Documentazione Completa del Progetto SmartFoodSelector

Tuo Nome

22 dicembre 2024

Sommario

In questo documento viene illustrata in dettaglio l'architettura del sistema **SmartFoodSelector**, che combina tecniche di Clustering, Apprendimento Supervisionato, Reti Bayesiane (sia con valori continui sia discretizzati) e l'utilizzo di Prolog per la logica formale. Vengono inoltre riportati i principali log di esecuzione, i risultati sperimentali e numerosi grafici per dimostrare l'efficacia dell'approccio proposto.

Indice

1 Introduzione Generale **2**

2 Log di Esecuzione **2**

3 Preprocessing dei Dati **3**
3.1 Selezione e Pulizia delle Feature 3

4 Clustering con k-Means **3**
4.1 Metodo del Gomito 3
4.2 Assegnazione e Distribuzione dei Cluster 3

5 Apprendimento Supervisionato **3**
5.1 Modelli Utilizzati 3
5.2 Bilanciamento e Cross-validation 4
5.3 Confronto delle Metriche 4
5.4 Curve di Apprendimento 5

6 Reti Bayesiane **5**
6.1 Rete Bayesiana con Valori Continui 5
6.2 Rete Bayesiana con Valori Discreti 6
6.3 Osservazioni 6

7 Interfaccia Prolog **7**
7.1 Generazione della Knowledge Base 7
7.2 Esempio di Query 7

8 Conclusioni e Sviluppi Futuri **7**
8.1 Possibili Estensioni 8

1 Introduzione Generale

SmartFoodSelector è stato progettato per analizzare e categorizzare i prodotti alimentari attraverso l'utilizzo di tecniche avanzate di *machine learning* e *inferenza probabilistica*. Il sistema si compone di diverse fasi:

1. **Preprocessing dei Dati**: pulizia, selezione delle feature ed eventuale normalizzazione.
2. **Clustering** (*k-Means*) per identificare gruppi di prodotti con caratteristiche nutrizionali simili.
3. **Apprendimento Supervisionato** per classificare automaticamente nuovi prodotti.
4. **Reti Bayesiane** per modellare le relazioni probabilistiche tra le variabili in forma sia continua sia discreta.
5. **Interfaccia Prolog** per la rappresentazione della conoscenza e l'esecuzione di query logiche.

L'obiettivo finale è fornire uno strumento flessibile e robusto per assistere gli utenti (tecnici e non) nella valutazione nutrizionale e nella categorizzazione di prodotti alimentari, garantendo al contempo la possibilità di gestire e inferire informazioni mancanti o incertamente note.

2 Log di Esecuzione

Di seguito uno stralcio del log di esecuzione prodotto dal comando:

```
C:\Users\gianluca\Progetti\bot\.venv\Scripts\python.exe main.py

Inizio del programma...
1) Preprocessing dei dati...
Preprocessing completato, dati salvati in: data/newDataset.csv
Preprocessing completato in 21.82 secondi.

2) Clustering dei dati...
Assegnazione cluster completata, salvato in: data/clustered_dataset.csv
Clustering completato in 3.01 secondi.

3) Training e valutazione dei modelli...
Caricati 26032 campioni su 260316 totali
[...omissis per brevità...]

Cross-validation per il modello Decision Tree...
F1 scores: [0.99613294 0.99613209 0.9980666  0.99765264 0.99903307]
Mean F1 score: 0.9974
Deviazione Standard: 0.001130
Varianza: 0.000001
```

```
Generazione della curva di apprendimento per Decision Tree...
Grafico salvato in: data/learning_curve_decision_tree.png

[...simile per Random Forest e Logistic Regression...]

4) Creazione della rete bayesiana...
[...Bayesian Network continua e discretizzata...]
Programma completato in 89.19 secondi.
Process finished with exit code 0
```

3 Preprocessing dei Dati

3.1 Selezione e Pulizia delle Feature

Il dataset originario (`en.openfoodfacts.org.products.tsv`) contiene un ampio numero di campi; le feature ritenute più rilevanti per l'analisi nutrizionale sono state:

$$\{\text{energy\_100g}, \text{fat\_100g}, \text{carbohydrates\_100g}, \text{sugars\_100g}, \text{proteins\_100g}, \text{salt\_100g}\}.$$

Dopo l'identificazione di record con valori errati o mancanti, si è proceduto alla loro rimozione o sostituzione (quando possibile). Successivamente, si è applicato il *MinMax-Scaler* per normalizzare ogni feature nel range [0,1]. Il risultato finale è stato salvato in `data/newDataset.csv`.

4 Clustering con k-Means

4.1 Metodo del Gomito

Per individuare il *numero ottimale di cluster*, si è valutata l'*inertia* (somma delle distanze quadrate tra i campioni e il centro del cluster) al variare di k. Dall'analisi è emerso che $k = 3$ garantisce un buon trade-off tra complessità e accuratezza.

4.2 Assegnazione e Distribuzione dei Cluster

Una volta impostato $k = 3$, l'algoritmo *k-Means* ha prodotto il file `data/clustered_dataset.csv` con l'assegnazione di ogni prodotto al rispettivo cluster.

Nella Figura 1 è riportata la *distribuzione dei cluster* sotto forma di grafico a torta.

5 Apprendimento Supervisionato

5.1 Modelli Utilizzati

Sono stati sviluppati tre modelli principali per assegnare un nuovo prodotto al cluster corretto:

- **Decision Tree**: facilmente interpretabile, ottime prestazioni (F1 medio ≈ 0.9974).

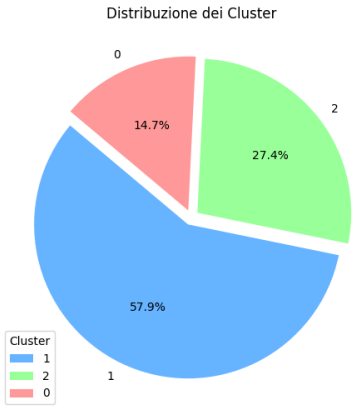

Figura 1: Distribuzione dei prodotti nei cluster 0, 1 e 2.

- **Random Forest**: ensemble di alberi, risultati paragonabili a Decision Tree (F1 medio ≈ 0.9974).
- **Logistic Regression**: modello lineare, leggermente inferiore (F1 medio ≈ 0.9902).

5.2 Bilanciamento e Cross-validation

Per mitigare eventuali squilibri nelle classi, è stato impiegato **SMOTE** (Synthetic Minority Over-sampling Technique). La valutazione è avvenuta tramite **5-Fold Cross-validation**, calcolando *Accuracy*, *F1 Score*, *Precision* e *Recall*.

5.3 Confronto delle Metriche

La Figura 2 e la Figura 3 mostrano un confronto tra i tre modelli. Tutti i valori metrici risultano molto alti, confermando l'eccellente qualità delle predizioni.

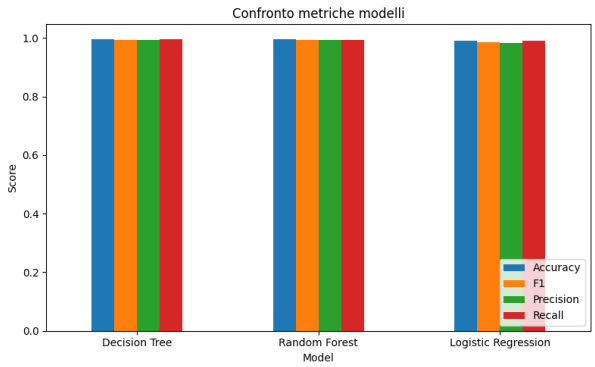

Figura 2: Confronto (Accuracy, F1, Precision, Recall) per Decision Tree, Random Forest e Logistic Regression.

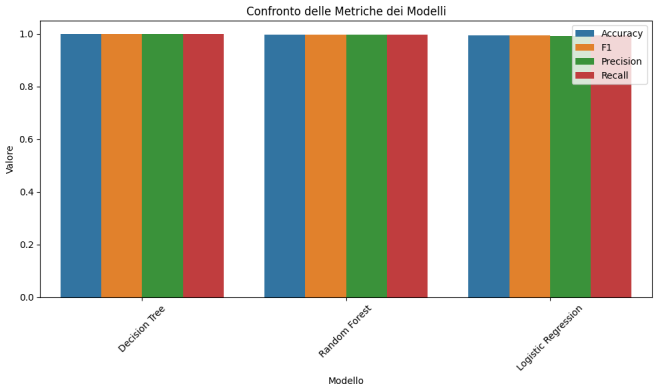

Figura 3: Versione alternativa del confronto delle metriche.

5.4 Curve di Apprendimento

Per valutare la *capacità di generalizzazione* dei modelli, sono state generate **Learning Curves** in funzione della dimensione del training set.

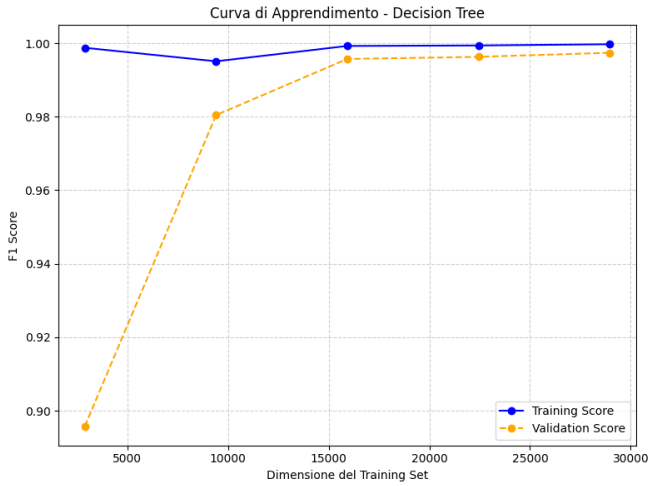

Figura 4: Curva di apprendimento per **Decision Tree**: training vs validation (*F1 Score*).

Dai grafici emerge che Decision Tree e Random Forest si assestano su prestazioni molto elevate già con un numero relativamente contenuto di campioni, mentre Logistic Regression continua a migliorare in modo più graduale, raggiungendo comunque ottime performance.

6 Reti Bayesiane

6.1 Rete Bayesiana con Valori Continui

Per la rete bayesiana con valori *continui*, è stata sfruttata l'informazione numerica diretta (senza discretizzazione). La struttura è stata appresa tramite `HillClimbSearch` con `BicScore`. In Figura 7 si può osservare il grafo risultante, dove i nodi rappresentano le variabili principali (ad esempio `energy_100g`, `carbohydrates_100g`, `proteins_100g` e così via), mentre le frecce indicano relazioni di dipendenza.

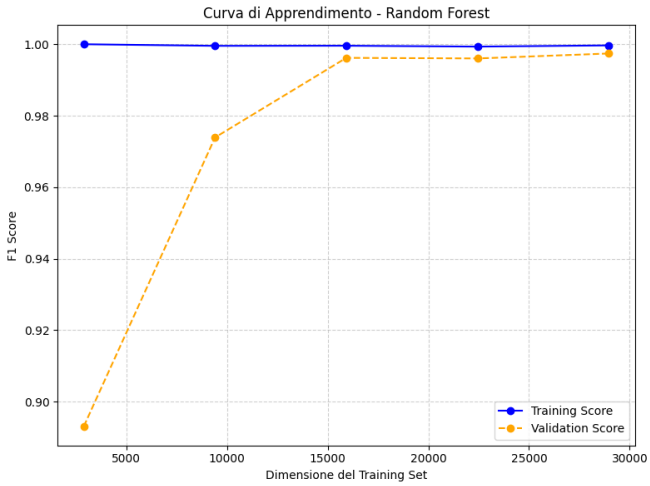

Figura 5: Curva di apprendimento per **Random Forest**.

Figura 6: Curva di apprendimento per **Logistic Regression**.

6.2 Rete Bayesiana con Valori Discreti

Parallelamente, si è realizzata una seconda rete *discreta*, discretizzando le variabili in 5 intervalli (*bins*) tramite `KBinsDiscretizer` (strategia `uniform`). Anche in questo caso, la struttura è stata ottimizzata con `HillClimbSearch` e `BicScore`.

La Figura 8 illustra la topologia ottenuta, che, pur presentando simili relazioni tra le variabili, organizza i dati in categorie invece che trattarli come continui.

6.3 Osservazioni

Le reti bayesiane così apprese consentono di:

- Effettuare **inferenza probabilistica** (es. query del tipo $P(\text{cluster} \mid \text{energy\_100g} = 1)$).
- Calcolare **probabilità congiunte** o predire la probabilità di un certo valore nutrizionale, date altre variabili.

Figura 7: Rete bayesiana appresa con valori **continui**: strutture di dipendenza rilevate nei dati.

- Gestire **dati mancanti** completando automaticamente valori ignoti sulla base delle dipendenze statistiche apprese.

7 Interfaccia Prolog

7.1 Generazione della Knowledge Base

A partire dal file `clustered_dataset.csv`, sono stati generati automaticamente *fatti* e *regole* (archiviati in `knowledge_base.pl`) per gestire query logiche. Un esempio di regola:

```
product_info(E, F, C, Su, P, Sa, Cl) :-
    product(E, F, C, Su, P, Sa),
    clustered_product(E, F, C, Su, P, Sa, Cl).
```

7.2 Esempio di Query

```
?- product_info(0.2, 0.05, 0.3, 0.25, 0.1, 0.02, Cluster).
```

Tale query identifica il `cluster` di appartenenza di un prodotto caratterizzato dai valori nutrizionali `energy_100g=0.2`, `fat_100g=0.05`, ecc.

8 Conclusioni e Sviluppi Futuri

SmartFoodSelector evidenzia come algoritmi di *machine learning* classico (clustering e classificazione), *inferenza probabilistica* (reti bayesiane) e *logica formale* (Prolog) possano interagire per fornire un sistema di supporto decisionale nel dominio alimentare.

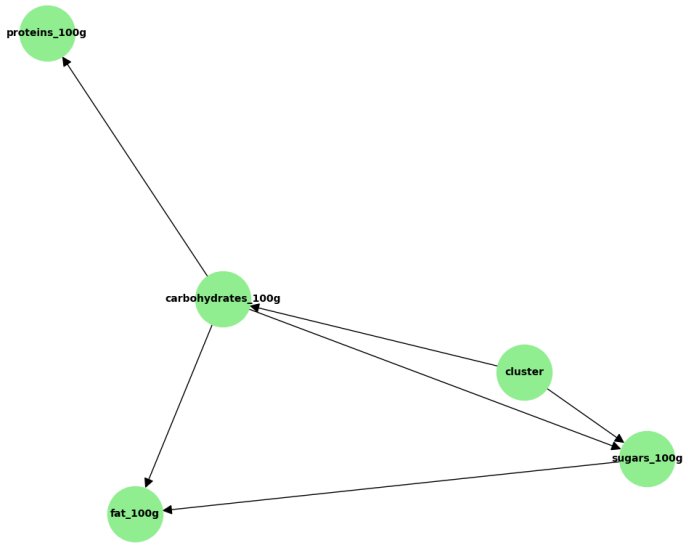

Figura 8: Rete bayesiana con valori **discretizzati** in 5 bin. Si notino le relazioni tra `cluster`, `fat_100g`, `sugars_100g` e altre variabili.

8.1 Possibili Estensioni

1. **Integrazione con API di terze parti**: es. piattaforme e-commerce per caricare e aggiornare in tempo reale nuovi prodotti.
2. **Interfaccia Grafica Avanzata**: dashboard intuitiva per utenti non tecnici, con filtri e suggerimenti personalizzati.
3. **Arricchimento del Dataset**: inserimento di ulteriori informazioni (allergeni, certificazioni biologiche, ingredienti, ecc.).
4. **Raccomandazioni Personalizzate**: estensione verso un sistema di *recommender* che suggerisca prodotti in base a esigenze dietetiche o preferenze dell'utente.

L'approccio proposto può costituire la base per futuri sviluppi in ambito nutrizionale e di e-Health, favorendo un uso consapevole dei dati e migliorando la qualità delle scelte alimentari.

Fine del Documento.