

**SISTEMA INTELLIGENTE PER L’ORIENTAMENTO AGLI STUDI PRESSO IL DIPARTIMENTO DI INFORMATICA**

**Docente:** Nicola Fanizzi

**Corso:** Ingegneria della Conoscenza (ICon)

**Realizzato da:**

* *PANTALEO G.* – **Matricola:** 708237 – **E-mail:** [g.pantaleo11@studenti.uniba.it](mailto:g.pantaleo11@studenti.uniba.it)
* *PRODON N.* – **Matricola: ???** – **E-mail:** [n.prodon@studenti.uniba.it](mailto:n.prodon@studenti.uniba.it)

# **Architettura del sistema**

Il sistema realizzato è in grado di determinare la compatibilità di uno studente (e non) con il dipartimento di Informatica grazie a una sequenza di domande, la cui risposta dovrà essere positiva (‘S’) o negativa (‘N’). In base alle risposte, verrà determinato se lo studente avrà affinità bassa, media o alta con il profilo da studente universitario informatico.

Il sistema di orientamento si basa su un *forward chaining:* inizialmente, lo studente è sottoposto a tutte le domande per determinare conseguentemente se quest’ultimo è portato per lo studio dell’informatica in ambito universitario.

L’idea alla base del sistema è una regola di derivazione, una forma generalizzata della regola di inferenza chiamata ***modus ponens***:

Se “ℎ ← 𝑎1 ∧ … ∧ 𝑎𝑚” è una clausola definita nella base di conoscenza e ogni  
𝑎𝑖 è stato derivato, allora ℎ può essere derivato.

Dove:

• ℎ è la “testa” dell’atomo;

• 𝑎1 ∧ … ∧ 𝑎𝑚 è il “corpo” della clausola, formato da 𝑎𝑖 atomi

Se 𝑚 > 0, la clausola è detta regola; se 𝑚 = 0, il corpo è vuoto e la clausola è detta clausola atomica (o fatto), e tutte le clausole atomiche nella base di conoscenza sono sempre derivate in maniera diretta.

# **Sistema esperto**

Un sistema esperto è un’applicazione dell’intelligenza artificiale atta alla risoluzione di determinati problemi, cercando di riprodurre i comportamenti di persone esperte in un determinato dominio. È principalmente composto da:

* una “knowledge base”, che rappresenta e memorizza fatti e regole riguardanti il mondo;
* un “inference engine”, che si occupa di mettere in pratica le nozioni apprese dalla base di conoscenza;
* una “user interface”, che permette una facile interazione tra il sistema e l’utente.

## **Implementazione del sistema esperto**

Il sistema si basa su un sistema esperto realizzato in **Python**,principalmente con la libreria **Experta**, che permette di associare fatti accaduti a regole relative agli stessi.

Le regole sono formate da due componenti:

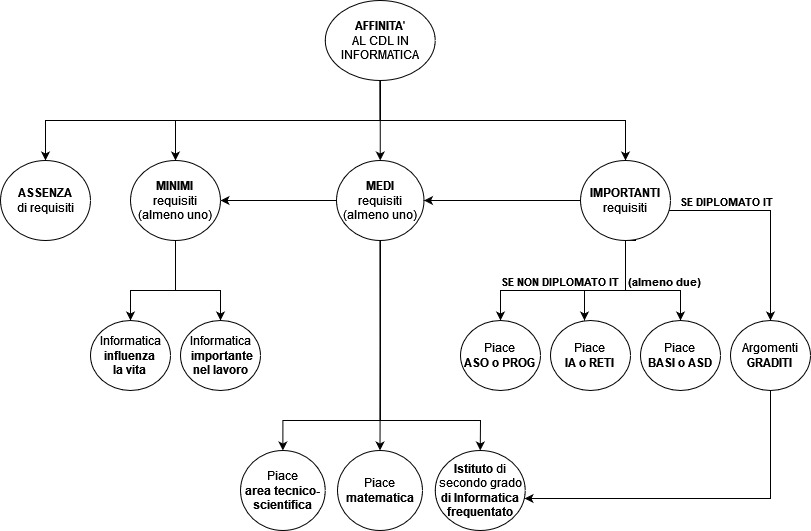
* **LHS** (Left-Hand-Side): è l’insieme delle condizioni che si devono verificare affinché la regola venga applicata;
* **RHS** (Right-Hand-Side): è l’insieme di azioni che vengono compiute quando viene applicata la regola.

L’interesse dell’intervistato dipende da una serie di fattori, quali:

* l’importanza della tecnologia nella vita quotidiana;
* l’importanza della tecnologia nel mondo del lavoro;
* un eventuale interesse negli studi dell’area STEM;
* un eventuale interesse per gli studi matematici;
* una eventuale istruzione presso un istituto tecnico e relativo interesse per gli studi già affrontati a scuola;
* nel caso di istruzione secondaria diversa dall’istituto tecnico, un eventuale interesse per argomenti come programmazione, basi di dati, reti e AI.

Ogni domanda posta all’intervistato, alla quale è associata una regola, ha il fine di rilevare in quest’ultimo l’interesse verso uno specifico argomento e, in base alla sua risposta (positiva o negativa), la presenza dell’interesse per tale argomento viene impostata a True o False; dopodiché, il sistema interpreterà la situazione dell’utente e di conseguenza applicherà altre regole relative ad altri argomenti d’interesse. In alcune occasioni, il sistema porrà all’utente domande diverse in base al tipo di risposte fornite da quest’ultimo (vedi 3a, 3b).

1. Il sistema comincia a porre all’utente due domande a carattere generale (importanza della tecnologia nella vita privata e nel lavoro):
   1. se l’utente risponde positivamente ad una delle due domande (o ad entrambe), vorrà dire che l’utente ha un minimo di interesse per l’informatica;
   2. nel caso in cui l’utente risponda negativamente ad entrambe, il programma terminerà.
2. Verificata la presenza di un interesse minimo nell’intervistato, il sistema pone domande leggermente più specifiche (interesse per area STEM e studio della matematica):
   1. se l’utente risponderà positivamente ad una sola delle due domande, l’interesse aumenterà moderatamente;
   2. se l’utente risponderà positivamente ad entrambe, l’interesse aumenterà ulteriormente;
   3. se l’utente risponderà negativamente ad entrambe le domande, l’interesse a fine analisi sarà molto basso.
3. Viene chiesto all’utente se ha frequentato un istituto tecnico:
   1. in caso di risposta positiva, all’utente verrà chiesto se ha gradito gli argomenti già precedentemente affrontati:
      1. se l’utente ha gradito gli argomenti trattati, l’interesse sarà alto;
      2. se l’utente non ha gradito gli argomenti trattati, l’interesse sarà basso
   2. in caso di risposta negativa, all’utente verrà chiesto se conosce gli argomenti cardine che andrà ad affrontare (programmazione e architettura degli elaboratori, reti e intelligenza artificiale, basi di dati):
      1. se l’utente ha trovato interessanti più di due argomenti, l’interesse sarà alto;
      2. se l’utente ha trovato interessante un solo argomento, l’interesse sarà medio;
      3. se l’utente non ha mostrato interesse in nessun argomento, l’interesse sarà basso.



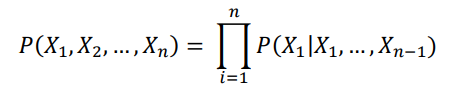
*Schema rappresentativo del sistema esperto.*

\***INSERISCI PEZZI DI CODICE**\*

# **Rete bayesiana**

Come secondo modello abbiamo usato una rete bayesiana per mezzo di un grafo aciclico orientato, dove ogni nodo è una feature del sistema e la dipendenza tra feature è espressa tramite un arco, anch’esso orientato. In questo modo, possiamo esprimere che l’attivazione di una feature dipende dall’attivazione di una o più feature, chiamate **genitore/i**. Per far ciò, dobbiamo creare una relazione di ordinamento tra le feature. La rete bayesiana usa le probabilità e il teorema di Bayes per poter calcolare la probabilità di un determinato evento, dipendendo (o meno) da altri eventi (probabilità condizionata). Nel caso della rete bayesiana, quindi, ogni nodo con genitori ha una probabilità condizionata di 𝑃(𝑛𝑜𝑑𝑒|𝑝𝑎𝑟𝑒𝑛𝑡𝑠(𝑛𝑜𝑑𝑒)), dove 𝑝𝑎𝑟𝑒𝑛𝑡𝑠(𝑛𝑜𝑑𝑒) è una funzione che restituisce i nodi genitore del nodo parametro della funzione.

In questo modo, ogni nodo avrà una tabella delle probabilità condizionate dai suoi genitori. Per calcolare le probabilità di ogni nodo, ci serviamo della formula:



dove *Xi sono le feature della rete bayesiana.*

Nel nostro modello, l’affinità al corso di laurea in Informatica dipende da requisiti, che dipendono a loro volta da specifiche domande (vedere fig. 2). Utilizzando la rete bayesiana possiamo esplicitare la dipendenza tra domande e requisiti andando a creare un nodo per ogni requisito e per ogni domanda con degli archi che partono dall’affinità e finiscono nei nodi relativi ai requisiti e poi ancora alle relative domande. In questa maniera, possiamo imporre una relazione d'ordine tra le feature ed assegnare ad ogni feature dei genitori (se esistono).

Una volta costruita una rete bayesiana con una sua struttura (la DAG) e le sue tabelle per la probabilità condizionata, possiamo inferire la probabilità che un determinato evento si verifichi avendo osservato il verificarsi di alcuni eventi.

Per far ciò, abbiamo due tipi di inferenza:

* inferenza esatta: si va ad enumerare i mondi coerenti con le osservazioni per poi sfruttare un algoritmo per calcolare la probabilità esatta dell’evento su cui si sta indagando. Ci sono vari algoritmi per il calcolo e noi useremo quello dell’eliminazione di variabili (che è simile all’omonimo algoritmo nei CSP);
* inferenza approssimata: si va a stimare la probabilità di un certo evento.

Nel caso in cui non si conoscano a priori le tabelle delle probabilità condizionate degli eventi, è possibile ricavarsele andando ad analizzare i dati e, usando opportuni algoritmi, stimare le probabilità condizionate dei nodi di una DAG.  
Per stimare le probabilità, è possibile procedere con lo stimatore di massima verosimiglianza o con altri metodi come regressioni, alberi di decisione, etc.

## **Implementazione rete bayesiana**

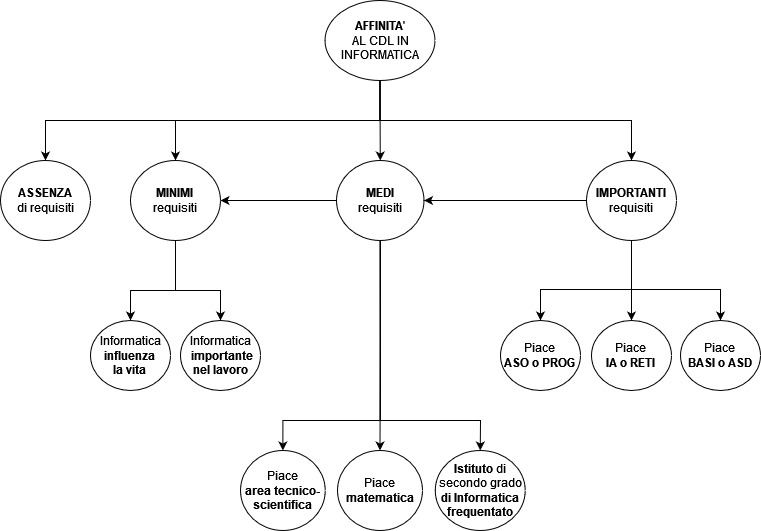
Per implementare la rete bayesiana in Python abbiamo fatto uso di **bnlearn**, una libreria *wrapper* contenente un’altra libreria, **pgmpy**.

La libreria *bnlearn* consente la creazione di una DAG, la creazione ed assegnazione delle tabelle di probabilità condizionata per ciascun nodo (con la classe TabularCPD) e permette poi di inferire le probabilità per un nodo della DAG, andando a segnalare le osservazioni sull’utente tramite il metodo della eliminazione delle variabili. Inoltre, *bnlearn* permette di imparare anche le tabelle delle probabilità, andando a stimare da un dataset fornito in input. In questo caso, viene permesso di scegliere come opzione lo stimatore di massima verosimiglianza. Per la creazione della DAG, è stato creato un vettore di coppie, ciascuna formata da due nodi del grafo che indicano un arco che li collega.

Creata una rete bayesiana funzionante, bisogna raccogliere le osservazioni dell’utente (i requisiti che pensa di rispettare) ed inferire la probabilità che l’utente sia compatibile con il CdL. Per far ciò, vengono poste delle domande all’utente; ogni requisito osservato avrà valore 0 se assente e 1 se invece è presente. Dopodiché, andiamo ad usare la funzione di inferenza di *bnlearn* che va ad usare l’eliminazione di variabili per ottenere la probabilità di affinità.

In alternativa, se non abbiamo le tabelle di probabilità per ogni nodo, possiamo apprendere le probabilità usando diversi stimatori (descritti precedentemente). Semplicemente diamo una DAG in input combinata con un dataset: ogni riga del dataset rappresenta un utente intervistato.

Una volta passati questi input, è possibile creare una rete bayesiana con stimatore di massima verosimiglianza, stavolta con probabilità apprese e non date.



*Schema rappresentativo della rete bayesiana.*