Mental Diagnosis

Generated by Doxygen 1.9.2

Luca Lobascio, matr. 676614

Repository progetto: https://github.com/ll2909/ICon_mental_diagnosis

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 BBN Namespace Reference

Functions

- def getDiagProb (symCatList)
- def getWeightProb (list1, list2)
- def getFinalProbCat (list)
- def predictDiag (conn, probCat, percOcc)

3.1.1 Function Documentation

3.1.1.1 getDiagProb()

```
def BBN.getDiagProb (
              symCatList )
```

```
Funzione che riceve in input la lista delle categorie dei sintomi inseriti dall'utente,
e tramite l'uso di una Bayesian Belief Network viene calcolata la probabiltà della categoria
di una diagnosi condizionata dalla categoria dei sintomi, infine restituita sotto forma di lista

Parameters
----------
symCatList: list
    Lista contenente le categorie dei rispettivi sintomi forniti in input

Returns
-------
condProbs: list
    Lista contenente delle tuple, ogni tupla contiene la categoria e la
    corrispondente probabilità di diagnosi
```

3.1.1.2 getFinalProbCat()

```
def BBN.getFinalProbCat (
              list )
```

```
Funzione che effettua la somma di tutte le probabilità della stessa categoria
all'interno della lista in input, restituendo una lista contenenti le probabilità
finali

Parameters
----------
list: list
    lista contenente le probabilità pesate delle diagnosi

Returns
-------
res: list
    lista contenente le probabiltà finali per ciascuna categoria di diagnosi
```

3.1.1.3 getWeightProb()

```
def BBN.getWeightProb (
              list1,
              list2 )
```

```
Funzione che prende in input una lista con le percentuali delle categorie dei sintomi
e una lista con le probabilità delle diagnosi. Restituisce una lista con le probabilità
pesate delle diagnosi, ottenute moltiplicando ogni probabilità delle categorie dei sintomi
con le corrispettive delle diagnosi

Parameters
----------
list1: list
    Lista contenente le percentuali per ciascuna categoria di sintomi
list2: list
    Lista contenente le percentuali per ciascuna categoria di diagnosi
Returns
-------
res: list
    Lista contenenti le probabilità pesate delle diagnosi
```

3.1.1.4 predictDiag()

```
def BBN.predictDiag (
              conn,
              probCat,
              percOcc )
```

```
Funzione che calcola le probabilità di ciascuna diagnosi, il cui risultato
è un dizionario con diagnosi e rispettiva percentuale

Parameters
----------
conn: connection
    Connessione al database
probCat: list
    Lista contenente le probabilità che la diagnosi appartenga ad una categoria
percOcc: dict
    Dizionario contenente per ciascuna diagnosi le probabilità che l'utente abbia una
    specifica diagnosi
Returns
-------
pred: dict
    Dizionario le cui chiavi sono i codici delle diagnosi e i valori sono le
    probabilità pesate di ogni diagnosi, sotto forma di percentuale
```

3.2 database Namespace Reference

Functions

- def mySQLConnect (hostname, username, pw)
- def createDatabase (conn, dbName)
- def useDatabase (conn, dbName)
- def checkIfDbExists (conn, db_name)
- def createTablefromFile (filename, tablename, conn)
- def getValue (conn, table, col, value)
- def ifValueExists (conn, table, col, value)
- def selectJoinID (conn, data, table1, table2, v1, v2)
- def getNameFromID (conn, table, id)
- def getValueCategory (conn, table, value)
- def getValuesCategory (conn, table, list_values)
- def closeConnection (conn)

3.2.1 Function Documentation

3.2.1.1 checkIfDbExists()

```
def database.checkIfDbExists (
              conn,
              db_name )

Funzione che verifica l'esistenza del database

Parameters
----------
conn: connection
    Connessione al database
dbName: str
    Nome del database

Returns
-------
boolean
    True se il database esiste, false altrimenti
```

3.2.1.2 closeConnection()

```
def database.closeConnection (
              conn )
```

```
Funzione che chiude la connessione al database

Parameters
----------
conn: connection
    Connessione al database
```

3.2.1.3 createDatabase()

```
def database.createDatabase (
              conn,
              dbName )
```

```
Funzione che crea il database, se non esiste, e automaticamente selezionato

Parameters
----------
conn: connection
    Connessione al database
dbName: str
    Nome del database
```

3.2.1.4 createTablefromFile()

```
def database.createTablefromFile (
              filename,
              tablename,
              conn )
```

```
Funzione che crea una tabella e la popola con i valori all'interno
del file csv in input

Parameters
----------
filename: str
    Nome/percorso del file csv da aprire
tablename: str
    Nome della tabella da creare
conn: connection
    Connessione al database
```

3.2.1.5 getNameFromID()

```
def database.getNameFromID (
              conn,
              table,
              id )
```

```
Funzione che restituisce il nome della tupla all'interno di una specifica tabella, con
uno specifico id

Parameters
----------
conn: connection
    Connesisone al database
table: str
    Nome della tabella
id: str
    Id della tupla
Returns
-------
cursor: str
    Nome della tupla
```

3.2.1.6 getValue()

```
def database.getValue (
              conn,
              table,
              col,
              value )
```

```
Funzione che restituisce l'id di un valore all'interno di una tabella, in un preciso campo

Parameters
----------
conn: connection
    Connessione al database
table: str
    Nome della tabella
col: str
    Nome del campo
value: str
    Valore da verificare

Returns
-------
result: str
    Stringa contenente l'id del valore
```

3.2.1.7 getValueCategory()

```
def database.getValueCategory (
              conn,
              table,
              value )
```

```
Funzione che restituisce una stringa contenenta le categoria del valore di una specifica tabella

Parameters
----------
conn: connection
    Connesisone al database
table: str
    Nome della tabella
values: str
    Nome della tupla

Returns
-------
cursor: str
    Categoria del campo della rispettiva tabella
```

3.2.1.8 getValuesCategory()

```
def database.getValuesCategory (
              conn,
              table,
              list_values )
```

```
Funzione che restituisce una lista contenente le categorie dei valori di una specifica tabella

Parameters
----------
conn: connection
    Connesisone al database
table: str
    Nome della tabella
list_values: list
    Lista contenenti i nomi delle tuple

Returns
-------
cat: list
    Lista contenenti le categorie dei campi della rispettiva tabella
```

3.2.1.9 ifValueExists()

```
def database.ifValueExists (
              conn,
              table,
              col,
              value )
```

```
Funzione che verifica l'esistenza di un valore all'interno di una tabella, in un preciso campo

Parameters
----------
conn: connection
    Connessione al database
table: str
    Nome della tabella
col: str
    Nome del campo
value: str
    Valore da verificare
Returns
-------
boolean
    True se il valore esiste, false altrimenti
```

3.2.1.10 mySQLConnect()

```
def database.mySQLConnect (
              hostname,
              username,
              pw )
```

```
Funzione che crea la connessione a MySQL

Parameters
----------
hostname: str
    Nome dell'host del db server
username: str
    Username di accesso
password: str
    Password di accesso
Returns
-------
conn: connection
    Connessione a MySQL
```

3.2.1.11 selectJoinID()

```
def database.selectJoinID (
              conn,
              data,
              table1,
              table2,
              v1,
              v2 )
```

```
Funzione che effettua un right join tra due tabelle, restituendo una lista contenente i valori in comune
tra la prima e la seconda tabella, data una lista di valori in input

Parameters
----------
conn: connection
```

```
    Connessione al database
data: list
    Lista dei valori
table1: str
    Nome della prima tabella
table2: str
    Valore della seconda tabella
v1: str
    Campo criterio da verificare
v2: str
    Campo da selezionare
Returns
-------
id_1: list
    Lista contenente i valori della seconda tabella in comune con quelli della prima
```

3.2.1.12 useDatabase()

```
def database.useDatabase (
              conn,
              dbName )
```

```
Funzione che seleziona il database da utilizzare

Parameters
----------
conn: connection
    Connessione al database
dbName: str
    Nome del database
```

3.3 main Namespace Reference

Functions

- def inputSymptoms (conn)
- def mergeList (listOfLists)
- def getPercList (list)
- def getDictOccurences (l)
- def getPercOcc (dictOcc, listLenght)
- def tabulateData (conn, predDiag)
- def main ()

3.3.1 Function Documentation

3.3.1.1 getDictOccurences()

```
def main.getDictOccurences (
              l )
```

```
Funzione che restituisce il numero di occorrenze degli elementi di una lista

Parameters
----------
l: list
    Lista contenente gli elementi

Returns
-------
occDict: dict
     Dizionario le cui chiavi sono gli elementi della lista, i valori sono il numero di occorenze
```

3.3.1.2 getPercList()

```
def main.getPercList (
              list )
```

```
Funzione che calcola la percentuale delle occorenze in una lista,
ottenuta dividendo il numero di occorenze per la lunghezza della lista

Parameters
----------
list: list
    Lista contenente gli elementi

Returns
-------
res: defaultdict
    Dizionario le cui chiavi sono gli elementi della lista, i valori sono le percentuali
```

3.3.1.3 getPercOcc()

```
def main.getPercOcc (
              dictOcc,
              listLenght )
```

```
Funzione che calcola la percentuale delle occorenze in un dizionario,
data la lunghezza della lista

Parameters
----------
dictOcc: dict
    Dizionario le cui chiavi sono gli elementi della lista, i valori sono il numero di occorenze
listLenght: int
    Lunghezza della lista di partenza

Returns
-------
dictOcc: dict
     Dizionario le cui chiavi sono gli elementi della lista, i valori sono le percentuali
```

3.3.1.4 inputSymptoms()

```
def main.inputSymptoms (
              conn )

Funzione per inserire i sintomi dell'utente

Parameters
----------
conn: connection
    Connessione al database. Usato per verificare che l'input sia corretto

Returns
-------
input_list: list
    Lista dei sintomi inseriti dall'utente
```

3.3.1.5 main()

```
def main.main ( )
```

3.3.1.6 mergeList()

```
def main.mergeList (
              listOfLists )

Funzione per unire piú liste in una sola

Parameters
----------
listOfLists: list
    Lista di liste

Returns
-------
mergedList: list
    Unione delle liste in input
```

3.3.1.7 tabulateData()

```
def main.tabulateData (
              conn,
              predDiag )

Funzione che organizza i dati in forma tabulare

Parameters
----------
conn: connection
    Connessione al database
predDiag:
    Dizionario le cui chiavi sono le diagnosi, i valori sono le probabilità (in percentuale)
Returns
-------
df: DataFrame
     Dati organizzati in forma tabulare, le cui colonne sono il codice della diagnosi,
     il rispettivo nome e percentuale
```

3.4 xml_parser Namespace Reference

Functions

- def getDiags (file_in, file_out)
- def getDiagInfo (xml, code)

3.4.1 Function Documentation

3.4.1.1 getDiagInfo()

```
def xml_parser.getDiagInfo (
              xml,
              code )
```

```
Funzione che stampa a video tutte le informazioni su una specifica diagnosi
e le sue sottodiagnosi, con i rispettivi codici

Parameters
----------
xml: str
    Nome/percorso del file xml
code: str
    codice/id della diagnosi
```

3.4.1.2 getDiags()

```
def xml_parser.getDiags (
              file_in,
              file_out )
```

```
Funzione che legge il file xml in input e salva il codice e il nome delle diagnosi in formato csv

Parameters
----------
file_in: str
    Nome/percorso del file in input
file_out: str
    Nome/percorso del file in output
```

Chapter 4

File Documentation

4.1 BBN.py File Reference

Namespaces

- namespace BBN

Functions

- def BBN.getDiagProb (symCatList)
- def BBN.getWeightProb (list1, list2)
- def BBN.getFinalProbCat (list)
- def BBN.predictDiag (conn, probCat, percOcc)

4.2 database.py File Reference

Namespaces

- namespace database

Functions

- def database.mySQLConnect (hostname, username, pw)
- def database.createDatabase (conn, dbName)
- def database.useDatabase (conn, dbName)
- def database.checkIfDbExists (conn, db_name)
- def database.createTablefromFile (filename, tablename, conn)
- def database.getValue (conn, table, col, value)
- def database.ifValueExists (conn, table, col, value)
- def database.selectJoinID (conn, data, table1, table2, v1, v2)
- def database.getNameFromID (conn, table, id)
- def database.getValueCategory (conn, table, value)
- def database.getValuesCategory (conn, table, list_values)
- def database.closeConnection (conn)

4.3 main.py File Reference

Namespaces

- namespace main

Functions

- def main.inputSymptoms (conn)
- def main.mergeList (listOfLists)
- def main.getPercList (list)
- def main.getDictOccurences (l)
- def main.getPercOcc (dictOcc, listLenght)
- def main.tabulateData (conn, predDiag)
- def main.main ()

4.4 xml_parser.py File Reference

Namespaces

- namespace xml_parser

Functions

- def xml_parser.getDiags (file_in, file_out)
- def xml_parser.getDiagInfo (xml, code)

Index