)
```

Figura 3.5: Esempio di codice Python sull'utilizzo di CupsReader e StairsReader.

Le differenze strutturali nei diagrammi generati dalle librerie BobcatParser, CupsReader e StairsReader riflettono distinti approcci nella rappresentazione sintattica delle frasi. Mentre i diagrammi di BobcatParser incorporano relazioni sintattiche precise tra i token, i diagrammi prodotti da CupsReader e StairsReader risultano essere più semplici e lineari, basandosi esclusivamente sull'ordine posizionale delle parole all'interno di una frase.

Tuttavia, l'approccio di BobcatParser ha presentato sfide significative in termini di complessità strutturale. La rilevazione di diverse interpretazioni sintattiche per una stessa frase ha portato alla generazione multipla di diagrammi, generando conseguenti difficoltà durante la fase di addestramento del modello. Per risolvere tale problematica, ci siamo rivolti all'uso dei StairsReader. Quest'ultimo ha dimostrato la capacità di produrre un unico diagramma per ciascuna frase in input, semplificando notevolmente la successiva fase di addestramento.

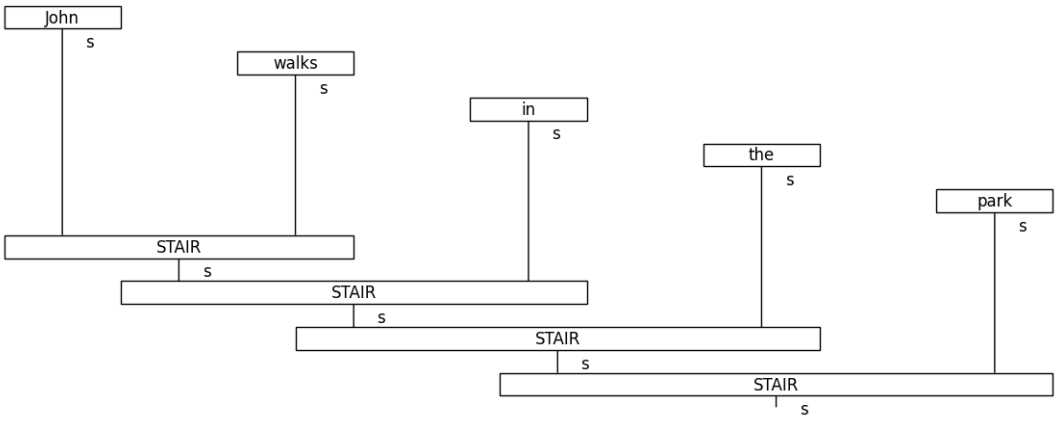

Figura 3.6: Esempio di diagramma di stringhe dato in output da uno StairsReader

È importante sottolineare che queste considerazioni sono state applicate sia al contesto classico che a quello quantistico. L'obiettivo è stato quello di evitare problemi analoghi a quelli precedentemente riscontrati e di utilizzare componenti omogenei per entrambi i contesti. Questo approccio ha inoltre consentito un confronto più accurato del comportamento dei modelli nelle diverse situazioni.

Da questo momento descriveremo le successive fasi suddividendole in approccio classico e approccio quantistico, cioè in base alle differenti scelte fatte sia per la pipeline classica che per la pipeline quantistica.

3.6.3 Approccio classico

Parsing in circuiti

Come i diagrammi di stringhe anche i circuiti hanno un ruolo fondamentale nella rappresentazione morfosintattica delle parole in una frase, ma in particolare un circuito quantistico vuole descrivere e modellare processi e calcoli che coinvolgono le proprietà della meccanica quantistica.

Un circuito quantistico è composto da una serie di qubit e gate quantistici che rappresentano le operazioni eseguite sul sistema quantistico, per ottenere i nostri

circuiti applichiamo ai nostri diagrammi di stringhe un ansatz.

Un ansatz definisce partendo da un diagramma un'assunzione sulla forma funzionale di uno stato quantistico per ogni parola della frase e assegnando un numero di qubit a cui è associato ogni filo del diagramma di stringhe.

Nel caso di una pipeline classica, le scelte che abbiamo adottato dipendono dalla semplicità costruttiva dei nostri diagrammi come tensori e alla risoluzione di problemi di calcolo che non dipendono da alcun limite di potenza computazionale, per questo ci siamo avvalsi della facoltà di utilizzare come ansatz lo **SpiderAnsatz**. Quest'ultimo permette la creazione di circuiti quantistici, basati sul concetto di 'spiders', che hanno una struttura regolare e facilmente comprensibile da qualsiasi macchina.

```
from lambeq import SpiderAnsatz
from discopy.tensor import Dim

spider_ansatz = SpiderAnsatz({N: Dim(4), S: Dim(2)})
spider_diagram = spider_ansatz(diagram)
```

Figura 3.7: Esempio di codice Python sull'utilizzo di SpiderAnsatz.

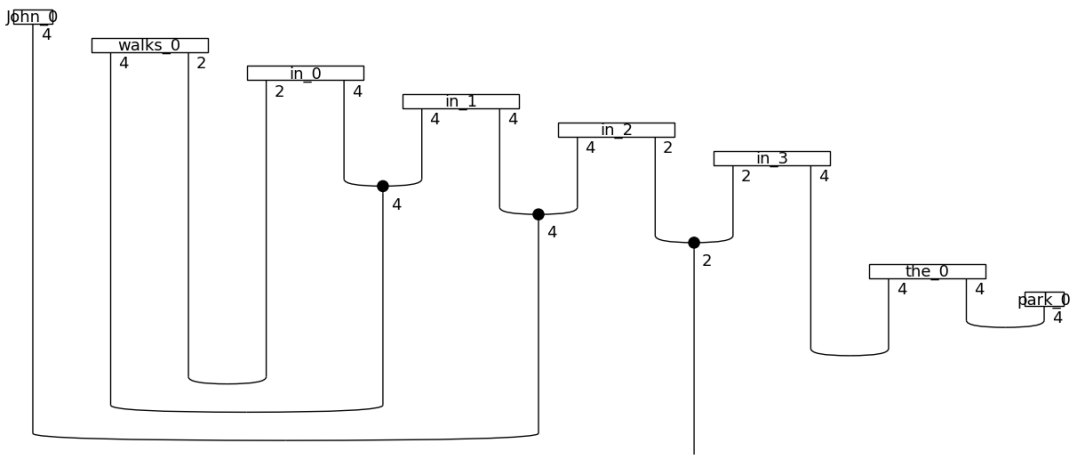

Figura 3.8: Esempio di un circuito dato in output da uno SpiderAnsatz

Scelta del modello di machine learning

In questa fase ci poniamo davanti il problema che riguarda che modello utilizzare per la nostra classificazione binaria e allo stesso tempo come allenare tale modello. Come precedentemente specificato la scelta del modello è fortemente influenzata dal caso che vogliamo testare sul nostro dataset.

Per la scelta del modello della nostra pipeline classica abbiamo scelto di seguire i suggerimenti della libreria Lambeq che utilizza come modello PytorchModel proposta dalla libreria PyTorch per l'apprendimento automatico per il machine learning e deep learning. PyTorch utilizza per le sue operazioni i tensori che sono alla base di tutte le operazioni di calcolo della libreria e permette di utilizzarli sia su CPU che su GPU per accelerare tali calcoli.

```
from lambeq import PytorchModel

all_circuits = train_circuits + val_circuits + test_circuits
model = PytorchModel.from_diagrams(all_circuits)
```

Figura 3.9: Esempio di codice Python sull'utilizzo di PytorchModel

Per creare un PytorchModel, si passano in input i circuiti precedentemente creati con l'ansatz. Questi circuiti sono rappresentati come tensori di PyTorch, che vengono utilizzati dal modello per imparare le relazioni tra i circuiti. Il modello viene poi utilizzato per fare predizioni sui circuiti dati in input.

Per allenare il modello, abbiamo utilizzato il PytorchTrainer di PyTorch. Questo trainer semplifica l'addestramento dei modelli, offrendo un'interfaccia semplice e intuitiva. Il PytorchTrainer consente di scegliere le componenti necessarie per l'addestramento, senza concentrarsi sui dettagli tecnici.

```
from lambeq import PytorchTrainer

trainer = PytorchTrainer(
        model=model,
        loss_function=torch.nn.BCEWithLogitsLoss(),
        optimizer=torch.optim.AdamW,
        learning_rate=0.03,
        epochs=10,
        evaluate_functions={'acc': acc, 'precision': prec, 'recall': rec,
        'f1': f1},
        evaluate_on_train=True,
        verbose='text',
        seed=0)
```

Figura 3.10: Esempio di codice Python sull'utilizzo di PytorchTrainer

Il frammento di codice presentato illustra l'utilizzo del PytorchTrainer per l'addestramento di un modello. Questo trainer offre un controllo dettagliato sul processo di addestramento, comprese le metriche di valutazione, la scelta della funzione di perdita e altri parametri rilevanti.

Nel codice, l'istanza di PytorchTrainer fornisce i seguenti parametri:

- **model:** il modello che intendiamo addestrare.
- **loss_function:** la funzione di perdita utilizzata durante l'addestramento. In questo caso, viene utilizzata la 'BCEWithLogitsLoss', tale scelta è legata ai suggerimenti della libreria Lambeq.
- **optimizer:** l'ottimizzatore utilizzato per aggiornare i pesi del modello. Utilizziamo l'optimazer di torch AdamW.
- **learning_rate:** il tasso di apprendimento impostato a 0.03, che regola l'ampiezza degli aggiornamenti dei pesi.
- **epochs:** il numero di iterazioni per l'addestramento, fissato a 10 per risparmiare risorse di calcolo e per ridurre il tempo di addestramento.
- **evaluate_functions:** la lista di metriche di valutazione da calcolare durante l'addestramento, tra cui l'accuratezza, la precisione, la recall e l'F1-score.

- **evaluate_on_train:** la scelta di valutare le metriche anche sul set di addestramento.
- **verbose:** il parametro "text" imposta la modalità di output durante l'addestramento su testo, fornendo informazioni dettagliate sull'andamento.
- **seed:** il seed utilizzato per la riproducibilità dei risultati.

3.6.4 Approccio quantistico

Parsing in circuiti

Nel caso della pipeline quantistica invece, le scelte adottate dipendono da come vogliamo simulare una esecuzione quantistica e su come vogliamo che il nostro modello funzioni su un'ipotetica macchina quantistica, per questo motivo l'uso di IQPAnsatz è stato cruciale per la creazione dei nostri circuiti quantistici.

I circuiti generato da IQPAnsatz sono molto piú complessi rispendo ad altri ansatz siccome sono portati per essere elaborati da una macchina quantistica e difficili da simulare su una macchina che segue meccanismi classici.

```
from lambeq import AtomicType, IQPAnsatz

N = AtomicType.NOUN
S = AtomicType.SENTENCE

ansatz = IQPAnsatz({N: 1, S: 1}, n_layers=2)
discopy_circuit = ansatz(diagram)
```

Figura 3.11: Esempio di codice Python sull'utilizzo di IQPAnsatz.

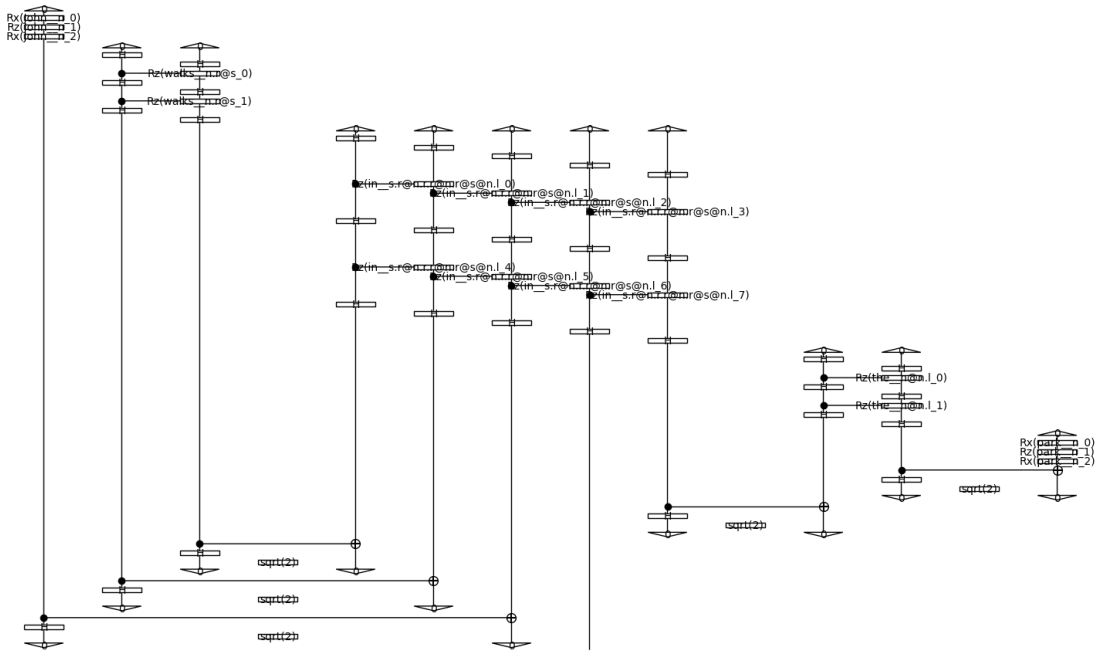

Figura 3.12: Esempio di circuito dato in output da uno IQPAnsatz

Scelta del modello di machine learning

Come lo è stato per la pipeline classica, anche per la scelta del modello della nostra pipeline quantistica abbiamo scelto di seguire i suggerimenti della libreria Lambeq che utilizza come modello NumpyModel proposta dalla libreria NumPy per l'apprendimento automatico per il machine learning e deep learning, grazie all'utilizzo di simulatori di circuiti quantistici converte i circuiti quantistici in una rete tensoriale.

```
from lambeq import NumpyModel

all_circuits = train_circuits + val_circuits

model = NumpyModel.from_diagrams(all_circuits, use_jit=False)
```

Figura 3.13: Esempio di codice Python sull'utilizzo di NumpyModel

Il modello descritto nel codice oltre a prendere in input la lista di circuiti quan-

tistici, prende in input un flag **use_jit** che se messo a *True* può essere utilizzato per abilitare la compilazione **Just-In-Time**, questo permette di migliorare le prestazioni del codice Python, ma è inefficiente in termini di tempo e risorse di calcolo richieste, per questo motivo abbiamo scelto di tenerlo a *False*.

Per allenare il modello, abbiamo utilizzato il QuantumTrainer utilizzato per l'addestramento di modelli di apprendimento quantistico. Obiettivo di questo trainer è fornire un'interfaccia semplice da configurare e che permetta la sperimentazione dell'apprendimento quantistico ad un pubblico piú ampio.

```
from lambeq import QuantumTrainer, SPSAOptimizer

trainer = QuantumTrainer(
    model,
    loss_function=torch.nn.BCEWithLogitsLoss(),
    epochs=10,
    optimizer=SPSAOptimizer,
    optim_hyperparams={'a': 0.2, 'c': 0.06, 'A':0.01*10},
    evaluate_functions={'acc': acc, 'precision': prec, 'recall': rec,
    'f1': f1},
    evaluate_on_train=True,
    verbose = 'text',
    seed=0
)
```

Figura 3.14: Esempio di codice Python sull'utilizzo di QuantumTrainer

Il frammento di codice presentato illustra l'utilizzo del QuantumTrainer per l'addestramento di un modello. Questo trainer offre gli stessi parametri che abbiamo già visto per il PytorchTrainer, ma in piú permette la modifica degli iperparametri del nostro ottimizzatore.

Nel codice, l'istanza di QuantumTrainer fornisce i seguenti parametri:

- **model:** il modello che intendiamo addestrare.
- **loss_function:** la funzione di perdita utilizzata durante l'addestramento. In questo caso, viene utilizzata la 'BCEWithLogitsLoss', tale scelta è legata ai

suggerimenti della libreria Lambeq.

- **optimizer:** l'ottimizzatore utilizzato per aggiornare i pesi del modello. In questo caso abbiamo utilizzato un optimizer fornito da Lambeq SPSAOptimizer.
- **optim_hyperparams:** questi parametri controllano il comportamento dell'ottimizzatore e possono influire sulle prestazioni del modello.
- **epochs:** il numero di iterazioni per l'addestramento, fissato a 10 per risparmiare risorse di calcolo e per ridurre il tempo di addestramento.
- **evaluate_functions:** la lista di metriche di valutazione da calcolare durante l'addestramento, tra cui l'accuratezza, la precisione, la recall e l'F1-score.
- **evaluate_on_train:** la scelta di valutare le metriche anche sul set di addestramento.
- **verbose:** il parametro "text" imposta la modalità di output durante l'addestramento su testo, fornendo informazioni dettagliate sull'andamento.
- **seed:** il seed utilizzato per la riproducibilità dei risultati.

3.7 Tecniche di validazione

La valutazione delle prestazioni dei due classificatori è stata condotta attraverso l'impiego di una **K-fold cross validation**, sia per il classificatore binario classico che per il classificatore binario quantistico.

La K-fold cross validation rappresenta una metodologia fondamentale per l'analisi dei modelli, che implica la suddivisione del dataset originale in K sottoinsiemi distinti, noti come **"fold"** ognuno avente la medesima dimensione. In ogni iterazione di questa procedura, uno di tali sottoinsiemi, designato come "insieme di validazione," è destinato a valutare le performance del modello, mentre gli altri K-1 sottoinsiemi sono impiegati per l'addestramento dello stesso modello. Questo ciclo di valutazione e addestramento viene iterato K volte, assicurando che ciascun sottoinsieme funga da insieme di validazione almeno una volta.

Al fine di questa valutazione specifica, è stata scelta una configurazione di **5-fold cross validation**. Per ogni fold, sono state calcolate e registrate le metriche di valutazione delle prestazioni del modello precedentemente selezionate. Inoltre, i risultati ottenuti sono stati rappresentati graficamente, permettendo così una comprensione visiva delle differenze sostanziali tra i vari fold dei due classificatori presi in esame.

Questo approccio non solo consente di ottenere una stima affidabile delle prestazioni dei modelli, ma rende anche possibile rilevare eventuali variazioni significative tra le diverse iterazioni della cross validation. La rappresentazione grafica dei risultati contribuisce a cogliere in modo immediato le divergenze tra i classificatori in termini di performance su ciascun fold. Tale analisi fornisce un'ulteriore profondità di comprensione e costituisce una base solida per l'interpretazione dei risultati e le conclusioni tratte in seguito.

3.8 Metriche di valutazione

Le metriche che abbiamo scelto di valutare le prestazioni sono quelle di *accuracy, precision, recall e F1-score*. Per il calcolo dei valori di queste metriche abbiamo innanzitutto diviso le nostre predizioni in quattro tipologie importanti:

1. **True Positive (TP):** descrivono le previsioni che sono positive corrette;
2. **False Positive (FP):** descrivono le previsioni che sono positive errate;
3. **True Negative (TN):** descrivono le previsioni che sono negative corrette;
4. **False Negative (FN):** descrivono le previsioni che sono negative errate;

Chiariti e calcolati questi valori, possiamo definire l'**accuratezza** come il rapporto tra le predizioni corrette e il totale delle previsioni effettuate dal modello:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

La **precisione** è definita come il rapporto tra il numero delle previsioni positive corrette (TP) e il totale delle previsioni positive del modello:

$$Precision = \frac{TP}{TP + FP}$$

La **recall** è definita come il rapporto tra le previsioni positive corrette e il totale dei casi in cui si verifica realmente:

$$Recall = \frac{TP}{TP + FN}$$

La **F1-score** viene calcolata tramite la media armonica di precisione e recall:

$$F1 - score = 2 \times \frac{P \times R}{P + R}$$

CAPITOLO 4

Risultati

4.1 Risultati a confronto

Come spiegato in precedenza, i risultati che abbiamo ottenuto dalla nostra sperimentazione sono stati estratti dopo aver eseguito come tecnica di validazione la **5-fold cross validation**, in questa sezione metteremo a confronto singolarmente i risultati dei fold della pipeline classica con i risultati dei fold della pipeline quantistica. I risultati inseriti verranno poi raggruppati e verranno fatte delle considerazioni su come un approccio puó essere vantaggioso rispetto all'altro sia per quanto riguarda la qualitá dei risultati che sulla quantitá di risorse di calcolo occupate.

4.1.1 Primo fold

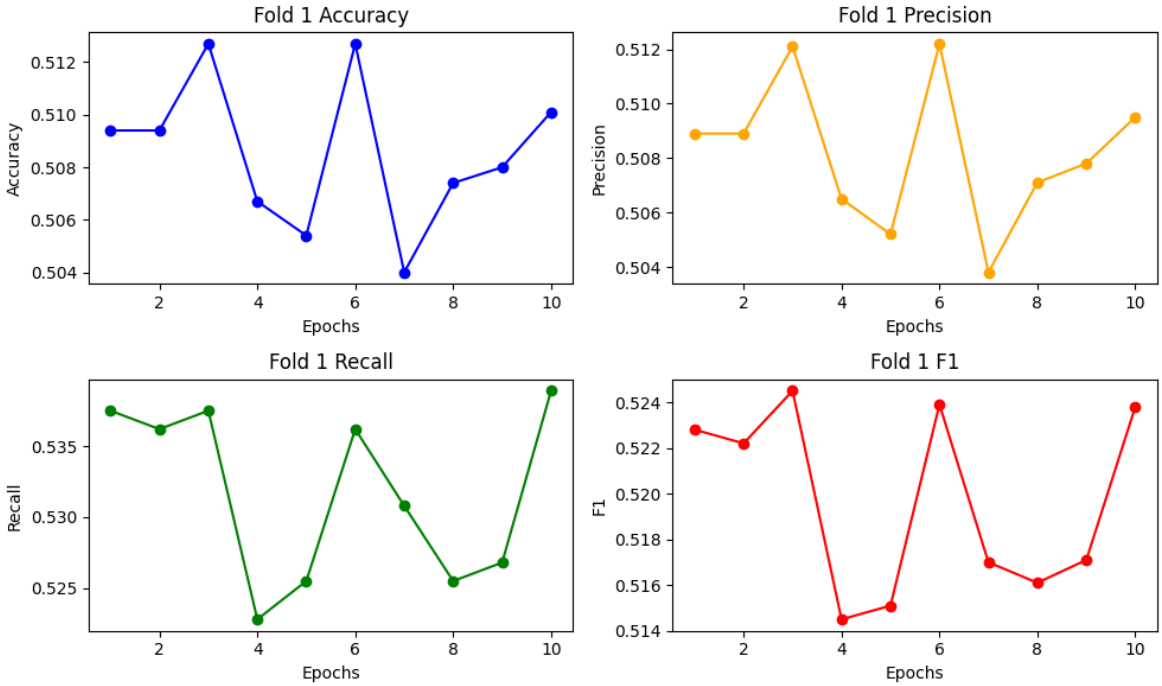

Figura 4.1: Risultati del primo fold per la pipeline classica

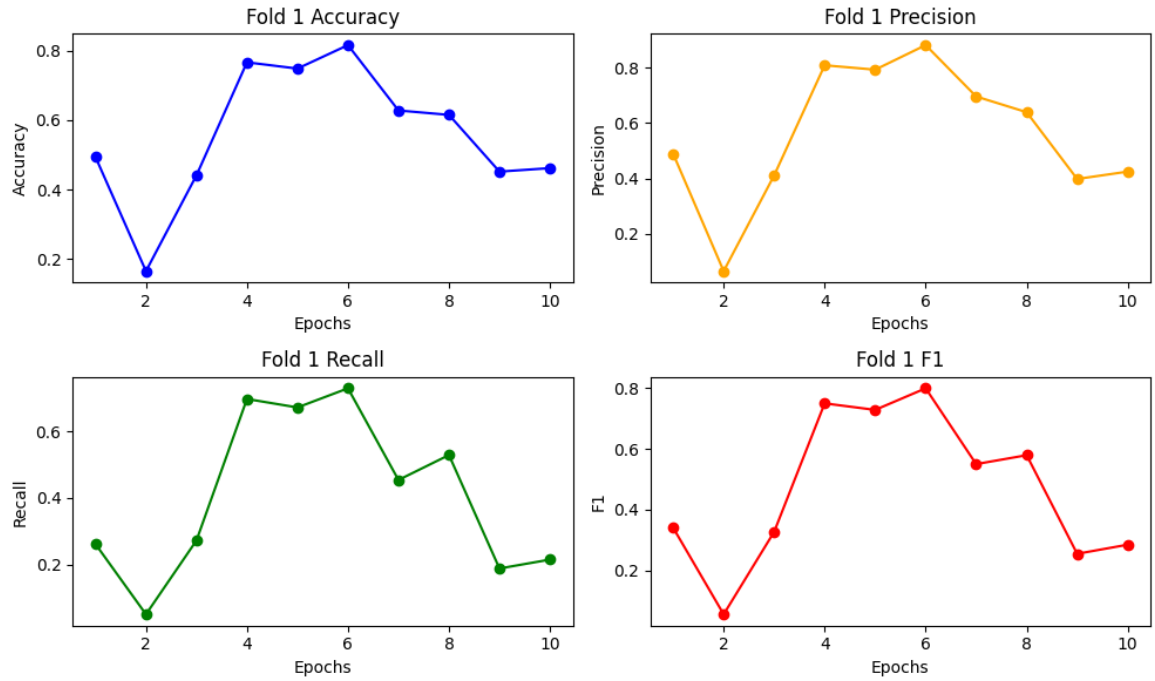

Figura 4.2: Risultati del primo fold per la pipeline quantistica

Discutendo del primo fold, nella **Figura 4.1** possiamo osservare come nel caso classico ci sia una crescita netta dei risultati per ogni metrica, in particolare nella **Recall** che ha superato il 53% piú velocemente rispetto alle altre metriche. In generale questo fold risulta il miglior risultato per quanto riguarda il caso classico.
Per quanto riguarda il caso quantistico nella **Figura 4.2** si vede come la crescita é stata molto piú lenta e in alcuni casi c'é stato un calo drastico dei risultati, l'andamento peggiore l'hanno avuto la **Recall** e la **F1-score**, mentre il miglior risultato l'ha avuta la **Precision** superando il 50%.

4.1.2 Secondo fold

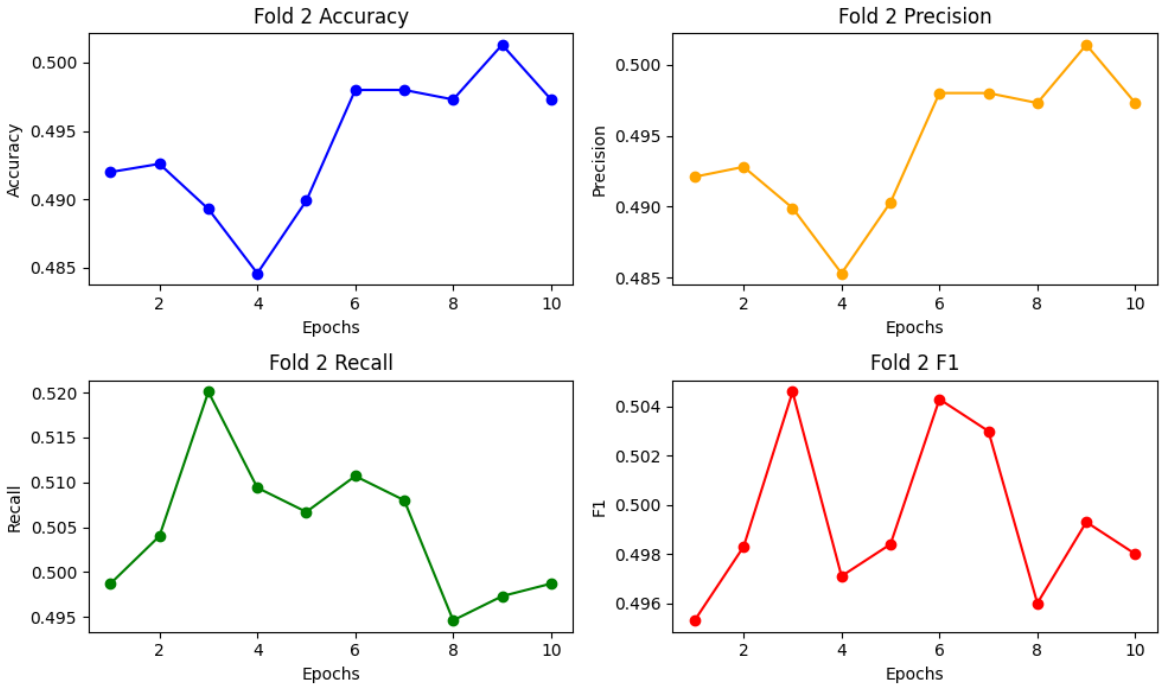

Figura 4.3: Risultati del secondo fold per la pipeline classica

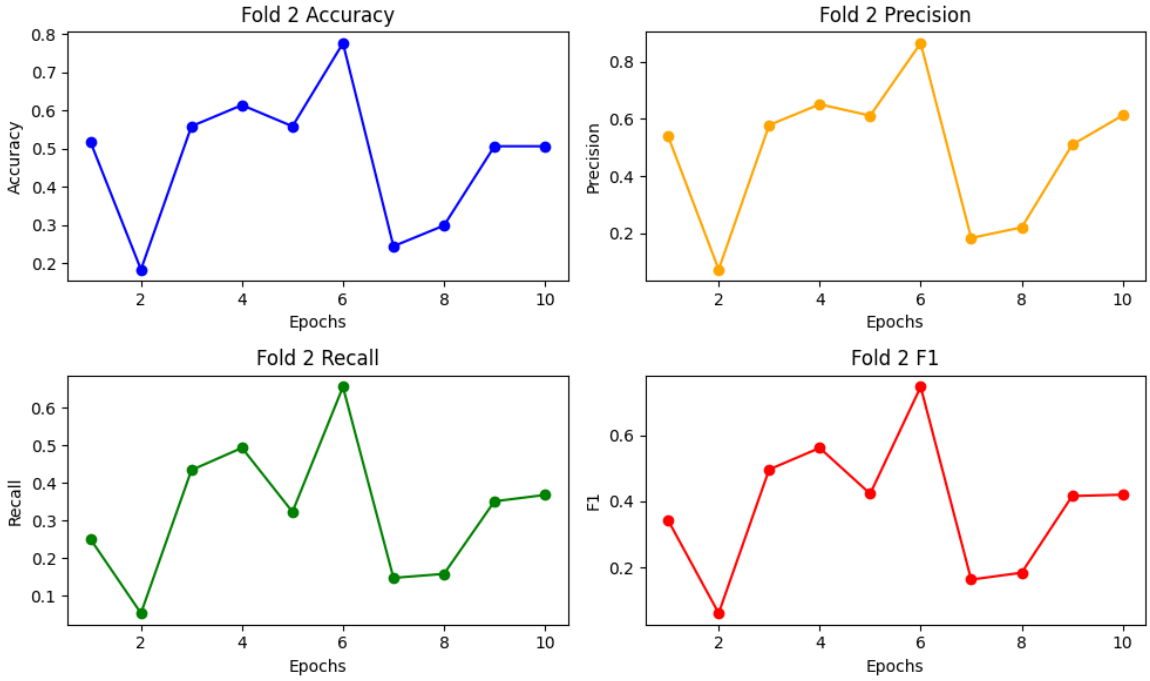

Figura 4.4: Risultati del secondo fold per la pipeline quantistica

Per quanto riguarda il secondo fold, nella **Figura 4.3** possiamo osservare come nel caso classico ci sia una leggero decremento dei risultati rispetto al primo fold, nonostante questo la **Recall** risulta il risultato migliore con il 50%.
Il caso quantistico nella **Figura 4.4** ha riscontrato risultati migliori rispetto al fold precendente, nonostante ci siano comunque stati dei cali drastici durante il training, il miglior risultato lo ha ottenuto la **Precision** con il 60%.

4.1.3 Terzo Fold

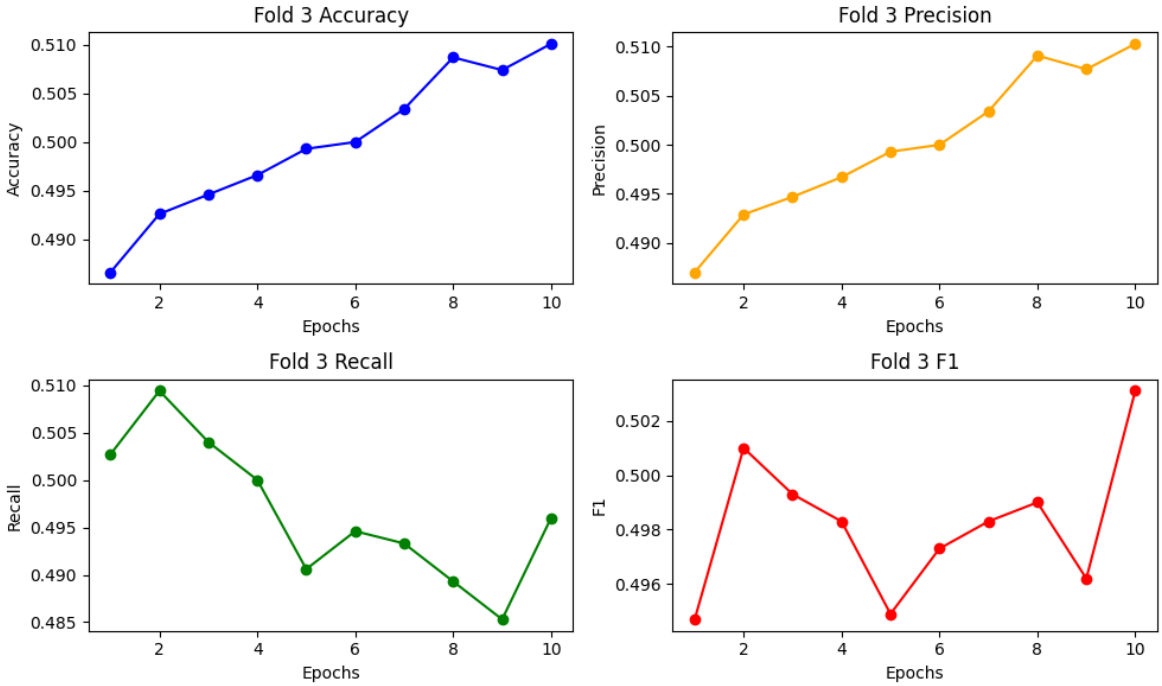

Figura 4.5: Risultati del terzo fold per la pipeline classica

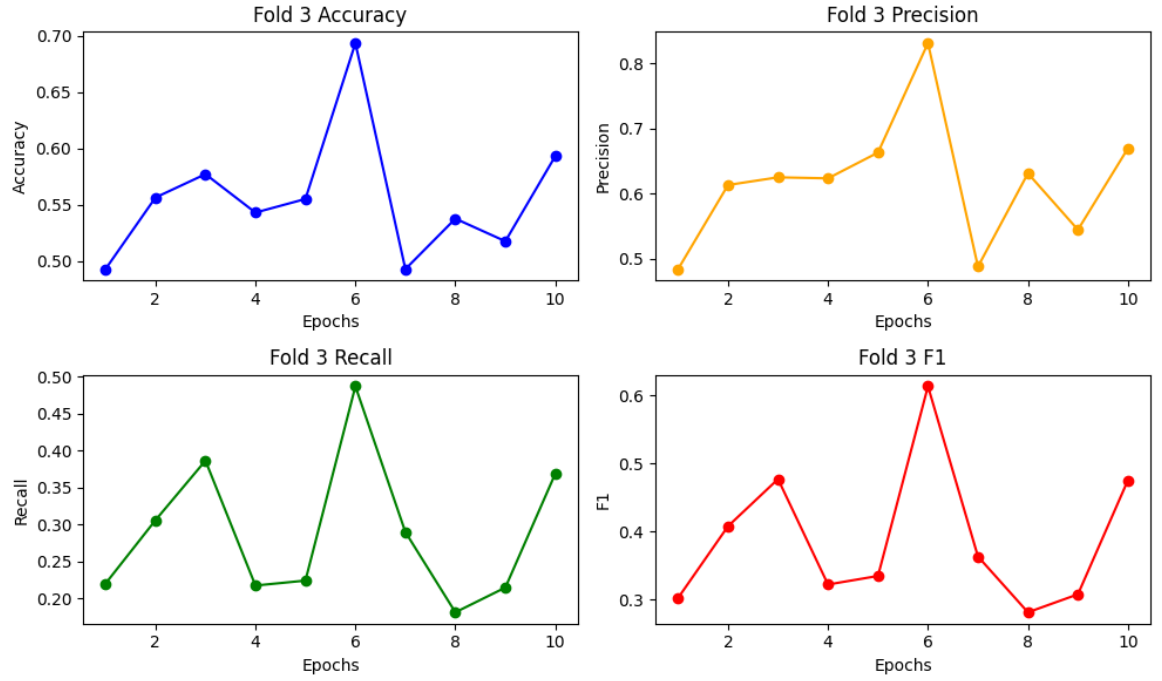

Figura 4.6: Risultati del terzo fold per la pipeline quantistica

Il terzo fold nel caso classico nella **Figura 4.5** si presenta con un leggero incremento rispetto al secondo fold e stavolta come risultati predominanti ci sono l'**Accuracy** e la **Precision** con il 51%. Nel caso quantistico nella **Figura 4.6** c'é stato un notevole incremento dei risultati per tutte le metriche ricevendo il miglior risultato tra tutti i fold e con ben 70% per la **Precision**.

4.1.4 Quarto Fold

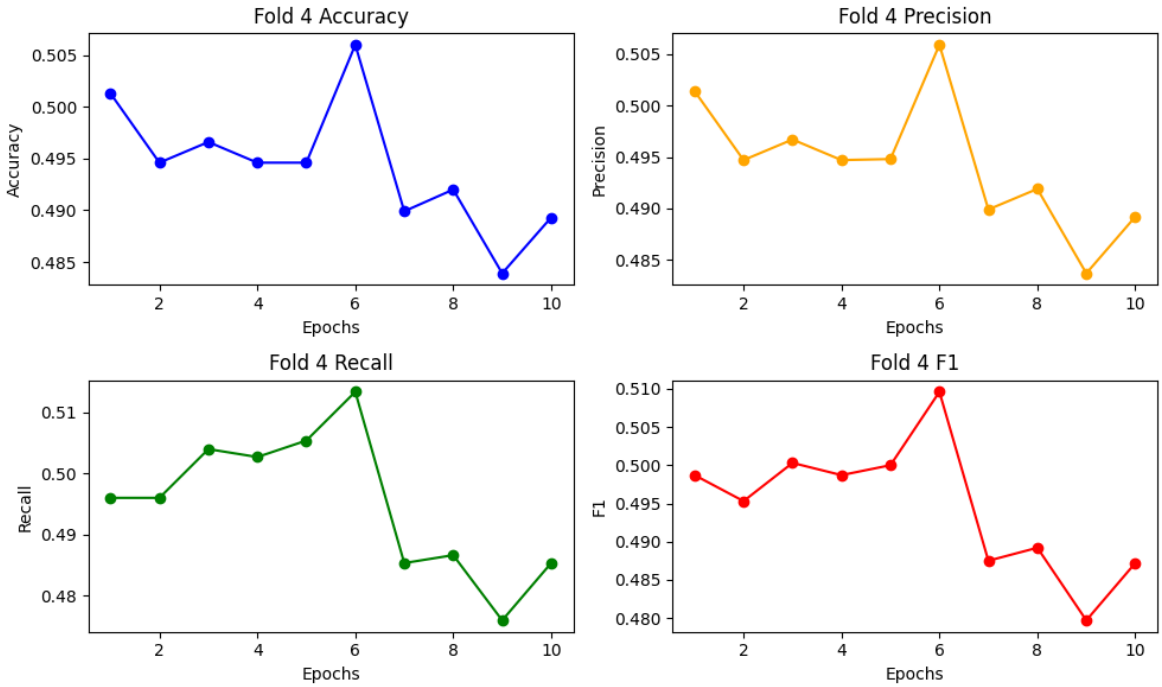

Figura 4.7: Risultati del quarto fold per la pipeline classica

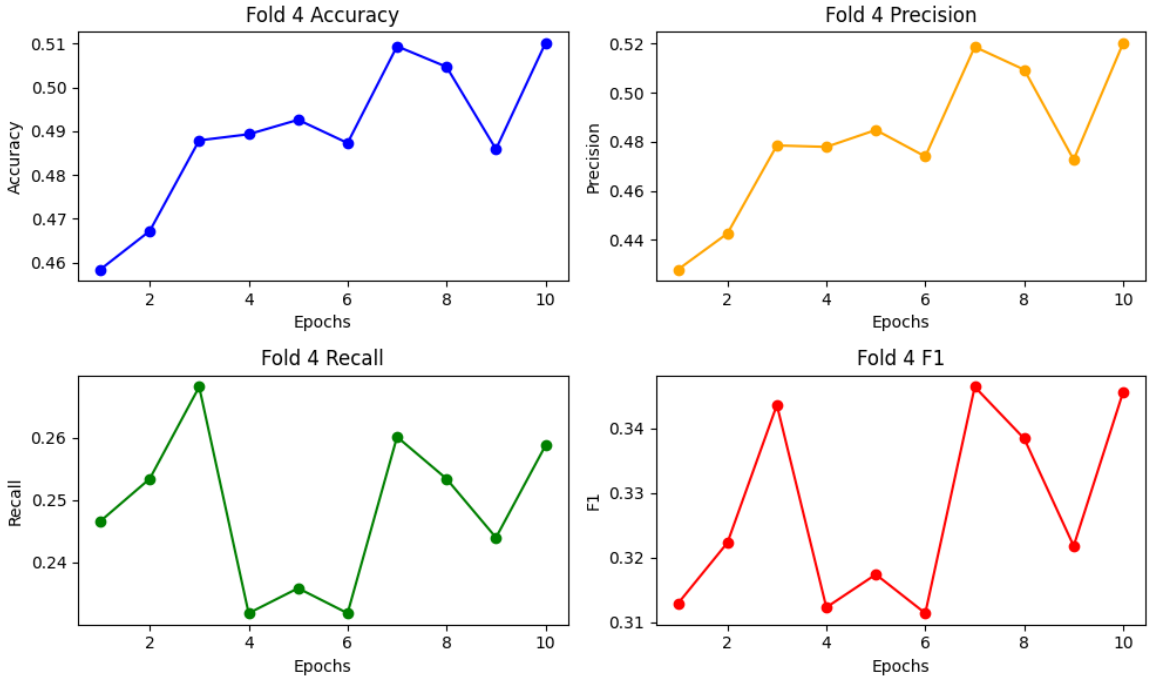

Figura 4.8: Risultati del quarto fold per la pipeline quantistica

Il quarto fold nella **Figura 4.7** presenta nuovamente un leggero decremento dei risultati portando per il caso classico ad avere per tutte le metriche registrate un 49%. In questo fold il caso quantistico nella **Figura 4.8** ha nuovamente un notevole decremento dei risultati per ogni metrica, **Recall** e **F1-score** risultano molto bassi con rispettivamente il 26% e il 35%, risultato migliore lo ha avuto anche stavolta la **Precision** con il 52%.

4.1.5 Quinto Fold

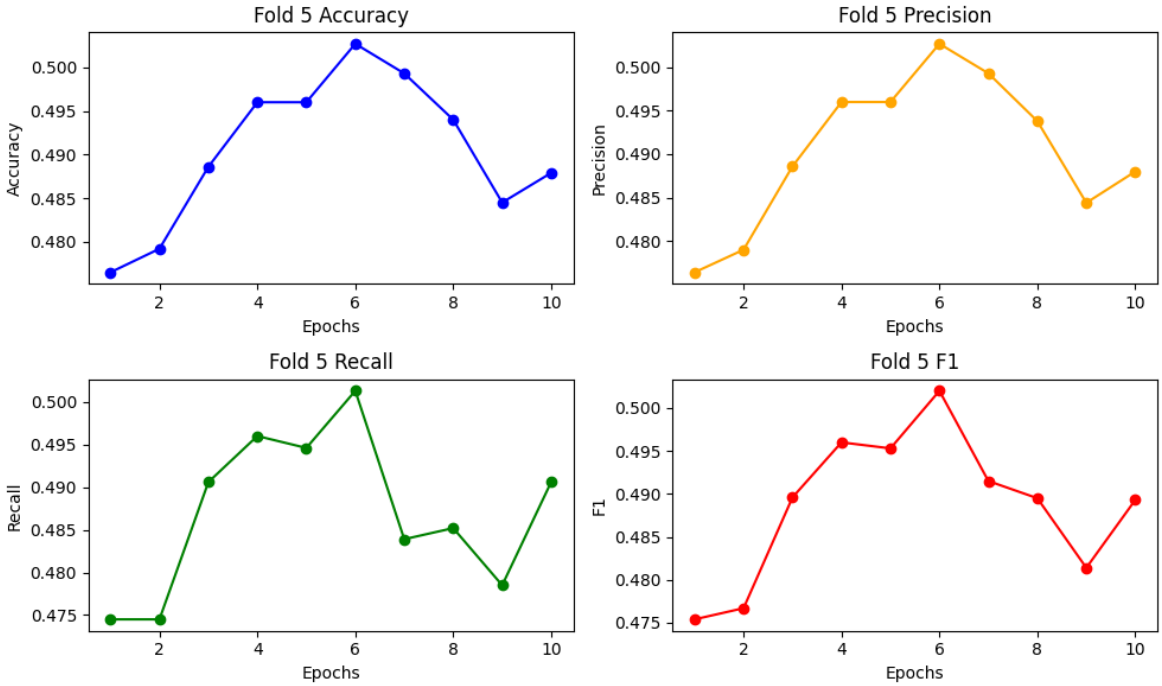

Figura 4.9: Risultati del quinto fold per la pipeline classica

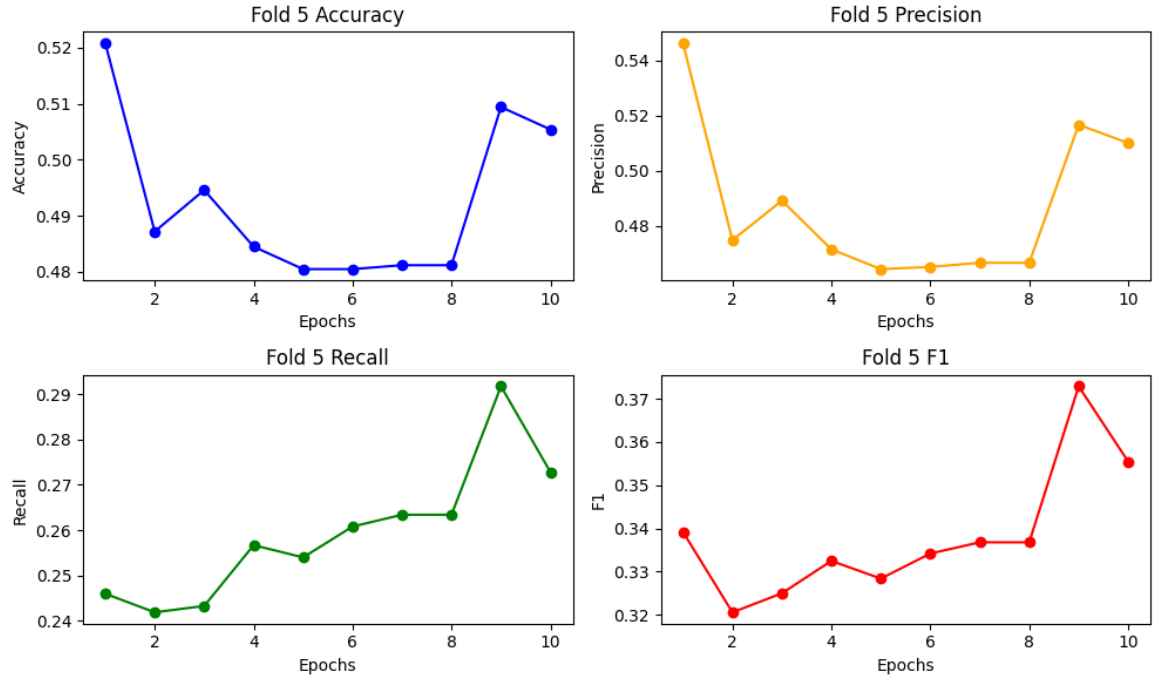

Figura 4.10: Risultati del quinto fold per la pipeline quantistica

Per il quinto fold sia per l'approccio classico **Figura 4.9** che per l'approccio quantistico **Figura 4.10** i risultati sono rimasti quasi invariati rispetto al quarto fold. Anche qui il risultato migliore é stata la **Recall** per il caso classico con il 49%, lo stesso per la **Precision** con il 52% é il miglior risultato del caso quantistico nel quinto fold.

4.2 Considerazioni sui risultati

Fold	Accuracy	Precision	Recall	F1-Score
1	0.51	0.51	0.54	0.52
2	0.49	0.49	0.50	0.49
3	0.51	0.51	0.49	0.50
4	0.49	0.49	0.49	0.49
5	0.48	0.48	0.49	0.49
Media	0.496	0.496	0.502	0.498

Tabella 4.1: Risultati dei fold nel caso classico

Fold	Accuracy	Precision	Recall	F1-Score
1	0.50	0.50	0.20	0.30
2	0.50	0.60	0.40	0.50
3	0.60	0.70	0.40	0.50
4	0.51	0.52	0.26	0.35
5	0.51	0.52	0.27	0.36
Media	0.524	0.568	0.306	0.402

Tabella 4.2: Risultati dei fold nel caso quantistico

Osservando i risultati riportati nella tabella 4.1 e nella tabella 4.2 possiamo rispondere alle domande di ricerca che ci siamo posti nell'obiettivo della sperimentazione:

👉 **Answer to RQ_1.** La pipeline classica, considerando la media dei risultati, ha avuto risultati migliori sulla Recall con il 50% nell'identificazione dei requisiti di privacy

Analizzando i risultati dei fold dell'approccio classico 4.1 risulta che il picco piú alto é stato rivelato nel fold 1 con una Recall del circa 54%, tale risultato ha confermato ció che riporta la media dei risultati ottenendo migliori prestazioni medie sulla Recall.

👉 **Answer to RQ_2.** La pipeline quantistica, considerando la media dei risultati, ha avuto risultati migliori sulla Precision con il 56% nell'identificazione dei requisiti di privacy

Analizzando i risultati dei fold dell'approccio classico 4.2 risulta che il picco piú alto é stato rivelato nel fold 3 con una Precision del circa 70%, tale risultato ha confermato ció che riporta la media dei risultati ottenendo migliori prestazioni medie sulla Precision.

👉 **Answer to RQ_3.** La pipeline quantistica si differenzia dalla pipeline classica nell'aver avuto mediamente risultati migliori sulla Precision e sull'Accuracy nell'identificare i requisiti di privacy.

Confrontando ogni risultato di singolo fold tra le pipeline rispetto alle risultati medi delle metriche, l'approccio quantistico ha risultati nettamente piú alti sulla Precision e sull'Accuracy rispetto all'approccio classico.

Bisogna precisare che ai risultati ottenuti da entrambi gli approcci bisogna associare le tempistiche e le risorse di calcolo occupate dall'esecuzione delle pipeline. La pipeline classica nella sua esecuzione ha occupato in media 14 GB di RAM e ha impiegato circa 30 minuti per ciascun fold. D'altro canto la pipeline quantistica nella sua esecuzione ha impiegato ben 3 ore e 30 minuti per ciascun fold e occupato in media 11 GB di RAM. Quindi oltre a considerare i risultati bisognerebbe associare questi dati per fare delle considerazioni piú precise su quale approccio sia migliore,

inoltre ricordiamo che stiamo sperimentando su un dataset di 4032 entry e con solo 10 epoche impostate per fold, che é un dettaglio da non escludere.

Come potevamo aspettarci da altri lavori osservati i risultati dell'approccio quantistico 4.2 sono mediamente migliori rispetto ai risultati ottenuti dall'approccio classico 4.1, nonostante questo i risultati quantistici predominano su **Accuracy** e **Precision**, ma non sul resto. Infatti, i risultati classici riportano come **Recall** e **F1-score** siano in media piú alte rispetto al caso quantistico. Risultati del genere peró sono accompagnati da una mole di calcolo ed una quantitá di tempo che per sole 10 epoche non é plausibile, soprattutto se effettuiamo tale apprendimento quantistico su un hardware poco potente e abbiamo bisogno di risultati in tempi brevi, in questo caso sarebbe meglio utilizzare l'approccio classico che permette di avere risultati accettabili in tempi brevi con poche risorse di calcolo.

CAPITOLO 5

Conclusioni e sviluppi futuri

Al giorno d'oggi l'avanzamento tecnologico é in rapida evoluzione, quelli che una volta erano sistemi solidi ed efficienti oggi sono sovrastati da nuovi sistemi molto piú veloci ed efficienti. Sistemi che sfruttano la meccanica classica come l'apprendimento classico man mano stanno lasciando il posto a modelli basati sulla meccanica quantistica per sfruttarne la velocitá e l'efficienza. Per questo motivo bisogna effettuare un lungo studio dei casi d'uso dell'apprendimento quantistico, testando questi sistemi su vari parametri e dataset per poi analizzare i risultati per verificare l'effettiva utilitá di questi modelli quantistici.

Il lavoro svolto in questa tesi vuole fare proprio questo, sviluppare un classificatore per i requisiti di privacy che possa essere un esempio di caso d'uso per testare le performance di un modello di machine learning classico e un modello quantistico. É stato utilizzato un dataset relativo all'healthcare analizzando ogni requisito per verificare la sua struttura e testare gli strumenti di parsing utilizzati, facendo ulteriori considerazioni su di essi. Abbiamo infine validato entrambi i classificatori eseguendo una 5-fold cross validation, ottenendo una recall del 50% per l'approccio classico e una precision che varia tra il 50% e il 70% per l'approccio quantistico.

Una sperimentazione del genere sicuramente non é completa, ma sicuramente ha permesso di fare numerose argomentazioni sui vantaggi e svantaggi di entrambi

gli approcci, infatti un lavoro del genere puó essere considerato come un punto di partenza su cui altri ricercatori potranno basarsi per lavori futuri.
Per sviluppi futuri possiamo descrivere diversi possibili miglioramenti al nostro lavoro:

- Utilizzare un dataset associato ad un utilizzo piú generico e essendo che abbiamo effettuato una classificazione binaria, bilanciare il numero delle categorie presenti nel dataset.
- Validare il classificatore su dispositivi con specifiche tecniche e risorse di calcolo adeguate per migliorare le tempistiche ed evitare sovraccarichi di memoria non richiesti.
- Utilizzare simulatori quantistici differenti che possono permettere di avere una panoramica dei risultati piú ampia tra le libreria quantistiche presenti attualmente.
- Cambiare i parametri e gli optimizer utilizzati nella sperimentazione e testarne di nuovi per confrontarne le performance.

Ringraziamenti

Vorrei ringraziare tutte le persone che mi hanno accompagnato in questi tre anni. Ringrazio il professore Fabio Palomba per la sua disponibilitá e per il supporto dato durante questo studio, grazie anche alle sue lezioni che mi hanno fatto appassionare maggiormente a questo campo di studi. Un altro grazie al dott. Francesco Casillo per i suoi suggerimenti e per l'attenzione che mi ha dedicato per tutto il periodo della stesura di questa tesi.

Un grazie di cuore ai miei genitori e a mia sorella che sono stati pazienti e mi hanno sempre spronato nel mio corso di studi.

Un grande grazie a mia nonna che ha sempre tifato per me e ha sempre voluto vedermi continuare questo percorso con successo.

Ringrazio Antonio, Tony, Emanuele, Loredana, Aurora, Rebecca e tutti gli amici storici per avermi ascoltato e supportato nei momenti piú difficili. Ringrazio Adriano, Giovanni, Michele, Marco, Vincenzo, David, Luca e tutti gli amici conosciuti all'universitá che mi sono sempre stati vicini nelle situazioni complicate rivolgendomi parole sempre di conforto e strappandomi un sorriso.

Bibliografia

[1] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*. New York, NY, USA: Wiley, 1997.

[2] L. Cao and B. Ramesh, "Agile requirements engineering practices: An empirical study," *IEEE Software*, pp. 60–67, 2008.

[3] xDaryamo. (2023) Nfr-security-extraction-classification. [Online]. Available: https://github.com/xDaryamo/NFR-Security-Extraction-Classification

[4] I. Masnari, "Computazione quantistica: un'introduzione."

[5] A. Di Pierro, "Quantum computing," *Teaching Report*, 2010.

[6] T. M. Mitchell, "Machine learning," 1997.

[7] E. Alpaydın, *A Gentle Introduction to Machine Learning*. MIT Press, 2004.

[8] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[9] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[10] S. Aaronson, "The limits of quantum," *Scientific American*, vol. 298, no. 3, pp. 62–69, 2008.

[11] W. Yamany, N. Moustafa, and B. Turnbull, "Oqfl: An optimized quantum-based federated learning framework for defending against adversarial attacks in intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, pp. 1–11, 12 2021.

[12] A. Alanezi, B. Abd-El-Atty, H. Kolivand, and A. A. Abd El-Latif, "Quantum based encryption approach for secure images," in *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*. IEEE, 2021, pp. 176–181.

[13] E. D. Canedo, I. N. Bandeira, A. T. S. Calazans, P. H. T. Costa, E. C. R. Cançado, and R. Bonifácio, "Privacy requirements elicitation: a systematic literature review and perception analysis of it practitioners," *Requirements Engineering*, vol. 28, no. 2, pp. 177–194, 2023.

[14] C. Kalloniatis, E. Kavakli, and S. Gritzalis, "Addressing privacy requirements in system design: the pris method," *Requirements Engineering*, vol. 13, pp. 241–255, 2008.

[15] F. Casillo, V. Deufemia, and C. Gravino, "Detecting privacy requirements from user stories with nlp transfer learning models," *Information and Software Technology*, vol. 146, p. 106853, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584922000246

[16] F. Ebrahimi and A. Mahmoud, "Unsupervised summarization of privacy concerns in mobile application reviews," *Information Systems*, vol. 112, p. 102401, 2022.

[17] P. Sangaroonsilp, M. Choetkiertikul, H. K. Dam, and A. Ghose, "An empirical study of automated privacy requirements classification in issue reports," *Automated Software Engineering*, vol. 30, no. 2, p. 20, 2023.

[18] Y. Chen, S. Li, Y. Wang, and L. Liu, "A unified framework for classifying privacy and security requirements," *IEEE Transactions on Software Engineering*, vol. 47, no. 11, pp. 2178–2200, 2020.

[19] N. I. of Standards and Technology, "Privacy requirements framework," 2019. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-63-3/rev-1/final

[20] Y. Chen, S. Li, Y. Wang, and L. Liu, "A classifiers experimentation with quantum machine learning," *IEEE Access*, vol. 8, pp. 22 745–22 754, 2020.

[21] J.-X. Ling, H. Zhu, X. Li, X.-M. Wang, and X.-J. Zhang, "Quantum machine learning for text classification with lambeq," *Phys. Rev. A*, vol. 104, p. 032308, 2021.

[22] M. A. Nielsen and I. L. Chuang, "Quantum machine learning for image classification," *arXiv preprint arXiv:1104.3040*, 2011.

[23] "Colaboratory," https://research.google.com/colaboratory/faq.html, verified: 2022-03-24.

[24] D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark, and B. Coecke, "lambeq: An Efficient High-Level Python Library for Quantum NLP," *arXiv preprint arXiv:2110.04236*, 2021.

[25] C. Quantum. (2023) lambeq. [Online]. Available: https://github.com/CQCL/lambeq

[26] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134