

# EcoOptima

## Componenti del gruppo

- Gabriele Di Liso, [MAT. 801378], [g.diliso3@studenti.uniba.it](mailto:g.diliso3@studenti.uniba.it)

Link GitHub: [progetto](#)

A.A. 2025-2026

# Indice

---

## Sommario

Introduzione e Architettura del Sistema .....	3
0.1 Obiettivo del progetto e Architettura Ibrida .....	3
0.2 Requisiti Funzionali e Scenari Operativi .....	3
0.3 Stack Tecnologico .....	4
Creazione del Dataset .....	5
1.1 Generazione e Struttura del Dataset.....	5
1.2 Preprocessing e Pulizia dei Dati.....	6
1.3 Variabilità dei dati e prevenzione dell'overfitting .....	7
Apprendimento Supervisionato (previsione dei costi).....	7
2.1 Il Task Predittivo .....	7
2.2 Scelte Progettuali e Iper-parametri .....	9
2.3 Addestramento e Validazione .....	10
2.4 Valutazione delle Prestazioni .....	10
2.5 Analisi dei risultati .....	11
Ragionamento Probabilistico e Reti Bayesiane .....	11
3.1 Modellazione del Rischio tramite Rete Bayesiana.....	12
3.2 Struttura della Rete (Grafo Diretto Aciclico - DAG) .....	12
3.3 Definizione delle Conditional Probability Tables (CPT) .....	13
3.4 Inferenza (Variable Elimination) e Risultati sugli Scenari .....	14
Ragionamento Logico e Ottimizzazione (Prolog + CSP) .....	15
4.1 La Knowledge Base in Prolog: Rappresentazione e Deduzione.....	15
4.2 Interfaccia Architettonica Python-Prolog (pyswip) .....	17
4.3 Formulazione del Constraint Satisfaction Problem (CSP).....	17
4.4 Risoluzione dello Spazio di Ricerca e Profilo Ottimale.....	19
Conclusioni e Sviluppi Futuri .....	20
5.1 Valutazione Complessiva del Sistema .....	21
5.2 Sviluppi Futuri .....	21
Riferimenti Bibliografici.....	22

# Introduzione e Architettura del Sistema

---

L'obiettivo di questo progetto è la progettazione e lo sviluppo di **EcoOptima**, un agente intelligente per la gestione energetica domestica (Home Energy Management System - HEMS). Il sistema è concepito per monitorare l'ambiente, prevedere i costi energetici, valutare il rischio di sovraccarico della rete e intervenire attivamente per ripianificare l'accensione degli elettrodomestici, garantendo la sicurezza elettrica dell'abitazione.

Per operare efficacemente, EcoOptima costruisce una rappresentazione interna dell'ambiente domestico (sensori e vincoli degli elettrodomestici) e, dato un obiettivo di sicurezza, calcola autonomamente la configurazione ottimale per raggiungerlo.

## 0.1 Obiettivo del progetto e Architettura Ibrida

Rispetto a soluzioni software tradizionali, EcoOptima si distingue per l'adozione di un'**architettura ibrida** che integra tre diversi paradigmi dell'Ingegneria della Conoscenza, permettendo all'agente di superare i limiti di un singolo approccio:

1. **Apprendimento Supervisionato (Machine Learning)**: Utilizzato per analizzare i dati storici e astrarre un modello predittivo sulle fasce di costo dell'energia.
2. **Ragionamento Probabilistico**: Utilizzato per gestire l'incertezza fisiologica dei sensori (es. meteo e carico di rete) e calcolare dinamicamente la probabilità di un blackout.
3. **Ragionamento Logico e a Vincoli (Prolog + CSP)**: Utilizzato per rappresentare la conoscenza ferrea e immutabile del dominio elettrico (le incompatibilità di potenza) e risolvere i conflitti generando un piano di accensione sicuro e ottimizzato.

## 0.2 Requisiti Funzionali e Scenari Operativi

Il sistema deve essere in grado di acquisire input dall'utente (la richiesta di accensione di specifici dispositivi) e valutare lo stato dell'ambiente circostante. Per validare l'agente, il progetto analizza due scenari operativi principali:

- **SCENARIO A (Emergenza):** L'ambiente presenta condizioni critiche (Meteo Nuvoloso e Carico di rete Alto). L'agente deve rilevare un elevato rischio di blackout, bloccare l'accensione simultanea dei dispositivi richiesti e consultare la propria base di conoscenza (Knowledge Base) per generare uno scheduling temporale (Load Shifting) che mantenga i consumi sotto il limite di 3.5 kW.
- **SCENARIO B (Tranquillo):** L'ambiente presenta condizioni ottimali (Meteo Soggiato e Carico di rete Basso). L'agente, grazie alla valutazione probabilistica, deduce l'assenza di rischio e lascia all'utente totale libertà operativa sui dispositivi, mettendo il risolutore di vincoli in standby.

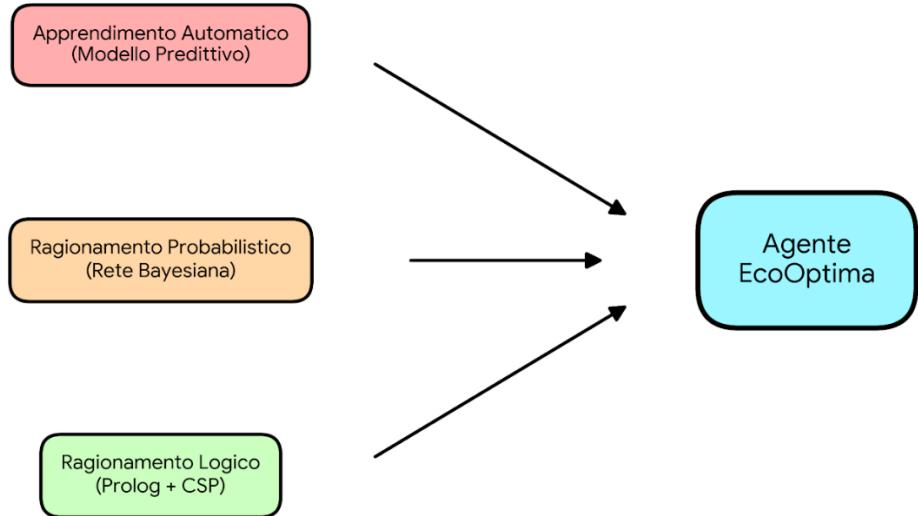
### 0.3 Stack Tecnologico

Il progetto è stato implementato interamente in **Python** (versione 3.x), sfruttando la sua ricca integrazione con librerie scientifiche e logiche, affiancato da **Prolog** per la modellazione della base di conoscenza.

L'ambiente di sviluppo (IDE) utilizzato è PyCharm, e le principali librerie impiegate per soddisfare i requisiti funzionali sono:

- scikit-learn: utilizzata per la creazione, l'addestramento e la validazione (K-Fold Cross Validation) del modello di Machine Learning (Random Forest).
- pgmpy: impiegata per la definizione della struttura a grafo diretto aciclico (DAG) e per l'inferenza esatta (Variable Elimination) della Rete Bayesiana.
- pyswip: libreria ponte fondamentale per interfacciare l'algoritmo di Constraint Satisfaction Problem (CSP) in Python con il motore inferenziale di Prolog.
- pandas e numpy: essenziali per l'importazione, la generazione e il preprocessing del dataset in formato .csv.
- matplotlib: utilizzata per la visualizzazione dei risultati, in particolare per la generazione dei profili di consumo finali (grafici a barre).

## Architettura Ibrida del Sistema EcoOptima



## Creazione del Dataset

---

L'addestramento di un modello di Machine Learning richiede un solido database di partenza. Per EcoOptima è stato sviluppato un modulo interno in Python (basato sulle librerie numpy e pandas) in grado di generare autonomamente un dataset storico. Questo approccio permette di simulare il comportamento di un vero database domotico che registra, ora per ora, le condizioni ambientali e i consumi della casa.

### 1.1 Generazione e Struttura del Dataset

Il dataset (`energy_data.csv`) è costituito da 1000 record, ognuno dei quali rappresenta un'ora di una giornata passata. Per permettere all'Agente Intelligente di apprendere il contesto ambientale, sono state definite le seguenti *features* (variabili di input):

- **temperature:** La temperatura esterna misurata in gradi Celsius. Questa variabile è fondamentale in quanto influisce direttamente sull'accensione di impianti di climatizzazione o riscaldamento.
- **hour:** L'ora del giorno, espressa in un formato da 0 a 23. Si tratta della metrica più importante per modellare lo stress della rete elettrica nazionale e i conseguenti picchi di prezzo serali.
- **day\_of\_week:** Il giorno della settimana (valori da 0 a 6). Modella i cambiamenti nelle abitudini umane (es. l'utenza domestica è più attiva nei weekend rispetto ai giorni lavorativi).
- **solar\_production:** L'energia (in kW) generata in quel momento dall'impianto fotovoltaico domestico. Se il valore è alto, la casa si auto-alimenta abbattendo i costi in bolletta.

Oltre alle *features*, è presente la variabile **Target** che il sistema supervisionato dovrà imparare a prevedere:

- **high\_cost:** Variabile booleana (0 o 1). Se il valore è 1, significa che in quell'ora l'energia si trovava in una "Fascia di Punta" (costo elevato). La regola base inserita nel generatore prevede che i costi si innalzino nella fascia serale (tra le 17:00 e le 22:00), quando il carico sulla rete nazionale è massimo.

## 1.2 Preprocessing e Pulizia dei Dati

Il preprocessing si è dimostrato una fase critica per garantire sia la leggibilità umana dei dati che la stabilità in fase di lettura da parte del sistema. Sono state applicate le seguenti ottimizzazioni:

1. **Risoluzione dei conflitti di parsing (Il separatore):** I file CSV (*Comma-Separated Values*) utilizzano nativamente la virgola per separare le colonne. Tuttavia, questo crea ben noti conflitti di incomprensione con le versioni europee dei software di fogli di calcolo (come Excel), che usano la virgola per i decimali, causando l'ammassamento di tutti i dati in un'unica colonna. Per risolvere definitivamente il problema a livello software, il modulo Python è stato forzato a salvare e leggere il dataset utilizzando il punto e virgola come separatore (`sep=';'`).
2. **Arrotondamento per realismo:** I valori continui generati originariamente da numpy (temperatura e produzione solare) presentavano fino a 15 cifre

decimali. Per simulare la precisione di veri sensori fisici, i valori sono stati arrotondati a due cifre decimali, migliorando drasticamente l'estetica e la leggibilità della tabella senza alcuna perdita di informazione utile.

### 1.3 Variabilità dei dati e prevenzione dell'overfitting

Se il dataset seguisse esclusivamente la regola matematica fissa (costo alto sempre e solo tra le 17:00 e le 22:00), l'algoritmo di Machine Learning non avrebbe bisogno di compiere generalizzazioni, limitandosi a memorizzare la regola e risultando perfetto ma irrealistico.

Per rendere il dataset fedele alla realtà e prevenire fenomeni di *overfitting* (sovradattamento), la generazione dei dati non segue una regola puramente deterministica. È stata introdotta una variabilità casuale (rumore stocastico) nel **5% dei record** per simulare situazioni anomale del mercato (es. picchi di prezzo improvvisi in orari diurni o cali tariffari serali).

Questo costringe il modello di Machine Learning a non limitarsi a memorizzare ciecamente i dati di addestramento, ma a ricercare pattern soggiacenti che permettano una reale generalizzazione su esempi mai visti prima, migliorando enormemente la robustezza del sistema in fase di test.

## Apprendimento Supervisionato (previsione dei costi)

---

L'apprendimento supervisionato, all'interno di EcoOptima, viene utilizzato per conferire all'agente intelligente la capacità di fare previsioni sui futuri costi dell'energia elettrica, permettendo così al sistema di pianificare l'accensione dei dispositivi in modo ottimizzato.

### 2.1 Il Task Predittivo

Il problema è stato formulato come un task di classificazione binaria. L'obiettivo dell'agente è prevedere se, in una determinata condizione ambientale e temporale, il costo dell'energia sarà alto o basso.

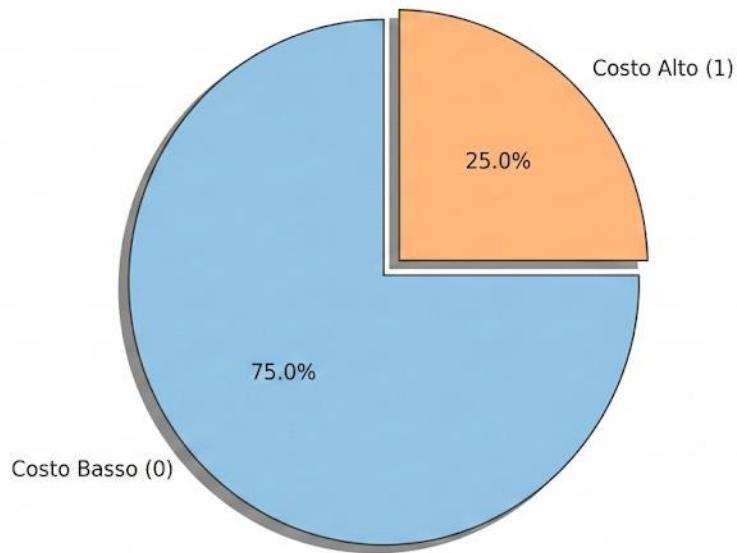
Per effettuare questa predizione, il modello osserva le *features* di input descritte nel capitolo precedente (*temperature*, *hour*, *day\_of\_week*, *solar\_production*).

La variabile di output (il *target*) è *high\_cost*, che assume valore 1 (costo elevato) o 0 (costo basso). Come documentato nel Capitolo 1, il target nel dataset è generato seguendo una regola basata sulle fasce orarie (costi alti tra le 17:00 e le 22:00) a cui è stato volutamente aggiunto un 5% di rumore stocastico (inversione casuale delle etichette) per simulare l'incertezza reale del mercato e testare la robustezza dell'algoritmo.

Riproduzione delle prime 5 righe della tabella excel:

Temperature	Hour	Day of week	Solar production	High cost
13.11	14	6	4.26	0
33.28	11	5	3.34	0
25.62	15	2	2.97	0
20.95	23	5	4.46	0
5.46	18	2	0.93	1

Distribuzione delle Classi di Costo (Dataset Energetico)



## 2.2 Scelte Progettuali e Iper-parametri

Per risolvere questo problema di classificazione, si è scelto di utilizzare il modello Random Forest Classifier. Il Random Forest è un classificatore che si ottiene creando tanti Decision Tree. Il valore in output si ottiene mediando sulle predizioni di ogni albero appartenente alla foresta tramite una tecnica nota come *bagging*.

La scelta del suo utilizzo è stata motivata da diverse ragioni pratiche attinenti al dominio:

- 1. Gestione di dati eterogenei:** Il Random Forest gestisce nativamente dataset che mescolano variabili continue (temperatura) e variabili categoriche ordinali (ora e giorno).
- 2. Robustezza al rumore:** Essendo un metodo *ensemble* risulta particolarmente resistente al rumore del 5% inserito nel dataset, riducendo drasticamente il rischio di overfitting.

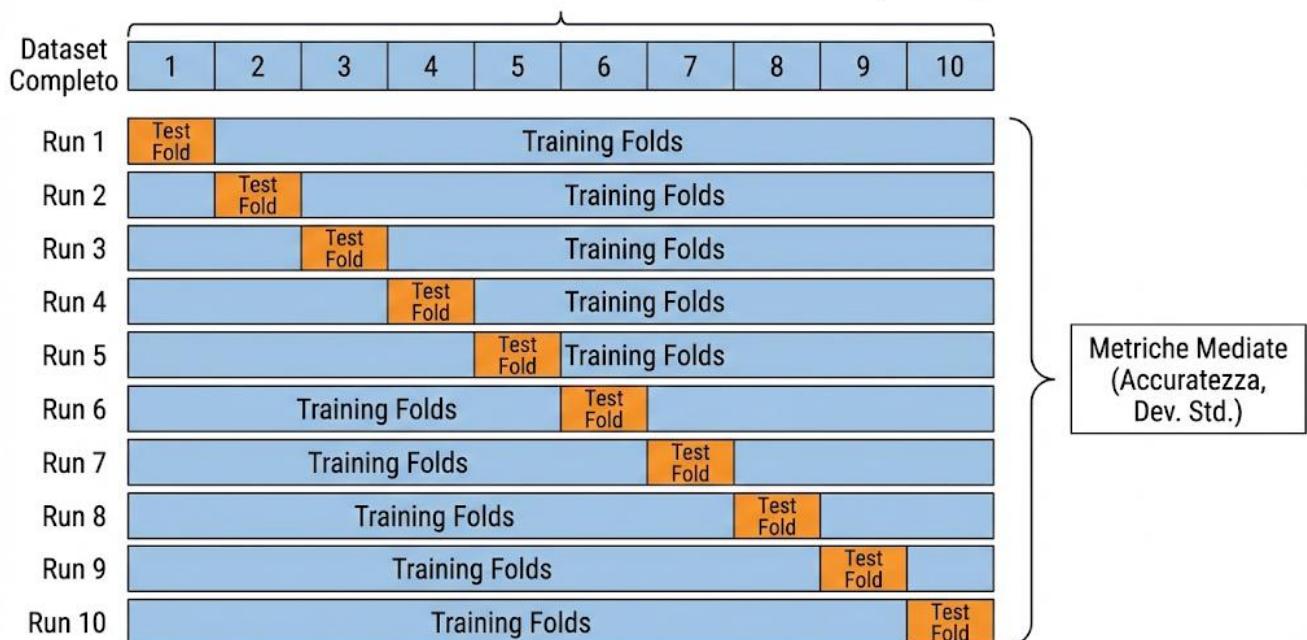
In fase di inizializzazione, gli iper-parametri sono stati settati a n\_estimators=100 (per garantire una foresta sufficientemente vasta da stabilizzare la varianza) e random\_state=42 (per garantire la riproducibilità degli esperimenti).

## 2.3 Addestramento e Validazione

Per evitare valutazioni fuorvianti basate su un singolo e fortunato partizionamento dei dati, la fase di validazione del modello è stata condotta utilizzando la tecnica della K-Fold Cross Validation.

Nello specifico, è stato utilizzato un approccio a 10-Fold (`n_splits=10`, `shuffle=True`), che suddivide il dataset di 1000 record in 10 partizioni disgiunte. Il parametro `shuffle=True` ha garantito che gli esempi venissero mescolati prima della divisione.

Schema di K-Fold Cross Validation (K=10)



## 2.4 Valutazione delle Prestazioni

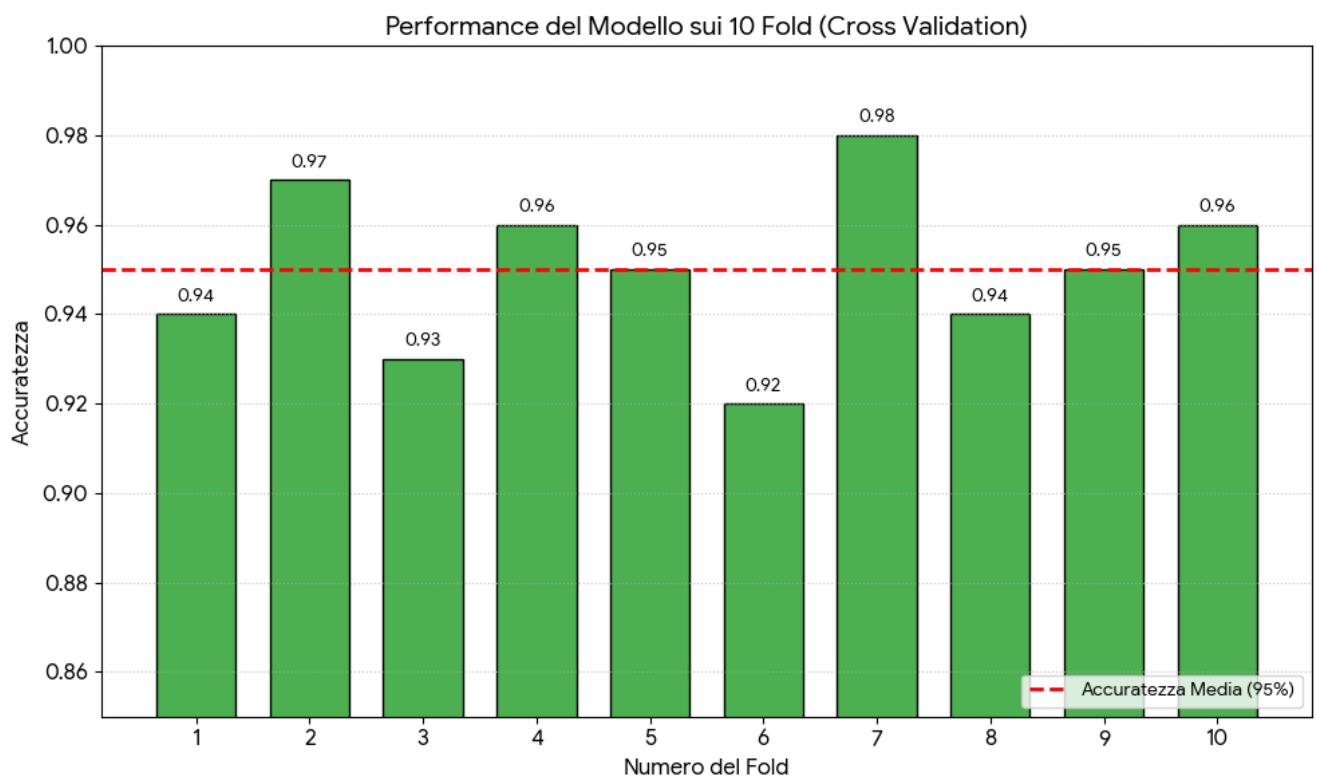
La valutazione finale viene presentata attraverso metriche statistiche mediate su tutti i run della K-Fold Cross Validation.

I risultati ottenuti sull'intero dataset sono i seguenti:

- Mean Accuracy (Accuratezza Media): ~95.00%
- Standard Deviation (Deviazione Standard): ~0.02 (2%)

## 2.5 Analisi dei risultati

L'accuratezza media attorno al 95% è un risultato eccellente e coerente con la logica del sistema. Dato che nel dataset è stato inserito artificialmente un 5% di rumore imprevedibile, il modello ha raggiunto il limite teorico massimo di apprendimento del pattern reale senza tentare di memorizzare il rumore. Inoltre, la deviazione standard molto bassa indica che le prestazioni del modello rimangono stabili indipendentemente dal *fold* utilizzato per il test. Questo ci permette di concludere con sicurezza che il modello non soffre di overfitting e generalizza in modo robusto.



## Ragionamento Probabilistico e Reti Bayesiane

---

Nel dominio della gestione energetica domestica, non tutti gli eventi sono deterministici o predicibili con assoluta certezza. Il sistema EcoOptima deve valutare il rischio di un sovraccarico (Blackout) basandosi su sensori ambientali e di rete, i cui dati sono intrinsecamente affetti da incertezza. Per dotare l'agente

della capacità di gestire questa incertezza, si è implementato un modulo di ragionamento probabilistico.

### 3.1 Modellazione del Rischio tramite Rete Bayesiana

Per valutare la probabilità di un blackout, si è scelto di utilizzare una **Rete Bayesiana Discreta** (implementata tramite la libreria pgmpy). Rispetto a un approccio logico "rigido" (se A allora B), la Rete Bayesiana permette di quantificare *quanto* sia probabile un sovraccarico date certe evidenze (es. cielo nuvoloso), consentendo all'agente di attivare lo *Scheduler Ottimizzato* solo quando il rischio supera una certa soglia di allerta (fissata nel sistema al 50%, ovvero  $> 0.5$ ).

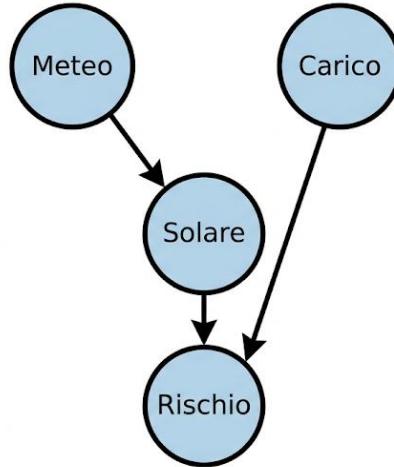
### 3.2 Struttura della Rete (Grafo Diretto Aciclico - DAG)

La struttura della rete è stata progettata definendo esplicitamente le relazioni di dipendenza causale tra quattro variabili casuali discrete:

- **Meteo** (0 = Sole, 1 = Nuvoloso): Variabile indipendente che funge da evidenza ambientale a priori.
- **Carico** (0 = Basso, 1 = Alto): Variabile indipendente che rappresenta lo stato di assorbimento della rete elettrica in quel momento.
- **Solare** (0 = Basso, 1 = Alto): Variabile dipendente dal Meteo, rappresenta la stima della produzione dei pannelli fotovoltaici.
- **Rischio** (0 = Basso, 1 = Alto/Blackout): Variabile target dipendente sia dal Carico della rete che dalla produzione Solare.

La struttura del DAG definita nel codice è dunque la seguente: Meteo  $\rightarrow$  Solare  $\rightarrow$  Rischio  $\leftarrow$  Carico

Struttura della Rete Bayesiana (DAG)



### 3.3 Definizione delle Conditional Probability Tables (CPT)

Trattandosi di un sistema esperto, le probabilità a priori e condizionate (CPT) sono state modellate per riflettere le dinamiche fisiche di un impianto fotovoltaico domestico.

#### Probabilità a Priori:

- $P(\text{Meteo} = \text{Sole}) = 0.7, P(\text{Meteo} = \text{Nuvoloso}) = 0.3$
- $P(\text{Carico} = \text{Basso}) = 0.6, P(\text{Carico} = \text{Alto}) = 0.4$

**CPT del nodo "Solare" (Dipende dal Meteo):** Si è modellato il fatto che una giornata di sole garantisce un'alta probabilità di avere molta energia solare, ma non la certezza assoluta (potrebbero esserci ombreggiamenti).

| Meteo |  $P(\text{Solare} = 0 | \text{Meteo})$  |  $P(\text{Solare} = 1 | \text{Meteo})$  |

| :--- | :--- | :--- |

| **Sole (0)** | 0.10 | 0.90 |

| **Nuvoloso (1)** | 0.80 | 0.20 |

**CPT del nodo "Rischio" (Dipende da Solare e Carico):** Questa tabella rappresenta il cuore logico probabilistico del modello. Il rischio di blackout schizza al 90% quando il carico è alto e la produzione solare è bassa (perché

l'energia prelevata dalla rete supera il limite del contatore). Al contrario, se c'è molta produzione solare (Solare=1), la probabilità di blackout scende drasticamente anche in presenza di un carico elevato (solo il 30% di rischio), poiché il fabbisogno energetico viene autoprodotto e compensato dai pannelli.

Produzione Solare	Carico Rete	P(Rischio = 0) [Basso]	P(Rischio = 1) [Blackout]
0 (Basso)	0 (Basso)	0.90	0.10
0 (Basso)	1 (Alto)	0.10	0.90
1 (Alto)	0 (Basso)	0.95	0.05
1 (Alto)	1 (Alto)	0.70	0.30

Una volta definita la struttura (DAG) e i parametri (CPT), la rete bayesiana viene utilizzata per il task di inferenza. Date delle evidenze (variabili di cui osserviamo lo stato reale), si utilizza il Teorema di Bayes e algoritmi di inferenza esatta (come la *Variable Elimination*) per calcolare la distribuzione di probabilità a posteriori delle variabili nascoste o target (il Rischio).

### 3.4 Inferenza (Variable Elimination) e Risultati sugli Scenari

Per calcolare la probabilità finale del blackout (nodo Rischio), l'agente interroga la rete fornendo le evidenze (Meteo e Carico) attraverso l'algoritmo di **Variable Elimination** (fornito dall'oggetto pgmpy.inference.VariableElimination).

Nel simulatore (main.py) sono stati testati due scenari per validare il ragionamento dell'Agente:

- SCENARIO A (Emergenza - Nuvoloso, Carico Alto):** Fornendo come evidenza {'Meteo': 1, 'Carico': 1}, la rete propaga l'incertezza. Il meteo nuvoloso sposta la probabilità di Solare verso il valore basso, che incrociandosi con il Carico alto porta l'inferenza a restituire una probabilità di Blackout calcolata del **78% (0.78)**. Essendo  $> 0.5$ , l'agente rileva correttamente l'allarme e blocca le azioni non sicure, invocando lo Scheduler logico (Prolog).
- SCENARIO B (Tranquillo - Sole, Carico Basso):** Fornendo come evidenza {'Meteo': 0, 'Carico': 0}, l'elevata probabilità di avere energia solare e il basso carico fanno crollare il calcolo del Rischio al **6.5% (0.065)**. In questo

caso, l'agente classifica lo stato come "Sicuro" e lascia all'utente la libertà di accendere i dispositivi.

```
[Bayes] Lettura sensori: Meteo=Nuvoloso, Carico=Alto  
Probabilità di Blackout calcolata: 0.78  
>>> ATTENZIONE: Rischio elevato rilevato. L'Agente attiva lo Scheduler Ottimizzato.
```

## Ragionamento Logico e Ottimizzazione (Prolog + CSP)

---

Mentre la Rete Bayesiana (descritta nel Capitolo 3) gestisce l'incertezza e stima il rischio di blackout in modo probabilistico, l'effettiva risoluzione del problema e la ripianificazione dei dispositivi richiedono un approccio deterministico basato su vincoli rigorosi. Quando il modulo di rischio rileva una probabilità di blackout superiore al 50%, l'Agente EcoOptima passa da uno stato di monitoraggio passivo a uno stato di intervento attivo, attivando lo *Scheduler Ottimizzato*.

Questo modulo sfrutta un approccio ibrido che unisce la logica relazionale (Prolog) e la programmazione a vincoli (Constraint Satisfaction Problem) per trovare un piano d'azione sicuro.

### 4.1 La Knowledge Base in Prolog: Rappresentazione e Deduzione

Per modellare i dispositivi domestici, i loro assorbimenti e le regole ferree del dominio elettrico, si è scelto di utilizzare il linguaggio **Prolog**. All'interno del progetto, la base di conoscenza (KB) memorizzata nel file dataset/devices\_rules.pl non agisce come un semplice database passivo, ma come un vero e proprio motore inferenziale in grado di dedurre nuova conoscenza esplicita a partire da fatti impliciti.

Nella KB di EcoOptima, la conoscenza è strutturata mediante fatti (verità incondizionate) e regole deduttive.

Oltre ai vincoli di potenza, la KB è stata arricchita per inferire vincoli di precedenza temporale tra i dispositivi. Invece di limitarsi a un semplice pattern matching, il sistema deduce una relazione asimmetrica di sequenzialità. Dichiarendo un fatto base di dipendenza funzionale (es. `depends_on(asciugatrice, lavatrice)`), il motore Prolog inferisce dinamicamente che un dispositivo deve terminare prima

dell'altro tramite la regola: *must\_run\_before(Y, X) :- depends\_on(X, Y)*. Tale conoscenza esplicita viene poi propagata al risolutore CSP.

**Fatti strutturali (Assiomi):** I dispositivi sono modellati come individui del dominio tramite il predicato *device/3*, che associa a un nome (atomo logico) le sue proprietà fisiche, tra cui la potenza nominale espressa in kW e una categoria di appartenenza.

```
% Sintassi: device(Nome, Potenza_kw, Categoria).  
device(lavatrice, 2.0, standard).  
device(forno, 2.0, riscaldamento).  
device(auto_elettrica, 3.0, ricarica).
```

**Regole deduttive e Vincoli Logici:** La vera potenza del Prolog all'interno del sistema risiede nella definizione di regole generali che deducono l'incompatibilità temporale tra due apparecchi. Due dispositivi non possono funzionare contemporaneamente se la somma delle loro potenze supera il limite fisico del contatore domestico (assunto convenzionalmente a 3.5 kW).

```
% Regola per dedurre l'incompatibilità tra due dispositivi  
incompatible(X, Y) :-  
    device(X, P1, _),  
    device(Y, P2, _),  
    X \= Y,  
    Somma is P1 + P2,  
    Somma > 3.5.
```

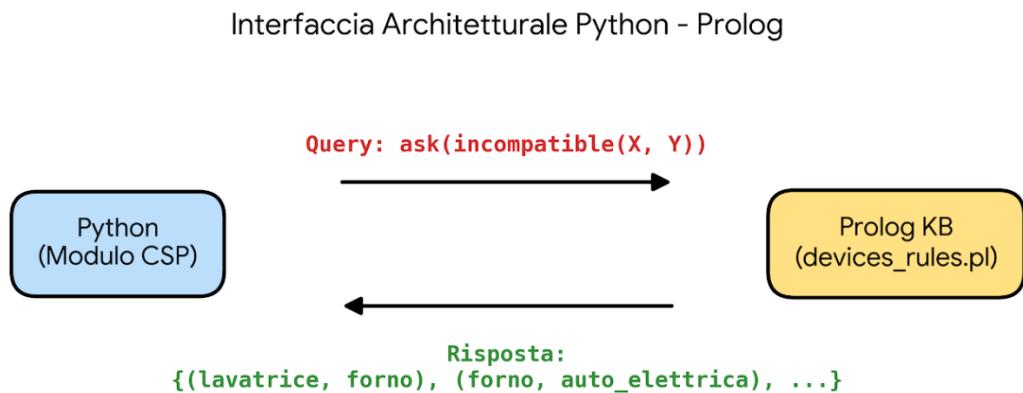
Attraverso questa regola, il sistema non ha bisogno di mappare a mano tutte le possibili combinazioni pericolose. Il motore di Prolog utilizza il meccanismo di **unificazione e risoluzione SLD (backward chaining)** per istanziare le variabili X

e Y, recuperare le rispettive potenze P1 e P2, sommarle e verificare matematicamente se il limite viene superato. Il controllo  $X \neq Y$  garantisce che il sistema non confronti un dispositivo con sé stesso.

## 4.2 Interfaccia Architetturale Python-Prolog (pyswip)

Per integrare l'inferenza logica appena descritta con l'algoritmo di ottimizzazione scritto in Python, è stata sviluppata la classe DeviceKB (all'interno di prolog\_interface.py), che sfrutta la libreria pyswip.

Questa scelta progettuale è fondamentale per mantenere separata la logica di dominio (Prolog) dalla logica di controllo (Python). All'avvio della pianificazione, Python interroga il Prolog inviando la query testuale "incompatible(X, Y)". Il motore Prolog esegue il *backtracking*, esplorando l'intero albero di derivazione e restituendo a Python l'insieme completo delle coppie di dispositivi che violano i vincoli di potenza. Python raccoglie questi risultati in un set (per evitare duplicati simmetrici come A-B e B-A) e li passa al risolutore. In questo modo, il codice Python è completamente agnostico rispetto alle regole elettriche: se in futuro il contatore dovesse passare da 3.5 kW a 4.5 kW o venissero aggiunti 50 nuovi dispositivi, il codice Python non subirà alcuna modifica; basterà aggiornare i fatti nella KB in Prolog.



## 4.3 Formulazione del Constraint Satisfaction Problem (CSP)

Una volta ottenuta la "mappa delle incompatibilità" dalla base di conoscenza, il problema di assegnare un orario di accensione a ciascun dispositivo richiesto

dall'utente è stato formulato rigorosamente come un problema di soddisfacimento di vincoli (CSP), implementato nella classe EnergyScheduler.

Nello specifico del dominio energetico di EcoOptima, il CSP è stato formalizzato come segue:

- **Insieme delle Variabili (V):**  $V = \{D_1, D_2, \dots, D_n\}$  dove ogni  $D_i$  rappresenta un dispositivo richiesto dall'utente per la giornata odierna (es. lavatrice, forno, auto\_elettrica).
- **Domini (D):** Il dominio di ciascuna variabile rappresenta gli slot temporali disponibili per l'accensione. Nel nostro scenario, il dominio è un insieme discreto e finito di ore:  $\text{dom}(D_i) = \{8, 9, 10, \dots, 23\}$  (dalle 08:00 alle 23:00).
- **Vincoli Rigidi (Hard Constraints):**
  1. **Vincolo Logico di Mutua Esclusione:** Se l'interfaccia Prolog ha dedotto che la coppia  $(D_i, D_j)$  appartiene all'insieme delle incompatibilità ***incompatible(X, Y)***, allora l'assegnazione temporale deve essere rigorosamente disgiunta:  $\text{Time}(D_i) \neq \text{Time}(D_j)$ .
  2. **Vincolo di Capacità Cumulativa del Contatore:** Per ogni ora  $h$  nel dominio temporale, la somma delle potenze dei dispositivi a cui è stato assegnato l'orario non deve superare il limite fisico:

$$\sum_{\{i \in V : \text{Time}(D_i) = h\}} \text{Power}(D_i) \leq 3.5$$

3. **Vincolo di Precedenza Sequenziale:** Se l'interfaccia Prolog ha dedotto una dipendenza temporale tramite la query *must\_run\_before(Di, Dj)*, il dominio temporale subisce una restrizione direzionale. Il CSP traduce questa inferenza forzando un hard constraint algebrico:  $\text{Time}(D_i) < \text{Time}(Dj)$ .

#### **4.4 Risoluzione dello Spazio di Ricerca e Profilo Ottimale**

Per dimostrare l'efficacia del KBS, il sistema viene testato nello **SCENARIO A** (Meteo Nuvoloso, Carico Alto). Il modulo di Machine Learning e la Rete Bayesiana hanno già rilevato una probabilità di blackout estremamente elevata (superiore al 70%). Nonostante ciò, l'utente richiede l'accensione simultanea di tre dispositivi ad alto consumo: lavatrice (2.0 kW), asciugatrice (2.0 kW) e forno (2.5 kW).

Se il sistema fosse privo di intelligenza e acconsentisse, il carico istantaneo balzerebbe a **7.0 kW**, ben oltre i 3.5 kW sopportabili, causando uno sgancio immediato dell'interruttore generale (blackout). L'Agente blocca la richiesta e attiva il ConstraintScheduler.

#### **Dinamica di esplorazione del risolutore:**

1. Lo scheduler tenta di inserire i carichi. Tenta di associare la lavatrice alle 21:00.
2. Il risolutore tenta di assegnare l'asciugatrice. Consulta la KB e rileva il vincolo `must_run_before(lavatrice, asciugatrice)`. L'asciugatrice viene quindi rigettata dalle 21:00 e spinta obbligatoriamente a uno slot successivo.
3. Il forno viene posizionato ottimizzando il carico residuo e cercando di non sovrapporsi."

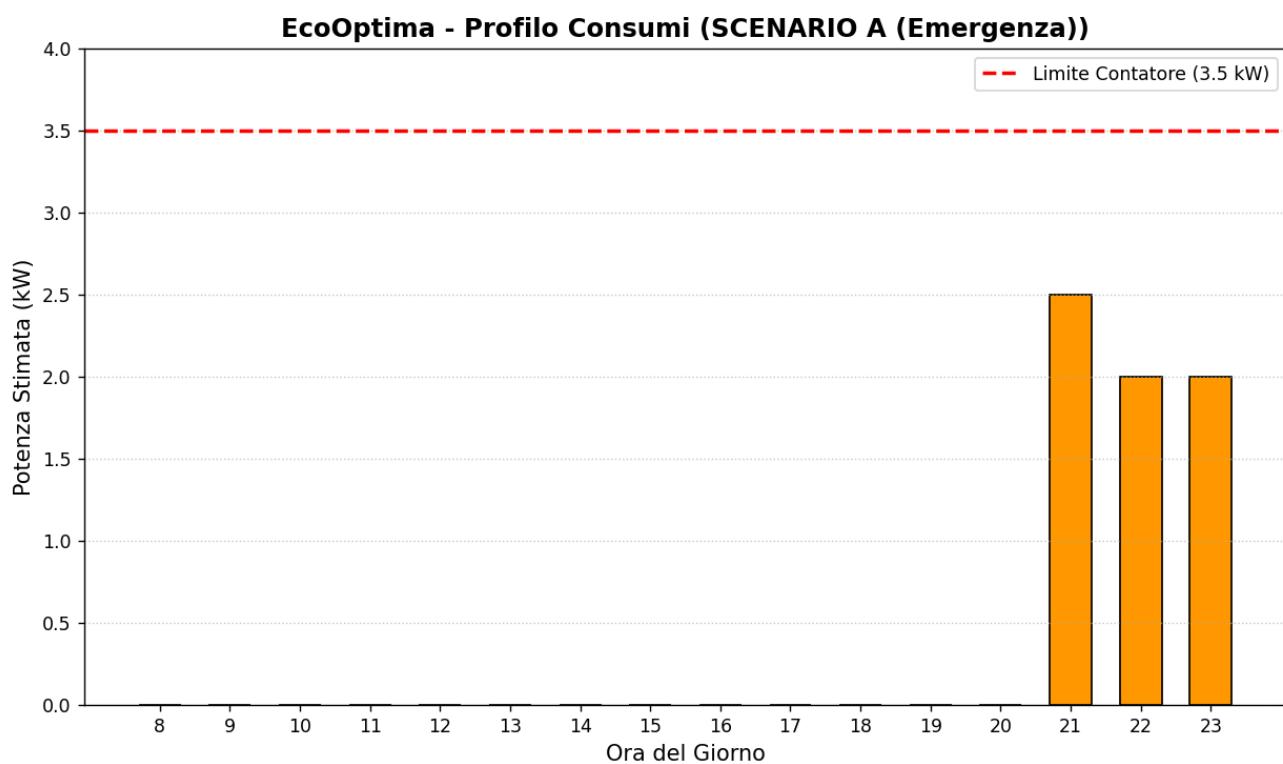
#### **Piano Soluzione (Output Testuale dell'Agente):**

Trovate pianificazioni valide. Ecco la migliore (ottimale):

- forno (2.5 kW): Accendere alle ore 21:00
- lavatrice (2.0 kW): Accendere alle ore 22:00
- asciugatrice (2.0 kW): Accendere alle ore 23:00

Invece di negare categoricamente il servizio all'utente, il sistema ha agito da mediatore intelligente: ha "spalmato" i carichi su slot temporali successivi (operazione nota nel dominio energetico come *Load Shifting*).

Il risultato finale dell'ottimizzazione viene fornito all'utente tramite una visualizzazione grafica dettagliata, generata con la libreria matplotlib, che rappresenta il profilo di consumo spalmato nell'arco della giornata, dimostrando che la curva dei consumi non interseca mai la linea di pericolo.



In conclusione, l'integrazione tra la rappresentazione logico-relazionale (Prolog) e la risoluzione algebrica dei vincoli (CSP) ha permesso di creare un KBS non solo intellettualmente robusto, ma in grado di operare in modo efficiente e flessibile per la risoluzione di problemi reali.

## Conclusioni e Sviluppi Futuri

---

L'obiettivo di questo progetto è stato lo sviluppo di un agente intelligente (EcoOptima) capace di gestire i consumi energetici domestici operando in un ambiente complesso, dinamico e parzialmente incerto. Per raggiungere questo scopo, si è scelto di allontanarsi da soluzioni basate su un singolo algoritmo, abbracciando invece un'architettura ibrida che integra i tre pilastri dell'Ingegneria della Conoscenza: l'Apprendimento Automatico, il Ragionamento Probabilistico e il Ragionamento Logico.

## 5.1 Valutazione Complessiva del Sistema

L'architettura sviluppata si è dimostrata robusta ed efficace nel soddisfare i requisiti funzionali proposti:

1. **Flessibilità ai Dati (Machine Learning):** L'uso del *Random Forest* ha permesso al sistema di astrarre regole sui costi dell'energia a partire da dati storici rumorosi, superando i limiti di un approccio puramente procedurale. L'eccellente accuratezza validata tramite K-Fold Cross Validation ha confermato la validità del modello.
2. **Gestione dell'Incertezza (Rete Bayesiana):** La formulazione del rischio tramite un DAG probabilistico ha evitato falsi allarmi. Invece di bloccare i dispositivi a ogni nuvola passeggera, l'agente calcola l'effettivo rischio di blackout ponderando il carico e la stima solare, intervenendo solo quando matematicamente necessario (soglia > 0.5).
3. **Affidabilità e Sicurezza (Prolog + CSP):** Quando il rischio si concretizza, il sistema non procede per tentativi probabilistici, ma si affida a vincoli logici rigorosi (Hard Constraints). Delegando la deduzione delle incompatibilità elettriche a un motore logico come Prolog, lo *Scheduler CSP* ha generato pianificazioni ottime in modo deterministico e intrinsecamente sicuro, rispettando il limite fisico del contatore.

Come evidenziato nelle buone pratiche di progettazione dei sistemi intelligenti, l'estrazione di pattern dai dati storici (Apprendimento) ha dimostrato il suo reale valore solo quando integrata con una solida rappresentazione interna formale (come i vincoli relazionali) permettendo all'agente di migliorare le decisioni e le azioni intraprese nel mondo fisico (Ottimizzazione e Ricerca).

## 5.2 Sviluppi Futuri

Sebbene l'architettura attuale sia funzionale e completa per gli scenari di test, il progetto si presta a diverse evoluzioni tecnologiche e concettuali, fortemente allineate con le frontiere della ricerca sui Knowledge-Based Systems (KBS):

- **Dai Fatti Prolog alle Ontologie (Knowledge Graphs):** Attualmente, la conoscenza sui dispositivi è memorizzata come fatti testuali in Prolog. Uno sviluppo futuro prevede la migrazione di questa base di conoscenza verso il *Web Semantico*, descrivendo i dispositivi, le loro classi (es. *ElettrodomesticoDaCucina*, *DispositivoCritico*) e i loro assorbimenti tramite un'Ontologia OWL. Questo permetterebbe di sfruttare un

*Knowledge Graph* per interrogazioni più ricche e per integrare l'agente con API di domotica standardizzate.

- **Modelli Probabilistici Relazionali (Apprendimento + Logica):** Un'altra naturale estensione consisterebbe nell'adottare Modelli Probabilistici Relazionali. Invece di mantenere separati il modulo Prolog (logico) e la Rete Bayesiana (probabilistica), si potrebbero definire regole logiche che includono pesi probabilistici, permettendo al sistema di ragionare sull'incertezza direttamente a livello relazionale (es. dedurre dinamicamente la probabilità di guasto di un dispositivo in base al suo utilizzo storico).
- 

## Riferimenti Bibliografici

- **Ragionamento logico e Proposizionale:** D. Poole, A. Mackworth: *Artificial Intelligence: Foundations of Computational Agents*. 3/e Cambridge University Press [Ch.5].
- **Prolog e Rappresentazione Relazionale:** D. Poole, A. Mackworth: *Artificial Intelligence: Foundations of Computational Agents*. 3e, Cambridge University Press [Ch.15].
- **Apprendimento Supervisionato:** D. Poole, A. Mackworth: *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press. 3rd Ed. [Ch.7].
- **Ragionamento Probabilistico e Incertezza (Reti Bayesiane):** D. Poole, A. Mackworth: *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press. 3/e. [Ch.9].
- **Ragionamento con Vincoli (CSP):** Dispense dell'insegnamento: *Knowledge Engineering and Intelligent Systems Programming* [Ch.3 - Ragionamento con Vincoli].