# O îmbunătățire a performanțelor algoritmului KNN în sistemele de recomandare pe web

**Costin-Gabriel Chiru**
Universitatea „Politehnica” Bucuresti
Splaiul Independenței nr. 313 – București, România
chirucos@gmail.com

**Ştefan Trăuşan-Matu**
Universitatea „Politehnica” Bucuresti
Splaiul Independenței nr. 313 – București, România
trausan@gmail.com

**Traian Rebedea**
Universitatea „Politehnica” Bucuresti
Splaiul Independenței nr. 313 – București, România
trebedea@gmail.com

**REZUMAT**
Lucrarea de față prezintă o abordare ce poate conduce la îmbunătățirea rezultatelor întoarse de sistemele de recomandare bazate pe o implementare de tipul filtrării colaborative. Recomandările oferite de acest tip de sisteme sunt generate pe baza preferințelor utilizatorilor care au comportamente asemănătoare cu cel al utilizatorului curent. Pentru identificarea acestor utilizatori se utilizează foarte frecvent o metodă de clasificare automată bazată pe algoritmul „cei mai apropiați K vecini”. Scopul acestei lucrări este să trateze o modalitate de a prezenta şi corecta unele probleme ale acestui algoritm, prin aplicarea unei interpolări asupra claselor întoarse de către varianta standard a algoritmului de clasificare, astfel încât să se îmbunătățească acuratețea metodei şi, implicit, a rezultatelor fumizate de sistemele de recomandare.

**Cuvinte cheie**
Sisteme de recomandare, KNN, clasificare.

**Clasificare ACM**
Interfețe inteligente om-calculator, agenți inteligenți

## INTRODUCERE

Sistemele de recomandare încearcă să furnizeze utilizatorilor propuneri de itemi (pagini web, ştiri, cărți, filme, muzică, poze, CD-uri etc.) care pot fi interesanți pentru aceştia, plecând de la filtrarea informațiilor pe baza preferințele altor persoane, cu un profil similar. De aceea, aceste sisteme reprezintă o alternativă foarte bună pentru algoritmii de căutare, deoarece ajută utilizatorii să descopere itemi pe care nu i-ar putea descoperi singuri datorită multitudinii de opțiuni din care pot alege [3].

Dintr-o altă perspectivă, putem spune că se schimbă direcția interacțiunii om-calculator. În loc ca utilizatorul să ceară explicit un item, sistemele de recomandare îi propun acestuia un item care a fost preferat de persoane cu caracteristici asemănătoare. De exemplu, unul dintre cele mai cunoscute astfel de sisteme este cel de la amazon.com, care, plecând de la caracteristicile pe care le deduce din cărțile cumpărate sau chiar doar examinate de un utilizator pe situl web, recomandă automat cărți cumpărate de persoane care au avut interese similare.

Sistemele de recomandare compară profilul utilizatorului cu anumite caracteristici de referință, pe baza acestei comparații calculând o listă de itemi ce pot fi interesanți pentru utilizator. Caracteristicile de referință pot fi de două feluri: caracteristici ce țin de contextul comportamentului social al utilizatorului (abordare de tipul filtrării colaborative) sau caracteristici ce țin de natura itemilor (abordare bazată pe conținut) [2].

Majoritatea abordărilor de tipul filtrării colaborative folosesc metode de tipul „cei mai apropiați K vecini” (engl. „K Nearest Neighbors” - KNN). Aceasta este o metodă de învățare automată din exemple pentru aproximarea funcțiilor cu valori reale sau discrete. Acest tip de învățare stochează instanțele furnizate ca exemple pentru antrenarea sistemului şi amână învățarea efectivă până când este primit un exemplu de test. Învățarea are loc de fiecare dată când se primeşte un nou test. Procesul constă din extragerea din memorie a instanțelor similare şi clasificarea exemplului de test pe baza acestora [6].

În următoarele două secțiuni vom face o scurtă descriere a sistemelor de recomandare şi a algoritmului clasic de KNN. Următoarele secțiuni descriu îmbunătățirile aduse algoritmului clasic şi datele cu care s-a lucrat. Lucrarea se încheie cu prezentarea rezultatelor obținute şi cu concluziile derivate din aceste rezultate.

## SISTEME DE RECOMANDARE

Sistemele de recomandare încearcă să ajute utilizatorii să identifice rapid care sunt produsele şi serviciile interesante dintr-o gamă de oferte extrem de variată. Numărul şi complexitatea acestora fac practic imposibilă incercarea utilizatorilor de a le trece în revistă cu scopul de a lua o decizie în privința lor. Sistemele de recomandare sunt folosite de către foarte multe magazine on-line, putând să ofere recomandări despre aproape orice: începând de la lucruri simple, ca pagini web, ştiri, cărți, filme, muzică, poze, CD-uri, şi până la itemi complecşi de genul serviciilor financiare sau serviciilor de e-government.

Recomandările pot fi generate prin două metode: prin purtarea de dialoguri cu utilizatorii on-line de pe sit sau prin analiza informațiilor existente cu privire la itemii cumpărați de un individ sau de către o comunitate de utilizatori. Datorită faptului că metoda dialogului s-a demonstrat a fi inadecvată, s-a trecut la construcția de sisteme utilizând metoda analizei. Primul sistem de recomandare bazat pe această metodă a fost construit la mijlocul anilor 90 [2]. De atunci, interesul pentru astfel de sisteme a crescut extrem de mult datorită cererii mari de tehnologii personalizate de către aplicațiile de succes din domeniul comerțului electronic (ex. amazon.com).

Una din cele mai des utilizate metode folosite în sistemele de recomandare este filtrarea colaborativă [4, 5], care

adoptă, în general, o metodă de KNN aplicată unei matrice de evaluări. În filtrarea colaborativă, opiniile prietenilor şi rapoartele ce compară diferiţi itemi reprezintă principalele elemente care influenţează decizia utilizatorilor de a cumpăra unul din acei itemi [2]. Alte metode utilizate frecvent în sistemele de recomandare sunt metodele bazate pe conţinut (pe ceea ce a cumpărat respectivul utilizator în trecut) şi respectiv cele bazate pe cunoştinţe (pentru cumpărături complexe care necesită atât informaţii suplimentare despre context cât şi modalităţi de interacţiune cu sistemul) [2].

## ALGORITMUL KNN

KNN este cea mai simplă metodă de clasificare bazată pe instanţe. Algoritmul presupune că toate exemplele sunt plasate într-un spaţiu n-dimensional. Fiecare instanţă de test este clasificată în funcţie de cele mai apropiate k exemple din memorie, fără a încerca să se determine funcţia de distribuţie pe baza căreia instanţele sunt plasate în spaţiul n-dimensional [6].

Pentru a găsi şi extrage din memorie aceste exemple, este nevoie de o metodă de a compara cât de similare sunt două instanţe. În cazul în care se lucrează cu valori reale, se pot aplica metrici de genul distanţei Euclidiene, distanţei normate sau funcţiei cosinus între instanţe. Pentru situaţiile în care valorile sunt discrete, se calculează numărul caracteristicilor distincte între cele două instanţe. În lucrarea curentă, ne vom concentra doar asupra ultimei variante, cazul valorilor reale fiind similar, unica diferenţă fiind modul de calcul al similarităţii dintre două instanţe. După definirea unei metrici pentru determinarea similarităţii dintre două instanţe, următorul pas este să se calculeze distanţa dintre exemplul de test şi toate exemplele de învăţare aflate în memorie. În ultimul pas se determină pentru fiecare instanţă de test care sunt cele mai apropiate k instanţe din setul din memorie şi pe baza acestor instanţe similare se decide care este cea mai probabilă clasificare a acesteia. Acest lucru se realizează prin votarea între cele mai similare k instanţe.

Din modul de calcul al clasificării instanţelor de test se observă faptul că algoritmul KNN utilizează o aproximare diferită pentru fiecare instanţă de test în parte. Acest lucru oferă un mare avantaj în cazul în care funcţia de distribuţie este foarte complexă, dar poate fi descrisă printr-o colecţie de aproximări locale mai puţin complicate.

## ÎMBUNĂTĂŢIREA ALGORITMULUI CLASIC DE KNN

Algoritmul KNN este o metodă inductivă de inferenţă foarte eficientă pentru multe probleme practice. Acest algoritm este rezistent la zgomotul ce poate să apară în datele de antrenament şi este foarte eficient în cazul în care i se furnizează un set de date de antrenament suficient de mare. În ciuda rezultatelor bune obţinute folosind KNN, acest algoritm are totuşi câteva probleme, pe care ne propunem să le corectăm prin îmbunătăţirile propuse. Una din cele mai mari probleme ale algoritmului KNN este faptul că nu oferă informaţii despre ce valoare ar trebui să fie aleasă pentru „k” astfel încât să se obţină cele mai bune rezultate. O altă problemă a algoritmului KNN este aceea a ponderii pe care o au instanţele în calculul rezultatului final. Se pune problema dacă toate instanţele din setul de antrenare trebuie sa aibă aceeaşi pondere. Ultima problemă considerată în lucrarea de faţă a fost aceea a distanţei medii dintre instanţa de test şi instanţele din setul de antrenament. Se pune întrebarea dacă se pot obţine rezultate mai bune în cazul în care se foloseşte distanţa ca modalitate de ponderare a rezultatelor obţinute.

Pentru a elimina problemele identificate la algoritmul KNN, s-a construit un sistem care să poată aborda fiecare dintre probleme în parte.

Acest sistem ia toate valorile lui „k” începând de la valoarea 1 şi până la o limită maximă şi calculează care este clasificarea instanţei de test pentru fiecare valoare a lui „k” în intervalul specificat şi pentru datele de antrenare date. Această limită poate fi impusă de utilizator sau poate fi aleasă astfel încât să fie egală cu numărul total de instanţe de antrenare furnizate sistemului. În continuare, există două variante de utilizare a rezultatelor întoarse de aplicările algoritmului cu diferite valori ale lui „k”:

- prima variantă, în care nu se foloseşte nici o pondere pentru valorile întoarse (toate aplicările au pondere egală)
- a doua variantă, în care valorile întoarse de algoritmul KNN sunt ponderate cu valorile folosite pentru „k”.

Aceste posibilităţi au fost luate în calcul pentru a putea adresa problema ponderării, adică pentru a vedea dacă rezultatele obţinute trebuiesc (sau nu) ponderate în vreun fel în funcţie de valoarea lui „k”.

În afara acestor două variante, se mai dă posibilitatea utilizatorului de a alege dacă doreşte sau nu să folosească distanţa medie dintre instanţa de test şi cele de antrenare considerate în aplicarea respectivă a algoritmului KNN, pentru a pondera în plus rezultatele obţinute.

### Alegerea lui „k”

Aşa cum s-a specificat în secţiunile anterioare, alegerea valorii lui „k” pentru care se va aplica algoritmul KNN ridică numeroase probleme.

Valorile mici ale lui „k” permit identificarea mai eficientă a structurilor de dimensiuni mici din spaţiul problemei, dar pot conduce la crearea unui model prea complex (engl. „overfitting”). În plus, dacă se consideră valori mici pentru „k”, atunci rezultatele sunt influenţate in mod sensibil de datele incorecte (engl. „noisy data”).

Dacă se aleg valori mari pentru „k”, se evită problema zgomotelor din datele de antrenare şi, de asemenea, se pot estima mai bine probabilităţile în cazul în care se lucrează cu valori discrete.

În schimb, pe lângă avantajele oferite de o astfel de alegere apar şi 2 dezavantaje majore. O primă problemă este că nu se pot aplica valori mari pentru „k” în cazul în care se lucrează cu seturi mici de date de antrenare. Cel de-al doilea dezavantaj este că nu se cunoaşte cât de mult se poate mări valoarea lui „k” fără a scădea performanţele – mărirea valorii lui „k” înseamnă de fapt simplificarea modelului din spaţiul problemei şi dacă acest model se simplifică prea mult, este posibil să nu mai obţinem o soluţie corectă.

Cercetătorii din domeniu au arătat că nu există o valoare general valabilă pentru „k” astfel încât obţinerea rezultatelor optime să fie garantată şi că această valoare

depinde de fiecare problemă în parte. De asemenea, aceştia au ajuns la concluzia că valoarea lui „k” trebuie aleasă din intervalul 1 – 30 .

Abordarea noastră încearcă să elimine aceste probleme prin evitarea alegerii între diferite valori ale lui „k”, algoritmul KNN rulându-se cu toate valorile posibile mai mici decât o valoare limită. Cu alte cuvinte, în loc să se aplice algoritmul KNN pentru o singură valoare a lui „k”, acesta se aplică de mai multe ori având valori diferite pentru „k”. În acest fel, se încearcă eliminarea problemelor apărute din alegerea unei valori prea mici sau prea mari pentru „k”.

### Ponderarea

Problema ponderării constă în a determina dacă datele de antrenament trebuie să aibă ponderi diferite în rezultatele furnizate. Aceasta presupune să se dea o importanță mai mare (tradusă printr-o pondere mai mare) instanțelor care sunt mai aproape de instanța de test şi o importanță mai mică (tradusă printr-o pondere mai mică) instanțelor care sunt departe de aceasta.

Datorită îmbunătățirilor prezentate anterior, această problema se rezolva de la sine. Abordarea noastră rezolvă în mod implicit această problemă prin considerarea multiplă a instanțelor ce sunt mai aproape de instanța de test. Cu cât o instanță este mai aproape de instanța de test, cu atât valoarea acesteia va fi luată în considerare în mai multe aplicări ale algoritmului KNN având valori diferite ale lui „k”. În acest fel, instanțele aflate în apropierea instanței de test capătă ponderi mai mari ca celelalte în final, când se decide care va fi valoarea instanței de test.

Chiar dacă această problemă este rezolvată implicit, o altă problemă apare: valorile întoarse de aplicările diferite ale algoritmul KNN trebuiesc ponderate în raport cu numărul de vecini considerați sau nu? Ceea ce încercăm să exprimăm prin această întrebare este posibilitatea ca valorile întoarse de aplicările algoritmului KNN pentru diferite valori ale lui „k” să aibă importanță diferită în votarea finală. Din această cauză, sistemul permite cele două variante: valorile întoarse nu sunt ponderate în nici un fel sau ele sunt ponderate cu valorile folosite pentru „k”. Dând aceasta posibilitate, am încercat sa vedem dacă clasificările obținute folosind valori mici ale lui „k” ar trebui să aibă mai mare importanță (deci şi ponderi mai mari) în raport cu cele obținute utilizând valori mai mari pentru „k”.

### Distanța

O ultima problemă este aceea a identificării în ce măsură considerarea distanței în ponderarea rezultatelor este utilă. Ideea care a generat posibilitatea considerării acesteia ca factor de ponderare a fost că în cazul în care se obține o valoare mai mică a distanței medii între instanța de test şi cele de antrenament avem de-a face cu o structură mai compactă decât în cazul în care această valoare este mare, şi de aceea structura ar putea fi mai aproape de realitate. Această idee este mai uşor de înțeles urmărind exemplul următor în care valoarea lui „k” este 3 şi în care avem două posibilități pentru instanțele de test:

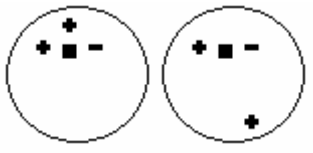

**Figura 1 Exemplu 3NN**

În ambele figuri instanța de test (reprezentată printr-un pătrat) va fi clasificată ca fiind pozitivă, dar încrederea cu care se face aceasta clasificare nu este aceeaşi pentru cele două figuri. Dacă se calculează distantă medie în ambele cazuri, şi se ponderează clasificarea returnată de algoritmul 3NN cu această distanță se observă că avem mai multă încredere în primul caz decât în al doilea.

Datorită acestei situații, s-a realizat un test care să testeze influenta distanței în clasificarea instanței de test. Astfel, pentru fiecare „k” în parte, s-a calculat valoarea medie a distanței între instanța de test şi instanțele de antrenament considerate.

## DATELE

Pentru efectuarea testelor s-au folosit cinci seturi diferite de date, dintre care trei (Connect-4, Clasificarea ciupercilor şi X şi 0) au fost preluate din baza de date a UCI Machine Learning Repository [1]. Al patrulea set de date a fost preluat de pe situl Universității din Antwerp de pe pagina cursului de tehnologia limbajului („Language Technology” [7]). Ultimul set de date a fost preluat de pe situl Universității din Texas la Dallas de pe pagina cursului de învățare automată (engl. „Machine Learning”) [8]. În continuare vom prezenta fiecare din aceste seturi de date:

### Jocul Connect-4

Acest set de date conține toate pozițiile legale din jocul connect-4 pe o tabla de 6 x 7 pătrățele, în care nici unul dintre jucători nu a câştigat încă şi în care următoarea mutare nu este forțată [9]. Setul de date conține 67.557 instanțe, fiecare având câte 42 atribute, corespunzătoare celor 42 de pătrățele disponibile în joc. Aceste atribute pot să ia una din valorile X, 0 sau B ele semnificând că jucătorul X respectiv 0 au ocupat poziția respectivă sau că nici unul din cei doi nu a ocupat pătrățelul şi acesta este gol (engl. blank). Exemplele nu conțin valori lipsă şi sunt clasificate din perspectiva jucătorului X, şi anume: victorie (engl. win), înfrângere (engl. loss) sau egalitate (engl. draw). Din acest set, pentru experimente s-au folosit doar 300 de instanțe, împărțite în 200 instanțe pentru antrenare şi 100 pentru test.

### Clasificarea ciupercilor

Acest set de date conține o descriere a unor exemple de ciuperci făcând parte din 23 de specii din familiile Agaricus şi Lepiota [10]. Fiecare exemplu este clasificat ca fiind o ciupercă otrăvitoare, bună de mâncat sau suspectă. Setul de date are 8.124 instanțe, fiecare având câte 22 de atribute. Aceste atribute au ca valori clasificările obținute pe baza criteriului ce definește atributul. Setul conține 2.480 de valori lipsă în coloanele atributelor şi 124 în coloana clasei specificând faptul că ciupercile respective nu au putut fi clasificate exact. Din acest set, s-au reținut pentru teste numai instanțele ce au valori în coloana rezultatelor (8.000 exemple), acestea

fiind împărţite în 7.000 instanţe de antrenare şi 1.000 instanţe de test.

**Jocul X şi 0**

Acest set de date codifică toate configuraţiile posibile ale unei table de joc de 3 x 3 pătrăţele pentru jocul „X şi 0" în care se presupune că jucătorul X este primul care joacă [11]. Setul conţine 958 de instanţe, fiecare având câte 9 atribute corespunzătoare celor 9 pătrăţele de pe tabla de joc. La fel ca la jocul connect-4 prezentat anterior, şi aici atributele pot avea valoare X, 0 sau B având aceeaşi semnificaţie ca mai sus. Exemplele nu conţin valori lipsă iar clasificarea folosită este binară, reprezentând capacitatea jucătorului x de a obţine victoria: pozitiv (victorie) sau negativ (înfrângere sau egalitate). Din acest set se folosesc doar 250 de instanţe, grupate în setul de antrenament (200 instanţe) şi setul de test (50 instanţe).

**Formarea diminutivelor în limba olandeză**

Acest set de date descrie modul de formare a diminutivelor în limba olandeză [12]. Setul de date conţine 3.949 de instanţe, fiecare având câte 12 atribute. Atributele au ca valori clasificările diminutivelor realizate pe diferite criterii, iar clasificarea instanţelor se poate face în cinci clase distincte. Setul de date este complet specificat (nu are valori lipsă) şi a fost împărţit într-o parte de antrenare de 2.999 de instanţe şi o parte de test de 950 de instanţe.

**Setul 5**

Acest set de date a fost creat în scopuri didactice şi descrie o clasificare binară bazată pe şase atribute care sunt şi ele tot binare [8]. Setul de date conţine 800 de instanţe de antrenare şi 203 instanţe de test.

**PREZENTAREA REZULTATELOR**

Pentru a evalua îmbunătăţirile propuse, s-au construit două sisteme: unul care calculează clasa unei instanţe de test folosind algoritmul clasic de KNN şi altul care foloseşte îmbunătăţirile algoritmului KNN propuse în secţiunile anterioare. Cele două sisteme au fost aplicate în paralel asupra fiecăruia din cele 5 seturi de date prezentate anterior.

Pentru varianta îmbunătăţită de KNN s-au considerat mai multe funcţii de ponderare posibile – sin(x), exp(-x), sum(x) – pentru a vedea care este cea mai bună alegere. De asemenea au fost considerate combinaţii ale lui „k" cu distanţa medie a exemplelor de antrenament faţă de instanţa de test.

S-a observat că dintre toate aceste funcţii, cele mai bune rezultate au fost obţinute în cazul în care se foloseşte votarea între clasele returnate de aplicările algoritmului KNN pentru diferite valori ale lui „k", fără a considera nici un fel de funcţie de ponderare între termenii implicaţi în votare.

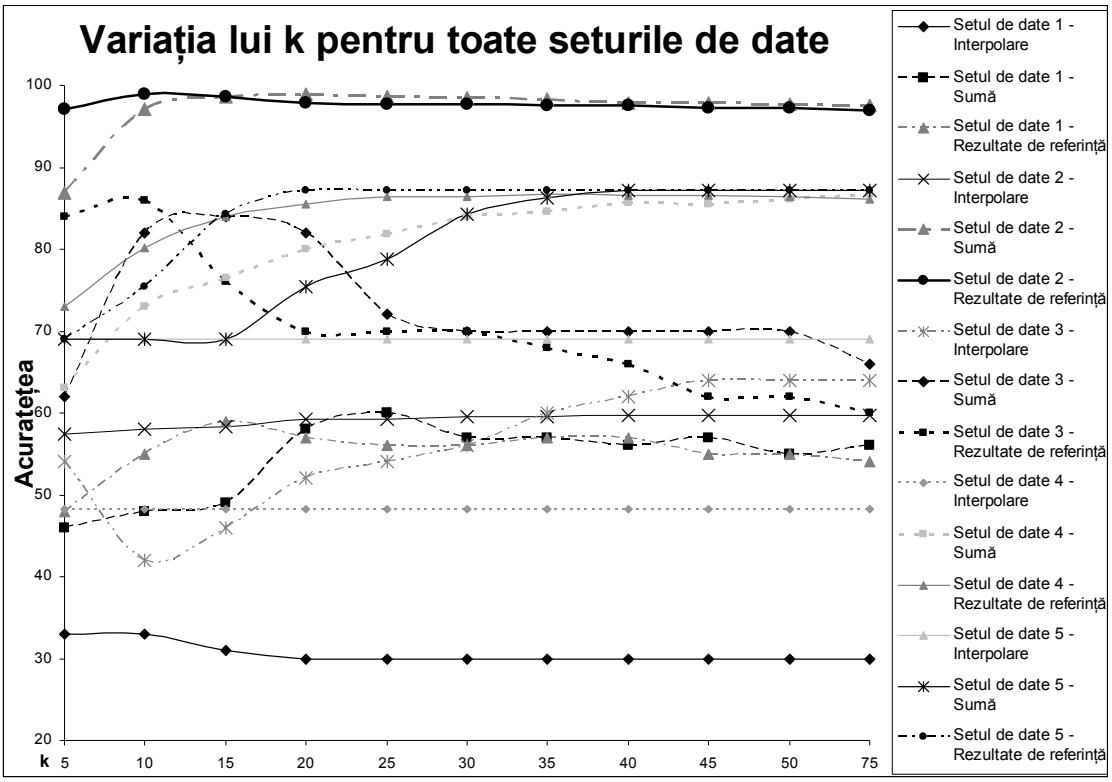


**Figura 2 Variaţia lui „k" pentru toate seturile de date**

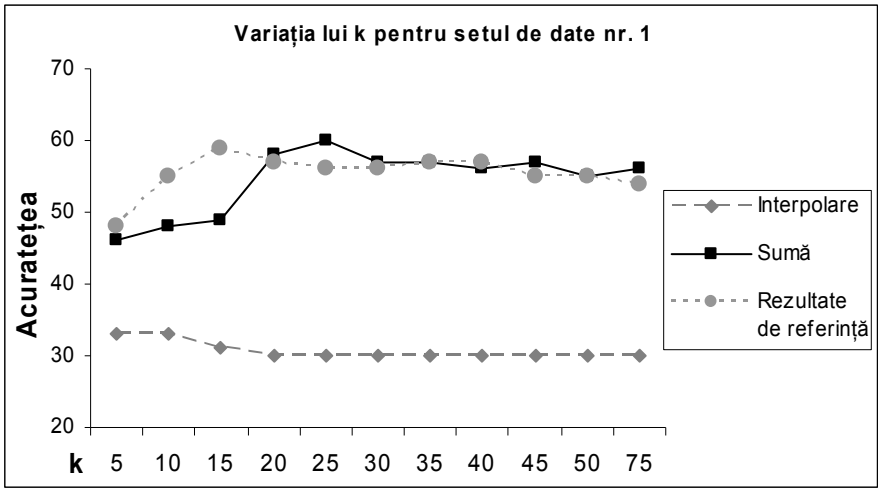


**Figura 3 Variația lui „k" pentru setul de date numărul 1**

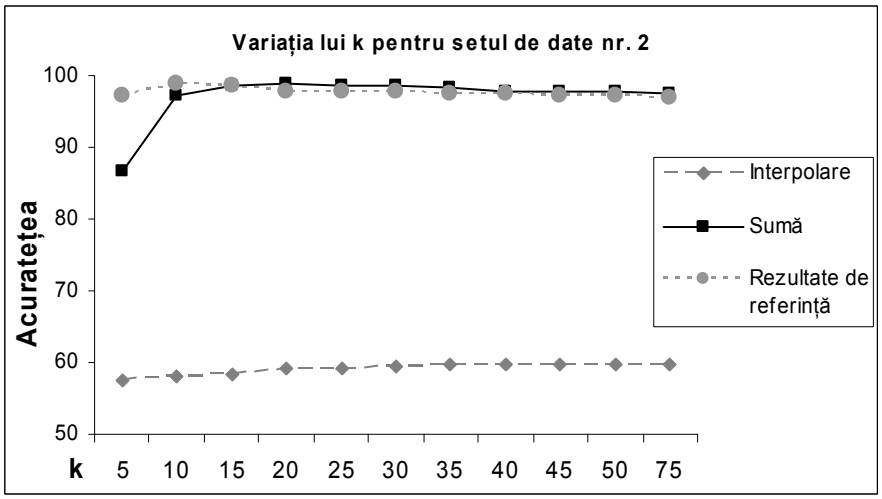


**Figura 4 Variația lui „k" pentru setul de date numărul 2**

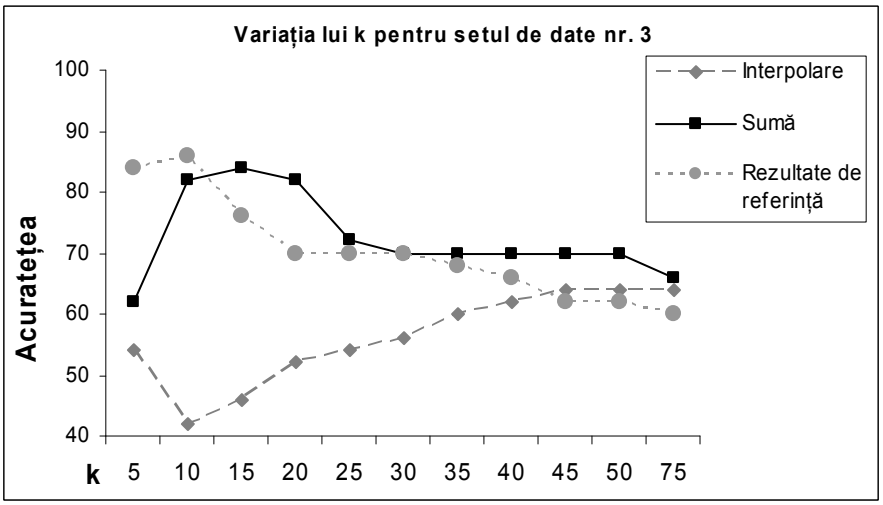


**Figura 5 Variația lui „k" pentru setul de date numărul 3**

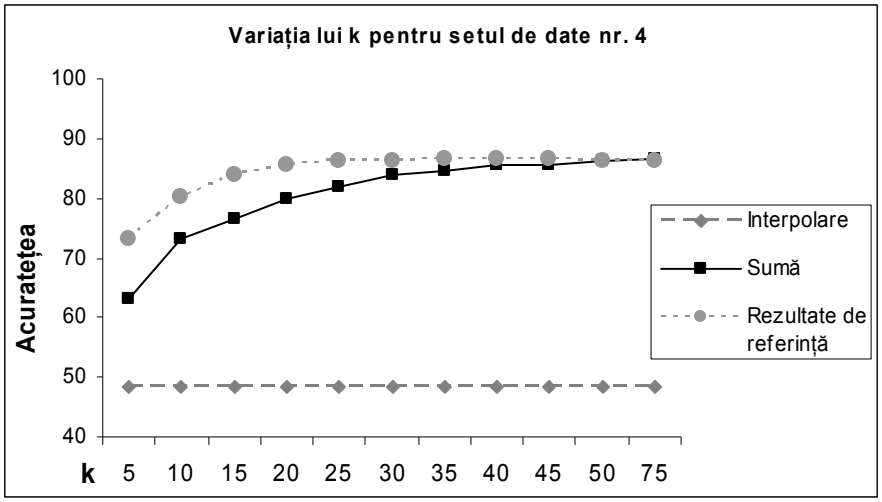


**Figura 6 Variația lui „k" pentru setul de date numărul 4**

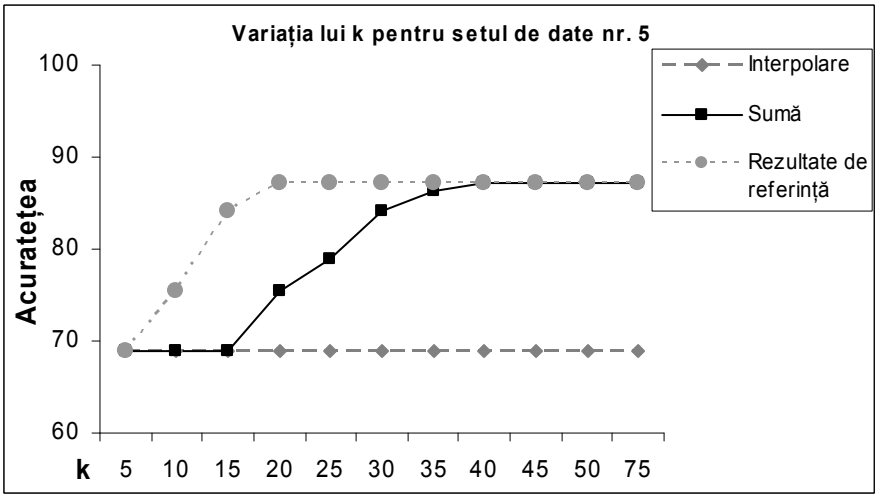


**Figura 7 Variația lui „k" pentru setul de date numărul 5**

Funcția considerată pentru interpolare în graficele prezentate mai sus a fost sin (1/(K*Avg(D))). În graficul de mai jos prezentăm alte funcții de ponderare încercate. Pentru testarea diferitelor funcții de interpolare, am reprezentat în grafic doar rezultatele obținute pe setul de date numărul 1 (Jocul Connect-4), pentru celelalte seturi de date curbele fiind similare.

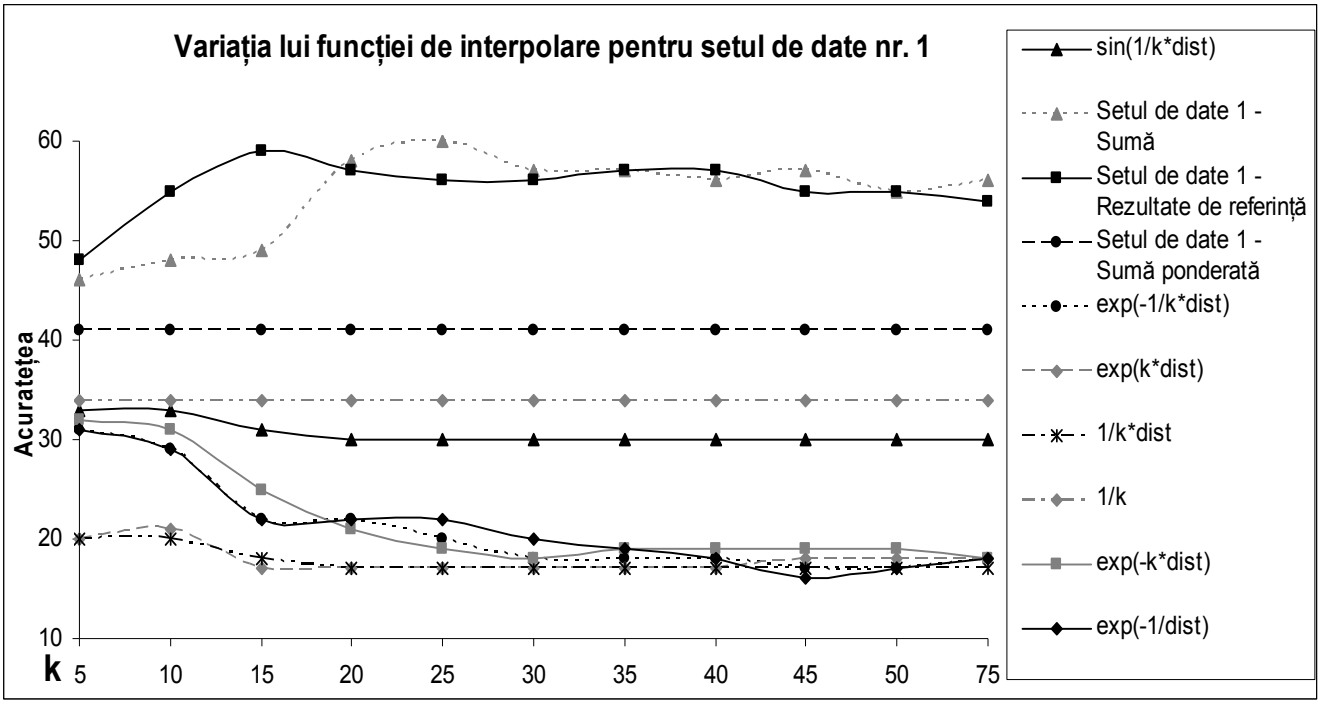


**Figura 8 Variația funcțiilor de interpolare pentru setul de date numărul 1**

## CONCLUZII

În urma cu 100 de ani, Henry Ford a revoluționat industria manufacturieră prin introducerea producției de masă (fabricarea în serie a mai multor produse identice). În prezent, producția de masă a devenit un model învechit, companiile trebuind să ofere produse şi servicii care să corespundă nevoilor cumpărătorilor [2]. În acest context, dezvoltarea de tehnologii avansate de recomandare ce pot fi aplicate unor produse şi servicii configurabile astfel încât nevoile utilizatorilor să fie satisfăcute cât mai bine şi mai uşor reprezintă un nou pas înainte. Unul dintre cei mai utilizați algoritmi în construcția de sisteme de recomandare este algoritmul KNN şi orice îmbunătățire ce se poate aduce acestui algoritm poate atrage după sine o îmbunătățire a recomandărilor generate de sistemele care se bazează pe el. În cele ce urmează vom prezenta concluziile autorilor relativ la experimentele efectuate cu scopul de a îmbunătăți performantele algoritmului KNN.

Graficele prezentate în secțiunea anterioară permit realizarea unei clasificări a metodelor descrise din punctul de vedere al acurateței. Astfel, cea mai mare acuratețe se obține fie folosind algoritmul clasic de KNN, fie suma clasificărilor rezultate din aplicarea algoritmului KNN pentru diferite valori ale lui „k”. Pe următorul nivel de acuratețe s-a situat o variantă de ponderare în funcție de valoarea lui „k” a sumei clasificărilor rezultate din aplicarea algoritmului KNN pentru diferite valori ale lui „k”.

Valori mai mici ale acurateței au fost obținute în momentul în care s-a încercat combinarea a două mărimi de ponderare (valoarea lui „k” şi distanța medie față de instanța de test). Acest comportament este explicat de faptul că în momentul în care cele două mărimi sunt combinate, se combină şi efectele pe care acestea le au asupra rezultatelor.

Cea mai slabă acuratețe a fost obținută de funcția exp(k*dist) deoarece aceasta este o funcție construită greşit din punct de vedere conceptual, dar care a fost intenționat folosită pentru a se vedea rezultatele obținute.

Testele referitoarea la ponderare au demonstrat că această abordare nu este foarte utilă, deoarece în acest fel se diminuează mult prea mult importanța instanțelor mai puțin similare cu cea de test. Importanța acestora este deja micşorată prin considerarea lor de mai puține ori decât cele apropiate de instanța de test, aşa că o nouă ponderare nu mai este necesară.

Testele efectuate pentru a identifica necesitatea considerării ca valoare de ponderare a distanței medii între instanța de test şi cele de antrenare au arătat că situația descrisă în exemplul din Figura 1 este un caz particular şi că în majoritatea cazurilor, considerarea acesteia are ca efect micşorarea acurateței. De aceea, considerăm că este mai bine să se evite această modalitate de ponderare.

Din rezultatele obținute se pot trage câteva concluzii importante:

rezultatele obținute folosind metoda votării între clasele returnate de aplicările algoritmului KNN pentru diferite valori ale lui „k”au fost mai bune decât cele returnate de algoritmul clasic de KNN pentru valori mari ale lui „k”;

cea mai bună soluție este să nu se folosească ponderări ale rezultatelor claselor întoarse de către algoritmul KNN, deoarece fiecare astfel de funcție de ponderare introduce efecte negative;

în cazul în care se combină mai multe funcții de ponderare se observa o scădere mai mare a acurateței comparativ cu situațiile în care acestea se folosesc individual;

ponderarea claselor întoarse de aplicările algoritmului KNN (cu diferite valori ale lui „k”) folosind ca măsură de

ponderare valoarea lui „k” este mai puțin dăunătoare pentru acuratețe decât ponderarea folosind distanța medie a exemplelor de antrenament față de instanța de test;

atunci când valoarea lui „k” devine prea mare, se încearcă construcția unui sistem mult prea simplu ce nu mai corespunde realității.

## MULȚUMIRI

Mulțumim UC Irvine Machine Learning Repository pentru punerea la dispoziție a seturilor de date folosite la testarea variantei de îmbunătățire a algoritmului KNN propusă în lucrarea de față.

Cercetările prezentate în această lucrare au fost parțial finanțate din proiectul European FP7 STREP LTfLL şi din proiectul național CNCSIS K-Teams.

## REFERINȚE

1. Asuncion and D.J. Newman, (2007). UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science: http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Felfernig, A., Friedrich, G., Schmidt-Thieme, L.: Recommender Systems. In IEEE Intelligent Systems Special Issue on Recommender Systems, Vol. 22(3) (2007)
3. Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating collaborative filtering recommender systems. In: ACM Trans. Information Systems, vol. 22, no. 1, pp. 5-53 (2004)
4. T. Hofmann. Latent Semantic Models for Collaborative Filtering. ACM Trans. Information Systems, vol. 22, no. 1, 2004, pp. 89–115.
5. J. Konstan et al. GroupLens: Applying Collaborative Filtering to Usenet News. Comm.ACM, vol. 40, no. 3, 1997, pp. 77–87.
6. T. Mitchell. Machine Learning. McGraw Hill, 1997
7. Situl cursului de tehnologia limbajului al Universității din Antwerp: http://ilk.uvt.nl/~antalb/ltua/week2.htm
8. Situl cursului de învățare automată (CS6375) al Universității din Texas la Dallas: http://www.hlt.utdallas.edu/~vince/cs6375/assignments/hw1/
9. http://archive.ics.uci.edu/ml/datasets/Connect-4
10. http://archive.ics.uci.edu/ml/datasets/Mushroom
11. http://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame
12. http://ilk.uvt.nl/~antalb/ltua/dimin.data.arff