

NutriScore Ai

Documentazione sul caso di studio
Ingegneria della Conoscenza

Realizzato da:

Elmadhi Nazim, 758331, n.elmadhi@studenti.uniba.it

Repository GitHub:
[nutri-score](#)

AA 2025-2026

Indice

- 1. Introduzione3
- 2. Requisiti e Installazione4
 - 2.1 Requisiti funzionali4
 - 2.2 Installazione e avvio5
- 3. Dataset e Preprocessing.....5
 - 3.1 Descrizione del Dataset5
 - 3.2 Pipeline di Preprocessing.....7
 - 3.3 Arricchimento Semantico8
- 4. Knowledge Base e Regole9
 - 4.1 Architettura della Knowledge Base 10
 - 4.2 Motore Inferenziale e Regole 11
 - 4.3 Calcolo del Risk Score 12
- 5. Apprendimento supervisionato e Risultati..... 12
 - 5.1 Metodologia di Addestramento 12
 - Classificazione..... 13
 - Validazione Cross-Validation..... 13
 - Integrazione Neuro-Simbolica..... 14
 - Ottimizzazione Soglia 14
 - 5.2 Analisi dei Modelli..... 15
 - Decision Tree..... 15
 - K-Nearest Neighbors (KNN) 16
 - Logistic Regression..... 17
 - Random Forest..... 18
 - Multi-Layer Perceptron (MLP) 19
 - 5.3 Confronto Finale20
- 6. Sviluppi futuri21
- 7. Bibliografia21

1. Introduzione

La valutazione automatica della qualità nutrizionale dei prodotti alimentari è una problematica di crescente interesse nel campo dell'analisi dei dati e della nutrizione. L'ampia diffusione di prodotti industriali ultra-processati, spesso accompagnati da etichette complesse o potenzialmente fuorvianti, rende difficile per i consumatori interpretare correttamente le informazioni nutrizionali e compiere scelte consapevoli.

In questo scenario, l'analisi automatica dei dati alimentari assume un ruolo centrale, offrendo la possibilità di supportare decisioni informate attraverso modelli intelligenti e riproducibili. Tuttavia, i tradizionali sistemi di valutazione basati su punteggi fissi o soglie rigide presentano limiti significativi, in quanto non riescono a catturare la complessità delle relazioni tra i diversi componenti nutrizionali.

Il progetto NutriScore AI si propone di affrontare questa problematica mediante la realizzazione di un sistema di classificazione automatica della salubrità dei prodotti alimentari. L'obiettivo è distinguere tra prodotti *sani* e *non sani* sfruttando tecniche di Machine Learning supervisionato integrate con un sistema di ragionamento logico basato su Prolog.

L'approccio adottato è di tipo Neuro-Symbolic, combinando la capacità del Machine Learning di apprendere pattern complessi dai dati con la possibilità di incorporare conoscenza esperta sotto forma di regole logiche. Questa integrazione consente non solo di migliorare le prestazioni di classificazione, ma anche di aumentare la spiegabilità delle decisioni del modello, rendendo il sistema più trasparente e affidabile.

2. Requisiti e Installazione

2.1 Requisiti funzionali

Per garantire il corretto funzionamento del progetto, il sistema richiede una configurazione ibrida che supporti sia l'esecuzione di script Python sia il motore inferenziale logico.

Requisiti Software:

- Versione *Python 3.11*: Linguaggio base per la manipolazione dati e il Machine Learning.
- *SWI-Prolog*: Motore inferenziale necessario per l'esecuzione della Knowledge Base.

Librerie Python: Le dipendenze principali, elencate nel file `requirements.txt`, includono:

- *Pandas & Numpy*: Per la manipolazione vettoriale dei dati nutrizionali e la gestione del dataset strutturato.
- *Scikit-learn*: Per l'implementazione degli algoritmi di classificazione, le metriche di valutazione e le pipeline di preprocessing standard.
- *Imbalanced-learn*: Utilizzata per gestire il bilanciamento delle classi tramite tecniche di campionamento (*RandomOverSampler*) integrate nella pipeline di validazione.
- *Matplotlib*: Per la generazione delle curve ROC e la visualizzazione della stabilità dei modelli.
- *PySWIP*: Libreria "bridge" che consente a Python di interrogare la base di conoscenza Prolog, iniettare fatti e recuperare i risultati delle deduzioni logiche.

2.2 Installazione e avvio

Per avviare correttamente il software è necessario individuare, nella directory principale del progetto, il file `requirements.txt`, che contiene l'elenco completo delle dipendenze Python richieste.

L'installazione delle librerie può essere effettuata eseguendo il seguente comando:

```
pip install -r requirements.txt
```

È inoltre necessario assicurarsi che l'eseguibile di SWI-Prolog sia correttamente aggiunto alle variabili d'ambiente di sistema (PATH), in modo da consentire a *PySWIP* di invocare il motore inferenziale durante l'esecuzione del programma.

Una volta completata la configurazione dell'ambiente, il software può essere avviato eseguendo lo script principale:

```
python main.py
```

3. Dataset e Preprocessing

3.1 Descrizione del Dataset

Il sistema proposto si basa su un dataset di prodotti alimentari ottenuto a partire da *OpenFoodFacts* [4], una piattaforma open-source che fornisce informazioni nutrizionali strutturate su alimenti commercializzati a livello globale. Il dataset originale è stato filtrato e riorganizzato per costruire un insieme di dati coerente con gli obiettivi del progetto.

Dopo una fase di selezione e pulizia preliminare, è stato ottenuto un campione di circa 20.000 prodotti, scelti in modo da garantire una buona varietà nutrizionale e una distribuzione bilanciata delle classi. L'analisi si concentra su un insieme ridotto ma significativo di attributi nutrizionali, ritenuti particolarmente rilevanti per la valutazione della qualità alimentare.

Le feature numeriche considerate sono:

- *sugars_100g*: quantità di zuccheri per 100 grammi di prodotto;
- *carbohydrates_100g*: contenuto di carboidrati totali;
- *fat_100g*: contenuto di grassi totali;
- *salt_100g*: contenuto di sale (sodio);
- *fiber_100g*: quantità di fibre alimentari;
- *fruit_veg_100g*: percentuale di frutta e verdura;
- *additives_n*: numero di additivi presenti;
- *proteins_100g*: contenuto proteico.

Inoltre, viene utilizzato l'attributo testuale *product_name*, essenziale per l'analisi semantica e per inferire la categoria merceologica del prodotto all'interno del modulo ontologico.

Il problema è formulato come una classificazione binaria, in cui la variabile target è stata ottenuta discretizzando l'attributo originale *nutrition_grade_fr* (che corrisponde alla classe NutriScore [5]). In particolare:

- **0 (Sano)**: prodotti appartenenti alle classi NutriScore A o B;
- **1 (Non Sano)**: prodotti appartenenti alle classi C, D o E.

Questa discretizzazione consente di trasformare una valutazione multi-classe in un problema decisionale più diretto, focalizzato sull'individuazione dei prodotti nutrizionalmente sbilanciati.

Oltre alle feature grezze, il dataset viene successivamente arricchito con attributi derivati tramite ragionamento neuro-simbolico. Le regole definite nella Knowledge Base Prolog permettono di generare feature logiche complesse (es. *incoerenza funzionale*, *protein washing*, *trappola low-fat*), che vengono incorporate nel dataset finale per guidare l'apprendimento dei modelli di Machine Learning.

3.2 Pipeline di Preprocessing

I dati grezzi provenienti da *OpenFoodFacts* presentano spesso inconsistenze, valori mancanti o formati eterogenei. Per rendere tali dati idonei all'elaborazione tramite algoritmi di Intelligenza Artificiale, è stato necessario applicare una rigorosa pipeline di preprocessing.

Data la frequente assenza di alcuni valori nutrizionali (come fibre o sale) nel database originale, l'eliminazione delle righe incomplete avrebbe comportato una significativa perdita di informazioni. Si è pertanto optato per una strategia di imputazione basata sulla mediana statistica. Tale scelta, preferita alla media aritmetica, garantisce una maggiore robustezza rispetto agli outlier, preservando la distribuzione statistica dei dati anche in presenza di valori estremi.

Considerata l'eterogeneità delle scale di misura (ad esempio, il sodio in milligrammi rispetto ai carboidrati in grammi), è stata applicata una standardizzazione tramite *StandardScaler*. Questa operazione ha permesso di ricentrare le distribuzioni attorno allo zero con varianza unitaria, evitando che le variabili con ordini di grandezza maggiori influenzassero sproporzionatamente l'apprendimento degli algoritmi sensibili alla scala (come KNN o MLP).

Il dataset è stato ripartito in due sottoinsiemi distinti:

- Training Set (80%): utilizzato per l'addestramento dei modelli.
- Test Set (20%): preservato per la valutazione finale delle prestazioni. La suddivisione è stata effettuata in modo stratificato, mantenendo inalterata la proporzione tra prodotti "Sani" e "Non Sani" in entrambi gli insiemi, al fine di garantire una validazione statisticamente rappresentativa.

Per mitigare il rischio che i modelli privilegiassero la classe maggioritaria, è stata integrata una tecnica di *Random Oversampling* all'interno della pipeline di addestramento. Tale procedura prevede la

replicazione casuale degli esempi appartenenti alla classe minoritaria fino al raggiungimento dell'equilibrio tra le classi. L'operazione è stata eseguita esclusivamente sui dati di training, escludendo rigorosamente il test set per assicurare una valutazione imparziale e aderente alla realtà.

3.3 Arricchimento Semantico

Questa fase costituisce una parte importante dell'architettura e differenzia il presente lavoro dai classici approcci di Machine Learning puramente statistici. Prima della fase di addestramento, il dataset grezzo viene sottoposto a un processo di arricchimento semantico gestito dal modulo logico.

Il processo avviene secondo i seguenti step:

1. Interrogazione del Motore Inferenziale: Per ogni istanza del dataset, lo script Python interroga la Knowledge Base Prolog passando i dati nutrizionali e, fattore cruciale, il nome del prodotto normalizzato.
2. Ragionamento Contestuale: Il sistema non si limita a verificare soglie numeriche, ma attiva un ragionamento ontologico per inferire la categoria del prodotto e le sue proprietà funzionali attese (es. distinguere una fonte di energia da una semplice fonte di zuccheri).
3. Generazione delle Feature: Il risultato delle inferenze viene codificato in nuove colonne binarie (feature logiche) e in un punteggio di rischio semantico, che vengono aggiunti al dataframe originale.

Attraverso questa procedura di *Knowledge Injection*, il modello statistico riceve in input non solo i valori grezzi (osservazione), ma anche il risultato di una valutazione esperta pre-calcolata (ragionamento).

Un aspetto distintivo del sistema risiede nella gestione duale dei dati. Mentre gli algoritmi di Machine Learning necessitano di feature

matematicamente scalate (prive di unità di misura fisica) per ottimizzare la convergenza, il motore inferenziale Prolog richiede valori assoluti. Le regole nutrizionali, infatti, si basano su soglie biologiche concrete (es. grammi di zuccheri o milligrammi di sodio) e non su deviazioni statistiche astratte. Pertanto, il sistema preserva l'integrità dei valori originali esclusivamente per la fase di ragionamento logico, garantendo che le deduzioni siano semanticamente corrette e direttamente interpretabili dal punto di vista dietetico.

4. Knowledge Base e Regole

Viene implementato un paradigma Neuro-Symbolic [3], integrando la potenza predittiva del Machine Learning con la profondità semantica della logica Prolog. Questo approccio ibrido mira a superare i limiti dei modelli puramente statistici, aggiungendo uno strato di "comprensione" del dominio.

Il modulo Prolog non agisce solo come un validatore di soglie, ma come un motore di ragionamento ontologico e contestuale. Grazie alla struttura ontologica, il sistema inferisce la categoria del prodotto e le sue proprietà funzionali attese (es. un cereale deve essere una *Energy Source* valida, non solo zucchero).

Vengono generate feature avanzate come *is_functionally_inconsistent* o *violates_category_expectation*. Queste variabili forniscono al modello un'informazione qualitativa cruciale: segnalano quando un prodotto tradisce le aspettative nutrizionali della sua stessa categoria.

Tramite il predicato *compute_risk_score*, il sistema calcola un indice di gravità basato su pesi (es. penalità elevate per *misleading_label*). A differenza della probabilità statistica, questo score offre una spiegabilità causale, dettagliando matematicamente perché un prodotto è considerato a rischio in base alle regole violate.

4.1 Architettura della Knowledge Base

La componente simbolica del sistema è implementata tramite una base di conoscenza (Knowledge Base, KB) realizzata in Prolog [1]. La KB ha l'obiettivo di modellare in forma esplicita concetti nutrizionali di alto livello che non sono immediatamente osservabili a partire dai soli valori numerici.

La conoscenza è organizzata secondo una distinzione classica:

- ***facts***, che rappresentano dati e soglie nutrizionali;
- ***rules***, che codificano relazioni logiche e criteri.

Questa separazione consente una modellazione modulare e facilmente estendibile della conoscenza di dominio.

Il file facts.pl non si limita a un elenco di soglie, ma implementa una struttura ontologica che permette al sistema di "comprendere" la natura del prodotto.

È stata definita una gerarchia tramite il predicato isa(Sottocategoria, Categoria). Il sistema sa, ad esempio, che lo yogurt è un dairy e che la cola è un beverage. Grazie alla proprietà transitiva, i prodotti ereditano le aspettative nutrizionali della loro categoria madre, permettendo un ragionamento contestuale e non solo assoluto.

Sono stati definiti i parametri oggettivi che delimitano i concetti di "Alto", "Basso" o "Buono", basati su standard nutrizionali consolidati:

- *Zuccheri*: *High* (>15g) e *Very High* (>30g).
- *Grassi*: *Low* (<3g, utile per i claim "light") e *High* (>20g).
- *Sale*: Soglia di rischio fissata a 1.5g.
- *Fibre e Proteine*: Definite *Good* sopra i 3g e 5g (apporto positivo).
- *Additivi*: Un numero superiore a 3 indica un prodotto ultra-processato (*max*).

In fine è presente un dizionario di marketing_keyword (es. *bio*, *vegan*, *fit*, *detox*) utilizzato per analizzare le etichette testuali e individuare discrepanze tra il messaggio pubblicitario e il contenuto reale.

4.2 Motore Inferenziale e Regole

Il file `rules.pl` contiene le regole logiche che combinano i fatti per identificare profili di rischio complessi. Queste regole agiscono come Feature Constructors intelligenti per il Machine Learning.

Oltre alle verifiche di base, sono state implementate logiche avanzate:

- *violates_category_expectation* (Coerenza Contestuale): Sfruttando l'ontologia, questa regola verifica se un prodotto devia dagli standard della sua categoria. Permette di distinguere, ad esempio, un contenuto di zuccheri accettabile per un *biscotto* ma inammissibile per un'acqua aromatizzata.
- *is_functionally_inconsistent* (Coerenza Funzionale): Analizza se il prodotto supporta chimicamente la sua funzione. Un esempio chiave è la distinzione tra Carboidrati e Zuccheri: la regola penalizza i prodotti che dovrebbero fornire energia (come i cereali) ma contengono prevalentemente zuccheri semplici invece di amidi complessi.
- *is_low_fat_sugar_trap*: Identifica la trappola dei prodotti "Light": si attiva se i grassi sono bassi (< 3g) ma compensati da zuccheri alti (> 15g).
- *is_misleading_label*: Incrocia dati testuali e numerici. Si attiva se il nome vanta proprietà salutiste (es. "Barretta Fit") ma i valori nutrizionali reali superano le soglie di rischio.
- *is_protein_wash*: Segnala prodotti che usano termini come "Protein" o "Sport" nel nome senza raggiungere la soglia minima di proteine definita nei fatti.
- *is_hidden_sodium* e *is_empty_calories*: Rilevano rispettivamente l'eccesso di sale non bilanciato da vegetali e l'alta densità calorica priva di fibre.
- *is_hyper_processed* (Indice di Ultra-Processazione): Non guarda solo al singolo nutriente, ma rileva la "firma" del cibo industriale di bassa qualità. Si attiva in presenza di un numero eccessivo di

additivi (> 3) o quando c'è una co-presenza critica di zuccheri alti, grassi e sale (il cosiddetto *bliss point* artificiale).

4.3 Calcolo del Risk Score

Infine, il sistema implementa un meccanismo di quantificazione tramite il predicato *compute_risk_score*. Ad ogni regola violata è associato un peso ($\text{weight}/2$) che riflette la gravità nutrizionale (es. peso 5 per *misleading_label*, peso 3 per *hidden_sodium*). Il sistema calcola una somma ponderata di tutte le violazioni attive. Questo score fornisce una misura deterministica e spiegabile dell'insalubrità del prodotto, complementare alla probabilità statistica fornita dal modello neurale.

5. Apprendimento supervisionato e Risultati

5.1 Metodologia di Addestramento

L'approccio metodologico adottato per la costruzione del classificatore si basa sul paradigma dell'Apprendimento Supervisionato [2]. In questo contesto, l'algoritmo non opera "alla cieca", ma apprende la relazione sottostante tra le caratteristiche di un prodotto (input) e la sua effettiva qualità salutistica (output o *target*), sfruttando un vasto dataset di esempi precedentemente etichettati.

Il problema viene formulato come un task di classificazione binaria. La scelta di discretizzare il target in due sole categorie ("Sano" vs "Non Sano") risponde a una precisa esigenza applicativa: fornire al sistema la capacità di discriminare in modo netto i prodotti consigliabili da quelli da limitare.

L'obiettivo dell'addestramento, pertanto, è minimizzare l'errore di predizione su prodotti mai visti, generalizzando le regole apprese durante la fase di training per identificare correttamente la classe di appartenenza anche in presenza di combinazioni di nutrienti complesse o inedite.

Classificazione

In questo progetto, la classificazione viene utilizzata per predire la qualità nutrizionale dei prodotti alimentari, assegnando ciascun prodotto a una classe binaria sulla base di un insieme di caratteristiche nutrizionali raw e di feature derivate tramite reasoning simbolico.

Per identificare l'architettura più idonea alla valutazione della salubrità nutrizionale, è stato previsto un'analisi comparativa tra cinque diversi algoritmi, scelti per la loro eterogeneità e capacità di modellare relazioni differenti nei dati:

- **Decision Tree** (approccio basato su regole gerarchiche);
- **K-Nearest Neighbors (KNN)** (approccio basato sulla distanza);
- **Logistic Regression** (approccio lineare baseline);
- **Random Forest** (metodo ensemble basato su bagging);
- **Multi-Layer Perceptron (MLP)** (rete neurale artificiale).

Validazione Cross-Validation

La validazione dei modelli è stata condotta tramite Cross-Validation stratificata a 10 fold, utilizzando l'Area Under the Curve (ROC-AUC) come metrica guida. Questa scelta metodologica ha permesso di ottenere stime di performance robuste, minimizzando la dipendenza da specifiche partizioni dei dati e prevenendo l'overfitting. Per la ricerca degli iperparametri ottimali è stata impiegata la GridSearchCV per la maggior parte degli algoritmi, ricorrendo alla RandomizedSearchCV per l'MLP al fine di gestire efficientemente il suo ampio spazio di ricerca. La stabilità delle predizioni è stata ulteriormente verificata analizzando la varianza delle curve ROC tra i diversi fold.

Integrazione Neuro-Simbolica

A differenza delle architetture in cui il ragionamento logico interviene solo a valle come validatore, in questo progetto l'integrazione Neuro-Simbolica avviene a monte, nella fase di Feature Engineering. Le regole semantiche definite in Prolog (es. *is_sugar_trap*) vengono utilizzate per generare nuove variabili che arricchiscono il dataset di addestramento. In questo modo, il modello di Machine Learning non apprende solo dai dati grezzi, ma "internalizza" la conoscenza esperta durante il training stesso. Questo permette al classificatore di costruire alberi decisionali che integrano nativamente sia la statistica dei nutrienti sia le regole logiche di salubrità.

Ottimizzazione Soglia

Infine, per perfezionare la capacità decisionale, non è stata utilizzata acriticamente la soglia standard di classificazione (0.5). È stato invece eseguito un Threshold Tuning specifico: per ogni modello, la soglia è stata calibrata sul valore in grado di massimizzare l'F1-Score (ad esempio 0.45 per il Random Forest), bilanciando così in modo ottimale Precision e Recall.

5.2 Analisi dei Modelli

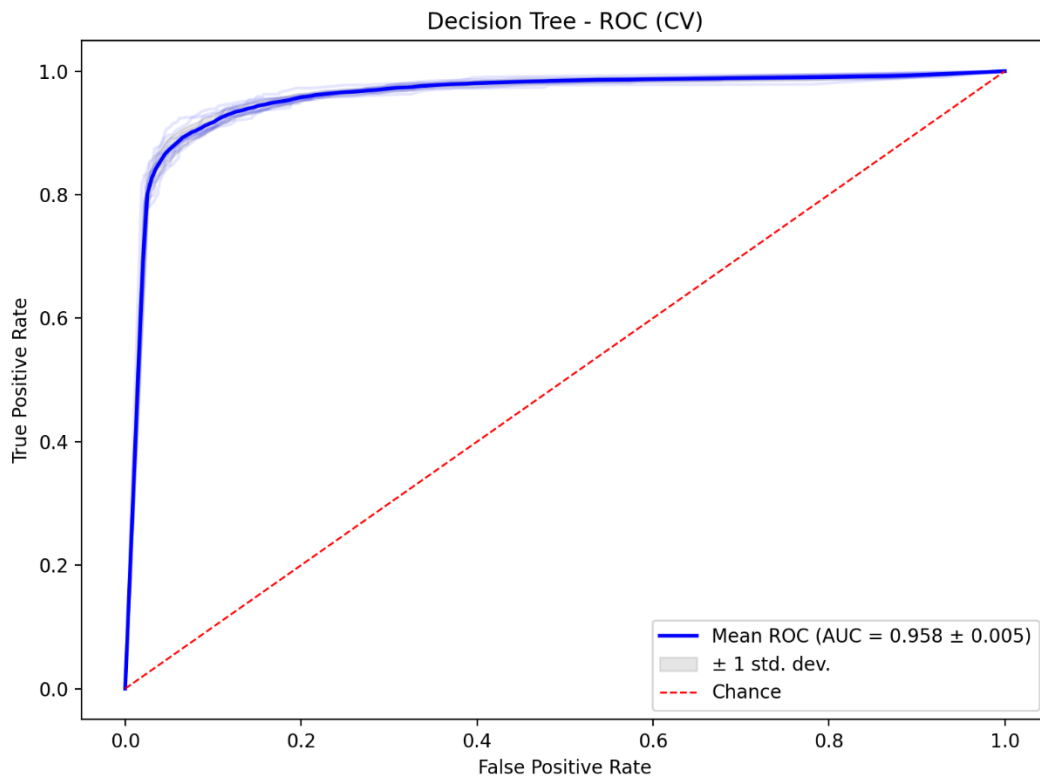
Decision Tree

Modello basato su regole decisionali gerarchiche.

Iperparametri migliori: $max_depth = 10$, $min_samples_leaf = 5$.

Performance: Il protocollo di validazione Stratified 10-Fold Cross-Validation ha confermato un'elevata stabilità del modello, con una varianza minima tra i fold. Il confronto tra le metriche di validazione e quelle di test mostra un ottimo allineamento, escludendo fenomeni significativi di overfitting:

- ROC-AUC (Cross-Validation): $0.958 (\pm 0.005)$.
- ROC-AUC (Test Set): 0.9582.



K-Nearest Neighbors (KNN)

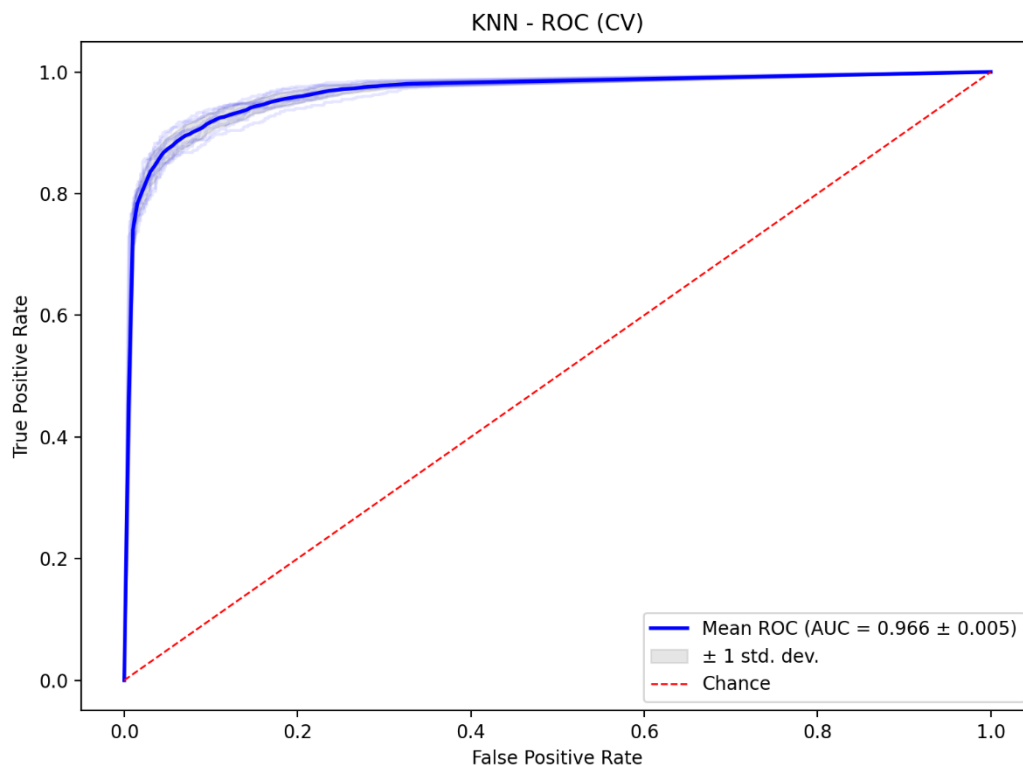
Algoritmo basato sulla distanza nello spazio delle feature.

Iperparametri migliori: $n_neighbors = 7$, $weights = 'distance'$.

Performance: Le metriche di generalizzazione indicano un'ottima capacità di ranking:

- ROC-AUC (Cross-Validation): $0.966 (\pm 0.005)$.
- ROC-AUC (Test Set): 0.9636.

Sebbene l'AUC sia alto, l'analisi della soglia ottimale rivela un bias estremo: per massimizzare l'F1-Score, il modello ha abbassato la soglia decisionale a 0.05. Ciò significa che il KNN classifica un prodotto come "Non Sano" anche con una probabilità minima (5%). Questo approccio garantisce una Recall altissima ma crolla in Precisione sul test set, generando un numero elevato di Falsi Positivi. È il modello meno equilibrato.



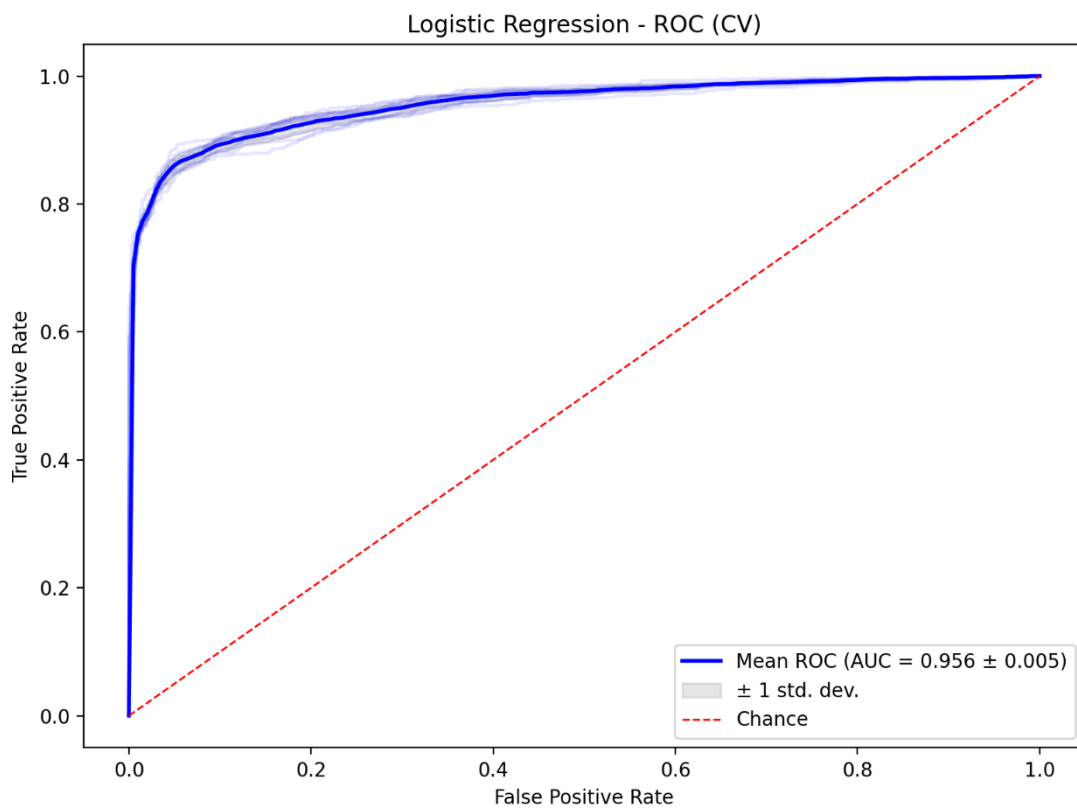
Logistic Regression

Modello lineare utilizzato come baseline.

Iperparametri migliori: $C = 1.0$, $class_weight = None$.

Performance: La sua semplicità lo rende molto interpretabile, ma fatica leggermente rispetto ai modelli non lineari nel catturare le interazioni complesse introdotte dalle nuove feature logiche.

- ROC-AUC (Cross-Validation): $0.956 (\pm 0.005)$.
- ROC-AUC (Test Set): 0.9559



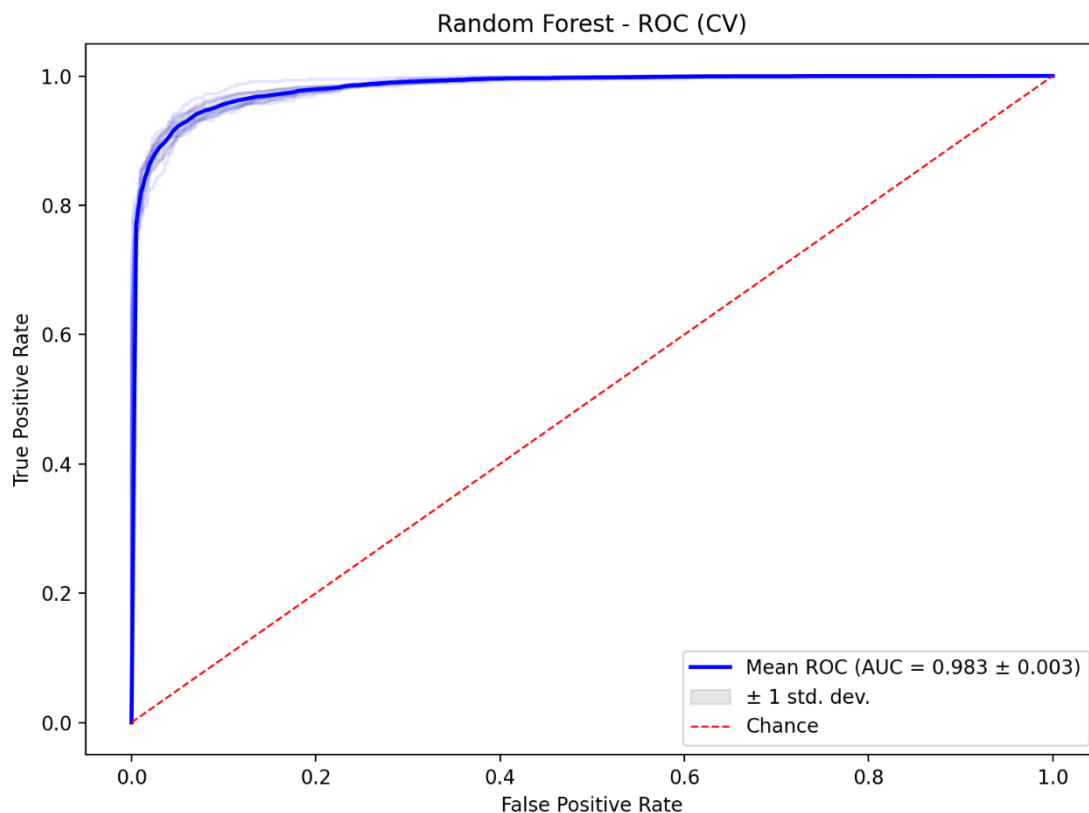
Random Forest

Ensemble di alberi decisionali che riduce la varianza e il rischio di overfitting.

Iperparametri migliori: $n_estimators = 100$, $max_depth = 20$, $min_samples_leaf = 1$.

Performance: Si è confermato il modello migliore in assoluto. Il mantenimento di performance eccellenti su dati nuovi conferma che il modello ha appreso con successo sia le relazioni nutrizionali di base sia i pattern complessi suggeriti dall'arricchimento semantico.

- ROC-AUC (Cross-Validation): $0.983 (\pm 0.003)$.
- ROC-AUC (Test Set): 0.9823.



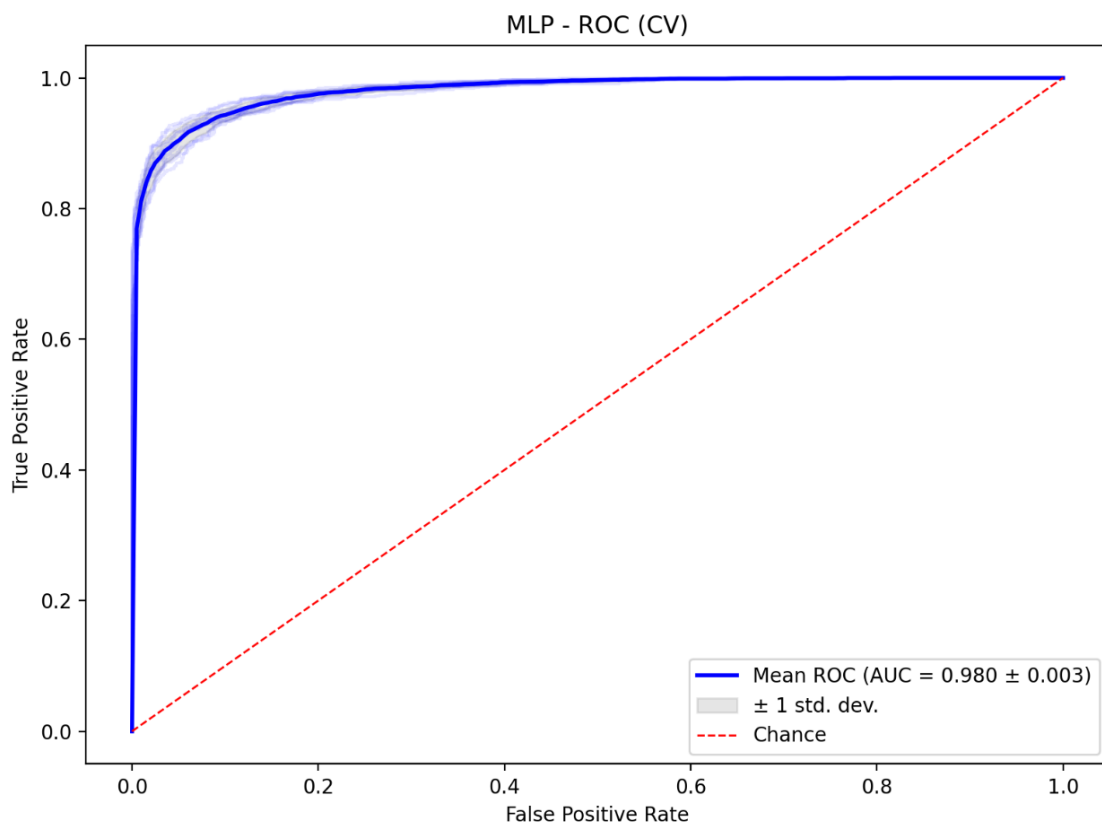
Multi-Layer Perceptron (MLP)

Rete neurale feed-forward.

Iperparametri migliori: *hidden_layer_sizes = (64, 32), alpha = 0.0001.*

Performance: L'alta precisione dimostra che le reti neurali riescono a mappare efficacemente la non-linearità del problema nutrizionale.

- ROC-AUC (Cross-Validation): 0.980 ± 0.003 .
- ROC-AUC (Test Set): 0.9767.



5.3 Confronto Finale

Modello	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	0.9359 (±0.0050)	0.9491 (±0.0075)	0.9213 (±0.0089)	0.9349 (±0.0052)	0.9841 (±0.0016)
MLP	0.9281 (±0.0051)	0.9457 (±0.0123)	0.9088 (±0.0130)	0.9267 (±0.0053)	0.9813 (±0.0016)
Decision Tree	0.9110 (±0.0085)	0.9313 (±0.0085)	0.8875 (±0.0157)	0.9088 (±0.0091)	0.9598 (±0.0043)
KNN	0.9137 (±0.0048)	0.9366 (±0.0075)	0.8876 (±0.0081)	0.9114 (±0.0049)	0.9668 (±0.0019)
Logistic Regression	0.9039 (±0.0043)	0.9490 (±0.0077)	0.8539 (±0.0120)	0.8988 (±0.0051)	0.9570 (±0.0031)

Il confronto decreta il Random Forest come modello migliore, capace di superare la baseline lineare e il singolo albero decisionale in tutte le metriche grazie alla robustezza dell'approccio *Ensemble*. Degna di nota è anche la rete neurale (MLP), che registra la precisione media più alta, confermando l'efficacia nel modellare relazioni non lineari. Infine, la bassissima deviazione standard (< 0.01) trasversale a tutti gli esperimenti certifica l'estrema stabilità dell'architettura neuro-simbolica, garantendo predizioni affidabili indipendentemente dal partizionamento dei dati.

6. Sviluppi futuri

Un'evoluzione naturale del sistema riguarda l'introduzione della personalizzazione. Attualmente, il motore Prolog valuta la qualità del prodotto in senso assoluto, ma in futuro la Knowledge Base potrebbe essere estesa per gestire profili utente dinamici (ad esempio per atleti o persone con esigenze dietetiche specifiche), adattando le regole di inferenza alle necessità individuali. Sul fronte pratico, l'applicazione potrebbe essere ulteriormente potenziata integrando un modulo di riconoscimento ottico (OCR) capace di estrarre i valori nutrizionali direttamente dalla fotografia dell'etichetta, rendendo lo strumento utilizzabile in tempo reale durante l'acquisto.

7. Bibliografia

- [1] Bratko, I. (2012). *Prolog Programming for Artificial Intelligence*. Addison-Wesley.
- [2] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
- [3] Besold, T. R., et al. (2017). *Neural-Symbolic Learning and Reasoning: A Survey and Interpretation*. arXiv preprint.
- [4] OpenFoodFacts Contributors. *OpenFoodFacts Database*. Disponibile su: <https://world.openfoodfacts.org/> .
- [5] Santé Publique France. *Nutri-Score: Scientific and Technical Guiding Principles*.