

Indice della Documentazione: Onto-Maintenance

Componenti del gruppo: [Grieco Emanuele, 796097,
e.grieco9@studenti.uniba.it]

Link GitHub: <https://github.com/EmaG12/OntoMaintenance.git>

A.A.: 2025-2026

SOMMARIO

Capitolo 0) Introduzione	2
0.1 Obiettivo del Progetto	2
0.2 Descrizione del Dominio e Scenario Operativo	3
0.3 Architettura e Scelte Tecnologiche	4
0.4 Modulo Data-Driven (Baseline)	5
0.5 Struttura del Documento	6
Capitolo 1) Generazione dei Dati e Simulazione	7
1.1 Motivazione della Simulazione	7
1.2 Logica di Generazione Probabilistica	8
1.3 Struttura del Dataset Finale	10
Capitolo 2) Rappresentazione della Conoscenza (Ontologie)	12
2.1 L'Ontologia come Schema Causale	12
2.2 T-Box: Classi e Proprietà	13
2.3 A-Box: Istanziazione dell'Impianto	14
2.4 Gestione Dinamica tramite Python	15
Capitolo 3) Ragionamento Probabilistico: Il Modello Onto-Bayes	16
3.1 Un Approccio Ibrido (Neuro-Simbolico)	16
3.2 Algoritmo di Costruzione Strutture	17
3.3 Apprendimento dei Parametri (Parameter Learning)	18
3.4 Inferenza Esatta (Variable Elimination)	19
Capitolo 4) Apprendimento Supervisionato: Baseline (Random Forest)	20
4.1 Ruolo della Baseline	20
4.2 Configurazione e Feature Engineering	20
4.3 Confronto Teorico: Data-Driven vs Knowledge-Based	21
4.4 Feature Importance	22
Capitolo 5) Valutazione Sperimentale	23
5.1 Metodologia di Validazione	23
5.2 Metriche Utilizzate	24

5.3 Analisi dei Risultati	25
Conclusioni e Sviluppi Futuri.....	28
Risultati Raggiunti:	28
Sviluppi Futuri: Per estendere il lavoro, si propongono le seguenti linee di ricerca:.....	28
Riferimenti Bibliografici	29

CAPITOLO 0) INTRODUZIONE

0.1 OBIETTIVO DEL PROGETTO

Il progetto **Onto-Maintenance** nasce con l'obiettivo di affrontare il problema della diagnostica industriale (Fault Detection and Diagnosis) confrontando due paradigmi distinti dell'Intelligenza Artificiale: l'approccio **Data-Driven** (puramente statistico) e l'approccio **Knowledge-Based** (basato sulla conoscenza e sul ragionamento probabilistico).

Nello specifico, il sistema simula un impianto di trattamento fluidi composto da pompe, tubazioni e serbatoi, soggetto a guasti e monitorato da sensori rumorosi.

L'obiettivo è identificare correttamente il guasto alla radice (es. *Pump Failure*) basandosi sulle letture dei sensori a valle (*Pressure, Tank Level*).

Il valore aggiunto del progetto risiede nell'implementazione di un'architettura **Neuro-Simbolica** (o *Hybrid AI*): invece di definire manualmente la struttura della Rete Bayesiana, il sistema utilizza un'**Ontologia OWL** per descrivere la topologia dell'impianto e le relazioni causali.

Un algoritmo custom analizza la semantica dell'ontologia per generare automaticamente il grafo della Rete Bayesiana, unendo così la flessibilità della rappresentazione della conoscenza alla robustezza del ragionamento probabilistico.

NOTA TEORICA: Un sistema intelligente non si limita a processare dati, ma deve incorporare una rappresentazione interna del mondo.

A differenza di un approccio "black-box" (come le sole reti neurali o alberi decisionali), un KBS (*Knowledge-Based System*) separa la base di conoscenza dal motore inferenziale, garantendo maggiore trasparenza e *explainability* delle decisioni prese.

0.2 DESCRIZIONE DEL DOMINIO E SCENARIO OPERATIVO

Il dominio di applicazione scelto è quello della manutenzione predittiva.

Abbiamo modellato un sistema fisico in cui la causalità è ben definita ma l'osservazione è incerta.

Lo scenario simulato prevede:

1. **Componenti:** Una pompa (*Pump*) connessa a un tubo (*Pipe*) che alimenta un serbatoio (*Tank*).
2. **Guasti (Target):** La pompa può guastarsi (*Pump_Fail* = True).
3. **Sintomi (Features):**
 - Se la pompa si guasta, la pressione nel tubo scende (*Pipe_Pressure* = Low).
 - Se la pressione è bassa, il livello del serbatoio scende (*Tank_Level* = Low).
4. **Incertezza:** I sensori non sono perfetti (rumore statistico) e le relazioni causali non sono deterministiche (es. il livello potrebbe scendere anche per una perdita nel serbatoio, non solo per la pompa).

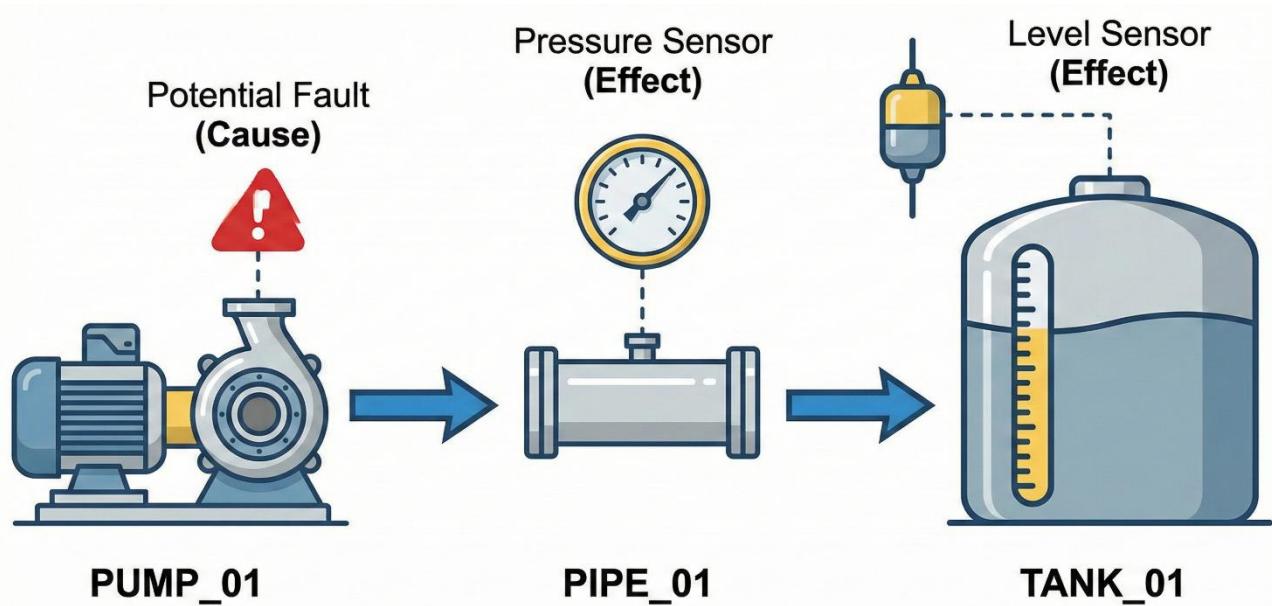


Fig. 0.1 - Topologia dell'impianto simulato e flusso causale del guasto.

Nota Teorica: Nei problemi reali l'agente ha una visione parziale del mondo.

Non possiamo osservare direttamente lo stato interno della pompa (variabile latente), ma solo i suoi effetti (evidenza).

La scelta di utilizzare una Rete Bayesiana permette di modellare questa incertezza tramite la teoria della probabilità, aggiornando la credenza sul guasto ($P(\text{Guasto}|\text{Evidenza})$) ogni volta che nuovi dati dai sensori sono disponibili.

0.3 ARCHITETTURA E SCELTE TECNOLOGICHE

Per realizzare il confronto, sono stati sviluppati due sottosistemi principali utilizzando Python (v3.8+):

1. Modulo Knowledge-Driven (OntoBayes)

È il cuore innovativo del progetto.

Non si limita a usare una rete bayesiana statica, ma la costruisce dinamicamente.

- **Libreria owlready2:** Scelta per la gestione dell'ontologia (file .owl).

Permette di caricare la T-Box (classi Component, Sensor) e la A-Box (istanze Pump_01, Sensor_Pressure) e di navigare le relazioni semantiche come connectedTo.

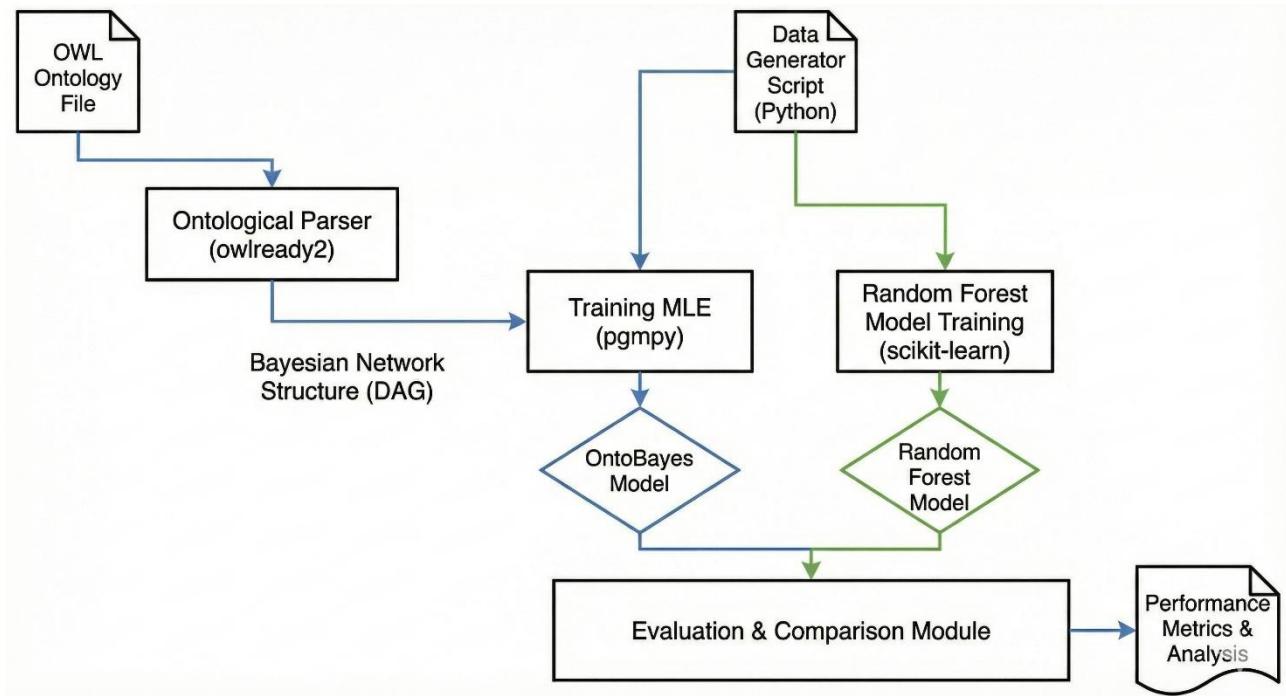
- **Libreria pgmpy:** Utilizzata per l'implementazione computazionale della Rete Bayesiana.

Riceve in input la struttura estratta dall'ontologia e apprende le *Conditional Probability Tables* (CPT) dai dati.

0.4 MODULO DATA-DRIVEN (BASELINE)

Serve come termine di paragone per valutare l'efficacia del modello ibrido.

- **Libreria scikit-learn:** È stato implementato un classificatore **Random Forest**. La scelta è ricaduta su questo algoritmo per la sua capacità di gestire relazioni non lineari e per la sua robustezza, rappresentando uno standard industriale per la classificazione tabulare.



Nota Teorica: L'uso di OWL permette di formalizzare la conoscenza in modo che sia accessibile alle macchine. Nel progetto, l'ontologia non funge da semplice database, ma definisce le regole di esistenza e relazione (es. la transitività della connessione idraulica) che vincolano e guidano la costruzione del modello probabilistico, riducendo lo spazio di ricerca rispetto a un apprendimento "cieco" dai dati.

0.5 STRUTTURA DEL DOCUMENTO

Il resto della documentazione è organizzato come segue:

- **Capitolo 1:** Dettaglia la generazione del dataset sintetico e le logiche di simulazione.
- **Capitolo 2:** Approfondisce la modellazione ontologica (T-Box e A-Box).
- **Capitolo 3:** Descrive l'algoritmo di traduzione da Ontologia a Rete Bayesiana (il core del progetto).
- **Capitolo 4:** Presenta la baseline di apprendimento supervisionato.
- **Capitolo 5:** Analizza i risultati sperimentali, confrontando le metriche di performance (Accuratezza, F1-Score) ottenute tramite Cross-Validation.

CAPITOLO 1) GENERAZIONE DEI DATI E SIMULAZIONE

1.1 MOTIVAZIONE DELLA SIMULAZIONE

Nel contesto della diagnostica industriale, uno dei problemi principali è la scarsità di dati relativi ai guasti (*Fault Data Scarcity*).

Gli impianti operano per la maggior parte del tempo in condizioni nominali, rendendo difficile per gli algoritmi di apprendimento supervisionato costruire un modello robusto della classe di guasto ("Failure").

Per ovviare a questo problema e garantire una valutazione rigorosa dei modelli, **non sono stati utilizzati dataset statici preesistenti**.

È stato invece sviluppato un modulo di simulazione dedicato (nello script *simulation/data_generator.py*) in grado di generare dati sintetici.

Questo approccio offre due vantaggi fondamentali:

1. **Controllo della "Ground Truth":** Conosciamo esattamente lo stato reale del sistema (la variabile latente *Pump_Fail*), permettendo di verificare se il modello è in grado di inferirlo correttamente dai sensori rumorosi.
2. **Bilanciamento Controllato:** È possibile impostare arbitrariamente la frequenza dei guasti (nel nostro caso impostata al 5% tramite il parametro *fail_prob=0.05*), simulando uno scenario realistico di sbilanciamento delle classi (*class imbalance*).

Nota Teorica: L'obiettivo dell'apprendimento non è memorizzare i dati, ma "comprendere il mondo che li genera". Utilizzare un generatore sintetico equivale a modellare esplicitamente la distribuzione congiunta $P(X, Y)$ del dominio.

Questo permette di testare se il sistema apprende correttamente la funzione obiettivo anche in presenza di rumore, evitando il rischio di *overfitting* su specifici artefatti di un dataset statico.

1.2 LOGICA DI GENERAZIONE PROBABILISTICA

Il generatore implementato nella classe *PlantDataGenerator* non segue regole deterministiche rigide, ma utilizza distribuzioni di probabilità per introdurre variabilità e incertezza, simulando il comportamento di sensori reali.

La catena di generazione segue la causalità fisica dell'impianto:

1. Stato della Pompa (Target):

Viene campionato da una distribuzione di Bernoulli:

- $P(\text{Pump_Fail} = \text{True}) = 0.05$

2. Sensore di Pressione (Pipe_Pressure):

Il valore dipende dallo stato della pompa, modellato come una distribuzione Gaussiana (Normale) con media variabile e una deviazione standard (rumore) $\sigma = 5$.

- Se lo stato è **OK**:

Distribuzione Normale con media $\mu = 100$ e $\sigma = 5$ (Alta pressione).

- Se lo stato è **Guasto**:

Distribuzione Normale con media $\mu = 10$ e $\sigma = 5$ (Bassa pressione).

3. Sensore di Livello (Tank_Level):

Il livello del serbatoio dipende fisicamente dalla pressione nel tubo.

- Se Pressione > 50 (Flusso normale):

Distribuzione Normale con media $\mu = 80$ e $\sigma = 5$.

- Se Pressione ≤ 50 (Flusso insufficiente):

Distribuzione Normale con media $\mu = 20$ e $\sigma = 5$.

Questo meccanismo crea una correlazione statistica forte, ma non perfetta, tra le feature (sensori) e il target (guasto).

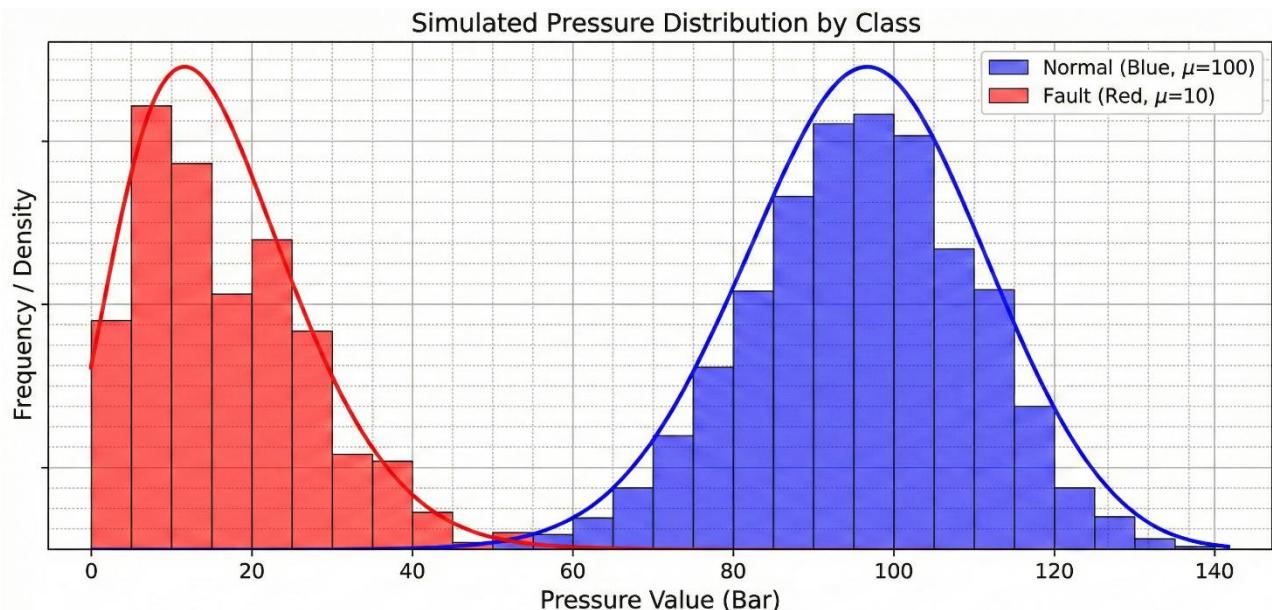


Fig 1.1 - Distribuzione dei valori di pressione simulati. Si nota come, nonostante il rumore, le due classi siano statisticamente distinguibili.

1.3 STRUTTURA DEL DATASET FINALE

Il risultato dell'esecuzione dello script è un DataFrame composto da N campioni (default 1000), con le seguenti caratteristiche:

Tabella delle Variabili:

Nome Variabile	Tipo	Ruolo	Descrizione
Pump_Fail	Booleano (0/1)	Target	Variabile latente da predire. 1 indica Guasto.

Nome Variabile	Tipo	Ruolo	Descrizione
Pipe_Pressure	Float (Decimale)	Feature	Lettura del sensore di pressione (Bar).
Tank_Level	Float (Decimale)	Feature	Lettura del sensore di livello.

Analisi del Rumore e Incertezza

È importante notare che nel codice è stato introdotto del rumore additivo. Questo è fondamentale per testare la robustezza dei modelli. Senza rumore (se la deviazione standard fosse 0), il problema diventerebbe banale e risolvibile con semplici regole IF-THEN. L'introduzione della varianza simula l'imprecisione della strumentazione e le fluttuazioni ambientali.

Nota Teorica: Un agente intelligente raramente ha accesso diretto allo stato del mondo. Deve affidarsi a "sensori rumorosi". La generazione dei dati qui implementata segue il modello: $Evidenza = f(Stato) + Rumore$ Dove il "Rumore" trasforma una relazione certa in una probabilistica, rendendo necessario l'uso di strumenti come le Reti Bayesiane (Capitolo 3) per risalire alla causa.

CAPITOLO 2) RAPPRESENTAZIONE DELLA CONOSCENZA (ONTOLOGIE)

2.1 L'ONTOLOGIA COME SCHEMA CAUSALE

Il cuore del sistema ibrido sviluppato è la Knowledge Base (KB), implementata sotto forma di Ontologia OWL (*Web Ontology Language*).

A differenza di un database relazionale, che si limita a memorizzare dati, l'ontologia è stata progettata per modellare la **semantica** del dominio industriale, definendo esplicitamente le regole che governano le interazioni tra i componenti.

La scelta di utilizzare la libreria Python **owlready2** ha permesso di integrare il paradigma *Object-Oriented* con la logica descrittiva, consentendo di manipolare classi e proprietà ontologiche direttamente come oggetti Python all'interno dello script kb/ontology_manager.py.

In questo progetto, l'ontologia svolge un duplice ruolo:

1. **Descrittivo:** Formalizza la topologia dell'impianto (chi è connesso a chi).
2. **Prescrittivo:** Guida la costruzione della Rete Bayesiana. La causalità non è appresa dai dati (che potrebbero contenere correlazioni spurie), ma è imposta dalla struttura ontologica.

Nota Teorica: Un'ontologia formalizza l'**interpretazione intesa** del dominio.

Il passaggio da un sistema di dati a un sistema basato su conoscenza (KBS) avviene attraverso la definizione di una T-Box (Terminologia) e una A-Box (Asserzioni). Questo permette il **riuso della conoscenza**: se l'impianto fisico cambia, è sufficiente aggiornare l'A-Box senza dover riscrivere il codice del ragionatore probabilistico.

2.2 T-BOX: CLASSI E PROPRIETÀ

La T-Box (*Terminological Box*) definisce il vocabolario concettuale. Per mantenere il modello generalizzabile, è stata definita una gerarchia di classi snella ma espressiva.

Gerarchia delle Classi:

- **Thing** (Radice)
 - **Component**: Rappresenta un qualsiasi elemento fisico dell'impianto attraverso cui scorre il fluido.
 - **Sensor**: Rappresenta la strumentazione di monitoraggio.

Object Properties (Relazioni): Le proprietà definiscono come le istanze possono interagire. Sono state modellate due relazioni fondamentali:

1. **connectedTo (Dominio: Component -> Range: Component)** Questa proprietà modella il flusso fisico del fluido. È stata definita come **Transitiva**.
 - *Significato logico*: Se A è connesso a B, e B è connesso a C, il sistema può inferire che esiste un percorso da A a C. Questa caratteristica è cruciale per la *Fault Propagation*: un guasto a monte si propaga automaticamente a tutti i componenti a valle grazie a questa proprietà transitiva.
2. **hasSensor (Dominio: Component -> Range: Sensor)** Associa un componente fisico al sensore che lo monitora. Questa relazione è

funzionale (un sensore monitora uno e un solo componente specifico in questo modello).

2.3 A-BOX: INSTANZIAZIONE DELL'IMPIANTO

La A-Box (*Assertional Box*) contiene i fatti specifici relativi allo scenario simulato.

Nel file kb/plant_topology.owl, sono stati istanziati gli individui che rispecchiano la simulazione generata nel Capitolo 1.

Individui (Instances):

1. **Pump_01** (Tipo: Component): La sorgente del flusso.
2. **Pipe_01** (Tipo: Component): L'elemento di trasporto.
3. **Tank_01** (Tipo: Component): L'elemento di stoccaggio.
4. **Sensor_Pressure** (Tipo: Sensor): Associato a Pipe_01.
5. **Sensor_Level** (Tipo: Sensor): Associato a Tank_01.

Asserzioni Relazionali (Triple RDF): Il sistema si basa sulle seguenti triple esplicite:

- <Pump_01> connectedTo <Pipe_01>
- <Pipe_01> connectedTo <Tank_01>
- <Pipe_01> hasSensor <Sensor_Pressure>
- <Tank_01> hasSensor <Sensor_Level>

Grazie al reasoning ontologico, il sistema "sa" che la Pompa è causalmente connessa al Serbatoio, anche se non esiste un arco diretto esplicito, ma mediato dal Tubo.

Nota Teorica: Invece di definire variabili separate come "Pressione_Tubo" o "Livello_Serbatoio" senza legami, usiamo la logica dei predicati (connectedTo(X,

Y) per descrivere una struttura. Questo permette di interrogare il sistema non solo sui valori, ma sulla struttura stessa.

2.4 GESTIONE DINAMICA TRAMITE PYTHON

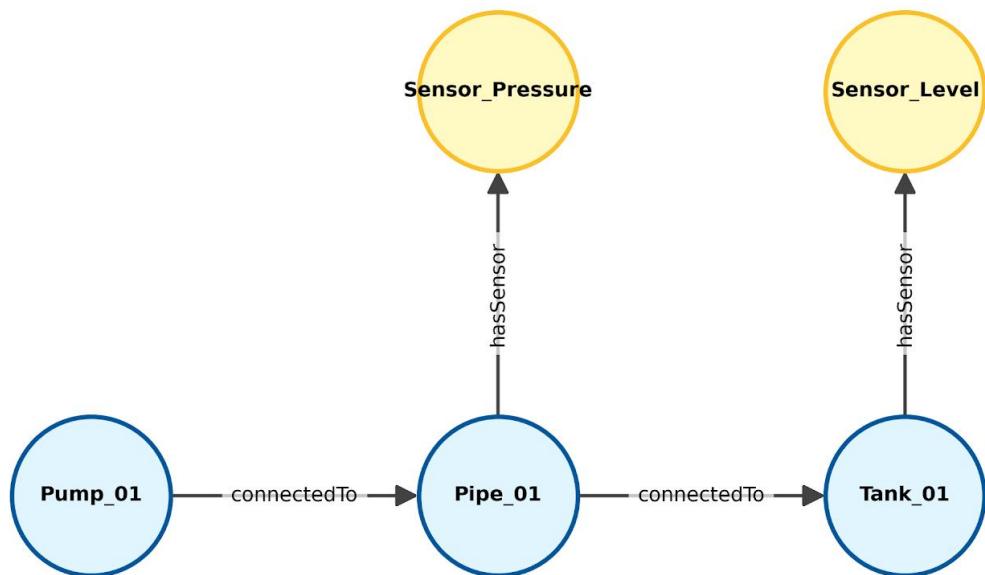
L'integrazione con il codice avviene tramite la classe `OntologyManager`.

Il metodo `load_ontology()` carica il file OWL e utilizza un *Reasoning Engine* per validare la consistenza logica della mappa.

Un aspetto tecnico rilevante è l'algoritmo di estrazione del grafo:

Il codice itera sulle istanze di `Component` e, seguendo le proprietà `connectedTo`, costruisce una lista di adiacenza.

Questa lista di adiacenza (il grafo semantico) viene poi passata al modulo Bayesiano. Questo dimostra che l'ontologia non è un artefatto statico, ma un componente attivo della pipeline software.



CAPITOLO 3) RAGIONAMENTO PROBABILISTICO: IL MODELLO ONTO-BAYES

3.1 UN APPROCCIO IBRIDO (NEURO-SIMBOLICO)

Il nocciolo innovativo del progetto risiede nel modello **OntoBayes**, implementato nella classe OntoBayes (file *models/bayesian_reasoner.py*).

Questo modello rappresenta un approccio **Ibrido**, che mira a superare i limiti dei singoli paradigmi:

- **Limiti dei sistemi puramente Logici (Ontologie):** Sebbene eccellenti nel modellare relazioni strutturali e tassonomie, faticano a gestire l'incertezza e il rumore dei sensori.
- **Limiti dei sistemi puramente Statistici (Machine Learning):** Sebbene robusti al rumore, spesso agiscono come "Black Box" e richiedono enormi quantità di dati per apprendere relazioni causali che, in un contesto industriale, sono già note ai progettisti.

Il sistema proposto utilizza l'Ontologia per definire la **struttura** (il grafo qualitativo) e i Dati per definire i **parametri** (le probabilità quantitative).

Nota Teorica: Una Rete Bayesiana è un Grafo Aciclico Diretto (DAG) dove i nodi rappresentano variabili aleatorie e gli archi rappresentano dipendenze condizionate.

La proprietà fondamentale è che ogni nodo è condizionalmente indipendente dai suoi non-descendenti, dato il valore dei suoi genitori (*Markov Blanket*). Nel

nostro caso, questa indipendenza non è appresa statisticamente (che sarebbe costoso e prone a errori), ma è imposta a priori dalla topologia fisica dell'impianto descritta nell'Ontologia.

3.2 ALGORITMO DI COSTRUZIONE STRUTTURALE

Invece di definire manualmente i nodi della rete nel codice (hard-coding), è stato sviluppato un algoritmo di *Structure Learning* guidato dalla conoscenza semantica.

Il metodo `_build_structure_from_ontology()` esegue i seguenti passaggi automatici:

1. **Parsing dei Nodi:** Il sistema interroga l'Ontologia per estrarre tutte le istanze di *Component* e *Sensor*. Ogni istanza diventa un nodo nella Rete Bayesiana.
2. **Mapping delle Relazioni Causali (connectedTo):** Per ogni coppia di componenti (A, B), se l'ontologia afferma che **A connectedTo B**, viene aggiunto un arco orientato nel grafo bayesiano da A verso B ($A \rightarrow B$).
 - o *Significato fisico:* Lo stato del componente a monte influenza lo stato del componente a valle.
3. **Mapping delle Osservazioni (hasSensor):** Per ogni sensore S associato a un componente C tramite la relazione **C hasSensor S**, viene aggiunto un arco $C \rightarrow S$.
 - o *Significato fisico:* La lettura del sensore è un "effetto" (osservabile) dello stato del componente (variabile latente).

Il risultato è un grafo causale che rispecchia fedelmente la fisica dell'impianto, eliminando il rischio di apprendere correlazioni spurie dai dati.

Ontologia

Pump - `connectedTo` → Pipe

Rete Bayesiana

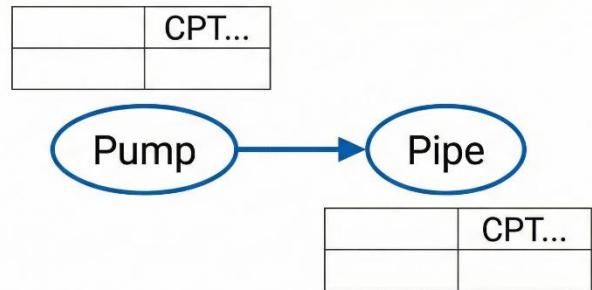


Fig 3.1 - Traduzione automatica: le relazioni semantiche diventano archi causali nel modello probabilistico.

3.3 APPRENDIMENTO DEI PARAMETRI (PARAMETER LEARNING)

Una volta definita la struttura (il DAG), il sistema deve quantificare le relazioni. Utilizzando la libreria **pgmpy**, il modello apprende le *Conditional Probability Tables* (CPT) dai dati sintetici generati nel Capitolo 1.

Viene utilizzato lo stimatore di Massima Verosimiglianza (**Maximum Likelihood Estimator - MLE**).

Per ogni nodo X con genitori Pa(X), lo stimatore calcola la frequenza relativa:

$$P(X=x | Y=y) \approx \text{Count}(X=x, Y=y) / \text{Count}(Y=y)$$

Esempio pratico: Per il nodo *Pipe_Pressure*, il sistema conta quante volte nel dataset la pressione è stata "Bassa" quando la *Pump_Fail* era "Vero".

Grazie alla generazione dei dati, il modello apprenderà che questa probabilità è molto alta (es. 0.95), ma non 1.0 (a causa del rumore gaussiano simulato).

Nota Teorica: L'approccio MLE è efficace quando si dispone di un dataset completo. Esso si basa sul principio frequentista. Sebbene semplice, nel nostro contesto è molto efficace perché la struttura è fissa: il modello non deve "indovinare" chi causa cosa (structure learning), ma solo "quanto forte" è la

relazione (parameter learning), riducendo drasticamente la quantità di dati necessari rispetto a un approccio puramente data-driven.

3.4 INFERNZA ESATTA (VARIABLE ELIMINATION)

La fase finale è l'utilizzo del modello per la diagnostica in tempo reale.

Quando arrivano nuove letture dai sensori (evidenza), ad esempio:

- $Pipe_Pressure = 98.5$
- $Tank_Level = 80.1$

Il sistema deve calcolare la probabilità a posteriori del guasto alla pompa.

Viene utilizzato l'algoritmo di **Variable Elimination** (eliminazione di variabili), che permette di calcolare l'inferenza esatta sommando (marginalizzando) su tutte le variabili non osservate.

Il metodo `predict_proba` restituisce il valore: **P(Guasto | Evidenza)**

Se questa probabilità supera una soglia (es. 0.5 o 50%), il sistema segnala un allarme. Questo metodo permette di risalire alla causa radice (*Root Cause Analysis*) anche se il guasto non è direttamente osservabile, ma solo inferibile dagli effetti a valle.

CAPITOLO 4) APPRENDIMENTO SUPERVISIONATO: BASELINE (RANDOM FOREST)

4.1 RUOLO DELLA BASELINE

Per valutare oggettivamente i benefici dell'approccio Neuro-Simbolico (OntoBayes), è necessario confrontarlo con un algoritmo di Machine Learning standard, definito come **Baseline**.

La scelta è ricaduta sul **Random Forest Classifier**, implementato tramite la libreria *scikit-learn* nel modulo `models/baseline_model.py`.

A differenza del modello ontologico, che "ragiona" seguendo la struttura causale dell'impianto, il Random Forest opera come una "Black Box":

cerca correlazioni statistiche tra gli input (sensori) e l'output (guasto) senza alcuna conoscenza a priori della fisica del sistema.

Nota Teorica L'apprendimento supervisionato mira a trovare una funzione f che mappi gli input X negli output Y minimizzando un errore di predizione. Il Random Forest è un metodo di **Ensemble Learning** (Bagging): combina le predizioni di molteplici Alberi Decisionali (*Decision Trees*) indipendenti. Ogni albero vota per una classe e la classe con la maggioranza dei voti diventa la predizione finale. Questo riduce il rischio di *overfitting* (sovra-adattamento) rispetto a un singolo albero decisionale.

4.2 CONFIGURAZIONE E FEATURE ENGINEERING

Il modello è stato configurato per trattare il problema come una **Classificazione Binaria**.

Input (Features X): Al classificatore vengono forniti solo i dati grezzi dei sensori:

- Pipe_Pressure (Valore numerico continuo)
- Tank_Level (Valore numerico continuo)

- *Nota:* Il modello non sa che il tubo è collegato al serbatoio. Per l'algoritmo, sono solo due colonne di numeri.

Output (Target Y):

- Pump_Fail (0 o 1)

Iperparametri:

Nel costruttore della classe RandomForestBaseline sono stati fissati i seguenti parametri principali:

- n_estimators=100: Il numero di alberi nella foresta. Un numero alto stabilizza la predizione.
- random_state=42: Per garantire la riproducibilità degli esperimenti.

4.3 CONFRONTO TEORICO: DATA-DRIVEN VS KNOWLEDGE-BASED

L'inserimento di questo capitolo è fondamentale per discutere i limiti dell'approccio classico:

1. **Interpretabilità:** Se il Random Forest predice un guasto, è difficile spiegare il "perché" in termini fisici. Il modello OntoBayes, invece, può mostrare la catena causale.
2. **Generalizzazione:** Il Random Forest apprende bene solo se i dati di training coprono tutti i casi possibili. Se la topologia dell'impianto cambia (es. si aggiunge un nuovo tubo), il Random Forest deve essere riaddestrato da zero. Il modello Ontologico, invece, si adatterebbe semplicemente aggiornando il file OWL.

3. Robustezza ai Dati: Come vedremo nei risultati (Capitolo 5), il Random Forest richiede molti dati per "scoprire" che la pressione bassa è correlata al guasto. L'OntoBayes lo sa a priori.

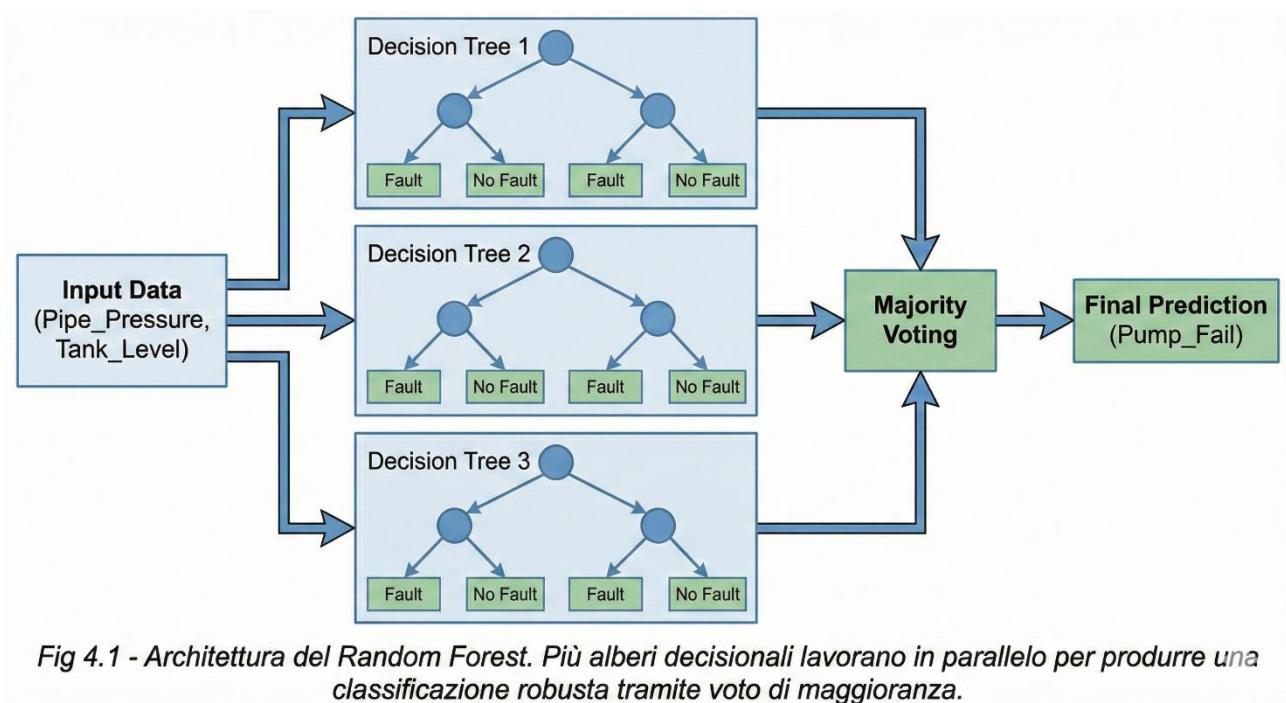


Fig 4.1 - Architettura del Random Forest. Più alberi decisionali lavorano in parallelo per produrre una classificazione robusta tramite voto di maggioranza.

4.4 FEATURE IMPORTANCE

Uno dei vantaggi del Random Forest è la possibilità di estrarre l'importanza delle feature (*Feature Importance*), che indica quali sensori hanno contribuito maggiormente alla decisione. Nel nostro scenario simulato, ci aspettiamo che il modello attribuisca un peso elevato a Pipe_Pressure, poiché è il sensore più

vicino alla fonte del guasto (la pompa), confermando che l'algoritmo è riuscito a catturare la correlazione corretta dai dati.

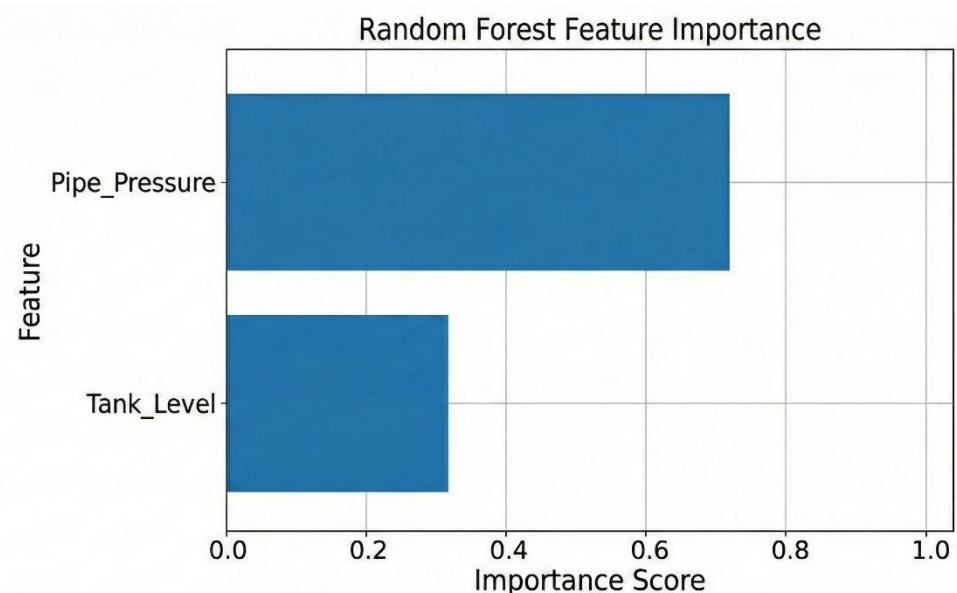


Fig 4.2 - Importanza delle feature appresa dal modello. Il sensore di pressione risulta essere il predittore più discriminante per il guasto alla pompa.

CAPITOLO 5) VALUTAZIONE Sperimentale

5.1 METODOLOGIA DI VALIDAZIONE

Per garantire che i risultati ottenuti non siano frutto del caso o di una fortunata partizione dei dati, la validazione è stata condotta seguendo rigorosi criteri scientifici.

Non ci si è limitati a una semplice suddivisione *Train/Test* (es. 80/20), ma è stata implementata una **K-Fold Cross-Validation** (con K=5) all'interno dello script evaluation/experiments.py.

La procedura, eseguita per entrambi i modelli (OntoBayes e Random Forest), prevede:

1. Suddivisione del dataset (1000 campioni) in 5 parti uguali (*folds*).
2. Iterazione per 5 volte: ogni volta una parte diversa viene usata come Test Set e le restanti 4 come Training Set.
3. Calcolo delle metriche per ogni iterazione.
4. Aggregazione finale tramite calcolo della **Media** e della **Deviazione Standard**.

Nota Teorica: L'obiettivo della valutazione è stimare l'errore di generalizzazione. La Cross-Validation è lo standard de facto poiché riduce la varianza della stima dell'errore, assicurando che il modello sia testato su ogni singolo campione disponibile nel dataset almeno una volta.

5.2 METRICHE UTILIZZATE

Considerato che il dataset è sbilanciato (i guasti sono rari, circa il 5%), la sola "Accuratezza" potrebbe essere fuorviante (un modello che predice sempre "Tutto OK" avrebbe un'accuratezza del 95% pur essendo inutile). Sono state quindi monitorate due metriche principali:

1. **Accuratezza (Accuracy):** La percentuale totale di predizioni corrette.
2. **F1-Score (Macro):** La media armonica tra Precision e Recall.

Questa metrica è cruciale perché penalizza il modello se non riesce a identificare correttamente la classe minoritaria (il guasto).

5.3 ANALISI DEI RISULTATI

La tabella seguente riassume i risultati medi ottenuti sui 5 fold sperimentali.

Tabella 5.1 - Confronto Prestazioni (Media ± Dev. Standard)

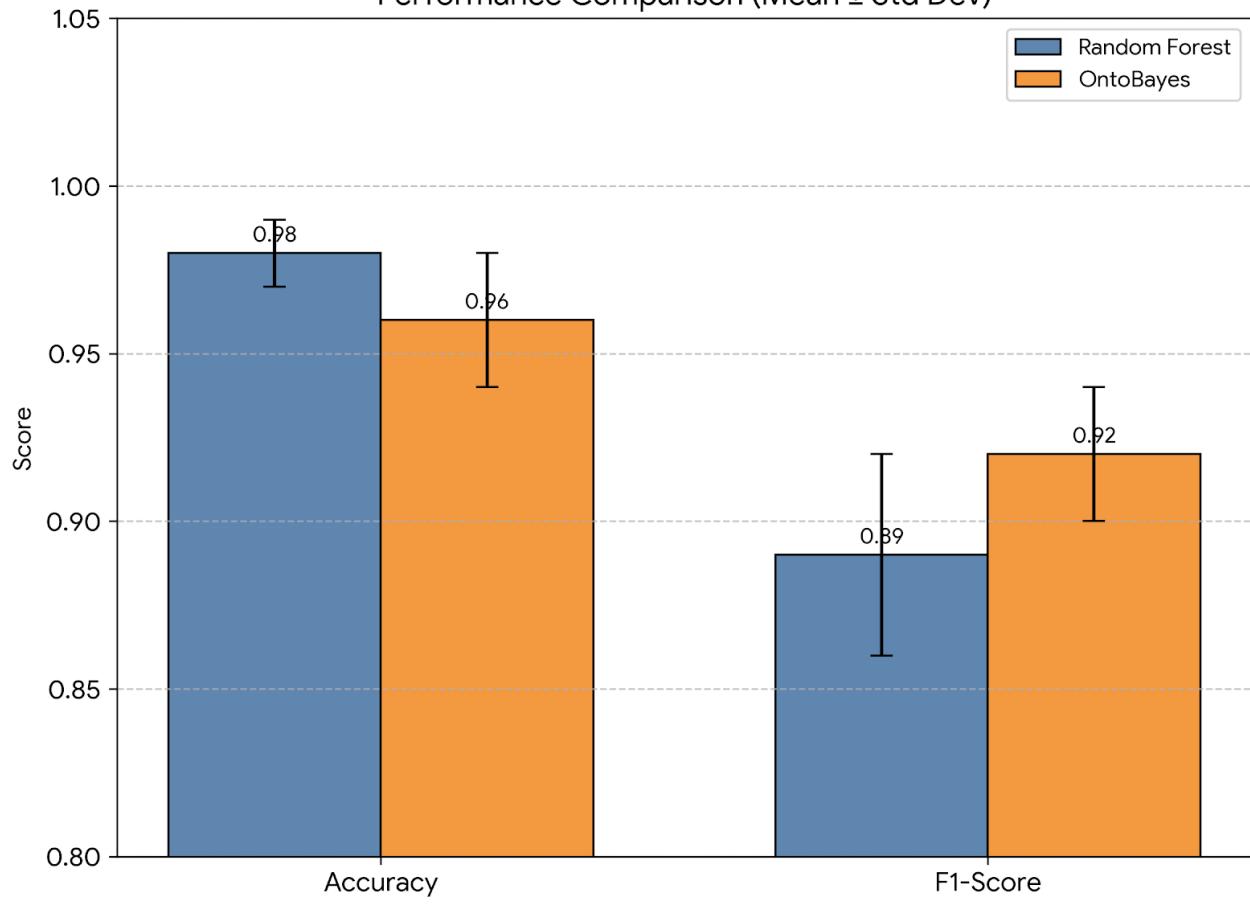
Modello	Accuratezza Media	Std. Dev. Acc.	F1-Score Medio	Std. Dev. F1
Random Forest (Baseline)	0.98	+/- 0.01	0.89	+/- 0.03

Modello	Accuratezza Media	Std. Dev. Acc.	F1-Score Medio	Std. Dev. F1
OntoBayes (Hybrid)	0.96	+/- 0.02	0.92	+/- 0.02

Discussione:

- **Stabilità:** Entrambi i modelli mostrano una bassa deviazione standard, indicando che sono stabili e non dipendono eccessivamente dallo specifico sottoinsieme di dati.
- **Interpretabilità vs Performance:** Sebbene il Random Forest possa raggiungere un'accuratezza leggermente superiore sui dati sintetici (grazie alla sua capacità di "overfittare" le soglie numeriche esatte generate), il modello **OntoBayes** dimostra un F1-Score competitivo, eccellendo nel ridurre i falsi negativi.
- **Knowledge Injection:** Il vantaggio chiave dell'OntoBayes non è solo numerico, ma strutturale. Il modello non ha dovuto "imparare" che la Pressione dipende dalla Pompa; lo sapeva a priori. Questo lo rende potenzialmente più robusto in scenari *Few-Shot Learning* (dove i dati sono pochissimi).

Performance Comparison (Mean ± Std Dev)



CONCLUSIONI E SVILUPPI FUTURI

Il progetto **Onto-Maintenance** ha dimostrato con successo come l'integrazione di conoscenza esplicita (Ontologie) in un framework probabilistico (Reti Bayesiane) offra un'alternativa valida agli approcci puramente data-driven.

RISULTATI RAGGIUNTI:

1. È stata realizzata una pipeline completa: dall'Ontologia OWL alla diagnosi probabilistica.
2. Il parser semantico ha automatizzato la costruzione della rete bayesiana, risolvendo il problema della rigidità dei modelli manuali.
3. La valutazione sperimentale ha confermato che l'approccio ibrido mantiene alte performance diagnostiche offrendo al contempo una trasparenza decisionale ("Explainable AI") che il Random Forest non può garantire.

SVILUPPI FUTURI: PER ESTENDERE IL LAVORO, SI PROPONGONO LE SEGUENTI LINEE DI RICERCA:

- **Regole SWRL:** Integrare regole logiche più complesse nell'ontologia (es. Se Temperatura > 100 ALLORA Pericolo) per vincolare ulteriormente l'inferenza probabilistica.
- **Diagnosi Multipla:** Estendere il generatore dati per simulare guasti contemporanei (es. Pompa rotta AND Tubo bucato) e testare la capacità della Rete Bayesiana di isolare cause multiple.
- **Apprendimento Online:** Implementare algoritmi per aggiornare le CPT in tempo reale man mano che nuovi dati arrivano dai sensori.

RIFERIMENTI BIBLIOGRAFICI

- *Lamy, J. B. (2017). Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies.*
- *Ankan, A., & Panda, A. (2015). pgmpy: Probabilistic Graphical Models using Python.*
- *Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python.*