ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA
CAMPUS DI CESENA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

GEAR: UNA PIATTAFORMA BIG DATA PER L'ELABORAZIONE DI STREAM DI DATI ATTRAVERSO MACHINE LEARNING E BUSINESS RULES

Tesi in
Big Data

Relatore
Dott. GALLINUCCI ENRICO

Correlatore
PROSCIA ALESSANDRO

Presentata da
CAVINA EUGENIO

ANNO ACCADEMICO 2018–2019
SESSIONE III

KEYWORDS

big data

streaming

fraud detection

business rules

machine learning

brms

Indice

Introduzione

La digitalizzazione sta portando ad una profonda trasformazione del mercato dei pagamenti elettronici, e nel panorama Europeo i regolatori intervengono a livello normativo per garantire concorrenza leale e diritti ai consumatori. I regolamenti Payment Services Directive (PSD e PSD2), definiscono le modalità con cui tutti gli operatori del mercato devono agire in quest'ottica. Fondamentali in queste norme, sono gli aspetti riguardanti le misure e i meccanismi di sicurezza a tutela degli utenti, effettuate anche attraverso una valutazione del rischio delle singole operazioni finanziarie. L'utilizzo di sistemi digitali finanziari da parte degli utenti è in continua crescita e questo porta i sistemi esistenti a dover adattarsi a carichi di lavoro sempre più grandi. Inoltre, automatizzare questi processi di valutazione e reazione rispetto a eventi di sicurezza, diventa fondamentale per rendere scalabili e sostenibili i sistemi, sgravando gli operatori umani da compiti non essenziali. Nel contesto indicato, l'obiettivo della tesi consiste nella costruzione di un sistema che sfrutti le potenzialità degli attuali strumenti di big data per analizzare in tempo reale un'elevata quantità di dati, fornendo una valutazione del rischio che porti ad una reazione automatica. La tesi è stata svolta in collaborazione con l'azienda Imola Informatica Spa che ha messo a disposizione la conoscenza, le infrastrutture e i dati per poterla costruire.

Sebbene esistano già sistemi di Online Fraud Detection (OFD) proprietari che supportano i team di analisi di frodi, questa tesi ha l'obiettivo di definire un modus operandi più generale per costruire questi sistemi, estendibili e modularizzabili rispetto alle necessità previste in un dominio specifico. Oltre

allo studio dell'architettura necessaria ad elaborare in real-time big data, si illustrano le tecniche di analisi per valutare il rischio di frode. Le più utilizzate sono basate su due tipologie di profilazione comportamentale: quella basata su anomalie rispetto a pattern normali e quella basata su pattern espliciti di abuso. Ibridare queste tecniche bilancia vantaggi e svantaggi di entrambe e per questo, il modello definito in tesi cerca di fondere questi due approcci. Anche per la parte di reazione e valutazione è necessario individuare lo strumento più adatto perché nei domini gestibili dal sistema costruito in tesi, la logica di valutazione è tipicamente complessa e in continua evoluzione. Per questo, si analizza l'approccio basato sulla definizione della logica di business attraverso regole e non quello più tradizionale di codifica della stessa direttamente all'interno dei programmi.

La soluzione proposta possiede caratteristiche generalizzabili e riutilizzabili in domini applicativi che astraggono dall'ambito dei pagamenti ma, che prevedono una sequenza concettuale di passaggi (raccogliere, arricchire, valutare e reagire) eseguita su uno stream di dati. Per questo motivo, la tesi non si limita alla creazione del sistema per un caso d'uso specifico ma prevede la definizione di un modello concettuale, di un'architettura e una possibile implementazione tecnologica. L'unione di tutte queste valutazioni e analisi ha portato alla definizione del modello concettuale GEAR (Gather Enrich Assess React), alla base della successiva architettura e pipeline di trasformazione effettuata sugli eventi. Successivamente si definisce un possibile stack tecnologico completamente open source e si illustra nello specifico e a livello implementativo, il funzionamento dei vari moduli che compongono il sistema. Tra le varie tecnologie e tecniche utilizzate si fa uso di Open Source Intelligence (OSINT) per l'arricchimento mentre, machine learning e Business Rules Management System (BRMS) per le valutazioni e reazioni.

Per quanto riguarda la struttura del documento, nel Capitolo 1 viene descritta la problematica da cui è nata la tesi e come le caratteristiche e requisiti, possano essere generalizzati in una soluzione riutilizzabile. Viene inoltre chiarito il concetto di OSINT perché centrale nella fase di arricchimento. Nel

Capitolo 2 si riporta lo studio effettuato in letteratura per le architetture di sistemi di valutazione del rischio e strumenti big data, le tecniche e gli algoritmi di analisi utilizzate per l'individuazione di frodi e anomalie e infine gli strumenti e le tecniche per la gestione di complesse logiche di business in evoluzione. Nel Capitolo 3 si illustra il processo che ha portato alla costruzione del modello, la definizione dell'architettura del sistema e la scelta dello stack tecnologico; oltre a questo, viene descritta l'implementazione vera e propria dei singoli moduli che compongono il sistema. Infine nel Capitolo 4 si mostra l'applicazione del modello al caso d'uso originale della tesi, partendo dall'analisi del dataset a disposizione fino ad arrivare alle valutazioni di efficacia e efficienza, sia sui singoli moduli che sull'intera pipeline di trasformazione.

Capitolo 1

Definizione del problema

In questo capitolo si definisce la problematica inizialmente valutata come obiettivo di tesi in Sezione 1.1, derivante da un cambiamento della normativa europea per quanto riguarda la valutazione del rischio nei pagamenti elettronici. Successivamente in Sezione 1.2, s'illustra come le caratteristiche di una soluzione possibile siano riapplicabili in altri contesti, sia finanziari che non, dove sia necessario valutare il rischio di un evento. Infine, in Sezione 1.3 si spiega una tipologia di tecnica di intelligence potente e accessibile, che possa risultare utile per aumentare il valore delle informazioni ai fini della valutazione del rischio.

1.1 Un sistema di valutazione del rischio per l'online banking

L'Europa ha definito un percorso chiaro per il rinnovo dell'infrastruttura dei mercati finanziari attraverso regolamenti come la Seconda Direttiva sui Servizi di Pagamento (PSD2). Questo sta portando ad un periodo di profonda trasformazione del mercato dei pagamenti in Europa. La digitalizzazione sta accrescendo le aspettative per pagamenti veloci sia per acquisti al dettaglio sia per transazioni commerciali. La tecnologia, che offre opportunità di innovazione ed efficienza, sta consentendo anche ai concorrenti (giganti della

vendita al dettaglio, reti di carte e start-up di banche digitali) di sfidare i fornitori e i modelli di business già esistenti. I regolatori intervengono per promuovere la concorrenza, per proteggere i diritti dei consumatori e promuovere l'efficienza dei pagamenti: ad esempio richiedendo alle banche di condividere i dati dei clienti a fornitori terzi per informazioni sui conti e servizi di pagamento (open banking) [1].

Il panorama normativo europeo legato ai pagamenti elettronici ha visto l'entrata in vigore del Payment Services Directive (PSD) nel 2007, la direttiva che definisce un quadro giuridico comunitario moderno e coerente. La PSD risponde a questi obiettivi:

- regolamentare l'accesso al mercato per favorire la concorrenza nella prestazione dei servizi;
- garantire maggiore tutela degli utenti e maggiore trasparenza;
- standardizzare i diritti e gli obblighi nella prestazione e nell'utilizzo dei servizi di pagamento;
- stimolare l'utilizzo di strumenti elettronici di pagamento per ridurre il costo di inefficienti soluzioni come quelle cartacee e il contante.

Nel 2016 è entrata in vigore la seconda direttiva sui servizi di pagamenti chiamata PSD2, da recepire negli stati membri entro due anni. A differenza della precedente, questa direttiva incide sulle nuove procedure di sicurezza per l'accesso al conto online e ai moderni servizi di pagamento, offerti dalle banche tradizionali e dai nuovi operatori del mercato [2]. Le norme disciplinate dalla PSD2 che regolano le nuove misure di sicurezza, sono contenute nel Regolamento delegato (UE) 2018/389, che ha trovato applicazione il 14 settembre 2017. Sui sistemi di pagamento, l'articolo 18 del Regolamento prevede esplicitamente la valutazione del rischio connessa alle operazioni, per poi proporre la modalità di autorizzazione della transazione più adatta [3].

L'utilizzo sempre maggiore dei pagamenti elettronici ha portato alla necessità di regolamentare il mercato, sia per favorire la concorrenza, sia per

aumentare il grado di tutele per gli utenti europei. La quantità di frodi identificate sta infatti aumentando ed esplicativo in questo senso, è il grafico 1.1 delle frodi informatiche denunciate negli ultimi 10 anni, ottenuto dai dati ISTAT. Questa tesi nasce all'interno dell'azienda Imola Informatica Spa come opportunità di ricerca inquadrata nel cambiamento normativo in atto.

La problematica prevede di ricercare una soluzione alternativa rispetto ai sistemi antifrode bancari presenti attualmente sul mercato (e utilizzati dai clienti dell'azienda). La soluzione deve sia rispettare la nuova normativa sia utilizzare tecnologie open source ma, la reale utilizzabilità di una soluzione di questo tipo, dipende dal livello di efficacia e efficienza che riesce a raggiungere. Per questo il sistema completo deve garantire un livello di identificazione dei rischi almeno pari alle soluzioni già presenti sul mercato e deve riuscire a processare grandi quantità di dati near-real time.

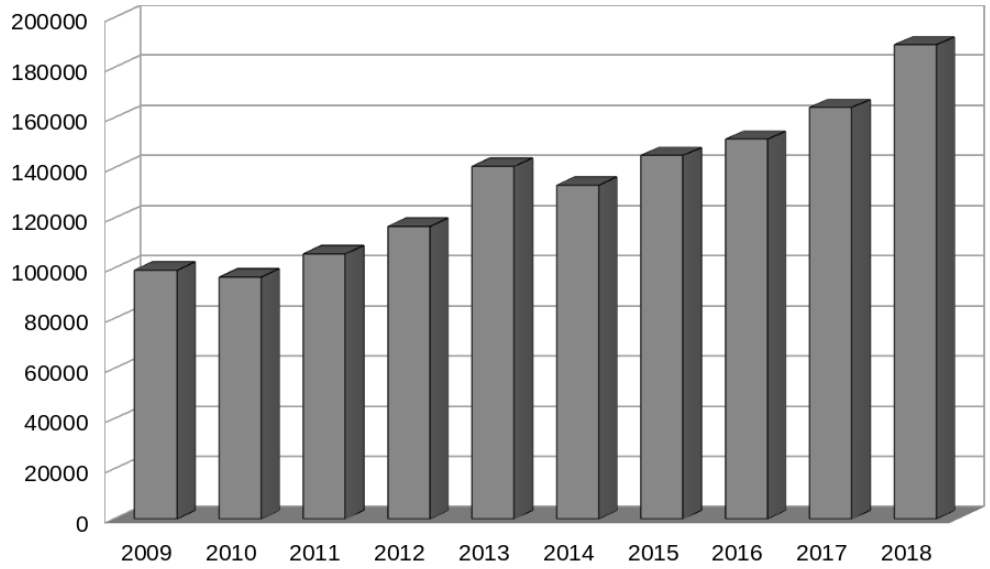

Figura 1.1: Frodi informatiche denunciate dalle forze di polizia all'autorità giudiziaria (dati ISTAT: http://dati.istat.it/Index.aspx?DataSetCode=DCCV_DELITTIPS)

1.2 Generalizzazione della soluzione

Un sistema in grado di analizzare in tempo reale una sequenza di eventi, fornendo una valutazione del rischio che porti ad una reazione automatica, può essere applicata ad altri contesti. Per questo la soluzione prevista in questa tesi ha un livello di generalizzazione più alto rispetto all'online banking che rimane però, il caso d'uso applicativo. La soluzione architetturale, il modello di analisi e la combinazione di tecnologie scelte deve essere quindi riutilizzabile in altri contesti con sforzi ridotti, modificando solamente la logica di business. A livello concettuale la soluzione potrebbe diventare un framework vero e proprio.

Il concetto di reazione automatica è fondamentale nella soluzione e se combinata a strumenti che mettono a disposizione API, può automatizzare completamente un processo (dagli input alle azioni intraprese di conseguenza). Il processo deve essere però composto da un insieme di attività ben definito e facilmente ripetibile. Il livello di complessità delle attività influenza poi l'applicabilità della soluzione che dipende dal livello espressivo delle tecnologie utilizzate.

Oltre agli ambiti simili all'online banking (come tutti i meccanismi di pagamento elettronico), un'applicazione possibile potrebbe essere l'automazione dei processi del Security Operation Center (SOC). Un SOC può essere definito sia come una squadra, che spesso opera a turni 24 ore su 24, sia come una struttura dedicata e organizzata per prevenire, rilevare, valutare e rispondere alle minacce e agli incidenti di cybersecurity, nonché per soddisfare e valutare le normative conformità [4]. All'interno del SOC è tipicamente presente il Security Information and Event Management (SIEM) il cui compito è quelo di supportare il rilevamento delle minacce, la conformità e la gestione degli incidenti di sicurezza attraverso la raccolta e l'analisi (sia in tempo reale sia storica) di eventi di sicurezza [5]. Questo modulo è necessariamente collegato alla gestione dei log e tipicamente ha queste funzionalità [6]:

- raccolta di log e dati contestualizzati;
- normalizzazione e categorizzazione;
- correlazione (in tempo reale e non);
- notifiche e avvisi;
- prioritizzazione degli eventi;
- visualizzazioni in tempo reale;
- reportistica;
- workflow delle operazioni di sicurezza.

In un contesto dove il numero di eventi di sicurezza nell'infrastruttura aumenta, l'automazione di questi task può portare a riduzione dei costi e tempi di gestione del SIEM. Esistono già gruppi di prodotti commerciali che si occupano di automatizzare questi processi.

1.3 Open source intelligence

1.3.1 Definizione

Il tipo di eventi processati dal sistema in questo documento possiede solitamente l'indirizzo IP, un'informazione facilmente correlabile ad altre sorgenti dati esterne. L'utilizzo di queste sorgenti esterne può aumentare il valore informativo ai fini della determinazione di un eventuale pericolo di sicurezza. Queste informazioni di arricchimento sono ottenute sia da piattaforme pubblicamente esposte tramite API sia da sorgenti dati interne e proprietarie. Si nota quindi che, informazioni di valore possono essere estratte anche da sorgenti pubbliche e questa attività è definita open source intelligence (OSINT).

OSINT deriva dalla raccolta, dalla elaborazione e dall'analisi delle informazioni solitamente ottenute da sorgenti che le rendono disponibili pubblicamente e con le seguenti caratteristiche [7]:

- tipicamente disponibili attraverso vie multimediali;
- non classificate (quindi non coperte da segreto di stato);
- non proprietarie eccetto quelle protette da diritti di autore;
- non ottenute clandestinamente.

Per chiarire meglio quali siano effettivamente le informazioni pubbliche con le precedenti caratteristiche è utile fornire una classificazione, sulla base della loro natura [8]:

- media (ad esempio, giornali, riviste, radio e televisione);
- Internet (ad esempio, pubblicazioni online, blog, forum, contenuti creati da utenti e social network);
- dati pubblici degli enti dello stato (ad esempio, report, budget, indici telefonici, conferenze stampa, siti web e discorsi);
- pubblicazioni professionali e accademiche;
- dati commerciali (ad esempio, immagini satellitari, valutazioni finanziarie o industriali e database);
- letteratura grigia (ad esempio, report tecnici, brevetti, documenti non pubblicati e newsletter).

L'utilizzo di API pubbliche che forniscano informazioni disponibili da Internet può risultare molto utile per inserire il concetto di OSINT all'interno dei sistemi informatici.

1.3.2 Vantaggi e svantaggi

OSINT ha le sue origini negli Stati Uniti, più precisamente dopo la creazione del Foreign Broadcast Monitoring Service (FBMS), un'agenzia responsabile del monitoraggio delle comunicazioni estere dell'Asse durante la Seconda Guerra Mondiale. Un esempio di OSINT eseguito da questa agenzia, è stato quello di trovare una correlazione nella variazione del prezzo delle arance a Parigi con il bombardamento dei ponti ferroviari da parte degli Alleati. In questo modo, riuscivano a capire se i bombardamenti erano andati a buon fine o meno [9].

Secondo un report del 2005 redatto dalla Commissione sulle capacità degli Stati Uniti nella ricerca delle armi di distruzione di massa, OSINT dovrebbe essere inclusa in tutte le sorgenti dei processi di intelligence per i seguenti motivi [10]:

- la natura mutevole delle esigenze di intelligence obbliga le agenzie a comprendere rapidamente un'ampia gamma di paesi e culture straniere e questo può essere effettuato attraverso fonti aperte;
- le informazioni aperte forniscono una base per comprendere materiale classificato che preso da solo potrebbe portare ad avere prospettive limitate;
- informazioni aperte possono essere utilizzate per proteggere le informazioni classificate, per esempio per comunicazioni politiche o diplomatiche che non compromettano informazioni classificate;
- attraverso le informazioni aperte si può monitorare e "salvare" la storia di entità specifiche in tutto il mondo, per esempio una cultura.

L'utilizzo di OSINT deve però tenere in conto di alcune problematiche e rischi oltre all'esplosione dell'informazione. Secondo Arthur S. Hulnick, OSINT ha questi svantaggi [11]:

- il fatto che OSINT sia un input a basso costo non implica che sia gratuito;

- le sorgenti utilizzate potrebbero contenere disinformazione, dati ambigui, dati contrastanti, messaggi segreti o sciocchezze;
- le sorgenti potrebbero essere mal interpretate;
- le sorgenti possono essere manipolate volutamente per fornire informazioni errate agli analisi OSINT.

1.3.3 Esempio con l'indirizzo IP

Nella problematica analizzata in questa tesi si è in presenza di eventi che rappresentano azioni intraprese da dispositivi tipicamente su Internet. Questo implica la presenza dell'indirizzo IP come informazione allegata all'evento.

L'IP è un informazione utilizzata spesso come componente principale negli indicatori di compromissione (IoCs), che sono utilizzati nella difesa di rete, identificazione, risposta agli attacchi e associazione di identità di dispositivi. L'indirizzo è considerato quindi un elemento chiave nella cybersecurity ed è per questo consigliato dispiegare attività di logging come meccanismo di partenza nei sistemi di sicurezza [12]. Quando si utilizza l'IP è necessario essere a conoscenza dell'inaffidabilità di questa informazione e di conseguenza, attribuirla ad un fingerprinting unico e immutabile può portare a valutazioni sbagliate. Più precisamente [13]:

- esistono tecnologie comuni che oscurano il soggetto che sta utilizzando un indirizzo;
- identificare un dispositivo non individua direttamente l'utilizzatore;
- dispositivi mobili e reti pubbliche permettono accessi non controllati a Internet;
- lo spazio degli IP è ampio e può essere riassegnato;
- gli indirizzi possono essere condivisi.

Nella fattispecie in esame, l'IP viene utilizzato per ottenere informazioni aggiuntive tramite tool di OSINT come: natura del nodo di uscita della comunicazione (nodo vpn, proxy, tor), porte aperte, vulnerabilità in servizi in esecuzione sulla macchina e geolocalizzazione dell'indirizzo. Queste informazioni permettono fare valutazioni in termini di intenzionalità dell'evento, senza considerare però l'IP come un fingerprinting della sorgente dell'evento stesso.

Capitolo 2

Stato dell'arte

L'obiettivo progettuale di tesi prevede la definizione di un modello con relativa architettura e implementazione che permetta di raggiungere gli obiettivi definiti nella Sezione 1.2. Per costruire questo sistema rispettando i requisiti previsti è necessario studiare lo stato dell'arte per fare alcune valutazioni. I primi argomenti analizzati in Sezione 2.1 sono le possibili architetture di sistemi esistenti per la valutazione del rischio e quelle per l'elaborazione real-time di Big Data. Oltre a questo, in Sezione 2.2 si indicano studi sulle tecniche algoritmiche per l'elaborazione di un fattore di rischio di un evento e in Sezione 2.3 soluzioni per valutare una possibile reazione conseguente (solitamente definita da esperti di dominio).

2.1 Architetture

Questa sezione cerca di definire delle linee guida architetturali su come costruire il sistema, iniziando da valutazioni riguardo i sistemi esistenti per l'identificazione di frodi in real-time. Una volta identificate le caratteristiche di questi sistemi, è necessario studiare come strutturarle attraverso le analisi in real-time di Big Data, lo scenario previsto da requisiti. Il Data Streaming è fondamentale in questo senso ma necessita di ampliare il discorso architetturale riguardo ai tipici livelli previsti in una pipeline di elaborazione di

stream. Quest'ultima valutazione è necessaria per costruire un sistema flessibile, estendibile e modularizzabile, che permetta di gestire anche domini che evolvono nel tempo.

2.1.1 Sistemi di valutazione del rischio di transazioni

Le organizzazioni che accettano pagamenti su Internet dispiegano sistemi di online fraud detection (OFD) che aiutano nell'individuare e prevenire frodi in tempo reale. Questi meccanismi si occupano di controllare e individuare comportamenti malevoli come frodi nei pagamenti, riconducibili al furto degli account o alla creazione di finte identità. I vendor possono poi fornire funzionalità aggiuntive mirate alla sicurezza dei client utilizzati dagli utenti. Il concetto su cui si basano questi controlli è l'integrità delle pagine web o delle applicazioni mobile, comprese quelle ibride [14].

Per costruire la soluzione prevista in questa tesi è fondamentale capire come sono inquadrati questi sistemi all'interno delle organizzazioni oltre che le loro caratteristiche. Secondo un report Gartner [15] di aprile 2019 i prodotti presenti sul mercato forniscono uno o entrambi i servizi seguenti:

- eseguono in background dei processi trasparenti agli utenti e che utilizzano dalle centinaia alle migliaia di attributi contestuali - come ad esempio geolocalizzazione, caratteristiche dei device, comportamento dell'utente, navigazione e attività nelle transazioni - per determinare la probabilità di un'azione fraudolenta;

- combinano informazioni dell'utente salvate precedentemente con quelle ottenute dinamicamente dalla richiesta di transazione per confermare o meno l'identità dell'utente.

Attraverso questi meccanismi i sistemi OFD ritornano allarmi e risultati, tipicamente tramite un punteggio di rischio, ai team e sistemi deputati al controllo delle frodi. A questo punto avviene una reazione appropriata al rischio come ad esempio:

- sospensione della transazione se il comportamento ha la soglia superiore al livello di sospetto (comportamento automatizzabile);
- revisioni manuali della transazione e dell'utente;
- attivazione di ulteriori meccanismi di verifica dell'identità per garantire la legittimità dell'utente come le one-time password (comportamento automatizzabile).

In Figura 2.1 si può vedere una possibile scelta architetturale e tecnologica di un prodotto commerciale OFD. Da questi sistemi si possono estrapolare alcune caratteristiche funzionali che sono applicabili nella soluzione oggetto di tesi:

- il concetto di punteggio di rischio è fondamentale e presente in ogni sistema di OFD ma la modalità con cui viene calcolato e valutato cambia da implementazione a implementazione;
- essere online implica che ci siano valutazioni e reazioni in tempo reale, di conseguenza la tecnologia sottostante deve permetterlo;
- questi sistemi sono inseriti in un contesto più ampio che prevede:
  - client dinamici e diversificati (fondamentale la trasparenza);
  - sorgenti di informazioni esterne al sistema stesso per le valutazioni;
  - entità e meccanismi di protezione che reagiscono in modalità più o meno automatiche sulla base delle valutazioni eseguite.

2.1.2 Big Data

Il sistema progettato in questa tesi prevede scenari applicativi con caratteristiche riconducibili a un contesto di analisi near real-time di Big Data. Esistono infatti molte definizioni di Big Data ma tutte possono essere ricondotte al seguente concetto: si tratta di un insieme di dati talmente voluminoso e

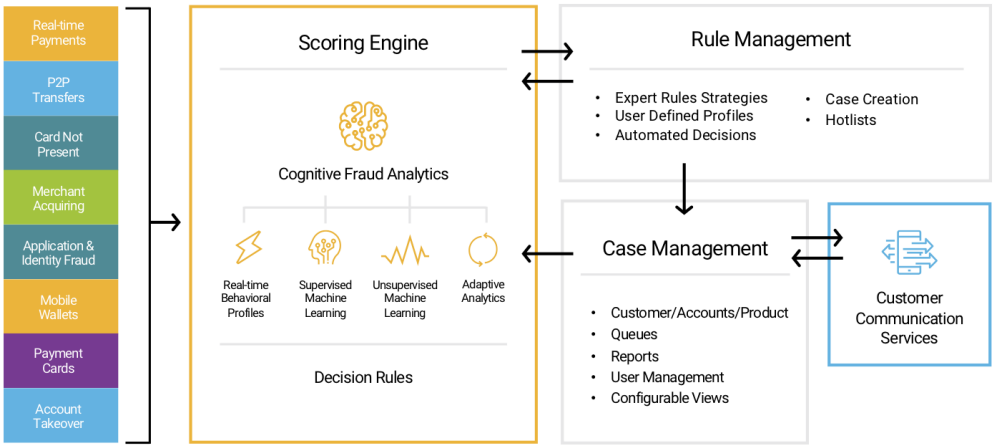

Figura 2.1: I componenti dell'OFD di Fico (https://www.fico.com/en/products/fico-falcon-platform)

complesso che il software utilizzato tradizionalmente per il processamento dei dati risulta inadeguato. Il concetto di complessità può essere ampliato in altri tre concetti che uniti al volume, sono definiti le quattro V dei Big Data:

- Volume, inteso come la scala dei dati con cui si ha a che fare;
- Velocità, intesa come la rapidità con cui vengono generati e devono essere gestiti i dati;
- Varietà, intesa come l'eterogeneità dei dati che provengono da sorgenti differenti e in formati completamente diversi tra di loro (strutturati, non strutturati, testuali e multimediali);
- Veracità, intesa come incertezza nella correttezza dei dati generati (inconsistenti, incompleti, ambigui e approssimati).

È già stato detto che l'utilizzo dei pagamenti elettronici è in crescita e di conseguenza, ci si può aspettare che le istituzioni finanziarie si troveranno a gestire sempre maggiori quantità di informazioni giornalmente. Il sistema inoltre, deve gestire in maniera continua e rapida le informazioni generate in un contesto in cui la quantità dei dati ricevuti sia incostante. Oltre a questo,

i dati da analizzare e quelli utilizzati per gli arricchimenti sono eterogenei perché possono provenire da sorgenti differenti, ad esempio client mobile o sistemi di backend. Infine, analisi corrette devono tenere conto anche del contesto in cui vengono generati e utilizzate le informazioni (ad esempio cambia il client utilizzato dall'utente oppure si cambiano le sorgenti di arricchimento).

Appurato che sia necessario costruire una soluzione che possa gestire Big Data, è importante definire quali sono i componenti logici di un'architettura di questa natura [16]:

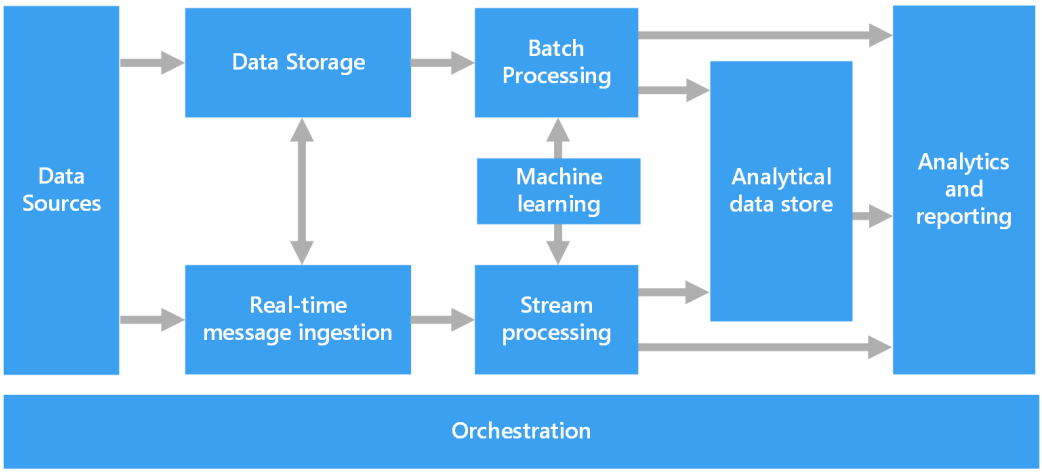

Figura 2.2: Componenti logici di un'architettura Big Data [16]

- sorgenti dati come archivi di applicazioni (ad esempio database), file statici generati dalle applicazioni (ad esempio file di log) e dati in tempo reale (ad esempio quelli generati da dispositivi IoT);
- archiviazione, perché i dati per le operazioni di elaborazione batch vengono in genere inseriti in un archivio di file distribuito che, può contenere volumi elevati di file di grandi dimensioni in vari formati;
- elaborazioni batch per filtrare, aggregare e preparare i dati di dimensioni considerevoli alle analisi;
- inserimento di messaggi in tempo reale per acquisire e archiviare i messaggi utilizzati dall'elaborazione del flusso;

- elaborazioni del flusso, perché dopo avere acquisito i messaggi in tempo reale la soluzione deve elaborarli filtrando, aggregando e preparando in altri modi i dati per l'analisi;

- archivio dei dati analitici, perché le soluzioni spesso forniscono i dati elaborati in un formato strutturato su cui è possibile eseguire query con strumenti analitici;

- analisi e creazione di report, perché l'obiettivo della maggior parte delle soluzioni per Big Data è fornire informazioni dettagliate sui dati tramite strumenti di analisi e report;

- orchestrazione, per automatizzare flussi di lavoro che vengono ripetuti lungo tutta la pipeline Big Data.

Lavorando con grandi quantità di dati, a causa della latenza, può essere necessaria una notevole quantità di tempo per fare analisi in modalità batch. Di conseguenza, queste elaborazioni non possono avvenire in tempo reale e i risultati vanno archiviati per essere utilizzati successivamente. La parte di elaborazione del flusso è quindi utile per avere risultati in tempo reale ma solitamente, con una perdita di precisione. Questo rende anche possibile combinare i risultati delle analisi del flusso durante le elaborazioni batch successive.

Per fornire queste due tipologie di elaborazioni sono presenti in letteratura due architetture simili ma alternative: architettura Lamba (Figura 2.3) e architettura Kappa (Figura 2.4). La Lamba (proposta da Nathan Marz) è quella più tradizionale che prevede di incanalare in due rami di elaborazione i dati in arrivo, ognuno dei quali ha la propria architettura e infrastruttura. In questo modo viene però duplicata la logica computazionale. Successivamente è stata proposta l'architettura Kappa (da Jay Kreps) che prevede un singolo ramo di elaborazione che gestisce i dati in real time affiancato ad un ramo di archiviazione. Si può quindi emulare l'elaborazione batch riapplicando l'elaborazione ai dati archiviati.

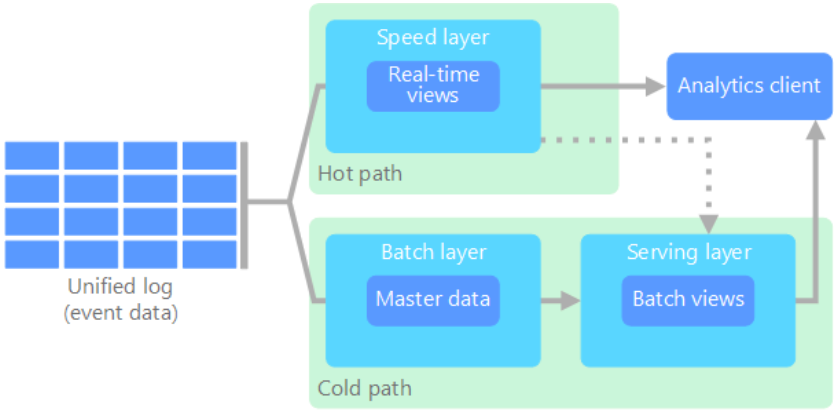

Figura 2.3: Architettura Lamba [16]

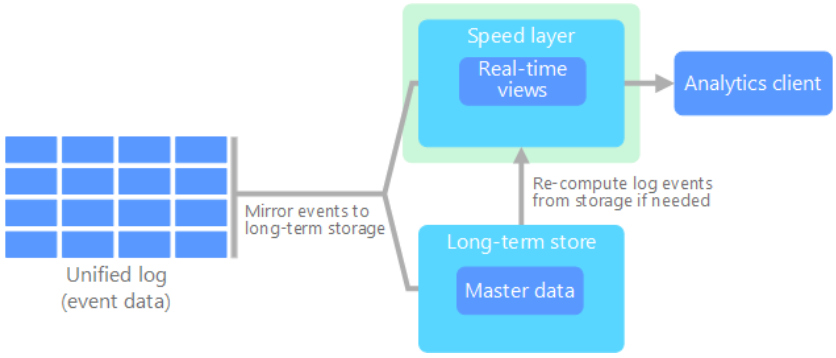

Figura 2.4: Architettura Kappa [16]

2.1.3 Data Streaming

La soluzione nella tesi prevede l'analisi near real time dei dati in maniera continua quindi, in un contesto di Big Data, si può parlare di un insieme di dati infinito da elaborare a bassa latenza. La continuità necessita di un sistema sempre attivo e in grado di regolarsi sulla base della quantità di dati ricevuti. Questi concetti sono quelli alla base del Data Streaming. L'architettura per gestire un data stream necessità dei seguenti livelli [17]:

1. livello di raccolta dei dati;
2. livello di accodamento dei messaggi;
3. livello di analisi;
4. livello di accesso ai dati.

A questi livelli è affiancata l'archiviazione a lungo termine per mantenere i dati per futuri elaborazioni e l'archiviazione in memoria di supporto al livello di analisi. L'architettura è mostrata in Figura 2.5.

Livello di raccolta dei dati

- prevede pattern di interazione diversi (request/response, publish/subscribe, one way, streaming)
- utilizza strategie per la gestione dei fallimenti
- gestisce dati in diversi formati

Una nota è necessaria per il pattern di streaming perché si differenzia da tutti gli altri per la tipologia di interazione che avviene. Normalmente un client fa una richiesta ad un servizio che potrà o meno rispondere mentre in quello di streaming il servizio si trasforma nel client che fa una richiesta e riceve risposte in maniera continua.

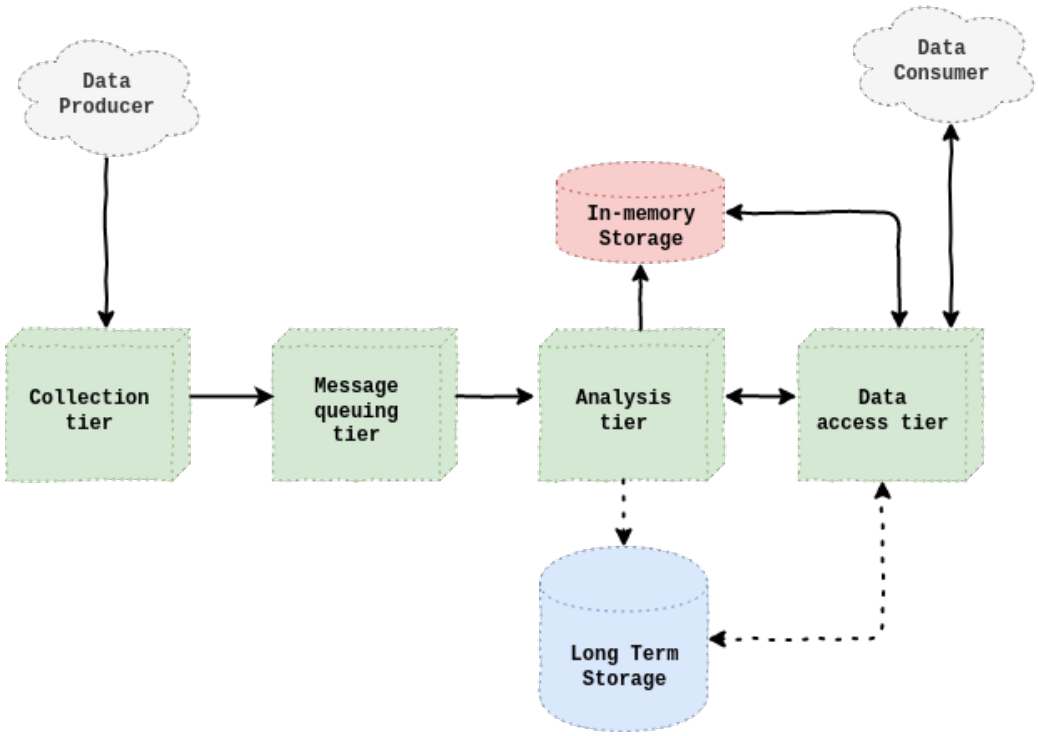

Figura 2.5: Architettura Data Streaming

Livello di accodamento dei messaggi

- gestisce il trasporto tra gli altri livelli
- permette di disaccoppiare le operazioni della pipeline
- fornisce un framework per la comunicazione sicura
- gestisce la comunicazione tra n data stream verso m consumer

Questo livello si basa su quattro concetti fondamentali: Producer, Consumer, Broker e coda dei messaggi (vedi in Figura 2.6). La coda è fondamentale per permettere anche a Consumer lenti di accedere ai dati oltre che per fornire garanzie semantiche. Esistono tre tipologie di garanzie semantiche:

- *esattamente una volta* prevede che un messaggio non sia mai perso e venga letto solo una volta;

- *al massimo una volta* prevede che un messaggio possa essere perso ma mai letto due volte;
- *almeno una volta* prevede che un messaggio non sia mai perso ma possa essere letto due volte.

Il primo tipo di garanzia è fondamentale in quelle applicazioni dove i dati sono strettamente legati ad un valore finanziario e può essere raggiunta più facilmente attraverso l'utilizzo della terza aggiungendo un controllo lato Consumer.

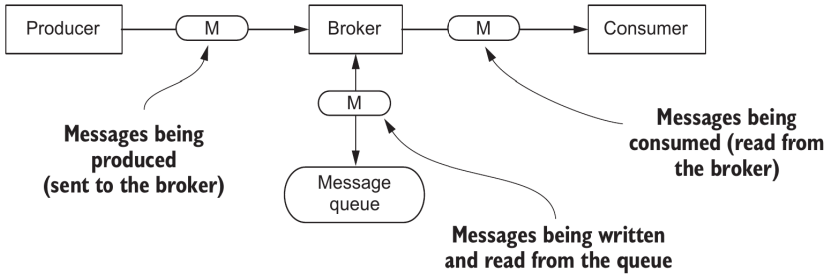

Figura 2.6: Relazione tra Producer-Broker-Consumer [17]

Livello di analisi

- adotta un modello a query continua
- utilizza degli algoritmi specifici per i problemi su streaming
- in caso di semantica *almeno una volta* è probabilmente necessaria un'esecuzione distribuita dei job idempotenti

Il modello a query continua prevede che l'interrogazione sia lanciata una volta ma venga eseguita ripetutatamente sui dati, tenendo conto dei vincoli di memoria e tempo.

Il primo vincolo è strettamente legato alla dimensione infinita dello stream e implica un singolo passaggio del dato all'interno dell'algoritmo (algoritmi

one-pass) oltre che tecniche per sintetizzare lo storico dei dati. Quest'assunzione può avere conseguenze significative sugli algoritmi utilizzati, per esempio alcuni algoritmi di Data Mining tradizionali prevedono passaggi multipli sui dati.

Il secondo vincolo prevede che i dati non elaborati in tempo vadano esclusi e che gli algoritmi possano perdere efficacia nel tempo (*concept drift*). Il fenomeno del concept drift può avvenire quando i dati evolvono nel tempo e quindi le proprietà statistiche dell'insieme variano. In base ai modelli predittivi utilizzati si dovrà o meno tenerne in conto.

Livello di accesso ai dati

- espone i dati analizzati tramite pattern e protocolli di comunicazione al Consumer (che può essere un ulteriore pipeline)
- utilizza strategie per la gestione dei fallimenti
- fa transitare i dati in diversi formati

I pattern di comunicazione mettono in relazione client con il componente di accesso e possono essere:

- sincronizzazione dei dati:
  - il client ha il dataset completo;
  - la sincronizzazione è solo iniziale e poi si procede gestendo solo le differenze;
  - il client ha una vista consistente dei dati;
  - se i dati sono numerosi è richiesta banda di trasferimento ampia;
  - i dati potrebbero non essere adatti al dispositivo;
  - bisogna risolvere i conflitti tra versioni differenti;
  - bisogna definire una politica di unione delle modifiche;
- invocazione remota di metodi o procedure (RMI e RPC):

  - il client ha i propri processi e reagisce quando viene chiamato l'handler;
  - il client deve esporre delle API;
  - individuare fallimenti è difficile (il client è disponibile? il client come fa a sapere che il componente non ha ricevuto nuovi dati?);
  - il client potrebbe non riuscire a gestire frequenze di aggiornamento troppo alte;

- scambio di messaggi:
  - il client richiede se ci sono nuovi dati;
  - sono inviati solo i dati più recenti;
  - il client necessita di mantenere metadati minimi per funzionare;
  - il componente di accesso non necessita di mantenere uno stato;
  - non c'è un meccanismo che avverta il client della presenza di nuovi dati;
  - il payload dei messaggi potrebbe essere molto ampio se il client non ha fatto richieste per un tempo relativamente lungo;

- publish-subscribe:
  - il client gestisce i suoi processi e reagisce quando arrivano nuovi dati;
  - il client non deve mantenere metadati sui dati correnti;
  - si può ottimizzare l'invio dei dati su più client;
  - il componente di accesso deve tracciare tutti i metadati relativi ai client e deve avere l'abilità di distribuirli nelle sue repliche in caso di fallimento.

Oltre ai pattern è importante valutare i protocolli da utilizzare, sulla base di frequenza di aggiornamento dei messaggi, direzione della comunicazione,

latenza nei messaggi, efficienza e tolleranza ai fallimenti. Quelli tipicamente utilizzati sono:

- Webhooks;
- HTTP Long Polling;
- Server-sent events (SSE);
- WebSockets.

2.1.4 Livelli multipli nel Data Streaming

La soluzione in questa tesi deve permettere l'analisi di uno stream di dati attraverso l'utilizzo di un'architettura estendibile e modularizzabile. Il sistema deve infatti prevedere la possibilità di posizionare nella pipeline di streaming multipli moduli di interazione con i dati, in base ai domini applicativi diversi o l'evoluzione degli stessi. L'aggiunta di complessità deve impattare nel minor modo possibile il sistema in essere, sia a livello di prestazioni che a livello di manutenibilità.

A livello concettuale ogni modulo dovrebbe quindi: essere il più possibile disaccoppiato dagli altri, comunicare in maniera asincrona e reattiva agli eventi, scalare e essere mantenibile singolarmente. Alcuni esempi di scenari che la soluzione deve supportare sono i seguenti:

- aggiunta di un livello di analisi che adotti uno specifico algoritmo di Machine Learning la cui elaborazione si sommi a un algoritmo di Data Mining tradizionale;
- divisione del flusso dello stream per effettuare due tipologie diverse di analisi (esempio offline e online);
- parallelizzazioni degli arricchimenti delle informazioni da sorgenti perché presentano latenze diverse;

- ottenimento di risultati meno accurati con bassa latenza e più accurati in un momento successivo partendo da un unico flusso di dati iniziale.

Per garantire queste caratteristiche risulta fondamentale il livello di accodamento messaggi tipico delle architetture Data Streaming oltre che, partizionare la logica di business in differenti moduli di elaborazione e analisi.

Un obiettivo simile è individuabile nella ricerca del 2018 di Sheik Hoque e Andriy Miransky [18]. Nel documento è proposta un'architettura multilivello che combina microservizi che si occupano della trasformazione dei dati, interconnessi attraverso il pattern publish-subscribe. In questo pattern ogni messaggio pubblicato in un topic viene ricevuto immediatamente da tutti quelli iscritti a quel topic. Un sistema così costruito assicura le caratteristiche definite nel paragrafo precedente.

Il numero di livelli dell'architettura è sintetizzato dalla formula $2n + 1$ dove n corrisponde ai livelli di comunicazione necessari. I livelli dispari eseguono la trasformazione dei dati e i livelli pari permettono la comunicazione.

Nel documento si descrive uno scenario di applicazione che prevede la fattorizzazione in livelli separati della logica relativa a: conversione, suddivisione, aggregazione e modellazione dei dati. Di conseguenza si definiscono 7 livelli totali, dove 3 sono di comunicazione e 4 di trasformazione. Quelli di trasformazione sono composti da microservizi e sono i seguenti:

- converter, dove è presente un microservizio per ogni sorgente che converte dati eterogenei in un formato universale e li invia al secondo livello;

- splitter, dove si applica la logica di business sui dati ricevuti inoltrandoli a uno o più topic nel quarto livello;

- aggregator, che inoltra i dati solo nei topic dove vi sono modelli di analisi interessati a specifici dati;

- modeller, dove è presente un microservizio per ogni modello, in attesa sui topic di suo interesse.

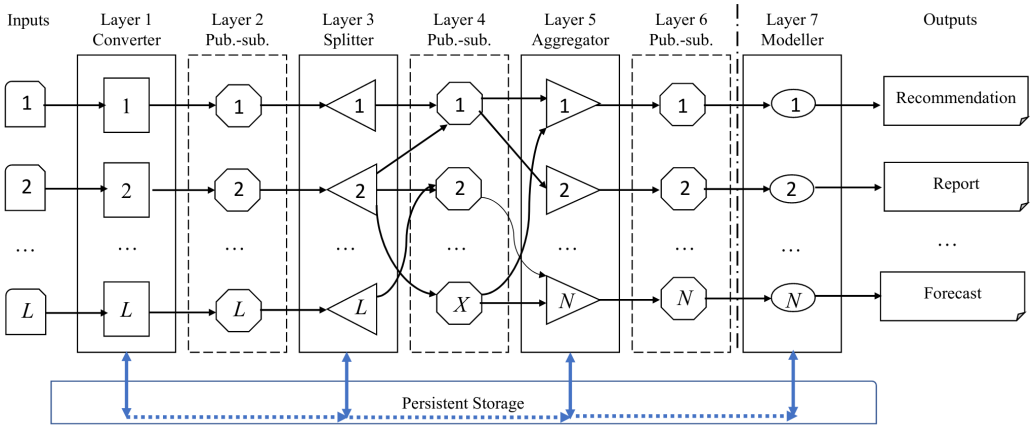

Figura 2.7: Diagramma dell'architettura a 7 livelli [18]

2.2 Tecniche di analisi

Parte fondamentale del sistema è quella della valutazione del rischio attraverso l'analisi dei dati. Il punto di partenza è stato individuare le tecniche più utilizzate nell'ambito dell'individuazione di frodi, problema riconducibile all'obiettivo di tesi. La ricerca ha fatto emerge due tipologie di tecniche utilizzati, l'anomaly based fraud detection e la misuse based fraud detection. Indipendentemente dalla tecnica è necessario considerare che questi sistemi hanno tipicamente a che fare con dataset problematici e successivamente vengono illustrati appunto questi problemi. Le tecniche inoltre, devono agire in real-time e il dominio in cui operano è caratterizzato da un ambiente in evoluzione. Si procede quindi ad un'analisi ulteriore dello stato dell'arte riguardo le tecniche e gli algoritmi utilizzati nel caso d'uso dell'individuazione di frodi nell'online banking. Questo dominio applicativo presenta infatti le caratteristiche ambientali previste.

2.2.1 Fraud Detection System

In questa tesi il rischio di un evento è considerato come la probabilità che questo possa essere illegittimo. Questo concetto può essere meglio spiegato dal termine frode, ovvero un inganno illecito o criminale destinato a deter-

minare guadagni finanziari o personali [19]. Per questo motivo nella tesi si è ritenuto necessario analizzare i Fraud Detection System (FDS) e utile in questo senso, è stata una survey del 2016 [20].

Un FDS ha il compito di scoprire e identificare le attività fraudolente nel momento in cui avvengono all'interno di un sistema per poi segnalarle [21]. Negli anni le tecniche manuali per individuare le frodi si sono evolute in sistemi automatici più o meno complessi. Questa complessità deriva principalmente dai tipi di meccanismi utilizzati per individuare la frode che possono variare dall'utilizzo di regole definite da esperti di dominio, fino ad arrivare all'utilizzo di metodi di Data Mining. Questi metodi utilizzano statistica, matematica, intelligenza artificiale e tecniche di machine learning per estrarre conoscenza da grandi quantità di dati. Utilizzando queste tecniche si possono ottenere i seguenti vantaggi:

- i pattern delle frodi sono ottenuti automaticamente dai dati;
- si ottiene la probabilità di frode per ogni caso e questo permette di avere un livello di priorità;
- si possono individuare tipi di frode non definiti precedentemente.

Per quanto concerne i metodi di Data Mining, quelli che si sono affermati recentemente nei FDS sono l'*anomaly based fraud detection* e *misuse based detection* [20].

Anomaly Based Fraud Detection

L'approccio utilizzato dai FDS con questo metodo di individuazione di anomalie si basa sul concetto di profilazione comportamentale. Viene creato un modello del comportamento normale per ogni entità e attraverso una fase di monitoraggio si possono individuare deviazioni dal pattern trovato. Per questo motivo queste tecniche sono considerate efficaci e utilizzate nell'ambito dei FDS [22]. Questo metodo ha il potenziale per individuare frodi non definite a priori e può essere categorizzato in tre tipologie [23]:

- supervisionato: algoritmi di classificazione (k-nearest neighbors, alberi decisionali, regressione logistica, Naïve–Bayes, Support Vector Machine SVM) e regressione (lineare, semplice e logistica);
- non supervisionato: clustering (k-means) e riduzione della dimensionalità (Principal Component Analysis);
- semi supervisionato.

Le tecniche supervisionate necessitano di un insieme di dati etichettato in "frode" o "non frode" e questo viene utilizzato per addestrare un modello di classificazione. Il vantaggio principale di questo metodo è che gli output degli algoritmi sono significativi per gli umani e possono essere utilizzati come pattern di discriminazione. Sono presenti però anche due limitazioni (oltre all'etichettamento sbagliato). La prima è causata dalla difficoltà di ottenere dei dati etichettati in grandi quantità. La seconda deriva dalla difficoltà di trovare etichette distinte per i dati in maniera che siano senza incertezza e ambiguità.

Le tecniche non supervisionate agiscono su un insieme di dati senza etichette sotto l'assunzione che la maggior parte delle istanze non siano frodi. A differenza delle precedenti, il modello non prevede l'utilizzo di classi. Il beneficio principale è che non sono necessari dati etichettati.

Le tecniche semi-supervisionate si pongono a metà tra le due precedenti perché necessitano di un piccolo numero di dati etichettati e un grande numero di dati non etichettati. L'obiettivo principale di queste tecniche è quello di addestrare un classificatore da entrambe le tipologie di dato mitigando eventuali difetti dei dati e il loro etichettamento [24].

Misuse Based Detection

Queste tecniche prevedono prima la definizione dei comportamenti fraudolenti attraverso pattern e successivamente, tutti gli altri comportamenti sono considerati come normali. Si utilizzano meccanismi basati su regole, metodi statistici o euristici per rivelare il verificarsi di specifiche transazioni sospette

[25]. Questo approccio è considerato un meccanismo di rilevamento semplice e veloce ma con un importante limite: non è possibile individuare tutte le tipologie di frodi perché riesce ad identificare solo pattern predefiniti di abuso [26]. Di fatto può essere ricondotto ad un sistema esperto, ovvero un sistema che cerca di riprodurre le prestazioni di una o più persone esperte in un determinato campo di attività [27]. L'approccio a regole definite da esperti per individuare frodi è ampiamente utilizzato da anni in tutti i domini impattati da questa problematica, dalle frodi finanziare alle telecomunicazioni ([28][29]).

Problemi e sfide nell'individuazione di frodi

Le due tecniche precedentemente illustrate hanno dei vantaggi e degli svantaggi che possono essere bilanciati costruendo soluzioni ibride. Utilizzando entrambi i metodi si compensa l'incapacità delle tecniche di misuse based detection di individuare nuove frodi mentre si limita la mancanza di generalizzazione e la presenza di falsi allarmi della anomaly based detection.

Anche ibridando le tecniche è necessario ricordare che l'individuazione di frodi è un dominio complesso e potrebbe quindi capitare di avere un sistema soggetto a fallimenti, con bassa accuratezza o che comunichi falsi allarmi. Ciò accade perché i sistemi di rilevamento delle frodi devono affrontare molteplici sfide da prendere in considerazione [20]:

- concept drift;
- distribuzione distorta;
- grandi quantità di dati;
- supporto alla real-time detection.

Il **concept drift** nelle analisi predittive e di machine learning, indica che le proprietà statistiche del fenomeno analizzato evolvono in modo imprevedibile con il passare del tempo. Questo porta ad una perdita di efficacia dei

modelli utilizzati. I FDS operano in ambienti dinamici dove i comportamenti degli utenti legittimi e non, cambia continuamente. Ad esempio, in un contesto di pagamenti elettronici, il comportamento dell'utente è influenzato da diverse cause esterne. Le cause esterne possono essere l'ammontare della transazione, la frequenza che è molto legata alle abitudini di spesa, il cambio di reddito, le risorse disponibili e lo stile di vita. Inoltre, i malintenzionati evolvono il proprio comportamento fraudolento quando viene bloccato il loro modus operandi.

La **distribuzione distorta** o problema delle classi sbilanciate avviene quando gli esempi di frode posseduti sono molti meno rispetto a quelli leciti. Questa situazione porta all'impossibilità di scoprire pattern della classe minoritaria; questo perché i classificatori possono essere sopraffatti dalla classe di maggioranza, ignorando completamente la classe di minoranza. In queste situazioni è fondamentale un meccanismo di bilanciamento efficace per rendere il rapporto tra eventi normali e frodi 1:1. Le tecniche di bilanciamento possono essere categorizzate su due livelli: a livello di dato o a livello algoritmico.

A livello di dato, in fase di pre-processing, si può fare sovracampionamento (si replicano i dati della classe minoritaria) o sottocampionamento (si rimuovono parte dei dati nella classe maggioritaria). Utilizzare il sovracampionamento può portare a diversi problemi tra cui l'overfitting di un modello, specialmente in caso di dati con rumore. Inoltre, fare sovracampionamento può portare alla produzione di modelli eccessivamente complessi [30].

A livello algoritmico si può agire in due modalità. La prima consiste nel inserire un fattore di costo per la classificazione errata della classe minoritaria (cost-sensitive learning). La seconda prevede di utilizzare direttamente algoritmi che siano robusti rispetto al problema delle classi sbilanciate: per le proprietà intrinseche che hanno, come per esempio Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [31] oppure rafforzandoli per esserlo, come modifiche interne di k-nearest-neighbor o SVM.

Quando si è in presenza di **grandi quantità di dati** con dimensionalità

alta, ovvero con numerosi attributi, il processo di individuazione di frodi può risultare complicato [32] e quindi rallentato. In un contesto di analisi real-time questo rallentamento potrebbe impattare sull'efficacia del sistema. I FDS utilizzano degli approcci per ridurre la dimensione dei dati, producendo un modello ridotto e quindi più rapido nelle analisi. Alcune delle tecniche utilizzabili per ridurre la quantità di informazione sono:

- riduzione della dimensionalità (come compressione dei dati, selezione degli attributi e costruzione di nuovi attributi);
- riduzione della numerosità (aggregazione di dati).

L'ultima sfida è il **supporto alla real-time detection** per l'individuazione online di frodi (il problema non si pone per le analisi offline che comunque potrebbero essere presenti). I FDS dovrebbero funzionare in maniera efficiente utilizzando tempo e memoria limitati per l'individuazione delle frodi. I sistemi hanno quindi beneficio se si utilizza un ridotto ammontare di informazioni ma anche se i meccanismi utilizzati hanno un livello di complessità ridotto.

2.2.2 Algoritmi di analisi comportamentale

L'analisi della letteratura FDS indica come tecniche più efficaci quelle basate sulla modellazione di un comportamento complessivo dell'entità rispetto ad una visione sulla singola transazione.

Tecniche basate su firma

In questo senso è importante il concetto di firma, ovvero una rappresentazione statistica del comportamento normale di un utente attraverso variabili. Confrontando le nuove transazioni di sistema con la firma di un utente si può stabilire la probabilità che rappresentino atti fraudolenti. La firma può essere inoltre ricalcolata a seguito dell'arrivo di una nuova transazione facendo evolvere la rappresentazione del comportamento. In Figura 2.8 è possibile vedere i componenti necessari per un'analisi basata su firma: è presente

un meccanismo di gestione delle firme di tutti gli utenti che si appoggia ad una base dati. I metodi basati su firma forniscono una soluzione che può supportare bene la profilazione di un account individuale, in questo modo la valutazione è indipendente dalle soglie definite a priori e globalmente per tutti gli utenti (che potrebbero portare a valutazioni errate) [33].

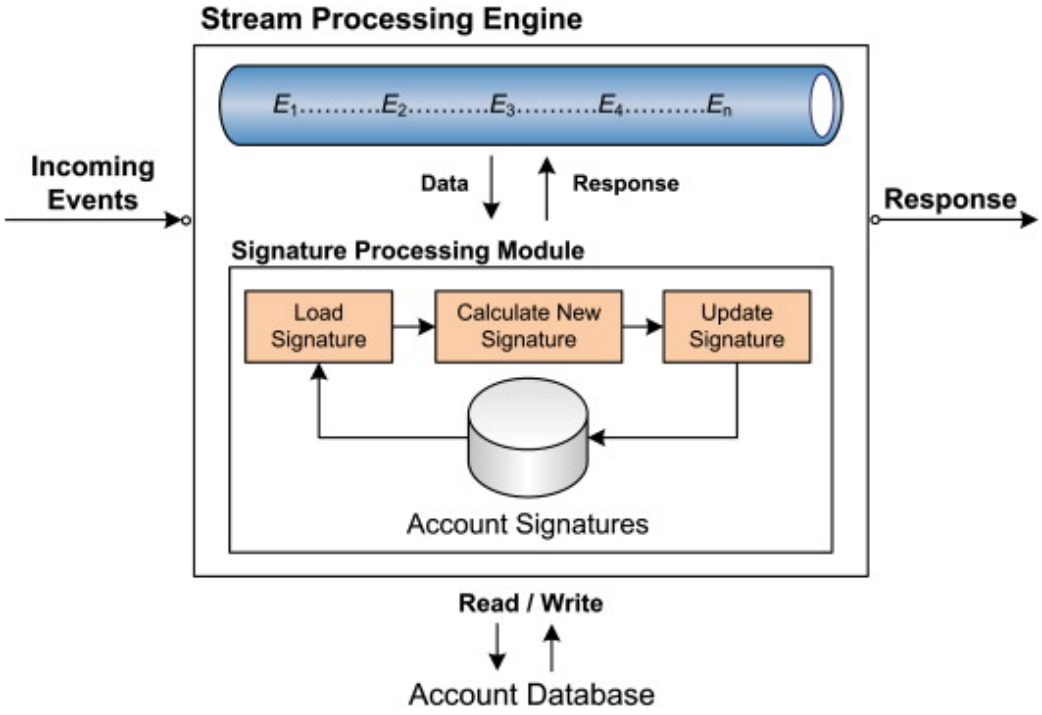

Figura 2.8: Componenti per processi basati su firma [33]

Cost Sensitive Neural Network (CNN)

Un framework efficace pensato per l'online banking proposto in questo documento [26] prevede la costruzione di un vettore per ogni transazione, oltre che un vettore costruito sulla base del comportamento storico dell'account. Si calcola poi il tasso di differenziazione di ogni transazione corrente rispetto al comportamento preferenziale del cliente calcolando la distanza tra i vettori. Nella fattispecie si utilizza il comportamento dell'utente dal momento del login al momento del logout quando esegue la transazione, salvando

tutte le operazioni effettuate e un timestamp. Questo scenario si distacca dal caso d'uso oggetto di tesi ma risulta interessante l'algoritmo utilizzato successivamente.

Si è scelto infatti di utilizzare una cost sensitive neural network (CNN), una rete neurale modificata appositamente per dare più importanza ad un certo tipo di errori rispetto agli altri. Esistono infatti due tipologie di errore nella classificazione:

- falso positivo che classifica una transazione fraudolenta come genuina;
- falso negativo che classifica una transazione genuina come fraudolenta.

Nel contesto dell'online banking le due tipologie di errore non hanno lo stesso costo. Una perdita derivata da un falso positivo è in generale più costosa di una perdita derivata un falso negativo [26]. Inoltre, una rete neurale ottiene performance migliori rispetto agli algoritmi di classificazione più classici nei problemi di classificazione estremamente sbilanciati. Utilizzando quindi una rete neurale e un approccio di learning basato su costo si può ottenere una buona soluzione per un problema di classificazione molto sbilanciato [26].

I benefici ottenuti combinando tecniche di profilazione comportamentali diverse

Un altro caso interessante presente in letteratura è BankSealer [34], un sistema di supporto alle decisioni pensato per l'analisi e l'individuazione di frodi nell'online banking. Durante la fase di addestramento, BankSealer costruisce un modello per ogni utente utilizzando le sue abitudini di spesa basandosi sulle transazioni passate. Combina tre modalità di profilazione per mitigare il problema di undertraining degli utenti con a disposizione uno storico di transazioni limitato. Queste modalità sono:

- quantificare l'anomalia di ogni transazione rispetto allo storico dell'utente;
- trovare un cluster a livello globale di utenti con abitudini di spesa simili;

- utilizzare una finestra temporale che misura l'anomalia del pattern di spesa corrente per ogni utente rispetto alle transazioni effettuate nella finestra.

A runtime, BankSealer supporta gli analisti segnalando le transazioni che deviano dai profili appresi precedentemente. Nel documento di BankSealer si effettua inoltre un'analisi di un dataset tipico di transazioni bancarie, illustrando valutazioni sugli attributi normalmente presenti. Oltre a questo, si forniscono dei test del sistema in scenari di frode bancarie definite da esperti per evidenziare l'utilità delle tre modalità di profilazione.

Algoritmi di anomaly detection nel contesto bancario

Infine si segnala una ricerca [35] che fornisce una panoramica delle varie classi di problemi e algoritmi per l'individuazione di anomalie nell'online banking (sia supervisionato che non). Il documento si concentra sui meccanismi più rilevanti nel contesto bancario, sia a livello di profilazione commerciale che a scopo di individuazione di frodi. Si indica che l'approccio supervisionato più immediato risulta quello di valutare un nuovo elemento in base alla distanza dagli elementi etichettati come "normali" e "anormali" (ad esempio usando k-nn). Altre tecniche per questa tipologia di problemi possono essere Support Vector Machines (SVM) e le Reti Neurali di tipologia Feed Forward oppure di tipologia Radial Basis Function (RBF).

Successivamente il documento [35] scende nello specifico dell'individuazione di anomalie con timestamp e quindi sequenzialmente posizionabili (nel caso precedente non si tiene in conto a livello implicito di un ordine temporale, vi sono solo punti che rappresentano le transazioni nello spazio). In questo contesto si possono utilizzare algoritmi basati su regressione oppure Hidden Markov Model (HMM) che costruiscono un modello a partire da un training e successivamente assegnano un punteggio di anomalia a una serie temporale da valutare. Vi è un'ulteriore suddivisione tra individuazione di anomalie all'interno di un'unica sequenza e tra più sequenze.

Fino ad ora si è dato per scontato di avere valori degli attributi solo numerici ma il sistema in questa tesi potrebbe avere casistiche con attributi categorici (o nominali). In ambito di individuazione di anomalie questo implica una problematica di misurazione della distanza perché complessa [35]. La distanza più immediata è binaria, 1 se il valore uguale o 0 se è diverso. Trasformare l'attributo categorico in un insieme di attributi categorici binari potrebbe non bastare per questi problemi. All'interno del documento [35] sono indicati riferimenti in letteratura ad algoritmi che si basato su item-set frequenti o basati su minimum description language (MDL) per ovviare a questa problematica.

2.3 Complessità dei fenomeni di business

Il sistema proposto in questa tesi necessita di agire in maniera autonoma e reattiva rispetto ad ogni singolo evento valutato. La reazione quindi dipende direttamente dal tipo di valutazione che è stata effettuata ma all'interno del sistema, queste valutazioni possono essere:

- fortemente legate alla logica di dominio;
- in continua evoluzione (per riflettere l'ambiente in cui opera);
- di cardinalità diversa in base al tipo di sistema;

Per gestire queste caratteristiche si riportano il concetto di logica di business e come questa, possa essere implementata in modo che il cambiamento e l'evoluzione della logica siano gestite più facilmente (attraverso i Business Rules Management System). Per avere una chiara idea della tecnologia si descrive infine l'algoritmo Rete, quello normalmente utilizzato nei BRMS.

2.3.1 Logica di business

Nel software, la logica di business o di dominio è la parte del programma che codifica le regole presenti nel mondo reale, determinando di fatto come i dati

vengono creati, modificati e salvati. Si contrappone alle parti del software che invece si occupano di gestire dettagli ad un livello di astrazione più basso. La logica di business è quindi collegata alle politiche dell'organizzazione stessa ma queste sono potenzialmente complesse (in termini di parametri e soglie da considerare) tanto da necessitare esperti nello specifico dominio.

Anche seguendo le buone pratiche di separazione dei dati dalle interfacce, la logica di business codificata è poi incorporata nelle strutture previste dalla programmazione, assieme al codice che gestisce e risolve gli errori, oltre che le funzioni di input e output. Conseguentemente, ogni cambiamento alle politiche di business richiede un controllo dettagliato e la modifica del codice stesso. L'evoluzione poi non è spontanea ma dipende dai cambiamenti sorti nell'ambiente dove opera il sistema e in un contesto dinamico, la frequenza delle modifiche può essere alta. Secondo Loucopoulos e Layzell il cambiamento può essere categorizzato in [36]:

- fisico, dove avviene un cambiamento materiale del sistema (ad esempio storage o sistema operativo);
- operazionale, dove evolve il modo in cui il sistema viene utilizzato operativamente (ad esempio modifiche alle interfacce di sistema o modifiche alle fasi di elaborazione o all'algoritmo);
- strategico, dove cambia il modo in cui un sistema deve adattarsi al modello di funzionamento di un'organizzazione, in quanto deve fornire nuove funzionalità e strutture.

La terza categoria è quella che interessa in termini di politiche organizzative e generalmente prevede un grosso volume di codice utilizzato a fronte di una complessità concettuale ridotta. Questo porta alle seguenti problematiche:

- le regole di business implementate possono essere valutate nell'ordine delle istruzioni del programma;

- controllare la correttezza delle implementazioni delle politiche è difficile perché le persone con la conoscenza del dominio dovrebbero capire anche l'implementazione;

- la manutenzione del programma è difficile perché è il programma stesso a descrivere come funziona la politica piuttosto che contenere solo la politica e il fattore di attivazione.

Per questi motivi si consiglia un approccio alternativo che eviti confusione tra politiche e operazioni, fornendo un più alto livello di formalismo che riesca facilmente a catturare, rappresentare, eseguire e mantenere la logica di business: il paradigma basato su regole [36].

Una regola di business è un affermazione compatta, atomica, ben formata, relativa ad un aspetto del business; può essere espressa in termini direttamente correlati al dominio, utilizzando un linguaggio semplice e non ambiguo, che sia accessibile a tutte le parti interessate: business owner, analista di business, architetto tecnologico, cliente, ecc. [37].

2.3.2 Business Rules Management System

I *Business Rule Management System (BRMS)* sono quei sistemi usati per definire, dispiegare, eseguire, monitorare e mantenere le regole di business presenti nei sistemi operazionali di organizzazioni e aziende. Questa logica è composta da politiche, requisiti e condizioni che sono usate per determinare le azioni da intraprendere da parte delle applicazioni e sistemi, in maniera autonoma e sulla base degli eventi che si verificano. In generale un BRMS deve includere almeno:

- un repository esterno rispetto alle applicazioni contenente le regole di business e quindi la logica decisionale;

- dei tool che permettano la definizione della logica sia da parte di utenti tecnici che di dominio;

- un motore di esecuzione che permetta alle applicazioni di invocare la logica decisionale prevista.

Il motore di regole deve elaborare i singoli eventi per valutare l'attivazione delle regole. Uno schema dei componenti è rappresentato in Figura 2.9.

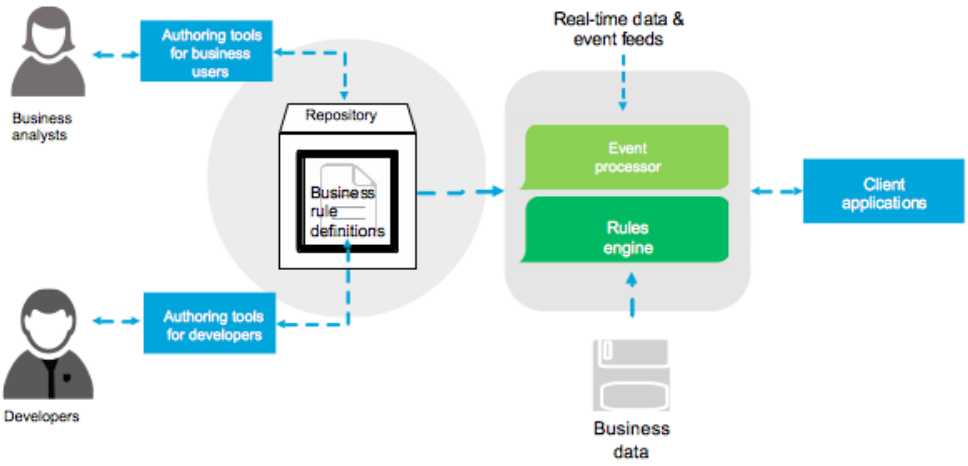

Figura 2.9: Componenti di un Business Rules Management System (http://www.schabell.org/p/jboss-bpm-suite-starter-kit.html)

Secondo Ian Graham [37], utilizzando un BRMS si hanno i seguenti vantaggi:

- sviluppo più veloce;
- manutenzione più veloce;
- audit più chiaro;
- componenti della logica di business maggiormente riutilizzabili;
- maggiore consistenza tra organizzazioni;
- miglior allineamento e comprensione tra business e IT.

Tuttavia, i BRMS sono accompagnati anche da insidie:

- fare debugging dei sistemi con migliaia di regole è problematico;
- insiemi di regole ampi e segmentati male rende difficile la gestione;
- più regole sono aggiunte per gestire casi particolari o eccezionali più la consistenza globale e la capacità di selezionare la regola da attivare viene minata.

2.3.3 Algoritmo Rete

I motori di regole utilizzano un processo di inferenza per giungere ad una conclusione valida a partire dalle premesse. Il funzionamento dell'inferenza è classificabile in tre tipologie le cui implementazioni si basano su un meccanismo di matching: *forward chaining (data driven), backward chaining (goal driven) e mixed*. Il diagramma in Figura 2.10 illustra graficamente la differenza.

Nella maggior parte dei motori di regole, la capacità di inferenza è fornita attraverso l'algoritmo Rete più o meno ottimizzato, efficace nel risolvere problemi di matching molti a molti nell'intelligenza artificiale. Rete è implementato costruendo una rete di nodi, ognuno dei quali rappresenta uno o più regole, in fondo alla rete ci sono i nodi che rappresentano la singola regola. I fatti sono processati da questa rete e se giungono in fondo significa che hanno passato tutti i filtri fino a giungere ad una regola particolare, che corrisponde all'attivazione [37].

Di seguito un semplice esempio per chiarire il meccanismo preso dalla documentazione di *Jess*, un motore di regole open source. Le due regole (definite nel Codice 2.1) vengono compilate nella rete in Figura 2.11. Dove i nodi marcati come x?, y? ecc. (a singolo input), controllano se un fatto contiene i dati previsti mentre i nodi marcati come "+" (a doppio input) ricordano i fatti provenienti da sopra e si attivano ogni volta che ricevono i dati da entrambi i genitori. Ogni nuovo fatto viene presentato in cima alla

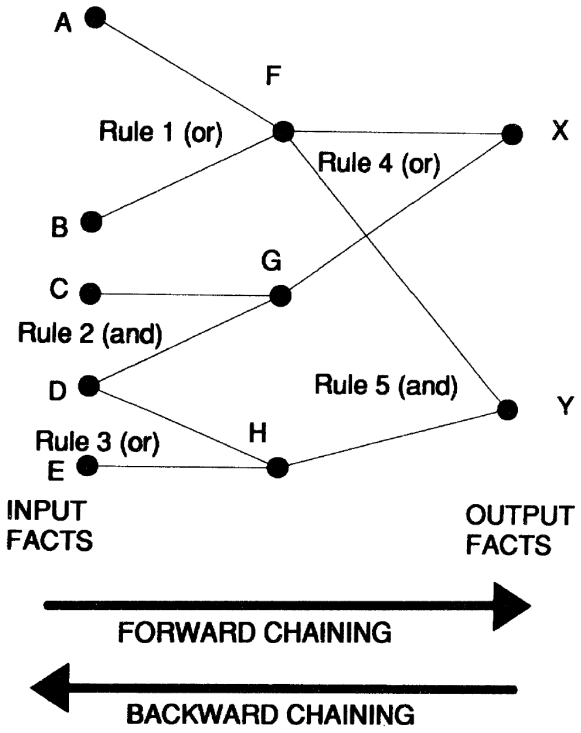

Figura 2.10: Insieme di fatti collegato da regole [38]

rete e spedito verso il basso. Ogni nodo a singolo input riceve il fatto, fa la sua valutazione e se è valida la passa al nodo successivo, altrimenti non fa nulla. Ogni nodo a doppio input integra i fatti da entrambi i genitori.

Questo meccanismo permette di valutare un predicato dichiarativo rispetto un insieme di fatti che cambia dinamicamente in real time, più efficacemente rispetto ai costrutti equivalenti di if/then/else o select/case. Al crescere delle regole questo vantaggio dell'algoritmo aumenta.

Codice 2.1: Esempio regole Jess

```
(defrule example-2
(x)
(y)
(z)
=> )

(defrule example-3
(x)
(y)
 => )
```

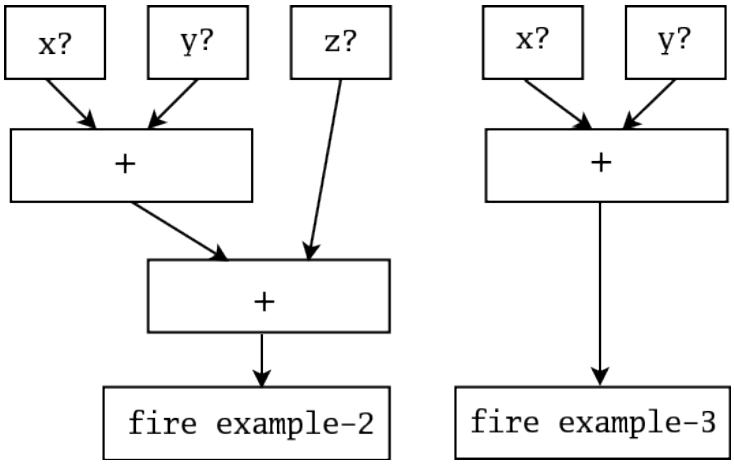

Figura 2.11: Rete compilata a partire dalle regole nel codice di esempio 2.1

Capitolo 3

Progetto GEAR

In questo capitolo viene descritto il prototipo del progetto di tesi nello specifico, a cui è stato dato il nome di GEAR (Gather, Enrich, Assess, React). Questi termini corrispondono alle fasi che compongono il modello. Partendo da quest'ultimo, è stato poi possibile definire i componenti architetturali e la pipeline di operazioni che il sistema esegue in Sezione 3.1. In Sezione 3.2 è stato definito lo stack tecnologico completo, tenendo conto del vincolo di utilizzare strumenti open source e un approccio di arricchimento basato su OSINT. Infine, in Sezione 3.3 si descrivono nel dettaglio tutti i singoli moduli che compongono il sistema e che rispecchiano il modello e l'architettura definiti.

3.1 Architettura

La struttura del progetto prevede il concetto di pipeline di passaggi, costruita partendo da un modello formale di ciò che concettualmente il sistema deve fare. Per questo si definisce il modello GEAR e a partire da questo, la pipeline di esecuzione sul flusso dei dati.

3.1.1 Modello concettuale GEAR

Il prototipo artefatto di tesi è costruito utilizzando come base i criteri di un sistema di OFD, definiti nella Sottosezione 2.1.1. Si tratta quindi di un prototipo che esegue analisi e valutazioni in tempo reale di eventi, integrandoli ad ulteriori informazioni storiche o da altre fonti. Se valuta eventi riconducibili ad anomalie o frodi deve reagire di conseguenza. Si estrapolano quindi diverse fasi nel funzionamento del sistema:

1. raccolta di eventi (Gather);
2. arricchimento con informazioni esterne (Enrich);
3. valutazioni sull'evento (Assess);
4. reazione conseguente (React).

Riguardo l'arricchimento il sistema ha come requisito quello di sfruttare OSINT oltre che sorgenti interne all'organizzazione. In questa fase vengono allegate nuove informazioni all'evento in analisi che possono essere utili nelle fasi successive.

Per quanto concerne la valutazione del rischio in termini di intenzionalità di un evento, è emerso dalla Sottosezione 2.2.1 che le tecniche utilizzate in letteratura sono basate su profili comportamentali; sia definendo pattern di normalità, sia definendo pattern fraudolenti, nello specifico Anomaly Based e Misuse Based. Per questo si è deciso di scomporre la fase di valutazione in due sottofasi in cui la seconda si occupi della decisione finale, valutando pattern di abuso e il risultato della precedente; sono chiamate rispettivamente: elaborazione del rischio e decisione tramite regole.

- La fase di elaborazione del rischio utilizza algoritmi di intelligenza artificiale (ad esempio, tecniche di machine learning) ma l'algoritmo specifico può cambiare totalmente da dominio a dominio. In questa fase è necessario utilizzare dati storici globali o individuali a seconda delle necessità, per costruire un modello algoritmico specifico per il caso

d'uso. L'obiettivo di questa sotto fase è quello di elaborare un risultato utile che viene allegato alle informazioni dell'evento, partecipando in questo modo alla fase di decisione finale (che porta ad una reazione). Alcuni esempi di risultato possono essere: una probabilità del rischio o una classificazione vera e propria.

- La fase di decisione tramite regole utilizza pattern pre-definiti da esperti di dominio e il sistema deve gestirne facilmente la continua evoluzione (riadattamento a nuovi comportamenti). Questi pattern sono parte della logica di business del sistema e per facilitarne lo sviluppo, l'evoluzione e la manutenzione si è deciso di estrarli, definendoli all'interno di un repository di regole. Gli eventi sono valutati nel loro complesso utilizzando le informazioni presenti in origine e quelle aggiunte dalla fase di elaborazione del rischio. Quest'ultimo passaggio può portare al calcolo di un punteggio di rischio dell'evento che riassuma tutti gli indicatori precedenti (compresa l'elaborazione del rischio).

Infine è presente la fase di reazione automatica a partire da una valutazione finale (punteggio di rischio). Il funzionamento di questo componente è basato sulle regole come nella fase di decisione. Si definiscono le soglie di reazione rispetto alla valutazione finale sempre attraverso delle regole. Opzionalmente si potrebbe allegare alle informazioni dell'evento anche le azioni intraprese. Per comodità si può quindi collassare la fase di decisione assieme a quella di reazione, dato che tutto avviene attraverso regole.

Questo modello è stato chiamato GEAR (Gather, Enrich, Assess, React) e di conseguenza il progetto collegato. Seppur non applicato nel prototipo, il modello può prevedere la possibilità di elaborazioni del rischio parallele, permettendo l'aggiunta di ulteriori elaborazioni con diversi algoritmi. La conclusione deve prevedere però il modulo di decisione, che a quel punto considera più elaborazioni. Alla fine del flusso, l'evento valutato viene salvato in un archivio perché ha acquisito valore nelle varie fasi che ha attraversato e può quindi essere riutilizzato. Vedi in Figura 3.1.

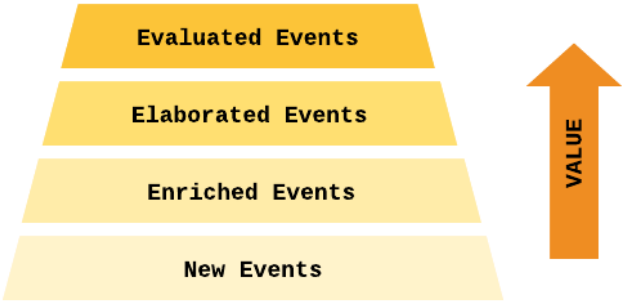

Figura 3.1: Crescita del valore informativo degli eventi nel sistema

Un'ultima nota è necessaria in questa sezione perché il modulo di elaborazione del rischio indica effettivamente la probabilità che un evento sia non intenzionale. Quest'informazione viene poi considerata nella fase di decisione. Perché aggiungere il componente di elaborazione? Cosa succede se emergono giudizi contrastanti tra modulo di elaborazione del rischio e modulo di decisione?

A livello concettuale questi due moduli sono stati disaccoppiati per garantire maggiore espressività nell'analisi, al fine di valutare aspetti impliciti e storici oltre che espliciti e sul singolo caso. Il modello prevede e assume che la fase di elaborazione sia una garanzia aggiuntiva rispetto alle regole ma che questa non basti da sola. La fase di decisione (definita da esperti di dominio) valuterà quanto peso dare alla fase di elaborazione sulla base di considerazioni come ad esempio: efficacia dell'algoritmo di machine learning e fiducia che l'utilizzatore ripone nelle singole fasi della pipeline.

Inoltre, il fatto stesso che il modulo di elaborazione del rischio dia un risultato contrastante con quello che darebbe il modulo basato su regole, può risultare molto significativo per scatenare reazioni specifiche a questo livello di incertezza. Come per esempio la richiesta di autenticazione forte all'utente o la valutazione da parte di un operatore umano.

3.1.2 Pipeline di trasformazione degli eventi

Il modello concettuale individua delle fasi di trasformazione distinte che vengono attraversate sequenzialmente dagli eventi. Questa concatenazione di elaborazioni può essere ben rappresentata da un'architettura data flow, nella quale l'eseguibilità e l'esecuzione di un istruzione è solamente determinata dalla presenza di un argomento in input [39]. Data la forte eterogeneità dei meccanismi e degli algoritmi nelle varie fasi di trasformazione, si è deciso di estrarli in moduli completamente disaccoppiati. Questa scelta permette di avere flessibilità nello sviluppo, nell'estendibilità, nell'evoluzione e nella manutenzione delle varie fasi.

Disaccoppiando le fasi in moduli è necessario però definire un pattern di comunicazione che permetta a questi di reagire agli eventi, eventualmente dividendo il flusso dei dati. Il pattern publish/subscribe è stato quindi scelto perché fornisce queste caratteristiche e una forte scalabilità. Questo pattern prevede il concetto di *topic* (un argomento) che può essere mappato rispetto alla fase che ha attraversato un evento. Si avranno quindi quattro topic: nuovo, arricchito, elaborato e valutato.

Le sorgenti degli eventi possono essere multiple e alla fine del flusso è necessario salvare gli eventi valutati in un archivio persistente in maniera trasparente. Per quest'ultima funzione è stato aggiunto quindi un ulteriore modulo. Utilizzando un flusso di questa tipologia si sta tendendo all'architettura Big Data detta Kappa, perché avviene il salvataggio in un sistema di archiviazione a lungo termine alla fine del flusso. Quindi in caso di rielaborazioni, queste potrebbero essere effettuate attraverso il *replay* dei dati dallo storage verso la pipeline. Non è a tutti gli effetti l'architettura perché un sistema di elaborazione batch non è implementato o previsto, almeno in questa fase iniziale.

Ogni modulo necessita poi di componenti aggiuntive oltre al collegamento ai topic:

- il modulo di arricchimento utilizza dei servizi esposti da API;

- il modulo di elaborazione necessità di un modello dell'algoritmo utilizzato;
- il modulo di decisione e reazione necessita di regole definite da esperti di dominio;
- il modulo di salvataggio deve potersi collegare ad un database.

Il flusso di trasformazione è visibile in Figura 3.2 ed è descritto di seguito:

1. multipli sorgenti generano gli eventi che vengono aggiunti in New;
2. il modulo di arricchimento legge da New, utilizza dei servizi esterni per aggiungere informazioni all'evento e scrive l'evento con le nuove informazioni in Enriched;
3. il modulo di elaborazione legge da Enriched, utilizza un modello precedentemente addestrato per dare un risultato e scrive l'evento con il risultato in Elaborated;
4. il modulo di decisione e reazione legge da Elaborated, definisce un punteggio di rischio utilizzando delle regole (codifica come regole anche le soglie di rischio per cui attivare processi proattivi) e scrive l'evento con la valutazione finale in Evaluated;
5. il modulo di salvataggio legge da Evaluated e scrive su una base di dati.

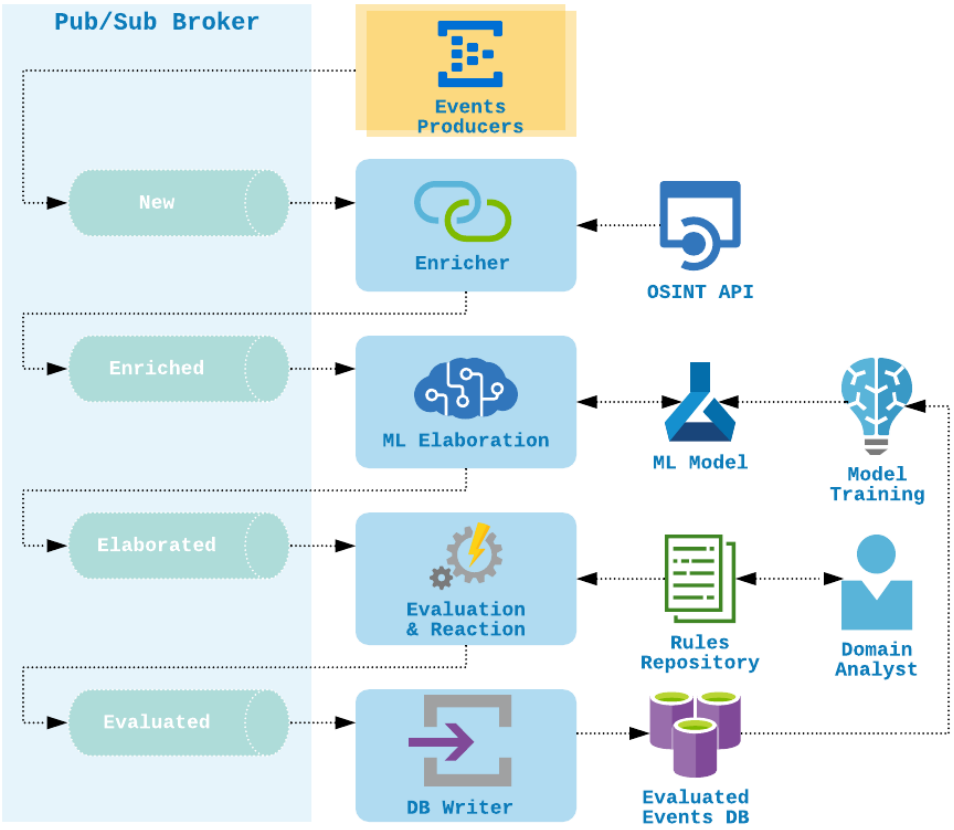

Figura 3.2: Architettura del sistema progetto di tesi

3.2 Tecnologie

La scelta dello stack tecnologico rispetta due importanti vincoli: il sistema deve essere costruito con tecnologie open source e deve elaborare Big Data in streaming. Di seguito si illustrano le tecnologie utilizzate e le motivazioni di scelta. Un diagramma dello stack tecnologico mappato sull'architettura è visibile in Figura 3.3.

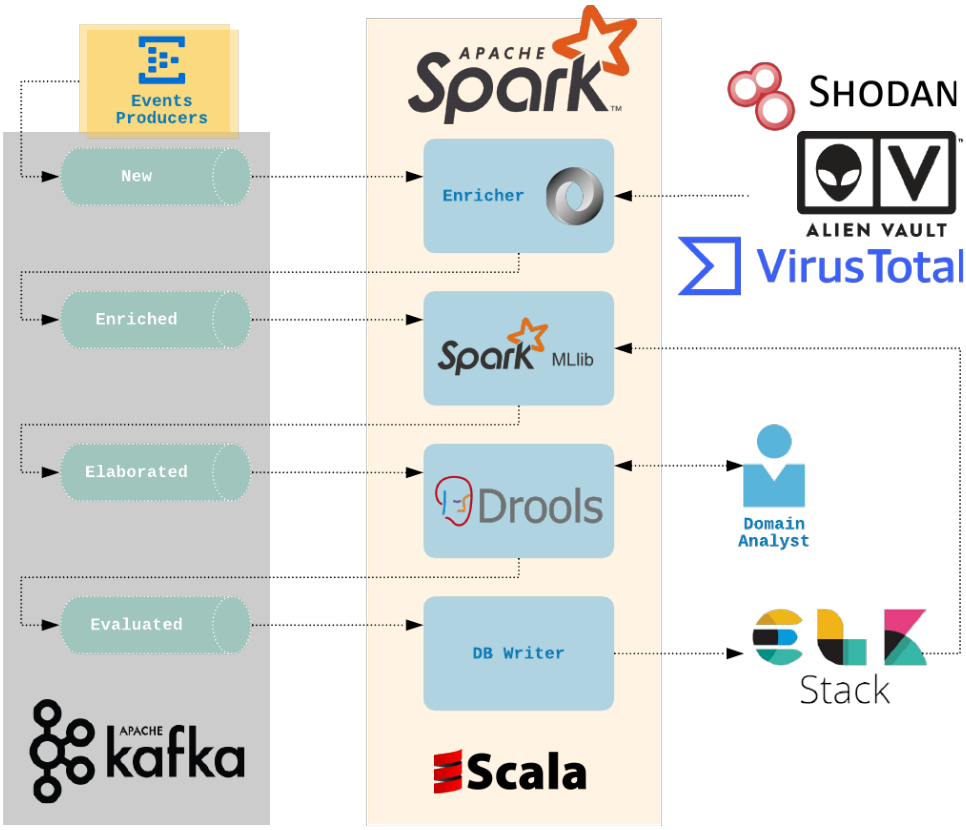

Figura 3.3: Stack tecnologico del sistema progetto di tesi

3.2.1 Apache Kafka

Apache Kafka è una piattaforma distribuita di streaming scalabile, tollerante ai guasti ed efficiente, che permette di costruire pipeline di streaming di dati

[40]. La piattaforma permette di pubblicare o sottoscriversi a degli stream di record, salvarli in maniera resistente ai fallimenti e processarli nel momento in cui avvengono.

Kafka è distribuito su un cluster di una o più macchine e salva gli stream di record in topic: un record è composto da una chiave, un valore e un timestamp; un topic è una categoria dove sono pubblicati i record e permette di avere sottoscrittori multipli. Per ogni topic il cluster mantiene un insieme di partizioni, dove vengono inseriti i record in maniera ordinata e immutabile. I record pubblicati sono persistenti per il periodo di validità previsto nella configurazione, indipendentemente che vengano consumati o meno.

Le partizioni sono fondamentali per quanto riguarda la scalabilità di un topic e la resistenza ai fallimenti perché possono essere replicate e distribuite su più macchine. Infatti, una partizione deve necessariamente essere sulla stessa macchina ma un topic può essere composto da infinite partizioni. All'interno di ogni partizione, i record sono individuati da un offset che li identifica univocamente; grazie alla combinazione di topic, partizione e offset un consumer può quindi leggere i record. Per ogni partizione è presente un server definito come "leader" che si occupa di gestire tutte le letture e scritture e, zero o più server aggiuntivi "followers" che replicano passivamente il leader. In caso di fallimento del leader uno dei follower viene automaticamente eletto.

Quando si valuta una piattaforma di comunicazione è importante valutare anche le garanzie messe a disposizione. Kafka garantisce che:

- venga mantenuta la semantica di ordinamento, ossia il consumer riceve i record di un particolare topic nello stesso ordine in cui sono stati inviati dal producer;
- quando più producer scrivono su stesso topic e stessa partizione, l'ordinamento avviene attraverso il timestamp di scrittura;
- per un topic con fattore di replicazione N, Kafka tollera fino a N-1 fallimenti dei server senza perdita di record.

Oltre all'infrastruttura, i producer e i consumer, Kafka fornisce una libreria client per gestire processare e analizzare gli stream chiamata Kafka Streams. Le caratteristiche di questa API sono le seguenti:

- leggera e semplice, può essere integrata in qualunque applicazione in Java con altre librerie;
- utilizza il modello di partizionamento di Kafka per ridimensionare orizzontalmente l'elaborazione;
- supporta la semantica di processo exactly-one garantendo che ciascun record sarà processato una e solo una volta, anche in caso di fallimenti di altri client o broker;
- diverse astrazioni nelle API, sia ad alto livello che primitive.

La libreria si basa sull'astrazione dello stream, ovvero un insieme di dati illimitato e in continuo aggiornamento dove i singoli record sono definiti come una coppia chiave-valore. La logica computazionale sui record è modellabile da un grafo i cui collegamenti sono gli stream e i nodi sono i *processor*, i quali rappresentano i passaggi di trasformazione dei record. Esistono due processor particolari all'interno della topologia del grafo:

- Source processor, che non hanno processor precedenti a loro ma leggono direttamente da un topic Kafka inoltrando ai processor successivi.
- Sink processor, che non hanno processor seguenti a loro ma scrivono direttamente su un topic Kafka ciò che hanno ricevuto dai processor precedenti.

In conclusione Kafka possiede le giuste caratteristiche per abilitare la costruzione di una pipeline di streaming real-time affidabile, scalabile e replicata che preveda applicazioni reattive di analisi e trasformazione. Questo perché fornisce sia la piattaforma adatta per la comunicazione sia le librerie per fare interagire in maniera reattiva i moduli del sistema.

3.2.2 Apache Spark

Spark è un motore di esecuzione veloce e general purpose che permette di eseguire query di analisi interattive in memoria su un cluster di macchine. L'architettura di Spark è di tipo master/slave con un coordinatore centrale (driver) e worker distribuiti (executors), dove ogni componente è un processo Java indipendente.

Spark è compatibile con il sistema di archiviazione di Hadoop e permette diversi modelli di elaborazione, come per esempio machine learning, query SQL e streaming. La forte efficienza del motore dipende dall'utilizzo della memoria rispetto al disco e dalle ottimizzazioni della pipeline di operazioni che effettua sulla logica dell'applicazione.

Spark si appoggia a due astrazioni principali, la prima è il Resilient Distributed Dataset (RDD) e la seconda è il Direct Acyclic Graph (DAG). Gli RDD rappresentano un insieme di dati su cui si opera in parallelo e in maniera distribuita perché internamente sono caratterizzati da queste proprietà:

- resilienza, che prevede la ricostruzione automatica del RDD in caso di fallimenti;
- distribuzione, attraverso la suddivisione in partizioni sparse tra i nodi del cluster (permette scalabilità);
- immutabilità, ovvero che a seguito della creazione non è possibile modificarli;
- valutazione lazy, ovvero che fino a ché non emerge la necessità di un risultato non avvengono elaborazioni sui dati (questo permette a Spark di fare ottimizzazione delle trasformazioni degli RDD prima di eseguire effettivamente le elaborazioni);
- salvabili in cache per essere riutilizzati più efficientemente;
- il tipo dei dati è inferito e non dichiarato.

Il DAG è la sequenza di operazioni eseguita sull'insieme dei dati e questa sequenza è rappresentabile da un grafo. Ogni nodo corrisponde ad un RDD mentre ogni arco, ad una trasformazione che viene effettuata da un RDD ad un altro. Il grafo creato è aciclico perché le trasformazioni non possono riportare ad un RDD vecchio, da qui il nome. Basandosi sull'applicazione definita dall'utente e dai collegamenti tra RDD, Spark calcola quindi un piano di esecuzione logico che successivamente viene trasformato in un piano di esecuzione fisico.

Il progetto di tesi necessita di moduli diversi tra di loro (machine learning, streaming, in sviluppi futuri eventuali batch) e Spark risulta una delle piattaforme più complete a livello di funzionalità per questo è stato scelto, anche in ottica di sviluppi futuri. Infine Spark può essere utilizzato con i linguaggi Scala, Java e Python, questi ultimi due molto popolari.

Spark Streaming

Spark Streaming è un estensione delle API di Spark, che permette l'elaborazione di stream di dati in real-time in maniera scalabile, tollerante ai fallimenti e con alto throughput [41]. Nello specifico permette di esprimere algoritmi complessi basandosi su funzioni ad alto livello, ad esempio map, reduce e join. Inoltre lo stream di dati può essere letto da diverse tecnologie sorgenti in maniera accessibile, tra cui Kafka. I dati così elaborati possono essere poi salvati su filesystem e database oppure inoltrati ad ulteriori piattaforme di comunicazione come Kafka.

Il funzionamento è basato sulla suddivisione in batch di dati dello stream in arrivo in real-time, ogni batch è poi processato dal motore di Spark e lo stream di output è quindi composto da batch di dati elaborati. Le API però permettono di lavorare sullo stream di dati continuo direttamente, utilizzando un'astrazione ad alto livello chiamata discretized stream (DStream). Internamente un DStream è composto da una sequenza di RDD, l'astrazione di base utilizzata da Spark normalmente. Nel progetto è stato scelto inizialmente per i seguenti motivi:

- soluzione flessibile che permette anche elaborazioni batch;
- facilità di integrazione con gran parte delle tecnologie Big Data presenti sul mercato;
- ampiamente adottato e supportato.

Un concorrente interessante è Flink [42], costruito con l'ottica di elaborare direttamente ogni record dello stream, minimizzando quindi la latenza il più possibile, a differenza di Spark Streaming che crea batch di dati. Flink fornisce garanzie simili a Spark Streaming e quindi risulta una soluzione alternativa e interessante.

Structured Streaming

Structured Streaming è un motore alternativo e nato successivamente a Spark Streaming per l'elaborazione di stream. Il motore è scalabile, tollerante ai fallimenti e costruito su Spark SQL [43]. Il modello si basa quindi sulle API Dataframe e Dataset che permettono di definire le elaborazioni come query SQL continue. Le differenze con Spark Streaming sono che [44]:

- ogni risultato dell'elaborazione di un record è aggiunto all'interno di una tabella illimitata (si perde il concetto di batch in uscita e DStream);
- Structured permette l'utilizzo dei DataFrame, un'astrazione più completa rispetto agli RDD;
- permette di processare i dati basandosi sul event-time se il timestamp è incuso tra i dati ricevuti (in Spark Streaming sono processati nell'ordine con cui arrivano al motore);
- fornisce garanzie maggiori per la gestione degli errori.

Nel progetto è subentrato successivamente a seguito di ricerche, al posto di Spark Streaming per alcuni moduli. Questo perché più orientato alle analisi real-time, più moderno, con astrazioni user-friendly che riducono la quantità

di codice necessaria per lo sviluppo e una maggiore facilità d'utilizzo. Inoltre, la libreria Spark MLlib utilizza come API principali i DataFrame invece che gli RDD dalla versione 2.0.

Spark MLlib

MLlib è una libreria machine learning di Spark il cui obiettivo è rendere facile e scalabile l'utilizzo degli algoritmi di machine learning [45]. Oltre agli algoritmi comuni di machine learning per classificazione, regressione e clustering fornisce anche strumenti per:

- il pre-processing del dataset (estrazione di attributi, trasformazione, selezione e riduzione di dimensionalità);
- fare pipeline per costruzione, valutazione e tuning delle pipeline machine learning;
- salvare in maniera persistente e caricare algoritmi, modelli e pipeline stesse;
- usare utility riguardanti algebra lineare e statistica.

L'astrazione principale su cui si basa la libreria è la Pipeline, che fornisce un insieme di API ad alto livello costruite sui DataFrame. Attraverso queste API è possibile combinare diverse operazioni e algoritmi machine learning in un flusso di lavoro unico. Oltre ai DataFrame emergono i seguenti concetti:

- Transformer, ovvero un algoritmo che trasforma un DataFrame in un altro (un modello ML non è altro che un Trasformer che trasforma un DataFrame con delle feature in un DataFrame con le predizioni);
- Estimator, ovvero un algoritmo che a partire da un DataFrame costruisce un Transformer (un algoritmo di learning è un Estimator che viene addestrato su un training set di dati e produce un modello).

Il Trasformer può quindi eseguire dalle semplici operazioni di mappatura o aggiunta di colonne fino ad arrivare all'applicazione di modelli machine learning. L'API principale del Transformer è la seguente: `DataFrame transform(Dataframe)`. L'Estimator astrae il concetto di algoritmo che viene addestrato con i dati, infatti l'api posseduta è concettualmente la seguente: `Transformer fit(DataFrame)`. La Pipeline non fa altro che concatenare multipli Transformer e Estimator in un unico flusso.

Fondamentali quando si applicano tecniche di machine learning, sono le fasi di estrazione, trasformazione e selezione delle features sui dati prima di applicare i modelli. L'estrazione è utile per fare emergere attributi significativi a partire da dati raw. La trasformazione permette di creare nuovi attributi usando quelli esistenti o modificandoli, in maniera che siano più significativi per il problema (ad esempio normalizzazioni, proiezioni e trasformazioni di attributi categorici in numerici). La selezione permette l'utilizzo di attributi specifici per un determinato problema riducendo eventualmente anche la dimensione dei dati in input. MLlib mette a disposizione funzioni che permettono di eseguire queste tre tipologie di operazioni.

MLlib è stata la scelta più naturale e immediata utilizzando Spark come motore di elaborazione dello stream di dati. Un'alternativa interessante e da valutare è Sparkling Water [46], un integrazione tra Spark e la piattaforma open source H2O, leader per quanto riguarda il machine learning distribuito in-memory. Integrando questi due ambienti open source è possibile eseguire query usando Spark SQL, inserire i risultati in H2O per costruire un modello e fare predizioni, reinserire poi i risultati in Spark.

3.2.3 Drools

La decisione e reazione deve avvenire attraverso un approccio a regole, implementabili attraverso un BRMS. Questi sistemi sono però concepiti tipicamente per lavorare su una singola macchina. Questo limite potrebbe impattare le performance nel caso di grandi quantità di fatti da valutare come potrebbe avvenire nel contesto di Big Data.

In letteratura viene proposta una soluzione chiamata SparkRE, per supportare il ragionamento basato su regole sui big data. SparkRE utilizza l'astrazione del DataFrame presente in Spark per rappresentare la memoria di lavoro distribuita che contiene i fatti [47]. Utilizza SparkSQL per implementare un efficiente matching delle regole spostando il calcolo sui dati (fatti) e non viceversa. Utilizzando Spark, il sistema eredita tutte le qualità della tecnologia come: scalabilità e tolleranza ai fallimenti. Questa soluzione risulta interessante e definisce anche l'algoritmo per il motore di inferenza ma non è un prodotto implementato e disponibile. La scelta di questa soluzione avrebbe significato attuare un implementazione custom perdendo di fatto alcuni vantaggi per gli utenti non tecnici derivanti dalla scelta di un BRMS esistente.

La scelta è quindi ricaduta su Drools, un BRMS sviluppato completamente in Java e open source ampiamente utilizzato. Il fatto che sia stato sviluppato in Java significa la possibilità di utilizzarlo su Spark (almeno a livello teorico), sfruttando i vantaggi di entrambe le piattaforme. Durante le ricerche è emerso un articolo [48] relativo ad una POC (proof of concept) relativamente a questa opportunità. La POC prevede di distribuire su ogni nodo worker un istanza di Drools che applichi le regole definite sui fatti che riceve.

Drools prevede un motore di regole basato su inferenza forward-chaining e backward-chaining (utilizza l'algoritmo Rete). Questo BRMS permette una valutazione veloce e affidabile delle regole di business e l'elaborazione di eventi [49]. Un BRMS prevede necessariamente un motore di regole, un blocco fondamentale per la costruzione di un sistema esperto, necessario in questa tesi per facilitare certi tipi di analisi e l'esecuzione di reazioni. La funzione di base del motore è quella di fare matching dei dati o dei fatti in arrivo rispetto alle regole, determinando quali regole e come eseguirle. Oltre alle regole e fatti, il motore prevede due memorie, una di produzione dove sono salvate le regole e una di lavoro dove finiscono i fatti. Un ulteriore componente è l'Agenda, dove vengono registrate e ordinate le regole che sono state attivate,

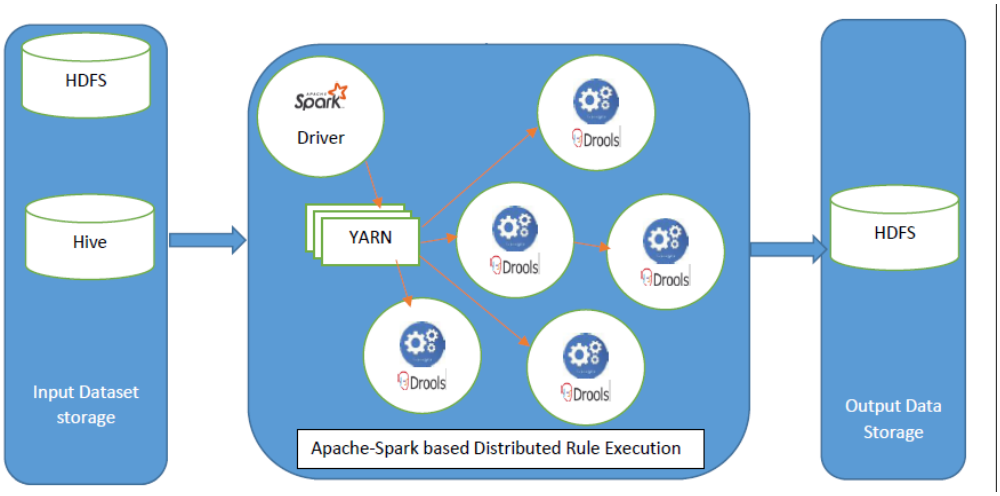

Figura 3.4: Diagramma dell'integrazione Spark-Drools dall'articolo [48]

in preparazione dell'esecuzione. Le aggiunte e gli aggiornamenti delle regole da parte di utenti o sistemi automatici sono inserite nella memoria di lavoro come fatti e questo permette di modificare la logica dinamicamente.

Il funzionamento del framework è basato sul concetto di Knowledge Session (KieSession), che mantiene le risorse richieste per eseguire le regole. I fatti vengono inseriti nella sessione e si eseguono le regole che fanno match. La Knowledge Base rappresenta invece la conoscenza nell'ecosistema Drools, ovvero contiene le informazioni sulle risorse dove sono definite le regole. La KieSession non è fornita direttamente dalla Knowledge Base ma dai Container (KieContainer) che sono creabili da KieServices sulla base delle informazioni della Knowledge Base. KieServices è un singleton thread-safe che ha funzioni di hub per esporre tutti i servizi di Drools.

La definizione delle regole avviene attraverso un file di estensione `.drl` in cui sono elencate le regole costruite nel formato when-then. Quando le condizioni in When sono rispettate allora si eseguono le azioni definite in Then.

Oltre ad usare direttamente il linguaggio messo a disposizione da Drools è possibile appoggiarsi alle Decision Table, che permettono di definire le regole utilizzando una tabella Excel pre formattata (utile per gli utenti non tecnici).

	RuleSet	com.baledung.spring.drools.excel		
	Import	com.baeldung.spring.drools.model.Product		
	Notes	This decision table is for deciding label of product		
	RuleTable Routing			
	CONDITION	CONDITION	CONDITION	ACTION
	product:Product			
	product.type	product.name	product.label	product.setLabel("$param");
Rule Comments	type	name	label	label
Rule1	Electronic			BarCode
Rule2	Book			ISBN
Rule3	CD			VOLUMENo.

Figura 3.5: Esempio di Decision Table di Drools

Drools è un framework che può essere facilmente integrato in qualunque progetto basato su Java perché le regole si trovano nella directory delle `resources` che viene allegata all'eseguibile. In conclusione, questo BRMS è stato scelto perché uno dei più utilizzati e più maturo tra quelli open source. Risulta inoltre ben integrabile nello stack tecnologico basato su JVM perché costruito in Java. Infine fornisce le decision table per la definizione delle regole da parte di utenti non tecnici come potrebbero essere i domain analyst delle frodi bancarie (vedi Figura 3.5).

3.2.4 Tool OSINT

Il modulo di arricchimento del progetto deve utilizzare informazioni esterne per arricchire l'evento e questo, può essere effettuato attraverso sorgenti dati interne all'organizzazione come blacklist, classifiche di pericolosità per specifiche informazioni (es. un evento proveniente dalla Cina può essere più sospetto che uno proveniente dall'Italia per un determinato campo) e database di informazioni collegabili all'evento sulla base di un identificativo. Oltre a questo però, si è deciso di inserire all'interno della pipeline la ricerca di informazioni relative all'evento tramite strumenti di OSINT. I framework teorici e gli strumenti sviluppati per fare OSINT sono molto eterogenei e specifici per il

tipo di ricerca che si vuole effettuare. Una buona panoramica delle categorie e degli strumenti disponibili viene fornita nel sito https://osintframework.com/.

Data la specificità degli strumenti è importante identificare quali informazioni di partenza utilizzare per fare OSINT. Nel progetto si è deciso di concentrare gli sforzi su un'informazione presente nel caso d'uso ma ricorrente in molti domini applicativi distribuiti in rete: l'IP. Ciò non toglie l'estendibilità del sistema ad altri strumenti per l'ottenimento di altre informazioni. Oltre a questo, i tool scelti devono essere integrabili in maniera automatica al flusso di lavoro quindi è necessario escludere tutti gli strumenti manuali. Per questi motivi sono emersi i tre motori di ricerca seguenti che espongono i loro servizi anche tramite delle API.

Shodan

Shodan è un motore di ricerca pensato per i dispositivi di qualunque genere connessi direttamente Internet. A differenza dei motori di ricerca web come Google che trovano siti web (solo la superficie di tutto Internet [50]), Shodan raccoglie informazioni pubbliche direttamente sui dispositivi connessi. I dispositivi possono essere di qualunque genere: server, stampanti, webcam, semafori, telecamere di videosorveglianza e sistemi di controllo di qualunque genere [51].

Per indicizzare Shodan si basa principalmente su metadati e Service Banner riguardanti il software in esecuzione sul device e utilizzando queste informazioni, Shodan può ottenere per esempio:

- località geografica;
- utente e password di default;
- versione del software;
- porte di rete aperte;
- costruttore e modello.

Codice 3.1: Esempio di Service Banner per una connessione tramite Netcat

```
[root@macchina]# nc www.targethost.com 80
HEAD / HTTP/1.1

HTTP/1.1 200 OK
Date: Mon, 11 May 2009 22:10:40 EST
Server: Apache/2.0.46 (Unix) (Red Hat/Linux)
Last-Modified: Thu, 16 Apr 2009 11:20:14 PST
ETag: "1986-69b-123a4bc6"
Accept-Ranges: bytes
Content-Length: 1110
Connection: close
Content-Type: text/html
```

Oltre a questo le ricerche effettuate su Shodan permettono di visualizzare dati aggregati ed è possibile aggiungere filtri che variano dalla località geografica al sistema operativo in esecuzione. Questo rende possibile ricerche come:

- quali nazioni hanno più dispositivi connessi;
- quale versione di sistema operativo per server è la più popolare;
- quanti dispositivi sono affetti da una vulnerabilità appena scoperta.

Le informazioni ottenute tramite questo servizio possono essere applicate in diverse aree come la sicurezza di rete, ricerche di mercato, Internet-of-things e tracciamento di vulnerabilità.

VirusTotal

VirusTotal è un aggregatore di informazioni che combina risultati di differenti prodotti antivirus, motori di scansione online, dataset e contributi degli utenti con l'obiettivo di controllare sia file che url per individuare malware. Sul sito è possibile caricare qundi sia file che URL sospetti. Il servizio dal sito web è gratuito e mette a disposizione delle API però soggette a diverse condizioni. VirusTotal è stato acquistato nel 2012 dal gruppo Alphabet Inc. Nel progetto è stato considerato come una possibile alternativa per ricondurre

l'origine di un evento a un URL sospetto. Nella documentazione è possibile vedere la lista dei prodotti e servizi utilizzati dall'aggregatore [52].

AlienVault

AlienVault Open Threat Exchange (OTX) è una comunità aperta di intelligence per le minacce che permette di avere accesso a indicatori di valutazione e strumenti per individuare compromissioni, tutto derivante da contribuzioni della comunità. OTX permette la collaborazione aperta tra compagnie private, ricercatori di sicurezza indipendenti e agenzie governative per condividere le informazioni riguardo minacce e metodi di attacco.

Le informazioni raccolte da questa infrastruttura vengono messe a disposizione tramite API ed è possibile quindi analizzare un IP in modo che, oltre alle informazioni ottenute da altri aggregatori, si possano avere tutte quelle segnalazioni dirette e indirette effettuate dalla comunità riguardanti quell'endpoint.

3.2.5 Elasticsearch, Logstash e Kibana

Il progetto prevede che gli eventi, completata la pipeline, debbano essere salvati in una base dati per i seguenti motivi: analisi a posteriori, reportistica e preparazione del training set per l'addestramento dell'algoritmo machine learning. Ragionando in termini di Big Data è necessario che questa base dati sia distribuita per garantire scalabilità. Il modello relazionale offre un meccanismo transazionale che garantisce consistenza e accessi concorrenti, a scapito di bassa latenza e scalabilità. Questa problematica è spiegata dal teorema CAP, che indica come impossibile per una base dati distribuita garantire più di due caratteristiche delle tre seguenti [53]:

- consistenza, che deve essere garantita anche in scrittura;
- disponibilità, ogni richiesta riceve una risposta non di errore (senza garanzie);

- tolleranza al partizionamento, il sistema continua a funzionare nonostante il numero arbitrario di messaggi persi sulla rete tra i nodi.

Quando avviene un fallimento nella rete si possono attuare infatti due azioni: cancellare l'operazione minando la disponibilità per garantire consistenza oppure procedere con l'operazione garantendo disponibilità ma rischiando inconsistenza. Il teorema implica quindi una scelta tra consistenza e disponibilità in caso di partizionamento di rete. Utilizzando modelli NoSQL si favorisce bassa latenza e scalabilità a scapito della consistenza.

I modelli NoSQL prevedono inoltre un approccio schemaless, in contrapposizione rispetto alla rigidità del relazionale e questo, può risultare utile in certi domini applicativi che evolvono nel tempo. Esistono diversi modelli dati ma il caso d'uso si presta alla tipologia NoSQL documentale, sopratutto in ottica di granularità adatta ad analisi e reportistica.

Per il prototipo è stato scelto lo stack ELK (Elasticsearch, Logstash e Kibana) perché mette a disposizione un parco completo di strumenti per interagire con i dati. I componenti dello stack sono:

- Elasticsearch è un motore di ricerca e analisi, con funzioni di archiviazione, scalabile, distribuito, open source e RESTful, che utilizza il formato JSON;

- Logstash è un software per la gestione della pipeline di elaborazione dei dati che li inserisce da sorgenti multiple, li trasforma e li salva in uno stash come Elasticsearch;

- Kibana è una UI che permette agli utenti di visualizzare i dati presenti su ElasticSearch.

La base dati vera e propria dello stack è quindi Elasticsearch, il quale internamente utilizza Apache Lucene per contenere le informazioni attraverso un *inverted index* (questo lo differenzia da altre base dati NoSQL). Un inverted index è una struttura dati che contiene i collegamenti tra un contenuto, come

potrebbe essere una parola o un numero, rispetto la sua locazione all'interno di un documento o un insieme di documenti. Lucene è una libreria scritta completamente in Java che fornisce un motore per effettuare query testuali espressive attraverso delle API. Permette infatti ricerche accurate e efficienti attraverso algoritmi che permettono *ranked searching*, *query* di prossimità attraverso la distanza e ricerche attraverso indici multipli con fusione dei risultati [54].

La scalabilità e la disponibilità vengono garantite da Elasticsearch attraverso gli *shard*, ovvero insiemi più piccoli di dati appartenenti allo stesso indice; un indice corrisponde al concetto di Database nel modello relazionale. Di seguito il collegamento concettuale tra le entità di Elasticsearch e quelle di MySQL:

- MySQL → Database → Tabelle → Colonne/Righe;
- Elasticsearch → Indici → Tipi → Documenti con proprietà

Quindi l'indice è distribuito tra shard multipli e una query rispetto ad un indice viene eseguita in parallelo tra tutti gli shard (questo garantisce migliori performance che un approccio sequenziale). Il risultato da ogni shard viene poi raccolto e inviato al client. In Elasticsearch sono presenti due tipologie di shard, primari e repliche, il cui numero viene definito al momento della creazione dell'indice (quando si indica il numero di repliche si intende per ogni shard primario). Gli shard primari sono distribuiti su tutti i nodi e vengono utilizzati per le query parallele mentre le repliche, sono in nodi diversi rispetto allo shard primario che replicano (questo garantisce disponibilità).

3.3 Implementazione dei moduli

3.3.1 Modulo standard

In questo progetto, il modulo standard è una possibile base di partenza per costruire un componente della pipeline utilizzando Structured Streaming. Il

Codice 3.2: Struttura dei record letti da Kafka in uno stream Spark

```
root
 |-- key: binary (nullable = true)
 |-- value: binary (nullable = true)
 |-- topic: string (nullable = true)
 |-- partition: integer (nullable = true)
 |-- offset: long (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- timestampType: integer (nullable = true)
```

modulo prevede le funzioni di streaming in input tramite Kafka, elaborazioni sui DataFrame e streaming in output sempre tramite Kafka. Attraverso la sessione di Spark è infatti possibile leggere uno stream di dati da una sorgente come se fosse un DataFrame e allo stesso modo può essere scritto come stream in output.

L'inizializzazione degli stream da Kafka prevede dei parametri di configurazione obbligatori e altri opzionali che permettono una configurazione granulare del consumer Kafka. I parametri obbligatori sono i topic di sottoscrizione e la configurazione per connettersi ai bootstrap server di Kafka. Di fatto è possibile anche avere il topic di input e il topic di output su cluster Kafka differenti. Tra quelli opzionali si segnala il timeout per leggere i dati da Kafka e il numero di tentativi per riprovare a leggere i dati.

Quando si integra Kafka con gli streaming di Spark il record ottenuto dallo stream ha la struttura indicata nel Codice 3.2, dove la chiave e il valore sono in formato binario. La deserializzazione deve essere effettuata tramite operazioni sui DataFrame come per esempio un operazione di cast del campo valore a stringa. A seguito delle elaborazioni tramite DataFrame è necessario preparare il valore per essere scritto nello streaming in output ad un altro topic Kafka. Il connettore a Kafka ha però un requisito, il DataFrame deve infatti possedere le colonne chiamate "key" e "value", di tipo stringa o binario (se non specificata la key viene aggiunta automaticamente).

3.3.2 Modulo di arricchimento

Questo modulo è costruito a partire da quello standard e si occupa di aggiungere al record dell'evento le informazioni esterne. Nel caso specifico, utilizza sorgenti esposte tramite API (i tool di OSINT) che contatta tramite richieste HTTP. Il modulo è composto da un motore che espone gli arricchimenti possibili tramite interfaccia, che rispondono con oggetti del modello.

Il motore implementa quindi l'interfaccia degli arricchimenti e all'interno effettua delle chiamate ai tool di OSINT per poi creare oggetti dal modello elaborando la risposta ottenuta in JSON. Il motore non effettua delle chiamate HTTP direttamente ma è stato sviluppato in maniera trasparente. Infatti, le chiamate sono wrappate dalla sorgente di arricchimento che si utilizza e ognuna implementa il metodo di chiamata remota definito da un'interfaccia; questo permette al motore di fare la chiamata ad una classe che rappresenta la sorgente di arricchimento.

Le funzioni illustrate fin'ora sono autonome rispetto a Spark e per questo è necessario wrapparle tramite le User-Defined Function (UDF) di SparkSQL per utilizzarle su un DataFrame. Le UDF sono una funzionalità messa a disposizione per estendere le funzioni presenti nel vocabolario di Spark definendo nuove funzioni basate sulle colonne dei DataFrame. Se possibile è sempre meglio utilizzare le funzioni standard perché le UDF sono una scatola nera per Spark e quindi non può ottimizzarle. Per questo, il modulo può essere probabilmente ottimizzato in questo senso.

3.3.3 Modulo di elaborazione

Nel prototipo oggetto di tesi questo modulo è composto da un job di streaming per l'elaborazione e da uno batch utilizzato per la costruzione di un modello machine learning. Il componente streaming è costruito sempre partendo dal modulo standard definito precedentemente. Di seguito si illustra la struttura generale del modulo, senza scendere nel dettaglio dell'algoritmo specifico di machine learning utilizzato. Questo perché l'algoritmo dipende

fortemente dal dominio e il modulo è predisposto per essere collegabile a qualunque algoritmo disponibile nella libreria MLlib. Nel capitolo finale di valutazioni sul caso d'uso si scende nel dettaglio riguardo i possibili algoritmi ed emerge la loro forte relazione con il caso d'uso.

Costruzione del modello

In questo componente è necessario definire il formato del dataset che verrà utilizzato per costruire il modello perché necessario a configurare il DataFrame di input oltre che le elaborazioni. Nel caso d'uso in tesi si è utilizzato un file `.csv` ma le API di Spark rendono facile la configurazione della lettura da diversi formati. Per effettuare delle valutazioni significative si consiglia di suddividere il dataset utilizzato in una parte per l'addestramento e una parte utilizzato solo per la verifica a posteriori. Una volta disponibile il DataFrame di input è necessario operare sui singoli attributi per renderli adatti ad essere utilizzati nel modello di machine learning scelto. Se si applicano trasformazioni sulle feature attraverso funzioni nella libreria MLlib queste possono essere inserite nella pipeline, mentre si mantengono fuori le trasformazioni effettuate direttamente sul DataFrame tramite UDF.

L'espressività di MLlib è tuttavia molto ampia e permette di applicare algoritmi di pre-processing per ogni necessità, dalla normalizzazione alla creazione di una colonna vettoriale costruita sulla base di altri attributi.

Tra le funzioni in pipeline è necessario inserire anche l'algoritmo da utilizzare definendo le sue configurazioni come: le colonne di attributi da considerare e l'eventuale attributo di etichettamento nel caso di algoritmi di classficazione. A questo punto si può creare la pipeline concatenando tutte le operazioni da effettuare e sull'istanza generata si richiama il metodo `fit()` inserendo il dataset di addestramento. Il modello così generato viene salvato tipicamente su un file system distribuito come per esempio HDFS con tutte le informazioni per applicare la pipeline.

Su un modello creato è possibile poi eseguire tutte le valutazioni di performance tipiche degli approcci machine learning, sia sul training set sia sul

test set. Si può procedere definendo manualmente il calcolo di certi indicatori oppure si possono utilizzare librerie messe a disposizione da MLlib.

Elaborazione in streaming

Questo componente è quello effettivamente collegato alla pipeline di progetto principale, che comunica quindi tramite Kafka. Al DataFrame in input è necessario riapplicare quelle trasformazioni che non sono state inserite nella pipeline e bisogna assicurarsi che i nomi delle colonne corrispondano a quelli definiti durante la creazione del modello.

Si carica poi il modello precedentemente salvato e su di esso si può eseguire il metodo `transform()` con il DataFrame di input trasformato. Il metodo di trasformazione riapplicherà tutte quelle operazioni definite nella pipeline generando un nuovo DataFrame. Le trasformazioni derivanti dall'applicazioni di algoritmi machine learning veri e propri aggiungono automaticamente le colonne con le informazioni legate al risultato dell'algoritmo. A questo punto si modifica il DataFrame risultato per essere inserito nel topic Kafka di output come previsto negli altri moduli (vedi modulo 3.3.1).

3.3.4 Modulo di decisione e reazione

Questo è il modulo che completa la pipeline logica del modello e si deve occupare di fare una valutazione finale, intraprendendo eventuali azioni attive. A livello di implementazione la soluzione è stata adottata per funzionare su Spark Streaming; una migrazione su Spark Structured è consigliata per i vantaggi che fornisce ma non è ancora attuata. Il modulo prevede tre componenti: il job vero e proprio, una classe di valutazione che applica le regole e la classe di modello per il tipo di oggetto valutato nelle regole. Il modello rappresenta il record analizzato ed è fondamentale per agire attraverso le regole, queste infatti valutano sulla base dei nomi dei campi.

Il valutatore è stato implementato sfruttando i generici di Scala e in questo caso è fondamentale che la classe sia serializzabile per essere distribuita sui worker di Spark. Il job è stato costruito su Spark Streaming e prevede alcune

operazioni pre-valutazione, tra cui suddivisione della stringa (formato .csv) e la mappatura della stringa sul modello. Nell'implementazione attuale si utilizzano alcune regole di base che valutando le caratterisitche del record (tra cui l'elaborazione del modulo precedente) calcolano un punteggio di score agendo tramite un metodo sull'oggetto definito dal modello. Il record con il nuovo parametro di score viene poi scritto su un topic kafka attraverso un Producer Kafka implementato in Scala.

3.3.5 Modulo di salvataggio su base dati

Questo modulo finale si occupa di salvare tutti gli eventi letti dall'ultimo topic (Evaluated) in un archivio o una base dati distribuita. Il prototipo prevede un implementazione derivante dal modulo standard che, grazie all'espressività di Spark Structured, permette di salvare i dati dallo stream direttamente su ElasticSearch. Il modulo può essere esteso per operare su diverse tecnologie di salvataggio, eventualmente salvando in maniera parallela su storage differenti. Durante l'inizializzazione della sessione di Spark si inserisce la configurazione per comunicare con il cluster ElasticSearch, a questo punto lo stream viene letto normalmente da Kafka ma la scrittura avviene direttamente su ElasticSearch.

Capitolo 4

Valutazioni sul caso d'uso

Quest'ultimo capitolo illustra l'applicazione del prototipo al caso d'uso originario, ovvero l'individuazione delle frodi nell'online banking. Inizialmente è necessario comprendere le caratteristiche del dataset che si sta utilizzando facendo un'analisi degli attributi presenti, successivamente nella Sezione 4.2 si definisce la logica di evoluzione dei dati da modellare attraverso le varie fasi della pipeline del modello GEAR (dagli arricchimenti alla definizione delle regole di valutazione e reazione). Infine, nella Sezione 4.3 si valuta efficacia in termini di individuazione effettiva di frodi e efficienza in termini di performance del sistema per analizzare una quantità di transazioni predefinita.

4.1 Dataset

Il dataset fornito dall'azienda Imola Informatica Spa è utilizzato normalmente in fase di testing dei sistemi già in essere ma è costruito a partire da informazioni reali. Questo implica la necessità di oscuramento delle informazioni che potrebbero ricondurre a dati reali o clienti, per motivi di sicurezza e privacy. Gli identificativi degli utenti, il numero di carte, IBAN e i numeri di telefono sono stati mascherati per mantenere il valore informativo dell'informazione (ad esempio unicità di un ID), rendendo però ragionevolmente

difficile risalire al dato originario. Il vantaggio di questo dataset è l'alta qualità che possiede in termini di assenza di valori mancanti, grande dimensione e etichette. Infatti ogni transazione è stata etichettata come fraudolenta o meno da parte del sistema antifrode proprietario utilizzato dalla banca. Quest'informazione è allegata all'evento (attraverso un punteggio di rischio calcolato e l'azione intrapresa dall'antifrode). Il dataset è fornito in formato `.csv` e i record hanno la seguente struttura:

```
"TIPODISPOSIZIONE","IMPORTO","DESCRIZIONE","CONTOADDEBITO","
    ↪ CONTOACCREDITO","TIPOAUTENTICAZIONE","INDIRIZZOIP","
    ↪ RISKSCORE","ACTIONCODE","IDUTENTE","DATA"
```

Riguardo gli attributi si hanno queste informazioni da dominio:

- TIPO_DISPOSIZIONE, 4 possibili valori (bonifico italia, bonifico estero, ricarica cellulare, ricarica carte di credito)
- CONTOADDEBITO, sempre IBAN;
- CONTOACCREDITO, IBAN per i bonifici, numeri di telefono o di carta per le ricariche;
- IMPORTO, molto variabile, range da 0.01 a 10.000.000, media 2488.71 e scarto quadratico medio a 52938.47;
- TIPOAUTENTICAZIONE, 5 possibili valori (SMS, CSP, SCARD, SSCARD2, SPIN);
- INDIRIZZOIP, indirizzo di provenienza dell'operazione;
- RISKSCORE, punteggio definito dal sistema antifrode esistente;
- ACTIONCODE, azione intrapresa dal sistema antifrode esistente;
- DATA, formato YYYY-MM-DDTHH24:MI:SS.

In Tabella 4.1 si riporta la distribuzione dei record rispetto alla tipologia di disposizione, assieme agli utenti univoci per ogni sezione. I record totali risultano 696.519 e la maggior parte risultano bonifici verso l'Italia.

Tipo Disposizione	Transazioni	Utenti
BONIFICO_ITALIA	564.952	51.097
BONIFICO_ESTERO	22.423	4.342
RICARICA_CELLULARE	104.864	22.753
RICARICA_CARTA	4.280	2.387

Tabella 4.1: Distribuzione dei record in base al tipo di disposizione

Emerge come il dataset presenti una distribuzione distorta che prevede circa 3000 record identificati come frode. Come indicato nella Sottosezione 2.2.1 questa sfida è spesso prevista nel dominio dell'individuazione di frodi in generale ed è necessario adottare tecniche di bilanciamento. Nello specifico si è agito a livello di dato, sotto campionando la classe maggioritaria (non frode).

4.2 Mappatura sul modello GEAR

4.2.1 Arricchimento

Come già indicato più volte durante la tesi, l'arricchimento può avvenire in varie modalità, eventualmente parallelizzando meccanismi e tecniche diverse (sorgenti interne e esterne). In questo caso si è deciso di utilizzare l'indirizzo IP e sorgenti OSINT per aggiungere informazioni all'evento. Utilizzando infatti tool come Shodan, attraverso l'IP si riesce a risalire ad informazioni come:

- porte aperte;
- parole chiave associate ad un particolare IP;
- eventuali vulnerabilità conosciute o segnalate in passato;
- località geografica dell'IP.

Queste informazioni sono importanti per fare delle valutazioni del rischio esplicite all'interno del modulo finale della pipeline, ma possono portare contenuto informativo anche nella fase di elaborazione.

Si nota come l'utilizzo di sorgenti esterne esposte su Internet, possa aumentare la latenza dell'evento nell'attraversare l'intera pipeline. Questo ritardo derivante dalla richiesta HTTP su Internet, può risultare superiore a tutta l'esecuzione nella pipeline. Inoltre, richiamare servizi esterni porta a cedere parte del controllo sulla pipeline ad altre organizzazioni o sistemi. Per mitigare queste problematiche si possono adottare alcune strategie:

- ottimizzare il modulo per l'esecuzione del minor numero di richieste possibili;
- inserire un sistema di caching interno all'organizzazione;
- gestire eventuali fallimenti nell'arricchimento con sistemi di *fallback*.

4.2.2 Elaborazione del rischio

Lo studio del dominio ha indicato come la visione comportamentale sul singolo utente sia molto efficace ma allo stato attuale, questo non avviene. Nella prima implementazione infatti, è stato utilizzato l'algoritmo di regressione logistica per la classificazione delle frodi, utilizzando il dataset nella sua interezza e quindi, l'algoritmo agisce avendo una visione globale. Questa scelta è dettata dal fatto che si vuole dare un'idea di come utilizzare il modello GEAR e non analizzare il caso d'uso specifico in maniera profonda. Si effettua una fase di pre-processing sugli attributi che prevede questi passaggi:

- TIPODISPOSIZIONE, categorico e utilizzabile senza modifiche;
- IMPORTO, si trasforma in categorico per avere gruppi di quantità;
- DESCRIZIONE, utile in caso di text mining ma per la regressione viene rimossa;

- CONTOADDEBITO, se considerato il dataset globalmente questa informazione non da valore informativo all'algoritmo di regressione quindi può essere rimosso;
- CONTOACCREDITO, se considerato il dataset globalmente questa informazione non da valore informativo all'algoritmo di regressione quindi può essere rimosso;
- TIPOAUTENTICAZIONE, categorico e utilizzabile senza modifiche;
- INDIRIZZOIP, se considerato il dataset globalmente questa informazione non da valore informativo all'algoritmo di regressione quindi può essere rimosso;
- ACTIONCODE, da non considerare nell'algoritmo perché è un dato costruito dal sistema antifrode in essere;
- RISKSCORE, da non considerare nell'algoritmo perché è un dato costruito dal sistema antifrode in essere;
- ID_UTENTE, se considerato il dataset globalmente questa informazione non da valore informativo all'algoritmo di regressione quindi può essere rimosso;
- DATA in formato di stringa, si può trasformare in categorico per raggruppare il timestamp in un momento della giornata;
- TAGS, categorico e utilizzabile senza modifiche;
- PORTE, categorico e utilizzabile senza modifiche;
- VULNERABILITA', numerico di quantità e utilizzato in fase di valutazione;
- CODICE_STATO, categorico e utilizzabile senza modifiche;
- MALWARE, numerico di quantità e utilizzato in fase di valutazione.

Le possibilità di elaborazione più complesse introducibili in questo modulo possono essere valutate a partire dal documento relativo a BankSealer [34]. In questo documento, oltre a progettare un sistema di supporto alle decisioni per l'analisi di frodi, si effettua un'analisi degli attributi normalmente presenti nel contesto dell'online banking e di come questi possano essere utilizzati per definire dei comportamenti. Si riportano di seguito i ragionamenti effettuati nel paper.

La distribuzione dell'ammontare delle transazioni indica che la maggior parte di esse ha importi bassi, si decide di discretizzare attraverso la tecnica di *binning standard equi-frequency*. L'ultimo bin copre un ampio raggio di importi perché la distribuzione è *long-tailed* e quindi si può decidere di rompere a sua volta l'ultimo bin.

Osservando la distribuzione delle transazioni lungo le ore del giorno si nota che la maggior parte delle operazione avviene durante le ore di lavoro. Si possono discretizzare i timestamp in un valore categorico che prevede: mattina presto, mattina, pomeriggio, sera e notte.

Si nota una notevole differenza tra le distribuzioni di attributi con una visione globale rispetto ad una visione locale del singolo utente. Quando analizzati a livello globale, alcuni attributi mostrano distribuzioni uniformi; quando analizzati localmente, gli stessi attributi mostrano una distribuzione sbilanciata e distorta, spesso con più di una modalità. Per questo può avere senso valutare il comportamento del singolo utente per individuare anomalie.

Infine si può decidere se fare valutazioni riguardo le correlazioni tra attributi, attraverso coefficienti come Kendall-tau oppure Spearman [34]. Successivamente e in base ai risultati ottenuti, si decide se lavorare sotto l'assunzione di avere attributi indipendenti oppure correlati.

Questo modulo può essere quindi adatto a valutare il comportamento del singolo utente. Caratteristiche utili per individuare frodi considerandole anomalie, possono essere le seguenti:

- ammontare medio delle transazioni;
- numero di transazioni da stati stranieri;

- intervallo di tempo tra una transazione e l'altra;
- transazioni con beneficiari stranieri.

Altre caratteristiche che non possono essere utilizzate per mancanza di dati ma indicate in letteratura [20]:

- informazioni sui dispositivi utilizzati dall'utente (mobile app oppure browser);
- numero di giorni dall'ultimo login o cambio password;
- numero alto di account acceduto da uno stesso indirizzo.

4.2.3 Decisione e reazione tramite regole

Il modulo di decisione ha il compito di definire, attraverso caratteristiche esplicite, un punteggio di rischio che tenga conto anche del risultato della fase precedente. Sempre in questo modulo sono definite regole che in base al punteggio di rischio possono attivare azioni conseguenti. Le regole relative alla definizione del punteggio aumentano o riducono il livello di rischio. Esempi di regole definite per il caso d'uso sono le seguenti (* non implementate):

- nazione dell'indirizzo IP differente dall'Italia;
- IP in una blacklist definita attraverso honeypot*;
- definita una lista di paesi pericolosi, nazione all'interno di questa lista*;
- provenienza da VPN;
- provenienza da nodo TOR;
- provenienza da proxy;
- porte aperte nell'indirizzo utilizzato;
- segnalazione riguardo malware e vulnerabilità segnalate da tool di OSINT;

- classificazione effettuata dal modulo di elaborazione;
- azioni proattive come blocco della transazione o invio di una mail di segnalazione*.

4.3 Valutazione delle prestazioni

In questa sezione si analizzano le prestazioni del sistema in termini di efficacia ed efficienza. Per quanto riguarda l'efficacia si valuta l'algoritmo di machine learning (fondamentale nel modello per definire il peso del modulo di elaborazione nel giudizio finale) e il risultato finale di individuazione effettiva di transazioni fraudolente o problematiche. Riguardo l'efficienza invece, si analizza nello specifico il modulo di arricchimento perché impatta maggiormente sulla latenza dell'interno sistema a causa delle chiamate remote, successivamente si effettua una valutazione finale dei tempi necessari per analizzare una quantità predefinita di transazioni dall'inizio alla fine della pipeline. I test dei singoli moduli sono effettuati virtualizzando sul notebook (8 core, 16GiB di RAM) i cluster attraverso docker-compose [55] (Docker permette l'esecuzione trasparente di più nodi rispetto ai core disponibili):

- stack ELK, 1 nodo Elasticsearch, 1 nodo Kibana, 1 nodo Logstash;
- cluster Kafka, 1 nodo zookeeper, 3 nodi Kafka;
- cluster Spark, 1 nodo master, 3 nodi worker.

La pipeline escludendo lo stack ELK, viene invece eseguita su di un'infrastruttura fisica composta da 5 server (40 core, 250GiB di RAM, 18TiB di disco) all'interno del datacenter di Imola Informatica, dove al di sopra è installato il *layer* di virtualizzazione Proxmox [56] (infrastruttura visibile in Figura 4.1). Ogni nodo seguente corrisponde ad un container LXC [57] virtualizzato attraverso Proxmox ed essi vengono distribuiti tra i server fisici:

- cluster Spark, 1 nodo master in S5, 4 nodi worker (ognuno con 4 core e 1gb di ram) in S1, S2, S3, S4;

- cluster Kafka, 4 nodi Kafka (ognuno con 2 core e 4gb di ram) in S1, S2, S3, S4.

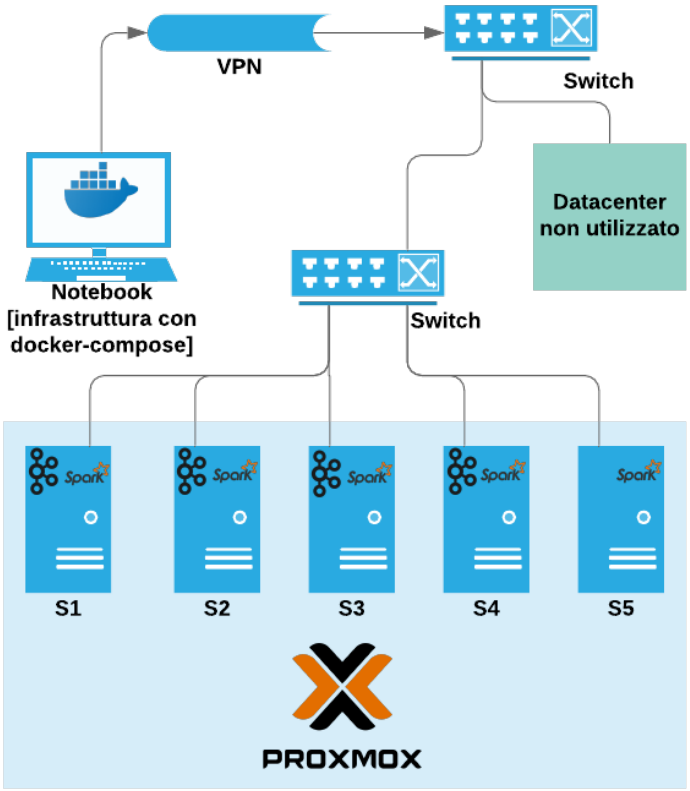

Figura 4.1: Infrastruttura utilizzata per eseguire i test

4.3.1 Efficacia

La valutazione del modello machine learning inserito nel modulo di elaborazione è fondamentale perché l'efficacia nella classificazione determina quanto influisce il risultato del modulo di elaborazione del rischio rispetto al calcolo del punteggio finale. Come specificato nella Sottosezione 4.2.2 di elaborazione del rischio, si utilizza una regressione logistica con visione globale, a seguito di un bilanciamento del dataset. Come iperparametri dell'algoritmo si è definita una colonna di pesi per bilanciare ulteriormente lo sbilanciamento e si configurano massimo 10 iterazioni. Si nota come le performance non

siano nulla di eccezionale; questa difficoltà di classificazione può derivare dallo sbilanciamento e dall'analisi globale che effettua l'algoritmo. Per esempio, attributi come importo e momento della giornata sono considerati fondamentali in questa fase ma sarebbero molto più efficaci valutando l'utente singolarmente.

Indicatore	**Valore**
Accuracy	0.66
Precision	0.68
Recall	0.61
F-Measure	0.64
Area under precision-recall curve	0.64
Area under ROC	0.66
True Positive in training	41019
False Positive in training	19133
True Negative in training	47778
False Negative in training	26109

Tabella 4.2: Performance modello su training set

La transazione uscita dalla pipeline prevede la presenza dei seguenti campi oltre agli arricchimenti: momento del giorno, risultato del modulo di elaborazione e punteggio finale. Lo scenario di test avviene sull'infrastruttura fisica e viene effettuato su 350.000 record (di cui 1250 etichettati come frodi) e consiste nel valutare quanto il punteggio finale è ragionevole o meno tenendo conto dello score precedente e l'etichetta. I risultati in termini di efficacia di individuazione di frodi non sono soddisfacenti: il modello di regressione logistica machine learning non individua frodi e quindi si conferma come una visione globale per l'individuazione di anomalie in questo dominio non sia adatta. Le performance delle regole sono più soddisfacenti perché tutti i casi di frodi risultano avere uno score superiori al minimo. Il problema della valutazione è che tutti i falsi positivi individuati nel modulo machine learning

Indicatore	Valore
Accuracy	0.73
Precision	0.68
Recall	0.60
F-Measure	0.64
Area under precision-recall curve	0.64
Area under ROC	0.66
True Positive in testing	4392
False Positive in testing	2077
True Negative in testing	5336
False Negative in testing	2813

Tabella 4.3: Performance modello su test set

risultano avere uno score più alto delle frodi reali (il peso del modello machine learning impatta notevolmente in questo scenario). Per quanto riguarda le valutazioni aziendali sull'efficacia, è da tenere presente che notevoli miglioramenti sono possibili effettuando il tuning dei modelli e degli algoritmi utilizzati, il cui lavoro in sé, può essere visto come argomento di tesi a sé stante e dedicato più a temi di data science.

4.3.2 Efficienza

La valutazione del modulo di arricchimento è fortemente legata alle chiamate HTTP effettuate verso i tool di OSINT e questo influenza la latenza dell'intero sistema. Internamente effettua due chiamate HTTP (una a Shodan e una ad AlienVault, la richiesta a VirusTotal è simulata) e si prevede un timeout di 5s in questa fase sperimentale. Per arricchire circa 100.000 transazioni sull'infrastruttura virtualizzata attraverso docker-compose, il modulo necessita di circa 17 ore senza parallelizzazione (i topic hanno una sola partizione in questo scenario). Si tratta di 1.3 arricchimenti al secondo di media. Il

timeout è relativamente alto per una richiesta HTTP e si ipotizza impatti sul valore medio degli arricchimenti. Questo ritardo potrebbe essere ridotto in presenza di un modulo di fallback per i fallimenti. Bisogna inoltre specificare che Spark Structured assegna ad un worker/executor una partizione specifica di un topic Kafka e per l'intera durata di vita, applica l'intero grafo di esecuzione all'interno dello stesso executor sulla partizione specifica. Per parallelizzare quindi l'arricchimento, è necessario produrre eventi su più partizioni Kafka in modo che Spark Structured bilanci automaticamente il carico di ogni partizione su un worker diverso. Essendo l'arricchimento atomico rispetto alla singola transazione, nello scenario che preveda più partizioni la parallelizzazione è totale. La latenza introdotta nel modulo di arricchimento rallenta di fatto i passaggi successivi che prevedono machine learning e regole di business, i quali si adattano quindi a gestire immediatamente le transazioni arricchite con latenza trascurabile. Si riporta in conclusione il secondo scenario di test in cui l'analisi di 100.000 transazioni ha impiegato circa 4 ore utilizzando l'infrastruttura fisica e definendo 4 partizioni per ogni topic Kafka.

Il progetto di tesi può essere valutato in termini di una POC (Proof Of Concept) per una piattaforma di arricchimento e valutazione di stream di eventi. Il punto saliente e più interessante in termini aziendali è sicuramente la parte architetturale. Sebbene in termini di efficienza i risultati possano sembrare poco lusinghieri, bisogna considerare che in uno scenario di uso reale, i dati arrivano distribuiti nel tempo e possibilmente elaborati su ambiente distribuito: la piattaforma presentata può facilmente scalare orizzontalmente, distribuendo così il carico fra più nodi. È inoltre da sottolineare come 1.3 secondi come tempo di elaborazione medio ben si concilia con una navigazione utente: miglioramenti in tal senso possono essere raggiunti utilizzando base dati offline e aumentato il throughput delle richieste verso le API di terze parti (possibile utilizzando piani commerciali).

Conclusioni

L'obiettivo iniziale della tesi era quello di costruire un sistema che individuasse frodi nel dominio dell'online banking in near real-time, processando grandi quantità di dati attraverso strumenti di big data open source. Durante il procedere degli studi e della progettazione è emerso come le caratteristiche che si stavano modellando potessero essere riutilizzate anche in altri domini. Per questo si è deciso di focalizzarsi sulla costruzione di un sistema il più astratto possibile, che avesse certe caratteristiche generiche e la capacità di essere calato in un dominio specifico con sforzo ridotto. Il sistema creato nella tesi può essere quindi definito come una piattaforma big data per l'elaborazione di stream di dati attraverso machine learning e business rules. Questo perché la piattaforma permette di gestire una sequenza di passaggi concettuali grazie alla combinazione di algoritmi di machine learning *pluggable* e l'espressività garantita da un approccio a regole. Gli elementi messi a fattor comune possono essere sintetizzati dal modello GEAR definito in tesi, che prevede una sequenza di trasformazioni e analisi da effettuare su uno stream di dati, allegando i risultati di ogni passaggio ai dati stessi.

Per costruire la pipeline di trasformazione che implementasse il modello, è stato necessario individuare paradigmi e approcci utilizzabili per ogni modulo del sistema attraverso l'analisi dello stato dell'arte. Il modulo di arricchimento ha necessitato di individuare sorgenti da cui trarre nuovi dati collegati agli eventi partendo dalle informazioni a disposizione. Sicuramente sarebbe stato possibile utilizzare sorgenti interne all'organizzazione ma è stato deciso di adottare un approccio OSINT per valutarne l'efficacia. Ispirandosi ai sistemi

per individuare frodi e anomalie già ben analizzati in letteratura, la valutazione è stata suddivisa in due moduli, quello di elaborazione e quello di decisione. Il primo che si occupa di applicare tecniche di machine learning per abilitare anomaly based detection e il secondo che utilizza un approccio basato su regole per la misuse based detection. Dato che la valutazione e la reazione utilizzano lo stesso paradigma, si è deciso di accorpare l'attivazione di queste ultime sempre nel modulo di valutazione.

Le prestazioni dell'intero sistema sono difficilmente valutabili aldilà del caso d'uso, perché fortemente dipendenti dal dominio. Sono però presenti le garanzie fornite dai paradigmi e tecnologie utilizzate (Spark e Kafka). I risultati ottenuti in questa tesi costituiscono un punto di partenza per il perfezionamento complessivo del sistema. Riguardo al caso d'uso infatti, una migliore analisi del dominio potrebbe portare all'utilizzo di algoritmi robusti rispetto al dataset sbilanciato (come per esempio la Sensitive Neural Network) oppure a focalizzarsi maggiormente sulla profilazione individuale dei singoli utenti. Le implementazioni successive sarebbero poi relativamente semplici grazie all'architettura pluggable e allo stack tecnologico definito in tesi. Può essere anche interessante parallelizzare più moduli di elaborazione che eseguano analisi, utilizzando algoritmi e tecniche diverse, per poi aggregare i risultati prima del modulo di decisione. Ulteriori proseguimenti di interesse aziendale sull'argomento, possono riguardare la trasformazione in una piattaforma SIEM (Security Information and Event Management), al fine del miglioramento della sicurezza della rete aziendale interna. Infine, come società di consulenza, la piattaforma GEAR sarà sicuramente uno spunto per lo sviluppo di piattaforme di elaborazione di stream di dati presso i propri clienti. In conclusione il modello GEAR, l'architettura e l'implementazione del sistema creato in tesi risulta adatto ad applicare una sequenza di operazioni (Gather, Enrich, Assess, React) su uno stream di dati garantendo performance applicabili a casi d'uso reali e una forte flessibilità di implementazione e evoluzione.

Elenco delle figure

Bibliografia

[1] iCorporate - Eleonora Meneghelli / Ilaria Mastrogregori. *La trasformazione nel panorama dei pagamenti europei*. A cura di SWIFT — iCorporate. URL: https://www.swift.com/resource/transformation-european-payments-landscape.

[2] *definizione ABI di: direttiva europea sui servizi di pagamento*. URL: https://www.abi.it/Pagine/Mercati/Sistemipagamento/Direttiva-europea-sui-servizi-di-pagamento/Direttiva-europea-sui-servizi-di-pagamento.aspx.

[3] *Regolamento Delegato (UE) 2018/389 della Commissione*. URL: https://eur-lex.europa.eu/legal-content/IT/TXT/PDF/?uri=CELEX:32018R0389&from=EN.

[4] gartner.com, cur. *Definition of Security Operation Center (SOC)*. [Online; 24-Gennaio-2020]. URL: https://www.gartner.com/en/newsroom/press-releases/2017-10-12-security-operations-centers-and-their-role-in-cybersecurity.

[5] gartner.com, cur. *Definition of Security Information And Event Management (SIEM)*. [Online; 24-Gennaio-2020]. URL: https://www.gartner.com/en/information-technology/glossary/security-information-and-event-management-siem.

[6] Anton Chuvakin. *La guida completa alla gestione di log ed eventi*. A cura di NetIQ. URL: https://www.microfocus.com/media/white-paper/the_complete_guide_to_log_and_event_management_wp_it.pdf.

[7] Martin H. Weik. «open-source intelligence». In: *Computer Science and Communications Dictionary*. Boston, MA: Springer US, 2001, pp. 1145–1145. ISBN: 978-1-4020-0613-5. DOI: 10.1007/1-4020-0613-6_12818.

[8] Jeffrey T Richelson. *The U.S. Intelligence Community 7th*. Routledge; 7 edition, 2015.

[9] Lowenthal Mark e Clark Robert. *The Five Disciplines of Intelligence Collection*. CQ Press. p. 18. ISBN 978-1483381114, 2015.

[10] Commission on the Intelligence Capabilities of the United States Regarding Weapons of Mass Destruction. *The Commission on the Intelligence Capabilities, 378–379*.

[11] Arthur S. Hulnick. *The Dilemma of Open Sources intelligence: Is OSINT Really Intelligence?* OUP USA, 2010.

[12] Cybersecurity Spotlight. *Internet Protocol*. URL: https://www.cisecurity.org/spotlight/cybersecurity-spotlight-internet-protocol/.

[13] Chris Vickery. *What Is IP Attribution, and Why Is It Doomed?* A cura di upguard.com. [Online; updated 15-Gennaio-2020]. Gen. 2020. URL: https://www.upguard.com/blog/ip-attribution.

[14] CLEAFY. «CLEAFY Threat Detection and Protection - solution whitepaper». In: (2016). A cura di CLEAFY.

[15] Jonathan Care e Akif Khan. «Market Guide for Online Fraud Detection». In: (2019). A cura di Gartner.

[16] docs.microsoft.com, cur. *Architetture per Big Data*. [Online; updated 23-Gennaio-2020]. Feb. 2018. URL: https://docs.microsoft.com/it-it/azure/architecture/data-guide/big-data/.

[17] A.G. Psaltis. *Streaming Data Understanding the Real-Time Pipeline*. Manning, 2017.

[18] Sheik Hoque e Andriy Miranskyy. «Online and Offline Analysis of Streaming Data». In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)* (apr. 2018). DOI: 10.1109/icsa-c.2018.00026. URL: http://dx.doi.org/10.1109/ICSA-C.2018.00026.

[19] oxford dictionary, cur. *Definition of fraud in English*. [Online; 24-Gennaio-2020]. URL: https://www.lexico.com/en/definition/fraud.

[20] Aisha Abdallah, Mohd Aizaini Maarof e Anazida Zainal. «Fraud detection system: A survey». In: *Journal of Network and Computer Applications* 68 (2016), pp. 90–113. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2016.04.007. URL: http://www.sciencedirect.com/science/article/pii/S1084804516300571.

[21] M. Behdad et al. «Nature-Inspired Techniques in the Context of Fraud Detection». In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (nov. 2012), pp. 1273–1290. ISSN: 1558-2442. DOI: 10.1109/TSMCC.2012.2215851.

[22] John Akhilomen. «Data Mining Application for Cyber Credit-Card Fraud Detection System». In: *Advances in Data Mining. Applications and Theoretical Aspects*. A cura di Petra Perner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 218–228.

[23] Bo Sun et al. «Enhancing security using mobility-based anomaly detection in cellular mobile networks». In: *IEEE Transactions on Vehicular Technology* 55.4 (lug. 2006), pp. 1385–1396. ISSN: 1939-9359. DOI: 10.1109/TVT.2006.874579.

[24] Xiaojin Zhu e Andrew B Goldberg. «Introduction to semi-supervised learning». In: *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009), pp. 1–130.

[25] David J. Hand e Martin J. Crowder. «Overcoming selectivity bias in evaluating new fraud detection systems for revolving credit operations». In: *International Journal of Forecasting* 28.1 (2012). Special

Section 1: The Predictability of Financial Markets Special Section 2: Credit Risk Modelling and Forecasting, pp. 216–223. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2010.10.005. URL: http://www.sciencedirect.com/science/article/pii/S0169207011000124.

[26] Wei Wei et al. «Effective detection of sophisticated online banking fraud on extremely imbalanced data». In: *World Wide Web* 16.4 (lug. 2013), pp. 449–475. ISSN: 1573-1413. DOI: 10.1007/s11280-012-0178-0. URL: https://doi.org/10.1007/s11280-012-0178-0.

[27] Peter Jackson. *Introduction to Expert Systems*. 3rd. USA: Addison-Wesley Longman Publishing Co., Inc., 1998. ISBN: 0201876868.

[28] Saharon Rosset et al. «Discovery of fraud rules for telecommunications—challenges and solutions». In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 1999, pp. 409–413.

[29] Ashutosh Deshmukh e Lakshminarayana Talluru. «A rule-based fuzzy reasoning system for assessing the risk of management fraud». In: *Intelligent Systems in Accounting, Finance & Management* 7.4 (1998), pp. 223–241.

[30] Peter Brennan. «A comprehensive survey of methods for overcoming the class imbalance problem in fraud detection». In: *Institute of technology Blanchardstown Dublin, Ireland* (2012).

[31] Chan K Philip et al. «Distributed data mining in credit card fraud detection». In: *IEEE Intelligent Systems* 14.6 (1999), pp. 67–74.

[32] Constantinos S Hilas e John N Sahalos. «An application of decision trees for rule extraction towards telecommunications fraud detection». In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer. 2007, pp. 1112–1121.

[33] Michael Edward Edge e Pedro R. Falcone Sampaio. «A survey of signature based methods for financial fraud detection». In: *Computers & Security* 28.6 (2009), pp. 381–394. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2009.02.001. URL: http://www.sciencedirect.com/science/article/pii/S0167404809000091.

[34] Michele Carminati et al. «BankSealer: A decision support system for online banking fraud analysis and investigation». In: *Computers & Security* 53 (2015), pp. 175–186. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2015.04.002. URL: http://www.sciencedirect.com/science/article/pii/S0167404815000437.

[35] Chilukuri K. Mohan e ishan G. Mehrotra. «Anomaly detection in banking operations». In: *IJBT* (). URL: https://pdfs.semanticscholar.org/b48a/fd871d0f2d846a9d1d8ab682bc70da12dde3.pdf.

[36] P. Loucopoulos e P.J. Layzell. «Improving information system development and evolution using a rule-based paradigm». In: *Software Engineering Journal* (1989). URL: https://ieeexplore.ieee.org/document/42959.

[37] Ian Graham. *Business Rules Management and Service Oriented Architecture: A Pattern Language*. Wiley, 2006.

[38] Robert Laurini e Derek Thompson. «17 - Spatial Knowledge: Intelligent spatial information systems». In: *Fundamentals of Spatial Information Systems*. A cura di Robert Laurini e Derek Thompson. Boston: Academic Press, 1992, pp. 620–670. ISBN: 978-0-12-438380-7. DOI: https://doi.org/10.1016/B978-0-08-092420-5.50022-0. URL: http://www.sciencedirect.com/science/article/pii/B9780080924205500220.

[39] Arthur H. Veen. «Dataflow Machine Architecture». In: *ACM Comput. Surv.* 18.4 (dic. 1986), pp. 365–396. ISSN: 0360-0300. DOI: 10.1145/27633.28055. URL: https://doi-org.ezproxy.unibo.it/10.1145/27633.28055.

[40] kafka.apache.org, cur. *Documentazione Apache Kafka*. [Online; 10-Gennaio-2020]. URL: https://kafka.apache.org/.

[41] spark.apache.org, cur. *Documentazione Spark Streaming*. [Online; 10-Gennaio-2020]. URL: https://spark.apache.org/docs/latest/streaming-programming-guide.html.

[42] flink.apache.org, cur. *Documentazione Flink*. [Online; 10-Gennaio-2020]. URL: https://ci.apache.org/projects/flink/flink-docs-release-1.9/.

[43] spark.apache.org, cur. *Documentazione Spark Structured*. [Online; 10-Gennaio-2020]. URL: https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html.

[44] dzone.com, cur. *Differenze Spark Streaming e Structured Streaming*. [Online; 10-Gennaio-2020]. URL: https://dzone.com/articles/spark-streaming-vs-structured-streaming.

[45] spark.apache.org, cur. *Documentazione Spark MLlib*. [Online; 10-Gennaio-2020]. URL: https://spark.apache.org/mllib/.

[46] h2o.ai, cur. *H2O Sparkling Water*. [Online; 10-Gennaio-2020]. URL: https://www.h2o.ai/products/h2o-sparkling-water/.

[47] J. Zhang, J. Yang e J. Li. «When Rule Engine Meets Big Data: Design and Implementation of a Distributed Rule Engine Using Spark». In: *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*. Apr. 2017, pp. 41–49. DOI: 10.1109/BigDataService.2017.17.

[48] Saikrishna Pujari. *Apache Spark Integration with DROOLS*. A cura di linkedin.com. [Online; 13 gennaio 2020]. Apr. 2019. URL: https://www.linkedin.com/pulse/spark-drools-integration-poc-sai-krishna-pujari.

[49] drools.org, cur. *Documentazione Drools*. [Online; 10-Gennaio-2020]. URL: https://docs.jboss.org/drools/release/7.32.0.Final/drools-docs/html_single/index.html.

[50] Michael K Bergman. «White paper: the deep web: surfacing hidden value». In: *Journal of electronic publishing* 7.1 (2001).

[51] shodan.io, cur. *What is Shodan*. [Online; 11-Gennaio-2020]. URL: https://help.shodan.io/the-basics/what-is-shodan.

[52] virustotal.com, cur. *Contributors VirusTotal*. [Online; 11-Gennaio-2020]. URL: https://support.virustotal.com/hc/en-us/articles/115002146809-Contributors.

[53] Seth Gilbert e Nancy Lynch. «Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services». In: *SIGACT News* 33.2 (giu. 2002), pp. 51–59. ISSN: 0163-5700. DOI: 10.1145/564585.564601. URL: https://doi.org/10.1145/564585.564601.

[54] lucene.apache.org, cur. *Apache Lucene Features*. [Online; 28-Febbraio-2020]. URL: https://lucene.apache.org/core/.

[55] docker.com, cur. *Docker Compose Overview*. [Online; 8-Marzo-2020]. URL: https://docs.docker.com/compose/.

[56] proxmox.com, cur. *Proxmox*. [Online; 8-Marzo-2020]. URL: https://www.proxmox.com/en/.

[57] linuxcontainers.org, cur. *Linux Containers - LXC*. [Online; 10-marzo-2020]. URL: https://linuxcontainers.org/lxc/introduction/.