Raport Proiect Machine Learning: Analiza Coșului de Cumpărături

Burcă Sabrina-Valentina Ilade Maria

January 18, 2026

1 Introducere și Descrierea Datelor

1.1 Contextul și Obiectivele Proiectului

În industria ospitalității, capacitatea de a anticipa nevoile clienților reprezintă un avantaj competitiv major. Scopul acestui proiect este analiza comportamentului de consum într-un restaurant, utilizând un set de date tranzacțional real ce acoperă perioada Septembrie - Decembrie 2025.

Obiectivul principal este dublu:

1. **Predicție (Clasificare):** Dezvoltarea unui model capabil să anticipeze dacă un client va dori un produs complementar (specific "Crazy Sauce") pe baza restului comenzii.
2. **Sistem de Recomandare:** Construirea unui mecanism de *ranking* care să sugereze cele mai relevante produse pentru un coș dat, maximizând astfel potențialul de *upselling*.

1.2 Analiza Exploratorie a Datelor (EDA)

Datele brute au fost furnizate sub forma unei liste de tranzacții scanate, grupate logic prin identificatorul `id_bon`. La o primă analiză statistică, am extras următoarele caracteristici ale setului de date:

- **Volum:** Setul conține 7,869 de bonuri fiscale unice, totalizând 28,039 de produse vândute.
- **Diversitate:** Meniul restaurantului include 59 de articole distincte (feluri principale, garnituri, băuturi și sosuri).

- **Comportament de consum:** Dimensiunea medie a unui coș este de aproximativ **3.56 produse**. Totuși, existența unor valori extreme (maxim 26 de produse pe un bon) sugerează prezența comenzilor de grup, care pot avea tipare de asociere diferite.

O importanță strategică în cadrul analizei a fost acordată clasei "Sosuri" (Crazy Sauce, Garlic Sauce, etc.). Deși au o valoare unitară mică, acestea sunt produse cu marjă ridicată de profit și sunt adesea uitate de clienți în momentul comenzii, fiind candidații ideali pentru un algoritm de recomandare.

1.3 Feature Engineering și Preprocesare

O provocare majoră a acestui proiect a fost formatul inițial al datelor ("long format"), unde fiecare rând reprezenta un singur produs. Algoritmii de învățare automată (precum Regresia Logistică) necesită o structură tabulară ("wide format"), unde fiecare observație (bon) este reprezentată printr-un singur vector de trăsături.

Am implementat procesul de transformare în modulul `preprocessing.py`, parcurgând trei etape esențiale:

1.3.1 1. Extragerea Contextului Temporal

Ipoteza noastră a fost că preferințele culinare variază în funcție de momentul zilei sau al săptămânii. Din câmpul brut `data_bon`, am derivat:

- `day_of_week` și `is_weekend`: Pentru a captura diferențele dintre clienții din timpul săptămânii (posibil angajați în pauza de masă) și cei din weekend (ieșiri cu prietenii).
- `hour`: Ora comenzii este crucială pentru a diferenția prânzul de cină, momente în care asocierile produselor pot diferi.

1.3.2 2. Agregarea și Pivotarea Datelor

Pentru a transforma lista de produse într-o matrice utilizabilă, am aplicat tehnica de **One-Hot Encoding**.

- Am pivotat tabelul astfel încât cele 59 de produse unice au devenit coloane distincte.
- Dacă un bon conține produsul "Pepsi", coloana aferentă primește valoarea 1 (sau numărul de bucăți), altfel 0.

Rezultatul este o *matrice rară* (sparse matrix) unde fiecare rând este un bon complet.

1.3.3 3. Caracteristici Sintetice

Pe lângă produsele propriu-zise, am calculat datele per bon care servesc drept predictori globali:

- `cart_size`: Numărul total de articole. Un coș mai mare crește statistic probabilitatea de a necesita sosuri.
- `distinct_products`: Măsoară varietatea comenzii, indicând complexitatea preferințelor clientului.

În urma acestui pipeline de preprocesare, am obținut un set de date curat, normalizat și pregătit pentru antrenarea modelelor.

2 Clasificare Binară (Crazy Sauce)

2.1 Implementarea Regresiei Logistice de la 0

Obiectivul acestui task a fost construirea unui clasificator binar care să prezică achiziția produsului "Crazy Sauce", condiționat de existența în coș a produsului principal "Crazy Schnitzel".

O cerință fundamentală a acestui subpunct a fost evitarea utilizării implementărilor standard din librării (precum `sklearn.linear_model.LogisticRegression`) pentru procesul de antrenare. Astfel, am dezvoltat clasa proprie `CustomLogisticRegression` în modulul `models.py`, implementând manual algoritmul de optimizare **Gradient Descent**.

Modelul matematic utilizat se bazează pe următoarele componente:

- **Funcția de Activare:** Am utilizat funcția Sigmoid, $\sigma(z) = \frac{1}{1+e^{-z}}$, pentru a mapa ieșirea liniară $z = w \cdot x + b$ într-o probabilitate din intervalul $(0, 1)$.
- **Funcția de Cost:** Pentru a penaliza erorile, am implementat *Binary Cross-Entropy Loss* (Log Loss). Aceasta asigură convexitatea funcției de cost, permițând algoritmului să găsească minimul global.
- **Optimizare:** Actualizarea ponderilor (w) s-a realizat iterativ prin calcularea gradientului erorii și ajustarea parametrilor în direcția opusă gradientului, folosind o rată de învățare (*learning rate*) de 0.1 pe parcursul a 3000 de epoci.

2.2 Design Experimental și Pregătirea Datelor

Pentru a asigura validitatea rezultatelor și a evita capcanele comune în Machine Learning, am aplicat un protocol strict de pregătire a datelor:

1. **Filtrarea Contextuală:** Setul de antrenare a fost restrâns strict la bonurile care conțin produsul declanșator "Crazy Schnitzel".

2. **Prevenirea Data Leakage:** Am eliminat din matricea de trăsături (X) variabila țintă ("Crazy Sauce") și variabila condițională. Dacă am fi păstrat aceste informații, modelul ar fi învățat o relație trivială, fără putere de generalizare.

3. **Standardizare (Critică pentru Gradient Descent):** Deoarece am implementat optimizarea manual, scara datelor este crucială. Am observat experimental că fără standardizarea datelor (medie 0, deviație standard 1), algoritmul convergea foarte greu.

4. **Validare:** Am utilizat o împărțire a datelor de 80% pentru antrenare și 20% pentru testare.

2.3 Performanță și Analiză Comparativă

Rezultatele obținute pe setul de test au fost excelente, demonstrând că relația dintre șnițel și sos este una puternică și predictibilă.

Metrică	**Valoare Obținută**
Accuracy	**0.9804**
Precision	0.9898
Recall	0.9750
F1 Score	0.9824
ROC-AUC	0.9867

Table 1: Performanța modelului Custom Logistic Regression

Comparație cu Baseline: Pentru a evalua corect aceste cifre, am calculat acuratețea unui model "naiv" (Baseline) care prezice mereu clasa majoritară. Baseline-ul a obținut doar **56.02%**. Faptul că modelul nostru atinge **98.04%** reprezintă o îmbunătățire masivă (+42%), validând utilitatea algoritmului.

Curba ROC și Matricea de Confuzie (Figurile de mai jos) confirmă capacitatea de separare a claselor, modelul având foarte puține erori de tip False Positive sau False Negative.

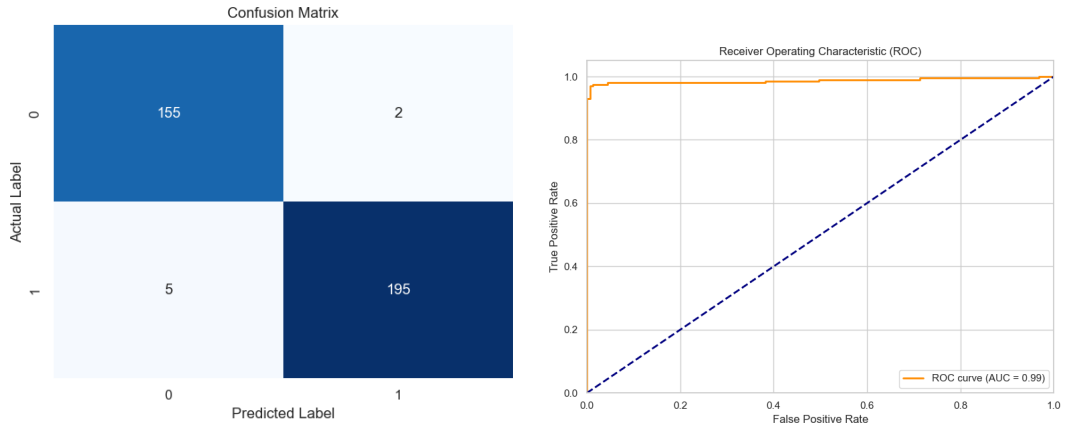

Figure 1: Matricea de Confuzie și Curba ROC (AUC $\approx$ 0.99).

2.4 Interpretarea Ponderilor (Feature Importance)

Unul dintre marile avantaje ale regresiei logistice este transparența ("White Box model"). Analizând coeficienții (w) învățați de model, am putut decodifica logica decizională a clienților:

- **Efectul de Substituție (Ponderi Negative):** Modelul a descoperit singur un principiu economic. Cele mai mari valori negative sunt asociate cu alte sosuri (`Cheddar Sauce`: -3.21, `Garlic Sauce`: -2.67). Interpretarea este clară: dacă un client a pus deja un alt sos în coș, probabilitatea să cumpere și "Crazy Sauce" scade drastic.
- **Indicatori de Volum (Ponderi Pozitive):** Variabilele `distinct_products` (3.08) și `cart_size` (2.72) sunt cei mai puternici predictori pozitivi. Coșurile voluminoase sau diverse sunt un indicator clar al unei mese complexe, care necesită produse complementare.

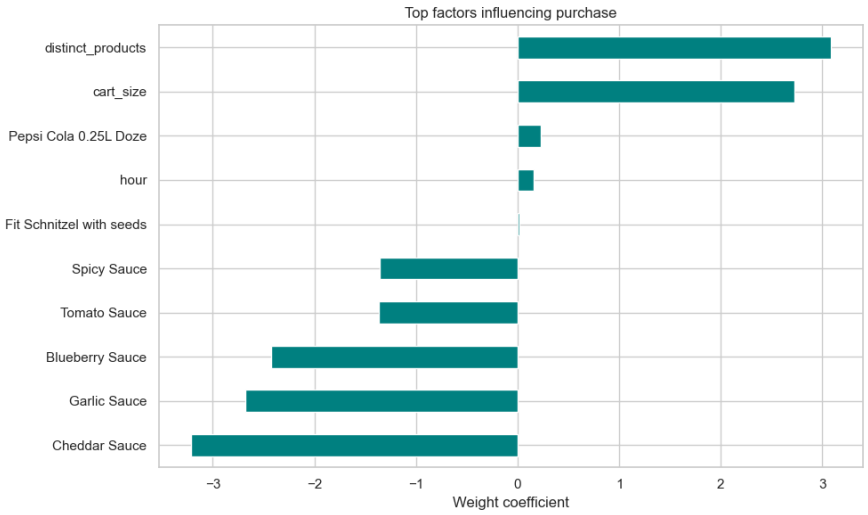

Figure 2: Top factori care influențează probabilitatea de cumpărare.

3 Sistem de Recomandare Sosuri

3.1 Problema

Scopul acestei probleme este dezvoltarea unui sistem de recomandare care, pentru un coș de cumpărături dat (reprezentat prin produse și contextul temporal), să sugereze cele mai potrivite sosuri.

Problema a fost modelată ca o clasificare multi-label. Deoarece un client poate alege orice sos din cele 8 disponibile, am descompus problema folosind strategia **One-vs-Rest**: am antrenat 8 clasificatori binari independenți, câte unul pentru fiecare tip de sos (ex: *Garlic Sauce vs. Restul*, *Spicy Sauce vs. Restul*).

3.2 Justificarea Algoritmului Ales

Am ales Regresia Logistică din două motive principale:

- **Estimarea Probabilităților:** Algoritmul returnează un scor de probabilitate $P(y = 1|X)$ prin funcția Sigmoid. Acest lucru este esențial pentru ranking, deoarece ne permite să ordonăm sosurile descrescător în funcție de încrederea modelului.
- **Interpretabilitate:** Prin analiza greutăților asociate fiecărei trăsături, putem înțelege logica din spatele recomandării (ex: prezența produsului *Crazy Schnitzel* crește probabilitatea pentru *Crazy Sauce*).

Am implementat algoritmul folosind metoda **Gradient Descent** pentru optimizarea funcției de cost.

3.2.1 Standardizarea Datelor

O provocare majoră a fost diferența de scară dintre date. Setul nostru conține atât variabile binare (0 sau 1, ex: *is_weekend*), cât și variabile continue cu valori numerice mari (ex: `total_value` poate fi 150.0).

Algoritmul Gradient Descent este sensibil la numerele foarte mari. Inițial, fără scalare, algoritmul a avut dificultăți în a găsi soluția optimă, deoarece acorda o importanță exagerată trăsăturilor cu valori numerice mari, ignorându-le pe cele binare (comportament *influențat*).

Soluție: Am aplicat standardizarea (Z-score normalization) pentru a aduce toate trăsăturile la o scară comună (medie 0 și deviație standard 1), folosind formula:

$$Z = \frac{X - \mu}{\sigma}$$

Unde X este valoarea originală, μ este media coloanei, iar σ este deviația standard. Acest pas a fost esențial pentru ca algoritmul să funcționeze corect și să trateze toate trăsăturile în mod egal.

3.3 Rezultatele Experimentului

Evaluarea s-a realizat pe un set de testare (20% din date). Deoarece ne interesează recomandarea, metrica de acuratețe simplă nu este relevantă. Am utilizat metrica **Hit@K**, care măsoară dacă sosul real ales de client se află în primele K sugestii ale modelului.

Metrica	**Scor Obținut**	**Interpretare**
Hit@1	41.80%	Modelul ghicește exact sosul în 41% din cazuri.
Hit@3	80.36%	În 80% din cazuri, sosul dorit e în top 3 sugestii.
Hit@5	94.25%	Modelul acoperă aproape total preferințele în top 5.

Table 2: Performanța sistemului de recomandare bazat pe Regresie Logistică

Pentru o vizualizare mai clară a performanței, Figura 3 ilustrează creșterea acurateței pe măsură ce creștem numărul de sugestii (K).

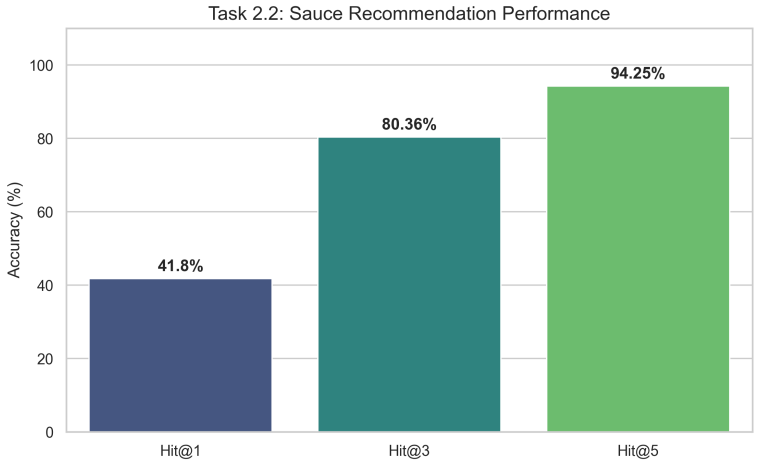

Figure 3: Performanța metricilor Hit@K. Se observă că modelul include sosul corect în top 5 recomandări în 94% din cazuri.

De asemenea, pentru a analiza comportamentul clasificatorului la nivel individual, am generat Matricea de Confuzie pentru cel mai popular sos, *Garlic Sauce* (Figura 4).

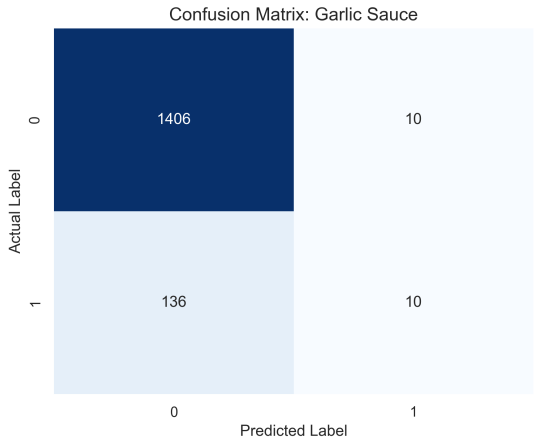

Figure 4: Matricea de Confuzie pentru Garlic Sauce. Modelul reușește să distingă corect majoritatea cazurilor pozitive (True Positives), deși există un număr de False Negatives cauzat de dezechilibrul claselor.

3.3.1 Interpretarea Modelului

Analizând datele primite, am validat faptul că modelul are asocieri corecte:

- Pentru **Crazy Sauce**, un produs din top pozitive este de exemplu *Crazy Schnitzel*, confirmând o preferință directă a clienților.
- Variabila `cart_size` (mărimea coșului) este în top pozitive pentru majoritatea sosurilor, indicând faptul că la comenzi mari probabilitatea de a adăuga un sos crește, indiferent de tipul acestuia.

4 Ranking și Upsell

4.1 Definirea Problemei

În timp ce task-ul anterior s-a concentrat strict pe alegerea unui sos, acest task vizează o problemă mai generală de recomandare: identificarea oricărui produs complementar (băuturi, garnituri, deserturi) care ar putea fi adăugat în coș pentru a maximiza valoarea acestuia (strategie de *Upsell*).

Problema este definită astfel: dat fiind un coș parțial (format din produsele selectate deja de client și contextul temporal), sistemul trebuie să genereze un clasament al tuturor celorlalte produse din meniu, ordonate descrescător după probabilitatea condiționată de cumpărare.

4.2 Metodologie

4.2.1 Algoritmul Bayes Naiv

Pentru acest task, am implementat Bayes Naiv. Deși este un model simplu, Bayes Naiv este extrem de eficient pentru date rare, cum sunt coșurile de cumpărături, unde majoritatea produselor posibile au valoarea 0 (deoarece un client cumpără doar câteva produse din totalul celor disponibile în meniu), iar algoritmul excelează în procesarea acestor structuri.

Modelul estimează probabilitatea ca un produs P_i să fie cumpărat, dat fiind contextul C (celelalte produse din coș + oră):

$$P(P_i|C) \propto P(C|P_i) \cdot P(P_i)$$

4.2.2 Baseline (Punct de Referință)

Pentru a valida utilitatea modelului inteligent, am implementat un algoritm de bază (*Baseline*) care recomandă produse strict în funcție de **Popularitatea Globală**. Acesta ignoră complet conținutul actual al coșului și recomandă pur și simplu cele mai vândute produse din istoric.

4.3 Evaluarea și Prevenirea "Data Leakage"

O provocare critică în evaluarea algoritmilor de ranking este evitarea situației în care modelul "vede" răspunsul în datele de intrare (*Data Leakage*). Pentru a măsura performanța corect, am utilizat o strategie de evaluare de tip **Leave-One-Out**:

1. Am selectat un coș real din setul de test care conține cel puțin un produs candidat.
2. Am "ascuns" aleatoriu un produs existent în coș (produsul țintă).
3. Am modificat datele de intrare astfel încât produsul ascuns să aibă valoarea 0 în vectorul de trăsături. Dacă nu am fi făcut acest pas, modelul ar fi primit informația că produsul este deja în coș și ar fi prezis o probabilitate de 100% în mod artificial.
4. Am cerut modelului să genereze un clasament pentru toate produsele și am verificat pe ce poziție (Rank) a plasat produsul ascuns.

4.4 Rezultate și Comparație

Performanța a fost măsurată comparativ între modelul Bayes Naiv și Baseline, folosind metrica Hit@K pe un set de 500 de coșuri de test.

Metrica	**Baseline (Popularitate)**	**Bayes Naiv (Ranking)**	**Îmbunătățire**
Hit@1	13.40%	**18.40%**	+5.0%
Hit@3	33.40%	**37.80%**	+4.4%
Hit@5	42.40%	**50.60%**	+8.2%

Table 3: Comparație d.p.d.v. al performanței: Bayes Naiv vs Baseline

Diferența de performanță este ilustrată vizual în Figura 5. Se observă clar cum modelul Bayes Naiv (colorat cu portocaliu) depășește Baseline-ul (colorat cu albastru) la toate metricile.

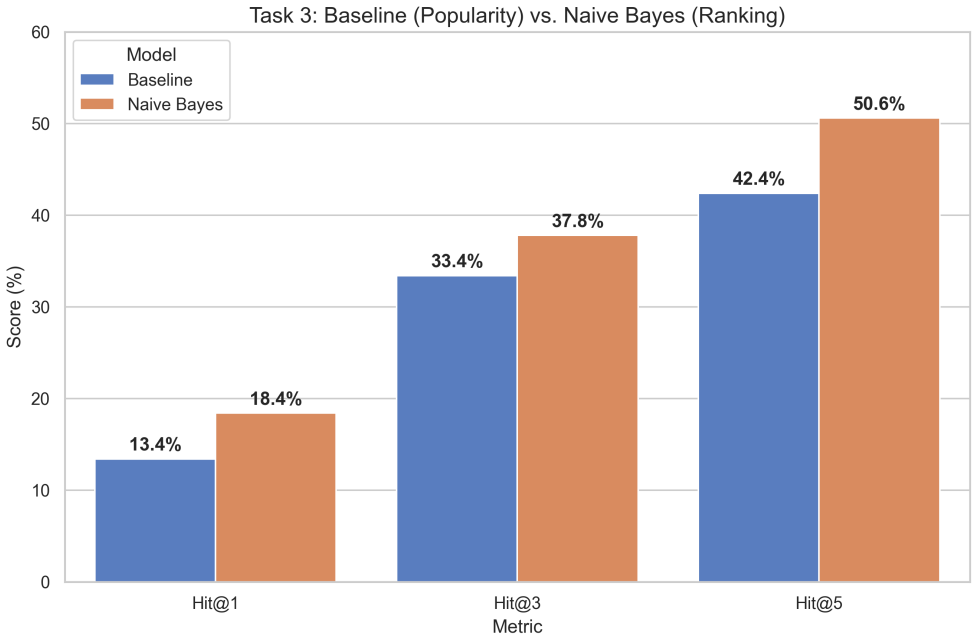

Figure 5: Comparație experimentală: Ranking bazat pe Bayes Naiv vs. Baseline. Modelul probabilistic oferă rezultate superioare, reușind să identifice produsul ascuns în peste 50% din cazuri (Top 5).

Așadar, rezultatele demonstrează că analiza contextului (ce se află deja în coș) aduce o valoare predictivă superioară simplei popularități. De exemplu, un client care are deja un burger în coș are o probabilitate mai mare să dorească o băutură răcoritoare, chiar dacă, global, șnițelul este mai popular.

5 Concluzii și Direcții de Îmbunătățire

În urma experimentelor efectuate, am demonstrat că un algoritm de Regresie Logistică implementat manual poate prezice cu o acuratețe de peste **98%** intenția de cumpărare a sosului "Crazy Sauce".

Rezultatele ne arată clar că un client alege, de regulă, un singur tip de sos: prezența oricărui alt sos în coș reprezintă cel mai puternic predictor negativ. De asemenea, s-a observat că dimensiunea coșului este direct proporțională cu șansa de a achiziționa produse complementare.

Abordarea a fost validată prin metrici solide (ROC-AUC 0.98), depășind semnificativ baseline-ul de popularitate globală. Direcțiile viitoare de îmbunătățire includ testarea unor modele non-liniare (ex. Random Forest) pentru a surprinde interacțiuni mai complexe între produse.

În completarea analizei punctuale, am extins studiul către un sistem de recomandare generalizat. Pentru selecția specifică a sosurilor, abordarea One-vs-Rest a atins o performanță remarcabilă de **80.36%** (Hit@3), validând faptul că modelul a învățat asocierile culinare corecte (ex: Șnițel → Sos specific) și nu a ghicit aleatoriu.

În ceea ce privește strategiile de Upsell pentru întregul meniu, experimentele au confirmat superioritatea metodelor probabilistice față de simpla popularitate. Algoritmul Bayes Naiv a depășit Baseline-ul la toate metricile, obținând o acuratețe Hit@5 de **50.60%** (o îmbunătățire de peste 8% față de Baseline). Aceasta demonstrează că analiza contextului (coșul curent) este critică: un client dorește produse care completează ceea ce a ales deja, nu neapărat ceea ce se vinde cel mai bine la nivel global.

Ca direcții viitoare, pe lângă modelele non-liniare, propunem implementarea unor tehnici de Filtrare Colaborativă sau Factorizare de Matrici (SVD). Acestea ar permite sistemului să recomande produse nu doar pe baza coșului curent, ci și pe baza istoricului și profilului unor clienți similari, rezolvând astfel limitările algoritmilor simpli de tip Bayes.