DIGITAL FUTURE

OR2 – Piattaforma Digital Future: Metodi e Tecniche

Metodi e Tecniche per il Midlleware

TR 2.1

AUTORE: EXPRIVIA S.P.A

Indice dei contenuti

INDICE DEI CONTENUTI **3**
INDICE DELLE FIGURE **7**
INDICE DELLE TABELLE **9**
1. INTRODUZIONE **10**
2. ARCHITETTURA LOGICA DEL MIDDLEWARE **14**
3. CONNECTOR LAYER **21**
3.1 COMMUNICATION MANAGER 21
3.1.1 IoT Broker *22*
3.1.1.1 Il protocollo MQTT 23
3.1.1.2 Aspetti sulla sicurezza nella comunicazione 30
3.1.2 Event Bus *32*
3.1.3 API Gateway *36*
3.2 SECURITY MANAGER 38
3.2.1 Architettura Logica *43*
3.2.1.1 Identity Manager 43
3.2.1.2 Authentication Manager 43
3.2.1.3 Authorization Manager 44
3.2.2 Modelli di autorizzazione *44*
3.2.3 Protocolli di Autenticazione ed Autorizzazione *46*
3.2.4 Autenticazione e Autorizzazione dei Client *51*
3.2.4.1 Autenticazione e Autorizzazione dei sensori 52
3.2.4.2 Security Pattern 53
4. OPPORTUNISTIC INTEGRATOR LAYER **56**
4.1 ARCHITETTURA LOGICA DEL LAYER 58
5. ANALYTICS LAYER / DATA GOVERNANCE **63**

5.1 ARCHITETTURA LOGICA DEL LAYER 66
5.2 DATA SCIENCE TOOL 66
5.2.1 Data Preparation 69
5.2.1.1 Data Exploration 71
5.2.1.2 Data Characterization 73
5.2.1.3 Data Correlation 76
5.2.2 Data Modeling 79
5.2.3 Data Prediction 82
5.3 MACHINE LEARNING LIBRARIES (ML) 84
5.3.1 Classificazione 84
5.3.2 Regressione 85
5.3.3 Clustering 86
5.3.4 Deep Learning 87
5.3.4.1 Integrazione di TensorFlow in Digital Future Middleware 89
5.3.4.2 Soluzioni raggiungibili tramite Deep Learning 90
5.3.4.3 Approcci Deep Learning per l'analisi dei dati 92
5.4 PROCESSING ENGINE 115
5.4.1 Batch Processing Engine 115
5.4.2 Stream Processing Engine 116
5.5 EXPLANATION TOOL 117
5.6 ALGORITHM RECOMMENDER TOOL 120
5.6.1 Recommender di metodi di Classificazione in Machine Learning 122
5.6.1.1 Fattori di valutazione degli algoritmi di classificazione in apprendimento automatico supervisionato 128
5.6.1.2 Ranking Recommender per metodi di classificazione in apprendimento automatico supervisionato 135
5.6.2 Recommender di metodi di Regressione in Machine Learning 136

5.6.2.1 Fattori di valutazione degli algoritmi di regressione in apprendimento automatico supervisionato 142

5.6.2.2 Ranking Recommender per metodi di regressione in apprendimento automatico supervisionato 147

5.6.3 Recommender di metodi di Clustering in Machine Learning 148

5.6.3.1 Fattori di valutazione degli algoritmi di clustering in apprendimento automatico non supervisionato 153

5.6.3.2 Ranking Recommender per metodi di clustering in apprendimento automatico non supervisionato 157

5.7 DATA STORAGE 159

5.7.1 Structured & Unstructured Data Store 160

5.7.2 ML Model Store 164

6. MULTIMEDIA LAYER 165

6.1 FUNZIONALITÀ PREVISTE 165

6.2 TECNICHE E TECNOLOGIE 166

6.3 ARCHITETTURA LOGICA DEL LAYER 167

6.3.1 Media Repository 168

6.3.2 Media Index Manager 168

6.3.3 Media Search Manager 169

6.3.4 Annotation Service Manager 169

7. SEMANTIC AND ONTOLOGICAL LAYER 171

7.1 FUNZIONALITÀ PREVISTE E COMUNICAZIONE CON ALTRE COMPONENTI DEL MIDDLEWARE . 171

7.2 TECNICHE E TECNOLOGIE 172

7.3 ARCHITETTURA LOGICA DEL LAYER 174

7.3.1 Semantic Annotation Manager 175

7.3.2 Inference Engine 189

7.3.3 Ontological Connector 196

7.3.4 Ontology Repository 198

8. EVENT LAYER 201

8.1 FUNZIONALITÀ PREVISTE E COMUNICAZIONE CON ALTRE COMPONENTI DEL MIDDLEWARE . 201

8.2 TECNICHE E TECNOLOGIE 203
8.3 ARCHITETTURA LOGICA DEL LAYER 207
9. REFLECTIVE LAYER 209
9.1 REFLECTION PATTERN 210
9.2 ARCHITETTURA LOGICA DEL LAYER 211
9.2.1 Meta-Object Store 213
9.2.2 Meta-Object Editor 214
9.2.3 Reflective Rule Engine 216
10. BIBLIOGRAFIA 218

Indice delle figure

Figura 1: Architettura Logica del Middleware 15
Figura 2: Architettura Logica del sub-layer Communication Manager 22
Figura 3: Generico Modello Publisher - Subscriber 33
Figura 4: Architettura del componente Event Bus 36
Figura 5: API Gateway Workflow 38
Figura 6: Security Manager 39
Figura 7: Gestione accessi in modalità autonoma 41
Figura 8: Gestione centralizzata degli accessi 42
Figura 9: Modalità di funzionamento di KERBEROS 48
Figura 10: Modalità di funzionamento di OAuth 2.0 50
Figura 11: Processo di verifica delle regole di autorizzazione 53
Figura 12: PSM Security Model 54
Figura 13: PDM Security Model 55
Figura 14: BPMN del Processo Inizializzazione Dispositivo 58
Figura 15: Architettura Logica del Opportunistic Integrator Layer 58
Figura 16: Architettura Logica del Opportunistic Integrator Layer 62
Figura 17: Attività previste nel Data Science Life Cycle 65
Figura 18: Architettura Logica del Data Governance Layer 66
Figura 19: Framework CRISP-DM 68
Figura 20: Livelli di Astrazione di una rete neurale 88
Figura 21: Calcolo distribuito con Spark e TensorFlow 89
Figura 22: Topologia del MultiLayer Perceptron 94
Figura 23: Topologia di una RBM 95
Figura 24: Topologia di un AutoEncoder 96
Figura 25: Topologia di un VAE 97
Figura 26: Topologia di una DBN 98
Figura 27: Topologia di una DSN 99
Figura 28: Topologia di una GAN 101
Figura 29: CNN e Computer Vision 102
Figura 30: Architettura CNN ad alto livello 103
Figura 31: Strato convolutivo con volumi di input e output 104
Figura 32: Operazione di Convoluzione 105
Figura 33: Generazione di un volume di output di attivazione 106
Figura 34: Topologie di RNN 108
Figura 35: Rete Feed-Forward multistrato "appiattita" 110
Figura 36: Connessioni ricorrenti su nodi degli strati nascosti 110
Figura 37: RNN srotolata lungo l'asse del tempo 111
Figura 38: Diagramma del blocco LSTM 111
Figura 39: Cella GRU 113
Figura 40: Esempio di rete neurale ricorsiva 114
Figura 41: Stream Processing Pipeline 117
Figura 42: Data Storage Layer 159
Figura 43: Architettura Logica del Multimedia Layer 167

Figura 44: Architettura Logica del Semantic and Ontological Layer 175
Figura 45: Classificazione ontologica: le frecce spesse rappresentano relazioni di specializzazione [129] 197
Figura 46: Tipica Architettura CEP [143] 204
Figura 47: Batch and Stream Processing [145] 205
Figura 48: Complex Event Processing Pattern Detection [146] 206
Figura 49: Complex Event Processing with Apache Flink [149] 207
Figura 50: Architettura Logica dell'Event Layer 207
Figura 51: Architettura Logica del Reflective Layer 212
Figura 52: Entità definite nel File di Configurazione e loro Relazioni 215
Figura 53: Dettaglio Architetturale del Reflective Layer 217

Indice delle tabelle

Non è stata trovata alcuna voce dell'indice delle figure.

1. Introduzione

Con l'avvento dell'Internet Of Things, il numero di dispositivi e delle apparecchiature connesse alla rete sta aumentando in maniera vertiginosa, basti pensare ai sensori per le automobili, elettrodomestici, telecamere o anche quelli utilizzati per il rilevamento dei parametri vitali.

Tutti questi dispositivi elettronici scambiano continuamente una grande mole di dati sulla rete globale al fine di ottimizzare in tempo reale i processi produttivi. La diffusione di questo fenomeno, insieme ad altri fattori come l'utilizzo sempre più crescente dei social media, dove gli utenti pubblicano quotidianamente un'enorme quantità di dati come tweet, post, video, immagini, hanno messo in evidenza come le infrastrutture di calcolo tradizionali non soddisfano più in modo adeguato la richiesta crescente di capacità computazionale e storage, al fine di elaborare in tempi brevi questa "esplosione" di dati.

Esistono diversi casi significativi in cui non basta solo processare una grande quantità di dati, ma bisogna farlo il più velocemente possibile.

Proporzionalmente a questi fattori anche la mole di dati che vengono generati e immagazzinati sta aumentando in maniera vertiginosa dando luogo alla nascita di un nuovo concetto: i BigData. All'interno dei giganteschi dataset, inoltre, possono essere presenti informazioni che elaborate correttamente, permettono quindi di ottenere delle informazioni di valore strategico che aiutano il processo di analisi decisionale aziendale a qualsiasi livello, dalla produzione di un prodotto fino all'azione di manutenzione che gli concerne. Nasce, di conseguenza, la necessità di far ricorso a nuovi strumenti che possano sfruttare la potenza di calcolo oggi offerta dalle architetture più complesse che comprendono, sotto un unico sistema, un insieme di host utili per l'analisi, come ad esempio i Cluster, oltre alle nuove modalità di analisi dei dati che fanno sempre più utilizzo delle tecniche di intelligenza artificiale.

Nel panorama più recente, sono stati introdotti, a tal fine, numerosissimi framework, proprietari e open-source, che agevolano l'intero processo, fornendo la possibilità di elaborare grandi dataset sfruttando la potenza di calcolo di cluster di commodity hardware, mettendo a disposizione nuove tecnologie e strumenti consoni a esigenze diverse dalla classica analisi batch, come, ad esempio, il Data Stream Processing. La grande quantità di dati, infatti, non sempre necessita di obbligatoria storicizzazione. Ai fini di analisi è possibile elaborare flussi continui di dati, elaborati direttamente in memoria, raccolti da sorgenti (e.g. sensori, web, architetture di rete) geograficamente distribuite che li producono con data rate non predicibili, al fine di ottenere risposte in modo tempestivo.

Al tal fine, la piattaforma di interoperabilità che si vuole realizzare nel progetto mira ad affrontare in maniera innovativa la problematica scientifico-applicativa della *Big Data Intelligence*, investigando e sperimentando l'insieme di strumenti e tecniche che consentano la *raccolta*, l'*archiviazione* e la *gestione* di grandi patrimoni informativi, avendo poi la capacità di estrarre da essi informazioni utili per *fare delle predizioni su eventi futuri, attraverso tecniche di machine learning e deep learning*. La piattaforma che sarà progettata, porrà particolare attenzione al processo di **Data Mining** che è la scienza più efficace quando non si hanno informazioni a priori sul contesto da studiare, andando a bilanciare la conoscenza degli esperti dell'ambiente del dato e la potenza di calcolo degli elaboratori. Come già accennato, l'analisi del dato permetterà di effettuare due tipi di attività, ovvero l'analisi predittiva e descrittiva, i cui risultati sono complementari al raggiungimento dello scopo. Dove l'analisi descrittiva si ferma a trovare i pattern e nuove informazioni che l'esperto può trarre per effettuare degli studi, la predittiva invece permette di produrre un modello eseguibile sotto forma di codice, utile alla predizione, estimazione e identificazione di un processo. A tal scopo la piattaforma d'interoperabilità avrà un componente definito *Data Governance Layer* che dovrà consentire efficacemente ed efficientemente

- integrazioni di dati voluminosi, eterogenei, complessi (caratteristiche peculiari dei Big Data)

- loro gestione
- analisi anche complesse pervenendo a modelli predittivi
- esecuzione degli algoritmi in modalità batch e modalità data streaming

La piattaforma di interoperabilità dovrà anche recepire i più recenti trend architetturali come l'Edge Computing e i Microservizi.

Con il termine **Edge Computing** si indica l'elaborazione delle informazioni ai margini della rete, dove i dati stessi vengono prodotti. I benefici principali derivanti dall'utilizzo di queste tecnologie sono la riduzione della latenza di elaborazione, che permette risposte in tempo reale, e il risparmio di banda, inviando al data center informazioni già elaborate e quindi di minori dimensioni. L'Edge Computing rappresenta, quindi, un nuovo concetto di architettura per ambienti IoT, ma non mette in gioco nuovi componenti di rete. Piuttosto, vengono utilizzate tecnologie già consolidate in una struttura più compatta e con un nuovo nome. La piattaforma che si vuole progettare, dovendo consentire la registrazione di sensori e dispositivi di qualunque genere, collezionare dati dal campo, sottoporli ad attività di data analytics con strumenti di intelligenza artificiale e data stream processing, dovrà essere opportunamente ripensata con una architettura operante a due livelli con differenti trade-off tra completezza e impiego di risorse di calcolo:

- al margine (edge) della rete possono essere effettuate elaborazioni real-time di tipo opportunistico, che privilegiano analisi rapide anche se meno approfondite, utilizzando dispositivi vicini al livello di campo dai costi e dalle capacità computazionali contenuti;
- al nucleo dell'infrastruttura informatica – proprietaria, in cloud o ibrida – restano i compiti di raccolta dei dati, opportunamente aggregati ed eventualmente anonimizzati, per l'analisi massiva anche off-line, caratterizzata da volumi più elevati, da complessità e diversificazione maggiori delle metodologie analitiche e da vincoli laschi sui tempi di risposta.

L'architettura a **Microservizi** rappresenta uno stile architetturale emergente, grazie al quale ciascuna delle sotto-parti atomiche di un sistema viene isolata in un applicativo

indipendente, che collabora con gli altri per creare il comportamento del sistema nel suo complesso. La piattaforma sarà, pertanto, suddivisa nelle sue funzioni di base, dove ogni funzione, denominata servizio, potrà essere compilata e implementata in modo indipendente. Pertanto, i singoli servizi dovranno poter funzionare, o meno, senza compromettere gli altri.

In definitiva, la piattaforma di interoperabilità può essere considerata un insieme di servizi / funzionalità che potranno essere utilizzati in maniera opportunistica da parte di chi, in una organizzazione, ha bisogno di raccogliere, analizzare e comprendere i dati generati dal contesto operativo che si vuole tenere sotto controllo, al fine di individuare fenomeni di interesse. Ovviamente, data la natura opportunistica, non tutti i componenti della piattaforma dovranno essere obbligatoriamente sollecitati durante il suo utilizzo. La piattaforma sarà assimilabile ad un insieme di strumenti attraverso i quali organizzare le attività di analisi sui dati raccolti fornendo insight significativi a chi ne ha bisogno.

Le soluzioni verticali / tenant che utilizzeranno la piattaforma potranno istanziare tali servizi personalizzandoli per le proprie necessità e facendo in modo che operino sui dati di proprio interesse. Alla luce di quanto esposto, la piattaforma, con i servizi di analisi dati attivati a run time sarà assimilabile ad una estensione alla logica di business che verrà implementata dalle varie soluzioni verticali, occupandosi essenzialmente di individuare i fenomeni di interesse per le soluzioni verticali afferenti a diversi domini applicativi, attraverso l'analisi approfondita dei dati che arrivano dal campo.

2. Architettura Logica del Middleware

In linea di massima, il Middleware della Piattaforma DF dovrà trattare le seguenti tematiche:

- *Interoperabilità*: si ha spesso a che fare in IoT con la necessità di far condividere informazioni tra dispositivi (e applicazioni) di tipo diverso, che devono cooperare a uno scopo comune. L'interoperabilità si può suddividere in tre tipologie:
    - Di rete: l'interoperabilità di rete riguarda protocolli per lo scambio di informazioni attraverso le reti, senza però pensare al contenuto delle informazioni. Riguarda i problemi di connettività di base reltivi ai livelli dello stack TCP-IP (in questo caso livello fisico, data link, di trasporto, sessione e di applicazione).
    - Sintattica: riguarda il formato e la struttura della codifica delle informazioni scambiate (in questo caso riguarda il livello di presentazione e di applicazione dello stack TCP-IP).
    - Semantica: riguarda le regole per interpretare le informazioni.
- *Consapevolezza del contesto*: il middleware deve essere consapevole del contesto in cui opera, in modo tale da fornire all'utente le informazioni necessarie nel momento in cui ne ha bisogno. Si può raggiungere questo obiettivo in due passaggi:
    - Rilevazione del contesto: vengono raccolti i dati e identificati i fattori significativi che possono incidere nella risposta da fornire all'utente.
    - Elaborazione del contesto: il tutto viene elaborato e viene scelta la risposta da fornire in base al contesto.
- *Privacy e sicurezza:* dato che in gioco c'è la raccolta di dati, questi devono essere protetti a dovere. Un modo per realizzare questo è utilizzare reti peer to peer sicure, oppure implementare dei sistemi di autenticazione dei dati, gestione di permessi e altro.

- *Gestione dei dati:* i dati raccolti in gioco sono innumerevoli, e di ogni tipo. Questi vanno quindi gestiti, filtrati, salvati ed elaborati in modo opportuno.

Da questo ne scaturisce una Architettura Logica che può essere suddivisa nei Layer indicati nella Figura seguente:

1. Connector
2. Opportunistic Integrator
3. Analytics / Data Governance
4. Multimedia Manager
5. Semantic and Ontological
6. Event Manager
7. Reflective Architecture

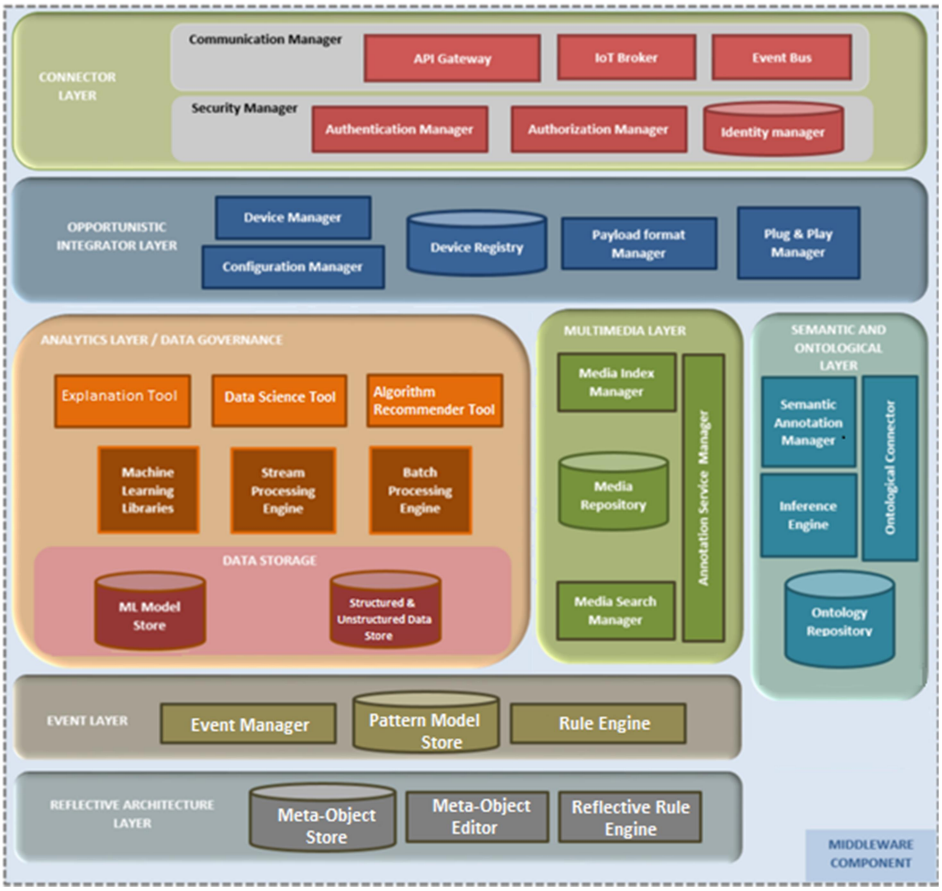


Figura 1: Architettura Logica del Middleware

Di seguito si fornisce una macro descrizione delle singole componenti costituenti il Middleware. Per una descrizione più dettagliata delle stesse componenti si rimanda alla lettura dei capitoli specifici.

La piattaforma Digital Future comunica con il mondo esterno tramite il **Connector Layer**, suddiviso, per questioni strutturali logiche, in due micro layer: *Communication Manager* e *Security Manager*.

Il Communication Manager, oltre alle varie User Interface, esposte dalle singole componenti e non esplicitamente rappresentati nell'architettura, espone delle API che permettono ai sistemi verticali di interagire con la stessa. Tutte le varie API sono gestite nella componente denominata *API Gateway.* Oltre ai sistemi verticali, anche i dispositivi IoT, che solitamente utilizzano protocolli dedicati e diversi dall'HTTP, potranno dialogare con la piattaforma e lo effettueranno attraverso la componente denominata *IoT Broker*. Infine, sia per la comunicazione tra piattaforma e soluzioni verticali, che per la connessione delle componenti interne della piattaforma, sarà necessario predisporre un *Event Bus*. L'Event Bus è un elemento cruciale dell'architettura, funge da collettore di tutti gli altri elementi e rende l'architettura scalabile e fault-tolerant.

All'aspetto legato alla comunicazione si affianca obbligatoriamente l'aspetto legato alla sicurezza, gestita appunto dalla componente denominata Security Manager. E' importante specificare che questa componente si preoccupa di gestire gli utenti e i sistemi che dovranno interagire con la piattaforma e di regolamentare tutte le varie interazioni che ciascuno di essi dovrà avere con le varie risorse interne alla stessa piattaforma, definendo e gestendo le varie autorizzazioni e i relativi livelli.

Dato l'elevato numero di componenti e dei relativi servizi offerti da ognuno, si rende necessario un ulteriore componente, in grado di ridurre le azioni di configurazione ed assicurarsi che ci sia coerenza di configurazione tra le componenti, riducendo così effort e rischio di commettere errori. La componente denominata **Opportunistic Integrator Layer** potrà avviare il processo di auto-censimento ed auto-configurazione, abilitando le connessioni necessarie tra IoT Broker ed Event Bus, definendo i canali dell'Event Bus su

cui confluiranno gli eventi generati dal dispositivo, le strutture dati per la memorizzazione degli stessi, etc..

Esso è un elemento centrale per gli aspetti di configurazione e si compone dei seguenti elementi:

- Plug&Play Manager - dialoga con il device e di conseguenza utilizza gli altri servizi del layer per predisporre e configurare in maniera automatica ogni componente della piattaforma per predisporre i servizi richiesti dal dispositivo
- Payload Format Manager - permette al dispositivo di dialogare con la piattaforma utilizzato il formato dati desiderato
- Device Manager - espone i servizi relativi alla gestione del dispositivo
- Configuration Manager - espone i servizi relativi alla configurazione di tutte le altre componenti della piattaforma così come delle configurazioni dei dispositivi
- Device Registry - è il repository dove sono memorizzate tutte le informazioni relative ai dispositivi

Uno dei layer core e più interessanti del middleware è sicuramente il Data Governance and Analytics Layer, che espone all'utente finale tre macro-funzionalità raggruppate nei rispettivi componenti:

- Data Science Tool
- Explanation Tool
- Algorithm Recommender Tool

Il *Data Science Tool* permette di creare ed eseguire delle pipeline di analisi, dalla raccolta, alla preparazione del dato, all'addestramento di modelli di Machine Learning e di Deep Learning, fino al loro utilizzo su stream di eventi in tempo reale. E' importante che questa componente dia la massima libertà all'esperto di dominio, di eseguire pipeline di analisi complesse, composte da molti step, ma allo stesso tempo deve assicurare che l'analisi sia ripetibile. E' bene notare inoltre che nella pratica è spesso necessario realizzare più varianti di una stessa pipeline, dove variano parametri di configurazione, set di dati, algoritmi utilizzati, alla ricerca della combinazione che porti i migliori risultati. Le pipeline devono ovviamente essere eseguite nativamente tramite

calcolo distribuito, siano esse in batch o in streaming. Tutte le analisi e gli algoritmi , siano essi di tipo online che in batch, dovranno essere eseguibili in ambiente di calcolo distribuito. Per farlo si dovranno avvalere delle componenti di *Stream Processing Engine* e *Batch Processing Engine* che offrono appunto supporto per il data processing su larga scala**.** L'engine di Stream Processing è utile anche ai fini del "cleaning" e "filtering" dei dati in tempo reale: in questo caso la pipeline consiste nel prelevare i dati dall'event bus, modificarli e filtrarli, per poi reintrodurli in un differente canale del bus diretto alla destinazione di storage finale.

Nel momento in cui l'esecuzione di una pipeline porti ad un risultato è necessario che il middleware sia in grado di spiegare all'esperto di dominio "come" vi sia giunto. Il compito è appunto demandato alla componente denominata *Explanation*, che si occuperà appunto di fornire spiegazioni facilmente comprensibili da un essere umano su cosa ha contribuito alla generazione del risultato.

Infine è presente la componente denominata *Algorithm Recommender*. La risposta alla domanda "Quale algoritmo di machine learning dovrei usare?" è sempre "dipende". Dipende dalla dimensione, dalla qualità e dalla natura dei dati. Dipende dal fatto se si ha necessità di avere insight sul modello, quanto tempo e risorse si possono dedicare all'addestramento e da che performance si vogliono ottenere. Ma alla domanda "Quale algoritmo, ad esempio, di classificazione è più utilizzato in uno specifico dominio?", la risposta è un po' più complessa, in quanto si renderebbe necessario uno scouting tecnologico o una valutazione statistica sull'utilizzo di quello specifico algoritmo nel dominio interessato.

Sempre nel layer Data Governance è previsto il *Data Storage Layer*, nel quale sono stati considerati tre componenti: due per trattare altrettante tipologie distinti di dati (strutturati e non strutturati) e una per la memorizzazione dei modelli ML. Le tecnologie utilizzate per le due tipologie sono solitamente distinte. I dati storici presenti in questo layer saranno utili sia ai fini dell'analisi statistica che dell'addestramento di modelli di machine learning, ma anche per la necessità di gestire eventi in tempo reale: il flusso di gestione di un evento può dipendere da informazioni di carattere storico. L'accesso ai

dati avverrà dunque sia da parte delle pipeline di analisi che da parte dell'Event Layer. Il popolamento potrà avvenire facendo confluire una parte di interesse dei dati che transitano sull'Event Bus, ma il verticale potrebbe anche avere necessità di caricare altri dati di suo interesse da integrare nell'analisi, come dati generati dai suoi applicativi, dati presenti in un DBMS relazionale, dati di pubblico dominio.

L'utente che accede alle funzionalità messe a disposizione dal Middleware deve avere dunque la possibilità di definire nuove strutture dati e popolare il data store strutturato o non strutturato da varie fonti attraverso funzioni di import. Essendo il middleware una soluzione di tipo SaaS, il layer di storage dovrà assicurare che solo i soggetti autorizzati possano accedere ai dati. Per quanto riguarda la memorizzazione dei modelli di machine learning, questi necessiteranno di un componente apposito, in grado di associare alcuni metadati ai modelli.

Altro componente presente nel disegno architetturale è **l'Event Layer**. Esso si dovrà occupare di identificare, gestire e processare tutti gli eventi di interesse che intervengono o che sono generati dal sistema e di fatto contemplerà le funzioni di un Complex Event Process (CEP). Il layer attingerà dall'Event Bus quale fonte di eventi, ma anche come destinazione degli eventi complessi generati.

La varietà ed eterogeneità di sistemi, dispositivi e soggetti, che dovranno interagire con una piattaforma che si colloca nell'ambito dell'Internet of Everything, genera la necessità di introdurre infrastrutture architetturali cosiddette *reflective*, in grado cioè di consentire la realizzazione di servizi applicativi adattivi al contesto. È stata prevista una componente identificata con il nome di **Reflective Architecture** che si occuperà appunto di definire un insieme di modelli operativi che rendono parametrico il comportamento di un servizio applicativo al fine di giungere ad ottenere variazioni dinamiche dello stesso sulla base del contesto. Le componenti della piattaforma che generano eventi, come l'Event Layer, la Data Governance o lo stesso Opportunistic Integrator potranno far uso di questa componente a tale scopo.

Il middleware mette a disposizione un patrimonio comune di contenuti multimediali attraverso la componente denominata **Multimedia Layer**, nel quale saranno trattati

video, audio, immagini, elementi testuali. Questi contenuti dovranno essere annotati semanticamente attraverso l'*Annotation Service*, che funge da elemento di congiunzione tra il Multimedia Layer ed il Semantic Layer. In questo modo il *Media Search Manager* potrà basarsi su un matching semantico tra la richiesta elaborata e la descrizione disposta nella fase di memorizzazione del contenuto multimediale nel repository, al fine di fornire servizi avanzati per il discovery dei contenuti.

Infine è presente il **Semantic Layer** che è al servizio del Multimedia Layer per annotare i documenti e contenuti multimediali con tag contestualizzati per consentire una rapida ricerca intelligente degli stessi o di parti specifiche di essi.

Quanto descritto è appunto una overview generale dell'architettura del Middleware. Nel capitolo che segue si entrerà più nello specifico di ciascuna componente con lo scopo di meglio dettagliare le loro caratteristiche.

Come già accennato nel capitolo introduttivo, il modello architetturale si presta ad essere anche scalabile a livello edge dal momento che le componenti essenziali dell'architettura possono essere istanziati a seconda dei vari casi d'uso.

3. Connector Layer

Il Connector Layer è il layer del Middleware che si occupa di operare nell'ambito della comunicazione tra la piattaforma e le soluzioni verticali. Tale layer incapsula al suo interno due sublayer:

- **Communication Manager**: necessario per definire ed individuare i diversi tipi di connessioni possibili
- **Security Manager**: gestisce l'autenticazione e l'autorizzazione degli utenti dei "pillar" che accedono al Middleware.

3.1 Communication Manager

Il **Communication Manager** deve essere in grado di interfacciare la piattaforma con i vari pillar, permettendo a questi l'utilizzo e la gestione dei sensori, l'attivazione di algoritmi di analisi dati, la ricezione di risultati a fronte della esecuzione di tali algoritmi, l'invocazione di servizi di elaborazione specifici.

Nel realizzare tale strato architetturale, si dovrà tenere in considerazione il concetto dell'indipendenza dall'origine dei dati, ciò equivale a dire che si dovrà permettere la connessione di qualsiasi dispositivo alla piattaforma, senza che questi venga configurato, analogamente alla modalità *plug-and-play*.

Il modello dovrà essere progettato in maniera tale da permettere la comunicazione bidirezionale con tutti i dispositivi che accederanno alla piattaforma, contemplando adeguati livelli di sicurezza che saranno gestiti dal *Security Manager* (vedi paragrafo specifico). Come si evince dalla figura sotto riportata

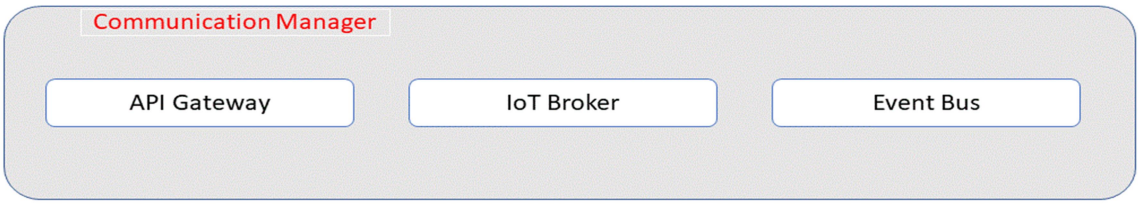


Figura 2: Architettura Logica del sub-layer Communication Manager

il Communication Layer si compone delle seguenti componenti:

- **API Gateway -** Permette l'interazione tra gli utenti delle soluzioni verticali e alcuni servizi applicativi esposti dal l Middleware.
- **Event Bus** – Consente il transito di eventi tra le soluzioni verticali e il Middleware e tra i componenti stessi del Middleware
- **IoT Broker** - Permette la comunicazione tra dispositivi IoT e Middleware.

3.1.1 *IoT Broker*

L'*IoT Broker* è la componente che all'interno del *Communication Manager* si occupa dell'acquisizione ed instradamento dei flussi provenienti dal dispositivo. L'IoT Broker è un *message broker*, cioè una componente software che, fungendo da intermediario tra sistemi con differenti protocolli di comunicazione, si occupa di instradare i messaggi ai corretti destinatari. Invece di inviare messaggi ad un determinato set di destinatari, i mittenti pubblicano i messaggi su un certo argomento (detto *topic*). Ogni destinatario si iscrive agli argomenti che lo interessano e, ogni volta che un nuovo messaggio viene pubblicato su quel determinato argomento, il *message broker* lo distribuisce a tutti i destinatari.

L'adozione di un *message broker*, nella progettazione di architetture, riduce notevolmente le dipendenze tra componenti o sistemi; inoltre permette di nascondere i dettagli implementativi agli utilizzatori, fornendo esclusivamente un'interfaccia per consentire la ricezione e l'invio di messaggi.

Tra i principali vantaggi derivanti dall'utilizzo di un *message broker* è possibile annoverare:

- Semplicità di comunicazione: i consumer e producer comunicano esclusivamente con il message broker.
- Affidabilità nella consegna del messaggio: è il message broker che garantisce il recapito dei messaggi.
- Asincronia: quando il produttore invia il dato non deve preoccuparsi di assicurare che il consumatore sia disponibile in quel determinato momento e viceversa.
- Persistenza: se all'invio dei dati, il consumatore non è connesso, il message broker può conservare i messaggi fino alla consegna.
- Varietà di modelli di comunicazione: message broker evoluti implementano differenti modi di consegnare i messaggi (point-to-point, publish/subscribe)

La scelta di un componente di message broker viene determinata dal protocollo. Tra i più comuni e popolari protocolli di messaggistica basati su TCP / IP, ad oggi disponibili, abbiamo MQTT e AMQP.

Nella valutazione del protocollo, il dominio applicativo gioca un ruolo fondamentale. Dal momento che in ambito Internet of Things non si può prescindere dalla rete mobile e dal risparmio energetico dei dispositivi mobili, nel prossimo paragrafo verrà approfondito il protocollo MQTT, pensato per dispositivi con poca banda e scarsa potenza di calcolo a disposizione, a vantaggio di AMQP.

L'IoT Broker sarà utilizzato non solo come punto di ingresso per i messaggi che proverranno dall'esterno ma anche per quelli che la piattaforma invierà verso i dispositivi registrati, in base alla pertinenza del messaggio. Sarà quindi introdotto un connettore che si occuperà di indirizzare verso i dispositivi interessati i messaggi inviati sul topic dell'Event Manager dalle applicazioni del Middleware che svolgeranno servizi di analisi dati e che, potrebbero avere la necessità di inviare messaggi / alerts ai dispositivi.

3.1.1.1 *Il protocollo MQTT*

MQTT, acronimo per Message Queue Telemetry Transport, è un protocollo over TCP/IP di messaggistica basato sul modello publish/subscribe, cioè basato su nodi che inviano i

messaggi (Publisher) e nodi che ricevono i messaggi (Subscribe). Inventato da IBM e Link nel 1990 è stato successivamente standardizzato dall'organizzazione OASIS arrivando alla versione MQTT 3.1.1.

Questo protocollo è stato principalmente progettato per i dispositivi con scarsa potenza di calcolo in situazioni in cui la larghezza di banda a disposizione è piuttosto limitata o appartengono a reti poco affidabili.

Lo scopo di questo protocollo è disaccoppiare publisher e receiver portando i benefici di:

- *Space decoupling*: publisher e subscriber condividono solamente l'indirizzo IP del broker e non hanno la necessità di conoscersi fra loro;
- *Time decoupling*: la pubblicazione e ricezione dei messaggi non vengono eseguiti nello stesso istante, quindi il broker può memorizzare i messaggi da inoltrare ai receiver in un secondo momento;
- *Syncronization decoupling*: un client già impegnato in altra operazione non viene interrotto per la ricezione di un messaggio, ma potrà scaricare lo stesso dalla coda al termine dell'operazione esistente.

Caratteristiche

MQTT offre una serie di funzionalità che gli hanno permesso di affermarsi come uno standard di riferimento per quanto riguarda l'IoT, tra cui:

- Topic matching
- Last Will and Testament
- Persistenza
- Sicurezza.

Topic Matching

In MQTT i topic sono strutturati gerarchicamente, come un file system. All'atto della pubblicazione di un messaggio il topic deve essere identificabile e privo di ambiguità, mentre in fase di sottoscrizione è possibile utilizzare delle wildcard per poter osservare la gerarchia completa: il carattere speciale "+" rappresenta un livello della gerarchia, mentre il carattere "#" rappresenta l'intero albero a partire da un certo punto della

gerarchia, ad esempio, a/+/d fa match con a/b/d ma non con a/b/c/d, mentre a/#/d farà match anche con a/b/c/d.[5]

Last Will and Testament

MQTT è spesso utilizzato in scenari che prevedono utilizzo di reti instabili, ciò comporta che i client MQTT possano disconnettersi in modo improvviso. La feature "Last Will and Testament" fornisce un modo per permettere ai client di rispondere in caso di disconnessioni improvvise in modo appropriato. Ciascun client MQTT, in fase di connessione, può registrare presso il broker un "last will message". Questo messaggio verrà storicizzato dal broker, fin quando esso rileva che il client si è disconnesso in modo improvvisa. In risposta, il broker invia il "last-will message" a tutti i client sottoscrittori del topic "last-will message". Questi messaggi possono essere, quindi, utilizzati per segnalare ai sottoscrittori di un determinato topic che un dispositivo non è più disponibile.[6]

Persistenza

MQTT pur non essendo un middleware completo orientato ai messaggi, supporta comunque funzionalità atte a garantire gli aspetti relativi alla persistenza. Questa funzionalità diventa essenziale quando si ha a che fare con client che agiscono in ambienti limitati, con client con bassa potenza di elaborazione e con connettività intermittente. È qui che la funzionalità di persistenza del broker MQTT diventa utile, in quanto ogni volta che un client si connette al broker MQTT, esso avvia una nuova sessione per la sottoscrizione o la pubblicazione su topic; se la connessione viene persa, il processo riparte nuovamente con il client che stabilisce una nuova sessione.

Questo processo inficia negativamente sulle prestazioni del sistema, pertanto al momento della connessione, un client impostando il flag 'Clean Session' a "false", informerà il broker che deve conservare le informazioni sulla sessione anche dopo una eventuale disconnessione. Il broker, da quel momento in poi, inizierà a mantenere la sessione per quel client. Ogni sessione conterrà, quindi, dettagli quali gli argomenti

sottoscritti o pubblicati dal client, i messaggi QoS 1 e QoS 2 inviati, mentre il client era offline, e i messaggi QoS 2 che devono ancora essere riconosciuti dal client.

Sicurezza

Il protocollo MQTT consente l'autenticazione da parte dei client tramite username e password. La username deve corrispondere ad una stringa codificata in UTF-8, mentre la password è un binario con capacità massima di 65535 byte. Un client può facoltativamente specificare la sola username.

In risposta al meccanismo di autenticazione implementato, il broker risponde con uno dei seguenti codici:

- 0 - connessione accettata
- 4 - connessione rifiutata, username o password errata
- 5 - connessione rifiutata, non autorizzato

Il protocollo MQTT si basa sul protocollo di trasporto TCP. Di default, le connessioni TCP non utilizzano una comunicazione crittografata. Molti broker, per crittografare la comunicazione, consentono l'uso di TLS invece del semplice TCP. E' fortemente consigliato associare l'utilizzo del TLS al meccanismo di autenticazione e autorizzazione, tale scelta assicura che nessun "interceptor" malevolo possa intercettare le comunicazioni, ottenendo quel livello aggiuntivo di sicurezza che è importante per l'autenticazione e l'autorizzazione a livello applicativo.[7,8]

Struttura dei messaggi MQTT

MQTT, essendo un protocollo message-oriented, supporta l'interazione tra client e broker mediante messaggi. In un messaggio è distinguibile un header di dimensione fissa, un header di dimensione variabile ed il payload.

HEADER FISSO

L'header fisso, come è possibile osservare nella figura sottostante, è lungo 2 byte e contiene un Message Type, un DUP flag, un QoS livel ed un RETAIN.

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

- Message Type

Il Message Type definisce il tipo di comando che si andrà ad inviare con il messaggio secondo la lista mostrata in figura:

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Completed (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

- DUP flag

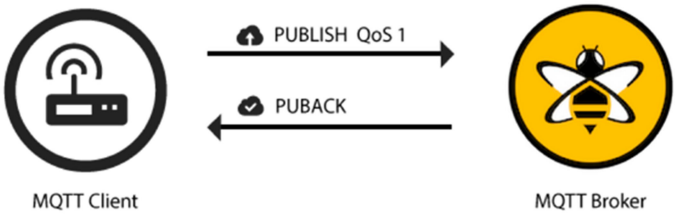
Rappresenta un flag con cui è possibile specificare se il messaggio che si sta inviando è un duplicato di un precedente pacchetto andato perso o non confermato dal relativo ACK.

- QoS level

Specifica il livello di qualità di servizio richiesta per un messaggio di tipo PUBLISH. Esistono tre livelli di QoS:

- **Livello 0** – *At most once*: rappresenta la soluzione più veloce, non c'è nessuna conferma di ricezione del messaggio da parte del broker. Un client quindi non potrà mai sapere se un dato messaggio è giunto o meno a destinazione.

- **Livello 1** – *At least once*: garantisce che il messaggio venga consegnato almeno una volta, ma non esclude che ci siano duplicati. Il broker comunica al client di aver ricevuto il messaggio facendone l'ACK. Nel caso in cui un messaggio non raggiunga il broker o vada perso l'ACK, il messaggio verrà re-inviato con la specifica che il messaggio è duplicato.



- **Livello 2** – *Exactly once*: assicura che i messaggi vengano consegnati una ed una sola volta. Questo livello impone che ci sia un doppio scambio di ACK tra cliente e broker. Affinché si sappia sempre a quale messaggio i differenti ACK fanno riferimento, questi dovranno contenere all'interno dell'header l'identificativo del messaggio a cui si riferiscono.

La scelta del QoS risente fortemente del contesto applicativo. In generale il QoS 2 è fondamentale per applicazioni che richiedono la ricezione univoca di un messaggio e l'eventuale consegna duplicata può causare malfunzionamenti ai client sottoscrittori. E' auspicabile usare il QoS 1 quando è necessario garantire che nessun messaggio venga perso e sono ammesse consegne duplicate. Si può, invece, usare il QoS 0 quando la perdita di messaggi è accettabile e assicurarsi tempi di consegna molto veloci. In genere

per messaggi la cui ricezione puntuale non ha fondamentale importanza o la frequenza di invio viene effettuata a brevi intervalli il QoS 0 è fortemente consigliato. Contesti di applicazione tipici dell'IoT possono tollerare la perdita di un messaggio di tanto in tanto e adottare un QoS 0 [9].

- Retain

Specifica se il messaggio di tipo PUBLISH deve essere mantenuto in memoria persistente e consegnato ai client che si sottoscriveranno al topic in un secondo momento (valore 1), o se la consegna deve avvenire solo a chi è sottoscritto al momento della pubblicazione.

HEADER VARIABILE

L'header variabile, quando presente, conterrà campi in ordine ben definito dalla specifica, quali:

- **Protocol Name**: nome del protocollo (MQIsdp) codificato in UTF e presente in tutti i messaggi di tipo CONNECT;
- **Protocol Version**: versione del protocollo, lungo 8 bit e presente nei messaggi di tipo CONNECT;
- **Flag**: specificano la presenza di particolari parametri di connessione all'interno di messaggi di tipo CONNECT:
    - User Name Flag: specifica se nel payload è presente lo username
    - Password Flag: specifica se nel payload è presente la password
    - Will Retain: se impostato, indica che il broker deve persistere il messaggio Will
    - Will QoS: definisce il QoS del messaggio Will
    - Will Flag: specifica la presenza o meno del messaggio Will
    - Clean Session: se impostato a false il broker conserva le sottoscrizioni dei client e memorizza eventuali messaggi da recapitare a seguito di una disconnessione

- **Keep Alive timer**: specifica intervallo massimo, trascorso il quale il client viene riconosciuto come disconnesso;
- **Connect return code**: codice restituito dall'header in risposta a messaggi di tipo CONNACK;
- **Topic name**: corrisponde all'identificativo del canale su cui deve essere pubblicato il payload.

- *PAYLOAD*

Il payload è l'effettivo contenuto del messaggio, obbligatorio per le seguenti tipologie di messaggi: PUBLISH, CONNECT, SUBSCRIBE, SUBACK.

3.1.1.2 *Aspetti sulla sicurezza nella comunicazione*

Nella valutazione degli aspetti di sicurezza legati alla comunicazione, è necessario contemplare diversi aspetti. La valutazione olistica degli stessi permette di scegliere in maniera opportuna standard e framework di comunicazione che meglio si prestano agli obiettivi preposti. Gli aspetti che quindi verranno presi in considerazione riguarderanno pertanto:

1. le componenti che prendono parte alla comunicazione,
2. il livello di sicurezza da garantire alle componenti in quanto tali (asset),
3. il livello di sicurezza da garantire all'informazione che si intende condividere tra le parti interessate,
4. la grado di sicurezza sui diversi livelli della pila ISO/OSI
5. l'environment e quindi il tipo di rete su cui si intende instaurare una comunicazione; infatti una comunicazione che avviene in ambito cloud avrà sicuramente considerazioni ed accorgimenti diversi a comunicazioni che avvengono on-premises, così come differenze possono essere evidenziate in comunicazioni M2M, D2D, D2M e viceversa.

Contestualizzando quanto appena detto nell'ambito del Message Broker, incaricato della comunicazione con i vari sensori e quindi posizionato nella tipologia di comunicazioni di tipo D2M, nella scelta del protocollo va tenuto conto anche di altri

fattori che risultano essere fondamentali. Pertanto, in ambito IoE, e a maggior ragione in ambito IoT, vanno tenuti in considerazione i limiti in termini di risorse disponibili, oltre ai fattori elencati precedentemente, quali ad esempio:

- Affidabilità
- Larghezza di banda di comunicazione
- Latenza di comunicazione
- Compatibilità con device IoT
- Carico di lavoro in device IoT
- Energia consumata in device IoT
- Scalabilità e Costi legati all'uso della tecnologia/protocollo, infatti l'uso di determinati protocolli implica l'uso di tecnologie apposite, un esempio può essere la scelta di tecnologie ZigBee o LoraWan in ambito WSN (Wireless Sensor Network).

MQTT tra I suoi principali vantaggi offre la possibilità di usare SSL/TLS per gestire aspetti di sicurezza essendo basato su TCP, e permette l'uso di Username e Password per l'autenticazione all'interno di messaggi di tipo CONNECT. Tali aspetti rendono tale protocollo adatto all'uso che si intende fare soddisfacendo i requisiti di sicurezza richiesti. Infine, l'ultima caratteristica è che il payload di MQTT è agnostico quindi è "possibile" cryptarlo, ma quest'ultima feature, non essendo gestita direttamente dallo standard, non permette che venga fatto uso direttamente su smart devices. Per garantire la sicurezza di MQTT ci affidiamo quindi completamente alla famiglia di protocolli crittografici SSL/TLS. Di questo protocollo esistono diverse versioni; per evitare i problemi di vulnerabilità note [7] sarà adoperato tale protocollo TLS nella versione 1.2.

Nel caso venga ampliato il supporto di TLS 1.3 [10] alla maggioranza dei Browser più usati [11], verrà proposta (in fase di industrializzazione) l'adozione di quest'ultimo in quanto come riportato in [12]. L'adozione di tale protocollo implica miglioramenti in termini di velocità oltre che di sicurezza dal punto di vista crittografico. L'adozione dell'ultima versione proposta dall'IETF non esula da problematiche, infatti alcune

organizzazioni potrebbero avere la necessità di sfruttare i punti di debolezza delle vecchie versioni di TLS, come punto cruciale del loro modello di business [13].

3.1.2 *Event Bus*

L'Event Bus dovrà operare secondo il meccanismo publish-subscribe e, trattando Big Data, si rende necessario usare il calcolo distribuito in quanto indispensabile per elaborare una gran mole di dati in tempi brevi, ad esempio per effettuare analisi statistiche, o per eseguire ricerche veloci.

Le tecniche di calcolo distribuito utilizzate dagli strumenti adottati comprendono l'utilizzo di file system distribuito e di opportuni framework, come ad esempio MapReduce. MapReduce è un framework sviluppato da Google in grado di gestire la potenza di calcolo distribuita su una grande quantità di dati, suddividendo gli stessi in modo da essere computati separatamente su dispositivi diversi. Si basa sul concetto di *functional programming* nel quale a differenza del multithreading i dati non sono condivisi durante l'elaborazione, ma vengono passati tra le funzioni come parametri o valori di ritorno. L'iter prevede una iniziale suddivisione del dataset in *chunk* indipendenti che saranno processati nei task Map; l'output dei task map faranno da input ai task Reduce, gli output/input consistono in coppie chiave-valore. Il framework si occuperà di schedulare e monitorare i task ed eventualmente far rieseguire quelli falliti.

In Digital Future abbiamo la necessità di operare su Big Data presenti in maniera persistente così come su stream di dati che arriveranno alla piattaforma dai device associati. La procedura prevede che gli stream vengano discretizzati in maniera tale da poter ottenere la singola parte del flusso discretizzato, applicando su di essa anche algoritmi di Machine Learning sia per scopi di analisi storiche sia per poter intercettare situazioni che necessitano una gestione particolare (ad esempio una anomalia nella supply chain di una fabbrica 4.0).

Relativamente il meccanismo "publish-subscribe", un publisher pubblica un evento verso l'Event Bus dall'altra parte il subscriber (già sottoscritto all'Event bus) riceve tale evento. Tra publisher e subscriber vige l'anonimato in quanto è l'Event Bus che si occupa

di tutte le comunicazioni, in questo modo essi sono indipendenti e sostituibili. In pratica i messaggi, anziché essere inviati verso destinatari puntualmente identificati, vengono instradati in base a "oggetti" o "argomenti" chiamati *topic;* ogni destinatario ("subscriber") interessato all'argomento viene appunto sottoscritto al topic. Non , appena un publisher pubblica un evento su quel topic allora l'Event Bus lo distribuisce ai subscriber.

Un singolo topic può avere N publisher e N subscriber; l'Event Bus mantiene un una lista di publisher/subscriber per ogni topic e inoltre ne gestisce i permessi.

Figura 3: Generico Modello Publisher - Subscriber

Il principale beneficio è quello di permettere il collegamento di più componenti che hanno necessità di produrre, consumare ed elaborare eventi ad altissimo throughput, gestendo anche situazioni in cui per un certo lasso di tempo il subscriber non riesce a processare gli eventi alla velocità del publisher. L'utilizzo di tale meccanismo abilita alla creazione di una architettura distribuita ed orientata al real-time. Attraverso l'Event Bus si potranno collegare e di conseguenza anche importare tutta una serie di strumenti atti alla memorizzazione come database, sia relazionali che non (Key Value store) o FileSystem; ma anche indici di ricerca e sistemi Batch. Questo sarà possibile attraverso l'uso di opportuni connettori che a seconda del compito che svolgeranno saranno di input o di output. In base al tipo di connettore la configurazione può avvenire attraverso tool specifici, chiamate alle API REST oppure dal connettore stesso.

Nel contesto progettuale di Digital Future, i dispositivi (fisici o virtuali) saranno in grado sia di mandare messaggi che eventualmente di riceverli, di conseguenza gli stessi dispositivi si configureranno sia con il ruolo di publisher che con il ruolo di subscriber, permettendo di fatto una comunicazione bidirezionale tra dispositivo e piattaforma. Nel ruolo di *publisher*, il dispositivo dovrà inviare i dati secondo uno schema comunicato in

fase di registrazione, tale schema sarà memorizzato in uno *schema registry* che conterrà gli schema di tutti i dispositivi. Questo compito è affidato al componente "Plug & Play Manager" che a sua volta si rifà al componente "Payload Manager" per la gestione delle grammatiche, entrambi i componenti appartenenti al "Opportunistic Integrator Layer". Nel ruolo di *subscriber*, invece, il dispositivo si occuperà di ricevere eventi generati dalla piattaforma; la struttura di tali messaggi varia a seconda della tipologia del dispositivo e delle sue finalità. Sarà compito dell'esperto di dominio tenere conto del tipo di dispositivo e di conseguenza modellare il tipo di messaggi che potranno essere inviati allo stesso.

Per permettere questa comunicazione bilaterale, si dovranno necessariamente associare due topic ad ogni dispositivo:

- Topic "Events": Usato dal dispositivo per comunicare verso la piattaforma;
- Topic "Operations": Usato dalla piattaforma per comunicare verso il dispositivo.

Dunque gli eventi generati da dispositivi IoT (fisici o virtuali) confluiranno nell'Event Bus e da qui potranno confluire verso un sistema di memorizzazione o, piuttosto, essere processati in tempo reale da un engine o, ancora, potranno a loro volta generare nuovi eventi sulla base delle logiche richieste dalla soluzione verticale e specifiche del contesto a cui i dati si riferiscono.

L'Event Bus si occuperà anche di far comunicare:

- Piattaforma e Soluzione verticale del Pillar tra loro
- Componenti interne del middleware

Per quanto riguarda le comunicazioni tra Piattaforma e Soluzioni verticali relativi ai domini applicativi di riferimento, sarà associato e creato per ogni tenant (utilizzatore) della soluzione verticale il topic "Results". Tramite esso fluiranno, verso la soluzione verticale, gli eventi / risultati derivanti dalla esecuzione, sul Middleware, di algoritmi di analisi su dati relativi al contesto applicativo del tenant di riferimento.

Per quanto riguarda le componenti interne del Middleware, sono previste le seguenti comunicazioni:

- Connector Layer: L'Event Bus è strettamente collegato con l'IoT Broker, in quanto i connettori si sottoscrivono ai topic sopracitati creati dell'IoT broker.
- Opportunistic Integrator: In particolare il Configuration Manager si occupa della corretta configurazioni dei connettori alla registrazione di un nuovo dispositivo.
- Data Governance: I DataBase usati per la persistenza dei dati sono direttamente collegati con l'Event Bus in quanto alimentati con i flussi che arrivano dai connettori di output. Lo Stream Processing Engine usa il Bus come fonte dati su cui applicare gli algoritmi di Machine Learning necessari.
- Event Manager: Strettamente collegato al Bus è l'Event Manager. Quest'ultimo potrà intercettare il flusso dei dati in arrivo per:
    1. elaborarli così come sono;
    2. estrarre da essi eventuali eventi che seguono predefiniti pattern. Tali eventi potranno essere:
        a) filtrati, per produrre un ulteriore streaming di eventi che potrà confluire nell'Event Bus e poter ricadere nuovamente sotto l'attenzione dell'Event Manager, comportandosi a tutti gli effetti come un Complex Event Processor su dati realtime / near realtime;
        b) inviati al motore a regole allocato operativamente nel "Reflective Layer" per essere valutati al fine di:
            - attivare opportuni algoritmi di analisi;
            - generare trigger per le notifiche che usciranno dalla piattaforma per raggiungere i dispositivi e/o altri sistemi.

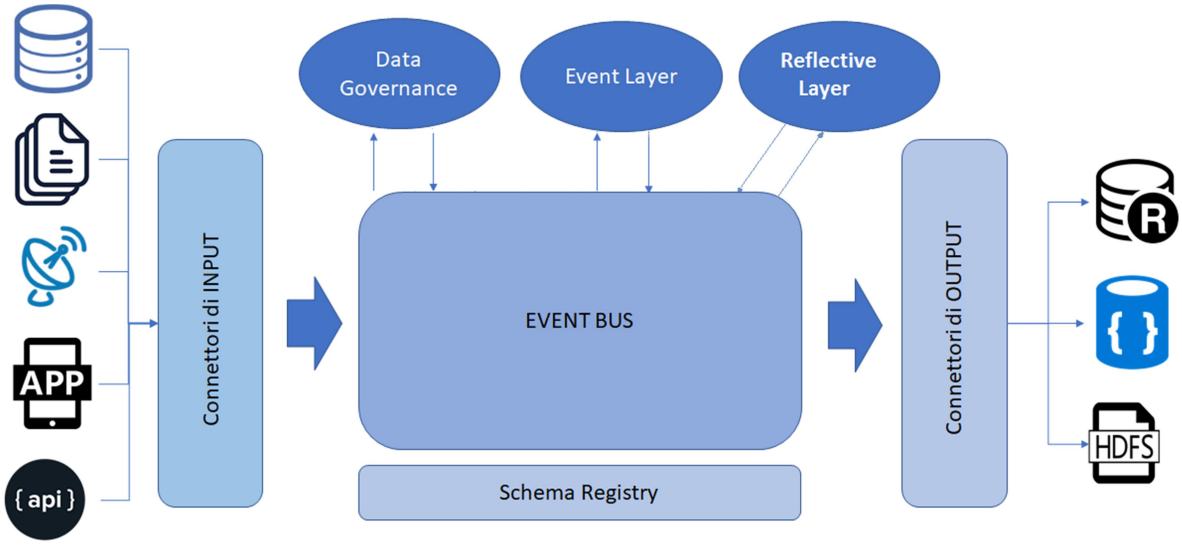


Figura 4: Architettura del componente Event Bus

La figura suddetta descrive l'architettura logica dell'Event Bus. I connettori in input permettono la comunicazione dei dati da molteplici fonti; essendo una piattaforma Internet of Everything, i dati possono pervenire da Data Base di ogni tipo, file, App, API, oltre che dai dispositivi IoT. Il connettore di output permette la comunicazione dei dati su sistemi di persistenza come i Data Base Relazionali e NoSql oltre che su Filesystem distribuiti. Si notano anche la connessione con lo Schema Registry per memorizzare la grammatica di comunicazione delle varie tipologie di dispositivi. Infine è presente la comunicazione con:

- il *Data Governance Layer* e l'*Event Layer* per la elaborazione diretta delle informazioni da parte degli algoritmi di analisi in modalità Batch e Streaming;
- il *Reflective Layer* per la valutazione degli eventi da parte di un Rule Engine al fine di realizzare supportare l'esecuzione di servizi applicativi riflessivi e adattivi al contesto.

3.1.3 *API Gateway*

Api Gateway, in generale, è uno strumento che permette di esporre in maniera strutturata i servizi (o microservizi) dal Middleware verso l'esterno ed agisce da proxy rispetto alle interrogazioni dei client, monitorando e gestendo il traffico delle chiamate.

La componente è presente nel sub-layer Communication Manager e permetterà ad ogni soluzione verticale di poter interagire con il Middleware attraverso l'interrogazione dei servizi messi a disposizione per il proprio dominio, sviluppati in chiave Microservizi. Per definizione le architetture a Microservizi sono l'evoluzione naturale delle architetture orientate ai servizi (SOA) e sono costituite da un insieme di servizi autonomi, dove ogni servizio è indipendente e implementa una singola funzionalità di business. La gestione di un numero elevato di endpoint può essere molto difficoltosa per il Middleware, pertanto, può essere molto comodo disporre di un livello intermedio di riferimento come il gateway. [14]

Se non si disponesse di un API Gateway, gli applicativi dei Pillar dovrebbero inviare le richieste direttamente ai singoli Microservizi del Middleware. Questa condizione aprirebbe le porte ad una serie di diversi problemi. Per permettere una migliore identificazione della situazione, riportiamo alcune considerazioni in tal senso. Un tipo di problematica che si potrebbe avere è legato al potenziale aumento di latenza sulle interrogazioni dei servizi, a seguito dei cosiddetti "round trip".

Altro problema, molto più grave rispetto al precedente, potrebbe incorrere nell'ambito della sicurezza, in quanto tutti i Microservizi verrebbero direttamente esposti verso l'esterno senza attivare sistemi di controllo identificazione/autorizzazione, rendendo più ampia la superficie di attacco rispetto a quando si mascherano i Microservizi interni.

Introducendo un livello intermedio, quindi, si migliora gli aspetti prestazionali e si riesce a garantire un più alto livello di sicurezza. Relativamente alla latenza, si demanda all'API Gateway aspetti legati alla scalabilità, alla gestione dell'aggregazione dei Microservizi e alla gestione delle varie chiamate agli stessi. Relativamente invece agli aspetti legati alla sicurezza, si aumenterebbe il livello demandando alla componente la gestione e soprattutto il controllo delle autorizzazioni all'uso dei servizi e al mascheramento degli stessi attraverso il disaccoppiamento tra il servizio esposto dalla componente interna e quello effettivamente esposto dal Middleware verso l'esterno.

La figura sotto riportata mostra appunto il disaccoppiamento tra il servizio di back-end ed il servizio invocato dai tenant di ogni Pillar. I tenant non chiameranno direttamente i

servizi del dominio ma chiamano l'API Gateway che inoltra la chiamata ai servizi appropriati sul back-end. [15]

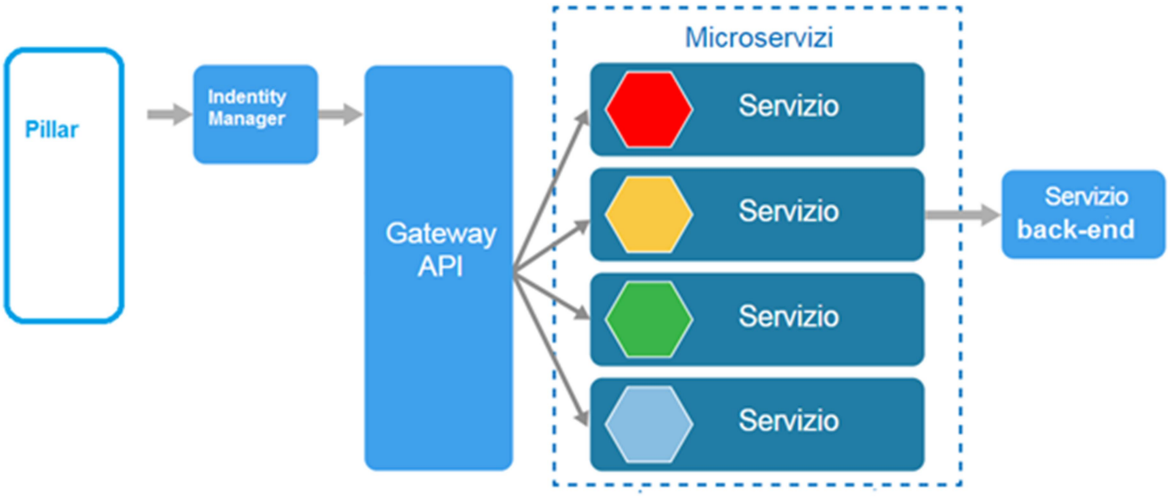


Figura 5: API Gateway Workflow

3.2 Security Manager

Il Security Manager è la componente che si preoccupa di gestire gli utenti e i sistemi che dovranno interagire con la piattaforma, definendo e gestendo le varie autorizzazioni e i relativi livelli. Il problema del riconoscimento dell'identità associata ad una qualsiasi entità che intende interagire con un sistema e dell'attribuzione dei relativi permessi è una difficoltà ricorrente in tutte le situazioni in cui vengono fornite risorse, servizi, dati, ecc. previa autenticazione e autorizzazione. Tale problematica viene gestita usualmente da sistemi di Identity e Access Management (IAM) che gestiscono l'intero processo, inteso come applicazione di policy appropriate e impiego di strumenti tecnologici, volto a gestire le informazioni riguardanti le identità degli utenti e controllarne l'accesso alle risorse informatiche. Al fine di poter soddisfare tali funzionalità, nel Security Layer vengono individuate tre componenti visibili nella Figura seguente: Identity Manager, Authentication Manager ed Authorization Manager. Tali componenti così come riportato in [17] sono gestite dal sistema denominato "Identity and Access Management" che è un sistema a cui vengono demandate le responsabilità di gestione e quindi organizzative associate alla gestione degli accessi degli utenti e dell'autorizzazione verso software applicativi siano essi interni o esterni.

Figura 6: Security Manager

I primi ad aver affrontato i concetti di Identificazione furono l'OECD - Organisation for Economic Co-operation and Development – organizzazione che ha come obiettivo l'integrazione e cooperazione economica finanziaria, e dal NIST[1] – National Institute of Standards and Technology, un'agenzia di governo americana che si occupa di gestione delle tecnologie e delle linee guida riguardanti la protezione delle informazioni personali. A tal proposito l'OECD afferma che la gestione dell'identità digitale è fondamentale per l'ulteriore sviluppo della Internet Economy; solleva inoltre dal problema di come tradurre i meccanismi attraverso cui gli individui si fidano l'un l'altro (trust), prerequisito per l'interazione nel mondo digitale [18].

Sussiste in questo contesto dare una definizione di identità, la quale viene caratterizzata da un insieme di attributi che permettono di individuare univocamente oltre che discriminare le utenze sulla base di determinati valori, al fine di poter definire un'esperienza utente personalizzata e dinamica in base a tali attributi. In sintesi, un'identità consiste in:

- **attributi**: informazioni specifiche di ogni utente distinguibili in attribuiti anagrafici necessari per l'identificazione utente, attributi relativi ai possibili ruoli ricoperti e attributi necessari per l'accesso alle risorse;

[1] Standard NIST SP 800-63-3: Digital Identity Guidelines

- **ruoli**: funzioni che gli utenti ricoprono nel sistema; generalmente un ruolo definisce l'insieme di privilegi assegnati ad un utente.

Aspetto molto rilevante risulta essere l'univocità dell'identità, diviene quindi fondamentale definire un gruppo di attributi in grado di identificare senza ambiguità ogni identità, in modo da potersi riferire a tale identità attraverso tali attributi chiave.

Un'identità nel corso del tempo potrebbe subire delle variazioni a seguito di operazioni come modifica delle credenziali, cambi di ruoli o attributi e così via. Si evince, dunque, la necessità di un Identity LifeCycle Management. La gestione del ciclo di vita delle identità comprende i processi e le tecnologie che consentono l'implementazione, la gestione degli attributi, la sincronizzazione, aggregazione ed eliminazione dell'identità digitale.

I sistemi di Identity and Access Management solitamente sono scomposti nei seguenti livelli [19]:

- Directory services: repository per le informazioni richieste dai servizi di identity management, come informazioni aziendali e regole di utilizzo del servizio;
- Autenticazione: servizio che permette alle varie applicazioni di verificare l'identità utente e passare le informazioni di identificazione alle altre applicazioni;
- Autorizzazione: servizio che determina quali informazioni e funzionalità possono essere accedute e/o utilizzate da un utente autenticato;
- Provisioning: aggiunge utenti al sistema instradando la richiesta all'interno del sistema per l'approvazione, modifica le autorizzazioni utente quando l'utente cambia ruolo, rimuove l'utente dal sistema di autenticazione;
- Password management: fornisce servizi per razionalizzare la gestione delle password utente tramite funzioni di self-service e sincronizzazione;
- Federation e trust: permette di condividere le informazioni utente in modo strutturato tra società diverse o sedi remote.

Quando si ha a che fare con sistemi complessi, ovvero sistemi composti da molteplici sottosistemi, vi è la necessità di dover gestire le identità precedentemente definite, al fine di poter garantire l'accesso della stessa identità alle molteplici componenti che costituiscono l'ecosistema. Tale problematica viene solitamente affrontata in due modi di seguito definiti:

- ogni componente gestisce gli accessi e le autorizzazioni ai propri servizi in modo autonomo,
- centralizzazione degli accessi e delle autorizzazioni.

Nella realtà si possono avere anche casi misti, in cui alcune risorse hanno una gestione degli accessi centralizzata ed esistono risorse che gestiscono gli accessi in proprio; questo caso lo si può chiaramente ricondurre al primo punto in quanto non si ha una vera e propria centralizzazione degli accessi, ma sottoinsiemi di risorse che gestiscono autonomamente l'autenticazione e l'autorizzazione in modo autonomo.

Figura 7: Gestione accessi in modalità autonoma

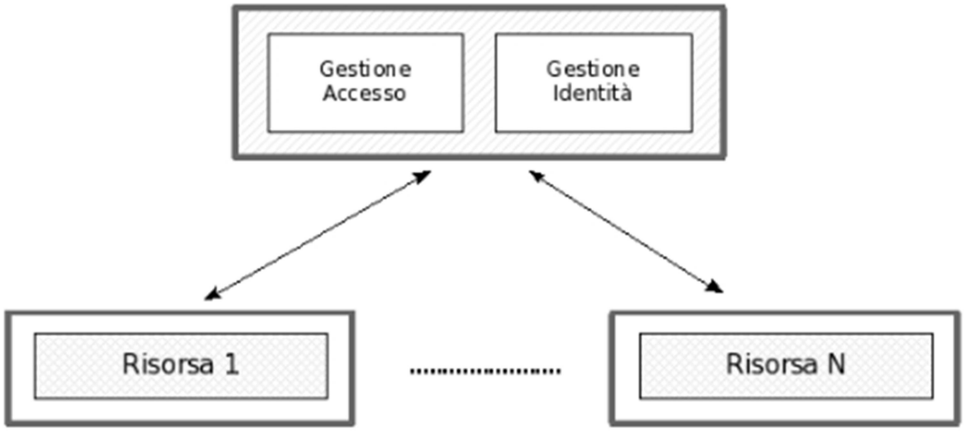


Figura 8: Gestione centralizzata degli accessi

Il primo metodo comporta evidentemente diversi problemi, dal punto di vista del gestore delle risorse:

- le informazioni delle identità sono replicate per ogni risorsa;
- l'aggiunta di una nuova risorsa comporta l'aggiunta di un'intera infrastruttura di accesso;
- la replica delle informazioni su molti sistemi significa dover gestire la sicurezza in molti punti, quindi si ha un minor controllo sulla sicurezza;

e dal punto di vista dell'utente:

- credenziali di accesso diverse a seconda della risorsa a cui vuole accedere;
- minor sicurezza a causa di numerose password senza adeguato livello di complessità.

La gestione centralizzata, tuttavia, non è esente da altre problematiche non trascurabili, quali: performance e affidabilità. Centralizzando in una sola infrastruttura tutta la gestione degli accessi è doveroso stimare il carico di richieste a cui sarà sottoposto, quindi dimensionare opportunamente l'intero sistema in modo che riesca a rispondere a tutte le richieste in tempi sempre accettabili. L'altra problematica da risolvere è l'affidabilità, ovvero fare in modo che l'infrastruttura sia sempre disponibile anche in caso di guasti accidentali o compromissione del sistema.

3.2.1 *Architettura Logica*

3.2.1.1 *Identity Manager*

La componente Identity Manager evidenziata precedentemente, è quella che si occupa di memorizzare fisicamente le identità che tutto il layer ha la necessità di gestire ed i rispettivi ruoli. Senza tale componente quindi non avrebbero alcun senso le componenti di Authentication Manager ed Authorization Manager, che si occupano della verifica formale delle policy definite di accesso ed autorizzazione.

Il nome associato a tale componente non va confuso con quello che in letteratura viene definito "Identity Management" sebbene il nome sia molto similare, il concetto che viene espresso è diverso. Infatti, l'Identity Management è quel processo, parte costituente della "Identity and Access Management", che si occupa della gestione dell'Identità e soprattutto del ciclo di vita dell'Identità, gestendo gli attributi che fanno riferimento ad una determinata utenza, invece l'Access Management è relativo alla valutazione degli attributi basata su politiche e decisioni di tipo booleane.[20]

3.2.1.2 *Authentication Manager*

L'Authentication Manager certifica l'identità dell'utente con cui si sta comunicando. Col termine di autenticazione si definisce il processo tramite il quale una componente software verifica la corretta identità di un utente che vuole comunicare attraverso una connessione.

Il modello generale utilizzato da tutti i protocolli prevede che una entità A inizi la comunicazione inviando un messaggio ad una entità B o una terza entità considerata fidata – solitamente il KDC (Key Distribution Center).

I passi che sottendono al modello con KDC sono quattro, di seguito vengono riportati nel dettaglio:

- A invia un messaggio al KDC, comunicandogli di voler parlare con B.
- Il KDC restituisce un messaggio contenente una chiave condivisa segreta (KA,B)

- Il messaggio è codificato con la chiave segreta che A condivide con il KDC (KA,KDC)
- Il KDC delega A a contattare il ricevente B, inviandogli un ticket (KB,KDC). Il ricevente sarà l'unico che può fare uso del ticket, dato che è l'unico oltre al KDC che sa come decodificare l'informazione che contiene.

3.2.1.3 *Authorization Manager*

L'Authorization Manager fornisce il servizio di autorizzazione, ovvero il passo successivo dopo che un utente è stato autenticato. È il processo che consente l'accesso alle risorse solamente a coloro che hanno i diritti di usarle. Durante l'autorizzazione vengono quindi valutati i privilegi dell'identità digitale associata all'utente autenticato e viene consentito, limitato oppure impedito l'accesso alla risorsa, applicando opportune regole stabilite in precedenza. Il meccanismo permette di garantire l'autorizzazione degli utenti prende il nome di "Controllo degli Accessi". Esso viene definito come le policy o procedure che permettono, vietano o limitano l'accesso ad un sistema. L'utilizzo di tale sistema permette inoltre anche l'osservazione oltre che la memorizzazione di tutti i tentativi di accesso al sistema.

3.2.2 *Modelli di autorizzazione*

I primi modelli di controllo degli accessi erano basati su modelli discrezionali e garantivano un accesso unicamente sull'identità del soggetto o gruppo e dei relativi diritti posseduti sulla risorsa. Il modello che ha avuto larga diffusione, all'interno della famiglia dei modelli discrezionali, noti anche come Discretionary Access Control (DAC), prende il nome di **Identity-Based Access Control** (IBAC) e prevede l'implementazione delle Access Control List (ACL). Tale metodo consiste nella forma più elementare di controllo degli accessi: ogni risorsa controllata ha associata una lista di soggetti con le possibili azioni, che sono autorizzati ad eseguire su tale risorsa.

Questo modello ha il vantaggio di definire un controllo molto fine sulle risorse e non richiede specifiche infrastrutture tecnologiche per essere realizzato. Per questo motivo fu tra i primi metodi ad essere supportato direttamente nei sistemi operativi. La

dettagliata definizione delle regole di accesso è però causa di notevoli svantaggi, come l'alto numero di controlli necessari quando si accede a molte risorse contemporaneamente e la grande quantità di dati da conservare. Il modello IBAC mal si adatta, quindi, a contesti di dimensioni elevate o distribuiti in cui il problema dell'assegnazione di privilegi a singoli soggetti assume proporzioni difficilmente gestibili da uno schema statico.

Un modello che supera parzialmente l'approccio IBAC è basato su identificazione dei ruoli dei soggetti e prende il nome di modello **Role-Based Access Control** (RBAC). Tale modello associa ai soggetti classi di privilegi chiamate ruoli, quindi il controllo sull'accesso passa dalla verifica dei privilegi di un soggetto alla verifica dell'appartenenza di un soggetto a un determinato ruolo. Dal momento che più persone possono avere lo stesso ruolo, e di conseguenza accedere alle stesse risorse, il modello RBAC sopperisce alla carenza principale di scalabilità presente nell'implementazione delle ACL. Questo modello permette la creazione di strutture gerarchiche tra ruoli: i diritti di accesso definiti per ruolo possono essere ereditati da predecessori. La maggior scalabilità del modello, dovuta alla presenza dei ruoli, ha però aumentato la granularità dell'approccio: per assegnare a persone specifiche poche risorse è sempre necessario definire un nuovo ruolo.

Un ulteriore modello che supera il problema di granularità è il modello di controllo degli accessi basato su attributi, il cosiddetto **Attribute-Based Access Control** (ABAC). In tale modello il controllo degli accessi è basato sugli attributi e prevede la valutazione delle regole che determinano l'accesso in base alle proprietà delle entità, alle operazioni richieste e al contesto della richiesta. La richiesta di accesso fornisce al gestore degli accessi, direttamente o indirettamente, una serie di attributi necessari per valutare l'esito finale dell'autorizzazione. Le regole possono essere definite per una qualsiasi combinazione di attributi, offrendo la possibilità di creare regole specifiche per certe risorse o approcci più generali sullo stile del modello RBAC, considerando il ruolo come uno degli attributi.

Una standardizzazione e riorganizzazione del modello ABAC è rappresentata dal modello **Policy-Based Access Control** (PBAC). In tale modello, il sistema di controllo è formato da politiche, cioè insiemi di regole su attributi. La valutazione delle richieste autorizzative avviene combinando le decisioni delle politiche applicabili alla richiesta. Ogni politica calcola la sua decisione in base ad attributi forniti dalla richiesta o da altre entità esterne, che forniscono solitamente valori d'ambiente. Rispetto al modello ABAC, le politiche possono essere identificate e utilizzate in più entità distinte all'interno di un sistema, offrendo una maggior scalabilità del modello.[21]

3.2.3 *Protocolli di Autenticazione ed Autorizzazione*

Il processo di autenticazione può essere implementato in diversi modi. Di seguito verranno passati in rassegna i principali tipi di protocolli.

Una prima famiglia di protocolli fa riferimento al tipo di **protocollo AAA** (Authentication, Authorization, Accounting). Tale protocollo realizza le tre funzioni di autenticazione, autorizzazione e accounting. Tra i protocolli più noti si citano RADIUS e TACACS+.

Il protocollo **RADIUS** (RFC 2865) è stato sviluppato da Livingston Enterprises, Inc., prima versione di tale protocollo risalente al 1997. Le sue specifiche costituivano una proposta per un protocollo standard nel campo dell'autenticazione dell'autorizzazione e dell'accounting. Le comunicazioni tra un server di accesso alla rete (NAS) e un server RADIUS sono basate sul protocollo User Datagram (UDP).[22]

TACACS+ deriva da una serie di miglioramenti ed estensioni apportate da Cisco System al protocollo TACACS e al successore Extended TACACS (XTACACS). Originariamente TACACS nasce come protocollo per l'autenticazione e l'autorizzazione remota, mentre la successiva evoluzione, nota come XTACACS, aggiunge la funzionalità di accounting, rendendo quindi il protocollo compliant allo standard AAA. Il protocollo TACACS+ non utilizza UDP come trasporto, ma si avvale di connessioni TCP ereditandone le caratteristiche di affidabilità. TACACS+ offre più servizi di RADIUS, quali: trasporto orientato alla connessione, non richiede variabili aggiuntive utili per tentativi di ritrasmissione e time_out per compensare il miglior trasporto di pacchetti.

Tra le differenze principali si nota che il RADIUS cifra solo le password nei pacchetti di richiesta di accesso dal client al server, mentre il TACACS+ cifra l'intero corpo del pacchetto ad eccezione dell'header.

Per quanto riguarda l'autenticazione e l'autorizzazione, invece, è importante notare che il RADIUS invia i pacchetti dal server al client con informazione relativa ad entrambi i processi, rendendo difficile una eventuale separazione, mentre Il TACACS+ separa questi processi, permettendo tipi di autenticazione diversi ma compatibili con gli altri due processi: è possibile, ad esempio, utilizzare un sistema di autenticazione differentemente (ad esempio Kerberos, che verrà spiegata più avanti) e l'autorizzazione e l'accounting TACACS+. [23][24]

Un protocollo di autenticazione maggiormente usato in applicazioni client/server è **KERBEROS**. Tale protocollo è pensato per fornire un'autenticazione forte, utilizzando la crittografia a chiave segreta, su reti aperte ed insicure dove la comunicazione tra gli host che ne fanno parte può essere intercettata.

L'adozione di Kerberos permette di raggiungere tre importanti obiettivi:

- ***Autorizzazione***: la capacità del sistema di determinare se un individuo è autorizzato ad eseguire una certa operazione.
- ***Autenticazione***: la capacità del sistema di determinare se un individuo è veramente colui che dichiara di essere.
- ***Cifratura***: la capacità del sistema di prevenire che terze parti ascoltino i contenuti di qualunque comunicazione delle due parti o modifichino i contenuti delle comunicazioni.

Un sistema basato su Kerberos, dove sia il client che il server condividono una chiave comune usata per cifrare e decifrare le comunicazioni di rete, elimina la necessità di inviare le password attraverso la rete, impedendo così agli utenti non autorizzati di intercettare le password inviate attraverso la rete. Affinché questo avvenga, ogni utente ed ogni servizio dovranno essere dotati di una chiave per un algoritmo "generico" di crittografia forte. Kerberos supporta solo algoritmi simmetrici, ciò comporta che venga usata la stessa chiave sia per la cifratura che per la decifratura. Per poter funzionare,

Kerberos ha bisogno di tre componenti, ovvero un Authentication Server (AS), un Ticket Granting Server (TGS) e un Service Server ricevente (SS). Il corretto funzionamento del protocollo prevede che il client si autentica presso AS che gli fornisce un ticket di sessione per accedere a TGS, si autentica presso TGS e riceve il ticket per aprire una sessione di comunicazione con SS.

Si evince quindi che l'AS condivide una password segreta con ogni utente, mentre il TGS ha il compito di emettere dei ticket da usare per convincere i server del fatto che il loro possessore è realmente chi dichiara di essere. È importante comunque sottolineare che ogni server si prende carico delle proprie autorizzazioni: quando il client presenta al server il suo ticket, questo serve solo a dimostrare al server l'identità del client, stabilire che cosa il client sia autorizzato a fare è compito del server. [25]

In riepilogo a quanto descritto sopra, si riporta lo schema del funzionamento:

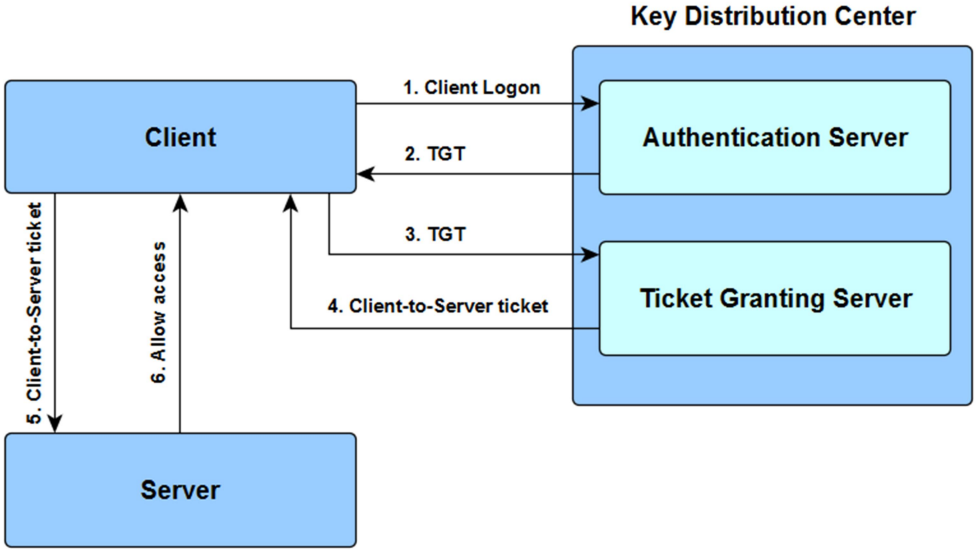


Figura 9: Modalità di funzionamento di KERBEROS

Kerberos non è comunque una soluzione esente da limitazione: essendo un protocollo di autenticazione per host fidati su rete non di fiducia, a nulla servono le strategie di Kerberos se qualcuno che ottiene l'accesso privilegiato ad un server, può copiare ed appropriarsi del file contenente la chiave segreta. All'impostore sarà così sufficiente mettere tale chiave su di un'altra macchina ed ottenere un semplice spoof di DNS o di

indirizzo IP per apparire nei confronti dei client come un server autentico. A tal riguardo è necessario, pertanto, che le macchine in gioco non siano vulnerabili: server di autenticazione, server applicativi e client siano mantenuti costantemente aggiornati affinché si possa garantire l'autenticità dei richiedenti e dei fornitori di servizi. [26]

Un altro protocollo molto utilizzato è l'Open Authorization più comunemente OAuth, è un protocollo di comunicazione open mediante il quale un'applicazione può gestire in modo sicuro l'accesso autorizzato a risorse protette. Il protocollo è compatibile con qualsiasi tipologia di applicazione: desktop, web e mobile. Il protocollo OAuth è stato ideato da Blaine Cook nel 2006, mentre lavorava all'implementazione Twitter di OpenID, proponendosi come alternativa aperta ai molti protocolli proprietari già esistenti, come Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Flickr API e tanti altri. Dalla versione 1.0 (pubblicata nel 2007) OAuth, ha subìto diverse revisioni, fino all'attuale versione 2.0 che riscrive, quasi completamente, il funzionamento del protocollo e quella su cui opera l'Authorization Manager.

OAuth nasce dallo standard SAML(Security Assertion Markup Language) garantendo la possibilità di autorizzare terze parti alla gestione ed uso di applicazioni senza lo scambio di password. L'idea che sta ha alla base di OAuth, è quella di introdurre un terzo ruolo tra i due classici, client e server [27].

In questo protocollo un client per accedere ad una risorsa deve ottenere il permesso dal proprietario delle risorse, autorizzazione che viene espressa sotto forma di token ed uno shared secret. La base del protocollo è il token, che permette di gestire gli accessi senza la condivisione di password e diversamente dalle credenziali dell'utente, può essere rilasciato con delle restrizioni e/o con durata limitata, nonché essere revocato in maniera indipendente dalla volontà del client. Tipicamente, questi token sono emessi dal server dopo una richiesta esplicita dell'utente, in seguito ad un'autenticazione di tipo credential-based da parte di quest'ultimo. La differenza principale presente fra le due versioni di OAuth insiste nell'obbligo, nel caso della seconda versione (OAuth2), di utilizzare una comunicazione sicura SSL.

La figura sottostante mostra il workflow per eseguire l'autenticazione facendo uso del protocollo OAuth 2.0:

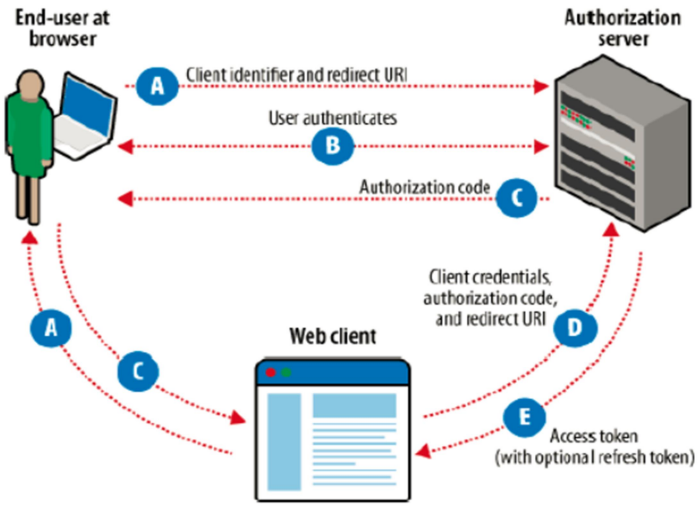


Figura 10: Modalità di funzionamento di OAuth 2.0

OpenID Connect è un livello di identificazione che si pone al di sopra del protocollo OAuth 2.0. L'uso del protocollo OpenID premette ai diversi Client di verificare l'identità degli utenti finali, che intendono accedere alle risorse, sulla base dell'autenticazione eseguita presso i così detti Authorization Server, ma anche ottenere informazioni basilari sul profilo in usando lo stile architetturale REST. OpenID Connect, consente a client di ogni tipo, inclusi client basati su Web, mobile e JavaScript, di richiedere e ricevere informazioni sulle sessioni autenticate e sugli utenti finali. La suite di specifiche è estensibile, consentendo ai partecipanti di utilizzare funzioni opzionali come la crittografia dei dati di identità, l'individuazione di OpenID Providers e la gestione delle sessioni. Secondo la modalità di funzionamento del protocollo OpenID Connect, questi restituisce un "access token" oltre ad un "token id". Quest'ultimo, in formato JWT[28] contiene informazioni sull'utente autenticato, e firmato dal provider del servizio che quindi fornisce le identità. Il vantaggio principale nell'uso di tale formato è associato alla possibilità di lettura dei certificati e dunque delle identità senza dover prevedere l'accesso e verifica delle stesse presso il fornitore dell'identità.

A supportare la scelta nell'uso di OpenID Connect piuttosto che in OAuth 2.0 risiede nella standardizzazione di modalità di erogazione dei servizi associati a endpoint discovery e registrazione dinamica dei clienti. Tale scelta semplifica la programmazione della funzionalità associata al modulo "Security Layer" al fine di supportare al contempo l'affiliazione a molteplici provider di identità.

3.2.4 *Autenticazione e Autorizzazione dei Client*

Nel contesto Digital Future i vari attori che possono interagire con il Middleware sono definiti Client, tali possono essere ricondotti a tre categorie principali, i sensori , gli utenti ed infine le componenti applicative. Quelli che nel documento vengono identificati come "sensori" individuano apparati elettronici di tipo IoT/IoE che sono in grado di gestire ed inviare misure legate a grandezze fisiche rilevate da trasduttori alla componente in esame. Un "utente" si identifica in una persona fisica o giuridica che opera con la componente middleware mediante il portale di accesso al middleware. Ciascun utente in base alle policy che saranno definite nel seguito avrà dei ruoli associati che gli permetteranno di operare con la piattaforma con gli opportuni privilegi. Dal momento che ciascuna componente deve poter interagire con le altre al fine di poter eseguire il workflow, è necessario garantire l'accesso oltre che la comunicazione sicura tra le diverse "componenti applicative".

Volendo dunque contestualizzare il processo di Autenticazione ed Autorizzazione internamente al middleware per le componenti applicative oltre che per gli utenti, sarà introdotta la componente WSO2 Identity Server, che si occuperà dunque vi voler perseguire i compiti ascritti alle funzionalità poc'anzi elencate.

Nel dettaglio avremo che tra i protocolli usati per l'autenticazione degli utenti presso il portale di accesso si farà uso del protocollo OAuth 2.0. Questa scelta non viene estesa come scelta architetturale per l'autenticazione verso altre componenti applicative, come conseguenza di limiti strettamente tecnologici. Dunque il protocollo di autenticazione presso la componente Identity Server per le diverse componenti verrà considerato di volta in volta.

3.2.4.1 *Autenticazione e Autorizzazione dei sensori*

Per quanto riguarda la Piattaforma "Digital Future" uno delle problematiche da prendere in considerazione è il dover permettere l'accesso e l'invio dei dati solo ad "oggetti" web identificati ed autorizzati. Pertanto, al fine di evitare che qualsiasi "oggetto connesso" possa inviare in maniera indiscriminata i dati verso la piattaforma, si dovrà realizzare una una funzione ad uso esclusivo dell'utente Pillar che gli permetterà di far identificare ed autorizzare i propri oggetti. In particolare tale funzione dovrà permettere al Pillar di inserire in input dei parametri che possano identificare il sensore che desidera comunicare con la piattaforma DF; tale funzione ritornerà una Shared-Secret, che di fatto rappresenta una password necessaria al Pillar per autorizzare quel particolare sensore alla comunicazione con la piattaforma. Nel momento della comunicazione, il sensore invierà verso la piattaforma, oltre ai dati sensoristici, sempre l'identificativo e la password affinchè gli stessi dati siano accettati.

La componente architetturale logica che viene interessata è l'IoT Broker, il framework che meglio soddisfa le caratteristiche identificate è individuato in EMQ Broker. Di fatto il broker EMQ permetterà di:

- autenticare un sensore tramite username e password
- autorizzare lo stesso tramite clientID, username, indirizzo IP

Tenendo in considerazione i concetti legati alle policy, la regola di autorizzazione dovrà permettere di definire:

- Il topic per il quale si stanno definendo i criteri di autorizzazione
- Il tipo di accesso (publish, subscribe o entrambi)

Il modulo di "Access Control" di EMQ effettuerà pertanto il matching di ognuna delle regole:

```
          ---------              ---------              ---------
Client -> | Rule1 | --nomatch--> | Rule2 | --nomatch--> | Rule3 | --> Default
          ---------              ---------              ---------
              |                      |                      |
            match                  match                  match
             \|/                    \|/                    \|/
        allow | deny           allow | deny           allow | deny
```

Figura 11: Processo di verifica delle regole di autorizzazione

Dunque in tale contesto si fa uso delle ACL (Access Control List) afferente alla tipologia IBAC (Identity Based Access Control) descritta nel paragrafo "Modelli di autorizzazione". A tale scopo, benchè siano previste altre modalità, è stata prevista l'integrazione del plugin di Autenticazione mediante Username e Password e regole ACL definite internamente al DBMS MySQL. Gli altri plugin previsti con lo stesso scopo, prevedono l'utilizzo di altre tipologie e tecnologie di data storage, oltre che l'uso di soluzioni di tipo LDAP. Il plugin scelto di interconnessione al database MySQL, permette al Middleware di avere a disposizione il database delle utenze e degli ACL, sulla base delle logiche di autorizzazione previste appunto dalla piattaforma integrata.

3.2.4.2 *Security Pattern*

Come riportato nel paragrafo "Sicurezza nei Sistemi Complessi" afferente al deliverable T2.3 (Metodi e tecniche per la gestione della Piattaforma) si è pensato di applicare il modello definito in [16], e nello specifico ci troviamo ad attualizzare il PSM (Platform Specific Model) adattato alla Piattaforma al fine di proteggerla da problematiche afferenti alla sicurezza. Il risultato di tale lavoro di contestualizzazione ed attuazione a livello tecnologico è mostrato nella figura seguente.

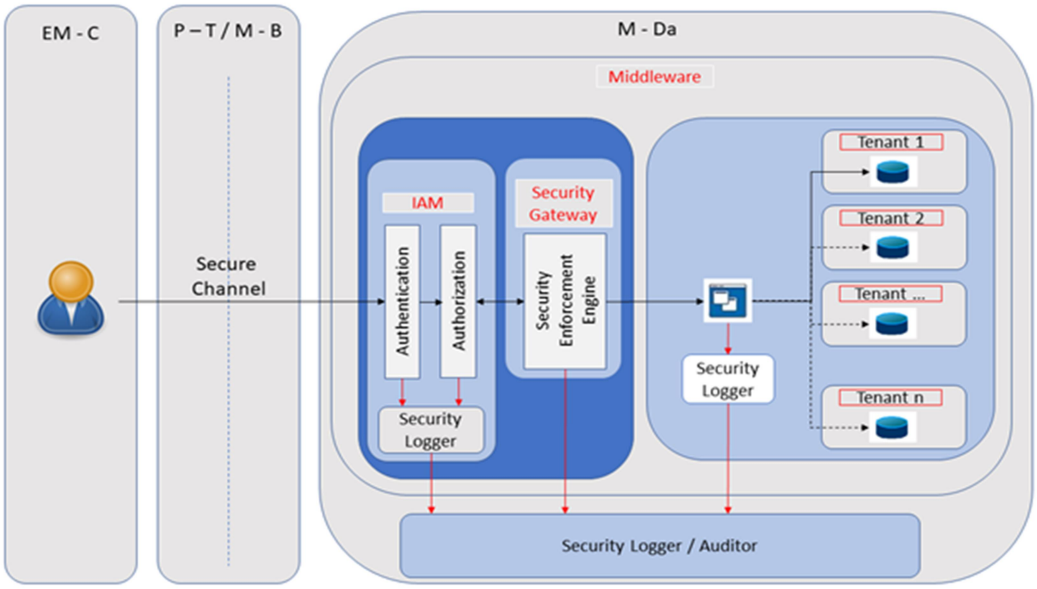


Figura 12: PSM Security Model

Come è possibile osservare si è contemplata la scelta di inserire un'unica componente "Identity and Access Management", anche denominata precedentemente Security Engine, che potesse gestire i Grant di accesso a tutte le funzionalità rese disponibili dalla Piattaforma integrata.

All'integrazione della componente denominata "IAM", segue l'applicazione del pattern definito di seguito. Il "Centralized Access Management" il quale prevede un'interfaccia grafica univoca che permetta di poter gestire graficamente in maniera semplificata le policy di accesso ai sistemi ed ai dati ivi contenuti.

Analizzando il workflow proposto in precedenza si evidenzia che ad (I) una richiesta proveniente da un determinato "Customer" che viene spedita in maniera sicura mediante il protocollo di trasporto TLS, (II) la richiesta giunge al sistema di Identity and Access management che ne verifica l'autorizzazione e l'autorizzazione ad eseguire determinate operazioni. (III) Dal momento che il sistema deve prevedere l'interoperabilità dei dati associati ai diversi tenant e pillar, è stato introdotto il modulo di "Security Enforcement Engine". Tale modulo terrà presente delle policy di accesso ai servizi e di conseguenza ai dati messi a disposizione dai diversi stakeholder garantendo i requisiti laddove imposto di confidenzialità delle stesse. (IV) Dunque internamente al Middleware le varie componenti in esso previste a livello applicativo, interagiranno con i dati in rispetto dei privilegi di accesso assicurati dal "Security Enforcement Engine". (V) Come si evidenzia ciascuna componente riportata integra il modulo "Security Logger" associato alla stessa, oltre all'aver previsto il modulo "Security Logger / Auditor" che funge da accentratore, ai quali sarà dunque possibile agganciare strumenti di Log Management, permettendo dunque analisi sugli stessi.

Contestualizzando il modello PSM a livello di PDM (Product Dependent Model) e quindi scendendo nel dettaglio delle tecnologie che meglio si prestano a sopperire ai moduli contemplati nel Platform Specific Model, si è giunti al modello descritto nella figura che segue.

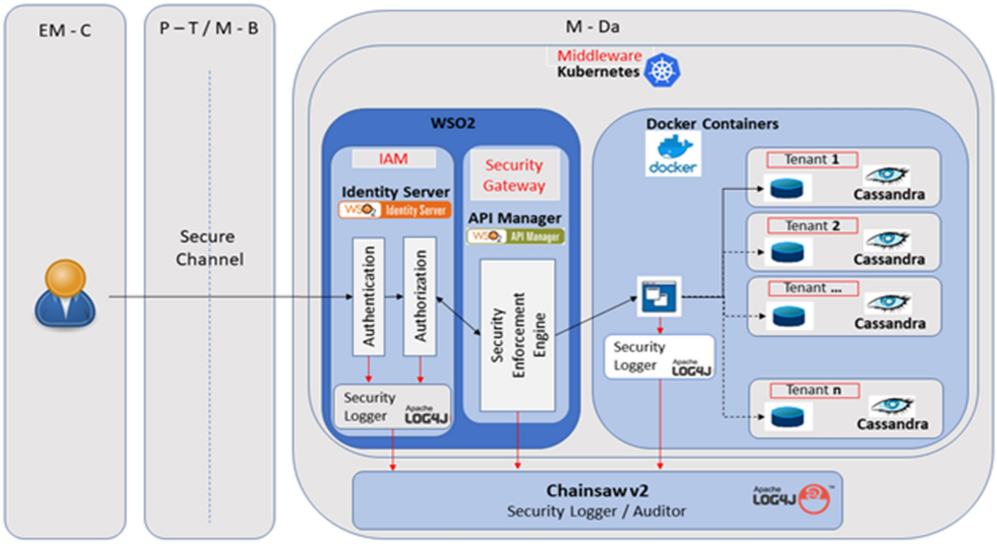


Figura 13: PDM Security Model

Nel modello PDM appena illustrato, andiamo ad evidenziare le componenti tecnologiche previste. I punti (II) e (III) descritti precedentemente sono integrati internamente al framework WSO2, e nello specifico dalle componenti "Identity Server" ed "API Manager". Il resto della logica applicativa e di storage sono identificate in immagini docker contenenti le diversi componenti contemplate. Tali componenti sono state integrate internamente alla tecnologia Kubernetes per sopperire ai problemi di scalabilità, e quindi anche di affidabilità e resilienza dei servizi offerti. logica applicativa e di storage che devono essere offerti dal Middleware.

4. Opportunistic Integrator Layer

L'Integratore Opportunistico ha il compito di gestire il censimento e la configurazione automatica dei dispositvi (virtuali o fisici) che potranno collegarsi alla piattaforma per inviare i dati rilevati.

In altre parole, nel momento in cui un dispositivo cercherà di comunicare per la prima volta con la Piattaforma, essa dovrà riconoscere se il device sia presente nel proprio registro e, qualora non individuato, attivare una configurazione in modalità plug&play. Dovrà pertanto prevedere tutto il processo di gestione della comunicazione e approntare la memorizzazione dei dati inviati dal dispositivo, il tutto quanto più possibile in automatico.

Nel seguente paragrafo viene invece descritto il processo di "Inizializzazione dispositivo", presentato attraverso la notazione BPMN 2.0 (Figura 1) , nel quale si considerano le diverse entità coinvolte:

- Utente del pillar che utilizza la Piattaforma
- Dispositivo / Sensore
- Moduli della Piattaforma

L'entità "User" del pillar, accedendo alla Piattaforma, con la attività "Census Request" richiede nuove credenziali da attribuire al dispositivo, inviando uno username ed ottenendo come risposta una passphrase; tale richiesta di attribuzione di credenziali viene gestita dal service task "Census Sensor Service", facente parte della Lane "Communication Manager" il quale delegherà il service task "Store Credential" afferente alla Lane "Security Manager" per rendere persistenti le credenziali create.

L'entità "User" del pillar a questo punto procederà con l'impostazione manuale dei parametri di connessione sul dispositivo all'interno della attività "Setting Communication Parameters".

A seguire il service task "Sensor Configuration Wizard Service", interno alla Lane "Sensor", procederà con l'invio di un messaggio verso la Piattaforma, contenente la username, la password generata precedentemente e le informazioni necessarie per avviare il processo di autoconfigurazione.

Il service task "Message Broker Service" presente nella Lane "Communication Manager" gestirà il messaggio ricevuto dal dispositivo che sta cercando di collegarsi alla Piattaforma attraverso il task "Authentication Sensor Check", affinché siano validate le credenziali inviate.

Validate le credenziali inviate dal dispositivo, sarà eseguito il processo denominato "Configuration Sensor" afferente alla Lane "Opportunistic Integrator". A seguire si attiverà il sotto-processo denominato "Plug&Play", al quale verranno demandate le attività di impostazione della piattaforma per la definizione del flusso di dati tra le varie componenti. Questo sotto-processo provvederà a :

1. Creare, all'interno dell'Event Bus, una struttura dati per ospitare i dati (messaggi) inviati dal dispositivo che sono in attesa di essere consumati.
2. Creare, all'interno dell'IoT Broker, un canale di comunicazione dedicato al dispositivo, in cui far fluire i dati e collegarlo con la struttura creata al punto 2);
3. Creare, nello Structured Data Store del Data Layer, lo spazio dati necessario per dare persistenza ai dati inviati dal dispositivo;
4. Persistere i dati provenienti dal dispositivo nello spazio creato al punto 3).

Terminata la configurazione, si ritorna al processo di "Configuration Sensor" per la definizione delle "ACL" per i canali di comunicazione, sui quali dovranno essere configurati i diritti di scrittura e/o lettura per lo specifico dispositivo.

La figura seguente mostra il workflow operativo:

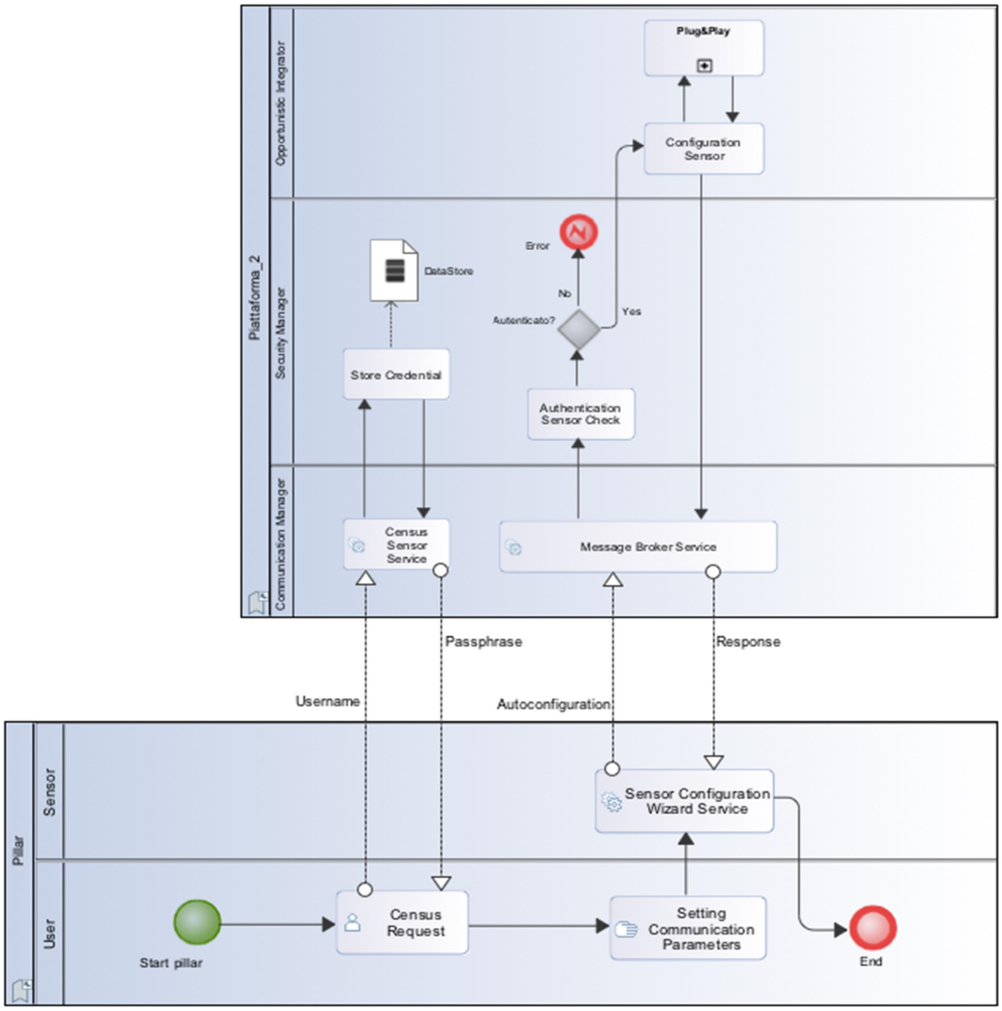


Figura 14: BPMN del Processo Inizializzazione Dispositivo

4.1 Architettura Logica del Layer

Dal punto di vista della architettura logica, il layer si compone dei componenti indicati nella figura seguente.

Figura 15: Architettura Logica del Opportunistic Integrator Layer

Il compito principale del "**Device Manager**" sarà gestire il censimento dei dispositivi che si collegheranno alla piattaforma. Esso sarà coinvolto nella fase di censimento della tipologia di dispositivi che richiedono di comunicare con la piattaforma e nella successiva fase di gestione di tale tipologia e delle relative istanze di dispositivi..

Nello specifico, il "Device Manager" avrà il compito, accedendo al "Device Registry" di:

- Creare l'identità del dispositivo
- Aggiornare i parametri del dispositivo
- Recuperare l'identità del dispositivo tramite ID
- Eliminare l'identità del dispositivo
- Esportare tutte le identità presenti

Il "**Configuration Manager**" è la componente che sarà coinvolta nella fase di configurazione di un nuovo dispositivo che intende avviare una comunicazione con la piattaforma. In particolare dovrà gestire la:

- Creazione e configurazione delle strutture dati dell'*Event Bus* e i canali di comunicazione e dell'*IoT Broker*, collegandoli tra loro; attraverso questi canali transiteranno i messaggi contenenti i dati inviati dal dispositivo.
- Creazione, nello *Structured Data Store*, della tabella in cui saranno persistiti i dati inviati dal dispositivo che sta collegandosi alla piattaforma.

È fondamentale che le strutture dell'Event Bus siano configurate in modo tale da impedire blocchi nella gestione dei dati qualora fosse inviato un messaggio non conforme. Un messaggio è "non conforme" quando nella struttura dati conosciuta al "Payload Manager", per quel tipo di dispositivo, ci sono campi non dichiarati oppure non ci sono campi attesi.

Il "Configuration Manager" inoltre, dovrà gestire le modalità di accesso autenticate, attraverso username e password e attraverso certificati SSL, all'*IoT Broker* ed allo *Structured Data Store*, ovvero username, password e certificati SSL.

Il "**Device Registry**" è lo store contenente le informazioni necessarie a caratterizzare e a rendere operativi i dispositivi che si collegheranno alla piattaforma.

Esso sarà alimentato, in momenti diversi, dalla componente "Device Manager" e dalla componente "Plug&Play Manager".

Infatti, prima che un dispositivo possa connettersi alla piattaforma sarà necessario che sia presente una voce relativa ad esso nel "Device Registry". Grazie a tale voce il dispositivo potrà autenticarsi con le credenziali archiviate in tale registro e solo dopo che le credenziali saranno state validate potrà inviare le proprie rilevazioni alla piattaforma.

Solo dopo che il "Device Manager" ha aggiunto una voce nel "Device Registry" per uno specifico dispositivo, gli altri componenti del Layer potranno creare un set di risorse per tale dispositivo non appena esso cerca di comunicare con la Piattaforma. La creazione di tali risorse sarà a cura del "Plug&Play Manager".

Le informazioni che popoleranno il Device Registry sono quelle che caratterizzano i dispositivi, ossia:

- L'identità del dispositivo
- Le credenziali di accesso alla piattaforma (salvate in modalità criptata)
- Le ACL (Access Control List) del dispositivo
- La struttura dati inviata dal dispositivo

Il "**Payload Manager**" sarà la componente che si occuperà di definire e gestire la "grammatica" delle informazioni che i dispositivi, censiti all'interno del "Device Registry" dal Device Manager, saranno in grado di interpretare allo scopo di consentire la corretta comunicazione con la Piattaforma.

Il "Plug&Play Manager" chiamerà il "Payload Manager" ogni volta che un nuovo sensore si collegherà alla piattaforma. Il "Payload Manager", una volta che il dispositivo è stato censito nel "Device Registry", predisporrà una voce con le regole della grammatica per ciascuno dei dispositivi che si connetteranno alla piattaforma, registrando lo schema dei dati sul registro degli schemi . La grammatica delle informazioni dovrà essere la più esaustiva possibile affinché i dati inviati dai dispositivi siano il più possibile auto esplicativi. L'ottimale sarebbe che la grammatica consenta di produrre un payload di messaggio tale da indicare:

- informazioni specifiche del dispositivo che sta inviando i dati
- i dati inviati
- informazioni sul contesto operativo in cui il dispositivo è inserito
- il pillar di appartenenza
- l'eventuale tenant del pillar che gestisce quel contesto operativo

I messaggi tra dispositivi e Piattaforma, coerenti con la grammatica prevista dal Payload Manager, saranno scambiati attraverso il protocollo MQTT; i messaggi, invece, che non rispetteranno le specifiche dettate dalla grammatica saranno rigettati. Come già specificato nel paragrafo "Struttura dei messaggi MQTT", un messaggio MQTT è costituito da un header di dimensione fissa, un header di dimensione variabile ed un payload. Il payload è di fatto l'effettivo contenuto del messaggio ed è obbligatorio per i messaggi di tipo PUBLISH, CONNECT, SUBSCRIBE e SUBACK. Per permettere che il messaggio inviato dal dispositivo (fisico o virtuale) sia accettato dalla piattaforma è opportuno che la grammatica relativa al payload del messaggio MQTT sia gestita dal Payload Manager.

Il "**Plug & Play Manager**" sarà responsabile del rilevamento dei dispositivi che si collegano alla piattaforma e della loro configurazione automatica all'interno di essa.

Questa componente avrà il compito di:

1. ricevere il messaggio di registrazione dal nuovo sensore;
2. censire il sensore all'interno del "Device Registry" contemplando tutti i vari parametri e ACL;
3. inserire la grammatica di comunicazione all'interno del Payload Manager in modo da registrare lo schema dei dati sul registro degli schemi;
4. attivare il "Configuration Manager" affinché possa essere eseguito tutto il processo che prevede la configurazione dell'*Event Bus* e del *IoT Broker*, necessaria per attivare la corretta canalizzazione del messaggio proveniente dal dispositivo e la creazione dell'area all'interno dello *Structured Data Store* per la persistenza dei dati ricevuti.

La figura seguente mostra l'interazione tra i componenti logici del Layer a fronte delle attività eseguite da:

- dispositivi nelle due fasi di richiesta comunicazione (RC) e trasmissione dati (TD) con la Piattaforma;
- utenti della piattaforma che gestisce i dispositivi (GD)

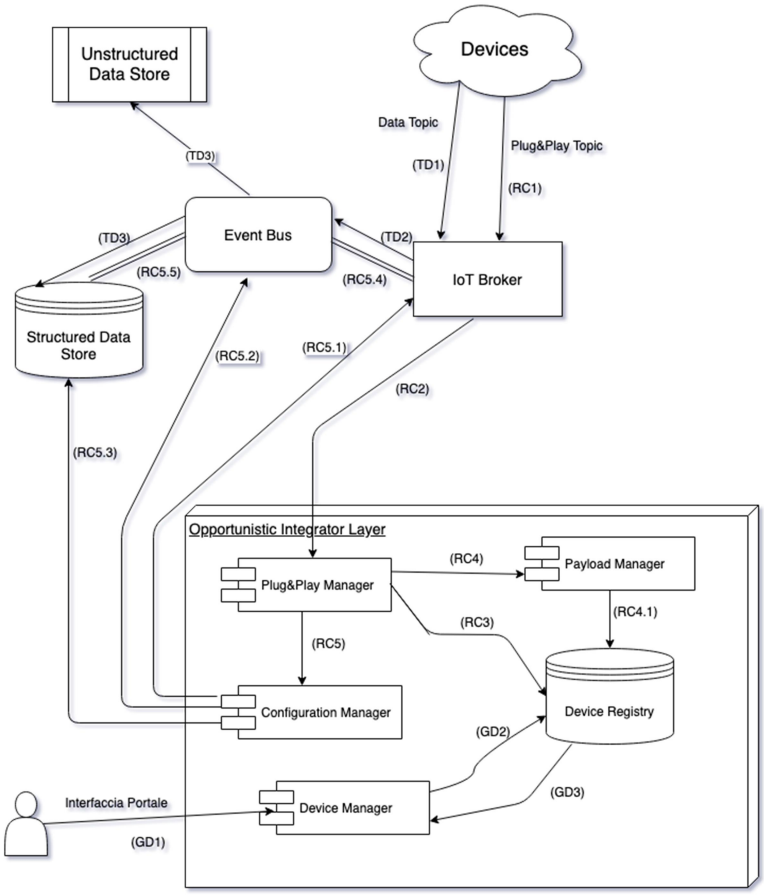


Figura 16: Architettura Logica del Opportunistic Integrator Layer

Viene mostrato anche l'interazione che i componenti del Layer hanno con altri componenti della Piattaforma al fine di attivare e creare i corretti collegamenti tra di essi al fine della persistenza ed elaborazione delle informazioni provenienti dai dispositivi.

5. Analytics Layer / Data Governance

In un contesto in cui aumenta sempre più la necessità di gestire mole di dati eterogenei provenienti da diverse fonti esterne, sensori, macchinari industriali e di gestire varie tecniche di analisi dei dati piuttosto che tecniche di apprendimento automatico, è necessario dotarsi di ambienti infrastrutturali con grandi capacità di elaborazione e di memorizzazione dei dati.

Attualmente sono molte le organizzazioni di diversi domini applicativi che per l'esecuzione di analisi di vario genere (es. statistiche, predittive, …) spendono molto tempo in maniera anche interattiva nella fase di raccolta, filtraggio e analisi utilizzando strumenti reperiti in rete per ogni diversa distribuzione al fine di individuare la soluzione migliore che soddisfa le specifiche esigenze del caso applicativo.

La Piattaforma DF, d'altro canto cerca di venire incontro alle esigenze delle organizzazioni, fornendo un insieme di metodi e strumenti standard, allo stato dell'arte, per garantire una gestione e integrazione dei dati ottimale al fine di consentire una analisi degli stessi in linea con gli obiettivi delle aziende.

Attraverso il Layer Data Governance del Middleware di DF si intende fornire agli esperti di dominio dei diversi Pillar la possibilità di operare in maniera veloce e distribuita. In particolare si desidera offrire da una parte soluzioni per la gestione di mole di dati piuttosto elevata, strutturati e non strutturati, provenienti dai sensori, log o altri tipi di fonte dati, mentre dall'altra soluzioni per la gestione di tutte quelle attività che si occupano di analizzare gli stessi dati attraverso algoritmi appositamente addestrati in modalità real time o con batch schedulati, con il fine ad esempio di estrarre informazioni utili per individuare eventuali legami esistenti tra fenomeni diversi o piuttosto di prevedere eventi futuri.

Questo layer rappresenta, quindi, un insieme di strumenti per i data scientist, esperti di dominio ed esperti informatici finalizzati alla visualizzazione, organizzazione ed analisi di enormi quantità di dati relativi alla problematica da analizzare con il fine di comprendere i pattern più significativi per la tipologia di indagine che si vuole effettuare. L'analisi di tali dati potrà consentire, laddove richiesto, la progettazione e validazione di modelli di classificazione, regressione e clustering che verranno successivamente implementati e istanziati nei componenti del layer preposti per tale scopo.

Il processo metodologico che sarà applicato dai componenti del layer si può considerare suddiviso in due macro-fasi:

a) **Design**, in cui i data scientist ed esperti informatici collaborano per operare all'estrazione dei dati da diverse fonti e al caricamento di un set di dati in un database, all'analisi e visualizzazione di dati storici e alla creazione di uno o più modelli dati.
b) **Deployment,** in cui si prevede che modelli definiti, progettati, testati e validati nella fase di Design siano deployati nei componenti del layer preposti e siano applicati ai nuovi dati in arrivo dai sensori e/o in generale dai sistemi di campo, per effettuare classificazione o segmentazione dinamica dei dai in modalità streaming o batch.

In dettaglio, nella fase di **Design** gli strumenti messi a disposizione dal layer consentiranno agli esperti di dominio:

- L'importazione dei dati da diverse fonti, come database, fogli di calcolo, file in formato CSV.
- La pulizia dei dati rimuovendo gli outlier e combinando le fonti dati. In questa fase si potranno identificare i picchi dei dati, i dati mancanti o i punti anomali da rimuovere. In seguito, si potranno unire insieme le diverse fonti di dati creando una singola tabella contenente tutte le informazioni utili al modello.
- La trasformazione dei dati al fine di generare il modello analitico dei dati, utilizzando tecniche di accorpamento, data reduction, data discretization, etc.

- L'analisi dei dati con tecniche di analisi grafiche come Box Plots, calcolo delle occorrenze etc.
- L'analisi statistica, al fine di permette di convalidare le ipotesi e testarle con l'utilizzo di algoritmi statistici standard.
- Lo sviluppo di un modello predittivo accurato basato sui dati aggregati utilizzando statistiche e tecniche di machine learning e deep learning.
- La validazione del modello al fine di testare la sua accuratezza.

Nella fase di **Deployment**, il layer consentirà di mettere in esecuzione il modello di apprendimento validato su nuovi dati in arrivo dagli scenari operativi gestiti dai pillar e di monitorarne le prestazioni. Saranno forniti anche gli strumenti per avviare una attività di riqualificazione del modello stesso, iterando su alcune attività della fase di Design.

La figura seguente descrive il processo metodologico suddetto:

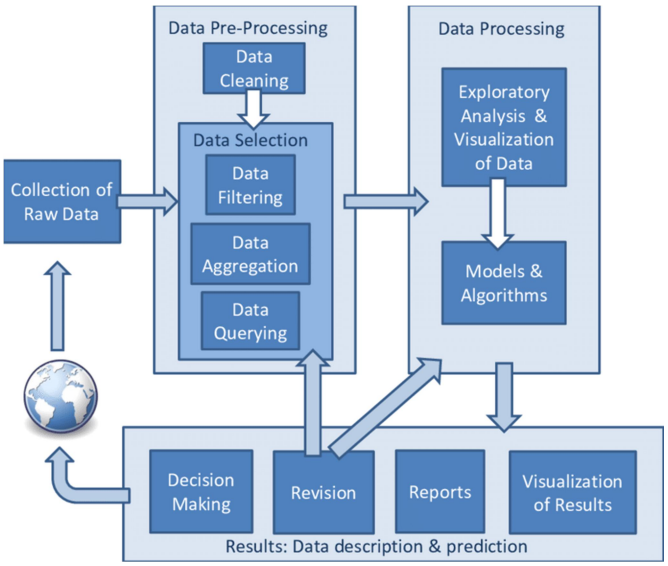


Figura 17: Attività previste nel Data Science Life Cycle

5.1 Architettura Logica del Layer

In generale, il layer Data Governance offrirà le seguenti funzionalità:

- ✓ Gestione, raccolta, elaborazione in tempo reale dati provenienti dai sensori.
- ✓ Gestione, raccolta ed elaborazione, con processi batch, dei dati caricati dagli utenti dei Pillar.
- ✓ Messa a punto di modelli di apprendimento automatico.
- ✓ Messa in esercizio dei modelli addestrati su nuovi dati al fine supportare la logica di business delle soluzioni verticali dei pillar attraverso l'individuazione / predizione di fenomeni significativi per il dominio di riferimento

Nella figura che segue viene rappresentata logicamente la componente Data Governance con tutte le sotto-componenti di cui è costituita. Nei paragrafi che seguono si offrirà inoltre la spiegazione di ognuna di essa.

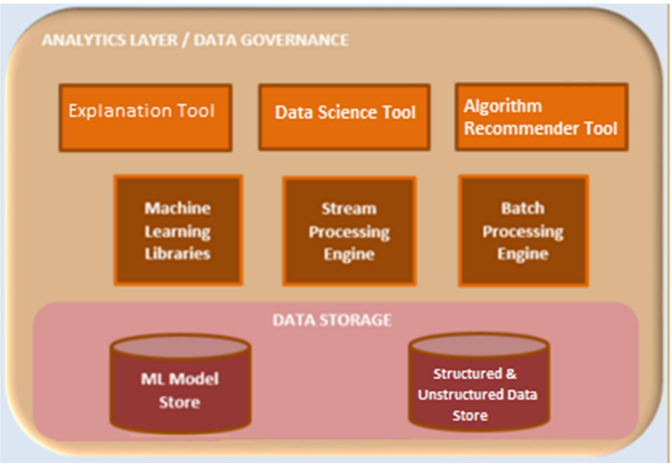


Figura 18: Architettura Logica del Data Governance Layer

5.2 Data Science Tool

Questa componente permetterà di creare e di eseguire delle pipeline di analisi dei dati, dalla raccolta, alla preparazione del dato, all'addestramento di modelli di Machine Learning e di Deep Learning, fino all'interfacciamento per la esecuzione degli stessi modelli su nuovi dati che entrano nella piattaforma.

In pratica questa componente dovrà consenta, all'esperto di dominio, di:

- ✓ definire nuove strutture dati e popolare il data store strutturato o non strutturato con dati rivenienti da varie fonti;
- ✓ creare pipeline di elaborazione ed analisi dati anche complesse, composte da vari step;
- ✓ utilizzare funzioni dedicate per l'import/export di dataset.

mettendo a disposizione:

- ✓ il supporto sull'utilizzo degli algoritmi di analisi;
- ✓ funzioni per la gestione, la valutazione e l'ottimizzazione delle pipeline di dati.

Nella progettazione della componente si dovrà anche tener conto che utenti con competenze più o meno avanzate potranno accedere alle funzionalità messe a disposizione. Di conseguenza, per utenti esperti, la componente includerà caratteristiche tipiche di un notebook per attuare forme di data-driven analytics interattive e collaborative con varie piattaforme di analisi dati, consentendo di scrivere codice in tempo reale, visualizzazioni e presentazioni.

Per utenti meno esperti il componente dovrà fornire un ambiente interattivo guidato / assistito per il caricamento dei dati e per i passi necessari per l'addestramento di un modello, rendendo il processo di training e tuning facilmente realizzabile anche da chi non ha conoscenze di programmazione.

IL componente "Data Science Tool" dovrà operare in accordo al paradigma operativo definito dallo standard CRISP-DM (*CRoss Industry Standard Process for Data Mining*), che definisce un framework teorico definito per condurre progetti di Data Mining.

Figura 19: Framework CRISP-DM

Il framework CRISP prevede di strutturare l'approccio di sviluppo di un progetto di Data Science secondo le seguenti fasi:

- ✓ **Business understanding**: analisi degli obiettivi di business e delle necessità del cliente per identificarne i processi e definire l'obiettivo finale del lavoro.
- ✓ **Data understanding**: fase di raccolta dati dalle sorgenti disponibili e loro analisi al fine di verificare la qualità delle informazioni fornite in funzione della loro accuratezza e completezza.
- ✓ **Data preparation**: fase di vitale importanza per l'ottenimento dei risultati pianificati; i dati necessari al raggiungimento dell'obiettivo sono ripuliti e ricondotti al formato più idoneo al proseguimento del lavoro. Può essere necessaria un'analisi aggiuntiva delle informazioni ottenute per evidenziare pattern emersi alla luce di quanto definito nella fase di Business Understanding.
- ✓ **Modeling**: selezione delle tecniche di modellazione del problema. Questo è il primo passo da fare, seguito dalla generazione di scenari di test per la convalida della qualità del modello. Sono generati diversi modelli che sono analizzati per identificare quello (o quelli) più in linea con l'obiettivo.

- ✓ **Evaluation**: i risultati dei modelli vengono valutati in base ai goal definiti; potrebbero essere identificati nuovi obiettivi derivanti dai risultati ottenuti dall'applicazione dei modelli. CRISP-DM è a tutti gli effetti un processo iterativo, la decisione di considerare o meno questi nuovi obiettivi è presa in questa fase, prima di passare alla fase finale.
- ✓ **Deployment**: le informazioni ottenute alla fine dei processi precedenti sono presentate in modo semplice ed intuitivo alle parti interessate, secondo le aspettative ed i requisiti aziendali del cliente.

Nello specifico, la fase di preparazione dei dati comporta il recupero e l'aggregazione delle diverse origini dati per creare "feature" che meglio descrivono le condizioni di salute di una entità da analizzare in un determinato momento; va da sé che l'aggregazione di dati va eseguita secondo quanto richiesto dall'algoritmo di Machine Learning individuato. È necessario, quindi, eseguire passaggi di pre-elaborazione delle diverse fonti dati al fine di preparare il set di dati finale corretto da dare in pasto all'algoritmo.

Il framework CRISP è iterativo e ciclico e si integra naturalmente con un approccio agile basato sul rilascio incrementale.

Nei paragrafi seguenti vengono fornite le caratteristiche che il componente "Data Science Tool" dovrebbe supportare per coprire le fasi del processo di gestione dei dati che partendo dalla loro preparazione consente la progettazione e valutazione di un modello di apprendimento e la sua messa in esecuzione su nuovi dati in arrivo.

5.2.1 *Data Preparation*

Nel ciclo di vita della Data Science, la preparazione dei dati è il primo step concreto verso la produzione di modelli predittivi, dopo aver compreso gli obiettivi di business da perseguire. Si tratta dell'atto di manipolare (o pre-elaborare) i dati grezzi, che possono provenire da diverse fonti di dati, in un modulo che può essere analizzato prontamente e accuratamente. Un buon processo di preparazione dei dati, infatti, rappresenta una

fase cruciale per la costruzione di un modello predittivo e si focalizza sull'organizzare e preparare i dati al fine di ottenere il massimo beneficio dall'analisi di tali dati.

Solitamente, la preparazione dei dati è composta dalle svariate tecniche. Si possono distinguere due macro-fasi distinte:

1. Fase di Data Wrangling, Data Cleaning, e Data Selection e Transformation;
2. Fase di Exploratory Data Analysis.

La prima fase riguarda nello specifico una serie di processi di elaborazione e mapping dei dati grezzi in dati pronti per essere processabili con l'intento di renderli più appropriati e valutabili per svariati task. In particolare, la pulizia dei dati (*data cleaning* o data cleansing) individua e rimuovi dati corrotti o inaccurati, e risulta fondamentale come processo che in genere occupa la maggior parte del tempo nel ciclo di vita della data science. Un buon processo di pulizia dei dati dovrebbe permettere di risolvere tutta una serie di problemi, quali tra cui l'utilizzo di dati doppi e ridondanti, dati mancanti, dati inconsistenti, dati non necessari, rumore e outliers. Un un buon processo di preparazione dei dati è necessario identificare quelle che sono le caratteristiche rilevanti (*feature selection*) che impattano sull'analisi e la costruzione del modello predittivo. Individuare queste caratteristiche permette di conoscere cosa influenza maggiormente i tuoi dati e di conseguenza anche il processo decisionale futuro. L'obiettivo in questa fase è identificare il più piccolo insieme di caratteristiche che meglio descrivono il problema che si vuole studiare. In questo modo si cerca di escludere gli attributi irrilevanti e ridondanti da dati che non contribuiscono all'accuratezza di un modello predittivo o possono in effetti ridurre l'accuratezza dell'analisi che si sta effettuando. In alcuni casi è richiesta anche la necessità di voler modificare il formato o il confronto tra le grandezze analizzate (*data transformation*) onde evitare di ottenere risultati fuorvianti. Questa operazione di trasformazione del range della caratteristica è definita scaling e permette di evitare che le caratteristiche con alti valori possano dominare i risultati dell'analisi.

La seconda fase riguarda il processo di analisi del dataset per riassumerne le principali caratteristiche, anche con metodi di visualizzazione dei dati, per vedere ciò che i dati possono dirci oltre l'attività di modellazione predittiva. Parte di questa attività riguarda il

massimizzare la comprensione di un dataset, scoprirne la struttura sottostante, estrarre variabili importanti, rilevare anomalie identificabili solo in maniera visiva, testare le ipotesi sottostanti, sviluppare modelli parsimoniosi e determinarne le impostazioni ottimali.

Sebbene la fase 1) sia una operazione totalmente a carico del data scientist e fortemente caratterizzata dalla tipologia dei dati collezionati, la fase 2) è una fase automatizzabile e fruibile in maniera programmatica, i cui risultati sono però oggetto di interpretazione umana.

Pertanto, il componente Data Science Tool dovrà predisporre funzionalità per l'esplorazione del dataset per supportare il data scientist nell'intera fase di ciclo di vita della data science. Di seguito vengono descritte le caratteristiche delle funzionalità di Exploratory Data Analysis (EDA) che dovranno essere supportate, ovvero Data Exploration, Data Characterization e Data Correlation.

5.2.1.1 *Data Exploration*

Quando si procede all'analisi dei dati si possono scoprire fatti già conosciuti, e di solito con un dettaglio maggiore, oppure delle relazioni tra i dati non conosciute (e quindi scoperta di nuove informazioni). Nel primo caso rientriamo nell'analisi descrittiva vera e propria, che descrive quanto accaduto. Nel secondo caso, invece, rientriamo dentro l'analisi esplorativa dei dati: essa non si focalizza solamente sulla comprensione dei dati, piuttosto si pone l'obiettivo di cercare le scoperte, le tendenze, le correlazioni o le relazioni tra i dati per generare idee o ipotesi. Indipendentemente dall'analisi, i dati utilizzati possono essere definiti come:

- Discreti e categorici: rappresentati da particolari valori finiti o infiniti in un dato insieme di dati numerici (ad esempio, il numero di personale utilizzato) o categorico (ad esempio, maschio o femmina).
- Continui: dati che risultano quando un'osservazione può assumere qualsiasi valore entro un certo intervallo (ad esempio, letture di temperatura).

Questi dati vengono riassunti e rappresentati tramite la statistica descrittiva. Quest'ultima è un tipo di statistica che da un insieme di dati analizzati riassume e interpreta le proprietà dello stesso insieme di dati. I principali metodi utilizzati dalla statistica descrittiva sono:

- frequenze, distribuzioni e tabulazioni: strumenti statistici che vengono utilizzati per esaminare il conteggio delle occorrenze di valori all'interno di un particolare gruppo o intervallo.
- tendenza centrale, media, mediana e moda dei dati: non sono altro che altri strumenti statistici che permettono di riepilogare i dati in un singolo valore tipico o rappresentativo di tutti i valori nel set di dati analizzato.
- Intervallo, quartili, varianza o deviazione standard: ci mostrano quanto sono sparsi i valori e quanto essi differiscono dal valore medio.

Le statistiche descrittive sono molto importanti perché se presentassimo semplicemente i nostri dati grezzi sarebbe difficile rendere visibile ciò che i dati mostrano, specialmente se ce ne sono tanti. Le statistiche descrittive ci consentono quindi di presentare i dati in un modo più significativo, che consente un'interpretazione più semplice dei dati. Infatti, l'obiettivo della statistica descrittiva è quello di ricavare da un insieme di dati raccolti in tabelle e grafici (troppo numerosi per poter essere esaminati singolarmente) alcune informazioni significative per il problema studiato.

Chi esegue l'analisi descrittiva dovrebbe cercare di riconoscere prima un fenomeno di interesse. A volte il fenomeno emerge dai dati; a volte nasce dall'esperienza e a volte deriva da lacune nella ricerca esistente. Non tutti i fenomeni o le domande sui fenomeni sono ben articolati o possono essere risolti con dati esistenti o raccolti e, spesso, chi analizza i dati deve ripensare, riconsiderare e rivalutare la domanda di studio fino a quando è ben articolata, concettualmente chiara e fattibile. Una volta che un fenomeno è stato identificato, l'analista deve considerare pienamente il fenomeno in questione, determinare quali caratteristiche sono più salienti e definire costrutti rilevanti (misure) che rappresentino queste caratteristiche. Questo approccio all'analisi descrittiva è

iterativo e richiede una riconsiderazione e una modifica man mano che avviene la comprensione del fenomeno.

La Data Governance del Middleware mette quindi a disposizione, in modalità esperto, un notebook Zeppelin con delle funzioni di statistica descrittiva (valore minimo, valore massimo, valore medio, mediana, e deviazione standard) già predisposte. Oltre a visualizzare l'eventuale presenza di valori nulli in una colonna, esso permette la creazione e la visualizzazione di:

- *grafici box plot*, utili per descrivere la distribuzione di un campione;
- *grafici di distribuzione*, utili per avere una rappresentazione grafica di una variabile continua;
- *grafici scatter*, per visualizzare la distribuzione dei dati confrontando due variabili.

5.2.1.2 *Data Characterization*

I dati devono essere interpretati per delinearne un significato. La statistica inferenziale offre strumenti per capire se un certo risultato è significativo o meno. Il principio generale è: più campioni misuriamo, minori sono gli errori che commettiamo nello stimare le grandezze statistiche della popolazione. L'inferenza è il processo di prendere conclusioni sulle popolazioni da campioni. L'inferenza include la maggior parte delle attività tradizionalmente associate a statistiche quali: stima, intervalli di confidenza, test di ipotesi e variabilità. L'inferenza ci obbliga a definire formalmente obiettivi di stime o ipotesi. Ci obbliga a pensare alla popolazione che stiamo cercando di generalizzare dal nostro campione. Un campione rappresentativo è quello che viene estratto senza pregiudizi (*bias*) dalla popolazione di interesse. La rappresentatività non dipende tanto dalla dimensione del campione, ma dipende dalla giusta composizione del campione. La tecnica per verificare questa rappresentatività è chiamata Test di Verifica dell'Ipotesi, o più semplicemente test di ipotesi. Il test di ipotesi viene utilizzato per confrontare due dataset. È un metodo di inferenza statistica, quindi, alla fine del test, si arriverà ad una conclusione riguardo l'esistenza di una differenza tra i gruppi che stiamo confrontando. Anche se nella vita reale a volte è possibile generare alcune statistiche di base o

conoscenze di base da un censimento o da alcune ricerche precedenti per utilizzare il test di ipotesi, il più delle volte non si hanno a disposizione dati come la deviazione standard e la media della popolazione. Quindi si dovrà essere abbastanza sicuri che il campione sia rappresentativo della popolazione. Tuttavia, se possiamo o non possiamo usare questa tecnica per assicurare la rappresentatività del nostro campione, nel contesto dell'apprendimento automatico, possiamo usare il test di ipotesi per verificare se il nostro test set è rappresentativo dei dati di apprendimento. Per questo, ci sono alcuni concetti chiave che dovremmo conoscere:

- L'ipotesi nulla H0: indica l'assunzione del test statistico, chiamata anche assunzione di default o "assunzione che non è cambiato nulla". Indica che l'assunzione del test è valida e non può essere rifiutata ad un certo livello di significatività.
- L'ipotesi alternativa H1: indica il risultato che si spera di dimostrare, ovvero che l'assunzione del test non non è valida e viene rifiutata a un certo livello di significatività.
- Il tipo di test: Un test è un tipo di statistica inferenziale che viene utilizzato per determinare se esiste una differenza significativa tra le medie di due gruppi che possono essere correlate in determinate caratteristiche. Viene utilizzato a seconda che il dataset segua una certa distribuzione e possa presentare varianze sconosciute. Ad esempio, il T-test viene utilizzato come strumento di verifica delle ipotesi, che consente di verificare un'ipotesi applicabile a una popolazione con una distribuzione normale. Per verificare l'ipotesi si fa affidamento sul Teorema del limite centrale, il quale stabilisce che se riuniamo un insieme di variabili casuali indipendenti (ad esempio, la media dei gruppi indipendenti) dovrebbero seguire una distribuzione normale anche se le stesse variabili originali non sono normalmente distribuite. Supponiamo quindi che si ottenga un campione contenente un gran numero di osservazioni, ciascuna osservazione viene generata casualmente in un modo che non dipenda dai valori delle altre osservazioni e che viene calcolata la media aritmetica dei valori osservati. Se questa procedura viene eseguita più volte, il teorema del limite centrale afferma

che la distribuzione della media sarà strettamente approssimata da una distribuzione normale.

- Il concetto di p-value: formulata un'ipotesi H0, ed effettuato un esperimento consistente nella misurazione di alcuni campioni di una popolazione, diciamo che l'esperimento conferma l'ipotesi se il suo p-value è al di sotto di una soglia, detta significatività, da noi stabilita a priori. Il p-value esprime la probabilità di ottenere risultati simili a quelli effettivamente ottenuti nel caso in cui H0 sia falsa.

Il componente Data Science Tool dovrà, quindi, mettere a disposizione funzionalità di statistica inferenziale da applicare ai dati selezionati. In particolare, dovrà essere possibile effettuare:

1. il *test statistico parametrico T-test* **[TT]** per verificare se le medie di due campioni indipendenti sono significativamente diverse:
    - le osservazioni in ciascun campione sono indipendenti e identicamente distribuite (iid);
    - le osservazioni in ciascun campione sono normalmente distribuite;
    - le osservazioni in ciascun campione hanno la stessa varianza.
    - Interpretazione H0: le medie dei campioni sono uguali.
    - Interpretazione H1: le medie dei campioni non sono uguali.
2. il *test statistico non parametrico di Kolmogorov-Smirnov* **[KS1, KS2]** per:
    - verificare la distanza tra una funzione di distribuzione empirica di un campione e una funzione di distribuzione cumulativa;
    - misurare il grado di separazione tra le distribuzioni positive e negative.
    - IL test non ha assunzioni sulla distribuzione dei dati.
    - Interpretazione H0: il campione proviene davvero da una distribuzione P.
    - Interpretazione H1: il campione non proviene dalla distribuzione P.
3. il *calcolo delle cumulate empiriche* **[CD]** nel caso in cui si vuole rappresentare la funzione di ripartizione della misura empirica di un campione. La funzione di

ripartizione empirica è una stima della vera funzione di ripartizione che ha generato il campione. Questa funzione di distribuzione cumulativa è una funzione a gradini che salta di 1/*n* in ciascuno degli *n* punti di dati. Il suo valore, a qualsiasi valore specificato della variabile misurata, è la frazione di osservazioni della variabile misurata che sono inferiori o uguali al valore specificato. La funzione di distribuzione empirica è una stima della funzione di distribuzione cumulativa che ha generato i punti nel campione. Converge con probabilità 1 a quella distribuzione sottostante, secondo il teorema di Glivenko-Cantelli. Esistono numerosi risultati per quantificare il tasso di convergenza della funzione di distribuzione empirica con la funzione di distribuzione cumulativa sottostante.

[TT] Student (William Sealy Gosset), The probable error of a mean, in Biometrika, vol. 6, n. 1, marzo 1908, pp. 1-25, DOI:10.1093/biomet/6.1.1.

[KS1] Kolmogorov, A. 1933. "Sulla Determinazione Empirica di una Legge di Distributione". Giornale dell'Istituto Italiano degli Attuari, 4: 83-91.

[KS2] Smirnov, N (1948). "Table for estimating the goodness of fit of empirical distributions". Annals of Mathematical Statistics. 19 (2): 279-281. doi:10.1214/aoms/1177730256.

[CD] Ayer, M., Brunk, H. D., Ewing, G. M., Reid, W. T., & Silverman, E. (1955). An empirical distribution function for sampling with incomplete information. The annals of mathematical statistics, 641-647.

5.2.1.3 *Data Correlation*

Studiare la correlazione significa investigare l'esistenza di un legame per cui a variazioni di un fenomeno corrispondono variazioni dell'altro fenomeno secondo una determinata relazione. In statistica, la correlazione è un'analisi bivariata che misura la forza dell'associazione tra due variabili e la direzione della relazione. In termini di forza della relazione, il valore del coefficiente di correlazione varia tra +1 e -1. Un valore esatto di ± 1 indica un grado perfetto di associazione tra le due variabili. Se il valore del coefficiente

di correlazione va verso 0, la relazione tra le due variabili sarà più debole. La direzione della relazione è indicata dal segno del coefficiente; un segno + indica una relazione positiva e un segno - indica una relazione negativa.

In letteratura sono noti tre diversi coefficienti di correlazione. Tutti e tre i metodi possono essere inquadrati come metodi non-parametrici. Di seguito vengono presentati i tre metodi di correlazione nel dettaglio.

1. *Coefficiente di Correlazione di Pearson* **[CP]**: testa se due campioni hanno una correlazione lineare. Le assunzioni di valida esecuzione del test sono le seguenti:
    - Le osservazioni in ciascun campione sono indipendenti e identicamente distribuite (iid).
    - Ogni osservazione dovrebbe avere una coppia di valori.
    - Ogni variabile dovrebbe essere continua.
    - Le osservazioni in ciascun campione sono normalmente distribuite.
    - Le osservazioni in ciascun campione hanno la stessa varianza.
    - Si assume che ci sia assenza di valori anomali (outliers).
    - Presuppone linearità e omoschedasticità.
    - Interpretazione di H0: i due campioni sono indipendenti.
    - Interpretazione di H1: esiste una dipendenza tra i campioni.
2. *Coefficiente di Correlazione per Ranghi di Spearman* **[CS]**: testa se due campioni hanno una correlazione monotona. Le assunzioni di valida esecuzione del test sono le seguenti:
    - Le osservazioni in ciascun campione sono indipendenti e identicamente distribuite (iid).
    - Le osservazioni in ciascun campione possono essere ordinate.
    - Due variabili devono essere misurate su una scala ordinale, di intervallo o di rapporto.
    - Presuppone che vi sia una relazione monotona tra le due variabili.
    - Interpretazione di H0: i due campioni sono indipendenti.
    - Interpretazione di H1: esiste una dipendenza tra i campioni.

3. *Coefficiente di Correlazione per Ranghi di Kendall* **[CK]**: testa se due campioni hanno una correlazione monotona. Le assunzioni di valida esecuzione del test sono le seguenti:
   - Le osservazioni in ciascun campione sono indipendenti e identicamente distribuite (iid).
   - Le osservazioni in ciascun campione possono essere ordinate.
   - Due variabili devono essere misurate su una scala ordinale, di intervallo o di rapporto.
   - Presuppone che vi sia una relazione monotona tra le due variabili.
   - Interpretazione di H0: i due campioni sono indipendenti.
   - Interpretazione di H1: esiste una dipendenza tra i campioni.

Data la similarità tra i metodi di Spearman e Kendall, di seguito si riportano differenze tra le due:

- Nel caso normale, la correlazione di Kendall è più solida ed efficiente della correlazione di Spearman. Significa che la correlazione di Kendall è preferita quando ci sono piccoli campioni o alcuni valori anomali.
- La correlazione di Kendall ha una complessità computazionale O (n^2) rispetto a O (n logn) della correlazione di Spearman, dove n è la dimensione del campione.
- Il rho di Spearman di solito è più grande della tau di Kendall.
- L'interpretazione della tau di Kendall in termini di probabilità di osservare le coppie concordanti e discordanti è molto diretta.

Il componente Data Science Tool dovrà, quindi, mettere a disposizione le funzionalità per scoprire eventuali relazioni lineari tra due variabili. Le funzioni di correlazione dovranno permettere di calcolare:

- il coefficiente di correlazione di Pearson;
- il coefficiente di correlazione per ranghi di Spearman;
- il coefficiente di correlazione per ranghi di Kendall.

Per ogni tecnica dovrà essere calcolato l'indice, il p-value e prodotto lo scatter matrix.

[CP] Pearson, K. (1895). "Notes on regression and inheritance in the case of two parents". Proceedings of the Royal Society of London, 58: 240–242.

[CS] Spearman, C. (1904). "The proof and measurement of association between two things". American Journal of Psychology. 15 (1): 72–101. doi:10.2307/1412159

[CK] Kendall, M. (1938). "A New Measure of Rank Correlation". Biometrika. 30 (1–2): 81–89. doi:10.1093/biomet/30.1-2.81

5.2.2 *Data Modeling*

La formulazione di una ipotesi e la modellazione sono le fasi cruciali in un progetto di data science che non si limitano ai campioni statistici. L'addestramento del modello consiste essenzialmente nel trovare il modello che risponde alle domande in modo più accurato. Il processo di formazione del modello prevede numerosi passaggi e iterazioni:

1. Determinare la forma funzionale del modello: definire la variabile target, determinando se questa debba essere trasformata in base all'obiettivo e individuandone la distribuzione: ad esempio, la variabile target è un valore binario (Sì o No) o un valore continuo come un importo in dollari? Queste decisioni tecniche fondamentali dovrebbero essere riviste e documentate.
2. Dividere i dati di input in modo casuale per la modellazione in un dataset di addestramento e uno di test: qui viene eseguita una fase secondaria chiave per la selezione del modello. Ciò comporta la separazione di una parte del dataset per l'addestramento dei modelli candidati di apprendimento automatico, una parte del dataset per la validazione e un'altra parte del dataset per il test atto alla fase confronto delle prestazioni del modello e la selezione del modello più performante, attraverso la misurazione dell'accuratezza del modello e la prevenzione del sovra-adattamento (overfitting).
3. Costruire i modelli utilizzando il dataset di addestramento: in genere l'idea è quella di applicare svariate tecniche di apprendimento automatico.

4. Valutare il modello addestrato sui dati di training e di test. Utilizzare una serie di metriche di valutazione delle performance.
5. Determinare la soluzione "migliore" per rispondere alla domanda confrontando le metriche di successo tra metodi alternativi.

Per la fase di **Data Modeling**, il componente "Data Science Tool" ha l'obiettivo di fornire all'utilizzatore finale uno strumento in grado di supportarlo nell'intero processo di creazione, test e validazione di modelli di machine learning, mettendo a disposizione differenti algoritmi di classificazione e clustering. Lo strumento dovrà, pertanto, mettere a disposizione un'interfaccia intuitiva attraverso la quale l'utente potrà creare e testare i propri workflow in maniera rapida ed interattiva. Le attività di **Data Modeling** dovranno essere possibili in due modalità differenti:

- ✓ Modalità Interactive Computing
- ✓ Modalità Assistita

Modalità Interactive Computing

Il componente "Data Science Tool" metterà a disposizione una interfaccia simile a quella di un notebook; un ambiente interattivo basato sul Web che permetterà di combinare l'esecuzione di codice, testo, modelli matematici e grafici in un singolo documento. Tale interfaccia consentirà di acquisire, rieseguire, presentare e condividere l'intero workflow. Permetterà, in modo interattivo, di importare file, creare modelli e migliorarli iterativamente.

Modalità Assistita

Il componente "Data Science Tool" consentirà anche ad utenti che non hanno esperienza di programmazione, di creare, lanciare e valutare il proprio modello di machine learning. Il modello sarà costruito in modo da guidare l'utente in ogni passo, fornendo suggerimenti di input, guida interattiva e flussi di esempio.

In entrambe le modalità una funzionalità di import dovrà consentire all'utente di caricare i propri dati e di procedere all'importazione e all'analisi degli stessi. Il requisito minimo sarà la gestione di file in formato CSV.

Per l'addestramento di modelli di classificazione, si dovrà poter scegliere la tipologia di algoritmo da lanciare sui propri dati, da una lista di algoritmi disponibili, quali, ad esempio: Cox Proportional Hazards (CoxPH), Deep Learning (DL), Distributed Random Forest (DRF), Generalized Linear Modeling (GLM), Gradient Boosting Machine (GBM), Naive Bayes, Stacked Ensembles, XGBoost.

L'utente potrà indicare il dataset di training e quello di validazione, la colonna di risposta da utilizzare come variabile dipendente, il numero di alberi in caso di scelta di algoritmi quali Gradient Boosting Machine (GBM), Distributed Random Forest (DRF) e XGBoost, ed altre innumerevoli specifiche utili a caratterizzare il modello sulla base della tipologia di algoritmo scelto.

Per l'addestramento di modelli di clustering, si dovrà poter selezionare la tipologia di algoritmo che si vuole lanciare sui propri dati, da una lista di algoritmi, quali ad esempio: Aggregator, Generalized Low Rank Models (GLRM), KMeans Clustering, Isolation Forest, Principal Component Analysis (PCA), Word2vec.

Una volta creato il modello di classificazione o clustering, l'utente potrà rivederlo tra i modelli salvati ed entrare nei dettagli associati.

I modelli addestrati potranno essere esportati e salvati in un apposito formato di serializzazione binaria, ad esempio il formato pickle, nell'apposito ML Model Store messo a disposizione dal layer di Data Governance.

Analogamente dovrà, inoltre, essere possibile importare un proprio modello per poterlo testare sul dataset selezionato e comparare i risultati ottenuti con quelli raggiunti attraverso altri modelli.

Relativamente alla validazione dei modelli, per valutare i risultati riportati dalla loro esecuzione, la componente dovrà mettere a disposizione dei grafici riepilogativi. Nel caso della classificazione, grafici significativi sono, ad esempio:

- ✓ Scoring history: (GBM, DL) Rappresenta il tasso di errore del modello così come è stato creato.

- ✓ Importanza variabili: (GBM, DRF, DL) Rappresenta il significato statistico di ciascuna variabile nei dati in termini di influenza sul modello. Le variabili sono elencate in ordine di maggiore o minore importanza.
- ✓ Confusion Matrix: (RF, GBM) Tabella che mostra le prestazioni dell'algoritmo in termini di falsi positivi, falsi negativi, veri positivi e veri negativi. I risultati effettivi vengono visualizzati nelle colonne e le previsioni vengono visualizzate nelle righe.
- ✓ Curva ROC: (DRF) Una curva ROC è un grafico che rappresenta il rapporto tra i veri positivi e i falsi positivi.

I workflow creati da ogni utente, inoltre, potranno essere salvati per essere utilizzati e modificati in un secondo momento. Tali flussi dovranno essere salvati nella directory dei flussi, sotto la directory dell'utente. Di tali flussi dovrà essere consentito anche il download in locale.

5.2.3 *Data Prediction*

L'obiettivo di questa fase è determinare se il modello è adatto alla produzione, distribuire i modelli in un ambiente di produzione o simile alla produzione per l'accettazione da parte dell'utente finale. Gli utenti devono convalidare le prestazioni dei modelli e, in caso di problemi con il modello, devono essere risolti in questa fase.

- *Valutare le prestazioni del modello*: la stabilità di un modello vuol dire che è in grado di continuare a funzionare nel corso del tempo. La stima si concentra sulla valutazione (a) dell'adattamento generale del modello, (b) del significato di ciascun predittore e (c) della relazione tra la variabile target e ciascun predittore.
- *Distribuire il modello per la predizione in tempo reale*: l'implementazione di modelli di machine learning in produzione può essere eseguita in un'ampia varietà di modi. La forma più semplice è la predizione batch. Dato un dataset, si esegue il tuo modello e si produce una predizione su base giornaliera o settimanale. Il tipo di predizione più comune è un semplice servizio Web. I dati

grezzi vengono trasferiti in tempo reale tramite una API REST. Questi dati possono essere inviati come JSON arbitrario che consente la completa libertà di fornire qualsiasi dato disponibile.

- *Ricostruire il modello*: nel tempo un modello perderà la sua predicibilità a causa di molte ragioni: l'ambiente di business può cambiare, la procedura può cambiare, possono diventare disponibili più variabili o alcune variabili diventano obsolete. Pertanto, è utile monitorare la predicibilità nel tempo e decidere di ricostruire eventualmente il modello se l'accuratezza va sotto una certa soglia.

Nella Data Governance del Middleware di DF, i modelli di ML validati potranno essere salvati in un apposito data storage, così da permettere all'utilizzatore di passare dalla fase di Design a quella di Deployment, potendo quindi mettere in esecuzione il modello predittivo su dati reali in real-time o inviati alla piattaforma DF a lotti batch. In ogni caso il modello potrà essere eseguito in esercizio in una sorta di fase di test continuo.

L'utente può essere sicuro dei risultati di predizione a seconda del grado di accuratezza certificato nella fase di design del modello. Ciò però non è sempre indice di alte performance nella fase di deployment. I dati possono cambiare o le caratteristiche del fenomeno analizzato possono variare nel tempo. Questo, a lungo termine, porta ad abbassare l'accuratezza predittiva del modello di ML addestrato. L'utente potrà utilizzare proprio i nuovi dati raccolti per avviare una nuova fase di re-fining del modello, che deve includere nella sua generalizzazione anche i nuovi casi non previsti in fase di addestramento iniziale. La fase di re-training può avvenire a seconda delle necessità:

- L'accuratezza del modello va sotto una certa soglia definita a priori;
- Il riaddestramento viene definito dopo aver collezionato nel tempo di esercizio un numero sufficiente di nuovi dati utili alla fase di refining.

Una volta riaddestrato, il modello potrà essere quindi salvato sempre nell'apposito data storage di modelli, indicando eventualmente l'avvenuto aggiornamento con una opportuna operazione di versioning.

5.3 Machine Learning Libraries (ML)

Machine Learning Libraries rappresenta un Framework costituito dato da un set di API che potranno essere inserite sia nelle pipeline di ML e DL per addestrare i modelli e sia nella fase di utilizzo dei modelli addestrati per effettuare previsioni. Inoltre, la componente Machine Learning Libraries fornirà funzioni basiche di statistica come minimo, media, varianza e il numero di non zero, nonché il conteggio totale e funzioni di Algebra Lineare. Metterà a disposizione anche funzioni di riduzione della dimensionalità (*Dimensionality Reduction*) che consentirà di ridurre il numero di variabili casuali in esame, ottenendo un insieme di variabili principali attraverso funzionalità di estrazione delle caratteristiche (Feature Extraction)**,** ossia trasformazione dei dati nello spazio con molte dimensioni in uno spazio con meno dimensioni. La trasformazione dei dati potrà essere lineare, come in Principal Component Analysis (PCA), ma si potranno attuare molte tecniche di riduzione della dimensionalità non lineare.

Di seguito vengono elencate le principali tecniche di ML che saranno supportate da tale componente.

5.3.1 *Classificazione*

La **Classificazione** è una procedura statistica che permette di associare ciascun oggetto (che sia un'immagine, un pattern, un dato numerico, etc.), appartenente a un generico spazio multidimensionale, a una o più etichette, corrispondenti alle possibili classi di cui può far parte un oggetto. Sono algoritmi che hanno la funzione di fornire in output la classe di appartenenza di ciascun campione in ingresso. Nello specifico identifica a quale insieme di categorie appartiene una nuova osservazione, sulla base di un insieme di dati formativi contenenti osservazioni (o istanze) la cui appartenenza alla categoria è nota. La classificazione è applicabile in contesti come riconoscimenti di immagini oppure per stimare la probabilità che una apparecchiatura possa guastarsi entro X unità di tempo. Riuscire ad interpretare ad esempio l'immagine ed estrarne informazioni utili non è mai un processo immediato e semplice, spesso la possibilità di codificare al meglio l'informazione è minata da diversi fattori tra i quali la scorretta acquisizione

dell'immagine o un errore nel modello di interpretazione. Per ovviare a questo diviene necessario un processo durante il quale si ricercano le caratteristiche salienti delle immagini, dalle quali si estraggono una serie di informazioni specifiche. Grazie all'utilizzo di queste informazioni e attraverso il riconoscimento delle caratteristiche individuate, diviene possibile classificare un oggetto mai visto prima, riuscendo ad attribuirgli una particolare classe di appartenenza o una label definita a priori. L'individuazione delle feature quindi ricopre una fase fondamentale ed un errore in questo particolare processo può pregiudicare l'esito della classificazione.

Gli Esperti di Dominio potranno applicare questa categoria di algoritmi per realizzare un proprio classificatore che, tenuto conto dei dati a disposizione e di classi di appartenenza pre-configurate e dopo un addestramento specifico, sarà in grado di riuscire a classificare un nuovo dato in ingresso. Il processo di classificazione consisterà nello specifico nell'adottare le funzioni algoritmi base di classificazione offerte Machine Learning Libraries (ML) per creare una nuova funzione che associa ad un certo dato in ingresso una sua specifica classe di appartenenza tenendo conto delle caratteristiche di ciascun dato. Si potranno costruire innumerevoli classificatori basati su varie tecniche e con differenti prestazioni in funzione della tipologia del problema da risolvere.

Quindi lo scopo principale dell'adozione da parte dei Pillar dell'algoritmo di classificazione mediante un approccio di apprendimento supervisionato sarà principalmente quello di definire delle buone regole di decisione per l'assegnazione delle etichette ad ogni oggetto sulla base delle informazioni raccolte dal dataset di training di cui si conosce la classe di appartenenza degli oggetti.

5.3.2 *Regressione*

La **Regressione** è un processo statistico per individuare e stimare le relazioni esistenti tra variabili e, in particolar modo, per comprendere ed esplorare le relazioni esistenti tra una variabile dipendente e una o più variabili indipendenti. Più specificamente, l'analisi di regressione aiuta a comprendere come il valore tipico della variabile dipendente cambia quando viene variata una qualsiasi delle variabili indipendenti, mentre le altre

variabili indipendenti sono mantenute fisse. In circostanze limitate, l'analisi di regressione può anche essere utilizzata per dedurre le relazioni causali tra le variabili indipendenti e dipendenti. Per determinare la relazione di dipendenza tra le diverse variabili del problema si possono utilizzare diversi modelli come la Regressione Lineare o la Regressione Logistica.

La Regressione è anche ampiamente utilizzata per analisi predittive o analisi previsionali; ad esempio, può essere applicata per stimare/prevedere il valore futuro di un dato avendo noto il suo valore attuale: previsione della quotazione delle valute o delle azioni di una società (borsa) o nel previsione del tasso di risposta di una campagna sulla base di un dato profilo di clienti (marketing) o variazione del fatturato dell'azienda al mutare della strategia (ambito commerciale) o, infine, previsione della vita utile rimanente di un asset (industria manifatturiera).

Gli esperti di dominio potranno definire modelli predittivi di vario genere da implementazioni di nuovi algoritmi adottando di base il processo statistico di regressione supportato da Machine Learning Libraries.

5.3.3 *Clustering*

Il **Clustering** è un insieme di metodi non supervisionati in grado di individuare raggruppamenti intrinseci (cluster) tra i dati presenti in uno spazio multidimensionale ed in grado di creare classi omogenee in corrispondenza di tali raggruppamenti. La complessità del Clustering è data dalla mancanza di conoscenza a priori sia delle classi da individuare che del numero delle stesse, utilizzando tecniche che si basano su valutazioni delle misure di similarità/dissimilarità tra i vari elementi analizzati. I metodi di clustering possono essere ricondotti a due famiglie:

- ✓ **Clustering gerarchico**: attraverso operazioni di tipo bottom-up o top-down organizza i dati in una sequenza innestata di gruppi che possono essere visualizzati come un albero.

- ✓ **Clustering partizionale**: individua (solitamente attraverso algoritmi euristici iterativi) delle partizioni che minimizzano un dato criterio di clustering, ad esempio la somma dei quadrati degli errori.

Esistono, inoltre, più tecniche che permettono di fare clustering un insieme di oggetti. Una prima importante suddivisione dipende dalla tecnica di generazione dei cluster stessi, che divide gli algoritmi in due categorie:

- ✓ **Algoritmi di clustering agglomerativi (bottom-up):** Iniziano inserendo ogni oggetto dell'insieme in un proprio cluster per poi raggrupparli iterativamente fino al raggiungimento di una condizione specifica (es. numero di cluster desiderato).
- ✓ **Algoritmi di clustering divisivi (top-down):** Iniziano inserendo tutti gli oggetti dell'insieme in un unico cluster per poi separarlo iterativamente in cluster più piccoli fino al raggiungimento di una condizione specifica.

Il risultato finale, in entrambi i casi, è un insieme di cluster contenenti uno o più oggetti. A prescindere dall'approccio utilizzato, gli algoritmi di clustering si basano tutti su una **metrica**, puramente geometrica, che permetta di identificare quanto simili siano due oggetti fra di loro.

Comuni a entrambe le famiglie sono i criteri di clustering che si possono scegliere per specificare il grado di ottimalità di ogni soluzione ammissibile.

5.3.4 *Deep Learning*

La sfida principale all'intelligenza artificiale riguarda la risoluzione di compiti che si dimostrano essere facili da eseguire per le persone, ma difficili da descrivere formalmente, che risolviamo intuitivamente e sentiamo essere "automatici", come il riconoscimento delle parole o dei volti nelle immagini. La soluzione a problemi intuitivi permette ai computer di apprendere dall'esperienza e a capire il mondo in termini di gerarchia di concetti. In tal senso, ogni concetto viene definito attraverso la sua relazione con concetti più semplici. Collezionando conoscenza dall'esperienza, questo approccio evita la necessità per le persone di specificare formalmente tutta la

conoscenza che richiede un computer. La gerarchia di concetti consente al computer di apprendere concetti complicati costruendoli da concetti più semplici. Se si disegna un grafo che mostra come questi concetti vengono costruiti uno sopra l'altro, il grafo è profondo ("deep"), con molti strati. Per questo motivo tale approccio, nell'ambito del Machine Learning, viene definito Deep Learning.

Quindi, un sistema di Deep Learning apprende multipli livelli di rappresentazione che corrispondono a differenti livelli di astrazione. Questi livelli formano dunque una gerarchia di concetti (cfr. Figura 20).

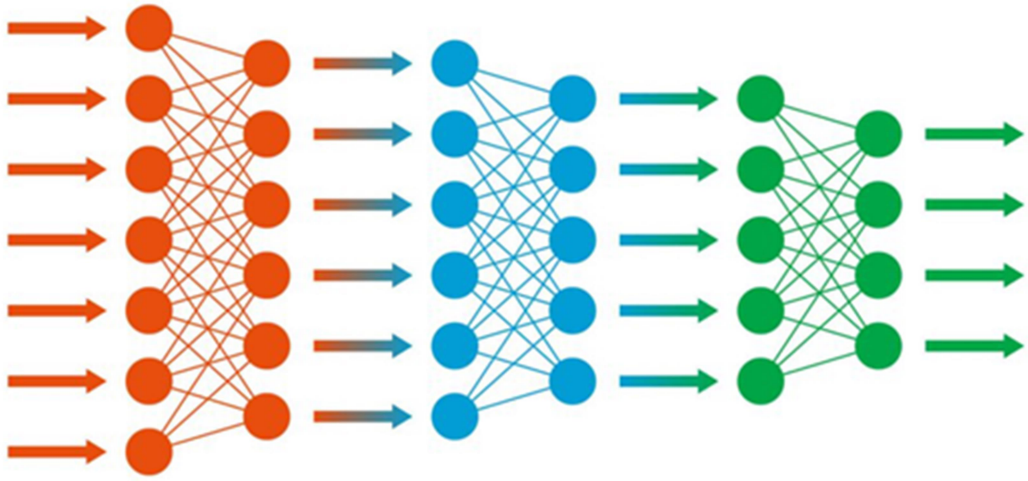


Figura 20: Livelli di Astrazione di una rete neurale

È importante precisare che il Deep Learning è un sottoambito della più ampia classe di algoritmi di apprendimento automatico, in quanto è in grado di eseguire problemi di apprendimento supervisionato (attraverso task di classificazione o regressione) e non supervisionato (attraverso l'analisi di pattern).

Gli algoritmi di Deep Learning usano vari livelli di unità non lineari a cascata, chiamati strati nascosti (*hidden layers*) per svolgere compiti di estrazione di caratteristiche (*features*), trasformazione e rappresentazione dei dati. Ciascun livello successivo utilizza l'output del livello precedente come input così che le caratteristiche di più alto livello possano essere derivate da quelle di livello più basso per creare una rappresentazione gerarchica. Ad esempio, nella classificazione di immagini, i nodi in input sono la codifica

numerica di ogni singolo pixel dell'immagine, mentre il nodo (o i nodi) finali di output consentono la classificazione di un oggetto (o più oggetti) all'interno dell'immagine.

5.3.4.1 *Integrazione di TensorFlow in Digital Future Middleware*

Il Data Governance / Analytics Layer consentirà di eseguire elaborazione dei dati con tecniche di Machine Learning e Deep Learning tramite una istanza creata ad hoc con Apache Zeppelin. Apache Zeppelin è una soluzione Notebook multi-purpose web-based che permette di attuare forme di Data-driven Analytics interattive e collaborative, Data Ingestion, Data Discovery e Data Visualization & Collaboration con SQL, Scala e altre risorse. Zeppelin è basato su un sistema di plugin e supporta Apache Spark, Python, JDBC, Markdown e Shell.

Tramite Zeppelin è possibile dunque dotarsi delle librerie opportune per poter eseguire attività di Deep Learning, eventualmente anche in forma di calcolo distribuito come in Figura 21.

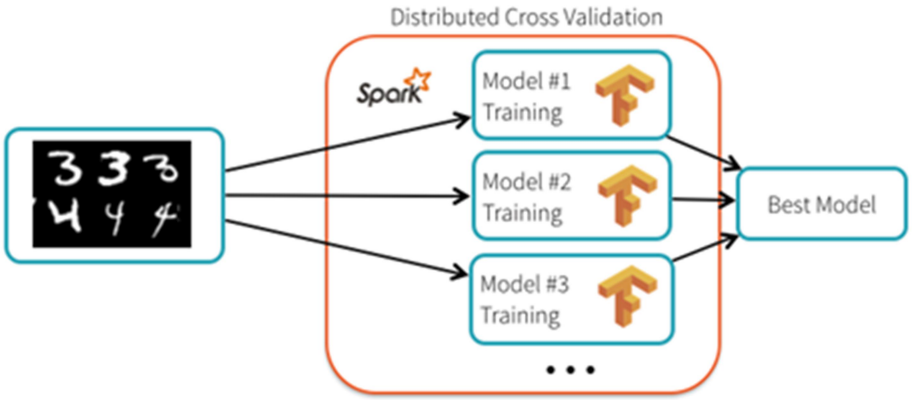


Figura 21: Calcolo distribuito con Spark e TensorFlow

Le librerie Python di riferimento sono appunto *tensorflow*, *tensorflow-gpu*, *pyspark*, installabili tramite gestori di pacchetti Python come *pip* o *conda*, inclusi nell'ambiente notebook di Zeppelin, dopo aver opportunamente installato anche le relative dipendenze di librerie di ogni pacchetto. Tale installazione potrebbe essere semplificata dall'esistenza di un Docker container che contenga già tutte le librerie in questione: in particolare, la configurazione della versione di TensorFlow con calcolo su GPU è spesso

legata anche alla configurazione di librerie CUDA di Nvidia; pertanto una versione dockerizzata di tutto l'ambiente consentirebbe di alleggerire l'integrazione di TensorFlow nella piattaforma Digital Future.

La combinazione di Apache Spark con TensorFlow ha degli enormi benefici nella gestione delle pipeline di deep learning e in particolare nel tuning degli iperparametri e nello scalare velocemente il deploy dei modelli addestrati. La distribuzione del miglior set di iperparametri nell'addestramento consente di ridurre di circa 10 volte il tempo di addestramento e di ridurre di circa il 30% il tasso di errore [29].

5.3.4.2 *Soluzioni raggiungibili tramite Deep Learning*

Nonostante le problematiche che abbiamo illustrato, i sistemi di Deep Learning hanno compiuto enormi passi evolutivi e sono migliorati moltissimo negli ultimi cinque anni, soprattutto per la grandissima quantità di dati a disposizione ma soprattutto per la disponibilità di infrastrutture ultra-performanti (CPU e GPU in particolare) [30].

Nell'ambito della ricerca sull'AI, l'apprendimento automatico ha riscosso un notevole successo negli ultimi anni, consentendo ai computer di superare o avvicinarsi alle prestazioni umane corrispondenti in aree che vanno dal riconoscimento facciale al riconoscimento vocale e linguistico (cfr. TR 1.1 – Stato dell'Arte e della Pratica, Paragrafo 3.5). Il Deep Learning invece consente ai computer di fare un ulteriore passo in avanti, in particolare di risolvere una serie di problemi complessi. Già oggi ci sono casi d'uso ed ambiti di applicazione che possiamo notare anche come "comuni cittadini" non esperti di tecnologia. Dalla computer vision per le auto senza conducente, fino ai droni e robot impiegati per la consegna di pacchi o anche per l'assistenza in casi di emergenza (per esempio per la consegna di cibo o sangue per trasfusioni in zone terremotate, alluvionate o in zone che devono affrontare crisi epidemiologiche, ecc.); riconoscimento e sintesi vocale e linguistica per chatbot e robot di servizio; riconoscimento facciale per sorveglianza; riconoscimento immagini per aiutare i radiologi a individuare i tumori nei raggi X, oppure per aiutare i ricercatori a individuare le sequenze genetiche correlate alle malattie e identificare le molecole che potrebbero portare a farmaci più efficaci o

addirittura personalizzati; sistemi di analisi per la manutenzione predittiva su una infrastruttura o un impianto analizzando i dati dei sensori dell'IoT.

Guardando invece ai tipi di applicazione (intesi come compiti che una macchina può svolgere grazie al Deep Learning), quelli di seguito sono quelli ad oggi più maturi:

1. *colorazione automatica delle immagini*: per la rete neurale significa riconoscere bordi, sfondi, dettagli e conoscere i colori tipici di una farfalla, per esempio, sapendo esattamente dove collocare il colore corretto;
2. *aggiunta automatica di suoni a filmati silenziosi*: per il sistema di Deep Learning significa sintetizzare suoni e collocarli correttamente all'interno di una situazione particolare riconoscendo immagini ed azioni, per esempio inserendo il suono del martello pneumatico, della rottura dell'asfalto e i sottofondi di una strada cittadina trafficata in un video in cui si vedono dei lavoratori che stanno rompendo l'asfalto con il martello pneumatico;
3. *traduzione simultanea*: per il sistema di Deep Learning significa ascoltare e riconoscere il linguaggio naturale, riconoscere la lingua parlata e tradurre il significato in un'altra lingua;
4. *classificazione degli oggetti all'interno di una fotografia*: il sistema in questo caso è in grado di riconoscere e classificare tutto ciò che vede in un'immagine, anche molto complessa dove per esempio c'è un paesaggio di sfondo, per esempio delle montagne, persone che camminano lungo un sentiero, degli animali al pascolo, ecc.;
5. *generazione automatica della grafia*: ci sono già oggi sistemi di Deep Learning capaci di utilizzare la grafia umana per scrivere addirittura apprendendo gli stimi della scrittura a mano degli esseri umani ed imitandola;
6. *generazione automatica di testo*: sono sistemi che hanno imparato a scrivere correttamente in una determinata lingua rispettando ortografia, punteggiatura, grammatica e persino imparando ad usare stili di scrittura differenti a seconda dell'output da produrre, per esempio un articolo giornalistico o una novella;
7. *generazione automatica di didascalie*: in questo caso il riconoscimelo delle immagini, l'analisi del contesto e la capacità di scrittura consentono ad un

sistema di scrivere in automatico le didascalie di una immagine descrivendone perfettamente la scena;

8. *gioco automatico*: un sistema in grado di giocare a giochi complessissimi, come il Go, riuscendo anche a battere il campione mondiale, umano, grazie ad AlphaGo, il sistema deep learning di Google DeepMind.

5.3.4.3 *Approcci Deep Learning per l'analisi dei dati*

Gli algoritmi di Deep Learning consistono in un insieme molto diversificato di modelli rispetto a un singolo algoritmo di apprendimento automatico tradizionale. Ciò è dovuto alla flessibilità offerta da una rete neurale nella fase di costruzione di un modello *end-to-end*, cioè il cui apprendimento ottimizza i pesi tra nodi della rete tenendo in considerazione sia gli input che gli output desiderati.

Una rete neurale può essere paragonata a dei blocchi lego, in cui è possibile costruire quasi tutte le strutture semplici e complesse che l'immaginazione aiuta a costruire. Possiamo quindi definire un'architettura avanzata come una che ha una ottenuto comprovati risultati di performance in uno o più settori di applicazione.

Infatti, gli attuali algoritmi di deep learning trascendono architetturalmente dalle classiche Artificial Neural Network (cfr. TR1.1. – Stato dell'Arte e della Pratica, Paragrafo 3.5.9), contestualmente al fatto di avere più potere computazionale, più neuroni nell'architettura delle reti, molte modalità più complesse per connettere gli strati e i neuroni tra loro nelle reti, ed infine l'enorme vantaggio di avere una estrazione automatica delle feature.

Il modo migliore per sfruttare questa potenza è abbinare i dati di input all'architettura di rete deep appropriata. In questo modo, si può applicare con successo il deep learning in modi nuovi e di interesse. Ignorando questo monito, non si aggiungerà alcun nuovo potere di modellazione oltre le tecniche di apprendimento automatico di base come la regressione logistica.

Fermi con l'obiettivo di descrivere i metodi e le tecniche per cui il Deep Learning risulti un valore aggiunto per la piattaforma Digital Future, possiamo elencare di seguito le

architetture di Deep Learning (supervised e unsupervised) più note in letteratura [31], [32]:

1. Deep Feedforward Neural Network o MultiLayer Perceptron (MLP)
2. Restricted Boltzmann Machine (RBM)
3. AutoEncoders (AE)
4. Variational AutoEncoders (VAE)
5. Deep Belief Network (DBN)
6. Deep Stacking Network (DSN)
7. Generative Adversarial Network (GAN)
8. Convolutional Neural Network (CNN)
9. Recurrent Neural Network (RNN)
10. Long-Short Term Memory (LSTM)
11. Gated Recurrent Unit (GRU)
12. Recursive Neural Network

Deep Feedforward Neural Network. Noto anche come **MultiLayer Perceptron** (MLP) [33], si tratta del tipo più comune di rete neurale nelle applicazioni pratiche. Il primo strato è l'input e l'ultimo è l'output. Questa architettura è costituita da più strati di neuroni nascosti. Ogni strato è completamente connesso al livello successivo nella rete (cfr. Figura 22). Tutti gli altri nodi mappano gli input alle uscite mediante una combinazione lineare degli input con i pesi del nodo e il bias associato, applicando una funzione di attivazione (cioè, una funzione sigmoide nel caso di apprendimento binario o una funzione softmax nel caso di apprendimento multiclasse).

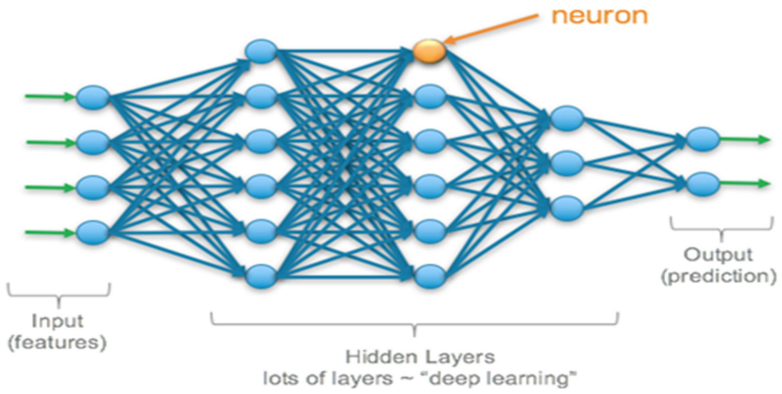


Figura 22: Topologia del MultiLayer Perceptron

Il MLP può rappresentare qualsiasi funzione ed impiega il metodo della backpropagation per l'apprendimento del modello (attraverso la tecnica della discesa del gradiente) [34], [35]. Nella sua implementazione in TensorFlow, viene data all'esperto tecnico di dominio una ampia scelta di costruzione dell'architettura neurale "deep", con una notevole flessibilità di decisione del numero di layer nascosti e del numero di neuroni per layer, oltre a poter definire il tipo di funzione di attivazione dei neuroni, il numero di epoche per l'addestramento e la funzione di loss che minimizzi l'errore tra l'output reale e quello predetto.

Il MLP può essere utilizzato in applicazioni del mondo reale come il riconoscimento di un oggetto in una immagine.

Restricted Boltzmann Machine. Una RBM è in grado di modellare le probabilità ed ha una ottima funzionalità di estrazione delle feature. Si tratta di una rete Feedforward in cui i dati vengono trasmessi in una direzione con due bias associati, invece che uno solo come nella tradizionale tecnica di backpropagation delle reti Feedforward [36]. Le RBM sono usate nel Deep Learning per eseguire feature extraction e riduzione della dimensionalità. La parte "restricted" del nome significa che le connessioni tra nodi dello stesso strato sono proibite, cioè non ci sono connessioni tra nodi visibili o tra nodi nascosti). Sono spesso usate per preaddestrare gli strati nascosti in reti più grandi come nelle Deep Belief Network. Una RBM è composta da delle unità visibili, delle unità

nascoste, pesi, unità di bias visibili ed unità di bias nascoste. Una standard RBM ha uno strato visibile ed uno nascosto, come in Figura 23.

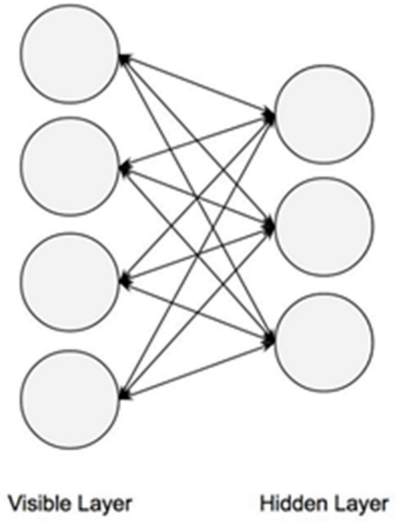


Figura 23: Topologia di una RBM

Con le RBM, ogni unità visibile è connessa ad ogni unità nascosta, ma nessuna unità dello stesso livello è connessa. Ogni livello di una RBM può essere immaginato come una fila di nodi. I nodi dei livelli visibili e nascosti sono collegati da connessioni con pesi associati, che rappresentano il modo in cui i segnali sono trasmessi, come se agissero da unità di decisione. In particolare, eseguono una decisione sull'essere accesi o spenti. "Acceso" vuol dire che i dati che passano dal nodo sono valutabili, viceversa, "spento" significa che la rete pensa che quel particolare input rappresenti rumore irrilevante. In questo modo è possibile conoscere quali feature/segnali si correlano con quali etichette per essere addestrate.

In tal senso, le RBM possono essere usate anche per fare classificazione, regressione, filtraggio collaborativo (*collaborative filtering*) e modellazione di topic (*topic modeling*).

AutoEncoder. Gli AutoEncoder sono una variante delle reti neurali Feedforward che includono un bias extra per il calcolo dell'errore di ricostruzione dell'input originale [37]. Dopo l'addestramento, gli AutoEncoder possono essere quindi utilizzati come normale rete neurale Feedforward per le attivazioni. Questa è una forma non supervisionata di

estrazione di feature perché la rete neurale utilizza solo l'input originale per i pesi di apprendimento piuttosto che il backpropagation, che ha le etichette. In genere, utilizziamo gli AutoEncoder per apprendere rappresentazioni compresse di dataset o per ridurne la dimensionalità. L'output della rete AutoEncoder è una ricostruzione dei dati di input nella forma più efficiente (la loro miglior approssimazione).

Gli AutoEncoder condividono una forte somiglianza con le reti neurali Feedforward in quanto hanno uno strato di input, strati nascosti di neuroni e d infine uno strato di output. La differenza fondamentale da notare tra un grafo di rete Feedforward e un grafo di un AutoEncoder è che lo strato di output in un AutoEncoder ha lo stesso numero di unità dello strato di input. La Figura 24 rappresenta la topologia di una rete AutoEncoder.

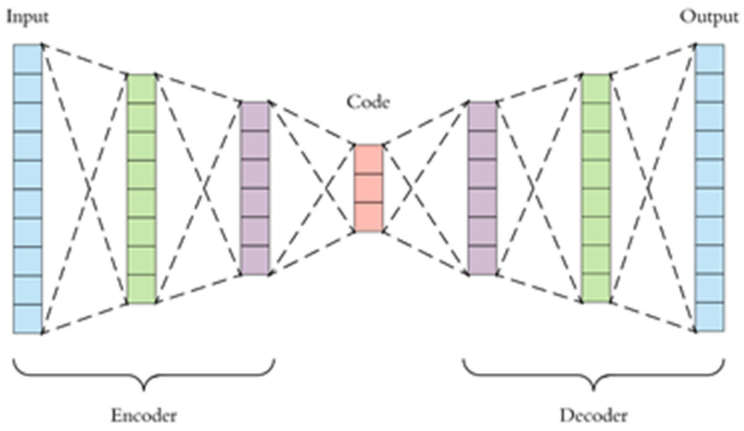


Figura 24: Topologia di un AutoEncoder

L'AutoEncoder apprende direttamente dai dati senza etichetta. Questo è collegato alla seconda grande differenza tra reti Feedforward e AutoEncoder. L'obiettivo di una rete Feedforward è generare predizioni su una classe, mentre un AutoEncoder è addestrato per riprodurre i propri dati di input.

Costruire un modello che rappresenti il dataset in input può non sembrare utile all'apparenza. In realtà, si è in genere meno interessati all'output stesso ma nella differenza tra le rappresentazioni tra input e output. Infatti, si può utilizzare la differenza

per capire quando i dati sono inusuali o anomali. Pertanto, un AutoEncoder può agire come un potente sistema per il ritrovamento di anomalie nei dati (*anomaly detection*).

Variational AutoEncoder. Una recente tipologia di AutoEncoder è il Variational AutoEncoder (VAE). Un VAE agisce similmente come un AutoEncoder, comprimendo i dati di input e rilevando il rumore (*denoising*) [38]. La differenza con l'approccio standard consiste in una diversa tecnica di attivazione dei neuroni attraverso gli strati nascosti, utilizzando un approccio probabilistico.

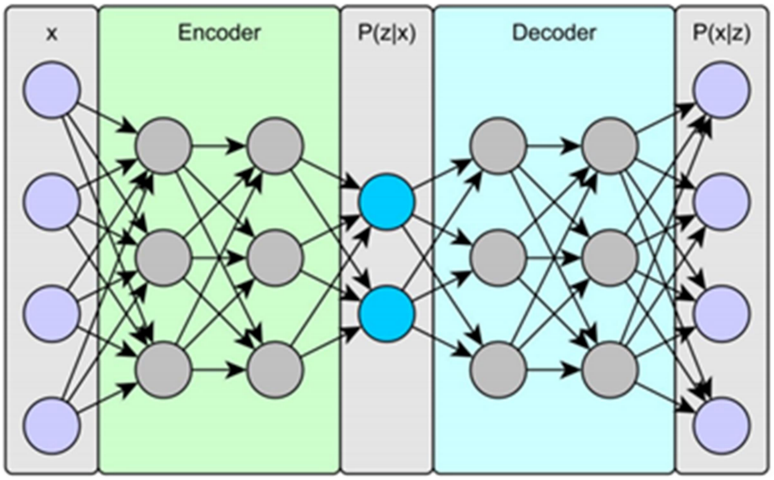


Figura 25: Topologia di un VAE

Il modello VAE assume che il generico dato di input *x* venga generato in due step: 1) viene generato un valore $z \sim P(z)$ da una distribuzione a priori; e 2) viene generata l'istanza del dato secondo una qualche distribuzione condizionale $x \sim P(x|z)$. Chiaramente, non è possibile conoscere il vero valore di *z*, dato che inferire $P(x|z)$ è un problema intrattabile, ma la rete è in grado di calcolare una buona approssimazione.

Questa architettura trae maggiore beneficio quando addestrata su serie temporali di dati.

Deep Belief Network. La DBN è una tipica architettura di rete che include però un nuovo algoritmo di addestramento. La DBN è una rete multistrato (in genere con molti strati nascosti) in cui ogni coppia di strati connessi è una Restricted Boltzmann Machines

(RBM) [39]. In questo modo, una DBN viene rappresentata come una pila di RBM (cfr. Figura 26).

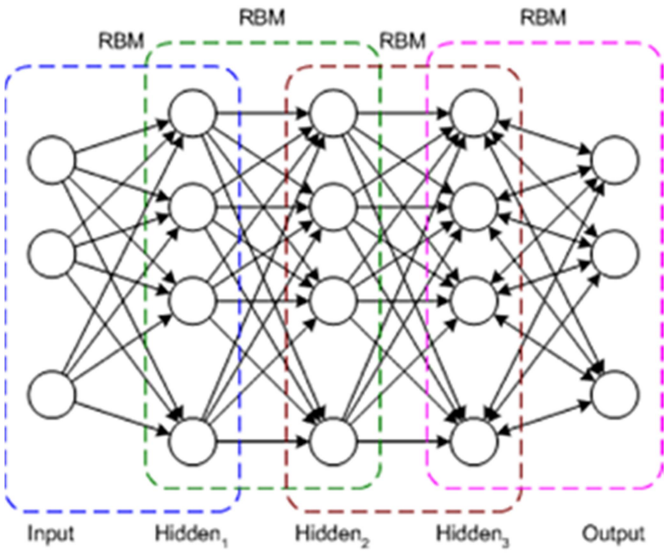


Figura 26: Topologia di una DBN

In una DBN, lo strato di input rappresenta gli input sensoriali grezzi e ogni livello nascosto apprende rappresentazioni astratte di tale input. Lo strato di output, che viene trattato in modo leggermente diverso dagli altri strati, implementa la classificazione della rete. L'addestramento avviene in due fasi: pre-addestramento non supervisionato per ogni RBM e fine-tuning supervisionato della rete complessiva Feedforward.

Nel pre-addestramento non supervisionato, ciascuna RBM viene addestrata a ricostruire il proprio input (ad esempio, la prima RBM ricostruisce lo strato di input sul primo strato nascosto). La successiva RBM viene addestrata in modo simile, ma il primo strato nascosto viene considerato come lo strato di input (o visibile) e l'RBM viene addestrata utilizzando gli output del primo strato nascosto come input. Questo processo continua fino a quando ogni strato non viene sottoposto a pre-addestramento. Quando il pre-addestramento è completato, inizia il fine-tuning. In questa fase, i nodi di output sono etichette applicate per dare loro un significato (ciò che rappresentano nel contesto della rete). L'addestramento completo sulla rete viene quindi applicato utilizzando la backpropagation per completare il processo di addestramento.

In confronto alle classiche reti neurali Feedforward, le DBN sono molto più efficaci quando applicate a problemi di classificazione con dati non etichettati.

Deep Stacking Network. L'architettura DSN, nota anche come Deep Convex Network si differenzia dai tradizionali framework di deep learning in quanto sebbene sia costituito da una rete deep Feedforward, è in realtà un insieme di singole reti deep, ciascuna con i suoi strati nascosti [40]. Questa architettura è una risposta a uno dei problemi con il deep learning: la complessità dell'addestramento. Ogni strato in un'architettura di deep learning aumenta esponenzialmente la complessità dell'addestramento, quindi le DSN considerano l'addestramento non come un singolo problema ma come una serie di problemi individuali.

Una DSN è costituita da un insieme di moduli, ognuno dei quali è una sottorete nella gerarchia complessiva della DSN. In un'istanza di questa architettura, vengono creati tre moduli. Ogni modulo è costituito da uno strato di input, un singolo strato nascosto e uno di output. I moduli sono impilati uno sull'altro, dove gli input di un modulo sono costituiti dagli output dello stato precedente e dal vettore di input originale (cfr. Figura 27). Questa stratificazione consente alla rete globale di apprendere una classificazione più complessa di quella che sarebbe possibile dare un singolo modulo.

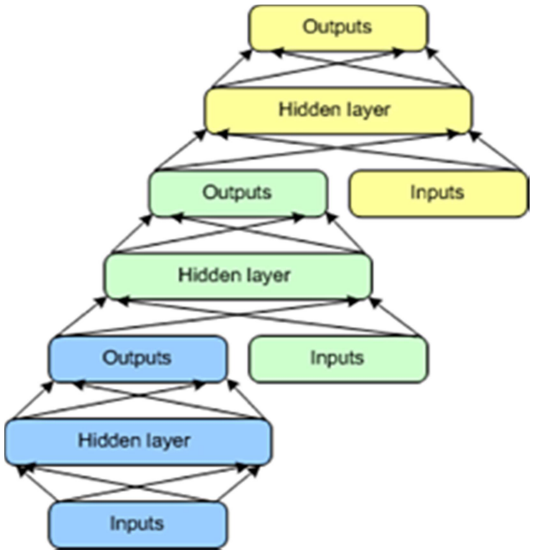


Figura 27: Topologia di una DSN

Le DSN consentono l'addestramento di singoli moduli da soli, rendendole efficienti data la capacità di addestramento in parallelo. L'apprendimento supervisionato viene implementato tramite backpropagation per ogni modulo piuttosto che come backpropagation sull'intera rete. Per molti problemi, le DSN possono ottenere prestazioni migliori rispetto alle classiche DBN, rendendole una delle architetture di rete più popolari ed efficienti.

Generative Adversarial Network. Le GAN si sono dimostrate essere molto abili nel sintetizzare nuove immagini basate su altre immagini di addestramento [41]. Si può estendere questo concetto per modellare altri domini come sequenze audio, video, o la generazione di immagini da descrizioni testuali [42].

Le GAN sono un esempio di rete che utilizza l'apprendimento non supervisionato per addestrare due modelli in parallelo. Un aspetto chiave delle GAN (e dei modelli generativi in generale) è il modo in cui usano un parametro di conteggio che è significativamente più piccolo del normale rispetto alla quantità di dati su cui si sta addestrando la rete. Tale rete è quindi forzata a rappresentare in modo efficiente i dati di addestramento, rendendo più efficace la generazione di dati simili ai dati di addestramento.

Se disponessimo di un grande corpus di immagini di addestramento (come il dataset *ImageNet* [43]), potremmo costruire una rete neurale generativa che produca immagini (al contrario delle classificazioni). Considereremmo queste immagini di output generate come "campioni" dal modello. Il modello generativo nelle GAN genera tali immagini mentre una rete secondaria "discriminante" tenta di classificare queste immagini generate. Questa rete discriminante secondaria tenta di classificare le immagini di output come reali o sintetiche. Durante l'addestramento delle GAN, si vuole aggiornare i parametri in modo che la rete generi immagini di output più credibili sulla base dei dati di addestramento. L'obiettivo qui è quello di rendere le immagini sufficientemente realistiche al punto che la rete discriminante sia ingannata al punto da non poter distinguere la differenza tra i dati di input reali e quelli sintetici, come in Figura 25.

Questo processo di apprendimento cerca di trovare il modo più efficace per rappresentare le feature nei dati, come gruppi simili di pixel, bordi e altri pattern.

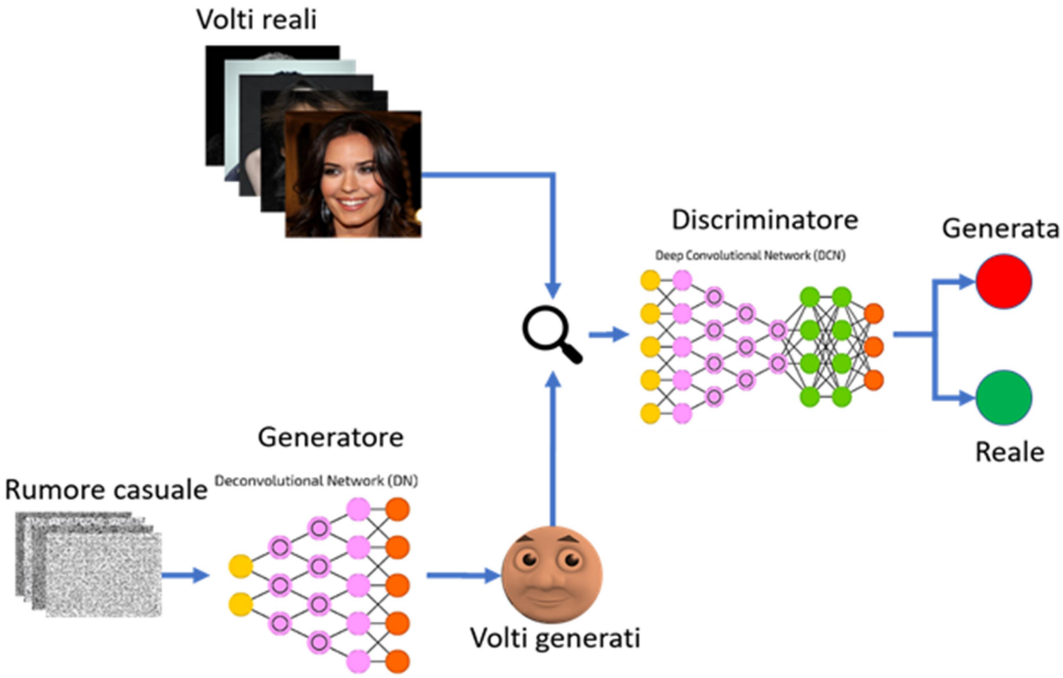


Figura 28: Topologia di una GAN

Una rappresentazione efficiente di modello GAN è garantita quando si ha a disposizione circa 100 milioni di parametri durante la modellazione di un dataset, come in ImageNet. Nel corso dell'addestramento, un dataset di input come ImageNet (200 GB) produce circa a 100 MB solo di parametri.

Quando si modellano le immagini, la rete discriminante è in genere una Convolutional Neural Network standard. Utilizzando una rete neurale secondaria, la rete discriminante consente alla GAN di addestrare entrambe le reti in parallelo in modo non supervisionato. Questa rete discriminante prende le immagini come input e produce quindi una classificazione. Il gradiente dell'output della rete discriminante indica come apportare piccole modifiche ai dati sintetici per renderle più realistiche. La rete generativa della GAN genera dati (o immagini) con un tipo speciale di strato chiamato strato "*deconvolutivo*". Durante l'addestramento, si utilizza la backpropagation su entrambe le reti per aggiornare i parametri della rete di generazione per generare

immagini di output più realistiche. L'obiettivo qui è quello di aggiornare i parametri della rete di generazione al punto in cui la rete discriminante è sufficientemente "ingannata" dalla rete di generazione perché l'output è così realistico rispetto alle immagini reali dei dati di addestramento.

Convolutional Neural Network. L’obiettivo di una CNN è di apprendere feature di ordine superiore nei dati tramite le convoluzioni [44]. Se si lavora con le immagini, le CNN mostrano grandi risultati. Si adattano bene al riconoscimento di oggetti nelle immagini e alle competizioni di classificazione delle immagini sono costantemente al top [45]. Possono identificare volti, individui, segnali stradali e molti altri aspetti dei dati visivi. Le CNN vengono usate anche nell'analisi del testo tramite riconoscimento ottico dei caratteri, e sono anche utili quando si analizzano le parole come unità testuali discrete [46]. Sono anche abbastanza performanti nell'analisi del suono [47].

L'efficacia delle CNN nel riconoscimento delle immagini è uno dei motivi principali per cui il mondo riconosce il potere del Deep Learning. Come illustrato nella Figura 26 le CNN sono in grado di costruire feature do posizione e invarianti alla rotazione dai dati di immagine grezzi.

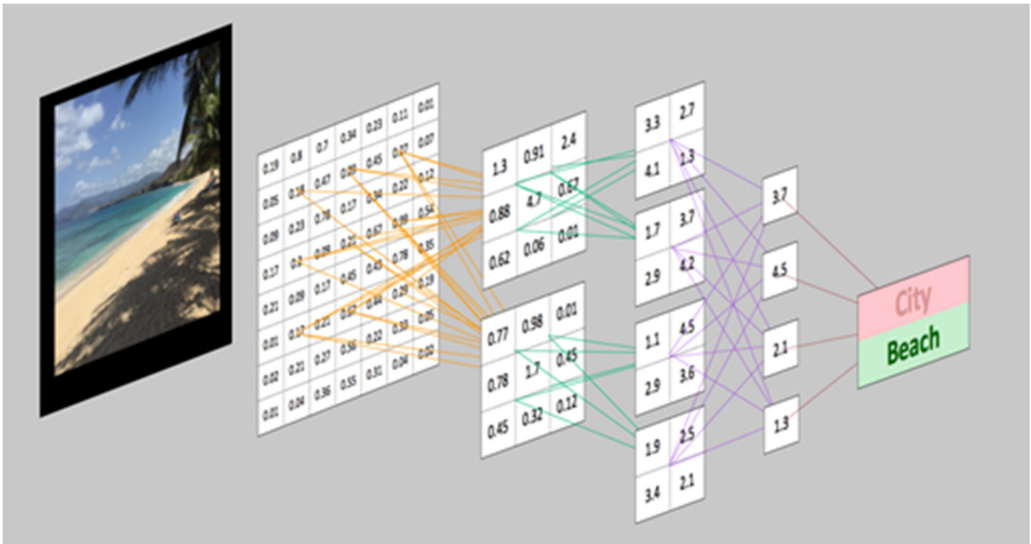


Figura 29: CNN e Computer Vision

Le CNN stanno alimentando importanti progressi nella visione artificiale, che ha applicazioni in svariati ambiti tra cui auto a guida autonoma, robotica, droni e trattamenti per ipovedenti. Le CNN sono state utilizzate anche in altri task come la traduzione/generazione in linguaggio naturale [48] e l'analisi del sentimento [49]. Una convoluzione è un concetto potente per aiutare a costruire uno spazio feature più robusto basato su un segnale.

Con le CNN, possiamo organizzare i neuroni in una struttura tridimensionale per la quale abbiamo le seguenti caratteristiche: larghezza, altezza, profondità. Questi attributi dell'input corrispondono alla struttura di una immagine per la quale abbiamo: larghezza dell'immagine in pixel, altezza dell'immagine in pixel, canali RGB come profondità. Possiamo considerare questa struttura come un volume tridimensionale di neuroni. Un aspetto significativo del modo in cui le CNN si sono evolute dalle precedenti varianti feed-forward è come hanno raggiunto l'efficienza computazionale con nuovi tipi di layer.

Le CNN trasformano i dati di input dallo strato di input attraverso tratti completamente collegati in un insieme di punteggi di classe forniti dal livello di output. Esistono molte varianti dell'architettura della CNN, ma sono basate sul pattern degli strati, come illustrato nella Figura 30.

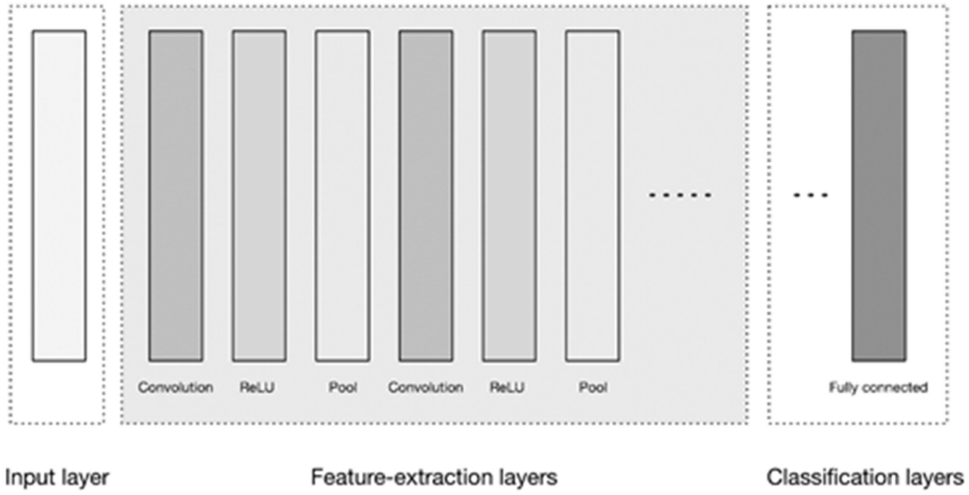


Figura 30: Architettura CNN ad alto livello

Come si può notare, la figura di cui sopra rappresenta tre gruppi principali: lo strato di input, gli strati di estrazione delle feature di apprendimento, gli strati di classificazione. Lo strato di input accetta dati di input tridimensionali generalmente nella forma spazialmente delle dimensioni (larghezza x altezza) dell'immagine e ha una profondità che rappresenta i canali di colore RGB. Gli strati di estrazione delle feature hanno uno schema generale ripetuto della sequenza: strato convolutivo e strato di raggruppamento (chiamato *pooling*). Questi strati trovano un numero di feature nelle immagini e costruiscono progressivamente feature di ordine superiore. Ciò corrisponde direttamente alla tematica di apprendimento gerarchico a vari livelli di astrazione del Deep Learning, grazie al quale le feature vengono apprese automaticamente rispetto a quelle tradizionalmente progettate a mano. Infine, abbiamo gli strati di classificazione in cui abbiamo uno o più strati completamente connessi per prendere le feature di ordine superiore e produrre probabilità o punteggi di classe. L'output di questi strati produce in genere un output bidimensionale delle dimensioni [*b x N*], dove b è il numero di esempi nel mini-batch e *N* è il numero di classi a cui siamo interessati.

I livelli convolutivi sono considerati i mattoni fondamentali delle architetture CNN. Come illustrato nella Figura 28, gli strati convolutivi trasformano i dati di input usando una patch di neuroni di connessione locale del livello precedente. Lo strato calcolerà un prodotto punto tra la regione dei neuroni nel livello di input e i pesi a cui sono localmente collegati nello strato di output.

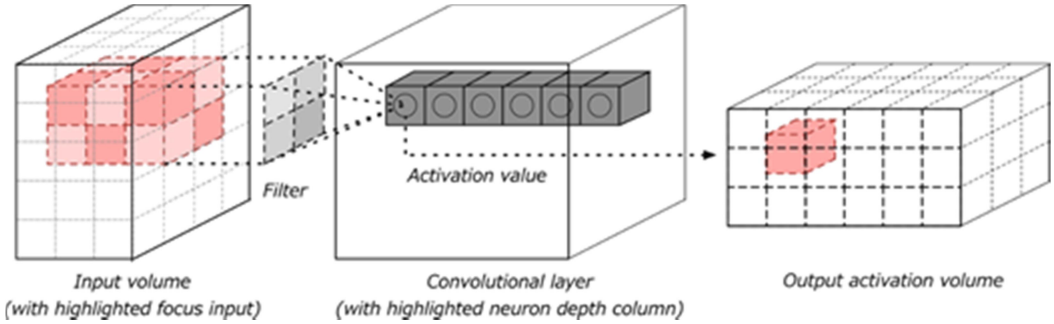


Figura 31: Strato convolutivo con volumi di input e output

L'output risultante ha generalmente le stesse dimensioni spaziali (o dimensioni spaziali più piccole) ma a volte aumenta il numero di elementi nella terza dimensione dell'output (dimensione di profondità). Il concetto chiave in questi strati è chiamato convoluzione. Una convoluzione è definita come un'operazione matematica che descrive una regola su come unire due serie di informazioni. È importante sia in fisica che in matematica e definisce un ponte tra il dominio spazio/tempo e il dominio della frequenza attraverso l'uso delle trasformate di Fourier. Prende l'input, applica un kernel di convoluzione e ci fornisce una mappa delle feature come output.

L'operazione di convoluzione, mostrata nella Figura 32, è nota come *feature detector* (rilevatore di feature) di una CNN. L'input di una convoluzione può essere un dato grezzo o una mappa di feature emessa da un'altra convoluzione. Viene spesso interpretato come un filtro in cui il kernel filtra i dati di input per determinati tipi di informazioni; per esempio, un kernel di bordo lascia passare solo le informazioni dal bordo di un'immagine.

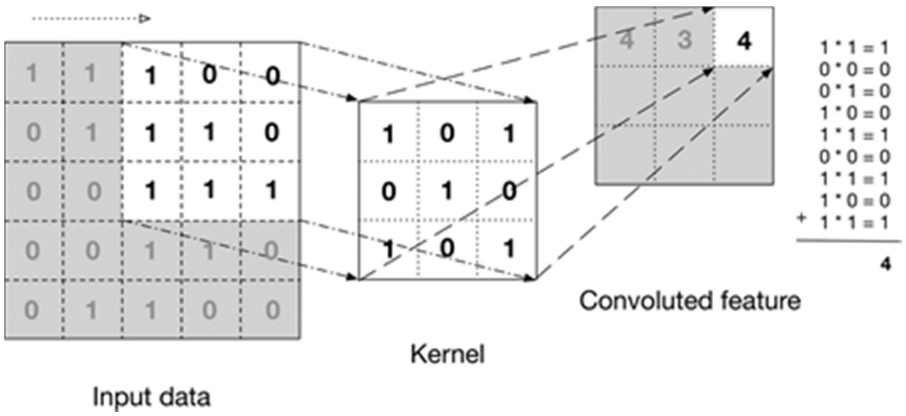


Figura 32: Operazione di Convoluzione

La figura illustra come il kernel viene fatto scorrere tra i dati di input per produrre le feature convolutive (output). Ad ogni passo, il kernel viene moltiplicato per i valori dei dati di input entro i suoi limiti, creando una singola entry nella mappa delle feature di output. In pratica, l'output è grande se la feature che stiamo cercando viene rilevata nell'input. L'insieme di pesi in uno strato convolutivo viene comunemente riferito come filtro (o kernel). Questo filtro è convogliato con l'input e il risultato è una mappa delle

feature (o mappa di attivazione). Gli strati convolutivi eseguono trasformazioni sul volume dei dati di input che sono una funzione delle attivazioni nel volume di input e dei parametri (pesi e bias dei neuroni). La mappa di attivazione per ciascun filtro è impilata lungo la dimensione della profondità per costruire il volume di output 3D, come in Figura 33.

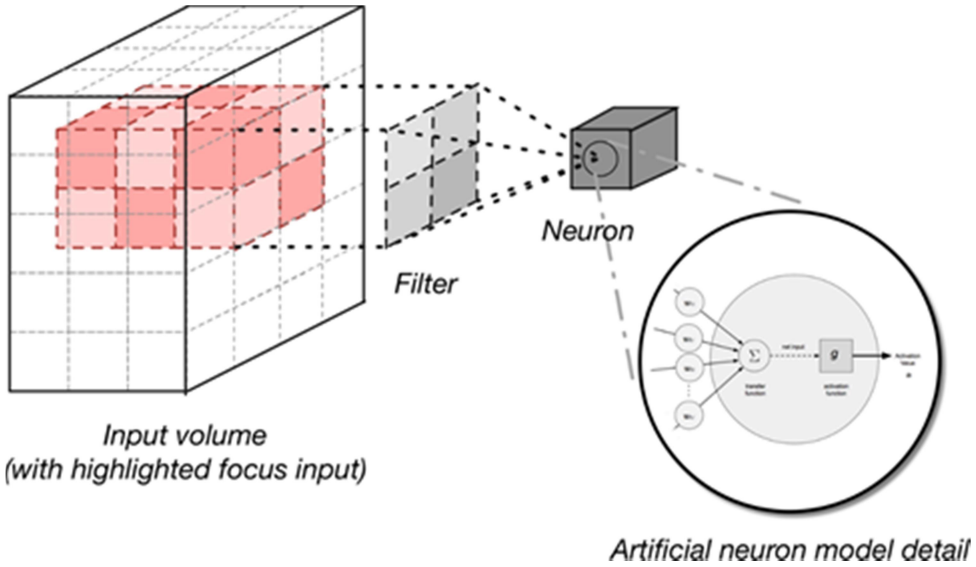


Figura 33: Generazione di un volume di output di attivazione

Gli strati convolutivi hanno parametri per ogni strato e iperparametri aggiuntivi. La discesa del gradiente viene utilizzata per addestrare i parametri in ogni strato in modo tale che i punteggi delle classi siano coerenti con le etichette nel training set.

Le CNN usano uno schema di condivisione dei parametri per controllare il conteggio totale dei parametri. Questo aiuta nel migliorare i tempi di addestramento perché si usano meno risorse per apprendere il set di dati di addestramento. Per implementare la condivisione dei parametri nelle CNN, prima si denota una singola fetta bidimensionale di profondità come una "slice di profondità". Quindi si vincolano i neuroni in ciascuna slice di profondità per utilizzare gli stessi pesi e bias. Questo ci dà un numero significativamente inferiore di parametri (o pesi) per un dato strato convolutivo.

Gli strati di raggruppamento sono comunemente inseriti tra strati convolutivi successivi. Si fa seguire gli strati convolutivi da strati di raggruppamento per ridurre progressivamente le dimensioni spaziali (larghezza e altezza) della rappresentazione dei dati. Gli strati di pooling riducono progressivamente la rappresentazione dei dati sulla rete e aiutano a controllare l'overfitting. Lo strato di pool opera indipendentemente su ogni slice di profondità dell'input.

Recurrent Neural Network. Le reti neurali ricorrenti (RNN) fanno parte della famiglia delle reti neurali feed-forward. Sono diverse dalle altre reti feed-forward nella loro capacità di inviare informazioni in fasi temporali [50]. Le RNN prendono ciascun vettore da una sequenza di vettori di input e li modellano uno alla volta. Ciò consente alla rete di mantenere lo stato durante la modellazione di ciascun vettore di input attraverso la finestra dei vettori di input. Le RNN sono adatte per funzioni di modellazione per le quali l'input e/o l'output sono composti da vettori che implicano una dipendenza temporale tra i valori. Le RNN modellano l'aspetto temporale dei dati creando cicli nella rete (da qui la parte "ricorrente" del nome). I dati in questi domini sono intrinsecamente ordinati e sensibili al contesto, dove i valori successivi dipendono da quelli precedenti. Troviamo dati sequenziali in molti domini di settore per i quali un modello neurale ha bisogno di generare una sequenza di vettori: sottotitoli di immagini [51], sintesi vocale [52], generazione di musica [53], riproduzione di videogiochi, modellazione linguistica, modelli di generazione di testo a livello di carattere. In altri domini, abbiamo bisogno di una sequenza di vettori di input: previsione delle serie temporali, analisi video, recupero di informazioni musicali. E poi abbiamo domini che richiedono sia una serie di vettori di input che di output: traduzione in linguaggio naturale, ingaggiamento nel dialogo, controllo robotico. Le reti neurali ricorrenti sono in contrasto con altre reti deep nel tipo di input che possono modellare (input non fisso): passaggi di calcolo non fissi, dimensioni di output non fisse Possono operare su sequenze di vettori, come i frame del video. Un aspetto importante delle RNN è il modo in cui possiamo lavorare con input e output in modi unici.

Una RNN include un ciclo di feedback che utilizza per imparare da sequenze, comprese sequenze di lunghezze variabili. Le RNN contengono una matrice di parametri aggiuntiva per le connessioni tra i passaggi temporali, che vengono utilizzati e addestrati per acquisire le relazioni temporali nei dati. Le RNN sono addestrate a generare sequenze, in cui l'output di ciascun passo temporale è basato sia sull'ingresso corrente che sull'ingresso in tutte le fasi temporali precedenti. Le normali reti neurali ricorrenti calcolano un gradiente con un algoritmo chiamato *backpropagation through time* (BPTT).

L'apprendimento automatico tradizionale funziona sul concetto di un singolo vettore di input di dimensioni fisse. Nelle attività di modellazione tradizionali, in genere vedremmo una relazione input-output di dimensioni di input fisse a dimensioni di output fisse. Questo è comunemente il pattern per la modellazione nella costruzione di classificatori. Le RNN invece modificano questa dinamica di input per includere più vettori di input, uno per ciascun passo temporale e ciascun vettore può avere molte colonne. La Figura 30 fornisce esempi di come le RNN operano su sequenze di vettori di input e output:

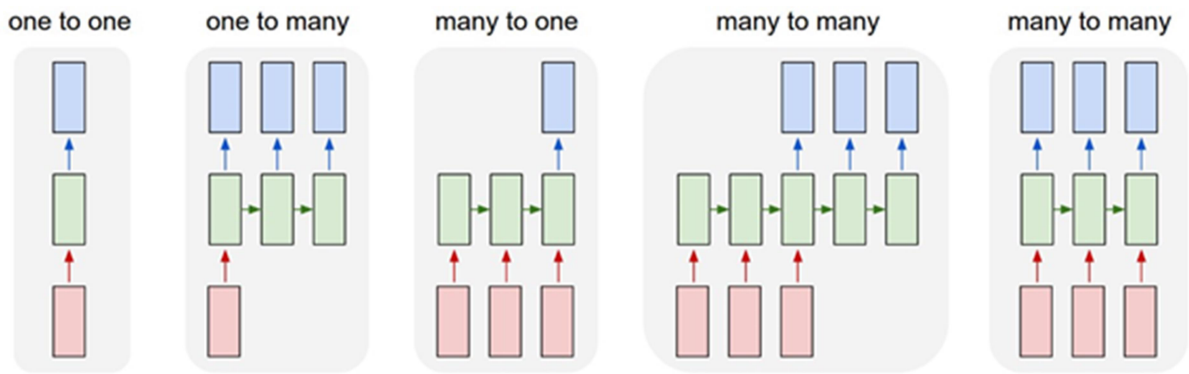


Figura 34: Topologie di RNN

- *One-to-many*: output di sequenza. Ad esempio, il sottotitolaggio delle immagini prende un'immagine e genera una sequenza di parole.
- *Many-to-one*: input di sequenza. Ad esempio, si consideri l'analisi del sentimento in cui viene inserita una determinata frase.

- *Many-to-many*: Ad esempio, classificazione video: task di etichettatura di ogni fotogramma.

Ad ogni passo temporale dell'invio dell'input attraverso una rete ricorrente, i nodi che ricevono l'input lungo i bordi ricorrenti ricevono le attivazioni di input dal vettore di input corrente e dai nodi nascosti nello stato precedente della rete. L'output è calcolato dallo stato nascosto al passo temporale specificato. Il precedente vettore di input nella fase temporale precedente può influenzare l'uscita corrente al passo temporale corrente attraverso le connessioni ricorrenti. Possiamo raggruppare strati di questi neuroni ricorrenti specializzati per costruire modelli migliori.

Long-Short Term Memory. Le reti LSTM sono la variante più comunemente utilizzata delle reti neurali ricorrenti [54]. Il componente critico dell'LSTM è la cella di memoria e i gate (inclusi il gate di dimenticanza e il gate di input). I contenuti della cella di memoria sono modulati dai gate d'ingresso e dai gate dimenticati. Supponendo che entrambe questi gate siano chiusi, il contenuto della cella di memoria rimarrà non modificato tra una fase temporale e la successiva. La struttura di gating consente di conservare le informazioni su più passaggi temporali e, di conseguenza, consente anche ai gradienti di scorrere su più passaggi temporali. Ciò consente al modello LSTM di superare il problema del gradiente di sparizione (vanishing gradient) che si verifica con la maggior parte dei modelli di reti neurali ricorrenti.

Le LSTM sono note per i seguenti casi d'uso: generazione di frasi (ad es. modelli linguistici a livello di carattere), classificazione di serie temporali, riconoscimento vocale, riconoscimento della scrittura a mano libera, modellazione polifonica musicale. Le architetture LSTM hanno mostrato negli ultimi anni benchmark leader del settore su attività come la generazione di didascalie delle immagini, traduzione della lingua, riconoscimento della grafia. Le reti LSTM consistono in molte celle connesse e si comportano bene in quanto sono efficienti durante l'apprendimento.

Per comprendere meglio la complessa disposizione delle connessioni tra unità e livelli nelle reti LSTM, si riconsideri la Figura 3 che rappresenta la topologia di una rete feed-

forward. Se dovessimo disegnare tale rete in una versione "appiattita", avrebbe la rappresentazione come in Figura 35.

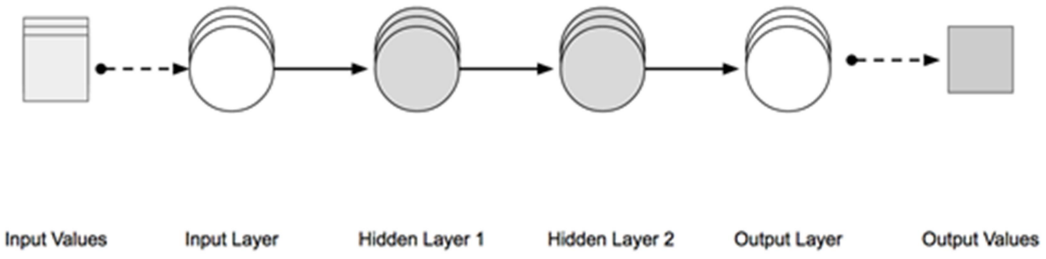


Figura 35: Rete Feed-Forward multistrato "appiattita"

Con le reti neurali ricorrenti, introduciamo l'idea di un tipo di connessione che collega l'uscita di un neurone del livello nascosto come input allo stesso neurone del livello nascosto. Con questa connessione ricorrente, possiamo prendere input dalla fase temporale precedente nel neurone come parte delle informazioni in arrivo, come rappresentato in Figura 36.

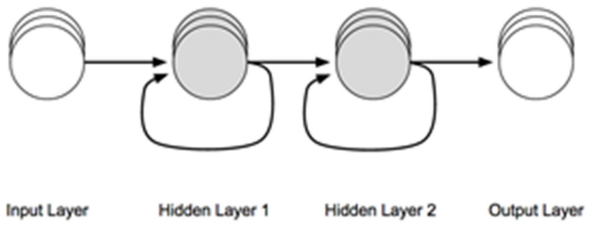


Figura 36: Connessioni ricorrenti su nodi degli strati nascosti

La Figura 37 illustra come possiamo ulteriormente comprendere visivamente questo "srotolamento" della rete in Figura 36 per mostrare come le informazioni fluiscono attraverso la rete in modo feed-forward e nel tempo.

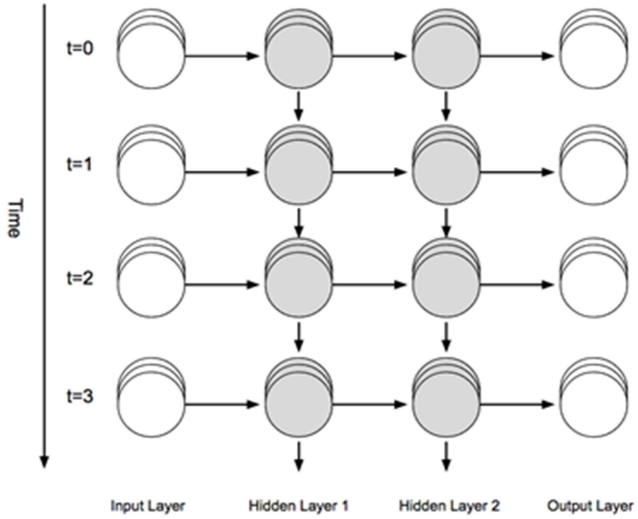


Figura 37: RNN srotolata lungo l'asse del tempo

Ogni unità LSTM ha due tipi di connessioni: collegamenti dal passo temporale precedente (output di tali unità), e collegamenti dal livello precedente. La cella di memoria in una rete LSTM è il concetto centrale che consente alla rete di mantenere lo stato nel tempo. Il corpo principale dell'unità LSTM è denominato *blocco LSTM*, come mostrato nella Figura 38.

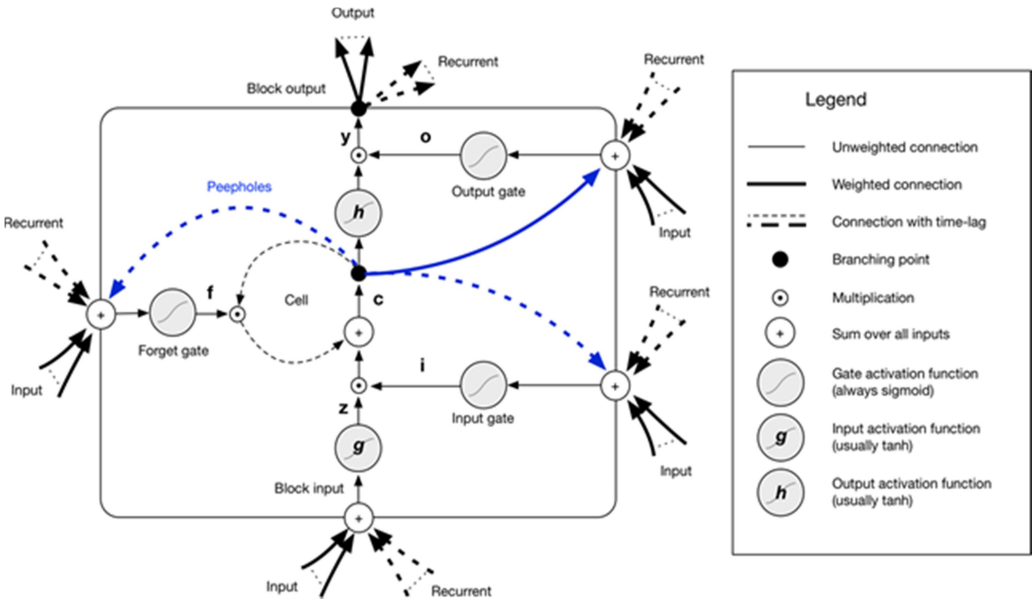


Figura 38: Diagramma del blocco LSTM

Ci sono tre unità gate, che imparano a proteggere l'unità lineare dai segnali fuorvianti:

- Il gate di input protegge l'unità da eventi di input irrilevanti.
- Il gate di dimenticanza aiuta l'unità a dimenticare il contenuto della memoria precedente.
- Il gate di output espone il contenuto della cella di memoria (o meno) all'uscita dell'unità LSTM.

L'output del blocco LSTM è ricollegato ciclicamente al blocco di input e a tutte i gate del blocco LSTM.

Le reti LSTM utilizzano l'apprendimento supervisionato per aggiornare i pesi nella rete. Si addestrano su un vettore di input alla volta in una sequenza di vettori. I vettori sono a valori reali e diventano sequenze di attivazioni dei nodi di input. Ogni unità non-input calcola l'attivazione corrente ad ogni time-step. Questo valore di attivazione è calcolato come la funzione non lineare della somma pesata delle attivazioni di tutte le unità da cui riceve le connessioni. Per ciascun vettore di input nella sequenza di input, l'errore è uguale alla somma delle deviazioni di tutti i segnali target dalle attivazioni corrispondenti calcolate dalla rete.

Gated Recurrent Unit. Un'altra unità ricorrente simile a LSTM è la Gated Recurrent Unit (GRU) [55]. La GRU ha un gate di reset e un gate di aggiornamento (cfr. Figura 39), simili ai gate di input/dimenticanza nell'unità LSTM. La differenza principale è che la GRU espone completamente il suo contenuto di memoria usando solo una integrazione "leaky", ma con una costante di tempo adattativa controllata dal gate di aggiornamento. La GRU è ispirata dall'unità LSTM ma è considerata più semplice da calcolare e implementare.

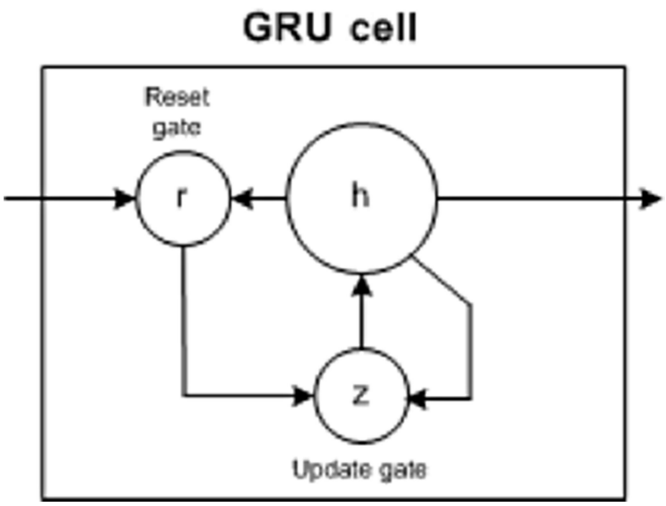


Figura 39: Cella GRU

La GRU è più semplice dell'LSTM, può essere addestrata più rapidamente e può essere più efficiente nella sua esecuzione. Tuttavia, l'LSTM può essere più espressiva e con più dati può portare a risultati migliori.

Recursive Neural Network. Le reti neurali ricorsive, come le reti neurali ricorrenti, possono gestire input di lunghezza variabile [56]. La differenza principale è che le reti neurali ricorrenti hanno la capacità di modellare le strutture gerarchiche nel dataset di addestramento. Le immagini hanno comunemente una scena composta da molti oggetti. Decostruire le scene è spesso un problema di interesse non banale. La natura ricorsiva di questa decostruzione consente non solo di identificare gli oggetti nella scena, ma anche come tali oggetti si riferiscono a formare la scena.

Un'architettura di rete neurale ricorsiva è composta da una matrice a peso condiviso e una struttura ad albero binario che consente alla rete ricorsiva di apprendere sequenze variabili di parole o parti di un'immagine. È utile come parser di frasi e scene. Le reti neurali ricorsive usano una variazione della backpropagation chiamata *backpropagation through structure* (BPTS). Il passaggio in feed-forward avviene dal basso verso l'alto e la backpropagation in maniera top-down. Si pensi all'obiettivo come al vertice dell'albero, mentre gli input sono in basso, come si nota in Figura 40.

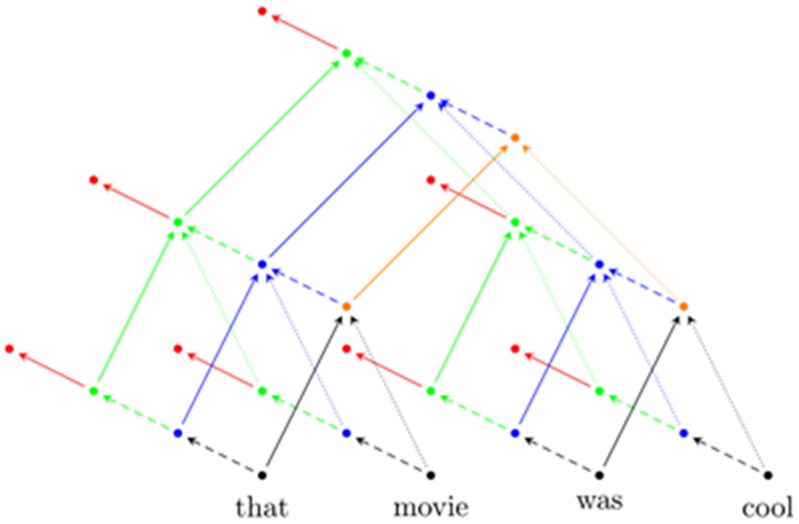


Figura 40: Esempio di rete neurale ricorsiva

Entrambe le reti neurali ricorsive e ricorrenti condividono molti degli stessi casi d'uso. Le reti neurali ricorrenti sono tradizionalmente utilizzate in NLP per via dei loro legami con alberi binari, contesti e parser basati sul linguaggio naturale. Ad esempio, i parser sono in grado di suddividere una frase in un albero binario, segmentandolo per le proprietà linguistiche della frase. Nel caso delle reti neurali ricorsive, è un vincolo che usiamo un parser che costruisce la struttura ad albero.

Le reti neurali ricorsive possono recuperare sia la struttura granulare che la struttura gerarchica di livello superiore in dataset di immagini o frasi. Le applicazioni per reti neurali ricorsive includono: scomposizione della scena di una immagine, trascrizione da audio a testo. Le reti neurali ricorrenti tendono ad essere più veloci da addestrare, quindi si usano tipicamente in più applicazioni temporali, ma si dimostrano essere performanti anche in domini basati su NLP come l'analisi dei sentimenti.

In definitiva, dopo questa carrellata sugli approcci Deep Learning per l'analisi dei dati, riepiloghiamo sinteticamente e raggruppiamo queste architetture per:

- ✓ generare dati (ad es. Immagini, audio o testo), utilizzeremmo:
  - GAN

- VAE
- Recurrent Neural Network

✓ modellare le immagini, probabilmente utilizzeremmo:

- CNN
- DBN

✓ modellare dati sequenziali, probabilmente utilizzeremmo:

- Recurrent Neural Network
- LSTM
- GRU

5.4 Processing Engine

Come intuibile dal nome dei due engine, la piattaforma DF dovrà essere in grado di poter operare sui dati sia in modalità real-time, processando i dati entranti nella piattaforma, che in modalità batch, sui dati memorizzati sul database o su altri supporti, come file CSV. Entrambi i modi di processing, sia online che in batch, dovranno essere eseguibili in ambiente di calcolo distribuito in maniera separata e indipendente con il supporto delle componenti di *Stream Processing Engine* e *Batch Processing Engine* che hanno come funzione base il data processing su larga scala.

5.4.1 *Batch Processing Engine*

Il **Batch Processing Engine** permetterà l'elaborazione in batch, ossia l'elaborazione di blocchi di dati che sono già stati memorizzati per un periodo di tempo. Questi dati contengono milioni di record per un giorno che possono essere archiviati come file o record, ecc. Questo particolare file verrà sottoposto a elaborazione alla fine della giornata per varie analisi che l'impresa vuole fare. Ovviamente ci vorrà molto tempo per l'elaborazione di quel file. Il Batch Processing sarà applicato in situazioni in cui non sono necessari risultati di analisi in tempo reale e, in particolare, quando si renderà necessario elaborare grandi volumi di dati o per ottenere elaborazioni di informazioni in maniera più dettagliate o ancora elaborazioni su finestre temporali prefissate. Dal punto di vista

prestazionale la latenza dell'elaborazione batch sarà compresa tra un minuto e qualche ora.

5.4.2 *Stream Processing Engine*

Lo **Stream Processing Engine** permetterà l'elaborazione dei dati in tempo reale, all'arrivo degli stessi dai sensori e rileverà rapidamente le condizioni entro un breve periodo di tempo dal momento della ricezione dei dati. L'elaborazione stream consentirà di alimentare gli strumenti di analisi messi a disposizione dal Layer e di ottenere di conseguenza dei risultati analitici immediati. La latenza dell'elaborazione del flusso sarà espressa in secondi o millisecondi quindi bassa.

Nello specifico questo processo, come mostrato in figura, verrà avviato continuamente per elaborare stream di dati attraverso una serie di operazioni. Riguarderà principalmente l'elaborazione dei dati "in motion", derivanti da dispositivi e da rilevazione real-time provenienti anche dagli applicativi dei pillar. Uno degli aspetti più importanti di tale fase riguarda la possibilità di processare i dati in parallelo senza problemi di dipendenza dei dati o sincronizzazione.

Lo stream processing avrà inizio alla ricezione di un evento proveniente dall'Event Bus. Esso potrebbe interrogare la componente di Machine Learning Libraries per apprendere in maniera automatica alcuni "pattern" che caratterizzano l'insieme dei dati analizzati e ottenere previsioni sul nuovo stream dati; successivamente interrogherà l'Event Layer. Quest'ultimo, rappresentando il Complex Event Processing (CEP) della Piattaforma Digital Future si occuperà di correlare eventi indipendenti attenuti in ingresso al fine di generare un Complex Event in uscita. Ossia permetterà di implementare logiche di business e regole legati al dominio applicativo, di arricchire lo stream dati in input con nuovi attributi ritenuti utili oppure e/o ripulire da informazioni ridondanti in modo tale da creare l'output finale, un nuovo event stream, da rinviare all' Event Bus.

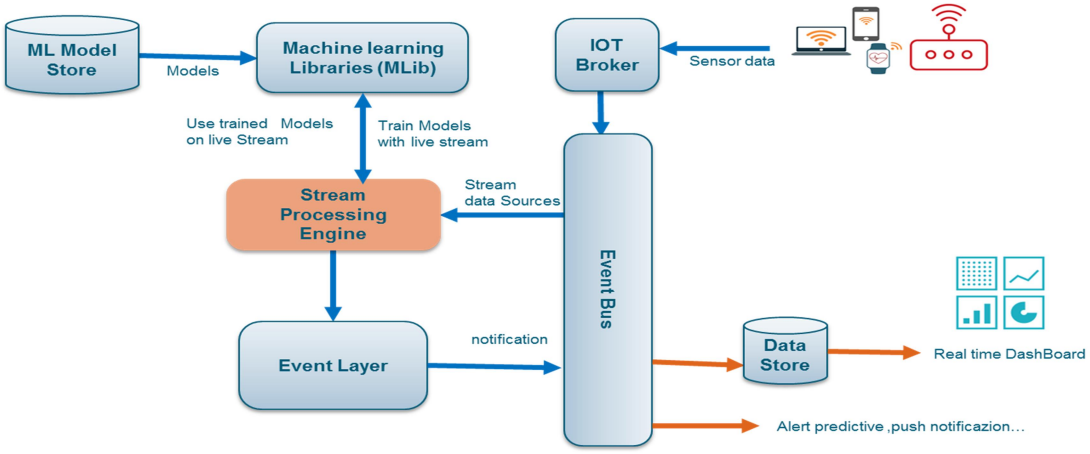


Figura 41: Stream Processing Pipeline

5.5 Explanation Tool

A causa del loro enorme potenziale in attività predittive, le tecniche di apprendimento automatico come le reti neurali artificiali hanno ricevuto grande attenzione sia dalla ricerca che dall'industria. Tuttavia, spesso questi modelli non forniscono risultati interpretabili che sono un requisito cruciale in molti settori ad alto rischio come l'assistenza sanitaria o i trasporti. Questa situazione è problematica perché può influire negativamente sulla comprensione, la fiducia e la gestione degli algoritmi di Machine Learning (ML) **[171]**. Sebbene non tutte le decisioni algoritmiche debbano essere spiegate in dettaglio, la *explanation* (cfr. spiegabilità) è necessaria quando si affrontano dichiarazioni di problemi incompleti, compresi aspetti di sicurezza, etica o trade-off **[172]**. Inoltre, le considerazioni legali sulla responsabilità dell'IA si aggiungono alla rilevanza dei sistemi spiegabili di supporto alle decisioni **[173]**.

Il termine generico EXplainable Artificial Intelligence (XAI) è spesso usato in ambito accademico per indicare una varietà di approcci che tentano di rendere i metodi ML spiegabili, trasparenti, interpretabili o comprensibili. Tuttavia, molti di questi approcci si basano su un'analisi puramente tecnica dei modelli ML black-box. Per tali approcci, in **[174]** sostengono che la explanation del modello non può essere raggiunta. Gli autori stabiliscono inoltre che la explanation dipende fortemente dall'uso della conoscenza del dominio e non dalla sola analisi dei dati.

L'incorporazione delle Semantic Web Technologies potrebbe essere la chiave per ottenere sistemi di IA veramente spiegabili **[175, 176]**. Infatti, le Semantic Web Technologies offrono strumenti semanticamente interpretabili che consentono il ragionamento su basi di conoscenza. Da un primo punto di vista, più generale, possiamo associare le Semantic Web Technologies a due tipi di modelli ML spiegabili: i task di classificazione supervisionata usando reti neurali e i task di *embedding* non-supervisionato. Combinando questi modelli con le Semantic Web Technologies un approccio è quello di mappare l'input di una rete neurale o i singoli neuroni a classi di un'ontologia o entità di un grafo della conoscenza (*knowledge graph*).

Ad esempio, sulla base della classificazione delle immagini emessa da una rete neurale, è possibile eseguire il ragionatore *DL-Learner* sull'ontologia per creare espressioni di classe che fungono da spiegazioni. Allo stesso modo, nel lavoro di **[177]**, i contenuti delle immagini vengono estratti come triple RDF e quindi abbinati a DBpedia tramite il predicato *"same-concept"*. Per rispondere alle domande fornite dall'utente su un'immagine, il sistema traduce ogni domanda in una query SPARQL che viene eseguita sulla knowledge base combinata. I risultati di questa operazione vengono quindi utilizzati per dare una risposta e confermarla con ulteriori prove che fungono da spiegazione.

Per quanto concerne l'apprendimento non supervisionato, significative ricerche accademiche mirano a creare embedding spiegabili di knowledge graph o grazie al loro ausilio **[178, 179]**. Per incrementare il livello di interpretabilità degli algoritmi di clustering si potrebbe far uso di conoscenza tassonomica codificata in WordNet per derivare una funzione di similarità semantica che porta a definire cluster più interpretabili.

Nell'ambito del progetto Digital Future, la componente del Middleware denominata "**Explanation Tool**" si occuperà di associare una ontologia di dominio specifico, ad esempio nel contesto del verticale Healthcare, per fornire una spiegazione dei risultati di classificazione derivante dalla applicazione di un modello ottenuto attraverso l'apprendimento automatico supervisionato, costruendo un individuo ontologico con le

caratteristiche della classe predetta in forma di Description Logic di tipo ALN (Attributive Language with Number restrictions).

La combinazione delle Semantic Web Technologies e ML offre un grande potenziale per facilitare modelli spiegabili. Abbiamo identificato la corrispondenza dei dati ML con le entità della knowledge base come una sfida centrale che deve essere superata dalla ricerca futura. In particolare, sono richiesti metodi automatizzati e affidabili per la corrispondenza delle conoscenze. Sono necessarie ulteriori ricerche in questo settore e campi correlati come l'annotazione semantica per consentire una corrispondenza delle conoscenze efficace ed efficiente.

Esiste inoltre una certa concentrazione su specifiche tecniche ML e Semantic Web Technologies. È necessario svolgere ulteriori lavori sull'apprendimento e il clustering spiegabili. In questo contesto, notiamo anche che il lavoro attraverso diverse discipline e attività rimane ancora un po' isolato anche se concetti come i *linked data* forniscono gli strumenti per integrare i vari domini. Pertanto, le aree di ontologia o apprendimento dei grafi di conoscenza, nonché la corrispondenza della knowledge base, svolgono un ruolo importante nel raggiungimento di questo obiettivo. I lavori futuri dovranno pertanto trovare modi per mitigare la potenziale mancanza di interconnessione dei dati e la maggiore complessità di tali sistemi.

Infine, sottolineiamo la necessità di lavorare in futuro per mirare a sistemi veramente spiegabili che incorporino ragionamenti e conoscenze esterne comprensibili all'uomo. Per raggiungere questo obiettivo, i futuri sistemi di spiegazione devono garantire che le spiegazioni fornite siano veritiere con l'algoritmo ML sottostante. Inoltre, tali approcci dovrebbero essere in grado di spiegare non solo il modo in cui un risultato è correlato a una rappresentazione di interesse, ma anche il modo in cui tale rappresentazione è stata ottenuta. Ad esempio, non è sufficiente giustificare che un volto umano è stato rilevato affermando che occhi, bocca e naso sono stati riconosciuti e che queste caratteristiche fanno parte di un volto umano (ad esempio inferito tramite ontologia). Un sistema veramente spiegabile dovrebbe anche essere in grado di spiegare perché queste caratteristiche sono state riconosciute.

5.6 Algorithm Recommender Tool

Nel contesto del Middleware della piattaforma DF, il sistema di raccomandazione avrà come scopo principale quello di somministrare in fase di ricerca da parte degli esperti di dominio di un Pillar un elenco ordinato, o meglio, una classifica dei possibili e più utili algoritmi da usare in funzione delle esigenze del proprio dominio di applicabilità e/o in funzione delle preferenze e/o valutazioni passate. A differenza dei classici algoritmi di raccomandazione, basati su filtraggio collaborativo dei dati e su misure di similarità tra oggetti, fornire un servizio di suggerimento di algoritmi di apprendimento automatico chiama in causa una serie di variabili e fattori che devono essere valutati e che concorrono a discriminare positivamente o negativamente qualsiasi tipo di algoritmo. Infatti, sviluppare la giusta soluzione di apprendimento automatico per un problema di vita reale è raramente solo un problema di matematica applicata. Alcune delle decisioni che si prendono quando si sceglie un metodo di apprendimento automatico hanno meno a che fare con l'ottimizzazione o gli aspetti tecnici del metodo, ma più sulle decisioni di business, sugli obiettivi degli stakeholders e richiedono una notevole esperienza: un esperto di solito sa in quale ordine provare vari algoritmi per un tipo specifico di problema, in modo tale da passare in modo approssimativo dalle tecniche più semplici a quelle più complicate. Idealmente ci si dovrebbe fermare in un punto in cui rendere l'algoritmo più complesso non dà molto miglioramento nelle prestazioni. Il nostro obiettivo è però quello di fornire soluzioni complete ed affidabili che siano in grado di supportare l'esperto nelle scelte che si troverà a dover effettuare nella fase di sperimentazione di un dataset, e magari riuscendo a sorprenderlo e convincerlo nell'utilizzare un algoritmo consigliato da noi piuttosto che un altro standard.

Nel risolvere un problema di apprendimento automatico, essere in grado di combinare e bilanciare tutti questi fattori è fondamentale. Infatti, l'apprendimento automatico supervisionato è in parte scienza e in parte "arte". Quando si esaminano i metodi di apprendimento automatico, non esiste una soluzione o un approccio che si adatti a tutti i tipi di dati. Ci sono diversi fattori che possono influenzare la decisione di scegliere un metodo di apprendimento automatico. Spesso accade che, se si sono identificati dei

metodi come potenziali buoni approcci, si va ad inserire i dati in essi e si eseguono tutti in parallelo o in serie e alla fine si valutano le prestazioni per selezionare i migliori metodi. Alcuni problemi sono molto specifici e richiedono un approccio unico. Mentre altri problemi sono aperti a più approcci e richiedono una metodologia *trial & error*. In particolare, i metodi di classificazione sono molto aperti a svariate tipologie di applicazione. Potrebbero essere utilizzati nel rilevamento di anomalie o potrebbero essere utilizzati per costruire modelli più generali di predizione.

Innanzitutto, a partire dalla categorizzazione del problema, si può distinguere se l'obiettivo può essere raggiunto con un approccio di apprendimento automatico *supervisionato* o *non supervisionato* (e ancora più nel dettaglio degli approcci di apprendimento automatico, si possono considerare anche metodi semi-supervisionati e metodi di apprendimento per "rinforzo" – il cosiddetto *Reinforcement Learning*).

Successivamente, per il problema specifico di apprendimento automatico supervisionato, bisogna distinguere tra un problema di **classificazione** o un problema di **regressione**.

Possiamo riassumere queste scelte a seconda della categorizzazione degli input e degli output [58]:

1. Categorizzazione degli input:
    - Se si hanno dati etichettati, si tratta di un problema di apprendimento supervisionato.
    - Se si hanno dati non etichettati, si tratta di un problema di apprendimento non supervisionato.
    - Se si hanno dati parzialmente etichettati e parzialmente non etichettati, si tratta di un problema di apprendimento semi-supervisionato.
    - Se si vuole ottimizzare una funzione obiettivo interagendo con un ambiente, è un problema di reinforcement learning.
2. Categorizzazione degli output:
    - Se l'output del modello è un numero, allora si tratta di un problema di regressione.

- Se l'output del modello è una classe (o categoria), allora si tratta di un problema di classificazione.
- Se l'output del modello è un insieme di raggruppamenti, allora si tratta di un problema di clustering
- Se si vuole rilevare una anomalia nei dati, allora si tratta di un problema di anomaly detection.

Questo approccio, insieme ad altri fattori che potranno variare a seconda del tipo di problema di apprendimento, sarà applicato da Algorithm Recommender per produrre una classifica degli algoritmi di apprendimento che potrà variare a seconda del contesto operativo, nel caso si vorrà sapere quale algoritmo è più utilizzato in uno specifico dominio., Inoltre, grazie alla componente Mediator di Digital Future, sarà possibile applicare uno scouting tecnologico per aggiungere informazioni aggiuntive discriminanti di algoritmi di apprendimento automatico rilevanti per il contesto operativo di riferimento. [59]

5.6.1 *Recommender di metodi di Classificazione in Machine Learning*

Ulteriore discriminante tra i metodi di classificazione è il dover distinguere se si tratta di un problema di **classificazione binaria** o di **classificazione multiclasse**. Si parla di classificazione binaria quando l'output del problema dovrà predire tra esattamente due categorie (pertanto, viene chiamata anche classificazione a due classi) e quindi si vuole ottenere una risposta semplice a domande a doppia scelta, come sì o no, oppure vero o falso. Esempi di tali domande possono essere: Questo tweet è positivo? Questo cliente rinnoverà il suo servizio? Quale dei due coupon attira più clienti?

Si parla invece di classificazione multiclasse quando l'output del problema dovrà predire tra più categorie e quindi si vuole ottenere una risposta complicata a domande con più possibili risposte. In maniera equivalente a quanto riportato per la classificazione binaria, esempi di tali domande possono essere: Qual è lo stato d'animo di questo tweet? Quale servizio sceglierà questo cliente? Quale delle diverse promozioni attira più clienti?

Successivamente, in maniera generica si possono considerare le seguenti **informazioni legate agli algoritmi**, ovvero le caratteristiche (più o meno) comuni a tutti i metodi classici più noti in letteratura —e non solo— di apprendimento automatico [[60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76]], fruibili dalla componente di Machine Learning Libraries del Data Governance Layer del Middleware, che possono discriminare la scelta e l'efficacia di un metodo rispetto ad un altro:

- Velocità di apprendimento (**Tra**)
- Velocità di predizione (**Pre**)
- Capacità di avere modelli scalabili (**Big**)
- Capacità di apprendimento incrementale (**Inc**)
- Capacità di lavorare con un piccolo dataset (**Sma**)
- Capacità di lavorare con dati categorici (**Cat**)
- Capacità di eseguire classificazione multiclasse (**Mul**)
- Capacità di gestire l'interazione tra features (**Fel**)
- Robustezza ai dati mancanti (**Mis**)
- Robustezza agli outliers (**Out**)
- Predisposizione all'interpretabilità (**Int**)
- Robustezza a dati ridondanti (**Red**)
- Robustezza ad un elevato numero di feature (**Dim**)
- Robustezza a dati standardizzati o normalizzati (**Std**)
- Necessità di predisporre feature engineering sui dati (**Eng**)
- Bassa complessità degli iperparametri (**Hyp**)
- Robustezza all'overfitting (**Ove**)

Eventuali vincoli di scelta sono legati alla capacità computazionale (computational overhead) o di storage del sistema/workstation.

Nello specifico, tra i metodi di classificazione che andremo a considerare tra quelli implementabili in Digital Future, troviamo:

- Librerie di Machine Learning di Spark MLlib [77]:

  - Binomial logistic regression (**Bin LR**)
  - Multinomial logistic regression (**Mul LR**)
  - Decision tree classifier (**DT**)
  - Random forest classifier (**RF**)
  - Gradient-boosted tree classifier (**GBT**)
  - Multilayer perceptron classifier (**MLP**)
  - Linear Support Vector Machine (**Lin SVM**)
  - One-vs-Rest classifier (**OvR**)
  - Naive Bayes (**NB**)
- Librerie di Deep Learning di Tensorflow / Keras [78]:
  - Feedforward Neural Network classifier
  - Deep Convolutional Neural Network classifier (**CNN**)
  - Deep Recurrent Neural Network (**RNN**)

Di seguito, quindi, si riporta una breve descrizione dei punti di forza e di debolezza di ogni metodo di classificazione appena introdotto, evidenziando il contesto operativo in cui in genere si ottengono dei buoni risultati.

Binomial logistic regression. Non si confondi questi algoritmi di classificazione con i metodi di regressione per l'uso del termine "regressione" nel nome. La regressione logistica binomiale esegue la classificazione binaria, quindi gli output dell'etichetta sono binari. Tale metodo prende una combinazione lineare di features e applica una funzione non lineare (cioè, una funzione sigmoide) ad esso, offrendo molti modi per regolarizzare il proprio modello, e non bisogna preoccuparsi di come le feature siano correlate (come si fa invece nel Naive Bayes). Possiede anche una buona interpretazione probabilistica e si può facilmente aggiornare il modello per acquisire nuovi dati (a differenza degli alberi di decisione o SVM). Viene usata se vuole avere un framework probabilistico o se si è in attesa di ricevere più dati da addestrare in futuro che si desidera incorporare rapidamente nel modello. La regressione logistica può anche aiutare a capire i fattori che contribuiscono alla predizione e non è solo un metodo black-box. La regressione logistica può essere utilizzata in casi quali predire il tasso di abbandono del cliente,

rilevamento di credito e frode, misurare l'efficacia delle campagne di marketing, classificare animali sulla base delle loro caratteristiche, risultati medici (studi di sopravvivenza) e infine scienze sociali (effetti del trattamento).

Multinomial logistic regression. La classificazione multiclasse è supportata dalla regressione logistica multinomiale (cioè, una funzione softmax). Le stesse assunzioni fatte per la regressione logistica binomiale valgono anche per quella multinomiale, con l'aggiunta di poter gestire più classi di apprendimento.

Decision tree classifier. Gli alberi di decisione gestiscono facilmente le interazioni tra le feature e sono non-parametrici, quindi sono robusti contro i valori anomali e non bisogna preoccuparsi se i dati sono separabili linearmente. Uno svantaggio è che non supportano l'apprendimento online, quindi è necessario ricostruire l'albero quando si presentano nuovi esempi. Un altro svantaggio è che sono facilmente sensibili all'overfitting, ma è qui che entrano in gioco metodi di ensemble come Random Forest o Boosted Trees. Gli alberi di decisione possono anche avere molta memoria (più funzioni hai e più profondo sarà l'albero); inoltre sono strumenti eccellenti per aiutarti a scegliere tra diverse azioni, come decisioni di investimento, fidelizzazione dei clienti previsioni metereologiche, diagnosi medica, analisi del rischio di credito e classificazioni astronomiche.

Random forest classifier. La Random Forest è un insieme (ensemble) di alberi di decisione: combina molti alberi di decisione al fine di ridurre il rischio di overfitting. Può risolvere problemi di classificazione con dataset di grandi dimensioni. Aiuta anche a identificare le variabili più significative da migliaia di variabili di input. La Random Forest è altamente scalabile per qualsiasi numero di dimensioni e ha prestazioni generalmente accettabili. Tuttavia, la fase di apprendimento può essere lenta (a seconda della parametrizzazione) e non è possibile migliorare iterativamente i modelli generati. La Random Forest può essere utilizzata in applicazioni del mondo reale come predire pazienti ad alto rischio, predire i guasti delle parti nella produzione, predire i debitori morosi, classificazioni bioinformatiche.

Gradient-boosted tree classifier. Noto anche con l'acronimo GBT, si tratta di un altro esempio di insieme (ensemble) di alberi di decisione: i GBT addestrano iterativamente alberi di decisione con lo scopo di minimizzare una funzione di loss. Infatti, l'idea del "boosting" (tradotto come potenziamento) può essere interpretato come un algoritmo di ottimizzazione su una funzione di costo adeguata. Sostanzialmente, le stesse assunzioni fatte per la Random Forest valgono anche per il GBT. Inoltre, il GBT è spesso efficace quando si ha a disposizione un dataset molto grande. In genere, il GBT viene utilizzato nel campo dell'apprendimento di una classifica di elementi (un insieme totalmente ordinato). I motori di ricerca web commerciali Yahoo e Yandex utilizzano infatti varianti del GBT nei loro motori di classifica appresi automaticamente.

Multilayer perceptron classifier. Noto anche come MLP, si tratta di un classificatore basato su rete neurale artificiale di tipo Feedforward. Il MLPC è costituito da più livelli di nodi. Ogni livello è completamente connesso al livello successivo nella rete. I nodi nel livello di input rappresentano i dati di input. Tutti gli altri nodi mappano gli input alle uscite mediante una combinazione lineare degli input con i pesi del nodo e il bias associato, applicando una funzione di attivazione (cioè, una funzione sigmoide). Nel caso di apprendimento multiclasse, i nodi del livello di output sono associati ad una funzione softmax. Il MLPC impiega il metodo della backpropagation per l'apprendimento del modello. Con le reti neurali, i modelli estremamente complessi possono essere addestrati e possono essere utilizzati in modalità black box, senza svolgere un'imprevedibile progettazione di feature complesse prima dell'addestramento del modello. Il MLP può essere utilizzato in applicazioni del mondo reale come il riconoscimento di un oggetto in una immagine. D'altra parte, il MLP è molto difficili da interpretare e la parametrizzazione è estremamente sconvolgente.

Linear Support Vector Machine. Meglio nota con l'acronimo SVM, si tratta di una tecnica di apprendimento automatico supervisionato ampiamente utilizzata nei problemi di riconoscimento e di classificazione dei pattern, quando i dati hanno esattamente due classi. Danno in genere buone garanzie teoriche riguardo il l'overfitting. Tale metodo è particolarmente popolare nei problemi di classificazione del

testo. Pregio della SVM è la grande capacità di memoria. Tuttavia, si tratta di un metodo di difficile interpretazione. La SVM può essere utilizzata in applicazioni del mondo reale come rilevare persone con malattie comuni come il diabete, riconoscimento dei caratteri scritto a mano, categorizzazione del testo per argomenti, predizione dei prezzi del mercato azionario e classificazione di proteine.

One-vs-Rest classifier. OneVsRest è un esempio di riduzione dell'apprendimento automatico per l'esecuzione della classificazione multiclasse dato un classificatore di base che può eseguire la classificazione binaria in modo efficiente. È anche conosciuto come "One-vs-All". OneVsRest è implementato come uno stimatore. Per il classificatore di base, prende l'istanza di un classificatore (ad esempio la regressione logistica) e crea un problema di classificazione binaria per ciascuna delle classi. Il classificatore per la classe k-esima è addestrato per predire se l'etichetta è proprio la k-esima o no, distinguendo la classe k da tutte le altre classi. Le predizioni vengono eseguite valutando ciascun classificatore binario e l'indice del classificatore più sicuro viene preso in output come etichetta.

Naive Bayes. È una tecnica di classificazione basata sul teorema di Bayes, molto facile da costruire, e particolarmente utile per set di dati molto grandi. Insieme alla semplicità, è noto che il Naive Bayes supera anche i metodi di classificazione estremamente sofisticati. Il Naive Bayes è anche una buona scelta quando la CPU e le risorse di memoria sono un fattore limitante. Il Naive Bayes è molto semplice: sostanzialmente si tratta solo di fare dei conteggi. Se l'assunzione di indipendenza condizionale è effettivamente valida, un classificatore di Naive Bayes convergerà più rapidamente dei modelli discriminativi come la regressione logistica, quindi è necessario un minore numero di dati di addestramento. E anche se l'ipotesi non regge, un classificatore Naive Bayes svolge spesso un ottimo lavoro nella pratica. Il suo principale svantaggio è che non può imparare le interazioni tra le feature. Ulteriore peculiarità è che evita l'overfitting. Il Naive Bayes può essere utilizzato in applicazioni del mondo reale come analisi dello stato d'animo e classificazione del testo, sistemi di raccomandazione come Netflix o Amazon, e infine per contrassegnare una email come spam o non spam.

Feedforward Neural Network. Si tratta sostanzialmente dello stesso caso del MLPC. Delle differenze si possono riscontrare proprio per la differenza di implementazione della libreria adottata (in tal caso, Tensorflow / Keras). In particolare, in questo caso viene data all'esperto tecnico di dominio maggiore scelta di costruzione dell'architettura neurale "deep", con una notevole flessibilità di decisione del numero di layer nascosti e del numero di neuroni per layer, oltre a poter definire il tipo di funzione di attivazione dei neuroni, il numero di epoche per l'addestramento e la funzione di loss che minimizzi l'errore tra l'output reale e quello predetto.

Convolutional Neural Network. Se si lavora con le immagini, le CNN mostrano grandi risultati. Le non linearità sono rappresentate da strati convolutivi e di raggruppamento, in grado di catturare le feature caratteristiche delle immagini. A partire dal 2017, non si può immaginare l'elaborazione delle immagini e dei video senza le CNN. Tali architetture all'avanguardia come AlexNet, GoogLeNet e ResNet sono tutte varianti di una rete neurale convolutiva.

Recurrent Neural Network. Per lavorare con testi e sequenze è meglio scegliere reti neurali ricorrenti. Le RNN contengono moduli LSTM (*Long Short Term Memory*) o GRU (*Gated Recurrent Unit*) e possono lavorare con dati per i quali conosciamo la dimensione in anticipo (cioè, la lunghezza della sequenza). Alcune delle applicazioni più conosciute per le RNN sono la traduzione automatica di testi, il riconoscimento di appartenenza del testo in una lingua, il riconoscimento vocale e il riconoscimento dell'audio.

5.6.1.1 *Fattori di valutazione degli algoritmi di classificazione in apprendimento automatico supervisionato*

Nel precedente paragrafo sono stati forniti delle brevi descrizioni degli algoritmi recuperate da fonti scientifiche. Da approfondimenti e una serie di valutazioni sono emerse delle forme di classificazioni degli stessi algoritmi derivanti da una mera osservazione, diretta o indiretta, dei risultati ottenuti a seguito di prove empiriche. Tali risultati hanno portato all'identificazione di aree afferenti ad alcuni domini nei quali l'algoritmo ha fornito risultati migliori rispetto ad altri. A titolo di esempio citiamo: l'algoritmo "Binomial logistic regression" e l'algoritmo "Multinomial logistic regression"

risultano aver raggiunto un miglior utilizzo nelle aree di *predizione del tasso di abbandono del cliente*, o per il *rilevamento di credito e frode* o ancora nella *misurazione dell'efficacia delle campagne di marketing*; piuttosto che l'algoritmo "Linear Support Vector Machine" più utilizzato in applicazioni del mondo reale sulla *rilevazione di persone con malattie comuni*, piuttosto che nel *riconoscimento dei caratteri scritti a mano* o nella *categorizzazione del testo per argomenti*, o ancora *predizione dei prezzi nel mercato azionario*, e così via. Questo modello di classificazione porta a definire una metodologia per l'identificazione di fattori discriminanti tra gli algoritmi di classificazione e il loro impiego. Il nostro obiettivo, infatti, è quello di poter suggerire *a priori* quali potrebbero essere gli algoritmi migliori a seconda di alcuni fattori, quali contesto operativo, la semplicità di utilizzo o la propensione ad ottenere ottime performance sulla base della complessità dei dati da addestrare. Questa analisi ci ha portato a studiare ulteriormente tali fattori, in modo che da essi noi possiamo esplicitare una conoscenza latente e fornire quindi un servizio di raccomandazione in grado di suggerire una classifica ordinata di algoritmi, dal più rilevante al meno rilevante, che di fatto genera un valore per l'esperto di dominio che si ritrova nella fase di primo approccio al problema di machine learning che gli si pone davanti.

Di seguito, le informazioni precedentemente descritte per ogni metodo di classificazione vengono generalizzate e condivise, in modo da indicare su quali parametri, derivati dalle informazioni legate agli algoritmi di classificazione, si può già determinare una preliminare "qualità" di scelta e ranking.

NFL	Bin LR	Lin SVM	NB	Mul LR	DT clf	RF clf	GBT clf	OvR	MLP clf	CNN clf	RNN clf
Tra	yes	no	yes	yes	yes	no	no	N/A	no	no	no
Pre	yes	yes	yes	yes	yes	no	yes	N/A	yes	yes	yes
Big	yes	no	yes	yes	yes	yes	yes	N/A	yes	yes	yes
Inc	yes	yes	yes	yes	no	no	no	N/A	yes	yes	yes
Sma	yes	yes	yes	yes	yes	no	no	N/A	no	no	no
Cat	no	no	yes	no	yes	yes	yes	N/A	no	no	no
Mul	no	no	yes	yes	yes	yes	yes	yes	yes	yes	yes
Fel	no	yes	no	no	yes	yes	yes	N/A	no	no	no
Mis	yes	yes	no	yes	yes	yes	yes	N/A	yes	yes	yes
Out	no	yes	no	no	yes	yes	no	N/A	yes	yes	yes
Ove	yes	yes	yes	yes	no	yes	yes	N/A	no	yes	yes
Red	yes	yes	no	yes	no	no	no	N/A	yes	yes	yes
Std	yes	no	no	yes	yes	yes	yes	N/A	no	no	no
Dim	no	yes	yes	no	yes	yes	yes	N/A	yes	yes	yes
Int	yes	no	yes	yes	yes	no	no	N/A	no	no	no
Hyp	yes	yes	yes	yes	yes	yes	no	N/A	no	no	no

Eng	no	no	no	no	yes	yes	no	N/A	yes	yes	yes

Informazioni legate agli algoritmi di Classificazione

Il Teorema "***No Free Lunch***" (NFL) [79] afferma che non esiste un modello che funzioni meglio per ogni problema. Le ipotesi di un modello ottimale per un problema potrebbero non reggere per un altro problema, quindi è comune in machine learning provare più modelli e trovarne uno che funzioni meglio per un particolare problema. Questo è particolarmente vero nell'apprendimento supervisionato, in cui la validation e la cross-validation sono comunemente utilizzate per valutare l'accuratezza predittiva di più modelli di varia complessità per trovare il modello migliore. Si noti che un modello che funziona bene potrebbe anche essere formato da più algoritmi. A seconda del problema, è importante valutare i compromessi tra velocità, accuratezza e complessità di diversi modelli e algoritmi e trovare un modello che funzioni meglio per quel particolare problema. A partire da queste assunzioni, e quelle derivate dallo studio di ogni algoritmo di classificazione che hanno portato alla generazione della Tabella 1, è possibile definire un ranking iniziale degli algoritmi semplicemente contando il numero di caratteristiche che ogni algoritmo possiede. Questo fattore iniziale prenderà quindi il nome di "*Fattore No Free Lunch*". Fermo restando quanto esposto che è una ipotesi chiara, nulla però vieta nel pensare comunque che quello utilizzato e inizialmente quotato come "modello ottimale" sia una sola parvenza a seguito della mancanza di valutazione di qualche altro modello mai preso in ipotesi e quindi mai utilizzato. Da qui nasce infatti il nostro approfondimento come modo per definire un metodo di valutazione degli algoritmi di classificazione, che di fatto determina già una prima valutazione discriminante.

Secondo fattore derivante dal teorema NFL è che, a parità di condizioni, si parta con la scelta dell'algoritmo di classificazione più semplice. Il **Rasoio di Occam** segue esattamente questa regola: è un principio di risoluzione dei problemi che afferma essenzialmente che "le soluzioni più semplici hanno maggiori probabilità di essere corrette rispetto a quelle complesse" [80]. Tale principio può essere quindi inglobato nel

nostro studio e possiamo determinare un fattore di semplicità da inglobare nel nostro sistema di raccomandazione, distinguendosi come fattore derivante altresì da uno studio empirico, ma in una veste originale e concorrente al nostro scopo. Quindi ad ogni algoritmo viene assegnato un "*Fattore Rasoio di Occam*" che indica il proprio grado di semplicità, in una scala da 1 a 10, dove 10 indica il massimo della semplicità.

Occ	Bin LR	Lin SVM	NB	Mul LR	DT clf	RF clf	GBT clf	OvR	MLP clf	CNN clf	RNN clf
Fattore Rasoio di Occam	9	9	10	8	7	6	6	8	5	4	4

Terzo fattore molto importante è il **tipo di problema di classificazione**. Sebbene sia stato già inserito come caratteristica da considerare tra metodi di classificazione binari (B) e multiclasse (M), questo fattore di fatto discrimina totalmente l'utilizzo di algoritmi di classificazione binaria nel caso ci si trovi davanti ad un problema di classificazione multiclasse. Pertanto, tali algoritmi devono essere esclusi dalla classifica. Viceversa, nel caso di un problema di classificazione binaria, non è detto che gli algoritmi limitati a questa peculiarità siano più performanti di altri che ammettano anche la possibilità di gestire più classi. Questo comportamento ci può tornare utile e dare un sintomo di "intelligenza" al suggerimento, riuscendo a scremare o riordinare la classifica di algoritmi da suggerire a seconda della natura di classificazione del problema da analizzare. Quindi, il "*Fattore Tipo di Problema di Classificazione*" deve agire da flag, potendo quindi consentire l'inserimento in classifica degli algoritmi utili al tipo di problema da affrontare.

Typ	Bin LR	Lin SVM	NB	Mul LR	DT clf	RF clf	GBT clf	OvR	MLP clf	CNN clf	RNN clf
Fattore Tipo di Problema di Classif.	B	B	M	M	M	M	M	M	M	M	M

Quarto fattore discriminante è quello di **complessità dei dati addestrabili**. Se si ha a che fare con dataset di classificazione di informazioni complesse, come immagini, video, segnali audio, è noto che gli algoritmi classici di machine learning (ovvero quelli della libreria Spark MLlib) non siano in grado di gestirli totalmente o di ottenere risultati accettabili quanto gli algoritmi di Deep Learning. Possiamo sfruttare questa conoscenza empirica e determinare quindi un altro fattore da inglobare nel nostro recommender. Quindi, il "*Fattore Training di Dati Complessi*" assegna ad ogni algoritmo un valore che indica la sua attitudine a gestire dataset con questo tipo di informazioni complesse.

Com	Bin LR	Lin SVM	NB	Mul LR	DT clf	RF clf	GBT clf	OvR	MLP clf	CNN clf	RNN clf
Fattore Training di Dati Complessi	4	6	5	5	5	6	7	5	9	10	10

Quinto fattore decisivo per dalle informazioni derivanti dal **Contesto Operativo**. I contesti operativi di cui Digital Future avrà conoscenza di dominio sono tre:

- Fabbrica Intelligence (FI)
- Healthcare (HC)
- Sicurezza ambientale e controllo del territorio (SACT)

A seconda dell'analisi condotta precedentemente sulle informazioni rilevanti e il contesto in cui gli algoritmi di classificazione si sono distinti in letteratura, abbiamo condotto uno studio di efficacia degli algoritmi. La nostra idea è quella di determinare un fattore di probabilità di efficacia di ogni algoritmo per il contesto operativo di riferimento. Si definiscono pertanto le seguenti stime percentuali di efficacia di ogni algoritmo di classificazione.

Contesto Operativo (CO)	Bin LR	Lin SVM	NB	Mul LR	DT clf	RF clf	GBT clf	OvR	MLP clf	CNN clf	RNN clf
FI	90	90	80	90	100	100	90	100	90	100	100
HC	100	100	60	100	70	80	90	100	90	100	100
SACT	50	70	50	50	70	80	90	100	80	100	100

Nonostante l'ampia gamma di soluzioni già presenti in Digital Future, per eseguire un task di classificazione l'esperto tecnico del dominio potrebbe voler ricevere dei suggerimenti su altri algoritmi di classificazione che, a seconda del contesto operativo, risultino essere efficaci oltre che essere una novità nello stato dell'arte della letteratura e nell'innovazione scientifico-tecnologica. A tal fine, un altro componente della Piattaforma DF, come il Mediator, potrebbe occuparsi di questo, ossia implementare funzionalità di scouting tecnologico e mettere a disposizione del componente "Algorithm Recommender" suggerimenti di una classifica ordinata di metodi di apprendimento automatico supervisionato di classificazione. In particolare, per ogni nuovo algoritmo che il Mediator potrebbe individuare come rilevante per il contesto operativo di riferimento, si potrebbe, ad esempio, valutare i seguenti parametri:

- Accuratezza su dati dello stesso contesto operativo (**Acc**)
- Numero Citazioni Paper scientifico di riferimento (**Cit**)

- h-Index del primo autore (**H1**)
- h-Index medio degli autori (**Havg**)
- Disponibilità codice online (**Ava**)
- Numero di revisioni/aggiornamenti dell'algoritmo (**Rev**)
- Data ultima revisione/aggiornamento (**Upd**)

5.6.1.2 *Ranking Recommender per metodi di classificazione in apprendimento automatico supervisionato*

Considerati i seguenti fattori:

1. Fattore "Tipo di Problema di Classificazione" (**Typ**)
2. Fattore "Rasoio di Occam" (**Occ**)
3. Fattore "Training di Dati Complessi" (**Com**)
4. Fattore "No Free Lunch" (**NFL**)
5. Fattore "Contesto Operativo" (**CO**)
6. Fattore "Scouting Tecnologico" (**ST**)

Viene proposta di seguito la formula che assegna un valore numerico ad ogni metodo di classificazione discusso precedentemente, in modo tale da poter restituire successivamente una classifica, ovvero un ranking ordinato, degli algoritmi di classificazione nell'ambito dell'apprendimento automatico supervisionato.

Definizione 1. (Classifier Recommender Evaluation Function) Sia ***p*** il problema di apprendimento, sia A l'insieme degli algoritmi di classificazione e $a \in A$ un algoritmo di tale insieme, e sia $r: A \rightarrow \mathbb{R}$ una funzione che mappa ogni elemento di A ad un numero reale, allora, $\forall a \in A$:

$$r(a) = \begin{cases} 0 & se\ Typ(a) = "B" \wedge Typ(p) = "M" \\ (Occ(a) + Com(a) + NFL(a)) \dfrac{CO(a)}{100} + ST(a) & altrimenti \end{cases}$$

dove:

- $Typ(\cdot) = \{"\mathrm{B}", "\mathrm{M}"\}$ indica la funzione che mappa il fattore "Tipo di Problema di Classificazione" per identificare il tipo di algoritmo e di problema come classificazione binaria o multiclasse;
- $Occ : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Rasoio di Occam";
- $Com : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Training di Dati Complessi";
- $NFL : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "No Free Lunch";
- $CO : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Contesto Operativo";
- $ST : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Scouting Tecnologico".

Quindi, la funzione $r(\cdot)$ restituisce un valore numerico che può essere ordinato totalmente tra tutti gli algoritmi di classificazione. Tale valore è il risultato della combinazione dei vari fattori descritti precedentemente che consentono di valutare le peculiarità di ogni algoritmo di classificazione e propensione ad ottenere delle performance migliori a seconda del contesto operativo, considerando anche l'apporto del Mediator come un "bonus" in cui a risultare tra i primi posti in classifica potrebbe esserci un algoritmo di classificazione non precedentemente considerato tra quelli già previsti nella componente Machine Learning Libraries del Data Governance di Digital Future.

Definizione 2. (Classifier Algorithm Recommender Ranking) Sia A l'insieme degli algoritmi di classificazione e $a \in A$ un algoritmo di tale insieme, e sia $r : A \rightarrow \mathbb{R}$ la funzione di valutazione su A, allora l'Algorithm Recommender Ranking associa ad A un ranking $\succeq_A^r$ su A tale che $\forall a, b \in A,\ a \succeq_A^r b$ se e solo se $r(a) \geq r(b)$.

In questo modo l'ordinamento totale su A genera una classifica ordinata a seconda delle specifiche richieste del problema da analizzare.

5.6.2 *Recommender di metodi di Regressione in Machine Learning*

L'analisi di regressione è un insieme di processi statistici per stimare le relazioni tra le variabili. Comprende molte tecniche per la modellazione e l'analisi di più variabili, quando il focus è sulla relazione tra una variabile dipendente e una o più variabili

indipendenti (o 'predittori'). Più specificamente, l'analisi di regressione aiuta a capire come il valore tipico della variabile dipendente (o 'variabile criterio') cambia quando viene variata una qualsiasi delle variabili indipendenti, mentre le altre variabili indipendenti sono fisse.

Ai fini di produrre una classifica ordinata per tale categoria di algoritmi, in maniera generica si possono considerare le seguenti **informazioni legate agli algoritmi**, ovvero le caratteristiche (più o meno) comuni a tutti i metodi di regressione più noti in letteratura — e non solo — di apprendimento automatico [[60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76]], fruibili dalla componente di Machine Learning Libraries del Data Governance Layer del Middleware, che possono discriminare la scelta e l'efficacia di un metodo rispetto ad un altro:

- Velocità di apprendimento (**Tra**)
- Velocità di predizione (**Pre**)
- Capacità di avere modelli scalabili (**Big**)
- Capacità di apprendimento incrementale (**Inc**)
- Capacità di lavorare con un piccolo dataset (**Sma**)
- Capacità di lavorare con dati categorici (**Cat**)
- Capacità di eseguire classificazione multiclasse (**Mul**)
- Capacità di gestire l'interazione tra features (**Fel**)
- Robustezza ai dati mancanti (**Mis**)
- Robustezza agli outliers (**Out**)
- Predisposizione all'interpretabilità (**Int**)
- Robustezza a dati ridondanti (**Red**)
- Robustezza ad un elevato numero di feature (**Dim**)
- Robustezza a dati standardizzati o normalizzati (**Std**)
- Necessità di predisporre feature engineering sui dati (**Eng**)
- Bassa complessità degli iperparametri (**Hyp**)
- Robustezza all'overfitting (**Ove**)

Eventuali vincoli di scelta sono legati alla capacità computazionale (computational overhead) o di storage del sistema/workstation.

Nello specifico, tra i metodi di regressione che andremo a considerare tra quelli implementabili in Digital Future, troviamo quelli disponibili dalla libreria di Machine Learning di Spark MLlib [77]:

- Linear regression (**LinR**)
- Generalized Linear Regression (**GLM**)
- Decision Tree Regression (**DT Reg**)
- Random Forest Regression (**RF Reg**)
- Gradient-boosted Tree Regression (**GBT Reg**)
- Survival Regression (**SurR**)
- Isotonic Regression (**IsoR**)

Di seguito, quindi, si riporta una breve descrizione dei punti di forza e di debolezza di ogni metodo di regressione appena introdotto, indicando ove possibile le situazioni in cui in genere si ottengono dei buoni risultati.

Linear Regression. Questo è probabilmente l'algoritmo più semplice nell'apprendimento automatico. Si considerano le feature $x_1, \ldots x_n$ di oggetti (matrice A) ed etichette (vettore b). L'obiettivo è trovare i pesi ottimali $w_1, \ldots w_n$ e bias per queste caratteristiche in base ad alcune funzioni di loss (perdita), ad esempio MSE o MAE. Nel caso di MSE esiste un'equazione matematica dal metodo dei minimi quadrati:

$$(A^T A)^{-1} A^T b$$

In pratica, è più facile ottimizzarlo con la discesa del gradiente, che è molto più efficiente dal punto di vista computazionale. Nonostante la semplicità di questo algoritmo, funziona abbastanza bene quando si hanno migliaia di feature, ad esempio, un sacco di parole o n-grammi nell'analisi del testo. Gli algoritmi più complessi soffrono di overfitting (sovradattamento) di molte feature e non di enormi set di dati, mentre la regressione lineare fornisce una qualità decente. Per evitare il sovradattamento usiamo spesso tecniche di regolarizzazione come LASSO e RIDGE. L'idea è di aggiungere la

somma dei moduli di pesi e la somma dei quadrati dei pesi, rispettivamente, alla nostra funzione di loss.

Generalized Linear Regression. Contrariamente alla regressione lineare in cui si presume che l'output segua una distribuzione gaussiana, i modelli lineari generalizzati (GLM) sono specifiche di modelli lineari in cui la variabile di risposta segue una distribuzione dalla famiglia di distribuzioni esponenziali. I GLM richiedono distribuzioni familiari esponenziali che possono essere scritte nella loro forma "canonica" o "naturale", ovvero distribuzioni di famiglia esponenziali naturali, come: Gaussiana, Binomiale, Poisson, Gamma, Tweedie.

La funzione di collegamento fornisce la relazione tra il predittore lineare e la media della funzione di distribuzione. Esistono molte funzioni di collegamento comunemente utilizzate e la loro scelta è influenzata da diverse considerazioni. Esiste sempre una funzione di collegamento canonico ben definita derivata dall'esponenziale della funzione di densità della risposta. Tuttavia, in alcuni casi ha senso cercare di far corrispondere il dominio della funzione di collegamento all'intervallo della media della funzione di distribuzione o utilizzare una funzione di collegamento non canonico per scopi algoritmici.

L'interfaccia di regressione lineare generalizzata di Spark fornisce anche statistiche riassuntive per diagnosticare l'adattamento dei modelli GLM, inclusi residui, p-value, deviazioni, il criterio di informazione di Akaike e altri.

Decision Tree Regression. Questo algoritmo osserva le caratteristiche di un oggetto e forma un modello nella struttura di un albero per prevedere i dati futuri per produrre un output continuo significativo. L'output continuo significa che l'output / risultato non è discreto, cioè non è rappresentato solo da un insieme discreto noto di numeri o valori, come ad esempio un modello di predizione del profitto che indica il probabile profitto che può essere generato dalla vendita di un prodotto.

Random Forest Regression. Le Random Forest costruiscono più alberi decisionali e uniscono le loro predizioni insieme per ottenere una predizione più accurata e stabile

anziché affidarsi a singoli alberi decisionali. Ogni albero in una foresta casuale apprende da un campione casuale delle osservazioni di addestramento. I campioni sono selezionati per sostituzione (*replacement*), nota come *bootstrap*, il che significa che alcuni campioni verranno utilizzati più volte in un singolo albero. L'idea è che addestrando ciascun albero su diversi campioni, anche se ogni albero potrebbe avere un'alta varianza rispetto ad un particolare insieme di dati di addestramento, nel complesso, l'intera foresta avrà una varianza inferiore ma non al costo di aumentare il bias. Questa maggiore diversità nella foresta porta a predizioni complessive più solide. Quando arriva il momento di fare una predizione, il modello di regressione di Random Forest prende la media di tutte le stime dei singoli alberi di decisione. Tra i vantaggi di questo modello abbiamo: una riduzione dell'overfitting; la relativa importanza di misurare ogni feature nella predizione; bassa propensione di feature engineering; si basa su poche ipotesi statistiche; non presuppone che i dati siano normalmente distribuiti, che la relazione sia lineare o che si abbia specificato le interazioni.

Gradient-boosted Tree Regression. L'algoritmo per Boosting Trees si è evoluto dall'applicazione dei metodi di potenziamento agli alberi di regressione. L'idea generale è di calcolare una sequenza di alberi (molto) semplici, in cui ogni albero successivo è costruito per i residui di predizione dell'albero precedente. Questo metodo costruirà alberi binari, ovvero partiziona i dati in due campioni su ciascun nodo di divisione. Supponiamo ora di limitare la complessità degli alberi solo a 3 nodi: un nodo radice e due nodi figli, cioè una singola divisione. Pertanto, ad ogni passo del boosting (algoritmo boosting trees), viene determinato un partizionamento (migliore) semplice dei dati, e vengono calcolate le deviazioni dei valori osservati dalle rispettive medie (residui per ogni partizione). Il successivo albero a 3 nodi verrà quindi adattato a quei residui, per trovare un'altra partizione che ridurrà ulteriormente la varianza residua (errore) per i dati, data la precedente sequenza di alberi. Si può dimostrare che tali "additive weighted expansions" di alberi possono alla fine produrre un eccellente adattamento dei valori previsti ai valori osservati, anche se la natura specifica delle relazioni tra le variabili predittive e la variabile dipendente di interesse è molto complessa (non lineare in natura). Quindi, il metodo di Gradient Boosting, adattando un'espansione additiva

ponderata di alberi semplici, rappresenta un algoritmo di apprendimento automatico molto generale e potente.

Survival Regression. Spesso abbiamo dati aggiuntivi oltre alla durata che vogliamo utilizzare. La tecnica è chiamata regressione di sopravvivenza: il nome implica che regrediamo le covariate (ad esempio, età, paese, ecc.) rispetto a un'altra variabile, in questo caso le durate. Non possiamo usare metodi tradizionali come la regressione lineare a causa della censura. Ci sono alcuni modelli popolari nella regressione di sopravvivenza: il modello di Cox, i modelli di cedimento accelerato (accelerated failure time models, AFT) e il modello addizionale di Aalen. Tutti i modelli tentano di rappresentare la frequenza di pericolo $h(t|x)$ in funzione di t e di alcune covariate x.

In Spark viene implementato il modello AFT che è un modello parametrico di regressione di sopravvivenza per i dati censurati. Descrive un modello per il registro del tempo di sopravvivenza, quindi è spesso chiamato un modello log-lineare per l'analisi di sopravvivenza. Diverso da un modello di rischio proporzionale progettato per lo stesso scopo, il modello AFT è più facile da parallelizzare perché ogni istanza contribuisce alla funzione obiettivo in modo indipendente.

Isotonic Regression. La regressione isotonica (o regressione monotona) è la tecnica di adattare una linea a forma libera ad una sequenza di osservazioni sotto i seguenti vincoli: la linea a forma libera adattata deve essere non-decrescente (o non crescente) ovunque, e deve essere il più vicino possibile alle osservazioni.

La regressione isotonica ha applicazioni in statistica inferenziale. Ad esempio, si potrebbe usarla per adattare una curva isotonica alle medie di alcuni set di risultati sperimentali quando è previsto un aumento di tali medie in base ad alcuni ordinamenti particolari. Un vantaggio della regressione isotonica è che non è vincolata da alcuna forma funzionale, come la linearità imposta dalla regressione lineare, purché la funzione sia in aumento monotono.

5.6.2.1 *Fattori di valutazione degli algoritmi di regressione in apprendimento automatico supervisionato*

Nel precedente paragrafo sono stati forniti delle brevi descrizioni degli algoritmi recuperate da fonti scientifiche. Da approfondimenti e una serie di valutazioni sono emerse delle forme di classificazioni degli stessi algoritmi derivanti da una mera osservazione, diretta o indiretta, dei risultati ottenuti a seguito di prove empiriche. Tali risultati hanno portato all'identificazione di aree afferenti ad alcuni domini nei quali l'algoritmo ha fornito risultati migliori rispetto ad altri. Questo modello di classificazione porta a definire una metodologia per l'identificazione di fattori discriminanti tra gli algoritmi di regressione e il loro impiego. Il nostro obiettivo, infatti, è quello di poter suggerire a priori quali potrebbero essere gli algoritmi migliori a seconda di alcuni fattori, quali contesto operativo, la semplicità di utilizzo o la propensione ad ottenere ottime performance sulla base della complessità dei dati da addestrare. Questa analisi ci ha portato a studiare ulteriormente tali fattori, in modo che da essi noi possiamo esplicitare una conoscenza latente e fornire quindi un servizio di raccomandazione in grado di suggerire una classifica ordinata di algoritmi, dal più rilevante al meno rilevante, che di fatto genera un valore per l'esperto di dominio che si ritrova nella fase di primo approccio al problema di machine learning che gli si pone davanti.

Di seguito, le informazioni precedentemente descritte per ogni metodo di regressione vengono generalizzate e condivise, in modo da indicare su quali parametri, derivati dalle informazioni legate agli algoritmi di regressione, si può già determinare una preliminare "qualità" di scelta e ranking.

NFL	LinR	GLM	DT Reg	RF Reg	GBT Reg	SurR	IsoR
Tra	yes	no	yes	no	no	yes	yes
Pre	yes	yes	yes	no	yes	yes	yes
Big	no	no	yes	yes	yes	no	no
Inc	yes	yes	no	no	no	yes	yes
Sma	yes	yes	yes	no	no	yes	yes
Cat	no	no	yes	yes	yes	no	no
Mul	*no*	*no*	*yes*	*yes*	*yes*	*no*	*no*
Fel	no	no	yes	yes	yes	no	no
Mis	yes	yes	yes	yes	yes	yes	yes
Out	no	yes	yes	yes	no	no	no
Ove	no	yes	no	yes	yes	no	no
Red	yes	yes	no	no	no	yes	yes
Std	yes	yes	yes	yes	yes	yes	yes
Dim	no	yes	yes	yes	yes	no	no
Int	yes	no	yes	no	no	yes	yes
Hyp	yes	yes	yes	yes	no	yes	yes
Eng	no	no	yes	yes	no	no	no

Informazioni legate agli algoritmi di Regressione

Il Teorema "No Free Lunch" (NFL) [79] afferma che non esiste un modello che funzioni meglio per ogni problema. Le ipotesi di un modello ottimale per un problema potrebbero non reggere per un altro problema, quindi è comune in machine learning provare più modelli e trovarne uno che funzioni meglio per un particolare problema. Questo è particolarmente vero nell'apprendimento supervisionato, in cui la validation e la cross-validation sono comunemente utilizzate per valutare l'accuratezza predittiva di più modelli di varia complessità per trovare il modello migliore. Si noti che un modello che funziona bene potrebbe anche essere formato da più algoritmi. A seconda del problema, è importante valutare i compromessi tra velocità, accuratezza e complessità di diversi modelli e algoritmi e trovare un modello che funzioni meglio per quel particolare problema. A partire da queste assunzioni, e quelle derivate dallo studio di ogni algoritmo di regressione che hanno portato alla generazione della **Tabella 1**, è possibile definire un ranking iniziale degli algoritmi semplicemente contando il numero di caratteristiche che ogni algoritmo possiede. Questo fattore iniziale prenderà quindi il nome di "Fattore No Free Lunch". Fermo restando quanto esposto che è una ipotesi chiara, nulla però vieta nel pensare comunque che quello utilizzato e inizialmente quotato come "modello ottimale" sia una sola parvenza a seguito della mancanza di valutazione di qualche altro modello mai preso in ipotesi e quindi mai utilizzato. Da qui nasce infatti il nostro approfondimento come modo per definire un metodo di valutazione degli algoritmi di regressione, che di fatto determina già una prima valutazione discriminante.

Secondo fattore derivante dal teorema NFL è che, a parità di condizioni, si parta con la scelta dell'algoritmo di regressione più semplice. Il **Rasoio di Occam** segue esattamente questa regola: è un principio di risoluzione dei problemi che afferma essenzialmente che "le soluzioni più semplici hanno maggiori probabilità di essere corrette rispetto a quelle complesse" [80]. Tale principio può essere quindi inglobato nel nostro studio e possiamo determinare un fattore di semplicità da inglobare nel nostro sistema di raccomandazione, distinguendosi come fattore derivante altresì da uno studio empirico,

ma in una veste originale e concorrente al nostro scopo. Quindi ad ogni algoritmo viene assegnato un "Fattore Rasoio di Occam" che indica il proprio grado di semplicità, in una scala da 1 a 10, dove 10 indica il massimo della semplicità.

Occ	LinR	GLM	DT Reg	RF Reg	GBT Reg	SurR	IsoR
Fattore Rasoio di Occam	9	5	7	6	6	5	5

Terzo fattore discriminante è quello di **complessità dei dati addestrabili**. Se si ha a che fare con dataset di regressione di informazioni complesse, come immagini, video, segnali audio, è noto che gli algoritmi classici di machine learning (ovvero quelli della libreria Spark MLlib) non siano in grado di gestirli totalmente o di ottenere risultati accettabili quanto gli algoritmi di Deep Learning. Possiamo sfruttare questa conoscenza empirica e determinare quindi un altro fattore da inglobare nel nostro recommender. Quindi, il "Fattore Training di Dati Complessi" assegna ad ogni algoritmo un valore che indica la sua attitudine a gestire dataset con questo tipo di informazioni complesse.

Com	LinR	GLM	DT Reg	RF Reg	GBT Reg	SurR	IsoR
Fattore Training di Dati Complessi	3	6	5	6	7	5	5

Quarto fattore decisivo per dalle informazioni derivanti dal **Contesto Operativo**. I contesti operativi di cui Digital Future avrà conoscenza di dominio sono tre:

- Fabbrica Intelligence (FI)
- Healthcare (HC)
- Sicurezza ambientale e controllo del territorio (SACT)

A seconda dell'analisi condotta precedentemente sulle informazioni rilevanti e il contesto in cui gli algoritmi di regressione si sono distinti in letteratura, abbiamo condotto uno studio di efficacia degli algoritmi. La nostra idea è quella di determinare un fattore di probabilità di efficacia di ogni algoritmo per il contesto operativo di riferimento. Si definiscono pertanto le seguenti stime percentuali di efficacia di ogni algoritmo di regressione.

CO	LinR	GLM	DT Reg	RF Reg	GBT Reg	SurR	IsoR
FI	100	90	100	100	90	50	50
HC	90	100	70	80	90	50	50
SACT	50	50	70	80	90	50	50

Nonostante l'ampia gamma di soluzioni già presenti in Digital Future, per eseguire un task di regressione l'esperto tecnico del dominio potrebbe voler ricevere dei suggerimenti su altri algoritmi di regressione che, a seconda del contesto operativo, risultino essere efficaci oltre che essere una novità nello stato dell'arte della letteratura e nell'innovazione scientifico-tecnologica. A tal fine, un altro componente della Piattaforma DF, come il Mediator, potrebbe occuparsi di questo, ossia implementare funzionalità di scouting tecnologico e mettere a disposizione del componente "Algorithm Recommender" suggerimenti di una classifica ordinata di metodi di apprendimento automatico supervisionato di classificazione. In particolare, per ogni nuovo algoritmo che il Mediator potrebbe individuare come rilevante per il contesto operativo di riferimento, si potrebbe, ad esempio, valutare i seguenti parametri:

- Accuratezza su dati dello stesso contesto operativo (**Acc**)
- Numero Citazioni Paper scientifico di riferimento (**Cit**)

- h-Index del primo autore (**H1**)
- h-Index medio degli autori (**Havg**)
- Disponibilità codice online (**Ava**)
- Numero di revisioni/aggiornamenti dell'algoritmo (**Rev**)
- Data ultima revisione/aggiornamento (**Upd**)

5.6.2.2 *Ranking Recommender per metodi di regressione in apprendimento automatico supervisionato*

Considerati i seguenti fattori:

1. Fattore "Rasoio di Occam" (**Occ**)
2. Fattore "Training di Dati Complessi" (**Com**)
3. Fattore "No Free Lunch" (**NFL**)
4. Fattore "Contesto Operativo" (**CO**)
5. Fattore "Scouting Tecnologico" (**ST**)

Viene proposta di seguito la formula che assegna un valore numerico ad ogni metodo di regressione discusso precedentemente, in modo tale da poter restituire successivamente una classifica, ovvero un ranking ordinato, degli algoritmi di regressione nell'ambito dell'apprendimento automatico supervisionato.

Definizione 3. (Regressor Recommender Evaluation Function) Sia A l'insieme degli algoritmi di regressione e $a \in A$ un algoritmo di tale insieme, e sia $r: A \rightarrow \mathbb{R}$ una funzione che mappa ogni elemento di A ad un numero reale, allora, $\forall a \in A$:

$$r(a) = (Occ(a) + Com(a) + NFL(a)) \frac{CO(a)}{100} + ST(a)$$

dove:

- $Occ: A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Rasoio di Occam";
- $Com: A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Training di Dati Complessi";
- $NFL: A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "No Free Lunch";

- $CO : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Contesto Operativo";
- $ST : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Scouting Tecnologico".

Quindi, la funzione $r(\cdot)$ restituisce un valore numerico che può essere ordinato totalmente tra tutti gli algoritmi di regressione. Tale valore è il risultato della combinazione dei vari fattori descritti precedentemente che consentono di valutare le peculiarità di ogni algoritmo di regressione e propensione ad ottenere delle performance migliori a seconda del contesto operativo, considerando anche l'apporto del Mediator come un "bonus" in cui a risultare tra i primi posti in classifica potrebbe esserci un algoritmo di regressione non precedentemente considerato tra quelli già previsti nella componente Machine Learning Libraries del Data Governance di Digital Future.

Definizione 4. (Regressor Algorithm Recommender Ranking) Sia A l'insieme degli algoritmi di regressione e $a \in A$ un algoritmo di tale insieme, e sia $r : A \rightarrow \mathbb{R}$ la funzione di valutazione su A, allora l'Algorithm Recommender Ranking associa ad A un ranking $\succeq_A^r$ su A tale che $\forall a, b \in A,\ a \succeq_A^r b$ se e solo se $r(a) \geq r(b)$.

In questo modo l'ordinamento totale su A genera una classifica ordinata a seconda delle specifiche richieste del problema da analizzare.

5.6.3 *Recommender di metodi di Clustering in Machine Learning*

Nell'apprendimento non supervisionato abbiamo meno informazioni sugli oggetti, in particolare se il dataset di addestramento non è etichettato. L'obiettivo diventa dunque quello di osservare alcune somiglianze tra gruppi di oggetti e includerli in cluster appropriati. Alcuni oggetti possono differire enormemente da tutti i cluster, in questo modo assumiamo che questi oggetti siano anomali.

La nozione di un cluster, come rilevato da diversi algoritmi, varia significativamente nelle sue proprietà. Comprendere questi "modelli di cluster" è la chiave per capire le differenze tra i vari algoritmi. I modelli di cluster tipici includono:

- *Modelli di connettività*: ad esempio, il *clustering gerarchico* crea modelli basati sulla connettività a distanza.
- *Modelli a centroidi*: ad esempio, l'algoritmo K-means rappresenta ciascun cluster per un singolo vettore medio.
- *Modelli di distribuzione*: i cluster sono modellati utilizzando distribuzioni statistiche, come le distribuzioni normali multivariate utilizzate dall'algoritmo di massimizzazione delle aspettative (EM).
- *Modelli di densità*: ad esempio, DBSCAN e OPTICS definiscono i cluster come regioni dense collegate nello spazio dei dati.
- *Modelli di sottospazio*: in biclustering (noto anche come co-clustering o clustering a due modalità), i cluster sono modellati sia con i membri del cluster che con gli attributi pertinenti.
- *Modelli di gruppo*: alcuni algoritmi non forniscono un modello raffinato per i loro risultati e forniscono semplicemente le informazioni di raggruppamento.
- *Modelli basati su grafi*: una cricca, cioè un sottoinsieme di nodi in un grafo tale che ogni due nodi nel sottoinsieme sono collegati da un arco può essere considerato come una forma prototipo di cluster. I rilassamenti del requisito di connettività completo (una frazione degli archi può mancare) sono noti come quasi-clique, come nell'algoritmo di clustering HCS.
- *Modelli di grafici firmati*: ogni tracciato in un grafo firmato ha un segno dal prodotto dei segni sugli archi. Sotto l'ipotesi della teoria del bilanciamento, gli archi possono cambiare segno e risultare in un grafo biforcato. Il più debole "assioma di clusterability" (nessun ciclo ha esattamente un margine negativo) produce risultati con più di due cluster, o sottografi con solo margini positivi.
- *Modelli neurali*: la rete neurale non supervisionata più conosciuta è la Self-Organizing Map (SOM) e questi modelli possono solitamente essere caratterizzati come simili a uno o più dei modelli sopra citati, e inclusi i modelli subspaziali quando le reti neurali implementano una forma di analisi di Principal Component Analysis (PCA) o Independent Component Analysis.

Ai fini di produrre una classifica ordinata per tale categoria di algoritmi, in maniera generica si possono considerare le seguenti **informazioni legate agli algoritmi**, ovvero le caratteristiche (più o meno) comuni a tutti i metodi di clustering più noti in letteratura —e non solo— di apprendimento automatico [[60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76]], fruibili dalla componente di Machine Learning Libraries del Data Governance Layer del Middleware, che possono discriminare la scelta e l'efficacia di un metodo rispetto ad un altro:

- Capacità di avere modelli scalabili (**Big**)
- Capacità di lavorare con diversi tipi di attributi (**Att**)
- Capacità di scoprire cluster con forme arbitrarie (**Sha**)
- Robustezza agli outliers (**Out**)
- Insensibilità all'ordine dei record di input (**Ord**)
- Capacità di lavorare ad alta dimensionalità (**Hig**)
- Predisposizione all'interpretabilità (**Int**)
- Possibilità di non definire a priori il numero di cluster (**NoK**)
- Possibilità di non definire partizionamenti stretti (**Stri**)
- Capacità di lavorare con un piccolo dataset (**Sma**)
- Robustezza all'overfitting (**Ove**)

Eventuali vincoli di scelta sono legati alla capacità computazionale (computational overhead) o di storage del sistema/workstation.

Nello specifico, tra i metodi di clustering che andremo a considerare tra quelli implementabili in Digital Future, troviamo quelli disponibili dalla libreria di Machine Learning di Spark MLlib [77] e Scikit-learn [81]:

- K-Means (**K-M**)
- Latent Dirichlet Allocation (**LDA**)
- Bisecting K-means (**BKM**)
- Gaussian Mixture Model (**GMM**)
- Density-Based Spatial Clustering of Applications with Noise (**DBSCAN**) [82]

Di seguito, quindi, si riporta una breve descrizione dei punti di forza e di debolezza di ogni metodo di clustering appena introdotto, indicando ove possibile le situazioni in cui in genere si ottengono dei buoni risultati.

K-means. Il K-means divide un insieme di oggetti in K gruppi sulla base dei loro attributi. Si assume che gli attributi degli oggetti possano essere rappresentati come vettori, e che quindi formino uno spazio vettoriale. L'obiettivo che l'algoritmo si prepone è di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centroide o punto medio. L'algoritmo segue una procedura iterativa. Inizialmente crea K partizioni e assegna ad ogni partizione i punti d'ingresso o casualmente o usando alcune informazioni euristiche. Quindi calcola il centroide di ogni gruppo. Costruisce quindi una nuova partizione associando ogni punto d'ingresso al cluster il cui centroide è più vicino ad esso. Quindi vengono ricalcolati i centroidi per i nuovi cluster e così via, finché l'algoritmo non converge.

Il K-means ha acquistato notorietà dato che converge molto velocemente. Infatti, si è osservato che generalmente il numero di iterazioni è minore del numero di punti. Comunque, l'algoritmo può essere molto lento nel caso peggiore. In termini di qualità delle soluzioni, l'algoritmo non garantisce il raggiungimento dell'ottimo globale. La qualità della soluzione finale dipende largamente dal set di cluster iniziale e può, in pratica, ottenere una soluzione ben peggiore dell'ottimo globale. Dato che l'algoritmo è di solito estremamente veloce, è possibile applicarlo più volte e fra le soluzioni prodotte scegliere quella più soddisfacente. Il più grande svantaggio è che il K-Means deve sapere in anticipo quanti cluster ci saranno nei tuoi dati, quindi questo potrebbe richiedere un sacco di prove per "indovinare" il miglior numero K di cluster da definire.

Latent Dirichlet Allocation. Il modello **LDA** è un modello generativo, utilizzato nello studio del linguaggio naturale, che consente di estrarre argomenti da un insieme di documenti di origine e di fornire una spiegazione logica sulla somiglianza di singole parti dei documenti. Ogni documento è considerato come un insieme di parole che, combinate tra loro, formano uno o più sottoinsiemi di argomenti latenti. Ciascun argomento (*topic*) è caratterizzato da una particolare distribuzione di termini. Il processo

generativo della LDA si basa sull'analisi dei dati contenuti nel testo. Le combinazioni di parole sono considerate come variabili casuali. Ad ogni topic viene associata una distribuzione di parole. Successivamente, ad ogni documento viene trovata una distribuzione di argomenti, poi, per ogni parola del documento, verifica la sua attribuzione a un argomento del documento e a una distribuzione di parole dell'argomento.

A seconda della tipologia di inferenza, l'algoritmo LDA consente di raggiungere un determinato livello di efficacia e di costo (efficienza) in termini di complessità temporale e spaziale.

Bisecting K-means. Il bisecting K-means è una tipo di *clustering gerarchico* che utilizza un approccio divisivo (o top-down): tutte le osservazioni iniziano in un cluster e le suddivisioni vengono eseguite in modo ricorsivo mentre ci si sposta verso il basso della gerarchia. L'algoritmo può spesso essere molto più veloce del normale K-means, ma generalmente produce un clustering differente.

Gaussian Mixture Model. Una **GMM** rappresenta una distribuzione composita in cui i punti sono tracciati da una delle *k* sottodistribuzioni gaussiane, ciascuna con la propria probabilità.

Qui, il dataset viene solitamente modellato con un numero fisso (per evitare overfitting) di distribuzioni gaussiane che sono inizializzate a caso e i cui parametri sono iterativamente ottimizzati per adattarsi meglio al dataset. Questo convergerà in un ottimo locale, quindi più esecuzioni potrebbero produrre risultati diversi. Per ottenere un cluster rigido, gli oggetti vengono spesso assegnati alla distribuzione gaussiana a cui probabilmente appartengono. Il clustering basato sulla distribuzione produce modelli complessi per cluster in grado di catturare la correlazione e la dipendenza tra gli attributi. Tuttavia, questi algoritmi mettono un ulteriore onere per l'utente: per molti dataset reali, potrebbe non esserci un modello matematico definito in modo conciso (per esempio assumendo che le distribuzioni gaussiane siano un'ipotesi piuttosto forte sui dati).

Density-Based Spatial Clustering of Applications with Noise. L'algoritmo **DBSCAN** è basato sulla densità, in maniera simile all'algoritmo *Mean-Shift* (basato su sliding-window nel tentativo di trovare aree dense di punti di dati), ma con un paio di notevoli vantaggi: non richiede un numero *k* pre-definito, identifica gli outlier come rumori ed è in grado di trovare facilmente cluster di dimensioni arbitrarie e di forma arbitraria.

Lo svantaggio principale di DBSCAN è che non ha prestazioni altrettanto buone rispetto ad altri quando i cluster sono di densità variabile.

5.6.3.1 *Fattori di valutazione degli algoritmi di clustering in apprendimento automatico non supervisionato*

Nel precedente paragrafo sono stati forniti delle brevi descrizioni degli algoritmi recuperate da fonti scientifiche. Da approfondimenti e una serie di valutazioni sono emerse delle forme di classificazioni degli stessi algoritmi derivanti da una mera osservazione, diretta o indiretta, dei risultati ottenuti a seguito di prove empiriche. Tali risultati hanno portato all'identificazione di aree afferenti ad alcuni domini nei quali l'algoritmo ha fornito risultati migliori rispetto ad altri. Tali risultati hanno portato all'identificazione di aree afferenti ad alcuni domini nei quali l'algoritmo ha fornito risultati migliori rispetto ad altri. Questo modello di classificazione porta a definire una metodologia per l'identificazione di fattori discriminanti tra gli algoritmi di clustering e il loro impiego. Il nostro obiettivo, infatti, è quello di poter suggerire a priori quali potrebbero essere gli algoritmi migliori a seconda di alcuni fattori, quali contesto operativo, la semplicità di utilizzo o la propensione ad ottenere ottime performance sulla base della complessità dei dati da addestrare. Questa analisi ci ha portato a studiare ulteriormente tali fattori, in modo che da essi noi possiamo esplicitare una conoscenza latente e fornire quindi un servizio di raccomandazione in grado di suggerire una classifica ordinata di algoritmi, dal più rilevante al meno rilevante, che di fatto genera un valore per l'esperto di dominio che si ritrova nella fase di primo approccio al problema di machine learning che gli si pone davanti.

Di seguito, le informazioni precedentemente descritte per ogni metodo di clustering vengono generalizzate e condivise, in modo da indicare su quali parametri, derivati dalle

informazioni legate agli algoritmi di clustering, si può già determinare una preliminare "qualità" di scelta e ranking.

NFL	K-M	LDA	BKM	GMM	DBSCAN
Big	Yes	Yes	Yes	Yes	Yes
Att	No	Yes	Yes	Yes	Yes
Sha	No	No	Yes	Yes	Yes
Out	No	No	Yes	No	Yes
Ord	No	Yes	No	No	No
Hig	No	Yes	No	Yes	No
Int	Yes	No	No	Yes	Yes
NoK	No	No	No	No	Yes
Stri	No	No	No	Yes	No
Sma	Yes	Yes	Yes	Yes	Yes
Ove	No	No	No	Yes	Yes

Informazioni legate agli algoritmi di Clustering

Il Teorema "No Free Lunch" (NFL) [79] afferma che non esiste un modello che funzioni meglio per ogni problema. Le ipotesi di un modello ottimale per un problema potrebbero non reggere per un altro problema, quindi è comune in machine learning provare più modelli e trovarne uno che funzioni meglio per un particolare problema. A

seconda del problema, è importante valutare i compromessi tra velocità, accuratezza e complessità di diversi modelli e algoritmi e trovare un modello che funzioni meglio per quel particolare problema. A partire da queste assunzioni, e quelle derivate dallo studio di ogni algoritmo di clustering che hanno portato alla generazione della **Tabella 1**, è possibile definire un ranking iniziale degli algoritmi semplicemente contando il numero di caratteristiche che ogni algoritmo possiede. Questo fattore iniziale prenderà quindi il nome di "Fattore No Free Lunch". Fermo restando quanto esposto che è una ipotesi chiara, nulla però vieta nel pensare comunque che quello utilizzato e inizialmente quotato come "modello ottimale" sia una sola parvenza a seguito della mancanza di valutazione di qualche altro modello mai preso in ipotesi e quindi mai utilizzato. Da qui nasce infatti il nostro approfondimento come modo per definire un metodo di valutazione degli algoritmi di clustering, che di fatto determina già una prima valutazione discriminante.

Secondo fattore derivante dal teorema NFL è che, a parità di condizioni, si parta con la scelta dell'algoritmo di clustering più semplice. Il **Rasoio di Occam** segue esattamente questa regola: è un principio di risoluzione dei problemi che afferma essenzialmente che "le soluzioni più semplici hanno maggiori probabilità di essere corrette rispetto a quelle complesse" [80]. Tale principio può essere quindi inglobato nel nostro studio e possiamo determinare un fattore di semplicità da inglobare nel nostro sistema di raccomandazione, distinguendosi come fattore derivante altresì da uno studio empirico, ma in una veste originale e concorrente al nostro scopo. Quindi ad ogni algoritmo viene assegnato un "Fattore Rasoio di Occam" che indica il proprio grado di semplicità, in una scala da 1 a 10, dove 10 indica il massimo della semplicità.

Occ	K-M	LDA	BKM	GMM	DBSCAN
Fattore Rasoio di Occam	9	5	7	6	6

Terzo fattore discriminante è quello di **complessità dei dati addestrabili**. Se si ha a che fare con dataset di clustering di informazioni complesse, come immagini, video, segnali audio, è noto che gli algoritmi classici di machine learning (ovvero quelli della libreria Spark MLlib) non siano in grado di gestirli totalmente o di ottenere risultati accettabili quanto gli algoritmi di Deep Learning. Possiamo sfruttare questa conoscenza empirica e determinare quindi un altro fattore da inglobare nel nostro recommender. Quindi, il "Fattore Training di Dati Complessi" assegna ad ogni algoritmo un valore che indica la sua attitudine a gestire dataset con questo tipo di informazioni complesse.

Com	K-M	LDA	BKM	GMM	DBSCAN
Fattore Training di Dati Complessi	3	6	5	6	7

Quarto fattore decisivo per dalle informazioni derivanti dal **Contesto Operativo**. I contesti operativi di cui Digital Future avrà conoscenza di dominio sono tre:

- Fabbrica Intelligence (FI)
- Healthcare (HC)
- Sicurezza ambientale e controllo del territorio (SACT)

A seconda dell'analisi condotta precedentemente sulle informazioni rilevanti e il contesto in cui gli algoritmi di clustering si sono distinti in letteratura, abbiamo condotto uno studio di efficacia degli algoritmi. La nostra idea è quella di determinare un fattore di probabilità di efficacia di ogni algoritmo per il contesto operativo di riferimento. Si definiscono pertanto le seguenti stime percentuali di efficacia di ogni algoritmo di clustering.

CO	K-M	LDA	BKM	GMM	DBSCAN
FI	90	50	80	70	90
HC	90	60	80	80	90
SACT	80	50	70	80	90

Nonostante l'ampia gamma di soluzioni già presenti in Digital Future, per eseguire un task di clustering l'esperto tecnico del dominio potrebbe voler ricevere dei suggerimenti su altri algoritmi di clustering che, a seconda del contesto operativo, risultino essere efficaci oltre che essere una novità nello stato dell'arte della letteratura e nell'innovazione scientifico-tecnologica. A tal fine, un altro componente della Piattaforma DF, come il Mediator, potrebbe occuparsi di questo, ossia implementare funzionalità di scouting tecnologico e mettere a disposizione del componente "Algorithm Recommender" suggerimenti di una classifica ordinata di metodi di apprendimento automatico supervisionato di classificazione. In particolare, per ogni nuovo algoritmo che il Mediator potrebbe individuare come rilevante per il contesto operativo di riferimento, si potrebbe, ad esempio, valutare i seguenti parametri:

- Accuratezza su dati dello stesso contesto operativo (**Acc**)
- Numero Citazioni Paper scientifico di riferimento (**Cit**)
- h-Index del primo autore (**H1**)
- h-Index medio degli autori (**Havg**)
- Disponibilità codice online (**Ava**)
- Numero di revisioni/aggiornamenti dell'algoritmo (**Rev**)
- Data ultima revisione/aggiornamento (**Upd**)

5.6.3.2 *Ranking Recommender per metodi di clustering in apprendimento automatico non supervisionato*

Considerati i seguenti fattori:

- Fattore "Rasoio di Occam" (**Occ**)
- Fattore "Training di Dati Complessi" (**Com**)
- Fattore "No Free Lunch" (**NFL**)
- Fattore "Contesto Operativo" (**CO**)
- Fattore "Scouting Tecnologico" (**ST**)

Viene proposta di seguito la formula che assegna un valore numerico ad ogni metodo di clustering discusso precedentemente, in modo tale da poter restituire successivamente una classifica, ovvero un ranking ordinato, degli algoritmi di clustering nell'ambito dell'apprendimento automatico non supervisionato.

Definizione 5. (Clustering Recommender Evaluation Function) Sia A l'insieme degli algoritmi di clustering e $a \in A$ un algoritmo di tale insieme, e sia $r: A \rightarrow \mathbb{R}$ una funzione che mappa ogni elemento di A ad un numero naturale, allora, $\forall a \in A$:

$$r(a) = (Occ(a) + Com(a) + NFL(a)) \frac{CO(a)}{100} + ST(a)$$

dove:

- $Occ : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Rasoio di Occam";
- $Com : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Training di Dati Complessi";
- $NFL : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "No Free Lunch";
- $CO : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Contesto Operativo";
- $ST : A \rightarrow \mathbb{N}$ indica la funzione che mappa il fattore "Scouting Tecnologico".

Quindi, la funzione $r(\cdot)$ restituisce un valore numerico che può essere ordinato totalmente tra tutti gli algoritmi di clustering. Tale valore è il risultato della combinazione dei vari fattori descritti precedentemente che consentono di valutare le peculiarità di ogni algoritmo di clustering e propensione ad ottenere delle performance migliori a seconda del contesto operativo, considerando anche l'apporto del Mediator come un "bonus" in cui a risultare tra i primi posti in classifica potrebbe esserci un

algoritmo di clustering non precedentemente considerato tra quelli già previsti nella componente Machine Learning Libraries del Data Governance di Digital Future.

Definizione 6. (Clustering Algorithm Recommender Ranking) Sia A l'insieme degli algoritmi di clustering e $a \in A$ un algoritmo di tale insieme, e sia $r : A \rightarrow \mathbb{R}$ la funzione di valutazione su A, allora l'Algorithm Recommender Ranking associa ad A un ranking $\succeq_A^r$ su A tale che $\forall a, b \in A,\ a \succeq_A^r b$ se e solo se $r(a) \geq r(b)$.

In questo modo l'ordinamento totale su A genera una classifica ordinata a seconda delle specifiche richieste del problema da analizzare.

5.7 Data Storage

Questo componente rappresenta il livello di persistenza del layer di Data Governance. Dovrà consentire la conservazione di dati provenienti dal campo, generalmente dati sensoriali, dati storici da utilizzare per l'addestramento dei modelli di apprendimento automatico, pipeline di analisi dei dati, modelli addestrati all'interno della piattaforma DF o modelli addestrati fuori della piattaforma ma da utilizzare al suo interno per fare inferenza sui dati che vengono caricati in essa.

L'accesso al layer di Storage sarà consentito solo agli utenti autorizzati. Questo layer sarà costituito da strumenti di storage con elevata scalabilità in grado di supportare un grande volume di operazioni di write. Il Layer sarà costituto principalmente da due componenti logici, come si evince nella figura sottostante:

- ML Model Store
- Structured & Unstructured Data Store

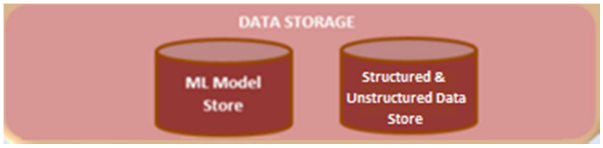


Figura 42: Data Storage Layer

5.7.1 *Structured & Unstructured Data Store*

Questo componente dovrà consentire di dare persistenza alle diverse tipologie di informazioni che potranno essere caricate sul Middleware della Piattaforma e consumati dai servizi messi a disposizione dalla stessa.

In generale esistono tre tipologie di dati:

- strutturati
- semi – strutturati
- non strutturati

I dati strutturati sono dati creati utilizzando uno schema predefinito e sono tipicamente organizzati in un formato tabellare. Basta pensare ad una tabella in cui ad ogni cella sia associato un valore specifico. Lo schema rappresenta l'infrastruttura, l'architettura che sottende all'organizzazione dei dati, l'intestazione della tabella utilizzata per descrivere valore e formato di ciascuna colonna. Lo schema inoltre impone i vincoli necessari per rendere i dati coerenti e calcolabili. Un database costituito da relazioni è un esempio perfetto di dato strutturato: le tabelle sono collegate usando ID univoci e un linguaggio di query come SQL per interagire con i dati.

I dati strutturati sono il modo migliore per fare interagire le macchine con le informazioni, al contrario dei dati *semi-strutturati* e *non strutturati*.

Spesso definiti come dati quantitativi, i dati strutturati sono fatti e numeri oggettivi rendendo i dati più facili da esportare, archiviare e organizzare in database tipici come Excel e SQL. Anche se i dati strutturati sono solo numeri o parole racchiusi in un database, è possibile estrarre facilmente informazioni dai dati strutturati eseguendoli attraverso metodi di analisi dei dati e strumenti come ad esempio l'analisi di regressione. Questo è l'aspetto più prezioso dei dati strutturati.

I **dati semi-strutturati** sono caratterizzati dall'assenza di una struttura rigida e formale. Contengono tipicamente dei tag o degli altri tipi di markup che separano il contesto testuale dagli elementi semantici. I dati semi-strutturati sono inoltre "autodescrittivi" (i

tag sono un buon esempio, lo schema fa parte dei dati e i dati si evolvono con il contenuto, ma mancano di coerenza).

I **dati non strutturati** possono trovarsi sotto forme differenti: dalle pagine web alle email, dai blog a i social media post, l'80% dei dati di cui disponiamo esiste sotto forma di dato non strutturato. Indipendentemente dal formato utilizzato per la loro archiviazione, stiamo parlando, per la maggior parte dei casi, di documenti testuali composti da sequenze di parole. Spesso definiti come dati qualitativi, i dati non strutturati sono di solito opinioni soggettive sotto forma di testo, che la maggior parte dei software di analisi non è in grado di raccogliere. Ciò rende difficile la raccolta, la memorizzazione e l'organizzazione dei dati non strutturati in database tipici come Excel e SQL. È anche difficile esaminare i dati non strutturati con metodi e strumenti di analisi dei dati standard. Poiché non è possibile archiviare e organizzare i dati non strutturati in database tipici, è necessario archiviarli in documenti Word o in database non relazionali come Elasticsearch o Solr, che possono eseguire query di ricerca per parole e frasi.

Nel contesto di utilizzo della Piattaforma Digital Future da parte dei Pillar, le tre tipologie di dati potranno essere persiste ed elaborate per finalità diverse. Infatti, già specificato nei paragrafi precedenti, il Data Governance Layer consentirà la elaborazione di dati storici e di dati a run time provenienti dal campo sia per addestrare modelli di apprendimento automatico e sia per fare inferenza su nuovi dati attraverso i modelli addestrati. Questi modelli potranno operare su: dati strutturati, ad esempio i dati di serie temporali, dati semi-strutturati, ad esempio nel caso di testo catturato attraverso tecniche di web scaping e dati non strutturati come l'analisi di documenti. Il tutto in sinergia con altri layer del Middleware DF come il Multimedia Manager Layer e il Semantic and Ontological Layer.

Il framework Apache Hadoop rappresenta lo stack tecnologico più adatto alle applicazioni distribuite che richiedono un elevato accesso ai dati. Uno dei componenti che dovrà essere utilizzato dalla piattaforma DF è l'Hadoop Distributed File System (HDFS), ovvero un file system distribuito, portabile e scalabile che sarà utilizzato per la

memorizzazione dei contenuti di pertinenza dei tenant che accederanno alla piattaforma. Se ad esempio si deve gestire un file le cui dimensioni superano quelle della memoria che si ha a disposizione Hadoop permette di archiviarlo "distribuendolo" su più server o su più nodi. Hadoop permette poi di gestire le funzioni di processing su questi dati distribuiti attraverso la funzione **MapReduce** che cambia le logiche di elaborazione dei dati, spostando la capacità applicativa del software a livello di dati con importanti vantaggi in termini di miglioramento delle prestazioni. MapReduce permette cioè di distribuire la capacità elaborativa così come HDFS permette di distribuire la capacità di archiviazione su diversi nodi. Con HDFS e MapReduce si può passare dalla elaborazione di tipo seriale a una elaborazione di tipo **parallelo.** Hadoop può, inoltre, gestire sia file strutturati e sia file destrutturati**.**

Nello specifico, per la piattaforma realizzata, ad ogni tenant sarà associato un workspace privato creato nel file system HDFS di Hadoop. Il workspace sarà composto da dalle seguenti cartelle:

- **data**: in questa cartella saranno memorizzati tutti i file CSV caricati dall'utente sulla piattaforma o generati tramite l'utilizzo degli strumenti della piattaforma
- **logs**: in questa cartella saranno salvati i file contenenti i log delle operazioni di import ed export di tabelle e di file CSV
- **multimedia**: in questa cartella saranno salvati i video che entreranno nella piattaforma tramite il modulo per l'ingestion delle immagini provenienti da telecamere
- **report**: in questa cartella saranno salvati i report prodotti durante le attività di analisi dei dati effettuate attraverso il componente Data Science Tool
- **scenari**: in questa cartella saranno salvati i file json creati per memorizzare l'intera struttura di uno scenario di analisi relativo relativo ad un servizio applicativo.

La tipologia di dati raccolti dai dispositivi IoT afferisce, generalmente ai dati di serie temporali, ossia un set di valori organizzati in base a criteri temporali. I dati di serie temporali potranno sia essere analizzati on fly per verificare le tendenze cronologiche, gli avvisi in tempo reale o per la modellazione predittiva e sia essere salvati per una

analisi successiva. In fase di salvataggio i dati verranno contrassegnati con un timestamp e inseriti nell'ordine di ricezione; solo raramente saranno aggiornati. Di conseguenza, il layer di persistenza dei dati dovrà essere sarà costituito da strumenti di storage con elevata scalabilità in grado di supportare elevati volumi di operazioni di write.

Questi dati provenienti da sensori e dispositivi di campo, potranno essere salvati in database NoSQL che vengono incontro alla necessità di gestire volumi di dati eccezionalmente grandi, con la necessità di avere dei tempi di risposta bassi anche con insiemi di dati enormi, pur mantenendo una elevata disponibilità del servizio. Queste problematiche sono difficilmente risolvibili attraverso un database relazionale a causa del modo in cui esso viene utilizzato nella maggior parte dei casi e cioè per gestire dati organizzati su tabelle normalizzate.

I principali vantaggi derivanti dall'uso di database NoSQL si possono riassumere in:

- Elevata velocità computazionale, anche al crescere del volume dei dati, legata alla mancanza di operazioni di aggregazione dei dati (no join) e ad una implementazione del database ottimizzata per la gestione di enormi quantità di dati; questa velocità è anche il frutto di alcune semplificazioni introdotte nella maggior parte dei database NoSQL: il mancato supporto delle transazioni ACID.
- Riduzione significativa dei tempi di sviluppo, grazie alla definizione di logiche di lettura dati molto più semplici rispetto a quelle da scrivere con database relazionali che operano su dati normalizzati (questo in realtà dipende dal fatto di dover denormalizzare i dati).
- Supporto per la scalabilità orizzontale, cioè la capacità di garantire tempi di risposta sostanzialmente invariati al crescere del carico, tramite l'aggiunta di nuovi server.
- Un elevato livello di disponibilità del servizio, anche grazie al supporto della data-replication (database distribuiti), più semplice da realizzare per i database NoSQL.
- Schemaless, cioè i database NoSQL sono privi di schema in quanto l'entità contiene tutti i campi necessari, senza necessità di una definizione formale e rigida del suo contenuto. In questo modo, è possibile arricchire le applicazioni di nuovi dati e informazioni, senza dover sottostare ad una rigida struttura dei dati.

All'interno della Piattaforma DF, sarà utilizzato Cassandra che un database NoSQL facente parte dell'ecosistema Apache Hadoop. Si tratta di un DBMS distribuito

sviluppato per gestire grandi quantità di dati dislocati in diversi server, fornendo inoltre un servizio orientato alla disponibilità, senza alcun point of failure.

In definitiva, i dati provenienti dai dispositivi esterni dovranno trovare persistenza all'interno di dataset organizzati attraverso il database NoSQL utilizzato dal Middleware di Piattaforma. L'operazione di persistenza dei dati dovrà essere contestuale al processo di Plug&Play dei dispositivi descritto nel Opportunistic Integrator Layer. Dovrà anche essere possibile creare dataset non collegati a dispositivi esterni e caricare in essi i dati contenuti in file testuali, ad esempio di tipo CSV, previo il loro caricamento nel file system distribuito che sarà messo a disposizione dal Middleware. Questa funzionalità consentirebbe, ad esempio di portare nella piattaforma dati storici che necessitano di essere utilizzati per il processo di training di modelli di apprendimento. Analogamente dovrà essere possibile effettuare l'export dei dati contenuti nei dataset persistiti nel database NoSQL per poi effettuarne il download in locale. L'operazione di download dovrà, in ogni caso, essere consentita per tutti gli oggetti conservati all'interno del file system distribuito nei workspace elencati precedentemente.

5.7.2 *ML Model Store*

Questa componente dovrà memorizzare i modelli ML addestrati, utilizzando il file system distribuito per dare persistenza agli oggetti che li rappresentano. Per cui si farà uso del file system HDFS del framework Apache Hadoop descritto precedentemente. Ad ogni tenant che accederà alla piattaforma, sarà associato un workspace privato in cui saranno salvati i modelli creati all'interno della Piattaforma DF attraverso il componente Data Science Tool, sia in Modalità Assistita che in Modalità Interactive Computing e i modelli creati esternamente alla piattaforma DF ma importati perché dovranno essere utilizzati al suo interno.

6. Multimedia Layer

Nell'ambito di un ecosistema in grado di fornire supporto trasversale ai differenti verticali di domini applicativi interconnessi con la piattaforma, diviene di fondamentale importanza l'introduzione della gestione di contenuti multimediali, sia come repository di un patrimonio comune di contenuti e sia come un manager di servizi avanzati per la loro fruizione.

Nell'architettura del Middleware è stata introdotta la componente denominata **Multimedia Manager**, nella quale saranno memorizzati e gestiti contenuti ad ampio raggio: da documenti testuali semplici e avanzati (come ad esempio PDF, documenti per l'automazione d'ufficio, fogli di calcolo e presentazioni multimediali) a contenuti propriamente multimediali come immagini, video, audio fino ad animazioni audio-visuali.

6.1 Funzionalità previste

Il Multimedia Layer dovrà mettere a disposizione una serie di funzionalità per l'acquisizione, gestione e fruizione di contenuti multimediali. Inizialmente si prevede una funzione che possa permettere ad un utente del Pillar di poter "caricare" nella piattaforma Digital Future un qualunque contenuto multimediale, tra quelli elencati precedentemente e, conseguentemente, di poterlo correttamente archiviare. Prima di procedere con l'archiviazione, si rende necessaria una fase di indicizzazione dei file e di annotazione semantica del contenuto, per cui il Multimedia Manager dovrà essere dotato di tali funzionalità. La funzione di indicizzazione si rende necessaria per poter ritrovare ed accedere a qualunque tipo di contenuto quasi istantaneamente. Per ritrovare un file memorizzato nella piattaforma, si dovrà considerare una funzione di ricerca di contenuti multimediali, così da consentirne la fruizione diretta. La funzione di ricerca si baserà su aspetti che consentiranno di operare con una selezione dinamica identificabile grazie all'utilizzo di informazioni semanticamente elaborate e informazioni legate al dominio di interesse. Questa funzione si baserà sostanzialmente su un matching semantico tra la richiesta elaborata e le informazioni elaborate in fase di memorizzazione del contenuto. Sarà prodotta una funzionalità avanzata di analisi semantica che permetterà di utilizzare la piattaforma Digital Future in maniera innovativa rispetto alle classiche funzionalità di gestione di file. In particolare si dovranno realizzare delle soluzioni applicative che permetteranno di effettuare una analisi semantica dei contenuti dei file caricati in piattaforma, grazie alla quale sarà possibile corredare gli stessi contenuti archiviati con degli specifici tag a loro volta

estrapolati dal contenuto stesso, sui quali avviare un processo di annotazione semantica. Questo consentirebbe, quindi, di poter procedere anche ad una classificazione del contenuto che a sua volta si dovrà basare su delle specifiche tassonomie. Questo processo permetterebbe di effettuare ricerche intelligenti, permettendo l'accesso a parti di interesse specifici a seconda del file archiviato: da estratti di documenti testuali a etichettature di contesto per immagini, fino a segmentazioni contestualizzate di file audio e/o video.

L'attività di ricerca e fruizione di un contenuto dovrà prevedere una componente di controllo dei permessi e dei diritti di visione dei contenuti a seconda dell'utente che ha effettuato l'accesso alla piattaforma Digital Future, onde evitare che tali contenuti siano utilizzati in maniera indiscriminata e priva di controlli. Infine, per permettere la fruizione di contenuti video per più utenti collegati alla piattaforma, si prevederanno soluzioni che possano eventualmente anche offrire modalità di visione in streaming, on demand o broadcasting.

6.2 Tecniche e Tecnologie

Per Realizzare quanto previsto per il Multimedia Layer un buon punto di partenza potrebbe essere quello di individuare soluzioni open source in grado di soddisfare, anche in parte, i requisiti del layer. La soluzione scelta andrebbe, poi, evoluta per raggiungere i requisiti definitivi. Tra i vari requisiti emerge l'interazione del Multimedia Layer con la componente semantica prevista all'interno del Middleware. Nasce, quindi, la necessità dell'ausilio di tecnologie e tecniche attinenti al mondo del web semantico.

Le caratteristiche della soluzione da individuare ed estendere o da realizzare partendo da zero, dovrebbe consentire la gestione di diverse tipologie di contenuti, fornendo servizi e controlli per la loro gestione. Sarebbe auspicabile che la soluzione sia dotata di un repository per consentire la persistenza dei contenuti, metadati, associazioni e indici di testo completi. Dal repository i contenuti dovrebbero poter essere accessibili ovunque siano necessari, con la possibilità di aggiungere un contesto ai contenuti, migliorando il processo decisionale, la produttività e la governance delle informazioni e rendendo la collaborazione all'interno e all'esterno dell'organizzazione più facile e sicura. Una soluzione che rispetta i macro requisiti suddetti è, ad esempio, la soluzione di content management open source denominata Alfresco[2].

[2] Alfresco Enterprise Content Manager: https://www.alfresco.com/it/

6.3 Architettura Logica del Layer

Il Layer in questione dovrà funzionalmente permettere di raccogliere varie tipologie di contenuti e di esporli in modo sicuro a vantaggio degli utilizzatori aventi diritto. A supporto, la componente esporrà i servizi descritti nel paragrafo precedente, alcuni dei quali permetteranno operativamente di recuperare i contenuti sulla base di necessità specifiche e metterli a disposizione.

Il modello logico del "Multimedia Layer" avrà dunque la seguente architettura logica:

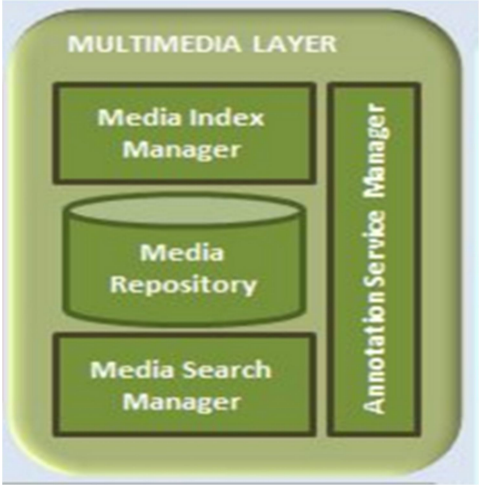


Figura 43: Architettura Logica del Multimedia Layer

Tra i principali componenti troviamo:

- *Media Repository*: si occuperà di inserire i contenuti multimediali in uno o più data storage, a seconda del tipo di file che viene caricato.
- *Media Index Manager*: si occuperà di fornire funzionalità a supporto della *indicizzazione* e del *versioning* del contenuto, del controllo degli accessi ai contenuti, della memorizzazione e della gestione degli stessi, e del controllo sulla sicurezza della fruizione.
- *Media Search Manager*: modulo di ricerca di contenuti che dovrà consentirne la selezione dinamica identificabile a partire dalle informazioni elaborate e specifiche al dominio di interesse. In particolare, il modulo implementerà funzioni di *smart search* e dovrà essenzialmente basarsi su un matching

semantico tra la richiesta elaborata e la descrizione disposta nella fase di memorizzazione del contenuto multimediale nel Repository.

- *Annotation Service Manager*: permetterà di compilare metadati descriventi il contenuto, annotandolo semanticamente, fungendo così da elemento di congiunzione tra il Multimedia Layer ed il Semantic & Ontological Layer.

Oltre ai servizi avanzati di search per contenuti multimediali, il Multimedia Layer si occuperà della gestione e della fruizione vera e propria di contenuto audio/video attraverso modalità di streaming, on demand oppure broadcasting.

6.3.1 *Media Repository*

Tra i requisiti di base che il Multimedia Layer deve soddisfare, troviamo la definizione e archiviazione dei documenti e la loro successiva ed eventuale revisione, sospensione, trasferimento e distruzione. Al fine di soddisfare tali requisiti, la componente dovrà mettere a disposizione dei servizi per garantire un processo automatizzato end-to-end che assicura coerenza di utilizzo e compliance, nascondendo la complessità tipicamente associata alla gestione dei documenti, che restano facilmente accessibili da parte degli utenti senza sparire in un piano di archiviazione incomprensibile. Le informazioni dovranno essere fornite alla persona giusta, al momento giusto, nelle applicazioni e nei dispositivi utilizzati per svolgere il loro lavoro.

Una gestione automatizzata del ciclo di vita di un contenuto multimediale, per l'integrazione diretta dalla scansione all'archiviazione, consentirebbe di far avanzare automaticamente i documenti attraverso un processo di business. Per tale necessità, anche se non è strettamente necessaria, il componente potrebbe esporre anche un servizio dedicato di acquisizione di documenti e dati da qualsiasi fonte mediante funzionalità OCR.

6.3.2 *Media Index Manager*

Per quanto concerne le funzionalità di indicizzazione ed accesso ai contenuti, il layer dovrà mettere a disposizione dei servizi per rendere semplice trovare rapidamente il documento esatto di cui si ha bisogno fra centinaia di migliaia di file, grazie a funzioni di ricerca avanzate grazie a suggerimenti di ricerca immediati e filtri di ricerca, accelerando il recupero dei contenuti pertinenti. Una funzione di cartelle intelligenti (Smart Folders), ad esempio, potrebbe raggruppare i documenti in base al contenuto e non in base a dove sono archiviati.

Riguardo il "Media Index Manager" esso dovrà contemplare funzionalità a supporto dell'indicizzazione e del versioning dei contenuti, oltre al controllo degli accessi alla

fruizione dei contenuti. In particolare, per le funzionalità di indicizzazione dovrà essere possibile abilitare o disabilitare l'indicizzazione automatica sia ai metadati che ai contenuti stessi. Per quanto riguarda le funzionalità di versioning, dovrà essere possibile rendere versionabili tutti i contenuti caricati; qualsiasi proprietà impostata su un file dovrà essere salvata con la versione corrente del file e scritta in una cronologia versioni dopo un aggiornamento importante, ad esempio, quando viene caricato un nuovo file. Per quanto concerne la sicurezza e i livelli di permessi sui file, dovrà essere possibile controllare le autorizzazioni appropriate per un utente prima di accedere ad un file.

6.3.3 *Media Search Manager*

Il "Media Search Manager" dovrà permettere la ricerca di contenuti presenti nel Media Repository e renderli fruibili all'utente. Dovrà essere realizzata, pertanto, una *Search API* per fornire l'accesso alle feature di ricerca, attraverso richieste contenenti strutture JSON per raggruppare opzioni relative alle diverse query che verranno sottoposte. Tra le varie funzioni di ricerca, per ottimizzare la ricerca di contenuti, dovrà essere possibile organizzare i contenuti secondo il principio degli "*Smart Folder*". Con questo approccio, i file dei contenuti multimediali potranno essere archiviati e successivamente visualizzati, in base ai requisiti di informazione. Le ricerche verranno visualizzate in un albero di cartelle, in modo che quando un utente apre una cartella, venga eseguita una query e i risultati visualizzati in un elenco. Anche i file verranno automaticamente classificati quando vengono caricati.

Parte importante del "Media Search Manager" è quella del ritrovamento di porzioni contestualizzate di frame video. Si dovrà realizzare funzionalità tali che ogniqualvolta un video dovrà essere memorizzato nel repository, la componente dovrà avviare tutte le varie operazioni specifiche di deframmentazione, contestualizzazione e annotazione, analogamente a tutti i vari contenuti inseriti in piattaforma. Tutto ciò per permettere al Media Search, analogamente ai contenuti testuali o immagini, di recuperare il segmento puntuale del video considerando il contesto, a sua volta opportunamente etichettato in una accezione semantica. Pertanto, ogni porzione opportunamente taggata verrà quindi automaticamente caricata nel Media Repository, consentendo quindi la possibilità di essere recuperato in maniera intelligente grazie al Media Search Manager.

6.3.4 *Annotation Service Manager*

L'Annotation Service Manager utilizzerà librerie di Deep Learning e Computer Vision (in particolare task di Scene Parsing, Scene Understanding, Semantic Video Segmentation) per l'attività di segmentazione intelligente di un contenuto video e il relativo tagging in base al contesto. Le stesse librerie possono essere utilizzate anche per la segmentazione, annotazione e tagging di immagini. Per quanto riguarda i documenti

testuali, si farà riferimento a tecniche di estrazione di porzioni di contenuti testuali contestualizzati grazie al supporto di ontologie e motori inferenziali semantici caricati nel *Semantic and Ontological Layer*.

Una volta generate le varie porzioni di contenuti contestualizzate, queste verranno automaticamente archiviate in uno *Smart Folder*.

L'Annotation Service Manager comunicherà, quindi, con il *Semantic and Ontological Layer* per acquisire conoscenza di fondo con ontologie ad hoc per l'annotazione semantica di contenuti caricati nel layer ed utilizzerà per il tagging le librerie contenute all'interno del *Analytics / Data Governance Layer*, consentendo, pertanto, una reale interoperabilità tra le varie componenti del Middleware.

7. Semantic and Ontological Layer

Nell'ambito della varietà di domini di applicazione, la Piattaforma opera con una vasta quantità di informazione creata e a sua volta rielaborata. Inoltre, la stessa informazione potrebbe essere affetta da problemi di sinonimia o di polisemia, noti ampiamente nella elaborazione del linguaggio naturale. Per poter quindi limitare quanto esposto e, soprattutto, evitare eventuali problematiche nelle fasi di rielaborazione o di ricerca, è importante intervenire definendo ed utilizzando vocabolari specifici comuni che permettano di operare coerentemente con le informazioni all'interno dei contesti applicativi semanticamente eterogenei tra loro.

Questo porta alla transizione dei vari dizionari, intesi come collezioni di informazioni, verso vere e proprie ontologie. Secondo un approccio collaborativo, al fine di evitare dispersioni e per raggiungere un modello di rappresentazione e di annotazione delle risorse comune, si punta principalmente verso quelle ontologie già presenti sul web e frutto appunto di approccio collaborativo.

Nell'architettura del Middleware è stata introdotta la componente denominata **Semantic and Ontological Layer**, che si occuperà, dunque, di gestire modelli di descrizione semantica e di annotazioni ontologiche e definire un motore semantico in grado di operare per riconoscere le informazioni nei domini nei quali si desidera procedere con l'elaborazione delle informazioni.

7.1 Funzionalità previste e comunicazione con altre componenti del Middleware

Il Semantic and Ontological Layer dovrà mettere a disposizione una serie di funzionalità per supportare la Piattaforma nel contestualizzare ed arricchire semanticamente le informazioni in entrata o frutto di elaborazione e/o rielaborazione. Dopo aver individuato come trattare le informazioni e come classificarle, si dovrà individuare uno strumento che sia in grado di poter ottemperare alla gestione di modelli di descrizione

semantica ed annotazione ontologica nei vari contesti applicativi semanticamente eterogenei, tenendo in considerazione i vocabolari individuati. Si dovrà, inoltre, prevedere un luogo dove saranno immagazzinati i vari dizionari e si dovranno prevedere funzioni in grado di effettuare interrogazioni di ontologie pubbliche.

Questo Layer ha il compito di servire l'Analytics Layer e il Multimedia Layer operando secondo logiche distribuite per supportare un elevato throughput.

Per quanto riguarda l'interazione con l' *Analytics Layer*, il Semantic and Ontological Layer dovrà supportare il processo di applicazione di un modello di machine learning addestrato ("*inference*") sui dati processati, cercando di fornire una *explanation* del risultato della inferenza. In pratica, l'informazione inferita sarà corredata di concetti semantici affinché possa fornire un significato in termini di rappresentazione della conoscenza.

Relativamente all'interazione con il *Multimedia Layer*, il Semantic and Ontological Layer dovrà annotare i documenti e contenuti multimediali gestiti dal layer, con tag contestualizzati per consentire una rapida ricerca intelligente degli stessi o di parti specifiche di essi.

7.2 Tecniche e Tecnologie

Per realizzare quanto previsto per il Semantic and Ontological Layer, verranno utilizzate tecniche e tecnologie di rappresentazione della conoscenza e ragionamento automatico (Knowledge Representation and Reasoning, KRR) per modellare l'informazione sulla base di un'interpretazione formale e rigorosa del suo significato (semantica). Questo consente l'interoperabilità tra differenti piattaforme e l'applicazione di procedure di inferenza per derivare nuova conoscenza implicita da quella esplicitamente asserita in una base di conoscenza. Tra i diversi strumenti e linguaggi di KRR, quelli derivanti dall'iniziativa del Semantic Web godono di ampia adozione, con un background teorico basato sulle Logiche Descrittive ben approfondito dalla comunità scientifica e numerose implementazioni ottimizzate per scenari Web-based. La convergenza tra il Semantic Web e l'Internet of Things (IoT) ha poi portato alla nascita del cosiddetto Semantic Web

of Things (SWoT) [83] che mira all'elaborazione intelligente dell'informazione e alla sua disseminazione in ambienti mobili e pervasivi. La ricerca allo stato dell'arte ha dimostrato come si possano sfruttare dispositivi di contenute capacità computazionali e memoria per estrarre dati in tempo reale, interpretarli in maniera efficiente e annotarli generando una caratterizzazione semantica in base al contesto operativo. In questa prospettiva, la piattaforma Digital Future mira a creare trasparenza dell'infrastruttura di sistemi complessi, sviluppare un approccio a basso sforzo per le parti interessate e configurare le interazioni tra dispositivi IoT al fine di raggiungere un obiettivo specificato dalle soluzioni verticali di business, ovvero la Smart Manufacturing, il controllo ambientale e l'assistenza sanitaria digitale.

Questo obiettivo è svolto specificamente dallo strato semantico e ontologico del Middleware, gestendo processi di scoperta automatica basati su specifici linguaggi di descrizione semantica e gestendo modelli di descrizione semantica e annotazioni ontologiche alimentate da un motore semantico in grado di riconoscere le informazioni nei domini opportunisticamente contestualizzate dai tenant.

Per la rappresentazione dei dati ontologici e delle annotazioni semantiche, sono stati definiti dal W3C diversi standard che costituiscono uno "stack" di specifiche per il "web semantico", in particolare:

- Resource Description Framework (RDF): metodo per la descrizione o modellazione di informazioni; si basa su espressioni della forma soggetto-predicato-oggetto, note come triplette; basi di dati specifiche per lo storage di grandi quantità di triple RDF prendono il nome di "triplestore". Un tale impiego è comune per la realizzazione di basi di conoscenza. Dallo standard RDF sono definibili diversi linguaggi di serializzazione, tra cui:
    - RDF/XML: basato su XML e definito nella RFC 3870 [84];
    - Turtle: basato su una notazione tabulare, consente di esprimere triplette RDF in modo più semplice rispetto a RDF/XML.
    - Notation3 (o N3): estende Turtle con caratteristiche aggiuntive rispetto a RDF.

  - RDF Schema (RDF/S): linguaggio per la descrizione di un vocabolario RDF.
  - SPARQL: linguaggio per effettuare query su dati memorizzati in formato RDF.
- Web Ontology Language (OWL) [85]: famiglia di linguaggi per la rappresentazione della conoscenza; in particolare, esistono diverse versioni di OWL:
  - OWL Lite: è decidibile e ha procedure di deduzione con complessità ben conosciute.;
  - OWL DL: è semplice da implementare ma ha basso potere espressivo;
  - OWL Full: utilizza la logica del primo ordine, addirittura espandendola con altri predicati, rendendolo quindi molto espressivo ma indecidibile.
  - OWL 2: raccomandato dal W3C dal 2009, comprende tre sottolinguaggi: OWL 2 EL è un frammento che ha una complessità di ragionamento in tempo polinomiale; OWL 2 QL è progettato per consentire un accesso e una query più semplici ai dati memorizzati nei database; OWL 2 RL è un sottoinsieme di regole di OWL 2.

7.3 Architettura Logica del Layer

Il Layer in questione dovrà funzionalmente permettere la progettazione, definizione ed elaborazione di annotazioni semantiche e generazione di nuova conoscenza attraverso processi di inferenza grazie al repository di ontologie derivanti dal mondo IoT generico e da altre specifiche a seconda del contesto operativo del pillar.

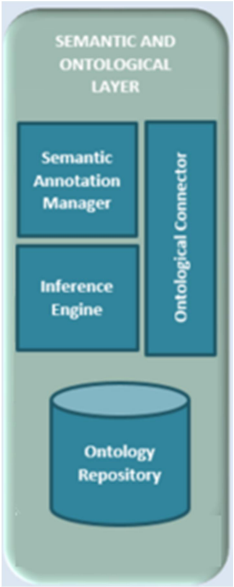


Figura 44: Architettura Logica del Semantic and Ontological Layer

Tra le principali componenti del Layer troviamo:

- *Semantic Annotation Manager*: si occuperà di etichettare gli oggetti (informazioni testuali strutturate e non strutturate, immagini e video) associando una informazione di contesto che ne arricchisca il significato.
- *Inference Engine*: si occuperà di generare nuova conoscenza attraverso processi di inferenza tipici del mondo del semantic Web.
- *Ontological Connector*: si occuperà di recuperare l'ontologia (o più ontologie) più adatta per l'elaborazione semantica dell'informazione di un determinato dominio.
- *Ontology Repository*: si occuperà di archiviare le ontologie recuperate in rete o costruite ad hoc per poi essere utilizzate dall'Ontological Connector.

7.3.1 *Semantic Annotation Manager*

La vertiginosa crescita di informazioni presenti sul Web ha creato la necessità di sviluppare tecniche di ritrovamento delle informazioni sempre più efficaci, capaci di sfruttare la semantica dei contenuti. La maggior parte delle informazioni presenti sul web è costituita da dati non strutturati come, ad esempio, testo libero. L'annotazione semantica è un approccio alla risoluzione di tale problema: si effettua un mapping

ontologico dei contenuti stessi e delle loro sottoparti, così da renderli semi-strutturati attraverso l'aggiunta di metadati. Ciò rende più agevole il recupero dei contenuti stessi con strumenti automatici. La creazione di un "Web semantico" è stata promossa dal W3C, portando alla creazione di progetti quali "Linked Open Data" (LOD) che ha portato alla pubblicazione e alla reciproca interconnessione di numerosi dataset in formato RDF, provenienti da sorgenti eterogenee quali DBpedia, Freebase, WordNet. Le annotazioni semantiche possono essere viste come informazioni semantiche associate a risorse Web, esse sono usate per arricchire il contenuto informativo dei documenti e per esprimere in maniera formale il significato di una porzione di testo in un documento. In pratica una annotazione semantica può essere vista come un commento su una risorsa digitale (o su una sua parte) che si può salvare e recuperare digitalmente. Questo commento viene esplicitato dall'espressione di annotazione ed è associato alla porzione di testo che è stata annotata. Collezionare annotazioni semantiche su un corpus di testi appartenenti ad uno specifico dominio offre una serie di vantaggi quali arricchire i testi associando espressioni in linguaggio naturale a porzioni di testo ed esprimere in modo formale il significato di una risorsa per poi permetterne il recupero in maniera intelligente aumentando le prestazioni di un sistema di ricerca tradizionale. L'annotazione delle risorse Web può essere effettuata con riferimento ad una specifica ontologia o, più in generale, di un vocabolario condiviso, come pure attraverso l'attività di tagging. Essa è un'attività centrale del Web Semantico.

Le annotazioni si usano quindi per arricchire il contenuto informativo dei documenti e per esprimere in maniera formale, il significato di una porzione di testo in un documento, dei Web service, delle strutture dati coinvolte in un processo di collaborazione ed interoperabilità tra applicazioni software. In pratica una annotazione semantica può essere vista come un commento su una risorsa digitale (o su una sua parte) che si può salvare e recuperare digitalmente. Vi sono molti criteri per classificare le annotazioni. Si può analizzare il livello di formalità del linguaggio usato, il posizionamento dell'annotazione, il destinatario dell'annotazione, il tipo di risorsa annotata, il livello di restrizione imposto sul linguaggio e la modalità di annotazione. Il livello di formalità può andare dallo strettamente formale all'informale. Nel livello

formale l'annotazione è espressa in un linguaggio formale di rappresentazione della conoscenza come per esempio RDF e RDFS: un framework per esprimere i metadati nel web, DAML, DAML+OIL, OWL usati per la definizione di ontologie. Il livello informale esprime l'annotazione in linguaggio naturale o in un linguaggio controllato e permette di aggiungere informazioni su documenti o risorse, fruibili da un utente umano. Per quanto riguarda il posizionamento un'annotazione può essere "embedded", ovvero inserita all'interno dello stesso documento annotato oppure "attached", ovvero memorizzata separatamente e collegata al documento da un link. Il destinatario di un'annotazione può essere un utente umano o un computer. Nel primo caso l'annotazione ha lo scopo di fornire informazioni aggiuntive su una risorsa, in questo caso l'annotazione in genere sarà poco formale probabilmente realizzata in linguaggio naturale, nel secondo caso l'annotazione ha lo scopo di esplicitare il significato di una risorsa, in tal caso l'annotazione deve essere di tipo formale per poi essere processabile da una macchina. Vari tipi di risorse possono essere annotati: documenti, frammenti di testo, pagine HTML, immagini, video, audio, dati sensoristici.

Collezionare annotazioni semantiche su un corpus di testi appartenenti ad uno specifico dominio offre una serie di vantaggi quali: arricchire i testi associando espressioni in linguaggio naturale a porzioni di testo e esprimere in modo formale il significato di una risorsa per poi permetterne il recupero in maniera intelligente aumentando le prestazioni di un sistema di ricerca tradizionale. L'annotazione delle risorse Web può essere effettuata con riferimento ad una specifica ontologia o, più in generale, di un vocabolario condiviso, come pure attraverso l'attività di tagging. Essa è un'attività centrale del Web Semantico. I vantaggi principali di un contenuto annotato semanticamente riguardano la possibilità di mettere in correlazione e di aggregare informazioni provenienti da fonti eterogenee e di conseguenza consentire all'utente di acquisire nuova conoscenza.

Per quanto concerne il processo di annotazione semantica, si devono considerare due fasi distinte: in una prima fase, di Information Extraction (IE), si generano le annotazioni, effettuando un mapping tra le informazioni presenti nel contenuto analizzato e

l'ontologia di riferimento; in una seconda fase, di Information Retrieval (IR), si utilizzano tali annotazioni per l'indicizzazione ed il recupero del contenuto stesso. L'IR può essere utilizzata per determinare quali documenti, presi da una collezione più o meno vasta, sono rilevanti per una query specificata da un utente, questa può essere semplificata come una lista di parole chiave. Ne consegue che le tecniche di IR permettono di determinare i documenti rilevanti in risposta ad una ricerca. Le tecniche di utilizzo nell'IE richiedono invece una comprensione maggiore di ciò che è scritto nei testi. A differenza dell'IR, tale ambito richiede di determinare contenuti rilevanti all'interno dei documenti, quindi non nella loro interezza. Il contenuto rilevante, esportato in un formato specifico, può essere integrato in sistemi intelligenti anche in coordinazione con sistemi IR per migliorare la ricerca ottenendo risultati più pertinenti. Per raggiungere l'obiettivo di comprendere in modo profondo il contenuto di un documento testuale, le tecniche Natural Language Processing (NLP) da applicare non si sono limitate solo al dividere il testo determinando i termini, eventualmente anche pesati in ordine di importanza, ma ad estrarre informazioni. I metodi statistici non sono sufficienti se non combinati con basi di conoscenza specifici, che devono essere definiti a priori. Questo fa sì le tecniche di IE dipendono dal dominio in cui vengono applicate, non potendo portare un sistema sviluppato per un ambito specifico in un dominio differente (es. giornalistico vs medico vs economico). Per ovviare a tale limitazione si sono sviluppate tecnologie adattive di IE che facendo uso di metodi empirici di NLP aiutano la costruzione di sistemi IE. Alcuni degli attuali sistemi di IE seguono un approccio a cascata, aggiungendo struttura ai documenti ed in alcuni casi filtrando le informazioni estratte in base a regole. Tali componenti possono essere riassunti in macro-elementi come segue:

- Modulo di pre-elaborazione di documenti:
    - "text zoner", riconoscono specifiche zone del testo quali: titolo, finale, bibliografia, ecc.
    - "segmenter" (o "splitter"), dividono il testo in segmenti che possono coincidere con i paragrafi di un testo o con frasi ("sentence splitter").
    - "tokenizer", dividono il testo in elementi lessicali (parole).

  - "language guesser", riconoscono la lingua in cui un documento è scritto tramite campionamento di parole.
  - "lexical analyzer", procedono con l'analisi lessicale del testo e può includere sia l'analisi morfosintattica che il riconoscimento di entità nominate (Named Entity) con relativa disambiguazione.
  - "POS tagger", etichetta ogni parola del testo (POS, Part Of Speech) con una etichetta relativa all'analisi grammaticale (nomi, congiunzioni, articoli, ecc).
  - "stemmer", è un normalizzatore che fornisce per ogni parola coniugata la sua forma base senza suffisso (o prefisso). Esempio: "studies" diventa "studi" avendo tolto il suffisso.
  - "lemmatizer", è un normalizzatore che fornisce per ogni parola coniugata il suo lemma di riferimento. Esempio: "studies" diventa "study" che è il verbo corrispondente all'infinito.
- Parser sintattico dei documenti, parsing in modo completo o parziale;
- Interpretatore semantico, per generare una forma logica (una espressione con una sintassi specifica che definisce una semantica precisa. Per esempio, se A=>B ed A è vero allora B è vero: "tutti gli uomini sono mortali", "Socrate è un uomo" allora "Socrate è mortale") o un template che viene riempito con gli elementi riconosciuti nel testo;
- Analizzatore del discorso, per legare i vari elementi semantici presenti nelle frasi di un testo (anaphora resolution è una di esse);
- Generatore finale, per trasformare le informazioni derivate nel formato di uscita finale.

I sistemi di IE per la pre-elaborazione dei documenti, fanno uso di risorse linguistiche o di componenti indipendenti dal dominio. Molto importante invece è il modulo di riconoscimento delle entità nominate (Named Entity Recognizer, NER), questi possono far uso di automi a stati finiti (Finite State Transducer) o ricerca in dizionari (dictionary lookup).

Con particolare riferimento ai *contenuti testuali*, la task più importante della fase di Information Extraction è la Named Entity Recognition (NER). Nel campo dell'elaborazione del linguaggio naturale (NLP) si considerano "Named Entity" (NE): persone, organizzazioni, località e tutto ciò che possiede un nome proprio; si possono includere, per estensione del concetto, anche date, numeri ed altri tipi di informazioni. Le Named Entity denotano individui o istanze ben particolari e quindi richiedono una conoscenza del mondo più specifica rispetto al riconoscimento di concetti e relazioni. Lo scopo di tali componenti è quello di identificare all'interno dei testi, le entità nominate cioè parole che assumono uno specifico significato nel testo quali: persone, luoghi, organizzazioni, ecc. È di comune accordo nella comunità scientifica di considerare entità nominate anche le quantità siano essere monetarie o temporali. Esistono diverse metodologie per il NER, ci sono sia approcci basate a regole grammaticali, che modelli statistici basati ad esempio su apprendimento automatico. I sistemi basati su regole, possono ottenere risultati molto precisi a scapito della copertura del numero di fenomeni riconoscibili. Inoltre, diventano non manutenibili all'aumentare del numero di regole da utilizzare, ed estremamente costose dovendo essere scritte manualmente da esperti di settore. C'è inoltre un vincolo intrinseco nel metodo e riguarda le logiche formali che soggiacciono il modello scelto. Considerando infatti la famosa gerarchia di Chomsky, se l'automa scelto è di tipo a stati finiti (equivalente per intenderci ad una regular expression) è relativamente semplice da costruire ma la possibilità di riconoscere correttamente le entità all'interno di un testo di per sé dipendente dal contesto è limitata. In un ambito ambiguo come la lingua, potrebbe infatti capitare di dover riconoscere sia persone e luoghi, e se non si considera il contesto, è impossibile discernere quando viene usata la parola "Washington" come persona (il primo Presidente degli Stati Uniti) dalla capitale degli Stati Uniti. Si tenga presente che all'altro estremo della gerarchia di Chomsky c'è la macchina di Turing, cioè il modello teorico che soggiace i computer odierni, e la grammatica naturale si trova in un livello intermedio tra il "tipo 1" e il "tipo 2". Quando fu definita tale gerarchia, si pensò che fosse possibile definire una grammatica formale che potesse descrivere la grammatica naturale vera a propria, diversi studiosi hanno provato ad ottenere ciò ma non esiste ancora nessuno

che ci sia riuscito e si hanno dubbi che questa possa essere raggiunta in un modo o nell'altro. E comunque anche se si ottenesse sarebbe estremamente complessa ed in alcuni casi non utilizzabile a causa dell'ambiguità della lingua. Il classico esempio è: "la vecchia porta la sbarra". La parola "porta" nella suddetta frase è "verbo" o "nome"? Non esiste una risposta giusta o sbagliata, sono purtroppo vere entrambi e nemmeno un essere umano riesce a capirlo se non utilizzando altri sensi, per esempio il tono di voce o il contesto visivo in cui si trova, e comunque non è sempre possibile. I sistemi che invece fanno uso dell'apprendimento automatico, invece affrontano il problema in modo diverso, si focalizzano sull'analizzare un dataset, scritto da un essere umano, apprendendo dei modelli che poi potranno essere riutilizzati in altri casi. Tali approcci però hanno bisogno di un grande quantitativo di materiale di addestramento e purtroppo non possono essere utilizzati in contesti differenti da quelli per cui sono stati pensati. Negli ultimi stanno prendendo piede piattaforme che applicano una logica di crowd contribution. In tali piattaforme una moltitudine di contributori, possono accedere e partecipare all'annotazione collettiva di documenti, utilizzando sistemi di etichette riferenti a categorie di Wikipedia. Esistono alcune tecniche che generalmente possono essere applicate alla NER. Solitamente le performance che si possono ottenere dipendono dalla qualità e numerosità del materiale di addestramento. Le tecniche più popolari nel recente passato sono state: Hidden Markov Model (HMM) e Maximum Entropy; recentemente invece stanno prendendo piede tecniche di deep learning quali le RNN (Recurrent Neural Networks) che vengono definite come state-of-the-art, anche se comunque i vantaggi che si ottengono rispetto ai passati approcci sono dell'ordine di pochi punti percentuali. In generale il problema del riconoscimento delle entità è equivalente a quello del POS Tagging e Chunk Tagging.

L'esplosione della visibilità dell'Intelligenza Artificiale sul Web portandolo da argomento e ambito di nicchia ad argomento disponibile un po' a tutti, ha fatto sì che sul World Wide Web siano disponibili librerie che permettono di trattare documenti non strutturati e permettere l'estrazione delle entità. Spark stesso fornisce alcuni dei passi necessari in una catena di processamento NLP, ma limitandosi alle fasi iniziali di elaborazione quali: tokenizzazione del testo, stop word filtering (rimozione delle parole

ritenute non rilevanti nel testo quali preposizioni o articoli), pesatura dei termini. Esistono tuttavia altre librerie o framework più o meno integrati rispetto a Spark che aggiungono funzionalità quali: POS Tagging, Entity Extraction, Named Entity Recognition. Il problema maggiore però sono le implementazioni e i modelli presenti all'interno che possono supportare solo un limitato insieme di lingue. Storicamente i primi lavori su queste tecnologie si sono focalizzate sull'utilizzo di dataset in inglese, perché facilmente reperibili e aperti all'utilizzo. Alcune cose sono cambiate, ma i dataset in alcune lingue continuano ad essere problematiche ed una di quella, la lingua italiana, rientra in tale caso. Dal punto di vista tecnologico le seguenti implementazioni sono ritenute interessanti:

- **John Snow Labs** [86], è ritenuto dalla comunità di Spark come un punto di riferimento rendendolo uno strumento facilmente integrabile nel framework di calcolo, ma i modelli disponibili sono limitati all'inglese (in realtà all'americano, ma la differenza è minima e quindi trascurabile). Per adattarlo all'italiano sarebbe necessario ottenere un corpus italiano per addestrare il POS Tagger e successivamente il componente di riconoscimento delle entità.
- **Stanford Parser** [87], è praticamente il punto di riferimento della comunità scientifica per l'alto livello di performance che ottiene dal punto di vista della qualità dei risultati ottenibili, sviluppato dalla medesima università americana. Sono supportate nativamente diverse lingue: inglese, cinese, tedesco e arabo. Altri gruppi di lavoro hanno utilizzato lo Stanford Parser per lavorare su altre lingue: italiano, bulgaro e portoghese. Nel particolare l'italiano è supportato dalla Fondazione Bruno Kessler (FBK) tramite la sua piattaforma Tint. Tale libreria però non è direttamente integrabile all'interno di Spark a causa delle tecnologie interne che utilizza. In particolare fa pesantemente uso delle librerie ProtoBuf di Google, e si è quindi proceduto a modificare il progetto iniziale operando tramite la tecnica dello shading, che permette di forzare la versione delle librerie da utilizzare procedendo ad una riscrittura del bytecode delle librerie operando sostituzioni. Lo shading ha funzionato correttamente, ma un'altra tecnica, la "reflection", non ha permesso l'integrazione all'interno della

versione di Spark 2.1 (e anche Spark 2.2) obiettivo della piattaforma HortonWorks da utilizzare. Esistono altri gruppi di lavoro che stanno lavorando a portare lo Stanford Parser su Spark, ma poggiando il loro lavoro su versioni successive di Spark, e nel particolare la 2.4 che non risulta influenzata dall'utilizzo della tecnica della reflection.

- **IXA Pipes** [88], è un parser che supporta più lingue sviluppato dall'Università Basca. Il POS Tagger interno supporta nativamente le lingue: basco, olandese, inglese, francese, galiziano, tedesco, italiano e spagnolo. Il NER supporta nativamente: basco, spagnolo, inglese, tedesco, olandese, tedesco, italiano e spagnolo. Le performance ottenibili dal parser, sono comparabili rispetto allo Stanford Parser attestandosi però un po' più in basso. Il vantaggio maggiore è che il codice sorgente non utilizza tecniche particolarmente complicate e quindi facilmente integrabili all'interno di Spark. Inoltre, è stato ampiamente testato su diverse lingue avendo partecipato a gare internazionali specifiche sulle lingue.

Dalle considerazioni sopra menzionate, la scelta preferita sarebbe stata il parser di John Snow Labs, ma la carenza di modelli già disponibili avrebbe richiesto il reperimento di materiale di addestramento difficilmente reperibile. D'altro canto, gli altri due supportano modelli per l'italiano, ma lo Stanford Parser non è integrabile in Spark, mentre IXA Parser è integrabile. Inoltre, l'implementazione interna fa ampiamente uso del framework Apache OpenNLP, sempre più un punto di riferimento per la costruzione di librerie NLP.

Considerando il caso di *contenuti multimediali* non necessariamente testuali, l'annotazione semantica automatica prevede generalmente la scelta di opportune feature che consentano di codificare il contenuto (o una sua parte) in un vettore, il quale è successivamente fornito ad un classificatore. Nel caso di immagini o video, un approccio alla risoluzione del problema di individuazione dei contenuti dell'immagine, tipico del Content Based Image Retrieval (CBIR) [89], è quello di suddividere l'immagine in sottoparti e classificarle, attraverso un confronto con le immagini di un dataset di addestramento. Tale dataset può essere anche costruito incrementalmente,

"imparando" dall'etichettamento manuale dell'utente ed assistendolo nelle successive scelte. Un caso particolare, ma molto importante, per quanto concerne le immagini è quello del riconoscimento dei volti (face recognition), in quanto consente l'identificazione di persone. Qualora siano già presenti metadati associati ad un contenuto multimediale, quali ad esempio il titolo, l'autore, una descrizione o dei tag, è possibile considerare la possibilità di annotare semanticamente tali metadati.

Nel caso di *contenuti sonori* contenenti del parlato è possibile utilizzare delle tecniche di speech recognition per ricondursi al caso più semplice di annotazione semantica di testi. Altre possibilità di annotazione semantica sono fornite dall'identificazione del parlante e dal riconoscimento di eventi sonori di interesse quali, ad esempio, esplosioni, spari, ecc.

Per quanto riguarda il caso più generale di *dati rivenienti da sensori*, il Semantic Web of Things (SWoT) è un nuovo paradigma che combina Semantic Web e Internet of Things per consentire l'annotazione di oggetti e processi in contesti pervasivi non predicibili come quelli industriali. Ne deriva una rappresentazione non ambigua su cui applicare algoritmi di ragionamento automatico in tempo quasi reale. Soluzioni innovative sono proposte per superare i classici limiti degli ambienti industriali ove è particolarmente rilevante l'esigenza di supportare le decisioni o attuarle per via automatica: i dati in campo vengono aggregati e annotati; da essi viene inferita conoscenza utile al processo decisionale che infine si riverbera sui dispositivi di attuazione. Il SWoT adatta le tecnologie di rappresentazione della conoscenza inizialmente sviluppate per il Semantic Web a contesti orientati all'Internet of Things. Ciò permette di annotare, rispetto ad una base di conoscenza condivisa, oggetti rappresentati tramite micro-dispositivi a basso costo e non intrusivi quali sensori, transponder per l'identificazione a radiofrequenza (RFID) e beacon Bluetooth Low Energy (BLE).

Il paradigma SWoT deve fronteggiare le criticità tipiche dei sistemi pervasivi: eterogeneità delle piattaforme, impredicibilità, volatilità e limitatezza delle risorse di calcolo, dipendenza dal contesto. Molte delle soluzioni attuali per l'IoT ereditano semplicemente tecnologie derivanti dalle reti infrastrutturate, con una gestione centralizzata dell'informazione. Il Semantic Annotation Manager definisce invece

un'evoluzione su base semantica dei protocolli di comunicazione standard IoT con lo scopo di costituire sistemi distribuiti basati sulla conoscenza, capaci di gestire autonomamente il discovery e l'elaborazione delle informazioni a partire da dati grezzi estratti dal campo. Tra i principali protocolli interessati vi sono Bluetooth [90], RFID [91], EIB/KNX [92] e CoAP [93]. Un approccio da considerare si basa su un modello di ubiquitous Knowledge Base (u-KB) [83] definita come base di conoscenza distribuita senza coordinamento centralizzato, in cui ogni risorsa (oggetto, contesto o processo) fa riferimento ad un'ontologia che modella il particolare dominio applicativo. La ricostruzione al volo di un frammento della u-KB abilita i servizi di inferenza di un ragionatore progettato per sistemi mobili ed embedded [94]. Si sfruttano comunque (i) algoritmi di compressione per lo storage efficiente delle annotazioni di oggetti/contesti/processi e (ii) servizi di inferenza non-standard, per identificare le caratteristiche in comune, in contrasto o assenti in un'annotazione rispetto ad un'altra. Il risultato è una misura di rilevanza semantica ed una giustificazione su base logica della comparazione tra due o più annotazioni. Tale approccio si distingue in flessibilità ed efficacia per il resource retrieval convenzionale basato su parole chiave e match esatti di attributi, risultando di grande utilità particolarmente in scenari in cui agenti mobili devono fornire supporto alle decisioni in tempo reale all'interno di ambienti dinamici.

Tra gli sviluppi più recenti della visione SWoT si annovera il Physical Semantic Web (PSW) [95], un'estensione del progetto Physical Web (PW) [96] proposto da Google, basato su BLE. Il PW prevede la disseminazione di una serie di beacon, che condividono in broadcast un URL (Uniform Resource Locator) sfruttando BLE, e dispositivi mobili personali in grado di rilevarli per identificare gli oggetti presenti in un dato contesto. Gli URL sono associati solitamente a pagine Web create per scopi informativi o ad applicazioni Web che offrono servizi o esperienze interattive. Nonostante i vantaggi del PW in termini di generalità ed immediatezza delle interazioni dell'utente con gli oggetti in prossimità, esso presenta alcuni limiti: (i) i beacon possono condividere solo semplici URL, mentre tutte le informazioni aggiuntive devono essere recuperate attraverso una connessione Internet; (ii) è sempre richiesta un'interazione esplicita con l'utente; (iii) la classificazione delle risorse si basa unicamente sulla distanza degli oggetti, senza

considerarne le caratteristiche intrinseche. Il framework PSW consente ad ogni oggetto di esporre attraverso il proprio URL non una semplice pagina Web, ma un'intera annotazione che ne descrive stato, caratteristiche e funzionalità. Le annotazioni possono essere aggiornate ed integrate durante la vita di un oggetto (si pensi ad esempio a sensori e attuatori) al fine di riflettere l'evoluzione delle capacità e percezioni dell'oggetto stesso (o di un contesto o di un processo). L'evoluzione protocollare proposta nel PSW consente inoltre una comunicazione peer-to-peer tra i dispositivi BLE in modo da abilitare nuovi scenari completamente autonomi o privi di connessione a Internet.

Le caratteristiche prestazionali da considerare ai fini della valutazione di un sistema di annotazione semantica sono: la velocità, la resilienza e l'affidabilità. La resilienza è la capacità di gestire in modo efficiente cambiamenti nei documenti indicizzati e/o nell'ontologia. Per quanto concerne l'affidabilità, si adottano in particolare tre misure: precision, recall, F-measure. Tali valori andrebbero confrontati rispetto alle prestazioni ottenute da un annotatore umano.

Di seguito si illustrano i principali software per l'annotazione semantica, con particolare riferimento alle soluzioni open source e/o gratuitamente disponibili a fini di ricerca o che siano stati, storicamente, di particolare rilievo, soprattutto scientifico.

Semantic Turkey [97] è un software per l'annotazione semantica e lo sviluppo di ontologie, realizzato dall'Università di Roma Tor Vergata. Semantic Turkey estende Mozilla Firefox e consente all'utente di tenere traccia dei contenuti visionati nelle pagine web, collegandoli a delle ontologie.

Annotea [98] è un software per l'annotazione semantica manuale, realizzato dal W3C. Le annotazioni sono salvate in formato RDF esternamente al documento, utilizzando XPointer per la localizzazione delle stesse all'interno del documento.

KIM [99] è una piattaforma sviluppata dall'OntoText Laboratory di Sirma AI Ltd. Per l'estrazione ed annotazione semantica delle informazioni fa uso del software di text mining **GATE** (General Architecture of Text Engineering). Le annotazioni sono salvate

all'interno dei documenti. Si compone di un'ontologia, di una base di conoscenza e di un server. L'ontologia, in linguaggio RDF(S), definisce le classi e le entità di interesse, mentre la base di conoscenza contiene le descrizioni delle entità. Un apposito plugin per i web browser consente all'utente di visualizzare le parti annotate con colori differenti.

Apolda (*Automated Processing of Ontologies with Lexical Denotations for Annotation*) [100] è un plugin per GATE. Analogamente ad un gazetteer annota un documento, ma non richiede una lista di termini poiché è in grado di utilizzare direttamente l'ontologia (in formato OWL).

Apache Jena [101] è un framework per la creazioni di applicazioni per il web semantico ed include sia dei tool sia librerie Java. In particolare il framework include:

- API di lettura, elaborazione e scrittura di dati RDF nei formati XML, N-triple o Turtle;
- API per la gestione di ontologie RDF e OWL;
- Un motore inferenziale per il ragionamento con dati RDF o OWL;
- Un sistema efficiente di storage di grandi quantità di triple RDF;
- Un sistema di query conforme alle più recenti specifiche SPARQL;
- Server per la pubblicazione di dati RDF.

Sesame [102] è un framework Java estensibile per il salvataggio, l'interrogazione e l'inferenza in presenza di dati RDF. Può essere utilizzato come web server o come libreria Java. Supporta i principali formati RDF (inclusi: RDF/XML, Turtle, N-triple) ed i principali linguaggi di interrogazione (SeRQL e SPARQL). Sviluppato da Aduna, Sesame è liberamente disponibile sotto licenza BSD.

Virtuoso OpenSource [103] è un database relazionale ad alte prestazioni con un web server ed un repository WebDAV integrati. Sviluppato da OpenLink in linguaggio C, è distribuito sotto licenza GNU GPL 2.0. Virtuoso OpenSource supporta l'inclusione di codice SPARQL all'interno di query SQL per l'interrogazione di dati RDF memorizzati all'interno del database.

4store [104] è un database RDF (triple store) progettato per fornire elevate prestazioni, scalabilità e stabilità. È sviluppato da Garlik, azienda del gruppo Experian. Distribuito sotto licenza GNU GPL 3.0, 4store è disponibile per i principali sistemi della famiglia Unix: Linux, Mac OS, FreeBSD.

OWLIM [105] è una famiglia di DBMS RDF appartenente al framework KIM, che fa uso dei software Sesame e Jena. Alla famiglia appartengono tre versioni del DBMS: la versione OWLIM-Lite riesce a gestire decine di milioni di statement su computer desktop di uso comune ed è l'unica disponibile gratuitamente; la versione OWLIM-SE consente di conseguire una elevata scalabilità, con elevate performance anche in presenza di query multi-utente e di decine di miliardi di statement; la versione OWLIM-Enterprise aggiunge affidabilità al sistema, mediante un'infrastruttura di tipo cluster con nodi ridondanti.

D2RQ [106] è una piattaforma per il mapping automatico di query SPARQL in query SQL così da far apparire un database relazionale come un grafo virtuale RDF memorizzato in un triplestore. L'accesso è in sola lettura: sono consentite ricerca e lettura, ma non modifica ed eliminazione. Un tool di questa piattaforma è **D2R Server** : si tratta di una servlet Java che consente la pubblicazione sul "web semantico" di database relazionali, poiché ne rende possibile la navigazione mediante query SPARQL e quindi l'interconnessione con Linked Data. La piattaforma D2RQ è sviluppata in linguaggio Java e distribuita sotto licenza Apache 2.0. Per approfondimenti si rimanda alla sezione dedicata del presente documento.

OWL API [107] è un API Java per la creazione, gestione e serializzazione di ontologie OWL, attualmente sviluppata presso la University of Manchester. È liberamente disponibile sia sotto licenza Apache sia sotto licenza GNU LGPL. Include funzionalità di parsing e scrittura di dati RDF/XML, OWL/XML e Turtle. Include inoltre delle interfacce per il collegamento a reasoner esterni.

Tema Tres [108] è un'applicazione web, sviluppata in PHP, per la gestione di rappresentazioni formali di conoscenza, thesauri, tassonomie e vocabolari multilingua. È distribuita sotto licenza GNU GPL 2. Parte della documentazione è in spagnolo.

7.3.2 *Inference Engine*

Ragionare con ontologie e basi di conoscenza è uno dei motivi per cui una specifica deve essere formale. Per ragionamento intendiamo derivare fatti che non sono espressi in modo esplicito in una ontologia o nella base di conoscenza. Tutti i formalismi discussi in questa sezione sono stati creati con la prospettiva dell'elaborazione automatica, ma a causa delle loro proprietà come decidibilità o complessità computazionale o anche a causa del livello di formalità, non è sempre possibile.

I linguaggi per la rappresentazione della conoscenza sono linguaggi formali utilizzati per rappresentare la conoscenza relativa ad un dominio di interesse, in modo che tale conoscenza possa essere utilizzata per effettuare deduzioni. La caratteristica rispetto ad altri linguaggi formali è nel tipo di rappresentazione, di "alto livello" nel senso che consente di rappresentare in modo semplice ma nello stesso tempo rigoroso informazioni complesse. Le logiche descrittive sono al momento il linguaggio più diffuso per la rappresentazione della conoscenza. Le logiche descrittive (*Description Logics, DL*), note anche come Linguaggi Terminologici o Concept languages, sono create con il focus sul ragionamento trattabile. Gli elementi sintattici di base dei DL sono: concetti (classi), che denotano insiemi di oggetti; ruoli (proprietà), relative coppie di concetti; individui (istanze), che denotano elementi speciali denominati in concetti. Una interpretazione semantica è costituita da una coppia di elementi che consistono in un dominio e una funzione di interpretazione che mappa ogni concetto in un sottoinsieme del dominio, ogni ruolo in un sottoinsieme del prodotto cartesiano dei domini, e ogni individuo in un elemento di un dominio. Questi elementi di base possono essere combinati usando dei costruttori per comporre concetti ed espressioni di ruolo. Un costruttore base usato nei DL è quello di congiunzione di concetti. I ruoli possono essere combinati usando quantificazioni esistenziali e universali.

Un ragionatore semantico, chiamato anche motore di ragionamento, motore a regole o semplicemente un ragionatore, è un pezzo di software in grado di inferire conseguenze

logiche da una serie di fatti asseriti o di assiomi. La nozione di ragionatore semantico generalizza quella di un motore di inferenza, fornendo un insieme più ricco di meccanismi con cui lavorare. Le regole di inferenza sono comunemente specificate per mezzo di un linguaggio ontologico e spesso un linguaggio logico descrittivo. Il ragionatore quindi, a partire dalle informazioni contenute nelle ontologie (conoscenza esplicita), deduce conoscenza implicita. Un ragionatore implementa algoritmi che permettono di manipolare i formalismi delle logiche descrittive. Ogni ragionatore è progettato per elaborare informazioni descritte in accordo con una determinata logica. Molti ragionatori usano la logica dei predicati del primo ordine per eseguire il ragionamento; l'inferenza comunemente procede con il concatenamento in avanti (*forward-chaining*) e il concatenamento all'indietro (*backward-chaining*).

Un sistema di rappresentazione della conoscenza (KR) basato su Description Logics fornisce servizi per impostare le basi di conoscenza, ragionare sui loro contenuti e manipolarli. Una knowledge base (KB) comprende due componenti: TBox e ABox. Il TBox introduce la terminologia, cioè il vocabolario di un dominio di applicazione (cioè una ontologia), mentre l'ABox contiene affermazioni su individui nominati in termini di questo vocabolario. Il vocabolario consiste di concetti che denotano insiemi di individui e di ruoli che denotano relazioni binarie tra individui. Oltre ai concetti e ai ruoli atomici (concetti e nomi di ruolo), tutti i sistemi DL consentono ai loro utenti di creare descrizioni complesse di concetti e ruoli. Il TBox può essere utilizzato per assegnare nomi a descrizioni complesse. Il linguaggio per la creazione di descrizioni è una caratteristica di ogni sistema DL e diversi sistemi si distinguono per i loro linguaggi descrittivi. Il linguaggio descrittivo ha una semantica teorica del modello. Pertanto, le affermazioni nel TBox e nell'ABox possono essere identificate con formule nella logica del primo ordine o, in alcuni casi, in una leggera estensione di esso.

Un sistema DL non solo memorizza terminologie e asserzioni, ma offre anche servizi che ragionano su di loro. Tipici compiti di ragionamento per una terminologia sono di determinare se una descrizione è *soddisfacibile* (cioè, non contraddittoria), o se una descrizione è più generale di un'altra, cioè se la prima *sussume* la seconda. Problemi

importanti per un ABox sono scoprire se il suo insieme di asserzioni è *coerente*, cioè se ha un modello e se le asserzioni nell'ABox implicano che un particolare individuo è un'*istanza* di una data descrizione del concetto. I controlli di attendibilità (*satisfiability check*) delle descrizioni e dei controlli di coerenza (*consistency check*) delle serie di asserzioni sono utili per determinare se una base di conoscenza è significativa.

Con i test di sussunzione, è possibile organizzare i concetti di una terminologia in una gerarchia secondo la loro generalità. Una descrizione del concetto può anche essere concepita come una query, descrivendo un insieme di oggetti a cui si è interessati. Pertanto, con i test di istanza, è possibile recuperare gli individui che soddisfano la query.

Riassumendo, le attività richieste ad un ragionatore sono le seguenti:

- *Soddisfacibilità di un concetto*: determinare se una descrizione del concetto non è contraddittoria, cioè se un individuo può esistere che possa essere istanza del concetto;
- *Sussunzione di concetti*: determinare se il concetto C sussume concetto D, cioè se la descrizione di C è più generale della descrizione di D;
- *Coerenza di una ABox rispetto alla TBox*: determinare se gli individui nella ABox non violano le descrizioni e gli assiomi descritti dalla TBox;
- *Controllo un individuo*: controlla se l'individuo è un'istanza di un concetto;
- *Recupero di individui*: trova tutti gli individui che sono esempi di un concetto;
- *Realizzazione di un individuo*: trova tutti i concetti a cui l'individuo appartiene, specialmente i più specifici.

Da un punto di vista logico, un concetto ha senso per noi se esiste *un'interpretazione* che soddisfi gli assiomi (cioè un modello) tale che il concetto denoti un insieme non vuoto in tale interpretazione. Si dice che un concetto con questa proprietà sia soddisfacibile rispetto al TBox e altrimenti insoddisfacibile. Il controllo della soddisfacibilità dei concetti è un'inferenza chiave. Come vedremo, una serie di altre inferenze importanti per i concetti può essere ridotta alla (non) soddisfacibilità. Ad esempio, per verificare se un modello di dominio è corretto, o per ottimizzare le query formulate come concetti,

potremmo voler sapere se un concetto è più generale di un altro: questo è il problema di sussunzione. Un concetto C è sussunto da un concetto D se in ogni modello di T l'insieme denotato da C è un sottoinsieme dell'insieme denotato da D. Algoritmi che controllano la sussunzione vengono anche impiegati per organizzare i concetti di un TBox in una tassonomia secondo il loro generalità. Ulteriori relazioni interessanti tra concetti sono quelli di equivalenza e disgiunzione. Questi task non sono semanticamente molto diversi. Ad esempio, la soddisfacibilità può essere testata come sussunzione di $\perp$, ovvero un concetto è insoddisfacibile se nessun individuo può esistere che sia istanza del concetto. Per tutti i task, è sufficiente essere in grado di controllare le conseguenze deduttive o derivare tutte le conseguenze deduttive di una teoria.

Tutti i sistemi di DL forniscono ai propri utenti servizi di inferenza standard come il calcolo della gerarchia di sussunzione e il test di coerenza di un ABox. In alcune applicazioni è emerso, tuttavia, che questi servizi non sono sufficienti per fornire un supporto ottimale quando si creano e si gestiscono basi di conoscenza di DL di grandi dimensioni. Per questo motivo, alcuni sistemi di DL (ad es. Classic) forniscono ai propri utenti servizi di sistema aggiuntivi, che possono essere ricostruiti formalmente come nuovi tipi di problemi di inferenza. In primo luogo, le inferenze standard possono essere applicate dopo che è stato definito un nuovo concetto per scoprire se il concetto è non contraddittorio o se il suo posto nella tassonomia coincide con l'intuizione dell'ingegnere della conoscenza; tuttavia, queste inferenze non supportano direttamente il processo di definizione del nuovo concetto. Per superare questo problema, sono stati proposti i servizi di inferenza non standard di calcolo del *least common subsumer* e il *most specific concept*. In secondo luogo, se una knowledge base viene gestita da diversi ingegneri della conoscenza, è necessario un supporto per rilevare più definizioni dello stesso concetto intuitivo. Poiché diversi ingegneri della conoscenza potrebbero utilizzare nomi diversi per lo "stesso" concetto primitivo, il test di equivalenza standard potrebbe non essere adeguato a verificare se descrizioni diverse si riferiscono alla stessa nozione. Il servizio di inferenza non standard dell'*unificazione* di descrizione di concetti affronta questo problema consentendo di sostituire i nomi dei concetti con descrizioni di concetti

appropriate prima di testare l'equivalenza. Il *matching* è un caso speciale di unificazione, che è stato utilizzato, ad esempio, per l'eliminazione di parti irrilevanti di descrizioni di concetti di grandi dimensioni prima di visualizzarle all'utente. In terzo luogo, e in modo molto astratto, la *riscrittura* delle descrizioni concettuali consente di trasformare una data descrizione C in una "migliore" descrizione D, che soddisfa determinati criteri di ottimalità (ad esempio, di piccole dimensioni) e si trova in una determinata relazione (ad esempio, equivalenza o sussunzione) con la descrizione originale C. Gli approcci per risolvere i nuovi problemi di inferenza sono di solito basati su una caratterizzazione appropriata della sussunzione, che può essere utilizzata per ottenere un algoritmo di sussunzione strutturale. Innanzitutto, le descrizioni dei concetti sono trasformate in una certa forma normale, in cui i fatti impliciti sono stati resi espliciti. In secondo luogo, la struttura delle forme normali viene confrontata in modo appropriato. Questo è uno dei motivi per cui la maggior parte dei risultati su inferenze non standard sono limitati a linguaggi che possono essere trattati da algoritmi di sussunzione strutturale. Si possono distinguere due tipi di forme normali proposte in letteratura. In un approccio, chiamato approccio basato sul linguaggio, la forma normale di una descrizione del concetto viene data in termini di determinati gruppi di parole finite o regolari sull'alfabeto di tutti i nomi dei ruoli. Quindi, la sussunzione può essere caratterizzata dall'inclusione di questi insiemi. Il secondo approccio, chiamato graph-based, trasforma le descrizioni dei concetti in cosiddetti grafi di descrizione. Qui, la descrizione della sussunzione di concetti è caratterizzata dall'esistenza di certi omomorfismi tra i corrispondenti grafi descrittivi. L'approccio basato sul linguaggio è particolarmente utile per caratterizzare la sussunzione tra descrizioni di concetti ciclici, cioè descrizioni definite mediante terminologie cicliche in FL_0 e *ALN*. D'altra parte, l'approccio basato su grafi può essere impiegato per gestire la negazione primitiva e le restrizioni esistenziali.

Di seguito si illustrano i principali software per il ragionamento semantico, con particolare riferimento alle soluzioni open source e/o gratuitamente disponibili a fini di ricerca o che siano stati, storicamente, di particolare rilievo, soprattutto scientifico.

Il software **Mini-ME** [109] (the Mini Matchmaking Engine) è un ragionatore compatto per la DL *ALN* (*Attributive Language with unqualified Number restrictions*). È finalizzato al matchmaking semantico per la scoperta di risorse / servizi in contesti mobili e ubiqui, sebbene sia anche un motore di inferenza del Web semantico di uso generale. Servizi di inferenza non standard ottimizzati consentono una categorizzazione a grana fine e un ranking delle risorse rispetto ad una richiesta, fornendo sia una metrica di distanza che una spiegazione logica dei risultati. Mini-ME è applicabile ad un'ampia classe di applicazioni in cui è possibile aggregare grandi set di risorse a componenti a bassa complessità per creare servizi composti con una crescente complessità semantica. Questo è adatto per i limiti di calcolo e di alimentazione dei fornitori di risorse in contesti onnipresenti e per la loro breve disponibilità di storage. Un'architettura "agile" di individuazione di servizi in grado di selezionare, assemblare e orchestrare al volo molti componenti elementari è più gestibile ed efficace in applicazioni mobili e pervasive. Mini-ME implementa entrambi i task di ragionamento standard per la gestione della Knowledge Base (*sussunzione, soddisfacibilità, classificazione*) e servizi di inferenza non standard per la scoperta e il posizionamento (*ranking*) di risorse basate sulla semantica (*abduzione, contrazione, copertura, bonus, differenza*). Il ragionatore Mini-ME in esecuzione sul dispositivo mobile esegue un matchmaking semantico tra il profilo utente e le annotazioni beacon. Per ogni coppia <profilo utente, risorsa>, viene calcolato un valore del punteggio per valutare la somiglianza tra il profilo utente e il beacon. Inoltre, sfruttando l'inferenza non standard di Concept Abduction [110], viene fornita all'utente una spiegazione completa del punteggio che prova quali caratteristiche compatibili e / o mancanti della risorsa determinano il risultato. Analogamente, in caso di profilo / risorsa di incompatibilità, il servizio Concept Contraction [110] viene utilizzato per rilevare proprietà e elementi in conflitto delle annotazioni dei beacon.

Bossam [111] è un motore di regole basato su RETE con supporti nativi per il ragionamento su ontologie OWL, regole SWRL e regole RuleML.

RacerPro [112] è un OWL Reasoner and Inference Server per il Semantic Web. RACER è l'acronimo di *Renamed ABox e Concept Expression Reasoner*. RacerPro è il nome commerciale del software. Con RacerPro puoi implementare progetti di forza industriale basati sugli standard W3C RDF e OWL ed è uno strumento ideale per la ricerca e lo sviluppo.

OntoBroker [113] è un motore di inferenza con ragionamento nativo su F-Logic, ObjectLogic, RIF e OWL (motore di inferenza elencato W3C).

Cyc [114] è un motore di inferenza con ragionamento in forward e backward chaining con numerosi moduli specializzati per la logica del second'ordine. OpenCyc è la soluzione open source.

KAON2 [115] è un'infrastruttura per la gestione di ontologie OWL-DL, SWRL e F-Logic.

CEL [116], OWL 2 EL reasoner (Apache 2).

Cwm [117], un ragionatore in forward-chaining utilizzato per querying, checking, trasformazione e filtraggio delle informazioni. Its core language is RDF,[118] extended to include rules, and it uses RDF/XML or N3 serializations as required (W3C software license).

Drools [119], un ragionatore in forward-chaining basato a regole che usa una implementazione avanzata dell'algoritmo RETE (Apache license 2.0).

ELK [120], OWL 2 EL reasoner (Apache 2).

EYE [118], un ragionatore in semi-backward chaining che supporta Euler paths e attraverso la notazione N3 è interoperabile con Cwm (MIT license).

FaCT++ Reasoner [121], un ragionatore basato su tableaux per Description Logics (DL) espressive come OWL e OWL 2 ma che manca di un supporto per i vincoli di chiave e alcuni datatypes. Scritto in C++ (LGPL).

Flora-2 [122], un ragionatore object-oriented basato a regole per la rappresentazione della conoscenza e per meccanismi di inferenza (Apache 2.0).

HermiT [123], OWL 2 DL reasoner (LGPL).

Jcel [124], OWL 2 EL reasoner (LGPL / Apache 2).

Jena [125], un framework open-source per semantic-web in Java che include una serie di moduli per il ragionamento semantico (Apache License 2.0).

OpenRules [126], un Sistema di gestione delle decisioni e delle regole di business open source. Insieme ad un motore a regole sequenziale, include un motore inferenziale a regole che utilizza un risolutore a vincoli.

Pellet [127], OWL 2 DL reasoner (AGPL).

RACER [128], OWL 2 DL reasoner (BSD-3). RACER è l'acronimo di *Renamed ABox e Concept Expression Reasoner* ed è un è un OWL Reasoner and Inference Server per il Semantic Web.

7.3.3 *Ontological Connector*

Come già detto, una base di conoscenza è composta da un'ontologia e dalle istanze (individui) descritti in accordo con l'ontologia. I ragionatori attuali permettono di effettuare inferenze sulle ontologie ipotizzando una netta separazione tra l'ontologia e gli individui. Ciò significa, da un punto di vista pratico, che è possibile descrivere individui solo tramite concetti e relazioni presenti nell'ontologia della base di conoscenza. Non è possibile descrivere individui che nella descrizione contengano altri individui. Dato questo vincolo, bisogna porre particolare attenzione al target dell'ontologia; tutto ciò che ci servirà per descrivere gli individui di interesse per la base di conoscenza deve essere definito nell'ontologia.

Le ontologie del Web semantico consentono alle macchine di interpretare e elaborare le informazioni sul Web, fornendo un modello comune che può essere compreso sia dagli esseri umani che dal computer, per condividere, scambiare e riutilizzare i dati in base ai loro significati previsti. L'uso delle ontologie mira a raggiungere l'interoperabilità semantica collegando e integrando contenuti digitali multipli ed eterogenei a livello semantico, che è esattamente l'idea centrale della visione del Semantic Web. Inoltre, non solo l'uso delle ontologie riduce le ambiguità semantiche offrendo una singola risorsa interpretativa, ma anche il contenuto informativo è reso disponibile per il

consumo della macchina, mentre la maggior parte del contenuto trovato oggi sul Web è principalmente destinata al solo consumo umano.

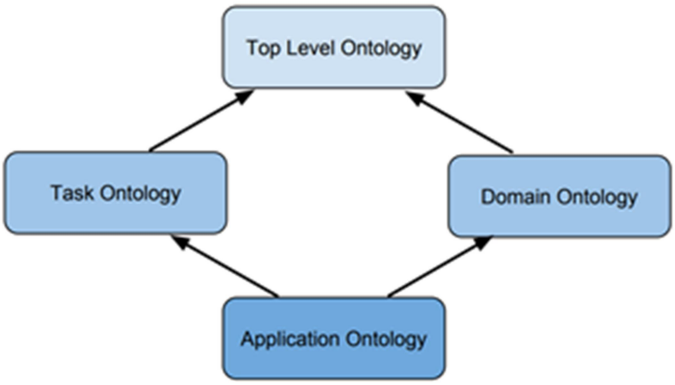


Figura 45: Classificazione ontologica: le frecce spesse rappresentano relazioni di specializzazione [129]

In letteratura esistono diverse classificazioni di ontologie. Come illustrato nella Figura 1, viene proposta una classificazione basata sul grado di generalizzazione [129]:

- *Top Level Ontologies*: descrivono concetti molto generici e astratti come spazio, tempo, materia, oggetti, eventi, azioni, ecc. Le ontologie di questo tipo sono valide indipendentemente dal problema specifico o dal dominio di interesse.
- *Domain Ontologies*: descrivono un vocabolario relativo ad un dominio generico (ad es. medicina o uno sport) specializzando i concetti forniti dalle top level ontology.
- *Task Ontologies*: descrivono il vocabolario dei termini necessari per svolgere task generici o attività (ad es. diagnosi) specializzando i concetti forniti dalle top level ontology.
- *Application Ontologies*: descrivono i termini dei concetti che dipendono sia da un particolare dominio che da un'attività. Le ontologie di questo tipo sono limitate solo ad un'applicazione specifica.

La sottocomponente dell'Ontological Connector agirà quindi da collettore, aggregatore e composer di ontologie, a più livelli della classificazione ontologica, recuperando le Top

Level Ontologies e le Domain Ontologies dalla sottocomponente di Ontology Repository e, laddove non fosse stato possibile preventivamente memorizzare tali ontologie, si predisporrà un modulo software per la ricerca e download online da repository pubbliche. Le Task Ontologies e le Application Ontologies verranno invece descritte a seconda dei casi e opportunamente sulla base delle prime due classi di ontologie, in modo da poter avere conoscenza di base di ogni attività specifica e applicativa di un particolare dominio o contesto di esecuzione, non riscrivendo tutta l'ontologia da zero, ma riusando i concetti già definiti a granularità diversa.

Le Top Level Ontologies e le Domain Ontologies non contemplate nell'Ontology Repository potranno quindi essere scaricate da repository pubbliche, come ad esempio **LOV4IoT** (*Linked Open Vocabularies for Internet of Things*), che conta quasi 500 ontologie basate su ambiti IoT [130].

Dopo che un'ontologia è stata sviluppata, viene utilizzata, riutilizzata e collegata ad altre ontologie e deve essere manutenuta. Queste attività possono essere più semplici se si pensa a un'ontologia con questi compiti in mente. Ad esempio, la costruzione di ontologie su un'ontologia superiore condivisa e l'utilizzo di un design modulare di solito comporta un utilizzo e una manutenzione più semplici.

7.3.4 *Ontology Repository*

Tra i requisiti di base che il Semantic & Ontological Layer deve soddisfare, troviamo la possibilità di archiviazione e utilizzo di basi di conoscenza ontologiche, tassonomie e vocabolari semantici. Al fine di soddisfare tali requisiti, si prevede di sviluppare un modulo di archiviazione per ontologie che potranno essere richiamate all'occorrenza a dai servizi della piattaforma che ne hanno bisogno.

Alcune delle ontologie di base (Top Level Ontologies) che si prevede di archiviare nella piattaforma potranno essere le seguenti:

- ✓ *Semantic Sensor Network Ontology* (**SSN**) [131]: è un'ontologia per descrivere i sensori e le loro osservazioni, le procedure coinvolte, le caratteristiche studiate di interesse, i campioni utilizzati per farlo e le proprietà osservate, così come gli

attuatori. SSN segue un'architettura di modularizzazione orizzontale e verticale includendo un'ontologia di base leggera ma autonoma denominata **SOSA** (Sensore, Osservazione, Sample e Attuatore) per le sue classi e proprietà elementari.

- *On the usage of the SSN ontology [132]*: Questo documento fornisce un'analisi dell'uso del nucleo di ontologia SOSA e l'ontologia della rete semantica dei sensori (SSN) e, ove appropriato, l'uso di classi e relazioni equivalenti nell'ontologia originale (SSN-XG) [SSNO], sviluppata dal gruppo di incubatori di reti di sensori semantici W3C e pubblicata nel 2011.
- *Extensions to the Semantic Sensor Network Ontology [133]*: Questa nota descrive alcune estensioni dell'ontologia SSN per abilitare (i) il collegamento all'ultima caratteristica di interesse per un'osservazione, un atto di campionamento o un'attuazione, accanto al collegamento alla caratteristica (di interesse), che potrebbe essere un campione; (ii) raccolte omogenee di osservazioni, in cui una o più delle caratteristiche di interesse, ultima caratteristica di interesse, proprietà osservata, procedura, sensore, tempo-fenomeno o tempo-risultato possono essere condivisi da tutti i membri della collezione.

✓ *IoT-Lite Ontology [134]: è una ontologia leggera per rappresentare risorse, entità e servizi di Internet of Things (IoT).* IoT-Lite è un'istanza dell'ontologia SSN.

Prima dell'inizio del lavoro di ingegneria ontologica sull'ontologia SSN, il gruppo W3C ha esaminato in modo approfondito ontologie e modelli di dati che descrivono i sensori e le loro capacità nonché le osservazioni. Questa parte del rapporto presenta le 17 ontologie incentrate sui sensori e sull'osservazione che sono state analizzate in dettaglio [135]: CSIRO Sensor Ontology, OntoSensor, SWAMO, SDO, MMI Device Ontology, SensorML Processes, CESN, WISNO, A3ME, Ontonym - Sensor, SEEK OBOE, SERES O&M, Stimuli-Centered, Sensei Observation and Measurement Ontology, OOSTethys, O&M-OWL (SemSOS), Socio-Ecological Research and Observation oNTOlogy (SERONTO).

I repository di ontologie sono in genere più specifici dei motori di ricerca web semantici e delle loro interfacce di navigazione / ricerca che possono variare notevolmente. Offrono strumenti che possono essere specifici per il tipo di applicazioni per cui è stato progettato il repository.

Per quanto riguarda repository di ontologie generiche che possono essere riutilizzabili troviamo in [136] le seguenti ontologie: BioPortal, Cupboard, DERI Vocabularies, Knoodl, Ontology Design Patterns, Prefix.cc, Schemapedia, SchemaWeb.

Ulteriori hub di repository di ontologie sono i seguenti: [137], [138], [139], [140], [141].

8. Event Layer

La sempre più presente pervasività delle reti di sensori e dei dispositivi intelligenti che raccolgono continuamente una quantità sempre più elevata di dati, pone come sfida l'esigenza di analizzare un flusso di dati in continua crescita quasi in tempo reale. Le componenti interne della piattaforma preposte alla analisi dei dati operano su una grande quantità di informazioni che vengono elaborate in modo asincrono. Gli eventi scatenati dal processo di elaborazione vengono gestiti in maniera asincrona rispetto al flusso di esecuzione del software. I concetti di tempestività e elaborazione del flusso sono fondamentali per giustificare la necessità di una nuova classe di sistemi specificamente progettati per elaborare le informazioni come un insieme di flussi in base a una serie di regole di elaborazione. I flussi continui di eventi di alto livello richiedono quindi elaborazioni intelligenti in tempo reale.

Nell'architettura del Middleware è stata introdotto l' **Event Layer**, con l'obiettivo di raccogliere ed elaborare tutti gli eventi di interesse che intervengono, al fine di organizzare articolate pipeline di analisi sui dati, utilizzando gli strumenti più idonei alla tipologia di analisi da effettuare per fornire avvisi, richieste di attuazione o più semplicemente archiviare pattern complessi di eventi in un apposito contenitore.

8.1 Funzionalità previste e comunicazione con altre componenti del Middleware

L'*Event Layer* dovrà mettere a disposizione una serie di funzionalità per l'acquisizione di flussi di dati derivanti da sensori, l'individuazione ed elaborazione di tali dati in pattern di eventi, anche complessi, la manipolazione di tali pattern ed infine l'esecuzione di azioni scatenate da un motore di regole su eventi. Per poter esporre tali servizi, l'*Event Layer* dovrà comunicare con più componenti del Middleware.

L'*Event Layer* dovrà comunicare con il *Connector Layer* per attingere agli eventi che vengono generati dai dispositivi IoT collegati alla piattaforma Digital Future. Pertanto, si

dovrà realizzare una soluzione applicativa che permetterà di direzionare verso l'*Event Layer* i flussi di dati grezzi che entrano nella Piattaforma transitando attraverso *l'IoT Broker* e l'*Event Bus*, pronti per essere gestiti ed elaborati.

Poiché tramite l'*Event Bus*, i dati possono essere salvati nello storage previsto dal "*Data Governance Layer*", oltre che essere elaborati dalle procedure in esecuzione nel "*Analytical Layer*" sia in modalità Stream Processing e sia Batch Processing, l'*Event Layer* dovrà comunicare anche con questo layer. Quindi la comunicazione con con "*Data Governance / Analytical Layer*" potrà avvenire per:

- attingere agli eventi (dati persistiti) da processare;
- ricevere i risultati della elaborazione, sotto forma di nuovi eventi da elaborare ulteriormente;
- inviare eventi preparati ad hoc a fronte di una attività di filtering e/o arricchimento anche complessi da sottoporre ad algoritmi di machine / deep learning in esecuzione nel "*Analytical Layer*".

Una volta che tali eventi sono stati "catturati" dall'*Event Layer*, in generale si dovrà mettere in atto un sistema di elaborazione di eventi, che potranno occorrere a differenti livelli di complessità di analisi.

Il *Complex Event Processing* (CEP) è uno dei modelli emergenti volti a monitorare e reagire agli eventi in arrivo in tempo reale o quasi reale. I fattori critici per i sistemi basati su eventi sono il rilevamento degli eventi e l'enorme quantità di informazioni disponibili sugli eventi. I flussi continui di eventi di alto livello richiederanno quindi processori intelligenti in tempo reale. In un mondo sempre più orientato alle tecnologie IoT, il CEP si configura come modello ideale per l'elaborazione dei dati derivanti dai sensori IoT: primo, perché i dati IoT sono strutturati come sequenze temporali (time series) di dati autocorrelati e il CEP è ideato per elaborare flussi continui di dati; secondo, perché la maggior parte dei casi d'uso IoT prevedono tempi di risposta brevissimi per cui il CEP ha un vantaggio considerevole rispetto ad altri strumenti di elaborazione di dati in streaming; terzo, perché generalmente i casi d'uso IoT sono complessi e vanno oltre la semplice computazione di dati aggregati, pertanto

necessitano di un supporto di operatori complessi come finestre temporali e pattern di query temporali.

Una funzione estremamente interessante legata al processo di gestione degli eventi complessi riguarderà la possibilità di utilizzare eventi precedentemente generati dal sistema come input di altri processi sia in maniera atomica che aggregata. Un evento o una sequenza di eventi potrà quindi fare da attivatore per l'esecuzione di un nuovo processo o servizio. Per garantire l'adeguatezza delle azioni al modificarsi del contesto nel quale gli eventi vengono generati, l'*Event Layer* potrà trarre vantaggio dalle funzionalità che il *Reflective Layer* metterà a disposizione del Middleware. Di conseguenza, le azioni che potranno essere *attivate in maniera riflessiva / adattiva* potranno afferire alle seguenti tipologie:

- generazione messaggi di avviso
- richieste di attuazioni effettuate verso i dispositivi
- invio di comunicazioni verso il pillar di riferimento
- attivazione di algoritmi / script per la produzione di ulteriori eventi

L'attivazione di queste azioni sarà possibile ponendo gli eventi alla attenzione di un motore a regole che sarà presente nel Reflective Layer.

8.2 Tecniche e Tecnologie

Per dotare la Piattaforma di un Layer avente le caratteristiche descritte precedentemente, un buon punto di partenza potrebbe essere quello di individuare soluzioni open source in grado di soddisfare, anche in parte, i requisiti del layer. La soluzione scelta andrebbe, poi, evoluta per raggiungere i requisiti definitivi.

In generale, l'elaborazione di eventi complessi (CEP) risolve il problema di corrispondenza degli eventi in ingresso continui con un pattern. Il risultato di una corrispondenza sono solitamente eventi complessi derivati dagli eventi di input. A differenza dei DBMS tradizionali in cui viene eseguita una query sui dati memorizzati, il CEP esegue i dati su una query memorizzata. Tutti i dati che non sono rilevanti per la query possono essere immediatamente scartati. I vantaggi di questo approccio sono

ovvi, dato che le *query CEP* vengono applicate su un flusso potenzialmente infinito di dati. Inoltre, gli input vengono elaborati immediatamente. Una volta che il sistema ha rilevato tutti gli eventi per una sequenza di matching, i risultati vengono emessi istantaneamente. Questo aspetto porta in modo efficace alla capacità di analisi in tempo reale del CEP.

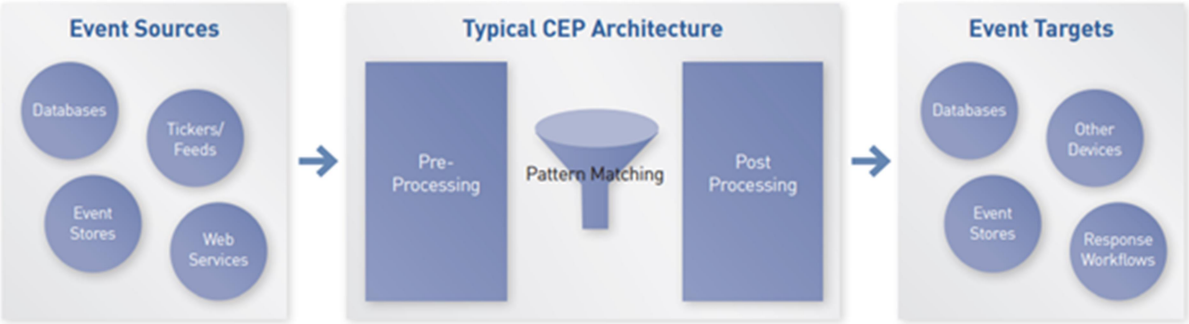


Figura 46: Tipica Architettura CEP [143]

Alla luce di queste modalità di generazione e gestione degli eventi, il paradigma di elaborazione CEP ha suscitato un interesse significativo. Tra le varie tecnologie open source analizzate su cui far riferimento per la realizzazione dei moduli del layer, sicuramente Apache Flink [144], risulta essere una delle più promettenti ed interessanti. Apache Flink, un sistema open source per l'elaborazione di dati in streaming e batch, si basa sulla filosofia che molte classi di applicazioni di elaborazione dati, tra cui analisi in tempo reale, pipeline di dati continui, elaborazione di dati storici (batch) e algoritmi iterativi (analisi di grafi o algoritmi di apprendimento automatico) possono essere espresse ed eseguite come linea di flussi di dati tollerante agli errori. Apache Flink segue un paradigma che comprende l'elaborazione del flusso di dati come modello unificante per l'analisi in tempo reale, i flussi continui e l'elaborazione batch sia nel modello di programmazione che nel motore di esecuzione.

In combinazione con code di messaggi durevoli che consentono la riproduzione quasi arbitraria di flussi di dati (come Apache Kafka), i programmi di elaborazione *stream* non fanno distinzione tra l'elaborazione degli ultimi eventi in tempo reale, l'aggregazione continua dei dati periodicamente in finestre di grandi dimensioni o l'elaborazione di

terabyte di dati storici. Invece, questi diversi tipi di calcoli iniziano semplicemente la loro elaborazione in diversi punti del flusso durevole e mantengono diverse forme di stato durante il calcolo. Attraverso un meccanismo di *windowing* altamente flessibile, i programmi basati su Flink possono fare calcoli sia in modo precoce e approssimativo, oltre che in modo ritardato e accurato, risultati nella stessa operazione, ovviando alla necessità di combinare sistemi diversi per i due casi d'uso. Flink supporta diverse nozioni di tempo (event-time, ingestion-time, processing-time) per dare ai programmatori un'elevata flessibilità nel definire come correlare gli eventi.

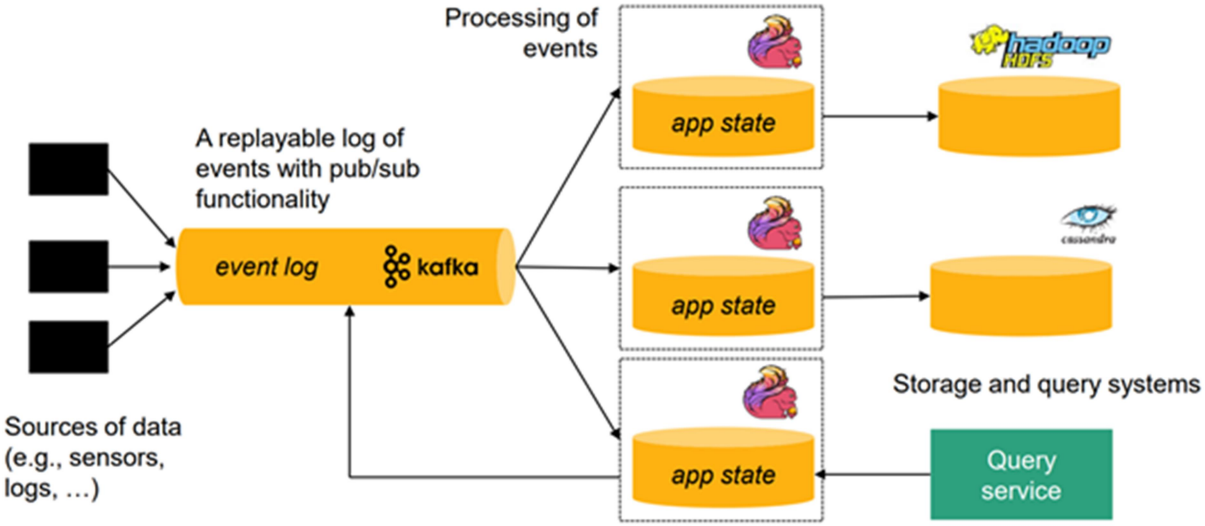


Figura 47: Batch and Stream Processing [145]

Apache Flink con la sua natura di streaming e le sue capacità per la bassa latenza e l'elevata elaborazione del flusso di throughput sono una scelta naturale per i carichi di lavoro CEP. In particolare, **FlinkCEP** è la libreria CEP implementata su Flink che consente di rilevare i pattern degli eventi in un flusso continuo di eventi che spesso arrivano non in ordine, offrendo l'opportunità di evidenziare ciò che è importante nei dati.

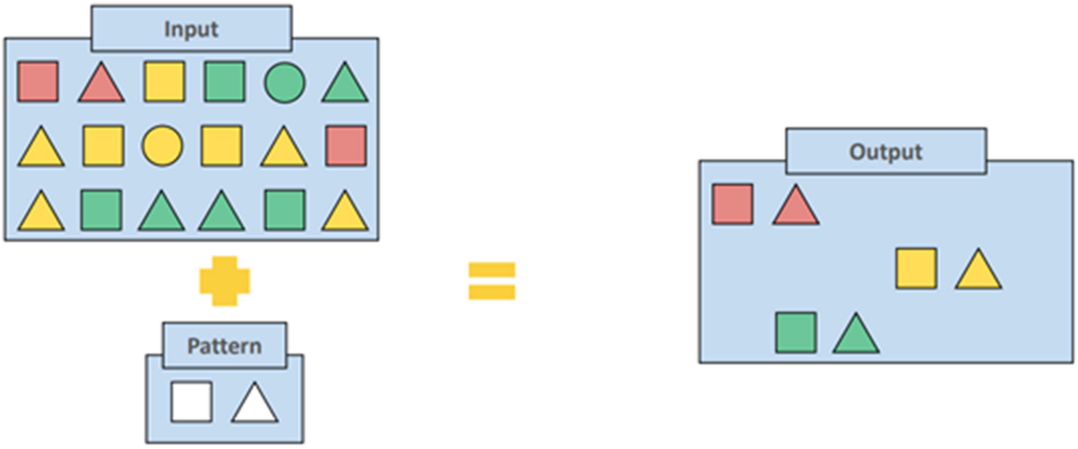


Figura 48: Complex Event Processing Pattern Detection [146]

Un programma FlinkCEP può essere scritto utilizzando delle "Pattern API" che consentono di definire pattern di eventi complessi [147]. In particolare, Apache Flink consente di analizzare i dati in streaming che possono essere analizzati in modo efficiente, inclusi filtri, aggregazioni, join, finestre di dati e così via. Sebbene non fornisca un linguaggio di query EPL (Event Processing Language), Apache Flink fornisce un'ampia collezione di API per eseguire l'elaborazione di eventi complessi [148]. Queste ampie API consentono agli utenti di sviluppare codice per collegare diversi tipi di fonti di dati o flussi di dati, che possono essere consumati utilizzando diversi tipi di formati (binario, XML, JSON, CSV, ecc.). Ogni modello è costituito da più fasi o stati. Il modello deve iniziare con lo stato iniziale e quindi passare da uno stato a quello successivo, specificando eventualmente delle condizioni. Ogni stato deve avere un nome univoco per identificare gli eventi corrispondenti in seguito. È possibile aggiungere stati per rilevare pattern complessi. FlinkCEP include varie caratteristiche come la capacità di: ottenere un throughput elevato e un'elaborazione a bassa latenza; produrre i risultati non appena è disponibile lo stream degli eventi di input; fornire aggregazione nel tempo e timeout tra due eventi di interesse; fornire avvisi e notifiche in tempo reale sul rilevamento di modelli di eventi complessi.

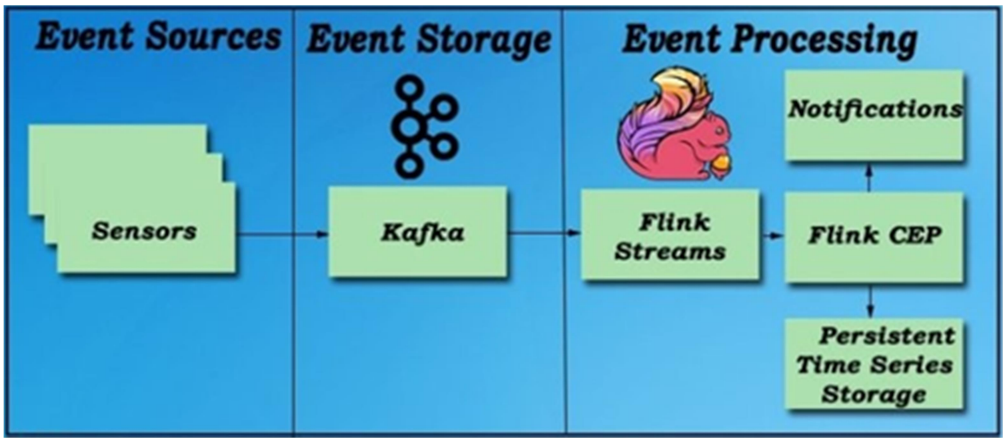


Figura 49: Complex Event Processing with Apache Flink [149]

8.3 Architettura Logica del Layer

Il Layer in questione dovrà funzionalmente permettere la definizione ed elaborazione di workflow di eventi complessi, direzionandoli all'interno della piattaforma ed eseguendoli grazie ad un motore a regole, creando messaggi di avviso (noti anche come *alert*), attivazione di altri processi di elaborazione e attuazioni appositamente ideati per fornire informazioni di contesto lato dispositivo e/o lato pillar.

Il Layer avrà la seguente architettura logica:

Figura 50: Architettura Logica dell'Event Layer

Tra le componenti troviamo:

- *Event Manager*: si occuperà di acquisire gli eventi in arrivo, direttamente dall'Event Bus o dal "*Data Governance/Analytical Layer*", ed elaborarli grazie ad un engine CEP in cui saranno definiti dei modelli di pattern per l'individuazione ed estrazione di pattern complessi (chiamati anche workflow). Tale componente potrà essere implementata attraverso la libreria FlinkCEP di Apache Flink.
- *Pattern Model Store:* si occuperà di memorizzare i modelli di pattern di eventi complessi e le istanze di tali modelli gestiti dal CEP. Anche per questo

componente, Apache Flink si candida come la soluzione ottimale per la implementazione.

- *Rule Engine:* permetterà di gestire le azioni da eseguire, a fronte della elaborazione di pattern di eventi, grazie alla valutazione di regole su eventi al fine di generare alert, attuazioni, semplice archiviazione di nuovi eventi prodotti da utilizzare in attività future, attivazione di algoritmi / script, da far eseguire all'Event Manager o alla componente di processing del "*Data Governance/Analytical Layer*", per la produzione di ulteriori eventi. Il Rule Engine è logicamente appartenente a questo layer ma operativamente sarà allocato nel "Reflective Layer".

9. Reflective Layer

L'Internet delle cose (IoT) e la sua estensione Internet of Everything (IoE) è di sicuro uno dei cambiamenti tecnologici più influenti che abbiamo affrontato negli ultimi anni. Grazie alla diffusione dei sensori e alla diffusione di risorse computazionali miniaturizzate a basso costo, siamo in grado di creare oggetti che producono dati e interagiscono tra loro producendo così una rete di oggetti, dati, processi interconnessi. Negli ultimi anni, c'è stato un enorme sforzo per fornire un accesso immediato a informazioni sul mondo fisico attraverso le tecnologie Internet. La visione dell'IoT mira a integrare il mondo virtuale delle informazioni con il mondo reale delle cose. Il ruolo di un IoT il middleware è quello di fornire la connettività tra il mondo virtuale e il mondo fisico e un'interfaccia tra dispositivi fisici eterogenei e applicazioni. Una delle sfide nella costruzione dell'Internet dei Servizi e delle Cose è il modo in cui il software sarà sviluppato e composto poggiandosi su infrastrutture flessibili e architetture di integrazione. Quando si sviluppa sistemi software basati su servizi e oggetti, le sfide principali riguardano il modo in cui un'architettura intelligente basata sull'internet degli oggetti è progettata per renderla robusta con riferimento ai cambiamenti contestuali che subisce continuamente. Pertanto, si rendono necessari nuovi paradigmi e modelli di composizione del software che riguardano l'eterogeneità, la dinamicità, l'adattività con riferimento a contesti esterni.

In generale, tutti i middleware hanno l'esigenza molto forte di rispondere a variazioni di funzionalità del sistema senza permettere una configurazione del sistema (spegnere tutto e ripartire). In questo filone, negli ultimi anni, la ricerca [203] si è molto focalizzata nella realizzazione di middleware adattivi e riflessivi:

- adattivo: vuol dire che ci si adatta alla situazione corrente dello scenario d'uso.
- riflessività: capacità di un componente di descrivere se stesso.

L'obiettivo di questo layer sarà, pertanto, aggiungere caratteristiche di riflessività al Middleware IoT della piattaforma DF in modo da poter realizzare servizi applicativi per

l'elaborazione dei dati che entrano nella piattaforma DF e facenti parte della logica di elaborazione prevista dalle soluzioni verticali realizzate per i pillar di riferimento, completamente configurabili e adattabili a diversi contesti operativi ed obiettivi specifici del momento, grazie alla capacità di eseguire automaticamente azioni in base:

- ai valori ricevuti dai dispositivi di campo
- al contesto e alla problematica che si vuole affrontare in un certo momento
- al destinatario dei risultati dell'azione, sia esso un dispositivo, sia un umano o un sistema informatico

innescando un insieme di regole. Il Layer consentirà la definizione di un insieme di astrazioni per abilitare l'osservazione e il controllo, a livello applicativo, della struttura e del comportamento dei servizi applicativi software che saranno realizzati, i quali potranno essere adattati dinamicamente alle caratteristiche del contesto operativo di riferimento e al tipo di utilizzatore che consumerà i risultati della loro esecuzione.

9.1 Reflection Pattern

Tra i diversi modelli disponibili per progettare un sistema software, il Reflection Pattern permette allo sviluppatore di produrre un'architettura estensibile fin dall'inizio.

Un sistema riflessivo può apportare modifiche a se stesso per mezzo di ispezione e/o adattamento. L'ispezione consiste nell'esporre il comportamento interno del sistema, in modo che diventi semplice inserire ulteriori comportamenti per controllare l'applicazione. L'adattamento consiste nel cambiare il comportamento interno del sistema dinamicamente, modificando funzionalità esistenti o aggiungendone altre.

Il **Reflection Pattern** è, quindi, un design pattern di tipo architetturale che fornisce una soluzione al problema, cioè un meccanismo per cambiare la struttura e il comportamento di sistemi software in modo dinamico. In un'architettura riflessiva, gli aspetti strutturali e comportamentali (attributi e metodi) di un sistema sono definiti all'interno di meta-oggetti e separati dai componenti che implementano la logica applicativa. Al fine di ottenere un comportamento dinamico, la logica applicativa "interroga" il meta-livello (che può cambiare nel tempo senza necessità che la prima sia

implementata nuovamente) al fine di chiamare le sue funzionalità. Grazie alla reflection, dunque, un sistema può gestire dinamicamente comportamenti non previsti in fase di progettazione.

Il concetto principale nel **Reflection Pattern** è la distinzione tra livello di base e meta-livello. Il livello base include la logica applicativa di base. Il suo comportamento a run time è osservato da un meta-livello che mantiene le informazioni sulle proprietà del sistema selezionato per rendere il software auto consapevole. Le modifiche alle informazioni mantenute nel meta-livello influenzano quindi il comportamento successivo del livello di base. [200] L'adattamento del meta-livello viene eseguito indirettamente con l'aiuto di una specifica interfaccia, il Meta-Object Protocol (MOP). Questo permette agli utenti di specificare una modifica, controllare la correttezza e integrare automaticamente la modifica nel livello meta. [201] Il MOP rende possibile anche il cambio di connessioni tra il livello di base e i meta-oggetti. Un meta-oggetto è un oggetto che manipola, crea, descrive, o implementa altri oggetti (compreso se stesso). Quindi, una corretta configurazione del livello di base e dei meta-oggetti definiscono il comportamento di un'applicazione. [202] Inoltre, utilizzando un linguaggio di programmazione che supporta la riflessività, è possibile modificare la struttura degli oggetti stessi a tempo di esecuzione e quindi rendere il sistema software più flessibile.

9.2 Architettura Logica del Layer

Il Layer in questione dovrà rendere il Middleware della piattaforma DF conforme al modello di programmazione riflessiva, che consente, in genere, ad un sistema software di cambiare la sua logica senza modifiche interne al codice.

Il modello riflessivo che sarà progettato avrà lo scopo di iniettare consapevolezza dei vari contesti di riferimento nel Middleware della Piattaforma DF. Questa caratteristica, unitamente alla estensione riflessiva consentirà ai servizi applicativi software, realizzati tramite gli strumenti messi a disposizione del Middleware e preposti alla elaborazione dei dati raccolti e gestiti dallo stesso, di cambiare dinamicamente la sua logica senza cambiamenti interni al codice.

Dal punto di vista architetturale, il layer dovrà implementare un modello formale che sia aderente ai tre livelli previsti dal reflective pattern. In pratica dovrà essere possibile sostenere e supportare la definizione dei seguenti livelli:

1. *Condition Level* : livello che definisce le relazioni (Condition) che collegano le misurazioni rilevate da un dispositivo al valore rilevato, ossia le condizioni sul flusso dati.
2. *Action Level*: livello che definisce le operazioni (Action) che devono essere eseguite, quando una condizione si verifica, applicata ad un argomento supportato dall'operazione.
3. *Rule Level* : livello che definisce le regole (Rule) ossia le funzioni che associano una Condition ad una Action.

Il mapping logico esistente tra i livelli suddetti e i livelli previsti dal reflective pattern è il seguente:

- Condition Level --> Meta Level
- Action Level --> Base Level
- Rule Level --> Meta-object Protocol

Di conseguenza, il Reflective Layer avrà la seguente architettura logica:

Figura 51: Architettura Logica del Reflective Layer

Le principali componenti sono il *Meta-Object Store*, il *Meta-Object Editor* e il *Reflective Rule Engine*. Con riferimento ai tre livelli descritti precedentemente, il Meta-Object Store si pone nel Meta Level mentre il Meta-Object Editor e il Reflective Rule Engine sono collocabili nel Metaobject Protocol. Il Base Level non è incluso nel Reflective Layer ma trova collocazione nelle procedure di elaborazione dati create e gestite nel "*Analytical Layer*" sia in modalità Stream Processing e sia Batch Processing e in quelle gestite nel "Event Layer".

9.2.1 *Meta-Object Store*

Questo componente avrà il compito di conservare i meta-object, ossia l'informazione che è implicitamente disponibile, ma esplicitamente accessibile e modificabile. Ciò che sarà rappresentato con i meta-object dipenderà da quello che si vuole rendere adattabile. In pratica, nei meta-object saranno incapsulati solo i dettagli del sistema, in questo caso i servizi applicativi messi a disposizione dal Middleware a supporto della logica di business delle soluzioni verticali che utilizzano la Piattaforma DF e che potrebbero cambiare.

In tale ottica, questo componente dovrà conservare le entità "Condition" e "Rule", nonché i segnaposti per le operazioni (Action) definite nel Base Level. Tali concetti saranno modellati attraverso il componente Meta-Object Editor e costituiranno un elemento base per un *Rule Based System* accedendo al quale il componente Rule Engine implementerà l'algoritmo di reasoning interpretando le regole ed istanzierà dinamicamente le operazioni.

Per essere efficaci nella descrizione dei servizi applicativi che si vorranno rendere adattivi al contesto e agli scenari di interesse ed estensibili al bisogno, i meta-object conservati nel componente Meta-Object Store dovranno necessariamente far riferimento ai flussi dati caratterizzanti il contesto e per la cui elaborazione sono state definite ed implementate le Azioni nel Base Level.

I flussi dati, che dovranno essere qualificati attraverso il Meta-Object Editor durante la fase di definizione delle "Condition" e delle "Rule" da applicare per il collegamento dinamico alle "Action", dovranno essere etichettati con identificativi univoci all'interno del *Rule Based System*.

Grazie alle entità "Condition" e "Rule", al loro collegamento con i flussi dati provenienti dai dispositivi e ai segnaposti delle "Action" da attivare, la logica dei servizi applicativi potrà essere cambiata dinamicamente a seconda dei dati e i cambiamenti potranno essere trasparenti alla struttura interna del software che implementa tali servizi.

Per ogni servizio applicativo da rendere adattivo al contesto, dovrà essere definito un File di Configurazione contenente i meta-object sopra descritti.

I File di Configurazione saranno persistiti, quindi, nel Meta-Object Store e insieme al componente Rule Engine costituiranno il *Rule Based System*.

La consapevolezza del contesto, sarà, pertanto, codificata per mezzo di un sistema basato su regole che guiderà il comportamento dinamico del Middleware.

9.2.2 *Meta-Object Editor*

Compito del livello meta sarà anche quello di implementare un **meta-object protocol interface**, ossia un'interfaccia che gli amministratori e chi si occupa dell'evoluzione del sistema potrà utilizzare per configurare e modificare dinamicamente i meta-oggetti. Poiché l'implementazione degli oggetti contenuti nel Base Level (quindi della logica dei servizi applicativi) si poggia esplicitamente su informazioni contenute nei meta-object, modificando questi ultimi si otterrà un effetto immediato nel cambiamento della logica applicativa dei servizi del Middleware. Per aggiornare o modificare i servizi applicativi del Middleware, quindi, sarà necessario modificare i meta-object, per definire la nuova logica, attraverso il componente Meta-Object Editor. Tramite esso dovrà essere possibile anche collegare dinamicamente il nuovo codice ai servizi applicativi mentre essi sono in esecuzione. Tale componente, dunque, dovrà fornire il centro unico di controllo delle modifiche che sono richieste a un'architettura riflessiva.

Questo componente, attraverso una interfaccia user friendly dovrà guidare l'amministratore a definire i meta-object che caratterizzeranno il servizio applicativo oggetto di implementazione. Saranno, pertanto acquisite informazioni su:

- Servizio Applicativo da modellare
- Casi d'uso che ne caratterizzano le funzionalità

Per ogni Caso d'uso, dovrà essere necessario definire:

- La Struttura del flusso dati, ossia lo la struttura dello event stream descritto in termini di tracciato dati

- L'entità "Condition", da applicare sui valori delle tuple caratterizzanti il flusso dati che dovrà arrivare a run time. Più entità "Condition" dovranno poter essere definite e collegate attraverso logica booleana.
- L'entità "Action", ovvero il segnaposto della operazione che dovrà essere eseguita qualora la "Condition" venga verificata. L'entità "Condition" e la relativa "Action" collegata determineranno la "Rule" che dovrà essere eseguita dal componente "Rule Engine" in fase di elaborazione dei flussi dati entranti nella Piattaforma DF.

Il segnaposto dell'azione, dovrà essere creato consentendo all'utilizzatore dell'Editor di selezionare la funzione e la classe, che saranno successivamente utilizzati per implementare riflessivamente l'operazione associata alla Action, attraverso il browsing del file contenete il codice da eseguire.

Quando definito attraverso il componente Meta-Object Editor per codificare il profilo della applicazione, ovvero le informazioni che, per ogni servizio applicativo, vengono associate per le politiche della configurazione del contesto, dovrà essere salvato in un File di Configurazione, uno per servizio applicativo. La notazione JSON potrebbe essere una possibile scelta per tale codifica.

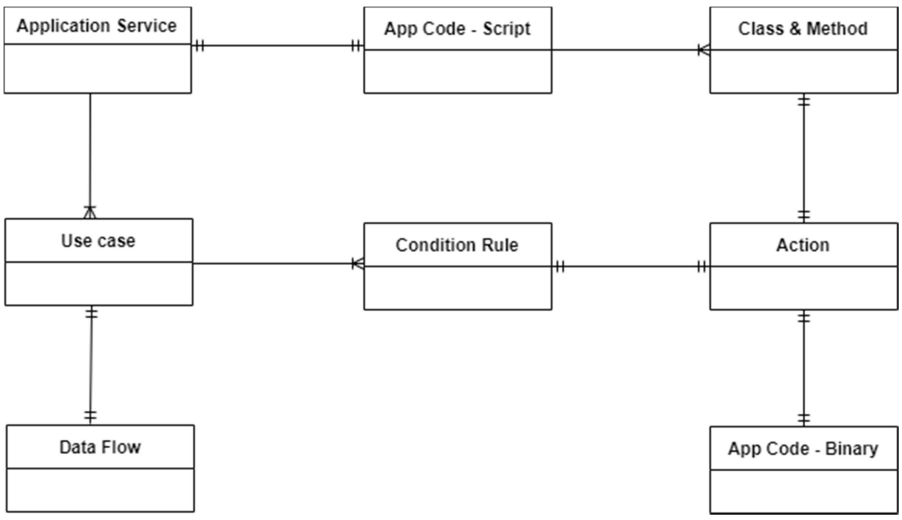


Figura 52: Entità definite nel File di Configurazione e loro Relazioni

9.2.3 *Reflective Rule Engine*

Questo componente avrà il compito di:

1. acquisire i flussi dati entranti nella Piattaforma DF;
2. individuare all'interno del Meta-Object Store i meta-object che ne definiscono il trattamento, ossia il File di Configurazione contenente i meta-object di riferimento;
3. valutare le Condition definite per i dati ricevuti;
4. estrarre la Action dalla Rule;
5. effettuare il mapping del segnaposto della Action con una operazione reale definita nel Base Level, collegando dinamicamente il codice ed eseguendolo sfruttando la riflessività del linguaggio di programmazione.

Come specificato precedentemente, il codice dei servizi applicativi (Action) che implementano parte delle logiche di business delle soluzioni verticali che utilizzano la Piattaforma DF, sarà collocato nelle procedure di elaborazione dati create nel "*Analytical Layer*" ed eseguite sia in modalità Stream Processing e sia Batch Processing e in quelle create e gestite dal "Event Layer".

Le procedure che saranno costruite nel "*Analytical Layer*" saranno costituite da script Python prodotti attraverso il componente Data Science Tool ed erogati attraverso i componenti di Processing Engine.

Le procedure gestite attraverso "Event Layer", invece, saranno job java che eseguono elaborazioni sui dati per l'individuazione ed estrazione di pattern complessi.

Il componente Rule Engine , dovrà essere in grado di poter agganciare dinamicamente queste due tipologie di Action garantendo una interoperabilità tra i layer e garantendo che i servizi applicativi del Middleware e, implicitamente, le soluzioni verticali che lo utilizzano, possano usufruire delle migliori pratiche / tecnologie per la elaborazione dei dati provenienti dai dispositivi.

Il componente Rule Engine dovrà sempre parsificare il File di Configurazione relativo, ogni volta che acquisisce un flusso dati. Questo parsing continuo dovrà rendersi necessario per recepire eventuali modifiche fatte, attraverso il Meta-Object Editor, alle meta entità che definiscono il servizio applicativo di trattamento del flusso, mentre il sistema è in esecuzione. Questa caratteristica consentirà anche di aggiungere "*a caldo*" nuove funzionalità, al trattamento dei dati, per soddisfare eventuali nuovi requisiti che potranno nascere nel dominio della soluzione verticale interessata ai servizi applicativi esposti dal Middleware della Piattaforma DF.

La figura seguente, fornisce una vista architetturale di dettaglio del Reflective Layer.

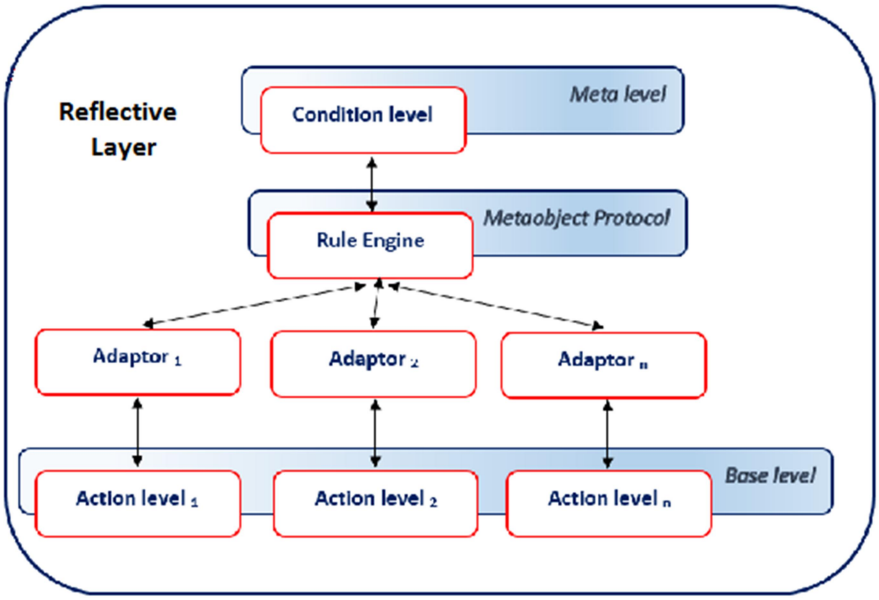


Figura 53: Dettaglio Architetturale del Reflective Layer

10. Bibliografia

1. Marfella S. Che cosa è il middleware di integrazione e perché dovrebbe interessarti [Internet]. [cited 5 Jul 2018]. Available: https://www.extrasys.it/it/redblog/che-cosa-è-il-middleware-di-integrazione-e-perché-dovrebbe-interessarti

2. Santucci G. Architetture Distribuite, Middleware, Distributed Object Computing e Legacy System [Internet]. [cited 5 Jul 2018]. Available: http://www.dis.uniroma1.it/pub/santucci/Dottorato/Dottorato99.PDF

3. Cugola G. Applicazioni distribuite e middleware [Internet]. [cited 5 Jul 2018]. Available: http://home.deib.polimi.it/cugola/pub/ImpII-Middleware.pdf

4. Vandi M. PROGETTAZIONE E SVILUPPO DI UN PROTOTIPO DI PIATTAFORMA PER WEB OF THINGS. Anno Accademico 2016 - 2017.

5. Antenucci. SERVIZI DI DISCOVERY IN IOT:UNA SOLUZIONE MQTT-BASED PER GATEWAY KURA. Università di Bologna. 2014.

6. MQTT Essentials Part 9: Last Will and Testament. In: HIVEMQ - Enterprise MQTT Browser [Internet]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament

7. HollyGraceful. TLS/SSL Vulnerabilities — GracefulSecurity [Internet]. [cited 5 Jul 2018]. Available: https://www.gracefulsecurity.com/tls-ssl-vulnerabilities/

8. MQTT Security Fundamentals: TLS / SSL. In: HIVEMQ - Enterprise MQTT Broker [Internet]. Available: https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl

9. MQTT Essentials Part 6: Quality of Service 0, 1 & 2 [Internet]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels

10. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. 2018; Available: https://tools.ietf.org/html/draft-ietf-tls-tls13-28

11. Can I use... Support tables for HTML5, CSS3, etc [Internet]. [cited 5 Jul 2018]. Available: https://caniuse.com/#feat=tls1-3

12. Sullivan N. Introducing TLS 1.3. In: Cloudflare Blog [Internet]. Cloudflare Blog; 20 Sep 2016 [cited 5 Jul 2018]. Available: https://blog.cloudflare.com/introducing-tls-1-3/

13. [TLS] Industry Concerns about TLS 1.3 [Internet]. [cited 5 Jul 2018]. Available: https://www.ietf.org/mail-archive/web/tls/current/msg21275.html

14. MikeWasson. Stile di architettura di microservizi - Azure Application Architecture Guide [Internet]. [cited 27 Mar 2019]. Available: https://docs.microsoft.com/it-it/azure/architecture/guide/architecture-styles/microservices

15. CESARDELATORRE. Confronto tra schema API Gateway e comunicazione diretta da client a microservizio [Internet]. [cited 27 Mar 2019]. Available: https://docs.microsoft.com/it-it/dotnet/standard/microservices-architecture/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern

16. WSO2 API Manager Documentation - API Manager 2.0.0 - WSO2 Documentation [Internet]. [cited 4 Apr 2019]. Available: https://docs.wso2.com/display/AM200/WSO2+API+Manager+Documentation

17. Edwards J. Identity Management Glossary of Terms. In: Top Identity & Access Management Software, Vendors, Products, Solutions, & Services [Internet]. Top Identity & Access Management Software, Vendors, Products, Solutions, & Services; 23 Nov 2015 [cited 4 Apr 2019]. Available: https://solutionsreview.com/identity-management/identity-management-glossary/

18. Gill NS. Data Ingestion, Processing and Big Data Architecture Layers - XenonStack

Blog. In: Data Ingestion, Processing and Big Data Architecture Layers - XenonStack Blog [Internet]. 3 Mar 2017 [cited 5 Jul 2018]. Available: https://www.xenonstack.com/blog/big-data-engineering/ingestion-processing-big-data-iot-stream/

19. Big Data Design Patterns. In: Big Data Design Patterns [Internet]. [cited 5 Jul 2018]. Available: http://www.bigdatapatterns.org/

20. Identity Management vs Access Management [Internet]. [cited 5 Apr 2019]. Available: https://www.globalsign.com/en/blog/identity-management-vs-access-management/

21. Khan AR. Access control in cloud computing environment. ARPN Journal of Engineering and Applied Sciences. 2012;7: 613–615.

22. Willens S, Rubens AC, Rigney C, Simpson WA. Remote Authentication Dial In User Service (RADIUS). 1997; Available: https://tools.ietf.org/html/rfc2138

23. The TACACS+ Protocol [Internet]. [cited 8 Apr 2019]. Available: https://tools.ietf.org/id/draft-ietf-opsawg-tacacs-08.html

24. What is TACACS (Terminal Access Controller Access Control System)? - Definition from WhatIs.com. In: SearchSecurity [Internet]. [cited 8 Apr 2019]. Available: https://searchsecurity.techtarget.com/definition/TACACS

25. ScienceDirect [Internet]. [cited 15 Apr 2019]. Available: https://www.sciencedirect.com/topics/computer-science/ticket-granting-service

26. MIT Kerberos Consortium - Protocol Tutorial [Internet]. [cited 15 Apr 2019]. Available: https://kerberos.org/software/tutorial.html

27. Spiegazione protocollo Open Authorization OAuth 2.0 - Noir Solutions. In: Noir Solutions [Internet]. [cited 4 Apr 2019]. Available: https://www.noirsolutions.net/virus-informatici-malware/spiegazione-protocollo-

open-authorization-oauth-2-0/

28. Jones M. JSON Web Signature (JWS) Unencoded Payload Option. 2016; Available: https://tools.ietf.org/html/rfc7797

29. Hunter T. Deep Learning with Apache Spark and TensorFlow - The Databricks Blog. In: Databricks [Internet]. 25 Jan 2016 [cited 10 May 2019]. Available: https://databricks.com/blog/2016/01/25/deep-learning-with-apache-spark-and-tensorflow.html

30. Boldrini N. Deep Learning: cos'è, come funziona e quali sono i casi d'uso. In: AI4Business [Internet]. 16 Oct 2018 [cited 10 May 2019]. Available: https://www.ai4business.it/intelligenza-artificiale/deep-learning/deep-learning-cose/

31. Patterson J, Gibson A. Deep Learning: A Practitioner's Approach. "O'Reilly Media, Inc."; 2017.

32. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE. A survey of deep neural network architectures and their applications [Internet]. Neurocomputing. 2017. pp. 11–26. doi:10.1016/j.neucom.2016.12.038

33. The Multilayer Perceptron [Internet]. Connectionism. pp. 108–113. doi:10.1002/9780470694077.ch17

34. LeCun Y, Bottou L, Orr GB, Müller K-R. Efficient BackProp [Internet]. Lecture Notes in Computer Science. 1998. pp. 9–50. doi:10.1007/3-540-49430-8_2

35. LeCun Y. Learning Invariant Feature Hierarchies [Internet]. Computer Vision – ECCV 2012. Workshops and Demonstrations. 2012. pp. 496–505. doi:10.1007/978-3-642-33863-2_51

36. Chen H, Murray AF. Continuous restricted Boltzmann machine with an implementable training algorithm [Internet]. IEE Proceedings - Vision, Image, and

Signal Processing. 2003. p. 153. doi:10.1049/ip-vis:20030362

37. Nifong N. Learning General Features From Images and Audio With Stacked Denoising Autoencoders [Internet]. 2000. doi:10.15760/etd.1549

38. Barello G, Charles A, Pillow J. Sparse-Coding Variational Auto-Encoders [Internet]. doi:10.1101/399246

39. Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets. Neural Comput. 2006;18: 1527–1554.

40. Deng L, Yu D, Platt J. Scalable stacking and learning for building deep architectures [Internet]. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2012. doi:10.1109/icassp.2012.6288333

41. Kalin J. Generative Adversarial Networks Cookbook: Over 100 recipes to build generative models using Python, TensorFlow, and Keras. Packt Publishing Ltd; 2018.

42. Zhang H, Xu T, Li H. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks [Internet]. 2017 IEEE International Conference on Computer Vision (ICCV). 2017. doi:10.1109/iccv.2017.629

43. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge [Internet]. International Journal of Computer Vision. 2015. pp. 211–252. doi:10.1007/s11263-015-0816-y

44. Lawrence S, Giles CL, Tsoi AC, Back AD. Face recognition: a convolutional neural-network approach. IEEE Trans Neural Netw. 1997;8: 98–113.

45. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks [Internet]. Communications of the ACM. 2017. pp. 84–90. doi:10.1145/3065386

46. Kalchbrenner N, Grefenstette E, Blunsom P. A Convolutional Neural Network for Modelling Sentences [Internet]. Proceedings of the 52nd Annual Meeting of the

Association for Computational Linguistics (Volume 1: Long Papers). 2014. doi:10.3115/v1/p14-1062

47. Piczak KJ. Environmental sound classification with convolutional neural networks [Internet]. 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP). 2015. doi:10.1109/mlsp.2015.7324337

48. Gehring J, Auli M, Grangier D, Dauphin Y. A Convolutional Encoder Model for Neural Machine Translation [Internet]. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017. doi:10.18653/v1/p17-1012

49. Severyn A, Moschitti A. Twitter Sentiment Analysis with Deep Convolutional Neural Networks [Internet]. Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '15. 2015. doi:10.1145/2766462.2767830

50. Funahashi K-I, Nakamura Y. Approximation of dynamical systems by continuous time recurrent neural networks [Internet]. Neural Networks. 1993. pp. 801–806. doi:10.1016/s0893-6080(05)80125-x

51. Wang F, Guo Q, Lei J, Zhang J. Convolutional recurrent neural networks with hidden Markov model bootstrap for scene text recognition [Internet]. IET Computer Vision. 2017. pp. 497–504. doi:10.1049/iet-cvi.2016.0417

52. Amberkar A, Awasarmol P, Deshmukh G, Dave P. Speech Recognition using Recurrent Neural Networks [Internet]. 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT). 2018. doi:10.1109/icctct.2018.8551185

53. Sheikholharam P, Teshnehlab M. Music Composition Using Combination of Genetic Algorithms and Recurrent Neural Networks [Internet]. 2008 Eighth International Conference on Hybrid Intelligent Systems. 2008. doi:10.1109/his.2008.46

54. Hochreiter S, Schmidhuber J. Long Short-Term Memory [Internet]. Neural Computation. 1997. pp. 1735–1780. doi:10.1162/neco.1997.9.8.1735

55. Tjandra A, Sakti S, Manurung R, Adriani M, Nakamura S. Gated Recurrent Neural Tensor Network [Internet]. 2016 International Joint Conference on Neural Networks (IJCNN). 2016. doi:10.1109/ijcnn.2016.7727233

56. Bowman SR, Potts C, Manning CD. Recursive Neural Networks Can Learn Logical Semantics [Internet]. Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality. 2015. doi:10.18653/v1/w15-4002

57. What is correlation? - Definition from WhatIs.com. In: WhatIs.com [Internet]. [cited 27 Mar 2019]. Available: https://whatis.techtarget.com/definition/correlation

58. Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media; 2013.

59. Machine Learning Algorithms for Recommender System - a comparative analysis. Volume 6–Issue 2, 97-100, 2017, ISSN:-2319–8656.

60. Hamilton LD. Machine Learning Algorithm Cheat Sheet - Laura Diane Hamilton [Internet]. [cited 10 May 2019]. Available: http://www.lauradhamilton.com/machine-learning-algorithm-cheat-sheet

61. The Most Important Machine Learning Algorithms. In: Semantic Bits [Internet]. [cited 10 May 2019]. Available: https://semanti.ca/blog/?the-most-important-machine-learning-algorithms

62. Modern Machine Learning Algorithms: Strengths and Weaknesses. In: EliteDataScience [Internet]. 16 May 2017 [cited 10 May 2019]. Available: https://elitedatascience.com/machine-learning-algorithms

63. HackingNote [Internet]. [cited 10 May 2019]. Available: https://www.hackingnote.com/en/machine-learning/algorithms-pros-and-cons

64. Mueller JP, Massaron L. Machine Learning For Dummies. John Wiley & Sons; 2016.

65. Markham K. Comparing supervised learning algorithms. In: Data School [Internet]. Data School; 27 Feb 2015 [cited 10 May 2019]. Available: https://www.dataschool.io/comparing-supervised-learning-algorithms/

66. Malik F. Machine Learning Algorithms Comparison. In: Medium [Internet]. FinTechExplained; 27 Aug 2018 [cited 10 May 2019]. Available: https://medium.com/fintechexplained/machine-learning-algorithm-comparison-f14ce372b855

67. Common supervised learning models – (pros and cons) [Internet]. [cited 10 May 2019]. Available: http://www.designemergente.com/common-supervised-learning-models-pros-and-cons/

68. ctufts. ctufts/Cheat_Sheets. In: GitHub [Internet]. [cited 10 May 2019]. Available: https://github.com/ctufts/Cheat_Sheets

69. Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms [Internet]. Proceedings of the 23rd international conference on Machine learning - ICML '06. 2006. doi:10.1145/1143844.1143865

70. Valcheva S. Supervised vs Unsupervised Learning: algorithms, example, difference. In: Business Intelligence, Data Science, and Management [Internet]. 11 Mar 2018 [cited 10 May 2019]. Available: http://intellspot.com/unsupervised-vs-supervised-learning/

71. Harlalka R. Choosing the Right Machine Learning Algorithm. In: Hacker Noon [Internet]. Hacker Noon; 18 Jun 2018 [cited 10 May 2019]. Available: https://hackernoon.com/choosing-the-right-machine-learning-algorithm-68126944ce1f

72. Czarnowski I, Jedrzejowicz P, Chao K-M, Yildirim T. Overcoming "Big Data" Barriers in Machine Learning Techniques for the Real-Life Applications [Internet].

Complexity. 2018. pp. 1–3. doi:10.1155/2018/1234390

73. Korbut D. Machine Learning Algorithms: Which One to Choose for Your Problem. In: Stats and Bots [Internet]. Stats and Bots; 26 Oct 2017 [cited 10 May 2019]. Available: https://blog.statsbot.co/machine-learning-algorithms-183cc73197c

74. Harrell F. Road Map for Choosing Between Statistical Modeling and Machine Learning | Statistical Thinking. In: Statistical Thinking [Internet]. 30 Apr 2018 [cited 10 May 2019]. Available: http://fharrell.com/post/stat-ml/

75. Website [Internet]. [cited 10 May 2019]. Available: https://www.datasciencecentral.com/profiles/blogs/how-to-choose-a-machine-learning-model-someguidelines

76. Donges N. Pros and Cons of Neural Networks. In: Towards Data Science [Internet]. Towards Data Science; 17 Apr 2018 [cited 10 May 2019]. Available: https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b

77. Classification and regression - Spark 2.4.3 Documentation [Internet]. [cited 10 May 2019]. Available: https://spark.apache.org/docs/latest/ml-classification-regression.html#classification

78. TensorFlow Guide | TensorFlow Core | TensorFlow. In: TensorFlow [Internet]. [cited 10 May 2019]. Available: https://www.tensorflow.org/guide

79. Wolpert DH. The Supervised Learning No-Free-Lunch Theorems [Internet]. Soft Computing and Industry. 2002. pp. 25–42. doi:10.1007/978-1-4471-0123-9_3

80. Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK. Occam's Razor [Internet]. Information Processing Letters. 1987. pp. 377–380. doi:10.1016/0020-0190(87)90114-1

81. scikit-learn: machine learning in Python — scikit-learn 0.21.2 documentation

[Internet]. [cited 23 Jul 2019]. Available: https://scikit-learn.org/stable/

82. sklearn.cluster.DBSCAN — scikit-learn 0.21.2 documentation [Internet]. [cited 23 Jul 2019]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

83. Scioscia F, Ruta M. Building a Semantic Web of Things: Issues and Perspectives in Information Compression [Internet]. 2009 IEEE International Conference on Semantic Computing. 2009. doi:10.1109/icsc.2009.75

84. [No title] [Internet]. [cited 15 May 2019]. Available: http://www.ietf.org/rfc/rfc3870.txt

85. OWL Web Ontology Language Reference [Internet]. [cited 15 May 2019]. Available: http://www.w3.org/TR/owl-ref/

86. John Snow Labs Spark-NLP [Internet]. [cited 17 May 2019]. Available: https://nlp.johnsnowlabs.com/

87. The Stanford NLP Group [Internet]. [cited 17 May 2019]. Available: https://nlp.stanford.edu/software/lex-parser.shtml

88. Agerri R, Bermudez J, Rigau G. Multilingual, Efficient and Easy NLP Processing with IXA Pipeline [Internet]. Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics. 2014. doi:10.3115/v1/e14-2002

89. Ma, Zongmin. Artificial Intelligence for Maximizing Content Based Image Retrieval. IGI Global; 2009.

90. Ruta M, Di Noia T, Di Sciascio E, Donini FM. Semantic-enhanced Bluetooth discovery protocol for m-commerce applications [Internet]. International Journal of Web and Grid Services. 2006. p. 424. doi:10.1504/ijwgs.2006.011713

91. Virgilio RD, De Virgilio R, Di Sciascio E, Ruta M, Scioscia F, Torlone R. Semantic-

Based RFID Data Management [Internet]. Unique Radio Innovation for the 21st Century. 2011. pp. 111–141. doi:10.1007/978-3-642-03462-6_6

92. Ruta M, Scioscia F, Di Sciascio E, Loseto G. Semantic-Based Enhancement of ISO/IEC 14543-3 EIB/KNX Standard for Building Automation [Internet]. IEEE Transactions on Industrial Informatics. 2011. pp. 731–739. doi:10.1109/tii.2011.2166792

93. Ruta M, Scioscia F, Pinto A, Di Sciascio E, Gramegna F, Ieva S, et al. Resource Annotation, Dissemination and Discovery in the Semantic Web of Things: A CoAP-Based Framework [Internet]. 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing. 2013. doi:10.1109/greencom-ithings-cpscom.2013.103

94. Scioscia F, Ruta M, Loseto G, Gramegna F, Ieva S, Pinto A, et al. A Mobile Matchmaker for the Ubiquitous Semantic Web [Internet]. International Journal on Semantic Web and Information Systems. 2014. pp. 77–100. doi:10.4018/ijswis.2014100104

95. Ruta M, Scioscia F, Ieva S, Loseto G, Gramegna F, Pinto A. Knowledge discovery and sharing in the IoT [Internet]. Proceedings of the Symposium on Applied Computing - SAC '17. 2017. doi:10.1145/3019612.3019701

96. The Physical Web [Internet]. [cited 17 May 2019]. Available: https://google.github.io/physical-web/

97. SEMANTIC TURKEY - A New Web Experience in between Ontology Editing and Semantic Annotation [Internet]. Proceedings of the Fourth International Conference on Web Information Systems and Technologies. 2008. doi:10.5220/0001526600900097

98. Kahan J, -R. Koivunen M, Prud'Hommeaux E, Swick RR. Annotea: an open RDF infrastructure for shared Web annotations [Internet]. Computer Networks. 2002.

pp. 589–608. doi:10.1016/s1389-1286(02)00220-7

99. KIM – semantic annotation platform. In: Ontotext [Internet]. [cited 30 May 2019]. Available: https://www.ontotext.com/knowledgehub/publications/17141-2/

100. Wartena C, Brussee R, Gazendam L, Huijsen W-O. Apolda: A Practical Tool for Semantic Annotation [Internet]. 18th International Conference on Database and Expert Systems Applications (DEXA 2007). 2007. doi:10.1109/dexa.2007.4312903

101. Apache Jena - [Internet]. [cited 30 May 2019]. Available: http://jena.apache.org/

102. Sesame - Semantic Web Standards [Internet]. [cited 30 May 2019]. Available: https://www.w3.org/2001/sw/wiki/Sesame

103. OpenLink Virtuoso [Internet]. [cited 30 May 2019]. Available: https://virtuoso.openlinksw.com/

104. 4store - Semantic Web Standards [Internet]. [cited 30 May 2019]. Available: https://www.w3.org/2001/sw/wiki/4store

105. Owlim - Semantic Web Standards [Internet]. [cited 30 May 2019]. Available: https://www.w3.org/2001/sw/wiki/Owlim

106. The D2RQ Platform – Accessing Relational Databases as Virtual RDF Graphs [Internet]. [cited 30 May 2019]. Available: http://d2rq.org/

107. OWL API [Internet]. [cited 30 May 2019]. Available: http://owlapi.sourceforge.net/

108. Ferreyra D. TemaTres Controlled Vocabulary server [Internet]. [cited 30 May 2019]. Available: https://www.vocabularyserver.com/

109. Scioscia F, Ruta M, Loseto G, Gramegna F, Ieva S, Pinto A, et al. Mini-ME Matchmaker and Reasoner for the Semantic Web of Things. Innovations, Developments, and Applications of Semantic Web and Information Systems. 2018; 262–294.

110. Ruta M, Di Noia T, Di Sciascio E, Scioscia F. Abduction and Contraction for Semantic-Based Mobile Dating in P2P Environments [Internet]. 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. 2008. doi:10.1109/wiiat.2008.413

111. Bossam Rule/OWL Reasoner. In: Bossam Rule/OWL Reasoner [Internet]. [cited 30 May 2019]. Available: https://bossam.wordpress.com/

112. RacerPro [Internet]. [cited 30 May 2019]. Available: https://franz.com/agraph/racer/

113. OntoBroker - Semantic Web Standards [Internet]. [cited 30 May 2019]. Available: https://www.w3.org/2001/sw/wiki/OntoBroker

114. ResearchCyc [Internet]. [cited 30 May 2019]. Available: https://web.archive.org/web/20120207001801/http://research.cyc.com/

115. KAON2 -- Ontology Management for the Semantic Web [Internet]. [cited 30 May 2019]. Available: http://kaon2.semanticweb.org/

116. Software. In: TU Dresden [Internet]. 25 Oct 2017 [cited 30 May 2019]. Available: https://tu-dresden.de/ing/informatik/thi/lat/forschung/software/?set_language=de

117. cwm - a general purpose data processor for the semantic web [Internet]. [cited 30 May 2019]. Available: https://www.w3.org/2000/10/swap/doc/cwm.html

118. josd. josd/eye. In: GitHub [Internet]. [cited 30 May 2019]. Available: https://github.com/josd/eye

119. Drools - Drools - Business Rules Management System (JavaTM, Open Source) [Internet]. [cited 30 May 2019]. Available: http://www.drools.org/

120. liveontologies. liveontologies/elk-reasoner. In: GitHub [Internet]. [cited 30 May 2019]. Available: https://github.com/liveontologies/elk-reasoner

121. FaCT++ reasoner | OWL research at the University of Manchester [Internet]. [cited 31 May 2019]. Available: http://owl.cs.manchester.ac.uk/tools/fact/

122. Flora-2 [Internet]. [cited 31 May 2019]. Available: http://flora.sourceforge.net/

123. HermiT Reasoner: Home [Internet]. [cited 31 May 2019]. Available: http://www.hermit-reasoner.com/

124. julianmendez. julianmendez/jcel. In: GitHub [Internet]. [cited 31 May 2019]. Available: https://github.com/julianmendez/jcel

125. Apache Jena - [Internet]. [cited 31 May 2019]. Available: http://jena.apache.org/

126. OpenRules Solver [Internet]. [cited 31 May 2019]. Available: http://openrules.com/rulesolver.htm

127. stardog-union. stardog-union/pellet. In: GitHub [Internet]. [cited 31 May 2019]. Available: https://github.com/stardog-union/pellet

128. ha-mo-we. ha-mo-we/Racer. In: GitHub [Internet]. [cited 31 May 2019]. Available: https://github.com/ha-mo-we/Racer

129. Guarino N. Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98), June 6-8, Trento, Italy. IOS Press; 1998.

130. Gyrard A. Linked Open Vocabularies for Internet of Things (LOV4IoT) [Internet]. [cited 3 Jun 2019]. Available: https://lov4iot.appspot.com/?p=ontologies

131. Semantic Sensor Network Ontology [Internet]. [cited 3 Jun 2019]. Available: https://www.w3.org/TR/vocab-ssn/

132. On the usage of the SSN ontology [Internet]. [cited 3 Jun 2019]. Available: https://w3c.github.io/sdw/ssn-usage/

133. Extensions to the Semantic Sensor Network Ontology [Internet]. [cited 3 Jun 2019].

Available: https://www.w3.org/TR/vocab-ssn-ext/

134. Website [Internet]. [cited 3 Jun 2019]. Available: https://www.w3.org/Submission/iot-lite/

135. Review of Sensor and Observations Ontologies - Semantic Sensor Network Incubator Group [Internet]. [cited 3 Jun 2019]. Available: https://www.w3.org/2005/Incubator/ssn/wiki/Review_of_Sensor_and_Observations_Ontologies

136. Ontology repositories - W3C Wiki [Internet]. [cited 3 Jun 2019]. Available: https://www.w3.org/wiki/Ontology_repositories

137. PeterYim S. OOR - Open Ontology Repository [Internet]. [cited 3 Jun 2019]. Available: http://oor.net/

138. Ontology Repositories | OWL research at the University of Manchester [Internet]. [cited 3 Jun 2019]. Available: http://owl.cs.manchester.ac.uk/tools/repositories/

139. AberOWL [Internet]. [cited 3 Jun 2019]. Available: http://aber-owl.net/#/

140. VIVO Ontologies Repository [Internet]. [cited 3 Jun 2019]. Available: http://swl.slis.indiana.edu/repository/

141. Protege Ontology Library - Protege Wiki [Internet]. [cited 3 Jun 2019]. Available: https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library

142. Nocera F, di Noia T, Mongiello M, Di Sciascio E. Semantic IoT Middleware-enabled Mobile Complex Event Processing for Integrated Pest Management [Internet]. Proceedings of the 7th International Conference on Cloud Computing and Services Science. 2017. doi:10.5220/0006369506380645

143. Jing X, Zhang J, Junhuai LI. Overview of complex event processing technology and its application in logistics Internet of Things [Internet]. Journal of Computer Applications. 2013. pp. 2026–2030. doi:10.3724/sp.j.1087.2013.02026

144. Carbone P, Ewen S, Fóra G, Haridi S, Richter S, Tzoumas K. State management in Apache Flink® [Internet]. Proceedings of the VLDB Endowment. 2017. pp. 1718–1729. doi:10.14778/3137765.3137777

145. Alexandrov A, Bergmann R, Ewen S, Freytag J-C, Hueske F, Heise A, et al. The Stratosphere platform for big data analytics [Internet]. The VLDB Journal. 2014. pp. 939–964. doi:10.1007/s00778-014-0357-y

146. Hueske F, Kalavri V. Stream Processing with Apache Flink: Fundamentals, Implementation, and Operation of Streaming Applications. "O'Reilly Media, Inc."; 2019.

147. Apache Flink 1.7 Documentation: FlinkCEP - Complex event processing for Flink [Internet]. [cited 18 Apr 2019]. Available: https://ci.apache.org/projects/flink/flink-docs-release-1.7/dev/libs/cep.html

148. Clemente PJ, Lozano-Tello A. Model Driven Development Applied to Complex Event Processing for Near Real-Time Open Data. Sensors . 2018;18. doi:10.3390/s18124125

149. DataFlair Team. Flink CEP - Complex Event Processing with Flink - DataFlair. In: DataFlair [Internet]. 11 Feb 2017 [cited 18 Apr 2019]. Available: https://data-flair.training/blogs/flink-cep/

150. Semantic Sensor Network Ontology [Internet]. [cited 18 Apr 2019]. Available: https://www.w3.org/TR/vocab-ssn

151. fiesta-iot. fiesta-iot/ontology. In: GitHub [Internet]. [cited 18 Apr 2019]. Available: https://github.com/fiesta-iot/ontology

152. MAS / BaseOntology · GitLab [Internet]. [cited 18 Apr 2019]. Available: https://git.onem2m.org/MAS/BaseOntology

153. Smart Appliances REFerence (SAREF) ontology - Smart Appliances Project [Internet].

[cited 18 Apr 2019]. Available: https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology

154. Shaw R, Troncy R, Hardman L. LODE: Linking Open Descriptions of Events [Internet]. The Semantic Web. 2009. pp. 153–167. doi:10.1007/978-3-642-10871-6_11

155. Raimond Y. Event Ontology [Internet]. [cited 18 Apr 2019]. Available: http://motools.sourceforge.net/event/event.html#

156. Time Event Ontology | NCBO BioPortal [Internet]. [cited 18 Apr 2019]. Available: https://bioportal.bioontology.org/ontologies/TEO

157. The Simple Event Model [Internet]. [cited 18 Apr 2019]. Available: https://semanticweb.cs.vu.nl/2009/11/sem/

158. Bob Schloss Rosario Uceda-Sosa. Intelligent Semantic Model Palette. In: GitHub [Internet]. [cited 18 Apr 2019]. Available: https://github.com/rschloss/ismp

159. Tsampoulatidis CI, Ntrigkogia V. Dem@Care Project - Ontologies [Internet]. [cited 18 Apr 2019]. Available: http://www.demcare.eu/results/ontologies

160. Pfisterer D, Romer K, Bimschas D, Kleine O, Mietz R, Truong C, et al. SPITFIRE: toward a semantic web of things [Internet]. IEEE Communications Magazine. 2011. pp. 40–48. doi:10.1109/mcom.2011.6069708

161. Shekarpour S, Alshargi F, Thirunaravan K, Shalin VL, Sheth A. CEVO: Comprehensive EVent Ontology Enhancing Cognitive Annotation on Relations [Internet]. 2019 IEEE 13th International Conference on Semantic Computing (ICSC). 2019. doi:10.1109/icosc.2019.8665605

162. Segers R, Caselli T, Vossen P. The Circumstantial Event Ontology (CEO) [Internet]. Proceedings of the Events and Stories in the News Workshop. 2017. doi:10.18653/v1/w17-2706

163. Apache Kafka. In: Apache Kafka [Internet]. [cited 18 Apr 2019]. Available: https://kafka.apache.org/

164. C-SPARQL – Stream Reasoning [Internet]. [cited 18 Apr 2019]. Available: http://streamreasoning.org/resources/c-sparql

165. W3C XML Schema [Internet]. [cited 18 Apr 2019]. Available: http://www.w3.org/XML/Schema

171. Gunning, D.: Explainable artificial intelligence (XAI). Defense Advanced Research Projects Agency (DARPA) (2017)

172. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)

173. Doshi-Velez, F., Kortz, M., Budish, R., Bavitz, C., Gershman, S., O'Brien, D., Schieber, S., Waldo, J., Weinberger, D., Wood, A.: Accountability of AI under the law: The role of explanation. Berkman Center Research Publication Forthcoming; Harvard Public Law Working Paper No. 18-07 (2017)

174. Cherkassky, V., Dhar, S.: Interpretation of black-box predictive models. In: Measures of Complexity, pp. 267–286. Springer, New York (2015)

175. Holzinger, A., Biemann, C., Pattichis, C.S., Kell, D.B.: What do we need to build explainable AI systems for the medical domain? arXiv preprint arXiv:1712.09923 (2017)

176. Holzinger, A., Kieseberg, P., Weippl, E., Tjoa, A.M.: Current advances, trends and challenges of machine learning and knowledge extraction: From machine learning to explainable AI. In: Holzinger, A., Kieseberg, P., Tjoa, A.M., Weippl, E. (eds.) Machine Learning and Knowledge Extraction. pp. 1–8. Springer, Cham (2018)

177. Wang, P., Wu, Q., Shen, C., Dick, A., Van Den Henge, A.: Explicit knowledge-based reasoning for visual question answering. In: Proceedings of the 26th International

Joint Conference on Artificial Intelligence. pp. 1290–1296. IJCAI'17, AAAI Press (2017)

178. Batet, M., Valls, A., Gibert, K.: Performance of ontology-based semantic similarities in clustering. In: Proceedings of the 10th International Conference on Artificial Intelligence and Soft Computing. pp. 281–288. Springer, Berlin, Heidelberg (2010)

179. Batet, M., Valls, A., Gibert, K., Sanchez, D.: Semantic clustering using multiple ontologies. In: Artificial Intelligence Research and Development - Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence. pp. 207–216. IOS Press, Amsterdam, The Netherlands (2010)

200. Frank Buschmann, Kevlin Henney, and Douglas C Schmidt. Pattern-Oriented Software Architecture, Volume 4, A Pattern Language for Distributed Computing. Wiley, 2007.

201. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern-oriented software architecture: a system of patterns. John Wiley & Sons, Inc., New York, NY, USA, 1996.

202. Pattie Maes. Concepts and experiments in computational reflection. In ACM Sigplan Notices, volume 22, pages 147–155. ACM, 1987.

203. M. Mongiello, T. di Noia, F. Nocera, E. di Sciascio and A. Parchitelli, "Context-aware design of reflective midlleware in the internet of everything", in Federation of International Conferences on Software Technologies: Applications and Foundations. Springer, 2016, pp. 423-435

204. Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. Communications Surveys & Tutorials, IEEE, 16(1):414–454, 2014.