CONTRIBUŢII LA ÎMBUNĂTĂŢIREA TEHNOLOGIEI DE CLASIFICARE A DATELOR

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI
de către

Ing. Claudiu-Raul ROBU

Conducător ştiinţific:	prof.univ.dr.ing Vasile STOICU-TIVADAR
Referenţi ştiinţifici:	prof.univ.dr.ing.mat. Dumitru Dan BURDESCU
	prof.univ.dr.ing. Viorel NEGRU
	conf.univ.dr.ing. Ştefan HOLBAN

Ziua susţinerii tezei: 27.07.2012

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica” din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Automatică şi Informatică Aplicată al Universităţii „Politehnica” din Timişoara.

Mulţumiri deosebite se cuvin conducătorului de doctorat prof.dr.ing. Vasile STOICU-TIVADAR, care m-a sprijinit, m-a încurajat, mi-a dat forţa şi inspiraţia ca să duc lucrurile până la capăt. Îi mulţumesc domnului prof. dr.ing Ştefan HOLBAN, pentru că mi-a fost mereu aproape, cu dânsul m-am consultat începând cu septembrie 2009 şi dânsul m-a direcţionat către algoritmii bioinspiraţi. Îi mulţumesc domnului profesor şi prieten Ioan FILIP, pentru sprijinul acordat în activitatea publicistică, cu dânsul am scris primele mele lucrări.

Mulţumesc familiei mele care m-a susţinut în această perioadă şi în mod special soţiei mele Mădălina care pe lângă faptul că mi-a fost alături, mi-a şi citit teza şi tatălui meu care mereu m-a încurajat să am încredere în mine şi să am forţa să finalizez teza.

Le mulţumesc tuturor celor care m-au ajutat în orice fel ca să-mi finalizez teza şi consider că acesta este un nou început de drum.

Timişoara, iulie 2012 Claudiu-Raul ROBU

Robu, Claudiu-Raul

Contribuţii la îmbunătăţirea tehnologiei de clasificare a datelor

Teze de doctorat ale UPT, Seria 14, Nr. 6, Editura Politehnica, 2012, 182 pagini, 51 figuri, 67 tabele.

ISSN: 2069-8216

ISSN-L:2069-8216

ISBN:978-606-554-504-5

Cuvinte cheie:
clasificare, data mining, Ant Colony Optimization, algoritmi genetici, predicţii, Weka

Rezumat:
Capitolul 1 prezintă definiţii şi terminologie, etapele procesului de data mining, domeniile de aplicare, instrumente utilizate, noţiunea de clasificare a datelor, algoritmi de clasificare, seturi de date utilizate la analiza algoritmilor, etc. Capitolul 2 trateaza preporcesarea datelor şi prezintă o sinteză a tehnicilor de preprocesare, aspecte legate de instrumentul conceput şi implmentat Arff Convertor şi realizează o analiză statistică a datelor unor studenţi. Capitolul 3 prezintă algoritmul ACO, se analizează comportametul lui rulând foarte multe configuraţii de parametrii, iar in urma analizei se emit recomandări privind alegerea parametrilor. Se prezintă diverse variante ale lui ACO şi se propun 4 imbunătăţiri care se implmentează în Ant-r-Miner. În capitolul 4 se prezintă aspecte legate de algorimii genetici utilizati la clasificarea datelor şi se propune un nou algoritm genetic care se implementaeza in Weka. Se analizează comportamentul său la diferite valori ale parametrilor de intrare şi se emite o listă de recomnadări privind alegerea acestora. În capitolul 5 este concepută o nouă metodă de a realiza predicţii în Weka, cu ajutorul unei interfete dinamice, funcţie de setul de date analizat. În capitolul 6 sunt analizate date legate de nasterile care au avut loc la Clinica de Obstretica si Ginecologie Bega, Timisoara în anul 2010. Se construiesc cu algoritmii propusi clasificatori în funcţie de indicele apgar.

CUPRINS

1. Introducere

1.1. Noţiuni generale

1.1.1. Definiţii şi terminologie

Oamenii au căutat tipare încă de la începutul omenirii: vânătorii au căutat tipare în comportamentul animalelor la migrare, fermierii au căutat tipare în ceea ce priveşte creşterea recoltelor, medicii caută tipare în simptomele pacienţilor. Procesul descoperirii de cunoştinţe este la fel de vechi ca şi omenirea, începând cu descoperirea focului şi ajungând la studiile actuale privind universul. Data mining este procesul de descoperire de cunoştinţe şi extragere de reguli şi tipare din date.

Nevoia explorării şi extragerii cunoştinţelor din baze de date de dimensiuni foarte mari este tot mai accentuată întrucât numărul bazelor de date de dimensiuni impresionante este în continuă creştere. Bazele de date din spitale, bănci, supermarketuri, au căpătat în timp dimensiuni impresionante. Senzorii de pe sateliţi, telescoapele care scanează cerul produc seturi de date de ordinul GB / oră [TAN05]. În multe baze de date au fost stocate informaţii care ascund cunoştinţe extrem de valoroase. Prin analizarea datelor despre clienţii unui supermarket se pot determina clienţii care sunt aproape pierduţi şi se pot iniţia campanii publicitare în vederea păstrării lor. Analizarea datelor despre clienţii unei bănci permite construirea profilul clientului de încredere, respectiv a clientului de neîncredere, foarte util în acordarea unui credit. Datele pacienţilor permit construirea profilului pacientului care suferă de o anumită afecţiune, iar acest profil poate ajuta la diagnostic sau prevenire [ATI09]. Prin intermediul *data mining* o universitate poate prezice cu o acurateţe de 85% care studenţi vor absolvi şi care nu. Universitatea poate folosi această informaţie pentru a-şi concentra atenţia asupra acelor studenţi care sunt în pericol de a nu absolvi. Administraţia Federală a Aviaţiei Statelor Unite foloseşte data mining pentru a analiza datele legate de prăbuşirea avioanelor, cu scopul de găsi defectele comune şi a recomanda măsuri de precauţie [SEI08].

Începând cu anii 90 [GOR06] se vorbeşte de data mining în foarte multe medii, plecând de la cele academice şi până la cele de afaceri sau medicale.

Revista tehnologică online ZDNET News afirma în 8 Februarie 2001 că Data mining va deveni "una dintre cele mai revoluţionare dezvoltări a deceniului următor". Recenzia MIT Technology a ales data mining ca fiind una dintre primele 10 tehnologii apărute care vor schimba lumea.

De-a lungul timpului au fost date multe definiţii procesului de data mining:

Data mining este o tehnologie care permite extragerea informaţiilor, utile, relevante din BD de dimensiuni mari [TWO99].

Prin data mining se înţelege procesarea datelor utilizând capabilităţi de căutare sofisticată şi algoritmi statistici pentru a găsi şabloane şi corelaţii în baze de date existente şi de dimensiuni mari [www1].

Data mining este procesul de extragere a modelelor din date. Explorarea datelor se realizează cu scopul de a transforma aceste date în informaţii [PON07].

Data mining este o tehnica de a căuta modele / şabloane în baze de date de dimensiuni mari. Scopul căutării este acela de a găsi corelaţii între variabile (anterior necunoscute) care pot fi valoroase din punct de vedere comercial [BER04].

Data mining înseamnă extragerea de şabloane şi alte informaţii folositoare dintr-un corp de date [www2].

Data mining este procesul care foloseşte tehnici statistice pentru a descoperi relaţiile subtile între date, precum şi construirea de modele predictive bazate pe ele [GOR06].

Data mining - un amestec de statistică, inteligenţă artificială şi cercetare în baze de date (Prigibon – cercetător de la Google Inc.) [GOR06].

Prin data mining – se înţelege practica căutării automate de pattern-uri (modele, şabloane, tipare) în mulţimi mari de date, utilizând tehnici computaţionale din statistică, învăţarea automată şi recunoaşterea formelor [GOR06].

O definiţie proprie, mai filozofică este următoarea: Data mining înseamnă învăţarea din experienţa trecutului reprezentată prin seturi de date, cu scopul aplicării cunoaşterii dobândite în acţiunile viitoare.

Disciplinele care stau la baza data mining sunt cele din figura 1.1.

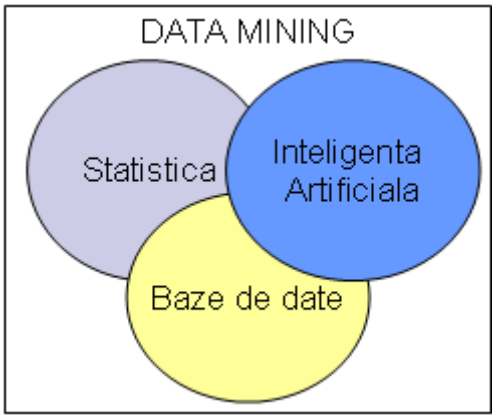

Fig. 1.1 - Disciplinele care stau la baza Data Mining

Statistica aduce în *data mining* tehnici ce permit identificarea de relaţii între variabile (regresie liniara, regresie logistica, Naive Bayes, etc).

Inteligenţa artificială contribuie la Data Mining prin tehnici de procesare a informaţiei bazate pe modelul raţionamentului uman.

Machine Learning este un domeniul al inteligenţei artificiale care este cel mai folosit în Data Mining.

Machine Learning se ocupă cu crearea de algoritmi şi tehnici care permit calculatoarelor să „înveţe”.

Finalitatea Machine Learning este achiziţia de descrieri structurale din exemple. Aceste descrieri structurale trebuie să poată fi folosite pentru predicţii şi înţelegerea fenomenului studiat.
Sistemele de baze de date reprezintă materialul care trebuie explorat.

Tehnicile de data mining cele mai utilizate sunt clasificare, regresie, clustering (grupare), reguli de asociere.

Clasificarea este o metodă predictivă care face parte din categoria metodelor de învăţare supervizată. Clasele sunt valorile pe care le poate lua un atribut categorial al setului de date considerat, numit atribut clasă. Atributele diferite de atributul clasă se numesc predictori. Scopul clasificării este de a prezice pentru o instanţă dată caracterizată de un set de valori a predictorilor a clasei căreia îi aparţine.

Setul de date este format din instanţe (exemple, obiecte).

Atributele sunt caracteristici sau proprietăţi ale exemplelor, exprimate prin valori numerice sau nominale (categorice). Altfel spus, atributele sunt denumirile coloanelor dintr-un tabel. Dacă într-un studiu de prognoză a vremii, dorim să prezicem cum va fi mâine (soare, înnorat, ploaie, lapoviţă, grindină, ninsoare), atunci aceste valori le putem considera clase iar atributul clasă îl putem numi vreme.

Regresia este tot o metodă predictivă, similară cu clasificarea, dar valorile care se prezic nu sunt categoriale ci sunt numerice [TAN05]. În cazul regresie exemplul precedent ar presupune prezicerea temperaturii de mâine.

Clusteringul (Gruparea) este procesul de determinare a grupurilor (numite clustere) de instanţe asemănătoare dintr-un set de date. Gruparea este o tehnică de învăţare nesupervizată, descriptivă, în urma utilizării ei se obţin informaţii cu privire la modul în care sunt organizate datele.

Detectarea asocierilor între diferite tipuri de informaţii care, în aparenţă nu au nici un fel de dependenţe semantice, poate oferi concluzii interesante. Regulile de asociere au ca scop modelarea dependenţelor între articole în datele tranzacţionale. Un exemplu clasic de studiu utilizând regulile de asociere este analiza produselor care se cumpără împreună (cineva care îşi cumpără un ciocan este foarte probabil să-şi cumpere şi cuie).

1.1.2. Etapele procesului de data mining

Procesul de data mining presupune parcurgerea următoarelor etape (vezi figura 1.2):

- Preprocesarea datelor
- Construirea şi testarea modelului (modelelor)
- Aplicarea modelului pe date noi

Preprocesarea datelor presupune citirea datelor bute, curăţarea lor şi prelucrarea lor în vederea optimizării procesului de *data mining*.

Datele brute pot conţine valori lipsă, date introduse greşit, date expirate, date duplicat, valori aberante sau pot fi afectate de zgomot [GOR06]. În acest sens se impune o primă preprocesare a datelor brute, prin estimarea valorilor lipsă, prin eliminarea datelor duplicat sau expirate, prin eliminarea sau corectarea datelor introduse greşit, prin utilizarea unor parametrii statistici toleranţi la valori extreme prin utilizarea de filtre hard şi soft pentru reducerea zgomotului, etc. În urma prelucrării datelor brute se obţin date curăţate.

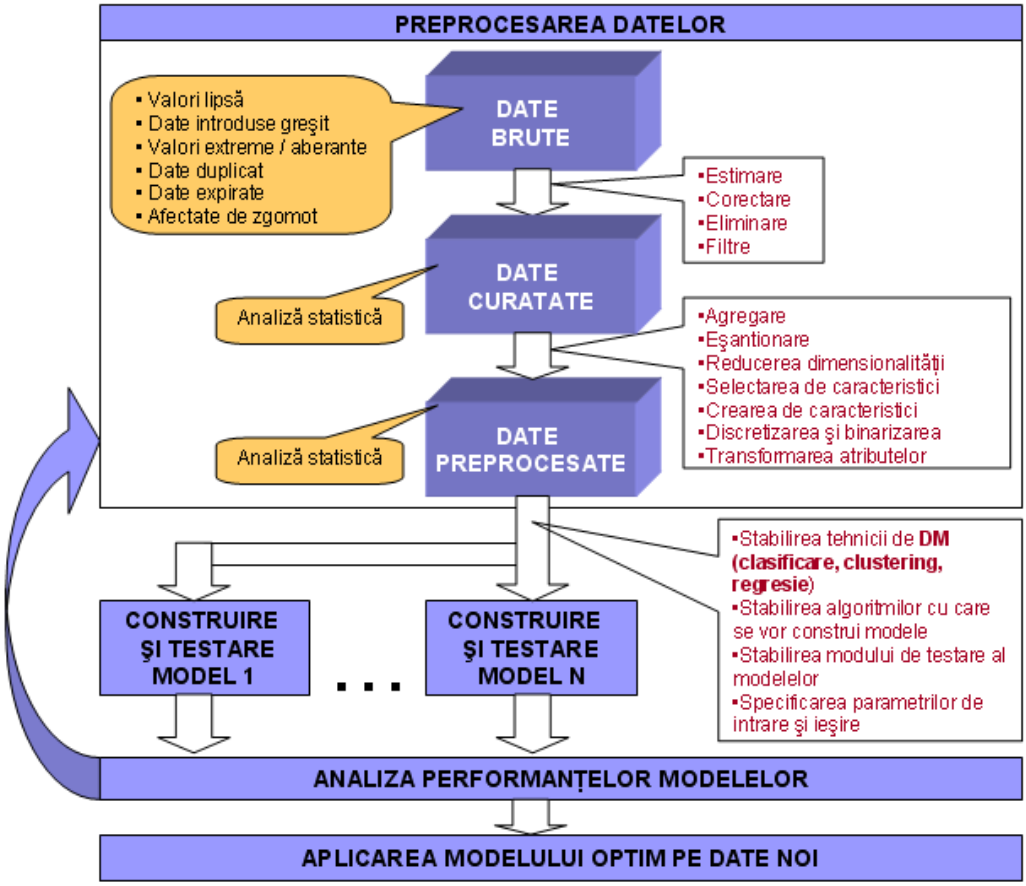

Fig. 1.2 - Etapele procesului de Data Mining

Datele curăţate trebuie preprocesate în continuare, în vederea optimizării procesului de *data mining*. Câteva tehnici care pot fi utilizate în acest scop sunt prezentate în continuare:

- **Agregarea** – combinarea mai multor atribute existente într-un singur atribut, cu scopul reducerii volumului de date şi al obţinerii de date mai stabile (exemplu: agregarea satelor în comuna de care aparţin, a oraşelor în judeţul de care aparţin, a vânzărilor săptămânale în vânzările lunare, etc.)
- **Eşantionarea** – se alege un eşantion reprezentativ pentru întreaga mulţime de date. O problemă importantă este alegerea optimă a volumului eşantionului astfel încât să nu se piardă date dar să nici nu fie mai multe decât este necesar.
- **Reducerea dimensionalităţii –** se ştie că atunci când dimensiunea (numărul atributelor) creşte va creşte în mod implicit şi împrăştierea datelor, ceea ce duce la scăderea vitezei de lucru. Acest fenomen este denumit în *data mining* ca şi blestemul dimensionalităţii (*curse of dimensionality*). Antidotul este reducerea dimensionalităţii prin tehnici ca: analiza factorială şi analiza componentelor principale.
- **Selectarea de submulţimi de caracteristici –** pentru eliminarea caracteristicilor redundante sau nerelevante
- **Crearea de caracteristici -** crearea de noi atribute care pot capta mai bine informaţiile din date decât cele originale
- **Discretizarea şi binarizarea -** trecerea de la date continue la date discrete (de exemplu trecerea de la numere reale la numere întregi)
- **Transformarea atributelor -** transformarea unor atribute vechi în atribute noi

În mod opţional, datele curăţate sau preprocesate pot fi supuse unei etape de analiza statistică a valorilor fiecărui atribut. Această analiză statistică poate extrage din date informaţii utile, privind distribuţia valorilor fiecărui atribut şi ajută la înţelegerea datelor.

Construirea modelului şi testarea lui. De regulă, în studiile de data mining se construiesc mai multe modele şi se alege cel mai performant dintre ele. O situaţie rară, în care nu se justifică construirea mai multor modele, este aceea în care primul model construit este cel mai bun cu putinţă (de exemplu am construit un model clasificator care are o acurateţe a predicţiei de 100 %). De obicei însă, este necesară construirea mai multor modele cu diferiţi algoritmi, compararea performanţelor obţinute de toate modelele şi alegerea celui mai bun model. Dacă toate modelele pe care încercăm să le construim au performanţe nesatisfăcătoare, este necesară revenirea în etapa de preprocesare, prelucrarea datelor fie prin discretizare, fie prin alte tehnici precum selectarea şi crearea de caracteristici şi reconstruirea modelelor, poate că în urma unor prelucrări a datelor de intrare vor obţine performanţe mai bune. Construirea oricărui model se face învăţând din datele disponibile. Pentru a învăţa din date se pot utiliza diverse tehnici (clasificare, regresie, grupare sau reguli de asociere) şi diverşi algoritmi în funcţie de tehnica aleasă. De exemplu pentru clasificare se pot folosi algoritmii Naive Bayes, C4.5, Suport Vector Machines, pentru regresie se pot folosi algoritmi de regresie liniara, regresie liniara multipla, regresie logistica, pentru grupare se pot folosi algoritmii K-Means, EM, FarthestFirst, pentru a descoperi reguli de asociere algoritmul Apriori,

FPGrowth etc. Un model poate fi testat pe datele de antrenare, sau întregul set de date se împarte în date de antrenare şi date de test (în acest caz va fi testat pe date noi şi trebuie ales procentul de instanţe folosit la antrenare), etc. În funcţie de algoritmul folosit pentru construirea modelului trebuie configuraţi parametrii specifici de intrare şi trebuie aleşi parametrii de ieşire.

Dacă cel puţin unul din modele construite obţine rezultate satisfăcătoare, atunci modelul poate fi aplicat pe date noi.

1.2. Domenii în care tehnicile de data mining se aplică cu succes

Data mining este o tehnologie care are o arie foarte largă de acoperire. Data mining se poate aplica în orice domeniu în care există date salvate, ca urmare a unei experienţe şi se doreşte să se înveţe din datele respective sau altfel spus, să se extragă cunoştinţe din acele date.

Domenii în care Data Mining s-a impus mai mult decât în altele sunt e-commerce, medicină, educaţie, detectarea fraudelor (în special în domeniul bancar), combaterea terorismului.

1.2.1. Data mining în e-commerce (Web Mining)

Clienţii sunt componenta ţintă a oricărei organizaţii. Achiziţionarea de noi clienţi, satisfacerea şi păstrarea celor existenţi, precum şi estimarea comportamentului cumpărătorilor poate îmbunătăţi disponibilitatea de produse şi servicii şi, prin urmare, profiturile. Astfel, obiectivul oricărui exerciţiu de *data mining* în e-commerce este de a îmbunătăţi procesele care contribuie la furnizarea de valoare către clientul final [RAG05].

Sistemele software destinate planificării resurselor companiilor (*ERP-enterprise resource planning*) oferă deseori facilităţi de data mining. Unul din cele mai puternice ERP-uri este sistemul SAP [PRO09]. Modelele de data mining din SAP permit determinarea unor şabloane în comportamentul clienţilor şi pot fi utilizate pentru a furniza răspunsuri la întrebări precum: care ofertă este cea mai potrivită, pentru care client şi când ar trebui făcută acea ofertă.

Fişierele log de pe *Web* ascund informaţii cu privire la ce caută potenţialii clienţi pe un site. Îşi propun să cumpere sau doar să navigheze? Încearcă să-şi cumpere ceva cu care sunt familiari sau ceva despre care ştiu puţine lucruri? Încearcă să cumpere de acasă, de la servici sau de la un hotel?

Informaţiile din fişierele log sunt adesea utilizate pentru a determina ce performanţă poate fi aşteptată de la servere şi reţea, pentru a sprijini service-ul şi pentru a face interacţiunea e- business productivă.

Instrumentele de *data mining* din spatele aplicaţiilor *Web*, au grijă de reprezentare cunoştinţelor, de construirea profilului clientului şi de modelarea predictivă a scenariilor interacţiunilor clientului. De exemplu, odată ce un client şi-a cumpărat un anumit număr de servere, este foarte probabil să aibă nevoie şi de anumite echipamente adiţionale cum ar fi routere, switch-uri, echipamente de back-up, etc. Sistemele bazate pe reguli ar putea fi folosite pentru a sugera asemenea alternative clienţilor [RAG05].

Data mining a fost de asemenea aplicată în detectarea modului în care clienţii pot răspunde la oferte promoţionale făcute de o companie e-commerce care se ocupă cu carduri de credit [ZHA03]. Tehnici precum fuzzy au fost utilizate pentru a genera reguli de tipul if-then. [RAG05].

E-commerce-ul este un domeniu foarte potrivit pentru *data mining* pentru că oferă toate ingredientele necesare pentru un *data mining* de succes: există o mulţime de înregistrări, colecţiile electronice oferă date fiabile, reflecţia poate fi uşor transformată în acţiune şi returnată în investiţii care se pot măsura. [ANS01]

Practic fiecare companie mare care vinde cu amănuntul are un site *Web* şi facilităţi de vânzare online poate utiliza Data Mining. Analizând fişierele log ale mai multor companii se pot identifica şabloanele de navigare ale clienţilor. Aceste şabloane de navigare se pot folosi pentru a sprijini un sistem de produse personalizate pentru vânzări online.

Exploatarea secvenţială este procesul de aplicare a tehnicilor de *data mining* unei baze de date, în scopul de a descoperi relaţiile de corespondenţă care există între o listă ordonată de evenimente. O aplicaţie importantă a tehnicilor de *data mining* secvenţial este explorarea fişierelor log, pentru ca în aceste fişiere log se află înregistrată ordinea în care au fost accesate paginile unui site de către diferiţi utilizatori, într-o anumită perioadă de timp [EZE05].

Autorii articolului [NIU02] prezintă o metoda de realizare a profilelor clienţilor în e-commerce bazată pe ierarhizarea produselor pentru o personalizare efectivă. Aceasta împarte profilul clientului în trei părţi: profilul de bază aflat pe baza datelor demografice ale clientului; profilul preferenţial bazat pe datele comportamentale şi profilul regulă care se referă în principal la regulile de asociere. Pe baza profilelor clienţilor, autorii generează două tipuri de recomandări, care sunt recomandări de interes şi recomandări de asociere. Aceştia propun o structură specială de date denumită arbore de profil pentru căutare şi potrivire efectivă.

Web mining - este punerea în aplicare a tehnicilor de data mining pentru a descoperi modele de pe *Web*. În conformitate cu obiectivele de analiză, *Web* mining poate fi împărţit în trei tipuri diferite, care sunt explorarea utilizării *Web*-ului, explorarea conţinutului *Web* şi a structurii *Web*.

Explorarea structurii *Web*

Explorarea structurii *Web* este procesul de utilizare a teoriei grafurilor pentru a analiza nodurile şi structura conexiunii într-un site *Web*. În funcţie de tipul datelor *Web* structurale, explorarea structurii *Web* poate presupune extragerea de şabloane din hyperlink-uri sau explorarea tag-urilor HTML sau XML din cadrul paginilor. Se construieşte o structură de arbore pentru a analiza şi descrie tag-urile HTML sau XML. Un hyperlink este o componentă structurală care conectează pagina *Web* la o pagina din cadrul site-ului sau din alt site [BIN11].

Datele de structură reprezintă viziunea designerului asupra organizării conţinutului în cadrul unui site. Aceasta organizare este captată prin intermediul structurii de legături între pagini, reflectată prin hyperlink-uri. Datele de structură includ de asemenea şi structura intra-pagină a conţinutului reprezentată de aranjarea tag-urilor HTML sau XML în cadrul paginii. Datele de structură pentru un site sunt captate în mod normal de o harta a site-ului generată automat, hartă care reprezintă structura hyperlink a site-ului[RAG05].

Explorarea conţinutului *Web*

Explorarea conţinutului *Web* este procesul de descoperire a informaţiilor utile din texte, imagini, date audio sau video de pe *Web*. Explorarea conţinutului *Web* este uneori numită *Web text mining* (explorarea textului de pe *Web*) pentru că aceasta este aria de cercetare cea mai răspândită. Tehnologiile care sunt de regulă folosite în explorarea textului de pe *Web* sunt NLP (Natural Language Processing) şi IR (Information Retrivial). Se analizează informaţiile citite sau cu care utilizatorii unui site interacţionează [www3].

Un exemplu de *Web text mining* este următorul: autorii articolului "Exploiting Wikipedia as External Knowledge for Document Clustering" [HU09] au observat că în metodele tradiţionale de grupare (clustering) a textului, documentele sunt reprezentate ca fiind "bagaje de cuvinte" fără să se ţină seama de informaţia semantică din fiecare document. De exemplu dacă două documente utilizează colecţii diferite de cuvinte de bază pentru a reprezenta aceeaşi temă, ele ar putea fi împărţite în mod fals în două grupuri distincte. În această lucrare autorii dezvoltă două abordări "potrivirea exactă" şi "potrivirea relaţională" pentru a pune în corespondenţă documentele text cu conceptele wikipedia şi ulterior cu categoriile Wikipedia. Apoi documentele text sunt grupate pe baza unei metrici de similaritate care combină informaţii legate de conţinutul documentului, informaţii legate de concept şi informaţii legate de categorie. Rezultatele experimentale arată că performanţa clusteringului s-a îmbunătăţit în mod semnificativ prin ralierea reprezentării documentului la conceptele şi categoriile Wikipedia.

Explorarea modului de utilizare a *Web*-ului

Web Usage Mining descrie cum este utilizată o pagină -data şi ora când a fost accesată, adresa IP a staţiei de pe care a fost accesată şi de către cine [www3].

Mobasher [MOB04] prezintă o analiză cuprinzătoare a procesului de personalizare bazat pe explorarea modului de utilizare a *Web*-ului. Scopul acestei lucrări este de a arăta cum pot fi cântărite şi utilizate efectiv, ca parte integrantă a unui sistem *Web* de personalizare, tehnicile de descoperire a şabloanelor cum sunt clustering-ul, explorarea utilizând reguli de asociere şi descoperirea şabloanelor secvenţiale efectuate asupra datelor de utilizare *Web*. Autorul observă că datele log colectate automat de către *Web* şi de serverele de aplicaţii reprezintă bine comportamentul de navigare al vizitatorilor.

Mobasher [MOB04] menţionează că odată ce s-a stabilit clar care sunt tipurile de date cu care urmează să se opereze (date structurale, date de conţinut, date de utilizare), urmează o etapă de pregătire a datelor care se realizează prin procese cum ar fi curăţirea datelor, identificarea paginilor vizualizate, identificarea utilizatorilor şi identificarea tranzacţiilor. Datele curăţate pot fi apoi analizate prin reguli de asociere, clustering, analiză secvenţială, etc. Se poate construi o matrice de tranzacţii – vizualizări care reţine care pagini au fost vizitate de către utilizatori şi câte secunde au stat aceştia pe fiecare pagină.

Lanţurile Markov [VAS09] pot fi utilizate pentru modelarea probabilităţilor de tranziţie între vizualizările paginilor. Sarukkai a propus utilizarea lanţurilor Markov drept un mecanism de modelare pentru minimizarea întârzierilor sistemului, în cadrul unei analize privind utilizarea *Web*-ului [SAR00].

1.2.2. Data mining în medicină

Tehnicile de data mining se aplică cu succes în domeniul medical. La ora actuală termenul Medical Data Mining este consacrat, existând multe cărţi şi conferinţe de Medical Data Mining.

Medicina modernă generează, aproape zilnic, cantităţi enorme de date eterogene. Astăzi provocarea cea mare este de a transforma aceste cantităţi uriaşe de date în informaţii folositoare şi cunoştinţe. Datele medicale pot conţine imagini (RMN), semnale (ECG), informaţii clinice cum ar fi temperatura, nivelul de colesterol, etc., precum şi interpretarea medicului. Tot mai multe şi mai multe proceduri medicale folosesc imagistica ca un instrument preferat de diagnostic, aşadar există o necesitate de a dezvolta metode de exploatare eficientă în bazele de date de imagini.

Învăţarea din datele unor bolnavi permite construirea profilului pacientului bolnav de o anumită boală, îmbunătăţirea diagnosticării [ATI09], se pot analiza relaţiile ce există între anumiţi parametri medicali în vederea realizării de prognoze medicale. Tehnicile de data mining se aplică cu succes în imagistica creierului uman, genetică (pentru prezicerea structurilor proteice, pentru determinarea structurii 3D a proteinelor dată fiind secvenţa lor amino-acidă). Tot în domeniul geneticii se aplică tehnici de data mining pentru descoperirea unor corelaţii între modificările secvenţelor de ADN ale diverşilor indivizi şi susceptibilitatea apariţiei unor boli. Scopul cercetărilor din acest domeniu este îmbunătăţirea diagnosticării bolilor, prevenirea mai eficientă şi tratarea mai uşoară a acestora.

Tehnicile de data mining permit descoperirea de medicamente şi susţin predicţia pe individ (personalizarea medicamentelor).

Datorită faptului ca fişierele ce conţin date medicale pe care se aplică tehnici de data mining sunt legate de subiecţi umani, referindu-se la viata lor privată, problema asigurării confidenţialităţii datelor este foarte importantă [GAN07]. În cadrul lucrări "Anonymizing Healthcare Data: A Case Study on the Blood Transfusion Service" [MOH09] autorii expun îngrijorările legate de confidenţialitatea informaţiilor privind transfuziile sanguine din cadrul sistemului de sharing existent între serviciul de transfuzii de sânge al Crucii Roşii din Hong Kong şi spitalele publice şi identifică provocările majore care fac metodele tradiţionale de anonimizare a datelor inaplicabile în acest caz. Autorii au propus un nou model de asigurare a intimităţii numit LKC-privacy, împreună cu un algoritm de anonimizare pentru a corespunde cerinţelor de intimitate a informaţiilor în cazul Serviciului de Transfuzii de Sânge. Experimentele făcute pe date reale au demonstrat că algoritmul de anonimizare propus poate reţine în mod eficient informaţiile esenţiale în date anonime, pentru a fi analizate şi este scalabil pentru anonimizarea unor seturi de date mari.

Tehnologiile transcriptomice reprezintă instrumente promiţătoare pentru identificarea noilor gene implicate în îmbătrânirea cerebrală sau în bolile neurodegenerative cum ar fi boala Alzheimer. Aceste tehnologii produc date biologice masive care până acum au fost dificil de exploatat. În acest context autorii articolului "GeneMining: Identification, Visualization, and Interpretation of Brain Ageing Signatures" [SAL09] propun explorarea genelor (GeneMining), o metodologie multidisciplinară care are drept scop dezvoltarea unor strategii noi pentru analizarea datelor şi proiectarea unor instrumente interactive pentru a ajuta biologii să identifice, vizualizeze şi să interpreteze semnalmentele îmbătrânirii creierului. Pentru a adresa problema specifică a descoperii semnalmentelor îmbătrânirii

creierului autorii combină şi aplică instrumentele existente punând accentul pe noile metode eficiente de Data Mining bazate pe şabloane secvenţiale.

O aplicaţie de Data Mining dezvoltată pe baza contractului cu NR. 99-II CEEX 03 – INFOSOC 4091 / 31.07.2006 de către Univ. Medicina şi Farmacie "VictorBabes" Timişoara – coordonator, Universitatea Politehnica din Timişoara, Universitatea de Vest din Timişoara, Spitalul Universitar de Obstetrică-Ginecologie "Dumitru Popescu" Timişoara şi Bridgeman SRL este aplicaţia MaternQual.

MaternQual este un sistem informatic integrat de evaluare complexă a factorilor asociaţi predicţiei riscului şi calităţii în obstetrică. MaternQual permite culegerea şi prelucrarea datelor medicale în obstetrică precum şi utilizarea tehnicilor de data mining pentru procesarea acestora. Aplicaţia permite identificarea factorilor de risc primari şi secundari în naşterea prematură, precum şi predicţia riscului de naştere prematură.

Un alt instrument de Data Mining în domeniul medical este aplicaţia Neonat, care permite înregistrarea, procesarea digitala şi analiza plânsetelor nou - născuţilor utilizând tehnici de data mining [ROB11a]. Aplicaţia a fost dezvoltată în cadrul grantului CNCSIS, tip A, nr 755, cu tema "Noi direcţii în diagnosticul şi managementul suferinţei cerebrale perinatale şi consecinţelor postnatale ale acesteia".

A fost propus un sistem de data mining în domeniul medical la nivel de regiune (cu adresare specială către euro-regiunea DKMT (Danube-Kris-Mures-Tisa)) [ROB11c]. Sistemul urmăreşte centralizarea datelor de la spitalele localizate pe teritoriul unei regiuni, într-o bază de date şi apoi analiza acestor date cu ajutorul instrumentului de data mining şi machine learning din *Weka*. Etapele propuse sunt următoarele: datele medicale din bazele de date ale spitalelor sunt exportate în format *XML*, potrivit standardului *HL7 CDA* [SCH06]. Acestea sunt apoi centralizate în mod automat pe un server, prin intermediul unor apeluri de servicii *web*, într-o bază de date. Datele de interes pentru studiul realizat sunt extrase din baza de date şi convertite în format *ARFF* [ROB10b]. Ultima etapă constă în preprocesarea şi analizarea acestora cu algoritmii din *Weka*, cu scopul de a obţine concluzii medicale relevante [ROB10c].

1.2.3. Data mining în educaţie

În zilele noastre, la fel ca şi în toate domeniile sociale, tehnologiile de informare şi comunicare, au explodat în cadrul sistemului educaţional. Astfel este posibilă capturarea şi compilarea mai multor tipuri de informaţii într-o modalitate foarte simplă şi necostisitoare cum ar fi datele administrative ale şcolilor, registrele liceelor şi universităţilor, înregistrările computerizate ale studenţilor, platformele e-learning, înregistrarea activităţilor, sistemele de învăţare cu ajutorul calculatorului, etc.

Educational Data Mining este procesul de transformare a datelor brute, compilate de sistemele educaţionale în informaţii folositoare care pot fi utilizate pentru luarea de decizii informate şi formularea răspunsurilor la întrebările legate de cercetare.

Data mining se aplică pe date care vin din mediul educaţional cu scopul de a obţine o mai bună înţelegere a procesului de învăţare a studenţilor şi de a îmbunătăţi calitatea şi costurile din sistemul educaţional.

EDM (*Educational Data Mining*) are 3 obiective mari:

- *Obiective pedagogice* – îmbunătăţirea conceperii materialelor didactice, cu scopul de a îmbunătăţi performanţa academică a studenţilor
- *Obiective de management* – pentru a optimiza organizarea şi întreţinerea infrastructurii educaţionale, zonele de interes, cele mai solicitate cursuri
- *Obiective comerciale* – realizarea de segmentări ale pieţei şi facilitarea recrutării studenţilor

Educational data mining este o disciplină aflata la intersecţia dintre *data mining* şi Pedagogie. Pedagogia contribuie cu cunoştinţele intrinseci procesului de învăţare. *Educational data mining* ca şi *data mining* este o tehnologie care integrează tehnici multidisciplinare, de la extragerea informaţiilor din baze de date prin simple comenzi SQL până la utilizarea unor algoritmi de învăţare automată extraşi din câmpul Inteligenţei Artificiale. Printre cele mai folosite tehnici în domeniul educaţiei se enumeră clusteringul, regulile de asociere şi analiza şabloanelor.

Prin intermediul *data mining* o universitate poate prezice cu o precizie între 75 şi 80 % care studenţi vor absolvi şi care nu. Universitatea poate folosi această informaţie pentru a-şi concentra atenţia asupra acelor studenţi care sunt în pericol de a nu absolvi [DEK09].

În universităţile care oferă învăţământ la distanţă numărul de studenţi exmatriculaţi este mult mai mare decât în universităţile care nu oferă această formă de învăţământ. Limitarea exmatriculărilor este esenţială în universităţi şi de aceea abilitatea de a prezice care studenţi vor fi exmatriculaţi este folositoare din mai multe motive [KOT09].

Prin intermediul regulilor de asociere se pot analiza erorile care apar împreună în timpul rezolvării exerciţiilor. Scopul găsirii acestor asocieri este ca profesorul să reflecteze şi să revizuiască materialele de curs sau să sublinieze aceste subtilităţi studenţilor [MAR05]. Reguli de asociere se pot aplica pentru a recomanda studenţilor cursurile pe care să le urmeze [BEN06].

Clustering-ul poate fi aplicat pentru a caracteriza studenţii cu dificultăţi. Clasificarea – arborii de decizie - se poate utiliza pentru a prezice rezultatele studenţilor la anumite examene [MAR05].

Un studiu de *Educational Data Mining* a fost realizat, pe datele studenţilor din anul universitar 2008-2009 de la Facultatea de Automatică şi Calculatoare a Universităţii "Politehnica din Timişoara" (2100 de studenţi). Studiul s-a realizat pe baza unor date civice şi şcolare obţinute din Sistemul Informatic de Gestiune a Şcolarităţii şi a constat în realizarea unei analize statistice la nivel de atribut, construirea unui model de regresie pentru media multianuală şi gruparea datelor în două grupuri pe baza caracteristicilor comune [ROB10b].

1.2.4. Data mining în combaterea terorismului

Tragedia de la 11 septembrie a avut efecte incomensurabile şi permanente atât asupra Statelor Unite cât şi asupra restului lumii şi a adus subiecte precum securitatea şi apărarea pe prim plan. Pentru a ajuta la prevenirea unor astfel de catastrofe, un program de succes de securitate ar include dispoziţii care să securizeze frontierele, sectorul de transport şi infrastructura critică. Un factor critic al unui astfel de program ar fi capacitatea de sintetiză şi analiză a datelor din surse multiple. În Mai 2003, a avut loc la San Francisco, un workshop la care s-a discutat

modul în care Data Mining şi Machine Learning pot ajuta la analizarea unor date complexe, din surse diferite, cu scopul de a preveni activitatea teroristă viitoare. Denumirea workshopului este: "Workshop on Data Mining for Counter Terrorism and Security". Subiectele de interes de la acest workshop au inclus:

- Metode de a integra surse de date eterogene, cum ar fi text, internet, video, audio, date biometrice, precum şi de vorbire
- Metode scalabile pentru a depozita surse de date răsfirate
- Identificarea tendinţelor în activităţile de grup sau singulare
- Identificarea şabloanelor cu scopul de a identifica persoana
- Data Mining în domeniul securităţii aviaţiei, securitatea portuară
- *Web* Data Mining pentru a detecta tendinţele teroriste

Detectarea teroriştilor presupune împărţirea indivizilor în două clase: indivizi periculoşi şi indivizi nepericuloşi. Acest proces este oarecum diferit de alte probleme clasice de Data Mining pentru că indivizii din prima clasa (cei periculoşi) fac tot posibilul să arate ca şi cei din a două clasă [SKI03].

Autorul D.B. Skilikorn a demonstrat în articolul [SKI03] că un aspect important al detectării teroriştilor este abilitatea de a detecta corelaţii locale la o scară mica, în contradictoriu cu corelaţiile difuze la scară mare.

În contextul securităţii naţionale, Data Mining poate fi un potenţial mijloc de a identifica activităţi teroriste cum ar fi transferuri de bani şi comunicaţii, precum şi de a identifica şi urmări teroriştii însăşi, folosind date de călătorie şi emigrare [SEI08].

Deşi Data Mining permite extragerea de şabloane şi relaţii, tehnologia are limitarea de a nu informa utilizatorul cu privire la valoarea sau semnificaţia acestor relaţii sau şabloane. Aceste lucruri trebuie să fie cunoscute de către utilizator [SEI08].

Câteva exemple notabile de utilizare a tehnicilor de data mining pe teritoriul Statelor Unite sunt următoarele: Departamentul de Justiţie al Statelor Unite utilizează Data Mining pentru a determina şabloane de crime şi a regla alocările de resurse în consecinţă; Administraţia Federală a Aviaţiei, foloseşte data mining pentru a analiza datele de prăbuşire a avioanelor, cu scopul de găsi defectele comune şi a recomanda măsuri de precauţie[SEI08].

Data mining contribuie la combaterea terorismului prin tehnici precum reguli de asociere, analiza legăturilor, clustering, clasificare şi detectarea anomaliilor. Aceste tehnici se aplică prin intermediul reţelelor neuronale, a arborilor de decizie, prin programare logică inductivă, mulţimi Rough, analiza legăturilor, care este bazată pe teoria grafurilor şi k-nearest- neighbor [BHA08].

Un exemplu de utilizare a regulilor de asociere pentru combaterea criminalităţii este următorul: daca John vine dintr-o tara X şi este asociat cu James care are un cazier substanţial şi se constată ca ţara X are un procent mare de atacuri teroriste atunci ar trebui ca John să fie pus sub urmărire [BHA08].

O altă tehnică de data mining folositoare în acest domeniu este detectarea anomaliilor. Un exemplu în acest sens ar fi cursanţii unui curs de pilotaj care se arată mult mai interesaţi de felul în care zboară un avion decât de felul în care decolează şi aterizează [BHA08].

Analiza legăturilor este o tehnica de Data Mining folosită la determinarea şabloanelor anormale. Aşa cum a fost menţionat anterior, analiza legăturilor foloseşte multe tehnici de teorie a grafurilor. Analiza legăturilor este de asemenea folosită în *Web* mining, în special în explorarea structurii *Web*. Unele motoare de căutare cum ar fi Google folosesc o formă de analiză a legăturilor pentru a afişa rezultatele unei căutări [BHA08].

Analiza legăturilor se foloseşte pentru a găsi asociaţii interesante şi pentru a determina cum se poate reduce un graf. Mulţi teoreticieni ai grafurilor lucrează la probleme de reducere a acestora. O altă provocare a utilizării analizei legăturilor în combaterea terorismului este aceea de a rezolva situaţiile de informaţii parţiale. De exemplu agenţia A dispune de un graf parţial, agenţia B de un alt graf parţial, şi agenţia C de cea de-a treia bucată. Întrebarea este cum se pot găsi asociaţiile dintre grafuri când niciuna din agenţii nu are o imagine completă [BHA08].

În timp ce de toate aceste aplicaţii de data mining pot beneficia oameni şi pot fi salvate vieţi, există, de asemenea, şi aspecte negative ale acestei tehnologii deoarece ar putea fi o ameninţare la adresa vieţii private a persoanelor fizice[BHA08].

1.2.5. Sistemele de Data Mining utilizate de agenţiile federale din SUA

În anul 2004 Oficiul de Contabilitate Generală al Statelor Unite (GAO - The General Accounting Office) a înaintat un raport către mai multe instituţii printre care şi Senatul Statelor Unite cu scopul de a prezenta şi a descrie caracteristicile sistemelor de Data Mining utilizate de agenţiile federale.

Misiunea GAO braţul de audit, evaluare şi investigare al Congresului, este de a asista Congresul în vederea îndeplinirii responsabilităţilor constituţionale şi de a ajuta îmbunătăţirea performanţei şi responsabilităţii guvernului federal pentru poporul american. GAO examinează utilizarea fondurilor publice; evaluează programele şi politicile federale; furnizează analize, recomandări şi alte rapoarte similare pentru a ajuta Congresul în luarea de decizii informate privind supravegherea, politica şi finanţarea. Angajamentul luat de GAO pentru o guvernare bună este reflectat de valorile sale intrinseci de responsabilitate, integritate şi încredere [GAO04].

Agenţiile federale utilizează Data Mining cu o varietate foarte mare de scopuri, de la îmbunătăţirea serviciilor şi performanţei până la detectarea şabloanelor şi activităţilor teroriste. Raportul realizat de GAO analizează dacă agenţiile şi departamentele federale utilizează sau intenţionează să utilizeze *data mining* în activitatea lor. Au fost analizate un număr de 128 de departamente federale şi agenţii. Dintre acestea un număr de 52 de instituţii utilizează sau intenţionează să utilizeze *data mining*. Aceste departamente şi agenţii au raportat un număr de 199 eforturi de *data mining* din care 68 erau în fază de plan iar 131 erau operaţionale. Raportul este foarte amplu se întinde pe 71 de pagini, în acest subcapitol este prezentat un scurt rezumat al acestui raport, cu scopul de a ilustra puterea şi importanţa tehnologiei *data mining*, precum şi nivelul de top la care s-a ajuns cu utilizarea ei.

O parte din cele 128 de agenţii şi departamente analizate pot fi vizualizate în tabelul următor:

Departamentul	Agenţia
Departamentul de Agricultură	Serviciul de Inspecţie al Sănătăţii Faunei şi Florei
	Serviciile Forestiere
Departamentul de Comerţ	Administraţia Naţională a Oceanelor şi Atmosferei
	Administraţia Dezvoltării Economice
Departamentul de Apărare	Serviciul de Securitate al Apărării
	Agenţia de Reducere a Ameninţării
Departamentul Serviciilor de Sănătate şi Umane	Institutele Naţionale de Sănătate
	Centrele pentru Îngrijire şi Servicii Medicale
Departamentul Securităţii Naţionale	Directoratul de Ştiinţă şi Tehnologie
	Biroul de Servicii pentru Cetăţenie şi Emigrări
Departamentul de Interne	Administraţia Funciară
	Biroul de Reclamaţii
Departamentul de Justiţie	Biroul de Alcool, Tutun, Arme de Foc şi Explozibili
Departamentul Transporturilor	Administraţia Federală a Aviaţiei
	Administraţia Federală a Tranzitului
Departamentul Trezoreriei	Controlul Monetar
	Biroul de Gravări şi Imprimare

Tabel 1.1 - O parte din departamentele şi agenţiile analizate

În tabelul 1.2 se poate vizualiza care sunt scopurile eforturilor de data mining demarate şi câte acţiuni au fost demarate în sprijinul acestor eforturi de către agenţiile şi departamentele analizate.

Scopurile eforturilor de data mining	Număr eforturi
Analizarea inteligenţei şi detectarea activităţilor teroriste	14
Detectarea activităţilor sau şabloanelor criminale	15
Administrarea resurselor umane	17
Analizarea informaţiilor ştiinţifice şi de cercetare	23
Detectarea fraudelor şi abuzurilor	24
Îmbunătăţirea serviciilor sau performanţei	65

Tabel 1.2 - Scopul şi numărul eforturilor de data mining

Departamentul Apărării a raportat cele mai multe eforturi utilizând data mining în vederea îmbunătăţirii performanţei, serviciilor şi gestiunea resurselor umane. Apărarea şi-a concentrat de asemenea cel mai frecvent eforturile în scopul analizării inteligenţei şi detectării activităţilor teroriste, pe locul doi în această ierarhie situându-se Departamentul Naţional de Securitate, Justiţie şi Educaţie.

Departamentul de Educaţie a raportat concentrarea celor mai multe eforturi în vederea detectării fraudei, deşeurilor şi abuzurilor în timp ce Administraţia Naţională a Aeronauticii şi Spaţiului îşi canalizează majoritatea eforturilor de data mining (21 din 23) către analizarea informaţiilor ştiinţifice şi de cercetare. Eforturile de data mining pentru detectarea activităţilor sau şabloanelor criminale, au fost însă împărţite în mod aproximativ egal între agenţiile raportatoare.

În plus, din cele 199 de eforturi de data mining identificate, 122 utilizează informaţii personale. Pentru aceste eforturi, scopurile principale erau detectarea fraudei, deşeurilor şi abuzurilor, detectarea activităţilor şi şabloanelor criminale; analizarea inteligenţei şi detectarea activităţilor teroriste; şi creşterea conformităţii fiscale.

Un exemplu al dezvoltării eforturilor la scara largă lansat în urma atacurilor de la 11 septembrie este Sistemul Antitero de schimb de informaţii între state, numit şi MATRIX. MATRIX este utilizat la acest moment în cinci state, furnizează capacitatea de a stoca, analiza şi schimba date legate de terorism precum şi alte date legate de serviciile criminale între agenţiile din diferite state, între state şi între state şi agenţiile federale. Informaţiile din bazele de date MATRIX includ date legate de istoricul criminalilor, înregistrări privind încarcerarea, înregistrări privind vehiculele, privind permisele de conducere, autovehicule şi fotografii digitale. Sistemul MATRIX a născut în rândul opiniei publice nemulţumiri, datorită încălcării intimităţii. Sistemul analizează datele personale ale foarte multor indivizi pentru a vedea care are profilul unui terorist.

Unele obiective ale data mining se concentrează pe activităţi umane, şi prin urmare există o posibilitate destul de mare să implice informaţii personale. Câteva exemple de astfel de obiective sunt detectarea fraudei, a abuzurilor, detectarea activităţilor şi şabloanelor criminale, gestiunea resurselor umane şi analiza serviciilor de inteligenţă. În continuare sunt prezentate exemple de eforturi de data mining concentrate în acest scop:

- *Detectarea fraudei, deşeurilor şi abuzurilor* Eforturile Administraţiei Beneficiilor Veteranilor C & P, prin Sistemul de Analiză a Datelor de Plată, explorează datele privind pensia şi compensaţiile veteranilor pentru a verifica şi proba eventualele fraude.
- *Detectarea activităţilor şi şabloanelor criminale*. Departamentului de Educaţie prin Sistemul Iniţiativa Furtului de Identitate, se concentrează către identificarea cazurilor de furt de identitate care implică împrumuturile pentru educaţie.
- *Gestiunea resurselor umane*. Departamentul de Resurse Umane Oracle al Forţei Aeriene Americane utilizează data mining pentru a furniza informaţii legate de promovări, gradaţii, permisii precum şi alte informaţii relevante pentru planificarea resurselor umane.
- *Analiza serviciilor de inteligenţă şi detectare a activităţilor teroriste*. Agenţia de servicii inteligente în domeniul apărării Verity K2 Enterprise explorează date din comunitatea agenţiilor de servicii şi din surse de pe Internet pentru a identifica terorişti străini sau cetăţeni americani conexaţi unor activităţi teroriste străine.

Pe de alta parte, o altă categorie de eforturi nu se concentrează în mod neapărat pe activităţi umane şi deci nu implică informaţii personale. Exemple de astfel de eforturi sunt cele efectuate în scopul analizării informaţiilor ştiinţifice şi de cercetare. Administraţia Naţională de Aeronautică şi Spaţiu, spre exemplu, explorează seturi complexe de date ştiinţifice legate de planetă pentru a identifica şabloane şi legături pentru a detecta evenimente ascunse (sistemul este denumit Maşina de Învăţare şi Data Mining pentru o Înţelegere mai bună a Datelor Dimensionale legate de Pământ - Machine Learning and Data Mining for Improved Data Understanding of High Dimensional Earth Sensed Data).

În tabelul de mai jos se pot vedea câteva proiecte care sunt în stadiul operaţional

Organizaţia	Denumirea sistemului	Descrierea sistemului	Scopul sistemului
Departamentul de Agricultură	Depozitul de date Financiare	Este sistemul de raportare intern de gestiune financiară. Data mining este făcută pentru rapoarte efectuate ad-hoc şi la cerere.	Gestiunea financiară
Agenţia de Management al Riscului	CAE	Este parte a unui proiect mandatat de Congres pentru a ajuta Agenţia de Management al Riscului în controlarea fraudei, deşeurilor şi abuzurilor în programul Corporaţiei de Asigurări Federal Crop.	Detectarea fraudei, deşeurilor şi abuzurilor
Oficiul de Invenţii şi Mărci al SUA	Modelul Proiecţiei compensaţiilor în Depozitul de date al companiilor	Generează şi pune la dispoziţie date privind proiecţia compensaţiilor, atât salariile cât şi sporurile în cazul angajaţilor curenţi şi asupra angajărilor planificate.	Gestionarea resurselor umane
Agenţia Comisariatului Apărării	Sistemul de suport al Deciziilor Corporative/ Sistemul de Management al Operaţiunilor Comisariatului	Explorează date pentru a produce date analitice privind operaţiunile comisariatului. Furnizează informaţii cum ar fi care produse se vând mai bine sau cât de cinstite sunt casierele.	Îmbunătăţirea serviciilor sau performanţei
Agenţia Serviciilor de Apărare	Întreprinderea Verity K2	Explorează date din comunitatea serviciilor secrete şi căutări pe Internet pentru identificarea teroriştilor străini sau a cetăţenilor americani conexaţi cu activităţi de terorism din străinătate.	Analizarea serviciilor secrete şi detectarea activităţilor teroriste
Agenţia Serviciilor de Apărare	Chestionarea Sistemului de Date ambulatorii	Pune în corespondenţă diagnosticul iniţial al pacienţilor cu rezultatul testărilor şi diagnosticării ulterioare. Permite identificarea timpurie a bolilor şi afecţiunilor.	Monitorizarea sănătăţii publice
Agenţia	Modus	Este un instrument de	Detectarea

Serviciilor de Apărare	Operandi	investigaţie utilizat pentru identificarea şi urmărirea tendinţelor în comportamentul criminalistic. Leagă caracteristicile crimelor şi furnizează detalii privind locul producerii crimelor precum şi alţi factori criminalistici.	activităţilor şi şabloanelor criminale
Administraţia Alimentelor şi medicamentelor	Analiza fonetică şi ortografică computaţională	Este un motor de căutare care furnizează rezultate indicând cât de similare sunt denumirile a două medicamente din punct de vedere fonetic şi ortografic. Scopul său este de a ajuta la siguranţa evaluării denumirilor private propuse pentru a reduce confuzia legată de numele medicamentului după ce o cerere în acest sens a fost aprobată de FDA.	Îmbunătăţirea siguranţei
Serviciul de Management al Minereurilor	Data Mining pentru sistemul de Management al Informaţiilor Tehnice Baza de date (TIMS)	Este o bază de date corporativă pentru închirierile de gaze şi petrol. Baza de date este explorată pentru sprijinirea dezvoltării politicii. O zonă a data mining este identificarea închirierilor care vor fi în viitorul apropiat. Data mining a relevat faptul că închirierile cu şase sau mai multe puţuri productive într-un an aproape niciodată nu vor fi abandonate în anul următor. O altă aplicare a data mining este în siguranţa operaţiunilor petroliere şi cu gaze. De exemplu, data mining a arătat faptul că accidentele ating o cotă maximă joia dimineaţa.	Îmbunătăţirea serviciilor sau performanţei
Administraţia	Maşina de	Va găsi şabloane şi relaţii	Analiza

Naţională de Aeronautică şi Spaţiu	Învăţare şi Data Mining pentru o Înţelegere mai bună a Datelor Dimensionale legate de Pământ	în seturi de date vaste şi complexe legate de ştiinţa ce studiază planeta, pentru evenimentele mici şi ascunse în semnalele datelor mai mari. Va construi noi capabilităţi de înţelegere a datelor ştiinţifice ale NASA.	informaţiilor ştiinţifice şi de cercetare
Administraţia Naţională de Aeronautică şi Spaţiu	Analiza Seriilor de Timp Geofizice	Va dezvolta un set de algoritmi pentru identificarea şabloanelor în cadrul activităţilor temporale. Datele vor constitui traiectoriile şi mişcările obiectelor în cadrul imaginilor.	
Administraţia Naţională de Aeronautică şi Spaţiu	Data Mining pentru Modelul Numeric 3-D pentru Previzionarea Randamentului şi Aplicarea sa în Cercetarea Atmosferică	Va automatiza analiza randamentului modelului de vreme, observarea şi datele din satelit pentru a permite o înţelegere mai bună a ştiinţei privind dinamica vremii pentru a previziona evenimentele viitoare.	Analizarea informaţiilor ştiinţifice şi de cercetare
Comisia de Reglementare Nucleară	Datele de Raportare a Evenimentelor Titularilor (Licenţiaţilor)	Identifică şabloanele şi tendinţele siguranţei nucleare în evenimentele nucleare comerciale.	Îmbunătăţirea siguranţei

Tabel 1.3 - Proiecte de data mining în stadiul operaţional

1.3. Clasificarea datelor

1.3.1. Noţiuni generale

Clasificarea este una dintre cele mai populare tehnici de data mining. Clasificarea permite construirea unui model clasificator pe baza datelor de antrenare, testarea acestuia pe datele de test şi apoi utilizarea lui pentru a realiza predicţii. Construirea modelului clasificator se poate face cu ajutorul unor algoritmi specializaţi precum *Naive Bayes, k-Nearest Neighbor, C4.5* etc. Modelul construit poate fi utilizat pentru a prezice valoarea nominală a unui atribut clasă.

Clasificarea se poate utiliza de exemplu în domeniul medical, pentru a extrage din datele unor pacienţi reguli pe care apoi să le utilizăm pentru a diagnostica noi pacienţi pe baza simptomelor lor. Se pot clasifica pacienţii cu probleme la inimă pe baza unor diferite tipuri de boli de inimă.

Presupunând că *D* este un set de date, el poate fi privit ca un set de tupluuri $x_1, x_2 \dots x_n$ unde $x_1, x_2 \dots x_n$ sunt valorile unor atribute $A_1, A_2 \dots A_n$ relevante pentru un anumit studiu. Se pot defini diferite clase sau categorii $C = \{C_1, C_2 \dots C_m\}$ de instanţe în funcţie de caracteristicile lor. Problema clasificării este de a definii o funcţie $f : D \rightarrow C$ în care fiecare instanţă $t_i \in D$ este pusă în corespondenţă cu ajutorul lui *f (ti)* către o anumită clasă C_j.

Diverşi algoritmi realizează diverse procesări cu scopul de a clasifica datele. În literatură există un număr foarte mare de algoritmi de clasificare şi numărul lor continuă să crească pentru că încă nu s-a descoperit un algoritm care să funcţioneze mai bine decât toţi ceilalţi algoritmi pe orice set de date şi pentru că nici unul din algoritmii existenţi nu asigură o acurateţe a predicţiei de 100 pe orice set de date. Capitolul prezintă sintetic trei algoritmi de clasificare foarte populari şi consideraţi algoritmi de referinţă în domeniul clasificării (*C4.5*, *Naive Bayes* şi *k-Nearest Neighbor).* Aceştia au fost consideraţi de IEEE International Conference on Data Mining (ICDM 2006) ca făcând parte din topul celor mai influenţi 10 algoritmi de data mining utilizaţi de comunitate [WU07]. Algoritmii menţionaţi au fost utilizaţi în capitolele 4, 5 şi 6 pentru a construi clasificatori şi pentru a compara rezultatele obţinute de aceşti algoritmi cu cele obţinute de algoritmii propuşi.

1.3.2. Algoritmul Naive Bayes

Principiul algoritmului: Algoritmul urmăreşte construirea unui modelul predictiv bazat pe calculul unor probabilităţi.

Pentru fiecare instanţă pe care o clasifică, algoritmul calculează care este probabilitatea să aparţină la fiecare din clasele existente. Clasa estimată va fi cea care are probabilitatea de apariţie cea mai mare [WU07]. Determinarea probabilităţii ca o instanţă să aparţină unei clase implică realizarea unui produs de probabilităţi în care factorii produsului sunt, în cazul atributelor nominale, probabilităţile valorilor atributelor, iar în cazul atributelor numerice, funcţiile de densitate de probabilitate. Probabilităţile valorilor tuturor atributelor nominale precum şi media şi deviaţia standard (necesare pentru calculul densităţii de probabilitate) în cazul atributelor numerice se vor calcula pe baza datelor din mulţimea de antrenament, deci aceste calcule vor fi făcute în etapa de antrenare. În continuare este prezentată funcţionarea algoritmului pe paşi.

Etapa de antrenament:

a) Se calculează pentru fiecare clasă probabilitatea ca o instanţă să aparţină clasei respective (numărul de instanţe care au clasa considerată, raportat la numărul total de instanţe)
b) Pentru fiecare valoare a fiecărui predictor nominal, se calculează probabilitatea condiţionată de fiecare clasă a valorii respective (numărul de apariţii ale valorii nominale printre cele m instanţe care aparţin etichetei clasei considerate)

c) Pentru fiecare atribut numeric se calculează media aritmetică şi deviaţia standard cu privire la numărul de instanţe din mulţimea de antrenament care fac parte din clasa considerată

Etapa de testare:

În această etapă se parcurg instanţele din mulţimea de testare, se calculează, pentru fiecare instanţă, pe baza datelor obţinute în etapa de antrenament, care este probabilitatea de a aparţine la fiecare din clasele existente. Pentru fiecare instanţă va fi estimată clasa care are probabilitatea cea mai mare. Se compară clasele estimate cu clasele reale. Se calculează acurateţea modelului care este egală cu numărul de instanţe din mulţimea de test clasificate corect, raportat la numărul total de instanţe din mulţimea de test. Dacă rezultatul este mulţumitor, modelul poate fi utilizat pentru predicţii viitoare.

1.3.3. Algoritmul k-Nearest Neighbor

Principiul algoritmului: Algoritmul urmăreşte găsirea pentru fiecare instanţă din mulţimea de test a instanţelor similare din mulţimea de antrenament. Algoritmul se bazează pe principiul „Dacă merge ca o raţă, măcăie ca o raţă şi arată ca o raţă, atunci probabil că este vorba de o raţă...” [TAN05].

Algoritmul ***k-Nearest Neighbor*** construieşte un model clasificator, care clasifică instanţele pe baza clasei majoritare pe care o au cele mai apropiate **k** instanţe de clasa considerată. Pentru a calcula distanţele dintre instanţe se folosesc diferite măsuri cum ar distanţă Euclidiana [DEZ09], distanţă Mincovski, distanţă Manhatan.

Etapa de antrenament:

a) Se alege valoarea lui K
b) Se alege care va fi distanţa care se va utiliza ca măsură a apropierii (Euclidiana, Mincovski, Manhatan)

Etapa de testare:

a) pentru fiecare instanţă din mulţimea de testare se calculează distanţa faţă de fiecare instanţă din mulţimea de antrenament
b) se determină care sunt cele mai apropiate k instanţe din mulţimea de antrenament, de instanţa de testare considerată
c) se determină care este clasa dominantă printre cele mai apropiate k instanţe, pe baza votului majorităţii
d) instanţei din mulţimea de test îi va fi atribuită clasa rezultată prin votul majorităţii
e) se analizează performanţele modelului şi dacă aceasta este mulţumitoare, modelul poate fi utilizat pentru predicţii

1.3.4. Algoritmul C4.5

Principiul algoritmului: Algoritmul face parte din categoria arborilor de clasificare [QUI93] (sau de decizie). Algoritmul urmăreşte construirea pe baza mulţimii de antrenament a unui model compus din reguli care poate fi reprezentat sub forma

unui arbore de decizie. Rădăcină arborelui corespunde întregii mulţimi de antrenament. Fiecare nod al arborelui reprezintă un test după un atribut şi corespunde unei submulţimi a mulţimii de antrenament. Ramurile care pornesc de la un nod reprezintă rezultatele testului. Frunzele reprezintă clasele. Structura unui arbore de decizie construit cu algoritmul C4.5 poate fi vizualizată în figura 5.1. Pentru trasarea figurii s-a considerat că atributul 1 este nominal (cu trei valori posibile), atributul 2 este numeric, atributul 3 este nominal (cu două valori posibile) iar atributul clasă poate lua două valori.

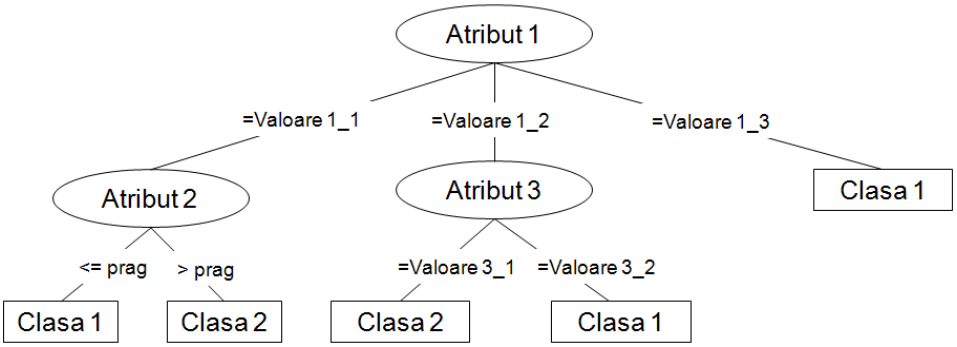

Fig. 1.3 - Structura unui arbore de decizie construit cu C4.5

Construcţia arborilor este de tip inductiv [QUI86]. Inducţia semnifică generalizarea unei reguli pe baza unor instanţe (sau exemple). Arborii sunt creaţi în mod inductiv pe baza valorilor instanţelor din mulţimea de antrenament. Pentru crearea arborelui se foloseşte un algoritm recursiv: divide et impera. Se va determina care este atributul care împarte cel mai bine datele, apoi instanţele se vor diviza în două subseturi. Dacă în urma divizării un subset conţine mai multe clase, algoritmul se reia pentru subsetul respectiv. Dacă un subset indică o singură clasă algoritmul se opreşte pentru acel subset.

Etapa de antrenament:

a) în prima iteraţie la nivelul rădăcinii sunt prezente toate instanţele din mulţimea de antrenament. Se calculează capacitatea de separare a datelor (se calculează entropia şi câştigul informaţional) pentru fiecare atribut. Entropia este o măsură a dezordinii informaţionale [HOL09]. *Information Gain* (câştigul informaţional) oferă o măsură cantitativă a informaţiei câştigate prin divizare (ordonare) după un anumit atribut Atributul câştigător va fi cel care reuşeşte să inducă cea mai multă ordine în setul de date (cel care are entropia cea mai mică şi câştigul informaţional cel mai mare). Acest atribut se adaugă arborelui.

b) atributele pot să fie nominale sau numerice. Din fiecare atribut nominal pot porni un număr de ramuri egal cu numărul de valori nominale al atributului, iar din fiecare atribut numeric pot porni doar două ramuri: una corespunde valorilor numerice care sunt mai mici sau egale cu o valoare limită (în figura 1.3 a fost numită prag), iar cealaltă corespunde valorilor numerice care sunt peste prag (atributele numerice sunt tratate ca şi atribute nominale cu două valori posibile). Pragul se determină calculând valoarea care conduce la un câştig informaţional cât mai mare din mai multe puncte de împărţire posibile [QUI98]. Punctele de împărţire posibile sunt valorile aflate la jumătatea distanţei dintre valorile numerice consecutive obţinute prin ordonare.

c) instanţele sunt divizate în subseturi, în funcţie de valorile atributului câştigător. Pot să apară două situaţii:
   - în unul dintre subseturi există reprezentată o singură clasă, în acest caz pentru acel subset se creează o frunză şi se opreşte procedura
   - în unul dintre subseturi există reprezentate cel puţin două clase, în acest caz se trece la iteraţia următoare cu subsetul respectiv (se reia algoritmul pentru acest subset)

d) algoritmul se opreşte când nu mai există nici un subset cu două clase reprezentate, au fost epuizate toate atributele sau au fost epuizate toate instanţele. În plus, mai pot fi specificate şi alte reguli de oprire cum ar fi impunerea unei limite pentru adâncimea arborelui

e) arborele construit în această etapă este posibil să se suprapotriveasacă datelor de antrenament (să nu fie suficient de general pentru a clasifica destul de corect instanţele din mulţimea de test, concept cunoscut sub numele de overfitting). Din această cauză se parcurge o etapă de tăiere a arborelui numită *prunning*. Tăierea se realizează parcurgând arborele şi înlocuind cu noduri frunză ramurile care nu sunt relevante [WIT05].

Etapa de testare:

Pentru fiecare instanţă din mulţimea de testare se parcurge arborele de la rădăcină şi până la o frunză, ţinând cont de valorile atributelor instanţelor testate şi de testele din nodurile arborelui. Frunza la care se ajunge reprezintă clasa estimată.

Algoritmul J48 este implementarea în *Weka* a algoritmului C4.5 [MOO09].

1.4. Instrumente utilizate la realizarea de studii de data mining

În prezent există un număr foarte mare de instrumente cu care se pot realiza studii de data mining. Acestea se pot împărţii în:

- instrumente comerciale (*SPSS Clementine, SAS E-Miner, MATLAB, Oracle data mining, SQL Server, SAP data mining, etc)*
- instrumente open-source (*Weka*, *R*, *Orange, Rapid Miner, Ant Miner, etc)*
- instrumente dezvoltate în urma unor prevederi contractuale pentru a rezolva anumite probleme concrete (*MaternQual* - sistem informatic integrat care utilizând tehnici de data mining permite identificarea factorilor de risc primari si secundari, pentru naşterile premature, precum si predicţia riscului de naştere prematură, *NEONAT* - instrument care permite înregistrarea, procesarea digitala si analiza plânsetelor nou născuţilor [ROB11a].

În cadrul tezei atenţia a fost concentrată asupra instrumentelor Open Source, deoarece acestea au permis autorului să îmbunătăţească tehnologia existentă prin îmbunătăţirea unor algoritmi implementaţi în ele, prin adăugarea unor algoritmi noi, etc. Două dintre cele mai folosite instrumente la nivel mondial sunt *Weka* şi R [HOR09]. Acestea vor fi prezentate pe scurt în continuare.

1.4.1. Weka

Weka este un program open-source de machine learning / data mining, dezvoltat în Java de către Univeristatea Waikato din Noua Zeelanda. Prima versiune internă a lui *Weka* a fost lansată în anul 1994, iar prima versiune publică a lui, versiunea 2.1 a fost lansată pe piaţă în anul 1996. În prezent s-a ajuns la versiunea 3.6.3 care este ultima versiune stabilă. *Weka* este un soft deosebit de util în educaţie, cercetare şi aplicaţii [BOU08].

Versiunea 3.6.3 a lui *Weka* oferă următoarele facilităţi:

- cuprinde 70 de instrumente de preprocesare (pentru discretizare, reducerea zgomotului, selectarea atributelor, etc)
- 117 de algoritmi de clasificare şi regresie (printre aceştia se găsesc J48, NaiveBayes, Random Forest)
- 11 algoritmi de clustering (cum ar fi SimpleKMeans, XMeans)
- 6 algoritmi pentru găsirea regulilor de asociere, printre care se găseşte şi algoritmul Apriori
- 3 interfeţe grafice: Explorerul, Experimenter şi KnoledgeFlow

Fişierele de date asociate cu *Weka* sunt fişierele arff (Attribute-Relation File Format), dar se pot deschide şi fişiere C4.5, csv, dat, se pot importa date de pe internet sau din baze de date SQL.

Un exemplu de fişier arff este prezentat în figura 1.4.

```
@relation vreme

@attribute vremea {soare, innorat, ploios}
@attribute temperatura numeric
@attribute umiditate numeric
@attribute vant {da, nu}
@attribute jucam_golf {da, nu}

@data
soare,85,85,nu,nu
soare,80,90,da,nu
innorat,83,86,nu,da
ploios,70,96,nu,da
ploios,68,80,nu,da
ploios,65,70,da,nu
innorat,64,65,da,da
soare,72,95,nu,nu
soare,69,70,nu,da
ploios,75,80,nu,da
soare,75,70,da,da
innorat,72,90,da,da
innorat,81,75,nu,da
ploios,71,91,da,nu
```

Fig. 1.4 - Structura unui fişier *.arff

Atributele sunt precedate de eticheta @attribute şi pot fi numerice, nominale (de genul soare, înnorat, ploios), mai pot fi de tip *string* şi *data*. În cazul tipului *data*

se va specifica formatul de dată [www4]. Datele sunt precedate de eticheta @data, instanţele se plasează pe rânduri diferite separate prin virgulă.

Interfaţa de start a lui *Weka* este cea din figura de mai jos:

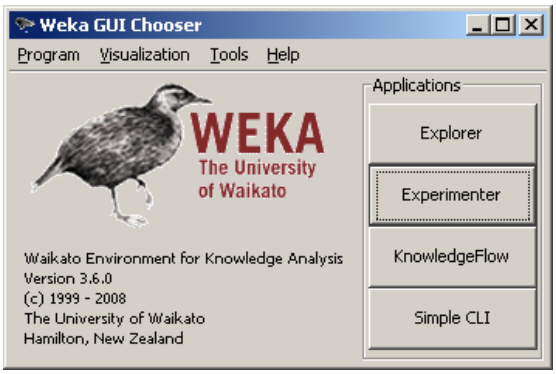

Fig.1.5 - Interfaţa principală a lui *Weka*

De pe interfaţa principala se pot rula patru aplicaţii: Explorer, Experimenter, KnowledgeFlow şi Simple CLI. Explorer este o aplicaţie care permite realizarea analizei exploratorii a datelor. Interfaţa lui se poate vedea în figura 1.6.

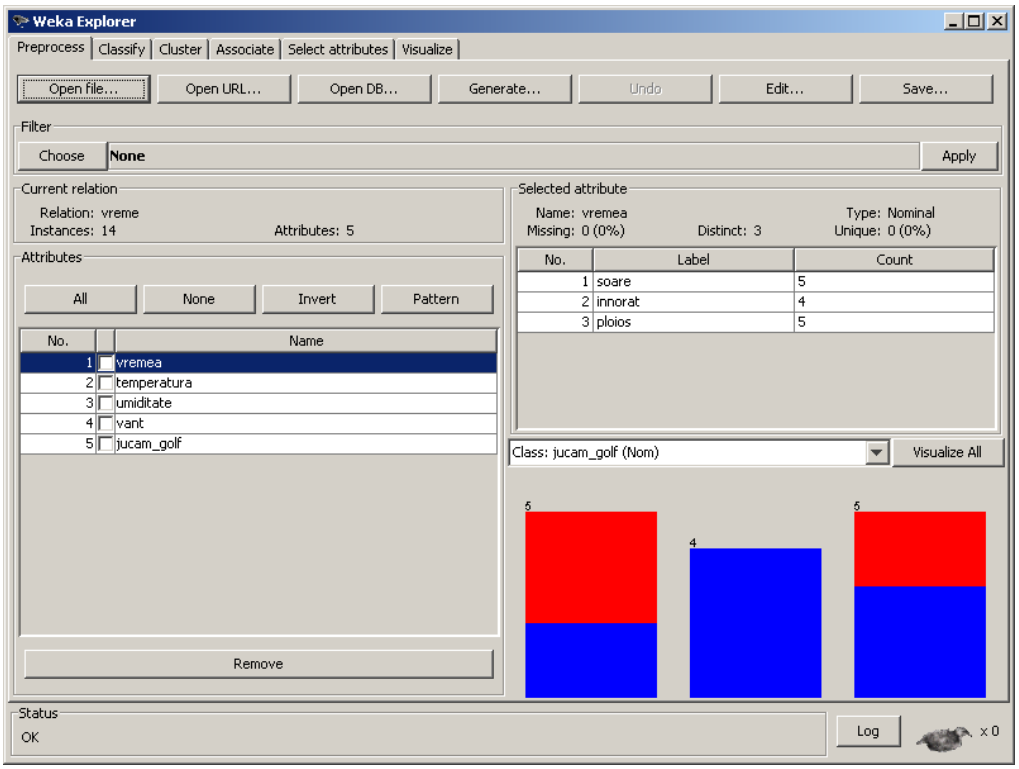

Fig.1.6 - Aplicaţia Explorer

Cu ajutorul comenzii *Open file…* se pot deschide fişierele de date. Odată ce acestea au fost deschise se pot vizualiza atributele din baza de date. Făcând click pe un atribut nominal se poate vizualiza care sunt valorile posibile ale acelui atribut nominal, precum şi numărul de instanţe corespunzătoare fiecărei valori nominale (în exemplul considerat se poate vedea că în total avem 14 instanţe, în cinci din ele atributul vreme a avut valoarea soare, în 4 din ele a avut valoare înnorat, iar 5 a avut valoarea ploios). În cazul în care se selectează un atribut numeric se pot vizualiza automat valoarea minimă a atributului, valoarea maximă, media şi deviaţia standard. În cadranul din partea dreaptă jos este un grafic care foloseşte un număr de culori egal cu numărul de clase, fiecare culoare corespunzând unei clase. Graficul ilustrează cate din instanţe aparţin unei clase şi câte aparţin celeilalte clase.

Datele încărcate pot fi preprocesate, se pot selecta atributele relevante, se pot aplica filtre pentru reducerea zgomotului etc. Se poate folosi comanda Filter şi apoi Apply. Pentru diverse filtre se pot stabili diverşi parametri.

În continuare Tab-ul classify permite selectarea de algoritmi de clasificare şi regresie, menţionarea modului în care se va efectua testarea modelului construit (se poate folosi pentru testare mulţimea de antrenament, sau se poate menţiona că un procent din setul iniţial de date să fie utilizat pentru antrenament şi cealaltă parte pentru testare, etc.). După ce se da comanda start şi începe etapa de antrenare a modelului şi de testare a lui, sunt prezentate rezultatele obţinute.

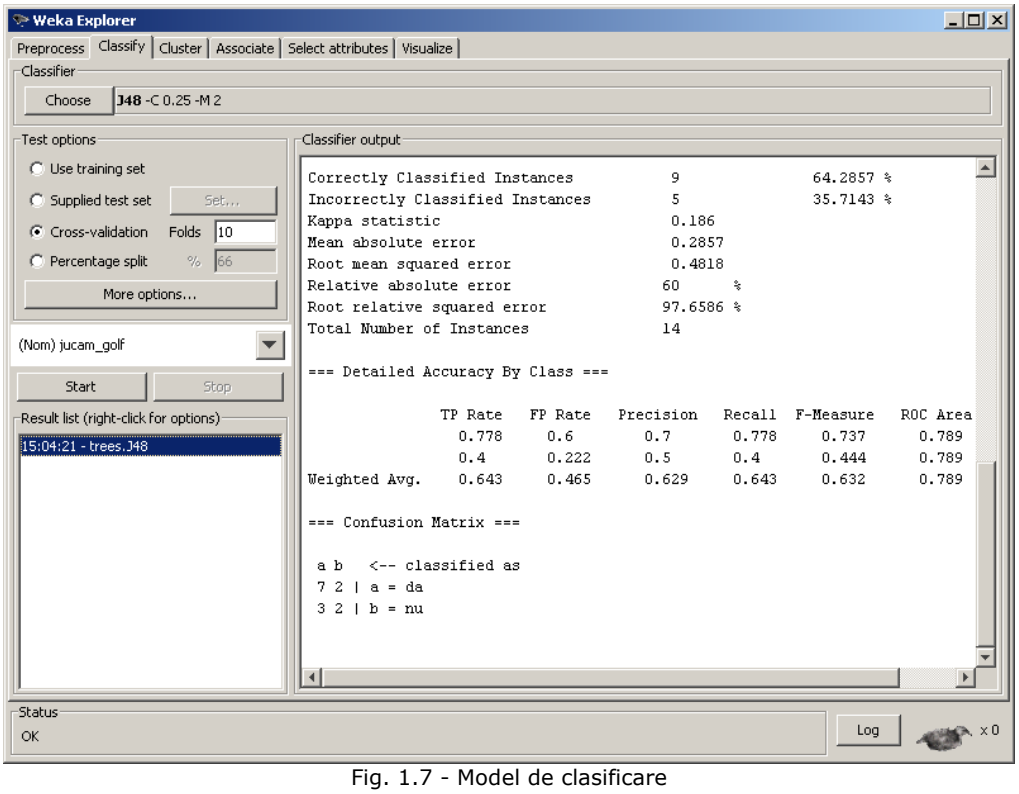

Fig. 1.7 - Model de clasificare

Se poate vedea o exprimare textuală a modelului construit şi alţi parametri care ne informează asupra performanţelor modelului (numărul de instanţe clasificate corect, numărul de instanţe clasificate greşit, matricea de confuzie).

Matricea de confuzie mai este numită şi tabel de contingenţă. Numărul de elemente clasificate corect este egal cu suma elementelor de pe diagonala principală. Elementele clasificate greşit nu se găsesc pe diagonala principală. Dimensiunea matricei depinde de numărul de valori ale atributului clasă.

Alte măsuri utile pentru a compara clasificatorii sunt [BOU08]:

TP (true positive) - proporţia de exemple care au fost clasificate de clasă x, printre toate exemplele care au întradevăr clasa x.

FP (false positive) – proporţia de exemple care au fost clasificate de clasa x, dar de fapt aparţin unei alte clase printre toate exemplele care nu sunt de clasă x.

Precizia este numărul de exemple care au întradevăr clasa x, dintre toate care au fost clasificate ca fiind de clasă x.

În cazul în care se aplică un clasificator de tip arbore de decizie se poate vizualiza arborele de decizie creat:

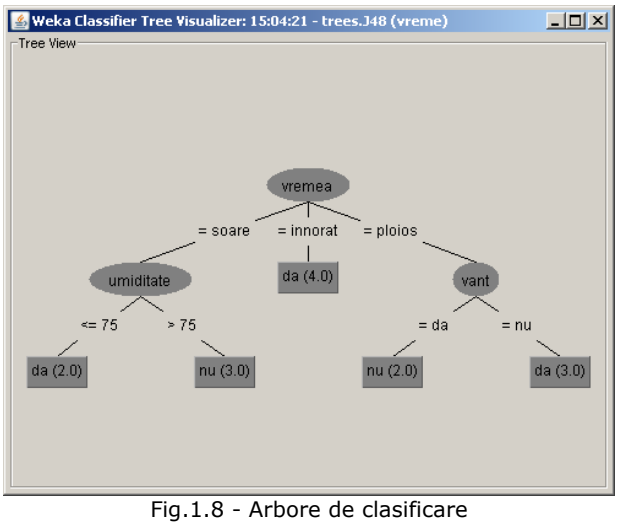

Fig.1.8 - Arbore de clasificare

Aplicaţia *Simple CLI* (*Command Language Interpretor*) este greu de utilizat, întrucât comenzile au foarte mulţi parametri.

Aplicaţia *Experimenter* permite selectarea mai multor fişiere şi a mai multor algoritmi, simultan. În cazul în care dorim să aplicăm mai mulţi algoritmi aceluiaşi fişier sursă (sau mai multor fişiere sursă) putem face acest lucru manual cu explorerul sau automat utilizând *Experimenterul*. *Experimenterul* ne scuteşte de a repeta aceleaşi acţiuni.

Aplicaţia *KnowledgeFlow* reprezintă o alternativă la *Explorer*, care permite construirea de modele în mod grafic. Anumite funcţionalităţi din *Explorer* nu sunt încă implementate în *KnowledgeFlow*.

Tabelul 1.4 prezintă avantajele şi dezavantajele utilizării lui *Weka*. Acesta este uşor de utilizat, este flexibil, dar dacă vrem de exemplu să realizăm predicţii cu un model pe care l-am construit şi l-am testat trebuie să căutăm pe Internet cum se

face asta pentru că interfaţa nu este intuitivă. Totodată dacă vrem să înţelegem cum funcţionează algoritmii, ce procesări fac, ce rezultate intermediare obţin, *Weka* nu ne este de folos, el furnizează doar rezultatul final.

Avantaje	Dezavantaje
Dispune de foarte mulţi algoritmi	Unele lucruri nu sunt intuitive
Uşor de utilizat	Ascunde înţelegerea adevărată
Poate fi extins si particularizat	

Tabel 1.4 - Avantajele şi dezavantajele lui *Weka*

În concluzie *Weka* este un instrument puternic, utilizat pentru a realiza diverse studii de *Data Mining*, şi este folosit la scară largă. El oferă o paletă largă de algoritmi de clasificare, clustering, regresie, precum şi multe instrumente foarte utile în analiza datelor. Fiind Opensource poate fi extins şi personalizat.

1.4.2. R

R-ul oferă facilităţi pentru manipularea datelor, calcule, analiză, trasare de grafice şi dezvoltare de soft. R este un soft şi un limbaj şi este considerat un dialect al limbajului S care a fost creat de AT & T Bell Laboratories [www5]. Cele mai multe din programele S (S-ul fiind un instrument clasic pentru comunitatea statistică) pot fi utilizate direct în R. Putem afirma despre R că este un vehicul pentru dezvoltarea de noi metode de analiză interactivă a datelor. R este un limbaj interpretat şi orientat pe obiecte. Utilizatorul poate realiza acţiuni asupra obiectelor cu ajutorul operatorilor (aritmetici, logici şi de comparare) şi a funcţiilor. R spre deosebire de alte pachete statistice stochează toate rezultatele tot în obiecte. Sintaxa R este simplă, iar instrumentele de machine learning sunt distribuite în pachete [www5]. Tipurile de date din R pot fi numerice (numeric) sau nominale (factor). Câteva exemple de comenzi R sunt următoarele:

Comanda	**Descriere**
setwd("D:/SCOALA/doctorat/statistical aspects of data mining/Data")	Setarea directorului curent
Getwd()	Obţinerea directorului curent
vector1<-c(1,10,12) **vector2<-c(3,5,1)**	Se creează doi vectori, unul iniţializat cu valorile 1,10,12, iar celălalt cu 3,5,1
length(vector)	Returnează numărul de elemente al vectorului
My_data_set<-data.frame(attributeA=vector1,attributeB=vector2)	Crearea unui data set
mean(my_data_set[,1])	Furnizează media elementelor de pe prima linie a data set-ului creat
write.csv(my_data_set,"my_data_set_file.csv")	Salvarea datelor într-un fişier csv
Plot()	Trasarea unui grafic

Tabel 1.5 - Exemple de comenzi în R

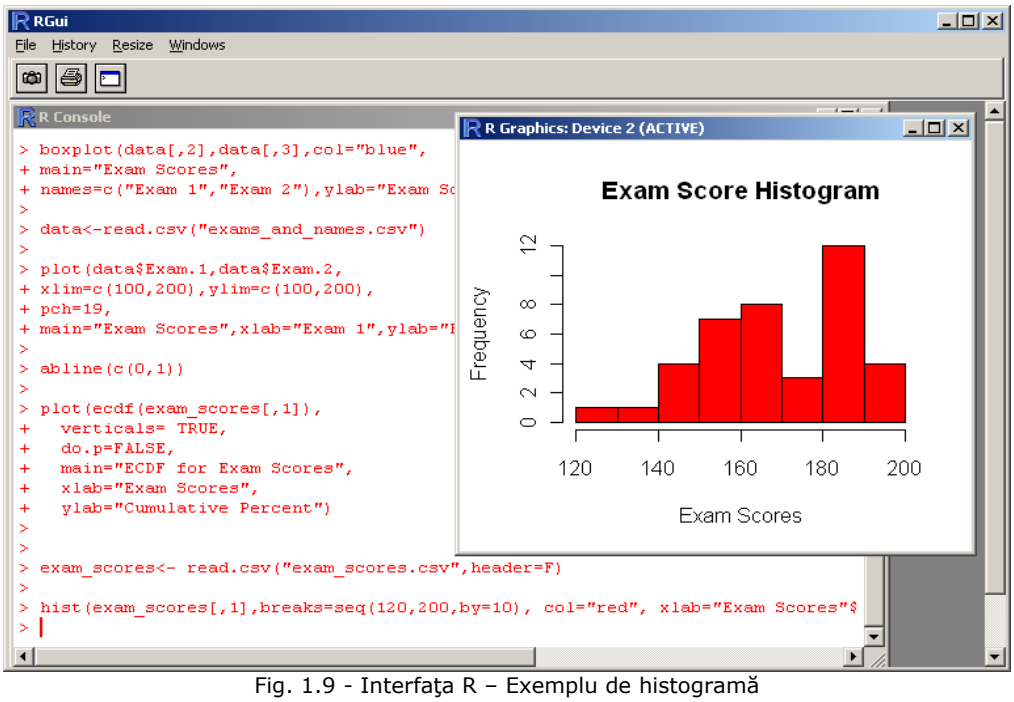

Fig. 1.9 - Interfaţa R – Exemplu de histogramă

Interfaţa R este prezentată în figura 1.9 împreuna cu un exemplu de trasare a unei histograme. În figura 1.10 sunt prezentate alte doua exemple de grafice trasate cu R, în cadrul unui studiu de comparare a rezultatelor la două examene cu scopul determinării care din examene este mai dificil.

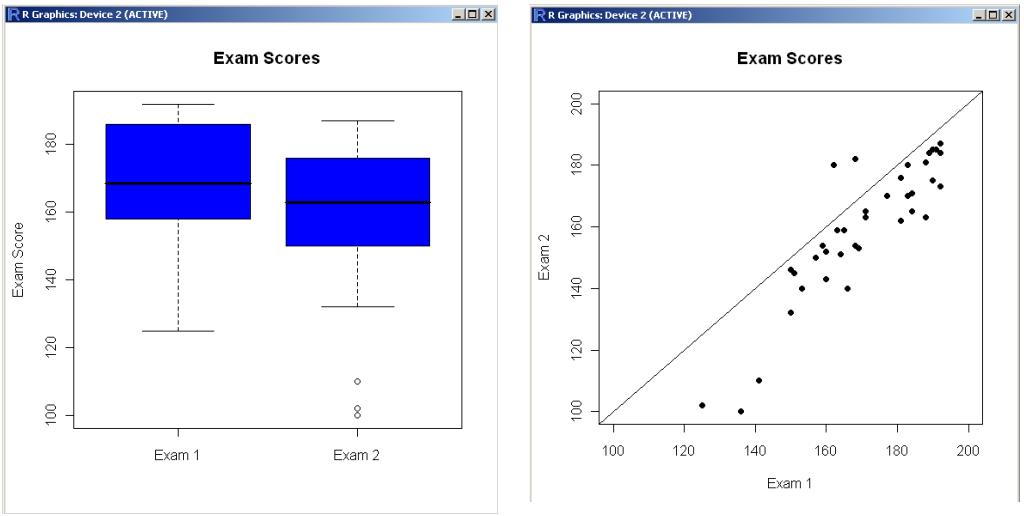

Fig.1.10 - Exemple de grafice trasate cu R

Tabelul 1.6 prezintă câteva avantaje şi dezavantaje ale R-ului. R-ul are o sintaxă simplă, putem să-l programam să facă lucrurile de care avem nevoie, cu ajutorului lui se pot crea grafice spectaculoase. Utilizatorii avansaţi pot să scrie cod C pentru a manipula obiectele R în mod direct. În R sunt implementate multe tehnici statistice şi oferă un bun control al erorilor.

Un dezavantaj al R-ului este legat de pachete care pot sa fie foarte dezorganizate, pot exista mai multe pachete cu implementări ale aceluiaşi algoritm, sau lucruri care ar trebui să se găsească într-un pachet pot să apară în alte pachete [www5]. Interfaţa grafică nu este foarte prietenoasă, iar pentru a scrie şi rula un algoritm, este necesară o documentare cu privire la ce funcţii trebuiesc utilizate şi modul în care trebuie utilizate.

Pentru a face diferitele seturi de instrumente din mediile *Weka* şi R accesibile într-un singur sistem unificat, autorii articolului "Open-Source Machine Learning: R Meets *Weka*" [HOR09] au sugerat un pachet R, numit RWEKA care să interfaţeze funcţionalitatea *Weka* către R.

Avantaje	Dezavantaje
Grafice spectaculoase	Interfaţă grafică nu foarte prietenoasă
Este un mediu în care sunt implementate multe tehnici statistice	Pachete destul de dezordonate
Oferă un control bun al erorilor	Necesită timp pentru utilizare
Poate executa cod C	
Se poate conecta la *Weka*	

Tabel 1.6 - Avantajele şi dezavantajele lui R

1.4.3. Comparaţie între Weka şi R

Atât *R* cât şi *Weka* sunt sisteme Open Source utilizate pentru analiza datelor. Tabelul 1.7 prezintă o comparaţie între aceste două instrumente. Plusul arată că instrumentul corespunzător stă mai bine decât celălalt la capitolul considerat.

	Weka	R
GUI	+	
Algoritmii implementaţi	+	
Graficele trasate		+
Uşurinţa de utilizare	+	
Control		+
Încurajează învăţarea		+
Popularitate	+	

Tabel 1.7 - Comparaţie între *Weka* şi *R*

Interfaţa lui *Weka* este mult mai prietenoasă decât cea a *R*-ului, întrucât în *Weka* deschiderea unui set de date, rularea unui algoritm se face cu ajutorul unor click-uri de mouse pe opţiunile de meniu corespunzătoare, în vreme ce R-ul este un instrument dar şi un limbaj, iar încărcarea datelor şi execuţia unui algoritm nu se poate face fără a apela din cod funcţiile corespunzătoare.

Weka are în plus faţă de R mult mai mulţi algoritmi de clasificare, regresie, grupare şi găsire a regulilor de asociere.

Ambele instrumente permit trasarea unor grafice spectaculoase, dar R-ul, în plus faţă de *Weka,* permite trasarea unor diagrame de tip *boxplot* (vezi figura 1.10 stânga).

Utilizatorii de *Weka* pot rula cu uşurinţă algoritmii, selectând-i din interfaţa grafică, pe când utilizatorii R-ului trebuie să apeleze algoritmii într-un mod mai greoi, din cod.

În *Weka* există lucruri pe care vrem să le facem şi nu putem sa le facem fără să intervenim în codul sau, pe când R-ul este un instrument dar şi un limbaj deci poate fi programat să facă ce doreşte utilizatorul.

Mulţi dintre utilizatorii de *Weka* rulează algoritmii de cele mai multe ori cu parametrii impliciţi, fără sa-şi pună prea multe probleme cu privire la aceştia şi în acest fel ajung să nu ştie ce se întâmplă în profunzime. În R, întrucât pentru apelul oricărei funcţii este necesara scrierea ei şi a parametrilor, asta obligă la cunoaşterea parametrilor dar complică procesul şi îi cere utilizatorului să dedice mult mai mult timp.

Datorită uşurinţei de utilizare şi a paletei largi se algoritmi oferiţi de *Weka*, acesta este mai popular decât R.

Weka este pe primul loc în preferinţele autorului şi de aceea a fost folosit în studiile următoare.

1.5. Seturi de date utilizate pentru analiza performanţelor algoritmilor

University of California Irvine găzduieşte o arhivă numită *UCI Machine Learning Repository* [FRA10] care conţine la ora actuală 211 baze de date reale, donate care sunt utilizate de comunitatea de data mining şi machine learning pentru analiza empirică a comportamentului algoritmilor.

Arhiva a fost creată ca o arhivă *ftp* în anul 1987 de către David Aha şi colegi absolvenţi de la UC Irvine. De atunci a fost folosită de cercetători din toată lumea ca sursă primară de seturi de date. Arhiva a fost citata de mai mult de 1000 de ori, devenind astfel una din cele mai citate ”lucrări” din domeniul calculatoarelor [FRA10] .

În cadrul acestei teze au fost utilizate 10 seturi de date reale obţinute de pe UCI Machine Learning Repository. Seturile de date utilizate sunt cele din tabelul 1.9. Capitolele în care sunt utilizate (o parte sau toate) aceste seturi de date sunt capitolele 3, 4 şi 5. Coloana *Număr atribute* din tabelul 1.8 conţine doar numărul de atribute predictive, nu conţine şi atributul clasă.

Nr	Setul de date	Număr instanţe	Număr atribute	Număr valori atribut clasa
1	Car	1728	6	4
2	Zoo	101	17	7
3	Ljubljana breast cancer	286	9	2
4	Diabetes	768	8	2
5	Cleveland heart disease	303	13	5
6	Iris	150	4	3
7	Mushrooms	8124	22	2
8	Nursurey	12960	8	5
9	Tic	958	9	2
10	Dermatology	336	34	6

Tabel 1.8 – Bazele de date utilizate pentru analiza comportamentului algoritmilor

Seturile de date conţin următoarele informaţii:

- ***Car***, conţine 1728 de instanţe şi 6 atribute categoriale cu privire la evaluările unor maşini. Acestea sunt preţul de cumpărare al maşinii, preţul de întreţinere a maşinii, ambele cu valorile posibile foarte mare, mare, mediu, mic, numărul de uşi (2, 3, 4, 5 sau mai multe), numărul de persoane pe care poate sa le transporte (2, 4, mai multe), dimensiunea portbagajului (mică, medie sau mare), siguranţa (mică, medie, mare), atributul clasă cu valorile posibile (neacceptabilă, acceptabilă, bună, foarte bună).
- ***Zoo*** conţine 101 instanţe şi 17 atribute. Acestea sunt numele animalului care poate lua 100 de valori distincte, numărul de picioare poate lua valori întregi de la 0 la 9, tipul animalului este atributul clasa cu 7 valori posibile: mamifer, pasăre, reptilă, peşte, amfibiu, insectă, nevertebrată, restul atributelor pot lua două valori. Acestea indică dacă animalul are păr, pene, face ouă, dă lapte, poate zbura, poate înota, este prădător, este dinţat, are coloană vertebrală, respiră, este veninos, are aripioare, are coadă, este domestic.
- ***Ljubljana breast cancer*** – Datele au fost furnizate de medici de la Institutul de Oncologie a Centrului Medical Universitar din Ljubljana, Yugoslavia şi conţine 286 de instanţe şi 9 atribute. O parte din acestea sunt vârsta (împărţită în intervale de 10 ani), dacă femeia este la menopauză sau nu, dimensiunea tumorii (12 intervale), gradul de malignitate (3 nivele), sânul (stâng sau drept), careul în care se găseşte tumoarea, dacă sânul este iradiat sau nu. Atributul clasa indică dacă reapariţia sau nu a unor evenimente specifice bolii.
- ***Diabetes*** – are un număr de 768 de instanţe şi 8 atribute. Datele privesc un grup de amerindieni Pima, de lângă Phoenix, Arizona şi sunt furnizate de Institutul Naţional de Diabet şi Boli Digestive şi de Rinichi. Atributele caracterizează femei care au cel puţin 21 de ani şi sunt următoarele: numărul de sarcini, concentraţia de glucoză din plasmă la 2 ore de la

efectuarea unui test oral de toleranţă la glucoză, tensiunea arterială diastolică, grosimea pielii de pe triceps, cantitatea de insulină serică la 2 ore, indicele de masă corporală, atributul clasă care indică dacă rezultatul testului este negativ sau pozitiv.

- ***Cleveland heart disease*** –datele au fost furnizate de Cleveland Clinic Foundation, şi conţin 303 instanţe şi 13 atribute. Câmpul clasă indică dacă pacientul are vreo boală de inimă. Valorile acestui câmp sunt cuprinse între 0 (nici o boala) şi 4 (bolnav grav). Numele şi alte informaţii confidenţiale precum codul numeric personal au fost eliminate din baza de date. Atributele luate în considerare în studiu sunt vârsta, sexul, tipul de durere în piept (cu 4 valori posibile), tensiunea arterială în repaus, colesterolul, cantitatea de zahăr din sânge, rezultatul electrocardiogramei, ritmul cardiac maxim atins, angină pectorală indusă de exerciţiu – cu valorile da sau nu, depresia ST indusă de exerciţiu relativ la odihnă, panta segmentului ST extras, numărul de vase (0-3) colorate la flouroscopie, diagnosticul bolii de inimă (stadiul din punct de vedere angiografic)
- ***Iris*** – Aceasta este probabil cea mai cunoscuta bază de date referită în literatură care vizează descoperirea şabloanelor. Dataset-ul conţine 150 de instanţe şi 3 clase, de fiecare din aceste clase aparţin câte 50 de instanţe. Atributele sunt numerice şi conţin informaţii cu privire la 3 specii de flori IRIS, Iris Setosa, Iris Versicolour şi Iris Virginica. Informaţiile care se memorează sunt lungimea şi lăţimea sepalei şi lungimea şi lăţimea petalei. Aceste 4 atribute sunt numerice.
- ***Mushrooms*** conţine informaţii despre ciuperci. Atributul clasă indică dacă acestea sunt comestibile sau otrăvitoare. Setul de date are 8124 de instanţe şi 22 de atribute precum forma şi culoarea pălăriei, mirosul etc.
- ***Nursurey*** conţine informaţii despre evaluarea aplicaţiilor depuse de părinţi la grădiniţe. Dataset-ul are 12960 de instanţe şi 8 atribute categoriale. Informaţiile care se reţin privesc ocupaţia părinţilor, creşa copiilor, forma familiei, numărul de copii, condiţiile de locuit, veniturile financiare ale familiei, imaginea socială şi de sănătate a familiei, rezultatul evaluării (atributul clasă).
- ***Tic-Tac-Toe*** – este o bază de date pentru cunoscutul joc X şi 0. Baza de date codifică toate situaţiile posibile la sfârşitul jocului X şi 0 în care X are mutarea de început. Setul de date are 958 de instanţe şi 9 atribute, fiecare atribut corespunde unui pătrat de pe tabla de X şi 0. Valorile posibile pentru fiecare pătrat sunt fie X, fie 0, fie B (jocul s-a încheiat şi pătratul a rămas liber). Clasa are valorile posibile pozitiv şi negativ (pozitiv înseamnă victorie pentru X).
- ***Dermatology*** – baza de date conţine 34 de atribute, din care unul este întreg (vârsta) şi a fost discretizat iar restul sunt nominale. Clasa reprezintă diagnosticul diferenţial al bolilor *eritematoase, scuamoase* care este o adevărată problemă în dermatologie. Aceste boli împărtăşesc toate caracteristicile clinice de eritem şi scalare, cu diferenţe foarte mici. Bolile din acest grup (valorile clasei) sunt psoriazis, dermatită seboreică, lichen plan, pitiriazisul rozat, dermatită cronică, şi pitiriazis rubra pilaris. De obicei este necesară o biopsie pentru a diagnostica aceste boli, dar din păcate ele împart şi multe caracteristici histopatologice de asemenea. Altă dificultate pentru diagnosticul diferenţiat este aceea că o boală poate arata caracteristicile altei boli în stadiul de început şi poate avea caracteristici

specifice în stadiile următoare. Pacienţii au fost întâi evaluaţi din punct de vedere clinic, reţinându-se 12 caracteristici. Apoi au fost luate mostre de piele pentru evaluarea a 22 de caracteristici histopatologice. Valorile acestor 22 de caracteristici s-au obţinut prin analiza mostrelor de piele la microscop. În setul de date construite pentru acest domeniu, caracteristica privind istoria familiei are valoarea 1 în cazul în care oricare dintre aceste boli au fost observate în familie, şi 0 altfel. Fiecare caracteristică clinică şi histopatologica, a primit o valoare în intervalul de la 0 la 3. 0 indică faptul că caracteristica nu a fost prezentă, 3 indică cea mai mare prezenţă posibilă, şi 1, 2 indică valorile relative intermediare.

1.6. Structura tezei

Capitolul 1 - prezintă definiţii şi terminologie, etapele procesului de data mining, domenii în care tehnicile de data mining se aplică cu succes, algoritmi de referinţă pentru clasificare datelor, seturi de date utilizate pentru evaluarea performanţelor algoritmilor şi două din cele mai populare instrumente utilizate la realizarea de studii de data mining (*Weka* şi *R*). Cele două instrumente au fost comparate, şi prezentate plusurile si minusurile fiecăruia.

Capitolul 2 - prezintă o sinteză privind tehnicile de preprocesare a datelor, aspecte legate de preprocesarea datelor cu *Weka* şi aplicaţia concepută şi implementată de autor (Arff Convertor), care uşurează procesul de preluare a datelor din diferite baze de date (MySql, Oracle, Microsoft Access, Microsoft Excel, Microsoft FoxPro) şi încărcare a lor în *Weka*, etapă care ţine de preprocesarea datelor. Pentru validarea soluţiei propusă s-a realizat o analiză a datelor unor studenţi de la facultatea de Automatică şi Calculatoare, preluate din aplicaţia GISC cu care este asigurată gestiunea şcolarităţii la această facultate.

Capitolul 3 – prezintă aspecte legate de clasificarea datelor cu ajutorul algoritmului Ant Colony Optimization.

- A fost realizată o prezentare detaliată a algoritmului, bazată atât pe documentaţia existentă cât şi pe studiul codului algoritmului.
- Au fost studiaţi parametrii algoritmului rulând 64 de configuraţii pe 10 seturi de date reale (prezentate în capitolul 1), obţinute de pe *UCI Machine Learning Repository.* Rezultatele obţinute au fost analizate atât vizual (comparativ) cât şi matematic. Analiza matematică s-a realizat construind modele de regresie pe baza unor seturi de date noi construite pe baza testelor realizate.
- A fost realizată o sinteză a îmbunătăţirilor aduse algoritmului din anul 2001 şi până în prezent
- Au fost aduse 4 îmbunătăţiri algoritmului care au fost implementate în aplicaţia open source *Ant Miner*. Aplicaţia care conţine algoritmul îmbunătăţit a fost numită *Ant-r-Miner*. S-a demonstrat că îmbunătăţirile aduse conduc la o mai bună performanţă utilizând seturi de date reale.

Capitolul 4 – prezintă aspecte legate de structura generală a unui algoritm genetic, aspecte legate de algoritmii genetici care au fost propuşi în literatură pentru clasificarea datelor şi propune un nou algoritm genetic în acest scop. Algoritmul

genetic propus a fost numir AGR şi implementat în *Weka*, contribuind în acest fel la îmbunătăţirea tehnologiei de clasificare a datelor. Algoritmul genetic propus diferă de alţi algoritmi propuşi în literatură pentru clasificarea datelor prin funcţia fitness utilizată, facilităţile de recalculare a clasei în urma încrucişării, facilităţile de execuţie repetată până se ajunge la un procent de acoperire dorit, facilităţile de utilizare a tuturor regulilor descoperite sau doar a celei mai bune reguli descoperite de fiecare iteraţie, facilităţile de adăugare a unei reguli implicite listei regulilor descoperite, etc. Analiza comportamentului algoritmului propus la diferite valori ale parametrilor de intrare s-a realizat rulând 14 seturi de parametri pe 5 seturi de date. Seturile de date au fost obţinute de pe *UCI Machine Learning Repository* şi prezentate în introducere. Analiza parametrilor s-a materializat într-o listă de recomandări privind alegerea acestor parametri. Rezultatele obţinute de algoritm pe 5 seturi de date au fost comparate cu cele obţinute de algoritmii *Naive Bayes, k-Nearest Neighbor şi J4.8* şi au fost favorabile, în ceea ce priveşte acurateţea predicţiei, algoritmului genetic propus.

Capitolul 5 – Capitolul prezintă modul în care se pot realiza predicţii cu *Weka*, soluţiile care au fost găsite pentru a îmbunătăţi acest proces şi îmbunătăţirile care au fost aduse interfeţei de clasificare a datelor. Pentru a îmbunătăţii procesul de realizare al predicţiilor, au fost realizate modificări în secţiunea *Classify* a *Explorer*-ului şi acestea constă în transformarea acestei interfeţe într-o interfaţă dinamică în funcţie de setul de date analizat, interfaţă cu care se pot realiza predicţii într-un mod foarte simplu şi intuitiv. Alte modificări realizate au transformat această interfaţă într-o interfaţă prietenoasă, uşor accesibilă chiar şi începătorilor în *Weka*. Extensiile concepute şi realizate în *Weka* contribuie la îmbunătăţirea tehnologiei de clasificare a datelor.

Capitolul 6 – Capitolul prezintă o analiză a datelor legate de naşterile realizate în anul 2010 la *Clinica de Obstetrică - Ginecologie Bega, Timişoara.* Pentru realizarea studiului au fost disponibile date legate de 2326 de naşteri. Studiul a fost realizat cu ajutorul cunoştinţelor şi algoritmilor, respectiv instrumentelor propuse în primele 5 capitole ale tezei. Analiza datelor s-a realizat prin următoarele etape:

- Datele disponibile au fost întâi preprocesate, în Microsoft Excel prin analiza fiecărei coloane, eliminarea instanţelor cu valori lipsă sau valori introduse greşit, corectarea valorilor introduse greşit, eliminarea atributelor neimportante pentru analiză, crearea unor noi atribute, etc (vezi capitolul 2). Transformarea datelor din format *xls* în formatul *ARFF*, specific *Weka* cu ajutorul aplicaţiei *Arff Convertor*, concepută, implementată şi prezentată de autor în cadrul capitolului 2.
- Preprocesarea datelor cu ajutorul lui *Weka*, prin discreditarea atributelor numerice (vezi capitolul 2)
- Analiza statistică la nivel de atribut, care a condus la obţinerea unor informaţii noi şi interesante privind naşterile
- Dezvoltarea unor modele de clasificare cu algoritmii *Naive Bayes, J48, k-Nearest-Neighbour* din *Weka*, algoritmi care au fost prezentaţi în introducere
- Dezvoltarea unor modele de clasificare cu algoritmii *Ant-Miner şi Ant-r-Miner*, prezentaţi în capitolul 3
- Dezvoltarea unui model de clasificare care poate estima cu o precizie de aprox 85 % unul din cele 5 intervale în care indicele apgar al nou-născutului

va lua valoare. Modelul a fost construit cu ajutorul algoritmului AGR, care a fost conceput, implementat şi prezentat de autor în cadrul capitolului 4

- Realizarea de predicţii cu ajutorul noii interfeţe dinamice a lui *Weka,* concepută şi prezentată de autor în cadrul capitolului 5. Setul de reguli descoperit a fost transpus în format XML cu ajutorul unei aplicaţii concepute şi implementate şi prezentate de autor în capitolul 6 pentru a servi ca intrare unui sistem care le transpune în *ARDEN Syntax* [HEE05].

Capitolul 7 - prezintă concluziile şi contribuţiile pe care autorul le-a adus la îmbunătăţirea tehnologiei de clasificare a datelor.

1.7. Concluzii

În cadrul acestui capitol a fost realizată o sinteză amplă, folosind 47 de surse bibliografice, a celor mai importante aspecte legate de domeniul Data Mining. Au fost prezentate definiţii şi terminologia folosită în domeniu, etapele procesului de data minig, domenii în care tehnicile de data mining se aplică cu succes, aspecte legate de clasificarea datelor şi de algoritmii folosiţi pentru a construi modele clasificatoare, aspecte legate de instrumentele software utilizate în studii de data mining şi de seturile de date utilizate pentru testarea şi validarea algoritmilor. Structura tezei a fost de asemenea prezentată.

Contribuţiile aduse de către autor la cunoaşterea în domeniu, prin intermediul acestui capitol, sunt prezentate la nivel de subcapitol în cele ce urmează:

a) În subcapitolul 1.1, a fost conturat domeniul, subliniind importanţa lui, au fost prezentaţi succint termenii existenţi, un număr mare de definiţii existente în domeniu, a fost introdusă o nouă definiţie, au fost prezentate original etapele procesului de Data Mining, pe baza experienţei practice a autorului
b) În subcapitolul 1.2 – a fost realizată o sinteză amplă asupra aplicării tehnicilor de data mining în domeniile e-commerce, educaţie, medicină, combaterea terorismului. De asemenea a fost prezentată o sinteză a sistemelor de *data mining* utilizate de agenţiile federale din SUA cu scopul de sublinia puterea şi importanţa tehnologiei *data mining*, precum şi nivelul de vârf la care s-a ajuns cu utilizarea ei.
c) În subcapitolul 1.3 a fost prezentate clar, succint aspecte generale legate de clasificarea datelor, precum şi principiile de bază privind funcţionarea a trei algoritmi de clasificare de referinţă (*Naive Bayes, k-Nearest Neighbor, C4.5*).
d) În subcapitolul 1.4 au fost prezentate pe scurt aspecte legate de două dintre cele mai populare instrumente, la nivel mondial, utilizate pentru a realiza studii de data mining (*Weka* şi R). Au fost analizate avantajele şi dezavantajele pe care le are fiecare instrument şi a fost realizată o comparaţie între cele două instrumente.
e) În subcapitolul 1.5 au fost prezentate aspecte legate de cele 10 seturi de date donate, reale, obţinute de pe UCI Machine Learning Repository, care au fost utilizate pe parcursul tezei, în capitolele 3,4 şi 5 pentru a testa şi valida algoritmii propuşi şi îmbunătăţirile tehnologice aduse.

2. Preprocesarea datelor

2.1. Noţiuni introductive

Preprocesarea datelor este o etapă deosebit de importantă a procesului de *data mining*, practica industrială arătând că mai mult de 80 % din procesul de *data mining* se concentrează pe pregătirea datelor [ZHA05].

Datele brute pot conţine valori lipsă, date introduse greşit, date expirate, date duplicat, valori aberante, etc. În acest sens se impune curăţarea datelor brute, prin estimarea valorilor lipsă, prin eliminarea datelor duplicat sau expirate, prin eliminarea sau corectarea datelor introduse greşit, prin utilizarea unor parametri statistici toleranţi la valori extreme, etc (vezi figura 2.1).

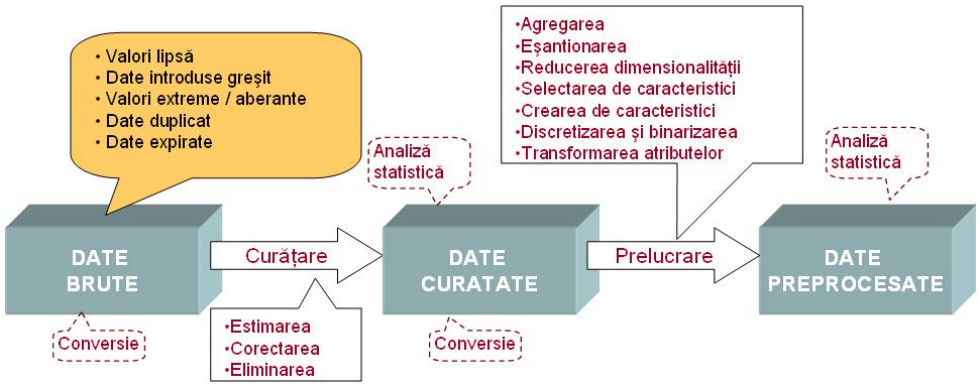

Fig. 2.1 – Preprocesarea datelor

Curăţarea datelor nu este suficientă pentru ca acestea să poată fi explorate cu anumiţi algoritmi de data mining. De foarte multe ori este necesară reducerea volumului datelor, pentru ca acestea să poată fi explorate cu algoritmi de data mining care sunt lenţi. Reducerea volumului datelor se poate face reducând numărul de instanţe (de exemplu prin agregare şi eşantionare) sau reducând numărul atributelor (prin tehnici de reducere a dimensionalităţii sau selectarea de submulţimi de caracteristici).

Datele iniţiale pot fi nepotrivite cu anumiţi algoritmi de data mining şi în acest sens se impune o transformare a acestora prin tehnici precum discretizarea, binarizarea, crearea caracteristicilor etc.

Instrumentele de data mining dispun de mulţi algoritmi de preprocesare a datelor care permit agregarea, eşantionarea, reducerea dimensionalităţii, etc. Pentru a putea aplica aceşti algoritmi asupra datelor ele trebuiesc de multe ori convertite în formatul acceptat de instrumentul software cu care se realizează studiul de data

mining. Conversia este tot o etapă a preprocesării şi se poate aplica fie asupra datelor brute fie asupra datelor curăţate.

Versiunea 3.6 a lui *Weka* dispune de 70 de algoritmi de preprocesare care permit discretizarea atributelor numerice, transformarea atributelor nominale în atribute binare, eşantionarea datelor, selectarea atributelor de interes pentru studiu, unirea a doua valori a unui atribut nominal într-una singură, analiza componentelor principale, etc. Fişierele de date cu care *Weka* lucrează sunt în format *ARFF*, dar *Weka* permite şi încărcarea datelor din diferite baze de date utilizând fie tehnologia *JDBC* fie tehnologiile *JDBC* şi *ODBC* împreună, dar acest proces este relativ complicat pentru că necesită realizarea unor configurări în *Weka* şi în sistemul de operare, precum şi cunoaşterea limbajului *SQL*. Pentru a simplifica acest proces a fost concepută şi dezvoltată de autor o aplicaţie numită *Arff Convertor* care permite conectarea cu uşurinţă la diferite tipuri de baze de date, preluarea datelor de interes din una sau mai multe tabele relaţionate şi exportarea lor în format *ARFF*, fără să fie necesară realizarea de configurări în Windows sau cunoaşterea limbajului SQL. Aplicaţia concepută şi dezvoltată simplifică procesul de preprocesare a datelor şi contribuie în acest fel la îmbunătăţirea tehnologiei de preprocesare a datelor cu *Weka*.

Capitolul prezintă o sinteză amplă şi riguroasă a principalelor tehnici de preprocesare a datelor. De asemenea este prezentată aplicaţia *Arff Convertor* şi pentru a demonstra utilitatea ei un studiu de caz realizat cu *Arff Convertor* şi cu *Weka* pe datele studenţilor din anul universitar 2008-2009 de la Facultatea de Automatică şi Calculatoare a Universităţii "Politehnica" din Timişoara.

2.2. Tehnici de preprocesare a datelor

2.2.1. Agregarea

Agregarea este procesul de combinare a două sau mai multe obiecte (instanţe) într-un singur obiect (instanţă), pe baza caracteristicilor comune ale acestora. Presupunând că dispunem de o bază de date cu vânzările realizate în fiecare zi, de mai multe magazine din locaţii diferite pe parcursul unui an de zile, un exemplu de agregare ar fi înlocuirea tuturor tranzacţiilor corespunzătoare unui magazin cu o singură tranzacţie, care să reprezinte totalul vânzărilor realizate de magazinul respectiv pe parcursul unui an de zile. În acest exemplu agregarea se realizează prin însumare, în alte cazuri se poate realiza prin medie. Un alt exemplu de agregare ar fi reducerea datelor de la 365 de zile la 12 luni.

Agregarea oferă câteva avantaje:

- seturile de date mai mici care rezultă în urma agregării necesită pentru analiză mai puţină memorie şi mai puţin timp de procesare, aşadar se pot folosi algoritmi mai consumatori de timp
- Agregarea permite o privire de sus asupra datelor în locul unei priviri de jos
- Comportamentul grupurilor de obiecte este mai stabil decât cel al indivizilor.

Un dezavantaj al agregării este posibila pierdere a amănuntelor interesante [TAN05]. În cazul în care se face o agregare la nivel de luni ale anului nu se mai pot determina zilele cu vânzările cele mai mari.

2.2.2. Eşantionarea

Eşantionarea este necesară pentru că de multe ori procesarea întregului set de date este mare consumatoare de timp.

În unele cazuri utilizarea unui algoritm de eşantionare poate reduce numărul instanţelor până în punctul în care se poate utiliza pentru analiza datelor un algoritm mai bun dar mai mare consumator de timp.

Eşantionul ales trebuie să fie reprezentativ. Un eşantion este reprezentativ dacă are aceleaşi proprietăţi ca şi setul de date original. Dacă proprietatea de interes este media obiectelor se va alege un eşantion la care obiectele au aceeaşi medie ca şi cea a setului de date original [TAN05].

Exemplul cel mai familiar de eşantionare ne este oferit de sondajele de opinie (cu scop
de informare politică sau socială) în care doar o proporţie foarte mică a populaţiei (câteva mii din câteva milioane) este intervievată.

Există mai multe tehnici de eşantionare, în continuare vor fi prezentate pe scurt cele mai cunoscute şi anume:

- eşantionarea aleatoare
- eşantionarea prin stratificare
- eşantionarea adaptivă

Cea mai simplă tehnică de eşantionare este **eşantionarea aleatoare (simple random sampling)**, în cadrul acestei tehnici toate instanţele din populaţie au probabilităţi de selecţie egale. Această tehnică poate fi implementată în două moduri:

- eşantionarea fără înlocuire (o instanţă aleasă în mod întâmplător este lăsată în continuare în populaţie putând ca apoi sa fie aleasă din nou)
- eşantionarea cu înlocuire (o instanţă aleasă în mod întâmplător este eliminată din populaţie ca sa nu mai poată fi aleasă din nou)

Când populaţia este formată din diferite tipuri de obiecte şi fiecare tip de obiect apare de un anumit număr de ori, atunci prin tehnica descrisă mai sus este posibil ca obiectele care au o frecvenţă scăzută să nu fie reprezentate adecvat (unele tipuri de obiecte pot să nu fie selectate deloc). Acest aspect poate conduce la probleme atunci când este necesară o reprezentare corespunzătoare a tuturor tipurilor de obiecte. De exemplu când se construiesc modele de clasificare, o cerinţă critică este ca clasele rare să fie reprezentate în mod adecvat în eşantion. De aceea este necesară o schemă de eşantionare care este capabilă să se adapteze la diferite frecvenţe ale item-urilor de interes. O astfel de tehnică este **eşantionarea prin stratificare**. Varianta cea mai simplă a acestei tehnici este extragerea unui număr egal de obiecte din fiecare grup, chiar dacă grupurile au numere de obiecte diferite. O altă variantă extrage din fiecare grup un număr de obiecte proporţional cu dimensiunea grupului [TAN05].

Odată selectată tehnica de eşantionare trebuie aleasă dimensiunea eşantionului astfel încât să nu se piardă informaţie relevantă dar nici să nu avem un eşantion prea mare. Determinarea dimensiunii potrivite a eşantionului este de multe ori dificilă aşa că se folosesc tehnici de **eşantionare adaptivă**.

Eşantionarea adaptivă (progresivă) începe cu un eşantion mic pe care îl tot măreşte până se obţine un eşantion suficient de mare. Eşantionarea adaptivă

elimină nevoia de a determina iniţial dimensiunea eşantionului, însă necesită o metodă de a determina dacă eşantionul este suficient de mare. Dacă se utilizează un eşantion adaptiv pentru a învăţa un model predictiv, deşi acurateţea modelului predictiv creşte odată cu creşterea volumului eşantionului, la un moment dat, câştigul de acurateţe obţinut prin creşterea dimensiunii eşantionului este neglijabil. În acel moment se poate opri creşterea volumului eşantionului.

2.2.3. Reducerea dimensionalităţii

Termenul "Reducerea dimensionalităţii" este adeseori rezervat pentru acele tehnici care reduc dimensionalitatea prin faptul că creează noi atribute care sunt combinaţii ale vechilor atribute. Dimensionalitatea poate fi redusă prin tehnici de algebră liniară cum sunt Analiza Componentelor Principale (**Principal Component Analysis**), Descompunerea Valorilor Singulare (**Singular Values Decomposition**), etc.

Avantajele reducerii dimensionalităţii sunt:

- mulţi algoritmi de *data mining* lucrează mai bine dacă dimensionalitatea este redusă
- modelul rezultat poate să fie mai uşor înţeles dacă sunt mai puţine atribute implicate
- reducerea dimensionalităţii poate să elimine atributele nerelevante şi să reducă zgomotul
- datele pot să fie mai uşor vizualizate

Creşterea numărului de atribute al unui set de date, determină necesitatea creşterii numărului de instanţe în mod exponenţial, dacă dorim să se păstreze un anumit nivel de acurateţe a datelor. Acest fenomen este cunoscut sub numele de "Blestemul Dimensionalităţii".

Blestemul dimensionalităţii este problema cauzată de creşterea exponenţială în volum, asociată cu adăugarea unor extra-dimensiuni unui spaţiu matematic [VER05].

Pentru a ilustra fenomenul "curse dimensionality", se consideră că spaţiul complet al probabilităţii pentru o variabilă este reprezentat de intervalul unitate (0,1) şi se consideră generarea a 10 instanţe în cadrul acestui interval. Fiecare instanţă va reprezenta 10 % din spaţiul probabilităţii (în medie). Apoi considerăm o a doua variabilă definită pe un alt interval ortogonal (0,1), de asemenea fiind reprezentată de 10 instanţe. Se vor produce 10 puncte în planul definit de liniile ortogonale x1 şi x2, reprezentând un nou spaţiu de probabilitate. Dar noul spaţiu are 10 * 10 =100 de unităţi, aşa că fiecare dintre aceste 10 puncte reprezintă doar 1 % din spaţiul de probabilitate. Ar fi necesare 100 de puncte pentru ca fiecare punct să reprezinte acelaşi 10 % din spaţiul de probabilitate care era reprezentat de 10 puncte în spaţiul cu o singură dimensiune .

Acest lucru este ilustrat în figura 2.2 în care cele 10 puncte nu sunt generate aleator pentru a ilustra acoperirea diminuată. Numărul de puncte ar trebui să crească exponenţial pentru a menţine o anumită acurateţe.

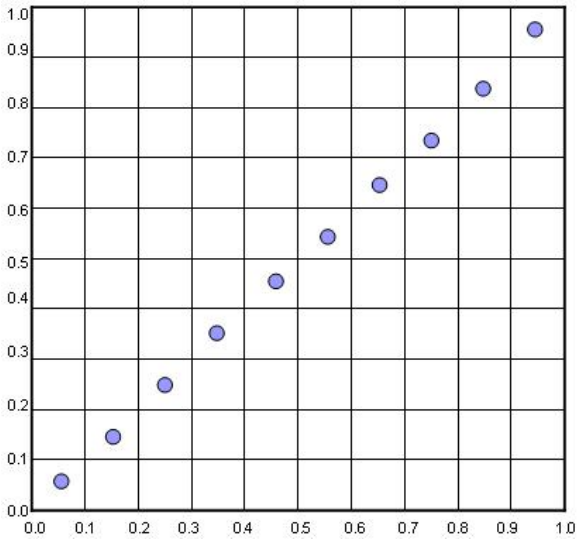

Fig. 2.2 – puncte în spaţiul bidimensional

Tehnicile de bază folosite în reducerea dimensionalităţii sunt următoarele:

- Analiza componentelor principale (*PCA*)
- Descompunerea valorilor singulare (*SVD*)
- Analiza factorială
- Încapsularea locală lineară (Locally linear embedding - *LLE*)
- Scalarea multidimensională (Multidimensional Scaling - *MDS*)

Analiza componentelor principale

Analiza componentelor principale (PCA) este o tehnică de algebră liniară, pentru atribute continue, care găseşte noi atribute (componente principale) care sunt :

- combinaţii liniare ale vechilor atribute
- ortogonale (perpendiculare) unele pe altele
- captează maximul de variaţie din date

Analiza componentelor principale caută să explice structura corelaţiilor unui set de variabile (atribute), folosind un set mai mic de combinaţii liniare a acestor variabile. Aceste combinaţii liniare sunt numite componente. Variabilitatea totală a unui set de date, produsă de un set complet de n variabile, poate fi adesea surprinsă printr-un set mai mic de k combinaţii liniare a acestor variabile, ceea ce înseamnă că în cele k componente există aproape la fel de multă informaţie ca şi în cele n variabile iniţiale. Dacă se doreşte, analiza poate apoi să înlocuiască cele n variabile originale cu cele $k < n$ componente, în aşa fel încât setul de date să fie format din m instanţe cu k componente în loc de m instanţe cu n variabile [LAR06].

Scopul Analizei Componentelor Principale este de a găsi noi atribute care captează mai bine variabilitatea datelor decât cele vechi [JOL02]. Mai explicit, prima dimensiune (atribut) este aleasă pentru a capta o variabilitate cât mai mare. A doua dimensiune este ortogonală faţă de prima şi captează cât de mult poate din variabilitatea rămasă, ş.a.m.d.

PCA are câteva caracteristici:

- tinde sa identifice şabloanele cele mai puternice din date. De aceea ea poate fi utilizată ca o tehnică de găsire a şabloanelor.
- adeseori cel mai mult din variabilitatea datelor poate fi captat printr-o mică parte din setul total de dimensiuni. Ca un rezultat, reducerea dimensionalităţii utilizând PCA, poate determina date de dimensiuni relativ mici şi este posibilă aplicarea tehnicilor care nu funcţionează bine cu date de dimensiuni mari.
- zgomotul din date este mai slab decât tiparele, prin urmare reducere dimensionalităţii poate sa reducă mult din zgomot [TAN05].

Fiind dată o matrice *D* cu *m* linii şi *n* coloane (*m* linii reprezintă obiectele sau instanţele şi *n* coloane reprezintă atributele) matricea de covarianţă a matricei D se notează cu S şi este formată din elementele [LAR06]:

$$s_{ij} = \mathrm{cov}\,arianta(d_{*i}, d_{*j}) = \frac{\sum_{k=1}^{m}(d_{ki} - \mu_i)(d_{kj} - \mu_j)}{m} \qquad (2.1)$$

În cuvinte s_{ij} este covarianţă atributelor *i* şi *j*. μ_i este valoarea medie a atributului *i*. Covarianţa a două atribute este măsură a cât de mult două atribute variază împreună. Valoarea pozitivă a unei covarianţe indică că atunci când o variabilă creşte şi cealaltă tinde să crească. Valoarea negativă a unei covarianţe indică că atunci când o variabilă creşte cealaltă tinde să scadă. Dacă o covarianţă are valoarea 0 atunci cele două variabile sunt independente[LAR06]. Dacă i=j atributele sunt aceleaşi şi atunci covarianţa este de fapt varianţa atributului. Dacă matricea D este preprocesată aşa încât media fiecărui atribut este 0, atunci $S = D^T D$. Scopul PCA este de a găsi o transformare a datelor care satisface următoarele proprietăţi [JOL02]:

1. Fiecare pereche de noi atribute are covarianţă 0 (pentru atribute distincte)
2. Atributele sunt ordonate în acord cu cât de mult surprinde fiecare atribut din varianţa datelor
3. Primul atribut captează cât de mult se poate din varianţa datelor
4. Subiect al cerinţelor de ortogonalitate, fiecare atribut succesiv captează cât de mult se poate din varianţa rămasă [TAN05]

O transformare a datelor care are aceste proprietăţi poate să fie obţinută analizând *valorile proprii (eigenvalues)* ale *matricei de covarianţă*.

Valorile proprii şi *vectorii proprii* sunt definiţi doar pentru matrice pătratice. Matricea de covarianţă exprimă varianţa atributelor şi este întotdeauna o matrice pătratică.

Valorile proprii măsoară cantitatea de varianţă „explicată” de fiecare componentă principală şi se calculează după cum este prezentat în continuare.

Fie S o matrice pătratică cu dimensiunea de *n x n*. Numărul λ este o valoare proprie a matricei S dacă există un vector *v* diferit de 0 astfel încât:

$$Sv = \lambda v \qquad (2.2)$$

În acest caz vectorul *v* este numit vector propriu al matricei S corespunzător valorii proprii λ. Relaţia (2.2) poate fi scrisă sub forma :

$$(S - \lambda I_n)v = 0 \tag{2.3}$$

Unde I_n este matricea unitate de dimensiune *n x n*. Dacă matricea $S - \lambda I_n$ este inversabilă, se poate înmulţi relaţia (2.3) cu inversa şi se obţine:

$$(S - \lambda I_n)^{-1}(S - \lambda I_n)v = (S - \lambda I_n)^{-1}0 \Rightarrow v = 0 \tag{2.4}$$

Dar noi căutăm un vector v diferit de 0, deci este necesar ca matricea $S - \lambda I_n$ să fie neinversabilă, adică să aibă determinantul egal cu 0. Calculând determinantul matricei $S - \lambda I_n$ se obţine un număr maxim de n valori proprii $\lambda_1, \lambda_2, ..., \lambda_n$. Pentru a determina un vector propriu $v = \begin{bmatrix} v_1 \\ v_2 \\ ... \\ v_n \end{bmatrix}$ diferit de 0 care corespunde unei valori proprii λ, se rezolvă sistemul de ecuaţii liniare din relaţia (2.3).

Valorile proprii ale matricelor de covarianţă S sunt nenegative şi pot să fie ordonate sub forma $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_{n-1} \geq \lambda_n$.Suma valorilor proprii este egală cu numărul variabilelor iniţiale. Fie $U = [u_1, u_2, ...u_n]$ matricea vectorilor proprii ai matricei de covarianţă S. Aceşti vectori proprii sunt ordonaţi în aşa fel încât vectorul de pe poziţia *i* corespunde valorii proprii de pe poziţia *i* (valorile proprii la rândul lor sunt ordonate descrescător). Matricea D a fost preprocesată în aşa fel încât media fiecărui atribut este 0.

În continuare se calculează matricea $D' = DU$. Fiecare nou atribut (din D') este o combinaţie liniară a atributelor originale. În mod specific, ponderile combinaţiei liniare pentru atributul de pe poziţia *i* sunt componentele vectorului propriu de pe poziţia *i*. Varianţa atributului de pe poziţia *i* este λ_i. Suma varianţei atributelor originale este egală cu suma varianţelor noilor atribute. Noile atribute sunt numite componente principale [TAN05]. Primul nou atribut este prima componentă principală, al doilea nou atribut este a doua componentă principală, etc. Vectorul propriu care este asociat cu cea mai mare valoare proprie indică care este direcţia în care datele au cea mai multă varianţă.

Vectorii proprii ai lui S definesc un nou set de axe. Analiza componentelor principale poate fi văzută ca o rotaţie a axelor originale către un nou set de axe care

sunt aliniate cu variabilitatea din date. Variabilitatea totală a datelor se păstrează dar noile atribute sunt acum necorelate.

Determinarea numărului optim de componente principale

O problemă care apare este *determinarea numărului optim de componente extrase*. Criteriile pentru această problemă sunt următoarele [LAR06]:

- *Criteriul valorilor proprii* - Valorile proprii exprimă importanţa componentelor principale. Criteriul valorilor proprii prevede că doar componentele cu valori proprii mai mari de 1 trebuie reţinute

- *Criteriul proporţiei varianţei* – analistul trebuie să specifice cât de mult din variabilitatea totală doreşte ca principalele componente să ia în calcul. Apoi analistul selectează componentele una după alta până la obţinerea proporţiei dorite a variabilităţii.

- *Criteriul comunalităţii minime* - Comunalitatea reprezintă proporţia de varianţă a unei variabile particulare care este folosită în comun cu alte variabile. Comunalităţile reprezintă importanţa globală a fiecărei variabile în PCA, ca şi un întreg. De exemplu o variabilă cu o comunalitate mult mai mică decât celelalte variabile indică că această variabilă împarte mult mai puţină variabilitate comună cu celelalte variabile şi contribuie mai puţin la soluţia PCA. Comunalităţile care sunt foarte scăzute pentru o variabilă anume ar trebui să fie o indicaţie pentru analist că acea variabilă s-ar putea să nu participe la soluţia PCA (spre exemplu s-ar putea să nu fie un membru al niciuneia dintre componentele principale). Per global, valori mari ale comunalităţii indică că principalele componente au extras cu succes o proporţie mare a variabilităţii din variabilele originale. Valori scăzute ale comunalităţii indică faptul că mai este multă variaţie în setul de date care nu a fost luată în calcul de principalele componente. Valorile comunalităţii sunt calculate ca fiind suma ponderilor componentelor pentru o variabilă dată. Comunalităţile mai mici de 0.5 pot fi considerate ca fiind prea scăzute având în vedere că aceasta ar însemna că variabila are în comun mai puţin de jumătate din variabilitatea sa în comun cu celelalte variabile.

- *Criteriul Scree Plot* – Un scree plot este o reprezentare grafică a valorilor proprii în funcţie de numărul pe care îl au componentele. Scree plot-urile sunt utile pentru găsirea pragului maxim în ceea ce priveşte numărul componentelor care trebuie reţinute. Cele mai multe scree plot-uri sunt destul de asemănătoare ca şi formă, începând de sus în partea stângă, căzând destul de rapid, aplatizându-se apoi la un anumit punct. Acesta datorită faptului că prima componentă explică de obicei cea mai mare parte din variabilitate, următoarele câteva componente explică într-o măsură moderată variabilitatea, în timp ce ultimele componente într-o măsură mică. Criteriul scree plot este următorul: numărul maxim de componente care ar trebui extrase este imediat anterior punctului în care graficul, începe pentru prima oară să se îndrepte formând o linie orizontală.

Descompunerea valorilor singulare (*SVD*)

SVD este o metodă pentru transformarea unor variabile corelate în variabile necorelate, care redau mai bine relaţiile dintre datele originale. În acelaşi timp *SVD* este o metodă pentru identificarea şi ordonarea dimensiunilor de-a lungul cărora datele au cea mai mare variaţie. Odată identificat unde este cea mai multă variaţie, este posibilă găsirea celei mai bune aproximări, a datelor originale, care foloseşte mai puţine dimensiuni. Prin urmare *SVD* poate fi văzută ca o metodă pentru reducere a dimensionalităţii. *SVD* se bazează pe o teoremă din algebra liniară, care spune că o matrice dreptunghiulară *A* cu *m* linii şi *n* coloane, poate fi împărţită într-un produs de 3 matrice în felul următor:

$$A = U\Sigma V^T \tag{2.5}$$

unde matricea *U* are dimensiunea m x m, Σ are dimensiunea *m x n*, iar *V* este de dimensiune *n x n*. *U* şi *V* sunt matrice ortogonale, aşadar $UU^T = I_m$ şi $VV^T = I_n$ [TAN05]. Matricea Σ este o matrice diagonală, entităţile de pe diagonala ei sunt ne-negative şi sunt ordonate în ordine descrescătoare: $\sigma_1 \geq \sigma_2 \geq ...\sigma_n \geq 0$. Valorile $\sigma_1, \sigma_2, ...\sigma_n$ se numesc **valori singulare** şi se calculează extrăgând radicalul din valorile proprii. Vectorii coloană $V, v_1, v_2, ..., v_n$ se numesc **vectori singulari dreapta**, iar coloanele lui *U* se numesc **vectori singulari la stânga**.

Vectorii proprii ai matricei $A^T A$ sunt vectorii singulari dreapta (coloanele matricei *V*), iar vectorii proprii ai matricei AA^T sunt vectorii singulari stânga (coloanele matricei *U*). Valorile proprii nenule ale matricelor $A^T A$ sau AA^T sunt pătratele valorilor singulare.

Algoritmul *SVD* urmăreşte descompunerea matricei *A* în produs de trei matrice, etapele algoritmului sunt următoarele:

- Se calculează produsul matricelor $A^T A$
- Se determină valorile proprii ale matricei $A^T A$, extrăgând radical din acestea se obţin valorile singulare, se construieşte matricea Σ amplasând valorile proprii pe diagonala principală în ordine descrescătoare
- Se determină vectorii proprii ai matricei $A^T A$ care corespund valorilor proprii calculate. Aceşti vectori proprii se amplasează pe coloanele matricei V, şi apoi se calculează V^T
- Se calculează produsul matricelor AA^T, apoi se determină vectorii proprii ai acestui produs, care reprezintă coloanele matricei *U*

După parcurgerea etapelor de mai sus matricea *A* a fost descompusă ca produs a matricelor *U*, Σ şi V^T. Reducerea efectivă a dimensionalităţii se realizează punând pe 0 valorile singulare cele mai mici [LES06] ale matricei Σ (luând în considerare doar valorile proprii cele mai mari - cele care surprind cea mai multă varianţă). Punerea pe 0 a anumitor valori din matricea Σ, care este o matrice diagonală, determină obţinerea unor coloane nule în această matrice. Coloanele nule

din matricea Σ vor determina anularea unor coloane din matricea *A*, atunci când se calculează produsul $U\Sigma V^T$. Calcularea produsului $U\Sigma V^T$, după ce au fost realizate prelucrările menţionate, determină obţinerea unei aproximări a matricei iniţiale, care va avea câteva coloane nule. Colonele nule vor fi eliminate în acest fel reducându-se dimensionalitatea.

Analiza factorială

Analiza factorială este o metodă care permite comasarea informaţiei dintr-un număr de variabile originale într-un set mai mic de dimensiuni (factori) cu o pierdere minimă de informaţie.

Tehnicile *PCA* şi *SVD* produc noi atribute care sunt combinaţii ale atributelor originale. Analiza factorială exprimă atribute existente ca şi combinaţii liniare a unui mic număr de atribute ascunse (factori ascunşi sau factori latenţi). Motivaţia este următoarea: deseori există caracteristici ale obiectelor, care sunt greu de măsurat în mod direct dar care par a fi legate de caracteristici măsurabile.

Un exemplu de analiză factorială este următorul [GOR06]: să presupunem că staff-ul unui lanţ de supermarket-uri doreşte să măsoare gradul de satisfacţie a cumpărătorilor cu privire la serviciile oferite. Se consideră că prezintă interes următorii doi factori: a) gradul de satisfacţie privind modul de servire şi b) gradul de satisfacţie privind calitatea produselor comercializate. Pentru măsurarea satisfacţiei clienţilor cu privire la serviciile menţionate se realizează un sondaj de opinie printre 1000 clienţi cărora li se adresează 10 întrebări. Fiecare răspuns a unui client reprezintă un scor. Scorul răspunsului dat de un client la o întrebare poate fi pus sub forma 7 * satisfacţie serviciu + 5 * satisfacţie produs (7 şi 5 reprezintă încărcările factorilor iar factorii ascunşi sunt satisfacţie serviciu şi satisfacţie client).

Fie $f_1, f_2, ..., f_p$ atributele ascunse. Aceste atribute sunt atribute noi şi au valori pentru fiecare obiect. Considerăm că matricea de date originale are dimensiunea de *m x n* şi se numeşte *D*, iar noua matrice se numeşte $F = f_1, f_2, ..., f_p$ şi are dimensiunea de *m x p*. Modelul de analiză factorială standard presupune că există următoarea relaţie între noile şi vechile obiecte [TAN05]:

$$d_{i*}^T = \Lambda f_{i*}^T + \varepsilon \tag{2.6}$$

$$d_{ij} = \lambda_{j1} f_{i1} + \lambda_{j2} f_{i2} + ... + \lambda_{jp} f_{ip} + \varepsilon_i \tag{2.7}$$

Λ care are intrările λ_{kl} este o matrice cu dimensiunea de *n x p* a încărcărilor factorilor (factor loadings) care indică, pentru fiecare din atributele originale, modul în care valorile originale depind de factori ascunşi, cum ar fi noile atribute.

Factorii vor fi ortogonali unul pe celălalt ceea ce înseamnă că vor fi necorelaţi.

Încapsularea locală lineară

Încapsularea locală lineară (*LLE – Locally linear embedding*) este un algoritm de învăţare nesupervizată care realizează maparea datelor de dimensiuni mari în mod neliniar spre un spaţiu de dimensiune mică [SAU00]. Algoritmul este cel de mai jos:

1. Se găsesc vecinii cei mai apropiaţi pentru fiecare punct de date x_i din spaţiul *n* dimensional. Pentru a determina distanţa dintre puncte se poate calcula distanţa Euclidiană şi vecinii cei mai apropiaţi se pot determina cu tehnica k-Nearest- Neighbours
2. Se calculează ponderile w_{ij} care reconstruiesc cel mai bine fiecare punct x_i pe baza vecinilor săi, x_j. Se exprimă fiecare punct x_i ca şi o combinaţie liniară a altor puncte $x_i = \sum_j w_{ij} x_j$, unde $\sum_j w_{ij} = 1$ şi $w_{ij} = 0$ dacă nu este un vecin apropiat a lui x_i
3. Se găsesc coordonatele fiecărui punct din spaţiul dimensional cu mai puţine dimensiuni, al dimensiunii specificate *p* utilizând ponderile găsite în pasul 2 [TAN05].

În etapa a doua, matricea de ponderi *W* ale cărei intrări sunt w_{ij}, se calculează minimizând eroarea de aproximare la pătrat aşa cum este măsurată în ecuaţia (2.8). Matricea W poate fi calculată rezolvând problema celor mai mici pătrate [SAU00].

$$error(W) = \sum_i \left(x_i - \sum_j w_{ij} x_j \right)^2 \qquad (2.8)$$

Erorile de reconstruire sunt măsurate cu funcţia de cost din relaţia 2.8 care adună pătratele distanţelor dintre toate punctele de date şi reconstrucţiile lor.

Etapa 3 efectuează reducerea efectivă a dimensionalităţii. Fiind date o matrice pondere şi un număr de dimensiuni *p* specificate de utilizator, algoritmul construieşte o vecinătate păstrând integrarea datelor în spaţiul dimensional cu mai puţine dimensiuni. Dacă y_i este vectorul din spaţiul dimensional cu mai puţine dimensiuni care corespunde lui x_i şi *Y* este noua matrice de date al cărei rând *i* este y_i atunci aceasta poate fi realizată găsind un *Y* care minimizează ecuaţia următoare:

$$error(Y) = \sum_i \left(y_i - \sum_j w_{ij} y_j \right)^2 \qquad (2.9)$$

Scalarea multidimensională

Tehnica standard *MDS*

Fiind date *M* obiecte în spaţiul *n* dimensional şi informaţia de distanţă dintre ele, cum ar fi matricea de distanţe de dimensiune *M x M,* ceea ce se urmăreşte este găsirea a *M* puncte în spaţiul *p* – dimensional (*p* < *n*) astfel încât distanţele dintre obiecte să fie menţinute atât de bine pe cât este posibil [FAL95].

Un exemplu clasic în domeniul scalării multidimensionale este următorul: având o matrice care conţine distanţele dintre cele mai mari *M* oraşe ale unei ţări, scalarea multidimensională încearcă să găsească coordonatele corecte pentru aceste oraşe.

În continuare se va descrie tehnica *MDS* standard pentru proiectarea datelor într-un spaţiu p – dimensional. Întâi se calculează matricea de distanţe *D*, în care fiecare poziţie d_{ij} reprezintă distanţa de la obiectul *i* la obiectul *j* [TAN05]. Cele mai uzuale măsuri pentru calculul distanţei dintre obiecte sunt *distanţa Minkovski*, *distanţa Euclidiană* şi *distanţa Manhattan [DEZ09]*. *Distanţa Minkovski* [STE02] între două obiecte *i* şi *j* care au *n* atribute, se calculează cu formula:

$$d_{ij} = \sqrt[r]{\sum_{k=1}^{n} \left(x_{ik} - x_{jk} \right)^{r}} \tag{2.10}$$

Dacă *r* are valoarea 2, distanţa este *Euclidiană*, iar dacă are valoarea 1, este *Minkovski*.

Fie $d_{ij}^{'}$ distanţa dintre obiecte după ce acestea au fost transformate în spaţiul p dimensional. Tehnica *MDS* clasică încearcă să asocieze fiecare obiect unui punct *p* - dimensional în aşa fel încât o cantitate numită *stress* să fie minimizată [TAN05]. Cantitatea numită *stres*s este definită după cum urmează:

$$stress = \sqrt{\frac{\sum_{ij} \left(d_{ij}^{'} - d_{ij} \right)^2}{\sum_{ij} d_{ij}^2}} \tag{2.11}$$

Versiunea clasică a lui *MDS* este un exemplu de tehnică *MDS metrică*, care presupune că disimilarităţile sunt variabile continue. Tehnicile *MDS non-metrice* presupun că disimilarităţile sunt categoriale. Tehnicile încearcă să asocieze obiectele punctelor *p*-dimensionale şi apoi să modifice punctele pentru a reduce mărimea s*tress*.

MDS începe cu o ghicire şi o îmbunătăţeşte în mod iterativ până când nu mai este posibilă nici o îmbunătăţire. În versiunile sale cele mai simple algoritmul asociază fiecare obiect unui punct p dimensional (folosind o euristică sau chiar aleator). Apoi algoritmul examinează fiecare punct, calculează distanţele dintre el şi celelalte puncte şi mută punctul pentru a fi cât mai aproape de situaţia iniţială.

FastMap

Algoritmul original *MDS* nu este potrivit pentru aplicaţii pe scară largă pentru că poate să necesite stocarea în memorie a unei matrice de distanţe de dimensiune *N x N* şi poate avea o complexitate de $O(N^3)$ [ATA10]. De-a lungul timpului au fost propuse diverse îmbunătăţiri pentru acest algoritm, printre care şi *FastMap*. *FastMap* determină o coordonată la un moment dat, examinând un număr constant de rânduri din matricea de distanţe [PLA10].

Scopul algoritmului este de a găsi M puncte în spaţiul p-dimensional, a căror distanţe Euclidiene să se potrivească cu distanţele Euclidiene din matricea de distanţe de dimensiune *M x M*. Algoritmul realizează proiectarea punctelor pe *k* direcţii mutual ortogonale. Se aleg două obiecte depărtate, numite pivoţi şi se alege linia care trece prin aceşti pivoţi, în spaţiul *n* dimensional. Ideea de bază a algoritmului este de a proiecta obiectele pe această linie. Determinarea proiecţiilor obiectelor pe această linie se realizează cu ajutorul *legii cosinusului*.

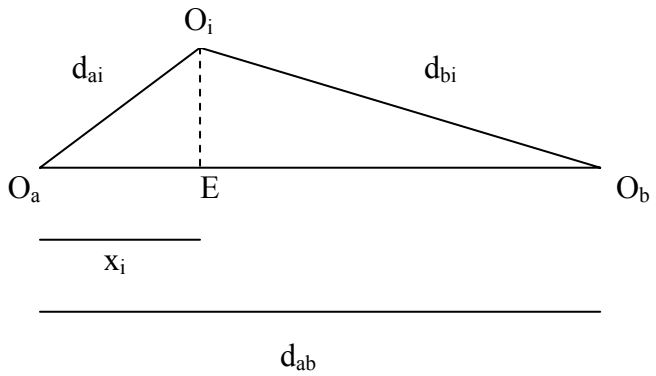

Fig. 2.3 – Proiectarea unui punct pe o dreaptă

Exprimând înălţimea triunghiului $O_a O_i O_b$ prin intermediul teoremei lui Pitagora în triunghiurile $O_a O_i E$ şi $E O_i O_b$ se obţin relaţiile de mai jos.

$$O_i E^2 = d_{ai}^2 - x_i^2$$
$$O_i E^2 = d_{bi}^2 - (d_{ab} - x_i)^2$$
$$d_{ai}^2 - x_i^2 = d_{bi}^2 - (d_{ab} - x_i)^2$$
$$d_{ai}^2 - x_i^2 = d_{bi}^2 - d_{ab}^2 + 2 d_{ab} x_i - x_i^2$$

$$d_{bi}^2 = d_{ai}^2 + d_{ab}^2 - 2 d_{ab} x_i \qquad (2.12)$$

Legea cosinusului (numită şi teorema lui Pitagora generalizată) spune că în orice triunghi este valabilă relaţia (2.12). Rezolvând ecuaţia pentru x_i se obţine prima coordonată a obiectului O_i .

$$x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{bi}^2}{2d_{ab}} \tag{2.13}$$

Se observă că datorită ecuaţiei (2.13) se pot mapa obiectele ca şi puncte pe o linie păstrând o parte din informaţiile legate de distanţă: de exemplu dacă O_i este aproape de pivotul O_a atunci x_i va fi mic [FAL95]. În felul acesta problema este rezolvată pentru *p=1* (pentru spaţiul unidimensional). Maparea obiectelor ca şi puncte în spaţiul 2 –dimensional sau p dimensional se bazează pe următoarea observaţie: considerăm un hiperplan de dimensiune *n-1* care este perpendicular pe linia (O_a, O_b). Fiecare punct din cele M puncte din spaţiul *n* dimensional va fi proiectat pe hiperplanul de dimensiune *n-1*. Se vor calcula distanţele dintre proiecţiile punctelor şi algoritmul se va relua recursiv. Fie $O_i^{'}$ şi $O_j^{'}$ proiecţiile obiectelor *i* şi *j* pe *hiperplanul* de dimensiune *n-1*. Distanţa $D^{'}(\ )$dintre $O_i^{'}$ şi $O_j^{'}$ poate fi calculată pe baza distanţelor dintre punctele O_i şi O_j în felul următor:

$$\left(D^{'}\left(O_i^{'}, O_j^{'}\right)\right)^2 = \left(D\left(O_i, O_j\right)\right)^2 - \left(x_i - x_j\right)^2 \tag{2.14}$$

Abilitatea de a calcula distanţa $D^{'}(\ )$ ne permite să calculăm proiecţiile pe o a doua dreaptă ortogonală pe prima. Astfel se poate rezolva problema pentru spaţiul bidimensional şi repetând acelaşi algoritm recursiv pentru spaţiul p – dimensional [FAL95].

ISOMAP

MDS nu este indicat pentru reducerea dimensionalităţii atunci când punctele au o relaţie non lineară complicată unele faţă de altele. ISOMAP care este o extensie a tradiţionalului MDS a fost dezvoltat pentru a lucra cu astfel de seturi de date. Algoritmul ISOMAP:

1. Se găsesc cei mai apropiaţi vecini ai fiecărui punct de date şi se creează un grafic de ponderi conectând un punct la cei mai apropiaţi vecini. Nodurile sunt punctele de date iar ponderile legăturilor sunt distanţele dintre puncte.
2. Se redefinesc distanţele dintre puncte pentru a fi lungimea celei mai scurte căi dintre 2 puncte în graficul de vecinătăţi
3. Se aplică clasicul *MDS* asupra noii matrice de distanţe.

Cei mai apropiaţi vecini pot fi definiţi fie prin luarea punctelor de proximitate *k*, fie prin luarea tuturor punctelor pe o rază specificată de la un punct. Scopul etapei a doua este de a calcula ***distanţa geodezică*** mai degrabă decât distanţa Euclidiană. Distanţa Euclidiană între 2 oraşe aflate în colţuri opuse ale pământului este lungimea unui segment de linie care trece prin pământ, în timp ce distanţa geodezică între 2 oraşe este lungimea celui mai scurt arc pe suprafaţa pământului [TAN05] .

Probleme uzuale

O problemă comună a tehnicilor de reducere a dimensionalităţii este *calitatea rezultatului*. Se pune problema dacă o tehnică poate să producă o reprezentare a datelor de o încredere rezonabilă într-un spaţiu cu mai puţine dimensiuni. O reprezentare a datelor în spaţiul cu mai puţine dimensiuni trebuie să capteze caracteristicile importante ale datelor şi să elimine aspectele care sunt irelevante sau chiar potrivnice. Reuşita acestei operaţii depinde de tipul de date şi de distribuţia datelor care pot fi analizate prin abordarea de reducere a dimensionalităţii.

Tehnicile precum *PCA*, *SVD* şi *analiza factorială* presupun o relaţie lineară între vechile şi noile seturi de atribute. Deşi această prezumţie poate fi adevărată în multe cazuri, în multe alte cazuri este necesară abordarea neliniară. Mai exact, algoritmi cum sunt *ISOMAP* şi *LLE* au fost dezvoltaţi pentru a trata relaţiile neliniare [TAN05].

Complexitatea spaţială şi temporală a algoritmilor de reducere a dimensionalităţii este o problemă cheie. Cei mai mulţi algoritmi de care am discutat au complexitate temporală sau spaţială de $O(M^2)$ sau mai mare, unde M este numărul de obiecte. Aceasta limitează aplicabilitatea lor asupra unor seturi mari de date deşi eşantionarea poate fi utilizată destul de eficient uneori. *FastMap* este singurul algoritm prezentat aici care are complexitate liniară, temporală şi spaţială.

Un alt aspect important al algoritmilor de reducere a dimensionalităţii se referă la *producerea aceluiaşi rezultat de fiecare dată când rulează*. *PCA, SVD* şi *LLE* produc acelaşi rezultat. *Analiza factorială* şi *tehnicile MDS* pot produce rezultate diferite la rulări diferite.

O altă problemă este determinarea numărului de dimensiuni pentru reducerea dimensionalităţii. Tehnicile pe care le-am luat în considerare pot realiza o reducere a dimensionalităţii în aproape orice număr de dimensiuni. În multe situaţii trebuie făcută o alegere între un număr mic de dimensiuni şi o eroare de aproximare mai mare şi o eroare de aproximare mai mică şi mai multe dimensiuni.

2.2.4. Selectarea de submulţimi de caracteristici

Selectarea submulţimilor de caracteristici este procesul de identificare şi de eliminare a cât mai multor caracteristici redundante sau irelevante[KOT06]. Caracteristicile pot fi clasificate în felul următor:

- *Relevante* – acestea au influenţă asupra ieşirii şi rolul lor nu poate fi asumat de celelalte caracteristici
- *Irelevante* – nu au nicio influenţă asupra ieşirii
- *Redundante* – există o redundanţă dacă o caracteristică poate prelua rolul alteia

Caracteristicile redundante dublează multe sau toate informaţiile conţinute în una sau mai multe alte atribute. Un exemplu de redundanţă este memorarea datei naşterii şi a CNP-ului în acelaşi tabel (CNP-ul include şi data naşterii). Caracteristicile irelevante conţin informaţii care sunt nefolositoare pentru procesul de data mining derulat. De exemplu numărul de ordine al pacienţilor este irelevant pentru analiza dacă suferă de o anumită boală sau nu. Caracteristicile redundante şi relevante pot reduce acurateţea clasificării.

Deşi anumite atribute irelevante şi redundante pot fi eliminate imediat utilizând bunul simt şi cunoştinţele din domeniu, pentru selectarea celui mai bun subset de caracteristici este necesară o abordare sistemică [TAN05]. Abordarea ideală pentru selectarea caracteristicilor, implică încercarea tuturor subseturilor posibile de caracteristici ca date de intrare pentru algoritmul de *data mining* care ne interesează şi selectarea subsetului care produce cele mai bune rezultate. Această metodă prezintă avantajul că reflectă obiectivul în funcţie de algoritmul de data mining care va fi folosit. Din păcate fiindcă numărul de subseturi care implică *n* atribute este 2^n, o asemenea abordare este impracticabilă în cele mai multe situaţii fiind necesare strategii alternative. Există 3 abordări standard pentru selectarea subseturilor de caracteristici şi anume: integrat, filtrat şi împachetat.

Abordarea integrată

Selectarea caracteristicilor are loc în mod natural ca parte a algoritmului de *data mining*. În mod specific în timpul operării algoritmului de *data mining*, chiar algoritmul decide atributele pe care le va folosi şi cele pe care le va ignora. Algoritmii utilizaţi pentru construirea clasificatorilor arborilor decizionali, operează deseori în această manieră.

Abordarea filtru

Caracteristicile sunt selectate înainte de rularea algoritmului de data mining utilizând o abordare care este independentă de sarcina de data mining. De exemplu putem selecta seturi de atribute a căror corelaţie aferentă este minimă.

Funcţiile de evaluare filtru pot fi împărţite în următoarele categorii [KOT06]:

- *Distanţă* – Pentru o problemă cu două clase, o caracteristică *X* este preferată faţă de altă caracteristică *Y* dacă *X* induce o mai mare diferenţă între probabilităţile condiţionate a celor două clase decât *Y*
- *Informaţie* – caracteristica *X* este preferată faţă de caracteristica *Y* dacă câştigul de informaţie de la caracteristica *X* este mai mare decât cel de la caracteristica *Y*
- *Dependenţă* – coeficientul este o măsură clasică de dependenţă şi poate fi utilizat pentru a găsi corelaţia dintre o caracteristică şi o clasă. Dacă corelaţia caracteristicii *X* cu clasa *C* este mai mare decât cea a caracteristicii *Z* cu clasa *C* atunci caracteristica *X* este cea preferată
- *Consistenţă* – două exemple sunt în conflict dacă au aceleaşi valori pentru un subset de caracteristici dar aparţin unor clase diferite

Abordarea împachetată

Aceste metode utilizează algoritmul de data mining ţintă, ca şi o cutie neagră, pentru găsirea celui mai bun subset de atribute, într-o modalitate similară algoritmului ideal descris mai sus, dar fără enumerarea tuturor subseturilor posibile.

Procesul de selectare al caracteristicilor constă în 4 părţi (vezi figura 2.4): un algoritm de selectare care generează subseturile propuse de caracteristici şi încearcă să găsească un subset optim, o măsură pentru evaluarea subsetului care determină cât de bun este subsetul propus, un criteriu de oprire şi o procedură de validare.

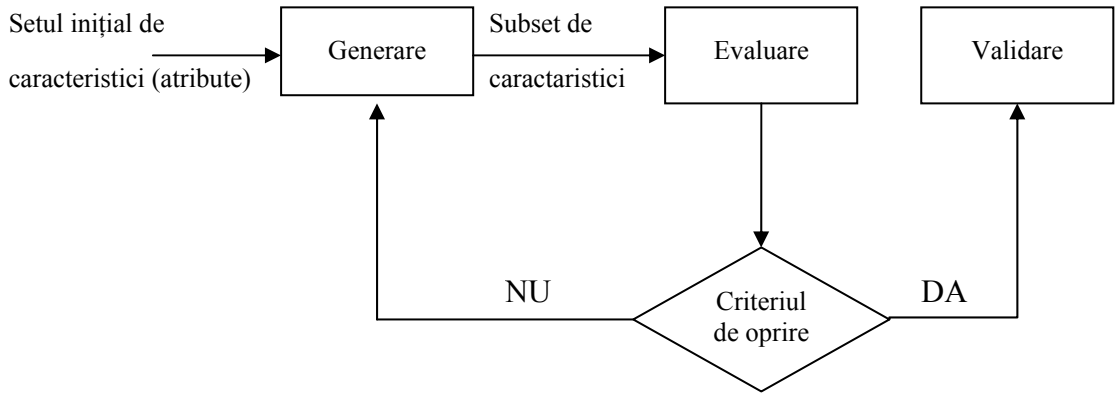

Fig. 2.4 – Fluxul procesului de selectare a unui subset de caracteristici

Selectarea caracteristicilor reprezintă o căutare a tuturor subseturilor de caracteristici. Diferite tipuri de strategii de căutare pot fi utilizate, însă strategia de căutare ar trebui să fie rapidă şi ar trebui să găsească seturi de caracteristici optime sau aproape optime. De obicei nu este posibilă satisfacerea ambelor cerinţe, astfel că sunt necesare compromisuri [TAN05].

Pentru că numărul de subseturi poate fi enorm şi examinarea tuturor acestora nu este practică, este necesar un gen de criteriu de oprire. Criteriul de oprire poate fi unul din următoarele:

- adăugarea sau ştergerea oricărei caracteristici nu produce un subset mai bun
- s-a obţinut un subset optim în acord cu o anumită funcţie de evaluare
- s-a executat un număr maxim de iteraţii, se alege cel mai bun subset găsit
- s-a obţinut un subset de o dimensiune dorită

În final, odată ce un subset de caracteristici a fost selectat, rezultatele algoritmului de *data mining* ţintă pe subsetul selectat, trebuie validate. O abordare de evaluare directă este rularea algoritmului cu întreg setul de caracteristici şi compararea rezultatelor complete cu rezultatele obţinute utilizând subsetul de caracteristici. În cel mai bun caz subsetul de caracteristici va produce rezultate care sunt mai bune sau aproape la fel de bune ca şi cele produse utilizând toate caracteristicile. O altă abordare de validare este utilizarea unui număr de algoritmi de selectare a caracteristicilor diferiţi, pentru a obţine subseturi de caracteristici şi a compara rezultatele rulării algoritmului de *data mining* pe fiecare subset.

Ponderea caracteristicilor

Ponderea caracteristicilor este o alternativă la păstrarea sau eliminarea caracteristicilor. Caracteristicilor mai importante li se desemnează o pondere mai mare, în timp ce caracteristicilor mai puţin importante li se dă o pondere mai mică. Aceste ponderi sunt conferite uneori pe baza cunoştinţelor în domeniu privind importanţa relativă a caracteristicilor. În mod alternativ, ele pot fi determinate automat. De exemplu anumite scheme de clasificare cum sunt *SVM*, produc modele de clasificare în care fiecare caracteristică are o anumită pondere. Caracteristicile cu pondere mai mare joacă un rol mai important în cadrul modelului. Normalizarea

obiectelor, atunci când se calculează similaritatea cosinusoidală poate fi privită ca un tip de pondere a caracteristicilor [TAN05].

2.2.5. Crearea caracteristicilor

Crearea caracteristicilor este un proces care descoperă informaţii lipsă despre relaţiile dintre caracteristici şi măreşte spaţiul caracteristicilor deducând sau creând noi caracteristici. Presupunând că iniţial dispunem de *n* caracteristici A_1, A_2, …,A_n după construirea de caracteristici putem avea încă *m* caracteristici A_{n+1}, A_{n+2} , …, A_{n+m}. De exemplu o nouă caracteristică A_k, $(n < k \leq n + m)$, poate fi obţinută realizând o operaţie logică între atributele A_i şi A_j. De exemplu o problemă bidimensională cu A_1 = *lăţimea* şi A_2 = *lungimea*, poate fi transformată într-o problemă unidimensională, după ce se construieşte aria $A_3=A_1 * A_2$ [LIU03b]. Prin crearea de noi caracteristici se poate introduce redundanţă. Aceasta poate fi apoi eliminată prin selectarea unor subseturi de caracteristici.

Deseori este posibilă crearea din atributele originale a unui nou set de atribute care captează în mod mai eficient informaţiile dintr-un set de date. Mai mult, numărul de noi atribute poate fi mai mic decât numărul de atribute originale, permiţându-ne să recoltăm toate beneficiile reducerii dimensionalităţii menţionate mai sus. Crearea unui nou set de caracteristici din datele brute, originale este cunoscută drept *extragerea caracteristicilor*.

Maparea caracteristicilor

Procesul de transformare al caracteristicilor, poate extrage un set de noi caracteristici din setul original de caracteristici printr-o mapare funcţională. După extragerea caracteristicilor vom avea $b_1, b_2, ...b_m$ unde $m < n$ şi $b_i = f_i(a_1, a_2, ..., a_n)$ şi f_i este o funcţie de mapare. De exemplu, pentru fiecare obiect *0* se pot defini: $b_1(x) = c_1 * a_1(x) + c_2 * a_2(x)$, unde c_1 şi c_2 sunt constante [KOT06].

Construcţia caracteristicilor

Uneori caracteristicile din seturile originale de date au informaţiile necesare dar nu au forma potrivită pentru algoritmul de data mining. În această situaţie una sau mai multe caracteristici construite din caracteristicile originale pot fi mai folositoare decât acestea. Un exemplu în acest sens este următorul: presupunem că dispunem de un set de informaţii despre artefactele istorice, printre care şi volumul şi masa fiecărui artefact. Dacă considerăm că aceste artefacte sunt realizate dintr-un număr mic de materiale cum ar fi lemn, bronz, lut, aur şi vrem să clasificam aceste artefacte din perspectiva materialului din care sunt construite, cel mai uşor ar fi să construim o nouă caracteristică densitatea =masa / volum şi să ne folosim de ea pentru a genera o clasificare corectă. Deşi au existat tentative de a efectua automat construcţia caracteristicilor (de exemplu algoritmul *Gala*), prin explorarea unor combinări matematice simple ale atributelor existente, cea mai utilizată abordare este construcţia caracteristicilor utilizând expertiza în domeniu [TAN05].

Un algoritm care realizează construcţia de caracteristici este algoritmul *Gala*. Acesta execută construcţia caracteristicilor în timpul construirii unui arbore de

clasificare a deciziilor. Noi caracteristici sunt construite la fiecare nod al arborelui efectuând o căutare *brench and bound* în spaţiul caracteristicilor. Căutarea este executată prin combinarea iterativă a caracteristicii cu cea mai mare valoare a câştigului de informaţie cu o caracteristică originală de bază care întruneşte anumite criterii de filtrare. Gala construieşte noi caracteristici binare utilizând operatori logici cum sunt conjuncţia, negaţia [KOT06].

2.2.6. Discretizarea şi binarizarea

Unii algoritmi de data mining în special anumiţi algoritmi de clasificare, necesită ca datele să fie sub forma atributelor categoriale [ZHA05]. Algoritmii care găsesc şabloane de asociere necesită ca datele să fie sub forma atributelor binare, astfel deseori este necesară transformarea unui atribut continuu într-un atribut categorial (discretizare), iar atât atributele continue cât şi cele discrete pot necesita a fi transformate în unul sau mai multe atribute binare (binarizare). Mai mult dacă un atribut categorial are un număr mare de valori (categorii) sau anumite valori nu au loc frecvent atunci poate fi benefic pentru anumite sarcini de data mining să reducă numărul de categorii prin combinarea unora dintre valori [TAN05].

La fel ca şi în cazul selectării caracteristicilor, cea mai bună abordare de discretizare şi binarizare este cea care produce cel mai bun rezultat pentru algoritmul de data mining care va fi utilizat pentru analiza datelor. De obicei aplicarea unui astfel de criteriu în mod direct nu este practică. Drept urmare discretizarea şi binarizarea sunt efectuate într-o modalitate în care satisfac un criteriu care are o bună performanţă pentru sarcina de data mining avută în vedere.

Discretizarea are multe avantaje: atributele discrete sunt mai uşor de utilizat, înţeles şi explicat decât cele continue. Discretizarea face învăţarea mai clară şi mai rapidă. În general rezultatele obţinute utilizând caracteristici discrete sunt mai compacte şi mai clare, putând fi mai uşor examinate, utilizate şi reutilizate.

Discretizarea atributelor continue

Discretizarea este folosită în general în problemele de clasificare şi cele de analiză a asocierilor. Procesul de inducţie al arborilor de decizie este mult mai complicat când se folosesc atribute continue decât atunci când se folosesc atribute discrete. Este necesară discretizarea atributelor continue fie înainte de inducţia arborelui de decizie, fie în timpul procesului de construire al arborelui. Algoritmii C4.5 şi CART, care sunt folosiţi pe scară largă utilizează diverse tehnici pentru a evita lucrul direct cu valori continue [LIU02b]. În general cea mai bună discretizare depinde de algoritmul folosit şi de celelalte atribute considerate.

Transformarea unui atribut continuu într-un atribut categorial presupune două acţiuni: determinarea numărului de categorii în care se va face transformarea şi deciderea modului în care se va face maparea valorilor continue în aceste categorii. În primul pas, după ce valorile atributului continuu sunt ordonate, ele sunt divizate în n intervale, specificând $n-1$ puncte de împărţire (split point). În continuare, toate valorile dintr-un interval sunt mapate la aceeaşi valoare categorială. De aceea problema discretizării este de a găsi câte puncte de împărţire vor fi şi unde vor fi ele plasate. Rezultatul poate fi ilustrat fie sub forma unor seturi

de intervale $\{(x_0, x_1], (x_1, x_2], ..., (x_{n-1}, x_n],\}$ sau sub forma unor inegalităţi $x_0 < x \le x_1, ..., x_{n-1} < x \le x_n$ [TAN05] .

Metodele de discretizare pot să fie împărţite după mai multe criterii în metode supervizate versus nesupervizate, metode locale versus globale, metode statice respectiv dinamice şi, metode sus – jos respectiv metode jos- sus [DOU95].

Cel mai comun criteriu de clasificare a algoritmilor de discretizare este clasificarea lor în algoritmi nesupervizaţi, care discretizează atributele fără să ia în considerare etichetele de clasă şi algoritmi supervizaţi care discretizează atributele ţinând cont de atributul clasă.

Metodele locale produc partiţii care sunt aplicate pentru a localiza regiuni ale spaţiului de instanţe. Metodele globale produc o mască peste întregul spaţiu *n* dimensional al instanţelor continue, în care fiecare caracteristică este împărţită în regiuni independente de celelalte atribute. Masca conţine $\prod_{i=1}^{n} k_i$ regiuni, unde k_i este numărul de partiţii ale atributului al *i*-lea [DOU95].

Multe tehnici de discretizare necesită un parametru *k* care indică numărul maxim de intervale care va fi produs în procesul de discretizare. Metodele statice realizează o trecere prin date pentru fiecare caracteristică şi determină valoarea lui *k* pentru fiecare caracteristică, într-un mod independent de celelalte caracteristici. Tehnicile dinamice realizează o căutare prin spaţiul valorilor *k* posibile, pentru toate caracteristicile simultan, astfel captând interdependenţele în discretizarea caracteristicilor [DOU95]. O metodă dinamică discretizează valorile continue când clasificatorul se construieşte (precum în algoritmul C4.5) în timp ce discretizarea statică se realizează înainte de procesul de clasificare [LIU02b].

Metodele de discretizare pot fi împărţite în metode sus-jos şi în metode jos-sus. Metodele sus-jos încep cu o listă goală de puncte de tăiere şi adaugă noi puncte la această listă împărţind intervalele pe măsură ce procesul de discretizare avansează. Metodele jos-sus încep cu o listă care conţine toate valorile continue, ca şi puncte de tăiere şi se elimină treptat din aceste puncte, unind intervalele pe măsură ce procesul de discretizare avansează [KOT06].

Etapele unui proces de discretizare tipic sunt următoarele: (1) se sortează valorile continue ale caracteristicii care urmează să se discretizeze, (2) se evaluează puncte de împărţire sau intervale adiacente pentru unire, (3) în acord cu un anumit criteriu se împart sau se unesc intervalele de valori continue, (4) se opreşte discretizarea.

Discretizarea nesupervizată

Cea mai simplă metodă de discretizare este metoda de discretizare nesupervizată numită discretizarea de dimensiune egală. Aceasta calculează maximul şi minimul caracteristicii care urmează să fie discretizate şi împarte domeniul observat în *k* intervale de dimensiuni egale. O altă metodă nesupervizată este discretizarea bazată pe frecvenţă egală. Aceasta numără câte valori sunt în atributul care urmează să fie discretizat şi îl împarte în intervale conţinând acelaşi număr de instanţe [KOT06]. Prin acest tip de discretizare se pierde informaţia legată

de clasă pentru că se combină în acelaşi interval valori care sunt legate de etichete de clase diferite.

Discretizarea supervizată

Discretizarea supervizată urmăreşte împărţirea caracteristicilor în intervale cât mai pure din punct de vedere al împrăştierii etichetelor de clasă. În acest sens este necesară localizarea punctelor de împărţire într-un mod care maximizează puritatea intervalelor. În continuare este prezentată o abordare simplă, bazată pe entropie.

Fie *k* numărul de etichete de clasă diferite, m_i numărul valorilor în intervalul al *i*-lea al unei partiţii şi m_{ij} numărul valorilor clasei *j* în intervalul *i*. Atunci entropia e_i a intervalului al *i*-lea este dată de ecuaţia:

$$e_i = \sum_{i=1}^{n} p_{ij} \log_2 p_{ij} \tag{2.15}$$

Unde $p_{ij} = \frac{m_{ij}}{m_i}$ este probabilitatea apariţiei clasei cu valoarea *j* în al *i*-lea interval.

Entropia totală a partiţiei este media ponderată a entropiilor intervalelor individuale:

$$e = \sum_{i=1}^{n} w_i e_i \tag{2.16}$$

Unde *m* este numărul valorilor, $w_i = \frac{m_i}{m}$ este fracţia valorilor în al *i*-lea interval şi *n* este numărul de intervale. În mod intuitiv entropia unui interval este o măsură a dezordinii acestuia. Dacă un interval conţine doar valorile unei clase (este perfect pur) atunci entropia este 0 şi nu contribuie la entropia totală. Dacă clasele valorilor într-un interval apar regulat (intervalul este cât de impur posibil) atunci entropia este maximă.

O abordare simplă pentru împărţirea unui atribut continuu începe prin împărţirea valorilor iniţiale astfel încât cele două intervale care rezultă să aibă entropii minime. Această tehnică necesită considerarea fiecărei valori ca un posibil punct de împărţire pentru că se prezumă că fiecare interval conţine seturi ordonate de valori. Procesul de împărţire este repetat apoi într-un alt interval alegând de obicei intervalul cu cea mai mare entropie până ce este atins un număr de intervale specificat de utilizator sau este satisfăcut un criteriu de oprire [TAN05].

Binarizarea

O tehnică simplă pentru binarizarea atributelor categoriale este următoarea: Dacă există *m* atribute categoriale, atunci se atribuie fiecare valoare categorială unui întreg din intervalul $[0, m-1]$. Dacă atributul este ordinal atunci ordinea

trebuie să se menţină prin atribuire. Chiar dacă atributul este reprezentat original utilizând întregi, această reprezentare este necesară dacă întregii nu fac parte din intervalul $[0, m-1]$. În continuare fiecare din aceşti întregi se convertesc la numărul binar corespunzător. Pentru a reprezenta în binar întregii sunt necesari *n* biţi unde $n = \lceil \log_2(m) \rceil$. De exemplu pentru o variabilă categorială $\{insufucient, suficient, bine, foarte\_bine\}$ sunt necesare două variabile binare, precum în tabelul de mai jos:

Valoare categorială	Valoare întreagă	x1	x2
Insuficient	0	0	0
Suficient	1	0	1
Bine	2	1	0
Foarte bine	3	1	1

Tabel 2.1 – binarizare calificative

O astfel de transformare poate cauza complicaţii cum ar fi crearea de relaţii neintenţionate între atributele create. De exemplu atributele x1 şi x2 sunt corelate pentru că codificarea valorii categoriale *foarte_bine* necesită ambele atribute. Mai mult, analiza asociaţiilor necesită atribute binare asimetrice, unde doar prezenţa valorii 1 este importantă [TAN05]. Pentru problemele de asociere este necesară introducerea unui atribut binar pentru fiecare atribut categorial, după cum se vede în tabelul 2.2.

Valoare categorială	Valoare întreagă	x1	x2	x3	x4
Insuficient	0	1	0	0	0
Suficient	1	0	1	0	0
Bine	2	0	0	1	0
foarte bine	3	0	0	0	1

Tabel 2.2 – binarizare în probleme de asociere

Dacă numărul atributelor categoriale care rezultă în acest fel este prea mare atunci tehnicile prezentate în continuare pot să fie utilizate pentru a reduce numărul valorilor categoriale înainte de binarizare.

2.2.7. Transformarea variabilelor

Transformarea variabilelor se referă la o transformare aplicată tuturor valorilor unei variabile. Transformarea se poate realiza prin funcţii simple sau normalizare.

Funcţii simple

Pentru acest tip de transformare a variabilei se aplică o funcţie matematică simplă în mod individual, fiecărei valori. Dacă x este o variabilă atunci exemple a unor astfel de transformări includ x^k, *log (x)*, e^x, $\sqrt{x}$, $\frac{1}{x}$, *sin(x)*. În statistică

transformările variabilelor în special radicalul, logaritmul şi *1 / x* sunt deseori utilizate pentru a transforma datele care nu au o distribuţie gausiană, în date care au o astfel de distribuţie.

Transformările variabilelor trebuie aplicate cu prudenţă pentru că modifică natura datelor. Deşi se doreşte acest lucru pot apărea probleme dacă natura transformării nu este evaluată complet. De exemplu, transformarea $\frac{1}{x}$, inversează ordinea elementelor variabilei. Pentru a clarifica efectul unei transformări este important să se urmărească dacă transformarea menţine ordinea, dacă se poate aplica tuturor valorilor (în special celor negative şi valorii 0) şi care este efectul acesteia asupra valorilor cuprinse între 0 şi 1 [TAN05].

Normalizarea şi Standardizarea

Un alt tip uzual de transformare a variabilelor este **standardizarea şi normalizarea** unei variabile.

Scopul standardizării sau normalizării este de a face ca un set întreg de valori să aibă o proprietate anume. Un exemplu tradiţional este cel al "Standardizării unei variabile în statistică". Dacă $\bar{x}$ este media valorilor atributelor şi s_x este deviaţia lor standard, atunci transformarea $x' = \frac{\left(x - \bar{x}\right)}{s_x}$ creează o nouă variabilă care are o medie de 0 şi o deviaţie standard de 1. Normalizarea menţionată este cunoscută sub denumirea *normalizarea z-score*. Dacă am combina într-un anumit fel variabile diferite, atunci o astfel de transformare este deseori necesară pentru a evita ca o variabilă cu valori mari să domine rezultatele calculului. De exemplu compararea oamenilor pe baza a două variabile: vârsta şi venitul. Pentru oricare doi oameni diferenţa de venit va fi mult mai mare în termeni absoluţi (milioane de lei) decât diferenţa de vârstă (mai puţin de 150 de ani). Dacă diferenţele în spectrul valorilor vârstei şi venitului nu sunt luate în considerare, atunci comparaţia dintre oameni va fi dominată de diferenţele de venit [TAN05]. Dacă similaritatea sau nesimilaritatea a doi oameni este calculată utilizând măsuri specifice, atunci în multe cazuri cum ar fi distanţa euclidiană, veniturile vor domina calculele.

Media şi deviaţia standard sunt puternic afectate de valori extreme, aşa că transformarea menţionată mai sus este adeseori modificată. Mai întâi media este înlocuită de mediană şi anume valoarea de mijloc. În al doilea rând, deviaţia standard este înlocuită de deviaţia standard absolută. Mai specific, dacă x este o variabilă, atunci deviaţia standard absolută a x este dată de $\sigma_A = \sum_{i=1}^{m} \left| x_i - \mu \right|$, unde x_i este a i-a valoare a variabilei, m este numărul de obiecte iar μ este fie media fie mediana.

O altă tehnică de normalizare este *normalizarea min-max* [KOT06]:

$$x' = \frac{x - \min}{\max - \min}\left(noul\_\max - noul\_\min\right) + noul\_\min \qquad (2.17)$$

în care x' reprezintă noua valoare iar *x* reprezintă vechea valoare.

Normalizarea se mai poate realiza utilizând tehnica *normalizarea prin scalare zecimală*:

$$x' = \frac{x}{10^i} \tag{2.18}$$

unde i este cel mai mic întreg astfel încât $\max(x') < 1$.

2.2.8. Consideraţii privind tehnicile de preprocesare

Datele brute pot fi afectate de zgomot, pot exista valori aberante, valori lipsă, date duplicat, date introduse greşit sau date expirate [GOR06]. În acest sens se impune *o prelucrare a datelor brute*, prin utilizarea de filtre hard şi soft pentru reducerea zgomotului, prin utilizarea unor parametri statistici toleranţi la valori extreme, prin estimarea valorilor lipsă, prin eliminarea datelor duplicat, a datelor introduse greşit sau a datelor expirate. În urma prelucrării datelor brute se obţin date curăţate.

Algoritmii de data mining care sunt lenţi necesită foarte mult timp de rulare dacă setul de date este foarte mare. De aceea este necesară reducerea volumului datelor. Aceasta se poate face reducând numărul de instanţe (de exemplu prin agregare şi eşantionare) sau reducând numărul atributelor (prin tehnici de reducere a dimensionalităţii sau selectarea de submulţimi de caracteristici).

Datele iniţiale pot fi nepotrivite cu anumiţi algoritmi de data mining şi în acest sens se impune o transformare a acestora prin tehnici precum discretizarea, binarizarea, etc. De exemplu algoritmul de clasificare ID3 nu funcţionează decât cu atribute nominale. Un set de date care conţine atribute numerice poate fi clasificat cu ID3 doar dacă atributele numerice se discretizează.

2.3. Îmbunătăţirea preprocesării cu *Weka*

2.3.1. Conectarea aplicaţiei Weka la baze de date

Încărcarea datelor din diferite baze de date (citirea datelor) în instrumentele software cu care se va realiza studiul de data mining este tot o etapă a preprocesării. De multe ori şi această etapă întâmpină greutăţi datorită multitudinii de formate existente şi datorită tehnologiilor de lucru care de multe ori nu sunt uşor accesibile utilizatorilor care nu sunt specialişti în calculatoare.

Conectarea lui *Weka* la baze de date, deşi este posibilă este greoaie. Ea se poate realiza în două moduri: utilizând doar tehnologia Java Database Connectivity (JDBC) sau utilizând atât tehnologia JDBC cât şi tehnologia Open Database Connectivity (ODBC) [www6].

Prima variantă de a lega *Weka* la diferite baze de date este utilizând *JDBC* driver-ul pentru baza de date la care se doreşte conectarea. Fişierul **.jar* ce conţine *JDBC* driverul trebuie adăugat în variabila de mediu care îi spune lui *JAVA* unde să caute după clase şi care se numeşte *CLASSPATH*. Dacă nu există această variabilă de mediu atunci ea se va crea. Următoarea etapă este cea de navigare în directorul *weka/experiment/* a arhivei *weka.jar* şi de setare a fişierului cu proprietăţi *DatabaseUtils.props* corespunzător bazei de date (DatabaseUtils.props.mysql, DatabaseUtils.props.oracle, etc). În mod normal în acest fişier trebuie setat doar *jdbcDriver* şi *jdbcURL*. Un exemplu de setări pentru o conectare la o bază de date *MySQL* locală este următorul:

```
jdbcDriver=com.mysql.jdbc.Driver
jdbcURL = jdbc:mysql://localhost:3306/db_name
```

Fişierul configurat trebuie redenumit în *DatabaseUtils.props* şi amplasat în directorul de start a lui *Weka* pentru a putea fi recunoscut. Ultima etapă presupune deschiderea bazei de date cu ajutorul comenzii *OpenDB…*a *Explorerului* din *Weka* şi construirea şi rularea interogării SQL cu scopul aducerii datelor din baza de date în *Weka*.

Cea de-a doua variantă de conectare utilizează *JDBC-ODBC bridge* şi presupune parcurgerea următoarelor trei etape: în prima etapă se utilizează *ODBC* pentru a crea un nou *Data Source Name (DSN)*, (se alege driverului corespunzător bazei de date la care se doreşte conectarea, se menţionează numele *DSN* –ului şi datele necesare realizării conexiunii la baza de date. Dacă acel driver nu se găseşte în *Windows*, el trebuie descărcat şi instalat). A doua etapă este cea de completare a fişierului cu proprietăţi *DatabaseUtils.props* cu datele necesare realizării conexiunii *JDBC*:

```
jdbcDriver=sun.jdbc.odbc.JdbcOdbcDriver
jdbcURL=jdbc:odbc:dbname
```

unde, *dbname* reprezintă numele *DSN*-ului creat. Acest fişier se găseşte în directorul *weka/experiment/* al arhivei weka. Fisierul *DatabaseUtils.props* trebuie să fie recunoscut de *Weka*, pentru aceasta cea mai simplă variantă este mutarea lui în directorul lui *Weka*. Următoarea etapă este cea de deschidere a bazei de date, din meniul *Open DB…*a *Explorerului* din *Weka*. Dacă deschiderea bazei de date a reuşit, urmează scrierea interogării *SQL* pentru aducerea datelor din baza de date în *Weka*. Datele odată aduse în *Weka* pot fi salvate în format *ARFF*.

Preluarea datelor dintr-o bază de date *Access* sau *FoxPro* cu ajutorul tehnologiilor prezentate mai sus nu este accesibilă utilizatorilor de *Weka* care nu sunt specializti în calculatoare (necesită realizarea unor configurări în *Windows* şi în *Weka* precum şi cunoaşterea limbajului *SQL*). Acesta este motivul pentru care a fost concepută şi dezvoltată aplicaţia *ARFF* Convertor care are rolul de a simplifica acest proces făcându-l accesibil oricui.

2.3.2. Preprocesarea datelor cu ajutorul lui Weka

Weka oferă foarte multe facilităţi pentru preprocesarea datelor. Instrumentele de preprocesare în *Weka* se numesc filtre. *Weka* conţine filtre pentru analiza componentelor principale, discretizare, binarizare, selectarea de atribute, crearea de atribute etc. În figura 2.5 pot fi vizualizate o parte din filtrele pe care *Weka* le oferă. Atributele de tip sir de caractere pot fi convertite în atribute nominale, iar atributele nominale pot fi convertite în atribute binare sau şir de caractere. Atributele numerice pot fi transformate în atribute binare sau nominale. Selectarea unor submulţimi de atribute (caracteristici) se poate realiza din tab-ul *Select attributes*. Utilizatorul trebuie să menţioneze metoda de căutare a unor submulţimi de atribute şi metoda de evaluare a lor. Printre metodele de căutare a unor submulţimi de caracteristici se află metodele: *BestFirst* şi *ExhaustiveSearch*. Două metode de evaluare a subseturilor sunt: *CfsSubsetEval* şi *InfoGainAttributeEval*. Evaluatorul *CfsSubsetEval* evaluează valoarea unui subset de atribute considerând abilitatea de prezicere individuală a fiecărei caracteristici împreună cu gradul de redundanţă dintre ele. Sunt preferate subseturile de caracteristici care sunt foarte corelate cu clasa şi mai puţin corelate între ele. Evaluatorul *InfoGainAttributeEval* – evaluează valoarea unui atribut măsurând câştigul de informaţie în raport cu clasa.

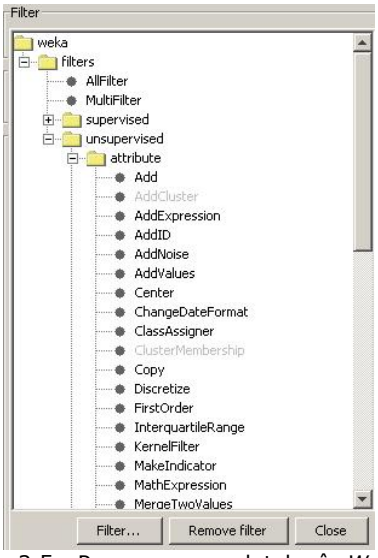

Fig. 2.5 - Preprocesarea datelor în *Weka*

2.3.3. ARFF Convertor

Noţiuni introductive

Weka este un program de machine learning şi *data mining* foarte puternic şi folosit pe scară largă. Fişierele de intrare pentru *Weka* sunt fişierele *ARFF* (*Attribute-Relation File Format*). Conectarea lui *Weka* la baze de date (vezi subcapitolul 2.10.1) este dificilă şi necesită cunoştinţe de *SQL*. O alternativă la acest proces complicat, o

reprezintă exportarea datelor din aceste baze de date în format *ARFF*, şi apoi încărcarea acestor date în *Weka*[ROM08]. O soluţie posibilă în acest sens este utilizarea soft-ului *Cahit Arf*, care se poate conecta la diferite baze de date şi poate realiza conversii în *ARFF* din diferite formate, dar utilizarea acestui instrument necesită cunoştinţe de SQL şi restrânge în acest fel numărul utilizatorilor [ALK10].

Ca o soluţie la problema menţionată, autorul a conceput şi dezvoltat un instrument (*Arff Convertor*), a cărui utilizare nu necesită cunoaşterea limbajului *SQL*, care permite conectarea cu uşurinţă la baze de date (în prezent *Oracle, MySQL, Mirosoft Access, Microsoft Visual FoxPro, Microsoft Excel*) şi generarea unui fişier de intrare pentru *Weka* (în format *ARFF*) cu datele dorite din una sau mai multe tabele relaţionate ale bazei de date (vezi figura 2.6).

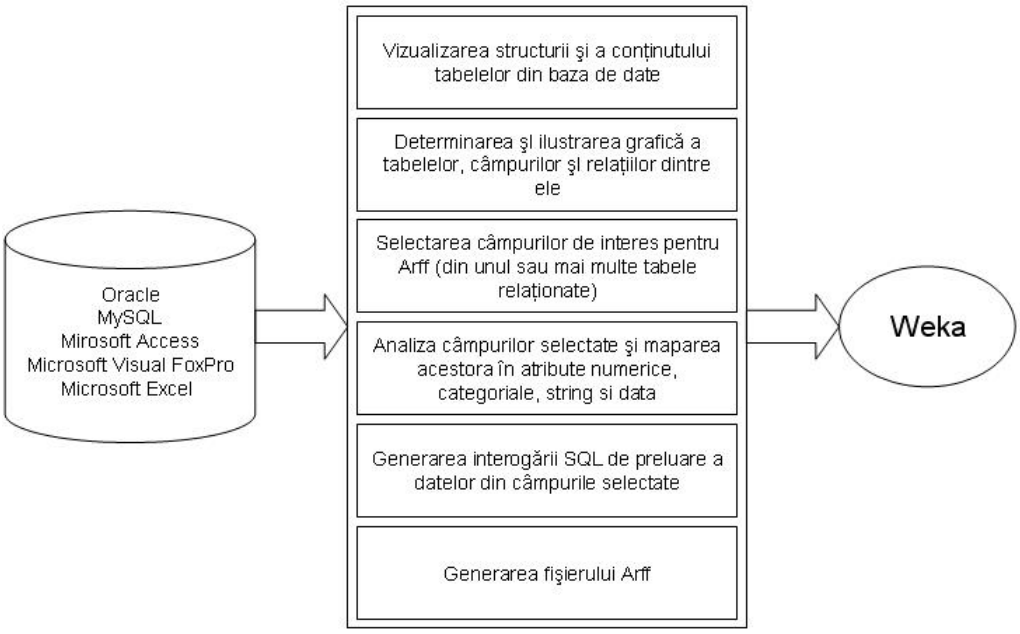

Fig. 2.6 - Funcţionalităţile lui *ARFF* Convertor

Arff Convertor permite vizualizarea structurii şi a conţinutului tabelelor din baza de date, ilustrarea grafică a tabelelor (un tabel va fi reprezentat printr-un control de tip *listbox* iar câmpurile lui ca şi item-uri în *listbox*) şi a relaţiilor dintre ele, selectarea cu ajutorul unor dublu-click-uri a câmpurilor de interes pentru generarea *ARFF*-ului, determinarea şi alegerea tipurilor de atribute care vor deveni câmpurile selectate, etc. Aplicaţia analizează tipul şi conţinutul câmpurilor selectate din diferitele baze de date la care se conectează şi sugerează tipul pe care îl vor avea câmpurile respective în format *ARFF*. De exemplu câmpurile de tip *tinyint, smallint, int, integer, bigint, real, double, float, decimal, numeric* dintr-o baza de date *MySQL* vor putea fi convertite în formatul *ARFF* fie în tip numeric fie într-un tip categorial. Variabilele de tip *varchar, char* din *MySQL* vor putea fi convertite fie în format *String*, fie în format categorial. Utilizatorul trebuie să aleagă tipul atributelor, iar dacă se determină că un atribut are mai puţin de 10 valori care se repetă atunci tipul implicit (sugerat) este categorial. Altfel poate fi numeric sau *String* în funcţie de tipul iniţial. Variabilele de tip boolean din toate bazele de date menţionate vor

putea fi convertite automat în variabile categoriale în *ARFF*, categoriile posibile fiind: *{Yes,No}, {Y,N}, {True, False}, {T,F},{Da,Nu},{D,N}*.

Câmpurile selectate sunt analizate şi dacă se constată că provin din tabele diferite se încearcă găsirea unei relaţii fizice între tabele. Dacă se determină o astfel de legătură, aceasta este utilizată pentru a genera o comandă SQL care va prelua datele din ambele tabele, (comandă ce va conţine clauza *left join*). Dacă nu se determină o legătură între tabele (este posibil ca legătura între ele să fie una logică) atunci se va afişa o interfaţă care va conţine tabelele selectate şi câmpurile lor, utilizatorul trebuind să menţioneze câmpurile de legătură. Dacă au fost selectate doar câmpuri dintr-un tabel, se generează comanda SQL care preia valorile câmpurilor selectate din tabelul ales. Aplicaţia dispune de interfeţe dinamice care se adaptează la conţinutul bazei de date deschise.

Pentru a demonstra buna funcţionare şi utilitatea soft-ului dezvoltat, autorul a preluat datele studenţilor de la Facultatea de Automatică şi Calculatoare, Universitatea Politehnica din Timişoara, anul universitar 2008-2009 din Sistemul informatic de gestiune a şcolarităţii, în format Microsoft Visual FoxPro. Datele preluate, sub forma a două tabele relaţionate în format *dbf*, au fost convertite cu ajutorul aplicaţiei propuse în format *ARFF*. Pentru a demonstra utilitatea acestei acţiuni s-a realizat o analiză statistică cu ajutorul lui *Weka* pe fiecare din atributele alese. Analiza statistică a permis extragerea unor concluzii interesante cu privire la nivelul de pregătire al studenţilor de la *AC*, cu privire la proveniența lor, numărul de credite obţinute după primul semestru, etc. Detalii despre acest studiu se găsesc în subcapitolul 2.10.4.

Detalii tehnice şi de implementare

Aplicaţia realizează următoarele prelucrări:

```
- explorarea structurii şi conţinutului bazei de date deschise
- selectarea tabelelor de interes
- afişarea dinamică a tabelelor, câmpurilor şi a legăturilor fizice dintre tabele
- selectarea câmpurilor de interes pentru ARFF

while (mai sunt câmpuri selectate){        //parcurgerea listei câmpurilor selectate
        Adaugă un combobox pentru acest câmp
        if (câmpul curent este numeric){
                - adaugă în combobox opţiunea de atribut numeric
                - determină categoriile posibile
                - adaugă în combobox opţiunea de atribut categorial
        }
        else
        if (câmpul curent este sir de caractere){
                - adaugă în combobox opţiunea de atribut de tip string
                - determină categoriile posibile
```

```
                - adaugă în combobox opţiunea de atribut categorial
        }
        else
        if (câmpul curent este boolean){
                - adaugă în combobox opţiunea de atribut categorial cu categoriile
posibile {Yes,No}, {Y,N}, {True,False}, {T,F},
{Da,Nu} şi {D,N}
        }
        else
        if (campul curent este de tip data)
                - adaugă în combobox opţiunea Data
}

- determinarea numărului de tabele din care provin câmpurile selectate

if (câmpurile selectate provin dintr-un singur tabel){
        - generează comandă select pentru preluarea datelor selectate
        - scrie datele în ARFF
}
else{
        determină relaţiile dintre tabelele de care aparţin câmpurile selectate
        if  (dacă au fost determinate relaţii)
                - generează o comandă SQL cu clauza tip left join care preia date
din tabele
        else{   //utilizatorul trebuie să menţionează legăturile
                 - generează o interfaţă cu tabele selectate şi toate câmpurile lor
                - utilizatorul menţionează legăturile dintre câmpuri
                - generează o comandă SQL cu clauza tip left join care preia date
din tabele
        }
```

Proiectul a fost realizat în mediul *Microsoft Visual Studio 2008*, *Visual C#* şi permite conectarea la baze de date *Oracle, MySql, Microsoft Access, Microsoft Excel, Microsoft Visual FoxPro* [FIL08]. Tehnologia utilizată pentru conectarea la aceste baze de date este *OLEDB* (Object Linking and Embedding, Database). Pentru conectarea la bazele de date menţionate au fost utilizaţi următorii furnizori: *MSDAORA.1* pentru *Oracle*, *MySQLProv* pentru *MySQL* (a fost instalat conectorul *MyOLEDB.3*), *Microsoft.Jet.OLEDB.4.0* pentru *Microsoft Access* şi *Microsoft Excel*, *VFPOLEDB.1* pentru *Microsoft Visual Fox Pro* (a fost instalat conectorul *Microsoft Visual FoxPro Oledb Provider*). Figura 2.7 prezintă interfaţa principală a aplicaţiei.

Tabelele din baza de date deschisă sunt afişate sub formă arborescentă într-un control *TreeView*. Declanşarea evenimentului *AfterSelect* al controlului *TreeView* (determinată de un click pe numele unui tabel) determină afişarea în controlul *ListView* a structurii tabelului selectat iar în controlul *DataGridView* a conţinutului tabelului.

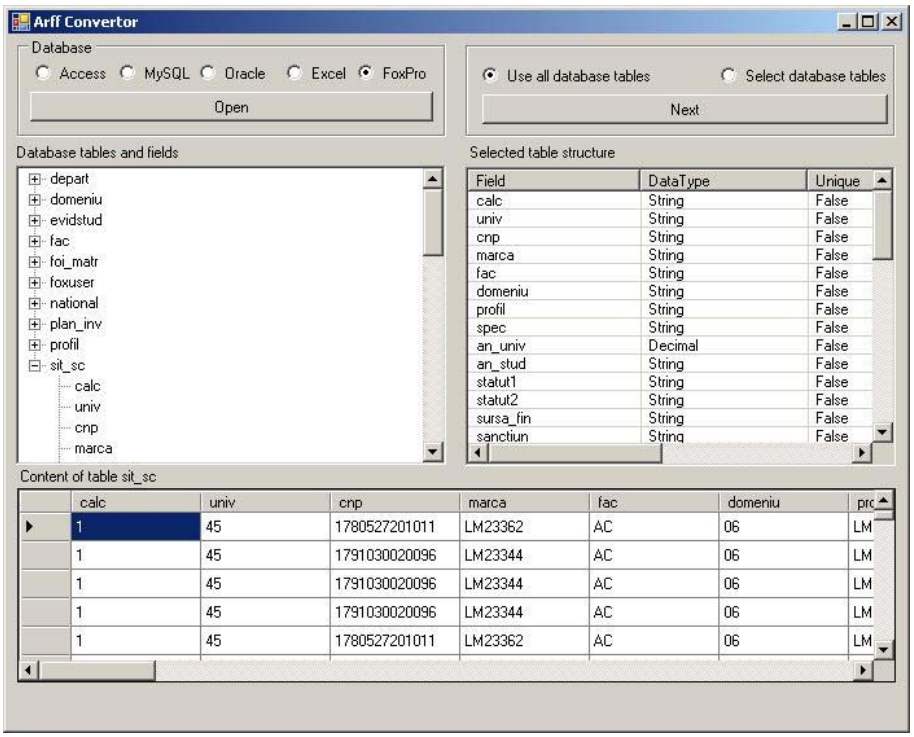

Fig. 2.7 - Fereastra principală a aplicaţiei

Obţinerea tuturor tabelelor din bazele de date (*Access*, *MySQL*, *Excel* şi *FoxPro*) a fost făcută utilizând clasa *CatalogClass* din spaţiul de nume *ADOX* şi interfaţa *Tables*.

```
CatalogClass cat = new CatalogClass();
cat.let_ActiveConnection(string_conectare);
Tables tb = cat.Tables;
```

În cazul bazei de date Oracle pentru obţinerea denumirilor tabelelor şi a câmpurilor din fiecare tabel au fost rulate următoarele doua interogări: *select table_name from use_tables* şi *select column_name from user_tab_columns where table_name = '" + tabel + "' order by column_id* [FIL09].

Obţinerea structurii tabelelor s-a realizat parcurgând colecţia de coloane a *DataTable*-ului şi accesând proprietăţile: *ColumnName, DataType, Unique, AlowDBNull, AutoIncrement, DefaultValue, MaxLength*.

În situaţia în care numărul de tabele din baza de date este foarte mare se pot alege doar tabelele de interes în generarea *ARFF*-ului (vezi figura 2.8).

Toate tabelele sau doar tabelele de interes (cele care au fost selectate prin interfaţa din figura 2.8) sunt afişate sub forma de *label*-uri iar câmpurile lor sunt

afişate în *listbox*-uri (vezi figura 2.9). Relaţiile dintre tabele sunt ilustrate prin linii de culori generate aleator. La producerea evenimentului *DoubleClick* al *ListBox*-ului asociat unui tabel, în *ListBox*-ul *Fields in ARFF* va fi adăugată denumirea tabelului apoi punct şi apoi denumirea câmpului pe care s-a făcut dublu click-ul.

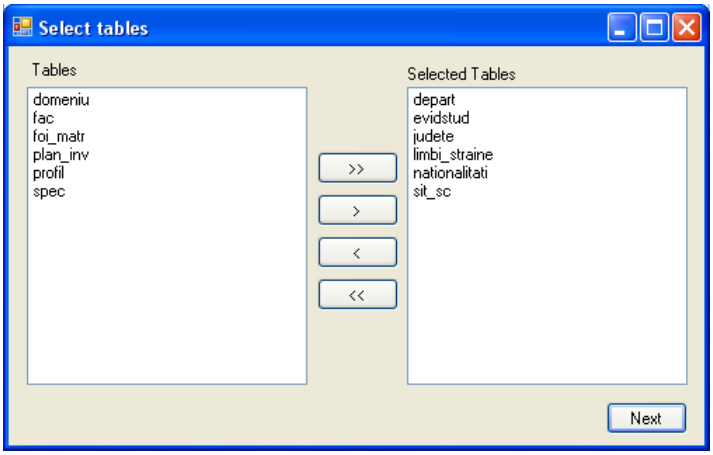

Fig. 2.8 - Selectarea tabelelor de interes

Atât label-urile cat şi *listbox*-urile au fost amplasate într-un control panel şi au fost create dinamic, din cod, în acord cu tabelele selectate din baza de date deschisă.

Ca şi structuri ajutătoare la trasarea legăturilor dintre tabele, s-a utilizat un *ArrayList*, format dintr-o colecţie de obiecte, fiecare obiect corespunzând unui tabel. Un obiect din *ArrayList* este de clasă *clsTabel* şi are proprietăţi care păstrează numele tabelului, coordonatele colţului stânga sus, numărul de câmpuri din tabel, denumirile câmpurilor.

Pentru determinarea legăturilor dintre tabele, s-a utilizat în cazul bazei de date Access, interfaţa *Tables*, clasa *CatalogClass* din spaţiul de nume *ADOX*. Au fost parcurse toate tabelele selectate care făceau parte din colecţia *Tables* a obiectului *CatalogClass*. În cadrul fiecărui *Tabel* au fost parcurse toate *Key*-urile, determinând tabelele de legătură cu ajutorul proprietăţii *Key.RealatedTable*. Pentru determinarea coloanelor de legătură s-au folosit *Key.Columns[i].Name* şi *k.Columns[i].RelatedColumn*. Relaţiile determinate au fost scrise într-un control *listbox*, dar a fost implementată şi facilitatea de trasare grafică a acestora (vezi figura 2.9).

Trasarea liniilor care ilustrează legăturile dintre tabele s-a implementat cu ajutorul unei metode care determină indexul coloanei primită ca şi parametru. Această informaţie serveşte la determinarea din dreptul cărui Item din cadrul Listbox-ului trebuie să pornească linia.

O altă metodă implementată returnează al câtelea tabel de pe interfaţă este tabelul primit ca şi parametru. Această metodă a fost utilizată pentru a determina în ce poziţii se află tabele pe care trebuie să le legăm, unul faţă de celălalt. Dacă indexul tabelului 1 modulo 5 este egal cu indexul tabelului 2 modulo 5 înseamnă că

tabelele sunt pe aceeaşi coloană (5 este numărul de tabele de pe un rând), dacă indexul tabelului 1 % 5 > indexul tabelului 2 %5 însemnă că tabelul 1 este în stânga tabelului 2, altfel înseamnă că tabelul 2 este în stânga tabelului 1.

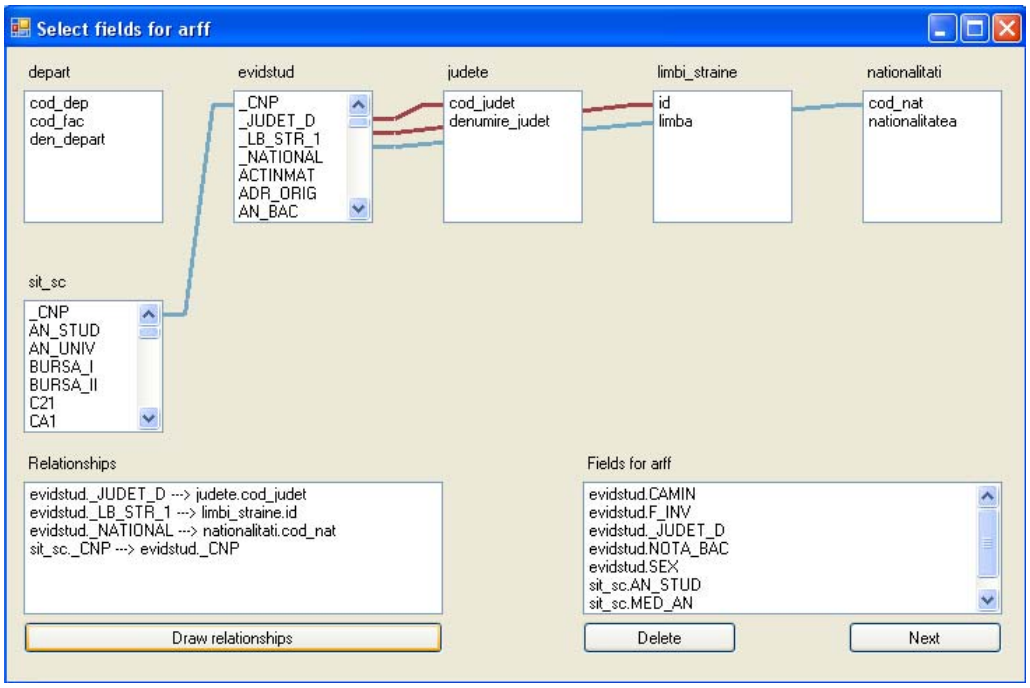

Fig.2.9 - Selectarea câmpurilor din tabele diferite pentru *ARFF*

Poziţiile relative ale tabelelor contează pentru a şti dacă linia trasata pleacă de pe partea stângă a listboxului corespunzător tabelului 1 sau de pe partea dreapta a lui, repectiv dacă ajunge pe partea stângă sau pe partea dreapta a listboxului corespunzător tabelului 2.

Stabilirea tipului pe care îl vor avea câmpurile care au fost selectate, în formatul *ARFF* (numerice, categoriale, string sau date) se realizează cu ajutorul unor controale de tip ComboBox (se adaugă dinamic câte un control pentru fiecare câmp selectat) (vezi figura 2.10). Conţinutul combobox - urilor este următorul: în cazul unui câmp de tip şir de caractere, în combobox - ul corespunzător se va adăuga un item cu denumirea atributului şi tipul string şi un alt item cu denumirea atributului şi categoriile determinate prin analiza conţinutului câmpului. În cazul unui câmp de tip numeric (*Int32, Double, Decimal, etc*) combobox-ul corespunzător va conţine un atribut numeric şi un atribut categorial cu toate valorile atributului (nu vor apărea duplicate).

În cazul câmpurilor de tip boolean, ele vor deveni atribute categoriale cu valorile posibile *{Yes,No}, {Y,N}, {True, False}, {T,F},{Da,Nu},{D,N}*. Dacă pentru un atribut de tip şir de caractere sau numeric se determină că are mai puţin de 10 categorii, atunci valoarea implicită a atributului va fi cea categorială. Dacă se determină că are mai mult de 10 categorii valoarea implicită va fi string sau numeric. Câmpurile de tip DateTime pot deveni doar atribute Date.

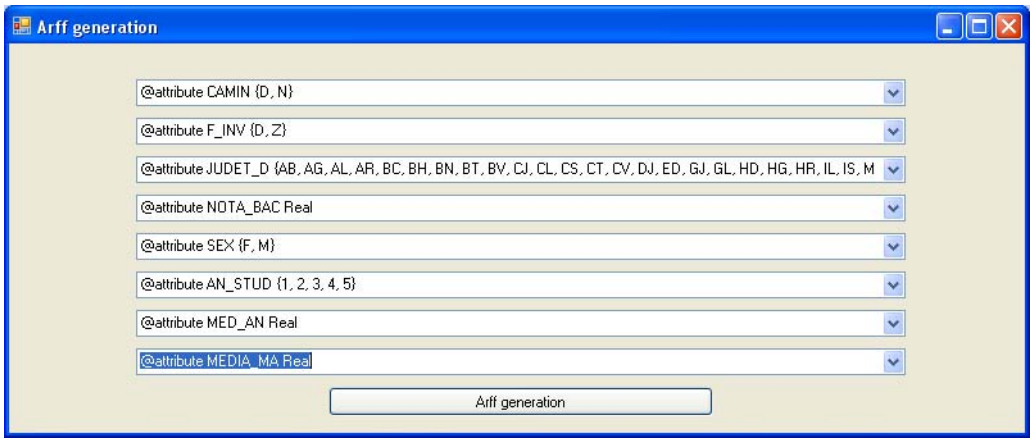

Fig. 2.10 - Stabilirea tipului atributelor

Înainte de generarea *ARFF*-ului se va analiza dacă câmpurile care au fost alese să devină atribute în *ARFF* aparţin unui singur tabel sau aparţin mai multor tabele. Dacă aceste câmpuri fac parte din tabele diferite, înseamnă că acele tabele sunt relaţionate şi interogarea care se generează va fi de tipul left – join. Se analizează dacă există o legătură fizică între tabelele selectate, iar dacă aceasta nu există, i se solicită utilizatorului să menţioneze tabelul părinte, tabelul copil şi câmpurile de legătură (vezi figura 2.11). Relaţia (fie introdusă de utilizator, fie determinată automat) este utilizată pentru a genera interogarea ce va prelua datele din câmpurile menţionate şi care este de tip left join. Un exemplu de interogare generată este următorul:

```
select evidstud.f_inv, evidstud.st_civ,evidstud.camin, evidstud.std_abs, evidstud.sex
,evidstud.judet_d,sit_sc.an_stud,sit_sc.co1,sit_sc.r11,sit_sc.co2,sit_sc.r22,sit_sc.c21,sit_sc.r21 from (evidstud  left join sit_sc on evidstud.cnp=sit_sc.cnp)
```

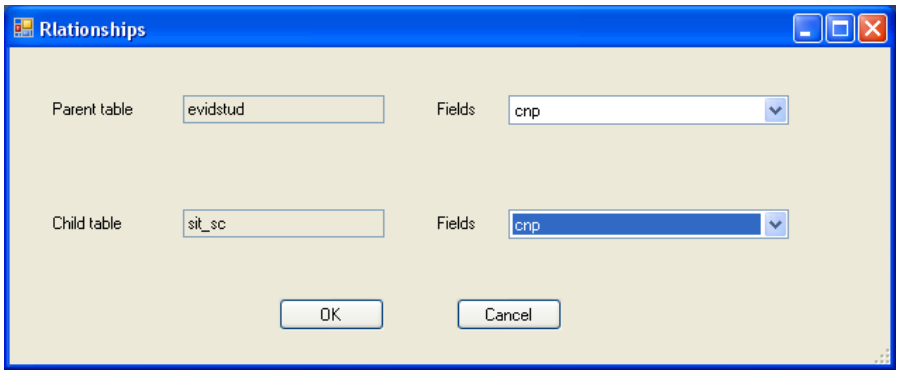

Fig. 2.11 – Specificarea legăturilor dintre câmpuri

În figura 2.12 poate fi vizualizat un exemplu de fişier *ARFF* generat.

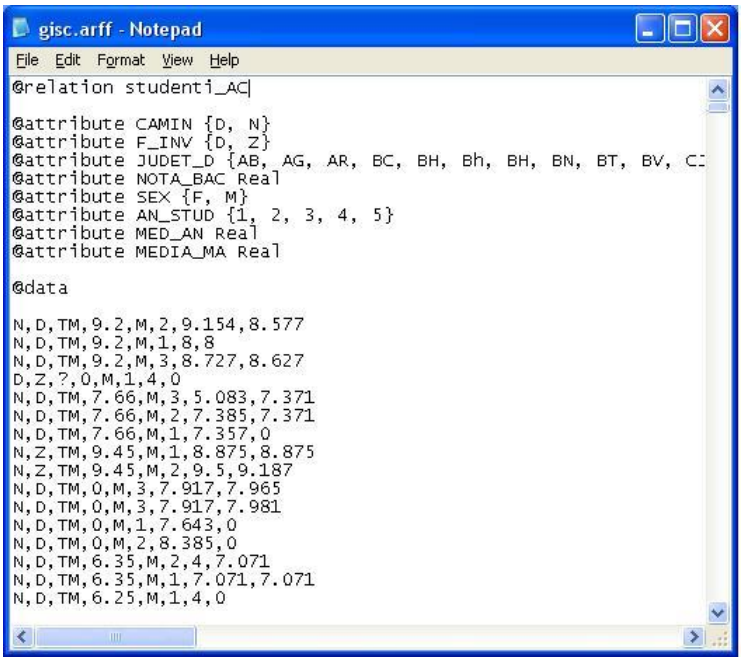

Fig. 2.12 - Fişierul *ARFF* generat

2.3.4. Studiu de caz: Analiza datelor civice şi şcolare ale studenţilor

Pentru a demonstra utilitatea aplicaţiei *Arff Convertor*, autorul a realizat un studiu de caz cu ajutorul aplicaţiei dezvoltate şi cu *Weka* pe datele studenţilor înscrişi în anul universitar 2008-2009 la Facultatea de Automatică şi Calculatoare din cadrul Universităţii Politehnica din Timişoara. Bazele de date pe care a fost făcut studiul au fost preluate din softul de gestiune al şcolarităţii utilizat de către secretariatul facultăţii, în format *dbf*. Au fost preluate două tabele: *evidstud.dbf* – care conţine datele civice ale studenţilor şi are 70 de câmpuri şi *sit_sc.dbf* - care conţine datele şcolare ale studenţilor şi are 43 de câmpuri. Cele două tabele sunt relaţionate prin *cnp*-ul studenţilor, relaţia fiind de unul la mai mulţi (unui rând din *evistud.dbf*, îi corespund mai multe rânduri din *sit_sc.dbf*). Un student are asociate în fişierul *sit_sc.dbf* un număr de rânduri egal cu numărul de ani universitari în care a fost student. Într-o primă fază au fost selectaţi în fişierul *sit_sc.dbf* doar studenţii din anul universitar 2008-2009 (din toţi anii de studiu şi de la toate specializările şi ciclurile atât batchelor cât şi master). În urma selecţiei au rezultat un număr de 2112 studenţi. În a doua fază a fost folosită aplicaţia *Arff Convertor*, pentru a genera un fişier *ARFF* care conţine doar atributele de interes: din cele 113 de câmpuri care sunt în tabela evidstud şi sit_sc s-au ales pentru analiză un număr de 17 câmpuri, 11 din fişierul *evidstud.dbf* şi 6 din fişierul *sit_sc.dbf*.

Din fişierul *evidstud.dbf* au fost alese pentru analiză următoarele: forma de învăţământ starea civilă, cămin, sex, judeţul de domiciliu, nota bacalaureat, media multianuală în liceu, media de admitere, cetăţenia, limba străină 1, limba străină 2.

Din fişierul *sit_sc.dbf* au fost alese pentru analiză următoarele câmpuri: numărul de credite la sfârşitul semestrului 1, numărul de restanţe la sfârşitul semestrului 1, numărul de credite la sfârşitul semestrului 2, numărul de restanţe la sfârşitul semestrului 2, media anuală, media multianuală. Selectarea câmpurilor de interes din cele două tabele *dbf* a fost făcută cu ajutorul aplicaţiei *Arff Convertor*.

Studiul a continuat prin realizarea cu ajutorul lui *Weka* a unei analize statistice pe fiecare din atributele alese[WIT05].

Rezultatele obţinute se referă la studenţii de la *AC* din 2008-2009 şi sunt următoarele:

- 96 % din studenţii (2027 studenţi) de la *AC* urmează învăţământul la zi şi doar 4 % (85 studenţi) învăţământul la distanţă
- Majoritatea studenţilor 89 % (1883 studenţi) sunt necăsătoriţi, 1% sunt căsătoriţi (22 de studenţi), 1.5 % sunt divorţaţi (33 studenţi) şi 0.15 % (3 studenţi) sunt văduvi
- 64 % din studenţi stau în cămin (vezi figura 2.13)

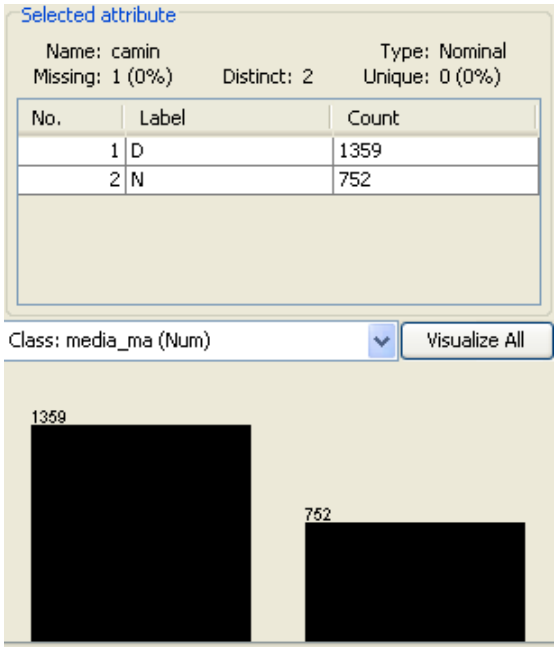

Fig. 2.13 - Numărul de studenţi care stau în cămin

- 70 % din studenţi sunt băieţi
- Cei mai mulţi studenţi de la Facultatea de Automatică şi Calculatoare sunt din Timiş 39 %, pe locul 2 se află Hunedoara 11.3 %, apoi Caraş- Severin 10.1 %, Arad 9.3 %, Bihor 5.9 %, Gorj 5.5 %, Mehedinţi 4.7%, etc. (vezi figura 2.14)

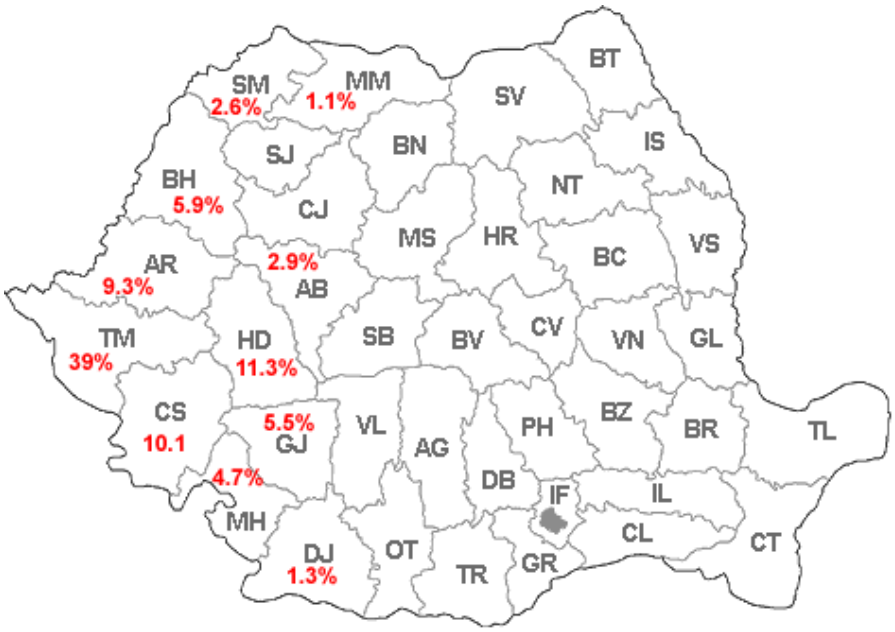

Fig. 2.14 - Domiciliul studenţilor de la Facultatea de Automatică şi Calculatoare

- Nota la examenul de bacalaureat este de tip numeric şi a fost discretizată cu ajutorul lui *Weka* în 10 intervale (vezi figura 2.15) . După cum se observă 74.5 % din studenţi au nota la bacalaureat cuprinsă între 9 şi 10, 15.5 % din studenţi au nota între 8 şi 9, 2.5 % din studenţi au nota între 7 şi 8 şi doar 0.7 % din studenţi au nota între 6 şi 7. În schimb pentru 6.6 % din studenţi nu a fost introdusă nota la bacalaureat în baza de date.

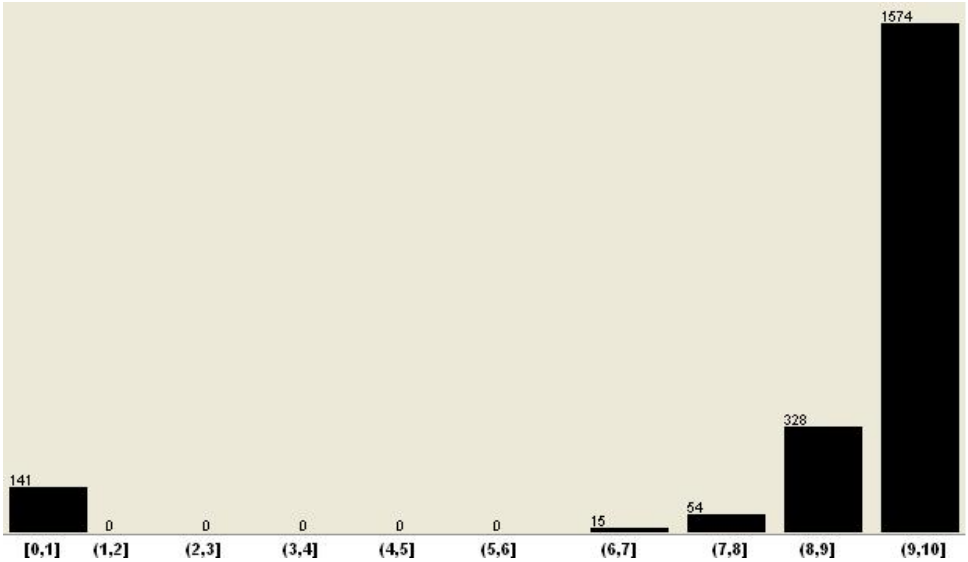

Fig. 2.15 - Nota la bacalaureat

- Media de admitere este destul de ridicată, dar scăzută faţă de cea de la bacalaureat: 40 % dintre studenţi au intrat cu medii de admitere între 9 şi 10, 30% cu medii între 8 şi 9, 13.8 % cu medii între 7 şi 8, 8.2 % cu medii între 6 şi 7 şi 2.6 % cu medii între 5 şi 6. Pentru 5.6 % dintre studenţi nu a fost completat câmpul media de admitere în baza de date

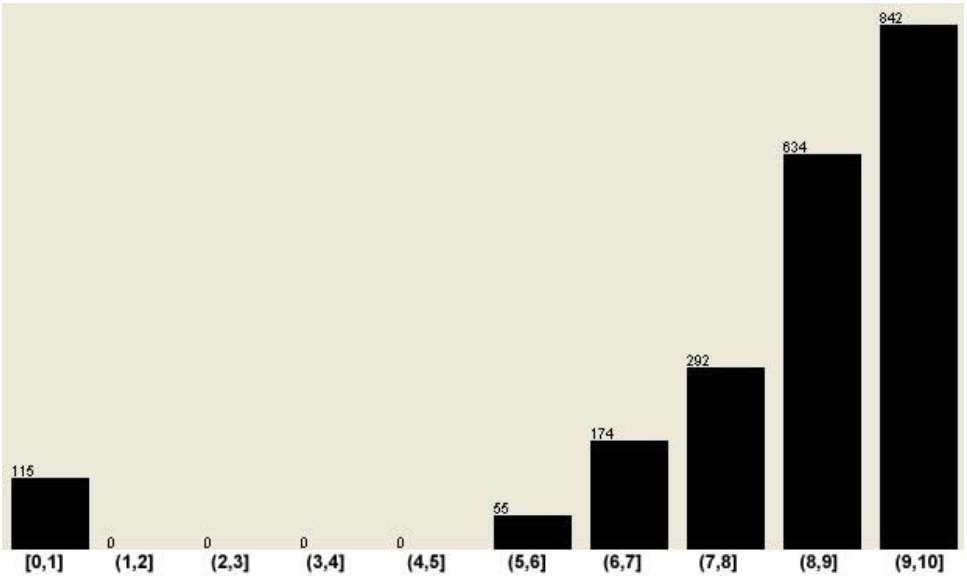

Fig. 2.16 - Media de admitere

- Media multianuală în liceu: 42.2 % de studenţi au avut media multianuală cuprinsă între 9 şi 10, 21.3 % au avut media intre 8 si 9, 8.71 % au avut media multianuală între 7 şi 8, 2.51 % au avut media între 6 şi 7, 1 % au avut media între 5 şi 6 şi pentru 1 % a fost introdus greşit că au avut media multianuală între 4 şi 5.

- Din punct de vedere al cetăţeniei 99.2 % din studenţi au cetăţenie română, dar există şi 6 studenţi moldoveni, 7 slovaci, un albanez, un sârb
- Prima limbă studiată în liceu de studenţi este engleza pentru 56.9 % studenţi, franceza pentru 13.4 % studenţi, germana pentru 4.1 % şi rusa pentru 0.2 % studenţi
- Limba străină 2, este franceză pentru 41 % studenţi, engleză pentru 15.3 %, germană pentru 14.4 %
- Numărul de credite la sfârşitul semestrului 1 este cuprins între 0 şi 32 (vezi figura 2.17). În medie studenţii au strâns 25.4 credite la sfârşitul semestrului 1. Valoarea a fost discretizată, împărţită în 2 intervale şi analizată. (89.9 % din studenţi au obţinut peste 16 credite, restul au obţinut sub 16 credite în primul semestru şi sunt în pericol de exmatriculare)

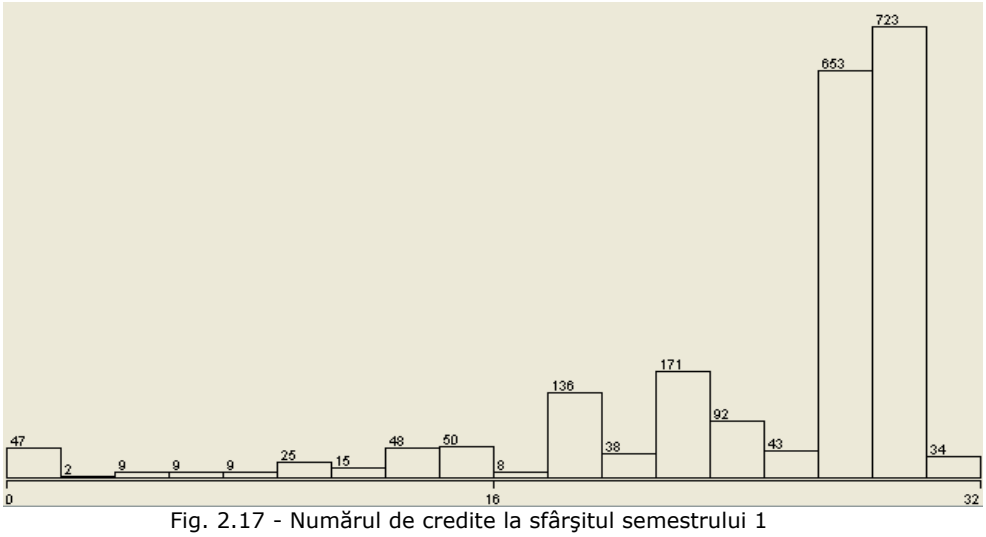

Fig. 2.17 - Numărul de credite la sfârşitul semestrului 1

- Numărul de credite la sfârşitul semestrului 2 a fost introdus doar pentru 946 de studenţi. Dintre aceştia 82.1 % au obţinut peste 15 credite
- Numărul de restanţe la sfârşitul semestrului 1 este 0 pentru 67.1 % studenţi. Numărul de restanţe la sfârşitul semestrului 2 este 0 pentru 71.26 % studenţi dar nu este un număr credibil pentru că aici nu au fost completate date despre creditele restante şi numărul de restanţe pentru mulţi studenţi

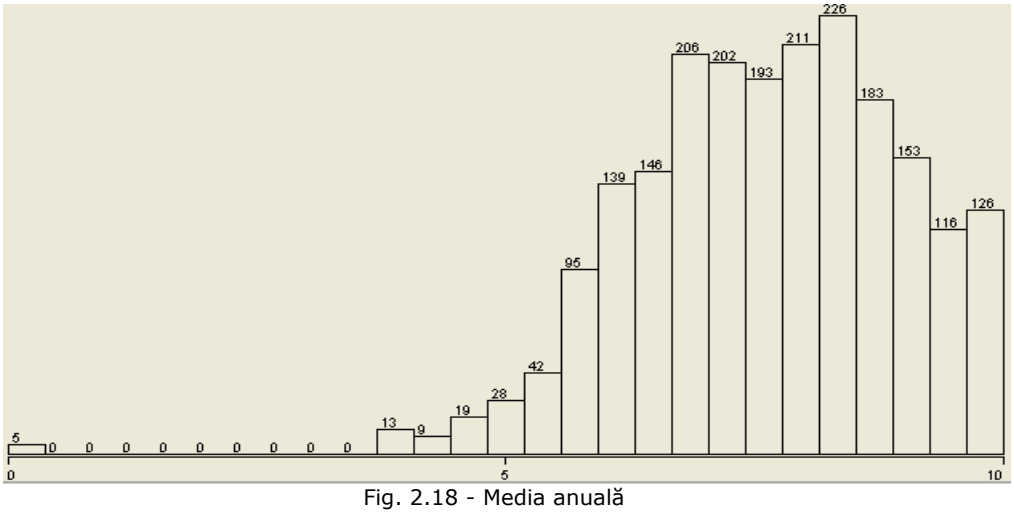

Fig. 2.18 - Media anuală

- Media anuală are o distribuţie gausiană, aspect care poate fi vizualizat în figura 2.18. Valoarea medie a mediei anuale este de 7.6 cu o deviaţie standard de 1.3
- Media multianuală are şi ea o distribuţie gausiană, valoarea medie este de 7.5 cu o deviaţie standard de 1.5.

2.4. Concluzii

Preprocesarea datelor este etapă care consumă cel mai mult timp în studiile de data mining [ZHA05].

Algoritmii de data mining extrag cunoştinţe din date. Succesul extragerii acestor cunoştinţe depinde de calitatea datelor pe care operează aceşti algoritmi. Dacă datele sunt nepotrivite sau conţin informaţie nerelevantă, atunci algoritmii pot produce rezultate de o calitate mai slabă sau pot să nu descopere nimic util în date. De aceea etapa de preprocesare a datelor este o etapă deosebit de importantă.

Când setul de date este prea mare, este posibil să nu poată fi folosiţi anumiţi algoritmi de data mining. Pentru a obţine mai puţine date se poate reduce numărul instanţelor (prin agregare sau eşantionare) respectiv numărul atributelor (prin tehnici de reducere a dimensionalităţii, selectare de submulţimi de caracteristici, creare de caracteristici, etc).

Datele iniţiale pot să fie nepotrivite cu anumiţi algoritmi, în acest sens se impune o transformare a acestora. Transformarea datelor se poate realiza prin discretizare, operaţie prin care atributele continue sunt transformate în atribute categoriale, prin binarizare, prin aplicarea unor funcţii matematice simple respectiv prin normalizare.

Weka este un program de data mining forte puternic, el oferind foarte multe facilităţi de preprocesare (pentru discretizare, reducerea zgomotului, selectarea de submulţimi de caracteristici, etc) dar şi numeroşi algoritmi de clasificare, regresie, clustering, reguli de asociere, etc. Înainte de preprocesarea datelor cu *Weka* prin tehnici precum cele menţionate este necesară parcurgerea altor etape ale preprocesării care constau în curăţarea datelor prin eliminarea duplicatelor şi a datelor expirate, corectarea sau eliminarea datelor introduse greşit, dar şi conversia datelor în formatul de intrare cerut de *Weka*. Deşi *Weka* permite preluarea datelor din baze de date cu ajutorul unor tehnologii, utilizarea acestora cere utilizatorului să facă configurări în Windows, în *Weka* şi să cunoască limbajul SQL pentru a prelua datele. Aceste cerinţe restrâng aria utilizatorilor de *Weka* la specialişti în domeniul calculatoarelor, deşi *Weka* ar putea fi utilizat de către oricine doreşte să analizeze date cărora le cunoaşte înţelesul. Ca o soluţie la această problemă autorul a conceput şi dezvoltat o aplicaţie care se poate lega foarte uşor la baze de date şi poate genera un fişier de intrare pentru *Weka*, în format *ARFF* cu datele din una sau mai multe tabele relaţionate, fără a fi necesare cunoştinţe de *SQL* şi fără să fie necesară realizarea de configurări în sistemul de operare sau în *Weka*. Aplicaţia dezvoltată se numeşte *Arff Convertor* şi permite conectarea la baze de date *Oracle, MySql, Microsoft Access, Microsoft Excel, Microsoft Visual FoxPro*. Printre facilităţile aplicaţiei se numără următoarele: selectarea câmpurilor de interes din unul sau mai multe tabele relaţionate (fără a fi nevoie de cunoaşterea limbajului SQL, prin dubluclick pe numele câmpului), vizualizarea grafică prin intermediul unor interfeţe dinamice a structurii, relaţiilor dintre tabele şi a conţinutului acestor baze de date, precum şi generarea de fişiere de intrare pentru *Weka* (fişiere *ARFF*) care conţin datele din unul sau mai multe tabele ale bazelor de date menţionate. Aplicaţia determină atributele numerice, string şi date şi propune atribute categoriale. Modul de vizualizare al structurii şi conţinutului tabelelor, precum şi al relaţiilor dintre ele,

facilitează înţelegerea datelor, lucru care este foarte important pentru procesul de data mining [BER09].

Pentru validarea aplicaţiei *Arff Convertor*, autorul a realizat cu aplicaţia dezvoltată şi cu *Weka* un studiu de caz pe o bază de date Microsoft Visual Fox Pro, cu datele a 2100 de studenţi, care a fost preluată din aplicaţia de gestiune a şcolarităţii care funcţionează în cadrul Universităţii "Politehnica" din Timişoara, facultatea Automatică şi Calculatoare. Studiul a constat în realizarea unei analize statistice a datelor studenţilor. O parte din concluziile desprinse în urma analizei sunt:

- din Timiş provin doar 39% dintre studenţii de la AC restul sunt din judeţele învecinate
- 64 % din studenţii de la AC stau în cămin
- 70 % sunt băieţi
- 74.5 % au media la bacalaureat între 9 şi 10
- 40 % dintre studenţi au media de admitere între 9 şi 10
- În medie studenţii obţin 25.4 credite la sfârşitul primului semestru

Studiul a dovedit că unealta propusă funcţionează bine şi este utilă, simplificând mult legătura dintre bazele de date sursă şi programul *Weka*, foarte utilizat în data mining.

Contribuţiile care au fost aduse prin intermediul acestui capitol sunt:

- Realizarea unei sinteze ample şi riguroase cu privire la principalele tehnici de preprocesare a datelor, bazată pe studiul a 30 de referinţe
- Îmbunătăţirea tehnologiei de preprocesare a datelor prin concepţia şi dezvoltarea unui instrument software permite conversia datelor din 5 formate de baze de date în formatul de date folosit de *Weka* (*ARFF*). Instrumentul propus dispune de interfeţe dinamice, în acord cu datele care se prelucrează, iar utilizarea lui este simplă şi nu necesită realizarea de configurări în sistemul de operare, în *Weka* sau cunoştinţe de *SQL*
- Realizarea unei analize a datelor studenţilor de la facultatea de Automatică şi Calculatoare din anul universitar 2008-2009 cu ajutorul lui *Arff Convertor* şi *Weka*, analiză care oferă o imagine de ansamblu asupra provenienţei studenţilor, pregătirii lor, etc.

3. Clasificarea datelor cu ajutorul algoritmului Ant Colony Optimization

3.1. Noţiuni introductive

Algoritmul *Ant Colony Optimization* face parte din rândul algoritmilor bioinspiraţi, el este utilizat la optimizarea funcţiilor din anul 1996, a fost propus de Dorigo et all [DOR96], iar utilizarea lui pentru descoperirea regulilor de clasificare a fost propusă de Parpinelli et all în anul 2001 [PAR01].

Autorii au demonstrat că acest algoritm poate descoperi în date reguli a căror acurateţe este comparabilă cu cea a unor algoritmi consideraţi clasici, chiar şi fără niciun fel de încercare de optimizare a parametrilor algoritmului. Din Anul 2001 şi până în prezent au fost făcute multe studii cu acest algoritm şi au fost aduse multe îmbunătăţiri în ceea ce priveşte calculul funcţiei euristice, modul de actualizare al feromonului, modul de selectare a termenilor pentru reguli, etc. Cu toate acestea, la fel ca în cazul tuturor algoritmilor de clasificare, nu a fost găsită o variantă care să fie mai bună decât altele pe orice set de date. Probleme precum alegerea parametrilor pentru algoritm în aşa fel încât să se obţină cele mai bune rezultate posibile precum şi îmbunătăţirea procesărilor algoritmului sunt deschise în continuare.

Algoritmul propus în [PAR01] şi [PAR02] a fost iniţial implementat în limbajul C şi testat pe diverse seturi de date. Ulterior algoritmul a fost implementat în Java şi distribuit sub forma unui sistem Open Source numit *Ant Miner*. Ultima versiune este versiunea 1.2.1, dezvoltată în anul 2007 de către Fernando Meyer în colaborare cu Parpinelli [PAR02].

În cadrul acestui capitol au fost tratate următoarele probleme:

- A fost realizată o prezentare detaliată a algoritmului, bazată atât pe documentaţia existentă cât şi pe studiul codului algoritmului.
- Au fost studiaţi parametrii algoritmului rulând câte 64 de configuraţii pe parametri pe fiecare din cele 10 seturi de date reale, obţinute de pe *UCI Machine Learning Repository.* Rezultatele obţinute au fost analizate atât empiric, comparativ cât şi matematic construind modele de regresie şi matrice de corelare. Pe baza analizei făcute a fost emis un set de recomandări cu privire la alegerea acestor parametri
- A fost realizată o sinteză a îmbunătăţirilor aduse algoritmului din anul 2001 şi până în prezent
- Au fost aduse 4 îmbunătăţiri algoritmului care au fost implementate în aplicaţia open source *Ant Miner*. Aplicaţia care conţine algoritmul îmbunătăţit a fost numită *Ant-r-Miner*. S-a demonstrat că îmbunătăţirile aduse conduc la o mai bună performanţă utilizând date reale.

3.2. Prezentarea algoritmului

Algoritmul *Ant Colony Optimization* este inspirat din comportamentul furnicilor. Acestea, deşi sunt aproape oarbe, reuşesc să găsească întotdeauna drumul cel mai scurt dintre muşuroi şi sursa de hrană. Studiile au arătat că acest lucru este posibil datorită unei substanţe, numită feromon cu care furnicile marchează traseul pe care îl urmează. O furnică lasă o urmă de feromon pe unde circulă şi este influenţată de mirosul de feromon lăsat de alte furnici [DOR04]. Feromonul este supus, odată cu trecerea timpului, fenomenului de evaporare. Studiile au arătat că dacă pe traseul dintre muşuroi şi sursa de hrană se plasează un obstacol, după un timp, toate furnicile îl vor ocoli pe drumul cel mai scurt. Explicaţia este următoarea: când primele furnici ajung la obstacol ele aleg să îl ocolească prin partea stângă sau prin partea dreaptă cu o probabilitate egală. La întoarcere, mirosul de feromon de pe traseul mai lung este mai slab decât cel de pe traseul mai scurt, datorită fenomenului de evaporare. Prin urmare, mai multe furnici vor urma calea mai scurtă. După un anumit număr de ture efectuate între cele două destinaţii, toate furnicile se vor plimba pe traseul cel mai scurt [MAR11].

Regulile descoperite sunt de forma:

IF <conditions> THEN <class>

Partea condiţiilor se numeşte antecedentul regulii şi conţine o combinaţie logică a atributelor predictoare în forma $T_{1,3}$ *şi* $T_{2,1}$ *şi* ... (sau *atributul1=valoarea3* şi *atributul2=valoarea1*). Clasa reprezintă consecventul regulii şi este de forma *atribut_clasa=valoare2*.

Pentru antrenarea şi testarea algoritmului se foloseşte tehnica numită *cross validation*. Setul iniţial de date se împarte într-un număr de subseturi (mulţimi) pe care utilizatorul îl doreşte şi îl specifică prin parametrul *Folds number*. Mulţimile obţinute au un număr aproximativ egal de elemente. Algoritmul va fi rulat de un număr de ori egal cu numărul de mulţimi. La fiecare rulare una din mulţimi va fi folosită pentru test şi celelalte pentru antrenare.

Parametrii de intrare ai algoritmului sunt numărul de subseturi în care se va împărţi setul de date (*NS*), numărul de furnici (*NF*), numărul minim de instanţe acoperite de orice regulă (*NMIR*), numărul maxim de instanţe care pot rămâne neacoperite de reguli (*NMINR*), numărul de reguli pentru convergenţă (*NC*), numărul de iteraţii (*NI*). Înainte de rularea algoritmului aceşti parametri de intrare trebuie să primească valori.

Algoritmul în pseudocod:

```
- alege aleator pentru fiecare instanţă din setul de date în care din cele NS subseturi face parte
ns=1;                    //index pentru numărul de subseturi
sumaAP =0;
while ns <= NS   // NS= numărul de subseturi
        mulţimea de test= subsetul ns
        mulţimea de antrenare =subseturile diferite de subsetul ns

        lista regulilor = 0 // iniţial lista regulilor este goală
```

```
        while (numărul de instanţe din mulţimea de antrenare > NMINR)
              f=1;  //indexul furnicii
              c=1; //index folosit pentru testul de convergenta
              - iniţializează toţi T_{i,j} cu aceeaşi cantitate de feromon
              repeta
                   - furnica Ant_f începe cu o regula goală şi construieşte regula adăugând rând
                     pe rând câte un  T_{i,j} la regulă
                   - calculează clasa regulii construite
                   - aplică elagajul asupra regulii construite (rule pruning)
                   - actualizează cantitatea de feromon în toţi T_{i,j}
                   if ( regula construită este identică cu regula precedentă)
                              atunci c=c+1
                   altfel c=1
                   end if
                   f=f+1; // se trece la următoarea furnică

              până când (f>=NF sau (c>=NC)
               - selectează cea mai bună regulă construita de toate furnicile
               - adaugă regula găsită la lista regulilor descoperite
               mulţimea_de_antrenare = mulţimea_de_antrenare - instanţele acoperite de
                                       regula descoperită
        end while

          - adaugă la sfârşitul listei de reguli o regulă care nu are nici o condiţie în antecedent,
          şi care are în  consecvent clasa majoritară dintre clasele mulţimii de antrenament

        t=0; //index pentru instanţele din mulţimea de testare
        while (t < nr_inst_mulţimea _testare)
                     nr=0; //index pentru lista de reguli
                     while (nr<nr_reguli)
                           if (antecedent regulii acoperă instanţa(t))
                                  dacă consecvent regula coincide cu clasa instanţei
                                        nr_clasificate_corect= nr_clasificate_corect+1
                                  altfel
                                        nr_clasificate_incorect= nr_clasificate_incorect+1
                                  end if
                     end while
                     t=t+1;
          end while

          acurateţea_predicţiei=  nr_clasificate_corect / nr_inst_mulţimea_testare
          sumaAP = sumaAP + acurateţea_predicţiei;
          ns=ns+1
end while

medieAP= sumaAP / nr_mulţimi
```

Împărţirea setului de date în NS subseturi

Împărţirea instanţelor în subseturi (mulţimi) se face parcurgând toate instanţele şi generând pentru fiecare instanţă un număr aleator de la 0 şi până la numărul de subseturi -1. Numărul generat reprezintă subsetul de care aparţine instanţa. Întrucât toate subseturile trebuie să aibă un număr aproximativ egal de elemente, iniţial se calculează cât este numărul de elemente maxim pe care trebuie să-l aibă fiecare subset= numărul de elemente din setul de date împărţit la numărul de subseturi +1. Dacă restul acestei împărţiri este 0 atunci se mai scade un element din dimensiunea grupului. O anumită instanţă este atribuită unei mulţimi obţinută prin generarea aleatoare a numărului dacă acea mulţime mai are loc sa primească instanţe (nu a depăşit deja numărul de instanţe permis). Dacă l-a depăşit, reia generarea aleatoare.

Lista de reguli

Iniţial lista regulilor este goală. Din regulile pe care le-au descoperit maxim *NF* furnici se alege cea mai bună şi se adaugă listei de reguli. Din mulţimea de antrenare se elimină instanţele acoperite de regula adăugată listei de reguli. Procesul continuă până când în mulţimea de antrenare rămân un număr de instanţe neacoperite de nici o regulă mai mic decât valoarea *NMINR* stabilită de utilizator.

Construcţia regulii

O furnică porneşte de la o regulă fără antecedent şi consecvent. Furnica construieşte întâi antecedentul regulii şi apoi alege valoarea consecventului. Construirea antecedentului se face adăugând câte un termen ($T_{i,j}$) la un moment dat. Un termen $T_{i,j}$ este o pereche atribut-valoare. Termenul $T_{i,j}$ are semnificaţia: atributul i are valoarea j. Alegerea termenului care va fi adăugat la regulă depinde de cantitatea de feromon asociata lui $T_{i,j}$ şi de o funcţie euristică. Fiecare termen $T_{i,j}$ are asociată o cantitate de feromon. Termenii din regula descoperită de furnică corespund traseului urmat de aceasta. Iniţial toţi $T_{i,j}$ din setul de date (toate perechile atribut valoare) au asociată o cantitate egală de feromon (cea din relaţia 3.1). După construcţia fiecărei reguli termenii folosiţi în regulă primesc o cantitate de feromon mai mare decât restul.

$$\tau_{ij}(t=0)=\frac{1}{\sum_{i=1}^{a} b_i} \tag{3.1}$$

O furnică continuă să adauge termeni la regula curentă până când, orice termen ar adăuga la regulă, ar face ca regula să acopere un număr de instanţe mai mic decât parametrul *NMIR* specificat de utilizator sau până când au fost epuizate toate atributele disponibile.

Elagajul regulilor (rule pruning)

După ce regula a fost construită se aplică asupra ei o funcţie care încearcă să o îmbunătăţească. Această îmbunătăţire încearcă pe de o parte să facă regula mai performantă din punct de vedere calitativ dar şi mai simplă. Performanţa unei reguli este calculată cu formula 3.2.

$$Q = \frac{TP}{TP + FN} \times \frac{TN}{FP + TN} \tag{3.2}$$

Unde,

- *TP* – este numărul de instanţe acoperite care satisfac şi antecedentul şi consecventul regulii
- *FP* – numărul de instanţe care satisfac antecedentul regulii dar nu satisfac consecventul
- *FN* – numărul de instanţe care nu satisfac antecedentul regulii dar satisfac consecventul
- *TN* – numărul de instanţe care nu satisfac nici antecedentul regulii şi nici consecventul ei

Îmbunătăţirea regulii se face eliminând acei termeni din regulă care o fac să fie mai puţin performantă (principiu cunoscut sub denumirea elagajul regulilor). Eliminarea termenilor se face eliminând rând pe rând toţi termenii din regulă şi calculând calitatea regulii după ce se elimină fiecare termen, cu ajutorul formulei 3.2. Dacă eliminarea oricărui termen nu duce la obţinerea unei reguli mai bune, elagajul e încheiat. Altfel se elimină din regulă termenul care prin eliminare îmbunătăţeşte cel mai mult regula. În urma eliminării unui termen, este posibil ca consecventul regulii să trebuiască schimbat. După eliminarea oricărui termen se recalculează consecventul regulii.

Actualizarea feromonului

După ce regula a fost construită trebuie actualizată cantitatea de feromon asociată cu toţi termenii i,j existenţi, indiferent dacă au făcut parte sau nu din regulă. Cantitatea de feromon a termenilor care fac parte din regulă trebuie mărită, pentru că ea corespunde traseului urmat de furnică, iar cantitatea de feromon din ceilalţi termeni trebuie scăzută pentru a simula fenomenul de evaporare a feromonului pe traseele neumblate [MAT08]. Creşterea cantităţii de feromon corespunzătoare termenilor care fac parte din regulă se face în mod proporţional cu performanţa regulii, după formula 3.3.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot Q \tag{3.3}$$

Scăderea cantităţii de feromon se face indirect, prin normalizare. Normalizarea se realizează împărţind cantitatea de feromon asociată fiecărui termen la suma cantităţii de feromon a tuturor termenilor. Normalizarea se face după ce feromonul asociat termenilor folosiţi de regulă a fost crescut.

Analiza regulii descoperite

Se verifică dacă regula curentă este identică cu regula descoperită de furnica precedentă. Dacă regula curentă este identică cu regula descoperită de *NC* furnici sau numărul de furnici curent a ajuns egal cu parametrul *NF* setat de utilizator atunci se selectează cea mai bună regulă descoperită de furnici, se adaugă aceasta la lista de reguli şi se elimină instanţele care sunt acoperite de regula din setul de antrenare. Condiţia de oprire legată de parametrul *NC*, corespunde în realitate la situaţia când toate furnicile urmează aceeaşi cale. Procesul se repetă până când numărul de instanţe din mulţimea de antrenare devine mai mic decât parametrul *NMINR* stabilit de utilizator. Dacă condiţia de oprire legată de numărul de furnici sau cea legată de testul de convergenţă nu sunt îndeplinite, atunci se

trece la următoarea furnică care va descoperi şi ea o regulă, ghidată fiind de traseul parcurs de furnicile precedente (cantităţile de feromon actualizate).

Regula implicită

În urma etapei de antrenare de mai sus, a fost descoperită o listă de reguli. Această listă este posibil să nu acopere toate instanţele din mulţimea de test. De aceea la sfârşitul ei se adaugă o instanţă care nu are nici o condiţie în antecedent, iar în consecvent are clasa majoritară din setul de antrenare. Dacă se întâmplă ca o instanţă să nu poată fi clasificată cu regulile descoperite de furnici, atunci regula de final va atribui instanţei clasa majoritară.

Testarea regulilor descoperite

În continuare se parcurg rând pe rând instanţele din mulţimea de test. Pentru fiecare instanţă se parcurge lista de reguli, în ordinea în care au fost descoperite, căutând o regulă care să poată clasifica instanţa (să o acopere prin antecedent). Odată găsită aceasta regulă se compară clasa prezisă de regulă cu clasa instanţei, şi dacă ele coincid, se măreşte numărul instanţelor clasificate corect, iar dacă nu, se măreşte numărul instanţelor clasificate greşit.

După epuizarea numărului de instanţe din mulţimea de test se calculează acurateţea predicţiei care este raportul dintre numărul de instanţe clasificate corect şi numărul total de instanţe.

Tot procesul descris mai sus se reia de un număr de ori egal cu parametrul *NS* menţionat de utilizator. La fiecare iteraţie mulţimea de test este un alt subset din cele *NS* subseturi iar mulţimea de antrenament este formată din cele *NS-1* subseturi.

Se va calcula media acurateţei predicţiei obţinută la cele *NS* rulări.

Probabilitatea de alegere a termenilor

Probabilitatea ca un termen $T_{i,j}$ sa fie adăugat la regula curentă depinde de cantitatea de feromon a termenului şi de valoarea unei funcţii euristice după cum se poate vedea în formula 3.4.

$$P_{ij}(t) = \frac{\tau_{ij}(t)\eta_{ij}}{\sum_{i}^{a}\sum_{j}^{bi}\tau_{ij}(t)\eta_{ij}}, \forall i \in I \qquad (3.4)$$

Unde,

- η_{ij} este valoarea funcţiei euristice şi reprezintă o măsură a puterii predictive a lui $T_{i,j}$
- τ_{ij} este cantitatea de feromon disponibilă curent, la momentul *t* în poziţia *i, j* care corespunde traseului urmat de furnică
- *a* este numărul total de atribute,
- *bi* este numărul total de valori ale atributului i
- I este mulţimea atributelor care nu au fost încă folosite de furnica curentă

Cu cât este mai mare valoarea lui η_{ij} cu atât mai relevant pentru clasificare este $T_{i,j}$ şi deci cu atât mai mare este probabilitatea de a-l alege. Cantitatea de

feromon este independentă de termenii care apar deja în regula curentă parţială, dar este dependentă total de calea urmată de furnicile anterioare.

Alegerea termenului *i,j* care să fie adăugat regulii curente se face în mod aleatoriu. Cu cât valoarea lui P_{ij} este mai mare cu atât $T_{i,j}$ are mai multe şanse să fie ales.

Funcţia euristică ilustrează abilitatea pe care o are $T_{i,j}$ de a îmbunătăţi acurateţea predicţiei [PAR01]. Valoarea acestei funcţii euristice pentru un termen implică o măsură a entropiei asociată cu acel termen (vezi formula 3.5). Pentru fiecare $T_{i,j}$ entropia lui este dată de formula 3.6.

$$\eta_{ij} = \frac{\log_2(k) - Entropie(T_{ij})}{\sum_{i}^{a}\sum_{j}^{b_j}\log_2(k) - Entropie(T_{ij})} \tag{3.5}$$

$$Entropie(T_{ij}) = -\sum_{w=1}^{k}\left(\frac{freqT_{ij}^{w}}{|T_{ij}|}\right) * \log_2\left(\frac{freqT_{ij}^{w}}{|T_{ij}|}\right) \tag{3.6}$$

Unde,

- *k* este numărul de clase
- $|T_{ij}|$ este numărul total de instanţe în partiţia $|T_{ij}|$ (numărul de instanţele în care atributul *i* are valoarea *j*)
- $freqT_{ij}^{w}$ este numărul de instanţe din partiţia $|T_{ij}|$ care au clasa *w*

Entropia redă dezordinea informaţională. Cu cât este mai mare valoarea entropiei cu atât mai uniform distribuite sunt clasele şi cu atât mai puţin puternică va fi puterea de predicţie a lui $T_{i,j}$ [PAR02]. În mod evident se urmăreşte să se adauge termeni care dau o mare putere predictivă regulii curente. Cu cât este mai mare valoarea lui $Entropie(T_{ij})$ cu atât este mai mică probabilitatea de a adăuga acest termen regulii curente.

Entropia termenilor nu se schimbă pe parcursul rulării şi de aceea se calculează ca un pas de preprocesare.

3.3. Analiza parametrilor algoritmului

3.3.1. Noţiuni generale

Înainte de utilizarea algoritmului *Ant Colony Optimization* este necesară configurarea parametrilor algoritmului care sunt: numărul de furnici (*NF*), numărul minim de instanţe acoperite de orice regulă (*NMIR*), numărul maxim de instanţe care pot rămâne neacoperite de reguli (*NMINR*), numărul de reguli pentru convergenţă (*NC*), numărul de iteraţii (*NI*).

Un algoritm de clasificare trebuie să ruleze rapid, să descopere reguli cât mai simple şi cel mai important, trebuie să aibă o acurateţe a predicţiei cât mai apropiata de 100 %.

Pentru a determina în ce mod trebuie aleşi parametrii algoritmului astfel încât obiectivele de mai sus să fie atinse, algoritmul a fost testat pe 10 dataset-uri din *UCI Machine Learning Repository*, pentru diverse valori ale parametrilor.

Informaţii detaliate cu privire la dataset-urile utilizate sunt prezentate în subcapitolul 1.5. Atributele continue din dataset-urile *Ljubljana breast cancer, Diabetes, Cleveland heart disease, Iris, Dermatology* au fost discretizate cu ajutorul lui *Weka*.

Din 2001 până în prezent s-au realizat în literatura de specialitate diverse studii asupra utilizării algoritmului *Ant Colony Optimization* la clasificarea datelor. În aceste studii, autorii au utilizat diferite seturi de date obţinute de pe *UCI Machine Learnig Repository* şi parametrii algoritmului au fost aleşi de multe ori fără vreo încercare de optimizare [PAR01][PAR02][SMA06]. Problema optimizării parametrilor, respectiv a analizei sensibilităţii algoritmului la parametrii săi, a rămas o direcţie de cercetare în multe studii [PAR01][PAR02][LIU02a][LIU03a]. Tabelul 3.1 prezintă valorile alese pentru parametrii algoritmului în diverse studii.

Autorii articolelor din tabelul 3.1 au demonstrat că rezultatele care se obţin cu algoritmul *Ant Colony Optimization*, respectiv cu diverse variante ale acestuia sunt competitive cu cele obţinute cu alţi algoritmi chiar şi fără a încerca să optimizeze aceşti parametri. Analiza parametrilor algoritmului şi alegerea lor astfel încât să se obţină rezultate optime este o problemă de cercetare deschisă în continuare.

Articolul	**NF**	**NMIR**	**NMINR**	**NC**
[PAR01] [PAR02]	3000 1500	10	10	10
[SMA06]	3000	5	10	5
[SAL11]	1500	5	10	10
[WAN04]	200	8	15	8
[NEJ08]	7000	5	10	10

Tabel 3.1 – parametrii aleşi pentru algoritm în diverse studii

După cum se vede în tabelul de mai sus, valorile alese pentru *NF* variază de la 200 la 7000, valorile pentru parametrii *NMIR* şi *NC* de la 5 la 10. Se pun probleme precum: care sunt avantajele alegerii unui număr mare de furnici? Se obţin reguli cu o acurateţe mai bună? Se obţin reguli mai puţine, mai uşor de înţeles? Timpul de execuţie este mai mic? Aceleaşi probleme se pun şi pentru restul parametrilor şi lucrarea de faţă încearcă să găsească răspuns la aceste întrebări.

Pentru ca seturile de date alese să poată fi analizate cu *Ant Miner* versiunea 1.2.1 (software Open-Source care implementează algoritmul *Ant Colony Optimization* în problema descoperirii regulilor de clasificare – vezi 3.1) a fost necesară o etapă de preprocesare care a constat în discreditarea atributelor continue, deoarece *Ant Miner* nu funcţionează decât pentru atribute categoriale. Discretizarea datelor s-a făcut cu ajutorul lui *Weka*.

Testele au fost făcute rulând aplicaţia în paralel pe un număr de 30 de calculatoare dintr-un laborator al Universităţii "Politehnica" din Timişoara. Toate

calculatoarele au configuraţie identică Intel Core 2 Quad CPU, 2,4 GHz, 2,4 GHz , 4 GB RAM.

Acurateţea predicţiei s-a obţinut în felul următor:

- fiecare set de date a fost împărţit în mod aleatoriu în 10 subseturi (mulţimi) care conţin un număr de instanţe aproximativ egal.
- algoritmul a fost rulat de 10 ori pe cele 10 mulţimi, la fiecare rulare o mulţime din cele 10 a fost folosită pentru testare şi celelalte 9 mulţimi pentru învăţare.
- valorile medii ale acurateţei predicţiei sunt cele prezentate în continuare

Au fost rulate în total 64 de configuraţii de parametri pe fiecare din cele 10 seturi de date alese.

Analiza parametrilor algoritmului s-a realizat în două moduri: empiric şi matematic. Analiza empirică a constat în compararea rezultatelor obţinute cu 31 de configuraţii de parametri, s-a analizat influenţa pe care o are modificarea fiecărui parametru de intrare asupra acurateţei predicţiei şi asupra timpului de execuţie a algoritmului.

Analiza matematică a încercat descoperirea unor relaţii matematice între parametrii de intrare şi acurateţea predicţiei (s-au construit modele matematice de regresie) şi s-au construit matrice de corelare, s-au analizat relaţiile dintre parametrii de intrare şi cel de ieşire cu ajutorul coeficienţilor din modelele de regresie şi din matricele de corelare. Pentru a realiza analiza matematică s-au construit noi seturi de date cu valorile alese pentru parametrii de intrare şi cu acurateţea predicţiei obţinută rulând algoritmul cu parametrii aleşi. Pentru a avea un volum de date mai mare pentru analiză s-au utilizat toate rezultatele obţinute cu cele 64 de configuraţii de parametri. Utilitarul cu care au fost construite modelele de regresie şi matricele de corelaţie este *DataFit* [www7].

3.3.2. Analiza empirică a parametrilor

3.3.2.1. Analiza parametrului *Număr de furnici (NF)*

Pentru a analiza efectul pe care îl are alegerea numărului de furnici asupra acurateţei predicţiei s-au realizat teste cu 13 numere de furnici care iau valori cuprinse între 5 şi 1500, în vreme ce ceilalţi parametri au fost păstraţi la valori fixe (vezi tabelul 3.2).

Parametru	**Varianta 1**
NF	5, 10, 15, 20, 25, 30, 40, 50, 100, 200, 500, 1000, 1500
NMIR	5
NMINR	10
NC	10
NI	100

Tabel 3.2 – Valorile parametrilor algoritmului

S-a determinat pentru fiecare set de date din tabelul 3.1 care este cea mai bună valoare pentru acurateţea predicţiei şi cu ce număr de furnici s-a obţinut acea valoare.

Rezultatele obţinute în ceea ce priveşte acurateţea predicţiei pentru seturile de date alese şi parametrii din tabelul 3.2 sunt prezentate în tabelul 3.3. Valorile maxime au fost subliniate cu **bold**, *italic* şi underline, iar cele minime doar cu **bold**.

Nr. furnici / Datasetul	5	10	15	20	25	30	40	50	100	200	500	1000	1500
Car	84.7	84.3	85	85.4	86.11	86.1	**84.2**	85.3	85.9	***86.9***	85.36	85.07	86.4
Zoo	92.47	87.62	91.23	90.95	***93.66***	92.53	92.78	90.69	91.73	92.05	91.44	**83.21**	91.63
Ljubljana breast cancer	74.79	74.16	72.74	73.24	75.13	73.53	74.04	76.58	**71.97**	72.37	***76.75***	72.43	75.93
Diabetes	74.85	73.69	73.56	74.32	73.7	71.74	73.3	72.39	**71.61**	73.42	72.4	***75.37***	72
Cleveland heart disease	78.99	78.24	***81.24***	76.85	75.15	**73.19**	75.83	74.41	77.54	78.08	77.96	77.41	76.29
Iris	**94**	95.3	94.6	95.3	***96***	95.3	**94**	***96***	95.33	**94**	94.67	***96***	94.67
Mushrooms	***98***	97.97	96.15	96.24	97.27	96.17	97.91	95.65	94.41	**93.88**	94.56		
Nursurey	**86.38**	86.45	***86.52***	86.49	***86.52***	86.47	86.39	86.51	86.48	86.45	86.5		
Tic	***72.13***	70.35	72.12	71.39	70.67	70.66	70.77	71.09	**69.32**	71.93	71.2	71.17	70.16
Dermatology	**94.29**	94.56	***95.59***	94.56	94.53	94.8	95.1	94.55	94.8	95.1	95.58	94.81	95

Tabel 3.3 – acurateţea predicţiei pentru varianta 1 de parametri

După cum se observă valorile predicţiei pentru seturile de date *Mushrooms* şi *Nursurey* în cazul a 1000 respectiv 1500 de furnici au rămas necompletate, datorită faptului că la trei rulări consecutive în care aplicaţia a fost lăsată să meargă vreme de 4 ore la fiecare rulare, aplicaţia nu a putut verifica acurateţea predicţiei pentru nici măcar o mulţime din cele 10.

Tabelul 3.4 arată valorile maxime şi minime pentru acurateţea predicţiei, diferenţa dintre valoarea minimă şi cea maximă, precum şi numărul de furnici pentru care s-au obţinut aceste valori.

Dataset	**Valoare maximă acurateţe**	**Nr furnici acurateţe maximă**	**Valoare minimă acurateţe**	**Nr furnici acurateţe minimă**	**Diferenţa între acurateţea maximă şi cea minimă**
Car	86.9	200	84.2	40	2.7
Zoo	93.66	25	83.21	1000	10.45
Ljubljana breast cancer	76.75	500	71.97	100	4.78
Diabetes	75.37	1000	71.61	100	3.76
Cleveland heart disease	81.24	15	73.19	30	8.05
Iris	96	50,1000	94	5,40,200	2
Mushrooms	98	5	93.88	200	4.12
Nursurey	86.52	15,25	86.38	5	0.14
Tic	72.13	5	69.32	100	2.81
Dermatology	95.59	15	94.29	5	1.3

Tabel 3.4 – valorile maxime şi minime ale acurateţei predicţiei şi numărul de furnici cu care s-au obţinut acestea, pentru varianta 1 de parametri

După cum se observă din tabelul de mai sus, alegerea numărului de furnici poate influenţa acurateţea predicţiei între 0.14 (cazul setului de date *Nursurey*) şi

10.45 procente (cazul setului de date Zoo). Tabelul 3.5 prezintă numărul de furnici şi pentru fiecare furnică, numărul de seturi de date în care ea a produs cea mai bună acurateţe a predicţiei şi cea mai slabă acurateţe a predicţiei.

Nr furnici	5	10	15	20	25	30	40	50	100	200	500	1000	1500
Nr dataseturi acurateţe maximă	2	0	3	0	2	0	0	1	0	1	1	2	0
Nr dataseturi acurateţe minimă	3	0	0	0	0	1	2	0	3	2	0	1	0

Tabel 3.5 – numerele de furnici şi la câte valori maxime şi minime au condus în ceea ce priveşte acurateţea predicţiei în varianta 1 de parametri

Analizând tabelul 3.5 putem afirma că numărul de furnici care are cea mai mare şansă să conducă la cea mai mare acurateţe a predicţiei este 15, întrucât cu acest număr de furnici s-a obţinut de 3 ori valoarea maximă a predicţiei şi niciodată valoarea minimă. Următorul număr de furnici care a produs rezultate bune este 25 de 2 ori maxim şi niciodată minim. Cel mai slab rezultat se poate spune ca l-a produs numărul de 100 de furnici, care a dus la 3 valori minime şi niciun maxim, urmat de numărul 40 de furnici, care a produs 2 valori minime şi niciun maxim, la egalitate cu numărul 1500 care deşi nu a produs valori minime şi maxime nu a putut fi rulat pe seturi de date voluminoase precum *mushrooms* şi *nursurey*.

Evoluţia acurateţei predicţiei la creşterea numărului de furnici se observă şi în figura de mai jos. Pe axa x este reprezentat vectorul de furnici cu v[0]=5, v[1]=10,…,v[12]=1500 furnici.

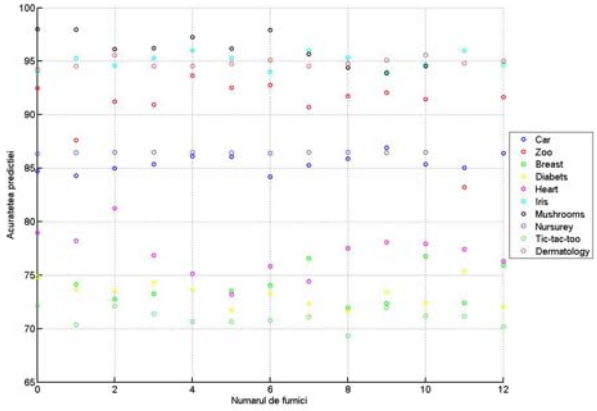

Fig. 3. 1 – Acurateţea predicţiei pentru varianta 1 de parametri

În ceea ce priveşte timpul de execuţie în care s-au obţinut rezultatele din tabelul 3.4, acesta este exprimat în secunde în tabelul 3.6.

Nr. furnici / Datasetul	5	10	15	20	25	30	40	50	100	200	500	1000	1500
Car	7	12	19	28	32	41	54	72	144	274	717	1414	2011
Zoo	9	19	28	37	47	58	79	93	191	381	977	1991	3074
Ljubljana breast cancer	2	4	6	9	10	14	16	22	40	81	227	420	633

Diabetes	3	7	8	11	14	16	22	20	37	86	277	414	779
Cleveland heart disease	1	3	4	5	6	7	11	13	24	51	130	263	396
Iris	0	0	0	0	1	1	1	2	4	8	24	47	64
Mushrooms	457	885	1297	1637	2320	3033	4358	5062	9485	18572	42351	∞	∞
Nursurey	72	127	186	247	309	376	513	644	1267	2569	6662	∞	∞
Tic	8	11	23	29	42	51	61	78	158	229	734	1405	2112
Dermatology	74	119	169	203	230	266	317	428	768	1320	3320	6312	9679

Tabel 3.6 – timpul de execuţie în varianta 1 de parametri

Analizând timpii de execuţie putem afirma că întotdeauna un număr mic de furnici este de preferat faţă de un număr mare de furnici, întrucât timpul de execuţie creşte direct proporţional cu numărul de furnici. În cadrul setului de date *Mushrooms* timpul de execuţie ajunge la 11 ore pentru 500 de furnici. Figura de mai jos arată evoluţia timpului de execuţie la creşterea numărului de furnici:

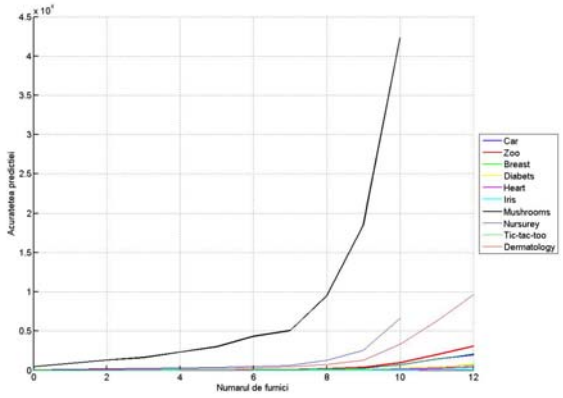

Fig 3.2 – Timpul de execuţie pentru varianta 1 de parametri

3.3.2.2. Analiza parametrului *Numărul minim de instanţe pe care orice regulă trebuie să le acopere* (*NMIR*)

Pentru a analiza influenţa parametrului *NMIR* asupra funcţionării algoritmului s-a considerat numărul de 15 furnici care a produs cele mai bune rezultate în testele anterioare. Pentru *NMIR* s-au luat valorile 5, 10 şi 15. Ceilalţi parametri au fost păstraţi la valorile din tabelul 3.7.

Parametru	Varianta 2	Varianta 3	Varianta 4
NF	15	15	15
NMIR	5	10	15
NMINR	10	10	10
NC	10	10	10
NI	100	100	100

Tabel 3.7 – Valorile parametrilor algoritmului

Rezultatele obţinute în ceea ce priveşte timpul de execuţie şi acurateţea predicţiei pentru cele 3 variante sunt prezentate în tabelul 3.8. Cu *AVn* s-a notat

acurateţea predicţiei obţinută cu varianta *n* de parametri, iar cu *TEVn* s-a notat timpul de execuţie obţinut cu varianta *n* de parametri. În cazul variantei 4 de parametri, au fost 4 seturi de date pentru care nu s-a obţinut un rezultat (datorită intrării aplicaţiei în buclă infinită). În cazul în care nu s-a putut ajunge la un rezultat, au mai fost încercate încă 2 rulări cu acelaşi set de parametri pe acelaşi set de date. Dacă în urma acestor 3 încercări s-a ajuns la un rezultat, atunci acurateţea predicţiei a fost completată în tabelul 3.8 iar dacă nu s-a ajuns la rezultat atunci numărul maxim de mulţimi, care a putut fi testat la una din cele 3 rulări, a fost completat în tabel în formatul cifră de la 0 la 9 şi apoi litera f (ex. 5f= s-au încercat 3 rulări, niciuna nu a putut fi finalizată din cauza faptului că aplicaţia a intrat în buclă infinită, performanţa maximă obţinută este că la una din aceste 3 rulări s-au putut verifica 5 folds).

Setul de date	AV2	AV3	AV4	TEV2	TEV3	TEV4
Car	***85***	83.68	3f	19	17	∞
Zoo	***91.23***	84.91	80.51	28	28	26
Ljubljana breast cancer	72.74	***74.89***	3f	6	6	∞
Diabetes	***73.56***	73.04	72.5	8	9	7
Cleveland heart disease	***81.24***	79.22	8f	4	4	∞
Iris	94.6	***95.33***	94	0	0	0
Mushrooms	96.15	96.95	***97.06***	1297	1212	1427
Nursurey	***86.52***	86.51	86.5	186	185	188
Tic	72.12	***72.44***	9f	23	28	∞
Dermatology	***95.59***	93.72	94.02	169	160	151

Tabel 3.8 - acurateţea predicţiei şi timpul de execuţie pentru variantele 2, 3 şi 4 de parametri

Analizând rezultatele din tabelul de mai sus observăm că varianta 2 de parametri conduce la acurateţea maximă în 6 cazuri, varianta 3 de parametri în 3 cazuri şi varianta 4 într-un singur caz. Deci creşterea *NMIR* conduce la o scădere a acurateţei predicţiei şi conduce la multe situaţii de blocare. Cele mai bune rezultate s-au obţinut pentru varianta 2, în care *NMIR* ia valoarea 5.

3.3.2.3. Analiza parametrului *Numărul maxim de instanţe neacoperite de nici o regulă* (*NMINR*)

Pentru a analiza comportamentul algoritmului, a fost considerată varianta 5 de parametri, în care au fost păstraţi parametrii din varianta 4, mai puţin numărul de instanţe neacoperite de nicio regulă care în loc să primească valoarea 10 primeşte valoarea 20.

Parametru	Varianta 4	Varianta 5
NF	15	15
NMIR	15	15
NMINR	10	20
NC	10	10
NI	100	100

Tabel 3.9 – Valorile parametrilor algoritmului în variantele 4 şi 5 de parametri

Rezultatele obţinute sunt prezentate comparativ în tabelul 3.10. Un prim lucru care se observă este că creşterea numărului de instanţe neacoperite de nici o regulă face ca programul sa găsească soluţii în cazul seturilor de date pentru care acest lucru nu a fost posibil cu varianta 4 de parametri.

Varianta 5 de parametri este mai bună pentru că a găsit valori maxime în cazul a 6 seturi de date.

Setul de date	**AV4**	**AV5**	**TV4**	**TV5**
Car	3f	***84.66***	∞	16
Zoo	***80.51***	79.42	26	25
Ljubljana breast cancer	3f	***75.53***	∞	6
Diabetes	72.5	***73.44***	7	7
Cleveland heart disease	8f	***73.92***	∞	3
Iris	***94***	93.33	0	0
Mushrooms	97.06	***97.7***	1427	1338
Nursurey	***86.5***	86.47	188	180
Tic	9f	***73.29***	∞	22
Dermatology	***94.02***	93.74	151	153

Tabel 3.10 – comparaţie între valorile maxime obţinute cu variantele 4 şi 5 de parametri

Deşi ne aşteptam ca creşterea numărului de instanţe neacoperite de nici o regulă să determine o scădere a performanţei, în acest caz lucrurile nu stau aşa. În 6 cazuri din 10 s-a obţinut o acurateţe mai bună a predicţiei dacă se lasă maxim 20 de instanţe neacoperite, în 4 cazuri s-a obţinut o acurateţe a predicţiei mai bună dacă se lasă maxim 10 instanţe neacoperite. O posibilă explicaţie pentru acest rezultat este numărul mare de rulări nereuşite, în cazul cu fiecare regulă trebuie să acopere minim 15 instanţe şi pot rămâne 10 instanţe neacoperite de nicio regulă. Pentru a lămuri acest aspect s-a realizat o verificare suplimentară, cea din tabelul 3.11.

Parametru	Varianta 6	Varianta 7
NF	15	15
NMIR	5	5
NMINR	10	30
NC	10	10
NI	100	100

Tabel 3.11 – Variantele 6 şi 7 de parametri

Setul de date	AV 6	AV 7	TE 6	TE 7
Car	***85***	84.83	19	21
Zoo	***91.23***	81.19	28	24
Ljubljana breast cancer	72.74	***73.86***	6	5
Diabetes	***73.56***	72.63	8	9
Cleveland heart disease	***81.24***	78.43	4	3
Iris	94.6	***95.33***	0	0
Mushrooms	96.15	***97.12***	1297	1583
Nursurey	86.52	***86.54***	186	176
Tic	***72.12***	70.23	23	19
Dermatology	***95.59***	95.07	169	148

Tabel 3.12 – acurateţea predicţiei şi timpul de execuţie pentru variantele 5 şi 6 de parametri

În varianta 6 se obţin în cazul a 6 seturi de date, valori ale acurateţei predicţiei mai mari decât în cazul variantei 7. În cazul setului de date *zoo* diferenţa este semnificativă (10 %). Deci acurateţea predicţiei este în general mai bună dacă configurăm numărul maxim de instanţe neacoperite de nici o regulă la o valoare cât mai mică, excepţie de la acest fapt fac situaţiile în care numărul de instanţe care trebuie acoperit de o regula este mare şi conduce la multe situaţii de blocare. În acest caz este mai bine să mărim *NMINR*, vom obţine rezultate mai bune.

3.3.2.4. Analiza parametrului *Numărul de reguli pentru convergenţă* (*NC*)

Pentru a analiza acest parametru considerăm valorile 2, 5, 10, 20. Cele 4 variante de parametri analizaţi sunt prezentate în tabelul 3.13.

Parametru	Varianta 8	Varianta 9	Varianta 10	Varianta 11
NF	15	15	15	15
NMIR	5	5	5	5
NMINR	10	10	10	10
NC	2	5	10	20
NI	100	100	100	100

Tabel 3.13 - variantele 7, 8, 9 şi 10 de parametri

Tabelul 3.14 prezintă rezultatele obţinute cu variantele de parametri 8, 9,10 şi 11.

Setul de date	**AV8**	**AV9**	**AV10**	**AV11**	**TEV8**	**TEV9**	**TEV10**	**TEV11**
Car	***85.71***	85.59	85	85.07	4	10	19	36
Zoo	84.57	***93.11***	91.23	90.55	6	14	28	58
Ljubljana breast cancer	***74.89***	73.81	72.74	74.79	1	3	6	13
Diabetes	73.7	72.28	73.56	***74.48***	1	2	8	19
Cleveland heart disease	78.92	77.03	***81.24***	77.04	0	1	4	8
Iris	94.67	93.33	94.6	***95.33***	0	0	0	1
Mushrooms	***97.4***	96.47	96.15	96.83	284	586	1297	2411
Nursurey	86.36	86.47	***86.52***	86.45	40	94	186	353
Tic	72.34	71.7	72.12	***72.85***	3	13	28	41
Dermatology	91.56	93.48	***95.59***	94.56		94	157	256

Tabel 3.14 - acurateţea predicţiei şi timpul de execuţie pentru variantele 8, 9, 10 şi 11 de parametri

Analizând rezultatele din tabelul 3.14 putem afirma că pentru variantele de parametri 8, 10 şi 11 s-au obţinut maxime ale acurateţei predicţiei de 3 ori, iar pentru varianta 9 s-a obţinut maximul acurateţei predicţiei o singură dată. Creşterea numărului de reguli pentru convergenţă determină creşterea timpului de execuţie. Deci cel mai bun rezultat, obţinut judecând prin prisma acurateţei predicţiei respectiv timpului de execuţie, este pentru varianta 8 de parametri.

3.3.2.5. Analiza parametrului *Numărul de iteraţii* (*NI*)

O regulă este descoperită fie dacă furnica curentă a descoperit aceeaşi regulă ca şi *NC-1* furnici, fie dacă furnica curentă nu a descoperit aceeaşi regulă ca şi *NC-1* furnici, dar s-a executat numărul de iteraţii stabilit.

Influenţa pe care o are acest parametru *NI* este analizată în continuare. S-au considerat 4 variante ale parametrilor:

Parametru	Varianta 12	Varianta 13	Varianta 14	Varianta 15
NF	15	15	15	15
NMIR	5	5	5	5
NMINR	10	10	10	10
NC	2	2	2	2
NI	10	50	100	200

Tabel 3.15 - variantele 12, 13, 14 şi 15 de parametri

Setul de date	**AV12**	**AV13**	**AV14**	**AV15**	**TEV12**	**TEV13**	**TEV14**	**TEV15**
Car	85.47	84.72	***85.71***	85.13	3	4	4	4
Zoo	***93.77***	90.18	84.57	91.83	6	6	6	5
Ljubljana breast cancer	73.73	***74.9***	74.89	74.16	1	1	1	1
Diabetes	72.65	***74.48***	73.7	72.64	1	1	1	1
Cleveland heart disease	77.91	77.46	***78.92***	78.33	0	0	0	1
Iris	94.67	***96***	94.67	94	0	0	0	0
Mushrooms	97.06	97.12	***97.4***	96.77	238	241	284	265
Nursurey	***86.42***	86.06	86.36	86.15	39	37	40	36
Tic	***74***	73.36	72.34	71.49	4	2	3	5
Dermatology	***94.02***	92.39	91.56	92.82	50	47	46	51

Tabel 3.16 - acurateţea predicţiei şi timpul de execuţie pentru variantele 12, 13, 14 şi 15 de parametri

În ceea ce priveşte acurateţea predicţiei, cel mai bun rezultat se obţine pentru varianta 12 de parametri (10 iteraţii) şi anume 4 valori maxime.

Analizând timpul de execuţie în cazul celor 4 variante, observăm că se obţin valori asemănătoare, cu toate că numărul de iteraţii ia valori într-o plajă mare de la 10 la 200. O posibilă explicaţie este că numărul mic de reguli pentru convergenţă face ca valoarea acestui parametru să nu mai conteze, bucla repetitivă oprindu-se de fiecare dată datorită neîndeplinirii condiţiei legate de numărul de reguli pentru convergenţă şi nu datorită depăşirii numărului de iteraţii. Pentru a verifica această teorie s-a ales un număr mare de reguli pentru convergenţă (30) şi valorile din tabelul 3.17:

Parametru	Varianta 16	Varianta 17	Varianta 18	Varianta 19
NF	15	15	15	15
NMIR	5	5	5	5
NMINR	10	10	10	10
NC	30	30	30	30
NI	10	50	100	200

Tabel 3.17 - variantele 16, 17, 18 şi 19 de parametri

Tabelul 3.18 prezintă acurateţea predicţiei pentru variantele 16, 17, 18 şi 19 de parametri, precum şi timpii de execuţie.

Setul de date	AV16	AV17	AV18	AV19	TV16	TV17	TV18	TV19
Car	85.77	85.25	***86.46***	84.96	14	55	56	64
Zoo	91.12	91.7	***92.73***	91.62	28	92	89	84
Ljubljana breast cancer	***75.56***	73.74	72.77	73.46	5	19	20	20
Diabetes	73.57	***74.22***	73.84	73.83	4	18	22	28
Cleveland heart disease	76.96	***80.1***	78.99	77.32	3	12	12	13
Iris	***96***	94	***96***	94	0	1	1	1
Mushrooms	***96.86***	96.48	95.38	96.76	1094	3825	4152	5709
Nursurey	86.42	86.47	86.46	***86.52***	169	528	536	541
Tic	70.12	72.75	69.29	***73.39***	15	59	61	54
Dermatology	94.53	95.38	***95.59***	94.55	97	346	347	360

Tabel 3.18 - acurateţea predicţiei şi timpul de execuţie pentru variantele 16, 17, 18 şi 19 de parametri

Comparând evoluţia timpilor de execuţie din tabelul 3.18 cu cea din tabelul 3.16, observăm că timpul de execuţie din tabelul 3.18 are o alură crescătoare iar cel din tabelul 3.16 este mai degrabă constant, ceea ce înseamnă că în cazul 3.16, valorile alese pentru numărul de instanţe nu au prea mare importanţă datorită valorii mici a lui *NC*. Analizând timpii de execuţie din tabelul 3.18 putem spune că ei sunt comparabili pentru variantele 17, 18 şi 19 de parametri şi sunt mult mai mici în cazul variantei 16 de parametri (excepţie face setul de date *mushrooms* în care timpul de execuţie este direct proporţional cu numărul de parametri). Deci un număr mic de iteraţii determină execuţia mai rapidă a algoritmului, iar creşterea numărului de iteraţii peste o anumită valoare (în cazul experimentelor noastre, valoarea 50) poate să nu modifice decât puţin timpul de execuţie. În ceea ce priveşte acurateţea predicţiei, cel mai bun rezultat se obţine pentru varianta 18 de parametri (100 de iteraţii) şi anume 4 valori maxime.

3.3.3. Analiza matematică a coeficienţilor algoritmului

În a doua parte a studiului s-a încercat descoperirea unor relaţii matematice între parametrii de intrare şi parametrul de ieşire al algoritmului (acurateţea predicţiei). S-au construit modele matematice de regresie şi s-au construit matrice de corelare, s-au analizat relaţiile dintre parametrii de intrare şi parametrul de ieşire cu ajutorul coeficienţilor din modelele de regresie şi din matricele de corelare.

Modelele matematice de regresie construite permit estimarea ieşirii (acurateţei predicţiei) în funcţie de parametri de intrare ai algoritmului (numărul de furnici -*NF*, numărul minim de instanţe acoperite de orice regulă -*NMIR*, numărul maxim de instanţe care pot rămâne neacoperite de reguli - *NMINR*, numărul de reguli pentru convergenţă -*NC,* numărul de iteraţii -*NI*).

Pentru determinarea acestor modele şi pentru construirea matricelor de corelare s-au construit 10 noi seturi de date. Fiecare set de date conţine valorile alese pentru parametrii de intrare şi acurateţea predicţiei obţinute rulând algoritmul cu parametrii aleşi. Toate seturile de date construite conţin 64 de instanţe, pentru că s-au realizat 64 de rezultate experimentale pentru fiecare set de date (s-au

analizat 64 de variante de parametri pentru fiecare set de date). Variantele de parametri analizate sunt prezentate în tabelul 3.20. Alegerea acestor variante de parametri a fost influenţată de valorile implicite pe care aceşti parametri le au în *Ant Miner* [PAR02], de valorile care au fost alese de alţi cercetători în studiile pe care le-au realizat (vezi tabelul 3.2), de rezultatele care s-au obţinut pentru acurateţea predicţiei pe parcursul execuţiei (vezi studiul comparativ din subcapitolul 3.3.2) şi de semnificaţia mărimilor. În cazul configuraţiilor de parametri de intrare care au determinat algoritmul să ruleze la infinit (acestea sunt de obicei configuraţiile de parametri în care *NMIR* are valoarea 15) în coloana aferentă variabilei dependente (acurateţea predicţie) a fost completată valoarea 100 împărţită la numărul de valori ale atributului clasă (s-a completat probabilitatea de a alege clasa corecta fără niciun fel de informaţie suplimentară).

NF	NMIR	NMINR	NC	NI
5, 10, 15, 20, 25, 30, 40, 50, 100, 200, 500, 1000, 1500	5	10	10	100
5, 10, 15, 20, 25, 30, 40, 50, 100, 200, 500, 1000, 1500	10	10	10	100
5, 10, 15, 20, 25, 30, 40, 50, 100, 200, 500, 1000, 1500	15	10	10	100
5, 10, 15, 20, 25, 30, 40, 50, 100, 200, 500, 1000, 1500	15	20	10	100
15	5	30	10	100
15	5	10	2, 5, 20	100
15	5	10	2	10, 50, 100, 200
15	5	10	30	10, 50, 100, 200

Tabel 3.19 – Valorile parametrilor de intrare ai algoritmului

Instrumentul software utilizat pentru a construi modelele de regresie şi pentru a determina matricele de corelare este *DataFit* [www7]. Iniţial s-a încercat construirea unor modele de regresie folosind un total de 10 variabile independente pe lângă cei 5 parametri de intrare menţionaţi, încă 5 parametri care caracterizează setul de date. Aceştia sunt: numărul de instanţe, numărul de atribute, numărul de valori ale atributului clasă, numărul de valori lipsă, numărul total de valori ale atributelor. Pe baza acestor 10 variabile independente utilitarul *DataFit* nu a putut construi modele de regresie.

Analiza modelelor de regresie

Modelele de regresie obţinute utilizând ca şi variabile independente cei 5 parametri de intrare ai algoritmului sunt prezentate în tabelul 3.20.

Setul de date	Modelul	R^2
Car	AP =0.013 *NF-3.876*NMIR+2.638*NMINR-0.002*NC-0.001*NI+ 78.518	0.55
Zoo	AP = exp(-0.011*NMIR-0.004*NMINR+0.001*NC+4.591)	0.83
	AP = -0.918*NMIR+-0.293*NMINR+0.075*NC-0.003*NI+97.48	0.83
Ljubljana breast cancer	AP = 0.001*NF-1.744*NMIR+1.269*NMINR-0.034*NC-0.003*NI + 71.674	0.58
Diabetes	AP =0.001 *NF-1.220*NMIR+0.758*NMINR+0.013*NC-0.001*NI+ 72.378	0.40
Cleveland heart disease	AP =0.004*NF-2.923*NMIR+2.101*NMINR-0.012*NC+0.001*NI+ 72.349	0.39
Iris	AP = 0.024*NMIR-0.006*NMINR+0.009*NC-0.006*NI+95.68	0.08
Mushrooms	AP = -0.035*NF+0.055*NMIR+0.002*NMINR-0.039*NC+98.136	0.90

	AP = exp(0.001*NMIR+4.589)	0.89
Nursurey	AP = -0.049*NF+0.125*NMIR-0.038*NMINR-0.026*NC+0.002*NI+ 88.341	0.87
	AP = exp(-0.001*NF+0.002*NMIR-0.001*NMINR+4.487)	0.83
Tic	AP = -1.579*NMIR+0.998*NMINR-0.042*NC+70.804	0.60
Detmatology	AP = 0.001*NF+0.035*NMIR+0.002*NMINR+0.07*NC-0.001*NI+ 93.395	0.29

Tabel 3.20 – Modelele de regresie determinate şi performanţele lor

R^2 este coeficientul determinării multiple şi se calculează cu formula 3.7.

$$R^2 = 1 - \frac{RSS}{TSS} \tag{3.7}$$

unde *RSS* este suma reziduală a pătratelor şi se calculează cu formula 3.8, iar *TSS* este suma totală a pătratelor şi se calculează cu formula 3.9.

$$RSS = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 \tag{3.8}$$

$$TSS = \sum_{i=1}^{n} \left(Y_i - \overline{Y}\right)^2 \tag{3.9}$$

Unde, Y_i este valoarea reală a variabilei dependente (a acurateţei predicţiei) pentru instanţa *i* şi $\hat{Y}_i$ este valoarea estimată de modelul de regresie pentru variabila dependentă şi $\overline{Y}$ este valoarea medie a variabilei dependente.

Mărimea *RSS* reprezintă suma reziduală a pătratelor (*Residual Sum of Squares*) şi se calculează ca şi sumă a pătratelor diferenţelor dintre valorile reale ale variabilei dependente şi valorile estimate de modelul de regresie, cu formula 3.8. O potrivire perfectă între acestea ar conduce la o sumă reziduală a pătratelor de valoare 0.

Mărimea R^2 ia valori cuprinse între 0 şi 1 şi măsoară cât de bine surprinde modelul de regresie variaţia variabilei dependente. De exemplu dacă R^2 are valoarea 0.8, atunci 80 % din variaţia variabilei dependente este explicată de modelul de regresie.

Utilitarul *DataFit* a determinat pentru fiecare set de date câte 3 modele. În tabelul 3.20 au fost prezentate doar modelele de regresie care au obţinut cele mai bune performanţe pentru fiecare set de date. După cum se observă unele din aceste modele surprind bine variaţia variabilei dependente (cele care au mărimea R^2 apropiată de valoarea 1) iar altele au o performanţă slabă.

Analizând cele 10 seturi de date alese pentru analiză se observă că 4 dintre ele conţin date medicale (breast, diabetes, heart, dermatology), 3 conţin date care caracterizează elemente din natură (animale sau plante: zoo, iris, mushrooms), unul conţine evaluările unor maşini (car), unul conţine evaluarea aplicaţiilor depuse la grădiniţe şi unul codifică mutările posibile la jocul X şi 0.

Modelele de regresie care s-au construit pe seturile de date medicale descriu între 29 % şi 58 % variaţia acurateţei predicţiei, deci sunt foarte slabe. Două din modelele de regresie construite pe seturi de date care caracterizează elemente din natură, zoo şi mushrooms, au obţinut o performanţă mai bună de 83 % respectiv 90% iar setul de date iris a obţinut cea mai slabă performanţă dintre toate seturile de date, de 8 %. Modelul de regresie construit pe setul de date car, are o performanţă slabă de 55 %. Pe baza setului de date ce conţine evaluarea aplicaţiilor la grădiniţe s-au construit modele de regresie bune de 83% şi 89 %. Modelul de regresie construit pe baza setului de date tic surprinde 60 % din variaţia mărimii dependente.

În continuare au fost analizaţi coeficienţii parametrilor din modelele de regresie determinate. Valorile acestora arată cât de mult influenţează aceşti parametri variabila dependentă. Valorile acestor parametri pot fi vizualizate în tabelul 3.21. S-au luat în considerare doar modelele care au o performanţă medie sau bună (pentru care parametrul R^2 este mai mare de 0.5). S-a determinat modulul pentru fiecare parametru şi maximul dintre valorile absolute pentru fiecare model. Valorile maxime au fost subliniate. Toate valorile maxime în valoare absolută au fost asociate coeficienţilor parametrului *NMIR*, ceea ce înseamnă că acurateţea predicţiei depinde într-o bună măsură de modul în care se alege acest parametru. Valorilor cele mai mari în modul ale acestui parametru le corespund valori negative, ceea ce însemnă că, de obicei, creşterea acestui parametru determină o scădere a acurateţei predicţiei.

NF	NMIR	NMINR	NC	NI	|NF|	|NMIR|	|NMINR|	|NC|	|NI|
0.013	-3.876	2.638	-0.002	-0.001	0.013	***3.876***	2.638	0.002	0.001
0	-0.011	-0.004	0.001	0	0	***0.011***	0.004	0.001	0
0	-0.918	-0.293	0.075	-0.003	0	***0.918***	0.293	0.075	0.003
0.001	-1.744	1.269	-0.034	-0.003	0.001	***1.744***	1.269	0.034	0.003
-0.035	0.055	0.002	-0.039	0	0.035	***0.055***	0.002	0.039	0
0	0.001	0	0	0	0	***0.001***	0	0	0
-0.049	0.125	-0.038	-0.026	0.002	0.049	***0.125***	0.038	0.026	0.002
-0.001	0.002	-0.001	0	0	0.001	***0.002***	0.001	0	0
0	-1.579	0.998	-0.042	0	0	***1.579***	0.998	0.042	0

Tabel 3.21 – Coeficienţii parametrilor din modelele de regresie determinate

S-a determinat maximul valorilor absolute, exceptând parametrul *NMIR* (următorul parametru ca şi importanţă). În 6 din cele 9 modele, valoarea maximă a corespuns parametrilor lui *NMINR*, în unul dintre ele lui NC şi în unul dintre ele lui *NF*. Deci următorul parametru ca şi importanţă este *NMINR*. Creşterea lui *NMINR* poate determina creşterea sau scăderea performanţei în funcţie de setul de date (pentru 4 seturi de date determină creşterea performanţei şi pentru 4 scăderea ei).

Analiza coeficienţilor de corelare

Următoarea etapă a analizei a constat în analiza coeficienţilor din matricele de corelare aferente fiecărui set de date. Tabelul 3.22 prezintă matricea de corelare aferentă setului de date car. Au fost analizaţi parametrii care influenţează cel mai mult parametrul de ieşire potrivit matricei de corelare. În cazul setului de date *car*,

se poate observa că, creşterea *NMIR* determină o scădere substanţială a performanţei, iar creşterea lui *NMINR* determină o creştere a performanţei.

	NF	**NMIR**	**NMINR**	**NC**	**NI**	**AP**
NF	1.00	0.13	0.02	-0.06	0.02	0.14
NMIR	0.13	1.00	**0.41**	-0.14	0.06	***-0.51***
NMINR	0.02	**0.41**	1.00	-0.06	0.03	***0.22***
NC	-0.06	-0.14	-0.06	1.00	-0.05	0.06
NI	0.02	0.06	0.03	-0.05	1.00	-0.03
AP	0.14	-0.51	0.22	0.06	-0.03	1.00

Tabel 3.22 – Matricea de corelare aferentă setului de date car

Tabelul 3.23 prezintă coeficienţii de corelare dintre variabila dependentă şi variabilele independente potrivit matricelor de corelare aferente fiecărui set de date.

Setul de date	NF	NMIR	NMINR	NC	NI
Car	0.14	-0.51	0.22	0.06	-0.03
Zoo	-0.01	-0.14	0.04	0.30	0.36
Ljubljana breast cancer	-0.06	-0.54	0.27	0.05	-0.04
Diabetes	-0.02	-0.49	0.15	0.07	-0.03
Cleveland heart disease	0.01	-0.44	0.21	0.05	-0.02
Iris	-0.07	-0.18	-0.1	0.1	-0.22
Mushrooms	-0.95	-0.11	-0.01	0.04	-0.02
Nursurey	-0.93	-0.10	-0.01	0.05	-0.02
Tic	-0.11	-0.60	0.20	0.06	-0.03
Detmatology	0.30	0.15	0.06	0.39	-0.04

Tabel 3.23 – Coeficienţii de corelare dintre variabila dependentă şi variabilele independente

Potrivit coeficienţilor de corelare, parametrul *NMIR* are o influenţă importantă asupra ieşirii şi creşterea lui determină de obicei o scădere a performanţei (excepţie face setul de date dermatology). Parametrul *NF* ia în general valori în jurul lui 0, deci nu are o influenţă foarte mare asupra ieşirii, excepţie de la această regulă o constituie seturile de date mari mushrooms şi nursurey, în care se observă că creşterea lui *NF* determină o scădere puternică a ieşirii.

Creşterea parametrului *NMINR* determină o creştere uşoară a ieşirii, dar sunt şi câteva situaţii în care ieşirea este influenţată slab, într-un mod negativ de creşterea acestui parametru. În cazul tuturor seturilor de date creşterea lui *NC* determină creşterea valorii variabilei dependente. Parametrul *NI* are de regulă o influenţă negativă asupra ieşirii, excepţie face doar setul de date zoo.

Valorile coeficienţilor din modelele de regresie şi din matricea de corelare demonstrează o parte din observaţiile făcute în analiza empirică şi anume:

- creşterea numărului de furnici nu conduce la o acurateţe mai bună şi din contră, dacă setul de date este mare, duce la o scădere a acurateţei prin faptul că algoritmul nu ajunge la rezultat
- creşterea valorii parametrului număr minim de instanţe acoperite de fiecare regulă descoperită (*NMIR*) determină o scădere a acurateţei

3.3.4. Consideraţii şi recomandări cu privire la alegerea parametrilor algoritmului Ant Colony Optimization

Rulările au fost realizate rulând aplicaţia în paralel pe un număr de 30 de calculatoare dintr-un laborator al Universităţii "Politehnica" din Timişoara. Toate calculatoarele au o configuraţie identică Intel Core 2 Quad CPU, 2,4 GHz, 2,4 GHz , 4 GB RAM. Au fost rulate şi analizate 64 de configuraţii de parametri pe fiecare din cele 10 seturi de date alese.

Analiza făcută a demonstrat că nu există o configuraţie de parametri universală care să conducă la rezultate mai bune decât altele pe orice set de date, dar a facilitat analiza comportamentului algoritmului pe diferite seturi de date şi extragerea unor concluzii şi recomandări utile cu privire la alegerea acestor parametri.

Concluziile cu privire la influenţa pe care o au parametrii algoritmului asupra acurateţei predicţiei şi asupra timpului de execuţie şi recomandările cu privire la alegerea lor sunt prezentate în continuare.

- Numărul de furnici este numărul maxim de reguli candidate, construite şi tăiate (sau fasonate) în timpul unei iteraţii a buclei repetitive a algoritmului. O furnică este asociată unei reguli. Cu cât este mai mare numărul de furnici cu atât mai multe reguli candidate sunt evaluate pe iteraţie şi cu atât mai lent este sistemul. Experimentele au arătat că, creşterea numărului de furnici determină creşterea direct proporţională a timpului de execuţie, fără să garanteze o creştere a performanţei. Dacă numărul de furnici este ales foarte mare (de genul 1000, 1500) şi setul de date este mare (comparabil ca dimensiune cu *mushrooms* sau *nursurey*) există o şansă mare ca aplicaţia să facă utilizatorul să aştepte mult fără să ajungă la un rezultat. Rezultate bune s-au obţinut pentru numărul de 15 furnici. Alegerea unui număr de furnici mare duce cu siguranţă la un timp de execuţie mare dar nu se poate spune ca duce şi la o creştere a acurateţei predicţiei, aşa ca nu recomandăm alegerea unui număr mai mare de 500 de furnici
- Creşterea *NMIR* conduce la o scădere a acurateţei predicţiei şi conduce la multe situaţii de blocare. Cele mai bune rezultate s-au obţinut pentru varianta 2, în care *NMIR* ia valoarea 5. Creşterea *NMIR* poate determina algoritmul să intre în buclă infinită în multe situaţii. Ieşirea din aceste situaţii de blocare se poate realiza prin creşterea *NMINR*. Datorita faptului că testele în care acest parametru a luat valoarea 15 au făcut ca algoritmul să nu poată fi rulat pe 4 seturi de date din 10, nu recomandăm alegerea unei valori mai mari de 10 pentru acest parametru. Dacă este necesară alegerea unei valori mai mari din diverse motive, atunci creşterea *NMINR* poate mări şansele de a obţine rezultate bune.
- Creşterea *NMINR* în general duce la o scădere a performanţei, (lucru confirmat şi de rezultatele din tabelul 3.12) excepţie fac cazurile în care sunt puse condiţii restrictive pentru *NMIR*, care conduc la bucle infinite, care pot fi rezolvate prin creşterea *NMINR* (lucru confirmat de rezultatele din tabelul 3.10). Nu recomandăm alegerea unui număr mai mare de 10 pentru acest parametru decât dacă *NMIR* este mai mare de 10.
- Cu cât numărul de reguli pentru convergenţă este mai mare cu atât timpul de execuţie este mai mare. Testele făcute au dus la un număr egal de 3

maxime pentru acurateţea predicţiei în cazul în care NC ia pe rând valorile 2, 10 şi 20. În această situaţie varianta preferată este cea care are valoarea cea mai mică pentru *NC* deoarece ea se execută mai repede. Întrucât în bucla repetitivă sunt puse condiţii de oprire legate de numărul de reguli pentru convergenţă dar şi de numărul de iteraţii, alegerea unei valori foarte mici pentru numărul de reguli pentru convergenţă face ca parametrul *NI* să nu mai conteze. De aceea influenţa parametrului *NI* a fost analizată şi în cazul în care *NC* are valoarea 2 dar şi în cazul în care are valoarea 30. Pe de altă parte numărul de reguli pentru convergenţă reprezintă o garanţie că regula găsită este bună. Cu cât mai multe furnici converg către o regulă cu atât mai multe şanse sunt ca regula să fie bună. Ţinând cont de rezultatele obţinute dar şi de semnificaţiile mărimilor credem că o valoare între 5 şi 10 este potrivită pentru acest parametru.

- Un număr mic de iteraţii determină execuţia mai rapidă a algoritmului, iar creşterea numărului de iteraţii peste o anumită valoare (în cazul experimentelor noastre, valoarea 50) poate să nu modifice decât puţin timpul de execuţie (datorită faptului că bucla *while* se va opri nu datorită condiţiei legate de numărul de iteraţii ci condiţiei legate de *NC*). În ceea ce priveşte acurateţea predicţiei cel mai bun rezultat s-a obţinut pentru *NI* =10 dacă *NC*=2, şi *NI*=100, dacă *NC*=30. Întrucât *NI* şi *NC* sunt condiţii de oprire a aceleaşi bucle repetitive, când se alege o valoare pentru *NI* trebuie să se ţină seama de valoarea lui *NC*.

Studiul empiric realizat a arătat că acurateţea predicţiei poate fi influenţată cu 10 % de valorile alese pentru parametrii algoritmului, iar timpul de execuţie poate fi de la câteva secunde la câteva ore în funcţie de valorile acestor parametri. Analiza unui set de date cu ajutorul aplicaţiei *Ant Miner* necesită testarea mai multor configuraţii de parametri şi alegerea celei care duce la cea mai bună performanţă. Recomandările de mai sus cresc şansele de a obţine un raport bun între acurateţea predicţiei şi timpul de execuţie.

Analiza matematică a coeficienţilor a permis construirea unor modele de regresie pe baza rezultatelor experimentale obţinute efectuând câte 64 de măsurători pe fiecare set de date şi construirea unor matrice de corelare care redau legătura dintre parametrii de intrare şi parametrul de ieşire ales (acurateţea predicţiei). Analiza matematică a demonstrat prin valorile obţinute pentru coeficienţii de regresie respectiv prin valorile termenilor din matricea de corelare că:

- Un număr mare de furnici nu duce la o acurateţe mai bună, ba chiar poate duce la o acurateţe slabă daca setul de date analizat este mare si *NF* este foarte mare
- Creşterea *NMIR* determină o scădere a performanţei.

O direcţie de dezvoltare ulterioară este analiza influenţei pe care o au parametrii de mai sus asupra numărului de reguli şi numărului de condiţii descoperite şi găsirea de secvenţe de parametri care se potrivesc pentru anumite tipare de seturi de date.

3.4. Variante ale algoritmului *ACO* pentru descoperirea regulilor de clasificare

În [PAR02] autorii au comparat rezultatele obţinute cu *Ant Miner* cu cele obţinute cu algoritmul clasic *CN2*, pe 6 seturi de date. Pe 4 din acestea a fost mai bun *Ant Miner*, pe unul a fost mai bun *CN2* şi în unul au fost la egalitate. De aceea au fost consideraţi echivalenţi în ceea ce priveşte acurateţea predicţiei, dar *Ant Miner* s-a dovedit net superior în ceea ce priveşte numărul de reguli descoperite şi numărul de condiţii din aceste reguli. Cu cât regulile sunt mai puţine şi mai simple cu atât sunt mai uşor de folosit. Direcţiile de dezvoltare sunt modificarea Ant Miner în aşa fel încât să poată lucra cu atribute continue, ca să nu mai fie necesară discretizarea lor într-un pas de preprocesare precum şi analiza modului în care se comportă algoritmul cu alte funcţii euristice sau de actualizare a feromonului.

În [PAR01] autorii au comparat algoritmul cu algoritmul clasic *C4.5*, pe 6 seturi de date. În ceea ce priveşte acurateţea predicţiei rezultatele au fost comparabile, dar *Ant Miner* a fost net superior în ceea ce priveşte numărul de reguli şi numărul de condiţii din fiecare regulă. O direcţie de cercetare din [PAR01] este analiza senzitivităţii algoritmului la parametrii săi. Alte direcţii de cercetare se referă la funcţiile de actualizare a feromonului, funcţia euristică, şi lucrul cu parametrii continui.

În [LIU02a] se subliniază faptul că funcţia euristică din Ant Miner se bazează pe calculul entropiei, care este destul de costisitor din punct de vedere computaţional. Autorii au propus o altă funcţie euristică, mult mai simplă care se bazează pe densitate, aşa cum se vede în formula 3.10. $Majority\_classT_{ij}$ reprezintă numărul de instanţe care aparţin clasei majoritare din partiţia în care atributul i ia valoarea j. $|T_{ij}|$ reprezintă numărul de instanţe pentru care atributul i are valoarea j. S-a demonstrat pe un set de date că rezultatele obţinute sunt aceleaşi în ceea ce priveşte acurateţea predicţiei şi numărul de reguli şi de condiţii ca şi în *Ant miner*.

$$\eta_{ij} = \frac{majority\_classT_{ij}}{\left|T_{ij}\right|} \tag{3.10}$$

În [LIU03a] autorii afirmă că, datorită modului de actualizare a cantităţii de feromon din *Ant Miner,* furnicile converg prea repede către o singură regulă construită şi au propus o metodă de actualizare a cantităţii de feromon din formula 3.11. Cu ajutorul parametrului ρ se poate controla cât de repede se evaporă feromonul. Cu cât ρ este mai mare cu atât mai repede se evaporă feromonul.

$$\tau_{ij}(t) = (1-\rho)\cdot\tau_{ij}(t-1) + \left(1 - \frac{1}{1+Q}\right)\cdot\tau_{ij}(t-1) \tag{3.11}$$

Pentru a mări probabilitatea de a alege termeni care nu au fost aleşi în regulile precedente, au fost introduse două numere aleatoare subunitare q_1 şi q_2. Dacă numărul q_1 generat aleatoriu este mai mic decât un prag φ care poate fi

configurat, atunci se alege în mod aleatoriu, proporţional cu performanţa, un termen *ij*, iar dacă este mai mare atunci se alege termenul care are cea mai mare valoare pentru P_{ij}.

If ($q1 \le \varphi$)
Loop
If ($q2 \le \sum_{j \in J_i} P_{ij}$) atunci alege T_{ij}
Endloop
Else
Alege termenul T_{ij} cu max P_{ij}

A fost demonstrat pe 2 seturi de date că *Ant Miner 3* dă rezultate mai bune pentru acurateţea predicţiei şi numărul de reguli este mai mare în *Ant Miner 3*. Acurateţea a fost calculată cu formula :

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3.12}$$

În [SAL11] autorii au propus 5 extensii ale algoritmului şi au testat 12 combinaţii ale acestor extensii pe 23 de seturi de date. De asemenea au fost evaluate performanţele a 3 algoritmi performanţi *Ripper*, *PART*, *C4.5* pe aceste 23 de seturi de date. S-a obţinut că C4.5 are cea mai bună performanţă. Apoi s-au comparat rezultatele celei mai bune variante de *Ant Miner* din cele 12 încercate cu performantele obţinute de *C4.5* şi s-au obţinut rezultate comparabile în ceea ce priveşte acurateţea predicţiei, iar în ceea ce priveşte numărul de reguli s-a obţinut că cea mai bună versiune de *Ant Miner* produce reguli mai puţine decât *C4.5*.

Cele 5 extensii propuse sunt următoarele:

- Utilizarea unor multiple tipuri de feromon, câte un tip pentru fiecare clasă; O furnica întâi selectează clasa regulii şi apoi depozitează tipul corespunzător de feromon. În *Ant Miner*, versiunea originală, întâi se construieşte antecedentul regulii şi apoi regula aste atribuită la clasa pe care antecedentul îl acoperă cel mai bine. În versiunea modificată întâi se alege clasa regulii şi apoi termenii care se vor adăuga în antecedentul regulii, alegerea termenilor se face ţinând seamă de clasa aleasă. O furnică este influenţată nu de toate furnicile ci doar de acelea care au construit reguli cu acelaşi consecvent (ca şi când ar fi diferite tipuri de feromon şi o furnică nu este influenţată de orice tip). Efectul acestei schimbări a fost nevoia de a modifica structurile bidimensionale (atribut, valoare) ce păstrează cantitatea de feromon şi valorile euristice în structuri tridimensionale (atribut, valoare, clasa)
- S-a utilizat un intensificator al calităţii pentru a premia regulile care au o calitate bună şi pentru a penaliza regulile care au o calitate slabă în termeni de feromon.

$$\Delta\tau(t)_k = \begin{cases} 2 \cdot Q(R_t), \ if\,(confidence(R_t) \ge \phi_1) \\ Q(R_t), \ if\,(\phi_1 \ge confidence(R_t) \ge \phi_2) \\ Q(R_t) - 2, if\,(\phi_2 \ge confidence(R_t)) \end{cases} \tag{3.13}$$

Unde $\Delta\tau(t)_k$ este cantitatea de feromon de tip *k* care va fi depozitată de furnica t pe fiecare pereche atribut-valoare care aparţine antecedentului regulii R_t precum şi clasei *k*. $Q(R_t)$ este calitatea regulii, iar ϕ_1 şi ϕ_2 sunt praguri de confidenţă la care calitatea este contrastată. Calitatea regulii este evaluată cu altă funcţie decât în *Ant Miner*, cea din formula:

$$Q(R_r) = \frac{TP}{\underbrace{|TrainingSet|}_{Support(R_t)}} + \frac{TP}{\underbrace{|Matches|}_{Confidence(R_t)}} \tag{3.14}$$

- S-a permis utilizarea operatorului de negare logică în antecedentul regulii, regulile descoperite fiind de forma de mai jos:
  If(A^i=V_{ij}) and (A_k NOT=V_{kl}) and…
  Noduri de negare sunt adăugate pentru un atribut dacă are cel puţin 3 valori. Regulile construite în acest fel au o mai puternică acoperire a setului de date decât celelalte, deci se va produce o lista de reguli mai scurtă
- Au fost incluse furnicile încăpăţânate care ţin cont de propria istorie atunci când adaugă termeni unei reguli. Fiecare furnică memorează cea mai bună regulă pe care a construit-o mergând pe urmele proprii. Fiecare furnică lasă un număr de urme care este configurabil.
- S-au utilizat furnici cu personalitate care au propriile valori pentru α şi β (coeficienţi pentru feromon şi pentru funcţia euristică). Personalitatea se manifestă prin faptul că unele furnici vor da mai multa importanţă informaţiei legate de feromon iar altele informaţiei legate de funcţia euristică. Valorile pentru α şi β sunt întregi şi cuprinse intre 1 şi 3. Ele sunt selectate probabilistic de către fiecare furnică înainte de construcţia regulii, folosind informaţia legată de feromon

În [LAL11] autorii propun modificări importante lui *Ant Miner* care îi permit să descopere reguli care în antecedent pot avea operatorii AND, OR, XOR, NOT. Antecedentul fiecărei reguli descoperite poate fi reprezentat sub forma unui arbore binar, în care nodurile interioare corespund operatorilor acceptaţi AND, OR, XOR, iar frunzele corespund perechilor *atribut=valoare* sau *atribut not =valoare*. Metoda propusă se numeşte *TREE MINER*. Pentru a construi o regulă o furnică trebuie să traverseze un graf de reguli. A altă diferenţă faţă de Ant Miner este că nu mai este necesar elagajul regulilor. Pentru actualizarea feromonului a fost introdus un nou parametru numit *conv_rate* care poate fi configurat astfel încât mărirea cantităţii de feromon pentru termenii adăugaţi în regulă să se poată face mai mult sau mai puţin. Dacă cantitatea de feromon este mai mare atunci regulile converge mai repede, altfel mai târziu, asta favorizând explorarea. Aplicaţia a fost testa pe 4 seturi de date şi comparata cu *Ant Miner*, în ceea ce priveşte acurateţea predicţiei rezultatele sunt comparabile iar numărul de reguli descoperit de *Tree Miner* este mai mic.

În [SAI11] a fost modificată funcţia care asigura elagajul regulilor şi a fost introdusă o funcţie care asigură evaporarea dinamică a feromonului. Algoritmul a fost rulat pe 5 seturio de date şi comparat cu *Ant Miner*, iar rezultatele arată că modificările propuse au dus la îmbunătăţirea acurateţei predicţiei, regulile descoperite sunt mai uşor de înţeles şi timpul de execuţie este mai scurt.

În [SMA06] a fost propus algoritmul *MulO-AntMiner* care permite predicţia a mai multe atribute clasa. Ca un efect consecventul unei reguli poate sa conţină diferite atribute.

În [WAN04] a fost propusă o nouă versiune a lui *Ant Miner*, versiune care poate determina un set de reguli neordonate (În ant miner regulile descoperite trebuie aplicate instanţelor în ordinea descoperirii). Regulile descoperite cu versiunea de *Ant Miner* din [WAN04] pot fi aplicate instanţelor în orice ordine.

3.5. Soluţii de îmbunătăţire a algoritmului ACO

Utilizarea algoritmului *Ant Colony Optimisation* la descoperirea regulilor de clasificare a fost propusa de *Parpinelli et al.* în anul 2001. Algoritmul propus de ei a fost întâi implementat în limbajul C şi testat pe diverse seturi de date. Ulterior algoritmul a fost implementat în Java şi distribuit sub forma unui sistem Open Source numit *Ant Miner*. Ultima versiune este versiunea 1.2.1, dezvoltata în anul 2007 de catre Fernando Meyer în colaborare cu lui Parpinelli. În fig. 3.3 este prezentată interfaţa principală a aplicaţiei. În cadrul acestui subcapitol au fost propuse soluţii de îmbunătăţire a algoritmului *Ant Colony Optimisation* şi a modului de utilizare a acestuia la descoperirea regulilor de clasificare. Îmbunătăţirile propuse vizează modul de calcul a cantităţii iniţiale de feromon asociat fiecărui termen, modul de actualizare a cantităţii de feromon pentru termenii ce fac parte din regula descoperită, modul de evaluare a calităţii regulilor, modul de alegere a termenilor care vor face parte din noua regulă. Modificările au fost implementate în Ant Miner, şi aplicaţia care le conţine a fost numită *Ant-r-Miner*. În figura 3.4 este prezentată interfaţa grafică a aplicaţiei extinsă.

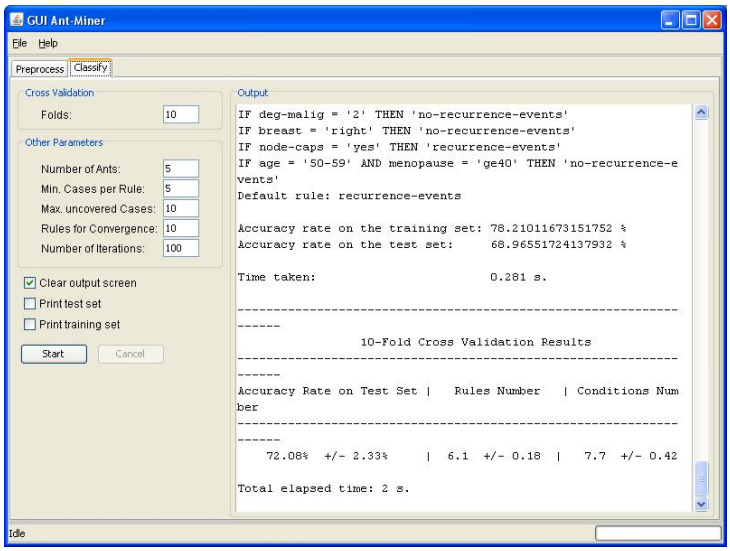

Fig. 3.3 - Ant Miner 1.2.1

Funcţia propusă pentru a evalua calitatea unei reguli în *Ant-r-Miner* realizează o sumă între acurateţea predicţiei şi comprehensibilitatea regulii. Acurateţea regulii este exprimată ca un raport între numărul de instanţe care au

întradevăr clasa prezisă de regulă raportat la toate instanţele acoperite de regulă (din care unele au clasa acoperită de regulă, iar altele nu au clasa acoperită de regulă). Comprehensibilitatea unei reguli a fost exprimată ţinând cont de numărul de condiţii din antecedentul regulii. Se consideră că o regulă este cu atât mai valoroasă cu cât este mai uşor de înţeles (are comprehensibilitatea mai mare). Comprehensibilitatea este subunitară şi este cu atât mai mare cu cât regula are mai puţine condiţii în antecedent. Funcţia propusă este cea din formula 3.15. Termenul *nar* reprezintă numărul de condiţii din antecedentul regulii (numărul de atribute din regulă), unite prin AND (un atribut nu poate apărea decât o data într-o regulă) iar termenul *nta* reprezintă numărul total de atribute.

$$Q = tp / (tp+fp) + (nta - nar) / nta \quad (3.15)$$

În *Ant-Miner* fiecărei perechi atribut-valoare i se atribuie o cantitate iniţială de feromon calculată cu formula 3.1. În tabelul de mai jos cu FI se notează feromonul iniţial iar cu AP acurateţea predicţiei. Cantitatea de feromon din coloana FI s-a obţinut cu formula 3.1. Acurateţea predicţiei pentru valorile iniţiale este în coloana AP.

Setul de date	FI	AP (FI)	AP (FI= 0.2)	AP (FI= 0.5)	AP (FI= 1)	AP (FI= 5)	AP (FI= 10)
Car	0.08	84.89	85.24	84.5	***85.59***	85.48	84.66
Zoo	0.019	88.15	90.12	***91.41***	89.12	90.1	90.03
Ljubljana breast cancer	0.018	73.86	***74.68***	73.14	72.69	73.46	72.32
Diabetes	0.052	***73.96***	72.26	73.31	73.55	71.34	72.13
Cleveland heart disease	0.068	77.31	77.54	78.18	75.49	***78.54***	74.85
Iris	0.105	93.33	94.67	94	***95.33***	94.67	94.67
Mushrooms	0.007	96.38	***97.93***	96.15	96.05	96.16	97.01
Nursurey	0.072	84.02	85.52	85.41	***85.63***	84.88	83.57
Tic	0.034	70.14	72.33	72.02	***74.2***	74.21	74.1
Dermatology	0.018	***92.64***	89.64	88.83	90.75	90.04	91.59

Tabel 3.24 - acurateţea predicţiei pentru diferite valori ale cantitatii initiale de feromon

Analizând rezultatele din tabelul 3.24 se poate constata că se obţin valori comparabile pentru acurateţea predicţiei chiar dacă feromonul iniţial se calculează cu formula 3.1, ca în *Ant Miner* sau se aleg pentru feromonul iniţial valori constante. Atribuirea unor valori constante feromonului iniţial este de preferat pentru că este mai rapidă computaţional decât calculul sumei numărului de valori pe care îl are fiecare atribut categorial şi împărţirea lui $\log_2(k)$ la acesta suma (unde k reprezintă numărul de clase). După cum se observă din tabelul 3.25 cele mai bune valori s-au obţinut dacă feromonul iniţial ia valoarea 1. În *Ant-r-Miner* iniţial fiecare termen are asociată cantitatea 1 de feromon.

În urma descoperirii unei reguli este necesară actualizarea cantităţii de feromon a tuturor termenilor. Termenii care au fost aleşi în regulă corespund traseelor urmate de furnică, deci cantitatea de feromon asociată lor se va mări, iar cantitatea de feromon a termenilor care nu fac parte din regulă trebuie diminuată pentru a simula fenomenul de evaporare. Creşterea cantităţii de feromon a termenilor din regulă se face în *Ant-r-Miner* pe baza principiului de recompensare a

regulilor bune respectiv de penalizare a celor slabe din [SAL11]. Diferenţa faţă de [SAL11] este ca se utilizează un singur factor care asigură intensificarea, respectiv diminuarea calităţii. Acest factor a fost numit *Quality factor* (factor de calitate) şi utilizatorul poate stabili care este valoarea lui, precum şi a limitelor în care el acţionează (vezi formulele 3.16 şi 3.17). Factorul de calitate a fost notat cu *n* şi poate lua doar valori mai $\geq$ 1. Dacă calitatea regulii este sub o limita inferioara l_1 stabilită de utilizator atunci cantitatea de feromon va creste cu $\tau_{ij} \cdot Q/n$, pentru a penaliza regula slabă. Dacă calitatea regulii este intre cele 2 limite stabilite, atunci cantitatea de feromon creste exact ca în Ant Miner cu $\tau_{ij} \cdot Q$. Dacă calitatea regulii este peste o limită superioară stabilită (l_2), atunci cantitatea de feromon se măreşte cu $\tau_{ij} \cdot Q \cdot n$, pentru a premia regula bună. În cazul în care se doreşte utilizarea acestei facilităţii se poate fie să se aleagă pentru l_1 valoarea 0 şi pentru l_2 valoarea 1 fie să se aleagă valoarea 1 pentru n. l_1 şi l_2 trebuie sa ia valori reale subunitare.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot \lambda \text{ ,} \tag{3.16}$$

$$\lambda = \begin{cases} Q/n, \forall Q < l_1 \\ Q, \forall Q, l_1 \leq Q \leq l_2 \\ Q \cdot n, \forall Q > l_2 \end{cases} \tag{3.17}$$

În [LIU02a] autorii au demonstrat că se pot obţine rezultate la fel de bune şi cu o funcţie euristică mai simplă, în care în loc de calculul entropiei, care este destul de costisitor, din punct de vedere computaţional se calculează o densitate. De aceea funcţia euristică utilizată în *Ant-r-Miner* este cea din formula 3.10, lucrarea [LIU02a].

În Ant Miner furnicile converg repede către regula construită, de aceea în *Ant-r-Miner* a fost introdus factorul de convergenţă ($0 \leq cf \leq 1$), configurabil de utilizator. Dacă utilizatorul alege valoarea 1 pentru acest factor, atunci se va alege termenul cu probabilitatea maximă. O valoare apropiată de 0 a factorului de convergenţă face ca alegerea termenilor să fie aleatoare.

```
double r=(random.nextInt() << 1 >>> 1) % 101 ;
r=r / 100; // r numar aleator intre 0 si 1

if ( r <= cf )
     alege Tij cu Pij maxim
else
     alege Tij prin selecţia de tip ruletă
```

Figura 3.4 prezintă interfaţa aplicaţiei *Ant-r-Miner/*. După cum se vede, în interfaţă a fost adăugat un nou *JPanel*, denumit *New Parameters* în care utilizatorul poate introduce valori pentru parametrii noi introduşi (*Quality Factor, Lower limit, Upper Limit, Convergence factor*).

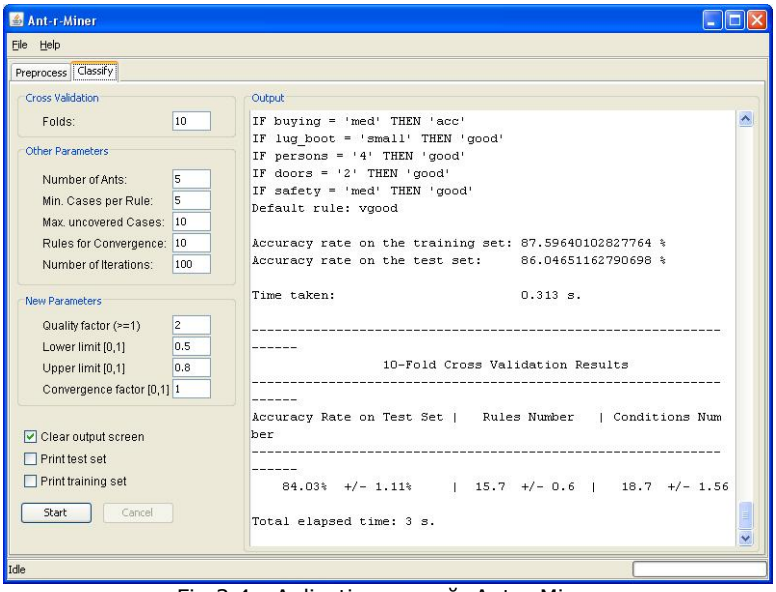

Fig 3.4 - Aplicaţia propusă, Ant-r-Miner

Algoritmul propus a fost testat pe 10 seturi de date şi rezultatele lui au fost comparate cu cele obţinute cu Ant Miner. Valorile alese pentru parametrii algoritmului au fost NF=15, NMIR=5, NMINR=10, NC=10, NI=100.

Pentru parametrii noi introduşi în Ant-r-Miner s-au ales valorile: *Quality Factor=2, Lower limit=0.5, Upper Limit=0.8, convergence factor=0.7,*

Tabelul 3.25 prezintă comparativ acurateţea predicţiei şi timpul de execuţie obţinute testând Ant Miner şi Ant-r-Miner pe seturile de date alese.

Setul de date	Ant Miner (AP)	Ant-r-Miner (AP)	Ant Miner (TE)	Ant-r-Miner (TE)
Ljubljana breast cancer	72.74	***72.95***	7	***6***
Car	85	***89.7***	19	***10***
Dermatology	***95.59***	89.9	158	***86***
Diabetes	73.56	***77.35***	10	***4***
Cleveland heart disease	***81.24***	78.51	***4***	9
Iris	94.6	***95.33***	0	***0***
Mushrooms	96.15	***99***	1566	***1002***
Nursurey	86.52	***88.68***	183	***107***
Tic	72.12	***81.83***	23	***8***
Zoo	***91.23***	86.26	31	***19***

Tabel 3.25 – acurateţea predicţiei cu Ant-Miner şi Ant-r-Miner

Rezultatele din tabelul 3.25 arată că se obţin rezultate puţin mai bune cu *Ant-r-Miner* decât cu *Ant Miner* în ceea ce priveşte acurateţea predicţiei, iar în ceea ce priveşte timpul de execuţie rezultatele obţinute cu *Ant-r-Miner* sunt mult mai bune. În ceea ce priveşte numărul de reguli şi de condiţii *Ant Miner* este superior.

3.6. Concluzii

Algoritmul *Ant Colony Optimization* este folosit cu succes la clasificarea datelor. Din anul 2001 când a fost propusă utilizarea lui pentru descoperirea regulilor de clasificare, au fost realizate multe studii cu ajutorul acestui algoritm şi problema alegerii unei secvenţe de parametri optime a fost lăsată ca o direcţie de cercetare de foarte mulţi cercetători. În cadrul acestui capitol a fost studiat codul algoritmului şi prezentată în detaliul funcţionarea lui, au fost analizate foarte multe configuraţii de parametri şi s-a emis o listă de recomandări privind alegerea acestora. De asemenea problema îmbunătăţirii algoritmului a fost atinsă de mulţi cercetători. Au fost prezentate sintetic îmbunătăţirile pe care o parte din aceştia le-au adus şi au fost propuse şi operate noi îmbunătăţiri. Acest lucru a fost posibil datorită faptului că aplicaţia care implementează algoritmul se numeşte Ant Miner şi este Open Source.

Contribuţiile aduse în cadrul acestui capitol sunt următoarele:

- A fost realizată o prezentare detaliată a algoritmului, bazată atât pe documentaţia existentă cât şi pe studiul codului algoritmului.
- Au fost studiaţi parametrii algoritmului rulând câte 64 de configuraţii pe fiecare din cele 10 seturi de date alese şi analizând rezultatele. Analiza rezultatelor s-a făcut atât vizual, comparativ cât şi matematic construind şi analizând modele de regresie şi matrice de corelare
- Analiza făcută a condus către o listă de recomandări privind modul de alegere al parametrilor, utile pentru toţi utilizatorii acestui algoritm
- A fost realizată o sinteză a îmbunătăţirilor aduse algoritmului din anul 2001 şi până în prezent
- Au fost aduse îmbunătăţiri algoritmului care vizează modul de calcul al cantităţi iniţiale de feromon a termenilor, modul de actualizare a feromonului, modul de evaluare a calităţii regulilor descoperite şi modul de selecţie a termenilor. Modificările propuse au fost implementate în aplicaţia open source *Ant Miner*. Aplicaţia care conţine algoritmul îmbunătăţit a fost numită *Ant-r-Miner*. S-a demonstrat că îmbunătăţirile aduse conduc la o mai bună performanţă utilizând date reale.

Pe viitor se doreşte extinderea aplicaţiei oferindu-i utilizatorului posibilitatea să aleagă dintr-o listă de funcţii, funcţia pe care doreşte să o folosească pentru a evalua calitatea regulilor. O altă direcţie de dezvoltare este de oferire a posibilităţii utilizatorului să-şi construiască în mod dinamic funcţia de calitate, cu ajutorul termenilor *TP, FP, TN, FN* şi a unor operaţii elementare gen adunare, scădere, înmulţire, împărţire. Analiza influenţei pe care o au valorile parametrilor asupra numărului de reguli şi asupra numărului de condiţii este de asemenea o direcţie de dezvoltare ulterioară.

4. Clasificarea datelor cu ajutorul algoritmului genetic *AGR*

4.1. Noţiuni introductive

Algoritmii genetici sunt tehnici adaptive care pot fi folosite cu succes în rezolvarea unor probleme de căutare şi optimizare complexe [MAU00]. Ei sunt bazaţi pe principiile geneticii şi ale selecţiei naturale enunţate de Darwin ("supravieţuieşte cel mai bine adaptat"). Algoritmii genetici au fost utilizaţi cu succes în *data mining*, pentru a determina reguli de clasificare [DEH06], pentru a căuta centrele de clustere adecvate [MAU00], pentru a selecta atributele de interes în prezicerea valorii unui atribut ţintă [FRE02], etc. Clasificarea instanţelor a fost realizată şi prin utilizarea unor algoritmi hibrizi, bazaţi pe algoritmi genetici şi *Particle Swarm Optimization* [DIN09], respectiv *Naïve Bayes* şi *k-Nearest Neighbor* [AIC10]. Câteva aplicaţii în care algoritmii genetici au fost aplicaţi cu succes pentru a rezolva probleme de clasificare sunt: clasificarea amprentelor, clasificarea bolilor de inimă, clasificarea emoţiilor de pe faţa umană, clasificarea imaginilor de la sateliţi [YAN06], etc.

Capitolul prezintă sintetic aspecte privind algoritmii genetici utilizaţi la clasificarea datelor, funcţiile *fitness* ale acestora şi propune un nou algoritmul genetic pentru clasificarea datelor. Algoritmul genetic propus a fost implementat în Open Source-ul *Weka*, utilizând limbajul *Java*. Mediul de dezvoltare utilizat a fost *Eclipse*. Sunt prezentate aspecte legate de procesările algoritmului, de implementare şi de rezultatele experimentale obţinute prin aplicarea algoritmului pe 5 seturi de date obţinute din *UCI Machine Learning Repository*. S-a analizat comportamentul algoritmului utilizând 14 variante de parametri. A fost analizat comportamentul algoritmului atât în cazul în care se adaugă la lista regulilor descoperite toate regulile descoperite după epuizarea numărului de generaţii cât şi în cazul în care se adaugă doar cea mai bună regulă descoperită după rularea numărului de generaţii stabilit. Sunt comparate rezultatele obţinute rulând algoritmul cu 3 funcţii *fitness* diferite. Funcţia *fitness* care a condus la cele mai bune rezultate a fost inclusă în algoritm.

Rezultatele obţinute de *AGR* pe seturile de date alese au fost comparate cu cele obţinute de algoritmii *Naive Bayes*, *k-Nearest-Neighbour* şi *J48* pe aceleaşi seturi de date şi în aceleaşi condiţii. Concluzia a fost că algoritmul propus este competitiv, poate fi utilizat la clasificarea datelor putând să conducă la rezultate mai bune pe anumite seturi de date decât alţi algoritmi clasici.

4.2. Structura unui algoritm genetic

Forma generală a unui algoritm genetic este următoarea [MUK09]:

```
Determină populaţia iniţială;
WHILE a condiţie de oprire nu este îndeplinită DO
BEGIN
        Selectează indivizii pentru reproducere;
        Creează moştenitori prin încrucişarea indivizilor selectaţi;
        Eventual aplică mutaţia unor indivizi;
        Calculează următoarea generaţie
END
```

Populaţia iniţială este aleasă aleatoriu. Selecţia este procedura prin care sunt aleşi indivizii ce vor participa la formarea generaţiei următoare. Indivizilor mai puternici, mai bine adaptaţi li se vor da mai multe şanse de a face parte din populaţia următoare. Performanţa indivizilor este măsurată cu ajutorul unei funcţii *fitness* care depinde de problema de rezolvat. Selecţia indivizilor pentru reproducere se poate face în mai multe moduri. Cea mai populară variantă este selecţia de tip ruletă. Fiecare cromozom are asociată o felie a ruletei. "Feliile ruletei sunt cu atât mai late cu cât *fitness*-ul cromozomului asociat feliei este mai mare".

O altă tehnică de selecţie este selecţia bazată pe rang. Aceasta previne convergenţa prematură ce apare la utilizarea schemelor de selecţie proporţională cu *fitness*-ul. Indivizii sunt ordonaţi în funcţie de valoarea *fitness*-ului iar probabilitatea de selecţie este proporţională cu rangul ocupat.

Selecţia prin trunchiere, elimină o parte din cromozomii cu cea mai slabă performanţă, iar în locul lor se generează alţii, după diferite scheme posibile. De exemplu se alege un nou cromozom x, folosind principiul ruletei. Dacă x diferă de toţi ceilalţi cromozomi ai populaţiei actuale, atunci el este inclus în populaţie; în caz contrar, este supus operatorului de "mutaţie" până ce devine diferit de ceilalţi cromozomi şi este inclus în populaţie.

Selecţia turneu alege în mod aleatoriu k indivizi iar dintre aceştia doar cei mai buni j sunt selectaţi pentru supravieţuire. Procedura se repetă până se obţine numărul dorit de indivizi. Este cea mai eficientă din punct de vedere al complexităţii timp.

Selecţia de tip elitist păstrează în generaţia următoare, cei mai buni k indivizi ai generaţiei curente. Dacă nu se reţin aceşti indivizi ei pot să dispară din cauză că nu au fost selectaţi pentru supravieţuire sau fiindcă au fost distruşi prin aplicarea operatorilor. Acest lucru garantează faptul că cel mai puternic membru al populaţiei din generaţia următoare nu va fi mai slab decât cel din generaţia curentă.

Încrucişarea a doi indivizi se poate face într-un singur punct sau în mai multe puncte. Noii indivizi ai populaţiei sunt obţinuţi prin combinarea unor părţi alternative de material genetic provenind de la părinţi.

Mutaţia permite algoritmului genetic să găsească noi soluţii în cadrul populaţiei şi îl protejează împotriva pierderii de informaţie în cazul unor încrucişări nepotrivite. Dacă operatorul de selecţie reduce diversitatea în populaţie, cel de mutaţie determină o nouă creştere a diversităţii. Cu cât probabilitatea mutaţiei este mai mare, cu atât mai redus este riscul convergenţei premature, dar apare un nou risc datorită faptului că o rată mare de mutaţie va transforma algoritmul genetic într-un algoritm de căutare aleatoare.

4.3. Algoritmi genetici utilizaţi la clasificarea datelor

Algoritmii genetici propuşi în literatură pentru clasificarea datelor, au structura din subcapitolul 4.2, dar utilizează diverse moduri de reprezentare a cromozomilor, de calcul al funcţiilor *fitness*, de încrucişare a indivizilor, etc.

În [DEH06] un cromozom e reprezentat ca o înşiruire de m biţi (vezi formula 4.1).

$$m = \sum_{i=1}^{n} nr(a_i) + n - 1 \tag{4.1}$$

Unde,

- *n* reprezintă numărul de atribute,
- a_i este atributul i
- $nr(a_i)$ reprezintă numărul de valori ale atributului i

Pentru reprezentarea fiecărei perechi atribut-valoare, $a_i=v_{ij}$, unde a_i este atributul *i* şi v_{ij} este valoarea j a atributului i, sunt necesari $nr(a_i)+1$ biţi, câte un bit pentru fiecare valoare posibilă a atributului şi un bit a cărui valoare indică dacă atributul face parte din regulă sau nu.

Funcţia *fitness* propusă în [DEH06] este cea din relaţia 4.2. Funcţia calculează performanţa regulii luând în considerare acurateţea predicţiei, claritatea regulii dar şi gradul de interes al regulii.

$$Fitness = \frac{w_1 C(R) + w_2 predAcc + w_3 RInt}{w_1 + w_2 + w_3} \tag{4.2}$$

Unde *C(R)* este comprehensibilitatea, *predAcc* este acurateţea predicţiei şi *RInt* este gradul de interes al regulii, iar w_1, w_2, w_3 sunt ponderi definite de utilizator. Comprehensibilitatea se calculează în (4.3) unde M este numărul maxim de condiţii al unei reguli.

$$C(R)=M\text{- number of condition }(R) \tag{4.3}$$

Acurateţea predicţiei este calculată cu relaţia (4.4).

$$predAcc = \frac{|A \,\&\, C|}{|A|} \tag{4.4}$$

Gradul de interes al regulii se calculează în (4.5) unde InfoGain(Ai) este câştigul informaţional adus de atributul i, şi |dom(G)| este cardinalitatea domeniului pentru atributul clasă G.

$$RInt = 1 - \frac{\sum_{i=1}^{n-1} InfoGain(A_i)}{\frac{n-1}{\log_2(|\,dom(G)\,|)}} \tag{4.5}$$

În [VIV10a] reprezentarea cromozomilor se face la fel ca în [DEH06]. După încrucişare sunt realizate unele prelucrări ale cromozomilor rezultaţi.

În primul rând, se determină pentru fiecare atribut care este valoarea care apare cel mai des în cromozomii rezultaţi. În următorul pas se înlocuieşte valoarea determinată cu o valoare obţinută aleatoriu. Dacă valoarea *fitness-ului* creşte în urma acestei înlocuiri atunci se păstrează valoarea obţinută aleatoriu.

A doua modificare constă în alegerea aleatoare a unui atribut din fiecare cromozom. Dacă el face parte din regulă, se elimină din ea şi se evaluează *fitness-ul* regulii. Dacă acesta a crescut atunci atributul rămâne eliminat. Selecţia utilizată este selecţia de tip ruletă, iar încrucişarea utilizată este încrucişarea în două puncte. Mutaţia se realizează asupra unui singur individ din populaţie ales aleatoriu şi basculând un bit al său ales aleatoriu. Cei mai buni *n* cromozomi sunt selectaţi din părinţi şi copii ca să facă parte din noua generaţie. Restul cromozomilor sunt aleşi din cromozomii copii. Algoritmul se poate opri dacă nu se constată nici o îmbunătăţire în valoarea *fitness-ului* vreme de *m* generaţii sau dacă încă nu a fost generat numărul de generaţii dorit. *Fitness-ul* se calculează cu formula 4.6.

$$Fitness = \frac{tp}{tp + A \cdot Fn} \cdot \frac{tn}{tn + B \cdot fp} \tag{4.6}$$

Unde 0.2<A<2 şi 1<B<20. A şi B sunt parametri definiţi de utilizator care determină rata de convergenţă a soluţiei şi sunt determinaţi experimental în acord cu cerinţele.

În [VIV10b] este propus algoritmul PGA (*Partitioning Genetic Algorithm*), care împarte setul de date în partiţii, algoritmul genetic este aplicat fiecărei partiţii şi un set de reguli este extras de pe fiecare partiţie. Apoi seturile de reguli se combină pentru a forma o listă de reguli candidat. În al doilea pas suportul şi confidenţa se calculează pentru fiecare regulă din regulile candidate folosind întregul set de date. Cromozomii se reprezintă utilizând câte un bit pentru fiecare valoare posibilă a unui atribut şi încă un bit care indică dacă perechea atribut=valoare considerată, face parte din regulă sau nu. Mutaţia face ca un atribut să primească o altă valoare în mod aleatoriu. Au fost dezvoltaţi 2 operatori care controlează dimensiunea regulilor adăugând sau ştergând termeni din reguli. Operatorul de inserare activează o gena punând pe 1 bitul care indică dacă perechea atribut valoare face parte din regulă, iar operatorul de ştergere pune pe 0 acest bit. Cei mai buni cromozomi din generaţia curentă sunt copiaţi în noua generaţie nealteraţi. Funcţia *fitness* propusă este cea din relaţia 4.7:

$$Fitness = w_1 \cdot \frac{|A \,\&\, C - 0.5|}{|A|} + w2 \cdot \frac{L - n}{L - 1} \tag{4.7}$$

Primul termen al funcţiei măsoară acurateţea predicţiei iar cel de-al doilea măsoară comprehensibilitatea regulii.

- |A & C| este numărul de instanţe care satisfac şi antecedentul şi consecventul regulii
- |A| este numărul de instanţe care satisfac antecedentul regulii.
- *L* este numărul maxim de condiţii ale regulii
- *n* este lungimea regulii.

Articolul [FRE02] este unul de sinteză şi prezintă două metode de codificare a cromozomilor, prima dintre ele este cea utilizată în [DEH06] [VIV10a] [VIV10b] iar cea de-a doua este prezentată în continuare. Metoda propusă utilizează pentru fiecare atribut un număr de biţi egal cu numărul de valori posibile ale atributului. Un atribut lipseşte dintr-o regulă dacă toţi biţii alocaţi atributului au fie valoarea 1 fie valoarea 0. Dacă reprezentarea valorii unui atribut conţine mai mulţi de 1, înseamnă că în regulă apare atributul respectiv cu mai multe valori unite prin SAU logic. În ceea ce priveşte reprezentarea consecventului, articolul prezintă mai multe variante:

- consecventul se include în genomul individului, făcându-l astfel subiect al evoluţiei
- valorile consecventului se asociază pe rând cu toţi indivizii din populaţie. Pentru a determina reguli care prezic diferite clase se rulează algoritmul de un număr de ori egal cu numărul de clase.
- se alege clasa cea mai potrivită pentru regulă, imediat ce antecedentul acesteia a fost determinat. Alegerea clasei se poate face după diverse criterii: fie se alege clasa care are cei mai mulţi reprezentanţi care satisfac antecedentul şi consecventul, fie se alege clasa care face ca *fitness*-ul regulii să fie maxim.

Funcţia *fitness* propusă în [FRE02] este cea din formula 4.8.

$$Fitness = w1 \times (CF \times Comp) + w2 \times Simp \quad (4.8)$$

Primul termen din relaţia 4.8 surprinde acurateţea predicţiei, iar al doilea surprinde cât de uşor de înţeles este regula. $CF = tp / (tp+fp)$ este factorul de confidenţă iar $Comp = tp / (tp+fn)$ este factorul de complitudine şi redă care este proporţia de instanţe acoperite de antecedentul regulii din cele care au clasa prezisă de regulă [FRE02]. *Simp* reprezintă simplitatea regulii (este invers proporţională cu numărul de condiţii din antecedentul regulii). *w1* şi *w2* sunt ponderi definite de utilizator.

4.4. *AGR* - Algoritmul genetic propus

Potrivit rezultatelor experimentale prezentate în articolele referite în subcapitolul 4.3, algoritmii genetici au un mare potenţial de aplicare la clasificarea datelor. Cu toate acestea, versiunea 3.6 a lui *Weka*, care este ultima versiune stabilă la ora actuală, nu dispune de un algoritm genetic. Algoritmii genetici din literatură prezentaţi în cadrul subcapitolului 4.3 diferă în ceea ce priveşte codificarea cromozomilor, funcţiile *fitness* care le utilizează, procesările care le efectuează, etc. Problema modului în care aceştia asigură că regulile descoperite pot clasifica orice instanţă nu este lămurită clar. Aşa cum se va vedea în capitolul 4.5 regulile

descoperite după rularea algoritmului pentru numărul de generaţii stabilit nu acoperă toate instanţele. Pe de altă parte câtă vreme în literatură nu există un algoritm genetic clasificator care să asigure o acurateţe a predicţiei de 100 % pe orice set de date, problema analizei şi propunerii unor noi algoritmi genetici în acest scop, rămâne una deschisă.

Pornind de la observaţiile de mai sus în cadrul acestui capitol a fost propus un nou algoritm genetic numit *AGR*, algoritm genetic care diferă după cunoştinţele autorului de cei propuşi în literatură prin următoarele:

- *permite recalcularea clasei în urma încrucişării*. Un cromozom codifică o regulă. În urma încrucişării cromozomilor, clasa cromozomilor rezultaţi este posibil să nu fie cea mai potrivită. De aceea a fost dezvoltată o facilitate care permite recalcularea clasei în urma încrucişării.
- *permite stabilirea unui procent minim de acoperire pentru regulile descoperite.* Algoritmul genetic rulează pentru numărul de generaţii stabilit şi descoperă un set de reguli. Se determină acurateţea regulilor dar şi acoperirea lor. Dacă aceste reguli descoperite au o acoperire mai mică decât cea impusa atunci ele se adaugă unei liste de reguli, instanţele din mulţimea de antrenament acoperite de ele se elimină din aceasta şi tot algoritmul se reia pana se ajunge la acoperirea dorită.
- *permite determinarea şi adăugarea unei reguli implicite listei regulilor descoperite.* Regula care se determină nu are condiţii în antecedent, poate acoperi orice instanţă şi o clasifică ca aparţinând de clasa majoritară. Aceasta este o metodă rapidă de a creşte acoperirea listei regulilor descoperite.
- *permite adăugarea la lista de reguli descoperite doar a celei mai bune reguli, obţinută după rularea algoritmului pentru numărul de generaţii stabilit sau a tuturor regulilor descoperite.* După rularea algoritmului pentru numărul de generaţii stabilit se descoperă mai multe reguli. Utilizatorul poate opta, dacă le consideră valide, pe toate sau doar pe cea mai bună dintre ele
- *utilizează o funcţie fitness diferită de cele utilizate de alţi algoritmi*. Au fost testate 3 funcţii *fitness* şi a fost aleasă cea cu care s-au obţinut cele mai bune rezultate.

Pseudocod-ul algoritmului genetic propus este prezentat în continuare:

```
if (a fost selectat clasificatorul algoritm genetic) then
        - împarte aleatoriu setul de date ales în mulţime de antrenament şi mulţime de test
        - alege aleatoriu instanţele care formează populaţia
        - codifică cromozomii, instanţele din mulţimea de antrenament şi cele din mulţimea
          de test

        lista_tuturor_regulilor_descoperite={}
        contor_iteraţii=1;

        do{     //iteraţia n
                contor_generaţii=0;

                do{
```

- determină valorile *fitness* ale cromozomilor şi suma acestora
- selectează o populaţie intermediară de cromozomi după principiul ruletei
- încrucişează unii cromozomi din populaţia intermediară
- aplică operatorul de mutaţie populaţiei
- recalculează clasa noilor cromozomi (opţiune a utilizatorului)

contor_generaţii+=1;
}while (contor_ generaţii < nr_ generaţii);

- afişează populaţia finală
- elimină regulile duplicat (cromozomii)
- adaugă regulile descoperite sau doar cea mai bună dintre regulile descoperite (opţiune a utilizatorului) la lista_tuturor_regulilor_descoperite
- decodifică cromozomii şi afişează regulile codificate de aceştia
- determină şi afişează gradul de acoperire al regulilor descoperite pe mulţimea de antrenament
- calculează şi afişează acurateţea predicţiei pe mulţimea de antrenament
- elimină din mulţimea de antrenament instanţele acoperite de regulile descoperite
- generează o nouă populaţie de cromozomi din mulţimea de antrenament

contor_iteraţii+=1;
}while (gradul de acoperire al regulilor descoperite < gradul de acoperire impus de utilizator)

- determină şi adaugă o regula implicită listei de reguli (opţiune a utilizatorului)
- afişează toată lista de reguli descoperite
- testează regulile pe mulţimea de test şi se afişează acoperirea şi acurateţea predicţiei
- afişează timpul de execuţie al algoritmului

Împărţirea setului de date în mulţime de antrenament şi mulţime de test se face în mod aleatoriu. Procentul de instanţe din mulţimea de test este preluat din interfaţă din caseta de text m_PercentText.

Instanţele din populaţia iniţială sunt alese în mod aleatoriu din setul de date.

Codificarea cromozomilor

Cromozomii se obţin prin codificarea binară a instanţelor din populaţie. O genă este reprezentată de valoarea unui atribut. Fiecare cromozom codifică o regulă. Regulile din populaţia iniţială sunt exprimate sub forma:

IF atribut1=valoare1 şi atribut2=valore2 şi… THEN clasa=c1

Fiecare regulă (cromozom) se codifică sub forma unui şir de biţi [VIV10a]. Un atribut care poate lua *m* valori va fi codificat pe *m* biţi (câte un bit pentru fiecare valoare posibilă). În cazul în care avem 3 atribute A1 cu 4 valori posibile, A2 cu 2 valori posibile şi A3 cu 3 valori posibile şi un atribut clasa cu 3 valori posibile, codificarea unei reguli se poate face ca în tabelul 1 (biţii sunt numerotaţi de la

dreapta la stânga). În cazul în care setul de date conţine atribute numerice acestea se discretizează.

Atribut	A1	A2	A3	C
Cod	1000	01	001	100
Regulă	If A1=valoare1 and A2=valoare2 and A3=valoare3 then C=valoare1			

Tabel 4.1 - Codificarea cromozomilor

Instanţele din mulţimea de antrenament şi cele din mulţimea de test se codifică şi ele binar.

Determinare performanţă cromozomi

În continuare începe o secvenţă repetitivă de cod, care se execută de un număr de ori egal cu numărul de generaţii stabilit de utilizator. Se determină valorile *fitness* pentru toţi cromozomii din populaţie. Funcţia *fitness* folosită este cea din formula 4.9. Primul factor reprezintă acurateţea predicţiei (vezi formula 4.10) iar cel de-al doilea senzitivitatea regulii (vezi formula 4.11). S-a analizat comportamentul algoritmului la diferite funcţii *fitness* (vezi subcapitolul 4.5.3) şi funcţia *fitness* cu care s-au obţinut cele mai bune rezultate este cea din formula 4.9, care a fost inclusă în algoritm.

$$Fitness = PA \cdot Senz \tag{4.9}$$

$$PA = \frac{tp}{tp + fp} \tag{4.10}$$

$$Senz = \frac{tp}{tp + fn} \tag{4.11}$$

Selecţie cromozomi

Algoritmul creează o populaţie intermediară în care sunt puşi cromozomii selectaţi în mod proporţional cu performanţa lor, după principiul ruletei. Pseudocodul algoritmului este prezentat mai jos.

```
- determină suma valorilor fitness corespunzătoare tuturor cromozomilor
- r = un număr aleatoriu în intervalul [0,suma]

suma_parţială=0
For i=0 to nr_indivizi_populaţie
        suma_partiala+=fitness(i);
        if(suma_partiala>r)
                Break;
Return i
```

Încrucişare cromozomi

Se parcurge populaţia intermediară şi pentru fiecare cromozom se generează un număr aleatoriu, cu distribuţie uniformă între 0 şi 1. Dacă acest număr este mai mare decât probabilitatea de încrucişare, stabilită de utilizator, atunci cromozomul va fi copiat în noua populaţie, dacă nu, el se va încrucişa cu altul pentru a forma doi descendenţi în noua populaţie. Încrucişarea se realizează într-un punct, iar punctul de încrucişare este ales aleatoriu (între 0 şi numărul de biţi din antecedent). Tabelul 4.2 exemplifică încrucişarea cromozomilor în punctul de taiere 3, obţinut aleatoriu. Spaţiile dintre grupurile de biţi nu fac parte din reprezentarea cromozomilor dar au fost puse pentru a ajuta utilizatorul să delimiteze atributele.

Cromozom1	1000 10 001 100
Cromozom2	0001 01 010 001
Cromozom1'	1001 01 010 001
Cromozom2'	0000 10 001 100

Tabel 4.2 - Încrucişarea cromozomilor

Mutaţie cromozomi

În următoarea etapă se parcurg toţi cromozomii din noua populaţie şi pentru fiecare individ se generează aleatoriu un număr între 0 şi 1. Dacă acest număr este mai mic decât probabilitatea de mutaţie (parametru stabilit de utilizator), atunci asupra cromozomului respectiv se va aplica operatorul de mutaţie. Operatorul de mutaţie generează un număr aleatoriu cuprins între 0 şi numărul de biţi din antecedent. Bitul astfel obţinut va fi basculat (dacă are valoarea 1 devine 0 şi invers).

Recalculare clasă

În reprezentarea cromozomilor a fost inclusă şi clasa. Este posibil însă ca în urma operaţiilor de încrucişare şi mutaţie cromozomii obţinuţi să prezică mai bine o altă clasă decât cea a părinţilor. În acest sens, dacă utilizatorul doreşte, bifează o opţiune care determină pentru fiecare cromozom din noua populaţie care este clasa cea mai potrivită. S-a considerat că, clasa cea mai potrivită este cea care conduce la cele mai multe instanţe clasificate corect.

Algoritmul prezentat mai sus se repetă pentru un număr de generaţii stabilit de utilizator. Indivizii din populaţia finală sunt afişaţi.

Convergenţa cromozomilor

În urma execuţiei numărului de generaţii stabilit se produce fenomenul de convergentă către cel mai bun individ. Astfel populaţia finală obţinută după numărul de generaţii stabilit, va avea foarte multe duplicate de cromozomi (spre exemplu, în cazul setului de date car.arff populaţia finală are între 1 şi 3 cromozomi unici). Metoda elimina_cromozomi_duplicat va lăsa în populaţia finală doar cromozomii unici. Cromozomii din populaţia finală codifică regulile descoperite.

Reţine doar cea mai bună regulă din fiecare iteraţie

Dacă este bifată opţiunea *Reţine doar cea mai bună regulă din fiecare iteraţie*, se determină regula (cromozomul) care are cel mai mare *fitness* din cele descoperite. Doar această regulă va fi considerată validă. Ea se va adăuga listei de

reguli descoperite şi va fi decodificată şi afişată. Dacă nu este bifată opţiunea menţionată atunci toate regulile descoperite după execuţia numărului de generaţii stabilit sunt considerate valide şi vor fi adăugate listei regulilor descoperite, respectiv vor fi decodificate şi afişate.

Decodificarea regulilor

Regulile codificate de cromozomi sunt decodificate şi afişate în caseta de text m_OutTextGA. Operaţia de decodificare poate întâlni câteva probleme specifice:

- *Atribute care au toţi biţii cu valoarea 0.* În urma operaţiilor de încrucişare şi mutaţie repetate este posibil ca unele atribute să aibă toţi biţii puşi pe 0. Acele atribute nu vor apărea în regulă.
- *Atribute care au toţi biţii cu valoarea 1.* Astfel de atribute vor fi eliminate din regulă pentru că nu influenţează cu nimic rezultatul clasificării.
- *Atribute care au mai multe valori de 1 (dar nu toate).* Dacă un atribut are mai multe valori de 1 atunci el va apărea în regula de mai multe ori, operatorul SAU va lega valorile posibile ale atributului (de ex. codul 1001 al unui atribut *n* se decodifică Atribut n =valoare 1 SAU atribut n=valoare 4)

Acoperirea regulilor

Se calculează şi se afişează acoperirea pe care o au regulile obţinute asupra mulţimii de antrenament, precum şi acurateţea predicţiei. Acoperirea regulilor s-a calculat numărând câte instanţe din mulţimea de antrenament sunt acoperite de antecedentul regulilor descoperite. Acurateţea predicţiei s-a calculat împărţind numărul de instanţe care respectă şi antecedentul şi consecventul regulilor la numărul de instanţe care respectă antecedentul regulilor.

Algoritmul a fost testat pe toată durata implementării pe setul de date *car.arff*. S-a constat că regulile descoperite după rularea a 50 de generaţii pe o populaţie de 100 de cromozomi au o acurateţe foarte bună, dar acoperirea acestor reguli este slabă. În acest sens au fost concepute şi implementate două soluţii de creştere a acoperirii regulilor.

O primă soluţie de creştere a acoperirii regulilor este prezentată în continuare. Utilizatorul are posibilitatea să impună ca algoritmul să continue căutarea de reguli până când se găseşte un număr de reguli care asigură procentul de acoperire al instanţelor din mulţimea de antrenament dorit de utilizator. Dacă regulile descoperite acoperă un procent de instanţe mai mic decât cel dorit de utilizator, atunci ele se adaugă listei de reguli, instanţele din mulţimea de antrenament acoperite de aceste reguli sunt eliminate, se reiniţializează populaţia cu instanţe alese aleatoriu din mulţimea de antrenament şi algoritmul începe o nouă iteraţie. Dacă acoperirea regulilor descoperite este peste valoarea minimă impusă de utilizator, regulile descoperite sunt testate pe mulţimea de test şi rezultatele sunt afişate. Se determină timpul de execuţie al algoritmului şi se afişează.

O altă soluţie de a creşte acoperirea regulilor este de a determina şi adăuga la lista regulilor descoperite o regulă implicită. Această regulă implicită nu are nici o condiţie în antecedent, în felul acesta ea va cuprinde toate instanţele care nu au putut fi clasificate cu regulile descoperite. Clasa regulii implicite este întotdeauna clasa cea mai populară printre instanţele rămase în setul de antrenare. Utilizatorul poate opta prin intermediul unui control din interfaţă dacă doreşte sau nu determinarea unei reguli implicite şi adăugarea ei la lista de reguli descoperite prin rularea repetată a algoritmului genetic.

4.5. *Weka* extinsă cu algoritmul genetic *AGR*

Algoritmul genetic propus a fost denumit *AGR* şi a fost implementat în versiunea 3.6.4 a lui *Weka* (limbajul Java). Figura 4.1 prezintă interfaţa aplicaţiei extinse.

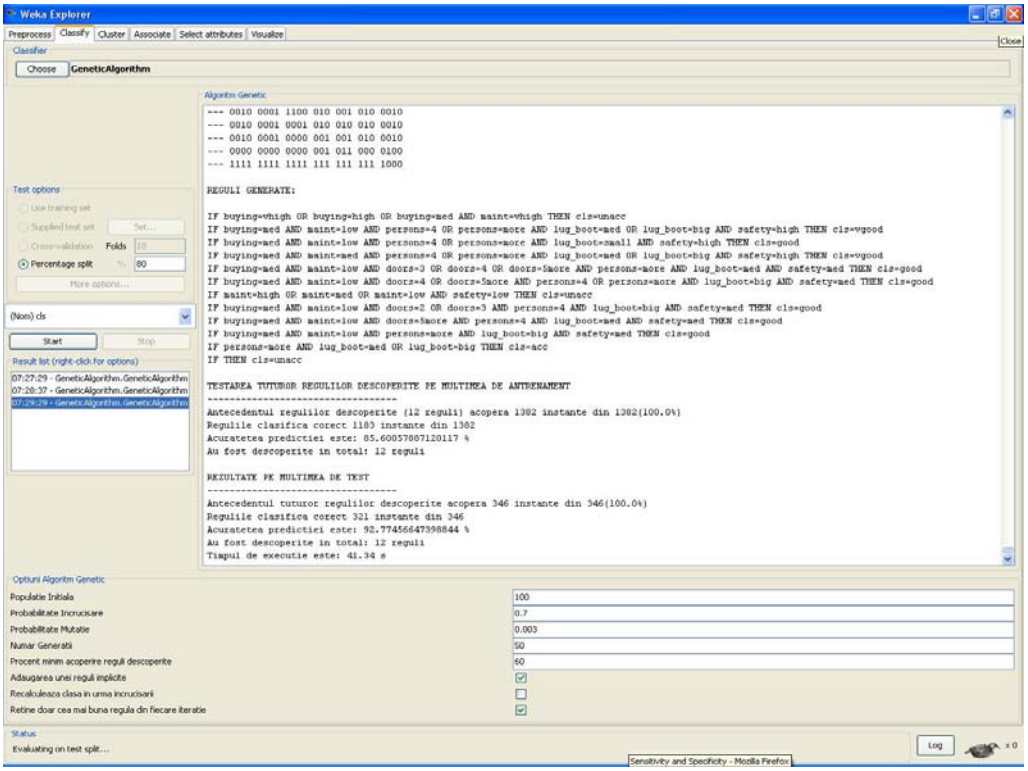

Fig. 4.1 - *Weka* cu algoritm genetic

Codul sursă al algoritmului a fost implementat în fişierul *ClassifierPanel.Java*. Pentru a adăuga clasificatorul *GeneticAlgorithm* în lista de clasificatori (vezi figura 4.2) a fost creat un nou pachet *weka.classifieres.GeneticAlgorithm* şi a fost completat numele pachetului în lista clasificatorilor din fişierul *GenericPropertiesCreator.props* din pachetul *weka.gui*.

În pachetul *weka.classifiers.GeneticAlgorithm* adăugat a fost creată clasa *GeneticAlgorithm*, derivată din *Classifier*, care implementează interfeţele *UpdateableClassifier, TechnicalInformationHandler*.

Codul sursă al algoritmului, precum şi codul aferent modificării interfeţei grafice a fost scris în fişierul *ClassifierPanel.Java*. În interfaţă au fost adăugate o serie de controale de tip *JPanel, JLabel, JTextField, JCheckBox,* pentru a permite utilizatorului să precizeze valorile unor parametri precum: dimensiunea populaţiei, probabilitatea de încrucişare, probabilitatea de mutaţie, procentul minim permis de

acoperire a regulilor descoperite, numărul de generaţii, dacă doreşte sau nu adăugarea unei reguli implicite la lista regulilor descoperite, dacă doreşte sau nu recalcularea clasei în urma încrucişării şi mutaţiei.

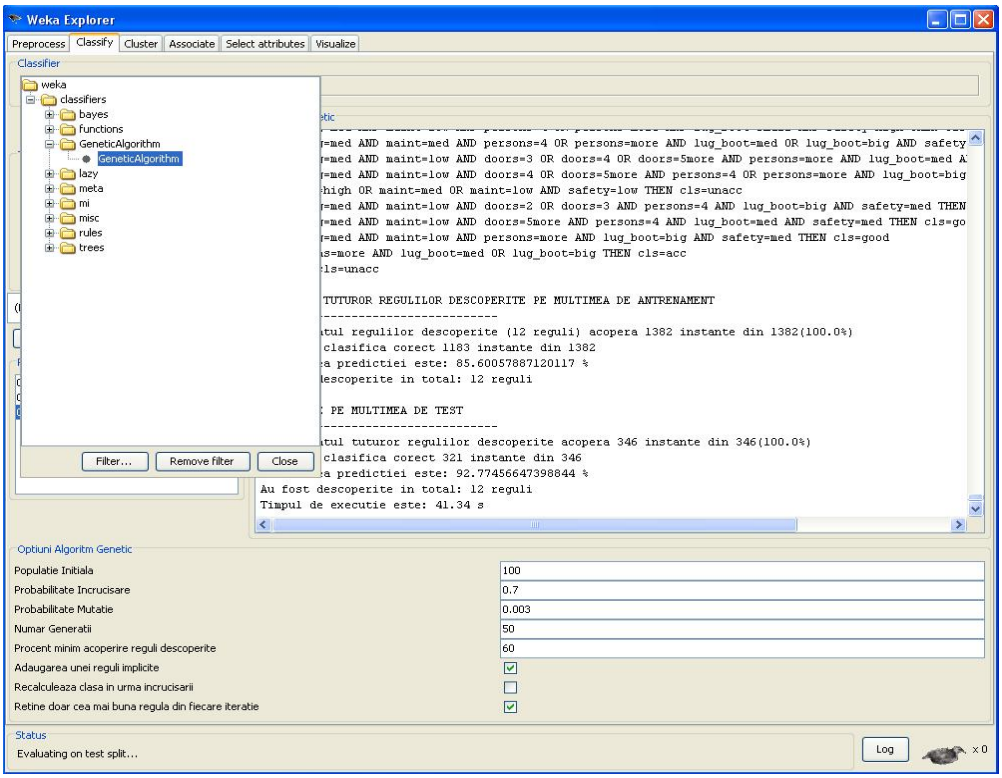

Fig. 4. 2 - Clasificatorul GeneticAlgorithm

Pentru a accesa instanţele din fişierul *ARFF* deschis s-a utilizat un obiect de tip *Instances*. Clasa *Instances* conţine metode ce permit determinarea numărului de atribute, a valorilor atributelor, a numărului de valori pe care îl are atributul clasă, a valorilor acestuia, a numărului de instanţe, permit accesul la orice instanţă, etc.

Petru toate procesările algoritmului au fost create metode în clasa *ClassifierPanel*, iar algoritmul a fost implementat în metoda *run*.

4.6. Analiza coeficienţilor algoritmului şi rezultate experimentale

Analiza comportamentului algoritmului s-a realizat în 3 situaţii: întâi s-au reţinut în lista de reguli toate regulile descoperite de fiecare iteraţie a algoritmului (prin iteraţie se înţelege execuţia algoritmului pentru numărul de generaţii stabilit). Al doilea experiment s-a realizat reţinând în lista de reguli doar cea mai bună regulă descoperită de fiecare iteraţie. Ultimul experiment s-a realizat utilizând 3 funcţii *fitness* şi comparând rezultatele obţinute cu ele. Comportamentul algoritmului a fost analizat utilizând în total 14 variante de parametri (8 variante de parametri pentru

primul experiment, 4 variante de parametri pentru cel de-al doilea experiment şi 2 variante de parametri pentru ultimul experiment).

Algoritmul a fost executat pe 5 seturi de date reale obţinute de pe UCI Machine Learning Repository. Aceste seturi de date sunt *Car, Zoo, Mushrooms, Breast Cancer, Heart Disease*. La fiecare rulare, 80 % din instanţe au fost folosite ca mulţime de antrenament şi 20 % ca mulţime de test. Fiecare variantă de parametri a fost rulată de 5 ori pe fiecare set de date şi tabelele de mai jos prezintă pentru toţi parametrii valorile medii obţinute la cele 5 rulări. Analiza făcută a condus la concluzii şi recomandări cu privire la alegerea parametrilor de intrare, care sunt de asemenea prezentate în cadrul acestui subcapitol.

4.6.1. Utilizarea tuturor regulilor descoperite de fiecare iteraţie

În cazul utilizării tuturor regulilor descoperite de fiecare iteraţie se consideră că toate regulile descoperite după ce algoritmul genetic a fost rulat pentru numărul de generaţii stabilit sunt suficient de bune încât să fie considerate reguli de ieşire ale algoritmului şi să fie adăugate listei regulilor descoperite. Configuraţiile de parametri de intrare alese sunt prezentate în tabelul 4.3.

Varianta de parametri	1	2	3	4	5	6	7	8
PI	100	100	100	100	100	100	100	100
Pc	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
Pm	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003
Număr generaţii	50	50	50	50	50	50	50	50
Procent minim acoperire reguli	0	0	0	60	60	60	60	80
Adăugare regulă implicită	nu	nu	da	nu	da	da	nu	da
Recalculare clasă	nu	da	nu	nu	nu	da	da	nu
Reţine doar cea mai bună regulă din fiecare iteraţie	nu	nu	nu	nu	nu	nu	nu	nu

Tabel 4.3 - Parametrii de intrare

Rezultatele din tabelul 4.4 arată că acurateţea predicţiei pe mulţimea de antrenament poate lua valori cuprinse între 52.74 % şi 100 % în funcţie de modul de alegere a parametrilor. Varianta de parametri care a produs cele mai slabe rezultate în ce priveşte acurateţea pe setul de antrenare este varianta 3, iar cea care a produs cele mai bune rezultate este varianta 1.

Varianta de parametri	v1	v2	v3	v4	v5	v6	v7	v8
Car	91.10	89.52	**73.22**	82.04	98.14	***99.61***	88.57	97.58
Zoo	***100.00***	***100.00***	**52.74**	***100.00***	87.00	84.50	***100.00***	***100.00***
Mushrooms	***100.00***	***100.00***	**57.04**	***100.00***	93.12	99.40	***100.00***	94.55
Breast cancer	88.00	83.15	**69.55**	83.55	83.85	79.21	83.77	***92.19***
Heart disease	***96.74***	93.76	**67.35**	86.89	81.32	78.01	82.34	90.58

Tabel 4.4 - Acurateţea pe mulţimea de antrenament (%)

Varianta de parametri care a produs cele mai bune rezultate pe mulţimea de test este varianta 2, iar cea care a produs cele mai slabe rezultate este la fel ca şi în cazul setului de antrenare varianta 3 (vezi tabelul 4.5).

Varianta de parametri	v1	v2	v3	v4	v5	v6	v7	v8
Car	94.44	94.62	**88.43**	90.89	97.80	99.88	93.83	***100.00***
Zoo	***100.00***	***100.00***	**63.80**	***100.00***	91.42	86.66	***100.00***	***100.00***
Mushrooms	***100.00***	***100.00***	**88.51**	***100.00***	92.65	99.74	***100.00***	93.47
Breast cancer	**60.00**	89.59	78.27	84.13	87.93	82.41	89.22	***95.86***
Heart disease	***100.00***	***100.00***	**70.82**	88.64	83.27	79.67	81.63	91.14

Tabel 4.5 - Acurateţea pe mulţimea de test (%)

Acoperirea regulilor descoperite arată capacitatea lor de a clasifica instanţe. Aşa cum era de aşteptat cea mai slabă acoperire o au regulile descoperite în variantele de parametri 1 şi 2, adică după rularea celor 50 de generaţii ale algoritmului genetic. Cea mai bună acoperire o au variantele de parametri ce utilizează pe lângă regulile descoperite rulând algoritmul genetic şi o regulă implicită. Nivelul de acoperire al regulilor descoperite poate să fie stabilit de utilizator la valoarea dorită. În cazul în care se impune execuţia repetată a algoritmului genetic până la descoperirea unei liste de reguli care acoperă minim 60 % din setul de antrenare şi nu este bifată opţiunea de determinare şi adăugare a unei reguli implicite la lista de reguli (versiunile 4 şi 7), acoperirea este cuprinsa între 60.28 şi 88.00 de procente (vezi tabelul 4.6).

Varianta de parametri	v1	v2	v3	v4	v5	v6	v7	v8
Car	**25.53**	26.05	***100.00***	77.09	***100.00***	***100.00***	88.00	***100.00***
Zoo	**7.25**	7.75	***100.00***	63.00	***100.00***	***100.00***	64.00	***100.00***
Mushrooms	3.45	**1.77**	***100.00***	60.28	***100.00***	***100.00***	61.05	***100.00***
Breast cancer	**3.86**	30.70	***100.00***	64.39	***100.00***	***100.00***	60.97	***100.00***
Heart disease	**9.83**	11.74	***100.00***	67.11	***100.00***	***100.00***	71.16	***100.00***

Tabel 4.6 - Acoperire pe mulţimea de antrenament (%)

Pe mulţimea de test, cea mai slabă acoperire o au variantele 1 şi 2 de parametri. Toate instanţele din mulţimea de test pot fi clasificate de variantele de parametri ce utilizează o regulă implicită. În variantele de parametri care nu utilizează o regulă implicită, dar în care algoritmul se execută repetat până la descoperirea unor reguli care acoperă minim 60 % din setul de antrenare, acoperirea pe mulţimea de test este cuprinsă între 23.76 % şi 94.22 %. Procentul de acoperire al regulilor pe mulţimea de test este prezentat în tabelul 4.7.

Varianta de parametri	v1	v2	v3	v4	v5	v6	v7	v8
Car	60.35	**59.54**	***100.00***	88.61	***100.00***	***100.00***	94.22	***100.00***
Zoo	**16.19**	17.15	***100.00***	70.48	***100.00***	***100.00***	71.43	***100.00***
Mushrooms	6.71	**6.53**	***100.00***	33.60	***100.00***	***100.00***	23.76	***100.00***

Breast cancer	**2.76**	24.14	***100.00***	57.59	***100.00***	***100.00***	57.59	***100.00***
Heart disease	**11.80**	**11.80**	***100.00***	59.67	***100.00***	***100.00***	68.85	***100.00***

Tabel 4.7 - Acoperire pe mulţimea de test (%)

Timpul de execuţie cel mai mic se obţine în varianta 1 de parametri. Un timp de execuţie mai ridicat au variantele de parametri care asigură acoperirea dorită de utilizator şi care execută repetat algoritmul genetic. Variantele de parametri în care se recalculează clasa cromozomilor în urma încrucişării au un timp de execuţie mai ridicat decât variantele similare care nu fac acest lucru. Tabelul 4.8 prezintă valorile medii obţinute pentru timpul de execuţie cu variantele de parametri alese.

Varianta de parametri	v1	v2	v3	v4	v5	v6	v7	v8
Car	***5.83***	16.61	5.90	37.60	34.58	**144.46**	112.52	39.07
Zoo	***0.75***	3.53	0.76	7.49	7.24	30.16	**33.30**	9.06
Mushrooms	***117.90***	151.09	73.18	2669.89	2981.55	9580.11	**9833.57**	4045.12
Breast cancer	***1.37***	3.23	1.42	21.89	17.38	21.11	18.92	**27.06**
Heart disease	***1.92***	6.88	1.96	7.59	7.59	**19.81**	18.30	6.48

Tabel 4.8 - Timpul de execuţie (s)

Tabelul 4.9 prezintă numărul de reguli descoperite. Variantele de parametri care descoperă cele mai puţine reguli sunt variantele 1 şi 2, iar varianta care descoperă cele mai multe reguli, este varianta 8. Recalcularea clasei în urma încrucişării conduce la reguli care au o acurateţe mai bună dar au o acoperire mai slabă. În cazul în care se impune ca acoperirea să aibă o anumită valoare minimă, variantele de parametri în care clasa se recalculează vor descoperi mai multe reguli pentru a asigura aceeaşi acoperire ca şi celelalte (comparaţie între numărul mediu de reguli obţinut cu variantele 6 şi 7 faţă de varianta 5).

Varianta de parametri	v1	v2	v3	v4	v5	v6	v7	v8
Car	***2.20***	2.40	4.20	18.20	15.40	25.00	15.40	**26.80**
Zoo	***2.20***	3.80	3.80	33.60	42.75	39.00	43.60	**65.40**
Mushrooms	3.80	***2.40***	4.00	180.33	186.67	311.20	346.00	**493.40**
Breast cancer	3.00	***1.60***	4.80	125.40	85.80	43.00	34.00	**184.60**
Heart disease	***1.80***	3.00	4.40	17.60	22.20	14.00	13.60	**25.80**

Tabel 4.9 - Numărul regulilor descoperite

Numărul de condiţii din reguli este direct proporţional cu numărul regulilor descoperite şi este prezentat în tabelul 4.10.

Varianta de parametri	v1	v2	v3	v4	v5	v6	v7	v8
Car	***7.80***	10.00	17.20	78.60	53.20	88.60	54.40	**142.40**
Zoo	***35.00***	57.80	41.00	511.80	626.50	557.20	640.40	**929.20**
Mushrooms	75.80	48.40	61.80	-	-	-	-	-
Breast cancer	27.20	***10.20***	33.20	892.40	710.00	334.60	392.60	**1634.00**
Heart disease	***13.20***	21.80	22.40	107.60	125.20	67.40	64.40	**150.20**

Tabel 4.10 - Numărul de condiţii din regulile descoperite

Variantele de parametri 1 şi 2 reprezintă ieşirea algoritmului genetic după rularea numărului de generaţii stabilit, fără nicio restricţie suplimentară. După cum se observă în tabelele de mai sus, aceste variante descoperă într-un timp de execuţie foarte bun, un număr mic de reguli, cu o acurateţe foarte bună. Acoperirea acestor reguli, atât pe setul de antrenare cât şi pe cel de testare, este foarte slabă, ceea ce înseamnă că foarte multe instanţe nu pot fi clasificate doar cu aceste reguli, deci utilizarea acestor variante de parametri nu reprezintă o soluţie.

Adăugarea unei reguli implicite la lista regulilor descoperite cu varianta 1 (varianta 3), rezolvă problema acoperirii instanţelor dar scade acurateţea predicţiei. Din punct de vedere al acurateţei predicţiei varianta 3 este cea mai slaba. Dacă se analizează global toţi parametrii se poate constata ca sunt situaţii în care varianta 3 este varianta optimă. De exemplu în cazul setului de date *Mushrooms*, variantele 4, 5, 6, 7 şi 8 produc rezultate ale acurateţei predicţiei între 92.65 şi 100 %, dar au un timp de execuţie foarte mare (minim 2669 s) şi descoperă minim 180 de reguli. Varianta 3 de parametri a descoperit în 73.18 secunde 4 reguli cu o acurateţe a predicţiei 88.51 % pe mulţimea de test. Deşi acurateţea predicţiei este puţin mai slabă decât în cazul celorlalte variante, ceilalţi parametri sunt mult mai buni şi de aceea varianta 3 este de preferat.

Variantele 4, 5, 6 şi 7 de parametri rulează în mod repetat algoritmul genetic până când se descoperă reguli ce acoperă minim 60 % din instanţele de antrenament. Această condiţie face ca regulile descoperite în variantele 4 şi 7 de parametri să acopere între 60.28 % şi 88 % din setul de antrenare, şi între 23.76 % şi 94.22 din setul de testare. Variantele de parametri 5 şi 6 acoperă toate instanţele din mulţimile de antrenament şi test, pentru că fiind bifată opţiunea de adăugare a unei reguli implicite se analizează datele care au rămas neclasificate de reguli şi se construieşte o regulă care acoperă toate instanţele şi le clasifică ca aparţinând de clasa cea mai frecventă.

Acoperirea impusă de 60 % este o acoperire medie. Dacă regulile descoperite cu aceasta restricţie acoperă prea puţin din mulţimea de test, atunci poate mări restricţia de 60 %, şi / sau se poate adăuga o regulă implicită. Cu cât se impune ca regulile descoperite să acopere mai mult din setul de antrenare, cu atât timpul de execuţie creste, la fel şi numărul regulilor descoperite (comparaţie între v5 şi v8).

În încercarea de a determina secvenţa optimă de parametri, dintre variantele care impun o acoperire de minim 60% (variantele 4, 5, 6 şi 7), trebuie trataţi împreună toţi indicatorii. În ceea ce priveşte acurateţea predicţiei, rezultatele sunt bune şi sunt comparabile în cazul variantelor menţionate. Din punct de vedere al acoperirii variantele de parametri care sunt preferate sunt cele care acoperă toate instanţele, deci 5 şi 6. Timpul de execuţie este mult mai bun în cazul variantei 5, decât în cazul variantei 6, la fel şi numărul regulilor descoperite este mult mai bun. Deci varianta optimă dintre cele menţionate este varianta 5.

Varianta 8 de parametri este similară variantei 5 cu menţiunea că impune algoritmului să descopere şi mai multe reguli decât versiunea 5, astfel încât regulile descoperite să acopere minim 80 % din instanţele de antrenare, spre deosebire de versiunea 5 care permite ca minim 60 % din instanţele de antrenare să fie acoperite de regulile descoperite. Ambele versiuni permit clasificarea tuturor instanţelor datorită regulii implicite care se adaugă listei regulilor descoperite. În ceea ce priveşte acurateţea predicţiei, ambele variante de parametri conduc la valori bune,

pe mulţimea de antrenament varianta 8 a obţinut valori mai mari decât varianta 5 pe 4 din 5 seturi de date, iar pe mulţimea de test varianta 8 a obţinut valori mai bune pentru acurateţea predicţiei pe toate seturile de date considerate. Timpul de execuţie este mai mare în cazul utilizării variantei 8 decât în cazul variantei 5. Numărul regulilor descoperite şi al condiţiilor este mult mai mare în cazul variantei 8 decât în cazul variantei 5.

Scopul oricărui algoritm de clasificare este să găsească cât mai repede reguli, care să clasifice cât mai corect datele, care să acopere toate instanţele, care să fie cât mai puţine, mai simple, mai uşor de înţeles şi utilizat. De multe ori nu se pot asigura valori maxime simultan, pentru toţi parametrii de ieşire şi trebuie alese configuraţii de compromis pentru parametrii de intrare, ţinând cont de datele care trebuiesc clasificate, astfel încât parametrii consideraţi mai importanţi dintre parametrii de ieşire să aibă valori bune iar ceilalţi să aibă valori satisfăcătoare. Dacă pe setul de date considerat, varianta de parametri 8 conduce la o acurateţe puţin mai bună decât varianta 5, dar la un timp de execuţie şi un număr de reguli mult mai mare, atunci varianta 5 este de preferat. Altfel este de preferat varianta 8.

4.6.2.Utilizarea celei mai bune reguli descoperite de fiecare iteraţie

În cazul în care se reţine în lista de reguli doar cea mai bună regulă descoperită în urma rulării algoritmului genetic pentru 50 de generaţii, s-au considerat din nou variantele de parametri 4, 5, 6, 7 şi 8 la care a fost modificată doar valoarea ultimului parametru (parametrul *Reţine doar cea mai bună regulă din fiecare iteraţie* a primit valoarea da). Noile variante de parametri au fost numite 4', 5', 6', 7' şi 8'. Tabelele de mai jos prezintă comparativ rezultatele obţinute utilizând toate regulile descoperite respectiv doar cea mai bună regulă din fiecare rulare. Valorile cele mai mari obţinute comparând versiuni similare de parametri (de ex. versiunea 4 cu versiunea 4' au fost subliniate).

Tabelul 4.11 prezintă acurateţea obţinută pe mulţimea de antrenament. Toate variantele de parametri asigură o acurateţe a predicţiei între 76.22 şi 100 %. Variantele de parametri ce reţin doar cea mai bună regulă conduc de 12 ori la rezultate mai bune decât cele care reţin toate regulile, iar variantele care reţin toate regulile, conduc de 6 ori la rezultate mai bune decât celelalte.

Varianta de parametri	v4	v4'	v5	v5'	v6	v6'	v7	v7'	v8	v8'
Car	82.04	***90.97***	***98.14***	91.28	***99.61***	98.00	***88.57***	85.44	***97.58***	95.67
Zoo	100.00	100.00	87.00	***89.25***	84.50	84.50	100.00	100.00	100.00	100.00
Mushrooms	100.00	100.00	93.12	***99.12***	99.40	94.40	100.00	100.00	94.55	***99.91***
Breast cancer	83.55	***88.70***	83.85	***94.38***	***79.21***	76.22	***83.77***	83.00	92.19	***95.96***
Heart disease	86.89	***87.06***	81.32	***81.90***	78.01	***82.97***	82.34	***84.77***	90.58	***92.47***

Tabel 4.11 - Acurateţea pe mulţimea de antrenament (%)

Rezultatele din tabelul 4.12 arată că variantele de parametri care folosesc toate regulile descoperite după 50 de generaţii au obţinut de 10 ori rezultate mai bune decât celelalte, iar variantele care reţin doar cea mai bună regulă, au obţinut

de 10 ori rezultate mai bune. Acurateţea pe mulţimea de test este cuprinsă între 79.67 şi 100 % pe cele 5 seturi de date considerate.

Varianta de parametri	v4	v4′	v5	v5′	v6	v6′	v7	v7′	v8	v8′
Car	90.89	***93.77***	***97.80***	96.36	***99.88***	99.77	***93.83***	89.09	***100.00***	98.15
Zoo	100.00	100.00	91.42	***94.28***	***86.66***	84.76	100.00	100.00	100.00	100.00
Mushrooms	100.00	100.00	92.65	***99.53***	***99.74***	93.19	100.00	100.00	93.47	***99.93***
Breast cancer	84.13	***93.58***	87.93	***95.51***	***82.41***	78.96	***89.22***	86.36	95.86	***97.58***
Heart disease	***88.64***	84.68	***83.27***	81.63	79.67	***83.93***	81.63	***83.63***	91.14	***91.47***

Tabel 4.12. Acurateţea pe mulţimea de test (%)

În ceea ce priveşte acoperirea instanţelor din mulţimea de antrenament, de cinci ori s-a obţinut o acoperire mai bună dacă s-au utilizat toate regulile şi de 5 ori dacă s-a utilizat doar cea mai bună regulă. Acoperirea obţinută pe datele de antrenare este prezentata în tabelul 4.13.

Varianta de parametri	v4	v4′	v5	v5′	v6	v6′	v7	v7′	v8	v8′
Car	77.09	***87.06***	100.00	100.00	100.00	100.00	***88.00***	77.39	100.00	100.00
Zoo	***63.00***	62.50	100.00	100.00	100.00	100.00	64.00	***66.00***	100.00	100.00
Mushrooms	60.28	***62.86***	100.00	100.00	100.00	100.00	61.05	***62.45***	100.00	100.00
Breast cancer	***64.39***	60.62	100.00	100.00	100.00	100.00	60.97	***63.86***	100.00	100.00
Heart disease	***67.11***	66.03	100.00	100.00	100.00	100.00	***71.16***	64.46	100.00	100.00

Tabel 4.13 - Acoperire pe mulţimea de antrenament (%)

Acoperirea pe mulţimea de test este mai bună dacă se reţine cea mai bună regulă la fiecare rulare (de şase ori s-a obţinut o acoperire mai bună dacă se reţine cea mai bună regulă şi de 4 ori s-a obţinut o acoperire mai bună dacă se reţin toate regulile).

Varianta de parametri	v4	v4′	v5	v5′	v6	v6′	v7	v7′	v8	v8′
Car	88.61	***91.73***	100.00	100.00	100.00	100.00	***94.22***	83.93	100.00	100.00
Zoo	***70.48***	67.62	100.00	100.00	100.00	100.00	71.43	***73.33***	100.00	100.00
Mushrooms	33.60	***41.24***	100.00	100.00	100.00	100.00	23.76	***37.54***	100.00	100.00
Breast cancer	***57.59***	54.48	100.00	100.00	100.00	100.00	57.59	***59.31***	100.00	100.00
Heart disease	59.67	***65.90***	100.00	100.00	100.00	100.00	***68.85***	62.95	100.00	100.00

Tabel 4.14 - Acoperire pe mulţimea de test (%)

Timpul de execuţie a fost în 10 situaţii mai mic atunci când s-a determinat şi utilizat cea mai bună regulă din cele descoperite de algoritmul genetic şi în 15 situaţii a fost mai mic când s-au utilizat toate regulile descoperite de algoritmul genetic. Valorile obţinute pentru timpul de execuţie sunt prezentate în tabelul 4.15.

Varianta de parametri	v4	v4′	v5	v5′	v6	v6′	v7	v7′	v8	v8′
Car	37.60	***35.85***	34.58	***31.95***	144.46	***120.15***	112.52	***81.43***	39.07	***38.65***

Zoo	***7.49***	8.19	***7.24***	7.62	***30.16***	31.87	***33.30***	33.97	***9.06***	33.97
Mushrooms	***2669.89***	3606.53	***2981.55***	4299.34	9580.11	***7983.96***	***9833.57***	11657.30	***4045.12***	5606.56
Breast cancer	***21.89***	28.41	***17.38***	40.52	21.11	***19.07***	18.92	***18.03***	***27.06***	38.72
Heart disease	7.59	***5.60***	7.59	***6.34***	***19.81***	24.32	***18.30***	19.60	***6.48***	9.68

Tabel 4.15 - Timpul de execuţie

Numărul regulilor descoperite, la fel ca şi numărul condiţiilor este mai mic în cazul în care la fiecare rulare se adaugă în lista de reguli doar cea mai bună regulă descoperită (vezi tabele 4.16 şi 4.17).

Varianta de parametri	v4	v4′	v5	v5′	v6	v6′	v7	v7′	v8	v8′
Car	18.20	***9.60***	15.40	***9.20***	25.00	***10.40***	15.40	***6.20***	26.80	***10.40***
Zoo	33.60	***15.00***	42.75	***14.80***	39.00	***14.20***	43.60	***14.00***	65.40	***14.00***
Mushrooms	180.33	***80.60***	186.67	***99.60***	311.20	***87.40***	346.00	***127.75***	493.40	***156.00***
Breast cancer	125.40	***34.40***	85.80	***47.00***	43.00	***10.60***	34.00	***9.80***	184.60	***50.40***
Heart disease	17.60	***3.80***	22.20	***5.40***	14.00	***5.80***	13.60	***3.80***	25.80	***11.80***

Tabel 4.16 - Numărul regulilor descoperite

Varianta de parametri	v4	v4′	v5	v5′	v6	v6′	v7	v7′	v8	v8′
Car	78.60	***38.40***	53.20	***32.60***	88.60	***38.00***	54.40	***27.20***	142.40	***50.60***
Zoo	511.80	***240.40***	626.50	***214.40***	557.20	***199.00***	640.40	***215.00***	929.20	***215.00***
Mushrooms	-	***1700.80***	-	***2095.40***	-	***1827.20***	-	***2740.75***	-	***3311.5***
Breast cancer	892.40	***290.40***	710.00	***397.40***	334.60	***73.40***	392.60	***76.00***	1634.00	***421.4***
Heart disease	107.60	***21.60***	125.20	***26.20***	67.40	***29.00***	64.40	***20.80***	150.20	***68.2***

Tabel 4.17. Numărul de condiţii din regulile descoperite

	Număr valori maxime dacă se reţin toate regulile	Număr valori maxime dacă se reţine doar cea mai bună regulă
Acurateţea pe mulţimea de antrenament	6	12
Acurateţea pe mulţimea de test	10	10
Acoperirea pe mulţimea de antrenament	5	5
Acoperirea pe mulţimea de test	4	6
Timpul de execuţie	15	10
Numărul regulilor descoperite	0	25
Numărul de condiţii din reguli	0	25

Tabel 4.18 - Numărul de valori maxime obţinute când parametrul BR=true, respectiv când parametrul BR=false

Rezultatele prezentate în tabelul 4.18 arată comparativ în câte situaţii s-au obţinut rezultate mai bune pe cele 5 seturi de date analizate dacă s-au considerat valide toate regulile descoperite după epuizarea celor 50 de generaţii, respectiv dacă s-a considerat validă doar cea mai bună regulă obţinută după rularea celor 50 de generaţii. Rezultatele sunt comparabile în ceea ce priveşte acurateţea, acoperirea şi

timpul de execuţie. În schimb numărul regulilor descoperite şi implicit şi numărul condiţiilor din reguli este mult mai mic în cazul în care se memorează doar cea mai bună regulă.

4.6.3.Analiza comportamentului algoritmului la utilizarea a diferite funcţii fitness

Analiza comportamentului algoritmului în cazul utilizării a diferite funcţii *fitness* s-a realizat utilizând parametrii care au dat cele mai bune rezultate bune în studiile precedente. Pe lângă funcţia Q_1 utilizată deja în studiile precedente au mai fost testate încă două funcţii *fitness*. Parametrii de intrare ai algoritmului au fost setaţi la valorile din tabelul 4.19. Rezultate obţinute pentru parametrii de ieşire sunt prezentate în continuare.

Varianta de parametri	v8'	v9	v10
PI	100	100	100
Pc	0.7	0.7	0.7
Pm	0.003	0.003	0.003
Număr generaţii	50	50	50
Procent minim acoperire reguli	80	80	80
Adaugare regulă default	da	da	da
Recalculare clasă	nu	nu	nu
Reţine doar cea mai bună regulă din fiecare rulare	da	da	da
Fitness	Q_1	Q_2	Q_3

Tabel 4.19 - Configuraţii de parametri

Q_1 este funcţia *fitness* din relaţia 4.9, Q_2 este funcţia *fitness* din relatia 4.12 iar Q_3 este funcţia *fitness* din relaţia 4.13.

$$Q_2 = PA \tag{4.12}$$

$$Q_3 = w_1 * PA + w_2 * CPH + w_3 * Senz \tag{4.13}$$

unde

- *PA este acurateţea predicţiei, PA= tp / (tp+fp)*
- *CPH este comprehensibilitatea, CPH =1- ANC / MNC*
- *ANC=numărul actual de condiţii din antecedent*
- *MNC = numărul maxim de condiţii din antecedent*
- *Senz este senzitivitatea, Senz=tp/(tp+fn)*
- *w1, w2 şi w3 sunt ponderi definite de utilizator, valorile alese pentru ele au fost w1=0.6, w2=0.2, w3=0.2, punând astfel accentul mai mult pe acurateţea predicţiei şi mai puţin pe comprehensibilitatea şi senzitivitatea regulilor* [ROB11b]

Tabelele de mai jos prezintă comparativ, acurateţea, timpul de execuţie, numărul de reguli descoperite, numărul de condiţii din reguli în cazul utilizării celor 3 funcţii *fitness* propuse. Cele mai bune rezultate s-au obţinut cu funcţia *fitness* Q_1,

acesta este motivul pentru care a fost inclusă în algoritm. În ceea ce priveşte acurateţea predicţiei, atât pe mulţimea de antrenament cât şi pe mulţimea de test, cu ajutorul funcţiei *fitness* Q_1 s-au obţinut valori maxime pe 4 seturi de date, cu ajutorul funcţiei Q_2 s-a obţinut o valoare maximă pe un set de date de test, iar cu ajutorul funcţiei Q_3 s-a obţinut o valoare maximă pe un set de date de antrenare (vezi tabelele 4.17 şi 4.18). Timpii de execuţie cei mai buni s-au obţinut cu funcţia *fitness* Q_1 (valori minime pe 3 seturi de date, vezi tabelul 4.19). În ceea ce priveşte numărul de reguli descoperite şi numărul de condiţii din reguli, cele mai bune rezultate s-au obţinut tot cu funcţia *fitness* Q_1 (valori minime pentru numărul de reguli şi numărul de condiţii pe toate seturile de date). Cu ajutorul funcţiei *fitness* Q_3 s-au obţinut cele mai bune rezultate pentru timpul de execuţie pe două seturi de date (vezi tabele 4.20 şi 4.21).

Parametrii	v8′	v9	v10
Car	***95.67***	88.98	90.78
Zoo	***100.00***	88.44	91.25
Mushrooms	***99.91***	90.23	
Breast cancer	***95.96***	91.01	91.07
Heart disease	92.47	89.75	***93.08***

Tabel 4.20- Acurateţea pe mulţimea de antrenament (%)

Parametrii	v8′	v9	v10
Car	***98.15***	93.49	94.60
Zoo	***100.00***	91.66	96.82
mushrooms	***99.93***	93.11	
Breast cancer	***97.58***	94.82	89.65
Heart disease	91.47	***93.11***	88.52

Tabel 4.21 - Acurateţea pe mulţimea de test (%)

Parametrii	v8′	v9	v10
Car	***38.65***	541.38	132.00
Zoo	33.97	11.52	***7.17***
Mushrooms	5606.56	6309.4	***5501.32***
Breast cancer	***38.72***	105.47	74.38
Heart disease	***9.68***	65.83	30.35

Tabel 4.22 - Timp execuţie (s)

Parametrii	v8′	v9	v10
Car	***10.40***	207.75	58.67
Zoo	***14.00***	29.50	18.67
Mushrooms	***156.00***	289.4	254.7
Breast Cancer	***50.40***	145.25	106.33
Heart disease	***11.80***	71.00	36.33

Tabel 4.23 - Numărul regulilor descoperite

Parametrii	v8′	v9	v10
Car	***50.60***	1288.00	249.33
Zoo	***215.00***	442.00	264.33
Mushrooms	***3311.5***		
Breast cancer Cancer	***421.4***	1287.75	812.33
Heart disease	***68.2***	495.20	206.33

Tabel 4.24 - Numărul de condiţii din reguli

4.6.4. Recomandări cu privire la alegerea parametrilor

Analiza realizată asupra parametrilor conduce la următoarele concluzii şi recomandări cu privire la alegerea acestora:

- Variantele de parametri care utilizează doar cea mai bună regulă descoperită la fiecare iteraţie au performanţe comparabile cu cele care utilizează toate regulile descoperite, în ceea ce priveşte acurateţea şi timpul de execuţie dar net superioare în ceea ce priveşte numărul regulilor descoperite şi numărului de condiţii din reguli, de aceea se recomandă utilizarea acestei facilităţi
- Se recomandă impunerea execuţiei repetate a algoritmului până se descoperă reguli care acoperă între 60 şi 80 % din instanţele de antrenare cumulat cu determinarea şi adăugarea unei reguli implicite la lista de reguli, care să clasifice toate instanţele ce nu pot fi clasificate cu regulile descoperite prin rularea repetată a algoritmului genetic. În acest fel regulile descoperite vor putea clasifica toate instanţele. Procentul de acoperire minim impus depinde de acurateţea obţinută, timpul de execuţie, numărul de reguli descoperite etc. Dacă setul de date este mare şi timpul de execuţie este foarte mare se poate impune o acoperire mică, cu condiţia ca acurateţea obţinută să fie satisfăcătoare sau se poate lucra doar cu un eşantion reprezentativ al setului de date
- Recalcularea clasei în urma încrucişării poate conduce la reguli cu o acurateţe mai bună dar o acoperire mai slabă. Deci pentru a asigura aceeaşi acoperire va fi nevoie de mai multe reguli şi de un timp de execuţie mai mare dacă este bifată această opţiune. Se recomandă utilizarea pentru seturi de date care nu sunt foarte mari
- Pentru parametrii clasici ai unui algoritm genetic, cum ar fi numărul de indivizi din populaţie, probabilităţile de încrucişare şi mutaţie, numărul de generaţii, se recomandă alegerea unor valori apropiate celor din studiile de mai sus, ţinând cont de următoarele aspecte:
    - Rata încrucişării stabileşte gradul de înnoire al populaţiei. Cu cât rata încrucişării este mai mare cu atât şansa de a avea mai mulţi indivizi noi în noua populaţie este mai mare. De obicei se alege între 0.6 şi 1.
    - Cu cât probabilitatea mutaţiei este mai mare cu atât riscul convergenţei premature este redus, dar dacă rata mutaţiei este prea mare algoritmul tinde să devină unul de căutare aleatoare. De obicei se aleg pentru mutaţie valori cuprinse între 0.001 şi 0.1.

4.7. Comparaţie cu alţi algoritmi

Întrucât *Weka* dispune de 117 algoritmi de clasificare şi regresie, pentru a compara rezultatele obţinute cu algoritmul genetic propus şi cele obţinute cu alţi algoritmi s-a utilizat tot *Weka* şi algoritmii *Naive Bayes, k-Nearest Neighbor* şi *J48*. Algoritmii au fost rulaţi în condiţii similare cu *AGR* propus, utilizând 80 % din instanţe la antrenament şi 20 % din instanţe la testare. Comparaţia s-a realizat între parametrii de ieşire pe care îi oferă atât *AGR* cât şi algoritmii aleşi pentru comparaţie. *AGR* a fost rulat cu configuraţia considerată optimă (v8').

După cum se observă rezultatele obţinute pentru acurateţe sunt mai bune în cazul algoritmului propus decât în cazul celorlalţi algoritmi pe patru seturi de date

din 5. În cazul setului de date *breast cancer* se obţin rezultate mult mai bune cu algoritmul propus.

	Acurateţea predicţiei			
Algorimii	Naive Bayes	k-Nearest Neighbor	J48	AGR
Car	87.57	92.77	91.04	***98.15***
Zoo	90	100	90	***100.00***
Mushrooms	95.38	***100***	***100***	99.03
Breast Cancer	70.17	71.92	71.92	***97.58***
Heart disease	86.88	77.04	85.24	***91.47***

Tabel 4.25 - Acurateţea pe mulţimea de test

Weka furnizează pentru algoritmii implementaţi în ea doar timpul de execuţie pentru construirea modelului nu şi pentru testarea lui aşa cum o face *AGR*. De aceea nu a putut fi realizată o comparaţie riguroasă la nivel de valori. În schimb se observă cu uşurinţă că timpul de execuţie al lui *AGR* este mai mare decât al celorlalţi algoritmi, acesta fiind dezavantajul algoritmului propus.

În ceea ce priveşte comparaţia între numărul de reguli şi condiţii descoperite de *AGR* şi cel determinat de ceilalţi algoritmi, aceasta se poate realiza doar între *J48* şi *AGR*, întrucât modelele construite de *Naive Bayes* şi *k-Nearest Neighbor* nu sunt compuse din reguli (vezi subcapitolele 1.3.2 şi 1.3.3).

Algoritmul *J48* determină un set de reguli ce sunt reprezentate sub forma unui arbore. Numărul de reguli determinate de algoritm nu este furnizat direct, dar este egal cu numărul de frunze. Numărul de condiţii nu este furnizat de *Weka*, el poate fi determinat prin numărare dar această operaţie poate consuma foarte mult timp dacă arborele este stufos.

Algoritmii	J48	AGR
Car	131	***10.40***
Zoo	17	***14.00***
Mushrooms	***25***	156.00
Breast Cancer	***4***	50.40
Heart disease	21	***11.80***

Tabel 4.26 - Număr reguli

Din punct de vedere al numărului de reguli descoperite există situaţii în care numărul de reguli descoperite cu *AGR* este mult mai mic decât cu J48 şi invers. Pe cele 5 seturi de date considerate, în 3 cazuri s-au obţinut rezultate mai bune (mai puţine reguli cu *AGR*) decât cu *J48*.

Comparaţiile cu alţi algoritmi arată că *AGR* este un algoritm performant, care poate determina reguli ce pot clasifica anumite seturi de date cu o acurateţe mai bună decât alţi algoritmi, poate determina un set de reguli mai redus decât alţi algoritmi, singura sa slăbiciune fiind timpul de execuţie.

4.8. Concluzii

Algoritmii genetici sunt tehnici adaptive care pot fi folosite cu succes în rezolvarea unor probleme de căutare şi optimizare complexe. O bază de date poate fi privită ca fiind un spaţiu de căutare foarte mare şi algoritmul genetic poate fi privit ca o strategie de căutare într-o bază de date. Ceea ce se caută sunt regulile de clasificare care vor constitui modelul clasificator. Capitolul de faţă prezintă structura generală a unui algoritm genetic, o scurtă sinteză cu privire la algoritmii genetici propuşi pentru clasificarea datelor şi propune un nou algoritm genetic în acest scop. Algoritmul genetic propus a fost implementat în *Java*, fiind adăugat instrumentului open source *Weka*. Mediul utilizat pentru dezvoltare a fost *Eclipse*. Testarea algoritmului şi analiza comportamentului său la diferite valori ale coeficienţilor de intrare s-a realizat utilizând 5 seturi de date obţinute din *UCI Machine Learning Repository*. În total au fost analizate 14 variante de parametri. Fiecare variantă de parametri a fost rulată de 5 ori pe fiecare set de date (la fiecare rulare au fost considerate 80 % din instanţele alese aleatoriu ca mulţime de antrenament şi 20 % din instanţe ca mulţime de test), iar valorile medii obţinute sunt prezentate în lucrare. Rezultatele obţinute au fost foarte bune şi în urma analizei s-au emis recomandări cu privire la alegerea parametrilor. Rezultatele obţinute cu *AGR* au fost comparate cu cele obţinute cu algoritmi clasici precum *Naive Bayes, k-Nearest Neighbor* şi *J48*, iar rezultatele arată că algoritmul propus este competitiv şi poate fi utilizat în studii de clasificare alături de aceşti algoritmi.

Contribuţiile aduse de acest capitol sunt următoarele:

- prezentarea succintă a structurii generale a unui algoritm genetic
- realizarea unei sinteze cu privire la algoritmii genetici utilizaţi la clasificarea datelor
- propunerea unui nou algoritm genetic pentru clasificarea datelor, care după cunoştinţele autorului, diferă de ceilalţi algoritmii genetici de clasificare prin următoarele:
  - permite recalcularea clasei în urma încrucişării
  - permite stabilirea unui procent minim de acoperire pentru regulile descoperite, rularea repetată a algoritmului până se atinge acest procent
  - permite adăugarea unei reguli implicite listei regulilor descoperite
  - permite adăugarea la lista de reguli descoperite doar a celei mai bune reguli obţinută după rularea algoritmului pentru numărul de generaţii stabilit sau a tuturor regulilor descoperite
  - utilizează o funcţie *fitness* diferită de cele utilizate de alţi algoritmi
- îmbunătăţirea tehnologiei de clasificare a datelor prin implementarea algoritmului genetic propus într-un instrument Open Source utilizat pe scară largă (*Weka*)
- analizarea comportamentului algoritmului la diferite valori ale parametrilor de intrare, şi la utilizarea a diferite funcţii *fitness* şi emiterea unui set de recomandări cu privire la alegerea acestora.

Direcţii de dezvoltare ulterioară sunt realizarea unei facilităţi care să-i permită utilizatorului atât să aleagă funcţia *fitness* pe care doreşte să o utilizeze în analiză, dintr-o listă de funcţii *fitness* propuse, dar şi să-şi poată construi funcţia *fitness* pe care doreşte să o utilizeze putând combina termenii tp, fp, tn, fn

împreună cu ponderi şi cu operatori de bază precum adunare, scădere, înmulţire împărţire, modulo, logaritm, etc. O altă direcţie de dezvoltare este implementarea mai multor metode de selecţie şi oferirea posibilităţii de a alege metoda de selecţie dorită într-un studiu.

5. Îmbunătăţirea procesului de realizare a predicţiilor

5.1. Noţiuni introductive

Weka permite construirea cu uşurinţă a unui model clasificator pe baza datelor de antrenament. Pentru a construi acest model clasificator se poate folosi unul din multitudinea de algoritmi de clasificare pe care *Weka* îi oferă. Testarea clasificatorului construit se poate face fie pe setul de antrenare, fie pe un set de test obţinut prin împărţirea setului iniţial de date în set de antrenare şi set de testare, fie pe un set de date bine precizat. Informaţii cu privire la rezultatele testării pot fi vizualizate în secţiunea *Classifier output*. Acestea sunt informaţii legate de rulare, modelul clasificator construit, rezultatele testării modelului, acurateţea detaliată pe clasă, matricea de confuzie, etc.

Dacă în urma analizării informaţiilor legate de testarea modelului se constată că acesta a obţinut o performanţă bună, el poate fi utilizat pentru a realiza predicţii.

Procesul de realizare a predicţiilor din *Weka* este foarte ne-sugestiv, se pot realiza predicţii doar în mod indirect re-testând modelul pe setul pe care dorim să-l prezicem. Întrebarea "How do I make predictions with a trained model?" de pe site-ul lui *Weka*, rubrica întrebări frecvente întăreşte afirmaţia făcută.

Capitolul prezintă modul în care se pot realiza predicţii cu *Weka*, soluţiile care au fost găsite pentru a îmbunătăţi acest proces şi îmbunătăţirile care au fost aduse interfeţei de clasificare a datelor. Pentru a îmbunătăţi procesul de realizare a predicţiilor au fost realizate modificări în secţiunea *Classify* a *Explorer*-ului şi acestea constau în transformarea acestei interfeţe într-o interfaţă dinamică în funcţie de setul de date analizat, interfaţă cu care se pot realiza predicţii într-un mod foarte simplu şi intuitiv. Alte modificări realizate au transformat această interfaţă într-o interfaţă prietenoasă, uşor accesibilă chiar şi începătorilor în *Weka*.

5.2. Realizare de predicţii cu *Weka*

Utilizarea unui model construit pentru a efectua predicţii în *Weka* se poate realiza doar indirect, folosind opţiunea de re-testare a modelului pe un set de date bine precizat. (nu există nici o opţiune clară de realizare de predicţii cu un model construit).

Realizarea predicţiilor necesită trecerea prin paşii următori:

- Se creează un nou fişier *ARFF* în care se pun instanţele care se doresc a fi prezise

- Întrucât este necesară completarea unei valori în dreptul clasei fiecărei instanţe din fişierul *ARFF*, se poate completa o clasă presupusă din cele existente sau se poate pune ? (vezi figura 5.1)

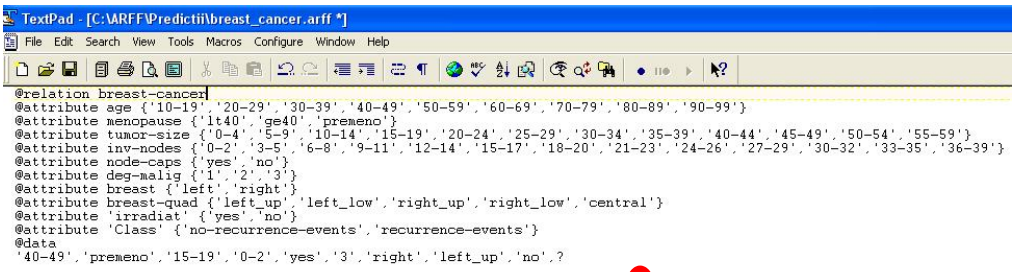

Fig. 5. 1 – Fişierul *ARFF* cu instanţe de prezis

- Se bifează opţiunea *Output prediction* (vezi figura 5.1)

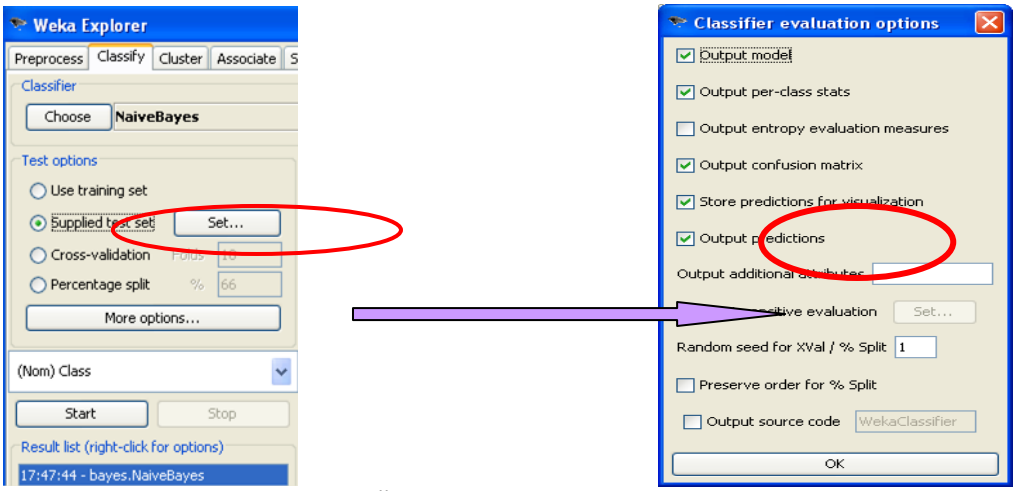

Fig.5.2 - Configurări necesare pentru realizarea de predicţii

- Se încarcă fişierul *ARFF* care conţine instanţele pentru care vrem să facem predicţii cu ajutorul opţiunii *Supplied test set*, din panel-ul *Test options* (vezi figura 5.2)
- Se face click dreapta pe modelul rulat şi se alege opţiunea de reevaluare a modelului pe setul curent de testare (vezi figura 5.3)

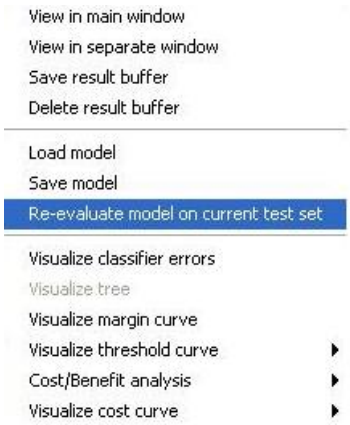

Fig. 5.3 - Reevaluarea modelului pe setul curent de testare

- modelul este de fapt retestat pe setul pe care noi vrem să-l prezicem, dar deoarece a fost bifată opţiunea *Output Prediction* în zona *Classifier output* se vor putea vizualiza valorile prezise (vezi figura 5.4)

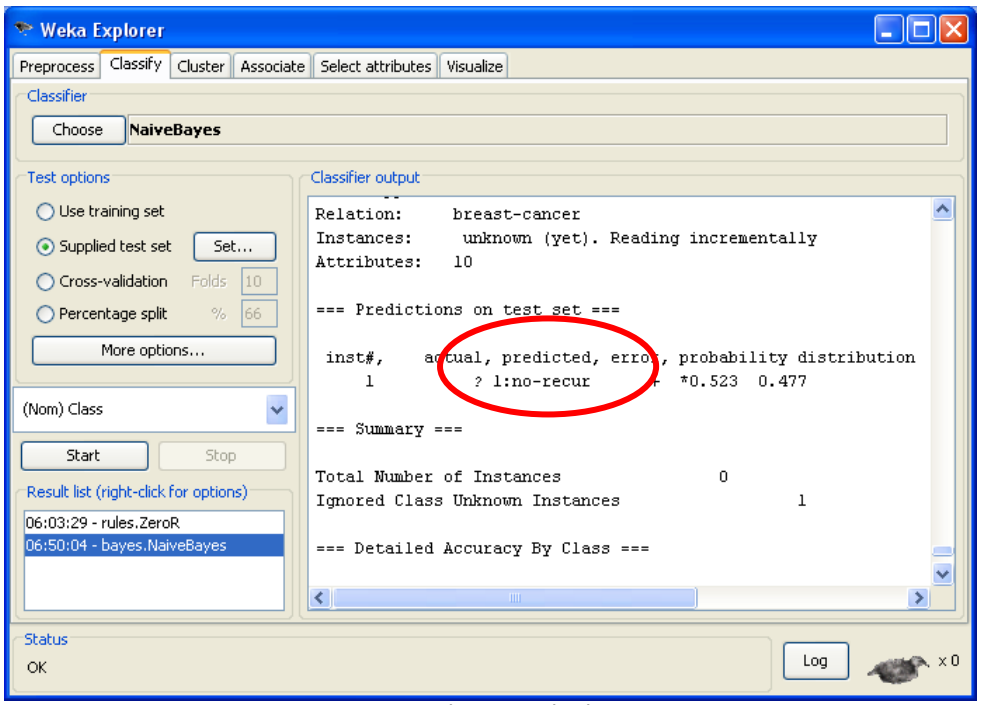

Fig. 5.4 - Vizualizarea valorilor prezise

Procesul prezentat este nesugestiv, predicţiile se realizează într-un mod indirect, printr-un artificiu, de fapt se retestează modelul pe setul pe care dorim să-l prezicem şi de aceea întrebarea "Cum se pot face predicţii cu un clasificator?" se găseşte în rândul întrebărilor frecvent adresate de utilizatorii de *Weka*. Procesul de

realizare a predicţiilor prezentat mai sus a fost mult simplificat şi transformat în unul rapid şi intuitiv, datorită faptului ca *Weka* este un instrument Open Source. Îmbunătăţirile aduse sunt prezentate în continuare.

5.3. Îmbunătăţirea procesului de realizare a predicţiilor

Pentru a îmbunătăţi acest proces, interfaţa de clasificare a datelor a fost modificată adăugând un nou panel, numit *Prediction of an instance.* Acest panel are un conţinut dinamic în funcţie de setul de date analizat. Panelul conţine pentru fiecare set de date un număr de label-uri egal cu numărul de atribute din setul de date, iar pentru fiecare atribut numeric se va afişa o casetă de text care permite introducerea valorii numerice, pentru fiecare atribut nominal în interfaţă va exista un control *JComboBox* cu valorile nominale posibile, din care utilizatorul va putea selecta valoarea nominală dorită. Utilizatorul introduce în interfaţa dinamică valorile instanţei pe care doreşte să o prezică, apasă comanda *Predict* şi clasa prezisă pentru acea instanţă este afişată imediat lângă butonul *Predict* (vezi figura 5.5) [ROB10a].

O altă modificare care a fost adusă secţiunii de clasificare a datelor a urmărit transformarea acesteia într-o secţiune prietenoasă. Pentru aceasta, panelul *Classifier output* care afişează multe informaţii care nu sunt accesibile decât utilizatorilor avansaţi de *Weka*, a fost înlocuit cu panelul *Test Results*, care afişează sub forma unui grafic 3d acurateţea clasificatorului construit. Accesul la informaţiile afişate în secţiunea *Classifier Output* nu a fost eliminat, dar utilizatorul care doreşte să vizualizeze aceste informaţii trebuie să apese un nou buton numit *Advanced information* care a fost introdus în interfaţă (vezi figurile 5.5 şi 5.6).

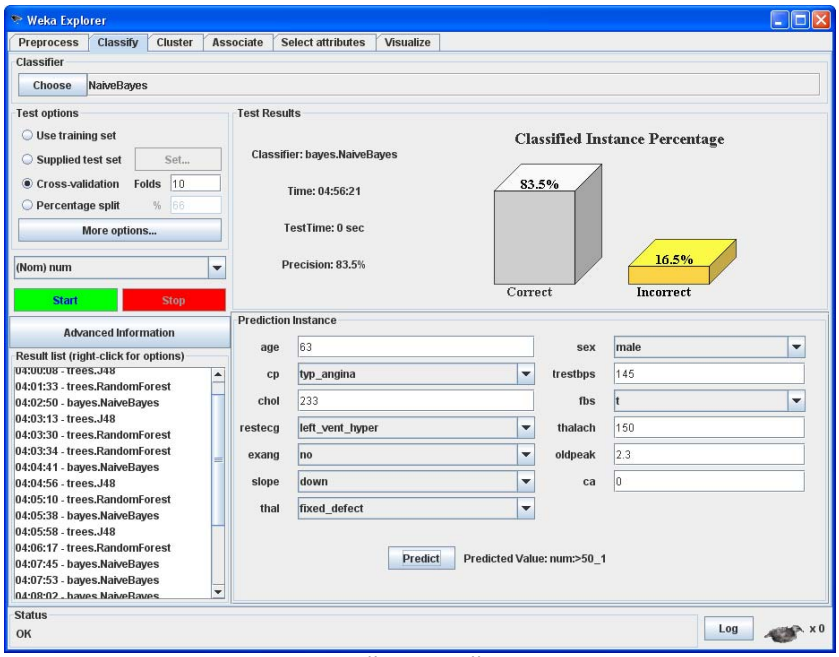

Fig. 5.5 - *Weka* cu interfaţă dinamică de realizare a predicţiilor

În continuare sunt prezentate o parte din informaţiile afişate în secţiunea *Classifier Output* cu scopul de a argumenta modificarea făcută.

Panel-ul *Classifier Output* din cadrul *GUI*-ului *Classify* afişează în mod implicit informaţii legate de rulare (numărul de instanţe, numărul de atribute, modul de testare al modelului), modelul clasificator, rezultatul testării (procentul de instanţe clasificate corect, procentul de instanţe clasificate greşit, *Kappa statistic, Mean absolute error, Root mean squered error, Relative absolute error, Root relative squered error*), acurateţea detaliată pe clasă (*True Positive Rate, False Positive Rate, Precision, Recall, F-Measure, Class*) şi matricea de confuzie [TAN05].

Ieşirile selectate în mod implicit pentru vizualizare sunt modelul, statisticile pe clasă, matricea de confuzie şi stocarea predicţiilor pentru vizualizare (vezi figura 5.2). Utilizatorul are posibilitatea să selecteze output-urile care le doreşte pentru vizualizare cu ajutorul comenzii *More Option*. Pe lângă ieşirile menţionate mai pot fi afişate măsurile de evaluare ale entropiei, predicţiile, atributele adiţionale, evaluarea cost-senzitivă, codul sursă, etc.

În cazul în care nu se selectează nici un output (sunt de-bifate toate opţiunile din interfaţa *Classifier evaluation options* (vezi fig. 5.2)) setul minimal de informaţii afişate în secţiunea *Classifier Output* cuprinde informaţii legate de rulare şi rezultatul testării exprimat atât prin procentul de instanţe clasificate corect respectiv greşit cât şi prin măsuri precum *Kappa statistic, Mean absolute error, Root mean squered error, Relative absolute error, Root relative squered error* (vezi figura 5.6).

Semnificaţia măsurilor menţionate este următoarea:

- *Kappa statistic*- măsoară acordul predicţiei cu clasa adevărată, valoarea 1.0 înseamnă acord complet [BOU08]
- *Mean absolute error* este o măsură utilizată pentru a analiza cât de aproape au fost predicţiile de valorile reale. Se calculează împărţind suma valorilor absolute a erorilor la numărul de predicţii
- *Root mean squered error* - este o modalitate de a cuantifica diferenţa dintre valorile reale şi de cele estimate. Eroarea este cantitatea prin care valoarea estimată diferă de cea reală [LEH98]. Mărimea se calculează extrăgând radical din suma pătratelor erorilor raportate la numărul de predicţii [WIT05].
- *Relative absolut error* – suma valorilor absolute a erorilor, raportată la suma valorilor absolute a diferenţelor faţă de medie) [WIT05]
- *Root relative squared error*. – Se calculează împărţind suma pătratelor erorilor la suma pătratelor diferenţelor faţă de medie. [WIT05]

Majoritatea măsurilor afişate în mod implicit, dar şi a celor afişate în cazul în care nu se selectează nici un output, sunt de folos doar utilizatorilor experimentaţi care le cunosc însemnătatea şi ştiu cum să le interpreteze. În rândul întrebărilor frecvente de pe site-ul *Weka* se află foarte multe întrebări legate de aceste măsuri, modul lor de calcul şi însemnătatea lor. În consecinţă autorul consideră că este mai bine ca accesul la aceste informaţii să se realizeze ca urmare a unei solicitări şi în mod implicit să fie prezentate informaţiile care pot fi interpretate chiar şi de utilizatori neexperimentaţi. În locul afişării măsurilor de mai sus a fost afişat un grafic 3d care redă acurateţea predicţiei, care poate fi înţeles şi interpretat chiar şi de începători. Cei interesaţi să vizualizeze măsurile menţionate, trebuie să apese butonul nou introdus *Advanced Information* [ROB10c].

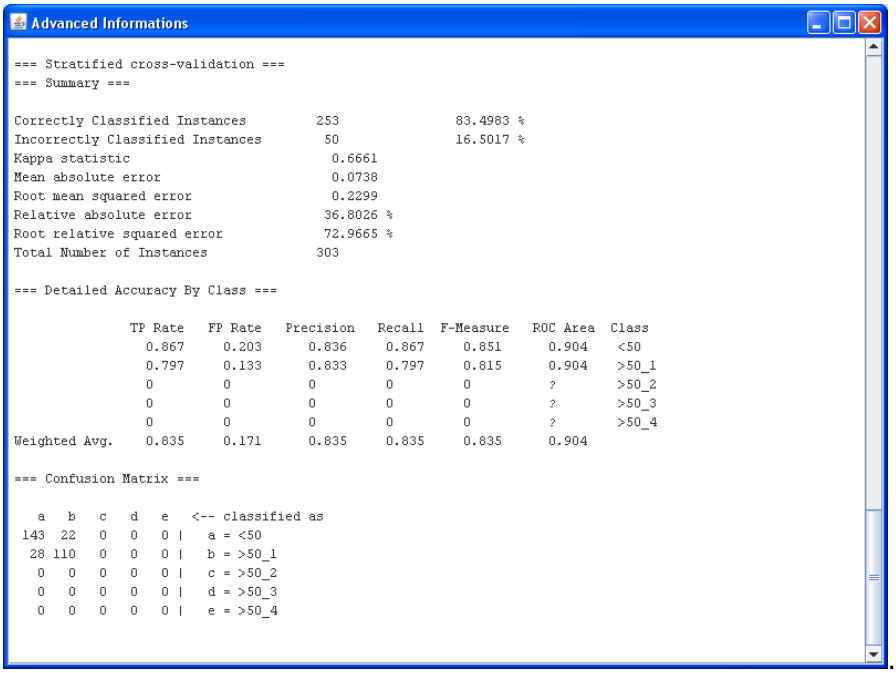

Fig.5.6 - Informaţiile din secţiunea Classifier Output

Modificările realizate simplifică mult acţiunile pe care un utilizator de *Weka* trebuie să le facă pentru a realiza predicţii cu un model construit şi testat. Totodată procesul a devenit unul clar, simplu şi intuitiv. Interfaţa a fost transformată în una simplă, prietenoasă care încurajează utilizarea lui *Weka*.

Aspecte legate de implementare

Weka realizează diverse prelucrări asupra unor seturi de date. În *Weka* setul de date este implementat de clasa *weka.core.Instances* [HAL09]. Fiecare instanţă este constituită dintr-un număr de atribute care pot fi nominale, numerice sau şiruri de caractere. Reprezentarea externă a unei clase Instances este un fişier ARFF. Algoritmii de clasificare din *Weka* sunt derivaţi din clasa abstractă *weka.classifiers.Classifier* care conţine la rândul ei metode pentru a genera un model de clasificare, pentru a evalua un model pe un set de date sau pentru a genera o distribuţie de probabilitate [WIT05]. Preprocesarea datelor este un pas important pentru algoritmii de învăţare automată. Un suport folositor pentru etapa de preprocesare este disponibil în pachetul *weka.filters* care este constituit din clase cu ajutorul cărora se pot face modificări asupra setului de date.

Arhitectura aplicaţiei *Weka* este cea din fig 1. *Weka* este alcătuită dintr-o multitudine de pachete. Pachetele sunt organizate ierarhic începând de la pachetul principal numit *weka*. Orice pachet poate să conţină la rândul lui alte pachete, fişiere cu cod sursă Java sau pe acestea amândouă în acelaşi timp. Fişierele în care au fost realizate modificări sunt: GuiChoser.java, ClasifierPanel.java, Explorer.java, PreprocessPanel.java. Fişierele SimpleBarChart.java ş SpringUtilities.java au fost adăugate.

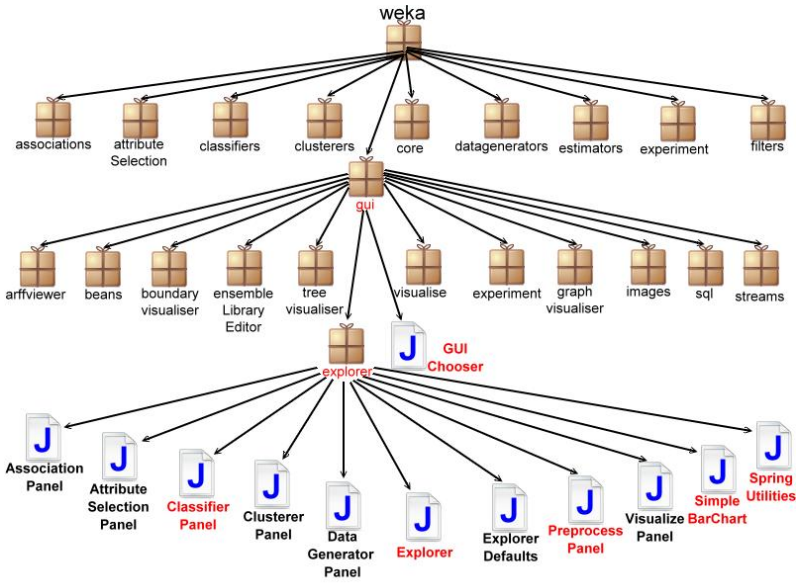

Fig. 5.7 – Arhitectura lui *Weka*

Testarea

Testarea aplicaţiei *Weka* extinsă s-a făcut construind clasificatori si realizând predicţii pe 3 seturi de date medicale obţinute de pe *UCI Machine learning Repository*. Acestea sunt *Ljubljana Breast Cancer, Heart Disease, Diabetes*. Informaţii cu privire la seturile de date menţionate se găsesc în introducere. Algoritmii cu care au fost construite modelele clasificatoare au fost *Naive Bayes [WU07], Random Forest* şi *J48*. Întrucât modelele care s-au obţinut cu aceşti algoritmi au obţinut rezultate bune, ele au fost utilizate pentru a realiza predicţii. Predicţiile au fost realizate cu ajutorul noii interfeţe dinamice adăugate în *Weka*.

Tabelul 5.1 arată rezultatele care s-au obţinut pentru acurateţea predicţiei folosind tehnica de testare cross validation. Figura 5.5 ilustrează rezultatele obţinute cu algoritmul *NaiveBayes* pe setul *Cleveland Heart Disease*. Figura 5.8 ilustrează rezultatele obţinute pe setul de date *Ljubljana breast cancer* de algoritmul J48 şi rezultatele obţinute de algoritmul *Random Forest* pe setul de date *Diabetes*. Aplicaţia îmbunătăţită a funcţionat corect.

Test data	Naïve Bayes	J48	Random Forest
Ljubljana breast cancer	71.68	75.52	69.23
Diabetes	76.3	73.83	73.83
Cleveland heart disease	83.5	77.56	81.52

Tabel 5.1 - Acurateţea predicţiei

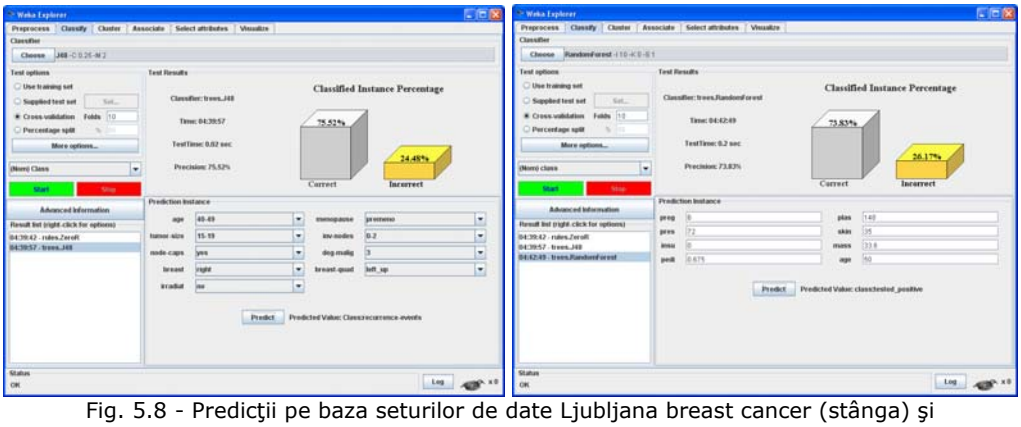

Fig. 5.8 - Predicţii pe baza seturilor de date Ljubljana breast cancer (stânga) şi diabetes (dreapta)

5.4. Concluzii

În cadrul acestui capitol a fost tratată problema realizării de predicţii cu ajutorul unui model construit şi testat de *Weka*. A fost prezentat modul în care clasificatorii pot fi utilizaţi pentru a realiza predicţii. A fost concepută şi implementată o interfaţă dinamică care conţine controale în acord cu setul de date analizat şi care uşurează şi clarifică mult procesul de realizare a predicţiilor. Interfaţa de clasificare a fost transformată într-o interfaţă mult mai prietenoasă, fiind adăugat în ea un grafic 3d care arată acurateţea modelului şi afişând indicatorii "complicaţi" din *Weka* într-o nouă fereastră *Advanced information*. Testarea modificărilor făcute s-a realizat construind modele clasificatoare pe trei seturi de date medicale, obţinute de pe *UCI Machine Learning Repository* şi realizând predicţii cu ele şi cu ajutorul interfeţei dinamice adăugate. Rezultatele testării au fost foarte bune.

Contribuţiile aduse în cadrul acestui capitol sunt:

- Prezentarea detaliată a modului în care *Weka* poate fi utilizat pentru a realiza predicţii
- Concepţia unei interfeţe dinamice în funcţie de setul de date analizat care permite realizarea cu uşurinţă a predicţiilor
- Prezentarea sintetică a mărimilor de ieşire furnizate de *Weka* în urma testării modelului
- Concepţia unor soluţii de transformare a interfeţei de clasificare a datelor într-o interfaţă mult mai simplă, inteligibilă şi prietenoasă, prin vizualizarea grafică a celei mai importante măsuri (acurateţea predicţiei) obţinute în urma testării modelului şi prin afişarea informaţiilor din secţiunea *Classifier output* într-o nouă fereastră *Advanced Information*.
- Modificările menţionate mai sus, realizate în unul din cele mai populare instrumente de data mining din lume, contribuie la îmbunătăţirea tehnologiei de clasificare a datelor.

6. Analiza şi clasificarea datelor legate de naşteri

6.1. Noţiuni introductive

Capitolul prezintă un studiu realizat pe datele legate de naşterile care au avut loc în anul 2010, la *Clinica de Obstetrică - Ginecologie Bega, Timişoara*. Setul de date disponibil pentru analiză cuprinde date despre 2326 de naşteri. Pentru fiecare naştere s-au reţinut în total 19 atribute care conţineau date despre mamă, date despre nou-născut şi date legate de intervenţii medicale. Capitolul prezintă setul iniţial de date, felul în care acestea au fost preprocesate pentru a putea fi şi ulterior analizate cu *Weka*, analiza statistică realizată pe fiecare atribut, modelele clasificatoare construite şi utilizarea acestora pentru a realiza predicţii.

Pentru realizarea analizei prezentate în cadrul acestui capitol au fost folosite instrumentele dezvoltate şi prezentate în capitolele 2, 3, 4 şi 5 şi noţiunile teoretice prezentate în toate capitolele anterioare.

Scopul analizei acestor date a fost de a extrage noi cunoştinţe şi informaţii folositoare din ele. Analiza s-a realizat în două etape: într-o primă etapă s-a realizat o analiză statistică la nivel de atribut, iar în a doua etapă s-a urmărit construirea unor modele clasificatoare din datele disponibile.

Analiza statistică realizată la nivel de atribut a permis determinarea unor informaţii statistice valoroase cum ar fi procentul din naşterile naturale în care este necesară realizarea unei tăieturi în zona pelvisului, care este procentul de femei care nasc natural, respectiv prin cezariană în zilele noastre, care este numărul mediu de ore de travaliu, etc.

Modelele clasificatoare permit realizarea de predicţii şi pot fi utilizate ca instrumente de suport ale actului medical. Modelul de clasificare construit permite estimarea cu o acurateţe de aprox 85 %, a indicelui apgar al nou născutului, în funcţie de parametrii cunoscuţi despre mamă, despre făt şi în funcţie de intervenţiile medicale care se realizează asupra mamei şi / sau fătului pentru a ajuta naşterea. Indicele apgar reprezintă o apreciere a funcţiilor vitale şi a capacităţii de adaptare a fătului la condiţiile din mediul extrauterin şi este de dorit ca valoarea acestui indice să fie maximă pentru fiecare nou născut.

Valoarea medicală a modelului de clasificare construit este următoarea: medicii pot analiza înainte de naştere care este intervalul în care se estimează că indicele apgar al viitorului nou născut va lua valori completând la intrarea modelului datele mamei, ale fătului, şi date legate de intervenţiile care intenţionează se le realizeze asupra mamei (de genul cezariană, sau naştere naturală cu epiziotomie, etc) pentru a ajuta naşterea. Modelul va furniza la ieşire intervalul în care se estimează că indicele apgar al nou născutului se va situa (1 din 5 intervale). Dacă se constată că acest interval este prea mic se poate verifica dacă se fac alt fel de

intervenţii (cum ar fi: se renunţă la naşterea naturală în favoarea cezarienei), sau dacă se ajută la îmbunătăţirea parametrilor mamei sau a fătului (de pildă se face un masaj care să ajute la ajungerea fătului în poziţia normală pentru naşterea naturală) ce valoare va lua indicele apgar după aceste intervenţii. Modelul construit nu este unul decizional, este un model consultativ pentru medici, care să-i ajute în luarea deciziilor în aşa fel încât fătul să aibă o trecere cât mai lină şi mai bună în mediul extrauterin, contorizată printr-un indice apgar cât mai mare.

6.2. Setul de date

Datele pe care a fost realizată analiza provin de la *Clinica de Obstetrică - Ginecologie Bega, Timişoara*. Setul de date iniţial cuprinde date despre 2326 de naşteri, realizate la acest spital în anul 2010. Datele despre naşteri au fost stocate într-un sheet al unui fişier Microsoft Excel.

Pentru fiecare naştere au fost disponibile următoarele informaţii:

- Un număr de ordine
- Luna naşterii
- Numele mamei
- Vârsta mamei
- Localitatea domiciliu
- Mediul - urban sau rural
- Gesta – a câta sarcină este
- Para – al câtelea copil este
- Numărul de săptămâni gestaţie
- Prezentaţie - Prezentaţia este aşezarea fătului la ieşirea din uter, mai precis partea din corpul copilului care va ieşi prima. Valorile posibile pentru acest câmp sunt: cefalică, pelviană, facială şi transversală. Prezentaţia **cefalică** este cea normală, fătul are coloana vertebrală paralelă cu cea a mamei şi capul în jos cu bărbia în piept. Prezentaţia **pelviană** înseamnă că fătul iese cu picioarele sau fundul înainte. Prezentaţia **facială** este când fătul priveşte înainte, faţa lui va ieşi prima. În cazul în care coloana fătului nu este paralelă cu coloana mamei (fătul este poziţionat oblic în burtă) prezentaţia este **transversală**.
- Indicele apgar - Imediat după naştere, chiar în primele 60 de secunde după expulzie, în sala de travaliu, se face o apreciere a stării de sănătate a nou-născutului, prin aprecierea funcţiilor vitale şi a capacităţii de adaptare la condiţiile din mediul extrauterin. Astfel, simultan cu acordarea primelor îngrijiri, medicul neonatolog va nota starea clinica şi comportamentul nou-născutului, cuantificând funcţiile vitale cu ajutorul scorului sau indicelui Apgar. Indicele apgar are valori cuprinse de la 0 la 10.
- Sexul copilului
- Greutatea
- Dacă a fost născut prin naştere naturală sau cezariană
- Videx – coloană care indică dacă fătului i-a fost aplicată pe cap sau nu o căciuliţă de metal care să ajute naşterea lui naturală
- Motivul pentru care a fost indicată naşterea prin cezariană
- Epiziotomia – coloană care indică dacă s-a făcut sau nu tăietura în perineu pentru a ajuta naşterea naturală (0 - nu s-a făcut, 1 – s-a făcut)

- Numărul de ore de travaliu
- EMP = extracţie manuală de placentă, valoarea 0 indică că nu s-a făcut o astfel de intervenţie, iar valoarea 1 că s-a făcut

6.3. Preprocesarea datelor

Preprocesarea datelor a fost realizată printr-o serie de acţiuni:

- Au fost eliminate coloanele neimportante pentru studiu, precum numărul de ordine, numele mamei, localitatea de domiciliu, recomandarea pentru cezariană care era şir de caractere. Coloana luna naşterii a fost păstrată pentru analiza statistică, dar a fost eliminată din rândul coloanelor folosite la construirea modelului
- Coloana număr săptămâni de gestaţie conţinea de regulă numărul exact al săptămânii de gestaţie (de exemplu săptămâna 40) dar la un număr de 178 de persoane nu se cunoştea săptămâna cu exactitate şi valorile completate erau de genul 37/38, 38/39, etc. Acestea valori au fost înlocuite cu valori medii (de exemplu săptămâna 37.5).
- Au fost eliminate instanţele care conţineau valori lipsă, sau valori introduse greşit. Valori lipsă sau introduse greşit au fost în coloanele:
    - luna naşterii avea valoarea F pentru o persoană
    - vârsta pentru o persoană avea valoarea 0 şi pentru 10 persoane nu a fost completată
    - pentru 6 persoane coloana număr săptămâni gestaţie avea fie valoarea 0 fie era necompletată
    - coloana mediul urban – rural nu a fost completată pentru două persoane
    - indicele apgar – nu a fost completat pentru 3 nou-născuţi şi pentru 4 nou-născuţi în loc să fie completată nota în coloana indice apgar a fost completat motivul pentru care acesta nu a putut fi acordat şi anume, datorită faptului că doi dintre aceştia au fost născuţi acasă iar alţii 2 pe ambulanţă
    - coloana ore travaliu avea valori lipsă pentru 4 persoane
- Au fost corectate greşelile pentru care s-a putut face acest lucru. De exemplu:
    - Greutatea consemnată la naştere pentru doi copii a fost de 34000 g, respectiv 34450 g. Am presupus că a fost adăugat din greşeală un zero la finalul greutăţii reale şi am eliminat această valoare.
    - Coloana ore travaliu avea date numerice, cuprinse între 0 şi 30, dar la două persoane era scris 20 min. Coloana trebuie să fie uniformă, să conţină fie numărul de ore de travaliu fie numărul de minute de travaliu, prin urmare am transformat totul în ore şi am înlocuit cele două valori 20 min cu 0.33 ore
- Au fost date coloanelor nume sugestive. De exemplu în fişierul primit litera G a fost folosită pentru a denumi atât coloana ce indică numărul de sarcini (gesta) cat şi cea care indică greutatea nou născutului. Noile denumiri alese pentru cele două coloane au fost Gesta şi Greutate.

Prelucrarea fişierului Excel a fost realizată cu ajutorul opţiunii *Filter*. A fost pus filtru pe fiecare coloană şi verificată fiecare coloană în parte ca să conţină doar date valide.

În urma eliminării instanţelor care conţineau date lipsă sau greşite a rămas un număr de 2277 instanţe valide (din 2324).

Următoarea etapă a preprocesării a constat în construirea fişierului *ARFF*, cu datele valide rămase şi încărcarea acestuia în *Weka*. Pentru această etapă a fost folosită aplicaţia *Arff Convertor* [ROB10b], propusă în capitolul 2. Figura 6.1 permite vizualizarea datelor încărcate din fişierul xls în *Arff Convertor*.

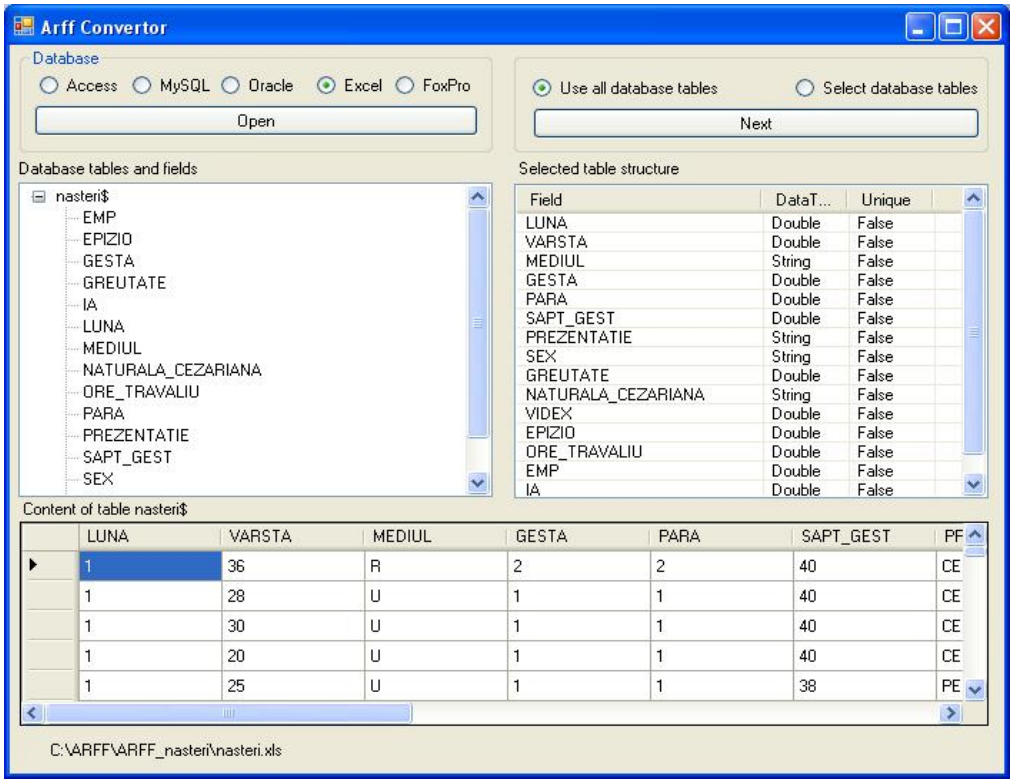

Fig. 6.1 - *Arff convertor* cu datele despre naşteri

Arff convertor a analizat valorile din fiecare câmp şi a propus ca acestea să fie ori numerice, ori categoriale (vezi figura 6.2), utilizatorul fiind cel care alege cu ajutorul unor controale de tip ComboBox tipul acestor atribute în fişierul *ARFF*.

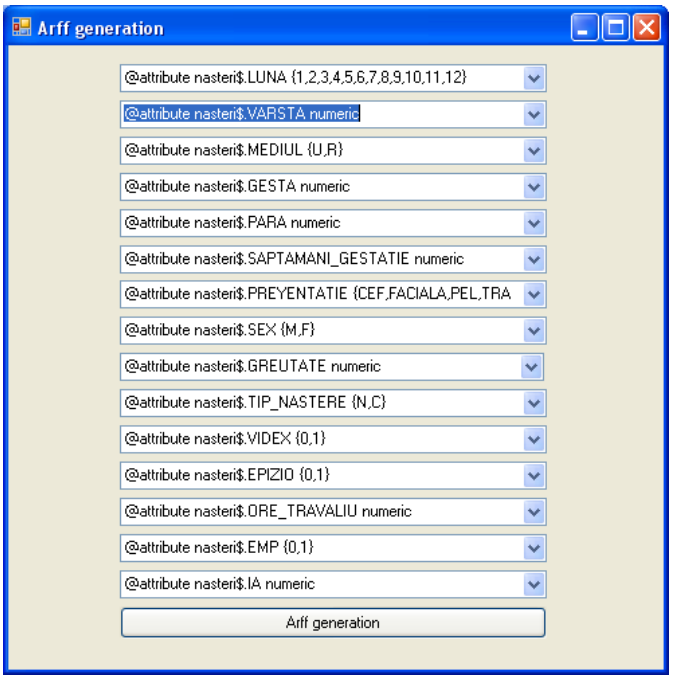

Fig. 6.2. - Alegerea tipului atributelor din fişierul *ARFF*

Figura 6.3 arată fişierul *ARFF* generat cu ajutorul lui *Arff Convertor*.

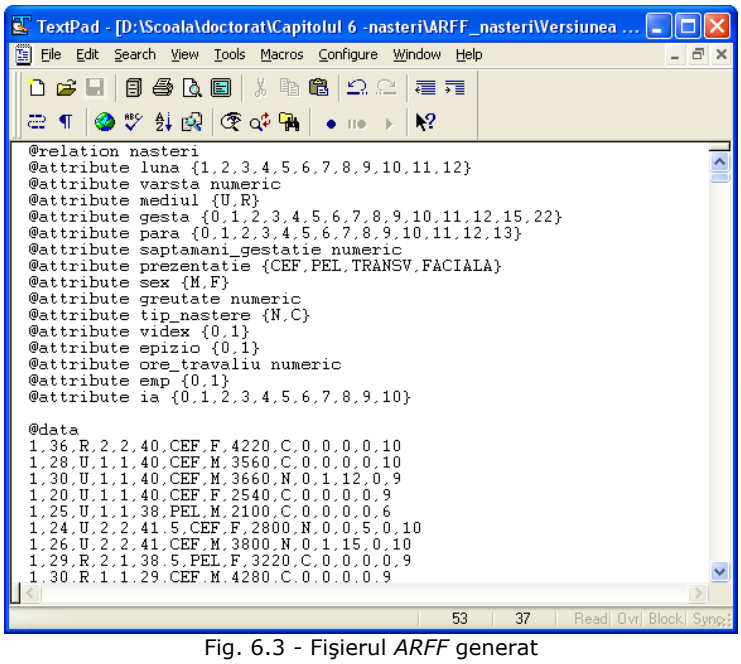

Fig. 6.3 - Fişierul *ARFF* generat

Preprocesarea datelor a continuat cu ajutorul lui *Weka*, dar abia după ce s-a încercat fără succes construirea unor modele clasificatoare rulând algoritmii clasici din *Weka* precum Naive Bayes [ROB11d], J48, k-Nearest Neighbour pe setul de date din figura 6.3. Atributele numerice au fost discreditate fiind împărţite în 5 intervale egale.

6.4. Analiza statistică

În continuare datele preprocesate au fost încărcate în *Weka* şi cu ajutorul acestuia a fost realizată o analiză statistică la nivelul fiecărui atribut (vezi figura 6.4). Numărul de instanţe este de 2277.

După cum se observă în figura 6.5, în luna mai s-au înregistrat cele mai multe naşteri, (245 de naşteri) ceea ce înseamnă că luna septembrie este luna optimă pentru a plănui un copil.

Vârsta femeilor gravide este cuprinsă între 12 şi 45 de ani. Cele mai multe femei rămân însărcinate în jurul vârstei de 28 de ani. 70 de femei minore au rămas însărcinate şi dintre acestea 17 au vârste cuprinse între 12 şi 15 ani. La polul opus 48 de femei cu vârste mai mari de 40 de ani au rămas însărcinate.

Din punct de vedere al mediului din care provin 1493 din femeile care au născut în 2010 la Spitalul Bega provin din mediul urban şi 784 provin din mediul rural.

În ceea ce priveşte indicatorul gesta (numărul de sarcini avute până în prezent) se observă că majoritatea femeilor au avut o sarcină (932 femei) sau două (646 femei), dar există şi situaţii extreme de femei care au avut peste 10 sarcini (24 de femei).

Indicatorul Para arată că majoritatea covârşitoare a femeilor sunt la prima (1313 femei) sau a doua naştere (646 femei). Ca de obicei situaţiile extreme nu lipsesc, avem 8 femei care au născut peste 10 copii.

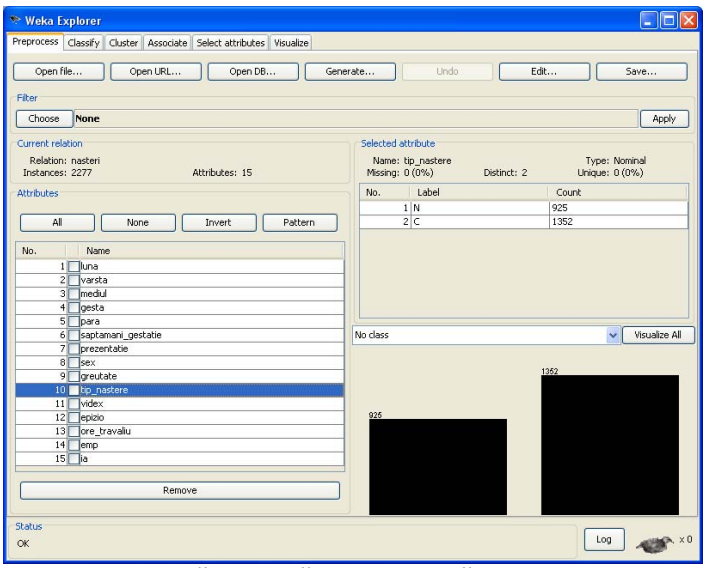

Fig. 6.4 - Analiză statistică la nivelul fiecărui atribut cu *Weka*

Numărul de săptămâni gestaţie este o valoare numerică cuprinsă între 19 şi 44.5, iar valoarea medie este de 38.442 cu o deviaţie standard de 2.438 %.

Prezentaţia normală, cefalică este cea mai răspândită (2115 cazuri), urmată de cea pelviană 156 de cazuri, prezentaţia transversală cu 6 cazuri, iar cea mai rară este prezentaţia facială (un singur caz).

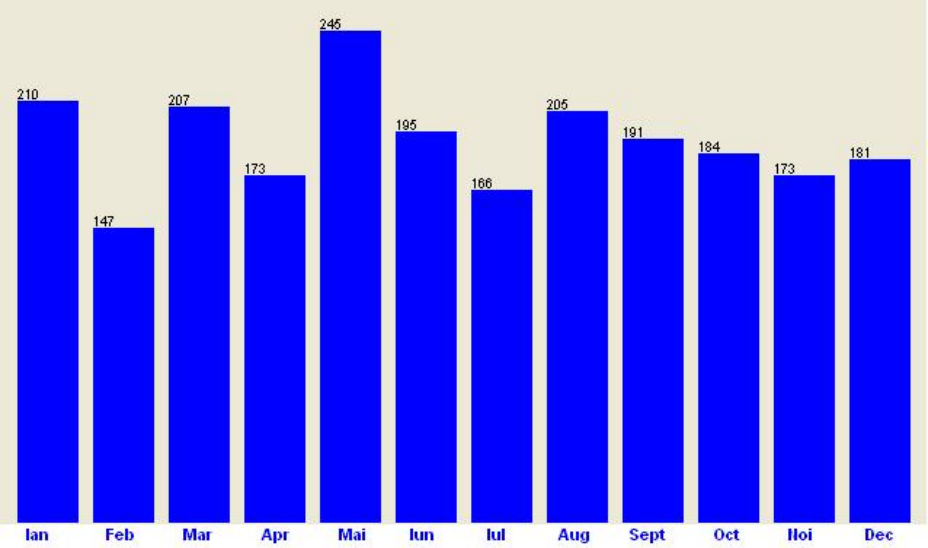

Fig. 6.5 - Numărul de naşteri pe lunile anului

În ceea ce priveşte sexul nou-născuţilor se observă un rezultat surprinzător, au fost născuţi mai mulţi băieţi (1188) decât fete (1089) la Spitalul Bega în 2010.

Greutatea medie a nou născuţilor este de 3182 de grame, cu o deviaţie standard de 589 de grame, dar au fost născuţi 2 bebeluşi care aveau peste 5000 de grame (vezi figura 6.6).

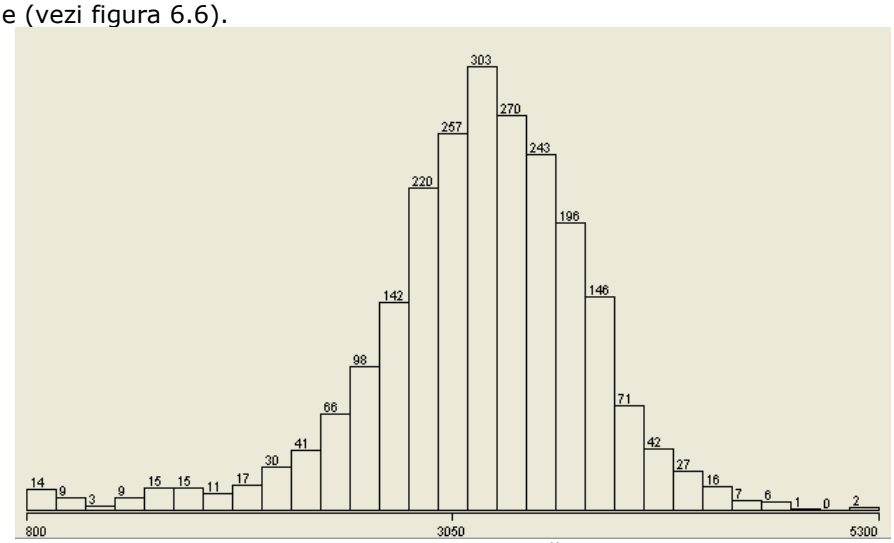

Fig. 6.6 - Greutatea nou născuţilor

Analizând numărul de femei care nasc normal respectiv prin cezariană în zilele noastre, am putea spune ca naşterea naturală este tot mai puţin recomandată sau dorită de femei întrucât în 2010 au născut natural 925 de femei şi prin cezariană 1352 de femei.

Căciuliţă de metal (videx) s-a pus doar pe capul a 20 de bebeluşi din cei 925 care au fost născuţi natural.

Epiziotomia indică dacă a fost necesară realizarea unei tăieturi în zona perineului la femeile care au născut natural. După cum se observă în 665 de cazuri din 925, (aprox. 70 %) a fost necesară această intervenţie.

Numărul de ore de travaliu este cuprins între 0 şi 30 de ore. De obicei 0 ore de travaliu sunt la naşterile prin cezariană.

Extracţie manuală de placentă a fost făcută doar la 7 femei care au născut natural.

Nota primită la naştere este de obicei 10 (990 de cazuri) sau 9 (811 de cazuri), dar după cum se vede în figura 6.7 toate notele cuprinse între 0 şi 10 au fost acordate, inclusiv cele foarte slabe.

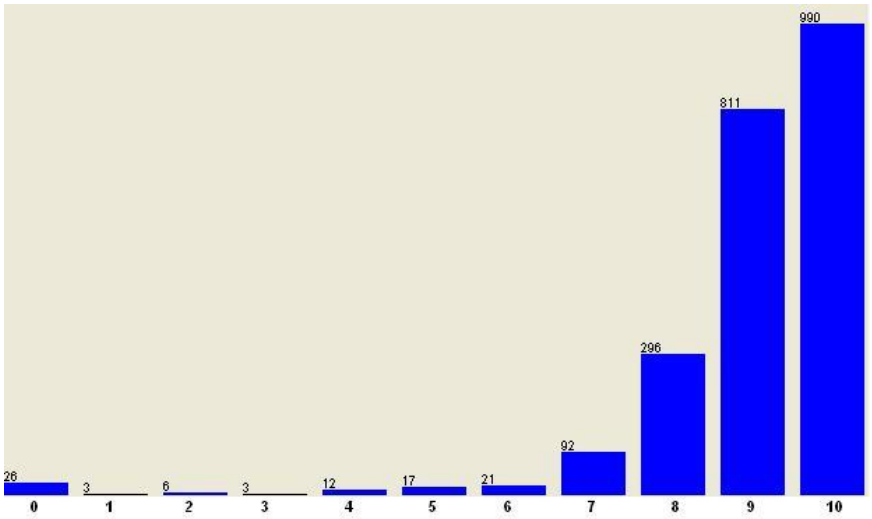

Fig. 6.7 - Indicele apgar

6.5. Clasificarea

În continuare s-a urmărit construirea unor modele clasificatoare, cu ajutorul algoritmilor propuşi dar şi cu ajutorul unor algoritmi clasici, implementaţi în *Weka*, care să clasifice cu o acurateţe cât mai mare datele propuse. Până la determinarea soluţiei optime au fost necesare multiple reveniri la etapa de preprocesare. Luna naşterii a fost păstrată în fişierul *arff* iniţial pentru analiza statistică dar ulterior a fost eliminată, cu ajutorul lui *Weka*, fiind considerată nerelevantă pentru studiu. Setul de date considerat întâi pentru analiză, are 2277 de instanţe şi atributele de tipurile de mai jos. Au fost construiţi clasificatori cu ajutorul algoritmilor clasici *Naive Bayes, J48, k-Neares-Neghbour* din *Weka*. Acurateţea acestor clasificatori este extrem de slabă şi este prezentată în tabelul 5.1. Întrucât au fost considerate atribute numerice, rularea algoritmului *Ant-Miner, Ant-r-Miner, AGR* nu a fost

posibilă. Acurateţea predicţiei prezentată în tabelele de mai jos este o acurateţe medie obţinută în urma rulării fiecărui algoritm de 5 ori pe fiecare set de date. La fiecare rulare s-au utilizat 80 % din instanţe pentru antrenare şi 20 % din instanţe pentru testare.

Atributele din fişierul *nasteri_1.arff:*

```
@attribute varsta numeric
@attribute mediul {U,R}
@attribute gesta {0,1,2,3,4,5,6,7,8,9,10,11,12,15,22}
@attribute para {0,1,2,3,4,5,6,7,8,9,10,11,12,13}
@attribute saptamani_gestatie numeric
@attribute prezentatie {CEF,PEL,TRANSV,FACIALA}
@attribute sex {M,F}
@attribute greutate numeric
@attribute tip_nastere {N,C}
@attribute videx {0,1}
@attribute epizio {0,1}
@attribute ore_travaliu numeric
@attribute emp {0,1}
@attribute ia {0,1,2,3,4,5,6,7,8,9,10}
```

Algoritm	Acuratete
Naive Bayes	46.81
J48	39.78
k-Nearest Neighbour	40.21

Tabel 5.1 – acurateţea predicţiei pe setul de date nasteri_1.arff

Întrucât clasificatorii obţinuţi au fost extrem de slabi, s-a revenit în etapa de preprocesare şi s-au discreditat atributele numerice *varsta, saptamani_gestatie, greutate, ore_travaliu* cu ajutorul lui *Weka.* Discretizarea s-a făcut împărţind fiecare din atributele numerice menţionate în 5 intervale egale (fiecare atribut numeric a devenit atribut nominal cu 5 valori posibile), după care s-a reîncercat construirea unui model clasificator. Algoritmii clasici din *Weka* au fost rulaţi din nou, dar datorita faptului că toate atributele obţinute sunt acum nominale au putut fi rulaţi şi algoritmii *Ant-Miner, Ant-r-Miner şi AGR*. Rezultatele obţinute sunt prezentate în tabelul 5.2.

Atributele din fişierul *nasteri_2.arff:*

```
@attribute varsta {'\'(-inf-18.6]\'','\'(18.6-25.2]\'','\'(25.2-31.8]\'','\'(31.8-38.4]\ ',
                   '\'(38.4-inf)\''}
@attribute mediul {U,R}
@attribute gesta {0,1,2,3,4,5,6,7,8,9,10,11,12,15,22}
@attribute para {0,1,2,3,4,5,6,7,8,9,10,11,12,13}
@attribute saptamani_gestatie {'\'(-inf-24.1]\'','\'(24.1-29.2]\'','\'(29.2-33.4\]'',
                               '\'(34.3-39.4]\'','\'(39.4-inf)\''}
@attribute prezentatie {CEF,PEL,TRANSV,FACIALA}
@attribute sex {M,F}
@attribute greutate {'\'(-inf-1700]\'','\'(1700-2600]\'','\'(2600-3500]\'',
```

```
                                '\'(3500-4400]\'','\'(4400-inf)\''}
@attribute tip_nastere {N,C}
@attribute videx {0,1}
@attribute epizio {0,1}
@attribute ore_travaliu {'\'(-inf-6]\'','\'(6-12]\'','\'(12-18]\'','\'(18-24]\'','\'(24-inf)\''}
@attribute emp {0,1}
@attribute ia {0,1,2,3,4,5,6,7,8,9,10}
```

Algoritm	Acurateţe
Naive Bayes	47.91
J48	45.93
k-Nearest Neighbour	43.73
AGR	54.6
Ant-Miner	43.3
Ant-r-Miner	45.98

Tabel 5.2 - Acurateţea predicţiei pe setul de date nasteri_2.arff

Chiar dacă algoritmul AGR propus a obţinut cel mai bun rezultat, şi acesta este neacceptabil de slab, prin urmare încercările de prelucrare a datelor şi de rulare a algoritmilor pe datele prelucrate au continuat. S-a revenit la etapa de preprocesare, atributele gesta şi para au fost transformate în atribute numerice şi apoi discreditate. Intervalele obţinute după discreditare pot fi vizualizate mai jos. Rezultatele obţinute rulând algoritmii pe aceste date discreditate pot fi vizualizate în tabelul 5.3

Atributele din fişierul *nasteri_3.arff:*

```
@attribute varsta {'\'(-inf-18.6]\'','\'(18.6-25.2]\'','\'(25.2-31.8]\',
                   '\'(31.8-38.4]\'','\'(38.4-inf)\''}
@attribute mediul {U,R}
@attribute gesta {'\'(-inf-4.4]\'','\'(4.4-8.8]\'','\'(8.8-13.2]\'','\'(13.2-17.6]\',
                  '\'(17.6-inf)\''}
@attribute para {'\'(-inf-2.6]\'','\'(2.6-5.2]\'','\'(5.2-7.8]\'','\'(7.8-10.4]\',
                 '\'(10.4-inf)\''}
@attribute saptamani_gestatie {'\'(-inf-24.1]\'','\'(24.1-29.2]\',
                               '\'(29.2-34.3]\'','\'(34.3-39.4]\'','\'(39.4-inf)\''}
@attribute prezentatie {CEF,PEL,TRANSV,FACIALA}
@attribute sex {M,F}
@attribute greutate {'\'(-inf-1700]\'','\'(1700-2600]\'','\'(2600-3500]\',
                     '\'(3500-4400]\'','\'(4400-inf)\''}
@attribute tip_nastere {N,C}
@attribute videx {0,1}
@attribute epizio {0,1}
@attribute ore_travaliu {'\'(-inf-6]\'','\'(6-12]\'','\'(12-18]\'','\'(18-24]\'','\'(24-inf)\''}
@attribute emp {0,1}
@attribute ia {0,1,2,3,4,5,6,7,8,9,10}
```

Algoritm	Acurateţe
Naive Bayes	48.35
J48	44.61
k-Nearest Neighbour	42.85

AGR	67.1
Ant-Miner	43.21
Ant-r-Miner	46.34

Tabel 5.3 - Acurateţea predicţiei pe setul de date nasteri_3.arff

Rezultatele obţinute arată că în urma reducerii împrăştierii valorilor, pentru atributele gesta şi para modelele construite au o acurateţe mai bună, dar nesatisfăcătoare în continuare. În următoarea încercare s-a discretizat atributul clasă, indicele apgar şi atributele gesta şi para au fost lăsate cu valori nominale la fel ca în fişierul *naşteri_2.arff*. Rezultatele obţinute sunt prezentate în tabelul 5.4.

Atributele din fişierul *nasteri_4.arff:*

```
@attribute varsta {'\'(-inf-18.6]\'','\'(18.6-25.2]\'','\'(25.2-31.8]\'',
                   '\'(31.8-38.4]\'','\'(38.4-inf)\''}
@attribute mediul {U,R}
@attribute gesta {0,1,2,3,4,5,6,7,8,9,10,11,12,15,22}
@attribute para {0,1,2,3,4,5,6,7,8,9,10,11,12,13}
@attribute saptamani_gestatie {'\'(-inf-24.1]\'','\'(24.1-29.2]\'',
                               '\'(29.2-34.3]\'','\'(34.3-39.4]\'','\'(39.4-inf)\''}
@attribute prezentatie {CEF,PEL,TRANSV,FACIALA}
@attribute sex {M,F}
@attribute greutate {'\'(-inf-1700]\'','\'(1700-2600]\'','\'(2600-3500]\'',
                     '\'(3500-4400]\'','\'(4400-inf)\''}
@attribute tip_nastere {N,C}
@attribute videx {0,1}
@attribute epizio {0,1}
@attribute ore_travaliu {'\'(-inf-6]\'','\'(6-12]\'','\'(12-18]\'','\'(18-24]\'','\'(24-inf)\''}
@attribute emp {0,1}
@attribute ia {'\'(-inf-2]\'','\'(2-4]\'','\'(4-6]\'','\'(6-8]\'','\'(8-inf)\''}
```

Algoritm	Acurateţe
Naive Bayes	78.78
J48	78.43
k-Nearest Neighbour	75.49
AGR	84.86
Ant-Miner	78.88
Ant-r-Miner	79.01

Tabel 5.4 - Acurateţea predicţiei pe setul de date nasteri_4.arff

Rezultatele din tabelul 5.4 arată că problema principală pentru care modelele construite pe seturile de date *nasteri_1.arff, nasteri_2.arff şi nasteri_3.arff*, au avut rezultate foarte slabe a fost numărul mare de valori al atributului clasa. Odată ce numărul de valori al acestui atribut a fost înjumătăţit prin transformarea valorilor în intervale, toţi algoritmii au obţinut rezultate substanţial mai bune.

O ultima încercare de a obţine rezultate mai bune a constat în revenirea în etapa de preprocesare şi reducere numărul de valori al atributelor gesta şi para din setul de date nasteri_4.arff, prin discretizare.

Atributele din fişierul *nasteri_5.arff:*

```
@attribute varsta {'\'(-inf-18.6]\'','\'(18.6-25.2]\'','\'(25.2-31.8]\','
                    \'(31.8-38.4]\'','\'(38.4-inf)\''}
@attribute mediul {U,R}
@attribute gesta {'\'(-inf-4.4]\'','\'(4.4-8.8]\'','\'(8.8-13.2]\'','\'(13.2-17.6]\',
                  '\'(17.6-inf)\''}
@attribute para {'\'(-inf-2.6]\'','\'(2.6-5.2]\'','\'(5.2-7.8]\'','\'(7.8-10.4]\',
                '\'(10.4-inf)\''}
@attribute saptamani_gestatie {'\'(-inf-24.1]\'','\'(24.1-29.2]\',
                                '\'(29.2-34.3]\'','\'(34.3-39.4]\'','\'(39.4-inf)\''}
@attribute prezentatie {CEF,PEL,TRANSV,FACIALA}
@attribute sex {M,F}
@attribute greutate {'\'(-inf-1700]\'','\'(1700-2600]\'','\'(2600-3500]\',
                     '\'(3500-4400]\'','\'(4400-inf)\''}
@attribute tip_nastere {N,C}
@attribute videx {0,1}
@attribute epizio {0,1}
@attribute ore_travaliu {'\'(-inf-6]\'','\'(6-12]\'','\'(12-18]\'','\'(18-24]\'','\'(24-inf)\''}
@attribute emp {0,1}
@attribute ia {'\'(-inf-2]\'','\'(2-4]\'','\'(4-6]\'','\'(6-8]\'','\'(8-inf)\''}
```

Algoritm	Acuratete
Naive Bayes	77.14
J48	78.46
k-Nearest Neighbour	77.14
AGR	82.67
Ant-Miner	78.7
Ant-r-Miner	79.53

Tabel 5.5 - Acurateţea predicţiei pe setul de date nasteri_5.arff

Cel mai bun rezultat l-a obţinut modelul creat cu algoritmul AGR (acurateţe a predicţiei de 84.86 %), pe setul de date 4.arff.

6.6. Predicţii

Setul de reguli determinat poate fi utilizat în domeniul medical pentru a prezice cu o acurateţe de aprox 85 % care va fi intervalul în care se va situa indicele apgar al nou-născutului, în funcţie de datele mamei, datele estimative ale copilului (greutatea) şi în funcţie de intervenţiile care se fac mamei.

Pentru realizarea predicţiilor s-a utilizat şi se recomandă utilizarea pe viitor a versiunii lui *Weka*, ce dispune de o interfaţă dinamică [ROB10a] cu care se pot realiza predicţii cu uşurinţă (vezi figura 6.8 şi capitolul 5) [ROB10c].

O altă utilitate a setului de reguli descoperit de algoritmul AGR a fost utilizarea lui ca set de date de intrare pentru un sistem care le transpune în sintaxă ARDEN [HEE05]. Pentru ca setul de reguli descoperit să poată fi utilizat ca set de date de intrare pentru sistemul menţionat a fost necesară punerea acestuia în

format XML. Pentru a genera acest fişier XML a fost dezvoltată o aplicaţie în C care primeşte la intrare un fişier text cu setul de reguli descoperit de AGR şi creează un fişier XML cu structura din figura 6.9.

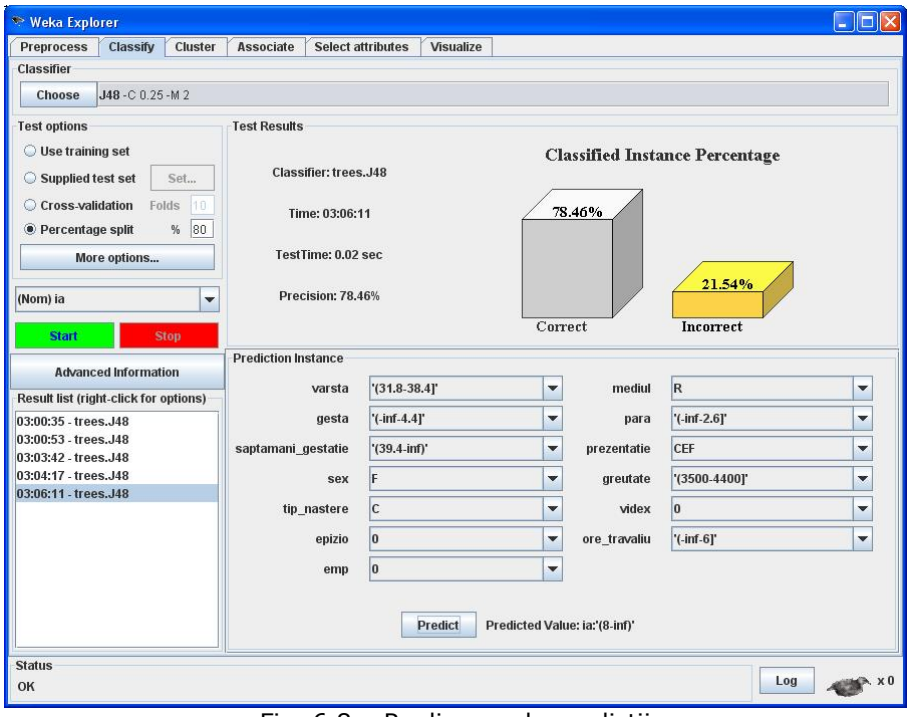

Fig. 6.8 – Realizarea de predicţii

O regulă obţinută de algoritmul AGR are structura de mai jos.

IF atribut1=valoare1 or atribut1=valoare2 and atribut2=valoare1 and…THEN clasa=calsa2

Prelucrările pe care aplicaţia dezvoltată le face sunt următoarele:

- deschide fişierul de intrare
- scrie în fişierul de ieşire versiunea XML şi codificarea
- scrie în fişierul de ieşire tag-ul rădăcină <naşteri>
- cât timp nu s-a parcurs tot fişierul de intrare:
- citeşte un grup de litere (grupurile de litere sunt separate prin spatii)
- dacă grupul de litere este egal cu "IF" atunci scrie tag-ul

  <regula id=contor>

- dacă grupul de litere conţine caracterul "=" şi în regula curentă NU s-a ajuns la THEN, atunci acesta se împarte în două grupuri cu ajutorul delimitatorului "=", primul grup

reprezintă atributul iar al doilea valoarea lui: <atribut>valoare</atribut>

- dacă grupul de litere curent este egal cu AND sau cu OR se adaugă în fişierul XML tag-ul <operator >valoarea_operatorului</operator >

- dacă grupul de litere este egal cu THEN, se reţine acest lucru

- dacă grupul de litere conţine caracterul ”=” şi în regula curentă s-a ajuns la THEN, atunci înseamnă că s-a ajuns la clasă, se împarte aceasta în două grupuri cu ajutorul delimitatorului ”=”, primul grup reprezintă clasa, iar al doilea valoarea ei şi se adaugă în fişierul XML tag-ul <atribut>valoare</atribut> şi se închide tag-ul </regula>

- dacă s-a parcurs tot fişierul de intrare, atunci închide tag-ul rădăcină </nasteri>

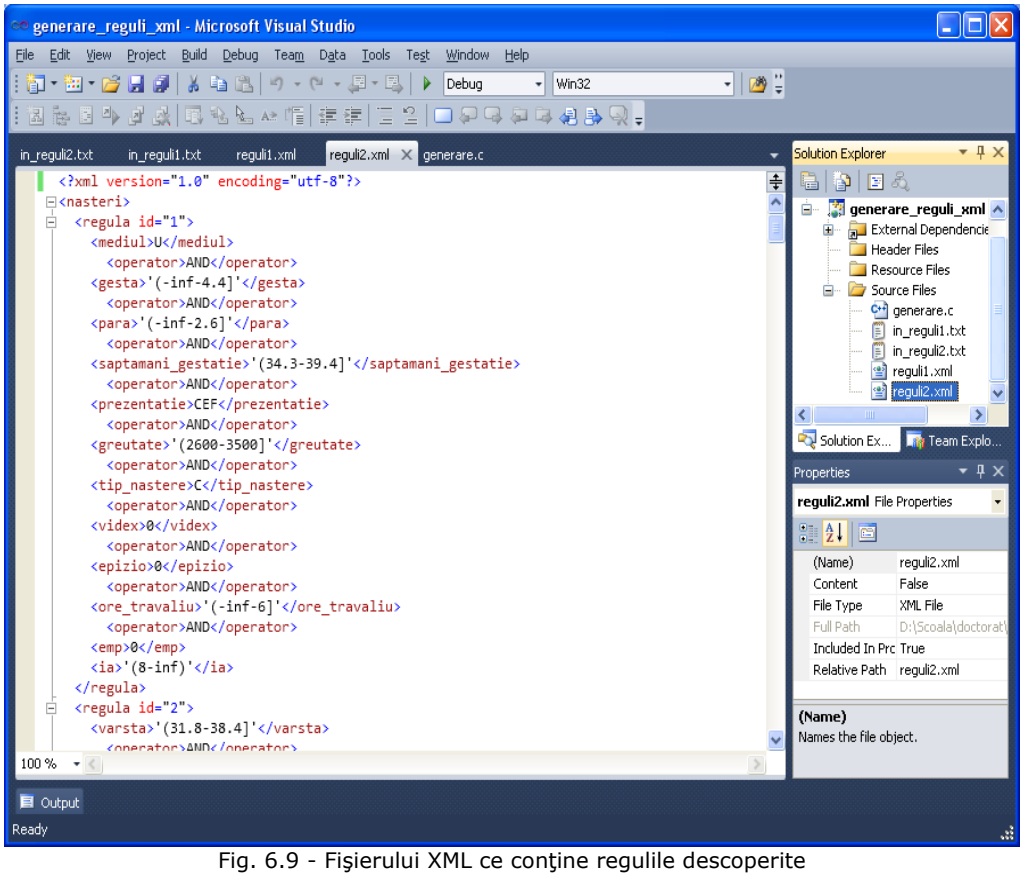

Fig. 6.9 - Fişierului XML ce conţine regulile descoperite

6.7. Concluzii

În cadrul acestui capitol s-a analizat un set de date privind naşterile realizate la *Clinica de Obstetrică - Ginecologie Bega, Timişoara* în anul 2010. Setul de date de la care a pornit analiza conţine date despre 2326 de naşteri. Datele obţinute au fost preprocesate deoarece conţineau valori lipsa, valori introduse greşit, valori neimportante pentru analiză şi apoi aduse în format *arff* ca sa poată fi analizate cu ajutorul lui *Weka*. O primă etapă a analizei datelor a constat în realizarea unei analize statistice la nivel de atribut. Această analiză a condus la determinarea unor informaţii interesante legate de naşterile din 2010, precum: în 70 % din cazuri femeilor care au născut natural a fost nevoie de ajutarea naşterii naturale printr-o tăietură în zona perineului, 60 % din femei au născut prin cezariană, majoritatea femeilor au născut în jurul vârstei de 28 de ani, s-au născut mai mulţi băieţi decât fete, etc. Următoarea etapă a constat în construirea cu ajutorul lui *Weka* a unui model clasificator, care să permită determinarea indicelui apgar pe baza unor variabile predictive precum gesta, para, epizionomie, etc. cu ajutorul unuia din algoritmii propuşi sau a celor existenţi în *Weka*. Cele mai bune rezultate s-au obţinut cu ajutorul algoritmului AGR (vezi capitolul 4). Modelul clasificator construit permite determinarea cu o precizie de aprox 85 % a intervalului în care se va situa indicele apgar (unul din 5 intervale) al nou-născutului, în funcţie de parametrii mamei, ai fătului şi ai intervenţiilor medicale care se vor face asupra mamei respectiv asupra nou-născutului.

Indicele apgar reprezintă o apreciere a funcţiilor vitale şi a capacităţii de adaptare a fătului la condiţiile din mediul extrauterin. Este de dorit ca valoarea acestui indicator să fie cât mai mare pentru fiecare nou-născut. Modelul construit poate fi utilizat înainte de naştere, pentru a realiza predicţii ale intervalului în care se va plasa acest indice (unul din cele 5 intervale) în funcţie de datele care se cunosc despre mamă, copil şi intervenţiile medicale care urmează să se realizeze pentru a ajuta naşterea. Dacă rezultatele predicţiei dau un indice apgar mic este necesar să se realizeze mai multe predicţii cu modelul construit, la fiecare predicţie modificând parametrii de intrare care pot fi modificaţi în practică de medici, pentru ca acest indice apgar să se maximizeze (de exemplu se poate modifica tipul de naştere sau se poate ajuta fătul să ajungă în poziţia normală, etc). Ceea ce se caută este modul în care pot fi modificate intrările astfel încât la ieşire să se obţină o valoare mulţumitoare. Un exemplu al utilizării modelului este: se realizează predicţii şi se analizează în ce interval ar primi nou născutul nota (indicele apgar) dacă fătul s-ar naşte natural respectiv prin cezariană. Dacă se estimează că ar primi notă mai mare dacă s-ar naşte natural, atunci poate că mamele şi / sau medicii se vor gândi de două ori înainte de a face cezariană. Situaţia poate fi valabilă şi invers.

O altă utilitate a modelului dezvoltat, pe lângă utilizarea lui pentru a realiza predicţii cu privire la intervalul în care se va situa indicele apgar, este utilizarea setului de reguli, transpuse în format XML, ca intrare într-un sistem care le transpune în Arden Syntax [HEE05]. Pentru aceasta a fost dezvoltată o aplicaţie care primeşte la intrare un set de reguli în formatul if…then… şi le transformă în format XML.

Dezvoltarea acestui capitol a fost realizată utilizând cunoştinţele şi îmbunătăţirile tehnologice aduse şi prezentate în toate capitolele anterioare. Punctual, contribuţiile aduse de autor în cadrul acestui capitol sunt:

- Preprocesarea în Microsoft Excel a datelor legate de naşterile din 2010 prin analiza fiecărei coloane, eliminarea instanţelor cu valori lipsă sau valori introduse greşit, corectarea valorilor introduse greşit, eliminarea atributelor neimportante pentru analiză, crearea unor noi atribute, etc
- Transformarea datelor din format *xls* în formatul *arff*, specific *Weka* cu ajutorul aplicaţiei *Arff Convertor*, concepută, implementată şi prezentată de autor în cadrul capitolului 2
- Preprocesarea datelor cu ajutorul lui *Weka*, prin discreditarea atributelor numerice
- Analiza statistică la nivel de atribut, care a condus la obţinerea unor informaţii noi şi interesante privind naşterile
- Dezvoltarea unor modele de clasificare cu algoritmii *Naive Bayes, J48, k-Nearest-Neighbour* din *Weka*, algoritmi care au fost prezentaţi în introducere
- Dezvoltarea unor modele de clasificare cu algoritmii *Ant-Miner şi Ant-r-Miner*, prezentaţi în capitolul 3
- Dezvoltarea unui model de clasificare care poate estima cu o precizie de 85 % unul din cele 5 intervale în care indicele apgar al nou-născutului va lua valoare. Modelul a fost construit cu ajutorul algoritmului AGR, care a fost conceput, implementat şi prezentat de autor în cadrul capitolului 4
- Realizarea de predicţii – modelul construit a fost utilizat pentru a realiza predicţii cu ajutorul noii interfeţe dinamice a lui *Weka,* concepută şi prezentată de autor în cadrul capitolului 5
- Setul de reguli descoperit a fost transpus în format XML cu ajutorul unei aplicaţii concepute şi implementate de autor, pentru a servi ca intrare unui sistem care le transpune în Arden Syntax [HEE05].

7. Concluzii

Clasificarea este una dintre cele mai populare tehnici de data mining. Clasificarea permite construirea unui model clasificator pe baza datelor de antrenare, testarea acestuia pe datele de test şi apoi utilizarea lui pentru a realiza predicţii. Construirea modelului clasificator se poate face cu ajutorul unor algoritmi specializaţi precum *Naive Bayes, k-Nearest Neighbor, C4.5* etc. Modelul construit poate fi utilizat pentru a prezice valoarea nominală a unui atribut clasă.

Algoritmii de clasificare au un domeniu de aplicabilitate foarte larg. Ei pot fi utilizaţi cu succes pentru clasificarea amprentelor, bolilor de inimă, a riscului de reapariţie a unor evenimente specifice cancerului de sân, emoţiilor de pe faţa umană, imaginilor de la sateliţi, etc. Prin intermediul algoritmilor de clasificare o universitate poate prezice cu o precizie între 75 şi 80 % care studenţi vor absolvi şi care nu. Universitatea poate folosi această informaţie pentru a-şi concentra atenţia asupra acelor studenţi care sunt în pericol de a nu absolvi [DEK09].

În continuare sunt prezentate aspectele tratate pe parcursul tezei şi contribuţiile aduse la nivel de capitol.

În *capitolul 1* a fost realizată o sinteză amplă, folosind 47 de surse bibliografice, a celor mai importante aspecte legate de domeniul Data Mining. Au fost prezentate definiţii şi terminologia folosită în domeniu, etapele procesului de data minig, domenii în care tehnicile de data mining se aplică cu succes, aspecte legate de clasificarea datelor şi de algoritmii folosiţi pentru a construi modele clasificatoare, aspecte legate de instrumentele software utilizate în studii de data mining şi de seturile de date utilizate pentru testarea şi validarea algoritmilor. Structura tezei a fost de asemenea prezentată.

Contribuţiile aduse în acest capitol sunt următoarele:

- A fost conturat domeniul, subliniind importanţa lui, au fost prezentaţi succint termenii existenţi, un număr mare de definiţii existente în domeniu, a fost introdusă o nouă definiţie, au fost prezentate original etapele procesului de date mining, pe baza experienţei practice a autorului
- A fost realizată o sinteză amplă asupra aplicării tehnicilor de data mining în domeniile e-commerce, educaţie, medicină, combaterea terorismului. De asemenea a fost prezentată o sinteză a sistemelor de *data mining* utilizate de agenţiile federale din SUA cu scopul de sublinia puterea şi importanţa tehnologiei *data mining*, precum şi nivelul de vârf la care s-a ajuns cu utilizarea ei
- Au fost prezentate clar, succint aspecte generale legate de clasificarea datelor, precum şi principiile de bază privind funcţionarea a trei algoritmi de clasificare de referinţă (*Naive Bayes, k-Nearest Neighbor, C4.5*) care au fost utilizaţi în studiile realizate în capitolele 4, 5 şi 6
- Au fost prezentate pe scurt aspecte legate de două dintre cele mai populare instrumente, la nivel mondial, utilizate pentru a realiza studii de data mining (*Weka* şi *R*). Au fost analizate avantajele şi dezavantajele pe care le are

fiecare instrument şi a fost realizată o comparaţie între cele două instrumente
- Au fost prezentate aspecte legate de cele 10 seturi de date donate, reale, obţinute de pe *UCI Machine Learning Repository*, care au fost utilizate pe parcursul tezei, în capitolele 3, 4 şi 5 pentru a testa şi valida algoritmii propuşi şi îmbunătăţirile tehnologice aduse.

În *capitolul 2* au fost prezentate aspecte legate de preprocesarea datelor, aceasta fiind o etapă foarte importantă a procesului de data mining, etapa cea mai mare consumatoare de timp. Contribuţiile care au fost aduse de capitolul 2 sunt următoarele:
- Realizarea unei sinteze ample şi riguroase cu privire la principalele tehnici de preprocesare a datelor, bazată pe studiul a 30 de referinţe. Au fost tratate următoarele tehnici de preprocesare, agregarea, eşantionarea, reducerea dimensionalităţii (prin analiza componentelor principale, descompunerea valorilor singulare, analiză factorială, încapsularea locală lineară, scalare multidimensională), selectarea de submulţimi de caracteristici, crearea caracteristicilor, discretizarea şi binarizarea, transformarea variabilelor, etc.
- Îmbunătăţirea tehnologiei de preprocesare a datelor prin concepţia şi dezvoltarea unui instrument software care permite conversia datelor din 5 formate de baze de date în formatul de date folosit de *Weka* (arff). Instrumentul propus dispune de interfeţe dinamice, în acord cu datele care se prelucrează, iar utilizarea lui este simplă şi nu necesită realizarea de configurări în sistemul de operare, în *Weka* sau cunoştinţe de SQL
- Realizarea unei analize a datelor studenţilor de la facultatea de Automatică şi Calculatoare din anul universitar 2008-2009 cu ajutorul lui *Arff Convertor* şi *Weka*, analiză care oferă o imagine de ansamblu asupra provenienţei studenţilor, pregătirii lor, etc.

În *capitolul 3* sunt tratate aspecte legate de clasificarea datelor cu ajutorul algoritmului *Ant Colony Optimization*. Din anul 2001 când a fost propusă utilizarea lui pentru descoperirea regulilor de clasificare, au fost realizate multe studii cu ajutorul acestui algoritm şi problema alegerii unei secvenţe de parametri optime a fost lăsată ca o direcţie de cercetare de foarte mulţi cercetători. În cadrul acestui capitol a fost studiat codul algoritmului şi prezentată în detaliul funcţionarea lui, au fost analizate foarte multe configuraţii de parametri şi s-a emis o listă de recomandări privind alegerea acestora. De asemenea problema îmbunătăţirii algoritmului a fost atinsă de mulţi cercetători. Au fost prezentate sintetic îmbunătăţirile pe care o parte din aceştia le-au adus şi au fost propuse şi operate noi îmbunătăţiri. Acest lucru a fost posibil datorită faptului că aplicaţia care implementează algoritmul se numeşte Ant Miner şi este Open Source.

Contribuţiile aduse în cadrul acestui capitol sunt următoarele:
- A fost realizată o prezentare detaliată a algoritmului, bazată atât pe documentaţia existentă cât şi pe studiul codului algoritmului.
- Au fost studiaţi parametri algoritmului rulând câte 64 de configuraţii pe fiecare din cele 10 seturi de date alese şi analizând rezultatele. Analiza rezultatelor s-a făcut atât vizual, comparativ cât şi matematic construind şi analizând modele de regresie şi matrice de corelare

- Analiza făcută a condus către o listă de recomandări privind modul de alegere al parametrilor, utile pentru toţi utilizatorii acestui algoritm
- A fost realizată o sinteză a îmbunătăţirilor aduse algoritmului din anul 2001 şi până în prezent
- Au fost aduse îmbunătăţiri algoritmului care vizează modul de calcul a cantităţi iniţiale de feromon a termenilor, modul de actualizare a feromonului, modul de evaluare a calităţii regulilor descoperite şi modul de selecţie a termenilor. Modificările propuse au fost implementate în aplicaţia open source *Ant Miner*. Aplicaţia care conţine algoritmul îmbunătăţit a fost numită *Ant-r-Miner*. S-a demonstrat că îmbunătăţirile aduse conduc la o mai bună performanţă utilizând date reale.

În *capitolul 4* este propus un nou algoritm genetic pentru clasificarea datelor. Algoritmul propus a fost denumit *AGR*. Contribuţiile aduse de acest capitol sunt următoarele:

- Prezentarea succintă a structurii generale a unui algoritm genetic
- Realizarea unei sinteze cu privire la algoritmii genetici utilizaţi la clasificarea datelor
- Propunerea unui nou algoritm genetic pentru clasificarea datelor care după cunoştinţele autorului diferă de ceilalţi algoritmii genetici de clasificare prin următoarele:
  - permite recalcularea clasei în urma încrucişării
  - permite stabilirea unui procent minim de acoperire pentru regulile descoperite, rularea repetată a algoritmului până se atinge acest procent
  - permite adăugarea unei reguli implicite listei regulilor descoperite
  - permite adăugarea la lista de reguli descoperite doar a celei mai bune reguli obţinută după rularea algoritmului pentru numărul de generaţii stabilit sau a tuturor regulilor descoperite
  - utilizează o funcţie *fitness* diferită de cele utilizate de alţi algoritmi
- Îmbunătăţirea tehnologiei de clasificare a datelor prin implementarea lui *AGR* într-un instrument Open Source utilizat pe scară largă (*Weka*)
- Analizarea comportamentului algoritmului la diferite valori ale parametrilor de intrare, şi la utilizarea a diferite funcţii *fitness* şi emiterea unui set de recomandări cu privire la alegerea acestora

În cadrul *capitolului 5* a fost tratată problema realizării de predicţii cu ajutorul unui model construit şi testat de *Weka.* Contribuţiile aduse în cadrul acestui capitol sunt:

- Prezentarea detaliată a modului în care *Weka* poate fi utilizat pentru a realiza predicţii
- Conceperea unei interfeţe dinamice care conţine controale în acord cu setul de date analizat şi care uşurează şi clarifică mult procesul de realizare a predicţiilor
- Prezentarea sintetică a mărimilor de ieşire furnizate de *Weka* în urma testării modelului
- Concepţia unor soluţii de transformare a interfeţei de clasificare a datelor într-o interfaţă mult mai simplă, inteligibilă şi prietenoasă, prin vizualizarea sub formă de grafic 3d a celei mai importante măsuri (acurateţea predicţiei) obţinute în urma testării modelului şi prin afişarea informaţiilor din secţiunea

Classifier output (a indicatorilor ”complicaţi”) într-o nouă fereastră *Advanced Information*.

- Modificările menţionate mai sus, realizate în unul din cele mai populare instrumente de data mining din lume, contribuie la îmbunătăţirea tehnologiei de clasificare a datelor. Modificările au fost testate pe 3 seturi de date medicale şi rezultatele au fost foarte bune.

În *capitolul 6*, a fost utilizată tehnologia prezentată şi îmbunătăţirile tehnologice propuse în capitolele precedente pentru a analiza şi clasifica datele despre cele 2326 naşteri care au avut loc la *Clinica de Obstetrică - Ginecologie Bega, Timişoara* în anul 2010. Punctual, contribuţiile aduse de autor în cadrul acestui capitol sunt:

- Preprocesarea în Microsoft Excel a datelor legate de naşterile din 2010 prin analiza fiecărei coloane, eliminarea instanţelor cu valori lipsă sau valori introduse greşit, corectarea valorilor introduse greşit, eliminarea atributelor neimportante pentru analiză, crearea unor noi atribute, etc
- Transformarea datelor din format *xls* în formatul *arff*, specific *Weka,* cu ajutorul aplicaţiei *Arff Convertor*, concepută, implementată şi prezentată de autor în cadrul capitolului 2
- Preprocesarea datelor cu ajutorul lui *Weka*, prin discreditarea atributelor numerice
- Analiza statistică la nivel de atribut, care a condus la obţinerea unor informaţii noi şi interesante privind naşterile
- Dezvoltarea unor modele de clasificare cu algoritmii *Naive Bayes, J48, k-Nearest-Neighbour* din *Weka*, algoritmi care au fost prezentaţi în introducere
- Dezvoltarea unor modele de clasificare cu algoritmii *Ant-Miner şi Ant-r-Miner*, prezentaţi în capitolul 3
- Dezvoltarea unui model de clasificare care poate estima cu o precizie de aproximativ 85 % unul din cele 5 intervale în care indicele apgar al nou-născutului va lua valoare. Modelul a fost construit cu ajutorul algoritmului AGR, care a fost conceput, implementat şi prezentat de autor în cadrul capitolului 4
- Realizarea de predicţii – utilizarea interfeţei dinamice a lui *Weka,* concepută şi prezentată de autor în cadrul capitolului 5 pentru realizarea predicţiilor
- Setul de reguli descoperit a fost transpus în format XML cu ajutorul unei aplicaţii concepute şi implementate de autor, pentru a servi ca intrare unui sistem care le transpune în *Arden Syntax* [HEE05]
- Valoarea medicală a modelului clasificator este următoarea: Indicele apgar reprezintă o apreciere a funcţiilor vitale şi a capacităţii de adaptare a fătului la condiţiile din mediul extrauterin. Este de dorit ca valoarea acestui indicator să fie cât mai mare pentru fiecare nou-născut. Modelul construit poate fi utilizat înainte de naştere, pentru a realiza predicţii a intervalului în care se va plasa acest indice (unul din cele 5 intervale) în funcţie de datele care se cunosc despre mamă, copil şi intervenţiile medicale care urmează să se realizeze pentru a ajuta naşterea. Dacă rezultatele predicţiei dau un indice apgar mic este necesar să se realizeze mai multe predicţii cu modelul construit, la fiecare predicţie modificând parametrii de intrare care pot fi modificaţi în practică de medici, pentru ca acest indice apgar să se maximizeze (de exemplu se poate modifica tipul de naştere, sau se poate ajuta fătul să ajungă în poziţia normală, etc). Ceea ce se caută este modul

în care pot fi modificate intrările astfel încât la ieşire să se obţină o valoare mulţumitoare. Un exemplu al utilizării modelului este: se realizează predicţii şi se analizează în ce interval ar primi nou născutul nota (indicele apgar) dacă fătul s-ar naşte natural respectiv prin cezariană. Dacă se estimează că ar primi notă mai mare dacă s-ar naşte natural, atunci poate că mamele şi / sau medicii se vor gândi de două ori înainte de a face cezariană. Situaţia poate fi valabilă şi invers.

Direcţii de dezvoltare

O direcţie de dezvoltare ulterioară este continuarea studiului din capitolul 3 şi extinderea aplicaţiei *Ant-r-Miner* astfel încât utilizatorul să poată alege dintr-o listă de funcţii, funcţia pe care doreşte să o folosească pentru a evalua calitatea regulilor, respectiv să-şi poată construi în mod dinamic funcţia de calitate, cu ajutorul termenilor *TP, FP, TN, FN* şi a unor operaţii elementare gen adunare, scădere, înmulţire, împărţire. Analiza influenţei pe care o au valorile parametrilor de intrare al algoritmului *Ant Colony Optimization* asupra numărului de reguli şi asupra numărului de condiţii este de asemenea o direcţie de dezvoltare ulterioară.

Continuarea studiului din capitolul 4 este de asemenea o direcţie de dezvoltare. La fel ca şi în cazul lui *Ant-r-Miner* se doreşte dezvoltarea unei facilităţi care să-i permită utilizatorului să aleagă sau să introducă funcţia *fitness* pe care doreşte să o utilizeze în analizele pe care le realizează cu *AGR*. O altă direcţie de dezvoltare este implementarea mai multor metode de selecţie şi oferirea posibilităţii de a alege metoda de selecţie dorită într-un anumit studiu cu *AGR*.

Autorul va aplica tehnologia propusă în domeniul medical, pe cât mai multe seturi de date medicale.

Un alt obiectiv care va fi urmărit este conceperea unei modalităţi de utilizare a algoritmului albină în problema clasificării datelor [PHA07].

Bibliografie

[AIC10] Aici M., Inan C. and Avci M., *A hybrid classification method of k nearest neighbor, Bayesian methods and genetic algorithm*, Expert Systems with Applications, Vol. 37, 2010, 5061-5067

[ALK10] Alkan A., "Cahit Arf", Available from: http://sourceforge.net/projects/cahitarf/, Accessed: 18-03-2010

[ANS01] Ansari S., Kohavi R., Mason L. and Zheng Z. *Integrating E-Commerce and Data Mining: Architecture and Challenges*. Proceedings IEEE International Conference on Data Mining, ISBN:0-7695-1119-8, pp. 27-34, 2001

[ATA10] Atallah M.J. and Blanton M., *Algorithms and Theory of Computation Handbook,* CRC Press, Taylor & Francis Group, LLC, ISBN 978-1-58488-822-2, 2010

[ATI09] Atilgan Y. and Dogan F. *Data Mining on Distributed Medical Databases: Recent Trends and Future Directions*. IT Revolutions, Volume 11, ISBN978-3-642-03977-5, pp. 216-224, 2009

[BEN06] Bendakir N. and Aimeur E. *Using Association Rules for Course Recomandation,* Proceedings of the Twenty-First National Conference on Artificial Intelligence, ISBN 978-1-57735-281-5, 2006

[BER04] Berry M. and Linoff G., *Data Mining Techniques For Marketing, Sales, and Customer Relationship Management,* Wiley Publishing, ISBN 0-471-47064-3, 2004

[BER09] Bertini E. and Lalanne D., *Surveying the complementary role of automatic data analysis and visualization in knowledge discovery*, Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery: Integrating Automated Analysis with Interactive Exploration, 2009

[BHA08] Bhavani Thuraisingham, *Data Mining for Counter-Terrorism*, http://www.utdallas.edu/~bxt043000/key-papers/%5B18%5DMining-Terrorism-NGDM04.pdf , 2008

[BIN11] Bing L., *Web Data Mining Exploring Hyperlinks, Contents, and Usage Data,* Springer, Second Edition, ISBN 978-3-642-19459-7, 2011

[BOU08] Bouckaert R., Frank E., Hall M., Kirkby R., Reutemann P.,

Seewald A., Scuse D., *Weka Manual for Version 3-6-0,* University of Waikato, Hamilton, New Zealand, 2008

[DEH06] Dehuri S., Ghosh A. and Mall R., *Genetic Algorithms for Multi-Criterion Classification and Clustering in Data Mining*, International Jurnal of Computing & Information Sciences, Vol 6, No 3, 2006, pp. 143-154

[DEK09] Dekker G.W., Pechenizkiy M. and Vleeshouwers J.M., *Predicting Students Drop Out: A Case Study*, Proceedings of 2nd International Conference On Educational Data Mining, Cordoba, Spain, July 1-3, ISBN: 978-84-613-2308-1, pp 41-50, 2009

[DEZ09] Deza E. and Deza M.M., *Encyclopedia of Distances*, Springer, ISBN 978-3-642-00233-5, 2009

[DIN09] Ding R., Dong H., Feng X. and Yin G., *A Hybrid Particle Swarm Genetic Algorithm for Classification*, Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization*,* vol.2, 2009, pp. 301

[DOR04] Dorigo M. and Stützle T., *Ant Colony Optimization*, Massachusetts Institute of Technology, ISBN 0-262-04219-3, 2004

[DOR96] Dorigo, M. and Maniezzo, V., *The ant system: optimization by a colony of cooperating agents,* IEEE Transactions on Systems, Man, and Cybernetics*, 26*(1), pp. 1-13, 1996

[DOU95] Dougherty J., Kohavi R. and Sahami M., *Supervised and Unsupervised Discretization of Continuous Features*, Proceedings of the Twelfth International Conference on Machine Learning (1995), pp. 194-202.

[EZE05] Ezeife C.I. and Yi Lu (2005). *Mining Web Log Sequential Patterns with Position Coded Pre-Order Linked WAP-Tree*, Data Mining and Knowledge Discovery, Volume 10, Number 1 / January, ISSN1384-5810, pp 5-38, 2005

[FAL95] Faloutsos C. and Lin K., *FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets*, in Proceedings of 1995 ACM SIGMOD, SIGMOD RECORD (June 1995), vol.24, no.2, pp 163-174

[FIL08] FILIP Ioan, ROBU Raul, SZEIDERT Iosif and ROBU Andreea, *Considerations Regarding an Unusual Cause of Database Concurrency Exception*, Annals of DAAAM for 2008 & Proceedings of the 19th International DAAAM Symposium, pag. 493-494, Trnava, Slovacia, 2008

[FIL09] FILIP Ioan, VAŞAR Cristian and ROBU Raul, *Considerations about*

an Oracle Database Multi-Master Replication, Proc. 5th International Symposium on Applied Computational Intelligence and Informatics, SACI, Timisoara, Romania, pag. 147-152, 2009

[FRA10] Frank, A. and Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[FRE02] Freitas A.A., *A survey of evolutionary algorithms for data mining and knowledge discovery*, Advances in Evolutionary Computation, Springer-Verlag, 2002, pp. 819-845

[GAN07] Gang K., Yi P., Yong S. and Zhengxin C. *Privacy-Preserving Data Mining of Medical Data Using Data Separation–Based Techniques*. Data Science Journal, Vol. 6, pp.429-434, 2007

[GAO04] United States General Accounting Office (GAO), *DATA MINING - Federal Efforts Cover a Wide Range of Uses*, Report to the Ranking Minority Member, Subcommittee on Financial Management, the Budget, and International Security, Committee on Governmental Affairs, U.S. Senate http://www.gao.gov/docsearch/locate?searched=1&o=0&order_by=rel&search_type=publications&keyword=data+mining&Submit=Search

[GOR06] Gorunescu F. *Data Mining, Concepte, Modele şi Tehnici*, Editura Albastra, Cluj-Napoca, ISBN 973-650-169-8, 2006

[HAL09] Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P. and Witten I.H., The *Weka* Data Mining Software: An Update, ACM SIGKDD Explorations Newsletter, Volume 11 , Issue 1, 2009, pp. 10-18

[HEE05] Hee K.P., Jae P.K., Jin W.C., Jae J.H. and Seung M.H., *Design of real-time arden syntax based decision support system with minimum workload and building cost*, Enterprise networking and Computing in Healthcare Industry, 2005. *HEALTHCOM 2005. Proceedings of 7th International Workshop on* , vol., no., pp. 430-433, 23-25 June 2005

[HOL09] Holban S. Data Mining and Machine Learning – Notiţe de curs, 2009

[HOR09] Hornik K., Buchta C. and Zeileis A., *Open-Source Machine Learning: R Meets Weka,* Computational Statistics, ISSN 0943-4062, pp 225-232, 2009

[HU09] Hu X., Zhang X., Lu C., Park E. K. and Zhou X., *Exploiting Wikipedia as External Knowledge for Document Clustering,* Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Paris, France, ISBN:978-

1-60558-495-9, pp 389-396, 2009

[JOL02] Jolliffe I.T., *Principal Component Analysis*, Springer-Verlag, Second Edition, October 2002

[KOT06] Kotsiantis S. B., Kanellopoulos D. and Pintelas P.E., *Data Preprocessing for Supervised Leaning*, International Journal of Computer Science, pp.111-117, 2006

[KOT09] Kotsiantis S., *Educational Data Mining: A case study for predicting drop-out Prone Students,* International Journal of Knowledge Engineering and Soft Data Paradigms, Inderscience Publishers, ISSN:1755-3210, pp 101-111, 2009

[LAL11] Lalbakhsh, P., Fasaei, M.S.K. and Fesharaki M.N., *Focusing on rule quality and pheromone evaporation to improve ACO rule mining*, Proceedings of the IEEE Symposium on Computers & Informatics (ISCI), ISBN: 978-1-61284-689-7, pp. 108 – 112, 2011

[LAR06] Larose D.T., *Data Mining Methods and Model*, Published by John Wiley & Sons, Inc., Hoboken, New Jersey, ISBN-13 978-0-471-66656-1, 2006

[LEH98] Lehmann E. L. and Casella G., *Theory of Point Estimation (2nd ed.),* New York: Springer, ISBN 0-387-98502-6, 1998

[LES06] Leskovec J., *Dimensionality reduction PCA, SVD, MDS, ICA, and friends*, *Machine Learning recitation*, April 27 2006

[LIU02a] Liu B., Abbass H. A. and McKay B, *Density-based heuristic for rule discovery with ant-miner,* Proceedings of the 6th Australasia–Japan joint workshop on intelligent and evolutionary systems (AJWIS2002), Canberra, Australia, pp. 180–184, 2002

[LIU02b] Liu H., Hussain F., Tan C. L. and Dash M., *Discretization: An Enabling Technique*, Data Mining and Knowledge Discovery, Springer Netherlands, Vol. 6, pp. 393-423, 2002

[LIU03a] Liu, B., Abbass H. A. and McKay B. (2003). *Classification rule discovery with ant colony optimization*, Proceedings IEEE/WIC international conference on intelligent agent technology (pp. 83–88). IEEE Comput. Soc.: Los Alamitos.

[LIU03b] Liu H., Yu L., and Motoda H., *Feature Extraction, Selection, and Construction*, The handbook of Data Mining, Edited by Nong Ye, Lawrence Erlbaum Associates, Inc., pp. 409-425, 2003

[MAR05] Marcelon A. and Yacef K. *Educational Data Mining: a Case Study.* Artficial Inteligence in Education: supporting learning through Intelligent and Socially Informed Technology, *IOS Press, 2005*

[MAR11] D. Martens, B. Baesens, and T. Fawcett, *Editorial survey: swarm intelligence for data mining*, Machine Learning, Volume 82, Number 1, pp. 1-42, 2011

[MAT08] Matthews D.C., *Improved Lower Limits for Pheromone Trails in Ant Colony Optimization,* Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X, Springer-Verlag Berlin, Heidelberg, pp. 508 – 517, 2008

[MAU00] Maulik U. and Bandyopadhyay S., *Genetic algorithm-based clustering technique*, Pattern Recognition 33*,* 2000, pp. 1455-1465

[MOB04] Mobasher B., *Web usage mining and personalization*. Practical handbook of internet computing (ed.) M P Singh (CRC Press), 2004

[MOH09] Mohammed N., Fung B., Hung P. and Lee C., *Anonymizing Healthcare Data: A Case Study on the Blood Transfusion Service*, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Paris, France, ISBN:978-1-60558-495-9, pp 1285-1294, 2009

[MOO09] Moore S., D'Addario D., Kurinskas J., and Weiss G. M., *Are Decision Trees Always Greener on the Open (Source) Side of the Fence?* In Proceedings of the 2009 International Conference on Data Mining, CSREA Press, pp185-188, 2009

[MUK09] Mukhopadhyay D.M., Balitanas M.O., Farkhod A., Jeon S.H., and Bhattacharyya D., *Genetic Algorithm: A Tutorial Review*, International Journal of of Grid and Distributed Computing, Vol.2, No.3, 2009, pp. 25-32

[NEJ08] Nejad N. Z., Bakhtiary A.H. and Analoui M., *Classification Using Unstructured Rules and Ant Colony Optimization*, Proceedings of the International MultiConference of Engineers and Computer Scientists, Vol I, ISBN: 978-988-98671-8-8, Hong Kong, pp. 506-510, 2008

[NIU02] Niu L., Yan X.W., Zhang C.Q. and Zhang S.C., *Product hierarchy-based customer profiles for electronic commerce recommendation*. In Int. Conf. on Machine Learning and Cybernetics, pp 1075–1080, 2002

[PAR01] Parpinelli R.S., Lopes H.S. and Freitas A.A., *An ant colony algorithm for classification rule discovery,* Abbas, H.A., Sarker, R.A., Newton, C.S. (eds.), Data Mining: A Heuristic Approach., Hershey: Idea Group Publishing, v. 1, ISBN 1-9307-0825-4, pp. 190-208, 2001

[PAR02] Parpinelli R.S., Lopes H.S. and Freitas, A.A., *Data mining with an ant colony optimization algorithm,* IEEE Transactions on Evolutionary Computation, special issue on Ant Colony Algorithms, v. 6, n. 4, pp. 321-332, August, 2002

[PHA07] Pham D.T., Otri S., Afify A., Mahmuddin M. and Al-Jabbouli H., *Data Clustering Using the Bees Algorithm,* Manufacturing Systems. Retrieved from http://bees-algorithm.com/modules/2/6.pdf , 2007

[PLA10] Platt J.C., *FastMap, MetricMap, and Landmark MDS are all Nystrom Algorithms*, Microsoft Research, Available at: http://research.microsoft.com/pubs/69185/nystrom2.pdf, Accessed: 09-05-2010

[PON07] Ponniah P., *Data Modeling Fundamentals, A Practical Guide for IT Professionals*, Wiley Interscience, A John Wiley and sons, Inc., Publication, ISBN-10: 0-471-79049-4, 2007

[PRO09] PROŞTEAN G., ROBU R., PROŞTEAN O. and TAROATĂ A., *Considerations regarding the integration of the cost centers based on SAP management accounting*, Proc. 6th International Conference on Management of Technological Changes, vol.1, Alexandroupolis, Greece, pag. 699-702, 2009

[QUI86] Quinlan, J.R., *Induction of Decision Trees*, Machine Learning 1, pp 81-106, 1986

[QUI93] Quinlan J. R., *C4.5: Programs for Machine Learning* , Morgan Kaufmann Publishers, Inc., pp.235-240, 1993

[QUI98] Quinlan. R., *C5.0: An Informal Tutorial*, http://www.rulequest.com/see5-unix.html, 1998, Accessed: 05-05-2012

[RAG05] Raghavan Srinivasa N.R. (2005) *"Data mining in e-commerce: A survey".* Sadhana Vol. 30, Parts 2 & 3, pp. 275–289, 2005

[ROB10a] Robu, R., Hora, C. and Stoicu - Tivadar, V. (2010). Improving the Classify User Interface in *Weka* Explorer (2010). 0171-0173, Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, ISBN 978-3-901509-73-5, ISSN 1726-9679, pp 0086, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2010

[ROB10b] Robu R. and Stoicu-Tivadar V., *Arff Convertor Tool for WEKA Data Mining Software*, Proceedings of the IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics, ICCC-CONTI, pag. 247-251, Timisoara, Romania,

2010

[ROB10c] Robu R. and Stoicu-Tivadar V., *Evaluation and Improvement of User Interfaces for Medical Data Mining with Open Source Programs*, Proceedings of the 31st National Conference on Medical Informatics, ISBN 978-606-8054-14-8, pp. 284-289, Arad, 18-20 November, 2010

[ROB11a] ROBU R., FEIER F., STOICU-TIVADAR V., ILIE C., ENĂTESCU I., *The analysis of the new-borns' cry using NEONAT and data mining techniques,* Proceedings of the 15th IEEE International Conference on Intelligent Engineering Systems (INES), Pages 235 - 238, Poprad, Slovacia, 2011

[ROB11b] ROBU R. and HOLBAN S., *A genetic algorithm for classification,* Proceedings of the International Conference on Computers and Computing, (ICCC), Pages 52-56 Lanzarote, Spania, 2011

[ROB11c] ROBU R. and STOICU-TIVADAR V., *Regional data mining system,* Studies in Health Technology and Informatics, Volume 165, (STC), Pages 117-122, Lasko, Slovenia, 2011

[ROB11d] ROBU R., TOLBUŞ A. and FILIP I., *DMCP - A data mining tool for classification and prediction*, Proceedings of the 7th International Conference on Management of Technological Changes (MTC 2011), vol.1, Pages 605-608, Alexandroupolis, Greece, 2011

[ROM08] Romero C., Ventura S. and García E., *Data mining in course management systems: Moodle case study and tutorial*, Computers & Education, Volume 51, Issue 1, pp. 368-384, 2008, Elsevier Science

[SAI11] Said N., Hammami M. and Ghedira K., *MulO-AntMiner: a new ant colony algorithm for the multi-objective classification problem*, Proceedings of the 2011 international conference on Computational science and its applications - Volume Part II, ISBN: 978-3-642-21886-6, 2011

[SAL09] Salle P., Bringay S., Teisseire M., Chakkour F., Roche M., Rassoul R.A., Verdier J.M., Devau G., *GeneMining: Identification, Visualization, and Interpretation of Brain Ageing Signatures*, Proceedings of XXII International Conference of the European Federation for Medical Informatics, 2009

[SAL11] Salama K., Abdelbar A. and Freitas A., *Multiple pheromone types and other extensions to the Ant-Miner classification rule discovery algorithm*, Swarm Intelligence (11 June 2011), pp. 1-34

[SAR00] Sarukkai R.R., *Link prediction and path analysis using markov chains*. Proceedings of the 9th International World Wide Web Conference, Amsterdam, 2000

[SAU00] Saul L.K. and Roweis S.T., *Nonlinear dimensionality reduction by*

locally linear embedding,Science, Vol. 290, pp. 2323–2326, 2000.

[SCH06] Schloeffel P., Beale T., Hayworth G., Heard S. and Leslie H., *The relationship between CEN 13606, HL7 and openEHR*, HIC 2006 and HINZ 2006: Proceedings, Sydney, Australia, 2006, pp.24-28

[SEI08] Seifert, Jeffery W. *Data Mining and Homeland Security: An Overview*. April 3, 2008. Congressional Research Reports for the People http://assets.opencrs.com/rpts/RL31798_20080403.pdf , Accessed:17-05-2008

[SKI03] Skillcorn D.B., *Cluster within Clusters: SVD and counterterrorism*, Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, ISBN 0-89871-545-8, 2003

[SMA06] Smaldon J. and Freitas, A. (2006). *A new version of the Ant-Miner algorithm discovering unordered rule sets,* Proceedings genetic and evolutionary computation conference (GECCO-2006) (pp. 43–50). San Francisco: Morgan Kaufmann

[SON08] Song J., *A new dissimilarity measure in time-dependent experiments,* Journal of the Korean Statistical Society, Volume 37, Issue 2, pp. 145-153, 2008

[STE02] Steyvers M, *Multidimensional scaling*, Encyclopedia of Cognitive Science*,* London, United Kingdom, Nature Publishing Group, 2002

[TAN05] Tan P.-N., Steinbach M. and Kumar V., *Introduction to Data Mining,* Editura Addison Wesley, US, ISBN 0-321-32136-7,2005

[TWO99] Two Crows Corporation, *Introduction to Data Mining and Knowledge Discovery, Third Edition,* ISBN: 1-892095-02-5, 1999

[VAS09] VASAR Cristian, PROSTEAN Octavian, FILIP Ioan, ROBU Raul and POPESCU Dan, *Markov Models for Wireless Sensor Network Reliability*, Proc. IEEE 5th International Conference on Intelligent Computer Communication and Processing, (ICCP), Cluj-Napoca, Romania, pag.323-328, 2009

[VER05] Verleysen M. and François D., *The Curse of Dimensionality in Data Mining and Time Series Prediction*, Proceedings of the 8th international conference on Artificial Neural Networks: computational Intelligence and Bioinspired Systems, pp. 758-770, ISBN: 978-3-540-26208-4, 2005

[VIV10a] Vivekanandan P. and Nedunchezhian D. R., *A New Incremental Genetic Algorithm Based Classification Model to Mine Data With Concept Drift*, Journal of Theoretical and Applied Information Technology, 2010, pp. 36-42

[VIV10b] Vivekanandan P. and Nedunchezhian D. R., *A Fast Genetic*

Algorithm for Mining Classification Rules in Large Datasets, International Journal on Soft Computing (IJSC), Vol.1, No.1, 2010, pp. 10-20

[WAN04] Wang Z. and Feng B., *Classification rule mining with an iproved ant colony algorithm*, Proceedings of the 17th Annual Australian Conference on Artificial Intelligence, 2004

[WIT05] Witten I.H. and Frank E., *Data Mining Practical Machine Learning Tools and Techniques, Second Edition,* Elsevier Inc., ISBN-13:978-0-12-088407-0, 2005

[WU07] Wu X., Kumar V., Quinlan R., Ghosh J., Yang Q., Motoda H., McLachlan G., Ng A., Liu B., Yu P., Zhou Z., Steinbach M., Hand D. Steinberg D. *Top 10 algorithms in data mining,* Knowledge and Information Systems, Volume 14, pp. 1-37, ISSN:0219-1377, 2007

[www1] *Data mining,* http://wordnetweb.princeton.edu/perl/webwn?s=data%20mining , Accessed: 02-06-2012

[www2] *Data mining,* http://emeld.org/school/glossary.html#data, Accessed: 28-06-2012

[www3] *Web Mining,* http://brodholt.com/WebMining.htm, Accessed: 03-05-2011

[www4] *Attribute-Relation File Format*, http://www.cs.waikato.ac.nz/~ml/weka/arff.html, Accessed: 22-06-2012

[www5] *Machime Learning with Weka and R,* http://videolectures.net/bootcamp07_belanche_mldm, Accessed: 03-04-2011

[www6] Connecting *Weka* to databases, Available from: http://weka.wikispaces.com/Windows+Databases, Accessed: 18-03-2010

[www7] *DataFit,* http://www.oakdaleengr.com/, Accessed: 28-06-2012

[YAN06] Yang Y. F., Lohmann P. and Heipke C., *Genetic Algorithms for the Unsupervised Classification of Satellite Images*, IntArchPhRS. Band XXXVI/3. Bonn, 2006, S. 179-184

[ZHA03] Zhang Y.Q., Shteynberg M., Prasad S.K. and Sunderraman R. *Granular fuzzy web intelligence techniques for profitable data mining*. In 12th IEEE Int. Conf. on Fuzzy Systems, FUZZ '03 (New York:IEEE Comput. Soc.) pp 1462–1464, 2003

[ZHA05] Zhang C., Yang Q. and Liu B., *Guest Editors' Introduction: Special Section on Intelligent Data Preparation*, IEEE Transactions On Knowledge And Data Engineering, Vol. 17, No. 9, pp. 1163-1165, 2005

[ZLH07] Zaharie D., Lungeanu D. and Holban S., "Feature ranking based on weights **estimated** by multiobjective optimization", *Proceedings of IADIS First European Conference on Data Mining*, pp.124-128, 2007

Lista figurilor

Lista tabelelor