Tab 1

**Proiect inferenta predicativa prin rationament inainte**

**Cuprins**

[**1. Descrierea problemei considerate 1**](#_is1kovnoremy)

[**2. Aspecte teoretice privind algoritmul 2**](#_fw2vykprc2te)

[**3. Modalitatea de rezolvare 2**](#_46qt1g7mgf95)

[**4. Analiza complexitatii algoritmilor 3**](#_ok3qjqc0o0sv)

[**5. Parti semnificative din codul sursa 3**](#_2hjeec3dpizu)

[**6. Rezultatele obtinute 8**](#_isju7heqzk9i)

[**7. Teste automate 10**](#_74dy485zr7ae)

[**8. Concluzii 10**](#_vistpzhh3cgk)

[**9. Bibliografie 11**](#_j86f6ut80424)

[**9. Lista contributiilor 11**](#_k0nol69flbhq)

# 1. Descrierea problemei considerate

Proiectul abordat se concentreaza pe aplicarea inferentei predicative prin rationament inainte pentru a demonstra doua teoreme cu relevanta matematica. In contextul inteligentei artificiale, acest tip de rationament este folosit pentru a deriva concluzii logice plecand de la un set de premise initiale.

Problemele selectate pentru demonstratie sunt:

1. Dezvoltarea binomiala a expresiei “(a + b) ^ n” utilizand Teorema binomiala a lui Newton.
2. Crearea unui sistem de recomandari bazat pe istoricul de achizitii al unui utilizator pentru un magazin online.

# 2. Aspecte teoretice privind algoritmul

Inferenta predicativa prin rationament inainte presupune utilizarea unui set de reguli logice pentru a deriva concluzii din premise. Algoritmul urmeaza pasii urmatori:

1. Se identifica premisele de intrare.
2. Se verifica regulile aplicabile fiecarei premise.
3. Se genereaza concluzii intermediare pana cand se ajunge la scopul final sau la o solutie satisfacatoare.

**Teorema Binomiala a lui Newton:** Aceasta afirma ca orice expresie de forma “(a + b) ^ n” poate fi dezvoltata ca o suma ponderata de termeni, unde coeficientii binomiali sunt calculati folosind formula:

C(n, k) = n! / [k!(n-k)!]

**Sistemul de Recomandari:** Un sistem de recomandari care utilizeaza istoricul utilizatorului pentru a anticipa interesele viitoare, bazandu-se pe principii precum:

* Categoriile preferate de utilizator.
* Gama de preturi a achizitiilor anterioare.
* Similaritatea dintre produsele achizitionate si cele disponibile.

# 3. Modalitatea de rezolvare

**Pentru teorema binomiala:**

* Implementarea formulei generale pentru dezvoltarea binomiala folosind Python.
* Calcularea coeficientilor binomiali pentru fiecare termen.
* Generarea termenilor expansiunii, afisarea rezultatelor intermediare si a rezultatului final.

**Pentru sistemul de recomandari:**

* Crearea de utilizatori si adaugarea istoricului de cumparare.
* Salvarea si citirea datelor despre utilizatori si produse din fisiere JSON.
* Analizarea categoriilor preferate si a intervalului de preturi.
* Selectarea produselor relevante utilizand o metoda de filtrare bazata pe reguli simple.

**Reguli folosite pentru recomandare:**

def matches\_category(user, product):

return product["category"] in user["categories"]

def matches\_price\_range(user, product):

return user["price\_range"][0] <= product["price"] <= user["price\_range"][1]

def not\_purchased\_before(user, product):

return product["name"] not in user["purchase\_history"]

# 4. Analiza complexitatii algoritmilor

**Teorema binomiala:**

* **Caz favorabil:** n = 1( cazul de baza).

**Complexitate:** O(1)

* **Caz defavorabil:** n este mare si toti termenii trebuie calculati

**Complexitate:** O(n^2) (n coeficienti sunt calculati in n pasi)

**Sistemul de recomandari:**

* **Caz favorabil:** Toate produsele sunt excluse rapid

**Complexitate:** O(p), unde p este numarul de produse

* **Caz defavorabil:** Toate produsele trebuie verificate complet(verifica toate regulile)

**Complexitate:** O(p \* c), unde c este numarul de reguli

# 

# 5. Parti semnificative din codul sursa

**Teorema binomiala:**

Aceasta este premisa si formula generala a functiei **newton\_binomial** ce va fi apelata pentru dezvoltarea binomului lui Newton:

def newton\_binomial(a, b, n):

steps = []

steps.append(f"Premisa: Formula generala pentru Binomul lui Newton este:")

steps.append(f"(a + b)^n = ∑(k=0 pana la n) [C(n, k) \* a^(n-k) \* b^k]\n")

Secventa de cod de mai jos acopera cazul de baza (in care valoarea puterii (n) este 1), in care codul prezinta formula simplificata “(a+b) ^ 1” si returneaza rezultatul expanded si numeric:

if n == 1:

steps.append("Caz de baza: Pentru n=1, formula devine:")

steps.append("(a + b)^1 = a + b")

expanded\_result = f"{a} + {b}"

numeric\_result = float(a) + float(b)

return steps, expanded\_result, numeric\_result

Calculul termenilor binomiali pentru celelalte cazuri posibile (n > 1)

expanded\_terms = []

numeric\_terms = []

for k in range(n + 1):

coef = binomial\_coefficient(n, k)

steps.append(f"Pasul {k + 1}: Calculam termenul pentru k={k}.")

steps.append(f"Coeficientul binomial C({n}, {k}) = {coef}")

expanded\_term = f"{coef} \* {a}^{n - k} \* {b}^{k}"

expanded\_terms.append(expanded\_term)

steps.append(f"Termenul expandat este: {expanded\_term}")

numeric\_term = coef \* (float(a) \*\* (n - k)) \* (float(b) \*\* k)

numeric\_terms.append(numeric\_term)

steps.append(f"Valoarea numerica este: {numeric\_term}")

steps.append(f"Adaugam termenul in expansiune.\n")

In secventa de cod prezentata mai sus, apelul functiei **binomial\_coefficient** are ca scop calculul coeficientului binomial C(n,k), pentru a putea determina coeficientii in expansiunea binomialului.

**def binomial\_coefficient(n, k):**

**return math.comb(n, k)**

Restul codului prezentat se ocupa de formatarea valorilor obtinute in coef pentru a putea fi afisate in forma expanded si numerica, urmand a fi adaugati in expansiune si repetand procesul pentru toate expansiunile necesare, dimensiune data de variabila n.

expanded\_result = " + ".join(expanded\_terms)

numeric\_result = sum(numeric\_terms)

steps.append("Toti termenii au fost calculati. Expansiunea este:")

steps.append(f"{expanded\_result}")

steps.append(f"\nIar valoarea numerica este: {numeric\_result}")

Aceasta ultima secventa de cod extrasa din functia **newton\_binomial** se ocupa de gruparea rezultatelor pentru a putea calcula rezultatul numeric final si pentru a putea afisa rezultatul simbolic complet.

Functia **main**

def main():

print("Demonstratie: Binomul lui Newton prin rationament inainte\n")

a = float(input("Introduceti valoarea lui a: "))

b = float(input("Introduceti valoarea lui b: "))

n = int(input("Introduceti valoarea lui n: "))

if(n < 0):

raise Exception("valoarea lui n trebuie sa fie mai mare sau egala cu 0.")

steps, expanded\_result, numeric\_result = newton\_binomial(a, b, n)

print("\nPasii demonstratiei:")

for step in steps:

print(step)

Functia main are ca scop obtinerea input-ului de la utilizator pentru valorile a,b si n, ca ulterior sa putem apela functia **newton\_binomial** pentru a obtine rezultatele dorite folosindu-ne de valorile oferite de utilizator.

**Cod relevant pentru sistemul de recomandari:**

**import json**

**def recommend():**

**users = load\_data(USERS\_DB)**

**products = load\_data(PRODUCTS\_DB)**

**user\_id = input("Reccommend for user(user\_id): ")**

**user = next((u for u in users if u["id"] == user\_id), None)**

**if not user:**

**print("The user id was not found")**

**return**

**recommendations = generate\_recommendations(user, products)**

**print("Recommendations:", recommendations)**

**def generate\_recommendations(user, products, max\_recommendations=3):**

**recommendations = []**

**for product in products:**

**if (**

**matches\_category(user, product) and**

**matches\_price\_range(user, product) and**

**not\_purchased\_before(user, product)**

**):**

**recommendations.append(product["name"])**

**if len(recommendations) > max\_recommendations:**

**recommendations = random.sample(recommendations, max\_recommendations)**

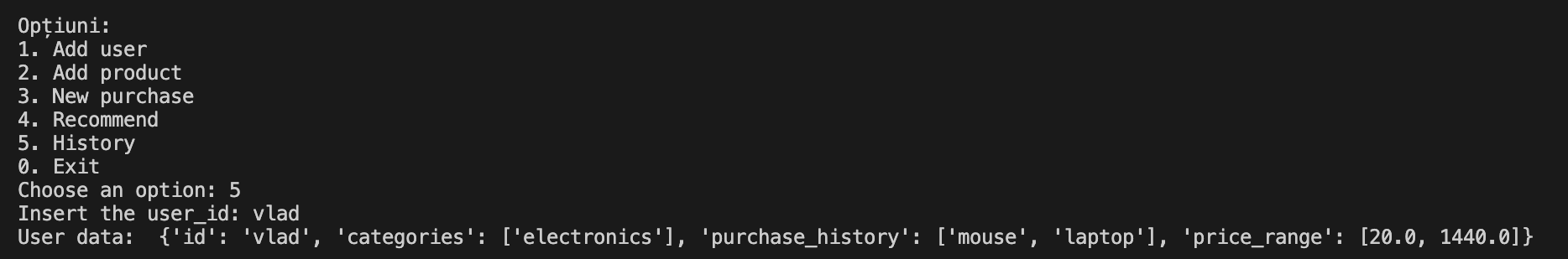
**return recommendations**

**E**xplicatie*:* Algoritmul selecteaza produse care corespund categoriilor si intervalului de preturi ale utilizatorului.

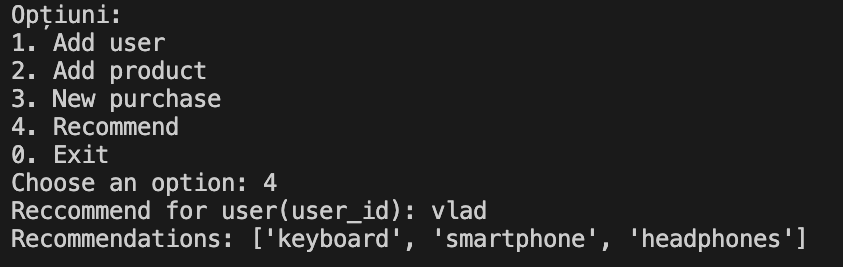
# 6. Rezultatele obtinute

**Pentru sistemul de recomandari:**

* Input: Utilizatorul Vlad(putem vizualiza datele userului folosind functia 5.History)

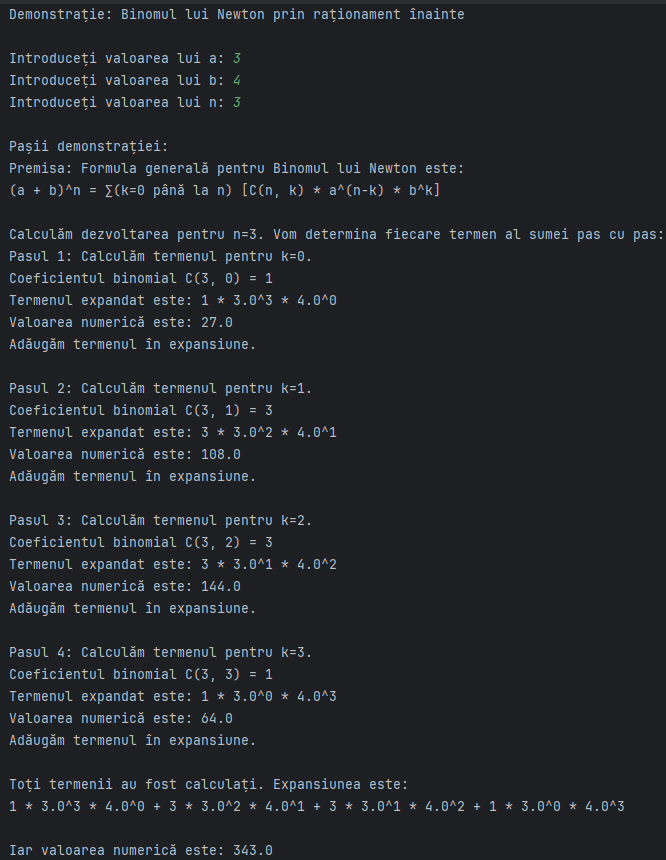


* Output: Recomandari: tastatura, telefon, casti.



**Pentru Teorema Binomiala:**

* Input: a = 3, b = 4, n = 3
* Output: 1 \* 3.0^3 \* 4.0^0 + 3 \* 3.0^2 \* 4.0^1 + 3 \* 3.0^1 \* 4.0^2 + 1 \* 3.0^0 \* 4.0^3



# 7. Teste automate

**Teorema binomiala:**

* test\_base\_n: Verifica functionarea pentru cazul de baza n=1.
* test\_\_n2: Testeaza expansiunea binomiala pentru n=2.
* test\_\_n3: Verifica expansiunea binomiala simbolica si numerica pentru n=3.
* test\_\_n0: Testeaza cazul special n=0n=0, unde rezultatul este 11.
* test\_large\_n: Verifica corectitudinea functiei pentru un n mare (n=10).
* test\_invalid\_n: Verifica gestionarea erorilor pentru n<0(intrare invalida).

**Sistemul de recomandari:**

* test\_category: Verifica daca produsul apartine unei categorii preferate de utilizator.
* test\_price\_range: Testeaza daca produsul se incadreaza in intervalul de pret al utilizatorului.
* test\_not\_purchased\_before: Verifica daca produsul nu a fost deja achizitionat de utilizator.
* test\_generate\_recommendations: Testeaza generarea corecta a recomandarilor pentru un utilizator, excluzand produsele deja achizitionate si cele care nu corespund preferintelor.
* test\_generate\_recommendations\_wrong: Verifica faptul ca un utilizator cu date invalide (necunoscut) nu primeste nicio recomandare.

# 8. Concluzii

Proiectul demonstreaza aplicabilitatea inferentei predicative in rezolvarea unor probleme practice si teoretice. Implementarea algoritmilor a condus la rezultate corecte si relevante, iar documentatia detaliaza abordarea completa.

Limitari:

* Algoritmul de recomandari nu include tehnici de machine nlearning, ci este bazat pe reguli simple.
* Complexitatea expansiunii binomiale creste exponential in functie de n, ce limiteaza utilizarea implementarii in cazul valorilor mari.

# 9. Bibliografie

<https://www.geeksforgeeks.org/forward-chaining-and-backward-chaining-inference-in-rule-based-systems/>

<https://www.geeksforgeeks.org/forward-chaining-in-ai-with-fol-proof/>

<https://www.appliedaicourse.com/blog/forward-chaining-and-backward-chaining-in-ai/>

Materiale curs Inteligenta Artificiala 2024.

# 9. Lista contributiilor

* **Lupu Eduard:** Implementarea teoremei binomiale si redactarea teoriei matematice.
* **Apostol Vlad:** Dezvoltarea sistemului de recomandari si testarea functionalitatii acestuia.