SmartDiet AI: Dalla Predizione del Gusto alla Pianificazione Automatica sotto Vincoli Nutrizionali

Corso didattico

- **Ingegneria della Conoscenza [cod. 063507]**
- **2025/2026**

Gruppo di lavoro

- **Dolidze Roland [796269]**
  r.dolidze@studenti.uniba.it

Sommario

1 Introduzione

Il presente progetto si pone l'obiettivo di realizzare un sistema intelligente per la generazione di piani alimentari settimanali personalizzati. La sfida risiede nel conciliare le esigenze fisiologiche dell'utente con le sue preferenze di gusto e ai vincoli logistici (intolleranze, ingredienti disponibili).

Per affrontare la complessità del problema, il sistema è stato progettato integrando tre diverse aree dell'Intelligenza Artificiale:

- **Machine Learning :** Utilizzo di un modello di classificazione per predire il gusto prevalente (*primary_taste*) di una ricetta a partire dai suoi ingredienti.
- **Knowledge Base:** Implementazione di una Base di Conoscenza (KB) per modellare il dominio nutrizionale, definendo fatti e regole logiche relative a diete specifiche (es. vegetariana, intollerante al lattosio) e proprietà degli alimenti.
- **Constraint Satisfaction Problem:** Modellazione della dieta settimanale come un problema di soddisfacimento di vincoli (CSP), dove il sistema deve incastrare ricette e pasti rispettando limiti, varietà alimentare e preferenze espresse dall'utente.

1.1 Dataset utilizzato

Il dataset utilizzato sarà:
https://www.kaggle.com/datasets/wafaaelhusseini/extended-recipes-dataset-64k-dishes

2 Fase di apprendimento

L'obiettivo di questo modulo è classificare il profilo di gusto prevalente di una ricetta partendo dalla lista dei suoi ingredienti. Questo ci permetterà di costruire una dieta gradevole per l'utente.

2.1 Raffinamento dati:

Data Cleaning: Il dataset fornisce dati grezzi, vi è quindi necessità di pulirli rimuovendo il testo non necessario "unità di misura, punteggiatura". Inoltre, il testo è stato normalizzato in minuscolo
Vettorizzazione: Le ricette vengono trasformate in vettori booleani "bag of words limitato alle 500 feature più frequenti" dove ogni posizione indica la presenza o l'assenza di un determinato ingrediente

2.2 Primo apporccio "Albero di decisione"

Si utilizza un modello di apprendimento supervisionato, nello specifico un albero di decisione, per predire il primary taste di una certa ricetta

Un albero di decisione è un albero binario costituito da:

nodi interni (non-foglia) etichettati con condizioni, ossia test booleani basati sui valori delle feature negli esempi e connessi con due figli, radici di sotto-alberi, attraverso archi tipicamente etichettati con true e false;

nodi-foglia etichettati con una stima puntuale per la feature obiettivo: la classe in un albero di classificazione oppure un valore reale in un albero di regressione.

2.3 Valutazione

Il modello è stato allenato sull' 80% del dataset per poi essere valutato sull' 20% restante. Sono stati calcolati i parametri di:

Precision

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

Recall

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

F1

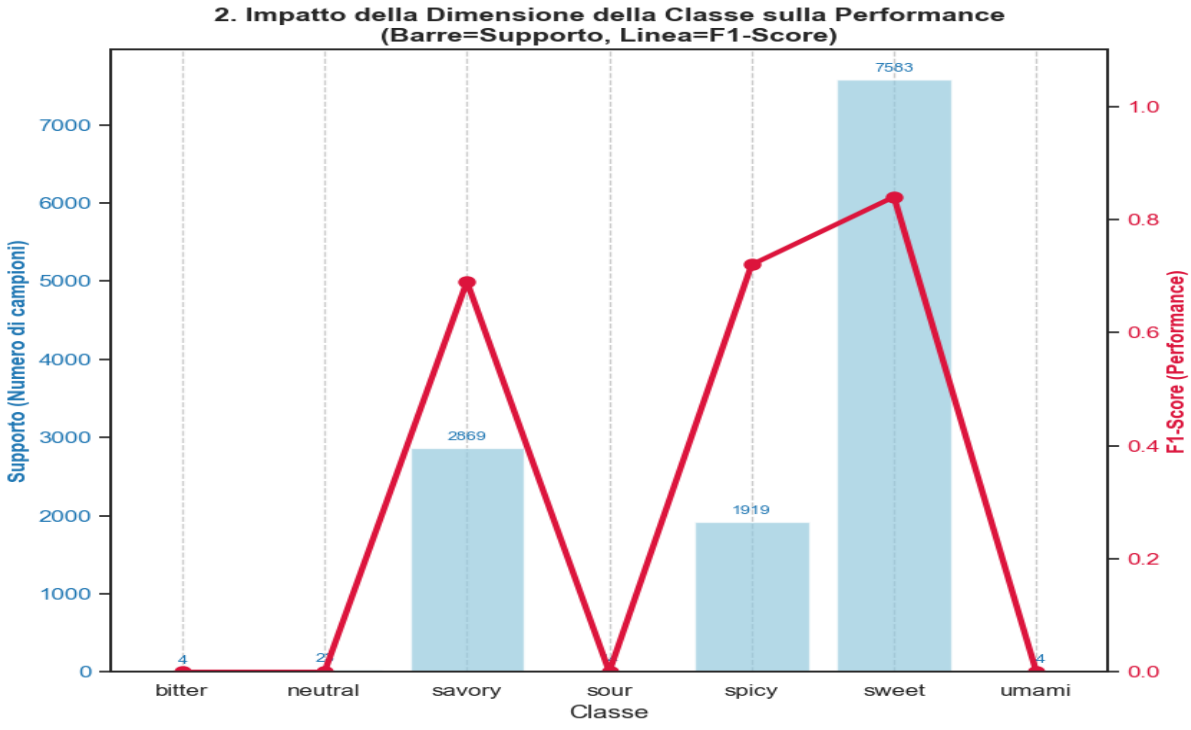
$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

ROC AUC (Compute Area Under the Receiver Operating Characteristic Curve), ovvero la misura dell'intera area bidimensionale sotto l'intera curva ROC, ci fornisce una misurazione aggregata del rendimento di un modello di classificazione in tutte le possibili soglie di classificazione. Questi sono stati i risultati iniziali:

```
Classification report:
              precision    recall  f1-score   support

      bitter       0.00      0.00      0.00         4
     neutral       0.00      0.00      0.00        23
      savory       0.53      1.00      0.69      2869
        sour       0.00      0.00      0.00        14
       spicy       0.82      0.64      0.72      1919
       sweet       1.00      0.73      0.84      7583
       umami       0.00      0.00      0.00        14

    accuracy                           0.77     12426
   macro avg       0.34      0.34      0.32     12426
weighted avg       0.86      0.77      0.78     12426

ROC score:  0.8259157811133768
```

Criticità

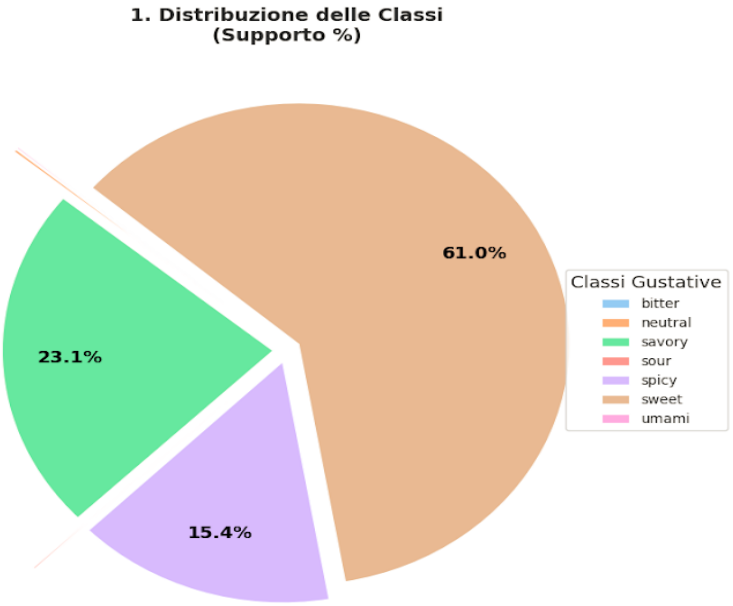
1) Un ROC score relativamente basso, risolvibile con una ottimizzazione degli iperparametri (K fold cross validation)
2) Cosa più importante precisione “0” per alcuni gusti “bitter, neutral, sour, umami”, precisioni comunque basse “0.53” per savory e precisione altissima “1” per sweet.
   Si è seguito per questo una analisi del dataset.



Con questo grafico riusciamo a vedere più facilmente quantò squilibrio ci sia nella classificazione per i vari “gusti”

Analisi dataset e bilanciamento dati

Analizzando meglio l’intero dataset si trova una distribuzione di classi molto sbilanciata



Quindi le basse precisioni e il “sweet bias” erano dovute ad uno sbilanciamento dei dati che portava il sistema a rispondere sempre “sweet”. Si scartano quindi i primary_taste con dati insufficienti e si sceglie di allenare il modello su un dataset ridotto che abbia 9500 piatti per ognuno dei 3 principali gusti “Sweet, Savory, Spicy”.

K fold cross validation

Per poter trovare i migliori iperparametri possibili per l’albero di decisione si è deciso di usare un:

Randomized Search: Una ricerca stocastica sugli iperparametri (criterio di split, profondità massima, campioni minimi per foglia). Si usa la funzione RandomizedSearchCV della libreria sklearn con:

n_iter=3: Questo parametro della RandomizedSearchCV indica che vengono pescate a caso 3 combinazioni diverse dal tuo dizionario di iperparametri.

Cross-Validation: Per garantire la robustezza del modello è stato utilizzato i k fold cross validation. Si usa la funzione RepeatedKFold della libreria sklearn. I parametri saranno:

n_splits=10: Il dataset di addestramento viene diviso in 10 parti. In ogni iterazione, 9 parti servono per l'addestramento e la decima per la validazione.

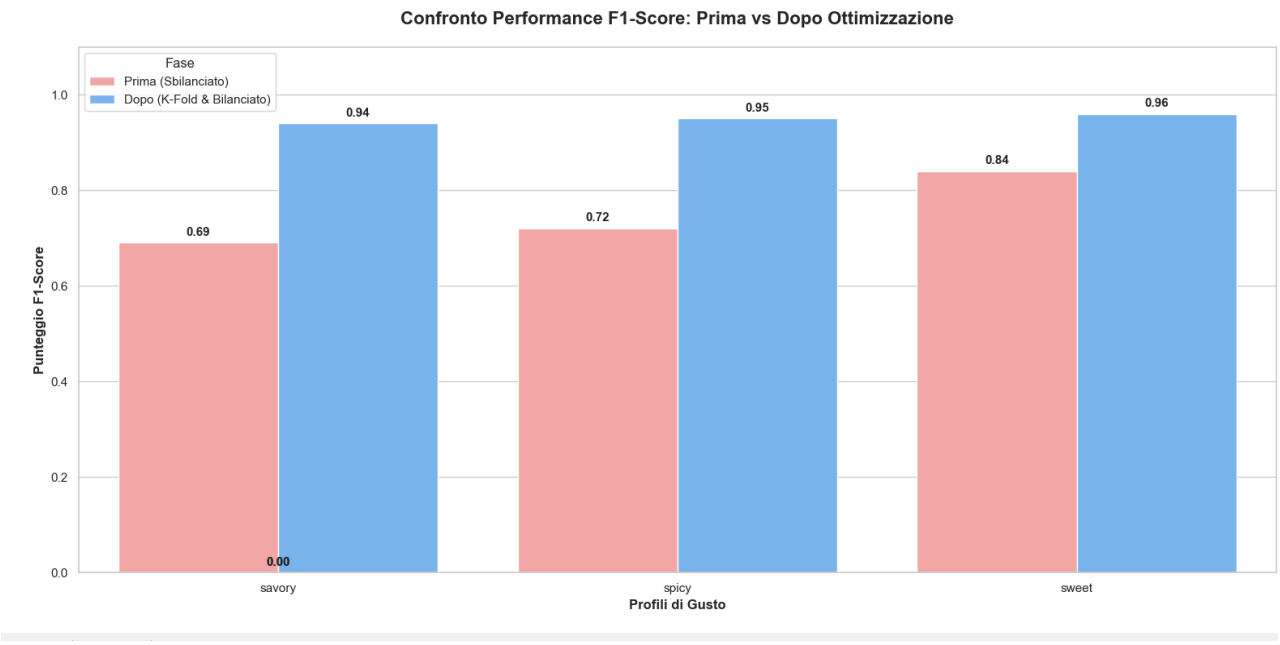
n_reapeats=2: l'intero processo di K-Fold viene ripetuto 2 volte. Ad ogni ripetizione, i dati vengono mescolati in modo diverso prima di essere divisi nei fold.

random_state=1: Garantisce che la suddivisione dei dati sia riproducibile.

Valutazione post modifiche

Dopo il bilanciamento dei dati e l'utilizzo del k fold cross validation si nota un miglioramento sostanziale in termni di prestazioni.

Da 0.87… a 0.97… di ROC score




```
--- MIGLIORI IPERPARAMETRI TROVATI ---
criterion: entropy
max_depth: 25
min_samples_leaf: 2
min_samples_split: 10

Ricomponiamo il modello utilizzando i nuovi iperparametri...
Classification report:
              precision    recall  f1-score   support

      savory       0.88      1.00      0.94      1900
       spicy       0.98      0.92      0.95      1900
       sweet       0.99      0.93      0.96      1900

    accuracy                           0.95      5700
   macro avg       0.95      0.95      0.95      5700
weighted avg       0.95      0.95      0.95      5700

ROC score:  0.9785560018467221

Fase di apprendimento completata con successo.
Il modello è ora addestrato su un dataset bilanciato (n=~28500 totali).
```

2.4 Secondo approccio Random Forest

Con il random forest si ha un algoritmo che invece di creare su un solo albero ne crea molti. Ogni albero viene addestrato su una porzione casuale dei dati e delle caratteristiche. Il risultato finale è la media dei voti per la classificazione di tutti gli alberi.
Si ha un notevole aumento in termini di complessità, ma si ha una riduzione nel rischio di overfitting e si ha un miglioramento in termini di performance.

Valutazione

Eseguendo anche sul random forest il K fold cross validation per la ricerca ottimale dei iperparametri e facendo il bilanciamento del dataset si ottengono tali misure di valutazione:

(IPERPARAMETRI DI DEFAULT)

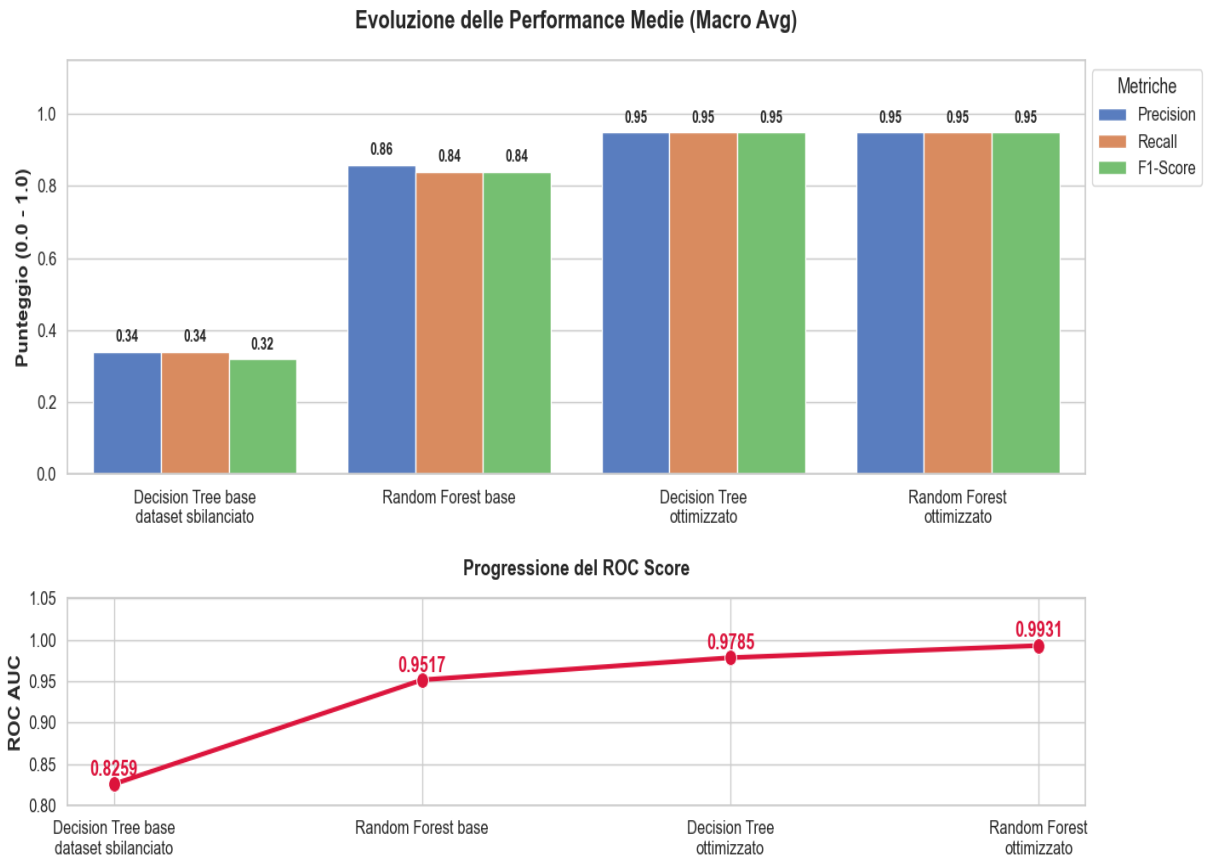
```
Iniziale composizione del modello con iperparametri basici...

Valutazione del modello base...
Classification report:
              precision    recall  f1-score   support

      savory       0.75      0.93      0.83      1900
       spicy       0.86      0.81      0.84      1900
       sweet       0.96      0.78      0.86      1900

    accuracy                           0.84      5700
   macro avg       0.86      0.84      0.84      5700
weighted avg       0.86      0.84      0.84      5700

ROC score:  0.9517692059095108
```

(IPERPARAMETRI SCELTI CON IL K FLOD CROSS VALIDATION)

```
Ricomponiamo il modello utilizzando i nuovi iperparametri...
Classification report:
              precision    recall  f1-score   support

      savory       0.89      0.99      0.94      1900
       spicy       0.96      0.93      0.95      1900
       sweet       1.00      0.92      0.96      1900

    accuracy                           0.95      5700
   macro avg       0.95      0.95      0.95      5700
weighted avg       0.95      0.95      0.95      5700

ROC score:  0.9931945983379502
```

si nota un ROC score alto da subito anche con gli iperparametri “basici”, che però con la ricerca dei iperparametri ottimali migliora fino ad arrivare ad uno score eccelente

2.5 Conclusione sull'evoluzione del Modello



L'analisi dei dati di performance conferma che l'ottimizzazione del modello ha seguito un percorso incrementale di successo, guidato da tre interventi chiave:

- **Risoluzione del Bias Iniziale**: Il passaggio dal dataset sbilanciato (F1-Score: **0.32**) al dataset bilanciato ha rimosso la distorsione del modello, permettendo una classificazione equa di tutte le categorie.
- **Efficacia del Tuning degli Iperparametri**: L'implementazione di parametri specifici (come max_depth: 25 e criterion: entropy) ha permesso al Decision Tree di passare da un'accuratezza dello **0.84** allo **0.95**, massimizzando la capacità di apprendimento dai dati.
- **Stabilità dell'Architettura Ensemble**: Il **Random Forest ottimizzato** rappresenta la configurazione finale superiore. Sebbene l'F1-Score si stabilizzi allo **0.95**, il **ROC Score di 0.9931** dimostra una capacità quasi perfetta di discriminazione tra le classi, minimizzando i falsi positivi e garantendo la massima robustezza predittiva.

Sintesi finale: L'attuale modello Random Forest è **altamente performante e pronto per l'implementazione**, avendo raggiunto un equilibrio ottimale tra precisione (0.95) e capacità di generalizzazione (ROC 0.99).

3 KNOWLEDGE BASE

Una Knowledge Base (KB) è una Infrastruttura di dati che va oltre il concetto di database tradizionale: essa non si limita ad archiviare dati, ma rappresenta la conoscenza attraverso fatti e regole logiche. Grazie a questa struttura basata su assiomi, la KB permette di dedurre nuove informazioni e identificare incongruenze logiche.

Nel contesto del nostro caso di studio, essa funge da nucleo informativo che permette all'utente di interrogare il dominio di riferimento in modo intelligente, ottenendo risposte derivanti da un processo di ragionamento sui dati.

3.1 GESTIONE DELLE KNOWLEDGE BASE

Per l'implementazione della Knowledge Base (KB) è stata utilizzata la libreria Pytholog, che permette di integrare la potenza della programmazione logica di Prolog direttamente all'interno dell'ecosistema Python.

L’architettura della KB è strutturata in tre livelli logici:

1) Popolamento Automatico (Fatti Dinamici): I fatti vengono generati programmaticamente a partire dal dataset. Questo approccio garantisce la sincronizzazione dei dati: ogni aggiornamento del

dataset sorgente si riflette immediatamente nella KB, mantenendo il sistema aggiornato senza interventi manuali sulla base di conoscenza.
2) Tassonomia del Dominio (Fatti Statici): Oltre ai dati estratti, sono stati codificati fatti di classificazione che definiscono le proprietà nutrizionali intrinseche degli ingredienti, indipendentemente dalle singole ricette.
3) Strato di Inferenza (Regole): È stato definito un set di regole logiche progettate per modellare le interazioni dell'utente. Queste regole permettono di trasformare i fatti grezzi in informazioni ad alto valore aggiunto (es. idoneità dietetica, bilanciamento macro-nutrizionale), abilitando un'interazione di tipo deduttivo.

3.2 Rappresentazione della Conoscenza

Fatti

I fatti rappresentano gli assiomi sempre veri della knowledge base, e sono la base per il funzionamento delle regole.

1) Fatti estratti dal Dataset Questi fatti sono generati processando ogni riga del file CSV e descrivono le caratteristiche intrinseche di ogni ricetta.

1. **“taste(recipe_title, primary_taste)”**: Rappresenta il profilo di gusto prevalente della ricetta. Es. taste(spaghetti_carbonara, savory)
2. **“is_veg(recipe_title, is_vegetarian)”**: Indica se la ricetta è vegetariana (yes/no). Es. is_veg(tofu_salad, yes)
3. **“no_gluten(recipe_title, is_gluten_free)”**: Specifica l'assenza di glutine nella ricetta. Es. no_gluten(risotto_ai_funghi, yes)
4. **“nut_free(recipe_title, is_nut_free)”**: Indica se la ricetta è sicura per chi soffre di allergie alla frutta a guscio. Es. nut_free(steamed_cod, yes)

5. **“contains(recipe_title, ingredient)”**: Associa una ricetta a ogni singolo ingrediente che la compone. Es. contains(margherita, tomato)

---

2) Fatti di Classificazione (Tassonomia del Dominio) Questi fatti definiscono categorie generali degli ingredienti basate su proprietà nutrizionali, indipendentemente dalle ricette.

1. **“is_protein_source(item)”: Definisce ingredienti che sono fonti primarie di proteine (carne, uova, tofu).**
2. **“is_fat_source(item)”: Identifica fonti principali di grassi (olio, burro, avocado).**
3. **“is_carb_source(item)”: Classifica ingredienti ricchi di carboidrati (pasta, riso, patate).**
4. **“is_fiber_source(item)”: Identifica ingredienti ad alto contenuto di fibre (lenticchie, broccoli).**
5. **“is_electrolyte_source(item)” / “is_potassium_source(item)”: Categorizza ingredienti ricchi di micronutrienti specifici. Es. is_potassium_source(banana)**
6. **“is_vegetable_source(item)”: Identifica ingredienti appartenenti alla categoria delle verdure (es. carote, spinaci, peperoni).**
7. **“is_fish_source(item)”: Specifica se un ingrediente è un prodotto ittico (es. salmone, tonno, gamberi).**
8. **“is_high_calorie(item)”: Identifica ingredienti ad alta densità energetica utilizzati per diete specifiche (es. burro d'arachidi, miele, olio d'oliva).**

Regole

Queste regole permettono al sistema di inferire proprietà nutrizionali e dietetiche complesse correlando gli ingredienti contenuti nelle ricette con la tassonomia definita.

1) Regole di Derivazione Nutrizionale

“has_protein(R)”: Una ricetta R possiede proteine se contiene un ingrediente che è fonte di proteine.

“has_fat(R)”: Identifica ricette contenenti fonti di grassi.

“has_carb(R)”: Identifica ricette contenenti fonti di carboidrati.

“has_fiber(R)”: Identifica ricette ad alto contenuto di fibre.

“is_safe_nut_allergy(Recipe)”: Una ricetta è sicura per allergici alla frutta a guscio se il fatto nut_free è "yes".

“has_veggies(R)”: Una ricetta R contiene verdure se include un ingrediente classificato come is_vegetable_source.

“has_fish(R)”: Identifica ricette che contengono pesce.

“has_electrolytes(R)” / “has_potassium(R)”: Verificano la presenza di micronutrienti essenziali per l'equilibrio idrosalino.

“has_high_energy(R)”: Una ricetta è considerata ad alta energia se contiene almeno un ingrediente ad alta densità calorica.

“high_fat(R)”: Regola di alias che identifica ricette con un contenuto lipidico rilevante (derivata da has_fat).

ES:

Carboidrati: $\text{has\_carb}(r) \iff \exists i(\text{contains}(r,i) \land \text{carb}(i))$

Fibre: $\text{has\_fiber}(r) \iff \exists i(\text{contains}(r,i) \land \text{fiber}(i))$

Verdure: $\text{has\_veggies}(r) \iff \exists i(\text{contains}(r,i) \land \text{veg}(i))$

2) Regole per Diete e Performance

“is_muscle_recovery(R)”: Una ricetta è adatta al recupero muscolare se contiene contemporaneamente proteine, elettroliti e potassio.

“is_weight_gainer(R)”: Classifica ricette adatte all'aumento di peso se includono ingredienti ad alta densità calorica, carboidrati e grassi.

“is_mediterranean(R)”: Definisce una ricetta come "mediterranea" se contiene pesce, verdure e carboidrati.

“is_complete(R)”: Una ricetta è considerata nutrizionalmente completa se contiene tutti i macronutrienti (proteine, carboidrati e grassi).

“is_super_veggie(R)”: Una ricetta è classificata come "super veggie" se è vegetariana e contiene sia fibre che verdure.

“is_vitamin_full(R)”: Una ricetta è considerata ricca di vitamine se combina la presenza di pesce e verdure.

“is_athlete_diet(R)”: Definisce un pasto bilanciato per l'atleta standard, richiedendo la presenza contemporanea di proteine, carboidrati e fibre.

“is_keto_style(R)”: Identifica ricette che seguono un approccio chetogenico, basate principalmente sulla combinazione di proteine e grassi.

“is_peak_performance(R)”: Combina la regola del recupero muscolare con la presenza di carboidrati per definire un pasto ottimale per l'atleta.

ES:

Aumento Peso	$\text{weight\_gainer}(r) \leftarrow \text{high\_energy}(r) \land \text{has\_carb}(r) \land \text{has\_fat}(r)$
Mediterranea	$\text{mediterranean}(r) \leftarrow \text{has\_fish}(r) \land \text{has\_veggies}(r) \land \text{has\_carb}(r)$
Completa	$\text{complete}(r) \leftarrow \text{has\_protein}(r) \land \text{has\_carb}(r) \land \text{has\_fat}(r)$

3.3 Ottimizzazione

Per garantire l'efficienza del sistema basato su conoscenza, è stata effettuata una scelta progettuale sulla scalabilità:

- **Campionamento Strategico:** Dato che il backtracking di Prolog può diventare oneroso su dataset molto vasti, la KB viene popolata con un sottoinsieme ottimizzato di 10.000 ricette. Questo garantisce tempi di risposta rapidi per il modulo successivo (CSP) senza sacrificare la varietà delle soluzioni.
- **Robustezza:** Il codice include controlli per gestire casi in cui il motore di inferenza non trovi soluzioni (gestione dei ritorni No), assicurando che il sistema non si blocchi durante la generazione della dieta.

3.4 Analisi Critica e Limiti Prestazionali

Nonostante l'efficacia del paradigma logico, l'implementazione pratica ha evidenziato dei compromessi necessari, legati principalmente allo strumento scelto per l'interfacciamento.

Criticità Pytholog

Utilizzando **Pytholog** per integrare Prolog in un ecosistema Python, introduciamo delle limitazioni significative in termini di velocità computazionale

1) Essendo un motore di inferenza interpretato interamente in Python, non gode delle ottimizzazioni a basso livello tipiche dei compilatori Prolog nativi .
2) La ricerca nello spazio degli stati, fondamentale per il backtracking di Prolog, risulta intrinsecamente lenta in un ambiente non ottimizzato, portando a tempi di latenza percepibili.

Impatto sul CSP

Questa lentezza strutturale della Knowledge Base ha un effetto a cascata sul modulo CSP**:**

Il tempo totale di generazione della dieta non è dato solo dalla risoluzione dei vincoli del CSP, ma è pesantemente influenzato dal tempo di risposta della KB. In scenari con molte query, il tempo di risposta di Pytholog diventa il fattore dominante nel rallentamento del sistema.

4 Constraint Satisfaction Problem (CSP)

Il modulo finale si occupa della pianificazione della dieta settimanale. Il problema è stato modellato come un **CSP**, dove l'obiettivo è assegnare una ricetta a ogni pasto della settimana rispettando vincoli nutrizionali, logici e di preferenza.

il CSP può essere definito come:

formalismo matematico e logico utilizzato nell'Intelligenza Artificiale per descrivere e risolvere problemi in cui l'obiettivo non è semplicemente trovare "un valore", ma trovare una combinazione di valori che rispetti un insieme di regole rigide

Ogni CSP è definito da una tripla P = < X, D, C> tale che:

X= Set di variabili che devono ricevere un valore

D= Dominio dei possibili valori per le x1, x2.. appartenenti a X

C= Regole da rispettare per le assegnazioni

4.1 Implementazione CSP

Per l'implementazione è stata utilizzata la libreria python-constraint. Il processo si articola in tre fasi principali:

Variabili

21 variabili totali (7 giorni x 3 pasti).

Definizione del Dominio e Filtraggio Primario

Prima di avviare il risolutore, il sistema interroga la KB per restringere il campo di ricerca. Una ricetta entra nel dominio di una variabile solo se supera i controlli di sicurezza:

- **Salute e BMI:** Se l'utente ha un $BMI > 25$, vengono automaticamente esclusi i piatti classificati come unhealthy.
- **Intolleranze:** Filtri rigorosi su Lattosio, Noci e Glutine basati sui fatti is_dairy_free, nut_free e no_gluten.
- **Tipologia Pasto:** Suddivisione dei piatti in "Colazione" (gusti *sweet* o *neutral*) e "Pasti Principali".

Nota di Varietà: Prima dell'assegnazione, viene eseguito un random.shuffle(domain). Questo assicura che il sistema non generi sempre lo stesso menu, garantendo dinamicità all'utente.

Vincoli

Il sistema applica tre tipologie di vincoli per garantire una dieta equilibrata:

1. **Vincoli di Varietà:** Utilizzo di AllDifferentConstraint per assicurare che i pranzi e le cene siano diversificati durante la settimana e che nello stesso giorno non si consumi lo stesso piatto due volte.
   ES:

   Pranzi diversi: $\text{AllDifferent}(\{g_i\_\text{Pranzo} \mid \forall g_i\})$

   Cene diverse: $\text{AllDifferent}(\{g_i\_\text{Cena} \mid \forall g_i\})$

   Giornaliero: $\forall g \in \text{giorni}, g\_\text{Pranzo} \neq g\_\text{Cena}$
2. **Vincoli Nutrizionali:** Impedisce l'accumulo di grassi eccessivi nello stesso giorno (es. non è permesso accoppiare un pranzo "High Fat" con una cena "High Fat").
   ES:

   $$\forall g \in \text{giorni}, \neg(\text{high\_fat}(g\_\text{Pranzo}) \land \text{high\_fat}(g\_\text{Cena}))$$
3. **Vincoli di Target:** Se l'utente è uno sportivo o ha un BMI basso, il CSP impone la presenza di almeno un piatto "Target" nella settimana (es. is_muscle_recovery o is_weight_gainer).

   ES:

   Mediterranea: $\exists v \in X_{Pasti} : KB \vdash \text{is\_mediterranean}(v)$

   Sport (se attivo): $\exists v \in X_{Pasti} : KB \vdash (\text{recovery}(v) \lor \text{peak}(v) \lor \text{athlete}(v))$

   BMI (se < 18.5): $\exists v \in X_{Pasti} : KB \vdash \text{weight\_gainer}(v)$

   Vegetariano (se attivo): $\exists v \in X_{Pasti} : KB \vdash \text{super\_veggie}(v)$

4.2 Problemi riscontrati

Durante le prove del programma si è notata una notevole latenza nella risoluzione del CSP, questa latenza è dovuta al fatto che quando si chiama il risolutre del CSP

il risolutore esplora lo spazio degli stati effettuando migliaia di tentativi d'assegnazione attraverso il backtracking**,** il tutto però utilizzando query Pythlog, che non solo hanno le varie problematiche già indicate sopra,
ma come già spiegato, effettuano una ricerca nell'albero logico della Knowledge Base.
Il tutto ci porta ad avere un "Backtracking Annidato", Backtracking del CSP che contiene al suo interno un Backtracking di Prolog

Un'altra problematica minore ma dello stesso genere "termini di tempo" è dovuto al caricamento della KB "serializzata con pickle", essa richiede un notevole quantitativo di tempo per essere caricata(1-2 min)

4.3 Ottimizzazioni e Limitazioni

Per garantire la reattività del sistema e gestire l'elevata complessità computazionale derivante dall'integrazione tra logica simbolica (Prolog) e ricerca combinatoria (CSP), sono state adottate le seguenti strategie:

1. **Campionamento Strategico del Dataset:** Il dataset è stato ridotto a 10.000 istanze rappresentative. Questa scelta permette di bilanciare la varietà delle ricette con la necessità di mantenere i tempi di risposta entro limiti accettabili per l'utente finale.
2. **Rilassamento dei Vincoli (Constraint Relaxation):** Si è optato per un modello di vincoli meno denso, privilegiando la consistenza nutrizionale rispetto a vincoli estetici o di varietà estrema, per ridurre la profondità dell'albero di ricerca.

4.4 Casi Limite

Si osserva che per profili utente estremamente restrittivi (es. coesistenza di molteplici intolleranze e vincoli BMI stringenti), la riduzione del dominio operata nel pre-processing può portare a un problema di **Over-constrained CSP**, ovvero una situazione in cui l'insieme delle soluzioni ammissibili è vuoto o eccessivamente ridotto per garantire la generazione di una dieta settimanale completa.

5 Conclusioni e Sviluppi Futuri

Il presente progetto ha dimostrato l'efficacia dell'integrazione di diverse branche dell'**Intelligenza Artificiale** per la risoluzione di un problema complesso e multidimensionale come la pianificazione alimentare personalizzata. L'approccio ibrido, che combina apprendimento automatico, ragionamento logico e soddisfazione di vincoli, ha permesso di ottenere un sistema capace di bilanciare esigenze oggettive (nutrizione) e soggettive (gusto).

5.1 Sintesi dei Risultati

Il lavoro svolto ha portato al raggiungimento di diversi traguardi tecnici:

- **Classificazione Accurata:** Superando il problema iniziale dello sbilanciamento del dataset (Sweet Bias), il modello **Random Forest** ha raggiunto prestazioni eccellenti, con un valore di ROC approsimativamente del 0.99. Questo garantisce che il sistema "comprenda" realmente il profilo organolettico delle ricette.
- **Modellazione della Conoscenza:** La **Knowledge Base** ha permesso di astrarre concetti complessi (es. "dieta per atleti" o "recupero muscolare") partendo da semplici fatti atomici, trasformando un database statico in un sistema dinamico capace di inferenza.
- **Pianificazione Intelligente:** Il modulo **CSP** ha dimostrato la capacità di gestire la logica combinatoria della settimana alimentare, garantendo varietà e rispetto dei limiti di salute (BMI) e intolleranze.

5.2 Riflessioni Critiche e Limitazioni

Nonostante il successo del prototipo, l'analisi ha evidenziato alcuni colli di bottiglia, in particolare riguardo alle prestazioni:

Il problema del "Backtracking Annidato": La criticità maggiore risiede nell'integrazione tra Pytholog e il risolutore CSP. La latenza computazionale derivante dall'interrogazione di un motore di inferenza interpretato in Python durante la ricerca nello spazio degli stati del CSP rappresenta il limite principale alla scalabilità del sistema su dataset massivi.

Inoltre, il rischio di **Over-constrained CSP** per utenti con restrizioni multiple (es. celiaci, allergici alle noci e con BMI elevato contemporaneamente) evidenzia la necessità di implementare algoritmi di *soft-constraints* o di rilassamento dinamico dei vincoli più avanzati.

5.3 Sviluppi Futuri

Il progetto getta le basi per diverse evoluzioni future che potrebbero renderlo un prodotto pronto per il mercato:

1. **Ottimizzazione delle Prestazioni:** Migrare la Knowledge Base verso un motore Prolog nativo o utilizzare tecniche di *caching* delle query per eliminare i tempi morti durante la risoluzione del CSP.
2. **Apprendimento del Feedback Utente:** Implementare un sistema di *Reinforcement Learning* dove il piano alimentare si adatta nel tempo in base al gradimento effettivo espresso dall'utente dopo il consumo dei pasti.
3. **Espansione del Dominio:** Integrare API esterne per il monitoraggio in tempo reale dei valori nutrizionali e la possibilità di generare automaticamente la "lista della spesa" basata sugli ingredienti mancanti.
4. **Interfaccia Utente (UI):** Sviluppare un'applicazione mobile che permetta una consultazione rapida del piano e la sostituzione *on-the-fly* di singoli pasti mantenendo la coerenza dei vincoli settimanali.

In conclusione, sebbene la complessità computazionale della logica simbolica ponga delle sfide, l'unione tra la precisione statistica del Machine Learning e il rigore della logica formale si è confermata la strada corretta per creare un consulente nutrizionale artificiale realmente affidabile.