SISTEMA INTELLIGENTE PER LA DIAGNOSTICA DI UNA MALATTIA CAUSATA DA UN VIRUS

-

Gruppo composto da:

Enrico Ciciriello

Matr.716044 (e.ciciriello4@studenti.uniba.it)

Pasquale Caporusso

Matr.725080 (p.caporusso1@studenti.uniba.it)

Marco Lacalamita

Matr.726010 (m.lacalamita16tudenti.uniba.it)

Link del repository GitHub del progetto: https://github.com/Hacryn/ICON2122

SISTEMA DI DIAGNOSTICA

Il sistema del progetto cercherà di diagnosticare la malattia del "virus di fantasia", basandosi sui dati forniti dall'utente, il quale viene coinvolto nella risposta a una serie di domande mirate a individuare i sintomi di cui l'utente soffre e le possibili relazioni che questi hanno col virus, in modo da stabilire la probabilità che ha l'utente di essere affetto dal patogeno.
La procedura adottata per il sistema di diagnostica è logicamente un backward chaining, in un linguaggio logico (come può essere il Prolog) ci limiteremmo a chiedere lo stato di malattia e il sistema notando che ha degli atomi dal valore che non conosce, andrebbe in automatico a porre le domande sugli atomi che abbiamo definito 'askable'; tuttavia per le limitazioni della libreria usata siamo costretti ad operare con il forward chaining andando a chiedere prima le domande (e ogni domanda controlla le sue condizioni per essere chiesta), per poi determinare se è presente o meno una forma di malattia nell'individuo. L'idea generale si basa su una regola di derivazione, una forma generalizzata della regola di inferenza chiamata ***modus ponens***:

> Se "$h \leftarrow a_1 \wedge \ldots \wedge a_m$" è una clausola definita nella base di conoscenza e ogni a_i è stato derivato, allora h può essere derivato.

Dove:

- h è la **testa** dell'atomo,
- $a_1 \wedge \ldots \wedge a_m$ è il **corpo** della clausola, formato da a_i **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.
Nel caso del sistema di diagnostica del virus la base di conoscenza è stata organizzata dalle seguenti clausole:

$$sintomi\_base \Leftarrow perdita\_di\_peso$$

$$sintomi\_base \Leftarrow diarrea$$

$$vomito \Leftarrow nausea$$

$$cisti \Leftarrow esame\_positivo$$

$$cisti \Leftarrow no\_esame \wedge dolore\_addominale \wedge rigonfiamento$$

$$ulcera \Leftarrow dolore\_addominale \wedge acidità\_di\_stomaco \wedge nausea$$

$$malattia\ lieve \Leftarrow sintomi\_base \wedge cisti \wedge vomito$$

$$malattia\ grave \Leftarrow sintomi\_base \wedge ulcera$$

La malattia può manifestarsi in due forme diverse, una lieve e una grave. Le due sono accomunate da dei sintomi di base: una perdita di peso imprevista e attacchi di diarrea.
I sintomi che caratterizzano la forma lieve sono cisti e vomito, mentre la forma grave è caratterizzata dall'ulcera.
La presenza di una cisti può essere certa in caso in cui l'utente abbia sostenuto un esame medico con risultato positivo, oppure in caso di referto mancante il sistema tiene conto dei sintomi che possano rilevarne la presenza (come il rigonfiamento e il dolore nella zona addominale).
A sua volta, anche la presenza dell'ulcera viene individuata secondo il riscontro dell'utente alla presenza di sintomi quali acidità di stomaco, nausea e dolore addominale.

SISTEMA ESPERTO

Un sistema esperto è un'applicazione dell'intelligenza artificiale che vede un programma cercare di risolvere dei dati problemi, cercando di riprodurre i comportamenti di persone esperte in un determinato campo di attività.

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, e da una "*user interface*" che permette una facile interazione tra il sistema e l'utente.

IMPLEMENTAZIONE DEL SISTEMA ESPERTO

La prima implementazione del sistema si basa su un sistema esperto realizzato tramite il linguaggio **Python**, utilizzando la libreria **Experta**, che permette di associare dei fatti accaduti a delle regole riguardo gli stessi. Le regole sono formate da due componenti chiamati **LHS** (Left-Hand-Side) e **RHS** (Right-Hand-Side). Il primo descrive le condizioni che si devono verificare affinché la regola venga applicata, mentre il secondo è l'insieme di azioni che vengono compiute quando viene applicata la regola.
Per ogni sintomo è stata associata una regola che, quando viene applicata, si occupa di domandare all'utente se riscontra il sintomo e, in base alla sua risposta (positiva o negativa), il fatto relativo al sintomo viene impostato a true o a false, dopo di che il sistema interpreterà la situazione dell'utente, e di conseguenza applicherà altre regole relative ad altri sintomi.
In base alla combinazione di fatti relativi ai sintomi impostati a true, possono essere richiamate altre regole, relative a sintomi correlati alla malattia del virus (ulcera, cisti e vomito). Alla fine, in base ai macro-sintomi presenti, viene applicata una regola tra quelle che si occupano di comunicare all'utente la diagnosi stabilita dal sistema quando richiamate.

Il sistema comincia a porre le domande all'utente relative ai sintomi di base. Se l'utente riscontra almeno uno tra i due sintomi basilari (perdita di peso e diarrea), si prosegue con l'analisi cercando di stabilire di quale delle due forme può essere affetto l'utente. Il sistema andrà a verificare prima se si è affetti dalla forma lieve del virus, controllando se l'utente è affetto da tutti i sintomi dello stadio. Si domanderà, quindi, all'utente se avverte una sensazione di nausea e in caso di risposta positiva, verrà posta una domanda per capire se l'utente è stato anche soggetto ad attacchi di vomito. Il sistema proseguirà andando a stabilire se l'utente è affetto da cisti: nel caso in cui l'utente si è sottoposto a un esame specifico con esito positivo, la presenza di cisti è certa, mentre nel caso di esito negativo, sarà nulla. Nel caso in cui l'utente non si è sottoposto a un esame, il sistema andrà a stabilire se l'utente soffre di cisti, chiedendogli se riscontra i relativi sintomi: dolore addominale e rigonfiamento.
Se l'utente riscontra tutti i sintomi della forma lieve, verrà fornita la diagnosi positiva. In mancanza di un sintomo tra cisti e vomito, il sistema andrà a indagare riguardo la possibilità di essere affetti dal secondo stadio della malattia. Il sistema cercherà, così, di stabilire se l'utente soffre di ulcera, sempre ponendogli delle domande riguardo l'aver riscontrato sintomi quali dolore addominale, acidità di stomaco e nausea. Se l'utente riscontra tutti e tre

i sintomi allora il sistema stabilisce che l'utente soffre di ulcera e di conseguenza comunicherà che questi ha contratto la forma grave del virus.

ESEMPI TRATTI DAL CODICE

```
# MALATTIA LIEVE
@Rule(AND(Fact(sintomi_base=True), Fact(vomito=True), Fact(cisti=True)))
def malattia_lieve(self):
    print("I sintomi indicano che potresti aver contratto il virus.")
    print("È consigliabile recarsi da un medico da un medico per ulteriori accertamenti.")
    self.reset()
```

Questa regola si occupa di comunicare la diagnosi all'utente nel caso in cui il sistema ha stabilito che ha contratto la forma lieve della malattia. Nella parte sinistra della regola (LHS), si nota che le condizioni necessarie affinché la regola possa essere applicata sono che i fatti relativi a sintomi di base, vomito e cisti, siano impostati a True. Nella parte destra della regola (RHS) il sistema si occupa di comunicare tramite delle stampe la diagnosi stabilita.

```
# NAUSEA
@Rule(Fact(sintomi_base=True))
def ask_nausea(self):
    self.declare(Fact(nausea=ask_question("Avverti un senso di nausea?")))
```

Questa regola viene applicata se il fatto all'interno della parte sinistra (LHS) è True. Se applicata verrà posta la domanda all'utente, e il sistema in base alla risposta imposterà il fatto relativo al sintomo a True o False.

```
# VOMITO
@Rule(Fact(nausea=True))
def ask_vomito(self):
    self.declare(Fact(vomito=ask_question("Hai avuto attacchi di vomito ultimamente?")))
```

Se il fatto "nausea" è stato impostato a True nella regola precedente, viene applicata questa regola che si occupa di stabilire se l'utente sta soffrendo di attacchi di vomito.

RETE BAYESIANA

Come secondo modello abbiamo usato una rete bayesiana, un grafo aciclico direzionato (ossia un grafo dove gli archi hanno una direzione e nessun nodo ha archi che possono creare un ciclo tra i nodi stessi) dove ogni feature è espressa come nodo e la dipendenza tra le feature è espressa tramite un arco direzionato. In questo modo possiamo esprimere che l'attivazione di una feature dipende dall'attivazione di una o più feature, che andiamo a chiamare come genitore/i. Per far ciò dobbiamo creare una relazione di ordinamento tra le feature.

La rete bayesiana, come possiamo intuire dal nome, usa le probabilità e il teorema di Bayes per poter calcolare quanto è probabile un determinato evento dipendendo (o meno) da altri eventi (probabilità condizionata). Nel caso della rete bayesiana quindi ogni nodo con dei genitori ha una probabilità condizionata di $P(node|parents(node))$ dove $parents(node)$ è una funzione che restituisce i nodi genitore del nodo. In questo modo ogni nodo avrà una tabella delle probabilità condizionate dai suoi genitori. Alla fine per calcolare le probabilità di ogni nodo abbiamo la formula generale di:

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^{n} P(X_1|X_1, \dots, X_{n-1})$$

dove X_i sono le feature/nodi della rete bayesiana.

Ad esempio, nel nostro modello la malattia può causare sintomi come la perdita di peso, la nausea, etc. Utilizzando la rete bayesiana possiamo esplicitare questa dipendenza andando a creare dei nodi 'Malattia', 'Perdita di peso' e 'Ciste' con degli archi che partono da 'Malattia' e finiscono negli ultimi due nodi.

Allo stesso modo anche 'Ciste' dipende da 'Malattia' ma tra 'Ciste' e 'Perdita di peso' non ci sono archi che gli collegano direttamente e quindi le due feature sono indipendenti. In questa maniera possiamo imporre una relazione d'ordine tra i feature ed assegnare ad ogni feature dei genitori (se esistono).

Nel nostro modello la features malattia non ha genitori, ma volendo espandere il modello è possibile indicare i fattori di rischio della malattia (che potrebbero essere una dieta squilibrata, un ambiente malsano, etc.), in modo da indicare quali potrebbero essere le cause della malattia. Così possiamo anche andare ad incrementare l'accuratezza del modello e la sua efficacia a livello diagnostico.

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: esatta e approssimata.

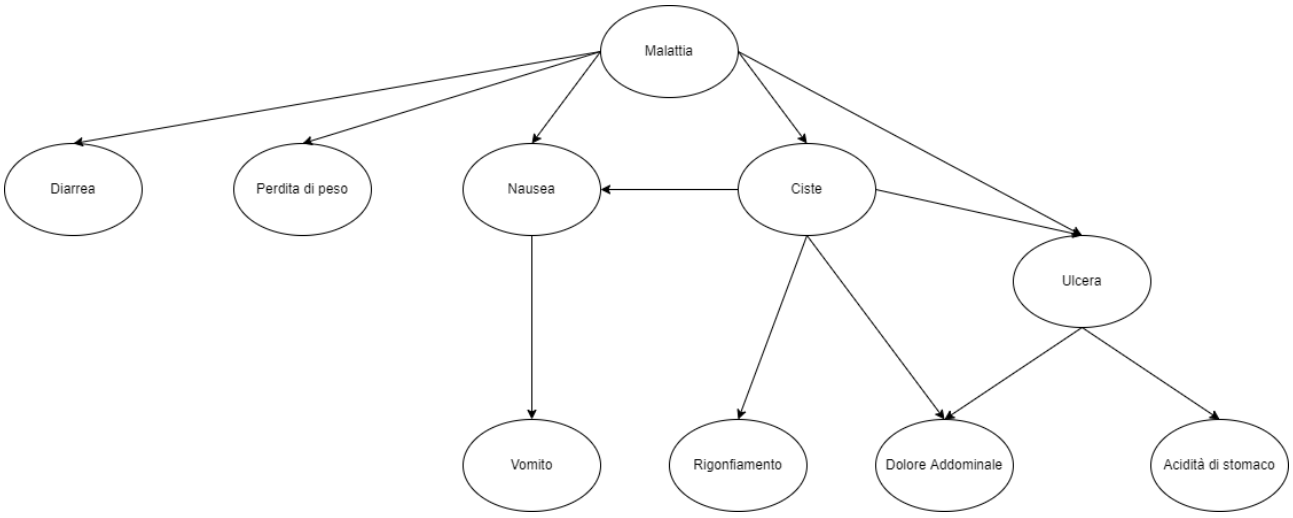
Nell'inferenza esatta si va ad enumerare i mondi coerenti con le osservazioni per poi sfruttare un algoritmo per calcolare la probabilità esatta dell'evento su cui stiamo indagando. Ci sono vari algoritmi per il calcolo e noi useremo quello dell'eliminazione di variabili (che è simile all'omonimo algoritmo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

Nell'inferenza approssimata invece 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 (nella realtà un problema di diagnostica medica soffre quasi certamente di questa problematica), 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 abbiamo una vasta scelta di soluzioni: stimatore della **massima verosimiglianza**, stimatori di **Bayes** (dove la stima non parte da una tabula rasa ma parte da una credenza precedente all'osservazione dei dati nella realtà; combinando la credenza precedente e i dati osservati si ottengono poi delle probabilità a posteriori che stimano ciò che stiamo cercando) oppure possiamo usare altri metodi come le regressioni, gli alberi di decisione, etc.

IMPLEMENTAZIONE

Sfruttando il fatto che stiamo lavorando con una malattia di fantasia abbiamo la libertà di poter sperimentare con una rete bayesiana con una struttura e delle tabelle date. Per prima cosa abbiamo la struttura della rete bayesiana così descritta:



Il grafo mostra chiaramente quali sono i sintomi della malattia, qual è la causa dei sintomi e anche delle variabili intermedie che a loro volta causano dei sintomi.

Per implementare la rete bayesiana in **Python** abbiamo fatto uso di **bnlearn**, una libreria che fa da wrapper ad un'altra libreria **pgmpy** che è il vero cuore pulsante.

bnlearn ci permette di creare una DAG, creare ed assegnare delle tabelle di probabilità condizionata per singolo nodo (con la classe TabularCPD) e poi di inferire le probabilità per un nodo della DAG andando a segnalare le osservazioni (ossia quali sintomi l'utente riporta al sistema), tramite il metodo della eliminazione delle variabili.

Inoltre, bnlearn permette di imparare anche le **tabelle delle probabilità** andando a **stimare** da un dataset che gli forniamo in input. In questo caso ci permette di scegliere due opzioni: **stimatore di massima verosimiglianza** e **stimatore di Bayes BDeu** (ossia la credenza precedente è costruita usando una distribuzione uniforme di Dirichlet e andando a confrontarla con i dati osservati normalizzati con dei pseudoconteggi).

Infine, ci permette di **apprendere** anche **la struttura** usando diversi metodi di ricerca (come la ricerca esaustiva, la ricerca Hillclimb, Chow-liu, etc.) che invece noi non useremo.

La prima cosa che dobbiamo fare (avendo una struttura data) è la **creazione della DAG**, per far ciò andiamo a creare un vettore di coppie, dove ogni coppia è formata da due nodi del grafo che vanno ad indicare un arco che li collega.

Come secondo punto (se stiamo lavorando con la distribuzione di probabilità data) andiamo a **creare e ad assegnare le tabelle delle probabilità**: ogni tabella e associata ad un nodo che andiamo ad esplicitare; andiamo anche ad esplicitare le evidenze per quel nodo (se il nodo ha genitori), e poi a dichiarare i possibili stati del nodo (nel nostro caso sono 2: presente, non presente) e infine a inserire le tabelle delle probabilità vere e proprie. La tabella ha come righe gli stati del nodo preso in esame e come colonne le combinazioni degli stati dei nodi genitori; così andiamo ad assegnare per ogni cella della tabella la probabilità che il nodo si trovi in un determinato stato dato la combinazione degli stati dei genitori. La somma degli elementi di una colonna deve essere uguale ad 1. Una volta creata una tabella per ogni nodo le andiamo ad assegnare alla DAG.

Arrivati a questo punto avremo una rete bayesiana funzionante, quello che ci rimane da fare è prendere le osservazioni del paziente (i sintomi che dichiara di avere) ed **inferire la probabilità** che il paziente abbia la malattia **conoscendo lo stato** dei sintomi. Per far ciò chiediamo (attraverso delle domande) i sintomi che il paziente ha; ad ogni sintomo osservato gli assegniamo il valore 0 se non è presente 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 della malattia, che andiamo poi a mostrare all'utente insieme ad un avvertimento se la malattia ha una probabilità maggiore o uguale al 50%.

Altrimenti come 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 **paziente osservato e testato** per la malattia a cui associamo i vari sintomi che possiede. Una volta passati questi input decidiamo il tipo di stimatore da usare (nel nostro programma lasciamo la libertà all'utente) e così otteniamo nuovamente una rete bayesiana funzionante ma questa volta le probabilità sono apprese e non date. Questa è sicuramente una situazione più realistica del primo caso, in quanto è impossibile sapere a priori quali sono le probabilità che una malattia abbia un certo sintomo. Nella realtà ciò ci è sconosciuto e tutto quello che abbiamo, non sono altro che osservazioni parziali della realtà.

VALUTAZIONE

Potremmo chiederci "**ma che cosa cambia tra i due metodi di stima? Uno dei due è il migliore?**". Dopotutto nella realtà non ci ritroveremo mai con una malattia con delle probabilità note, ci ritroveremmo molto più probabilmente con solo una osservazione parziale della realtà a cui dovremmo dare un senso, e quindi conoscere i pro e i contro dei due metodi di stima delle probabilità che usiamo risulta importante al paziente o al medico che si ritroverà ad usare, ipoteticamente, uno strumento simile a questo.

Per rispondere a questa domanda abbiamo **bisogno** di compiere **molte simulazioni** con i dati che abbiamo a disposizione; divideremo i dati che abbiamo in 2 dataset differenti (nel nostro caso 2 dataset non necessariamente uguali, ma esistono altri metodi come, per esempio, un dataset si può dividere in n parti uguali), la prima parte la useremo per la fase di **training** del modello (ossia quella parte in cui cerchiamo di stimare le probabilità dei nodi della BN) e la seconda parte come **test** del modello (ossia nel dataset abbiamo sia i dati dei sintomi del paziente e conosciamo anche se il paziente ha o meno la malattia, quello che facciamo e che forniamo come evidenza sono i sintomi della malattia e confrontiamo la risposta che inferiamo al modello con quella che già sappiamo nel dataset [ma che non forniamo al modello stesso]). In questo modo possiamo vedere come si comporta il modello con centinaia o migliaia di simulazioni di funzionamento. Ai fini della simulazione considereremo una probabilità maggiore o uguale al 50%, fornita dal modello, come una risposta positiva mentre una probabilità inferiore è considerata una risposta negativa (potremmo anche usare una diversa percentuale, in base allo scenario che ci troviamo ad analizzare potremmo preferire una diversa percentuale).

Come primo scenario da analizzare, forniamo un database di training fisso (che non cambia mai), e vediamo come i due stimatori si comportano al variare dei dati di test (in questo modo vediamo come si comportano i due stimatori in situazioni diverse ma a parità di apprendimento). Questo test ci servirà per analizzare il comportamento simulando situazioni diverse a quelle su cui si sono allenati i due modelli.

Verosomiglianza (cambio database test)				
Numero	Accuratezza	Precisione	Richiamo	F1
1	0,9977	0,7273	0,4848	0,5818
2	0,9967	0,6818	0,3659	0,4762
3	0,9974	0,6923	0,5000	0,5806

Bayes (cambio database test)				
Numero	Accuratezza	Precisione	Richiamo	F1
1	0,9969	0,5312	0,5152	0,5231
2	0,9967	0,6429	0,4390	0,5217
3	0,9969	0,5758	0,5278	0,5508

(i dati sono stati ottenuti caricando 1 dataset di training fisso e 3 dataset di test con la funzione di simulazione presente nel menù del modello a rete bayesiana. La funzione addestra la BN con il dataset di training e poi, per ogni riga nel dataset di test, inferisce le probabilità con i sintomi osservati e controlla che il risultato ottenuto corrisponda o meno a quello presente nel dataset di test)

Ad una prima vista possiamo notare che lo **stimatore di verosimiglianza** risulta in media più accurato e con una **precisione** di molto **superiore** rispetto a quello di Bayes (dove per precisione intendiamo i numeri di **veri positivi trovati** sui **positivi trovati**). Quindi saremmo tentati a decretare il primo stimatore come il migliore, ma qui sorgerebbe un problema. Infatti, come possiamo notare lo **stimatore di Bayes** ha in media un **richiamo maggiore** (dove per richiamo intendiamo i **veri positivi trovati** sui **positivi**). Dai dati possiamo quindi comprendere che in media lo stimatore di verosimiglianza sia più preciso e che quindi quando trova un positivo è più probabile che quel positivo sia un vero positivo, invece che risultare un falso positivo; d'altro canto, il richiamo maggiore dello stimatore di Bayes ci fa intendere che quest'ultimo sia più propenso a trovare i positivi rispetto a quello di verosimiglianza (a costo anche di aumentare il numero di falsi positivi).

A complicare le cose è il **punteggio F1** (un punteggio riassuntivo di precisione e richiamo) che in 2 casi su 3 è migliore nello stimatore di verosimiglianza, ma nel caso restante decreta come il migliore lo stimatore di Bayes. Un'altra cosa che possiamo notare è che lo stimatore di verosimiglianza tende a variare maggiormente rispetto a quello di Bayes. Quindi cosa possiamo ottenere da questa lettura dei dati?

Quello che possiamo comprendere è che non esiste uno stimatore migliore in assoluto: Ogni stimatore ha dei vantaggi e svantaggi e, in base al problema che ci ritroviamo ad affrontare, potrebbe essere preferibile uno rispetto all'altro. Ad esempio, come possiamo notare nelle matrici di confusione qui sotto, lo stimatore tende a trovare più positivi in generale e quindi potrebbe ritrovarsi più utile nel caso in cui una malattia sia particolarmente mortale (come il cancro) e che quindi una preferenza a trovare positivi anche quando non ce ne sono è più auspicabile rispetto a lasciare la malattia non diagnosticata. Viceversa, nel caso in cui la malattia non risulti particolarmente preoccupante e invece i costi di una diagnosi errata siano più alti, è preferibile una diagnosi il più precisa possibile.

Verosomiglianza: 1		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	16	17	33
	Non Malato	6	9961	9967

Bayes: 1		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	17	16	33
	Non Malato	15	9952	9967

Verosomiglianza: 2		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	15	26	41
	Non Malato	7	9952	9959

Bayes: 2		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	18	23	41
	Non Malato	10	9949	9959

Verosomiglianza: 3		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	18	18	36
	Non Malato	8	9956	9964

Bayes: 3		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	19	17	36
	Non Malato	14	9950	9964

Un altro aspetto da considerare è anche il nostro ambiente di test stesso. Infatti, visto che ci siamo basando su una malattia che non esiste e ci siamo generati da noi stessi il datset, non possiamo simulare un'altra condizione che avviene spesso nel mondo reale. Ossia, spesso nella realtà accade che i dataset vengano distorti da una serie di condizioni e che spesso possono quindi portare ad un dataset di training che non rispecchia fedelmente la realtà. Quindi, potenzialmente, ci potremmo ritrovare un modello di BN che è molto bravo nel riconoscere le situazioni di training&test, ma con delle pessime performance nel mondo reale (questo problema è chiamato **overfitting**). Evitare questo problema potrebbe essere indispensabile nel caso in cui si abbiamo dati la cui attendibilità nel mondo reale è incerta e quindi potrebbe risultare più interessante lo stimatore di Bayes, che sembra conformarsi di meno al dataset di training.

Infine, un ultimo problema del nostro ambiente di test è sicuramente la ridotta dimensione dei test stessi, che difficilmente possono avere un valore statistico vista la quantità di simulazioni ridotte. Questo problema è principalmente causato dalla poca performance del sistema nel simulare migliaia e migliaia di dati, e questo ci impedisce di condurre molti più test.

Questi test sono stati condotti con 1 dataset di training e 3 di test e sono stati usati gli stessi dataset di test per confrontare tra loro direttamente i due stimatori. La dimensione del dataset di training è stata fissata a 10.000 elementi e anche quella di test ha identica dimensione (per il test è stata scelta questa dimensione in quanto sperimentalmente si è visto che a dimensioni minori fornisce dati di scarsa rilevanza, infatti spesso i due modelli non trovano positivi, mentre dimensioni maggiori di 10.000 hanno un impatto peggiorativo delle performance del sistema).

Come **secondo caso** di test invece andiamo a testare qual è la quantità ottimale di dati da fornire al sistema nella fase di training, in questa fase del test andiamo quindi a variare la dimensione del dataset di training ma manteniamo costante il dataset di test.

Cercheremo, quindi, di capire qual è la quantità sufficiente di dati da fornire al modello per ottenere dei risultati accettabili (questo test è molto utile soprattutto quando non si hanno molti dati su cui basare l'addestramento).

Quindi eseguiamo 5 test con dataset di training di dimensione uguale a: 1.000, 5.000, 10.000, 25.000, 50.000. Mandando in esecuzione i 5 test per modello, otteniamo i seguenti dati:

Verosomiglianza (cambio database training)				
Dimensione training set	Accuratezza	Precisione	Richiamo	F1
1.000	0,9973	0,6667	0,3636	0,4706
5.000	0,9975	0,7000	0,4242	0,5283
10.000	0,9977	0,7273	0,4848	0,5818
25.000	0,9977	0,7273	0,4848	0,5818
50.000	0,9977	0,7273	0,4848	0,5818

Bayes (cambio database training)				
Dimensione training set	Accuratezza	Precisione	Richiamo	F1
1.000	0,9968	0,5152	0,5152	0,5152
5.000	0,9969	0,5312	0,5152	0,5231
10.000	0,9969	0,5312	0,5152	0,5231
25.000	0,9969	0,5312	0,5152	0,5231
50.000	0,9972	0,5862	0,5152	0,5484

(anche questi dati sono stati ottenuti con la funzione presente del menù del modello a rete bayesiana)

Quello che possiamo immediatamente osservare è come lo stimatore di verosimiglianza sembra raggiungere un massimo locale una volta arrivato a 10.000 elementi su cui si è addestrato. Mentre lo stimatore di Bayes sembra raggiungere un picco locale immediatamente a 5.000 (in termini di accuratezza e di precisione), per poi migliorare a 50.000 elementi. Tuttavia, il richiamo non sembra migliorare in nessuna simulazione, rimane identico per tutta la durata del test.

Quindi quello che possiamo dire è che 10.000 sembra un numero di elementi sufficienti per il training di verosimiglianza, e che andare oltre a tale numero di elementi non sembra portare benefici (tuttavia proseguendo con i test potremmo anche notare un marginale miglioramento andando avanti con dimensioni via via maggiori, forse a 100.000 potevamo apprezzare un miglioramento oppure no; purtroppo, le prestazioni limitano la quantità di test che possiamo effettuare).

Mentre quello che possiamo invece notare su Bayes è che forse aumentando la quantità di elementi aumenta la precisione (ma non il richiamo) e che quindi aumentare gli elementi per

Bayes potrebbe anche essere vantaggioso se vogliamo un sistema più preciso senza voler perdere in richiamo. Questo sembra in linea con il funzionamento dello stimatore di Bayes, più dati forniamo più è disposto a cambiare la credenza precedente in favore delle osservazioni. Tuttavia, non possiamo escludere che l'aumento della precisione a 50.000 elementi non sia una anomalia statistica. In basso sono riportare le matrici di confusione:

Verosomiglianza: 1000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	12	21	33
	Non Malato	6	9961	9967

Bayes: 1000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	17	16	33
	Non Malato	16	9951	9967

Verosomiglianza: 5000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	14	19	33
	Non Malato	6	9961	9967

Bayes: 5000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	17	16	33
	Non Malato	15	9952	9967

Verosomiglianza: 10000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	16	17	33
	Non Malato	6	9961	9967

Bayes: 10000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	17	16	33
	Non Malato	15	9952	9967

Verosomiglianza: 25000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	16	17	33
	Non Malato	6	9961	9967

Bayes: 25000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	17	16	33
	Non Malato	15	9952	9967

Verosomiglianza: 50000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	16	17	33
	Non Malato	6	9961	9967

Bayes: 50000		Predizione		Somma
		Malato	Non Malato	10000
Realtà	Malato	17	16	33
	Non Malato	12	9955	9967

I dati riportati qui sono trovabili anche in data/valutazione.xlsx