UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAŞI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Abordarea clasificării prin metoda programării genetice

propusă de
Constantin-Sebastian Stanciu

Sesiunea: *iulie, 2019*
Coordonator ştiinţific
Conf. dr. Mihaela Breabăn

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAŞI
FACULTATEA DE INFORMATICĂ

Abordarea clasificării prin metoda programării genetice

Constantin-Sebastian Stanciu

Sesiunea: *iulie, 2019*

Coordonator ştiinţific
Conf. dr. Mihaela Breabăn

Avizat,
Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele

_____

Data _____ Semnătura _____

DECLARAŢIE privind originalitatea conţinutului lucrării de licenţă

Subsemnatul(a) ……..
domiciliul în ……
născut(ă) la data de ……..…., identificat prin CNP ……..,
,absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de ……. specializarea ……,
promoţia ……., declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 şi 5 referitoare la plagiat, că
lucrarea de licenţă cu titlul:

_____

_____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice în vederea facilitării falsificării de către cumpărător a calităţii de autor al unei lucrări de licenţă,de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… Semnătură student ……

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul „Abordarea clasificării prin metoda programării genetice”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iaşi, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent Constantin Sebastian Stanciu

_____

CUPRINS

Introducere

Învăţarea automată (Machine Learning) are multiple aplicaţii, una dintre cele mai importante fiind extragerea de cunoştinţe din date (data mining). Aceasta este de doua tipuri: învăţare supervizată şi învăţare nesupervizata. Diferenţa dintre acestea este faptul ca în cazul primului tip de învăţare setul de date conţine pentru fiecare set de atribute şi o etichetare a acestora care reprezintă variabila ce trebuie prezisă, care lipseşte în schimb în cazul celui de-al doilea tip.

Oamenii sunt pasibili sa facă greşeli în analiza sau în încercarea de a stabili relaţii intre atributele dintr-un set de date, fapt ce îngreunează rezolvarea unor probleme, în schimb algoritmii de învăţare automata elimină eroarea umana şi sunt rapizi, datorită puterii computaţionale din ce în ce mai mari.

O mulţime de aplicaţii de învăţare automată pot include sarcini de învăţare supervizată, acest domeniu fiind şi tema pe care este orientată această lucrare, în special spre clasificare, realizată cu ajutorul programării genetice, care nu este o metodă clasica folosită în această problema, urmărindu-se dacă această metodă poate fi o soluţie viabila în anumite cazuri de clasificare. Aşadar scopul acestei lucrări de licenţă este studiul programării genetice ca şi metodă de clasificare.

Clasificarea este o problemă de actualitate, cu o vastă importanţă în domeniul informatic şi nu numai. Aşa cum este sugerat şi de nume, clasificarea reprezintă identificarea unui obiect sau fenomen necunoscut ca membru al unei clase de obiecte sau fenomene, cu ajutorul unor informaţii anterioare disponibile. O caracteristică principală a clasificării este faptul ca algoritmul de rezolvare selectează dintr-un set deja definit de soluţii, pe baza atributelor care îl definesc, neîncercând găsirea unor noi soluţii. O problema care poate apărea în acest caz este faptul ca uneori regulile de clasificare, care sunt învăţate pe baza setului de date de antrenament nu sunt potrivite pentru clasificarea anumitor noi obiecte, ceea ce poate duce la clasificări greşite. Acest lucru poate avea consecinţe mari, de exemplu în domeniul medical, un algoritm de detectare a unei boli clasifica un pacient drept sănătos, când defapt el este bolnav. Aceasta clasificare poate avea un cost mare, fiindcă se întârzie detectarea bolii şi începerea tratamentului. În aceasta lucrare accentul va fi pus pe clasificarea binara, elementele putând face parte din doua clase de obiecte. Eficienţa unui algoritm de clasificare fiind măsurată cu ajutorul a patru concepte: *true positives*, *true negatives*, *false positives* şi *false negatives*. Dintre acestea o deosebita importanţă le au ultimele doua, deoarece suma acestora reprezintă numărul de clasificări greşite ale algoritmului. În numeroase cazuri de clasificare exista un “trade off” intre acestea, deoarece costurile lor nu sunt aceleaşi, precum în cazul enunţat anterior, o clasificare false positive este mult mai puţin costisitoare decât una false negative.

Această problemă a clasificării binare are multiple aplicaţii în probleme din viaţa reală, precum în domeniul medical, în detectarea fraudelor, a e-mailurilor spam sau în cazul detectării produselor contaminate.

Principala temă a acestei lucrări este de a studia eficienţa programării genetice în clasificarea binara, pentru un set de date de antrenament care este dificil şi performanţele algoritmilor clasici de clasificare nu oferă rezultate suficient de bune. De asemenea se vor executa un număr ridicat de experimente cu scopul studierii performanţei pentru metoda clasificării cu programare genetică.Lucrarea este structurată în patru capitole, fiecare având sub-capitole.

În primul capitol intitulat "Problema detecţiei de evenimente în monitorizarea calităţii apei" se va prezenta mai în detaliu în ce consta problema clasificării, cum poate fi afectata performanţa clasificatorilor de către structura setului de date de antrenament şi explicarea conceptului de set de date dezechilibrat. De asemenea, va fi prezentat setul de date de antrenament care va fi folosit şi structura acestuia.

În al doilea capitol "Algoritmi de clasificare" sunt prezentaţi o parte dintre cei mai cunoscuţi şi folosiţi algoritmi pentru clasificare.

În al treilea capitol este prezentat conceptul de programare genetică, importanţa funcţiei fitness, modul în care va fi abordata problema clasificării cu ajutorul programării genetice şi funcţiile de fitness ce au fost folosite în experimentele realizate.

În al patrulea capitol numit în mod sugestiv "Analiză experimentală" sunt analizate şi comparate performanţele clasificatorilor clasici şi a celor obţinuţi prin programare genetică. De asemenea este prezentată şi construirea unui ansamblu de clasificatori obţinuţi prin programare genetică şi performanţele acestuia.

1.Problema detecţiei de evenimente în monitorizarea calităţii apei

1.1 Învăţarea supervizată

Învăţarea supervizată este reprezentată de algoritmi care reuşesc cu ajutorul unui set de date oferit să găsească ipoteze/modele generale prin care sa facă predicţii despre obiecte noi, sa facă mapări de la setul de date input la output. Cu alte cuvinte se urmăreşte crearea unui clasificator care sa asignează etichete unor noi instanţe despre care se ştie doar setul de atribute, dar nu şi eticheta acestora.

Abordările existente în activitatea denumită generic *mineritul datelor (Data Mining)* se împarte în doua tipuri de învăţare: supervizata şi nesupervizata. Ambele se bazează pe un set de date, însa în cazul învăţării supervizate setul de date include şi eticheta (outputul corespunzător) (Tabelul 1), iar în cazul învăţării nesupervizate acesta lipseşte. Învăţarea supervizata se împarte în continuare în doua categorii: clasificare şi regresie, diferenţă dintre acestea fiind faptul ca în cazul clasificării se prezice o etichetă, aşadar se urmăreşte crearea unei mapări de la variabilele de input la o variabila categorică, discretă de output, iar în cazul regresiei maparea se face de la variabilele de input la o variabila de output numerică, continuă. În cazul utilizării algoritmilor de clusterizare, numiţi şi învăţare nesupervizata se urmăreşte descoperirea unor clase de obiecte neştiute, dar care pot fi importante, care pot surprinde anumite legături, similarităţi, conexiuni intre obiectele respective.

	Atribut 1	Atribut 2	...	Atribut n	Eticheta
1	k	0		xy	1
2	r	1		xx	0
3	s	0		xx	2
...					
m	a	1		xy	6

Tabelul 1.
Instanţe cu etichetă cunoscută.

Există o expansiune continuă a acestui domeniu, care are o importanţă deosebită în multe domenii complexe şi variate, de la probleme de zi cu zi pană la probleme de securitate

naţională precum armată, medicină, finanţe. Multe dintre aceste probleme au date nebalansate, adică clasele din setul de date nu sunt reprezentate în mod egal sau cel puţin aproximativ în mod egal. De exemplu în cazul unei probleme de clasificare binară, adică setul de date are doua etichete dacă dintr-un număr total de 1000 de instanţe, 800 reprezintă clasa întâi, iar 200 reprezintă clasa a doua, setul de date este considerat nebalansat, din cauza diferenţei mari de instanţe reprezentative pentru cele doua clase, raportul dintre ele fiind 800:200 sau mai simplu 4:1.

În cazul clasificării binare, în cadrul a numeroase probleme clasele de elemente sunt 0 şi 1, clasa 0 fiind denumită clasa negativă, iar clasa 1 clasa pozitivă. În cadrul acestei lucrări se vor folosi de asemenea aceste notaţii şi denumiri, deoarece şi în cazul setului de date folosit ca instrument de studiu în această lucrare clasa 0 defineşte absenţa unui eveniment, iar clasa 1 prezenţa acestuia, clasa 0 defineşte instanţele de apa necontaminată iar clasa 1 pe cele de apa contaminată. Precum am precizat în introducere patru concepte vor fi de folos în evaluarea performanţei unui algoritm de clasificare binara: true positive (se prezice în mod corect clasa pozitivă), true negative (se prezice în mod corect clasa negativă), false positive (se prezice în mod greşit clasa pozitivă), false negative (se prezice în mod greşit clasa negativă). Aceste concepte se vor abrevia în această lucrare astfel: true positives ->TP, true negative ->TN, false positive ->FP, false negative ->FN.

1.2 Problematica datelor dezechilibrate

În multe situaţii una dintre clase este slab reprezentată, adică numărul de exemple aparţinând acesteia este mult mai mic decât numărul de exemple din cealaltă clasă (celelalte clase) - aceasta este denumită clasa minoritară. În astfel de situaţii poate apărea paradoxul acurateţii, în sensul ca acurateţea poate fi foarte mare deoarece algoritmul învaţă foarte bine sa clasifice clasa majoritara, în exemplul dat anterior o acurateţe de 80% poate părea destul de buna, dar este posibil ca algoritmul sa clasifice toate instanţele ca făcând parte din prima clasa, fiind incapabil sa clasifice obiectele care ar trebui sa facă parte din clasa minoritară.Acurateţea este definită după următoarea formulă:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

Rata erorii este definita drept raportul dintre clasificările greşite şi numărul total de clasificări, conform formulei:

$$RataErorii = \frac{FP + FN}{TP + TN + FP + FN}$$

Alte doua metrici foarte importante în analiza performanţei unui algoritm sunt: *precizia* (*precision* în limba engleza) şi *reamintirea* (denumita în engleza *recall*). În această lucrare vor fi folosiţi în continuare termenii din limba engleza. Acestea sunt definite astfel:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Cu ajutorul preciziei şi a recallului se calculează scorul F1. Această metrică combină aceste doua concepte având valoarea maxima 1 şi cea minima 0. Scorul F1 este în general folosit când numărul *TN* nu are o importanţă ridicată şi se doreşte ca modelul sa nu conţină multe FP sau FN. Din aceste motive această metrica este potrivită în cazul seturilor de date neechilibrate şi va fi folosită şi în cadrul acestei lucrări pentru a compara algoritmii. O valoare mai ridicată indică o performanţa mai buna a algoritmului, formula pentru scorul F1 fiind:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Un alt concept este matricea de confuzie (confusion matrix), care conţine în cazul clasificării binare doua linii şi doua coloane, iar în cazul unei set de date cu n clase conţine n linii şi n coloane. Valoarea de pe poziţia (i,j) reprezintă numărul de instanţe din clasa i, care au fost clasificate ca făcând parte din clasa j, de exemplu în figura 1, elementul de pe poziţia (0,0), reprezintă numărul de instanţe care fac parte din clasa negativă şi care au fost clasificate în mod corect.

	Negativ	Pozitiv
Negativ	TN	FP
Pozitiv	FN	TP

Figura 1.
Matrice de confuzie pentru clasificare binară

În cazul clasificării binare, matricea de confuzie ajuta pentru vizualizarea numărului total de clasificări corecte şi a celor greşite astfel: suma elementelor de pe diagonala principală reprezintă clasificările corecte, iar cel de pe diagonala secundara clasificările greşite.

Această problema are un interes major fapt ce se ilustrează prin numeroasele conferinţe, articole, al căror număr creste la o rata ridicată, devenind obiectivul principal al acestei lucrări în care se urmăreşte eficienţa unei metode mai puţin clasice pentru o astfel de problema, programarea genetică. În acest capitol va fi prezentată problema învăţării din seturi de date nebalansate în profunzime.

Teoretic orice set de date în care distribuţia claselor nu este egală poate fi considerat dezechilibrat, însa de obicei o diferenţă mica nu influenţează eficienţa clasificatorilor într-un mod care poate fi considerat semnificativ. De obicei printr-un set de date nebalansat se înţelege o diferenţă majora, fiind posibile reprezentări de tipul 1:100, 1:500 sau 1:1000, în fiecare dintre acestea o clasa are mult mai multe instanţe decât cealaltă(Figura 2). Deşi din definiţia anterioara s-ar putea înţelege că se face referire la problemele de clasificare binară, acest lucru se poate întâmpla şi în cazul unor seturi de date care conţin 3 sau mai multe clase, dezechilibrate având loc intre anumite clase din setul de date(Figura 3).

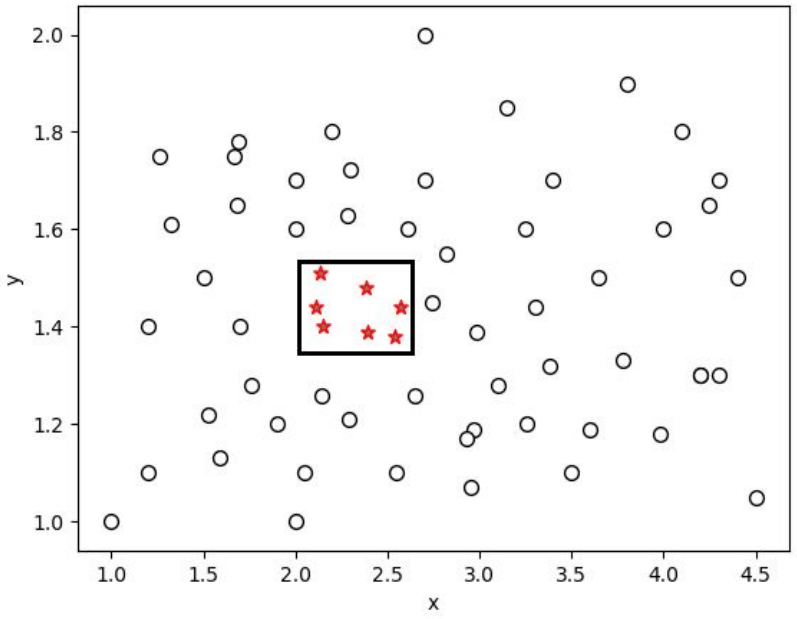

Figura 2.
Set de date în care raportul dintre cele doua clase este dezechilibrat, raportul fiind de 1:8.

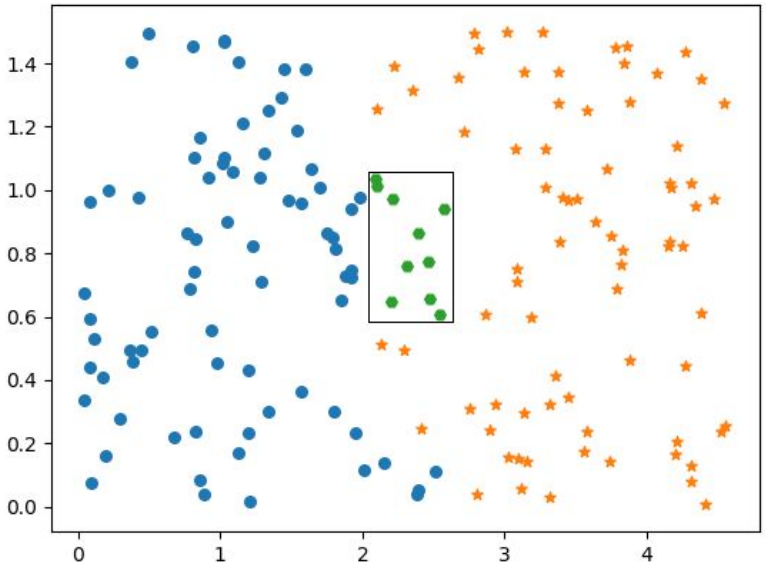

Figura 3.
Set de date dezechilibrat cu mai multe clase.Raportul dintre clasa minoritară şi clasele majoritare este 1:5

Dezechilibrele pot fi intrinseci, în cazul în care dezechilibrul este o consecinţa directă a naturii setului de date, de exemplu un set de date cu ajutorul căruia se încearcă clasificarea pacienţilor drept bolnavi de cancer sau sănătoşi, este normal ca acesta sa conţină un număr mai mare de instanţe al persoanelor sănătoase, decât al celor bolnave. Exista şi dezechilibre care nu ţin de natura setului de date sau a problemei care se încearcă a se rezolva, ci de factori precum timp sau puterea de stocare, acestea se numesc extrinseci. Dezechilibrele extrinseci sunt la fel de interesante ca şi cele intrinseci, acestea putând apărea chiar şi în cazul unor seturi de date ce nu sunt neapărat iniţial dezechilibrate, de exemplu dacă un set de date este obţinut treptat iar pe parcursul transmisiei apar întreruperi, în care datele nu sunt trimise, este posibil sa obţinem un set nebalansat extrinsec, obţinut dintr-un set de date balansat.

De asemenea, alte două concepte importante sunt dezechilibrul relativ şi dezechilibrul cauzat de instanţe rare, numit şi raritate absoluta. Un exemplu pentru dezechilibru relativ este un set de date binar care conţine 50.000 de exemple şi un raport intre clase de 50:1, astfel clasa minoritara având 1000 de instanţe. Dacă se mai obţin date, iar setul de date se dublează, iar raportul dintre clase rămâne acelaşi, clasa minoritara va conţine 2000 de instanţe, un număr destul de consistent de instanţe, dar scăzut relativ la clasa majoritara. Acest timp de dezechilibru apare în numeroase aplicaţii din viata reala şi va fi studiata în această lucrare. Dezechilibrul cauzat de instanţele rare apare în domenii în care clasa minoritară are număr limitat de exemple reprezentative, de aceea învăţarea este dificilă indiferent dacă există dezechilibru intre clase sau nu.

1.3 Detectarea anomaliilor pentru calitatea apei potabile

În studiul acestei probleme am folosit un set de date binar dezechilibrat. Am preluat aceste date din cadrul GECCO("Genetic and Evolutionary Computation Conference"), care a organizat în anul 2018 competiţia "Internet of Things: Online Anomaly Detection for Drinking Water Quality".

Setul de date este oferit de o companie de îmbuteliere a apei(Thüringer Fernwasserversorgung), fiind preluat din viata reală şi reprezintă măsurători alea parametrilor realizate de către această companie, făcându-se astfel o legătura directă cu industria. Datele au fost obţinute prin trecerea apei prin anumite zone speciale, cu senzori, unde sunt măsuraţi cei mai importanţi indicatori de calitate ai apei. Setul de date reprezintă de fapt, o serie de timp, instanţele fiind măsurători ale acestor atribute o data pe minut.

Setul de date are următoarele atribute: Time (timpul când a fost făcuta măsurătoarea), Tp (temperatura în grade celsius), Cl (cantitatea de dioxid de clor în mg/L), pH (valoarea PH-ului), Redox (potenţialul de reducere), Leit (conductibilitatea electrica a apei), Trueb (turbiditatea), Cl_2 (cantitatea de dioxid de clor la a doua măsurătoare), Fm (debitul în conducta 1), Fm_2 (debitul în conducta 2) şi EVENT (eticheta, variabilă booleana ce specifica dacă apa este contaminată).

Scopul acestei competiţii este realizarea unui clasificator, care sa detecteze schimbările de calitate ale apei, contaminarea acesteia. Acesta sarcina are o importanţă deosebită pentru companiile de apa publica, pentru cetăţeni, dar nu numai, apa fiind vitală pentru orice forma de viaţă. Un astfel de clasificator este esenţial în menţinerea şi distribuţia apei potabile pe întreaga planetă.

Acest set de date prezintă dezechilibru intrinsec, fiind evident ca pe parcursul acestor măsurători instanţele de apa contaminată sa fie mult mai slab reprezentate, decât cele ale apei potabile şi este de asemenea un dezechilibru relativ, clasa minoritară având un număr de 1726, nefiind slab reprezentată decât relativ la clasa majoritara, care conţine 137.840 de instanţe.

Datele sunt foarte dezechilibrate, raportul dintre instanţele clasei minoritare şi cele ale clasei majoritare fiind de aproximativ 1:80 (Figura 4).

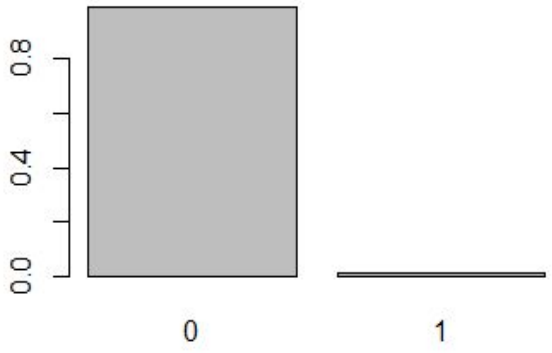

Figura 4.
Vizualizare a raportului dintre instanţele celor doua clase

1.4 Metode de sampling

Acesta metode constau în modificarea setului de date prin diferite mecanisme cu scopul de a micşora dezechilibrul dintre clase sau chiar de a echilibra instanţele ce reprezintă clasele din setul de date, cu scopul de a îmbunătăţi performanţele clasificatorilor.

Există numeroase astfel de metode, dar în cadrul acestei prezentări vor fi menţionate doar o parte dintre acestea. În primul rând există doua concepte fundamentale pentru aceste metode: oversampling, care consta în a adăuga în setul de date instanţe din clasele minoritare, şi undersampling, care constă în ştergerea unor instanţe din setul de date, care aparţin claselor majoritare.

Cel mai simplu mod de realizare a acestor doua concepte este în mod aleatoriu. În cazul oversampling-ului, se selectează în mod aleatoriu un subset de dimensiune dorită din clasa minoritară şi apoi se adaugă în setul de date. În cazul undersampling-ului, se selectează în mod aleatoriu un subset din clasa majoritară, ulterior instanţele din acest subset sub şterse din setul de date. În mod evident prin aceste metode se micşorează dezechilibrul din setul de date, însa pot cauza o serie de probleme. În cazul undersampling-ului se elimină instanţe din setul de date, fapt ce poate cauza pierderea unor informaţii, concepte importante pentru clasificarea corecta a instanţelor din clasa majoritara, iar în cazul oversampling-ului se adaugă în setul de date instanţe replicate, ceea ce poate duce la supra specializare.

O metodă mult mai interesantă este metoda SMOTE, ce a fost aplicată cu succes în multiple aplicaţii, performanţele acesteia fiind studiate şi menţionate în numeroase articole de specialitate. Această metodă creează în mod artificial instanţe din clasa minoritară cu ajutorul

celor deja prezente în setul de date cu algoritmului KNN, prin găsirea celor mai apropiaţi k vecini pentru instanţele din clasa minoritară. Apoi se sintetizează o noua instanţă intre instantă din clasa minoritară şi unul dintre vecinii sai, precum este ilustrat în figura 5

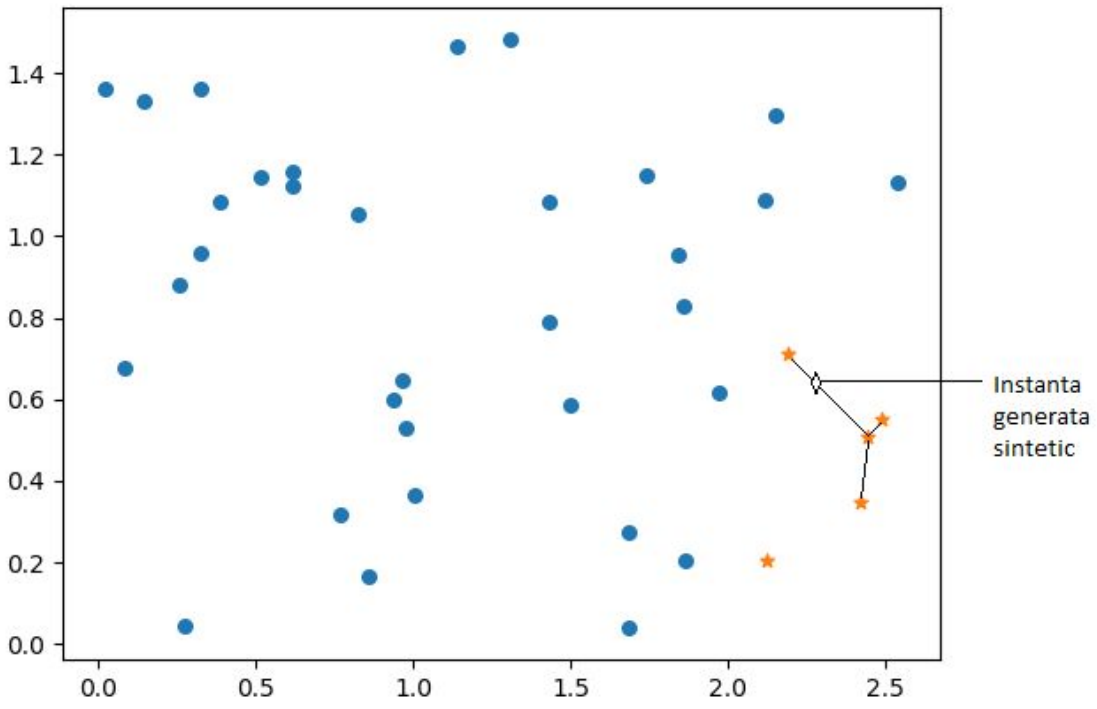

Figura 5.
Generarea unei instanţe în mod sintetic cu SMOTE

1.5 Învăţare bazată pe cost

Spre deosebire de metodele de sampling care modifică setul de date pentru a creste performanţele clasificatorilor, învăţarea bazată pe cost constă în asocierea unor costuri diferite pentru clasificările greşite, în funcţie de clasa din care face parte instanţa respectivă. Acest concept este o alternativă viabilă pentru metodele de sampling în încercarea de a îmbunătăţi performanţele algoritmului de clasificare.

Modul în care funcţionează acest concept este prin atribuirea unor costuri diferite pentru clasificările greşite, de exemplu în cazul unei probleme de clasificare binara costul clasificării greşite a unei instanţe din clasa majoritara va fi mai mic decât costul unei clasificări greşite din clasa minoritară şi se încearcă minimizarea costului pentru întreg setul de antrenament. În general nu se atribuie niciun cost pentru clasificările corecte.

Matricea de cost este un concept important în acest tip de învăţare, fiind o modalitate de a vizualiza costurile, care sunt furnizate fie de către experţi în domeniu sau stabilite prin diferite metode. În cadrul experimentelor din această lucrare costul unei clasificări greşite din clasa majoritară va fi 1, iar costul unei clasificări greşite din clasa minoritară va fi raportul dintre numărul de instanţe din clasa majoritară şi numărul de instanţe din clasa minoritară. În matricea de cost pe poziţia (i,j) se găseşte costul clasificării unei instanţe din clasa i în clasa j, şi în acest caz arata astfel:

	Negativ	Pozitiv
Negativ	0	1
Pozitiv	80	0

Figura 6.
Matrice de costuri

2. Algoritmi de clasificare

În acest capitol al acestei lucrării sunt prezentate o parte dintre cele mai cunoscute şi folosite metode de clasificare, o parte dintre acestea au fost studiate în cadrul materiei Învăţare automata din primul semestru al anului III, car vor fi prezentate şi metode care nu au fost studiate în cadrul facultăţii.

În primul rând aceşti algoritmi au câteva caracteristici comune, specifice pentru a rezolva problema clasificării, precum: se bazează pe un set de date care conţine instanţe de tipul atribute-valoare, cu scopul de a clasifica obiecte noi despre care se cunosc doar atributele, de asemenea atributele valoare din setul de date sunt discrete, adică care au un număr finit de valori spre deosebire de variabilele continue din cazul regresiei, celalalt tip de învăţare supervizată.

2.1 Arbori de decizie

Arborii de decizie sunt un tip de clasificator ce se reprezintă drept arbori prin sortarea atributelor în funcţie de valoarea acestora. Fiecare nod din arbore reprezintă un atribut, iar fiecare muchie conţine o valoare pe care nodul respectiv o poate avea. Clasificarea unei instanţe noi se realizează prin parcurgerea arborelui de la rădăcina la frunza care corespunde acestei parcurgeri şi care conţine eticheta clasei.

Această metodă se bazează pe împărţirea setului de antrenament în funcţie de valoarea atributelor, practic se considera ca nodul rădăcina realizează acest lucru. Apoi aceeaşi procedura se realizează pe fiecare partiţie din setul de date, obţinută prin împărţirile anterioare. Exista mai multe metode pentru o împărţire cat mai eficientă a setului de date, de exemplu câştigul de informaţie (Hunt et al., 1966) şi indexul gini (Breiman et al., 1984).

Termenul de câştig de informaţie este strâns legat de cel al entropiei, care este o măsura a incertitudinii, de aceea entropia în cazul aruncării unei monezi este maxima, probabilitatea de a cădea cap sau pajura fiind egale. Entropia este calculată în felul următor:

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

Valoarea entropiei este maximală atunci când exista o dezordine totală, în sensul ca toate probabilităţile sunt egale.Valoarea entropiei este calculata în acest caz astfel:

$$H = -n \cdot \frac{1}{n} \log_2 \frac{1}{n} = \log_2 n$$

Pentru calculul entropiei se face următoarea convenţie, necesara în cazul în care una dintre probabilităţi este egala cu 0:

$$0 \cdot \log_2 0 = 0$$

În cazul arborilor de decizie în care se foloseşte câştigul de informaţie se încearcă maximizarea acestuia, iar atributul cu cel mai mare câştig de informaţie va fi selectat pentru a partiţiona setul de date. Câştigul de informaţie este calculat cu ajutorul entropie şi reprezintă cat de multă informaţie oferă un atribut despre clasa din care face parte, de aceea un atribut care împarte setul de date în mod perfect oferă un câştig de informaţie maximal.

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

În cazul arborilor ID3 este folosit câştigul de informaţie, iar în cazul arborilor CART se foloseşte indexul gini, care indică cat de mixate sunt clasele în grupurile create de o împărţire, o tăietură a acestuia. Indexul Gini este calculat în cazul unui set de date care conţine n clase astfel:

$$gini(X) = 1 - \sum_{i=1}^{n} p_i^2$$

Indexul gini are valoarea 0 în cazul unei împărţiri perfecte şi valoarea maxima în cazul în care rezulta clase cu număr egal de instanţe, calculându-se astfel:

$$gini(X) = 1 - k(\frac{1}{k^2}) = 1 - \frac{1}{k}$$

2.2 AdaBoost

Boostingul (stimularea) este o metodă bazata pe ideea creării unui predictor puternic format din mai multe reguli mai slabe şi cu acurateţe mai slaba. Algoritmul AdaBoost, prescurtare de la "Adaptive Boosting" este o un algoritm bazat pe conceptul boosting care a dovedit o eficienţă crescută, fiind unul dintre cei mai populari algoritmi, cu aplicări în multiple domenii. Algoritmii de tip boosting crează predictorul puternic cu ajutorul unei mulţimi de clasificatori mai slabi în mod secvenţial.

Modul în care AdaBoost funcţionează este prin asignarea iniţială a unui cost de $1/n$ tuturor instanţelor din setul de antrenament. De asemenea, după ce clasificatorul a fost antrenat i se asignează o pondere, în funcţie de acurateţea sa, iar costul instanţelor clasificate greşit creste cu scopul clasificării corecte a acestora în iteraţia următoare. Acest proces continuă pana când se atinge o acurateţe îndeajuns de bună sau pană când se atinge un număr maxim de clasificatori. Clasificarea unui noi instanţe se realizează prin votul tuturor clasificatorilor, ţinând cont de ponderea acestora.
Algoritmul AdaBoost se bazează pe câteva formule matematice. Formula următoare reprezintă eticheta asignată de clasificator unui instanţe x:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

În formula aceasta h_t reprezintă eticheta asignată de clasificatorul slab t instanţei x, iar α_t este ponderea asignată clasificatorului t, având valoare negativă în cazul în care acurateţea este mai mica decât 0.5 şi pozitivă în caz contrar. Formula de calcul este următoarea, *E* fiind numărul clasificărilor greşite:

$$\alpha_t = 0.5 * ln(\frac{1-E}{E})$$

Recalcularea ponderii unei instanţe se realizează conform următoarei formule:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

D_t(i) reprezintă ponderea de la nivelul anterior, iar Z_t reprezintă suma tuturor ponderilor, cu scopul de a normaliza ponderile.

2.3. Regresie logistică

Regresia logistică este un algoritm de clasificare chiar dacă conţine cuvântul regresie în denumire, care ar putea dezorienta, indicând spre celalalt tip de învăţare supervizata. Aşadar regresia logistică caută un model care sa estimeze probabilitatea unei variabile dependente cu ajutorul unor variabile independente.

În cazul setului de date folosit în această lucrare variabila dependenta este EVENT, iar variabilele independente sunt restul variabilelor prezentate.Cu ajutorul probabilităţii obţinute se reuşeşte clasificarea unor noi instanţe, prin compararea acesteia cu un prag ales, în cazul în care probabilitatea este mai mare decât pragul instanta este clasificata pozitiv, iar în caz contrar este clasificata în mod negativ.

În cazul problemei abordate în această lucrare nu putem folosi regresia liniara deoarece aceasta presupune prezicerea unei unei variabile dependente cu ajutorul unei alte variabile independente, de asemenea nici regresia multipla care este asemănătoare cu cea multipla, dar cu mai multe variabile independente, aceste doua metode necesită un set de date cu distribuţie normala. Distribuţia normala este cel mai important tip de distribuţie din cadrul statisticii deoarece este reprezentativa pentru numeroase fenomene naturale, precum înălţimea, greutatea sau scorul IQ.

Abordarea problemei cu astfel de metode nu este posibila şi din cauza faptului ca acestea ar putea prezice valori pentru variabila dependenta EVENT care sa depăşească intervalul [0,1], fapt de ar încălca definiţia probabilităţii. În cazul regresiei logistice se obţine următoarea formula, care nu presupune problema aceasta, prin găsirea unei unei funcţii care asignează mulţimii de variabile independente o valoare care reprezintă probabilitatea de încadrare în valorile variabilei dependente.

$$0 \leq h_\theta(x) \leqslant 1$$

În formula data, $h_\theta(x)$ are următoarea formă:

$$h_\theta(x) = g(\theta^T x)$$

$$g(t) = \frac{1}{1 + e^{-t}}$$

Din aceste două relaţii se obţine în mod trivial următoarea relaţie, care reprezintă probabilitatea estimata ca pentru datele x clasificarea corecta sa fie clasa 1 (clasa pozitivă):

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^t x}}$$

Conform definiţiei anterioare se calculează probabilitatea apartenenţei la clasa pozitivă, iar fiind o problemă de clasificare binară, probabilitatea clasei negative este diferenţa dintre 1 şi această valoare. Pentru clasificarea unor date cu ajutorul modelului creat este necesară stabilirea unui prag care ia în mod evident valori din intervalul (0,1), care dacă este depăşit de către valoarea calculată $h_\Theta(x)$ datele sunt clasificare ca făcând parte din clasa pozitivă sau negativă în caz contrar.

Un rol important în funcţionarea buna a acestui algoritm o au parametrii Θ, iar aceştia sunt calculaţi cu ajutorul unei funcţii de cost. Scopul acestui procedeu este de a găsi parametrii Θ care minimizează această funcţie şi care vor fi ulterior folosiţi în cadrul clasificării. Costul clasificării unei instanţe din setul de date de antrenament este:

$$Cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

Funcţia de cost pentru regresia logistica în cazul unui set de antrenament cu m instanţe este suma costurilor fiecărei instanţe împărţita la numărul de instanţe:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i))]$$

2.4. Extra trees

Extra trees este o alta metodă de clasificare de tip ansamblu, precum algoritmul AdaBoost, dar spre deosebire de acesta elementele ansamblului funcţionează în mod paralel. această denumire provine de la prescurtarea numelui iniţial “Extremely randomized trees”. Acest algoritm funcţionează prin crearea mediei mai multor arbori care sunt creaţi prin partiţionarea setului de date în mod aleator.

Această abordare s-a dovedit foarte eficientă, crescând precizia şi având o complexitate computaţionala scăzută, evitându-se şi supra-adaptarea (overfitting), care se datorează unei adaptări prea mari la setul de date de antrenament, în sensul ca ipoteza este influenţata şi de date irelevante sau de zgomot, ceea ce cauzează o dificultate în clasificarea unor instanţe noi, din afara setului de antrenament.

Ideea algoritmului este asemănătoare ca în cazul clasificării cu ajutorul algoritmului Random Forest, dar exista şi anumite diferenţe, una dintre cele mai importante fiind faptul ca în cazul Extra trees punctul de taiere în cazul proprietăţilor selectate este ales în mod random, spre deosebire de Random Forest în cazul căruia se caută cel optim. Din acest motiv

algoritmul Extra trees este mult mai rapid din punct de vedere computațional, iar din punct de vedere al performantei algoritmii sunt asemănători.

Spre deosebire de metoda arborilor de decizie prezentaţi anterior în cadrul acestei lucrări, Extra trees propune un ansamblu de arbori, nu doar un arbore, de aceea se reduce supra-adaptarea şi de aceea în cele mai multe cazuri această metodă funcţionează mai bine pentru clasificarea unor noi instanţe.

La fel ca în cazul arborilor de decizie calitatea împărţirii setului de date de către un atribut este calculata cu ajutorul câştigului de informaţie sau a indexului gini. Aceste metode au fost prezentate în cadrul acestui capitol în cadrul prezentării arborilor de decizie şi nu vor mai fi reluate.

3. Programare genetică

3.1 Concepte şi algoritm

Programarea genetică reprezintă obiectul de studiu principal al acestei lucrări. Acest concept este împrumutat din natura în sensul că structurile biologice care reuşesc o adaptare buna supravieţuiesc şi se reproduc la o scară mai ridicată decât cele ce sunt inadaptate, conform teoriei evoluţioniste a lui Darwin, prin selecţie naturală. Adaptabilitatea unui individ poate fi măsurata prin conceptul de fitness.

Programarea genetică este o tehnică ce face parte din grupul algoritmilor evolutivi, din care fac parte şi algoritmii genetici, ce are scopul de a rezolva probleme în mod automat indiferent de domeniul acestora, fără a fi necesar sa se specifice forma sau dimensiunea soluţiei, ci doar în ce constă rezolvarea problemei, de exemplu în cazul unei probleme de optimizare, găsirea unei valori de maxim sau de minim.

Pentru o prezentare cat mai completă a programării genetice mai sunt necesare prezentarea anumitor concepte. Rezolvarea problemelor pe calculator se efectuează cu ajutorul algoritmilor, pentru care nu exista o definiţie standard, definiţia din dicţionarul Cambridge, considerat o sursă cu grad mare de încredere este: “Un set de instrucţiuni matematice care trebuie urmat într-o ordine fixată şi care cu ajutorul unui calculator va ajuta la calcularea răspunsului pentru o problemă matematică”. Exista doua strategii cu ajutorul cărora se rezolvă probleme cu ajutorul calculatorului, în mod determinist şi în mod probabilist.

Algoritmii determinişti sunt de doua tipuri: algoritmi exacţi şi euristici. Caracteristica comuna a acestora este ca pentru un anumit input produc mereu acelaşi output, urmând mereu aceeaşi secvenţă de stări. În cazul algoritmilor determinişti exacţi se obţin rezultate exacte, dar nu pot fi mereu folosiţi în practică în timp util, în special în cazul problemelor ce fac parte din clasa NP-complete, pentru care nu se cunoaşte încă un algoritm determinist exact care sa le rezolve în timp polinomial. Euristicile sunt algoritmi care produc soluţii aproximative, care nu sunt de cele mai multe ori optime, dar sunt suficient de bune în cazul în care se doreşte o rezolvare într-un timp limitat pentru o problema dificila precum cele NP-complete. Exemple de probleme ce fac parte din categoria NP-complete sunt: problema comisului voiajor, problema colorării unui graf, problema rucsacului.

Algoritmii probabilişti reprezintă o metodă de rezolvare a problemelor prin utilizarea noţiunii “random”, adică la cel puţin un pas din cadrul algoritmului decizia se realizează în

mod aleatoriu. Din acest motiv nu se poate prezice valoarea soluţiei ce se va obţine, la rulări succesive este posibil ca rezultatele pentru un acelaşi input sa difere, reieşind astfel ca soluţiile nu pot fi exacte, ci sunt aproximative.

Modelul tradiţional de rezolvare al problemelor se numeşte “hard computing” şi are trăsături precum faptul ca se bazează pe principii ca precizie şi rigoare şi nu are posibilitate de adaptare (exemple : calculul integral, calculul diferenţial). Aceste metode nu pot fi folosite în multe probleme din viata reala în care apar incertitudini, inexactităţi sau ambiguităţi, ci doar în probleme relativ simple.

O tehnică mai modernă sunt metodele “soft computing” care au posibilitatea de autoadaptare. Consider ca cel mai intuitiv mod de a prezenta acest principiu este prin antiteză cu conceptul prezentat anterior “hard computing”, care este descris prin rigiditate, inflexibilitate. Tehnica soft computing este tolerantă la incertitudini şi aproximări, imitând creaturile naturale flexibile şi adaptive. Această paradigmă se bazează pe un feedback intern cu ajutorul căruia proprietăţile individuale ale problemei sunt exploatate la maxim.

Programarea genetică face parte din categoria calculului evolutiv, ce este o sub-ramura a Inteligenţei Artificiale, ce cuprinde paradigme soft computing. Aşadar programarea genetică aparţine paradigmei soft-computing, fiind algoritmi probabilişti inspiraţi din biologie şi natură. În cadrul paradigmei programării genetice structurile supuse adaptării sunt programe pe calculator de dimensiune şi forme variabile. Încercarea de a rezolva o problema cu ajutorul programării genetice reprezintă practic o căutare a unui program pe calculator cu un fitness foarte bun în spaţiul programelor posibile. Această paradigma oferă posibilitatea căutării celui mai adaptat program pe calculator pentru problema ce se încearcă sa se rezolve, prin anumite metode inspirate din natura şi teoria evoluţionistă a lui Charles Darwin.

Execuţia programării genetice începe cu o populaţie iniţiala de programe generate random, care poate fi considerata o căutare oarba în spaţiul programelor pe calculator. Programele generate sunt compuse din funcţii şi terminali aleşi în funcţie de domeniul problemei.

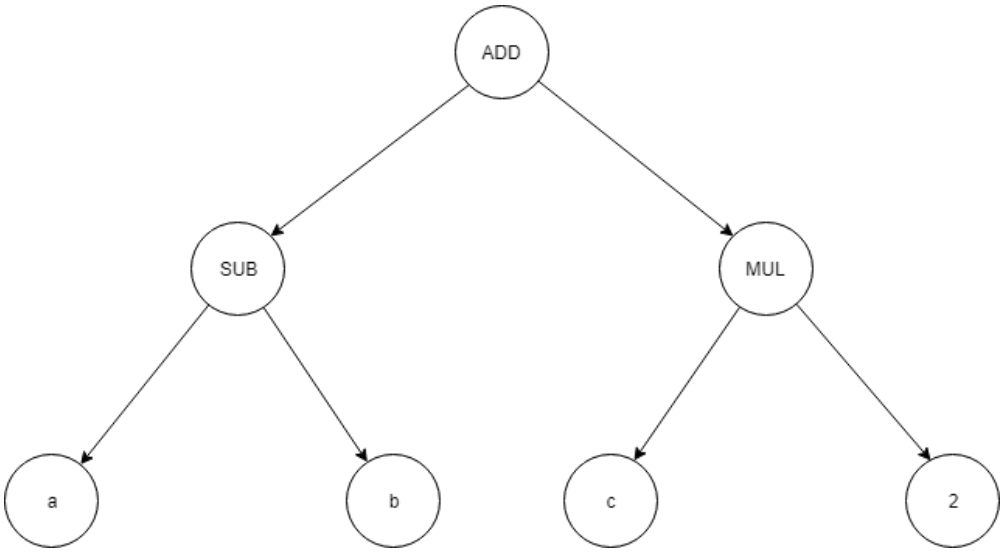

Figura 7. Reprezentarea unei soluţii candidat în Programarea genetică

În cadrul programării genetice programele sunt reprezentate sub forma de arbori. În arborele din Figura 7 este reprezentat programul ADD(SUB(a,b),MUL(C,2)). Frunzele arborelui sunt terminalii, iar nodurile interne sunt denumite primitive. Terminalii sunt împărţiţi în doua subtipuri, constante şi argumente, în exemplul anterior există o singura constantă(numărul 2) şi trei argumente(a,b,c). Constantele îşi păstrează valoarea pe întreaga execuţie, iar argumentele sunt datele de intrare ale programului.

Funcţia fitness reprezint un termen important în terminologia programării genetice, fiind modul în care se testează performanţa programelor din setul de programe. Cu ajutorul funcţiei fitness se pot determina cele mai bune programe, informaţie care este folosită ulterior pentru modificarea şi îmbunătăţirea acestora. Astfel se poate uşor intui faptul ca structura programelor pe calculator care reies din cadrul paradigmei programării genetice sunt o consecinţa a funcţiei de fitness, care reprezintă şi în cadrul natural factorul dominant al evoluţiei.

Programarea genetică dispune de doi operatori genetici de baza, încrucişarea şi mutaţia, cu ajutorul cărora se realizează căutarea în spaţiul posibilelor soluţii. Mutaţia este o operaţie care este controlată de un parametru probabilistic şi se realizează pe un singur individ din populaţie. În urma mutaţiei se obţine un nou individ prin modificarea structurii părintelui. Un exemplu simplu de mutaţie este următorul:

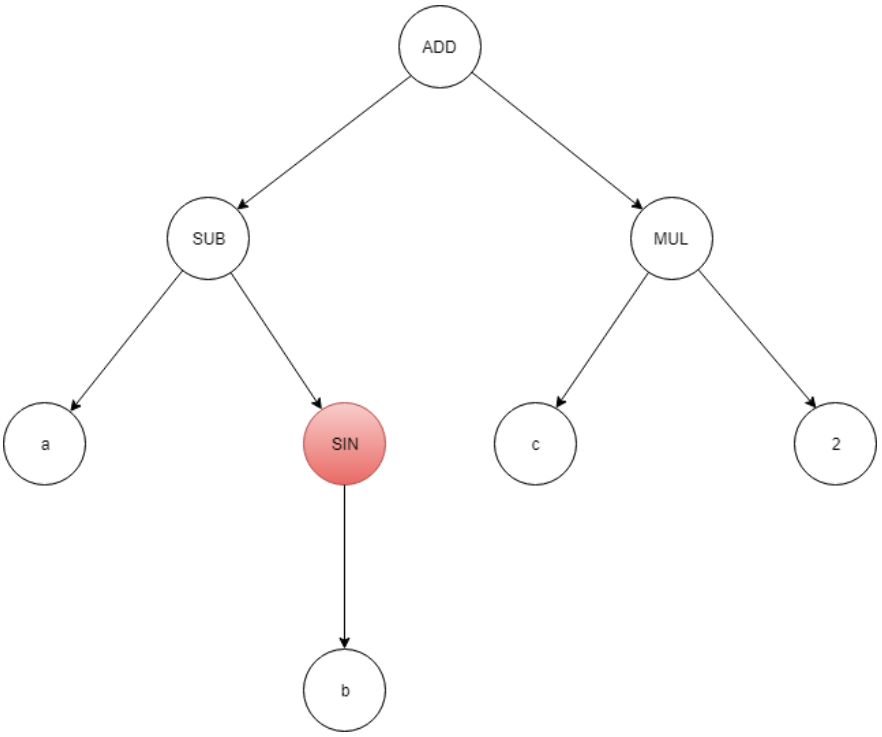

Figura 8.Individ înainte de realizarea mutaţiei

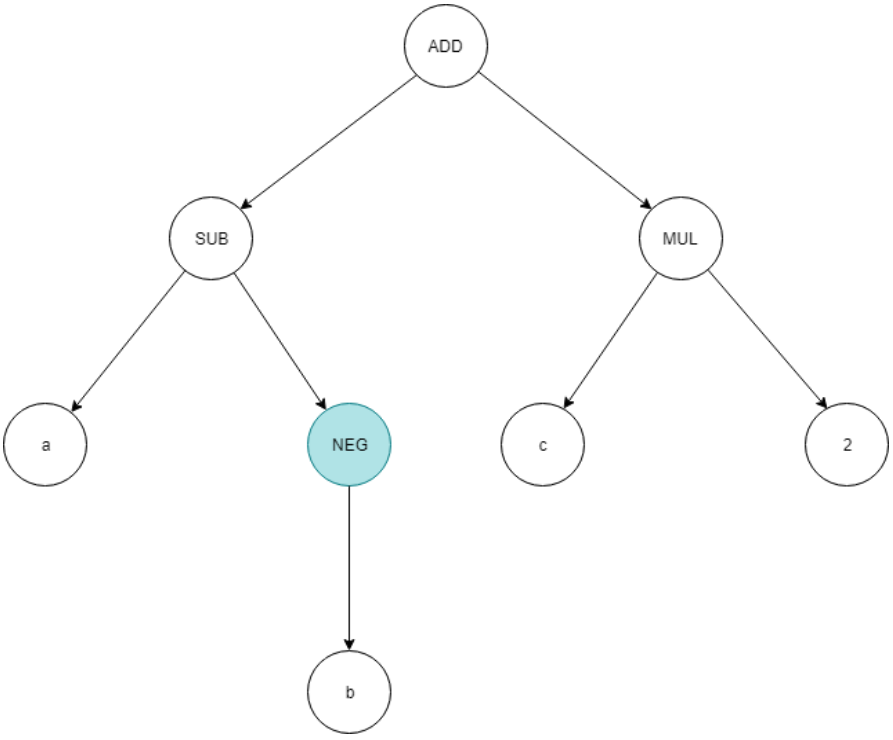

Figura 9.Individ după realizarea mutaţiei

În exemplul dat se observă ca mutaţia a constant în înlocuirea primitivei SIN cu cea NEG, astfel pornind de la structura ADD(SUB(a,SIN(b)),MUL(c,2)) obţinem prin mutaţie structura ADD(SUB(a,NEG(b)),MUL(c,2)).

Încrucişarea se realizează cu ajutorul a doua programe şi se obţin doua noi programe, care sunt în general diferite intre ele şi diferite faţă de părintii lor. Operaţia începe prin selectarea în mod random în cadrul fiecărui părinte a unui punct de încrucişare, astfel se obţin doua fragmente de încrucişare reprezentate de sub-arborii ce au drept rădăcina punctul de încrucişare ales anterior. Primul copil se produce prin ştergerea fragmentului de încrucişare din cadrul primului părinte şi adăugarea în locul acestuia a fragmentului de încrucişare din al doilea părinte, al doilea copil se produce în mod simetric.

În exemplul următor se ilustrează modul în care încrucişarea funcţionează. Nodurile colorate în galben vor fi considerate punctele de încrucişare. Părinţii au următoarele expresii: ADD(NOT(x),OR(y,z)) şi OR(AND(x,OR(x,y)),NOT(z), iar copii rezultaţi au expresiile AND(NOT(x),AND(x,OR(x.y)) şi OR(OR(y,z),NOT(z)).

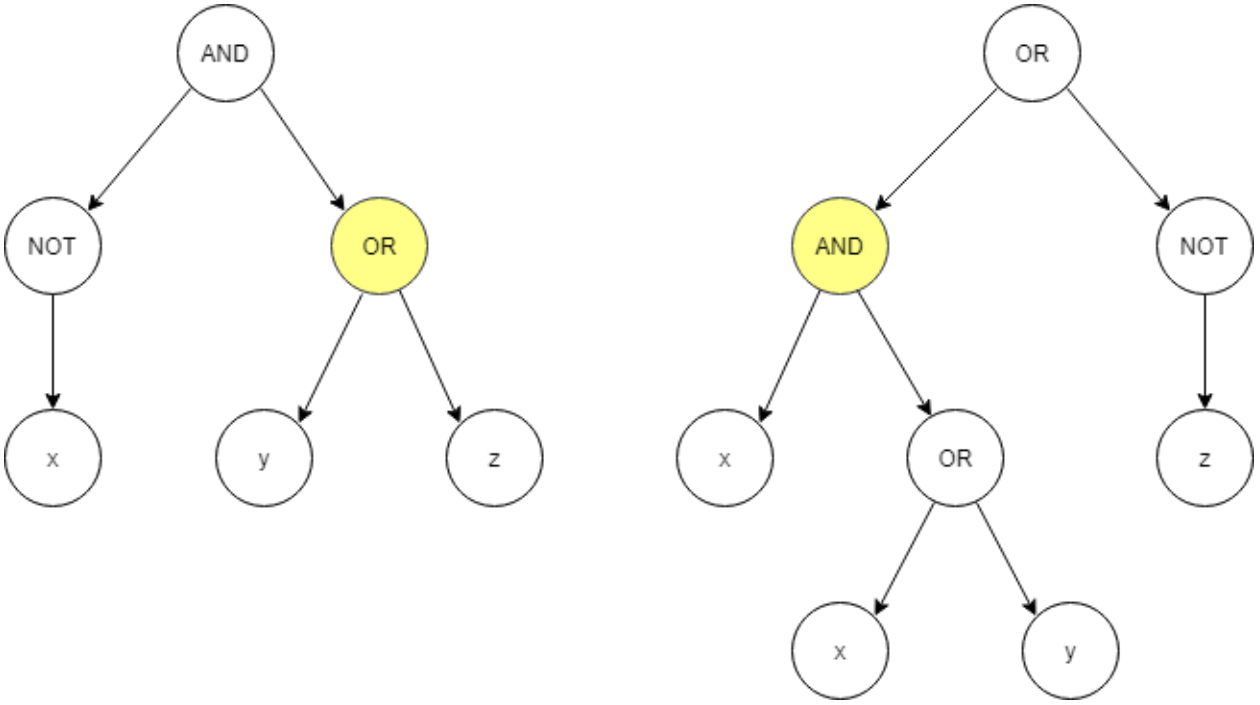

Figura 10.
Alegerea punctelor de încrucişare pentru doi indivizi

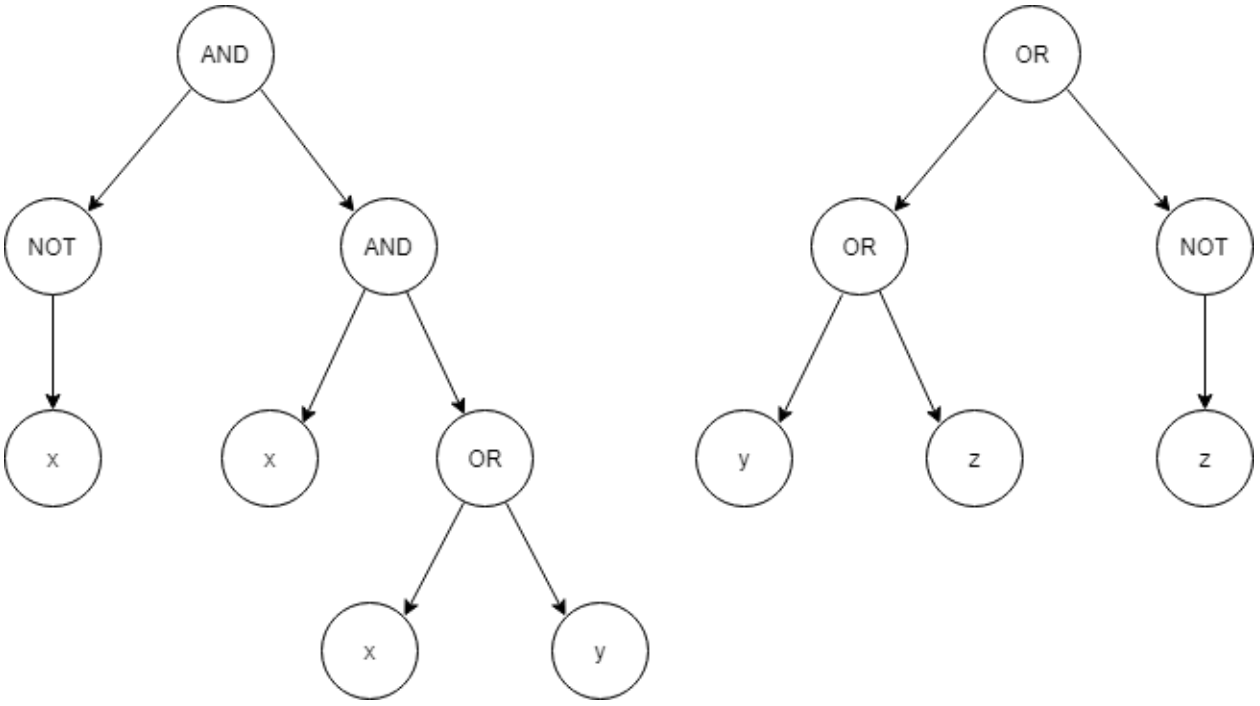

Figura 11.
Indivizii rezultaţi în urma încrucişării

3.2. Programare genetică pentru clasificare binară

În această lucrare vom studia dacă programarea genetică poate fi folosită cu succes în cazul clasificării binare. Modul în care acest studiu este realizat este prin căutarea unor expresii matematice. Argumentele problemei vor fi datele din setul de antrenament. Pentru fiecare expresie din setul de expresii se va parcurge întreg setul de date şi se va verifica dacă pentru acele date clasificarea realizată este corecta sau nu.

Modul în care este abordata această problema este prin găsirea unei probabilităţi în funcţie de rezultatul dat de expresia matematica găsită. Acest lucru este realizat prin următoarea formulă matematică:

$$predicţie(x) = \frac{1}{1 + e^x}$$

Argumentul funcţiei este chiar rezultatul expresiei matematice. Cu ajutorul acestei funcţii se obţine o valoare între 0 şi 1. Valoarea returnată va fi comparată cu un prag şi în funcţie de rezultatul comparaţiei se va face clasificarea. Dacă predicţia este mai mare sau egala decât valoarea pragului datelor vor fi clasificate ca făcând din clasa pozitivă, iar în caz contrar din clasa negativă.

Graficul funcţiei este ilustrat în figura 12. Se poate observa ca pentru un prag cu valoarea de 0.5 rezultatul expresiei trebuie sa aibă valoare mai mica sau egala cu 0 pentru ca datele sa fie clasificate în clasa pozitivă şi valoare mai mare de 0.5 pentru ca asignarea sa fie în clasa negativă.

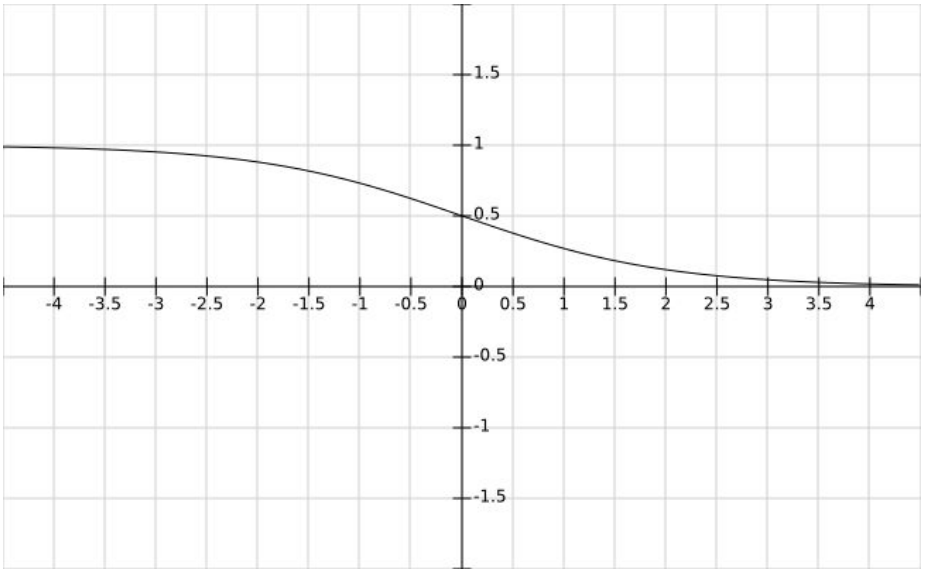

Figura 12.
Graficul funcţiei utilizate în clasificarea binară

3.3 Funcţii fitness pentru clasificare

Pentru găsirea unor expresii care sa fie folosite pentru clasificare prin metoda programării genetice este necesară definirea unei funcţii de fitness, cu ajutorul căreia sa se calculeze cat de bună este expresia în rezolvarea problemei de clasificare propusă, structura acestora fiind o consecinţă a funcţiei de fitness.

În studierea performantei programării genetice din această lucrare vor fi încercate mai multe funcţii de fitness.Prima metodă încercata este cross-entropia, definita astfel:

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

În formula data *p(x)* reprezintă probabilitatea vrută şi *q(x)* probabilitatea adevărată. În cazul acestei funcţii de fitness scopul va fi minimizarea fitness-ului.

O alta funcţie de fitness este numărarea numărului de erori. De asemenea şi în acest caz se va urmări scăderea fitness-ului.

Metrica de măsurare a performantei unui clasificator din această lucrare va fi scorul F1, aşadar o alta funcţie de fitness folosită va fi chiar calcularea scorului F1 obţinut pe setul de antrenament, aşadar scopul va fi maximizarea fitness-ului.

Una dintre metodele care au fost folosite cu succes pentru îmbunatăţirea performanţei unui clasificator în cazul învăţării dintr-un set de date dezechilibrat este învăţarea bazata pe cost. O alta funcţie de fitness este construită cu ajutorul acestui concept, calculându-se numărul de erori, însa costul clasificărilor greşite nu va mai acelaşi. Costurile au fost stabilite cu ajutorul raportului dintre instanţele clasei minoritare şi a celei majoritare(1:80). Costul unei clasificări greşite a unei instanţe din clasa majoritare va avea cost 1, iar costul unei clasificări greşite a unei instanţe din clasa minoritară va fi 80.

4. Analiză experimentală

În acest capitol sunt analizate rezultatele obţinute prin metodele de învăţare automată prezentate şi prin metoda programării genetice, astfel se determină eficienţa celei din urmă. Acest lucru este realizat cu ajutorul unui set de date de test, din cadrul aceleaşi competiţii ("Internet of Things: Online Anomaly Detection for Drinking Water Quality"), care a fost folosit pentru a verifica performanţele metodelor trimise de către concurenţi.

Astfel încât comparaţia dintre programarea genetică şi metodele de învăţare automată sa fie cat mai obiectivă şi relevantă în cazul unor metode de învăţare automata a fost realizata optimizarea hiper-parametrilor, adică se caută parametri pentru aceste metode astfel încât rezultatele sa fie cat mai bune. Un exemplu de hiper-parametru este adâncimea maxima a unui arbore de decizie, aşadar hiper-parametrii sunt parametrii ai metodelor, ce sunt setaţi înainte de începerea propriu zisa a procesului de învăţare.

O metodă foarte folosită pentru analiza performantei unui model de învăţare automată este K-fold Cross Validation, care consta în împărţirea setului de date de antrenament în k subseturi, dintre care k-1 sunt folosite pentru antrenament şi un subset este folosit pentru testare, procesul se repetă până când fiecare dintre subseturi a fost folosit pentru testare. Analiza performanţei se va realiza prin media rezultatelor obţinute. În figura 13 este ilustrat 3-fold Cross Validation, sub-seturile colorate în culoarea portocaliu fiind setul de test pentru iteraţia respectiva.

Figura 13.
Vizualizare a 3-fold Cross Validation

Căutarea grid se realizează prin definirea unui set de valori posibile pentru parametrii algoritmilor. Se realizează toate asignările posibile ale parametrilor cu aceste valori şi se calculează performanţa cu ajutorul Cross Validation. Se găseşte astfel cea mai buna combinaţie de parametri şi se îmbunătăţeşte performanţa algoritmului. Acest tip de căutare este folosit pentru găsirea hiper-parametrilor împreuna cu 10-fold Cross Validation, setul de date fiind împărţit în 10 sub-seturi, conform principiilor prezentate anterior.

Pentru a compara performanţele algoritmilor se va folosi scorul F1, care a fost şi criteriul oficial de departajare în cadrul competiţiei, de unde au fost preluate setul de date de antrenament şi cel de test. această metrică este potrivită deoarece nu se ia în considerare numărul clasificărilor corecte pentru elementele din clasa negativă, dar este sensibilă la clasificările greşite. Un scor F1 ridicat indica faptul ca instanţele din clasa pozitivă sunt identificate într-o proporţie ridicată în mod corect, fără a produce un număr ridicat de alarme false.

4.1 Arbori de decizie

Pentru arborii de decizie s-a realizat optimizarea hiper-parametrilor. Parametrul pentru evaluarea unei taieturi a fost ales criteriul Gini, parametrul înălţime maxima nu este limitata la o anumită valoare, iar argumentul min_samples_split, care reprezintă numărul minim de instanţe conţinut de către o frunza a fost setat la numărul 10.

Rezultatele clasificării se regăsesc în următoarea matrice de confuzie:

	Negativ	Pozitiv
Negativ	104937	7820
Pozitiv	784	1545

Cu ajutorul acestei matrice putem observa şi calcula cu uşurinţa următoarele aspecte ce ilustrează performanţa clasificatorului:

$$TN = 104937 \qquad TP = 1545$$

$$FN = 784 \qquad FP = 7820$$

$$Precision = \frac{1545}{1545 + 7820} = 0.16497$$

$$Precision = \frac{1545}{1545 + 784} = 0.66337$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = 0.26423$$

4.2 Regresie logistică

În cazul acestei metode de clasificare nu a fost cazul găsirii şi setării unor hiper-parametri. Rezultatele obţinute de acest clasificator sunt prezentate în următoarea matrice de confuzie:

	Negativ	Pozitiv
Negativ	112757	0
Pozitiv	1918	411

O prezentare mai în detaliu a performanţei acestui clasificator sunt următoarele rezultate.

$$TN = 112757 \qquad TP = 411$$

$$FN = 1918 \qquad FP = 0$$

$$Precision = \frac{411}{411 + 0} = 1$$

$$Recall = \frac{411}{411 + 1918} = 0.17647$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = 0.3$$

4.3 AdaBoost

Pentru această metodă de clasificare a fost identificat hiper-parametrul ce reprezintă numărul maxim de estimatori. Valoarea acestuia a fost setata la 50.Matricea de confuzie corespunzătoare acestui clasificator este:

	Negativ	Pozitiv
Negativ	109245	3512
Pozitiv	1351	978

Cu ajutorul matricei obţinem cu uşurinţă următoarele rezultate:

$$TN = 109245 \qquad TP = 978$$

$$FN = 1351 \qquad FP = 3512$$

$$Precision = \frac{978}{978 + 3512} = 0.21781$$

$$Recall = \frac{978}{978 + 1351} = 0.41992$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = 0.28683$$

4.4 Extra trees

În cazul acestei metode de clasificare au fost identificaţi următorii hyper-parametri: numărul de arbori a fost setat la valoarea de 40, criteriul de evaluare a unei tăieturi este indexul Gini, înălţimea nu este limitata la o valoare maxima, iar argumentul min_samples_split, care reprezintă numărul minim de instanţe conţinute de către o frunza a fost setat la numărul 10.

Matricea de confuzie corespunzătoare acestui clasificator este:

	Negativ	Pozitiv
Negativ	112616	141
Pozitiv	1636	693

Rezultatele pentru acest clasificator sunt:

$$TN = 112616 \qquad TP = 693$$

$$FN = 1636 \qquad FP = 141$$

$$Precision = \frac{693}{693 + 141} = 0.83093$$

$$Recall = \frac{693}{693 + 1636} = 0.29755$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = 0.43818$$

4.5 Programare genetică

Pentru programarea genetică a fost definit următorul set de primitive: minim, maxim, adunare, scădere, înmulţire, împărţire, negaţie, sinus şi cosinus. Cu ajutorul acestor primitive se vor crea expresii matematice care iau ca date de intrare valorile din setul de date. În urma experimentelor a fost aleasa o funcţie de fitness ce reprezenta numărul de clasificări greşite din setul de antrenament, iar în acest caz în mod evident se urmăreşte minimizarea fitness-ului.

O serie de alţi parametri cu importanţă deosebita au fost stabiliţi astfel: dimensiunea populaţiei are valoarea de 200, numărul de iteraţii realizate pentru căutarea expresiilor este 55 şi modalitatea de selecţie ale expresiilor este de tip turneu. Acest mod de selecţie este realizat prin alegerea în mod aleatoriu a unui număr k de indivizi din populaţie şi returnarea celui cu cel mai bun fitness. Numărul de indivizi pentru selecţie a fost stabilit la 25. În cazul mutaţiei punctul de mutaţie poate fi înlocuit cu un arbore produs în mod aleator, ce are înălţime cuprinsa intre 0 şi 3, iar probabilitatea de mutaţie a unui individ este 0.2. Încrucişarea a doi

indivizi este prin alegerea în mod aleator a cate unui punct de încrucişare pentru fiecare dintre ei, în modul prezentat vizual în capitolul anterior. Probabilitatea de încrucişare a doi indivizi a fost stabilita la valoarea de 0.5 .

Pentru aceşti parametri au fost obţinute rezultatele din tabelul următor, care conţine 30 de expresii. Este important de specificat ca pentru obţinerea acestuia au fost necesare mai mult de 30 de iteraţii, deoarece nu au fost incluse rezultatele care aveau probabilitate mare de overfitting, adică au fost eliminate soluţiile în cadrul cărora execuţia programului a înregistrat pentru un număr relativ mare de iteraţii scăderi mici ale valorii funcţiei fitness, fapt ce poate indica o adaptare a expresiei pentru zgomotul şi detaliile din setul de date, ceea ce afectează calitatea clasificatorului pentru date noi.

Expresia:	Fitness	Scorul F1
mul(sub(pH, add(neg(mul(Fm, Cl_2)), add(cos(Redox), sub(Fm, mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, sub(Cl, Cl_2))), mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), pH)), -1), Redox)), Cl_2))), pH)), -1), Redox))), pH)), Cl_2))), pH)), -1), Redox))), add(Redox, sub(Fm, mul(neg(pH), neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), add(neg(mul(Fm, sub(Cl, Cl_2))), mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), pH)), -1), Cl)), Cl_2))), pH)), -1), Redox))), pH))))), -1), Redox)), mul(Fm, mul(Cl_2, Cl_2))))))))))))), Redox)	1101	0.26544
mul(add(mul(add(mul(min(safeDiv(add(Redox, Fm), sub(Fm_2, add(mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), sub(Fm, Redox)), cos(max(Cl, pH))), sin(pH)))), mul(max(mul(Cl, pH), safeDiv(Fm, Redox)), sub(Fm, Redox))), Redox), min(sub(neg(cos(Redox)), Redox), mul(add(neg(pH), sub(Fm, Redox)), cos(max(Cl, pH))))), cos(max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sin(cos(Cl)))))), Cl), safeDiv(sub(sub(Fm, Redox), min(sub(Fm, Redox), Cl)), Cl_2))	1132	0.3811
mul(add(mul(add(mul(min(safeDiv(add(Redox, Fm), sub(Fm_2, add(mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), sub(Fm, Redox)), cos(max(Cl, pH))), sin(pH)))), mul(max(mul(Cl, pH), safeDiv(Fm, Redox)), sub(Fm, Redox))), Redox), min(sub(sub(sub(Fm, Redox), min(sub(Fm, Redox), Cl)), Redox), mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), max(Leit, Fm)), cos(max(Cl, pH))))), cos(max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sub(Fm, Redox))))), Cl), safeDiv(sub(sub(Fm, Redox), min(sub(Fm, Redox), Cl)), Cl_2))	1138	0.37574
safeDiv(max(safeDiv(min(Cl_2, Cl_2), Fm), cos(max(Cl_2, pH))), mul(neg(cos(sub(pH, min(Leit, Cl_2)))), safeDiv(sub(Fm, Redox), min(add(cos(max(Cl_2, pH)), sin(sin(pH))), Fm))))	1189	0.40356
mul(sub(safeDiv(add(cos(pH), Leit), add(Fm_2, Leit)), Cl_2), sub(mul(sin(Cl_2), sub(sub(pH, Cl_2), sin(add(Cl_2, sub(sub(pH, Cl_2), sin(add(sin(sub(pH, sin(sub(sub(pH, Cl_2), cos(pH))))), Tp)))))), cos(sin(sin(sub(pH, sin(sub(sub(pH, Cl_2), cos(pH))))))))	1040	0.34673
mul(neg(sin(sin(neg(pH)))), sub(sub(mul(Fm, Trueb), safeDiv(Fm, pH)), mul(sin(add(sub(safeDiv(Trueb, cos(sub(Redox, neg(sin(Cl))))), mul(sin(pH), neg(pH))), Trueb)), add(add(Tp, neg(sub(mul(add(add(Leit, neg(sub(mul(Fm, Trueb), neg(sub(mul(Fm, Trueb), safeDiv(Fm_2, pH)))))), sub(Redox, Fm)), Trueb), safeDiv(Fm, pH)))), sub(Redox, Fm)))))	1114	0.21975
mul(cos(safeDiv(cos(sin(neg(safeDiv(mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(Redox, Fm))), Fm), sub(Leit, Fm))))), Fm), sub(sub(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(sub(sub(Tp, pH), pH), Cl_2)), sub(Cl_2, neg(pH))), Fm))))), cos(sub(sub(Cl_2, neg(pH)), Cl)))), add(sub(Redox, Fm), mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(Leit, Fm))))), Fm)))	1137	0.44187
mul(add(neg(neg(neg(sin(safeDiv(Fm, Fm_2))))), mul(Cl_2, add(safeDiv(mul(Redox, pH), mul(Leit, Tp)), min(sub(sub(sub(Fm, max(safeDiv(Tp, cos(Redox)), add(Tp, mul(pH, Cl_2)))), max(safeDiv(Fm, Fm_2), Cl)), Trueb), max(Tp, Cl))))), min(mul(max(add(Trueb, Cl), max(Leit, Redox)), Redox), safeDiv(Redox, add(sub(sub(Fm, Redox), safeDiv(mul(Leit, Tp), add(sub(sub(Fm, Redox), cos(Redox)), neg(sin(safeDiv(mul(safeDiv(Trueb, Fm_2), max(max(Tp, Cl), Redox)), Fm_2)))))), Tp))))	1190	0.14204

min(add(safeDiv(Cl, neg(add(Cl_2, Cl))), mul(Cl_2, pH)), mul(sub(Fm, Redox), neg(safeDiv(safeDiv(Cl, min(pH, Fm)), safeDiv(neg(min(cos(pH), add(pH, Fm))), cos(add(pH, pH))))))	1101	0.3624
mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), Redox))), pH), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), add(Cl, sub(Redox, max(safeDiv(safeDiv(Redox, pH), Cl), max(mul(mul(Cl_2, safeDiv(mul(safeDiv(mul(mul(Cl_2, safeDiv(mul(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Fm, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), Tp), Cl_2)), Cl), mul(safeDiv(mul(Cl, pH), Cl), safeDiv(max(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), mul(mul(Cl_2, Cl), pH)), pH), Cl)), Cl), mul(Fm, safeDiv(Cl_2, mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), Cl_2), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox))))))))	995	0.29119
neg(safeDiv(cos(mul(add(-1, mul(Cl, Trueb)), pH)), mul(Fm, mul(safeDiv(safeDiv(Fm, Cl_2), Cl_2), mul(safeDiv(safeDiv(Fm_2, Cl_2), Cl_2), Cl_2), add(mul(safeDiv(pH, mul(add(-1, mul(Cl, pH)), pH)), Tp), mul(safeDiv(Tp, mul(safeDiv(-1, mul(Cl, pH)), Tp)), Tp)))))))	1052	0.25608
safeDiv(neg(add(min(min(mul(Cl, Redox), neg(add(pH, add(safeDiv(cos(safeDiv(min(Trueb, Trueb), mul(pH, Redox))), cos(max(Redox, Tp))), neg(sub(Cl_2, Tp)))))), min(neg(max(safeDiv(max(Cl, max(Redox, Fm)), sub(mul(Fm, Cl_2), min(pH, Trueb))), min(cos(add(max(Cl, Leit), sin(Cl_2))), Cl))), Cl)), add(mul(Cl, Redox), sub(add(mul(Cl, Redox), sub(neg(add(add(Tp, Cl_2), safeDiv(Tp, sub(add(mul(Cl, Redox), neg(safeDiv(mul(Fm_2, Leit), safeDiv(Redox, Trueb)))), safeDiv(pH, Cl_2))))), safeDiv(pH, Cl_2))), safeDiv(pH, Cl_2))))), sub(add(mul(Cl, Redox), sub(neg(add(pH, add(safeDiv(sin(Leit), mul(pH, Leit)), neg(sub(Cl_2, pH))))), safeDiv(pH, Cl_2))), safeDiv(pH, Cl_2)))	954	0.16601
safeDiv(max(safeDiv(min(Cl_2, Cl_2), Fm), cos(max(Cl_2, pH))), mul(neg(cos(sub(pH, min(Leit, Cl_2)))), safeDiv(sub(Fm, Redox), min(add(cos(max(Cl_2, pH)), sin(sin(pH))), Fm))))	1189	0.40356
neg(cos(sub(min(Trueb, sub(min(safeDiv(Trueb, add(sub(Cl_2, min(pH, pH)), Fm)), sub(max(Redox, Fm), add(pH, Redox))), min(min(add(sub(min(safeDiv(Trueb, add(sub(Cl_2, min(pH, pH)), Fm)), sub(max(Redox, Fm), add(pH, Redox))), min(min(add(sub(Cl_2, pH), add(Cl, Leit)), safeDiv(safeDiv(sub(Trueb, Redox), sub(Tp, Trueb)), add(neg(pH), add(Cl, max(mul(neg(Redox), cos(Trueb)), safeDiv(max(pH, Leit), sub(Leit, Fm)))))), add(pH, Cl_2))), add(Cl, Leit)), safeDiv(safeDiv(sub(pH, Redox), sub(Tp, Trueb)), add(neg(pH), Trueb))), add(pH, Cl_2)))), min(Fm_2, min(min(add(add(neg(pH), add(Cl, Trueb)), add(Cl, Leit)), safeDiv(safeDiv(sub(Trueb, Redox), sub(Tp, Trueb)), add(neg(pH), Trueb))), add(pH, Cl_2))))))	1173	0.32532
mul(sub(pH, add(neg(mul(Fm, Cl_2)), add(cos(Redox), sub(Fm, mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, sub(Cl, Cl_2))), mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), pH)), -1), Redox)), Cl_2))), pH)), -1), Redox))), pH)), Cl_2))), pH)), -1), Redox))), add(Redox, sub(Fm, mul(neg(pH), neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), add(neg(mul(Fm, sub(Cl, Cl_2))), mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), pH)), -1), Cl)), Cl_2))), pH)), -1), Redox))), pH)))), -1), Redox)), mul(Fm, mul(Cl_2, Cl_2))))))))))))), Redox)	1101	0.26544
neg(safeDiv(sin(cos(mul(safeDiv(mul(Tp, Cl), neg(cos(pH))), Cl))), sub(safeDiv(pH, sub(sub(cos(cos(cos(neg(cos(add(mul(safeDiv(mul(Tp, Cl), pH), Cl), add(Tp, Tp)))))), neg(mul(safeDiv(mul(Tp, Cl), pH), mul(safeDiv(mul(Tp, mul(safeDiv(mul(Tp, Cl), neg(cos(pH))), Cl)), neg(cos(pH))), Cl)))), safeDiv(safeDiv(mul(safeDiv(mul(Tp, Cl), neg(cos(pH))), Cl), neg(cos(pH))), neg(cos(pH))))), mul(safeDiv(cos(cos(Trueb)), neg(cos(pH))), Cl))))	968	0.23182
mul(mul(mul(mul(Leit, Cl_2), sub(mul(mul(Leit, Cl_2), sub(mul(mul(sub(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), Redox), Cl_2), add(mul(sub(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), Redox), Cl_2), neg(mul(Leit, Cl_2)))), Fm)), Fm)), safeDiv(sub(Fm, Redox), sin(pH))), add(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), neg(safeDiv(sub(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), Redox), sin(pH)))))	969	0.4774
mul(mul(Cl_2, mul(Cl_2, safeDiv(mul(max(neg(cos(neg(pH))), Cl_2), sin(pH)), mul(mul(Cl_2, safeDiv(pH, sin(max(neg(cos(pH)), 0)))), mul(Cl_2, mul(Cl_2, mul(Cl_2, safeDiv(mul(neg(cos(add(Cl_2, add(pH, pH)))), pH), sin(max(mul(cos(add(Cl, Cl_2)), mul(Cl_2, mul(Cl_2, safeDiv(mul(max(neg(cos(neg(pH))), Cl_2), sin(pH)), mul(mul(Cl_2, safeDiv(mul(sin(neg(cos(add(pH, mul(add(Cl, Cl_2), sub(add(pH, mul(add(Cl, Cl_2), sub(mul(Cl_2, safeDiv(mul(sin(neg(cos(add(pH, pH)))), pH), sin(max(neg(cos(pH)), 0)))), neg(mul(neg(mul(Cl_2, mul(Cl_2, mul(Cl_2, safeDiv(mul(neg(cos(add(Cl_2, add(pH, pH)))), pH), sin(max(mul(cos(add(Cl, Cl_2)), sub(mul(Cl_2, safeDiv(mul(sin(neg(cos(add(pH, mul(add(Cl, mul(add(Cl, sin(max(neg(cos(pH)), 0))), sub(mul(Cl_2, safeDiv(mul(sin(neg(cos(add(pH, pH)))), pH), sin(max(neg(cos(pH)), 0)))), neg(mul(Cl, cos(add(pH, mul(Cl, pH)))))))), sub(mul(Cl_2, safeDiv(mul(sin(neg(cos(pH))), pH), sin(max(neg(cos(pH)), 0)))), neg(cos(add(pH, mul(Cl, pH))))))))), pH), sin(max(neg(cos(pH)), 0)))), neg(cos(add(pH, Cl_2)))))), 0)))))), cos(add(pH, mul(Cl, pH)))))))), neg(cos(add(pH, Cl_2))))))))), pH), sin(max(neg(cos(pH)), 0)))), mul(Cl_2, mul(Cl_2, mul(Cl_2, safeDiv(mul(neg(cos(add(Cl_2, add(pH, pH)))), pH), sin(max(mul(cos(add(Cl, Cl_2)), sub(mul(Cl_2, safeDiv(mul(sin(neg(cos(add(pH, mul(add(Cl, mul(add(Cl, sin(max(neg(cos(pH)), 0))), sub(mul(Cl_2, safeDiv(mul(sin(neg(cos(add(pH, pH)))), pH), sin(max(neg(cos(pH)), 0)))),	751	0.24343

neg(mul(Cl, cos(add(pH, mul(Cl, pH))))))), sub(mul(Cl_2, safeDiv(mul(sin(neg(cos(pH))), pH), sin(max(neg(cos(pH)), 0)))), neg(cos(add(pH, mul(Cl, pH))))))))))), pH), sin(max(neg(cos(pH)), 0)))), neg(cos(add(pH, Cl_2))))), 0))))))))))), 0))))))))))), add(safeDiv(sub(Fm, max(Tp, Redox)), pH), pH))		
mul(mul(Cl_2, safeDiv(add(safeDiv(sin(Trueb), safeDiv(sin(add(add(sin(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))), Cl_2), Cl_2)), add(neg(mul(sin(add(add(sin(safeDiv(sin(sin(sin(add(add(neg(safeDiv(sin(sin(safeDiv(sin(sin(pH)), Cl_2))), Cl_2)), Cl_2), Cl_2), Cl_2)))), Cl_2)), Cl_2), Cl_2), Cl_2)), Cl_2)), Cl_2), Cl_2)), mul(sin(sin(sin(safeDiv(sin(sin(sin(sin(sin(sin(safeDiv(sin(sin(pH)), sin(pH))))))))), Cl_2)))), Cl_2))), Redox))), Redox), safeDiv(sin(add(add(sin(safeDiv(sin(sin(pH)), Cl_2)), Cl_2), add(add(sin(sin(sin(sin(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))))))), Cl_2), Cl_2))), add(neg(mul(safeDiv(sin(sin(pH)), Cl_2), mul(safeDiv(sin(sin(pH)), Cl_2), safeDiv(add(sin(add(add(sin(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))), Cl_2), Cl_2)), Cl_2), Cl_2)))), Redox)))), sub(Fm, add(safeDiv(sin(Trueb), safeDiv(sin(sin(add(add(sin(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))), Cl_2), Cl_2))), add(neg(mul(Fm, safeDiv(sin(sin(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2)))))), Cl_2))), Redox))), Redox)))	859	0.32385
min(add(safeDiv(Cl, neg(add(Cl_2, Cl))), mul(Cl_2, pH)), mul(sub(Fm, Redox), neg(safeDiv(safeDiv(Cl, min(pH, Fm)), safeDiv(neg(min(cos(pH), add(pH, Fm))), cos(add(pH, pH)))))))	1101	0.3624
mul(add(mul(add(mul(min(safeDiv(add(Redox, Fm), sub(Fm_2, add(mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), sub(Fm, Redox)), cos(max(Cl, pH))), Cl))), mul(max(mul(Cl, pH), safeDiv(Fm, Redox)), min(Cl, Fm))), Redox), min(sub(Fm, Redox), Cl)), cos(max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sin(cos(Tp))))))), Cl), safeDiv(sub(sub(Fm, Redox), min(sub(Fm, Redox), max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sin(cos(Cl))))))), Cl_2))	1154	0.44996
mul(add(neg(neg(neg(sin(safeDiv(Fm, Fm_2))))), mul(Cl_2, add(safeDiv(mul(Redox, pH), mul(Leit, Tp)), min(sub(sub(sub(Fm, max(safeDiv(Tp, cos(Redox)), add(Tp, mul(pH, Cl_2))))), max(safeDiv(Fm, Fm_2), Cl)), Trueb), max(Tp, Cl))))), min(mul(max(add(Trueb, Cl), max(Leit, Redox)), Redox), safeDiv(Redox, add(sub(sub(Fm, Redox), safeDiv(mul(Leit, Tp), add(sub(sub(Fm, Redox), cos(Redox)), neg(sin(safeDiv(mul(safeDiv(Trueb, Fm_2), max(max(Tp, Cl), Redox)), Fm_2))))))), Tp))))	1190	0.14204
neg(min(max(cos(min(Fm_2, Fm)), max(add(add(pH, min(max(add(Cl_2, mul(pH, add(safeDiv(mul(Cl_2, Fm_2), Redox), mul(pH, Cl_2)))), min(safeDiv(mul(Redox, pH), neg(pH)), Cl)), add(add(mul(pH, Cl_2), Cl_2), Cl_2))), pH), max(cos(Cl_2), sin(Cl_2)))), cos(add(pH, min(max(add(min(max(Fm_2, max(add(safeDiv(Cl_2, Redox), pH), max(cos(Cl_2), sin(Cl_2)))), cos(add(pH, min(max(add(safeDiv(Cl_2, sin(Redox)), pH), add(pH, Cl_2)), add(add(pH, Cl_2), Cl_2))))), mul(pH, add(safeDiv(mul(Cl_2, Fm_2), Redox), mul(pH, Cl_2)))), min(Cl, Cl)), add(add(pH, Cl_2), Cl_2))))))	1005	0.40762
mul(add(mul(add(mul(min(safeDiv(add(Redox, Fm), sub(Fm_2, add(mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), sub(Fm, Redox)), cos(max(Cl, pH))), sin(pH)))), mul(max(mul(Cl, pH), safeDiv(Fm, Redox)), sub(Fm, Redox))), Redox), min(sub(sub(sub(Fm, Redox), min(sub(Fm, Redox), Cl)), Redox), mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), max(Leit, Fm)), cos(max(Cl, pH))))), cos(max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sub(Fm, Redox))))), Cl), safeDiv(sub(sub(Fm, Redox), min(sub(Fm, Redox), Cl)), Cl_2))	1138	0.37574
neg(safeDiv(cos(mul(add(-1, mul(Cl, Trueb)), pH)), mul(Fm, mul(safeDiv(safeDiv(Fm, Cl_2), Cl_2), mul(safeDiv(safeDiv(Fm_2, Cl_2), Cl_2), add(mul(safeDiv(pH, mul(add(-1, mul(Cl, pH)), pH)), Tp), mul(safeDiv(Tp, mul(safeDiv(-1, mul(Cl, pH)), Tp)), Tp)))))))	1052	0.25608
mul(add(mul(add(mul(min(safeDiv(add(Redox, Fm), sub(Fm_2, add(mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), sub(Fm, Redox)), cos(max(Cl, pH))), sin(pH)))), mul(max(mul(Cl, pH), safeDiv(Fm, Redox)), sub(Fm, Redox))), Redox), min(sub(neg(cos(Redox)), Redox), mul(add(neg(pH), sub(Fm, Redox)), cos(max(Cl, pH))))), cos(max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sin(cos(Cl))))))), Cl), safeDiv(sub(sub(Fm, Redox), min(sub(Fm, Redox), Cl)), Cl_2))	1132	0.3811
min(sin(safeDiv(max(pH, Trueb), safeDiv(sin(safeDiv(Cl, Cl_2)), cos(Cl)))), min(safeDiv(safeDiv(Cl, sin(safeDiv(sin(add(pH, Cl_2)), Cl_2))), sin(min(Fm, pH))), sub(safeDiv(Trueb, cos(add(pH, min(safeDiv(Cl, Cl_2), sub(safeDiv(sin(safeDiv(Cl, Cl_2)), cos(add(pH, Cl_2))), safeDiv(max(pH, Trueb), safeDiv(sub(safeDiv(sin(add(min(mul(Cl_2, Tp), safeDiv(Fm, Fm_2)), cos(add(Redox, Cl)))), cos(add(pH, Cl_2))), safeDiv(max(pH, Trueb), sin(max(mul(Cl, Fm_2), cos(Leit))))), cos(add(pH, Cl_2)))))))), safeDiv(cos(sin(add(Cl, Cl_2))), safeDiv(add(pH, min(Cl_2, sub(safeDiv(sin(safeDiv(Cl, Cl_2)), cos(add(pH, Cl_2))), safeDiv(max(pH, Trueb), Cl_2)))), cos(add(pH, Cl_2)))))))	726	0.30679
mul(cos(safeDiv(cos(sin(neg(safeDiv(mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(Redox, Fm))), Fm), sub(Leit, Fm))))), Fm), sub(sub(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(sub(sub(Tp, pH), pH), Cl_2)), sub(Cl_2, neg(pH))), Fm))))), cos(sub(sub(Cl_2, neg(pH)), Cl)))), add(sub(Redox, Fm), mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(Leit, Fm))))), Fm)))	1137	0.44187

add(neg(cos(mul(sub(Cl, pH), Cl))), mul(min(sin(cos(mul(pH, max(sub(max(sin(sin(Cl)), pH), max(Fm, Cl)), sin(sin(sin(add(pH, Cl))))))))), min(sin(cos(mul(pH, Cl))), Cl)), mul(max(max(Cl, Leit), min(sin(min(Cl_2, cos(mul(sin(add(safeDiv(add(Leit, Cl_2), mul(pH, Tp)), neg(Cl))), sin(cos(mul(pH, Cl))))))), Cl)), mul(Cl, min(Cl_2, Cl)))))	1120	0.13705
mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), Redox))), pH), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), add(Cl, sub(Redox, max(safeDiv(safeDiv(Redox, pH), Cl), max(mul(mul(Cl_2, safeDiv(mul(safeDiv(mul(mul(Cl_2, safeDiv(mul(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Fm, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), Tp), Cl_2)), Cl), mul(safeDiv(mul(Cl, pH), Cl), safeDiv(max(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), mul(mul(Cl_2, Cl), pH)), pH), Cl)), Cl), mul(Fm, safeDiv(Cl_2, mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), Cl_2), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox))))))))	995	0.29119

Se pot face câteva observaţii cu ajutorul acestui tabel. În primul rând un fitness scăzut, adică un număr mic de erori pentru setul de date de antrenament nu înseamna neapărat ca expresia data va clasifica bine şi instanţele din setul de testare, acest lucru fiind cauzat de overfitting. Pentru expresiile din tabel s-au obţinut următoarele rezultate:

$$F1Minim = 0.13705$$

$$F1Maxim = 0.4774$$

$$MediaF1 = 0.31582$$

Deviaţia standard măsoară dispersia unui set de date relativ la medie şi are următoarea formula:

$$DevStandard = \sqrt{\frac{\sum_{i=1}^{n}(x_i - medie)^2}{n-1}} => DevStandard = 0.09661$$

4.6 Experimentări pentru clasificare cu programare genetică

În cadrul acestui sub-capitol vor fi prezentate câteva modalităţi prin care s-a încercat creşterea performanţelor clasificării prin metoda programării genetice. În primul rând au fost încercate mai multe funcţii fitness. Acestea au fost prezentate anterior în sub-capitolul "Funcţii fitness pentru clasificare", din capitolul anterior.

În cazul cross-entropiei rezultatele au fost destul de slabe în sensul ca evoluţia programului consta în apropierea probabilităţii instanţelor din clasa majoritara de 0, fără a realiza într-o măsura semnificativă apropierea instanţelor din clasa minoritara de 1, deci clasificările realizate cu expresiile care au fost evoluate în acest mod aveau un număr mare de false negative.

În cazul funcţiei fitness reprezentată chiar de valoarea metricii F1, rezultatele au fost asemănătoare cu cazul în s-a minimizat numărul de erori, din acest motiv nu s-ar justifica o prezentare mai detaliată a rezultatelor.

O alta funcţie fitness utilizată a fost una inspirata de învăţare bazata pe cost, deci costul unei clasificări incorecte nu a mai fost considerat egal pentru ambele clase din setul de date. Costurile au fost următoarele:

	Negativ	Pozitiv
Negativ	0	1
Pozitiv	80	0

În cazul acestei funcţii fitness s-a constat ca clasificatorii obţinuţi au înregistrat un recall bun, mai mare decât în cazul celorlalte metode, însa precizia a fost foarte slaba, din acest motiv scorul F1 a fost scăzut.

Un alt mod prin care s-a încercat creşterea performanţei a fost prin metoda de sampling SMOTE. Cu ajutorul acestei metode a fost dublat numărul de instanţe din clasa minoritara, fiind folosită valoarea 5 pentru determinarea celor mai apropiaţi vecini. Cu folosirea noului set de date nu s-au obţinut îmbunatăţiri semnificative pentru scorurile F1 obţinute, diferenţele fiind irelevante având în vedere nedeterminismul presupus de programarea genetică. Consider însă ca metodele de sampling reprezintă o direcţie importantă care trebuie luata în considerare în încercarea de a aprofunda studiul acestei probleme şi de a îmbunătăţi performanţele clasificării.

4.7 Ansamblu de clasificatori bazaţi pe programare genetică

Analizând rezultatele obţinute ce au fost introduse în tabelul prezentat se poate constata că apare supra-specializarea expresiilor, în sensul ca o valoare mica a fitnessului pentru un anumit individ nu presupune ca acesta va clasifica bine şi instanţele din setul de testare, acest fapt fiind cauzat de faptul ca modelul este influenţat şi de zgomot şi detaliile din setul de antrenament, afectând astfel performanţa clasificării de instanţe noi.

Pentru a micşora supra-specializarea se va încerca o metoda inspirata de modul în care metoda Extra trees funcţionează, adică se va crea un ansamblu de clasificatori mai slabi, decizia unei clasificări pentru o instanţe fiind data de către majoritatea clasificărilor oferite de către clasificatorii din ansamblu. Se poate observa ca expresiile create au forma, structura şi dimensiune diferita, de aceea un astfel de ansamblu reprezintă un model promiţător pentru scăderea supra-specializării. De asemenea, acest tip de ansamblu este diferit fata de cel propus de AdaBoost, în care construcţia clasificatorilor din ansamblu este influenţata de performanţele celor deja aflaţi în ansamblu şi li se atribuie ponderi, acest tip de ansamblu este de tip boosting. Ansamblul creat pentru acest experiment este de tip bagging, fiecare model

rulând în mod independent, iar apoi decizia clasificării se realizează în funcţie de majoritatea clasificărilor ale elementelor din ansamblu.

O parte dintre expresiile folosite pentru clasificare au fost preluate din tabelul prezentat anterior, dar au fost introduse şi altele obţinute prin modificarea parametrilor pentru programarea genetică, pentru a obţine expresii cat mai variate, scăzându-se astfel probabilitatea ca acestea sa fie afectate de supra-specializare în mod asemănător.

Ansamblul creat este format din expresiile din următorul tabel:

Expresii:
min(mul(mul(Cl, neg(cos(sub(Cl, pH)))), mul(mul(Cl, mul(sin(cos(sub(cos(sin(sin(mul(neg(Redox), cos(sub(cos(sin(pH)), pH)))))), pH))), pH)), add(neg(Trueb), Cl))), mul(mul(Cl, mul(sin(cos(sub(cos(sin(sin(mul(neg(mul(neg(Redox), neg(Cl_2))), neg(Cl_2))))), pH))), Cl_2)), add(neg(Cl_2), Cl)))
mul(sub(safeDiv(add(cos(pH), Leit), add(Fm_2, Leit)), Cl_2), sub(mul(sin(Cl_2), sub(sub(pH, Cl_2), sin(add(Cl_2, sub(sub(pH, Cl_2), sin(add(sin(sub(pH, sin(sub(sub(pH, Cl_2), cos(pH))))), Tp)))))), cos(sin(sin(sub(pH, sin(sub(sub(pH, Cl_2), cos(pH)))))))))
mul(neg(sin(sin(neg(pH)))), sub(sub(mul(Fm, Trueb), safeDiv(Fm, pH)), mul(sin(add(sub(safeDiv(Trueb, cos(sub(Redox, neg(sin(Cl))))), mul(sin(pH), neg(pH))), Trueb)), add(add(Tp, neg(sub(mul(add(add(Leit, neg(sub(mul(Fm, Trueb), neg(sub(mul(Fm, Trueb), safeDiv(Fm_2, pH)))))), sub(Redox, Fm)), Trueb), safeDiv(Fm, pH)))), sub(Redox, Fm)))))
mul(add(mul(add(mul(min(safeDiv(add(Redox, Fm), sub(Fm_2, add(mul(add(mul(min(Fm, add(neg(pH), Cl)), Redox), sub(Fm, Redox)), cos(max(Cl, pH))), Cl))), mul(max(mul(Cl, pH), safeDiv(Fm, Redox)), min(Cl, Fm))), Redox), min(sub(Fm, Redox), Cl)), cos(max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sin(cos(Tp))))))), Cl), safeDiv(sub(sub(Fm, Redox), min(sub(Fm, Redox), max(sub(pH, min(add(Cl, Cl_2), add(min(add(Cl, Cl_2), add(neg(pH), neg(Cl_2))), neg(Cl_2)))), sub(safeDiv(sin(Cl_2), add(Leit, Tp)), sin(cos(Cl))))))), Cl_2))
safeDiv(mul(add(neg(neg(sin(safeDiv(Cl, cos(mul(add(neg(neg(pH)), Cl), neg(sin(Leit)))))))), add(neg(neg(sin(pH))), Cl)), neg(Cl)), safeDiv(cos(safeDiv(max(safeDiv(cos(safeDiv(max(pH, pH), sub(neg(sin(pH)), Fm))), cos(mul(add(neg(neg(sin(pH))), Cl), neg(add(add(sub(pH, Cl), safeDiv(Trueb, Redox)), min(sin(Cl_2), add(Tp, Cl_2))))))), safeDiv(add(max(Trueb, pH), add(pH, Leit)), cos(neg(Cl_2)))), sub(Tp, Fm))), cos(mul(add(neg(neg(sin(pH))), Cl), neg(pH)))))
mul(cos(safeDiv(cos(sin(neg(safeDiv(mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(Redox, Fm))), Fm), sub(Leit, Fm))))), Fm), sub(sub(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(sub(sub(Tp, pH), pH), Cl_2)), sub(Cl_2, neg(pH))), Fm))))), cos(sub(sub(Cl_2, neg(pH)), Cl)))), add(sub(Redox, Fm), mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(Leit, Fm))))), Fm)))
mul(cos(pH), sub(mul(sub(Fm, Redox), neg(add(sin(sin(pH)), cos(pH)))), mul(pH, mul(pH, sin(add(Tp, Tp))))))
mul(mul(sin(sin(sin(add(sin(sin(add(add(Cl_2, pH), add(Cl_2, Cl_2)))), add(Cl_2, Cl_2))))), cos(neg(add(add(Cl_2, pH), add(Cl_2, pH))))), safeDiv(add(add(mul(safeDiv(neg(add(add(Cl_2, pH), add(Cl_2, pH))), cos(add(add(Cl_2, pH), add(Cl_2, Cl_2)))), pH), add(mul(Cl_2, Tp), neg(sub(Redox, add(Tp, neg(sub(sin(add(mul(Cl, pH), Cl_2)), add(add(add(sin(cos(sub(Fm_2, Cl_2))), mul(Cl_2, Redox)), pH), Fm)))))))), Fm), sub(Redox, add(add(add(Cl_2, mul(Cl_2, Redox)), add(Cl_2, pH)), Fm))))
neg(safeDiv(sin(cos(mul(safeDiv(mul(Tp, Cl), neg(cos(pH))), Cl))), sub(safeDiv(pH, sub(sub(cos(cos(cos(neg(cos(add(mul(safeDiv(mul(Tp, Cl), pH), Cl), add(Tp, Tp)))))), neg(mul(safeDiv(mul(Tp, Cl), pH), mul(safeDiv(mul(Tp, mul(safeDiv(mul(Tp, Cl), neg(cos(pH))), Cl)), neg(cos(pH))), Cl)))), safeDiv(safeDiv(mul(safeDiv(mul(Tp, Cl), neg(cos(pH))), Cl), neg(cos(pH))), neg(cos(pH)))))), mul(safeDiv(cos(cos(Trueb)), neg(cos(pH))), Cl))))
mul(mul(mul(mul(Leit, Cl_2), sub(mul(mul(Leit, Cl_2), sub(mul(mul(sub(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), Redox), Cl_2), add(mul(sub(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), Redox), Cl_2), neg(mul(Leit, Cl_2)))), Fm)), Fm)), safeDiv(sub(Fm, Redox), sin(pH))), add(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), neg(safeDiv(sub(mul(Leit, add(mul(Leit, Cl_2), neg(Tp))), Redox), sin(pH)))))
safeDiv(max(Fm, cos(max(cos(Fm), pH))), mul(neg(cos(sub(sub(pH, min(Leit, Cl_2)), min(Leit, Cl_2)))), safeDiv(sub(max(Redox, Fm), Redox), min(add(cos(max(Cl_2, pH)), sin(sin(pH))), Fm))))
mul(mul(Cl_2, safeDiv(add(safeDiv(sin(Trueb), safeDiv(sin(add(add(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))), Cl_2), Cl_2)), add(neg(mul(sin(add(add(sin(safeDiv(sin(sin(sin(add(add(neg(safeDiv(sin(sin(safeDiv(sin(sin(pH)), Cl_2))), Cl_2)), Cl_2), Cl_2)))), Cl_2)), Cl_2), Cl_2), Cl_2)), mul(sin(sin(sin(safeDiv(sin(sin(sin(sin(sin(safeDiv(sin(sin(pH)), sin(pH))))))))), Cl_2)))), Cl_2))), Redox))), Redox), safeDiv(sin(add(add(sin(safeDiv(sin(sin(pH)), Cl_2)), Cl_2), Cl_2), add(add(sin(sin(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))))), Cl_2), Cl_2))), add(neg(mul(safeDiv(sin(sin(pH)), Cl_2), mul(safeDiv(sin(sin(pH)), Cl_2), safeDiv(add(sin(add(add(add(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))), Cl_2), Cl_2)), Cl_2), Cl_2)))), Redox)))), sub(Fm, add(safeDiv(sin(Trueb), safeDiv(sin(sin(add(add(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2))))), Cl_2), Cl_2))), add(neg(mul(Fm, safeDiv(sin(sin(sin(sin(sin(safeDiv(sin(sin(pH)), Cl_2)))))), Cl_2))), Redox))), Redox)))

mul(sub(pH, add(neg(mul(Fm, Cl_2)), add(cos(Redox), sub(Fm, mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, sub(Cl, Cl_2))), mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), pH)), -1), Redox)), Cl_2))), pH)), -1), Redox))), pH)), Cl_2))), pH)), -1), Redox))), add(Redox, sub(Fm, mul(neg(pH), neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), add(neg(mul(Fm, sub(Cl, Cl_2))), mul(sub(Fm, add(neg(mul(Fm, Cl_2)), add(sub(sub(Fm, mul(neg(neg(mul(add(neg(mul(Fm, mul(Cl_2, Cl_2))), add(sub(sub(Fm, mul(neg(neg(mul(Fm, Cl_2))), pH)), -1), Cl)), Cl_2))), pH)), -1), Redox))), pH))))), -1), Redox)), mul(Fm, mul(Cl_2, Cl_2))))))))))))), Redox)
neg(min(max(cos(min(Fm_2, Fm)), max(add(add(pH, min(max(add(Cl_2, mul(pH, add(safeDiv(mul(Cl_2, Fm_2), Redox), mul(pH, Cl_2)))), min(safeDiv(mul(Redox, pH), neg(pH)), Cl)), add(add(mul(pH, Cl_2), Cl_2), pH))), pH), max(cos(Cl_2), sin(Cl_2)))), cos(add(pH, min(max(add(min(max(Fm_2, max(add(safeDiv(Cl_2, Redox), pH), max(cos(Cl_2), sin(Cl_2)))), cos(add(pH, min(max(add(safeDiv(Cl_2, sin(Redox)), pH), add(pH, Cl_2)), add(add(pH, Cl_2), add(safeDiv(mul(Cl_2, Fm_2), Redox), mul(pH, Cl_2)))))), mul(pH, add(safeDiv(mul(Cl_2, Fm_2), Redox), mul(pH, Cl_2)))), min(Cl, Cl)), add(add(pH, Cl_2), Cl_2))))))
mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), Redox))), pH), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), add(Cl, sub(Redox, max(safeDiv(safeDiv(Redox, pH), Cl), max(mul(mul(Cl_2, safeDiv(mul(safeDiv(mul(mul(Cl_2, safeDiv(mul(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Fm, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), Tp), Cl_2)), Cl), mul(safeDiv(mul(Cl, pH), Cl), safeDiv(max(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), mul(mul(Cl_2, Cl), pH)), pH), Cl)), Cl), mul(Fm, safeDiv(Cl_2, mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), Cl_2), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox))))))))
safeDiv(neg(safeDiv(sin(min(Fm, pH)), safeDiv(mul(min(mul(sub(Tp, Leit), cos(pH)), Trueb), mul(Cl_2, pH)), sub(add(Cl_2, mul(min(add(add(Cl_2, pH), add(cos(add(pH, Redox)), safeDiv(Cl_2, sin(max(Trueb, Leit))))), pH), mul(Cl_2, Fm))), Redox)))), cos(add(add(min(mul(Cl_2, min(sin(add(Tp, add(pH, Redox))), pH)), add(Cl_2, pH)), pH), add(Cl_2, pH))))
mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), Redox))), pH), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), add(Cl, sub(Redox, max(safeDiv(safeDiv(Redox, pH), Cl), max(mul(mul(Cl_2, safeDiv(mul(safeDiv(mul(mul(Cl_2, safeDiv(mul(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(max(mul(mul(Cl_2, safeDiv(mul(Redox, pH), Cl)), Cl), mul(safeDiv(mul(Fm, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), Tp), Cl_2)), Cl), mul(safeDiv(mul(Cl, pH), Cl), safeDiv(max(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), mul(mul(Cl_2, Cl), pH)), pH), Cl)), Cl), mul(Fm, safeDiv(Cl_2, mul(mul(mul(Cl_2, Cl), pH), Redox)))), pH), Cl)), Cl_2), mul(safeDiv(mul(Redox, pH), Cl), safeDiv(mul(Cl, pH), mul(mul(mul(Cl_2, Cl), pH), Redox))))))))
safeDiv(neg(add(min(min(mul(Cl, Redox), neg(add(pH, add(safeDiv(cos(safeDiv(min(Trueb, Trueb), mul(pH, Redox))), cos(max(Redox, Tp))), neg(sub(Cl_2, Tp)))))), min(neg(max(safeDiv(max(Cl, max(Redox, Fm)), sub(mul(Fm, Cl_2), min(pH, Trueb))), min(cos(add(max(Cl, Leit), sin(Cl_2))), Cl))), Cl)), add(mul(Cl, Redox), sub(add(mul(Cl, Redox), sub(neg(add(add(Tp, Cl_2), safeDiv(Tp, sub(add(mul(Cl, Redox), neg(safeDiv(mul(Fm_2, Leit), safeDiv(Redox, Trueb)))), safeDiv(pH, Cl_2)))), safeDiv(pH, Cl_2))), safeDiv(pH, Cl_2))))), sub(add(mul(Cl, Redox), sub(neg(add(pH, add(safeDiv(sin(Leit), mul(pH, Leit)), neg(sub(Cl_2, pH))))), safeDiv(pH, Cl_2))), safeDiv(pH, Cl_2)))
neg(safeDiv(neg(safeDiv(Cl_2, add(neg(neg(mul(add(Cl_2, mul(Cl, Cl)), neg(mul(Cl_2, pH))))), Cl))), mul(safeDiv(Fm_2, Cl_2), mul(safeDiv(Fm_2, Cl_2), add(neg(safeDiv(add(mul(safeDiv(Fm_2, Cl_2), add(neg(safeDiv(add(neg(mul(add(neg(mul(add(Cl_2, mul(Cl, Cl)), neg(mul(Cl_2, pH)))), mul(Cl, Cl)), neg(mul(Cl_2, pH)))), sub(Cl_2, Cl)), neg(Cl_2))), neg(mul(add(Cl_2, mul(Cl, Cl)), neg(mul(Cl_2, pH))))))), sub(Cl_2, Cl)), neg(Cl_2))), neg(mul(add(Cl_2, mul(Cl, mul(safeDiv(Fm_2, Cl_2), mul(safeDiv(Fm_2, Cl_2), add(neg(safeDiv(add(neg(mul(add(neg(mul(add(Cl_2, mul(Cl, Cl)), neg(mul(Cl_2, pH)))), mul(Cl, Cl)), neg(mul(Cl_2, pH)))), sub(Cl_2, Cl)), neg(Cl_2))), neg(mul(add(Cl_2, mul(Cl, Cl)), neg(mul(Cl_2, pH))))))))), neg(Cl))))))))
safeDiv(neg(safeDiv(Cl, mul(safeDiv(sub(safeDiv(sub(Cl, Cl_2), Cl_2), Cl), sub(add(add(Cl_2, add(safeDiv(neg(safeDiv(add(pH, Tp), sin(Redox))), Leit), pH)), pH), pH)), mul(cos(add(add(Cl_2, add(cos(sub(safeDiv(sub(Cl, Cl_2), Cl_2), min(safeDiv(safeDiv(sub(Cl, Cl_2), Cl_2), Cl_2), neg(safeDiv(safeDiv(Fm, Leit), sin(Redox))))), safeDiv(sub(Cl, Cl_2), Cl_2)))), pH)), pH)), Leit)))), cos(add(add(Cl_2, add(Cl_2, pH)), pH)))
neg(safeDiv(cos(mul(add(-1, mul(Cl, Trueb)), pH)), mul(Fm, mul(safeDiv(safeDiv(Fm, Cl_2), Cl_2), mul(safeDiv(safeDiv(Fm_2, Cl_2), Cl_2), add(mul(safeDiv(pH, mul(add(-1, mul(Cl, pH)), pH)), Tp), mul(safeDiv(Tp, mul(safeDiv(-1, mul(Cl, pH)), Tp)), Tp)))))))
mul(cos(safeDiv(cos(sin(neg(safeDiv(mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(Redox, Fm))), Fm), sub(Leit, Fm))))), Fm), sub(sub(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(sub(sub(Tp, pH), pH), Cl_2)), sub(Cl_2, neg(pH))), Fm))))), cos(sub(sub(Cl_2, neg(pH)), Cl)))), add(sub(Redox, Fm), mul(neg(sin(neg(safeDiv(mul(neg(safeDiv(mul(Cl, Fm), sub(pH, Fm))), Fm), sub(Leit, Fm))))), Fm)))
min(mul(mul(Cl, neg(cos(sub(Cl, pH)))), mul(mul(Cl, mul(sin(cos(sub(cos(sin(sin(mul(neg(Redox), cos(sub(cos(sin(pH)), pH))))))), pH))), pH)), add(neg(Trueb), Cl))), mul(mul(Cl, mul(sin(cos(sub(cos(sin(sin(mul(neg(mul(neg(Redox), neg(Cl_2))), neg(Cl_2))))), pH))), Cl_2)), add(neg(Cl_2), Cl)))

Rezultatul scorului F1 obţinut de către acest ansamblu demonstrează faptul ca acesta metoda poate îmbunătăţi performanţele clasificării prin metoda programării genetice într-o măsura considerabila.

$$F1Ansamblu = 0.52447$$

4.8 Interpretarea rezultatelor

În acest capitol au fost evaluaţi o serie de clasificatori clasici cu scopul de a compara performanţele programării genetice folosită în problema clasificării cu aceştia. Pentru algoritmii de clasificare au fost setaţii hiper-parametrii cu algoritmul de optimizare al acestora grid search. Acest lucru este esenţial pentru o comparaţie cat mai corecta, deoarece ei influenţează într-o proporţie ridicata performanţele algoritmilor.

Modul prin care se efectuează compararea performantelor este scorul F1 obţinut de clasificatori pentru setul de testare. Dintre algoritmii de clasificare metoda Extra trees a obţinut cel mai bun scor:0.43818, urmat de regresia logistica cu 0.3, AdaBoost cu 0.28683 şi arborele de decizie cu 0.26423.Media scorului F1 pentru clasificatorii obţinuţi prin programare genetică dintr-un grup format din 30 de expresii a fost de 0.31582, mai slaba decât scorul obţinut prin metoda Extra trees, dar mai buna decât al celorlaţi algoritmi de clasificare.

Pentru îmbunatăţirea performanţei programării genetice în cazul acestei probleme sau încercat o serie de metode. Un rezultat a fost notabil, crescând în mod semnificativ performanţele clasificării. A fost creat un ansamblu de clasificatori obţinuţi prin programare genetică. Crearea unui astfel de model de tip ansamblu presupune în mod evident mai mult efort computaţional, dar performanţele sunt îmbunătăţite foarte mult. Scorul F1 obţinut cu această metodă este 0.52447, mult mai mare decât media scorurilor F1 obţinute în mod individual.

Aşadar cea mai buna performanţa a fost obţinută de către ansamblul de clasificatori cu programare genetică, ceea ce arata faptul ca această metodă poate fi utilizata cu succes în practica. Vizualizarea comparării rezultatelor arata astfel:

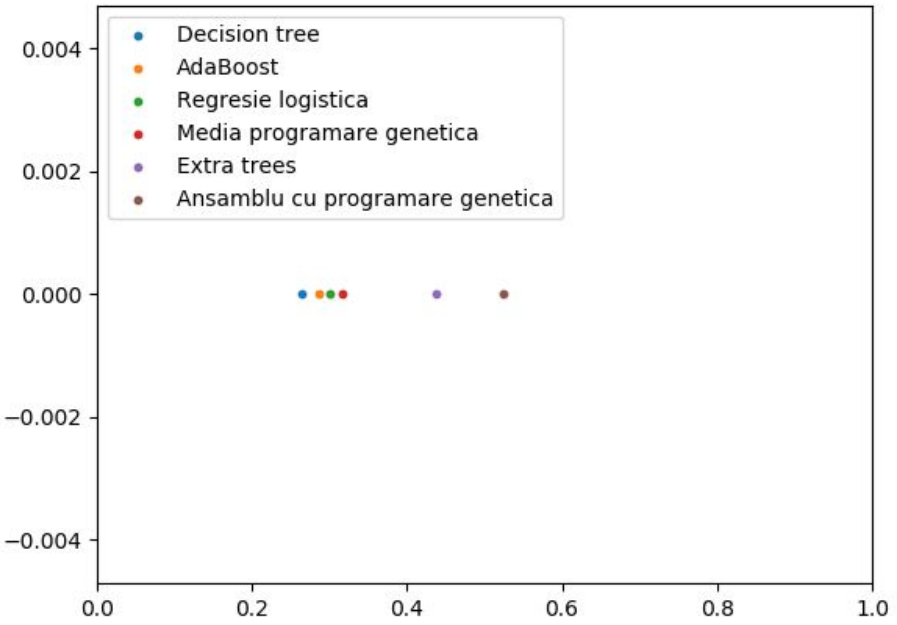

Figura 14.
Vizualizarea rezultatelor obţinute

Rezultatele competiţiei sunt publicate, aşadar se poate face o comparaţie intre rezultatele ansamblului şi rezultatele obţinute de către competitori. Cele mai bune 5 rezultate obţinute de către concurenţi sunt:

Autori	Scor F1
MINH NGUYEN, AFFAN HAMDANI, KALAICHELVI RAJENDRAN	0.56123041
FITORE MUHARREMI	0.47693977
VICTOR H. A. RIBEIRO (SVM)	0.45098039
VICTOR H. A. RIBEIRO (DT)	0.39030920
ABDUL_RASHEEQ, JAGADESH_DEVAN, JITHI_VARGHESE	0.35265018

Ansamblul nu obţine o valoare mai buna decât concurentul de pe primul loc, însa are un scor mai bun decât tot restul concurenţilor. Consider ca rezultatul obţinut evidenţiază faptul ca programarea genetică poate fi folosită cu succes în problema clasificării. Studiul acestei metode poate fi continuat, prin încercarea mai multor moduri de îmbunatăţire a performanţei, în special prin metode de sampling.

Concluzii

În această lucrare s-a realizat un studiu al performanţelor ce pot fi obţinute pentru problema clasificării cu ajutorul programării genetice şi compararea cu o serie de algoritmi populari pentru această problema.

Compararea performantelor s-a realizat în funcţie de scorul F1, care este o metrică potrivită pentru învăţarea din seturi de date dezechilibrate deoarece nu ia în calcul numărul de clasificări corecte din clasa majoritara, care nu este atât de relevant în acest caz fapt ilustrat prin paradoxul acurateţei. Un scor F1 mare indica un număr mare de clasificări true positive şi un număr scăzut de clasificări false positives şi false negatives.

Seturile de date de antrenament şi testare au fost preluate din cadrul competiţiei "Internet of Things: Online Anomaly Detection for Drinking Water Quality", acestea fiind măsurători reale ale parametrii importanţi în stabilirea calităţii apei. Unele variaţii ale acestor parametrii sunt normale, însa altele pot indica faptul ca aceasta este contaminata, nu este potabilă. Scopul clasificării este de a detecta dacă o instantă definita prin diverse valori pentru atributele măsurate este potabilă sau nu. Aşadar, performanţele obţinute cu clasificarea prin programare genetică în aceste experimente sunt reprezentative deoarece indica faptul ca această metodă poate fi folosită în probleme de clasificare din viata reală cu succes.

Modul în care a fost abordata această problema a fost prin căutarea unor expresii matematice cu ajutorul programării genetice. Parametrii expresiei sunt chiar atributele unei instanţe din setul de date. Cu ajutorul rezultatului obţinut pentru expresie se calculează ulterior o probabilitate pentru clasificare, cu ajutorul căreia se stabileşte dacă instanţa este clasificată în clasa pozitivă sau în cea negativă.

Media scorurilor F1 pentru un set de 30 de expresii generate prin programare genetică a fost apropiată şi puţin mai mare decât rezultatele obţinute pentru metodele de clasificare AdaBoost, regresie logistica şi arbori de decizie, dar mai mică decât scorul obţinut prin metoda Extra trees, care este de tip ansamblu.

Au fost încercate mai multe modalităţi de îmbunatăţire a performantei clasificării cu programarea genetică. Cel mai important rezultat a fost obţinut prin crearea unui ansamblu de clasificatori de tip bagging, obţinuţi prin programare genetică.

Acest ansamblu a oferit cel mai bun scor dintre cele obţinute în această lucrare, fiind la o diferenţa relativ mica faţă de cea mai buna submisie din cadrul competiţie. Aceste rezultate indică faptul ca programarea genetică poate oferi rezultate bune în problema clasificării datelor.

Concluzia principală a acestei lucrări este ca programarea genetică poate fi folosită cu succes pentru problema clasificării, însa este o metodă costisitoare din punct de vedere al timpului. Crearea unui ansamblu de tipul celui prezentat în această lucrare presupune un timp mai mare decât restul metodelor, algoritmilor cu care s-au făcut comparaţii.

.

Bibliografie

1.William J. Clancey - Heuristic Programming Project Computer Science Department Stanford University Stanford. CA 94305 Classification problem solving

2. Robert E. Schapire - Explaining AdaBoost

3. Tom M. Mitchell - Machine Learning, 1997

4.S. B. Kotsiantis - Supervised Machine Learning: A Review of Classification Techniques, 2007

5.Louis Wehenkel, Damien Ernst, Pierre Geurts - Ensembles of extremely randomized trees and some generic applications

6.Pierre Geurts, Damien Ernst, Louis Wehenkel - Extremely randomized trees

7.Dmitry Bystrov - Introduction to soft computing

8.Charles X. Ling, Victor S. Sheng - Cost-Sensitive Learning and the Class Imbalance Problem

9.John R. Koza - Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992, paginile 162-169

10.Henri Luchian, Mihaela Breabăn - Algoritmi genetici

11.Riccardo Poli, William B. Langdon, Nicholas F. McPhee - A field guide to genetic programming

12.John R. Koza -Genetic Programming: On the Programming of Computers by Means of Natural Selection

13. Haibo He, Edwardo A. Garcia - Learning from Imbalanced Data

Site-uri sau alte resurse:

14. Sursa originală a setului de date:
http://www.spotseven.de/gecco/gecco-challenge/gecco-challenge-2018/

15. https://deap.readthedocs.io/en

16. https://towardsdatascience.com/