MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

Tabăra de pregatire a lotului național de Inteligență Artificială – Iunie 2025

PROBA TEORETICĂ

Numele: _____

Prenumele: _____

Unitatea școlară: _____

_____

Problema 1

1. Se consideră următorul set de date referitor la utilizatori:

ID	VenitLunar (euro)	ScorSatisfacție (0–10)
1	2300	7
2	4500	8
3	1000	4
4	12000	6
5	3100	9
6	5000	8

a) Dacă antrenăm un model de învățare automată bazat pe distanță (ex. KNN), de ce este importantă normalizarea acestor coloane? Specificați o metodă prin care pot fi normalizate.

b) Metoda IQR este o tehnică robustă de detectare a valorilor atipice (outlieri) într-un set de date numerice. Se bazează pe setarea unor limite, după cum urmează:

- Q1 (prima cuartilă) – valoarea de la 25% din datele ordonate;
- Q3 (a treia cuartilă) – valoarea de la 75% din datele ordonate;
- Limita inferioară $= Q1 - 1.5 \times (Q3 - Q1)$ și Limita superioară $= Q3 + 1.5 \times (Q3 - Q1)$
- Orice valoare care se află în afara acestor limite este considerată un outlier.

Aplicați metoda IQR (descrisă mai sus) pentru a identifica eventualele valori atipice (outlieri) din coloana **VenitLunar**.

c) Specificați care dintre **Standard Scaling** și **Robust Scaling** ar fi mai potrivită pentru a fi aplicată pe coloana **VenitLunar**. Justificați alegerea!

Problema 2

2. Se consideră următorul set de date referitor la produse alimentare:

ID	CategorieAliment	ConținutGrăsime	Preț
1	*lactate*	*ridicat*	5.4
2	*carne*	*scazut*	13.5
3	*legume*	*mediu*	3.8
4	*carne*	*ridicat*	11.0
5	*legume*	*scazut*	3.5

a) Propuneți o strategie completă de codificare a coloanelor categoriale, justificând alegerea pentru fiecare.

b) Normalizați coloana *Preț* utilizând metoda **Min-Max Scaling**.

Problema 3

3. Considerăm următoarele situații în procesul de antrenare a unui model de predicție, în care dorim să aplicăm regularizare:

   a) Considerăm că avem un număr mare de caracteristici puternic corelate și dorim să le reducem gradual contribuția, fără a le elimina complet.

   b) Dorim să eliminăm caracteristicile irelevante și să realizăm automat selecția celor relevante.

   c) Presupunem că doar un subset redus de caracteristici este cu adevărat relevant pentru sarcina de predicție.

   d) Dorim să reducem overfitting-ul printr-o penalizare distribuită uniform asupra tuturor coeficienților modelului.

   Asociați fiecărei situații cea mai potrivită formă de regularizare, alegând între L1 (Lasso) și L2 (Ridge).

Problema 4

4. Să presupunem că dorim să determinăm dacă Josh se află în birou ($Y = 1$) sau nu ($Y = 0$), analizând datele furnizate de trei senzori instalați în acel spațiu:

   - `Xc`: un senzor de presiune al scaunului, care indică dacă o persoană este așezată pe scaun (`Xc = 1`) sau nu (`Xc = 0`);
   - `Xm`: un senzor de mișcare, care detectează dacă există mișcare în cameră (`Xm = 1`) sau nu (`Xm = 0`);
   - `Xp`: un contor de energie electrică, care arată dacă există consum de energie în cameră (`Xp = 1`) sau nu (`Xp = 0`).

   Fiecare senzor oferă doar o perspectivă parțială asupra prezenței lui Josh. De exemplu, dacă Josh este în birou, dar nu stă pe scaun, senzorul de presiune nu îl va detecta.

   Avem la dispoziție un set de date istorice care include valorile măsurate de senzori și informația dacă Josh era sau nu prezent în acele momente.

Day	1	2	3	4	5	6	7	8	9	10
`Xc`	0	0	0	0	0	1	0	1	0	1
`Xm`	0	0	0	0	0	0	0	0	1	1
`Xp`	1	1	0	0	1	1	1	1	0	0
`Y`	0	0	0	0	1	1	1	1	1	1

   a) Folosind clasificatorul Naive Bayes, modelați relația dintre măsurătorile senzorilor și prezența lui Josh. Completați tabelele de probabilitate utilizând estimarea de probabilitate maximă (**Maximum Likelihood Estimation – MLE**).

`Y`	`P(Y)`
⋮	⋮

`Xc`	`Y`	`P(Xc | Y)`
⋮	⋮	⋮

`Xm`	`Y`	`P(Xm | Y)`
⋮	⋮	⋮

`Xp`	`Y`	`P(Xp | Y)`
⋮	⋮	⋮

b) Presupunând că primim un nou set de observații de la senzori: `Xc = 0, Xm = 0, Xp = 1` – conform tabelelor de probabilitate estimate prin MLE, este mai probabil ca Josh să fie prezent sau absent?

c) Dacă, în loc de MLE, folosim **netezirea Laplace (Laplace Smoothing)** pentru a estima tabelele de probabilitate, presupunând un factor de corecție (k) egal cu 1, cum se modifică estimările?

Laplace Smoothing:

$$P_{\text{Laplace}}(x_i \mid y) = \frac{N(x_i, y) + k}{N(y) + k \cdot V}$$

unde

- $N(x_i, y)$ este numărul de apariții ale valorii x_i împreună cu clasa y;
- $N(y)$ este numărul total de exemple care aparțin clasei y;
- V este numărul total de valori posibile pentru features;
- k este factorul de corecție.

Problema 5

5. Dorim să construim un arbore de decizie care să prezică dacă un student promovează sau nu un examen. Pentru acest lucru, pornim de la următorul set de date ce conține 2 atribute: tipul punctajului de pe parcurs (mare, mediu, mic) și dacă a învățat sau nu pentru examen.

Parcurs	Învățat	Examen
mic	nu	picat
mic	da	trecut
mediu	nu	picat
mediu	da	trecut
mare	nu	trecut
mare	da	trecut

a) Calculați entropia $E(Examen)$ ($\log_2 3 \approx 1.58$).

b) Calculați entropia $E(Examen|Parcurs)$.

c) Construiți un arbore de decizie pentru setul de date furnizat.

$$E(X) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

$$E(Y|X) = -\sum_{x \in X} \sum_{y \in Y} P(y|x) \cdot P(x) \cdot \log_2(P(y|x))$$

Problema 6

6. Distanța **Manhattan** între două puncte $A(x_1, y_1)$ și $B(x_2, y_2)$ este definită prin formula:

$$\text{dist}_{\text{Manhattan}}(A, B) = |x_1 - x_2| + |y_1 - y_2|$$

Considerăm următorul set de puncte în planul 2D:

$$P = \{A(1,2),\ B(2,1),\ C(4,5),\ D(7,8),\ E(8,8),\ F(9,9)\}$$

Centroizii inițiali sunt:

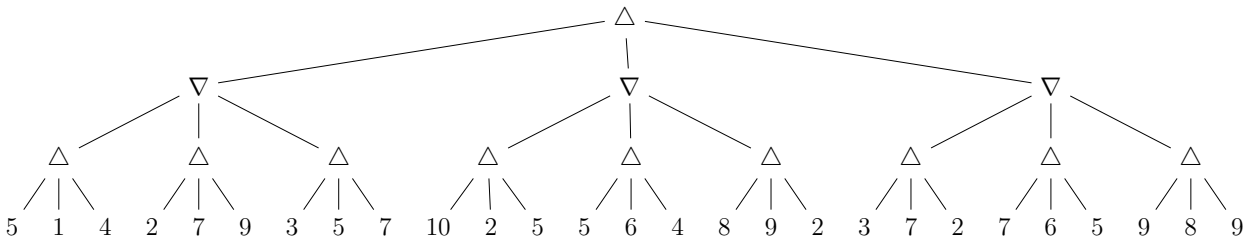
$$C_1^{(0)} = (2,2), \quad C_2^{(0)} = (8,9)$$

a) Aplicați un prim pas din algoritmul K-Means, folosind distanța Manhattan. Specificați pentru fiecare punct din ce cluster va face parte.

b) După formarea celor două clustere, recalculați centroizii acestora ca media aritmetică a punctelor din cluster (pe fiecare coordonată).

Problema 7

7. Pornim de la următorul arbore de joc pentru care considerăm nodurile △ corespunzătoare jucătorului MAX și nodurile ∇ corespunzătoare lui MIN.



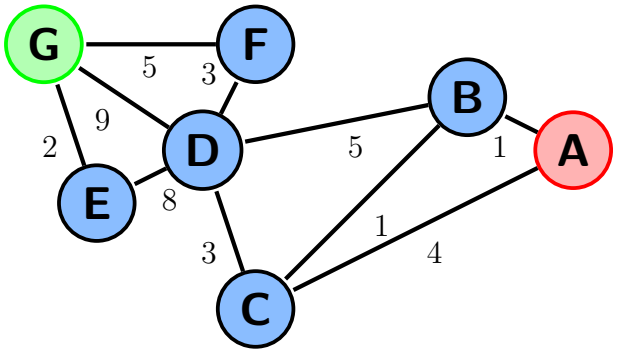
a) Propagați valorile în acest arbore de joc și arătați calea ce va fi aleasă de un algoritm **MINIMAX**.

b) Verificați dacă se pot aplica tăieri de tip α / β pentru acest arbore de joc. Evidențiați tăierile posibile (specificând valorile α / β pentru nodurile de interes) și specificați tipul pentru fiecare tăiere.

c) Presupunem că avem un joc în care jucătorul **MAX** are abilitatea de a controla acțiunile jucătorului **MIN**. Mai exact, jucătorul **MAX** va avea posibilitatea să îi impună jucătorului **MIN** ce acțiune să aleagă, dar pentru acest lucru va trebui să plătească un cost $c = 2$ (pondere egală cu valorile furnizate în frunzele arborelui de joc). Argumentați dacă este sau nu optimal pentru jucătorul **MAX** să folosească această abilitate, plătind costul c, pornind de la arborele de joc.

Problema 8

8. Considerăm următorul graf ponderat drept o reprezentare a spațiului stărilor pentru o problemă de căutare. Vom considera A starea inițială a problemei, iar G starea finală a problemei. Dacă avem în graf muchia (U, V) înseamnă ca pentru starea U avem starea V drept una din stările următoare și invers (putem face parcurgerea în ambele direcții). De asemenea, cunoaștem și o funcție euristică pentru care lipsește o valoare.



Starea	A	B	C	D	E	F	G
h	10	?	9	7	1.5	4.5	0

a) Specificați ce valoare trebuie să aibă $h(B)$ pentru ca euristica h să fie admisibilă și ce valoare ar trebui să aibă $h(B)$ pentru ca euristica h să fie consistentă. Argumentați răspunsurile!

b) Propuneți o funcție euristică pentru care rezultatul algoritmului A* aplicat pentru acest spațiu de căutare să fie $A \to C \to B \to D \to E \to G$.

c) Considerăm două funcții euristice $h_1(s)$ și $h_2(s)$ care se pot aplica pentru algoritmul A*, cunoscând faptul că ele sunt admisibile. Dorim să combinăm aceste două funcții astfel încât să obținem o funcție g. Putem construi o funcție g care să garanteze explorarea unui număr mai mic de noduri (fața de varianta în care aplicăm A* cu funcția h_1 sau A* cu funcția h_2) și care să garanteze condiția de admisibilitate? Explicați și oferiți un exemplu de definiție pentru funcția g. Puteți exemplifica pornind de la graful dat.

Problema 9

9. Se dau următoarele propoziții în limbaj natural:

   - *"Dacă un program este corect, atunci el produce rezultate conforme.* `PROG1` *este un program corect. Prin urmare,* `PROG1` *produce rezultate conforme."*

   a) Traduceți cele de mai sus în logica propozițională.

   b) Demonstrați propoziția prin reducere la absurd folosind rezoluția.
   Observație: rezoluția funcționează doar pe propoziții aduse la FNC.

Problema 10

10. Știind că sunt adevărate următoarele afirmații: *"Capul plecat nu este tăiat de sabie"*, și că *"I s-a tăiat capul lui Marcel."*, dorim să demonstrăm că *"Marcel nu avea capul plecat."*.

    a) Reformulați proverbul într-o propoziție logică (în logica de ordinul întâi).

    b) Demonstrați, prin metoda rezoluției, afirmația conform căreia *Marcel nu avea capul plecat.*.
    Atenție: demonstrațiile care nu utilizează metoda rezoluției nu se punctează.

Problema 11

11. Considerăm următoarea problemă de satisfacere a constrângerilor (CSP): variabilele X, Y și Z au domeniile:

    - $X : \{1, 2, \ldots, 10\}$
    - $Y : \{5, 6, \ldots, 10\}$
    - $Z : \{5, 6, \ldots, 15\}$

    și avem următoarele constrângeri:

    - $X > Y$
    - $Y + Z = 12$
    - $X + Z = 15$

    a) Există vreun arc care este **arc-consistent** pentru problema dată?

    b) Considerăm algoritmul de arc consistență (AC-3). Cum este inițializată coada de arce și care este numărul maxim de ori în care un arc poate fi adăugat în coadă?

    c) Aplicați algoritmul de arc consistență(AC-3) pentru a actualiza domeniile variabilelor.

Problema 12

12. Trei examene (Math, CS, Physics) trebuie programate în două săli (S1, S2), în trei intervale orare (T1, T2, T3). Pentru această programare avem următoarele constrângeri:

    - Math și CS nu pot fi în același timp (au studenți comuni)
    - Physics și CS nu pot fi în aceeași sală
    - O sală nu poate găzdui două examene în același interval

    a) Reprezentați problema sub forma unui probleme de tip CSP și explicați cum pot fi reprezentate variabilele și care sunt domeniile variabilelor.

    b) Precizați toate constrângerile într-o variantă formală, utilizând variabilele propuse la cerința anterioară.

    c) Folosiți backtracking pentru a găsi o soluție validă.

Problema 13

13. Într-o lume de tip grid, Pacman navighează un mediu periculos plin de fantome și puncte alimentare. Totuși, dimensiunea foarte mare a acestui spațiu de stări face imposibilă utilizarea directă a metodei **Q-learning** în forma sa clasică — care presupune memorarea valorilor Q pentru fiecare pereche stare-acțiune.

    Pentru a depăși această problemă, alegem o strategie bazată pe **caracteristici**: în loc să memoreze explicit fiecare valoare Q, Pacman folosește o funcție de **aproximare liniară**, în care valoarea Q este calculată pe baza unui set de caracteristici extrase din starea și acțiunea curentă.

    Caracteristicile alese, F_g și F_p, pentru această reprezentare sunt:

    - $F_g(s, a) = A(s) + B(s, a) + C(s, a)$: măsoară cât de aproape sau expus este Pacman față de fantome.
    - $F_p(s, a) = D(s) + 2 \cdot E(s, a)$: reflectă accesibilitatea punctelor alimentare.
    - $A(s)$: numărul de fantome aflate la un pas distanță de starea actuală.
    - $B(s, a)$: numărul de fantome atinse imediat după acțiunea a.
    - $C(s, a)$: numărul de fantome la un pas de poziția în care Pacman ajunge după acțiunea a.
    - $D(s)$: numărul de puncte alimentare aflate la un pas distanță de starea actuală.
    - $E(s, a)$: numărul de puncte alimentare consumate prin efectuarea acțiunii a.

    Pentru acest grid Pacman, fantomele vor fi întotdeauna **staționare**, iar spațiul de acțiune este {stânga, dreapta, sus, jos, rămâne}.

Figure 1: Starea curentă

a) Trebuie să calculați valorile caracteristicilor Fg și Fp pentru fiecare acțiune disponibilă: stânga, dreapta, sus, rămâne; în starea exemplificată în Figura 1 (pagina anterioară) .

b) După mai multe episoade de **Q-learning**, Pacman a învățat să evalueze acțiunile cu ajutorul unor ponderi asociate fiecărei caracteristici:

- $w_g = -10$: greutatea asociată expunerii la fantome.
- $w_p = +100$: greutatea asociată oportunităților de a consuma puncte alimentare.

Astfel, valoarea Q pentru o acțiune a în starea curentă s este calculată ca:

$$Q(s, a) = w_g \cdot F_g(s, a) + w_p \cdot F_p(s, a)$$

Trebuie calculate valorile Q pentru fiecare dintre acțiunile: **stânga**, **dreapta**, **sus**, **rămâne** din starea curentă prezentată în figură.

c) Se observă acum o tranziție specifică:

- Pacman pornește din starea s, alege acțiunea *sus*,
- Ajunge în noua stare s', unde există un punct alimentar consumat,
- Primește o recompensă $R(s, a, s') = 250$.

Din noua stare s', acțiunile posibile sunt *jos* și *rămâne*. Se presupune un factor de discount $\gamma = 0.5$.

Trebuie calculată noua estimare a valorii Q pentru starea și acțiunea inițială — adică pentru $Q(s, sus)$ — folosind regula de actualizare:

$$Q_{nou}(s, a) \leftarrow R(s, a, s') + \gamma \cdot \max_{a'} Q(s', a')$$

d) Având noua valoare estimată pentru Q și știind valoarea Q curentă estimată de funcția de caracteristici, folosim **regula de actualizare a ponderilor** în Q-learning:

$$w_i \leftarrow w_i + \alpha \cdot \delta \cdot F_i(s, a)$$

unde:

- $\alpha = 0.5$ este **rata de învățare**,
- $\delta = Q_{\text{nou}} - Q_{\text{curent}}$ este **eroarea de predicție**,
- $F_i(s, a)$ sunt valorile caracteristicilor F_g și F_p pentru acțiunea „*sus*”.

Rezultatul final este un set actualizat de **ponderi** care vor ghida deciziile viitoare ale lui Pacman într-un mod mai inteligent și adaptat experiențelor acumulate.

Pornind de la aceste relații, calculati valorile pentru ponderile w_g și w_p.

Problema 14

14. Un model de clasificare a fost evaluat pe un set de testare și a generat următoarea matrice de confuzie:

	Predicted A	**Predicted B**	**Predicted C**
Actual A	40	5	5
Actual B	4	35	11
Actual C	2	8	40

a) Calculați precizia pentru fiecare clasă (**A**, **B**, **C**).

b) Calculați **macro-average F1** pentru acest model, adică media aritmetică a F1-score-urilor obținute pentru fiecare clasă.

c) Calculați **micro-average Recall** pentru acest model.

d) Dacă aplicația care utilizează acest model de clasificare vizează detectarea corectă a cazurilor din clasa **C** (de exemplu, o boală rară și foarte gravă), ce metrică ar trebui să fie maximizată? Ce metodă puteți propune pentru a îmbunătăți performanța pentru această clasă?

Problema 15

15. În învăţarea automată, *funcţiile de similaritate* joacă un rol foarte important atât în clasificarea automată cât şi în clusterizare.

Considerăm X o mulţime formată din imagini dreptunghiulare de dimensiuni arbitrare, în care fiecare pixel este reprezentat sub forma unui număr întreg din mulţimea $\{0, \ldots, 255\}$. Fie $k_1 : X \times X \to \mathbb{R}$ o *funcţie de similaritate* a imaginilor, definită astfel: $k_1(x, x')$ = numărul de zone pătratice (engl., pixel patches), de dimensiune 16×16 pixeli, care apar atât în imaginea x cât şi în imaginea x'.

Precizare

Nu se va lua în considerare poziţia în care apare o astfel de *zonă pătratică*, ci doar *imaginea* / coloritul ei ca atare. O astfel de *zonă* poate să apară în mai multe poziţii într-o aceeaşi imagine, însă funcţia $k_1(x, x')$ va „contoriza" o singură dată o anumită *zonă pătratică* dacă ea apare [nu are importanţă de câte ori anume] atât în x cât şi în x'.

a) Demonstraţi că funcţia k_1 are proprietatea următoare: există $d \in \mathbb{N}^*$ şi o funcţie („mapare") $\phi : X \to \mathbb{R}^d$ astfel încât $k_1(x, x') = \phi(x) \cdot \phi(x')$ pentru orice x şi $x' \in X$. Altfel spus, funcţia k_1 este *funcţie-nucleu*. Am notat cu $\cdot$ produsul scalar al vectorilor.

b) Fie acum o altă *funcţie de similaritate*:

$$k_2(x, x') = \begin{cases} 1 & \text{dacă } k_1(x, x') \geq 1, \quad \text{adică, există cel puţin un "patch"} \\ & \qquad\qquad\qquad\quad \text{(zonă pătratică) comun(ă) pentru } x \text{ şi } x' \\ 0 & \text{în caz contrar.} \end{cases}$$

Demonstraţi că funcţia k_2 are proprietatea următoare: există o mulţime de imagini $x_1, \ldots, x_n$ şi un vector-coloană $f \in \mathbb{R}^n$ astfel încât $f^\top G f < 0$, unde $\top$ reprezintă operaţia de transpunere, iar G este matricea de tip $n \times n$ având elementele $G(i, j) \stackrel{def.}{=} k_2(x_i, x_j)$, pentru $i, j = 1, \ldots, n$. Altfel spus, funcţia k_2 **nu** este pozitiv semidefinită, deci **nu** este funcţie-nucleu (conform teoremei lui Mercer).

Sugestie

Puteţi face demonstraţia în manieră constructivă, adică alegând în mod convenabil numărul n şi „construind" / definind imaginile $x_1, \ldots, x_n$ astfel încât să existe $f \in \mathbb{R}^n$ cu propietatea $f^\top G f < 0$.

Problema 16

16. În sistemele de recomandare, similaritatea între utilizatori sau între itemi joacă un rol esențial în prezicerea preferințelor.

Considerăm un sistem de recomandare unde:

- $U = \{u_1, u_2, \ldots, u_n\}$ este mulțimea utilizatorilor;
- $I = \{i_1, i_2, \ldots, i_m\}$ este mulțimea itemilor (de exemplu, filme);
- $R \in \mathbb{R}^{n \times m}$ este matricea de ratinguri, unde R_{uj} este ratingul dat de utilizatorul $u \in \{u_1, u_2, \ldots, u_n\}$ itemului i_j, sau este necunoscut dacă nu există un rating.

Definim acum o funcție de similaritate între doi utilizatori u și v astfel:

$$\text{sim}(u, v) = \frac{\sum_{j \in I_{uv}} (R_{uj} - \bar{R}_u)(R_{vj} - \bar{R}_v)}{\sqrt{\sum_{j \in I_{uv}} (R_{uj} - \bar{R}_u)^2} \cdot \sqrt{\sum_{j \in I_{uv}} (R_{vj} - \bar{R}_v)^2}}$$

unde I_{uv} este mulțimea itemilor evaluați de ambii utilizatori, iar $\bar{R}_u$ este media ratingurilor date de utilizatorul u.

Precizare

Această funcție de similaritate este cunoscută sub numele de **corelație Pearson** și este frecvent folosită în filtrarea colaborativă user-based.

a) Presupunem că avem următoarea matrice de ratinguri (cu "-" indicând lipsa unui rating):

$$R = \begin{bmatrix} 5 & 3 & - & 1 \\ 4 & - & 3 & 2 \\ 1 & 1 & 5 & 5 \\ - & - & 5 & 4 \end{bmatrix}$$

Calculați:

- $\text{sim}(u_1, u_2)$
- $\text{sim}(u_1, u_3)$

Precizați care dintre cei doi utilizatori este *mai asemănător* cu utilizatorul u_1, considerând că $\sqrt{6} \approx 2.45$.

b) Folosind filtrarea colaborativă bazată pe utilizatori, preziceți ratingul pe care utilizatorul u_1 l-ar putea da itemului i_3, folosind doar valorile date de ceilalți doi utilizatori (u_2 și u_3), pe baza formulei:

$$\hat{R}_{u_1,3} = \bar{R}_{u_1} + \frac{\sum\limits_{v \in \{u_2, u_3\}} \text{sim}(u_1, v) \cdot (R_{v3} - \bar{R}_v)}{\sum\limits_{v \in \{u_2, u_3\}} |\text{sim}(u_1, v)|}$$

c) Discutăm o limitare a metodei de mai sus. Ce se întâmplă în cazul unui utilizator nou care nu a evaluat niciun item (așa-numita problemă *cold-start*)?

Propuneți o soluție bazată pe filtrare de tip **content-based** care ar putea atenua această problemă.

Sugestie

Puteți presupune că fiecare item este descris de un vector de caracteristici (de exemplu, genuri pentru filme). Gândiți-vă cum ar putea fi folosit acest vector pentru a recomanda itemi unui utilizator fără ratinguri anterioare.

Problema 17

17. Uneori, trebuie să comparăm clasamente. Cum putem cuantifica similaritatea dintre două clasamente? Pentru un top muzical, o astfel de măsură ne va spune: *"Cât de mult sunt de acord aceste două liste de albume/piese de top?"*

Putem privi un clasament ca o permutare x, unde x_i reprezintă poziția (rangul) elementului i.

Un întreg set de statistici de corelație pentru date ordinale se bazează pe numărul de perechi **concordante** și **discordante** dintre două variabile. Comparăm toate perechile posibile de elemente și numărăm câte sunt ordonate în același fel și câte în ordine opusă.

Pereche concordantă: dacă ambele clasamente sunt de acord asupra ordinii a două elemente (adică elementul i este clasat mai sus sau mai jos decât elementul j în **ambele** liste).

Pereche discordantă: dacă cele două clasamente nu sunt de acord (adică elementul i este mai sus decât j într-o listă, dar mai jos în cealaltă).

Matematic, numărul perechilor concordante este:

$$P = \left|\{(i,j) \,:\, 1 \le i < j \le n,\ (x_i - x_j)(y_i - y_j) > 0\}\right|.$$

În mod similar, numărul perechilor discordante este:

$$Q = \left|\{(i,j) \,:\, 1 \le i < j \le n,\ (x_i - x_j)(y_i - y_j) < 0\}\right|.$$

Kernelul PQ dintre două clasamente x și y este definit ca:

$$k_{PQ}(x,y) = P - Q.$$

Dorim să demonstrăm că PQ este într-adevăr un **kernel**, arătând cum putem construi explicit harta de caracteristici Ψ astfel încât:

$$k_{PQ}(x,y) = P - Q = \langle \Psi(x), \Psi(y) \rangle,$$

unde $\langle \cdot, \cdot \rangle$ denotă produsul scalar.