UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Documentazione di progetto per Ingegneria della conoscenza a.a. 2022/2023

Nome: Gabriele

Cognome: Grieco

Matricola: 700234

Mail istituzionale g.grieco19@studenti.uniba.it

Repository URL https://github.com/GaberBarnabass/Icon22-23Uniba

Questo progetto è un tool diagnostico che ha come dominio le malattie dell'apparato digerente, attualmente può individuare solo 3 malattie:

- Sindrome dell'intestino irritabile (IBS)
- Rettocolite ulcerosa (UC)
    - In diverse forme in base alla gravità:
        - Lieve
        - Grave
        - Fulminante
- Morbo di Crohn
    - Anche qui si presentano due forme in base alla gravità:
        - Lieve
        - grave

Al fine di identificare una malattia questo utilizza due modelli:

- una base di conoscenza
- una rete bayesiana

Il progetto è stato creato in python (versione 3.8) utilizzando le seguenti librerie:

- bnlearn per la rete bayesiana
- experta per la KB
- pandas per la gestione del dataset
- sklearn per la validazione

La base di conoscenza

Cosa è una base di conoscenza:

una base di conoscenza è un sistema esperto che utilizza un insieme di fatti e regole per prendere decisioni o fare inferenza su uno specifico dominio di interesse. Un fatto è un'informazione, l'unità base della conoscenza, che viene utilizzata per determinare se una regola deve essere attivata o meno, una regola invece, si compone di due parti: body e head; il body è l'insieme delle condizioni che si devono attivare affinché la testa, una o più azioni, si possa verificare.

Implementazione della base di conoscenza

La KB è stata implementata utilizzando la libreria Experta un'alternativa a CLIPS completamente scritta in python per creare dei rule-based systems, la quale utilizza l'algoritmo RETE, un pattern matching algorithm per l'implementazione di rule based-system. Questo determina quali regole devono essere attivate in base ai fatti dichiarati nella KB. È compatibile con python 3.X fino alla 3.8 (versione utilizzata in questo progetto).

L’ elemento principale della libreria è la classe Rule che può definire una regola, una regola si compone di due parti:

- Left Hand Side (LHS) chiamata anche body:
  un insieme di condizioni (pattern) che si deve verificare affinché la regola sia eseguita;
- Right Hand Side (RHS) chiamata anche head:
  un insieme di azioni che devono essere eseguite se le condizioni nella LHS si sono verificate.

I fatti invece sono istanze della classe Fact, i quali assieme alle regole sono racchiuse nel KnowledgeEngine, il cuore della libreria che rappresenta la KB vera e propria.

Per evitare interazioni inutili si sarebbe preferito creare il sistema come backward chaining, prendere le 6 regole principali che identificano le malattie e fare un dialogo con l’utente al bisogno, dichiarano atomi come askable. Poiché questo non è possibile si è costretti ad usare un forward chaining, ovvero chiedere tutti i fatti all’utente prima di fare inferenza. Experta però, nella definizione delle regole, permette di specificare un attributo `salience` che permette di dare importanza ad alcune regole rispetto ad altre, creando così un dialogo strutturato che parte dall’identificazione dell’IBS (quella con meno sintomi) fino al CROHN (quella con più sintomi). Poiché tutte le malattie hanno sintomi in comune questo approccio non causa un dialogo inutile, anzi, il dialogo è per la maggior parte strutturato seguendo sempre un certo ordine.

Le principali regole nella base di conoscenza sono:

ibs:

Ibd (categoria malattie infiammatorie dell'intestino)

```
@Rule(
    AND(
        NOT(Disease(ibs=yes)),
        Symptom(stool_type=diarrhea),
        Symptom(abdominal_pain=yes),
        Symptom(abdomen_cramps=yes),
        OR(
            Symptom(bloating=yes),
            Symptom(bloating=no)
        )
    ),
    salience=1999)
def is_ibd(self):
    self.declare(Disease(ibd=yes))
```

Ulcerative colitis

```
@Rule(
    AND(
        Disease(ibd=yes),
        OR(
            Symptom(mucus_with_stool=yes),
            Symptom(pus_with_stool=yes),
        ),
        Symptom(tenesmus=yes)
    ),
    salience=1998
)
def ulcerative_colitis(self):
    self.declare(Disease(uc=yes))
```

Mild ulcerative colitis

```
@Rule(
    AND(
        Disease(uc=yes),
        Symptom(bowel_movements_per_day=bmpd_lt_5),
        OR(
            Symptom(blood_with_stool=no),
            Symptom(blood_with_stool_frequency=bis_rare)
        )
    ),
    salience=1997
)
def mild_ulcerative_colitis(self):
    self.declare(Disease(mild_uc=yes))
```

Anemia

```
@Rule(
    AND(
        Fact(tested_for_anemia=no),
        Symptom(fatigue=yes),
        Symptom(weakness=yes),
        Symptom(pale_skin=yes),
        OR(
            Symptom(tachycardia=yes),
            Symptom(chest_pain=yes),
            Symptom(shortness_of_breath=yes)
        ),
        OR(
            Symptom(dizziness=yes),
            Symptom(lightheadedness=yes),
            Symptom(headache=yes)
        ),
        Symptom(brittle_nails=yes),
        Symptom(cold_feet=yes),
        Symptom(cold_hands=yes)
    ),
    salience=1996
)
def anemia(self):
    self.declare(Symptom(anemia=yes))
```

severe ulcerative colitis

```
@Rule(
    AND(
        Disease(uc=yes),
        NOT(Disease(mild_uc=yes)),
        Symptom(bowel_movements_per_day=bmpd_gt_5_and_lt_10),
        Symptom(blood_with_stool_frequency=bis_frequent),
        Symptom(anemia=yes),
        Symptom(weight_loss=yes),
        Symptom(appetite_loss=yes),
        OR(Symptom(fever=yes), Symptom(fever=no)),
        OR(Symptom(nausea=yes), Symptom(nausea=no)),
        OR(Symptom(vomiting=yes), Symptom(vomiting=no))
    ),
    salience=1995
)
def severe_ulcerative_colitis(self):
    self.declare(Disease(severe_uc=yes))
```

Fulminant Ulcerative Colitis

```
@Rule(
    AND(
        Disease(uc=yes),
        Symptom(bowel_movements_per_day=bmpd_gt_10),
        OR(
            Symptom(blood_with_stool_frequency=bis_frequent),
            Symptom(blood_with_stool_frequency=bis_continuous)
        ),
        Symptom(anemia=yes),
        Symptom(weight_loss=yes),
        Symptom(appetite_loss=yes),
        OR(Symptom(fever=yes), Symptom(fever=no)),
        OR(Symptom(nausea=yes), Symptom(nausea=no)),
        OR(Symptom(vomiting=yes), Symptom(vomiting=no))
    ),
    salience=1994
)
def fulminant_ulcerative_colitis(self):
    self.declare(Disease(fulminant_uc=yes))
```

Mild Crohn

```
@Rule(
    AND(
        Disease(ibd=yes),
        Symptom(tenesmus=no),
        Symptom(pus_with_stool=no),
        Symptom(mucus_with_stool=no),
        Symptom(mouth_sores=yes),
        Symptom(appetite_loss=yes),
        Symptom(weight_loss=yes),
        Symptom(fistula=yes),
        OR(
            AND(Symptom(blood_with_stool=yes),
                # if yes, frequency depends on the location
                OR(Symptom(blood_with_stool_frequency=L(bis_rare) | L(bis_frequent) |
L(bis_continuous)))),
            Symptom(blood_with_stool=no)
        ),
        OR(Symptom(fever=yes), Symptom(fever=no))
    ),
    salience=1993
)
def mild_crohn_disease(self):
    self.declare(Disease(mild_crohn=yes))
```

Severe Crohn

```
@Rule(
    AND(
        Disease(mild_crohn=yes),
        Symptom(anemia=yes),
        Symptom(red_tender_bumps_under_the_skin=yes),
        OR(Symptom(fever=yes), Symptom(fever=no)),
        OR(Symptom(eye_pain=yes), Symptom(eye_pain=no)),
        OR(Symptom(eye_redness=yes), Symptom(eye_redness=no)),
        OR(Symptom(joint_pain=yes), Symptom(joint_pain=no)),
        OR(Symptom(joint_soreness=yes), Symptom(joint_soreness=no))
    ),
    salience=1992
)
def severe_crohn_disease(self):
    self.declare(Disease(severe_crohn=yes))
```

Fintanto che la combinazione di sintmi corrisponde ad una regola della Knowledge Base il sistema riesce ad identificare correttamente le 3 malattie nelle loro forme di gravità. Quando però la combinazione di sintomi non corrisponde esattamente a quelle specificate la base di conoscenza non riesce a identificare la malattia. Qui entra in gioco la rete bayesiana.

La rete bayesiana

Cosa è una rete bayesiana:

una rete bayesiana o bayesian network o bayesian belief network (BBN) è un modello probabilistico utilizzato per fare inferenza quando c'è incertezza. Rappresenta un insieme di variabili e le loro relazioni di dipendenza mediante un grafo aciclico diretto (DAG) dove ogni nodo nel grafo corrisponde ad una variabile randomica ed ogni arco rappresenta una dipendenza. Ogni nodo e quindi ogni variabile viene associato ad con una distribuzione di probabilità per ogni possibile valore che la variabile può assumere. I nodi senza archi entranti sono detti nodi root, mentre gli altri sono detti nodi figli e rappresentano le variabili che sono dipendenti (condizionate) dalle variabili genitori.

Le probabilità condizionate sono determinate utilizzando il teorema di bayes che permette di calcolare la probabilità dello stato di una variabile date delle evidenze ovvero i genitori.

Come vengono calcolate le tavole di probabilità condizionata?

Le tavole di probabilità condizionata CPDs vengono calcolate mediante due metodi

- Maximum likelihood estimator MLE:

  metodo per trovare i parametri migliori di un modello che molto probabilmente produrrebbero i dati osservati e che quindi massimizzino la funzione di massima verosomiglianza, funzione che misura la probabilità dei dati osservati. MLE è noto per adattarsi troppo ai dati osservati (overfitting).

- Bayesian estimator:

  metodo per stimare i parametri del modello basandosi sui dati osservati usando l'inferenza bayesiana. A differenza di MLE usa una conoscenza a priori o credenze (beliefs) che mediante il teorema di bayes viene aggiornata creando così la distribuzione a posteriori dei parametri. Ottimo quando si ah un dataset ristretto poiché non soffre di overfitting o quando il modello è complesso

Quando la rete verrà costruita sarà possibile inferire l probabilità che una determinata malattia sia presente avendo osservato dei sintomi.

(P(disease|evidence1 & evidence2 & …. & evidenceN)).

Inferenza:

ci sono due tipi principali di inferenza

- Exact inference:
  computa esattamente la probabilità della variabile target usando algoritmi come variable elimination.
  - Variable Elimination consiste nell'eliminare iterativamente le variabili dalla rete in uno specifico ordine, determinato dalle dipendenze delle osservazioni. La CPD della variabile eliminata viene combinata con le CPDs vicine per produre una nuova CPD che rappresenta la probabilità congiunta delle variabili rimanenti.
- Approximate Inference:
  comporta l'utilizzo di metodi di campionamento come Markov Chain Monte Carlo per approssimare le probabilità a posteriori delle variabili target.

Implementazione della rete bayesiana:

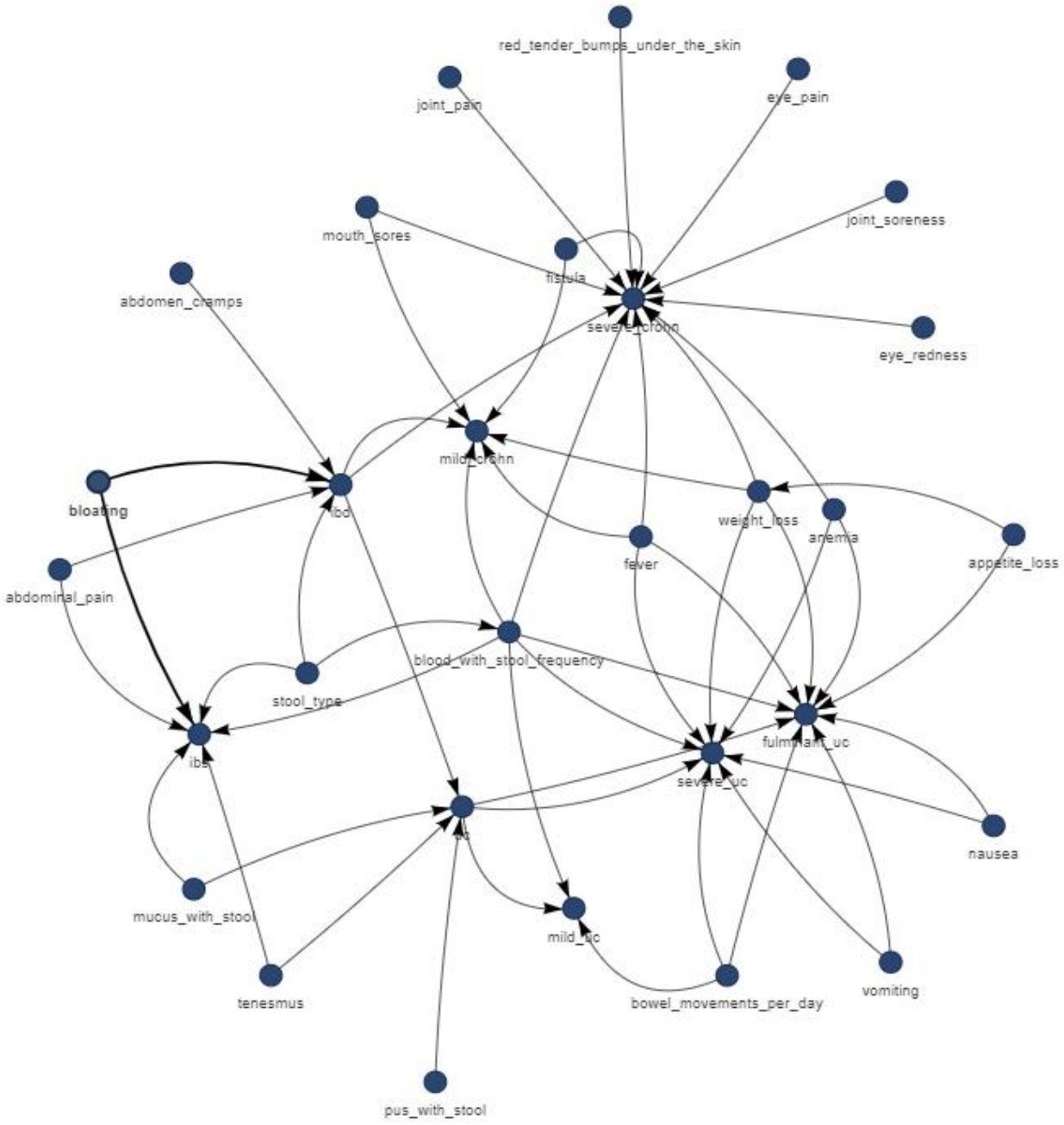
La BBN è stata implementata mediante l'uso della libreria python BNLearn la quale rende più semplice e intuitivo l'utilizzo di pgmpy che fornisce strumenti e algoritmi per manipolare e analizzare modelli probabilistici tra cui le reti bayesiane.

Il DAG

Per la creazione del grafo sono state definite le dipendenze tra i vari nodi:

```
self.edges = [
    # irritable bowel disease
    (bloating, ibs),
    (tenesmus, ibs),
    (mucus_with_stool, ibs),
    (abdominal_pain, ibs),
    (stool_type, ibs),
    (blood_with_stool_frequency, ibs),
    # dependency between the bleeding given the stool type
    (stool_type, blood_with_stool_frequency),
    # inflammable bowel disease
    (bloating, ibd),
    (stool_type, ibd),
    (abdominal_pain, ibd),
    (abdomen_cramps, ibd),
    # ulcerative colitis
    (ibd, uc),
    (mucus_with_stool, uc),
    (pus_with_stool, uc),
    (tenesmus, uc),
    # mild ulcerative colitis
    (uc, mild_uc),
    (bowel_movements_per_day, mild_uc),
    (blood_with_stool_frequency, mild_uc),
    # dependency of the weight loss given the appetite loss
    (appetite_loss, weight_loss),
    # severe ulcerative colitis
    (uc, severe_uc),
    (bowel_movements_per_day, severe_uc),
    (blood_with_stool_frequency, severe_uc),
    (anemia, severe_uc),
    (weight_loss, severe_uc),
    (fever, severe_uc),
    (vomiting, severe_uc),
    (nausea, severe_uc),
    # fulminant ulcerative colitis
    (uc, fulminant_uc),
    (bowel_movements_per_day, fulminant_uc),
    (blood_with_stool_frequency, fulminant_uc),
    (anemia, fulminant_uc),
    (weight_loss, fulminant_uc),
    (appetite_loss, fulminant_uc),
    (fever, fulminant_uc),
    (vomiting, fulminant_uc),
    (nausea, fulminant_uc),
    # mild crohn
    (ibd, mild_crohn),
    (mouth_sores, mild_crohn),
    (weight_loss, mild_crohn),
    (fistula, mild_crohn),
    (blood_with_stool_frequency, mild_crohn),
    (fever, mild_crohn),
    # severe crohn
    (ibd, severe_crohn),
    (mouth_sores, severe_crohn),
    (weight_loss, severe_crohn),
    (fistula, severe_crohn),
    (blood_with_stool_frequency, severe_crohn),
    (fever, severe_crohn),
    (anemia, severe_crohn),
    (red_tender_bumps_under_the_skin, severe_crohn),
    (eye_pain, severe_crohn),
    (eye_redness, severe_crohn),
    (joint_pain, severe_crohn),
    (joint_soreness, severe_crohn)
]
```

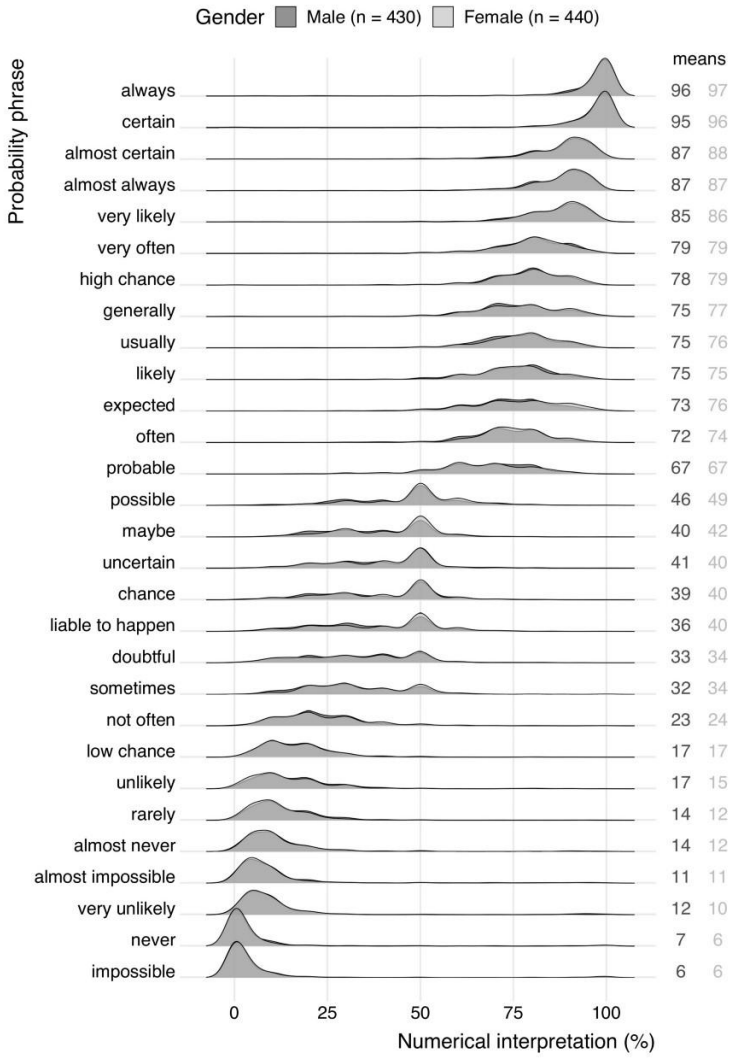
Questo è il grafo che si verrà a creare, mostrando tutte le dipendenze



Le CPDs possono essere specificate o possono essere create automaticamente da bnlearn, come in questo caso, dove si è utilizzato lo stimatore bayesiano. Per l'inferenza bnlearn usa l'algoritmo di Variable Elimination.

Il dataset:

Non avendo trovato dataset riguardanti il dominio di interesse, i dati sono stati generati seguendo le regole della base di conoscenza e di conseguenza le predizioni erano accurate fintanto che i sintomi combaciavano con quelli della base di conoscenza. Poiché l'idea della rete bayesiana è quella di rafforzare la base di conoscenza in caso di mancata identificazione della malattia si è generato del `rumore` ovvero con probabilità molto basse si è generato un dataset dove alcuni esempi della malattia X contengono anche sintomi della malattia Y e Z.



Seguendo come riferimento questa immagine presa da questo articolo di uno dei creatori della libreria dove spiega che impossibile non è mai impossibile.

Ovvero se un sintomo è impossibile nella malattia non è detto che non possa esserci, magari può essere causato da altro

Ad esempio, nell'IBS l'anemia non è una conseguenza della malattia ma potrebbe essere presente per altre motivazioni.

Quindi impossibile diventa 6% sì e 94% no.

Questo codice genera gli esempi della rettocolite ulcerosa lieve dove si può vedere che l'anemia sarà presente con un 6% di probabilità.

```
print('Generating mild ulcerative colitis examples')
for i in range(0, max_n):
    symptoms_dict_mild_uc0 = {
        bloating: random.choices([yes, no], weights=[46, 54])[0],
        tenesmus: yes,
        mucus_with_stool: yes,
        pus_with_stool: no,
        abdominal_pain: yes,
        abdomen_cramps: yes,
        stool_type: diarrhea,
        bowel_movements_per_day: bmpd_lt_5,
        blood_with_stool_frequency: random.choices([bis_rare, 'None'],
                                                   weights=[40, 60])[0],
        uc: yes,
        ibd: yes,
        anemia: random.choices([yes, no], weights=[6, 94])[0],
        fever: random.choices([yes, no], weights=[6, 94])[0],
        nausea: random.choices([yes, no], weights=[6, 94])[0],
        vomiting: random.choices([yes, no], weights=[6, 94])[0],
        weight_loss: random.choices([yes, no], weights=[6, 94])[0],
        appetite_loss: random.choices([yes, no], weights=[6, 94])[0],
        mouth_sores: random.choices([yes, no], weights=[6, 94])[0],
        fistula: random.choices([yes, no], weights=[6, 94])[0],
        red_tender_bumps_under_the_skin: random.choices([yes, no], weights=[6, 94])[0],
        eye_pain: random.choices([yes, no], weights=[6, 94])[0],
        eye_redness: random.choices([yes, no], weights=[6, 94])[0],
        joint_pain: random.choices([yes, no], weights=[6, 94])[0],
        joint_soreness: random.choices([yes, no], weights=[6, 94])[0],
        diagnosis: mild_uc
    }
```

Questo permette di gestire esempi che non possono essere identificati dalla base di conoscenza.

Il dataset si compone di 60000 esempi, 10000 per ogni malattia e si compone di 24 sintomi e la classe target `diagnosis`.

Per la maggior parte dei sintomi i valori sono 0 (no) / 1 (si) per altri invece ci sono più più valori come, ad esempio, blood_with_stool_frequency che può assumere quattro valori distinti.

Un esempio di uso della rete bayesiana:

```
{'bloating': 1, 'tenesmus': 1, 'mucus_with_stool': 1, 'pus_with_stool': 0, 'abdominal_pain': 1,
'abdomen_cramps': 0, 'stool_type': 'diarrhea', 'blood_with_stool_frequency': 'None', 'anemia': 0,
'fever': 0, 'nausea': 0, 'vomiting': 0, 'weight_loss': 0, 'appetite_loss': 0, 'mouth_sores': 0,
'fistula': 0, 'red_tender_bumps_under_the_skin': 0, 'eye_pain': 0, 'eye_redness': 0, 'joint_pain':
0, 'joint_soreness': 0, 'bowel_movements_per_day': '> 2 and <= 5 per day'}

The system detected a form of Irritable Bowel Syndrome with probability 63.732516472084l6.

For more information visit the following page of the U.S. institute of Diabetes and and Kidney
Disease:

https://www.niddk.nih.gov/health-information/digestive-diseases/irritable-bowel-
syndrome/definition-facts
```

Validazione della rete bayesiana:

per la validazione della rete bayesiana è stato usato il K-fold cross validation scegliendo k = 10 data la maggiore accuratezza

di seguito viene riportato il report creato mediante il classification_report di sklearn

```
              precision    recall  f1-score   support

         ibs       1.00      1.00      1.00     10000
    mild_uc        1.00      1.00      1.00     10000
  severe_uc        0.98      0.97      0.98     10000
fulminant_uc       1.00      1.00      1.00     10000
  mild_crohn       0.97      0.98      0.98     10000
severe_crohn       1.00      1.00      1.00     10000

    accuracy                           0.99     60000
   macro avg       0.99      0.99      0.99     60000
weighted avg       0.99      0.99      0.99     60000


Process finished with exit code 0
```