Documentazione del progetto: 118.py

118.py è un progetto realizzato per il corso “Ingegneria della Conoscenza” dell’Università di Bari “Aldo Moro”. Il progetto è stato realizzato da:

- Volpicella Savino Davide [MAT. 682552] s.volpicella3@studenti.uniba.it
- Valentino Federico [MAT. 721270] f.valentino7@studenti.uniba.it

Sommario

1. INTRODUZIONE

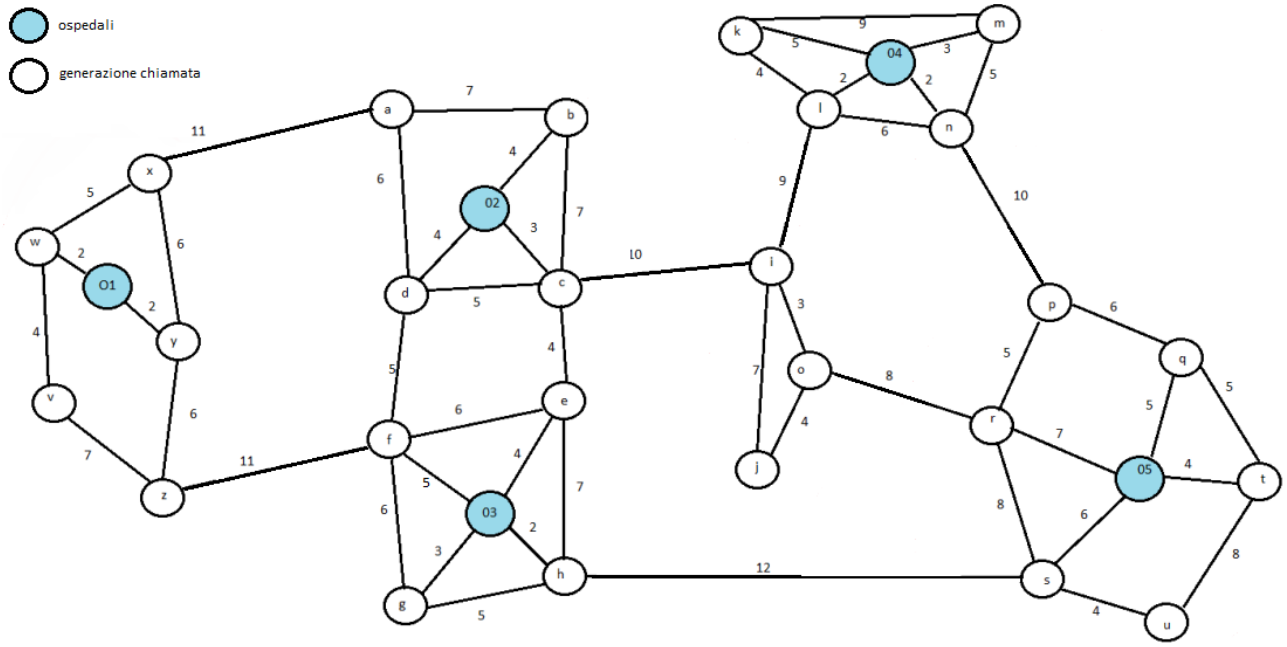
118.py è un programma che simula il funzionamento del servizio sanitario di urgenza ed emergenza medica. L’applicazione consente di generare una chiamata di richiesta di soccorso e in base alle condizioni del paziente, il programma individua l’ospedale che può intervenire nei tempi richiesti e con le giuste risorse.

Ad ogni chiamata generata viene associato un codice: rosso, giallo verde e bianco. Di seguito si riportano le indicazioni dei relativi codici:

- Codice Rosso: indica la massima urgenza. Il paziente ha una o più funzioni vitali compromesse (incosciente o in arresto respiratorio o cardiocircolatorio);
- Codice Giallo: in caso la condizione del soggetto sia a rischio e il paziente non sia stabile (forte dispnea ovvero difficoltà respiratoria, emorragie, ustioni di secondo grado non troppo estese o sospette lesioni ad organi interni);
- Codice Verde: è il codice di priorità minore e si usa quando non siano compromesse le funzioni vitali del paziente (piccole lesioni, dolori o patologie in cui il paziente è comunque stabile);

- Codice Bianco: indica l'assenza di urgenza e viene usato nei casi in cui il paziente presenta lievi patologie, che dovrebbero essere viste dal medico curante. Solo molto raramente viene effettuato dalle ambulanze del 118.

Il funzionamento del programma si poggia su un grafo, dove i nodi rappresentano sia i luoghi da dove una chiamata può essere generata e sia gli ospedali.



Ogni ospedale possiede delle risorse in termini di personale (medici, infermieri e soccorritori) e ambulanze. Esistono diverse tipologie di ambulanze e, in base alle condizioni del paziente da soccorrere, viene scelta quale utilizzare. Di seguito si riporta la descrizione delle tipologie delle ambulanze:

- Tipo A: adibita e attrezzata al trasporto di pazienti non gravi;
- Tipo B: è destinata al trasporto e al trattamento di pazienti gravi;
- Tipo C: è un'unità mobile di terapia intensiva, per il trasporto, monitoraggio e trattamento di pazienti gravi;
- Tipo SSP: non è un'ambulanza. È un mezzo di trasporto utilizzato dai soccorritori per pazienti che presentano patologie e richiedono l'assistenza a domicilio.

In base al codice generato in seguito ad una chiamata di richiesta di soccorso, si verifica quale ospedale ha le risorse disponibili per poter intervenire ed il tempo richiesto dall'ambulanza per poter arrivare.
Si riporta una tabella che indica le risorse necessarie per ogni tipo di codice generato:

	n. Medici	n. Infermieri	n. Soccorritori	Tipo Ambulanza
ROSSO	1	2	1	C
GIALLO	1	1	1	B
VERDE	0	1	1	A
BIANCO	0	0	1	SSP

2. REQUISITI FUNZIONALI

Il software è stato sviluppato in Python, utilizzando PyCharm come IDE.
Prima di poter testare il programma, è necessario:

- Installare swi-prolog (link al download: SWI-Prolog downloads)
- Importare le seguenti librerie:
    - sklearn (utilizzata per inizializzare e testare gli algoritmi di classificazione)
    - pandas (utilizzata per importare e modellare il dataset)
    - os (serve per utilizzare funzioni di sistema utili alla portabilità del programma)
    - random (serve per generare valori casuali)
    - nltk (utilizzata nella similarità del coseno per la rimozione di stopwords)
    - pickle (permette di serializzare e deserializzare i documenti)
    - warnings (evitare warnings generate da funzioni esterne)
    - prolog (permette di utilizzare prolog)
    - openpyxl (consente di lavorare con file xl)

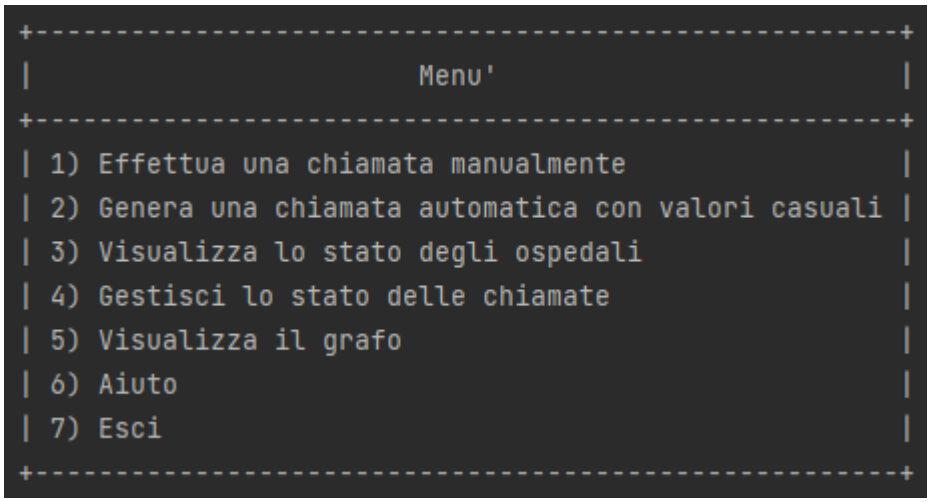
Per avviare il programma bisogna eseguire il file main.py

3. FUNZIONALITA' DEL PROGRAMMA

3.1 Avvio del programma

All'avvio del programma viene spiegato all'utente il funzionamento del programma e gli si indica come poter interagire con il menu principale. Il menu principale presenta diverse voci, ciascuna delle quali implementa una funzionalità diversa.

```
+-------------------------------------------------------------------------------------------------------+
|                                              118.py                                                   |
+-------------------------------------------------------------------------------------------------------+
| 118.py è un programma che simula il funzionamento del servizio sanitario di urgenza ed emergenza medica. |
| Il programma consente di generare delle chiamate di emergenza in due modi:                            |
|                                                                                                       |
| 1) tramite input attraverso la tastiera;                                                              |
| 2) auto-generata in modo casuale.                                                                     |
|                                                                                                       |
| In base alla richiesta di soccorso, si individuerà qual'è l'ospedale che puo' intervenire nei tempi   |
| richiesti e con le giuste risorse.                                                                    |
| Per interagire col programma digita i numeri affiancati alle diverse voci del menù,                   |
| ciascuna delle quali corrisponde ad una funzionalità diversa.                                         |
+-------------------------------------------------------------------------------------------------------+
```



3.2 Generazione di una chiamata

All'utente viene data la possibilità di generare delle chiamate di emergenza in due modi: manualmente o casualmente. La generazione manuale prevede l'utilizzo della tastiera, mentre quella casuale assegna dei valori casuali alle condizioni del paziente.

Nella generazione manuale, all'utente vengono poste delle domande per quanto riguarda la salute del paziente. Le domande sono le seguenti:

- Descrivi con poche parole i sintomi del paziente;
- Il paziente è cosciente?
- Il paziente presenta lesioni?
- Il paziente respira?
- Il pazienta presenta dolore toracico?
- Il paziente presenta un'emorragia?
- Il paziente presenta ustioni?

All'utente, per la maggior parte delle domande, viene richiesto di rispondere inserendo valori compresi in un certo range numerico, ad esempio 0 per rispondere "sì" e 1 per rispondere "no".
Per quanto riguarda la prima domanda, invece, viene richiesto all'utente di inserire una frase che descriva le condizioni del paziente. Da tale domanda si andrà a cercare di capire quale tipo di patologia presenta il paziente, ciò viene fatto utilizzando la similarità del coseno.

Nella generazione casuale, invece, le risposta alle domande elencate precedentemente vengono autogenerate, utilizzando dei valori casuali.

3.2.1 Similarità del coseno

Per poter estrapolare la patologia del paziente si utilizzano le parole chiave inserite dall'utente e si effettua la similarità del coseno tra i termini inseriti dall'utente e i termini presenti nei documenti, presenti nella cartella del progetto nominata "Patologie". In questa cartella sono presenti diversi documenti quante sono le patologie più comuni per cui viene richiesto l'intervento del 118.

Ogni documento rappresenta una patologia diversa e in questi vengono riportati i sintomi prevalenti della corrispettiva patologia.

Nello specifico, quando l'utente indica una patologia, si effettuano i passaggi seguenti:

- L'input inserito dall'utente viene elaborato, rimuovendo da esso le parole prive di significato semantico, come ad esempio congiunzioni e punteggiatura;
- Viene calcolata la similarità del coseno tra i documenti ed i termini estrapolati dall'input dell'utente;
- Il documento che ha una similarità più alta e se tale valore supera una soglia minima indicherà la patologia del paziente

3.2.2 Classificatore e dataset

Una volta generata la chiamata, questa può essere classificata come codice: Rosso, Giallo, Verde, Bianco. Dunque, nel caso di studio la classificazione viene utilizzata con lo scopo di predire, tramite addestramento su un dataset, il codice associato ad una relativa chiamata.
Il dataset utilizzato presenta esempi di classificazione di chiamate di emergenza e in ogni riga sono presenti 8 numeri dove i primi 7 indicano le condizioni del paziente, mentre l'ultimo indica il codice di emergenza associato.

Di seguito si riporta una tabella dove vengono indicati i domini delle risposte per ogni domanda effettuata durante la generazione di una chiamata:

Domande	**Risposte**
Patologia	0: non identificata 1: intossicazione 2: etilista 3: cardiocircolatoria 4: traumatica 5: respiratoria 6: psichiatrica 7: neoplastica
Il paziente è cosciente?	0: si 1:no
Il paziente presenta lesioni?	0: no 1: lievi 2: medie 3: gravi
Il paziente respira?	0: si 1: respiro irregolare 2: no

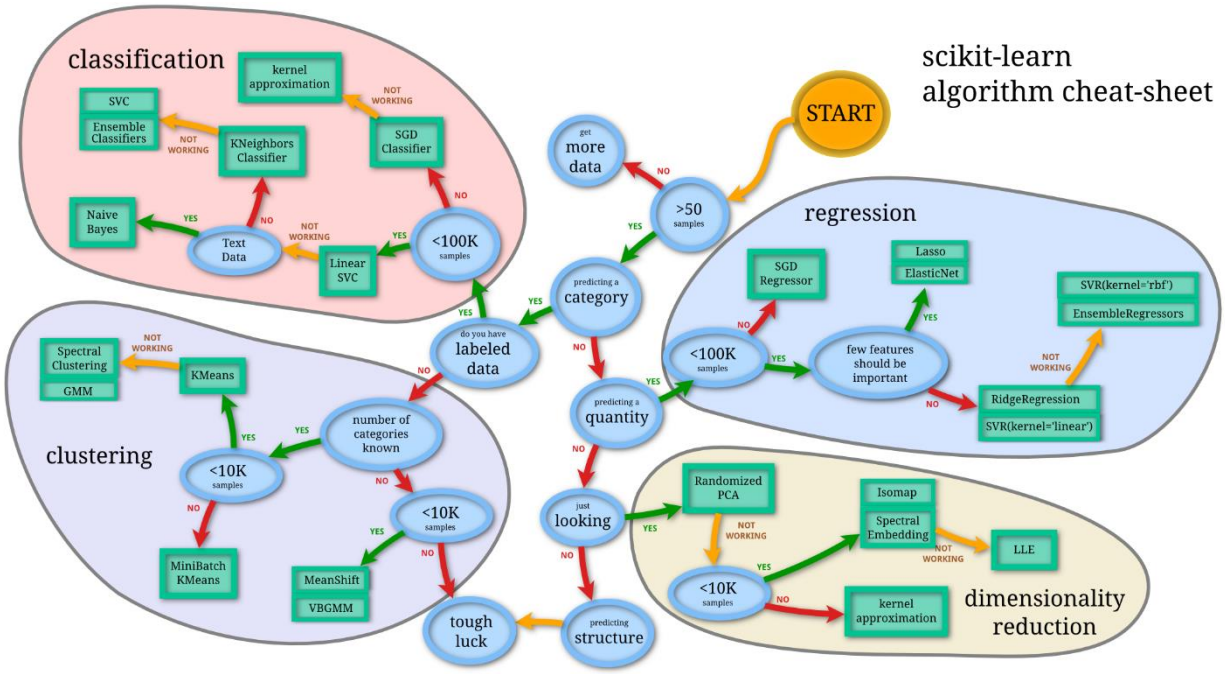
Il paziente presenta dolore toracico?	0: no 1: si
Il paziente presenta un'emorragia?	0: no 1: si
Il paziente presenta ustioni?	0: no 1: lievi 2: medie 3: gravi

Nel machine learning esiste il teorema del *"No free Lunch"* che afferma che non esiste un algoritmo che vada bene per qualsiasi problema e, di conseguenza, sono necessarie varie prove per trovare il modello di predizione più accurato, valutando le performance di ciascuna alternativa, al fine di trovare la più adatta.

Nel nostro caso, si è deciso di seguire la guida trovata sulla piattaforma "Scikit-Learn" e in base alle caratteristiche del nostro dataset, si è scelto di prendere in considerazione i seguenti classificatori:

- Random Forest;
- Bagging Classifier;
- Space Vector Classifier;
- KNN.

Di seguito si riporta l'immagine che rappresenta le linee guida considerate.



Per poter decidere quale classificatore utilizzare, sono state considerate le seguenti metriche:

- Precisione;
- Richiamo;
- Accuratezza.

Si riportano le considerazioni emerse durante la valutazione dei classificatori considerati:

- KNN: Dato che il dataset utilizzato contiene elementi di natura diversa, dove però le corrispettive caratteristiche sono molto simili (ad esempio un codice rosso “poco grave” per la sua categoria e un codice giallo “molto grave” per la sua categoria), il classificatore KNN potrebbe sbagliare un numero significativo di predizioni, andando a selezionare la categoria del caso sbagliato “più vicino” nello spazio N dimensionale. Questa ipotesi è stata verificata in quanto tale classificatore ha riportato valori bassi per precisione e richiamo.

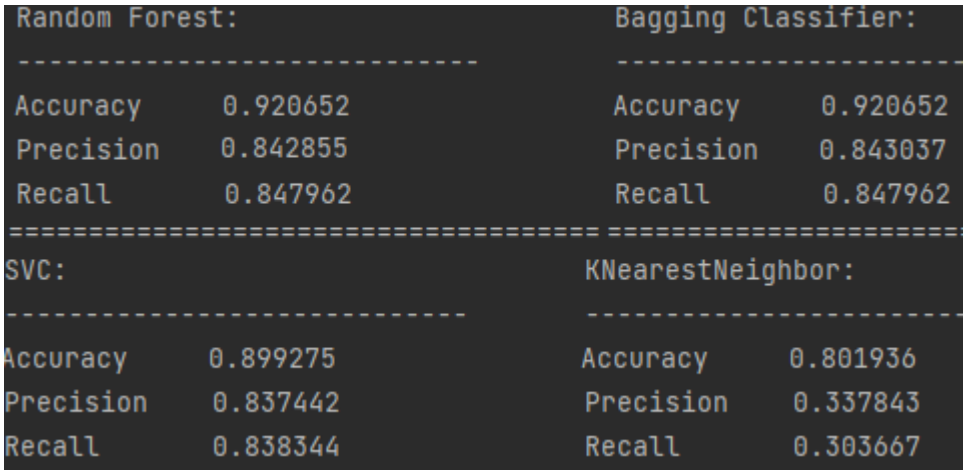
- SVM: Per testare tale classificatore si sono eseguite più verifiche con diverse funzioni in grado di andare ad individuare gli iperpiani che suddividono gli esempi nelle varie classi. La funzioni testate sono state le seguenti: Linear, Poly, Sigmoid, Radial Basis Function. Confrontando i diversi valori delle metriche utilizzate, si è scoperto che la Radial Basis Function è la più promettente.

- Bagging Classifier: Questo classificatore permette di ridurre l’overfitting. Dato che il nostro dataset possiede esempi spesso uguali o molto simili, il Bagging Classifier ha mostrato dati notevoli.

- Random Forest Classifier: Ha lo stesso vantaggio del Bagging Classifier; tuttavia, questo classificatore si dimostra molto lento nel creare previsioni.

Per testare e valutare il dataset, e per evitare overfitting, si è scelto di valutare e testare i vari modelli utilizzando la tecnica cross-validation mediante l’approccio di base, chiamato k-fold CV.
Per quanto riguarda la scelta del parametro k, si è scelto di utilizzare un valore pari a 10, in quanto permette di terminare il processo in tempi accettabili.

Di seguito si riportano i valori delle metriche dei classificatori considerati:



Si è constatato che gli algoritmi migliori per il nostro caso sono: il Bagging Classifier e il Random Forest Classifier, vista la loro natura di evitare l'overfitting.

Data la differenza minimale tra le metriche dei due algoritmi, abbiamo scelto di utilizzare il Bagging Classifier, basandoci sulle prestazioni, dato che il Random Forest Classifier è *veloce* nell'apprendimento ma lento nella *predizione*.

3.3 Elaborazione di una chiamata

In base al codice della chiamata di emergenza, si individuano quali ospedali sono in grado di rispondere alla chiamata.

3.3.1 Base di conoscenza

Ogni ospedale presenta delle risorse in termini di personale e numero di mezzi di trasporto disponibili.
Le informazioni relative alle risorse sono state inserite all'interno di una base di conoscenza, mediante l'utilizzo di Prolog e della libreria pyswip.

Per ogni ospedale, all'interno della base di conoscenza, vengono riportati:

- Il numero di medici
- Il numero di infermieri
- Il numero di soccorritori
- Il numero di ambulanze disponibili
- I reparti presenti nell'ospedale

Ogni codice di emergenza richiede delle risorse, di seguito si riportano le risorse necessarie per ogni tipologia di richiesta di soccorso:

CODICE	**RISORSE**
Rosso	Tempo per intervenire: 20 min Ambulanza di tipo C 1 medico 2 infermieri 1 soccorritore
Giallo	Tempo per intervenire: 40 min Ambulanza di tipo B 1 medico 1 infermiere 1 soccorritore
Verde	Tempo per intervenire: 60 min Ambulanza di tipo A 1 soccorritore 1 infermiere

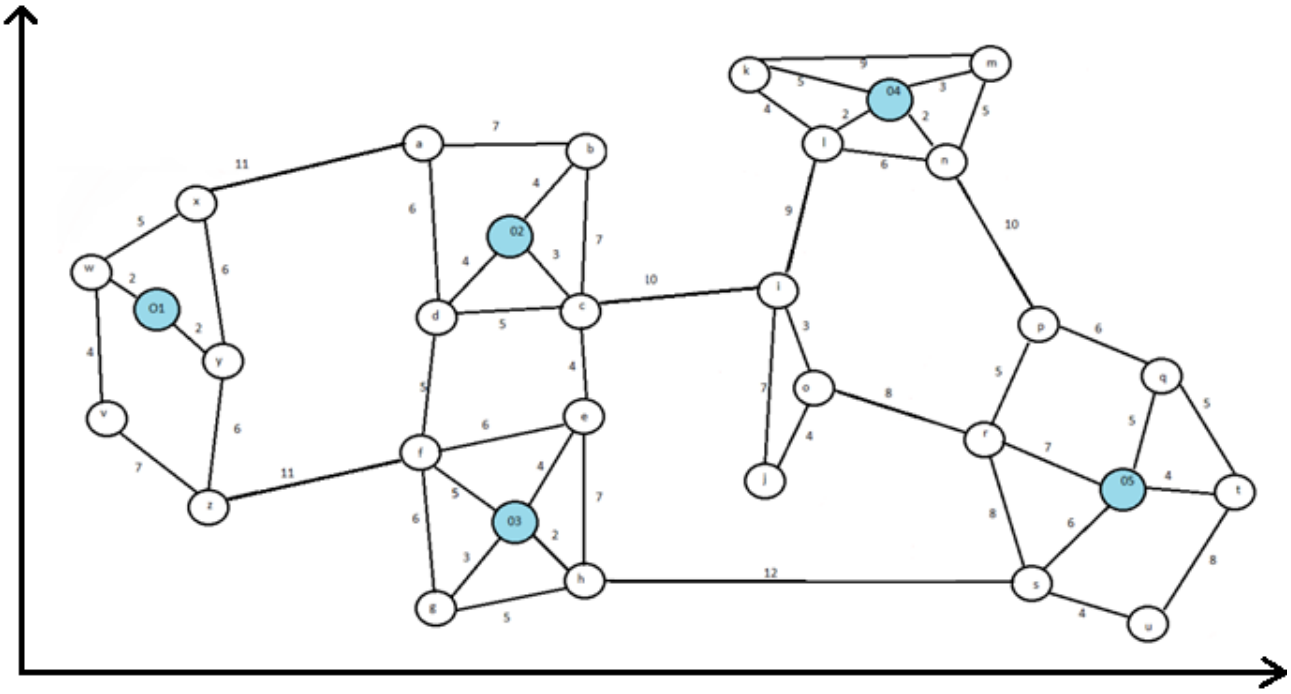
Bianco	Tempo per intervenire: 80 min Mezzo di trasporto semplice sanitario 1 soccorritore

Viene dunque interrogata la base di conoscenza attraverso delle query, con l'obiettivo di estrapolare gli ospedali che sono in grado di rispondere alla chiamata.

Una volta che viene individuato l'ospedale in grado di prendersi in carico il paziente, si aggiornano le risorse disponibili per quell'ospedale andando a modificare i fatti all'interno della base di conoscenza.

3.3.2 A*

Per individuare qual è il percorso che l'ambulanza inviata dall'ospedale deve intraprendere, viene utilizzato l'algoritmo di ricerca A*, utilizzando l'euristica di Manhattan.
L'euristica di Manhattan consiste nel considerare la distanza tra due punti come la somma del valore assoluto delle loro coordinate.
Per poter utilizzare tale euristica abbiamo proiettato il grafo su un piano cartesiano e si sono individuate le coordinate dei diversi nodi.



Dato che il grafo prevede cinque nodi goal possibili, che rappresentano i cinque ospedali, si sono definite cinque euristiche diverse. In ogni euristica si definisce la distanza di Manhattan tra ogni nodo con l'ospedale della corrispettiva euristica. Come esempio, si riporta uno screenshot che rappresenta l'euristica definita sull'ospedale 1:

```
def h_osp1(self, n):
    H = {
        'A': abs(5 - 15) + abs(15 - 22),
        'B': abs(5 - 20) + abs(15 - 22),
        'C': abs(5 - 20) + abs(15 - 16),
        'D': abs(5 - 15) + abs(15 - 15),
        'E': abs(5 - 20) + abs(15 - 12),
        'F': abs(5 - 15) + abs(15 - 9),
        'G': abs(5 - 16) + abs(15 - 4),
        'H': abs(5 - 20) + abs(15 - 5),
        'I': abs(5 - 26) + abs(15 - 16),
        'J': abs(5 - 27) + abs(15 - 8),
        'K': abs(5 - 26) + abs(15 - 25),
        'L': abs(5 - 29) + abs(15 - 23),
        'M': abs(5 - 35) + abs(15 - 26),
        'N': abs(5 - 33) + abs(15 - 23),
        'O': abs(5 - 29) + abs(15 - 12),
        'P': abs(5 - 36) + abs(15 - 15),
        'Q': abs(5 - 40) + abs(15 - 14),
        'R': abs(5 - 35) + abs(15 - 10),
        'S': abs(5 - 36) + abs(15 - 5),
        'T': abs(5 - 42) + abs(15 - 8),
        'U': abs(5 - 40) + abs(15 - 3),
        'V': abs(5 - 3) + abs(15 - 10),
        'W': abs(5 - 2) + abs(15 - 16),
        'X': abs(5 - 6) + abs(15 - 19),
        'Y': abs(5 - 7) + abs(15 - 13),
        'Z': abs(5 - 6) + abs(15 - 6),
        '01': 0,
        '02': abs(5 - 17) + abs(15 - 17),
        '03': abs(5 - 18) + abs(15 - 6),
        '04': abs(5 - 32) + abs(15 - 24),
        '05': abs(5 - 39) + abs(15 - 9),
    }

    return H[n]
```

Nel momento in cui viene generata una chiamata si andranno a calcolare cinque percorsi diversi, i quali avranno come nodo start il punto da dove viene generata la chiamata e avranno come nodi goal i diversi ospedali.

Il tempo necessario per percorrere i cinque percorsi viene memorizzato all'interno della base di conoscenza e, nel momento in cui si effettua la query per verificare quale ospedale possiede le giuste risorse per intervenire, si andrà a considerare anch'esso
Dunque, l'ospedale che presterà soccorso al paziente sarà quello avente le giuste risorse disponibili e che potrà accorrere il paziente nel minor tempo.

3.3.3 Esempio

L'immagine sottostante, riporta un esempio di generazione di chiamata casuale, classificazione della chiamata e individuazione dell'ospedale che è in grado di intervenire.

```
Generazione di una chiamata di soccorso in corso...
Informazioni sul paziente generate:

Patologia: respiratoria
Stato di coscienza del paziente: presente
Lesioni: gravi
Respiro: irregolare
Dolore toracico: assente
Emorragia: presente
Grado di ustioni: primo grado

La chiamata di emergenza e' stata effettuata dal nodo: T
Il codice generato per il paziente e': GIALLO

Interrogazione della base di conoscenza in corso...
Gli ospedali che sono in grado di rispondere alla richiesta di soccorso sono:
ospedale_2
ospedale_5

L'ospedale che riesce ad inviare un'ambulanza nel minor tempo possibile e': ospedale_5
Percorso effettuato: :['T', '05']
```

3.4 Visualizzazione delle informazioni sugli ospedali

Questa funzionalità permette all'utente di verificare quali sono le risorse disponibili per ogni ospedale.

3.5 Gestione dello stato delle chiamate

Ogni qual volta che si genera una chiamata, questa viene registrata all'interno della base di conoscenza.
La selezione della voce "Gestione dello stato delle chiamata" farà apparire un altro menu dal quale l'utente potrà:

- Visualizzare il registro delle chiamate;
- Aggiornare lo stato delle chiamate;
- Ritornare al menu principale.

Per ogni chiamata viene salvato:

- L'id univoco della chiamata;
- Il codice (ROSSO, VERDE, GIALLO, BIANCO);
- Il nodo di partenza;
- L'ospedale che interviene:
- Lo stato.

Lo stato di una chiamata può essere: "IN CORSO" o "COMPLETATA". Il primo valore indica che l'intervento della chiamata non è ancora terminato e le risorse impiegate sono ancora occupate, mentre il secondo valore indica che l'intervento è terminato e che le risorse impiegate sono nuovamente disponibili per una nuova chiamata.
L'utente, selezionando la voce "Aggiornare lo stato delle chiamate", potrà indicare, attraverso l'id della chiamata, quale intervento è stato completato.
Portare lo stato di una chiamata a "COMPLETATA", implicherà una modifica delle risorse disponibili degli ospedali all'interno della base di conoscenza effettuate mediante opportune query.

3.6 Visualizzazione del grafo

Questa funzionalità consente all'utente di visualizzare il grafo all'interno del programma.

3.7 Linee guida per l'utente

Nel menu è presente la voce "aiuto", la quale ha il compito di spiegare all'utente le diverse funzionalità del programma.

3.8 Terminare il programma

L'ultima funzionalità permette all'utente di chiudere il programma. Per evitare chiusure accidentali, si richiede la conferma di terminazione.