

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

STRUMENTI DI ORCHESTRAZIONE DI WORKFLOW NEL MACHINE LEARNING

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Presentata da:
Simone Boldrini

Correlatore:
Dott.
Stefano Pio Zingaro
Saverio Giallorenzo

Sessione II
Anno Accademico 2020/21

Indice

1	Introduzione	4
1.1	Intelligenza Artificiale	5
1.2	Apprendimento Automatico	5
2	Casi di Studio	8
2.1	Apprendimento automatico nel campo delle Reti	8
2.1.1	Panoramica	8
2.1.2	Workflow	9
2.1.3	Campi di Ricerca	10
2.1.4	Opportunità	12
2.2	Apprendimento automatico nella Sicurezza Informatica	14
2.2.1	Introduzione	14
2.2.2	Pre Processamento dei dati	14
2.2.3	Algoritmi di apprendimento	15
3	Analisi dei Casi di Studio	19
3.1	Introduzione	19
3.2	Analisi	19
3.2.1	Raccolta Dati	20
3.2.2	Elaborazione del Modello	21
3.2.3	Risultato	22
3.3	Conclusioni	23
4	Orchestrazione di Flussi di lavoro	24
4.1	Framework	25
4.1.1	ZenML	26
4.1.2	Kedro	27
4.1.3	Flyte	27
4.1.4	MLRun	29
4.2	Conclusioni	30

Glossario	31
Bibliografia	32

Capitolo 1

Introduzione

Il Machine Learning sta diventando molto diffuso nelle realtà aziendali IT e non solo, essenziale per fare previsioni basati su calcoli statistici.

In questo testo andremo ad analizzare due casi di studio, per comprendere le modalità in cui l'apprendimento automatico interagisce in diversi ambiti, al fine di illustrare una serie di strumenti che permettono di orchestrare in maniera automatica i flussi di lavoro del Machine Learning.

Metteremo in relazione i casi di studio trovati con altri due compagni di corso, con l'obiettivo di andare a trovare un'invariante il più possibile automatizzata che possa permettere di sfruttarne i benefici.

Questa analisi viene eseguita confrontando i diversi passaggi “fondamentali” tra i casi di studio presi in considerazione.

Una volta analizzate le differenze e le analogie presenti nei diversi casi di studio, tracciamo una linea di “workflow generica”, evidenziando le fasi di processo comuni, i quali diventeranno essenziali per capire come cercare di automatizzare la gestione dei flussi di lavoro del Machine Learning.

Senza la pretesa di essere completi studieremo e confronteremo un insieme di Framework, che permettono di orchestrare in maniera automatica diversi flussi di lavoro che, come vedremo, saranno simili al “workflow generico” illustrato in fase di analisi.

Per illustrare e capire il funzionamento di diversi strumenti di orchestrazione è necessario, in prima istanza, analizzare e confrontare come i processi di ML agiscono nei diversi casi di studio utilizzando in tal modo un approccio bottom-up.

1.1 Intelligenza Artificiale

“We spend a great deal of time studying history, which, let’s face it, is mostly the history of stupidity. So it’s a welcome change that people are studying instead the future of intelligence. ”

- Stephen Hawking,

L’intelligenza è l’abilità di imparare dalle esperienze, per applicare le conoscenze volte a risolvere dei problemi, adattandoli in differenti ambiti.

L’intelligenza artificiale è un ramo dell’informatica che si occupa di studiare metodologie e tecniche in grado di sviluppare programmi software capaci di fornire all’elaboratore elettronico prestazioni che possono apparire di pertinenza esclusiva dell’intelligenza umana.

Possiamo osservare diverse tipologie di AI:

- Thinking Humanly
- Action Humanly
- Thinking Rationally
- Action Rationally

La differenza in questi 4 campi mette in relazione il pensare dell’umano al risultato(“*action*”) oppure al processo(“*thinking*”) del sistema intelligente.

Nei primi due casi il sistema intelligente viene paragonato alla capacità di risolvere un problema o di svolgere un’azione in maniera simile a quella di un umano.

Mentre gli ultimi due campi svolgono un’azione seguendo un processo razionale.

1.2 Apprendimento Automatico

L’apprendimento automatico(*Machine Learning, ML*) è un ramo dell’intelligenza artificiale, che raccoglie metodi in grado di migliorare la performance di un algoritmo, autonomamente, nell’identificare pattern di dati.[15]

L’apprendimento automatico cerca di costruire algoritmi e modelli che possono imparare a prendere decisioni senza dati con regole predefiniti, gli algoritmi di ML esistenti si suddividono in tre categorie:

- **Supervised Learning:** algoritmi che cercano di condurre una classificazione o regressione su dati targettizzati.
- **Unsupervised Learning,** si focalizzano sulla classificazione di semplici insiemi in gruppi

- **Reinforcement Learning**, gli agenti imparano la migliore azione da eseguire che ti portino al beneficio superiore

L'*Apprendimento supervisionato* mira a istruire il sistema in modo da permettergli di elaborare automaticamente previsioni sui valori di uscita rispetto ad un input sulla base di una serie di “esempi ideali”, costituiti da coppie $\langle input, output \rangle$ fornitegli in input.[17] D'altrocanto l'*apprendimento non supervisionato* invece, riceverà un insieme di input (*Esperienza*) che in automatico verranno riclassificati sulla base di caratteristiche comuni, per cercare di eseguire ragionamenti e previsioni sugli input successivi.[18]

L'*apprendimento per rinforzo* mira a realizzare agenti autonomi in grado di scegliere azioni da compiere per il conseguimento di determinati obiettivi, tenendo in considerazione l'ambiente di cui fanno parte. A differenza dei precedenti paradigmi, si occupa di problemi decisionali sequenziali, in cui l'azione è fortemente condizionata dallo stato attuale del sistema che ne determina quello futuro.[16]

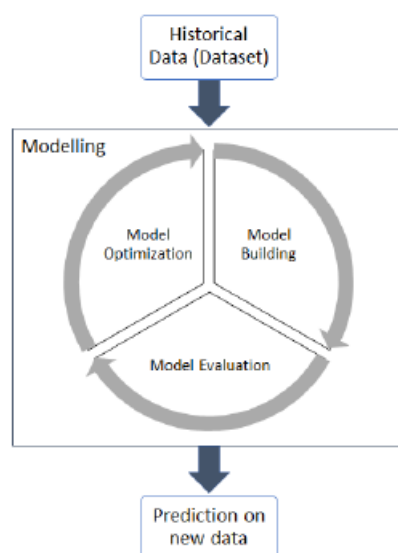


Figura 1.1: Model Building process

Gli algoritmi di ML assimilano i dati di addestramento, è possibile produrre modelli più precisi basati su tali dati.

Un modello di ML è l'output generato quando si addestra il proprio algoritmo con i dati(Figura 1.1); questo modello una volta addestrato sarà in grado di generare un output in base ai dati di addestramento che hanno contribuito alla generazione di esso. Dato l'alto tasso di accuratezza degli approcci di ML e la rapidità in cui il modello può essere generato, ML è usato ogni giorno da milioni di aziende per fare previsioni sulla più efficiente strategia di business.

Il processo di ML spesso dipende dalla natura dei dati di training, durante il processo di allenamento, il framework ML è allenato per raggiungere un determinato obiettivo come prendere decisioni, predire un valore o stilare una classificazione.

La fase di training permette di scoprire potenziali relazioni tra i dati di input e output senza alcun intervento umano. Esistono anche algoritmi di ML “online” nei quali ogni predizione aggiorna il modello per ogni nuova funzione di input.

In questo testo studieremo e confronteremo due casi di studi riguardanti la costruzione di questi modelli per problemi diversi, con l’obiettivo di cercare le fasi che accomunano i differenti casi di studio ed analizzarli.

Capitolo 2

Casi di Studio

2.1 Apprendimento automatico nel campo delle Reti

2.1.1 Panoramica

Il Machine learning è un sottoinsieme delle AI, si sta sviluppando in ogni campo, nelle pagine seguenti andremo a studiare le opportunità usando il Machine Learning applicato alle reti.

L'Apprendimento automatico permette ai sistemi di imparare automaticamente e prendere decisioni o predizioni basati sull'esperienza. Con lo sviluppo di internet, ricercatori e operatori di rete possono affrontare vari tipi di rete e applicazioni, le quali possono cambiare a seconda delle performance e dei requisiti.

L'incorporazione di ML nella gestione e nella pianificazione delle reti permette la possibilità di sviluppare nuove applicazioni di rete.

Il ML nelle reti può giocare un ruolo importante soprattutto nei problemi di reti più comuni, basti pensare a Intrusion Detection e Performance Prediction, inoltre è anche possibile aiutare a prendere decisioni facilitando lo scheduling di rete e l'adattamento di parametri in accordo con lo stato corrente dell'ambiente.

Altri problemi di rete invece, più complicati, hanno bisogno di interagire con sistemi complessi di rete, quali costruire e analizzare modelli che rappresentano sistemi complessi come il cambiamento di modelli delle CDN e delle caratteristiche di Throughput.

Machine Learning può fornire una stima di un modello di un sistema con un'accuratezza accettabile.

Infine, ogni scenario di rete ha diverse caratteristiche e i ricercatori spesso hanno bisogno di risolvere problemi per ogni scenario in maniera indipendente.

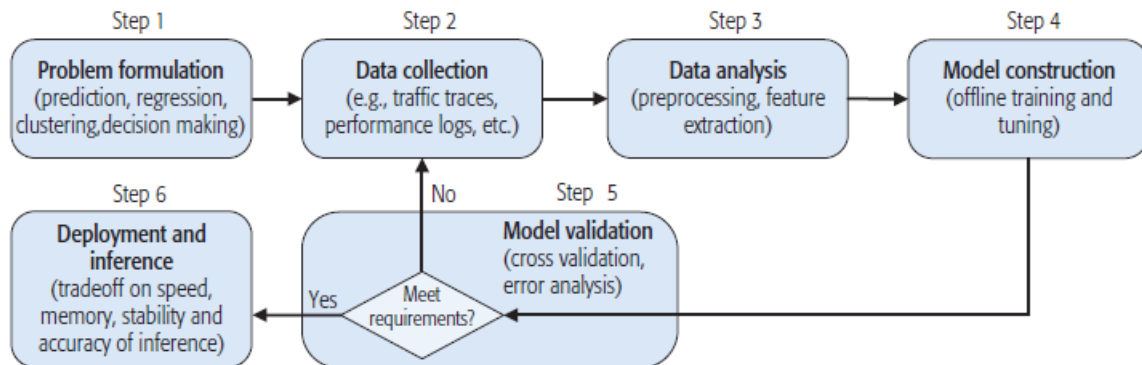


Figura 2.1: ML Networking Workflow

2.1.2 Workflow

Formulazione del problema: Il processo di allenamento di ML spesso implica alti costi di tempo, è importante, dunque, formulare e astrarre correttamente il problema al primo step.

In questa fase ci si occupa di dividere il problema in una delle Categorie ML: Classificazione, Raggruppamento o Decisionali[Rif.pag.23].

Questo ci aiuta a decidere che tipo e che mole di dati ci serve per il modello.

Un'astrazione impropria del problema può fornire un modello di apprendimento inadatto, che può risultare in prestazioni di apprendimento insoddisfacenti.

Collezioni Dati: In questa fase l'obiettivo è raccogliere un grande numero di dati rappresentativi senza Bias, i dati di rete vengono raccolti da diversi livelli di rete in accordo con l'applicazione di monitoraggio. Per esempio i problemi di classificazione del traffico richiedono dataset contenenti tracce a livello di pacchetto corrispondenti alle classi di applicazioni.

Nel contesto del MLN i dati sono spesso collezionati in due fasi. Nella fase Offline, si raccolgono dati storici di alta-qualità importanti per le analisi e l'addestramento del modello mentre nella fase online si utilizzano dati dello stato di rete in real-time usati come input o come segnali di feedback per addestrare il modello.

Analisi Dati: Ogni problema di rete ha le proprie caratteristiche ed è influenzato da molti fattori.

Nel nostro caso di MLN trovare le caratteristiche adeguate è la chiave per potenziare al meglio il nostro modello. Fondamentale in questa fase elaborare i dati grezzi attraverso processi di normalizzazione, discretizzazione e completamento del valore mancante, questa fase richiede una conoscenza specifica del dominio, in seguito si punterà ad estrarre le funzionalità effettive che possiamo trarre da questi dati.

Costruzione Modello: la costruzione del modello include la selezione e l'allenamento dello stesso. Un adatto modello di apprendimento o algoritmo ha bisogno di dati e dimensioni di essi il più coerenti possibili con le caratteristiche dello scenario di rete e la categoria del problema.

Validazione del modello: Validazione offline è uno step fondamentale nel workflow del MLN per valutare l'algoritmo di apprendimento. La convalida incrociata è spesso usata per testare l'accuratezza del modello(se il modello è Over-fitting o Under-fitting). Questo prevede una buona guida su come ottimizzare il modello(e.g. aumentando il volume dei dati o riducendo la complessità del modello in caso questo sia Over-fitting)

Deployment and Inference: Quando implementiamo un modello di apprendimento automatico nell'ambiente di rete dobbiamo considerare alcuni problemi, ci possono essere limitazioni sulle computazioni o nelle risorse, bisogna trovare un tradeoff tra accuratezza e overhead per i sistemi pratici di reti. L'apprendimento automatico spesso funziona in best-effort e non fornisce alcune prestazioni di garanzia però bisogna comunque tener conto della tolleranza degli errori.

2.1.3 Campi di Ricerca

Parlando di apprendimento automatico possiamo già separare i case study in 3 categorie:

La predizione del traffico e la classificazione sono i due campi più richiesti nel campo del MLN. La *Predizione del traffico* è un importante problema decisionale, la stima più accurata del volume del traffico è un beneficio del controllo della congestione, allocazione delle risorse, dei percorsi di Routing e anche ad alto livello dei livelli di applicazione. Ci sono due principali direzioni di ricerca: analisi delle serie temporali e tomografia di rete, possono dipendere semplicemente se la predizione del traffico viene condotta con osservazioni dirette o meno.

Tuttavia, è molto costoso misurare il volume del traffico direttamente, soprattutto a larga scala in una grande velocità di rete.

La *classificazione del traffico* trova corrispondenza con le applicazioni di rete e i protocolli con i rispettivi flussi di traffico.

I tradizionali metodi di classificazione del traffico sono Port-based e Payload-based. Il metodo port-base si considera inefficiente a causa del continuo cambiamento e riutilizzo delle porte, mentre il payload-based soffre di problemi di privacy dovuti alle analisi del contenuto dei pacchetti che risulta fallimentare in caso di traffico criptato. Gli approcci di ML sono basati su funzioni statistiche, studiati negli anni recenti specialmente nel campo della sicurezza del dominio di rete.



Figura 2.2: ML technique for MLN

Non è semplice considerare ML come una soluzione onnipotente e rilasciarla nel mondo reale, generalmente questi studi variano a seconda dello scenario da tutte le classificazioni verso più situazioni reali con traffico sconosciuto. Questa tabella riassuntiva(Figura 2.2) mette in relazione tecnologie che evolvono dal Supervised learning verso Unsupervised e semisupervised learning , il quale hanno permesso di importare l'apprendimento automatico nei campi di rete.

Un *Efficiente gestione delle risorse* e un *ottimale adattamento di rete* sono le chiavi per aumentare le performance dei sistemi di rete.

Alcuni esempi di problemi sono la pianificazione del traffico, Routing e controllo delle congestioni TCP. Tutti questi problemi possono essere formulati tramite problemi di decisione.

È difficile risolvere questi problemi con sistemi di regole basate su algoritmi euristici a causa delle complessità dei diversi sistemi, degli input (rumorosi) e delle difficoltà nell'ottimizzazione delle prestazioni in coda.

Soprattutto l'assegnamento di parametri arbitrari basati su esperienze o azioni prese in seguito a predeterminate regole, spesso si traduce in algoritmo di pianificazione che è compreso dalle persone ma tutt'altro che ottimale.

Il Deep learning è una soluzione promettente grazie alla sua capacità di caratterizzare le differenze tra input e output senza un coinvolgimento umano.

2.1.4 Opportunità

In questa sezione evidenziamo nuove potenziali richieste che possono apparire in discipline di rete grazie allo sviluppo dell'apprendimento automatico.

Open Dataset per la comunità di rete

Collezionare una grande quantità di dati che contengano sia i profili di rete che le prestazioni è uno dei problemi critici del MLN.

Tuttavia, acquisire abbastanza dati targettizzati è ancora molto costoso per la comunità dell'apprendimento automatico.

Per molte ragioni non è semplice per i ricercatori acquisire abbastanza dati di tracciamento reali, anche se esistono molti dati aperti nel dominio di rete. Questa realtà ci porta a sviluppare maggiori dataset aperti come ImageNet.

Con set di dati aperti, i test delle prestazioni sono un risultato inevitabile per fornire una piattaforma standard ai ricercatori per confrontare i loro nuovi algoritmi o architetture con quelli all'avanguardia.

Automatici Protocolli di rete e architetture

Con una profonda conoscenza delle reti, i ricercatori gradualmente scoprono che le reti esistenti hanno molte limitazioni.

Esiste tuttavia un ampio margine di miglioramento delle performance di rete e alla efficienza ridisegnando i protocolli di rete e la loro architettura. È abbastanza difficile riprogettare un protocollo o un architettura in maniera automatica, tuttavia grazie alla comunità del Machine Learning si sono trovate alternative più semplici in questa direzione, come consentire agli agenti di comunicare con altri per completare un task in maniera cooperativa.

In ogni modo nuovi risultati mostrano come i modelli di ML hanno l'abilità di generare elementi esistenti nel mondo reale e creare strategie che le persone non sono in grado di predire.

Tuttavia questi risultati generati sono tutt'ora lontani dalla possibilità di generare un nuovo protocollo di rete.

C'è un grande potenziale e una grande volontà creare nuovi componenti di rete senza il coinvolgimento umano, i quali possono aggiornare la loro conoscenza in base al sistema di rete.

Promuovere lo sviluppo del ML

Quando si applica il ML nei campi di rete, a causa di richieste dei sistemi di rete e pratici problemi implementativi, alcune limitazioni ereditarie e alcuni problemi emergenti del ML possono essere inoltrati a una nuova fase di compressione con il beneficio dell'unione di due comunità di ricerca. Ad esempio uno dei principali problemi da risolvere è proprio la robustezza degli algoritmi di ML.

La robustezza degli algoritmi dell'apprendimento automatico è la sfida chiave per le applicazioni nell'ambiente del mondo reale dove gli errori di apprendimento potrebbero portare a costi elevati.

È necessario un modello con un'alta abilità di generalizzazione che si adatta all'alta varianza e adattamento del traffico dinamico altrimenti sarà necessario aggiornare il modello verso i cambiamenti del traffico di rete (il che è inaccettabile).

Sebbene alcuni esperimenti sono stati condotti sotto specifici ambienti di rete, possono fornire buoni risultati in altri ambienti, non è scontato poiché la maggior parte degli algoritmi di apprendimento automatico presuppone che i dati seguano la stessa distribuzione, il che non è pratico negli ambienti di rete. [13, 12, 1]

2.2 Apprendimento automatico nella Sicurezza Informatica

2.2.1 Introduzione

La sicurezza informatica sta diventando il rischio chiave per qualsiasi azienda poiché il numero di attacchi sta crescendo e i nostri dati diventano sempre più vulnerabili.

L'Intelligenza Artificiale e il Machine Learning possono aiutare a rilevare minacce e fornire consigli agli analisti per sapere difendersi di conseguenza.

ML può essere usato per identificare target avanzati, le vulnerabilità dell'infrastruttura ed eventuali exploit.

Il numero di minacce è in aumento e rischia di compromettere la nostra privacy e la nostra professionalità quotidianamente, l'evolversi di questo campo rischia di lasciare indietro gli analisti che non riescono a tracciare i nuovi malware e i nuovi attacchi.

Nuovi attacchi e malware sofisticati sono stati in grado di aggirare il livello di sicurezza della rete e degli endpoint per fornire attacchi informatici a velocità allarmanti.

Le applicazioni di ML possono processare e analizzare grandi volumi di dati sperimentali in nuovi modi, gli algoritmi di ML possono fornire una visione unica dei "big data" ed elaborarli per ottenere un risultato ottimale.

Le reti e le piattaforme sono costantemente sotto attacco, questi attacchi sono molto efficienti dati il numero di strumenti che permettono di scannerizzare e valutare i target. Gli avversari stanno già usando il ML per rendere maggiormente evoluti i loro attacchi.

[6, 2, 4]

In questa sezione non andremo ad analizzare tutti i rischi della sicurezza informatica, perché sarebbe troppo dispendioso e non finalizzato verso gli obiettivi che ci eravamo prefissati. Bensì andremo ad analizzare un campo specifico, che dovrebbe permettere un confronto con il caso di studio precedente ovvero l'*Insider Threat Detection* (Individuazione delle minacce interne).

Andiamo di seguito ad analizzare le fasi principali del flusso di lavoro in questo campo, raggrupbandole più genericamente in 2 macrocategorie (preprocessamento dei dati e algoritmi di apprendimento) che analizziamo qui di seguito

2.2.2 Pre Processamento dei dati

Il primo passo è raccogliere i dati e iniziare una prima elaborazione degli stessi, come vediamo dalla figura 2.2 possiamo raggruppare questi dati in 2 categorie:

- Azioni degli Utenti
- Informazioni di Utenti e Struttura

Le azioni degli utenti, raccolti da diversi sistemi di log e altri sistemi di registrazione, devono essere raccolti in maniera costante per rendere utili i sistemi di analisi.

Mentre le informazioni degli utenti e dell'organizzazione della struttura, rappresentano invece dati statici, quali possono essere informazioni personali degli utenti, regole dell'organizzazione, ecc.

I dati di entrambe le categorie possono essere analizzati periodicamente, di solito giornalmente o settimanalmente, a seconda della configurazione dell'organizzazione, la quantità di dati e soprattutto i tempi requisiti per i sistemi di rilevamento.

L'analisi dei dati ha bisogno di incrociare correttamente le informazioni per ogni utente o per ogni dispositivo, basati su un insieme di proprietà come ID utente, ID host, ID azione e tempo.

Una volta raccolti i dati dalle varie sorgenti, le caratteristiche vengono estratte da addestramenti e valutazioni di algoritmi di apprendimento automatico. Possiamo distinguere i dati in dati sequenziali e dati numerici. I dati numerici vengono rappresentati per ogni istanza con un vettore di lunghezza fissato, i quali sono più comunemente applicabili al ML; mentre i dati sequenziali avendo una struttura intrinseca possono rilevare fenomeni maggiormente interessanti considerando l'azione di ciascun utente.

I dati numerici vengono esportati per rappresentare le caratteristiche degli utenti e le attività durante un determinato periodo, parliamo di caratteristiche degli utenti e caratteristiche delle attività. Le caratteristiche degli utenti includono ogni ruolo dell'utente, unità funzionale, dipartimento ecc. . .

Le caratteristiche delle attività invece sono per lo più estratte contando il numero di attività in ogni categoria(log on/off , dispositivo connesso/disconnesso, file, email) in un determinato periodo di tempo.

I dati sequenziali si riassumono come la sequenza ordinata di questi ultimi; e consiste in un ordinato elenco delle azioni intraprese da un utente raccolte di solito quotidianamente o settimanalmente. [10]

2.2.3 Algoritmi di apprendimento

Come raffigurato dallo schema 2.3 prendiamo in considerazione 3 algoritmi: Self-Organizing Map, Hidden Markov Model e Decision Tree; sviluppati per apprendere e modellare i dati per rilevare anomalie/minacce interne.

L'obiettivo è di valutare sia gli algoritmi di apprendimento supervisionati e non.

Gli algoritmi di apprendimento supervisionati sono adatti per analizzare i dati con basi di verità mentre gli algoritmi non supervisionati sono efficaci per generare avvertimenti di attività anomale.

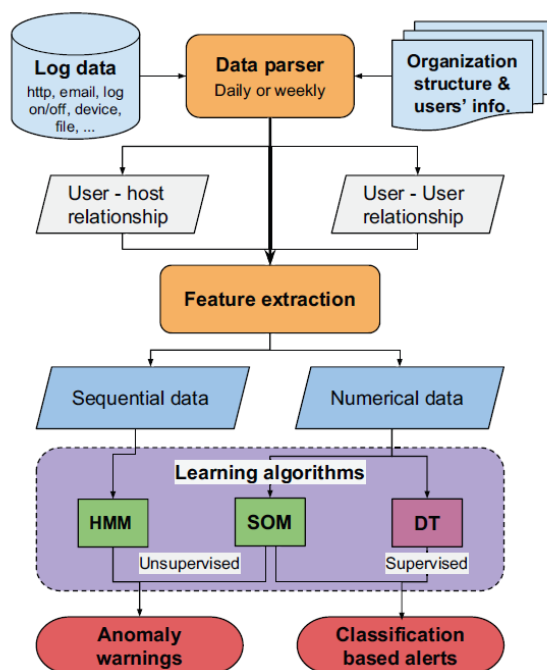


Figura 2.3: Workflow Intrusion Threat

Self-Organizing Map

SOM è un algoritmo non supervisionato basato sulle reti neurali; produce una proiezione di dati non lineare e ordinata da un input preso in spazio multidimensionale.

Il SOM è costituito da una rete di neuroni artificiali i cui pesi sono continuamente adattati ai vettori presentati in ingresso nel relativo insieme di addestramento.

La rete neurale descritta come un insieme di neuroni artificiali, ciascuno con una precisa collocazione e adiacenti gli uni dagli altri sulla mappa rappresentativa degli output, seguendo un processo dove il nodo avente un vettore di pesi più vicino a quello dell'input si aggiudica i restanti, mentre i restanti sono aggiornati in modo da avvicinarli al vettore in ingresso.

Quando un nodo vince una competizione, i pesi dei nodi adiacenti dove più un nodo è lontano dal cosiddetto “vincitore” meno deve essere evidente la sua variazione dei pesi.

Il processo viene ripetuto per ogni vettore dell'insieme di training per un certo numero di cicli, la mappa così riesce ad associare i nodi d'uscita con gruppi e/o schemi ricorrenti nell'insieme di dati in ingresso.

Vengono valutati 2 tipi di approcci per addestrare il SOM: i dati che rappresentano tutte le classi (minacce interne/esterne) altrimenti solo i dati che rappresentano i normali comportamenti dell'utente.

Il primo approccio è applicabile con delle “verità di base” per dati provenienti da più

classi sono disponibili, e la verità[ground-truth] di base vengono usate per etichettare i nodi in fase post-addestramento in base alle migliori unità corrispondenti(nodi) per i dati in ciascuna classe.

Nel secondo caso quando la base di verità per una classe, tipicamente normale è disponibile per l'addestramento, il secondo approccio può essere usato per modellare i dati ed in questo caso viene usato un rilevatore di anomalie post addestramento.

Quando non ci sono informazioni sui dati di addestramento, tutti i dati possono essere usati, l'obiettivo del SOM, in questo caso, è la fase di raggruppamento e visualizzazione dei dati per assistere l'analista umano.

Hidden Markov Model

HMM è un modello statistico nei quali gli stati sono nascosti. Ogni stato nascosto espone un simbolo in un insieme con probabilità prima di passare a un nuovo stato.

Un modello di Markov Nascosto è una Catena di Markov dove gli stati non sono direttamente osservabili direttamente: la catena ha un certo numero di stati, gli stati evolvono secondo una Catena di Markov, ogni stato genera un evento con una certa distribuzione di probabilità che dipende solo dallo stato, l'evento è osservabile lo stato no.

Possiamo usare il HMM con l'algoritmo di Baum-Welch che data una sequenza dell'uscita o un insieme di tali sequenze, l'obiettivo è di trovare l'insieme più probabile per il quale si possano dichiarare le probabilità dell'uscita e di transizione.

Ciò avviene addestrando i parametri dell'HMM mediante il gruppo di dati relativi alle sequenze di azioni di ciascun utente in un determinato periodo di tempo, in questo caso: settimanalmente.

Quindi, per ciascuna nuova sequenza di azioni dell'utente, l'HMM dell'utente viene utilizzato per calcolare la probabilità di log della sequenza.

La sequenza viene segnalata se il valore di probabilità di log è superiore a una certa soglia, se la sequenza non viene flaggata oppure viene considerata da un'analista "non rilevante" viene usata nella combinazione con le azioni precedenti per addestrare il modello HMM dell'utente.

Decision Tree

Un albero di decisione è un modello predittivo, dove ogni nodo rappresenta uno stato mentre ogni arco una determinata proprietà che porterà ad uno stato differente.

Per generare questo albero delle decisioni viene usato l'algoritmo *C4.5*, esso si basa sul costruire un decision tree usando il concetto di informazione entropica creando per ogni nodo una regola "if-then-else".

Per ogni nodo dell'albero, i dati vengono suddivisi in sottoinsiemi; il criterio è soddisfatto più efficacemente scegliendo la funzione e il punto di divisione che fornisce il massimo guadagno di informazioni normalizzate.

C4.5

L'algoritmo C4.5 ha l'obiettivo di costruire un albero decisionale da un insieme di dati di addestramento, usando il concetto di Informazione entropica.

Essendo S un insieme di addestramento

$$S = S_1, S_2, \dots, S_n$$

di campioni già classificati, ogni campione s_i è composto da un vettore p -dimensionale $(x_{(1,i)}, x_{(2,i)}, \dots, x_{(n,i)})$, che rappresentano gli attributi del campione nonché le classi di cui esso appartiene.

L'algoritmo sceglie l'attributo che più efficacemente divide l'insieme in più sottoinsiemi arricchiti in una classe o in un'altra.

Il criterio di divisione è basato sul concetto di informazione entropica, ottenere il massimo guadagno di informazione normalizzato; l'attributo con un alto guadagno di informazioni normalizzate è scelto per creare la divisione. C4.5 è ricorsivo sulle sottoliste partizionate.

[9]

Capitolo 3

Analisi dei Casi di Studio

3.1 Introduzione

L'obiettivo di questo capitolo è provare a trovare un invariante generica tra i diversi workflow riguardanti i case study trovati, e la difficoltà del lavoro sta proprio nel cercare di mettere insieme le varie ricerche nonostante siano molto diverse tra loro.

I case study trovati, insieme ai compagni di corso Luca Genova e Marco Benito Tomasone sono:

- IA nel campo della radiologia
- Previsione della domanda[Forecasting]
- ML nelle reti [Networking]
- IA nel campo della Cybersecurity
- Face2Face Traslation
- Recommender System

L'obiettivo è confrontare le tecniche usate in tutti i precedenti case study, cercando di trovare dei punti in comune con la possibilità di evidenziare un workflow il più possibile generico, al fine di adattare e gestire una Pipeline di lavoro e individuare in maniera ottimale e automatica il risultato che si va cercando.

3.2 Analisi

In tutti i casi di studio analizzati, anche se molto diversi tra loro abbiamo trovati dei passaggi in comune che andremo ad analizzare singolarmente, rilevando caso per caso le particolarità.

3.2.1 Raccolta Dati

La raccolta dati è fondamentale in ogni processo di apprendimento automatico; spesso si diversifica quando viene effettuata la prima elaborazione.

Nonostante i dati risultino in essere molto diversi tra loro, la loro raccolta può trovare dei punti d'accordo.

Spesso questi dati vengono suddivisi a seconda della metodologia di raccolta: il caso di studio dei recommendation system, che diversifica la raccolta dati come dati espliciti(input degli utenti) e impliciti(informazioni raccolte da flussi di dati), mentre nel campo dell'Intrusion Detection vengono differenziati tra dati statici(tramite l'informazione dell'organizzazione e dei dipendenti) e dinamici(quali gli input dei dipendenti stessi).

Nel campo della previsione abbiamo una valutazione dei dati in merito a diversi parametri quali: consistenza, precisione, rilevanza ecc.

Questi dati, però sono ideali di conseguenza verranno puliti e analizzati per lacune e anomalie, verificati per pertinenza e ripristinati.

Nel campo del Networking invece i dati vengono raccolti in due fasi:

- *Offline*: dati importanti per la costruzione e l'addestramento del modello
- *Online*: intendiamo dati che vengono utilizzati in real-time, usati come input o come segnali di feedback per il modello

Nel campo della radiologia non abbiamo questa distinzione, né tantomeno questa mole di dati, in questo campo la problematica della raccolta dati si basa sulla qualità dell'immagine. In questi casi vengono applicati metodi di Deep learning in grado di ridurre il disturbo e migliorare la qualità stessa.

3.2.2 Elaborazione del Modello

In questa fase intendiamo anche la ricerca oppure la costruzione del modello per quei particolari dati, a seconda del tipo di risultato sia una predizione o una classificazione.

Come abbiamo visto nel campo dell’Intrusion Detection per costruire o comunque elaborare un modello diventa fondamentale il tipo di problema che vogliamo andare a risolvere: se vogliamo aspettare che vengano generati degli avvertimenti per attività anomale useremo un modello costruito grazie ad un algoritmo non supervisionato, mentre ad esempio nel campo del Networking la classificazione del traffico diventa un problema per il quale una classificazione efficiente necessita di un algoritmo supervisionato.

Al contrario la maggior parte delle immagini mediche senza annotazioni potrebbero non essere utili per una varietà di scenari di Supervised learning diventa necessario in questo campo degli algoritmi di Reinforcement Learning.

Mentre una volta ottenuta un insieme di immagini mediche contenenti annotazioni, quindi targettizzati, inizia a diventare possibile utilizzare sia algoritmi Supervised Learning, per classificare e raggruppare radiografie simili, sia algoritmi non supervisionati che ad esempio, grazie ad un vettore di caratteristiche ciascuno associato a un caso, il vettore di caratteristiche della nuova immagine può essere confrontato con quelli esistenti utilizzando operatori vettoriali.

Nel campo della previsione, la scelta dei modelli di ML dipende da diversi fattori, tra cui l’obiettivo aziendale e le caratteristiche dei dati.

La maggior parte degli algoritmi di previsione (Regressione lineare, Smoothed Moving Average, ecc.) sono supervisionati, ciò significa che abbiamo a disposizione un set di dati di apprendimento, ad esempio, nella regressione lineare questo set di dati conterrà dei valori di y , dati i valori di x . Questi algoritmi di previsione sono simili a quelli che troviamo nel campo del networking per la previsione del traffico e nella rilevazione delle minacce interne come abbiamo visto con gli algoritmi SOM.

Ricordiamo che SOM non è un algoritmo supervisionato ma riesce comunque a fornire una proiezione sui dati non lineare basato sulle reti neurali.

3.2.3 Risultato

Definiamo l'output desiderato in base a una di queste categorie:

- *Classificazione*: Dato un insieme di più classi il sistema di apprendimento deve produrre un modello che assegni gli input non ancora visti a una o più di queste classi. Solitamente vengono usati sistemi di apprendimento Supervisionati.
- *Regressione*: In questa categoria troviamo sistemi di predizione, tipicamente l'output e il modello sono continui; la predizione di un evento futuro dati i suoi valori in tempi recenti.
- *Clustering*: L'obiettivo di questo tipo di problemi è suddividere gli input in gruppi, non noti precedentemente. Facilmente possiamo intuire che questo tipo di problemi verranno assegnati ad algoritmi tipicamente non supervisionati.

Nella tabella seguente, proviamo a suddividere i vari problemi affrontati nei singoli casi di studio nelle 3 categorie definite in precedenza:

Classification	Regression	Clustering
<ul style="list-style-type: none">• Traffic Classification• Image Recognition [Radiology]• Speech Recognition• Face2Face Traslation• Classification based Alerts	<ul style="list-style-type: none">• Traffic Prediction• Prediction/Forecasting	<ul style="list-style-type: none">• Resource Managment [Networking]• Anomaly warning

Figura 3.1: Partition ML Problem

3.3 Conclusioni



Concludendo cerchiamo di astrarre i nostri problemi evidenziando un pattern generico di invarianza tra i passaggi elencati nei capitoli precedenti. Come notiamo nello schema, i 3 passaggi che diventano fondamentali cercando di automatizzare un processo di ML sono:

1. Raccolta Dati
2. Estrazione dai Dati
3. Selezione del Modello

Come abbiamo detto in precedenza la raccolta dati è uno dei tasselli fondamentali, in seguito l'individuazione del problema rende possibile un'estrazione e una selezione dei dati più completa, cercando di ottenere un insieme conforme al problema che ci troviamo a svolgere.

Una volta individuato il problema e raccolto i dati necessari in input, si proverà a selezionare in maniera automatica il modello adatto per prendere in input la mole di dati individuata.

Il modello preso in input i dati, restituirà il risultato che sarà l'output risultante da tutto il processo.

L'unica tolleranza la possiamo individuare nell'identificazione del problema, che in alcuni casi necessiterà di un intervento manuale; ad esempio, nel forecasting le tecniche predittive sono assai diverse, abbiamo bisogno di qualche indizio in più su quale modello selezionare.

L'efficienza di questo processo, soprattutto nell'identificazione del problema e la selezione del modello, già scarsa nei singoli flussi di lavoro del ML, si ritrova un peggioramento notevole dovuto all'identificazione del problema e alla scelta del modello di riferimento.

Capitolo 4

Orchestrazione di Flussi di lavoro



Figura 4.1: MLops

Dopo aver visto che tra i vari casi di studio troviamo una linea comune nella Pipeline di utilizzo del Machine Learning, cerchiamo degli strumenti che ci permettono di automatizzare le invarianti trovate nel capitolo precedente.

Negli ultimi anni, i ricercatori stanno implementando diversi Framework automatici che permettono di automatizzare il flusso di lavoro al fine di semplificare la costruzione e l'addestramento dei modelli.

Come visto in precedenza i passaggi chiave dove si focalizza l'attenzione di questi framework sono: raccolta dati, creazione e implementazione del modello e successivamente la distribuzione, permettendo la riproduzione e il monitoraggio.

Le Pipeline ML, inoltre, aiutano a migliorare le prestazioni e la gestione dell'intero modello.

Possiamo semplificare lo schema visto in precedenza lasciando la gestione del modello assegnata al Framework.

Gli strumenti per l'orchestrazione ML sono usati per automatizzare e gestire i workflow e le infrastrutture delle Pipeline con semplici interfacce, aiutando i data scientists e il team di ML a concentrarsi solo sul necessario.

Orchestrare un flusso di lavoro è fondamentale per un'azienda che investe nel ML, diventa dunque necessario capire come automatizzare il maggior numero di risorse, tra cui l'estrazione dell'output del modello con dei monitoraggi costanti durante la fase di produzione.

Questi campi che stiamo analizzando sono un sottoinsieme di MLOps: un insieme di pratiche per la collaborazione e la comunicazione tra data scientist e professionisti delle operazioni. Applicando queste tecniche pratiche aumenta la qualità finale, viene semplificato il processo di gestione e si automatizza l'implementazione di modelli di ML e DL in ambienti di produzione su larga scala.(figura 4.1) [3, 7, 20]

4.1 Framework

Prima di elencare una lista di tools di gestori di risorse di ML, diventa necessario anticipare una distinzione tra un software in fase di ricerca e un software già a livello di produzione ovvero già pronto per essere utilizzato a livello aziendale.

Un software ancora in fase di ricerca, spesso utilizzato a livello accademico, offre delle funzionalità magari non ancora rilasciate nel mondo "Aziendale" avendo delle "particolarità" che altri software *"in produzione"* non possiedono.

Questi ultimi, a loro volta, si differenziano grazie a caratteristiche quali l'affidabilità e le performance.

Senza aver la pretesa di essere completi, citiamo di seguito alcuni strumenti più conosciuti, già rilasciati in produzione, per l'orchestrazione di flussi di lavoro dell'apprendimento automatico.

4.1.1 ZenML

ZenML è uno strumento open-source per le operazioni di ML. Lo strumento si focalizza sui problemi di riproduzioni basati sulla produzione, come le difficoltà di versioning e modelli, organizzazione di workflow di ML e la distribuzione. Può lavorare a fianco a un altro strumento di orchestrazione dei flussi di lavoro per fornire un semplice percorso per rilasciare il modello ML in produzione.

È possibile verificare con precisione dati, modelli e configurazioni. Ti consente di valutare in maniera semi-automatica il modello, confrontare le Pipeline di addestramento e distribuire la preelaborazione nel cloud.[19]

A screenshot of a Python terminal window with a dark background. The window title is 'python'. It shows a sequence of commands to create and run a ZenML pipeline. The commands are: 'import zenml', 'p = zenml.TrainingPipeline(name='My Awesome Pipeline')', 'p.add_split(RandomSplit())', 'p.add_preprocessing(StandardPreprocessor(features='all'))', 'p.add_trainer(FeedForwardTrainer())', '(...)', and 'p.run(orchestration='gcp', processing='dataflow')'.

```
python

» import zenml

» p = zenml.TrainingPipeline(name='My Awesome Pipeline')

» p.add_split(RandomSplit())

» p.add_preprocessing(StandardPreprocessor(features='all'))

» p.add_trainer(FeedForwardTrainer())

» (...)

» p.run(orchestration='gcp', processing='dataflow')
```

Figura 4.2: ZenML

Come vediamo dalla figura 4.2 ogni metodo implementa uno specifico passaggio del workflow, vincolando i passaggi da eseguire sul modello a discrezione del Framework stesso.

Se l'addestramento (*add_trainer*) appare prima del processamento dei dati (*add_preprocessing*), sarà compito del framework nel metodo (*run*) organizzare l'ordine corretto delle operazioni da eseguire nel modello.

4.1.2 Kedro

Kedro é uno strumento open-source di orchestrazione basato su Python, che permette di creare workflow per ML riproducibili, mantenibili e modulari, semplificando i processi e rendendoli più accurati.

Kedro integra l'ingegneria del software in un ambiente di apprendimento automatico, con concetti quali modularità, divisione degli interessi e versioning.

Astraendo la Pipeline, è possibile automatizzare le dipendenze tra il codice Python e la visualizzazione del Workflow; rendendo più flessibile lo sviluppo di progetti di *Data Science*.

L'obiettivo principale è la creazione di codice di data science gestibile per affrontare le carenze di Jupiter (applicazione web open-source, utile per condividere documenti che contengono codice live, equazioni e grafi ecc.)

Questo strumento crea un'interazione più semplice tra i vari livelli, e fornisce un efficiente ambiente di coding con codice modulare e riutilizzabile.[8]

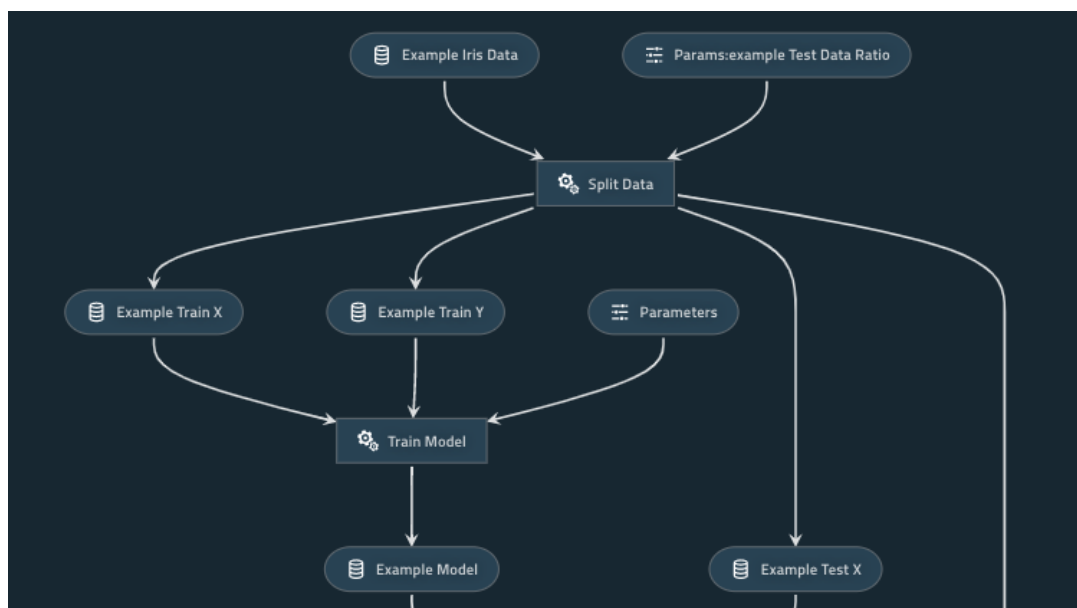


Figura 4.3: Kedro

4.1.3 Flyte

Flyte è uno strumento open-source di alta fascia che permette di facilitare la creazione di flussi di lavoro del ML. È una piattaforma di programmazione ad elaborazione distribuita e strutturata, con flussi di lavoro concorrenti, scalabili e mantenibili per l'apprendimento automatico e l'elaborazione dei dati.

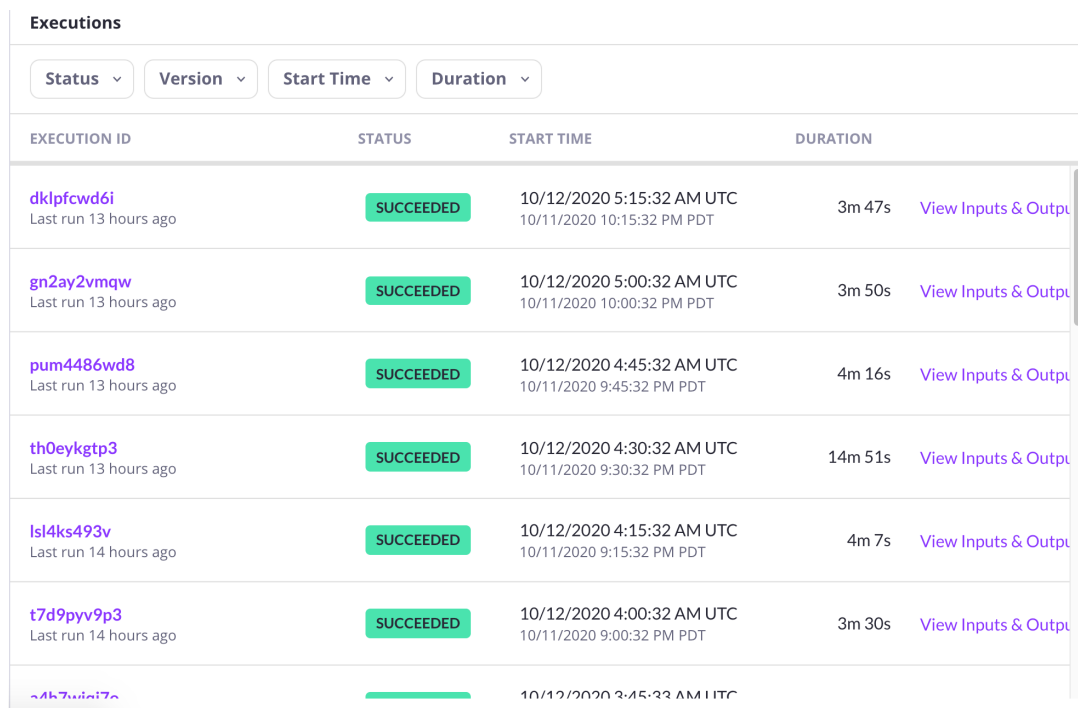
Flyte é scritto in Python ed é progettato per supportare flussi di lavoro ML complessi scritti in Python, Java e Scala.

Flyte gestisce già più di dieci mila workflow, è basato su Kubernetes e offre portabilità, scalabilità e affidabilità.

L'interfaccia è elastica, intuitiva e facile da usare; offre inoltre parametri, linee di dati e caching per organizzare i workflow.

L'intera piattaforma è dinamica ed estensibile attraverso vari plug-in per assistere la creazione e l'implementazione dei vari workflow. Tali Workflow, a loro volta, possono essere reiterati, annullati, sperimentati e condivisi per accelerare il processo di sviluppo dell'intero team.

Poiché ogni entità in Flyte è immutabile, insieme a ogni modifica esplicita assunta come nuova versione, diventa semplice ed efficiente iterare, sperimentare e tornare indietro nei Workflow.[5]



The screenshot shows the 'Executions' page in the Flyte Console. At the top, there are filter buttons for 'Status', 'Version', 'Start Time', and 'Duration'. Below these is a table with the following columns: 'EXECUTION ID', 'STATUS', 'START TIME', and 'DURATION'. The table lists several successful executions, each with a unique ID, a 'SUCCEEDED' status, a timestamp in both UTC and PDT, and a duration. A 'View Inputs & Outputs' link is provided for each entry.

EXECUTION ID	STATUS	START TIME	DURATION
dklpfcwd6i Last run 13 hours ago	SUCCEEDED	10/12/2020 5:15:32 AM UTC 10/11/2020 10:15:32 PM PDT	3m 47s
gn2ay2vmqw Last run 13 hours ago	SUCCEEDED	10/12/2020 5:00:32 AM UTC 10/11/2020 10:00:32 PM PDT	3m 50s
pum4486wd8 Last run 13 hours ago	SUCCEEDED	10/12/2020 4:45:32 AM UTC 10/11/2020 9:45:32 PM PDT	4m 16s
th0eykgtp3 Last run 13 hours ago	SUCCEEDED	10/12/2020 4:30:32 AM UTC 10/11/2020 9:30:32 PM PDT	14m 51s
lsl4ks493v Last run 14 hours ago	SUCCEEDED	10/12/2020 4:15:32 AM UTC 10/11/2020 9:15:32 PM PDT	4m 7s
t7d9pyv9p3 Last run 14 hours ago	SUCCEEDED	10/12/2020 4:00:32 AM UTC 10/11/2020 9:00:32 PM PDT	3m 30s
~4h7wini7e		10/12/2020 3:45:33 AM UTC	

Figura 4.4: Flyte Console

4.1.4 MLRun

MLRun è uno strumento open-source di orchestrazione di Pipeline di ML. Offre un livello di astrazione per un'ampia varietà di stack tecnologici e consente a *data engineer* e *data scientist* di definire le funzionalità e i modelli, semplificando e accelerando il percorso verso la produzione.

Esso integra un vasto reparto di strumenti e plugin per lavorare con caratteristiche e modelli insieme alla distribuzione del flusso di lavoro.

MLRun dispone di un archivio di funzionalità e artefatti per controllare l'inserimento, l'elaborazione e l'archiviazione dei dati su più repository e con tecnologie diverse, rendendo possibile anche l'automatizzazione.

È un servizio "Server-less" elastico per convertire semplice codice in scalabili e organizzati "Microservizi"; diventa così più semplice e leggero fare esperimenti, allenare i modelli e testarli, e implementare dei flussi di lavoro della Pipeline in tempo reale.

L'interfaccia utente complessiva ha una struttura centralizzata per gestire flussi di lavoro di ML. Le caratteristiche principali includono una rapida implementazione, elastica scalabilità, gestione delle funzionalità e un'usabilità flessibile.[11]

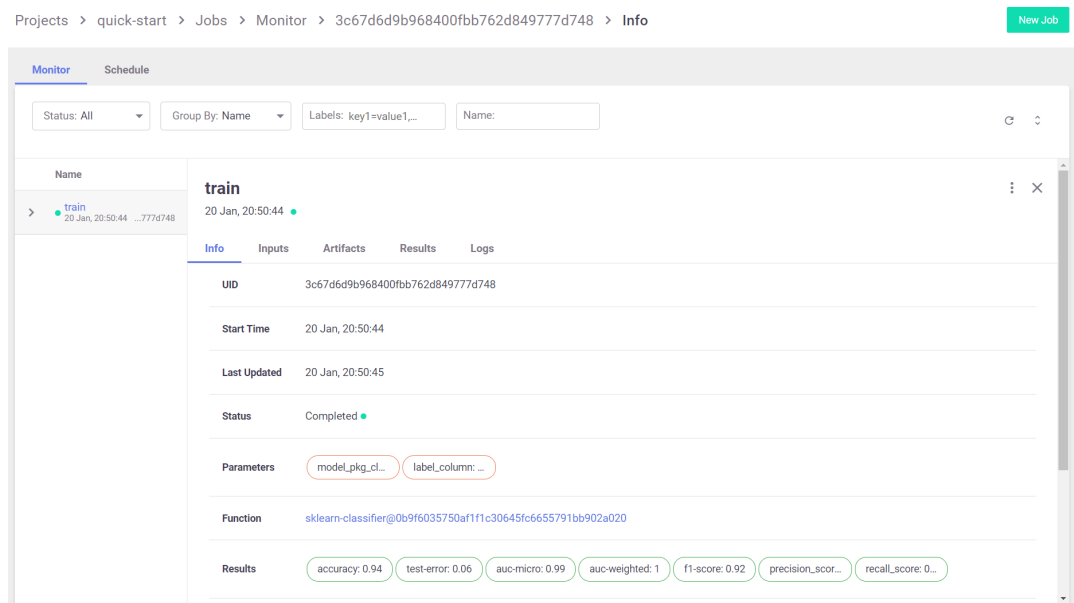


Figura 4.5: MLRun Console

4.2 Conclusioni

Come spesso accade nel campo del *Computer Science* uno stesso problema può essere risolto attraverso l'utilizzo di diversi strumenti.

I framework che abbiamo selezionato sono open-source (tutti disponibili su *github*) e forniscono tutti una orchestrazione di flussi di lavoro per l'apprendimento automatico.

Kedro e Flyte sono i più diffusi, tra cui Flyte, come citato nella sezione precedente, gestisce già più di 10 mila flussi di lavoro. Ovviamente in base al caso di studio che prenderemo in considerazione si andrà a preferire un Framework piuttosto che un altro, a seconda delle caratteristiche che li differenziano, e che in parte abbiamo visto nella sezione precedente.

Ad esempio nel campo della radiologia, i medici si troverebbero in difficoltà ad usare FW quali ZenML come libreria di Python come viceversa può essere un approccio più chiaro per sistemisti che lavorano nel campo del Networking, raccogliendo dati in real-time. Framework quali Flyte hanno un'interfaccia intuitiva che permette anche a utenti che non lavorano costantemente nell'apprendimento automatico di fare predizione attraverso dei modelli.

Sia MLRun e Flyte sono estendibili attraverso diversi plug-in i quali possono diventare fondamentali qualora i dati in entrata siano di tipologie particolari (quali le immagini nel campo della radiologia o il Face2Face Translation).

Nel campo dell'Intrusion Detection è necessario che ogni utente venga osservato *concorrentemente* e questa possibilità viene garantita da Flyte.

Glossario

bias costante che influenza una relazione tra le unità di input e quelle di output 9

catena di Markov Una catena di markov(finita) é un sistema dotato di un numero finito di stati $\{1, 2, \dots, n\}$ in cui la probabilità che il sistema passi dallo stato i allo stato j al tempo $k - 1$ é $p_{ij}(k)$ 17

CDN (*Content Delivery Network*) Sistema di computer collegati in rete attraverso Internet, che collaborano in maniera trasparente sotto forma di sistema distribuito per ripartire contenuti e servizi agli utenti finali, riducendo anomalie dovute all'eccessivo numero di richieste al server principale 8

deep learning il Deep Learning é un campo di ricerca del ML che si basa su diversi livelli di rappresentazione. Intendiamo un insieme di reti neurali artificiali organizzate in diversi strati, dove ogni strato calcola i valori per quello successivo affinché l'informazione venga elaborata in maniera sempre più completa. 11, 20

Framework architettura di supporto sulla quale un software può essere progettato e realizzato, facilitandone lo sviluppo 2, 4, 24-26, 30

informazione entropica l'entropia di una sorgente di un messaggio é l'informazione media contenuta in ogni messaggio emesso. Ovvero l'informazione contenuta in un messaggio é tanto più grande quanto meno probabile era. Un messaggio scontato, che ha un'alta probabilità di essere emesso dalla sorgente contiene poca informazione a confronto di un messaggio inaspettato che contiene maggiori informazioni. [14] 18

kubernetes Sistema open-source di orchestrazione e controllo di servizi containerizzati, uno dei modi ottimali al giorno d'oggi per distribuire e eseguire applicazioni 28

microservizi I Microservizi sono un approccio particolare alla realizzazione di applicazioni. Una applicazione diventa un insieme di servizi, i quali possono essere compilati e implementati in maniera indipendente. Inoltre ogni Microservizio può funzionare, o meno, senza compromettere gli altri. 29

over-fitting In presenza di un numero eccessivo di parametri rispetto al numero di osservazioni, un modello statistico si adatta ai dati osservati. Nei casi in cui l'apprendimento é stato effettuato troppo a lungo o dove c'era uno scarso numero di esempi di allenamento, il modello potrebbe adattarsi a caratteristiche che sono specifiche solo del *training set* ma che non hanno riscontro nel resto dei casi. 10

payload-based tecnica di classificazione basata sul contenuto del pacchetto, associazione *contenuto/protocollo* 10

pipeline Insieme di piú componenti collegati tra loro, in modo che il risultato prodotto da uno degli elementi sia l'ingresso di quello immediatamente successivo 19, 24–27, 29

port-based tecnica di classificazione basata sull'associazione *porta/protocollo* 10

server-less Il ServerLess computing é un modello di sviluppo Cloud Native che consente agli sviluppatori di creare ed eseguire applicazioni senza gestire i server, i quali saranno astratti dallo sviluppo dell'app. La gestione delle risorse, la manutenzione e la scalabilitá del server saranno tutti compiti affidati al provider dei servizi cloud. 29

throughput Quantitá di dati trasmessi in un unitá di tempo (*Tasso di elaborazione o trasmissione*) 8

under-fitting Scenario che si presenta quando in un modello statisco o nell'algoritmo di ML non si riesce a catturare adeguatamente la struttura sottostante i dati. Si verifica quando il modello o l'algoritmo non si adattano abbastanza ai dati. La classificazione soffre di un'eccessiva discrepanza. (Il modello é troppo semplice) 10

Bibliografia

- [1] Raouf Boutaba et al. “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities”. In: *Journal of Internet Services and Applications* 9.1 (2018), pp. 1–99.
- [2] Kirk Bresniker et al. “Grand challenge: Applying artificial intelligence and machine learning to cybersecurity”. In: *Computer* 52.12 (2019), pp. 45–52.
- [3] Author Prince Canuma. *Machine Learning Model Management: What It Is, Why You Should Care, and How to Implement It*. 2021. URL: <https://neptune.ai/blog/machine-learning-model-management>.
- [4] Akhilesh Deshmukh. *Debunking the Myths. How Machine Learning (ML) Benefits Cyber Security*. 2020. URL: <https://www.securityhq.com/blog/debunking-the-myths-how-machine-learning-ml-benefits-cyber-security/>.
- [5] *Flyte Framework*. URL: <https://flyte.org/>.
- [6] James B Fraley e James Cannady. “The promise of machine learning in cybersecurity”. In: *SoutheastCon 2017*. IEEE. 2017, pp. 1–6.
- [7] Author Krissanawat Kaewsanmua. *Best Workflow and Pipeline Orchestration Tools – Machine Learning Guide*. 2021. URL: <https://neptune.ai/blog/best-workflow-and-pipeline-orchestration-tools>.
- [8] *Kedro Framework*. URL: <https://kedro.readthedocs.io/en/stable/>.
- [9] Duc C Le e A Nur Zincir-Heywood. “Evaluating insider threat detection workflow using supervised and unsupervised learning”. In: *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2018, pp. 270–275.
- [10] Duc C Le, Nur Zincir-Heywood e Malcolm I Heywood. “Analyzing data granularity levels for insider threat detection using machine learning”. In: *IEEE Transactions on Network and Service Management* 17.1 (2020), pp. 30–44.
- [11] *MLRun Framework*. URL: <https://www.mlrun.org/>.
- [12] Mohammad Azmi Ridwan et al. “Applications of Machine Learning in Networking: A Survey of Current Issues and Future Challenges”. In: *IEEE Access* (2021).

- [13] Mowei Wang et al. “Machine learning for networking: Workflow, advances and opportunities”. In: *Ieee Network* 32.2 (2017), pp. 92–99.
- [14] Wikipedia. *Entropia (teoria dell’informazione)* — *Wikipedia, L’enciclopedia libera*. [Online; in data 29-settembre-2021]. 2021. URL: [http://it.wikipedia.org/w/index.php?title=Entropia_\(teoria_dell%27informazione\)&oldid=120890494](http://it.wikipedia.org/w/index.php?title=Entropia_(teoria_dell%27informazione)&oldid=120890494).
- [15] Wikipedia contributors. *Machine learning* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 27-September-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=1046345973.
- [16] Wikipedia contributors. *Reinforcement learning* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 27-September-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=1046562875.
- [17] Wikipedia contributors. *Supervised learning* — *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Supervised_learning&oldid=1033187514. [Online; accessed 27-September-2021]. 2021.
- [18] Wikipedia contributors. *Unsupervised learning* — *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=1043586308. [Online; accessed 27-September-2021]. 2021.
- [19] *ZenML Framework*. URL: <https://zenml.io/>.
- [20] Martin Zinkevich. *Best Practices for ML Engineering*. 2021. URL: https://developers.google.com/machine-learning/guides/rules-of-ml#rule_4_keep_the_first_model_simple_and_get_the_infrastructure_right.