Raport științific

1 Sumar

Obiective și rezultate estimate:

Etapa 1 - Modele bazate pe teoria jocurilor pentru clasificare de date cu numar mare de atribute

1.1 - Recenzia abordărilor recente legate de teoria jocurilor, extragerea atributelor și clasificarea

1.2 - Extinderea FROG pentru a face față unui număr mare de caracteristici din date. Explorați metode de boosting.

1.3 - Ingineria atributelor folosind metode de clustering de retele și de teoriei jocurilor

1.4 - Diseminarea rezultatelor

1.5 - Management de proiect

Livrabile:

- 2 articole trimise spre publicare
- raport de cercetare
- pagina web a proiectului: `https://www.cs.ubbcluj.ro/~mihai-suciu/cgtfe/`

Rezultate obținute:

- În lucrarea [P1] s-a propus o metodă de selectare a atributelor bazata de un arbore de decizie construit folosind conceptul de echilibru Nash;
- În lucrarea [P2] s-a propus un algoritm genetic (GA) selectarea de atribute; evoluarea și evoluția indivizilor se face pe baza informațiilor oferite de arbori de decizie evoluați folosind atributele oferite de GA.

Articole trimise spre publicare:

P1 : Mihai Suciu, Rodica Ioana Lung. A new Filter Feature Selection Method based on a Game Theoretic Decision Tree. HIS 2022. (articol acceptat)

P2 : Mihai Suciu, Rodica Ioana Lung. Feature Selection based on a Decision Tree Genetic Algorithm. evoAPPS 2023 : EvoApplications 2023 - 26th European Conference on the Applications of Evolutionary and bio-inspired Computation. (articol in evaluare)

2 Rezumat executiv

Direcția principală de cercetare se axează pe combinarea arborilor de decizie cu modele din teoria jocurilor pentru a rezolva problema selecției atributelor relevante pentru problema de clasificare binară. Problema selecției atributelor a devenit o activitate cheie în cadrul învățării automate. Pentru problemele de clasificare, se știe că reduce complexitatea de calcul a estimării parametrilor, dar adaugă și o contribuție importantă la aspectele de înțelegere și explicare a rezultatelor.

Un arbore de decizie bazat pe un model de joc este utilizat pentru selectarea atributelor. În timpul fazei de inducție a arborelui, atributul utilizat la împărțirea datelor este ales pe baza unui joc între instanțe din aceeași clasă. Presupunerea abordării este: componenta de joc va indica cele mai importante atribute. O măsură pentru importanța unui atribut este calculată pe baza numărului de apariții ale unui atribut și de adâncimea din arbore. Rezultatele sunt comparabile și mai bune în unele cazuri decât cele raportate printr-o abordare standard bazată și pe copaci.

A doua abordare analizată propune un algoritm genetic pentru selecția atributelor. Importanța, precum și eficacitatea atributelor alese de fiecare individ, este evaluată prin utilizarea arborilor de decizie. Importanța atributului indicată de arborele de decizie este utilizată în timpul selecției și recombinării. Arborele indus de cel mai bun individ din populație este folosit pentru clasificare. Experimentele numerice ilustrează comportamentul abordării.

Datorită legăturii structurale naturale dintre arborii de decizie și rețelele neuronale (NN), în ultimii ani s-au dezvoltat un tip de arbori de clasificare și regresie care folosesc pentru separarea datelor din noduri funcții împrumutate din NN, cum ar fi funcția sigmoid [31]. Urmând direcția folosirii arborilor de decizie pentru selectarea de atribute, se studiază efectul folosirii FROG/probit pentru separarea datelor în construcția arborelui și eficiența acestei abordări în identificarea importanței atributelor. O altă direcție studiată constă în folosirea unui algoritm random forest pentru extragerea atributelor, prin agregarea datelor din frunze și folosirea unei variante extinse FROG pentru probleme de clasificare cu mai multe clase împreuna cu un mecanism de boosting pentru atribute. Un articol care prezintă această abordare va fi trimis spre publicare la începutul anului 2023 la o revistă.

Arborii de decizie pentru clasificare sunt în general construiți într-o manieră recursivă, ”top-down”, pornind de la nodul rădăcină. Într-o abordare opusă, construcția arborilor se poate face și ”bottom-up”, prin popularea inițială a frunzelor cu date și agregarea lor mergând către rădăcină. Această abordare folosește un algoritm de *clustering* pentru a crea frunzele, împreună cu un algoritm care reprezintă separarea sub forma unui hiper-plan, cum ar fi SVM pentru a agrega datele. În cadrul proiectului se explorează folosirea jocului propus în [P1] pentru construirea unui arbore de decizie de tip bottom-up pornind de la datele din frunze. Distanța minimă dintre clusterii din frunze (sau fiecare nivel de nod) este calculată și clusterii cei mai apropiați sunt agregați, generându-se o regulă de separare a datelor agregare folosind echilibrul Nash. Importanța atributelor se calculează pe baza arborelui astfel construit. Avantajul acestei abordări constă în controlul mai mare asupra mărimii arborelui și a purității datelor din frunze.

3 Descrierea științifică

Principalele rezultate trimise spre publicare sunt prezentate mai jos, obiectivele etapei I fiind îndeplinite.

3.1 O nouă metodă de selecție a atributelor relevante bazată pe un arbore de decizie generat folosind concepte din teoria jocurilor

3.1.1 Introducere

Unul dintre pașii cheie în analiza datelor este reprezentat de selecția atributelor. Orice decizie luată pe baza rezultatelor unei analize trebuie să țină cont de limitările care reies în mod natural din date, precum și de metodele utilizate pentru a decide care sunt caracteristicile care sunt efectiv analizate. În timp ce selecția atributelor este obligatorie în contextul datelor mari, beneficiile acesteia pot fi avute în vedere și pe seturi de date mai mici pentru care reprezintă un prim pas în explicarea intuitivă a modelului de bază.

Metodele de selecție a atributelor [7] sunt, în general, clasificate în trei grupe: metode de filtrare, în care caracteristicile sunt selectate pe baza unor metrici care indică importanța lor [15], metode *wrapper* care iau în considerare subseturile setului de atribute evaluate prin potrivirea unui modelul de clasificare [14] și modelele încorporate care realizează intrinsec selecția caracteristicilor în timpul etapei de adaptare, de ex. arbori de decizie și păduri de arbori de decizie [28].

Arborii de decizie (*decision trees*) sunt, de asemenea, folosiți pe scară largă pentru a valida metodele de selecție a atributelor [11]. Diverse aplicații din lumea reală folosesc arbori de decizie pentru testarea și validarea metodelor de selecție a filtrelor. De exemplu, în detectarea intruziunilor în rețea [22], predicția stocului [25], predicția azotului în stații de ape [1], *code smell* [13] sunt câteva din abordările unde modelele bazate pe arbori de decizie au prezentat o îmbunătățire semnificativă a performanței după selecția atributelor.

Cu toate acestea, selecția atributelor în sine bazată pe inducerea arborelui de decizie este una dintre cele mai intuitive abordări pentru a evalua importanța atributelor unui set de date. Deoarece arborii de decizie sunt construiți recursiv și, la fiecare nivel de nod, trebuie alese anumite atribute pe baza cărora se vor împărți datele, este firesc să evaluăm că atributele implicate în procesul de împărțire sunt importante în explicarea datelor. Cu toate acestea, majoritatea metodelor de selecție a atributelor care se bazează pe arbori de decizie utilizează în cele din urmă o formă de pădure aleatoare, adică arbori multipli induși pe date și atribute eșantionate, în diferite forme și pentru diferite aplicații [12, 21, 28].

Credem că mai este loc de explorat în utilizarea unui singur arbore de decizie pentru selecția atributelor unui set de date, deoarece performanța oricărei abordări depinde în mod natural de metoda de inducție a arborelui. Propunem utilizarea unui arbore de decizie care împarte datele pe baza unei abordări teoretice de joc pentru a calcula importanța unei caracteristici și a o utiliza pentru selecție. Comparăm abordarea noastră cu o metodă de filtrare pentru o pădure aleatorie pe seturi de date sintetice și reale.

3.1.2 Selecția atributelor pe baza unui arbore de decizie și un model de joc (G-DTfs)

Fie un set de date $(\mathcal{X}, \mathcal{Y})$, cu $\mathcal{X} \subset \mathbb{R}^{n \times d}$ și $\mathcal{Y} \subset \{0, 1\}^n$, astfel încât fiecare instanță $x_i \in \mathcal{X}$ are eticheta $y_i \in \mathcal{Y}$. Dacă $\mathcal{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_d)$, cu $\mathbf{X}_j \in \mathbf{R}^n$, dorim să găsim o submulțime de $\{\mathbf{X}_1, \ldots, \mathbf{X}_d\}$ care explică cel mai bine etichetele $\mathcal{Y}$.

În această secțiune propunem utilizarea următorului arbore de decizie bazat pe teoria jocurilor pentru a identifica caracteristicile/atributele lui X care sunt cele mai influente în separarea datelor în cele două clase. La fiecare nivel de nod, atributul folosit pentru a împărți datele este ales prin simularea unui joc între cele două clase. Arborele este construit recursiv, de sus în jos, începând cu întregul set de date (de antrenament) de la nodul rădăcină. Următorii pași sunt utilizați pentru a împărți datele nodului curent (X, Y).

Verificarea datelor În primul rând se verifică dacă datele din nod trebuie să fie împărțite sau nu. Condiția folosită este: dacă toate instanțele au aceeași etichetă, sau dacă X conține o singură instanță, nodul devine o frunză.

3.1.3 Împărțirea datelor la nivel de nod pe baza jocului

Dacă datele (X, Y) dintr-un nod trebuie împărțite, se calculează un hiperplan paralel al axei pentru fiecare atribut $j = \overline{1, d}$ din date în felul următor.

Jocul nodului Se ia în considerare următorul joc $\Gamma(X, Y|j)$ compus din:

- jocul are doi jucători, L și R, corespunzător celor două sub-noduri (nodurile fiu ale nodului părinte din arborele binar), respectiv celor două clase;
- strategia fiecărui jucător este să aleagă un parametru pentru hiperplan, β: β_L și, respectiv, β_R;
- câștigul fiecărui jucător este calculat în felul următor:

$$u_L(\beta_L, \beta_R|j) = -n_0 \sum_{i=1}^{n} (\beta_{1|j} x_{ij} + \beta_{0|j})(1 - y_i),$$

și

$$u_R(\beta_L, \beta_R|j) = n_1 \sum_{i=1}^{n} (\beta_{1|j} x_{ij} + \beta_{0|j}) y_i,$$

unde $\beta = \frac{1}{2}(\beta_L + \beta_R)$, n_0 și n_1 reprezintă numărul de instanțe având etichetele 0 și, respectiv, 1.

Câștigul jucătorului din stânga însumează coeficienții care vor fi utilizați în construcția hiperplanului pentru toate cazurile cu eticheta 0 și minimizează această sumă înmulțită cu numărul lor pentru a deplasa produsele la stânga axei. Într-o manieră similară, suma corespunzătoare pentru instanțe care au eticheta 1 este maximizată pentru a le deplasa cât mai departe de instanțele cu cealaltă etichetă. Coeficientul β folosit pentru a calcula câștigurile este de fapt o combinație liniară a strategiilor celor doi jucători.

Echilibrul Nash (*Nash Equilibria* - NE) al acestui joc este reprezentat de o valoare β care combină β_L și β_R în așa fel încât niciunul dintre jucători să nu își mai poată muta sumele de produse la stânga sau la dreapta, respectiv, în timp ce celălalt își menține alegerea neschimbată. Echilibrul jocului poate fi aproximat folosind o imitație a procedurii de joc fictiv (*fictitious play*) [5].

Aproximarea echilibrului Nash Versiunea simplificată de joc fictiv folosită aici pentru a găsi o valoare potrivită pentru β este implementată după cum urmează: pentru un număr de η iterații, cel mai bun răspuns al fiecărui jucător față de strategia celuilalt jucător este calculat folosind un algoritm de optimizare. Deoarece ne propunem doar să aproximăm valorile β care împart datele într-un mod rezonabil, căutarea se oprește după ce a trecut numărul de iterații. În fiecare iterație, cel mai bun răspuns la media strategiilor celuilalt jucător în iterațiile anterioare este considerat drept cel fix. Procedura este prezentată în algoritmul 1.

Algorithm 1 Aproximarea echilibrului Nash

Input: X, Y - data to be split by the node; j - attribute evaluated
Output: $X_{L|j}, y_{L|j}, X_{R|j}, y_{R|j}$, and β_j to define the split rule for the node based on attribute j;
Initialize β_L, β_R at random (standard normal distribution)
for η iterations: **do**
  Find $\beta_L = \underset{b}{\operatorname{argmin}}\, u_L(b, \beta_R)$;
  Find $\beta_R = \underset{b}{\operatorname{argmin}}\, u_R(\beta_L, b)$;
end for
$\beta_j = \frac{1}{2}(\beta_L + \beta_R)$
$X_{L|j} = \{x \in X | x_j^T \beta \leq 0\}$, $y_{L|j} = \{y_i \in y | x_i \in X_{L|j}\}$
$X_{R|j} = \{x \in X | x_j^T \beta > 0\}$, $y_R = \{y_i \in y | x_i \in X_{R|j}\}$

Alegerea unui atribut pe baza echilibrului Nash Pentru a alege atributul folosit pentru a împărți datele nodurilor, NE pentru fiecare atribut $j = \overline{1, d}$ sunt aproximate și datele corespunzătoare ale sub-nodurilor sunt separate și evaluate în continuare pe baza câștigului de entropie. Oricare atribut care întoarce cel mai mare câștig de entropie este ales pentru împărțirea datelor și β_j corespunzător este utilizat pentru a defini hiperplanul de separare.

3.1.4 Atribuirea importanței unui atribut în problema selecției atributelor

Odată ce arborele a fost indus/generat, importanța fiecărui atribut în împărțirea datelor poate fi luată în considerare în funcție de dacă caracteristica este utilizată pentru împărțire și de adâncimea nodului care o folosește. Pentru fiecare atribut $j \in \{1 \dots d\}$ notăm cu

$$v_j = \{v_{jl}\}_{l \in I_j},$$

mulțimea care conține nodurile care împart datele pe baza atributului j, cu I_j setul de indici corespunzători din arbore și fie $\delta(v_{jl})$ adâncimea nodului v_{jl} în arborele de decizie, cu valori care încep de la 1 (nodul rădăcină). Atunci, importanța $\phi(j)$ a atributului j poate fi calculată ca:

$$\phi(j) = \begin{cases} \sum_{l \in I_j} \frac{1}{\delta(v_{jl})}, & I_j \neq \emptyset \\ 0, & I_j = \emptyset \end{cases}. \tag{1}$$

Astfel, importanța unui atribut depinde de adâncimea nodului care îl folosește pentru a împărți datele. Presupunem că atributele care sunt utilizate la începutul inducției arborelui de decizie pot fi mai influente. De asemenea, un atribut care poate apărea pe mai multe noduri cu o adâncime mai mare poate fi influent, iar indicatorul $\phi()$ cuprinde și această situație.

3.1.5 Experimente numerice

Experimentele numerice sunt efectuate pe seturi de date sintetice și reale cu diferite grade de dificultate pentru a ilustra stabilitatea și performanța abordării propuse.

Cadrul experimental

Datele de test Generăm și folosim seturi de date sintetice cu diferite grade de dificultate pentru a testa stabilitatea *G-DTfs*. Pentru reproductibilitate și control asupra setului de date sintetice generate, folosim funcția `make_classification` din biblioteca Python `scikit-learn`[1] [17]. Pentru a varia dificultatea seturilor de date generate, folosim valori diferite pentru numărul de instanțe și numărul de atribute.

Pentru seturile de date din lumea reală folosim setul de date Connectionist Bench (Sonar, Mines vs. Rocks) (R1) care are 208 instanțe și 60 de atribute, setul de date pentru Clasificarea bolii Parkinson (R2) care are 756 de instanțe și 754 de atribute și setul de date Musk (versiunea 1) (R3) care are 476 de instanțe și 168 de atribute. Seturile de date sunt preluate din baza de date UCI Machine Learning [8].

Toate seturile de date utilizate necesită clasificarea binară a instanțelor de date și prezintă grade diferite de dificultate.

Parametrii folosiți Pentru seturile de date sintetice folosim parametrii pentru funcția `make_classification`: numărul de instanțe (250, 500, 1000), numărul de atribute (50, 100, 150), seed (500), ponderea fiecărei etichete (0.5 - seturile de date sunt echilibrate) și separarea claselor (0.5 - există suprapunere intre instanțe din clase diferite). Creeăm seturi de date pentru teste folosind toate combinațiile ale parametrilor descriși anterior.

Pentru G-DTfs testăm diferiți parametri: adâncimea maximă a unui arbore (5, 10, 15), numărul de iterații pentru jocul fictiv (5).

Împărțim fiecare set de date, sintetice sau reale, în $M = 10$ subseturi. Raportăm rezultatele G-DTfs și abordarea comparată pe 10 rulări independente pentru fiecare set de date utilizat.

Comparăm rezultatele obținute de G-DTfs cu atributele alese de un clasificator *Random Forest* (RF) [4]. Pentru clasificatorul RF setăm parametrii: numărul de estimatori (100), criteriul de împărțire dintr-un nod (indexul *gini*), adâncimea maximă a fiecărui estimator (acest parametru ia aceeași valoare ca adâncimea maximă a G-DTfs).

Evaluarea performanțelor Pentru a evalua performanța G-DTfs, se folosește indicatorul de stabilitate, SC, [3, 16]. Indicatorul de stabilitate se bazează pe corelația Pearson între rezultatele raportate pe datele eșantionate și indică dacă metoda de selecție a atributelor este stabilă, adică cât de diferite/asemănătoare sunt atributelor alese pe baza unor eșantioane diferite din aceleași date. Deoarece aceasta este o caracteristică dorită a unei metode de selecție a atributelor, o folosim aici pentru a compara rezultatele raportate de G-DTfs cu o abordare standard *Random Forest* (RF) pentru selecția atributelor [17].

Pentru a calcula măsura stabilității, setul de date este împărțit în M subseturi prin utilizarea reeșantionării, iar metoda de selecție a atributelor este aplicată pe fiecare subset, rezultând M seturi de atribute, care sunt reprezentate ca vectori Z_i, $i = \overline{1, M}$, cu z_{ij} având valoarea 1 dacă atributul j a fost ales pe eșantionul i și 0 în caz contrar. Măsura stabilității face o medie a

[1] versiunea `1.1.1`

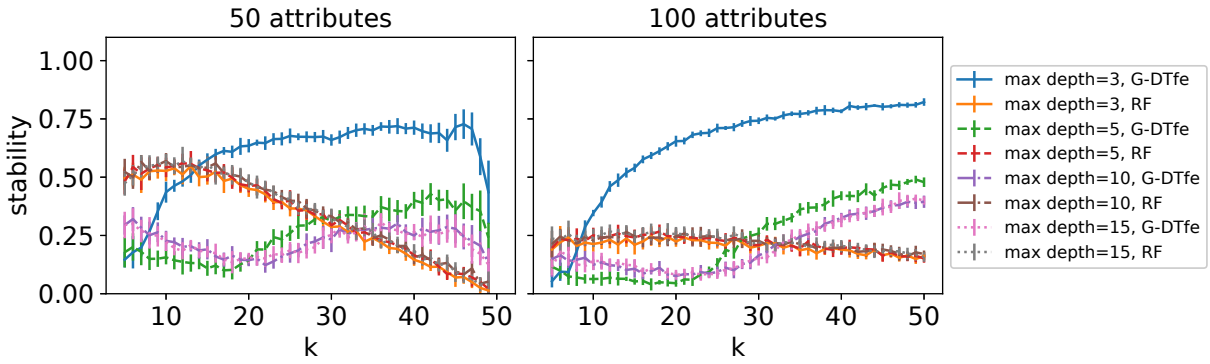

Figura 1: Efectul parametrului k asupra stabilității selecției atributelor pentru modelele G-DTfs și RF cu valori diferite pentru parametrul de adâncime maximă a arborelui de decizie $(3, 5, 10, 15)$ pe seturi de date sintetice cu 50 de atribute (stânga) și 150 de atribute (dreapta).

corelațiilor dintre toate perechile de vectori atribute, adică:

$$SC = \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} Cor(Z_i, Z_j), \tag{2}$$

unde $Cor(Z_i, Z_j)$ arată corelația liniară dintre Z_i și Z_j. O corelație ridicată indică faptul că aceleași atribute sunt identificate ca fiind influente pentru toate eșantioanele, în timp ce o valoare de corelație apropiată de 0 ar indica aleatoriu în selecția atributelor. Dacă scorul este utilizat pentru a evalua metodele de selecție a atributelor, acesta indică care dintre ele este mai stabilă, cu cât scorul este mai mare, cu atât mai bine.

Rezultate numerice Rezultatele sunt prezentate ca medie și abatere standard a scorului SC raportat de cele două metode pentru diferitele combinații ale parametrilor testați (Tabelele 1 și 2). Rezultatele unui test t statistic care compară scorurile de stabilitate raportate de cele două metode însoțesc datele. Pentru seturile de date sintetice, observăm că în 14 setări (combinații diferite de parametrii) rezultatele obținute de G-DTfs sunt semnificativ mai bune. De asemenea, în majoritatea cazurilor, testul t este de prisos, deoarece diferențele indicate de valorile medii și deviației standard sunt evident semnificative. Acest lucru este totuși adevărat în ambele sensuri: ori de câte ori rezultatele RF sunt mai bune, diferența este, de asemenea, evident semnificativă.

Aceeași situație apare și în cazul datelor din lumea reală (Tabel 2), cu observația suplimentară că creșterea numărului de atribute luate în considerare pare să scadă performanța RF și o crește pe cea a G-DTfs, în termeni de stabilitate. Deși este adevărat că se dorește un număr minim de atribute, trebuie luat în considerare și comportamentul unei metode atunci când se confruntă cu un număr mare de instanțe și atribute.

Efectul mărimii setului de atribute k asupra rezultatelor G-DTfs este ilustrat în Figura 1 pentru două seturi de date sintetice, cu 50 și 100 de atribute, comparativ cu cel al RF. Găsim măsuri de stabilitate mai mari pentru G-DTfs cu adâncime mică a arborelui (adâncime maximă de 3) și, de asemenea, tendința de scădere a măsurii de stabilitate pentru RF. Influența adâncimii arborelui asupra acelorași seturi de date este ilustrată în Figura 2 pentru diferite valori k. Rezultatele prezentate în aceste cazuri confirmă că scorul de stabilitate nu depinde de dimensiunea arborelui după un anumit prag, care, pentru aceste seturi de date, este în jur de 5.

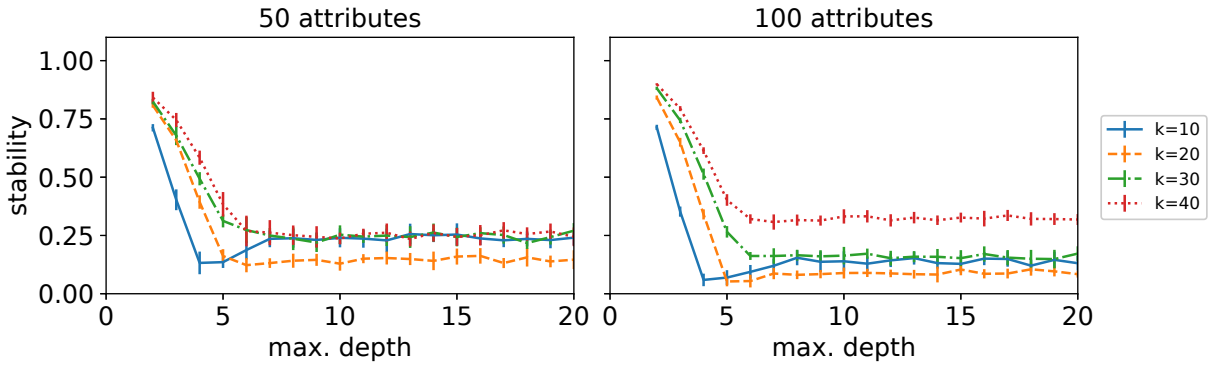

Figura 2: Efectul parametrului adâncime maximă a arborelui de decizie asupra stabilității selecției de atribute pentru G-DTfs pentru seturi de date sintetice cu 50 de atribute (stânga) și 150 de atribute (dreapta) și valori diferite pentru parametrul k $(10, 20, 30, 40)$.

3.1.6 Concluzii

Problema identificării atributelor cheie care pot fi utilizate pentru a explica o caracteristică a datelor este una centrală în învățarea automată. Similar altor sarcini de învățare automată, eficiența și simplitatea sunt dorite din abordările practice. În această lucrare se folosește un arbore de decizie pentru a atribui o măsură de importanță atributelor care pot fi utilizate pentru filtrarea lor. Noutatea abordării constă în utilizarea unui mecanism de divizare teoretică a jocului pentru datele nodurilor în timpul inducției arborelui. Importanța unei caracteristici este atribuită în funcție de poziția nodului (nodurilor) care este utilizat pentru împărțirea datelor. În timp ce utilizarea unui singur arbore de decizie a dat rezultate comparabile și chiar mai bune decât o abordare standard de pădure aleatoare, o direcție deschisă de cercetare constă în explorarea unei păduri de arbori de decizie bazați pe teoretică a jocului pentru selecția atributelor.

3.2 Selecția atributelor bazată pe un arbore de decizie și un algoritm genetic

3.2.1 Introducere

Explorarea diferitelor mecanisme de selecție a atributelor reprezintă un pas cheie în procesul de înțelegere a comportamentului modelelor de clasificare în prezența informațiilor redundante sau incomplete, contribuind, de asemenea, în mare măsură la înțelegerea modelului. Deoarece necesitatea metodelor de selecție a atributelor este evidentă în contextul datelor mari, din punctul de vedere al explicabilității și al atenției la detalii, poate adăuga valoare și atunci când este utilizat în contexte mai mici. În mod ideal, ar putea oferi o perspectivă asupra contribuției fiecărui atribut, sau grup de atribute, la rezultatele generale și să reducă complexitatea de calcul, crescând în același timp explicabilitatea.

Complexitatea problemei de selectare sau identificare a atributelor importante a condus la dezvoltarea mai multor abordări principale: metode de filtrare, în care atributelor li se atribuie o valoare care măsoară importanța lor [15], metode wrapper care aleg seturi de atribute evaluate pe baza performanței modelului de clasificare adaptat pe baza lor [14] și a metodelor încorporate, care efectuează selecția atributelor în timp ce construiesc modelul, cum ar fi arbori de decizie și păduri aleatorii.

Modelele de calcul evolutiv pentru selecția atributelor sunt abundente [6, 27, 30]. Algoritmii

Tabela 1: Rezultatele pentru seturile de date sintetice, abaterea standard medie ± pe zece rulări independente pentru indicatorul de stabilitate pentru G-DTfs și RF. Seturi de date cu 250 de instanțe de date și un număr diferit de atribute ($p_1 : 50, 100.150$), cu diferite valori maxime de adâncime ($p_2 : 5, 10, 15, 20$) și valori diferite pentru k parametru utilizat în procedura de selecție a caracteristicilor ($k : 30, 40$). Un ($-$) indică nicio diferență statistic semnificativă între rezultate, un simbol ($\triangle$) indică faptul că G-DTfs oferă rezultate mai bune din punct de vedere statistic, iar un simbol ($\times$) indică faptul că rezultatele RF sunt semnificativ mai bune.

p_1	p_2	k	G-DTfs	RF	significance
50	5	30	0.33(±0.03)	0.33(±0.03)	-
	5	40	0.39(±0.05)	0.18(±0.04)	$\triangle$
	10	30	0.26(±0.04)	0.32(±0.03)	$\times$
	10	40	0.24(±0.04)	0.18(±0.03)	$\triangle$
	15	30	0.26(±0.05)	0.32(±0.03)	$\times$
	15	40	0.26(±0.04)	0.17(±0.03)	$\triangle$
	20	30	0.24(±0.05)	0.33(±0.03)	$\times$
	20	40	0.26(±0.05)	0.17(±0.03)	$\triangle$
100	5	30	0.26(±0.03)	0.22(±0.03)	$\triangle$
	5	40	0.41(±0.02)	0.20(±0.03)	$\triangle$
	10	30	0.16(±0.03)	0.23(±0.03)	$\times$
	10	40	0.32(±0.03)	0.20(±0.03)	$\triangle$
	15	30	0.16(±0.03)	0.23(±0.02)	$\times$
	15	40	0.33(±0.03)	0.19(±0.02)	$\triangle$
	20	30	0.16(±0.03)	0.24(±0.02)	$\times$
	20	40	0.32(±0.03)	0.20(±0.02)	$\triangle$
150	5	30	0.31(±0.03)	0.28(±0.02)	$\triangle$
	5	40	0.45(±0.02)	0.25(±0.02)	$\triangle$
	10	30	0.18(±0.02)	0.27(±0.03)	$\times$
	10	40	0.35(±0.03)	0.25(±0.02)	$\triangle$
	15	30	0.20(±0.03)	0.27(±0.02)	$\times$
	15	40	0.37(±0.02)	0.25(±0.03)	$\triangle$
	20	30	0.18(±0.02)	0.27(±0.03)	$\times$
	20	40	0.36(±0.03)	0.25(±0.02)	$\triangle$

genetici (GA) au fost prima alegere naturală deoarece reprezentarea binară este potrivită în mod natural pentru această problemă. Multe exemple de algoritmi genetici sunt menționate în [30] cu aplicații în diverse domenii. În special, combinația cu arbori de decizie a fost atrăgătoare de la început [2], urmând multe variante ulterioare, extinzându-se la păduri aleatorii de arbori [10] sau optimizare multiobiectivă [29].

Atributele selectate prin utilizarea GA au fost utilizate în multe aplicații practice, cum ar fi detectarea intruziunilor în rețea [23], chimie [10] Performanța unui arbore de decizie și a unei metode Bayes naive folosind un set de caracteristici generate de algoritm genetic este analizată în [19]. O metodă complexă care combină selecția atributelor și a instanțelor este prezentată în [24].

În ciuda numărului mare de lucrări existente, provocările sunt încă multiple și există loc pentru explorarea comportamentului euristicii de căutare pentru această problemă. În seturile de date reale, atributele sunt rareori cu adevărat independente. Cu toate acestea, ele pot contribui

Tabela 2: Rezultate pentru seturi de date din lumea reală, medie și abatere standard pe zece curse independente pentru indicatorul de stabilitate pentru G-DTfs și RF. Diferite seturi de date din lumea reală (R1-R3) pentru modelele de selecție a atributelor G-DTfs și RF cu diferite valori maxime de adâncime (p_2 - 5, 10) și valori diferite pentru parametrul k utilizat în procedura de selecție a caracteristicilor . Un (−) arată că nu există nicio diferență statistică între modelele testate, un simbol (△) arată că G-DTfs oferă rezultate mai bune din punct de vedere statistic, iar un simbol (×) indică faptul că rezultatele RF sunt semnificativ mai bune.

Data-set	p_2	k	G-DTfs	RF	significance
R1	5	30	0.41(±0.03)	0.39(±0.04)	△
	5	40	0.48(±0.02)	0.31(±0.02)	△
	10	30	0.35(±0.04)	0.40(±0.03)	×
	10	40	0.44(±0.03)	0.31(±0.03)	△
R2	5	30	0.30(±0.03)	0.42(±0.03)	×
	5	40	0.48(±0.02)	0.42(±0.02)	△
	5	100	0.80(±0.01)	0.35(±0.01)	△
	5	150	0.86(±0.01)	0.31(±0.02)	△
	5	200	0.89(±0.01)	0.29(±0.01)	△
	10	30	0.08(±0.01)	0.43(±0.03)	×
	10	40	0.10(±0.01)	0.43(±0.03)	×
	10	100	0.61(±0.01)	0.35(±0.01)	△
	10	150	0.73(±0.01)	0.31(±0.01)	△
	10	200	0.79(±0.01)	0.28(±0.01)	△
R3	5	30	0.34(±0.02)	0.45(±0.02)	×
	5	40	0.52(±0.02)	0.45(±0.02)	△
	5	100	0.77(±0.02)	0.30(±0.02)	△
	5	150	0.78(±0.02)	0.14(±0.02)	△
	10	30	0.12(±0.02)	0.46(±0.01)	×
	10	40	0.17(±0.02)	0.45(±0.02)	×
	10	100	0.57(±0.03)	0.33(±0.02)	△
	10	150	0.51(±0.02)	0.17(±0.03)	△

într-un mod diferit la explicarea unui anumit atribut. Reducerea setului de date este întotdeauna o procedură imprevizibilă care trebuie efectuată cu atenție pentru a se asigura că nu se pierde nicio informație importantă.

Se propune o metodă *wrapper* pe baza unui algoritm genetic (GA) hibridizat cu un mecanism încorporat. Metoda wrapper și mecanismul încorporat folosesc arbori de decizie care oferă populației GA valori de fitness, precum și importanțele atributelor care sunt utilizate în timpul procesului de recombinare și selecție.

3.2.2 fsGA-DT: selecția atributelor folosind algoritmi genetici și arbori de decizie

Problema clasificării binare abordată aici constă în găsirea unei reguli care atribuie clase, sau etichete, datelor, pe baza informațiilor furnizate de un set de antrenament în care sunt cunoscute clase. Fie $X \subset \mathbb{R}^{n \times d}$ un set de date cu n instanțe și d atribute (sau caracteristici) și $Y \subset \{0, 1\}^n$ lor etichetele corespunzătoare. Problema de selecție a atributelor constă în găsirea subsetului de atribute care explică cel mai bine datele, adică care oferă o soluție bună la problema de clasificare. Minimizarea dimensiunii acestui set este, de asemenea, de dorit.

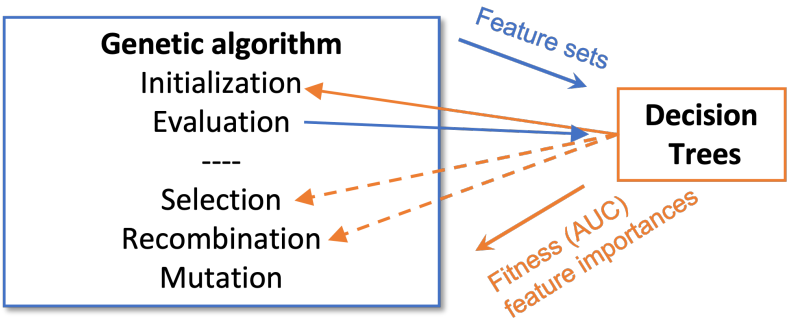

Figura 3: Comunicarea între GA și DT. Inițializarea se realizează pe baza informațiilor oferite exclusiv de DT. În timpul evaluării, seturile de atribute sunt trimise către DT și evaluate, primind o valoare de fitness și importanța atributelor care sunt utilizate în timpul selecției și recombinării.

Arborii de decizie [26] sunt modele de clasificare și regresie care împart spațiul de date în regiuni cât mai pure posibil, adică conținând majoritatea instanțelor din aceeași clasă. Reprezentarea arborelui este potrivită pentru acest tip de regulă, deoarece datele sunt împărțite recursiv, de obicei într-o abordare de sus în jos, începând cu întregul set de date la nodul rădăcină. Există multe variante de arbori de decizie, în funcție de modul în care este efectuată împărțirea, tipurile de hiperplanuri utilizate, criteriile de terminare etc. DT-urile paralele ale axelor au ales la fiecare nivel de nod cea mai bună caracteristică pentru a defini hiperplanul și a împărți datele pe baza unui criteriu. indicând puterea despărțirii. Importanța atributelor sunt apoi atribuite în funcție de contribuția lor la arbore. O modalitate standard de a calcula importanța atributelor în arbori este luarea în considerare a reducerii totale a criteriului atunci când se utilizează acea caracteristică (normalizată)[17].

fsGA-DT evoluează o populație de indivizi care codifică seturi de caracteristici și utilizează informațiile furnizate de arbori de decizie pentru a ghida căutarea. Comunicarea dintre populația GA și DT este bidirecțională: informațiile circulă de la GA la DT sub formă de seturi de caracteristici și de la DT la GA sub forma evaluării performanței și a importanței caracteristicilor care trebuie utilizate în timpul procesului de evoluție. Figura 3 ilustrează etapele de comunicare în cadrul fsGA-DT, iar algoritmul 2 prezintă schița acestuia. Detaliile fiecărei etape sunt explicate în cele ce urmează.

Codificarea fsGA-DT folosește codificare binară. Un individ x este reprezentat ca un șir de biți de dimensiunea d, unde valoarea 1 indică faptul că atributul corespunzător este ales și 0 că nu este. Pentru individul x notăm cu $FS(x) = \{j_1, \ldots, j_k\}$, $FS(x) \subset \{1, 2, \ldots, d\}$ setul de atribute care sunt selectate pe baza acestuia.

Evaluare Un individ x este evaluat prin inducerea unui arbore de decizie pe setul $FS(x)$ de atribute alese de acesta. Setul de date utilizat pentru selectarea atributelor este împărțit aleatoriu într-un set de antrenament care conține 70% din instanțe și un set de validare de 30% din instanțe, iar DT-ul este evaluat folosind setul de validare. AUC (aria de sub curba [9]) este utilizat ca indicator de performanță. AUC ia valori între 0 și 1 și măsoară robustețea clasificării în raport cu pragurile de probabilitate. Poate fi folosit pentru a compara rezultatele, cu cât valoarea este mai mare cu atât mai bine.

Algorithm 2 fsGA-DT

Input: X, y;
Parameters: population size p_{size}, maximum number of generations, crossover probability, mutation rate μ, mutation probability, maximum dept of DT, the flavor of DT used for evaluation.
Output: best individual, Ψ_{best}, over all generations;
Initialize population P (using the feature importance of DTs);
Set best individual Ψ_{best} at random from P;
Evaluate Ψ_{best} using 10 DTs and average AUC values.
for a number of generations **do**
    Evaluate P by computing $FS(x)$, $fs(x)$ for each $x \in P$;
    Identify x_{best} in P. Evaluate x_{xbest} using 10 DTs and average AUC values.
    Replace Ψ_{best} with x_{best} if better;
    Select p_{size} individuals from P using tournament selection (Section 3.2.2, paragraph Selecția);
    Apply feature recombination on selected individuals (Section 3.2.2, paragraph Încrucișare pe baza atributelor);
    bitflip mutation on each offspring created through recombination
    $P \leftarrow$ offsprings;
end for
Return: Ψ_{best}

Importanța atributelor în timpul evaluării DT sunt metode încorporate pentru selecția atributelor. În timpul inducției arborelui, se ia o decizie la fiecare nivel de nod cu privire la atributul utilizat pentru a împărți datele nodului. Pe baza structurii arborelui, fiecărui atribut din setul de date i se atribuie o măsură de importanță. Aceste valori sunt normalizate, aparțin $[0, 1]$ și sunt transmise indivizilor din populație în timpul fazei de evaluare. Astfel, fiecărui individ x i se atribuie un vector care conține importanța atributului $fs(x) \in \mathbb{R}^d$. $fs(x)$ va avea valori pozitive care se adaugă la 1 pentru atributele nodurilor atribuite în DT de evaluare și 0 pentru celelalte poziții.

Inițializarea Pentru a începe căutarea cu o populație de soluții bune, inițializarea inversează procesul de evaluare: sunt crescuți un număr de arbori de decizie egal cu dimensiunea populației, iar indivizii din populația inițială sunt creați din fiecare arbore prin setând la 1 toate atributele care au o importanță pozitivă în arbore. Arborii utilizați în această etapă sunt introduși/crescuți până când toate frunzele sunt pure, pentru a obține o imagine de ansamblu asupra atributelor potențial utile încă de la începutul căutării.

Cei mai buni indivizi În fiecare iterație, cel mai bun individ din populație este reevaluat prin media rezultatelor raportate de 10 arbori de decizie pentru a reduce variabilitatea valorilor AUC. fsGA-DT păstrează cel mai bun individ peste toate iterațiile, evaluate în acest mod.

Selecția Selecția turnir este utilizată pe baza valorii de fitness. Cu toate acestea, dacă doi indivizi au aceeași valoare fitness (AUC), este selectat cel care are valoarea medie a importanței caracteristicii pozitive mai mare. O motivație din spatele acestei scheme este de a conduce căutarea către un număr mai mic de caracteristici, deoarece pentru fiecare individ, suma $fs(x)$ se adaugă la 1, iar o medie mai mare indică un număr mai mic de valori. Un altul este de a

spori diversitatea, deoarece pentru unele probleme, multe setări distincte pot produce aceleași valori AUC, dar cu valori diferite de importanță a caracteristicilor. Scopul este de a păstra caracteristicile cu importanță mai mare și, de asemenea, de a utiliza valori de importanță fs în timpul variației.

Încrucișare pe baza atributelor Operatorul de încrucișare creează doi descendenți o_1 și o_2 din doi părinți x_1 și x_2 folosind următoarea abordare: pentru fiecare caracteristică (bit), valoarea de la x_1 sau x_2 cu cea mai mare importanță caracteristică este copiată în o_1 și este copiată și în o_2 dar cu o probabilitate egală cu probabilitatea de încrucișare p_{cross}. Alte caracteristici din o_2 sunt setate aleatoriu, cu o probabilitate de 0,5.

Mutația fsGA-DT folosește mutația bitflip și este aplicată cu rata de mutație pe o_1 și o_2 prin inversarea fiecărui bit cu o probabilitate de mutație.

Selecția pentru supraviețuire Pentru a spori diversitatea, descendenții înlocuiesc toți părinții, în timp ce cel mai bun individ din întreaga căutare, notat cu Ψ_{best}, este păstrat separat.

Rezultatul și evaluarea performanței Algoritmul produce cel mai bun $\Psi_{celmaibun}$ individ. Atributele indicate de acesta sunt folosite pentru a antrena un DT și apoi un set de testare este utilizat pentru a evalua performanța DT pe atributele furnizate de fsGA-DT.

3.2.3 Experimente numerice

Sunt efectuate experimente numerice pentru a testa performanța fsGA-DT. Efectuăm experimente pe seturi de date generate sintetic cu diferite grade de dificultate și seturi de date din lumea reală. Toate seturile de date reprezintă o problemă de clasificare binară. Comparăm rezultatele obținute de fsGA-DT cu un clasificator Decision Tree care are incorporat o metodă de selecție a atributelor încorporată.

Date de test Dificultatea seturilor de date sintetice este controlată prin varierea valorilor parametrilor utilizați pentru a le genera: numărul de instanțe din setul de date, numărul de atribute, suprapunerea între instanțele de date din diferite clase, numărul de atribute aleatorii adăugate la setul de date generat pentru a crește dificultatea. Pentru a asigura reproductibilitatea, folosim funcția `make_classification` din biblioteca Python `scikit-learn`[2] [18] pentru a generate seturile de date sintetice. Sunt utilizate toate combinațiile următorilor parametri pentru funcția `make_classification`: numărul de instanțe (250, 500, 1000), numărul de caracteristici (50, 100, 150), numărul de clase (2), separatorul de clasă (0.1, 1), ponderea fiecărei clase (0.5) și parametrul *seed* (500). Fiecare combinație generează un set de date diferit.

Seturile de date din lumea reală pe care efectuăm experimente sunt cele descrise în secțiunea 3.1.5.

Evaluarea performanței Pentru a evalua performanța fiecărui clasificator pe fiecare problemă de test, datele au fost împărțite în instanțe de antrenare și instanțe de testare. Datele de antrenare constă în 70% din instanțele de date ale problemei de testare (alese aleatoriu), iar restul de 30% de date compun datele de test. Antrenăm modelele pe datele de antrenare și raportăm AUC (aria

[2] versiunea `1.1.1`

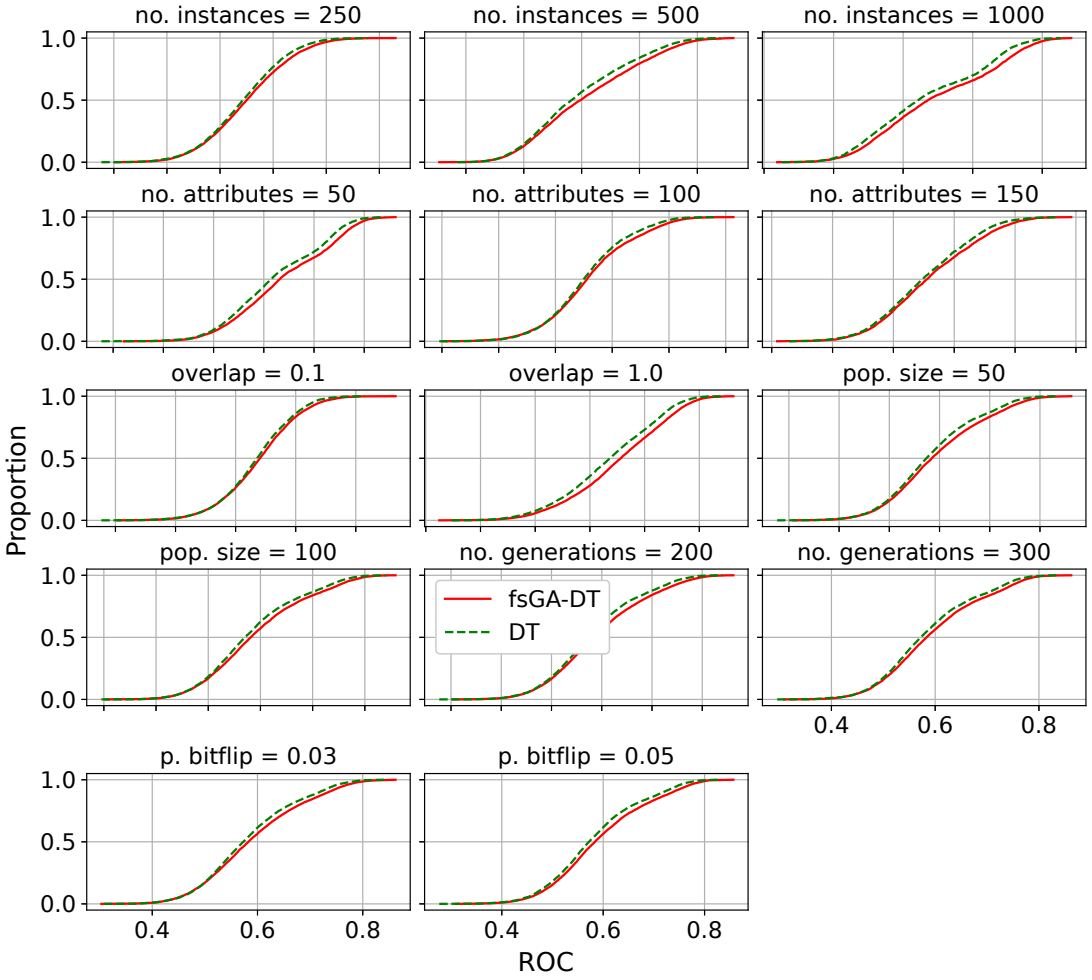

Figura 4: Grafice ECDF pentru rezultatele obținute de fsGA-DT și DT pe seturile de date sintetice pentru diferiți parametri. Rezultatele sunt raportate pentru 100 de rulări independente pe datele de test.

sub curba ROC) [9, 20] pentru datele de testare. AUC poate fi interpretată ca probabilitatea de a clasifica eșantioane pozitive selectate mai mare decât eșantioanele negative selectate.

Sunt efectuate 100 de rulări independente pentru fiecare problemă testată, datele sunt împărțite aleator în date de antrenare și date de test. Seturile de date sunt apoi grupate pe baza unui parametru al setului de date (număr de instanțe, număr de atribute, suprapunere) și parametri ai algoritmului genetic (dimensiunea populației, numărul de generații, probabilitatea de mutație pentru fiecare bit) și sunt generate figuri ECDF (*empirical cumulative distribution function*) pentru a ilustra distribuția valorilor AUC raportate de cele două metode. Graficele arată proporția elementelor care sunt mai mici sau egale cu un anumit punct din grafic; atunci când se compară mai multe curbe pe același figură, cea din dreapta indică o probabilitate mai mică de a lua valori mai mici și poate fi considerată mai bună.

Parametrii folosiți Parametrii utilizați pentru fsGA-DT sunt dimensiunea populației (50, 100), numărul de generații (200, 300), probabilitatea de încrucișare (0.7), probabilitatea de mutație (0.2), probabilitatea de mutare a unui bit (0.03, 0, 05) și dimensiunea turnirului (3, 5)). Pentru Arborele de Decizie, criteriul de împărțire este indexul *gini*, iar adâncimea maximă a arborelui este de 5.

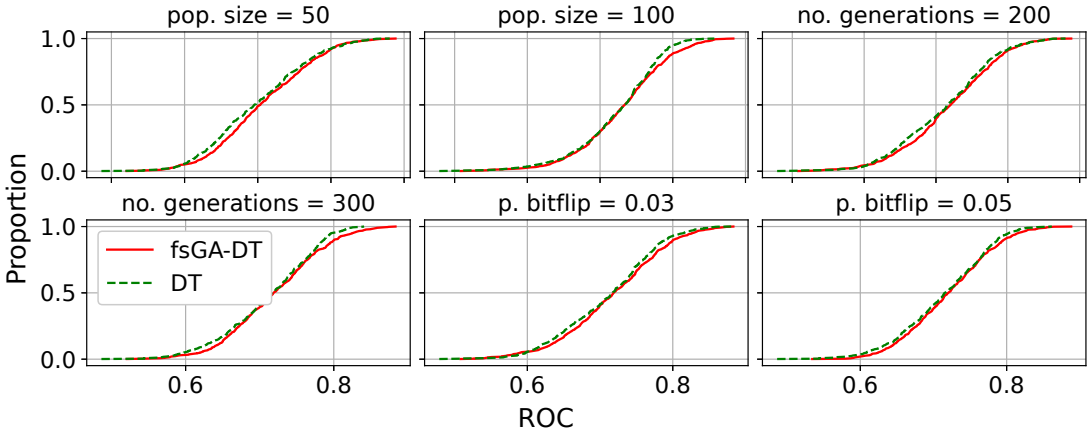

Figura 5: Grafice ECDF pentru rezultatele raportate pentru 100 de rulări independente de fsGA-DT și DT pe setul de date din lumea reală R_1.

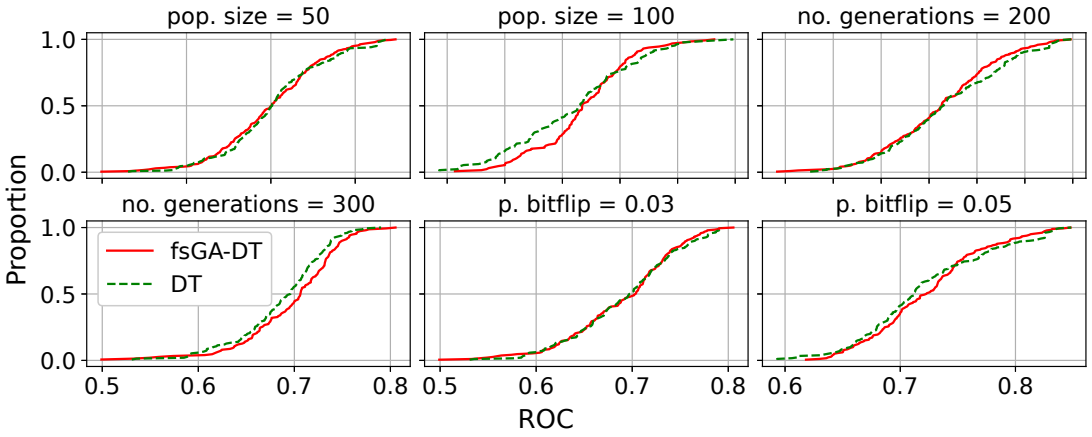

Figura 6: Grafice ECDF pentru rezultatele raportate pentru 100 de rulări independente de fsGA-DT și DT pe setul de date din lumea reală R_2.

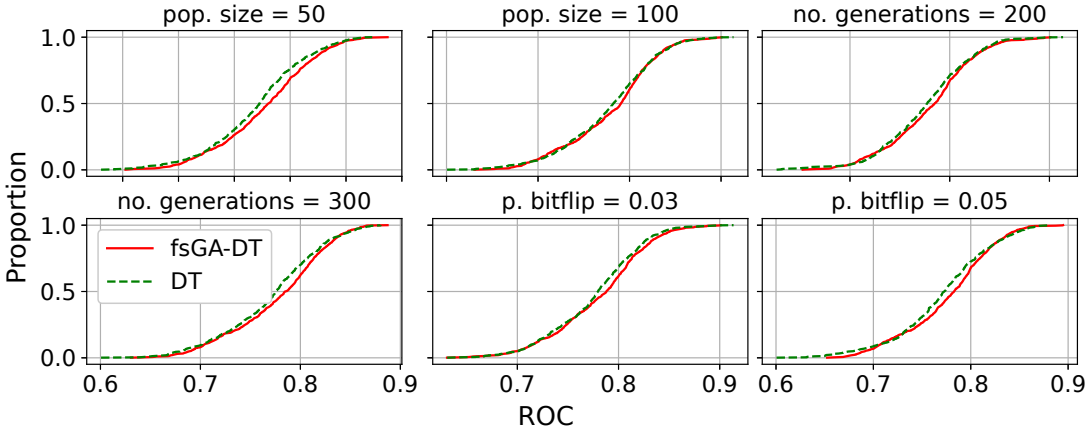

Figura 7: Grafice ECDF pentru rezultatele raportate pentru 100 de rulări independente de fsGA-DT și DT pe setul de date din lumea reală R_3.

Rezultate experimentale Figura 4 prezintă diagrame ECDF pe seturile de date sintetice pentru diferiți parametri și valorile acestora, figurile 5, 6 și 7 prezent diagrame ECDF pentru seturile de date din lumea reală. Tabelele 3 și 4 prezintă rezultate semnificative pentru un test care compară valorile AUC raportate de ambele metode pe cele 100 de rulări independente. Ipoteza nulă testată este că AUC medie raportată de fsGA-DT este mai mică decât cea raportată de DT; respingerea acesteia indică faptul că putem afirma că diferențele de rezultate sunt semnificative, iar rezultatul fsGA-DT poate fi considerat mai bun decât cel obținut de DT. Constatăm că pentru majoritatea combinațiilor de parametri diferențele între rezultate sunt semnificative. Cele mai multe rezultate pozitive sunt obținute pentru $d_3 = 1000$ instanțe și $p_2 = 1$, parametri care se suprapun. Rezultatele raportate pentru seturile de date din lumea reală sunt similare.

3.2.4 Concluzii

Este explorată o abordare hibridă a clasificării atributelor. Un algoritm genetic este conceput pentru a dezvolta seturi de atribute evaluate prin utilizarea arborilor de decizie. Deoarece arborii de decizie realizează intrinsec selecția atributelor și, prin urmare, reprezintă și modele de selecție a atributelor încorporate, ei oferă, de asemenea, o măsură a importanței atributelor care poate fi utilizată pentru a ghida căutarea algoritmului genetic. Astfel, comunicarea dintre populația GA și componenta DT este bidirecțională: atributele sunt trimise către arborii de decizie pentru a fi evaluate, iar DT, pe lângă faptul că oferă o măsură de fitness, indică importanța atributului care este utilizată în selecție și recombinare. Pentru a evita *overfitting*, cel mai bun individ este evaluat prin utilizarea mai multor copaci crescuți pe date împărțite aleator în date de antrenare și de validare. Deși există multe abordări bazate pe algoritmi genetici pentru selecția atributelor, rezultatele arată că există încă loc pentru a explora diferite mecanisme de comunicare și noi operatori genetici care utilizează informații specifice de la clasificatori în timpul evoluției. Problema de selecție a atributelor este una complexă și fiecare pas mic către îmbunătățirea metodelor existente poate adăuga valoare și poate contribui la înțelegerea problemei.

Referințe

[1] Bagherzadeh, F., Mehrani, M.-J., Basirifard, M., and Roostaei, J. Comparative study on total nitrogen prediction in wastewater treatment plant and effect of various feature selection methods on machine learning algorithms performance. *Journal of Water Process Engineering 41* (2021), 102033.

[2] Bala, J., Huang, J., Vafaie, H., Dejong, K., and Wechsler, H. Hybrid learning using genetic algorithms and decision trees for pattern classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1* (San Francisco, CA, USA, 1995), IJCAI'95, Morgan Kaufmann Publishers Inc., p. 719–724.

[3] Bommert, A., Sun, X., Bischl, B., Rahnenführer, J., and Lang, M. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis 143* (2020), 106839.

[4] Breiman, L. Random Forests. *Machine Learning 45*, 1 (Oct. 2001), 5–32.

[5] Brown, G. W. Iterative solution of games by fictitious play. *Activity analysis of production and allocation 13*, 1 (1951), 374–376.

[6] Cai, J., Luo, J., Wang, S., and Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing 300* (2018), 70–79.

Tabela 3: Comparații statistice între rezultatele obținute de fsGA-DT și DT pe seturile de date sintetice pentru diferite valori ale parametrilor algoritmului genetic. Parametrii setului de date sintetic sunt: numărul de instanțe din setul de date (d_1 - 250, d_2 - 500 și d_3 - 1000 instanțe de date), numărul de atribute ($p_1 : 50, 100.150$), suprapunerea claselor ($p_2 : 0.1, 1$). Parametrii pentru partea de algoritm genetic din fsGA-DT sunt: dimensiunea populației ($p_3 : 50, 100$), numărul de generații ($p_2 : 200, 300$) și probabilitatea de mutație a inversării biților ($p_5 : 0,03, 0,05$). Un simbol ($\triangle$) indică faptul că fsGA-DT raportează rezultate mai bune din punct de vedere statistic, un simbol ($\times$) indică faptul că abordarea DT obține rezultate mai bune din punct de vedere statistic, iar un simbol ($-$) indică nicio diferență semnificativă între rezultate.

p_1	p_2	p_3	p_4	p_5	d_1	d_2	d_3	p_1	p_2	p_3	p_4	p_5	d_1	d_2	d_3
50	0.1	50	200	0.03	$\times$	-	$\triangle$	100	1.0	50	200	0.03	-	$\triangle$	$\triangle$
50	0.1	50	200	0.05	$\triangle$	$\triangle$	$\triangle$	100	1.0	50	200	0.05	$\triangle$	$\triangle$	$\triangle$
50	0.1	50	300	0.03	-	$\triangle$	$\triangle$	100	1.0	50	300	0.03	$\triangle$	$\triangle$	$\triangle$
50	0.1	50	300	0.05	$\triangle$	-	$\triangle$	100	1.0	50	300	0.05	$\triangle$	$\triangle$	$\triangle$
50	0.1	100	200	0.03	-	$\triangle$	$\triangle$	100	1.0	100	200	0.03	$\triangle$	$\triangle$	$\triangle$
50	0.1	100	200	0.05	$\triangle$	-	$\triangle$	100	1.0	100	200	0.05	$\triangle$	$\triangle$	$\triangle$
50	0.1	100	300	0.03	-	$\triangle$	$\triangle$	100	1.0	100	300	0.03	-	$\triangle$	$\triangle$
50	0.1	100	300	0.05	$\triangle$	$\triangle$	$\triangle$	100	1.0	100	300	0.05	$\triangle$	$\triangle$	-
50	1.0	50	200	0.03	$\triangle$	$\triangle$	$\triangle$	150	0.1	50	200	0.03	-	-	$\triangle$
50	1.0	50	200	0.05	$\triangle$	$\triangle$	$\triangle$	150	0.1	50	200	0.05	$\triangle$	$\triangle$	$\triangle$
50	1.0	50	300	0.03	$\triangle$	$\triangle$	$\triangle$	150	0.1	50	300	0.03	-	$\times$	$\times$
50	1.0	50	300	0.05	$\triangle$	$\triangle$	$\triangle$	150	0.1	50	300	0.05	$\triangle$	-	-
50	1.0	100	200	0.03	$\triangle$	$\triangle$	$\triangle$	150	0.1	100	200	0.03	-	-	$\triangle$
50	1.0	100	200	0.05	$\triangle$	$\triangle$	$\triangle$	150	0.1	100	200	0.05	-	-	$\triangle$
50	1.0	100	300	0.03	$\triangle$	$\triangle$	$\triangle$	150	0.1	100	300	0.03	-	-	$\triangle$
50	1.0	100	300	0.05	$\triangle$	$\triangle$	$\triangle$	150	0.1	100	300	0.05	-	$\times$	-
100	0.1	50	200	0.03	$\times$	-	-	150	1.0	50	200	0.03	$\triangle$	$\triangle$	$\triangle$
100	0.1	50	200	0.05	-	-	-	150	1.0	50	200	0.05	$\triangle$	$\triangle$	-
100	0.1	50	300	0.03	-	-	$\triangle$	150	1.0	50	300	0.03	-	$\triangle$	-
100	0.1	50	300	0.05	$\times$	-	$\triangle$	150	1.0	50	300	0.05	$\triangle$	-	-
100	0.1	100	200	0.03	-	$\triangle$	$\times$	150	1.0	100	200	0.03	-	$\triangle$	$\triangle$
100	0.1	100	200	0.05	-	-	-	150	1.0	100	200	0.05	$\triangle$	$\triangle$	$\triangle$
100	0.1	100	300	0.03	$\times$	-	$\triangle$	150	1.0	100	300	0.03	$\times$	$\triangle$	-
100	0.1	100	300	0.05	-	$\triangle$	-	150	1.0	100	300	0.05	-	-	$\triangle$

Tabela 4: La fel ca mai sus: comparații statistice între rezultatele obținute de fsGA-DT și DT pe seturile de date din lumea reală (R_1, R_2, și R_3) pentru diferite valori ale parametrilor algoritmului genetic: dimensiunea populației ($p_1 : 50, 100$), numărul de generații ($p_2 : 200, 300$) și probabilitatea de mutație a inversării biților ($p_3 : 0,03, 0,05$).

p_1	p_2	p_3	R_1	R_2	R_3
50	200	0.03	-	$\times$	-
50	200	0.05	$\triangle$	$\times$	$\triangle$
50	300	0.03	$\triangle$	-	$\triangle$
50	300	0.05	-	$\triangle$	$\triangle$
100	200	0.03	$\triangle$	-	-
100	200	0.05	-	-	-
100	300	0.03	$\triangle$	$\triangle$	-
100	300	0.05	$\triangle$	-	$\triangle$

[7] Chandrashekar, G., and Sahin, F. A survey on feature selection methods. *Computers & Electrical Engineering 40*, 1 (2014), 16–28.

[8] Dua, D., and Graff, C. UCI machine learning repository, 2017.

[9] Fawcett, T. An introduction to roc analysis. *Pattern Recognition Letters 27*, 8 (2006), 861–874. ROC Analysis in Pattern Recognition.

[10] Hansen, L., Lee, E. A., Hestir, K., Williams, L. T., and Farrelly, D. Controlling feature selection in random forests of decision trees using a genetic algorithm: Classification of class i mhc peptides. *Combinatorial Chemistry & High Throughput Screening 12*, 5 (2009), 514–519.

[11] Hoque, N., Singh, M., and Bhattacharyya, D. K. EFS-MI: an ensemble feature selection method for classification. *Complex & Intelligent Systems 4*, 2 (June 2018), 105–118.

[12] Huljanah, M., Rustam, Z., Utama, S., and Siswantining, T. Feature Selection using Random Forest Classifier for Predicting Prostate Cancer. *IOP Conference Series: Materials Science and Engineering 546*, 5 (June 2019), 052031. Publisher: IOP Publishing.

[13] Jain, S., and Saha, A. Rank-based univariate feature selection methods on machine learning classifiers for code smell detection. *Evolutionary Intelligence 15*, 1 (Mar. 2022), 609–638.

[14] Kohavi, R., and John, G. H. Wrappers for feature subset selection. *Artificial Intelligence 97*, 1 (1997), 273–324.

[15] Lazar, C., Taminau, J., Meganck, S., Steenhoff, D., Coletta, A., Molter, C., de Schaetzen, V., Duque, R., Bersini, H., and Nowe, A. A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 9*, 4 (2012), 1106–1119.

[16] Nogueira, S., and Brown, G. Measuring the stability of feature selection. In *Machine Learning and Knowledge Discovery in Databases* (Cham, 2016), P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, Eds., Springer International Publishing, pp. 442–457.

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn - machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[18] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[19] Rahmadani, S., Dongoran, A., Zarlis, M., and Zakarias. Comparison of naive bayes and decision tree on feature selection using genetic algorithm for classification problem. *Journal of Physics: Conference Series 978*, 1 (mar 2018), 012087.

[20] Rosset, S. Model selection via the auc. In *Proceedings of the Twenty-First International Conference on Machine Learning* (New York, NY, USA, 2004), ICML '04, Association for Computing Machinery, p. 89.

[21] Saraswat, M., and Arya, K. V. Feature selection and classification of leukocytes using random forest. *Medical & Biological Engineering & Computing 52*, 12 (Dec. 2014), 1041–1052.

[22] Sheen, S., and Rajesh, R. Network intrusion detection using feature selection and decision tree classifier. In *TENCON 2008 - 2008 IEEE Region 10 Conference* (2008), pp. 1–4.

[23] Stein, G., Chen, B., Wu, A. S., and Hua, K. A. Decision tree classifier for network intrusion detection with ga-based feature selection. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2* (New York, NY, USA, 2005), ACM-SE 43, Association for Computing Machinery, p. 136–141.

[24] Theodoridis, P. K., and Gkikas, D. C. Optimal feature selection for decision trees induction using a genetic algorithm wrapper - a model approach. In *Strategic Innovative Marketing and Tourism* (Cham, 2020), A. Kavoura, E. Kefallonitis, and P. Theodoridis, Eds., Springer International Publishing, pp. 583–591.

[25] Tsai, C.-F., and Hsiao, Y.-C. Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems 50*, 1 (2010), 258–269.

[26] Utgoff, P. E. Incremental Induction of Decision Trees. *Machine Learning 4*, 2 (Nov. 1989), 161–186.

[27] Vafaie, H., and De Jong, K. Genetic algorithms as a tool for feature selection in machine learning. In *Proceedings Fourth International Conference on Tools with Artificial Intelligence TAI '92* (1992), pp. 200–203.

[28] Wang, S., Tang, J., and Liu, H. Embedded Unsupervised Feature Selection. *Proceedings of the AAAI Conference on Artificial Intelligence 29*, 1 (Feb. 2015).

[29] XUE, B., CERVANTE, L., SHANG, L., BROWNE, W. N., and ZHANG, M. Multi-objective evolutionary algorithms for filter based feature selection in classification. *International Journal on Artificial Intelligence Tools 22*, 04 (2013), 1350024.

[30] Xue, B., Zhang, M., Browne, W. N., and Yao, X. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation 20*, 4 (2016), 606–626.

[31] İrsoy, O., Yildiz, O. T., and Alpaydin, E. Soft decision trees. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)* (2012), pp. 1819–1822.

Director Proiect,
Suciu Mihai-Alexandru