Sistema diagnostica diabete

Davide Cellura, Matricola:721817

Email: d.cellura@studenti.uniba.it

Anno Accademico 2023-2024

Repository Github: https://github.com/DavideCellura/Progetto_ICON_2024

1:Introduzione

Il diabete è una malattia cronica caratterizzata da livelli elevati di glucosio (zucchero) nel sangue. Questo squilibrio è dovuto a un'alterazione nella produzione o nell'azione dell'insulina, un ormone prodotto dal pancreas che regola la quantità di glucosio nel sangue. Esistono principalmente due tipi di diabete:

1. **Diabete di tipo 1**: È una malattia autoimmune in cui il sistema immunitario attacca e distrugge le cellule beta del pancreas che producono insulina. Di conseguenza, l'organismo non è in grado di produrre insulina, rendendo necessaria la somministrazione di insulina dall'esterno.
2. **Diabete di tipo 2**: È la forma più comune e si verifica quando il corpo non utilizza efficacemente l'insulina (insulino-resistenza) o non ne produce abbastanza. Questo tipo è spesso associato a fattori di rischio come obesità, inattività fisica e dieta inadeguata.

Il diabete può portare a complicazioni gravi, come malattie cardiovascolari, danni ai reni, neuropatie e problemi alla vista, se non viene adeguatamente gestito.

Utilizzerò due moduli per la diagnostica del diabete:

- Un agente inteligente (rule-based) che tiene conto delle risposte dell’ utente per la diagnostica del diabete.
- L’ utilizzo di algoritmi di apprendimento supervisionato per la diagnostica del diabete.

2: Knowledge base

Per la diagnostica del diabete ho deciso di utilizzare un sistema esperto, per fare ciò ho utilizzato la libreria python “Experta”.

Un sistema esperto è un programma informatico progettato per simulare il processo decisionale di un esperto umano in un determinato campo. Utilizza una base di conoscenze, che raccoglie informazioni ed esperienze specifiche, e un motore di inferenza, che applica regole logiche per risolvere

problemi complessi o fornire consigli, proprio come farebbe un esperto umano.

In sintesi, un sistema esperto è uno strumento che "pensa" come un esperto in un particolare dominio, aiutando gli utenti a prendere decisioni informate.

2.1 Funzionamento Experta

“Experta” si basa su **fatti** e **regole**.

- **Fatti (Facts):** Le unità di informazione su cui vengono applicate le regole. Ogni fatto può essere visto come un insieme di attributi che descrivono una certa situazione.
- **Regole (Rules):** Dichiarazioni condizionali che descrivono un'azione da intraprendere quando certe condizioni sui fatti sono soddisfatte. Ogni regola è composta da:

  - LHS (sinistro): descrive le condizioni base alle quali la regola dovrebbe essere eseguita.

  - RHS (destro): è l’insieme delle azioni da eseguire quando la regola viene eseguita.

Ogni regola affinchè possa essere utilizzata deve essere un metodo di una sottoclasse KnowledgeEngine.

2.2 Gestione dei fatti

Un **fatto** può essere:

- Dichiarato: Aggiunta di un nuovo fatto all’elenco dei fatti.

- Ritrattato: Rimozione di un fatto esistente dall’elenco dei fatti.

- Modificato: Ritiro di un fatto e aggiunta del medesimo fatto modificato.

- Duplicato: Aggiunta di un nuovo fatto all’elenco utilizzando un fatto esistente come modello inserendo alcune modifiche.

2.3 Funzionamento sistema esperto

Durante l'esecuzione del sistema esperto l'utente è sottoposto ad una serie di domande attraverso il quale il sistema dichiara dei facts. Basandosi sulle regole il sistema sarà capace di fare una diagnosi del diabete.

2.4 Dataset

Per realizare il sistema ho utilizzato il seguente dataset:

https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset

Il dataset contiene diverese informazioni come:

- ***gender***: in questa colonna è indicato il sesso dei pazienti (maschio o femmina)
- ***age***: in questa colonna è riportata l'età del paziente
- ***hypertension***: colonna in cui è riportato se il paziente ha avuto problemi di ipertensione (0=no , 1=si)
- ***heart_disease***: colonna in cui è riportato se il paziente ha avuto malattie al cuore (0=no , 1=si)
- ***smoking_history***: questa colonna sta ad indicare se il paziente ha precedenti con il fumare.
- ***bmi***: Il BMI (Indice di Massa Corporea) è un indicatore che misura la relazione tra il peso e l'altezza di una persona. Si calcola dividendo il peso in chilogrammi per il quadrato dell'altezza in metri. Viene usato per stimare se una persona è sottopeso, normopeso, sovrappeso o obesa.
- ***HbA1c_level***: Il livello di HbA1c indica la quantità di emoglobina glicata nel sangue, riflettendo la media della glicemia negli ultimi 2-3 mesi. È usato per monitorare il controllo a lungo termine del diabete.
- ***blood_glucose_level*****:** Il livello di glucosio nel sangue si riferisce alla quantità di glucosio nel flusso sanguigno in un dato momento. Livelli elevati di glucosio nel sangue sono un indicatore chiave del diabete.
- ***diabetes:*** è la variabile target da prevedere, con valori pari a 1 che indica la presenza di diabete, 0 che ne indica l' assenza.

2.5 Ontologia

In informatica, un'ontologia è una rappresentazione formale, condivisa ed esplicita di una concettualizzazione di un dominio di interesse. Il termine ontologia formale è entrato in uso nel campo dell'intelligenza artificiale e della rappresentazione della conoscenza, per descrivere il modo in cui diversi schemi vengono combinati in una struttura dati contenente tutte le entità rilevanti e le loro relazioni in un dominio.

Le ontologie distribuite sulla rete vengono descritte attraverso il linguaggio OWL (che sta per Web Ontology Language) che permette di descrivere il mondo in termini di:

- **Individui**: entità del mondo

- **Classi**: insieme di individui, accomunati da delle caratteristiche comuni

- **Proprietà**: relazioni che associano agli elementi del proprio dominio (individui), un elemento del codominio (valori che può assumere).

In particolare, si dividono in;

- **Datatype property**: se il codominio contiene solo tipi primitivi (interi, stringhe, …)

- **Object property**: quando il codominio contiene altre classi

La strutturazione dell'ontologia è avvenuta tramite il software **protège.**

Con **protège** ho creato la ontologia sul diabete definendo le varie entità corrispondenti a ciascun sintomo con le rispettive proprietà.

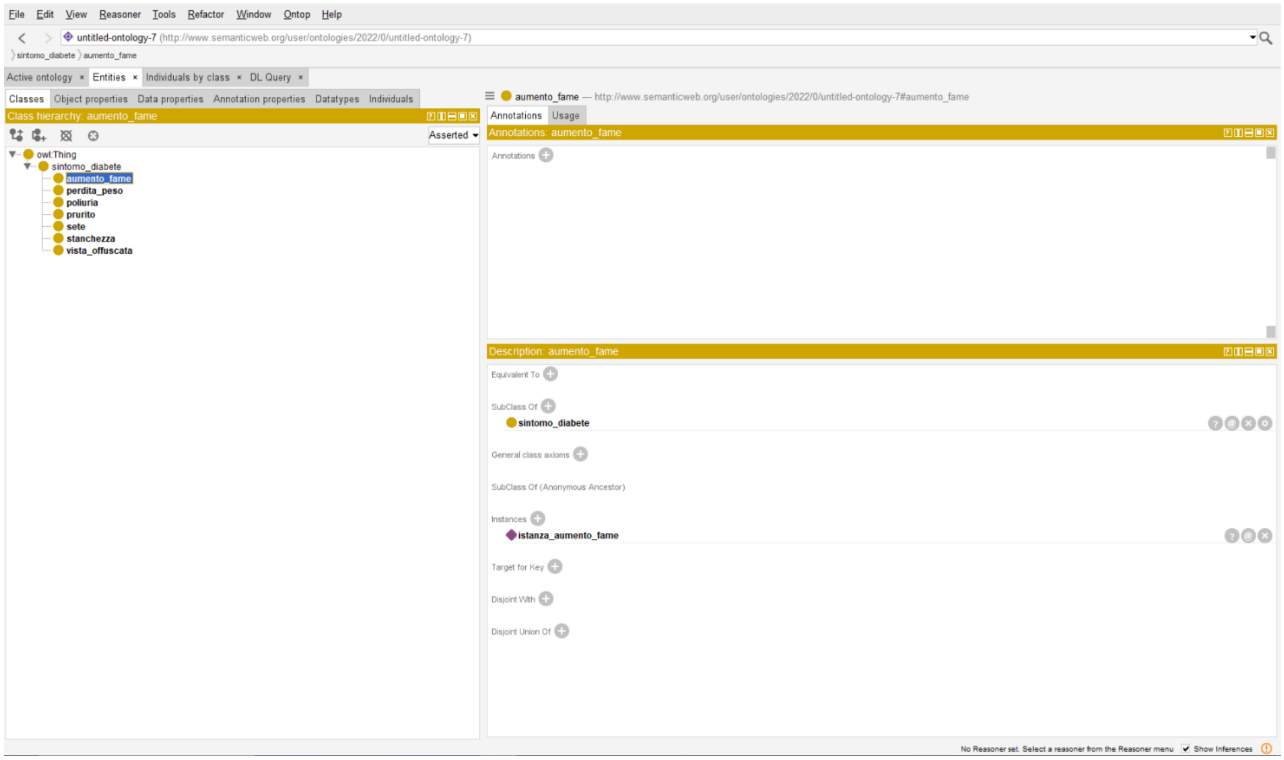


Figura 2.5.1: tool protege

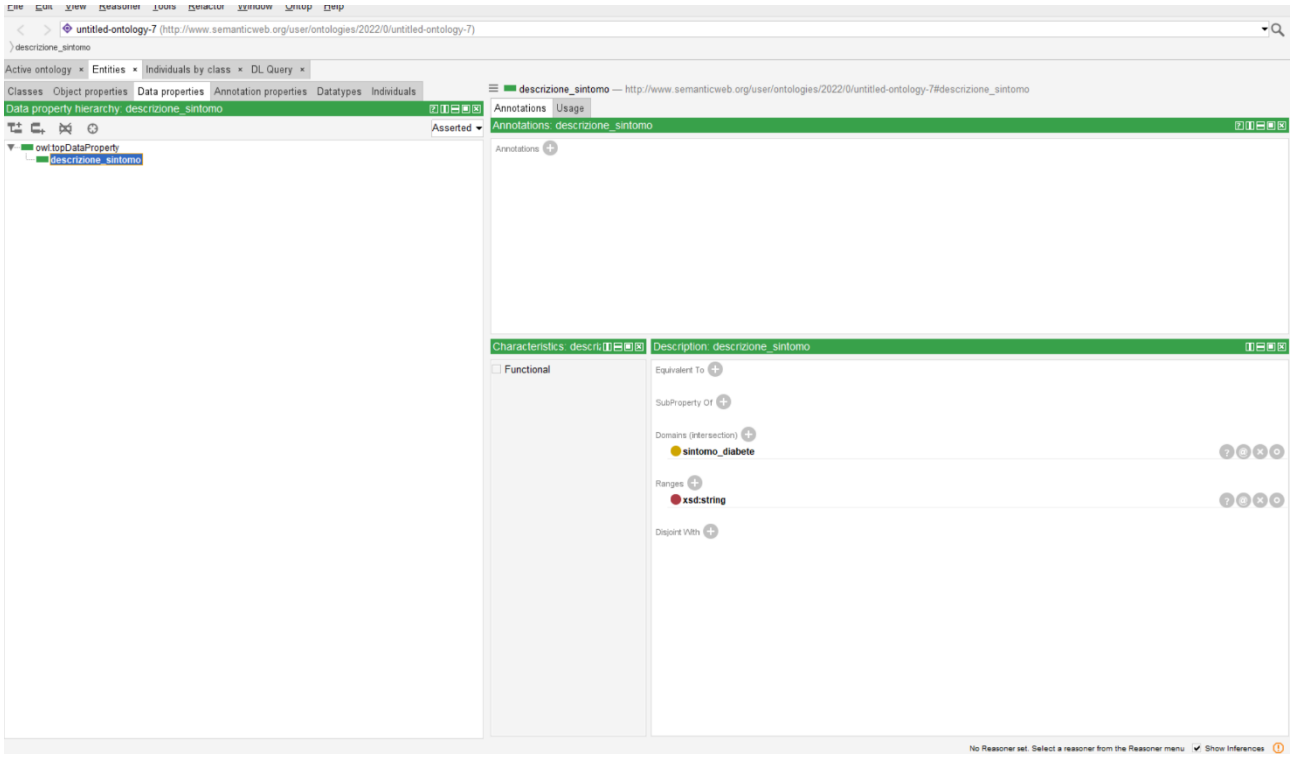


Figura 2.5.2: sezione “data properties” protege

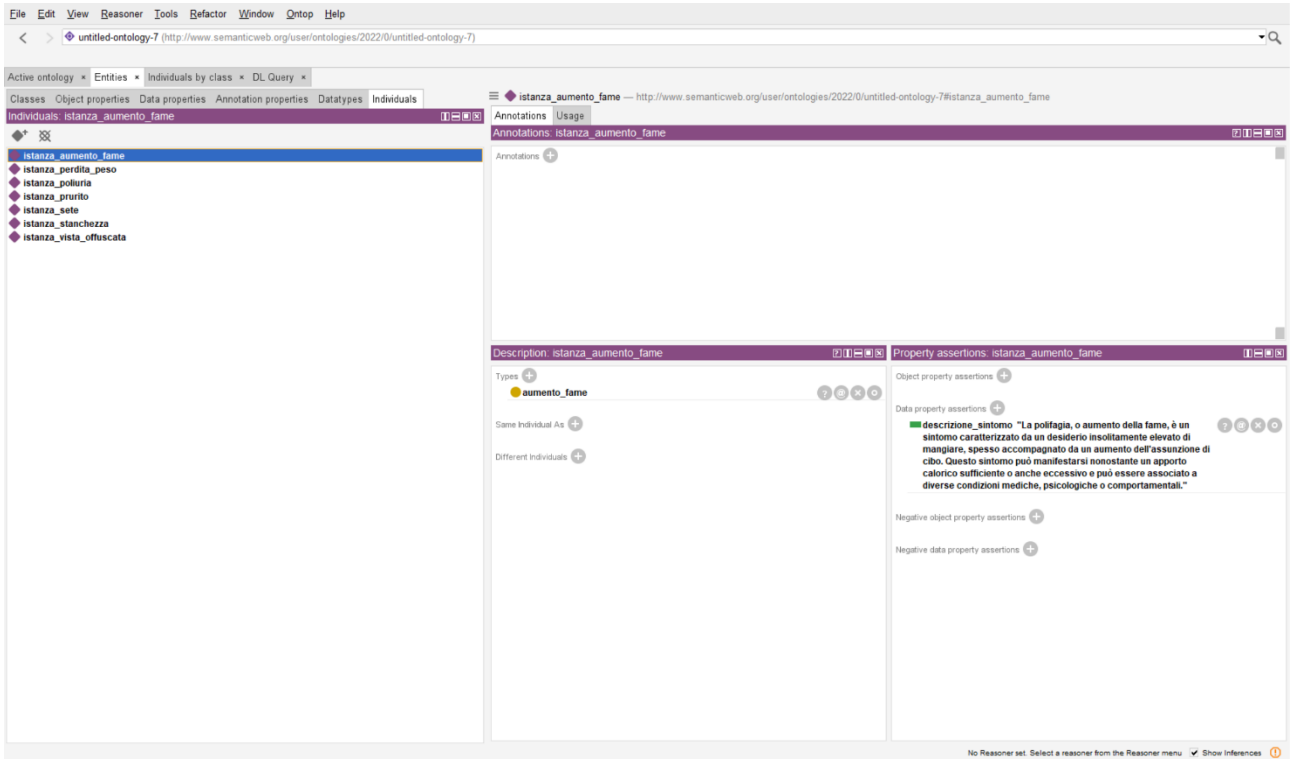


Figura 2.5.3: sezione “individuals” protege

Successivamente ho creato “sintomi_diabete.owl” in modo da gestire l’ontologia tramite la libreria **Owlready2** presente in python.

```
(base) C:\Users\Davide\Desktop\Università Davide\ICON\PROGETTO\Progetto Mio(Diabete)\Codice>python esperto_diabete.py
* Owlready2 * Warning: optimized Cython parser module 'owlready2_optimized' is not available, defaulting to slower Python implementat:
Benvenuto in Esperto Diabete, un sistema esperto per la diagnosi del diabete
----------->MENU<-----------
[1] Mostra i possibili sintomi del diabete
[2] Esegui una diagnosi
[3] Esci
1
Sintomo [1]: Nome: aumento_fame
Sintomo [2]: Nome: perdita_peso
Sintomo [3]: Nome: poliuria
Sintomo [4]: Nome: prurito
Sintomo [5]: Nome: sete
Sintomo [6]: Nome: stanchezza
Sintomo [7]: Nome: vista_offuscata

Seleziona il sintomo di cui vuoi conosere la descrizione, inserisci il numero del sintomo
3
Sintomo: poliuria, descrizione: La poliuria è un sintomo caratterizzato da un aumento anomalo della produzione di urina. Tecnicamente
zioni normali, un adulto urina circa 1-2 litri al giorno, quindi la poliuria rappresenta un aumento significativo di questo volume.



----------->MENU<-----------
[1] Mostra i possibili sintomi del diabete
[2] Esegui una diagnosi
[3] Esci
```

Figura 2.5.4: Lista dei sintomi con la relativa descrizione

2.6 CSP (Constraint Satisfaction Problem)

Molti problemi nell'ambito dell'Intelligenza Artificiale sono classificabili come Problemi di Soddisfacimento di Vincoli (Constraint Satisfaction Problem o CSP); Formalmente, un CSP può essere definito su un insieme finito di variabili (X1,X2, . . . ,Xn) i cui valori appartengono a domini finiti di definizione(D1,D2, . . . ,Dn) e su un insieme di vincoli(C1,C2, . . . ,Cn). Un vincolo su un insieme di variabili è una restrizione dei valori che le variabili possono assumere simultaneamente. Concettualmente, un vincolo può essere visto come un insieme che contiene tutti i valori che le variabili possono assumere contemporaneamente: un vincolo tra k variabili C(Xi1,Xi2, . . . ,Xik), è un sottoinsieme del prodotto cartesiano dei domini delle variabili coinvolte Di1 ,Di2 , . . . ,Dik che specifica quali valori delle variabili sono compatibili con le altre. Questo insieme può essere rappresentato in molti modi, come ad esempio, matrici, equazioni, disuguaglianze o relazioni.

La libreria che ho utilizzato è "constraint" che mi ha permesso di realizzare un semplice CSP (è un caso molto semplice, non si tiene conto di altri utenti) in grado di mostrare la disponibilità dello studio medico convenzionato, nel caso in cui il sistema preveda la prescrizione delle analisi del sangue. Il funzionamento è il seguente:

• Alla base vi è una sottoclasse di Problem (classe già definita in constraint, che modella un CSP).

• Vengono aggiunte variabili con proprio dominio associato in maniera esplicita.

• In base alle risposte dell' utente, il sistema decide se prescrivere o meno delle analisi.

• Se il sistema decide di prescrivere delle analisi, allora qui entra in gioco il CSP.

• Il CSP è relativo agli orari e giorni di apertura del laboratorio delle analisi.

• Per esempio: il laboratorio per l'analisi della glicemia nel sangue, fa le analisi dalle ore 8 alle 14 di lunedì.

• Il sistema quindi indica i possibili orari a cui l' utente può presentarsi per sottoporsi alle analisi.

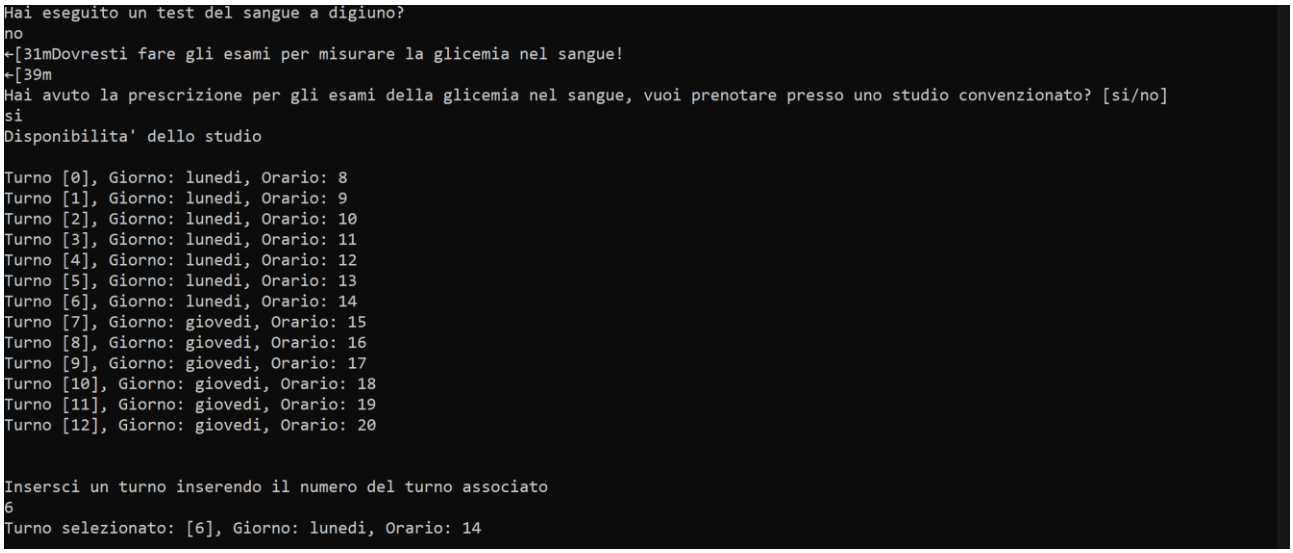


Figura 2.6.1: Esempio di prenotazione degli esami della glicemia nel sangue.

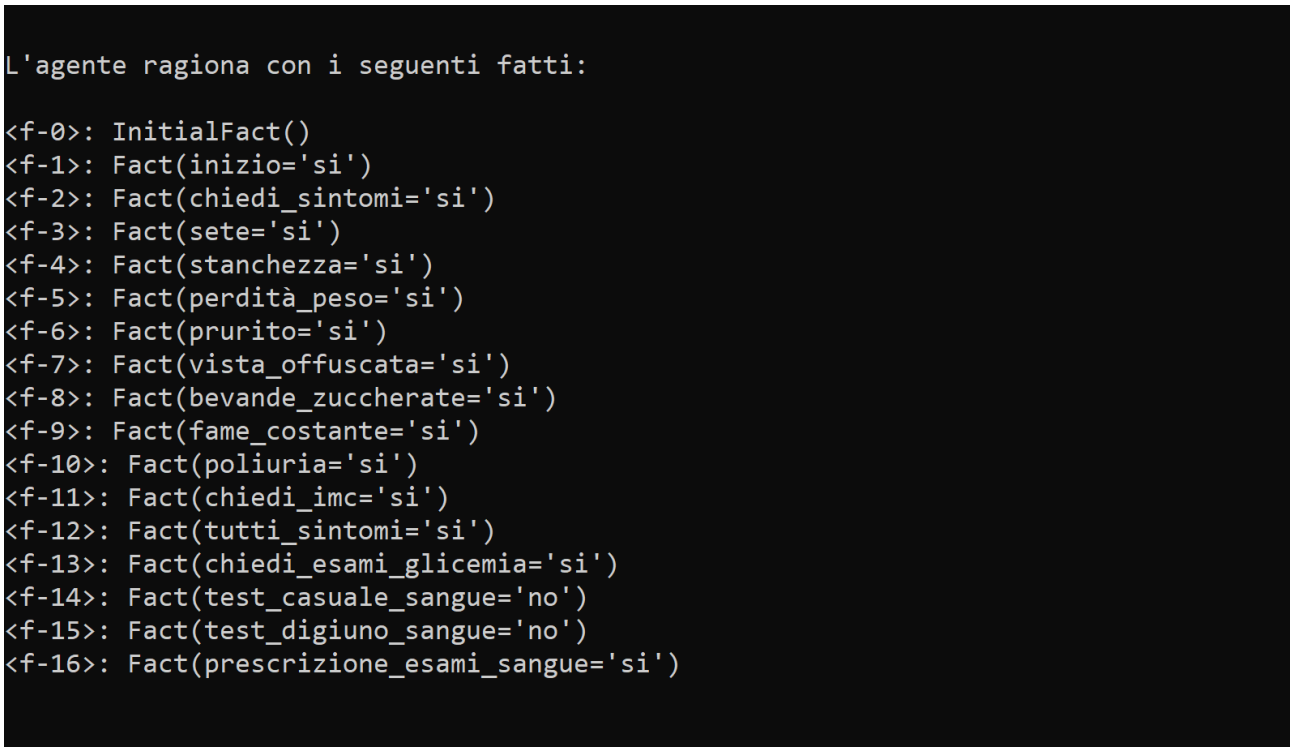


Figura 2.6.2: In base alle risposte fornite dall’utente, l’agente ragiona con i seguenti fatti

3 Apprendimento Supervisionato

In questa sezione di progetto sono stati implementati diversi algoritmi di Apprendimento Supervisionato appartenenti, in modo specifico, alla categoria degli Algoritmi di Classificazione.

Gli algoritmi utilizzati sono:

• Regressione logistica

• Albero di decisione

• K-nearest neighbor

3.1 Implementazione dei modelli di apprendimento supervisionato in sklearn

La libreria scelta per l' implementazione di questi algoritmi è sklearn, che permette tramite le classi e i metodi presenti di implementare, allenare e testare i modelli di apprendimento supervisionato.

1. Nel main (main_modello.py), per ogni modello si istanzia la relativa classe.

2. Per ogni modello viene eseguita la predizione.

3. Successivamente viene mostrata la matrice di confusione.

4. Vengono quindi calcolate le metriche (accuratezza, precision, recall, F1_precision).

5. Il calcolo di queste metriche avviene tramite l' ausilio di metodi messi a disposizione da Sklearn.

Le metrice che vengono utilizzate sono:

1. **Recall**: Misura la capacità del modello di identificare correttamente le istanze positive. Rappresenta la proporzione di veri positivi sul totale delle effettive classi positive.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

2. **Precision**: Misura la qualità delle predizioni positive del modello, ovvero la proporzione di veri positivi sul totale delle predizioni positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

3. **Accuracy**: Rappresenta la proporzione di tutte le predizioni corrette (sia positive che negative) sul totale delle istanze valutate.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

4. **F1-score**: È la media armonica tra precision e recall, utilizzata per bilanciare questi due aspetti in presenza di un dataset con classi sbilanciate.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.2 Regressione logistica

La regressione logistica è un modello statistico usato negli algoritmi di classificazione del machine learning per ottenere la probabilità di appartenenza a una determinata classe. L'algoritmo si basa sull'utilizzo della funzione logistica (sigmoid).Il risultato della funzione sigmoide è un valore compreso tra 0 e 1, che può essere interpretato come la probabilità che l'osservazione appartenga a una classe specifica (solitamente la classe "positiva").

- Se la probabilità calcolata è superiore a una certa soglia (tipicamente 0,5), l'osservazione viene classificata come appartenente alla classe positiva.
- Se la probabilità è inferiore alla soglia, l'osservazione viene classificata come appartenente alla classe negativa.

Nella fase di addestramento l'algoritmo riceve in input un dataset di training composta da N esempi.

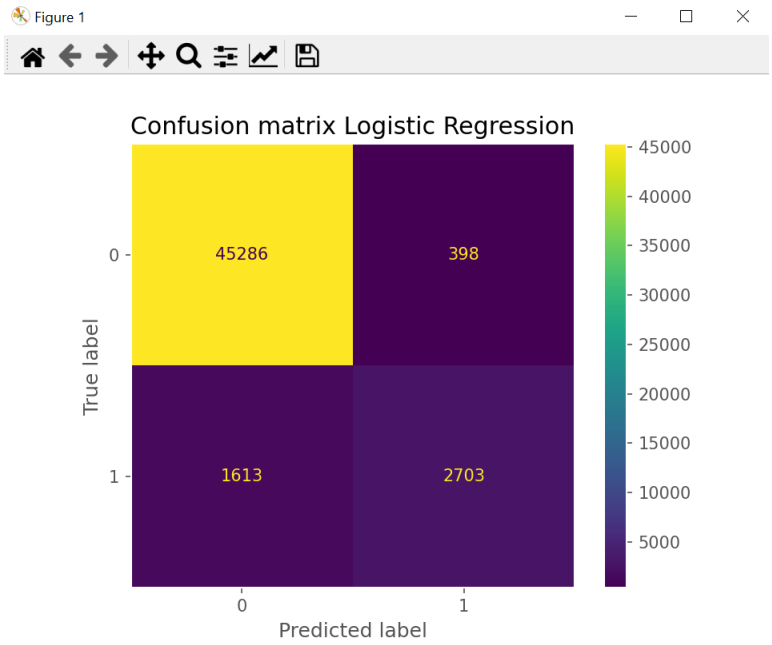
Ogni esempio è composto da m attributi X e da un'etichetta y che indica la corretta classificazione.

Nel nostro caso il dataset viene prima pre-processato creando l'oggetto di classe diabetes_data, tramite questa classe si elimina le feature relativa al sesso e ai precendeti con il fumare, successivamente definisce tramite il metodo get training data(), che restituisce i dati relativi a input features e target features. Nel nostro caso la target feature è una sola, cioè "diabetes". Questa feature indica se un soggetto è malato o meno di diabete.

Qui di seguito vengono mostrati a video i risultati delle metriche calcolate:

```
Logistic Regression metrics
Accurancy : 0.960
Precision : 0.872
Recall : 0.626
F1_precision : 0.729
```

La Matrice di confusione della regressione logistica è la seguente:



in questo caso abbiamo 2703 casi true-positive, ovvero che appartengono alle persone la cui predizione sul diabete ha dato esito positivo. Abbiamo 45286 true negative, quindi il modello ha predetto correttamente che 2703 soggetti non presentano il diabete, mentre il false-positive(398) ed il false-negative(1613) non sono stati predetti correttamente.

Analisi delle metriche al variare del parametro preso in input.

Nel caso della regressione logistica, il parametro che viene cambiato è il numero di iterazioni fatte dall' algoritmo di classificazione.

3.3 Albero di decisione

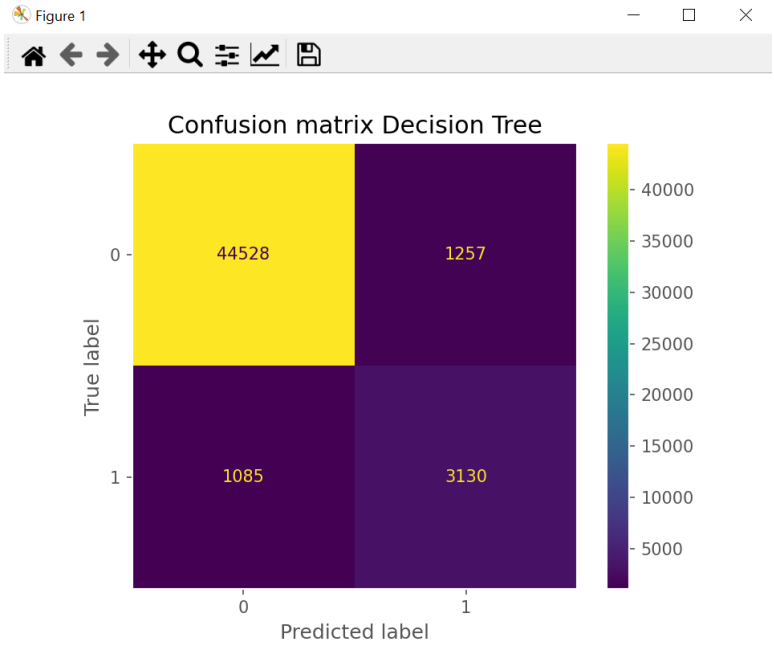
Gli alberi di decisione sono una tecnica di apprendimento automatico (machine learning) che viene utilizzata per la classificazione o la regressione di dati. Si basano su una struttura ad albero in cui ogni nodo rappresenta una caratteristica (o una variabile) del dataset e ogni arco rappresenta una regola di decisione basata su quella caratteristica.

Dal nodo dipendono tanti archi quanti sono i possibili valori che la caratteristica può assumere, fino a raggiungere le foglie che indicano la categoria associata alla decisione.

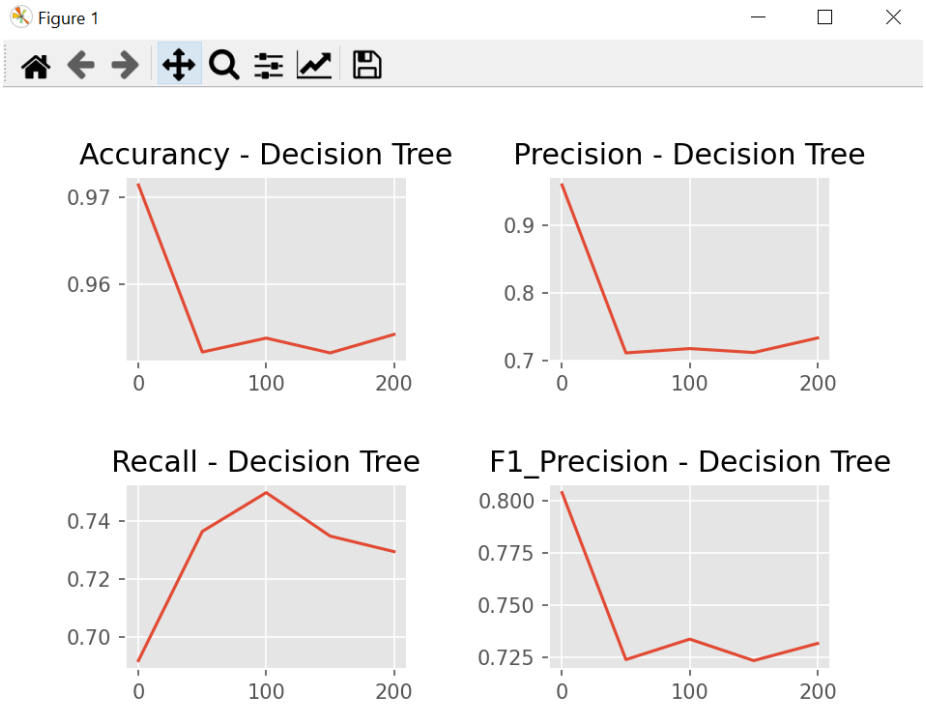
Qui di seguito vengono mostrati a video i risultati delle metriche calcolate:

```
Decision tree metrics
Accurancy : 0.953
Precision : 0.713
Recall : 0.743
F1_precision : 0.728
```

Utilizziamo nuovamente una Matrice di Confusione per analizzare gli errori compiuti da un modello di machine learning.



in questo caso abbiamo 3130 casi true-positive, ovvero che appartengono alle persone la cui predizione sul diabete ha dato esito positivo. Abbiamo 44528 true negative, mentre il false-positive(1257) ed il false-negative(1085) non sono stati predetti correttamente.



Nell’albero di decisione varia la profondità.

Le performance scendono verso la profondità 50 per accuratezza precisione e F1_precision.Mentre per il recall sono basse all'inizio ma alte a profondità maggiore.

3.4 K-nearest neighbor

Il **K-Nearest Neighbors (KNN)** è un algoritmo di machine learning utilizzato per problemi di **classificazione** e **regressione**. Il KNN classifica una nuova osservazione in base alle classi delle k osservazioni più vicine (i "vicini") nel dataset di addestramento.

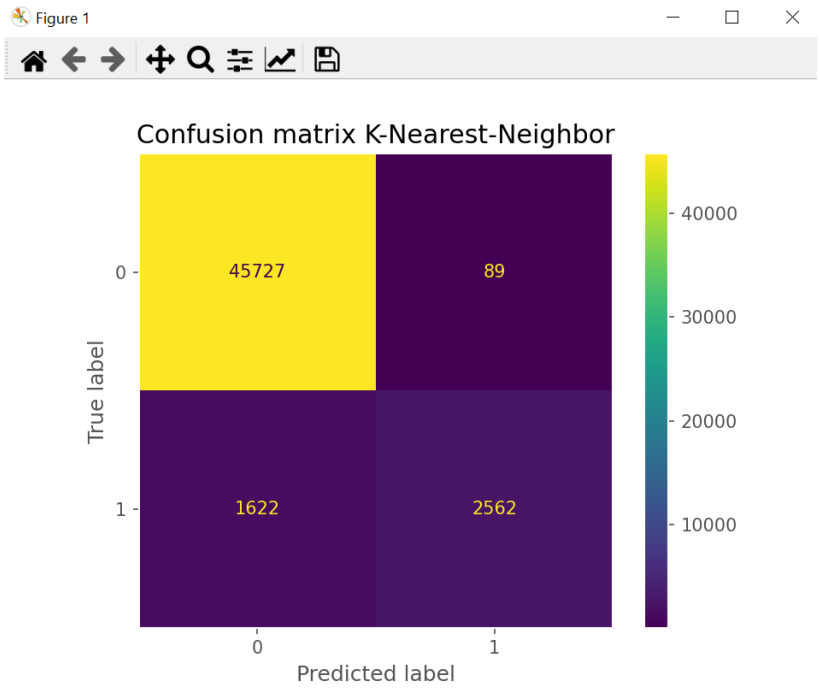
Passaggi fondamentali:

1. **Scegliere il numero di vicini (k)**: L'utente specifica il numero di vicini più prossimi k che verranno considerati per classificare un nuovo punto.
2. **Calcolare la distanza**: Per ogni nuova osservazione, KNN calcola la distanza tra questa osservazione e tutte le osservazioni del dataset di addestramento. La distanza più comunemente usata è la distanza euclidea, ma si possono usare altre metriche come la distanza di Manhattan.
3. **Identificare i K vicini più vicini**: Una volta calcolate le distanze, KNN seleziona i k punti dati più vicini alla nuova osservazione.
4. **Assegnare una classe** (per classificazione): L'osservazione viene assegnata alla classe più frequente tra i suoi vicini.

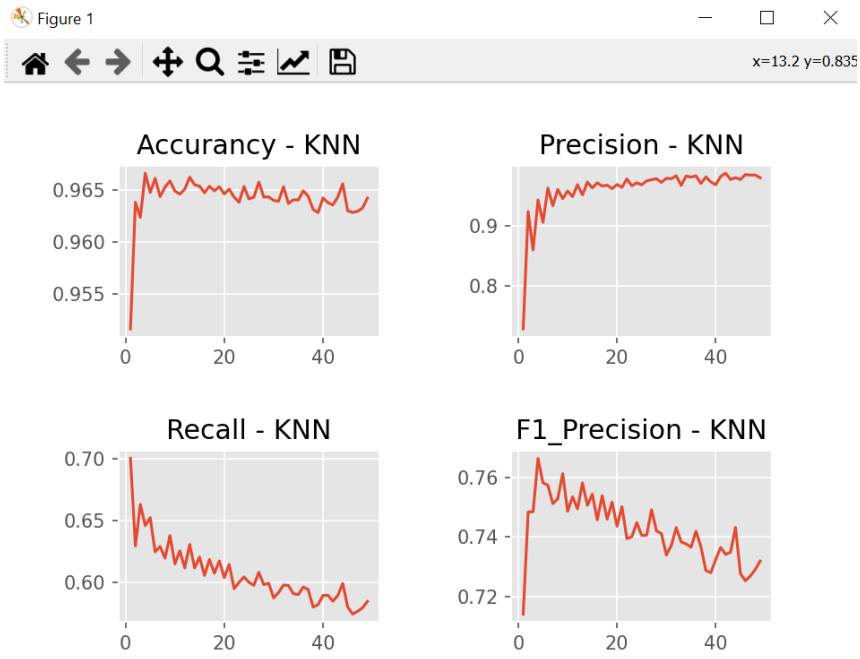
Qui di seguito vengono mostrati a video i risultati delle metriche calcolate:

```
Knn tree metrics
Accurancy : 0.966
Precision : 0.966
Recall : 0.612
F1_precision : 0.750
```

La Matrice di confusione del K-nearest neighbor è la seguente:

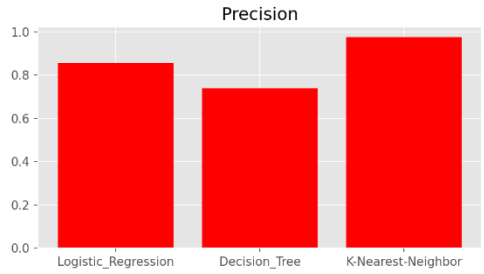


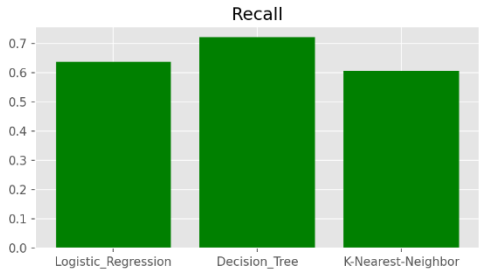
in questo caso abbiamo 2562 casi true-positive, ovvero che appartengono alle persone la cui predizione sul diabete ha dato esito positivo. Abbiamo 45727 true negative, mentre il false-positive(89) ed il false-negative(1622) non sono stati predetti correttamente.

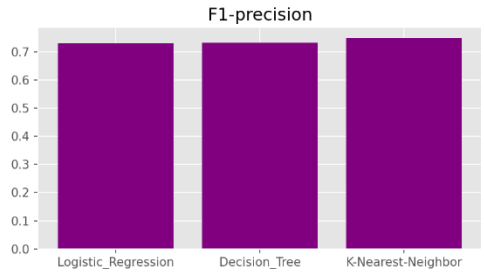


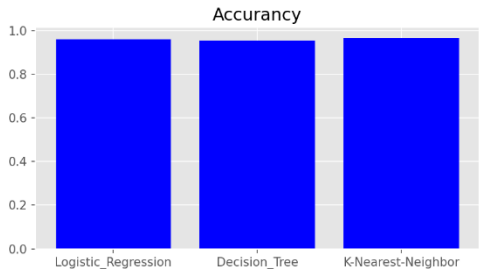
In questo caso variano i neighbors.

3.5 Confronto modelli di apprendimento supervisionato









In linea generale le prestazioni dei modelli sono simili con qualche differenza soprattuto tra le misure di precision e recall.

4 Considerazioni finali

In conclusione, il sistema realizzato soddisfa gli obiettivi da me stabiliti inizialmente. Per il sistema esperto con la relativa gestione dell'ontologia, ho riscontrato una funzionalità soddisfacente sia nell'implementazione del CSP con cui gestistco le prenotazioni delle visite del sangue, sia con la diagnostica dei sintomi dei pazienti relativa all'ontologia.

Inoltre, ho utilizato diversi modelli di apprendimento supervisionato sul dataset, per poter confrontare la loro efficenza su uno stesso dataset per un determinato caso di studio. Ho riscontrato buoni risultati nell'usare questi modelli e ho potuto vedere le differenze tra le misure calcolate ed i casi trovati correttamente e non, attraverso la matrice di confusione.