UNIVERSITATEA BABEŞ-BOLYAI, CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Clasificare nesupervizată prin mulțimi rough și modele de învățare supervizată cu aplicații în testarea de regresie

Rezumatul tezei de doctorat

Coordonator ştiinţific
Prof. Dr. Horia F. Pop

Student Doctorand
Arnold Szederjesi-Dragomir

2024

Cuvinte cheie: Clasificare nesupervizată, Mulțimi rough, Sisteme multi-agent, Clasificare supervizată, Prioritizarea cazurilor de testare, Integrarea Continuă

Cuprins

Capitolul 1

Introducere

Această teză de doctorat, intitulată *Clasificare nesupervizată prin mulțimi rough și modele de învățare supervizată cu aplicații în testarea de regresie*, prezintă cercetarea noastră privind dezvoltarea a noi modele de învățare automată, precum și aplicarea acestora în domeniul testării software. Contribuțiile noastre includ propunerea de metode și abordări noi, centrate în principal pe domeniile clasificării nesupervizate și prioritizării cazurilor de testare. În clasificarea nesupervizată, am dezvoltat noi metodologii pentru îmbunătățirea tehnicilor de clasificare în medii incerte, inclusiv studiul măsurilor de evaluare a performanței în astfel de contexte și strategii cuprinzătoare de evaluare a performanței. Pentru prioritizarea cazurilor de testare, această lucrare include mai multe abordări care simplifică și îmbunătățesc eficiența proceselor de prioritizare în diverse scenarii de testare regresivă. Am aplicat mai multe abordări pe seturi de date industriale din lumea reală sau chiar integrate în proiecte software industriale din lumea reală.

1.1 Motivație

Învățarea automată [1], o parte importantă a inteligenței artificiale, transformă modul în care interacționăm și extragem valoare din date. Calculatoarele învață să identifice tipare și să facă predicții cu un minim de ghidare umană, revoluționând industriile pe scară largă. De la experiențe personalizate la descoperiri medicale, învățarea automată stimulează inovația în domenii precum finanțele, procesarea limbajului și multe altele. Această tehnologie este o bază a progresului modern – automatizând sarcinile, aprofundând înțelegerea noastră asupra datelor și conducându-ne către un viitor de eficiență și perspicacitate fără precedent.

Învățarea nesupervizată (sau clasificarea nesupervizată) [11], o ramură esențială a învățării automate, explorează lumea datelor neetichetate. Spre deosebire de varianta sa supervizată, aceasta nu se bazează pe exemple preclasificate pentru a-și ghida procesul de învățare. În schimb, aceasta permite algoritmilor să descopere modele, structuri și relații ascunse în cadrul datelor în sine. Această explorare le permite să îndeplinească o varietate de sarcini, cum ar fi gruparea punctelor de date similare, reducerea dimensionalității pentru vizualizarea datelor și detectarea anomaliilor. Învățarea nesupervizată joacă un rol vital în domenii diverse, de la analiza imaginii și textului până la cercetarea de piață și descoperirile științifice, oferind un instrument puternic pentru deblocarea potențialului ascuns în date.

Clasificarea nesupervizată [3], o metodă centrală în recunoașterea tiparelor, se confruntă cu provocări în aplicațiile din lumea reală din cauza grupurilor suprapuse, a valorilor anormale și a formelor complexe de date. *Această cercetare propune metode inovative pentru a rezolva aceste provocări prin integrarea conceptelor din teoria mulțimilor rough în diferiți algoritmi de clasificare nesupervizată*

aglomerativă.

Testarea software-ului [4], baza dezvoltării software, asigură calitatea și fiabilitatea programelor pe care ne bazăm zilnic. Aceasta implică un proces minuțios de evaluare a software-ului în raport cu funcționalitățile sale preconizate și de identificare a eventualelor erori sau discrepanțe. Această etapă esențială acționează ca o plasă de siguranță, protejând utilizatorii de întâmpinarea unor probleme neașteptate și asigurând că software-ul funcționează conform așteptărilor.

Pe măsură ce software-ul evoluează prin noi funcționalități și corecții de erori, testarea de regresie devine crucială [20]. Aceasta implică re-rularea cazurilor de testare existente, utilizate inițial pentru a valida versiunea originală a software-ului, pentru a asigura că modificările recente nu au introdus neintenționat probleme noi. Această verificare continuă este esențială pentru menținerea stabilității și fiabilității software-ului, mai ales pe măsură ce acesta suferă modificări.

În ritmul accelerat al dezvoltării software, practicile de integrare continuă (CI) [10] au apărut pentru a eficientiza ciclul de dezvoltare și testare. CI implică integrarea regulată a modificărilor de cod de la diferiți dezvoltatori într-un depozit central, urmată de testare automată pentru a identifica potențiale probleme cât mai devreme. Această buclă continuă de integrare și testare facilitează feedback-ul rapid și detectarea mai rapidă a erorilor, asigurând calitatea software-ului pe parcursul întregului proces de dezvoltare.

Testarea de regresie, deși necesară, poate fi consumatoare de timp din cauza volumului mare de cazuri de testare implicate. Prioritizarea cazurilor de testare (TCP) [7] apare ca o metodă valoroasă în candrul testării de regresie. Aceasta implică reordonarea strategică a cazurilor de testare pe baza diferiților factori, cum ar fi probabilitatea de a descoperi erori, timpul de execuție și datele istorice. Această prioritizare permite testerilor să se concentreze pe cele mai critice cazuri de testare mai întâi, optimizând procesul de testare și minimizând timpul petrecut pentru identificarea problemelor critice.

1.2 Obiectivele tezei

În această teză, am identificat multiple obiective care au ghidat cercetarea noastră:

1. Studierea teoriei mulțimilor *fuzzy* și *rough* pentru a evalua eficacitatea acestora în contexte incerte
2. Propunerea de modele noi de clasificare nesupervizată bazate pe mulțimi *rough*
3. Implementarea metodelor de clasificare nesupervizată *rough* propuse
4. Studierea metodologiilor existente de evaluarea clasificărilor
5. Propunerea și implementarea unei metodologii pentru evaluarea rezultatelor clasificării *rough*
6. Evaluarea performanței acestor modele noi folosind metodologii standard și cele propuse de noi
7. Aplicarea noilor metode de clasificare nesupervizată *rough* pe seturi de date standard și din lumea reală
8. Cercetarea domeniului testării de regresie, în special prioritizarea cazurilor de testare (TCP)
9. Introducerea unei abordări unificate pentru construirea seturilor de date TCP
10. Analiza metodelor de învățare supervizată de ultimă generație pentru TCP
11. Propunerea de modele bazate pe rețele neuronale pentru TCP

12. Integrarea noilor metode bazate pe rețele neuronale într-o soluție reală și testarea pe un proiect din lumea reală
13. Aplicarea noilor abordări dezvoltate de clasificare nesupervizată *rough* în testarea software, mai exact în prioritizarea cazurilor de testare

1.3 Contribuții originale

Abordarea noastră inovatoare, ABARC [6], un algoritm de clasificare nesupervizată bazat pe mulțimi *rough*, identifică în mod eficient punctele de date care pot aparține mai multor grupuri, separând simultan valorile anormale de datele de bază. Acest proces este condus de agenți software care operează în paralel, promovând eficiența computațională. În plus, investigăm impactul diferitelor măsuri de similaritate asupra grupurilor din rezultate, în special în prezența datelor suprapuse.

Mai mult, experimentele efectuate pe seturi de date standard demonstrează rolul semnificativ al alegerii unei măsuri de similaritate adecvate pentru a obține o clasificare precisă [14, 15], în special atunci când se lucrează cu date suprapuse. Deoarece seturile de date standard nu conțin informații despre zonele de suprapunere, am implementat o metodă pentru a extrage aceste date în scopuri de referință. Această cercetare evidențiază potențialul teoriei mulțimilor *rough* și selecția a măsurilor de similaritate pentru o clasificare eficientă în scenarii reale.

În plus, am realizat, de asemenea, o evaluare cuprinzătoare a metodologiilor noastre de clasificare utilizând diferite tipuri de metrici [13]. Rezultatele arată performanța promițătoare a metodelor ABARC.

Testarea de regresie în medii CI necesită execuția eficientă a numeroase cazuri de testare pentru a asigura calitatea software-ului. Tehnicile de prioritizare a cazurilor de testare (TCP) abordează această provocare prin reordonarea testelor pentru a prioritiza pe cele cu o probabilitate mai mare de a descoperi erori, reducând astfel timpul și costurile de testare.

Investigăm eficiența unui model bazat pe rețele neuronale, NEUTRON [18], pentru TCP în CI. NEUTRON analizează diverse caracteristici precum durata execuției, rata de defecte și istoricul testelor pentru a prioritiza inteligent cazurile de testare. Studiul demonstrează că NEUTRON depășește prioritizarea aleatorie și obține rezultate similare sau mai bune comparativ cu tehnicile existente, în special atunci când se iau în considerare bugete de testare mai mari (75% și 100%).

Pe baza succesului NEUTRON, am implementat și MixTCP [16], un sistem aplicat conceput pentru a integra perfect modelul în fluxurile de lucru de dezvoltare software. Implementat în Elixir, MixTCP utilizează NEUTRON pentru a prioritiza testele, îmbunătățind eficient detectarea defectelor și reducând timpul de testare. În plus, arhitectura sa modulară permite integrarea viitoare a altor soluții TCP.

De asemenea, evidențiem potențialul NEUTRON și MixTCP pentru optimizarea proceselor de testare CI. NEUTRON abordează provocările prioritizării cazurilor de testare în medii CI, în timp ce MixTCP oferă dezvoltatorilor o soluție eficientă și ușor de utilizat pentru a valorifica acest model avansat în cadrul fluxului lor de lucru. Prin optimizarea testării de regresie, aceste avansări contribuie la cicluri de dezvoltare software mai rapide și mai fiabile.

RoughTCP [5] reprezintă abordarea noastră propusă pentru TCP, bazată pe clasificare nesupervizată aglomerativă utilizând mulțimi *rough*. Acest lucru îi permite să se adapteze la medii CI dinamice fără a necesita date etichetate. Experimentele demonstrează că RoughTCP performează

bine, în special cu bugete de testare mai mari.

Un framework nou [19] pe care l-am propus pentru TCP ia în considerare diferite aspecte, cum ar fi informațiile de trasabilitate, contextul și informațiile despre caracteristici. Acest framework își propune să ofere o viziune mai holistică asupra problemei TCP. Experimentele folosind un set de date sintetic demonstrează că metoda de clasificare nesupervizată bazată pe acest framework depășește constant celelalte metode, evidențiind eficiența sa.

În general, aceste progrese demonstrează eforturile continue de a dezvolta tehnici TCP eficiente și adaptabile pentru optimizarea dezvoltării software-ului în medii CI.

În concluzie, această teză face legătura între avansările teoretice în învățarea automată și aplicațiile în testarea software-ului. Nu doar că oferă contribuții teoretice în domeniul clasificării nesupervizate, ci demonstrează și valoarea acestor contribuții prin aplicarea lor la provocarea reală a testării software-ului.

Contribuțiile originale ale cercetării, care sunt prezentate în Capitolele 3, 4 și 5, sunt următoarele:

- **Cercetarea fundamentală asupra clasificării nesupervizate prin mulțimi *rough***
  - Metode noi de clasificare nesupervizată bazate pe teoria mulțimilor *rough* - Clasificarea nesupervizată *rough* bazată pe agenți (ABARC) (Secțiunea 3.1) [6].
  - O nouă metodologie pentru evaluarea rezultatelor în contexte incerte (Secțiunea 3.1) [6].
  - Studiu comparativ al mai multor metrici de similaritate, în special în medii *rough*, folosind ABARC. Metodologie pentru a le evalua (Secțiunea 3.2) [14, 15].
  - Analiza metricilor de evaluare a performanței (interne, externe și *rough*) și experimentele aferente pe ABARC în condiții necunoscute (Secțiunea 3.3) [13].
  - Experimente pe ABARC și pe metricile de similaritate pe diverse seturi de date standard (Secțiunile 3.1, 3.2) [6, 14, 15].
- **Cercetarea fundamentală asupra Prioritizării Cazurilor de Testare**
  - Modele noi bazate pe rețele neuronale - Prioritizarea Cazurilor de Testare Bazată pe Rețele Neurale în Integrarea Continuă (NEUTRON) (Secțiunea 4.1) [18].
  - O abordare unificată, unică și cuprinzătoare a prioritizării moderne a cazurilor de testare, care ia în considerare toate principiile utilizate în diverse cazuri de testare de regresie (Secțiunea 5.2) [19].
  - Un set de date sintetic unificat și demonstrarea eficienței acestuia folosind ABARC în contextul prioritizării cazurilor de testare (Secțiunea 5.2) [19].
  - Abordări de învățare nesupervizată bazate pe mulțimi *rough* pentru prioritizarea cazurilor de testare - RoughTCP (Secțiunea 5.1) [5].
- **Cercetare aplicativă**
  - Aplicarea lui NEUTRON pe multiple seturi de date industriale (Secțiunea 4.1) [18].
  - Integrarea lui NEUTRON într-o soluție din lumea reală - MixTCP (Secțiunea 4.2) [16].
  - Aplicarea lui NEUTRON prin MixTCP pe un proiect industrial din lumea reală (Secțiunea 4.2) [16].
  - Experimente pentru RoughTCP folosind seturi de date industriale standard (Secțiunea 5.1) [5].

1.4 Lista de publicații

Clasificarea publicațiilor a fost realizată conform listelor recente de reviste și conferințe publicate de CNATDCU (Consiliul Național de Atestare a Titlurilor, Diplomelor și Certificatelor Universitare) care sunt aplicabile începând cu 1 octombrie 2018.

1. [6] R. D. Găceanu, **A. Szederjesi-Dragomir**, H. F. Pop, and C. Sârbu, "ABARC: An Agent-Based Rough Sets Clustering Algorithm" Intelligent Systems with Applications, vol. 16, p. 200117, 2022.
   Indexed Web of Science and Scopus, Rank C, 1 punct.
2. [14] **A. Szederjesi-Dragomir**, R. D. Găceanu, H. F. Pop, and C. Sârbu, "A Comparison Study of Similarity Measures in Rough Sets Clustering" in Proceedings of the 2019 IEEE 15th International Scientific Conference on Informatics (INFORMATICS 2019). Editors: William Steingartner, Štefan Korečko, Anikó Szakál, pp. 399 – 405, IEEE Press, Poprad, Slovakia, Nov 20 – 22, 2019.
   Indexed IEEE, Rank D, 0.5 puncte.
3. [15] **A. Szederjesi-Dragomir**, R. D. Găceanu, H. F. Pop, and C. Sârbu, "Experiments on Rough Sets Clustering with Various Similarity Measures" IPSI BGD TRANSACTIONS ON INTERNET RESEARCH, vol. 16, pp. 75–83, 2020.
   Indexed Web of Science, Rank D, 0 puncte.
4. [18] A. Vescan, R. D. Găceanu, and **A. Szederjesi-Dragomir**, "Neural Network-Based Test Case Prioritization in Continuous Integration" in 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), pp. 68–77, IEEE, IEEE Computer Society, sep 2023.
   Indexed IEEE, Workshop at Rank A conference, 6 puncte.*
5. [13] **A. Szederjesi-Dragomir**, "A Comprehensive Evaluation of Rough Sets Clustering in Uncertainty Driven Contexts". Studia Universitatis Babes-Bolyai Informatica 69, 1 (2024), 41–56.
   Indexed Mathematical Reviews, Rank D, 1 point.
6. [19] A. Vescan, R. D. Găceanu, **A. Szederjesi-Dragomir**, "Embracing Unification: a Comprehensive Approach to Modern Test Case Prioritization" in Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2024, INSTICC, Angers, France, 29 april 2024, SciTePress, pp. 396–405.
   Indexed IEEE, short paper at Rank B conference, 2.66 puncte.
7. [16] **A. Szederjesi-Dragomir**, R. D. Găceanu, and A. Vescan, "Industrial Validation of a Neural Network Model Using the Novel MixTCP Tool" in Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2024, INSTICC, Angers, France, 29 april 2024, SciTePress, pp. 110–119.
   Indexed IEEE, Rank B conference, 4 puncte.

Puncte din publicații: 15.16 puncte.

Publicații trimise

1. [5] R. D. Găceanu, **A. Szederjesi-Dragomir**, and A. Vescan, "Leveraging Rough Sets for Enhanced Test Case Prioritization in a Continuous Integration Context" in 7th Workshop on Validation, Analysis and Evolution of Software Tests (VST 2024), at the 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2024), IEEE, Rovaniemi, Finland, 12 march 2024, **accepted for publication**.
   Indexed IEEE, Workshop at Rank A conference, 4 puncte.

Capitolul 2

Fundamentele teoretice

În acest capitol prezentăm contextul și contribuțiile de ultimă oră pe care le-am valorificat în cercetarea noastră. Acesta oferă mai mult context despre programarea funcțională și concurentă, sistemele multi-agent, învățarea nesupervizată, mulțimile *rough* și, în final, despre testarea automată a software-ului, în special prioritizarea cazurilor de testare. În continuare vom descrie structura acestui capitol în teza de doctorat împărțită în secțiuni.

Secțiunea 1, intitulată *Programarea funcțională concurentă*, prezintă cum funcționează programarea funcțională și concurentă, introduce limbajele de programare și framework-ul web pe care le-am folosit, ne oferă idei despre cum funcționează concurența în acele limbaje și explică de ce am ales aceste tehnologii.

Secțiunea 2, intitulată *Sisteme multi-agent*, descrie sistemele multi-agent, începe cu conceptul de agent, continuă cu tipurile de agenți și în final arată cum agenții pot lucra împreună pentru a atinge diferite obiective, enumerând aplicații concrete.

Secțiunea 3, intitulată *Învățarea supervizată*, definește învățarea supervizată, enumeră unele dintre metodele sale și în final descrie rețelele neuronale în detaliu.

Secțiunea 4, intitulată *Învățarea nesupervizată*, definește învățarea nesupervizată (sau clasificarea nesupervizată), conturează tipurile sale, specifică ce este metoda ierarhică și în final introduce mulțimile *rough* și explică cum pot fi folosite atunci când se aplică pentru clasificarea nesupervizată ierarhică.

Secțiunea 5, intitulată *Mulțimile Rough*, prezintă formal teoria mulțimilor *rough*, urmată de aplicațiile acesteia, în special în domeniul învățării automate, și în final stadiul actual al tehnologiei.

Secțiunea 6, intitulată *Testarea automată a software-ului*, este despre testarea automată a software-ului. Prezintă una dintre cele mai populare metode de testare: testarea de regresie. Apoi descrie problema prioritizării cazurilor de testare.

Capitolul 3

Abordări noi pentru clasificarea nesupervizată prin mulțimi rough

Acest capitol prezintă contribuțiile noastre teoretice la clasificarea nesupervizată folosind mulțimi *rough*. Aceste contribuții includ un algoritm original de clasificare nesupervizată bazat pe sisteme multi-agent și mulțimi *rough* și experimente asupra măsurilor de similaritate pentru această nouă abordare.

Secțiunea 3.1 introduce un algoritm nou de clasificare nesupervizată care utilizează mulțimi *rough* și agenți software pentru a îmbunătăți atât acuratețea pe seturi de date vagi, cât și timpul necesar pentru ca metoda aglomerativă să convergă. Această secțiune propune, de asemenea, o nouă metodologie pentru evaluarea rezultatelor clasificării nesupervizate care au instanțe *rough* sau instanțe atipice (outlier), denumite instanțe hibride.

Secțiunea 3.2 analizează impactul diferitelor metrici de distanță a similarității și metodologia folosită pentru a evalua performanța acestora atât pe baza măsurilor bine cunoscute, cum ar fi acuratețea, cât și într-un mod nou, cum ar fi evaluarea bazată pe instanțe *rough*.

Secțiunea 3.3 își propune să realizeze o comparație detaliată a algoritmului ABARC cu o gamă de algoritmi de învățare supervizată și nesupervizată, folosind o varietate de metrici de performanță pentru a evalua eficiența fiecărui algoritm pe seturi de date utilizate în mod comun.

Secțiunea 3.4 încheie capitolul și prezintă opțiuni pentru lucrări viitoare.

Abordările introduse în acest capitol reprezintă contribuții de cercetare originale publicate în [6, 14, 15, 13].

3.1 Clasificarea nesupervizată rough bazată pe agenți

Conținutul acestei secțiuni este obținut din lucrarea originală [6].

Clasificarea nesupervizată este o sarcină importantă în recunoașterea tiparelor, cu multe aplicații în științele naturale și sănătate. Totuși, în scenariile practice, deseori se întâmplă ca datele să nu poată fi ușor separate în grupuri bine distinse din mai multe motive, cum ar fi: forma grupurilor, prezența outlier-urilor sau problema grupurilor suprapuse (instanțe care pot aparține mai multor grupuri). Pentru a gestiona astfel de probleme, propunem o abordare de clasificare nesupervizată aglomerativă care identifică instanțele ce pot aparține mai multor grupuri și separă clar outlier-urile de restul instanțelor prin integrarea conceptelor din teoria mulțimilor *rough*. Întregul proces de grupare și regrupare este condus de agenți software care rulează în paralel. Abordarea noastră este prietenoasă din punct de vedere computațional, iar experimentele pe seturi de date standard indică avantajele sale.

Această secțiune prezintă abordarea noastră de clasificare nesupervizată. Introducem **ABARC** (**A**gent **BA**sed **R**ough **C**lustering), algoritmi care abordează problema grupurilor suprapuse mo-

delând grupurile folosind noțiuni din teoria mulțimilor *rough*. Aceștia identifică cu succes outlier-urile și, utilizând agenți software, sunt de asemenea scalabili. În plus față de această abordare de clasificare nesupervizată, introducem și ceea ce considerăm a fi o metodologie obiectivă pentru evaluarea calității unui rezultat de clasificare în cazul grupurilor suprapuse și outlier-urilor.

Procedura principală de clasificare nesupervizată este descrisă în Algoritmul 3.1, unde X este setul de date care conține instanțele de clasificat, i_{max} și λ sunt numere întregi care denotă numărul de încercări pentru sarcini specifice (detalii mai jos), σ_1 este limita de similaritate și δ este o metrică de distanță (de exemplu, distanța Euclidiană). Limita de similaritate, σ_1, este specifică fiecărui set de date și este un număr real care denotă valoarea maximă până la care două instanțe sunt considerate similare (ele aparțin cu siguranță aceluiași grup). Primul pas este inițializarea agenților și asocierea unui agent fiecărei instanțe. De asemenea, fiecărui agent i se atribuie un grup diferit, astfel încât, la început, numărul de grupuri este egal cu numărul de agenți, care este egal cu numărul de instanțe. Agenții rulează în paralel, în procese separate, iar acest comportament este indicat de operatorul $\|$ de la linia 4.

Algorithm 3.1: Agent Clustering

Data: $X, i_{max}, \lambda, \sigma_1, \delta$
Result: RC //the set of rough clusters
1 @ Let $\mathcal{AG}$ be the set of agents
2 **for** $i = \overline{1, i_{max}}$ **do**
3 **for** $k = \overline{1, |\mathcal{AG}|}$ **do**
4 $\| \; doCluster(agent_k, \lambda, \sigma_1, \delta, \mathcal{AG})$
5 **end**
6 **end**

Algoritmul 3.2 arată comportamentul asincron al fiecărui agent: dat un $agent_k$, acesta încearcă să găsească colegi similari în raport cu σ_1 și δ prin schimb direct de mesaje. Odată ce găsește un agent similar, se mută în grupul acestuia (linia 3).

Algorithm 3.2: Do Cluster

Data: $agent_k, \lambda, \sigma_1, \delta, \mathcal{AG}$
1 $sa_k = searchForSimilar(agent_k, \lambda, \sigma_1, \delta, \mathcal{AG})$
2 **if** $sa_k \neq null$ **then**
3 $changeCluster(agent_k, sa_k)$
4 **end**

Algoritmul 3.3 descrie procedura de găsire a unui agent similar. Argumentul λ denotă numărul maxim de încercări pe care agentul ar trebui să le efectueze pentru a găsi unul similar. La linia 4, un agent este selectat într-un mod nedeterminist, așa cum este indicat de operatorul $\bigcap$, și, dacă cei doi agenți nu se află în același grup, se calculează $similaritatea$ lor. Dacă această valoare este sub limita de similaritate, σ_1, atunci un agent similar este găsit și funcția se încheie prin returnarea $agentuluiSelectat$. În caz contrar, căutarea continuă și după λ încercări nereușite de a găsi agenți similari (care nu se află în grupul curent), funcția returnează $null$, ceea ce înseamnă că fie nu există agenți similari cu cel dat ($agent_k$), fie procesul a durat prea mult timp, caz în care sarcina este lăsată altor agenți sau unei alte iterații (linia 2 din Algoritmul 3.1) când procesul de căutare este probabil mai rapid, deoarece există mai puține grupuri. Funcția $computeSimilarity$ de la linia 6 calculează similaritatea între doi agenți, având o metrică de distanță δ. Dacă această valoare este

sub limita de similaritate, σ_1, înseamnă că cei doi agenți *sigur* aparțin aceluiași grup, astfel că se află în *aproximația inferioară* a grupului curent.

Algorithm 3.3: Search for Similar

Data: $agent_k, \lambda, \sigma_1, \delta, \mathcal{AG}$
Result: sa //similar agent
1 **if** $\lambda = 0$ **then**
2 | **return** $null$
3 **end**
4 $\bigcap\{selectedAgent = a_j, \forall j = \overline{1, |\mathcal{A}|}, a_j \in \mathcal{AG}\}$
5 **if** $getCluster(agent) \neq getCluster(selectedAgent)$ **then**
6 | $similarity = computeSimilarity(agent, selectedAgent, \delta)$
7 | **if** $similarity \leq \sigma_1$ **then**
8 | | **return** $selectedAgent$
9 | **else**
10 | | **return** $searchForSimilar(agent, \lambda - 1, \sigma_1, \delta, \mathcal{AG})$
11 | **end**
12 **else**
13 | **return** $searchForSimilar(agent, \lambda - 1, \sigma_1, \delta, \mathcal{AG})$
14 **end**

Algoritmul 3.4 reprezintă a doua fază a abordării noastre. Deoarece agenții sunt grupați împreună doar dacă aparțin cu siguranță aceluiași grup (pe baza limitei de similaritate, σ_1), prima fază a abordării noastre (Algoritmii 3.1, 3.2 și 3.3) va produce probabil un număr mare de grupuri. A doua fază (Algoritmul 3.4) unifică grupurile similare, producând grupurile *rough*. Algoritmul primește ca prim argument o mulțime de reprezentanți ale grupurilor. Un *reprezentant al grupului*, $\mathcal{R}k = \langle Ck, \mathcal{A}(C_k)\rangle$, este un tuplu unde $\mathcal{A}(C_k)$ este centroidul grupului C_k și este calculat astfel:

$$\mathcal{A}(C_k) = \frac{1}{|C_k|} \sum_{x^i \in C_k} x^i \tag{3.1}$$

Al doilea parametru, σ_2, este o limită de similaritate *rough* care indică până la ce punct doi agenți sunt *posibil* similari. Acest parametru este diferit de valoarea σ_1 (folosită în Algoritmii 3.1, 3.2 și 3.3) care indică punctul până la care doi agenți sunt *sigur* similari. Ultimul argument, $unified$, reprezintă mulțimea de grupuri unificate și este inițial egal cu mulțimea vidă. Similaritatea grupurilor este calculată pe baza valorilor centroidelor în același mod în care se realizează similaritatea agenților. Dacă un reprezentant este similar cu mai mulți reprezentanți, atunci datele corespunzătoare vor aparține mai multor grupuri în aproximația superioară. Rezultatul este o mulțime de reprezentanți ale grupurilor unificate. Funcția $updateRepresentatives$ de la linia 8 recalculează reprezentanții folosind Ecuația 3.1.

Chiar și după executarea Algoritmului 3.4, ar putea rămâne un număr semnificativ de grupuri, dar majoritatea acestora sunt de obicei compuse dintr-un număr foarte mic de entități care nu sunt similare cu niciunul dintre grupurile "normale". Instanțele din aceste grupuri mici vor fi marcate ca posibili *outliers*. Totuși, în Algoritmul 3.5, a treia fază a abordării noastre, le vom atribui grupului cel mai apropiat și vom obține structura finală de clasificare.

Algoritmul 3.5 primește ca date de intrare $clusters$, așa cum rezultă după aplicarea Algoritmului 3.4. În linia 1, outlier-urile (care sunt și ele grupuri) sunt separate de grupurile "normale". Această decizie se bazează pe valoarea ε și pe dimensiunea grupului: dacă numărul de instanțe

Algorithm 3.4: SimilarClusterUnification

Data: $representatives$, σ_2, $unified$
Result: $unified$
1 **if** $representatives = \emptyset$ **then**
2 | **return** $unified$
3 **end**
4 $\mathcal{R}_k = first(representatives)$
5 $\mathcal{S} = getSimilar(\mathcal{R}_k, representatives \setminus \{\mathcal{R}_k\}, \sigma_2)$
6 **if** $\mathcal{S} \neq \emptyset$ **then**
7 | $updateCluster(\mathcal{R}_k\langle C_k\rangle, \mathcal{S})$
8 | $newRepresentatives = updateRepresentatives(representatives)$
9 | **return** $SimilarClusterUnification(newRepresentatives, \sigma_2, \emptyset)$
10 **else**
11 | **return** $SimilarClusterUnification(representatives \setminus \{\mathcal{R}_k\}, \sigma_2, unified \cup \{R_k\})$
12 **end**

Algorithm 3.5: Outlier Elimination

Data: $clusters$, ε
Result: $finalClusters$
1 $\{outliers, clusters\} = detectOutliers(clusters, \varepsilon)$
2 $finalClusters = joinOutliers(outliers, clusters)$
3 **return** $finalClusters$

dintr-un grup este mai mic de ε, atunci grupul este marcat ca outlier. Valoarea lui ε este setată la 5% din numărul total de instanțe din setul de date. În linia 2, fiecare outlier este unificat cu grupul "normal" cel mai apropiat, pe baza reprezentanților grupelor. O explicație mai cuprinzătoare a algoritmilor, împreună cu metodologia noastră nouă pentru evaluarea grupelor *rough*, poate fi găsită în teză.

Pentru a evalua calitatea abordării noastre, folosim mai multe măsuri de evaluare a grupelor prezentate în teză. Pentru fiecare set de date, executăm algoritmul de 30 de ori și prezentăm valoarea medie pentru fiecare măsură de evaluare în Tabela 3.1. Dorim să menționăm că numărul de execuții a fost ales să fie 30 doar pentru a fi consistenți cu alte abordări cu care ne comparăm. În linia $Official$ din Tabela 3.1 arătăm valorile indicilor pentru grupele date în documentația oficială a setului de date. În linia $ABARC$ arătăm valorile medii ale indicilor pentru grupele obținute de algoritmul nostru, cu toate datele hibride incluse. Linia $ABARC - O$ arată valorile medii ale indicilor pentru grupele noastre după eliminarea outlier-urilor.

Tabela 3.1: Măsuri de calitate a grupurilor.

		$DB \downarrow$	$DN \uparrow$	$SI \uparrow$	$Entropy \downarrow$	$ARI \uparrow$	$Accuracy(\%) \uparrow$
Iris	$Official$	0.511	2.484	0.624	**0**	**1**	**100**
	$ABARC$	0.464	3.012	0.641	0.116	0.77	95.556
	$ABARC - O$	**0.354**	**3.616**	**0.716**	0.067	0.92	97.56
Seeds	$Official$	0.527	2.699	0.561	**0**	**1**	**100**
	$ABARC$	0.463	3.447	0.614	0.293	0.51	90.952
	$ABARC - O$	**0.21**	**5.972**	**0.772**	0.073	0.6	98.121
Wine	$Official$	0.98	1.474	0.465	**0**	**1**	**100**
	$ABARC$	0.968	1.6	0.48	0.11	0.65	96.985
	$ABARC - O$	**0.425**	**3.804**	**0.677**	**0**	0.85	**100**

Așa cum se poate vedea în Tabela 3.1, structura grupului propusă în documentația oficială depășește rezultatele propuse de noi doar în termeni de acuratețe, ARI și entropie (desigur). Cele mai bune rezultate pentru fiecare indice sunt marcate cu bold. Chiar și fără eliminarea outlier-

urilor, toate rezultatele (cu excepția acurateței, ARI și entropiei) sunt mai bune comparativ cu cele oficiale, dar după eliminarea outlier-urilor rezultatele sunt îmbunătățite semnificativ în unele cazuri. Mai multe experimente și evaluări pot fi găsite în teză.

3.2 Metrici de similaritate pentru clasificare nesupervizată prin mulţimi rough

Conținutul acestei secțiuni este obținut din lucrările originale [14, 15].

Selectarea adecvată a măsurii de similaritate poate fi de o importanță deosebită pentru rezultatul clasificării nesupervizate, în special dacă grupurile se suprapun. Prin urmare, scopul acestei secțiuni este de a studia influența unor măsuri de similaritate larg cunoscute asupra structurii de clasificare. Analiza utilizează un algoritm de clasificare nesupervizată bazat pe mulțimi *rough* care este capabil să descopere date hibride (anomalii și instanțe care sunt suficient de apropiate de mai mult de un grup). În această secțiune, examinăm de asemenea influența măsurilor de similaritate asupra zonelor de suprapunere. Deoarece seturile de date standard nu oferă informații specifice referitoare la zonele de suprapunere, propunem și o abordare pentru extragerea acestor date în scopul utilizării lor în scopuri de benchmarking. Experimentele efectuate pe seturi de date standard subliniază importanța selecției adecvate a măsurii de similaritate.

Pentru a evalua influența unei anumite măsuri de similaritate asupra instanțelor aspre, comparăm instanțele aspre raportate de algoritmul descris în Secțiunea 3.1.

Având o măsură de similaritate, executăm algoritmul de mai multe ori (N) pe un anumit set de date și colectăm instanțele aspre raportate din fiecare execuție.

Pentru fiecare instanță aspră x^i, calculăm *rata de apariție*: $occ_{x^i} = \frac{n}{N}$ unde n denotă numărul de apariții ale instanței în seria de N execuții.

Pentru o anumită distanță d, calculăm *scorul rough* conform Definiției 1.

Definition 1 Scorul rough *al unei măsuri de similaritate date este raportul dintre suma* ratelor de apariție *ale instanțelor rough validate și suma tuturor* ratelor de apariție*:*

$$\mathcal{R}_{sc} = \frac{\sum_{i=1}^{|RI_d|} \mathbb{1}\!\!\!\!\mathbb{1}_{RI}(x^i) \cdot occ_{x^i}}{\sum_{i=1}^{|RI_d|} occ_{x^i}},$$

unde:

- RI_d *este mulțimea de instanțe rough raportate de algoritmul de clasificare nesupervizată pentru o anumită măsură de similaritate (sau distanța, d)*
- RI *este mulțimea de instanțe rough produse de algoritmul din Secțiunea 3.1*
- $\mathbb{1}\!\!\!\!\mathbb{1}_{RI}$ *este funcția indicator a lui* RI*:*

$$\mathbb{1}\!\!\!\!\mathbb{1}_{RI}(x^i) = \begin{cases} 1, & \text{if } x^i \in RI \\ 0, & \text{otherwise.} \end{cases}$$

Așa cum se vede în Definiția 1, *scorul rough* depinde de un parametru fix κ care denotă încrederea fiecărei instanțe *rough* x^i — astfel, sunt luate în considerare doar instanțele care au o *rată de*

apariție mai mare decât κ. Dată fiind o măsură de similaritate, calculăm *scorul rough* pentru mai multe valori ale κ. Restul metodologiei noastre se poate găsi în teza de doctorat.

Vorbind despre rezultate, *scorurile rough* pot fi găsite în Tabela 3.2. Se pare că cele mai bune rezultate (*scoruri rough* ridicate pentru valori mari de încredere) sunt obținute în majoritatea cazurilor pentru valori ale lui p apropiate de 2.3. De exemplu, pentru Wine și Seeds cele mai bune rezultate sunt obținute pentru $p = \sqrt{2}$, în timp ce pentru celelalte seturi de date $p = 2\sqrt{2}$ dă cele mai bune rezultate, dar aceste valori ale lui p sunt ambele aproape de 2.3. Alte rezultate pot fi găsite în teză.

Tabela 3.2: Scoruri Rough cu încrederi minime (κ) variind între 0 și 1 cu un pas de 0.1.

	Similarity measure	***0***	***0.1***	***0.2***	***0.3***	***0.4***	***0.5***	***0.6***	***0.7***	***0.8***	***0.9***	***1***
Iris	Manhattan	2.61	11.0	25.5	25.5	25.5	20.0	20.0	NaN	NaN	NaN	NaN
	Chebyshev	6.36	9.5	13.13	14.31	19.5	11.0	11.0	0.0	0.0	0.0	0.0
	Euclidean	5.9	38.5	38.5	38.5	38.5	51.33	51.33	45.0	45.0	45.0	0.0
	Squared Euclidean	14.88	15.25	18.39	18.39	59.33	89.0	89.0	89.0	89.0	98.0	NaN
	Minkowski $p = \sqrt{2}$	5.56	36.0	36.0	36.0	36.0	48.0	46.0	46.0	46.0	46.0	0.0
	Minkowski $p = 2.3$	15.71	18.17	27.25	43.6	43.6	54.5	80.0	80.0	86.0	NaN	NaN
	Minkowski $p = 2\sqrt{2}$	7.61	18.88	25.17	31.71	62.0	62.0	62.0	62.0	93.0	98.0	NaN
	Minkowski $p = 3$	6.12	16.0	17.33	29.71	59.33	59.33	59.33	89.0	89.0	96.0	NaN
Wine	Manhattan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Chebyshev	7.73	11.94	20.24	20.0	20.29	33.33	33.33	33.33	50.0	100.0	100.0
	Euclidean	5.36	8.0	10.47	15.33	23.0	0.0	0.0	0.0	0.0	NaN	NaN
	Squared Euclidean	3.65	6.83	10.91	6.4	0.0	0.0	0.0	NaN	NaN	NaN	NaN
	Minkowski $p = \sqrt{2}$	5.72	8.66	10.63	11.47	14.0	16.55	10.57	12.33	0.0	NaN	NaN
	Minkowski $p = 2.3$	4.99	8.87	9.39	10.67	0.0	0.0	0.0	0.0	0.0	NaN	NaN
	Minkowski $p = 2\sqrt{2}$	4.03	6.33	8.29	8.5	0.0	0.0	0.0	0.0	NaN	NaN	NaN
	Minkowski $p = 3$	5.15	7.04	10.74	23.33	16.0	0.0	0.0	0.0	0.0	NaN	NaN
Seeds	Manhattan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Chebyshev	1.74	2.51	7.4	4.75	0.0	0.0	0.0	0.0	NaN	NaN	NaN
	Euclidean	3.62	4.87	10.34	17.5	18.62	19.0	15.0	NaN	NaN	NaN	NaN
	Squared Euclidean	4.49	6.94	8.4	8.64	27.56	41.6	52.0	86.0	86.0	NaN	NaN
	Minkowski $p = \sqrt{2}$	4.97	7.8	9.35	13.52	17.45	14.86	22.29	52.0	40.0	0.0	0.0
	Minkowski $p = 2.3$	3.58	6.07	11.64	21.62	25.08	37.6	33.5	35.0	NaN	NaN	NaN
	Minkowski $p = 2\sqrt{2}$	4.09	7.01	11.74	20.46	24.11	29.56	41.0	41.0	31.33	31.33	0.0
	Minkowski $p = 3$	3.41	4.73	7.04	8.38	18.92	7.78	14.0	14.0	0.0	0.0	NaN
Ionosphere	Manhattan	2.0	1.27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	NaN
	Chebyshev	14.5	19.33	19.33	19.33	29.0	29.0	NaN	NaN	NaN	NaN	NaN
	Euclidean	2.65	6.45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	NaN
	Squared Euclidean	1.85	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	NaN
	Minkowski $p = \sqrt{2}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN
	Minkowski $p = 2.3$	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN	NaN
	Minkowski $p = 2\sqrt{2}$	2.29	37.0	37.0	37.0	74.0	74.0	74.0	74.0	NaN	NaN	NaN
	Minkowski $p = 3$	5.0	8.24	0.0	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN
Ecoli	Manhattan	3.05	7.04	6.62	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NaN
	Chebyshev	4.64	22.93	47.0	52.0	56.0	56.0	56.0	48.0	48.0	48.0	NaN
	Euclidean	2.88	8.98	11.46	13.45	11.14	15.6	26.0	78.0	NaN	NaN	NaN
	Squared Euclidean	3.96	10.48	14.16	18.22	27.2	56.0	NaN	NaN	NaN	NaN	NaN
	Minkowski $p = \sqrt{2}$	3.23	8.57	13.83	28.4	47.33	47.33	47.33	37.0	NaN	NaN	NaN
	Minkowski $p = 2.3$	4.61	12.11	16.35	17.87	19.0	22.4	0.0	NaN	NaN	NaN	NaN
	Minkowski $p = 2\sqrt{2}$	3.31	9.31	13.43	19.5	26.0	39.0	78.0	78.0	NaN	NaN	NaN
	Minkowski $p = 3$	2.23	9.5	10.36	10.29	24.0	24.0	24.0	36.0	NaN	NaN	NaN
Glass	Manhattan	10.0	34.0	34.0	34.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Chebyshev	8.0	16.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Euclidean	6.44	34.0	34.0	34.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Squared Euclidean	9.2	34.0	34.0	34.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Minkowski $p = \sqrt{2}$	8.53	25.5	25.5	36.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Minkowski $p = 2.3$	11.67	17.0	34.0	34.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Minkowski $p = 2\sqrt{2}$	50.0	50.0	50.0	80.0	80.0	80.0	80.0	80.0	80.0	NaN	NaN
	Minkowski $p = 3$	35.0	35.0	52.0	52.0	72.0	72.0	72.0	72.0	NaN	NaN	NaN

3.3 Evaluarea clasificării nesupervizate bazate pe mulțimi rough

Conținutul acestei secțiuni este obținut din lucrarea originală [13].

Această secțiune își propune să efectueze o comparație cuprinzătoare a algoritmului ABARC față de mai mulți algoritmi de învățare supervizată și nesupervizată, folosind o suită de metrici de performanță pentru a evalua eficacitatea fiecărui algoritm pe seturi de date standard. Prin această analiză comparativă, ne propunem să arătăm punctele forte și limitările algoritmului ABARC și ale omologilor săi, contribuind astfel la cercetarea continuă privind abordările optime de clasificare nesupervizată în contextul incertitudinii a datelor.

Această analiză cuprinzătoare include trei tipuri de metrici: metrici de evaluare externi, interni și *rough*. Cele externe pe care le-am considerat sunt: Acuratețea, Precizia, Recall, F1-Score, Macro F1-Score, Weighted Average F1-Score, Micro F1-Score și Kappa Score. Comparațiile folosind acești metrici au fost realizate pe seturile de date Iris, Seeds și Wine. Cele interne utilizate sunt: Puritatea, Entropia, V-Measure. Pentru comparații am folosit din nou aceleași trei seturi de date. În final, avem următoarele metrici *rough*: Acuratețea medie (indicele α), Roughness medie (Indicele ρ), Acuratețea aproximării (indicele α^*) și Calitatea aproximării (indicele γ). Aceste metrici de evaluare au fost utilizate doar pe seturile de date Iris și Wine, deoarece nu am avut subiecte de comparație pentru setul de date Seeds.

Luând în considerare rezultatele, în Tabela 3.3 putem observa că pe Iris acuratețea și puritatea scad puțin pe măsură ce eliminăm outlierii și *rough*, dar entropia și V-Measure după eliminarea ambelor sunt semnificativ mai bune, ceea ce ne face să presupunem că outlierii și instanțele *rough* nu afectează într-adevăr omogenitatea, dar afectează completitudinea. Pe setul de date Seeds nu putem observa nicio diferență reală atunci când îi eliminăm, astfel că aceștia nu afectează performanța noastră. În final, pe Wine putem vedea că toate metricile se îmbunătățesc, entropia și V-Measure se îmbunătățesc semnificativ, deci pe acest set de date eliminarea lor face ca rezultatele noastre să fie aproape perfecte indiferent de metrica utilizată. Restul măsurătorilor de performanță pot fi găsite în teză.

Tabela 3.3: Măsurători de performanță a metodelor nesupervizate pentru seturile de date Iris, Seeds și Wine.

	Case Study	Inst	Acc	Entropy	Purity	V
Iris	Clusters with hybrids	150	98.66%	0.0803	0.987	0.733
	Clusters without outliers	139	98.56%	0.0847	0.986	0.717
	Clusters without outliers and rough	126	98.41%	0.0204	0.984	0.932
Seeds	Clusters with hybrids	210	92.857%	0.0839	0.929	0.721
	Clusters without outliers	190	92.105%	0.0863	0.921	0.711
	Clusters without outliers and rough	178	91.573%	0.0829	0.916	0.719
Wine	Clusters with hybrids	178	97.753%	0.0569	0.978	0.8
	Clusters without outliers	157	99.363%	0.0418	0.994	0.854
	Clusters without outliers and rough	148	99.324%	0.0088	0.993	0.97

3.4 Concluzii

În acest capitol, am introdus un algoritm de clasificare nesupervizată aglomerativ care utilizează teoria mulțimilor *rough* pentru a identifica date hibride, cum ar fi outlierii sau instanțele

rough. Datele hibride includ instanțe cu trăsături ale mai multor grupuri, oferind perspective mai profunde decât alte metode. Identificarea outlierilor ajută analiștii să elimine erorile potențiale de măsurare, îmbunătățind acuratețea clasificării. Instanțele *rough* pot indica noi clase sau indivizi mutanți în seturi de date precum Iris. Algoritmul nostru scalabil rulează rapid pe computere standard cu seturi mari de date.

De asemenea, am prezentat o metodologie obiectivă pentru a evalua calitatea datelor hibride, depășind multe măsuri tradiționale de calitate a grupurilor. Metodologia noastră de validare evidențiază performanța algoritmului nostru în raport cu informațiile documentate ale seturilor de date. În plus, am examinat impactul diferitelor măsuri de similaritate asupra acurateței clasificării nesupervizate și a instanțelor *rough*, propunând o metodă de determinare a instanțelor *rough* pentru benchmarking.

Algoritmul nostru depășește alte tehnici de clasificare în ceea ce privește Acuratețea, Entropia și Indexul Rand Ajustat, în special după eliminarea outlierilor. Rezultatele demonstrează performanța robustă a algoritmului și capacitatea sa unică de a identifica date hibride. Am evaluat extensiv algoritmul de clasificare nesupervizată prin mulțimi *rough* bazat pe agenți (ABARC) folosind seturi de date standard și l-am comparat cu multiple metode, analizând efectele diferitelor metrici în contexte imprevizibile.

Experimentele arată că distanța Minkowsky cu $p = 2.3$ este optimă pentru acuratețea și instanțele *rough*, probabil datorită formelor sferice similare ale grupurilor în toate seturile de date. Eliminarea hibridelor îmbunătățește performanța ABARC, în special pe seturile de date Iris și Wine, accentuând importanța detectării datelor hibride în medii incerte. Constatările subliniază, de asemenea, necesitatea unor metrici de validare adecvate pentru clasificarea nesupervizată în scenarii vagi sau imprevizibile.

Viitoarele lucrări includ experimentarea cu seturi de date care prezintă regiuni de suprapunere mai mari și forme variate de grupuri, precum și concentrarea pe gestionarea outlierilor prin validarea rezultatelor și examinarea influenței selecției măsurii de similaritate.

Capitolul 4

Contribuții la prioritizarea cazurilor de testare în contextul integrării continue

Acest capitol prezintă cercetarea noastră asupra prioritizării cazurilor de testare, și contribuțiile noastre utilizând o abordare bazată pe rețele neuronale pe seturi de date industriale și integrarea acesteia într-o soluție aplicată pentru un proiect real.

Secțiunea 4.1 descrie abordarea noastră pentru prioritizarea cazurilor de testare folosind rețele neuronale (NEUTRON), precum și aplicația și performanța sa pe seturi de date industriale, cum ar fi Google Shared Dataset of Test Suite Results.

Secțiunea 4.2 prezintă aplicarea și evaluarea abordării NEUTRON într-un proiect industrial real.

Secțiunea 4.3 încheie capitolul și menționează potențialele noastre lucrări viitoare.

Abordările introduse în acest capitol reprezintă contribuții de cercetare originale publicate în [18, 16].

4.1 Prioritizarea cazurilor de testare bazată pe rețele neurale în integrarea continuă

Conținutul acestei secțiuni este obținut din lucrarea originală [18].

În mediile de integrare continuă, execuția cazurilor de testare se realizează pentru fiecare caracteristică nou adăugată sau atunci când apare o corecție de bug. Prin urmare, testarea de regresie se efectuează considerând diverse strategii de testare. Abordarea Prioritizării Cazurilor de Testare (TCP) ia în considerare reordonarea cazurilor de testare astfel încât defectele să fie găsite mai devreme, cu un cost minim de execuție.

Scopul acestei secțiuni este să investigheze impactul modelelor de clasificare bazate pe rețele neuronale pentru a asista în prioritizarea cazurilor de testare. Sunt utilizate trei modele diferite cu diverse caracteristici (durată, rata de defecte, numărul de cicluri, numărul total de rulări) și se ia în considerare informația la fiecare 30 de cicluri sau la fiecare 100 de cicluri.

Rezultatele obținute subliniază că abordarea NEUTRON găsește o prioritizare mai bună în ceea ce privește NAPFD (procentul mediu normalizat al defectelor detectate) decât permutarea aleatoare și este comparabilă cu soluțiile care au folosit fie durata, fie defectele, considerând că le combină pe ambele. Comparativ cu alte abordări existente, NEUTRON obține rezultate competitive similare când se consideră un buget de 50% și cele mai bune rezultate când se consideră bugete de 75% și 100%.

După prezentarea stadiului actual al prioritizării cazurilor de testare, prezentăm diferitele noastre abordări, urmate de descrierea diferitelor modele de rețele neuronale (văzute în Figura 4.1) pe care le folosim și designul experimentelor (văzut în Figura 4.2). Am considerat trei seturi

de date industriale: două de la ABB Robotics Norvegia [1] (Paint Control și IOF/ROL, pentru testarea roboților industriali complecși) și Google Shared Dataset of Test Suite Results (GSDTSR) [2]. În ceea ce privește metricile pentru analiză, am folosit NAPFD, care se poate adapta mai bine la cazurile de utilizare în care nu toate defectele de testare sunt găsite. Etichetarea datelor de antrenament se face printr-un agent inteligent, care poate simula experiența și cunoștințele unui expert în acest domeniu. O descriere și explicație mai detaliată a abordărilor noastre se găsește în teză.

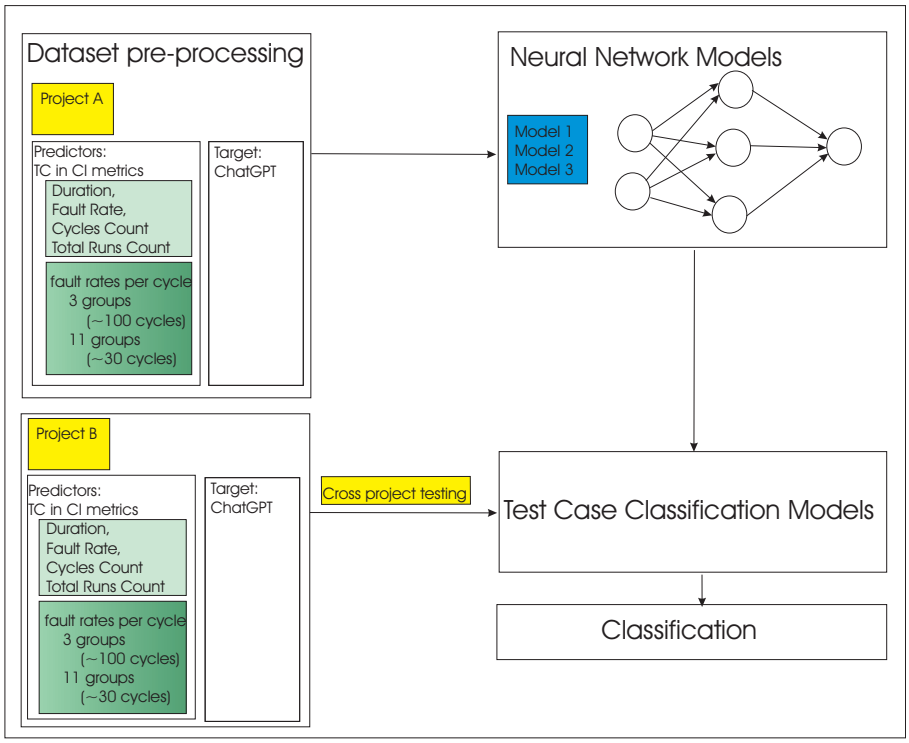

Figura 4.1: Prezentare generală a modelelor bazate pe rețele neuronale pentru TCP în Integrarea Continuă.

Pentru toate experimentele, antrenamentul s-a realizat în proiectul *IOF/ROL* (deoarece a fost cel mai echilibrat set de date dintre cele disponibile) și testarea s-a realizat în proiectele *Paint Control* și *GSDTSR*.

Experiment 1

Experimentul curent ia în considerare pentru fiecare caz de testare cele patru caracteristici generale menționate mai sus, împreună cu informațiile despre defecte pentru fiecare ciclu. Rețeaua neuronală folosește datele furnizate la fiecare 30 de cicluri.

După cum se poate observa în Figura 4.3, cele mai bune rezultate, în cazul bugetului de 50%, sunt obținute de RETECTS [12] și COLEMAN [8]. Trebuie menționat că NEUTRON obține rezultate mai bune decât soluțiile *Random* și, de asemenea, mai bune decât *Sorted by Duration* și *Sorted by Fault Rate* în fiecare dintre proiectele testate, însă soluția NEUTRON încorporează ambele caracteristici privind durata și rata de defecte.

În Figura 4.4, se arată că NEUTRON obține cea mai bună soluție pentru bugetul de 75% în cazul testării *Paint Control* și pentru ambele proiecte de testare în cazul bugetului de 100%. Tabela 4.1 conține valorile NAPFD obținute pentru lucrările anterioare: abordarea lui Elbaum [2],

[1] `https://new.abb.com/products/robotics`
[2] `https://bitbucket.org/HelgeS/atcs-data/src`

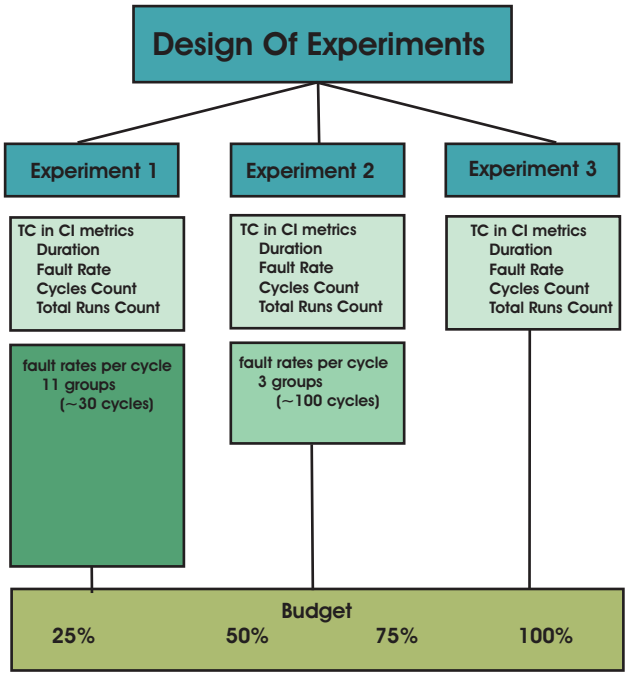

Figura 4.2: Proiectarea experimentelor

RETECTS [12], COLEMAN [8] și rezultatele pentru abordarea *NEUTRON*, împreună cu rezultatele pentru permutarea inițială, random și sortate după durată sau defecte. Trebuie menționat că valorile complete există doar pentru bugetul de 50%. Pentru 25% și 75%, am considerat în tabel rezultatele soluțiilor anterioare pentru 10% și respectiv 80%. Restul experimentelor pot fi găsite în teză.

Tabela 4.1: Experimentul 1

Project	NAPFD							
	Initial	Random	Sort Fault Rate	Sort Duration	NEUTRON	Elbaum	RETECS	COLEMAN
25%-Paint Control	0.275280	0.320224	0.067415	0.567415	0.251276		0.915	0.915
25%-GSDTSR	0.160602	0.208414	0.798651	0.692944	0.486858		0.9911	0.9893
50%-Paint Control	0.584485	0.589887	0.372999	0.780898	0.600487	0.9145	0.915	0.915
50%-GSDTSR	0.409727	0.486987	0.999755	0.882094	0.917503	0.9891	0.9911	0.9893
75%-Paint Control	0.666328	0.864736	0.700434	0.949570	0.937796		0.9162	0.9171
75%-GSDTSR	0.672442	0.697674	0.999878	0.945154	0.987035		0.9921	0.9893
100%-Paint Control	0.988700	0.980494	1	0.972667	0.994003			
100%-GSDTSR	0.982327	0.974485	0.999904	0.959203	0.998308			

4.2 MixTCP: o abordare pentru îmbunătățirea experienței dezvoltării software

Conținutul acestei secțiuni este obținut din lucrarea originală [16].

Prioritizarea cazurilor de testare (TCP) este crucială în lumea rapidă a dezvoltării software pentru a accelera și optimiza procedurile de testare, în special în configurațiile de integrare continuă (CI). Această secțiune își propune, în primul rând, să valideze un model de rețea neurală de ultimă generație pentru TCP în mediile CI, aplicându-l într-un context industrial real și, în al doilea rând, să propună MixTCP, o soluție aplicată care integrează modelul de rețea neurală și îmbunătățește semnificativ experiența testării de regresie din perspectiva dezvoltatorului de software. Este implementat în limbajul de programare Elixir și utilizează modelul NEUTRON, o abordare de ultimă generație care folosește rețele neuronale pentru a prioritiza inteligent cazu-

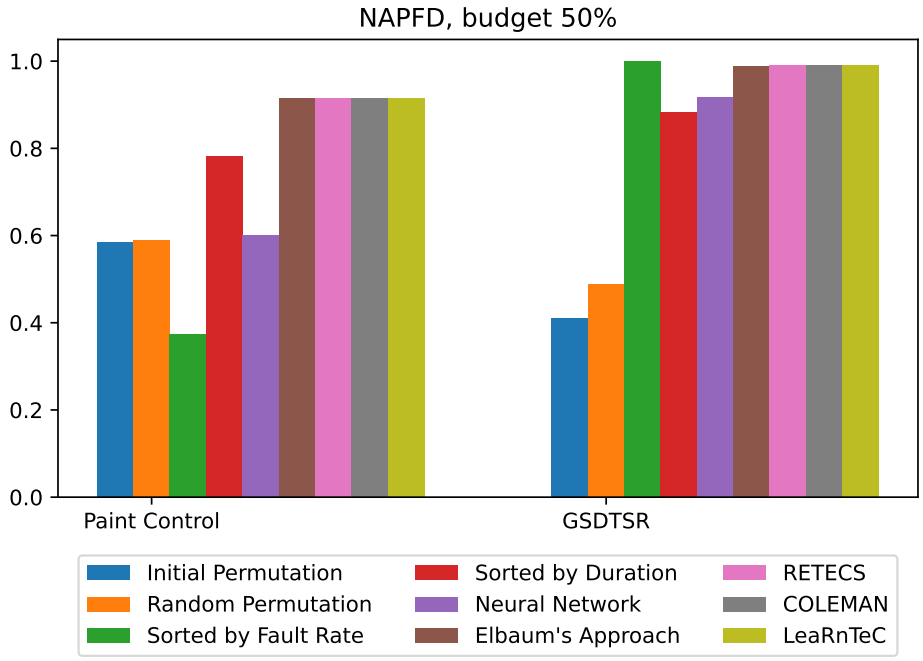

Figura 4.3: Experimentul 1 cu aproximativ 11x30 cicluri, considerând un buget de 50%

rile de testare, îmbunătățind efectiv detectarea defectelor și reducând timpul de testare. Sistemul este compus din componente slab cuplate (task-ul Mix TCP, serverul TCP și modelul NEUTRON, așa cum se vede în Figura 4.5), permițând astfel integrarea altor soluții de prioritizare a cazurilor de testare. Rezultatele arată că MixTCP are potențialul de a fi un atu valoros pentru metodele moderne de dezvoltare software, oferind inginerilor software o abordare TCP mai eficientă, mai prietenoasă cu utilizatorul și mai ușor de integrat.

În primul rând, prezentăm motivația noastră și stadiul actual al artei, apoi introducem abordarea MixTCP (așa cum se vede și în Figura 4.5). Am aplicat TCP și NEUTRON (așa cum este descris în Secțiunea 4.1) pe un proiect real Mix [21]. În părțile următoare detaliem arhitectura noastră, specificăm cum să utilizați acest software și discutăm avantajele acestuia, care sunt pe scurt: integrarea cu NEUTRON (o abordare de ultimă generație pentru TCP), scalabilitatea, validarea empirică în medii industriale, experiența utilizatorului și integrarea flexibilă. Toate acestea pot fi văzute în detaliu în teză.

Am proiectat un experiment care ia în considerare diverse cicluri de execuție, așa cum este specificat în următoarele și așa cum se vede în Figura 4.6.

Pașii experimentului sunt:

- testele au fost executate de mai multe ori, pe parcursul a 8 cicluri
- fiecare ciclu este o execuție unică a unui subset de teste
- pentru primele 3 cicluri am executat toate testele, fără a folosi MixTCP
- începând cu al patrulea ciclu, am început să folosim MixTCP, care a prioritizat testele prin analizarea datelor din toate ciclurile anterioare, astfel că am efectuat doar un subset selectat din aceste teste, alegându-le pe baza ordinii lor de prioritate
- pentru a calcula NAPFD, luăm rulările testelor din ciclul următor pentru a decide verdictul fiecărui fișier de test
- NAPFD este calculat pentru ciclurile între 3 și 7, NAPFD-ul ciclului 3 va indica cât de bine au fost prezise testele în ciclul 4

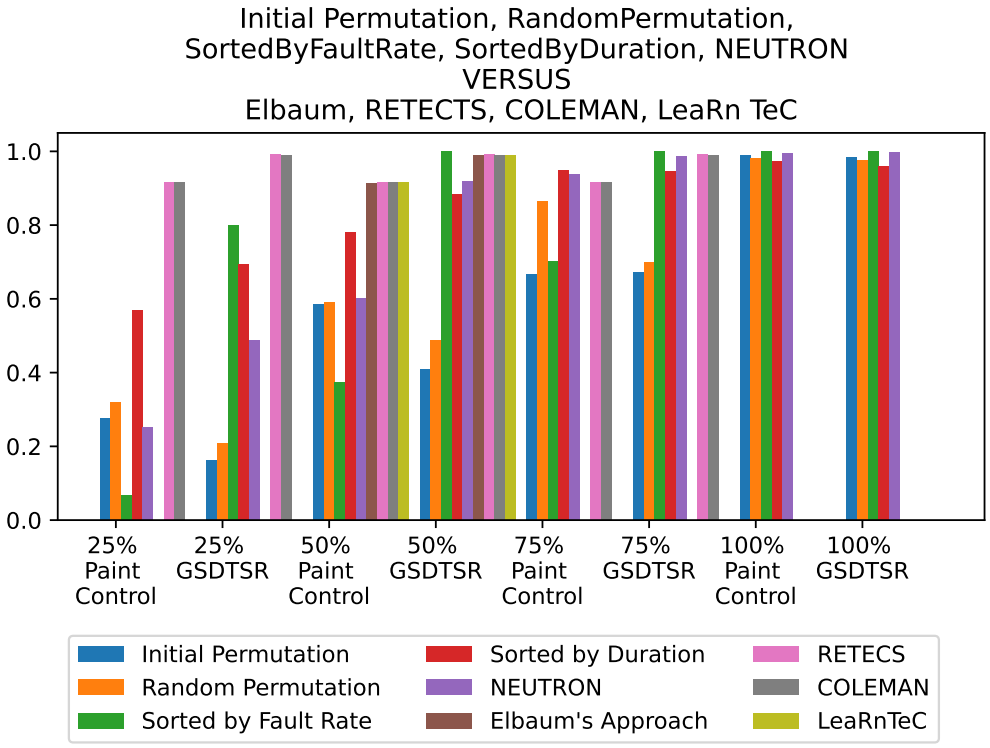

Figura 4.4: Experimentul 1 cu aproximativ 11x30 cicluri, considerând toate bugetele

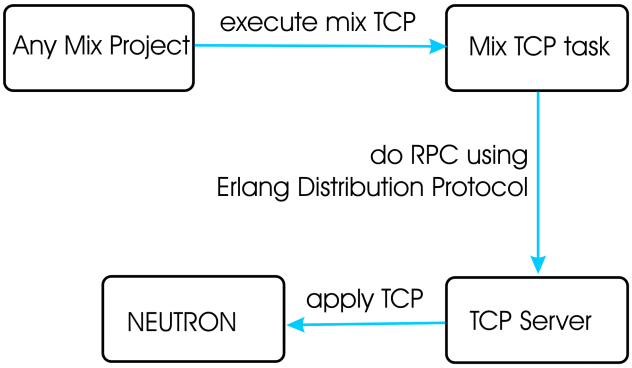

Figura 4.5: Prezentare generală a abordării MixTCP

- deoarece există doar 8 cicluri, nu se poate calcula un scor pentru ciclul 8 încă
- au fost expuse 8 defecte pe parcursul experimentului

Pentru a măsura rezultatele, am folosit aceeași metrică ca în abordarea NEUTRON (din Secțiunea 4.1), și anume NAPFD. Această metrică oferă o imagine de ansamblu suficientă asupra performanței generale a unei abordări TCP. Rezultatele sunt prezentate în Tabela 4.2. Așa cum se poate observa, performanța soluției este considerabil mai bună decât rularea testelor într-un mod aleatoriu (care este comportamentul implicit în Elixir).

Figura 4.7 ilustrează grafic rezultatele pentru Random și NEUTRON pe parcursul ciclurilor, arătând clar o îmbunătățire notabilă a performanței pentru ambele.

Mai multe detalii despre acest experiment pot fi găsite în teză.

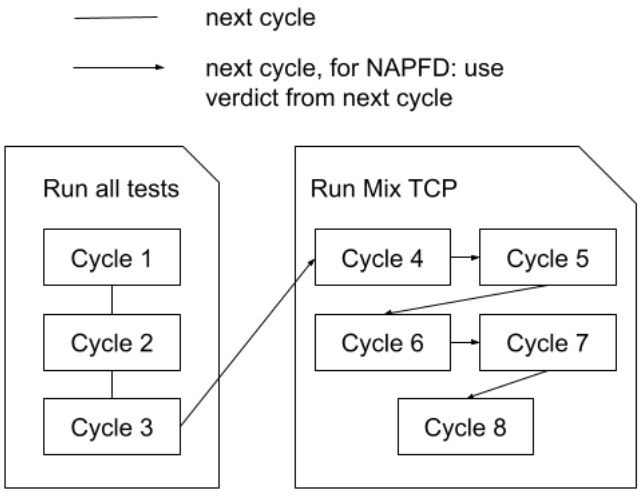

Figura 4.6: Ciclurile din Experiment.

Tabela 4.2: NAPFD Random și NEUTRON pe ciclu

Cycle	Random NAPFD	NEUTRON NAPFD
3	5.82%	12.18%
4	13.15%	22.25%
5	18.81%	35.06%
6	32.16%	48.20%
7	45.86%	81.46%

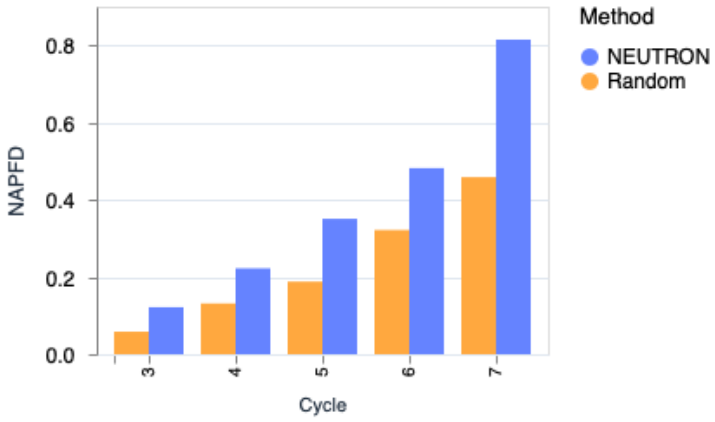

Figura 4.7: NAPFD per ciclu

4.3 Concluzii

În setările de integrare continuă, o suită de teste este rulat pentru fiecare caracteristică nouă sau corecție de bug. Există mai multe strategii pentru a selecta sau prioritiza execuția testelor. Prioritizarea Cazurilor de Testare reordonează cazurile de testare astfel încât defectele să fie găsite mai devreme, cu un cost minim de execuție.

Acest capitol a investigat utilizarea modelelor de clasificare bazate pe rețele neuronale pentru a ajuta la prioritizarea testelor, precum și integrarea acestora în MixTCP, o soluție aplicată pentru proiecte reale. Au fost utilizate trei modele diferite cu diverse caracteristici (durată, rata de defecte, numărul de cicluri, numărul total de rulări) și s-a luat în considerare informația la fiecare 30 de cicluri sau la fiecare 100 de cicluri.

Abordarea NEUTRON găsește o prioritizare mai bună în ceea ce privește NAPFD decât permutarea aleatorie. Rezultatele NEUTRON sunt comparabile cu alte soluții bazate pe sortare care au folosit fie durata, fie defectele; în plus, consideră ambele caracteristici atunci când construiește soluția. Comparativ cu alte abordări de ultimă generație existente, NEUTRON obține rezultate competitive similare când se consideră un buget de 50% și cele mai bune rezultate când se consideră bugete de 75% și 100%.

Evaluarea MixTCP în medii industriale reale validează beneficiile sale în îmbunătățirea procesului TCP. De asemenea, validează abordarea teoretică, NEUTRON. Soluția a arătat o îmbunătățire clară în prioritizarea cazurilor de testare, ceea ce duce în cele din urmă la detectarea mai timpurie a defectelor și, de asemenea, la utilizarea mai eficientă a resurselor de testare. Aceasta are implicații semnificative pentru viteza și calitatea dezvoltării software, deoarece permite lansări mai rapide și produse software mai fiabile.

Eforturile noastre viitoare vor fi direcționate către experimente suplimentare care vor fi efectuate cu mai multe proiecte și cu scenarii mai complexe, luând în considerare diverse cicluri și mai multe cazuri de testare care descoperă aceleași defecte, împreună cu legătura cu schimbările în codul sursă sau în cerințe. În plus, dorim să ne asigurăm că MixTCP se integrează cu succes în orice mediu de dezvoltare, planificăm de asemenea să integrăm alte abordări de TCP de ultimă generație în acesta, facilitând adoptarea mai rapidă de către industria software.

Capitolul 5

Clasificarea nesupervizată prin mulțimi rough pentru prioritizarea cazurilor de testare

Acest capitol prezintă aplicarea algoritmului nostru inovator de clasificare nesupervizată în contextul prioritizării cazurilor de testare și propune o abordare modernă și unificată a prioritizării cazurilor de testare validată pe un set de date sintetic folosind abordarea noastră de clasificare nesupervizată.

Secțiunea 5.1 aplică metoda noastră de clasificare nesupervizată *rough* pe seturi de date industriale în contextul prioritizării cazurilor de testare în integrarea continuă.

Secțiunea 5.2 ilustrează o idee nouă și originală privind prioritizarea cazurilor de testare, care unește toate conceptele bine cunoscute într-un singur set de date. Mai mult, validează eficacitatea acestei noi abordări folosind metoda noastră de clasificare nesupervizată.

Secțiunea 5.3 încheie capitolul și descrie lucrările noastre viitoare.

Abordările introduse în acest capitol reprezintă contribuții de cercetare originale publicate în [5, 19].

5.1 RoughTCP: o abordare privind TCP în CI bazată pe clasificare nesupervizată prin mulțimi rough

Conținutul acestei secțiuni este obținut din lucrarea originală [5].

În peisajul în continuă evoluție al Integrării Continue (CI), execuția cazurilor de testare devine esențială cu fiecare modificare de cod, făcând strategiile de testare de regresie indispensabile. Printre acestea, Prioritizarea Cazurilor de Testare (TCP) a devenit o metodă populară pentru a îmbunătăți eficiența și eficacitatea testării software. Recent, cercetătorii au analizat în principal metode de învățare supervizată, cum ar fi rețelele neuronale, pentru a aborda TCP în CI. Cu toate acestea, datorită naturii dinamice a acestor medii, ar putea fi util să explorăm abordări nesupervizată care se pot adapta la incertitudinile inerente fără date etichetate. Această secțiune propune RoughTCP, o abordare inovatoare care utilizează un algoritm de clasificare nesupervizată aglomerativ bazat pe mulțimi *rough*, pentru a prioritiza cazurile de testare.

RoughTCP grupează și clasează automat testele pe baza tiparelor și corelațiilor lor intrinseci (de exemplu, defecte, durata testelor, numărul de cicluri și numărul total de rulări) fără un model predefinit. Acest lucru îmbunătățește detectarea defectelor fără necesitatea unei supervizări constante și oferă o înțelegere mai cuprinzătoare a rezultatelor prin incorporarea mulțimilor *rough*. Au fost efectuate trei grupuri de experimente, luând în considerare date din contexte de integrare continuă în proiecte industriale.

Comparativ cu lucrările recente, experimentele noastre arată că abordarea RoughTCP produce rezultate mai bune pentru bugete mai mari sau egale cu 75% pe toate seturile de date, în timp ce uneori depășește toate celelalte metode și pe bugete mai mici. Acest lucru subliniază potențialul

metodelor nesupervizate și, în special, capabilitatea lui RoughTCP de a revoluționa stadiul actual al TCP în mediile de integrare continuă (CI).

Secțiunea începe prin prezentarea problemei prioritizării cazurilor de testare și a contextului (care este același ca în Secțiunea 4.1), alături de soluțiile de ultimă generație. În continuare, schițează abordarea noastră *RoughTCP* (așa cum se vede în Figurile 5.1 și 5.2). Secțiunea continuă cu detalierea algoritmului de clasificare nesupervizată *rough* pe care l-am folosit pentru TCP. Mai multe detalii pot fi găsite în teză.

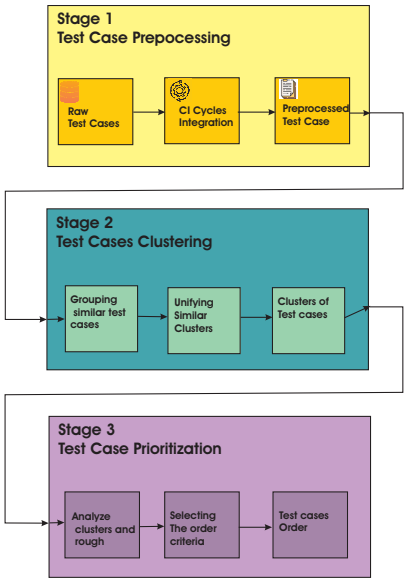

Figura 5.1: Etapele abordării

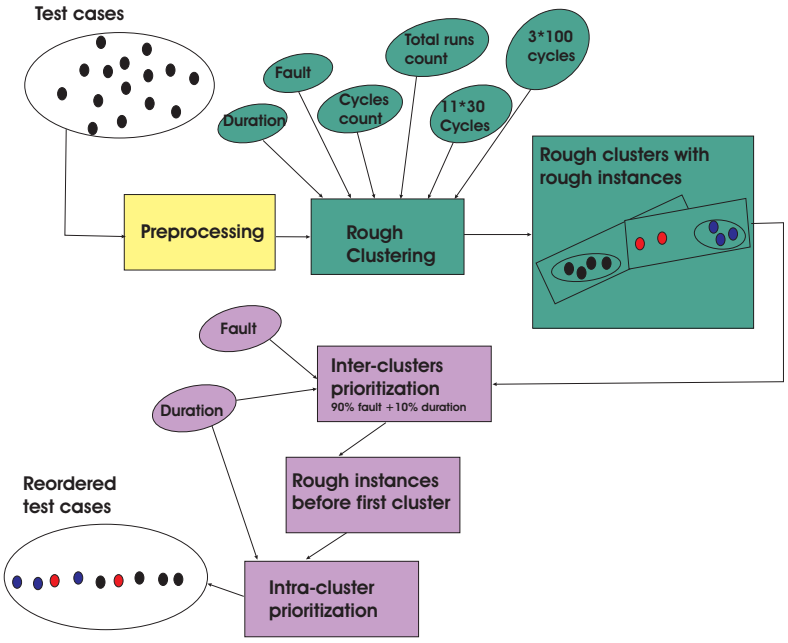

Figura 5.2: Prezentare generală a abordării.

Rezultatele experimentelor efectuate utilizând algoritmul de clasificare nesupervizată *rough* sunt prezentate în Tabela 5.1, împreună cu rezultatele altor abordări de la Elbaum [2], la RETECS [12], COLEMAN [8] și NEUTRON.

Investigațiile noastre utilizând experimentele bazate pe clasificarea nesupervizată *rough* (RC)

Tabela 5.1: Experimente

Project	Budget	NAPFD								
		Elbaum	RETECS RNFail	RETECS Time-Rank	COLEMAN RNFail	COLEMAN TimeRank	NEUTRON	RC-Reduced	RC-3x100-cycles	RC-11x30-cycles
Paint Control	10		**0.9078**	**0.9077**	**0.9076**	**0.9076**		0.1605	0.0393	0.0421
	25						0.2512	**0.5104**	0.4877	0.4767
	50	**0.9145**	**0.915**	0.9138	**0.915**	**0.915**	0.6004	0.7579	0.7352	0.7354
	75						0.9377	**0.9490**	0.9264	0.9265
	80		0.9162	0.9160	0.9171	0.9171		**0.9827**	0.9264	0.9265
	100						0.9940	0.9940	**0.9946**	**0.9947**
GSDTSR	10		**0.9893**	**0.9893**	**0.9894**	**0.9894**		0.8904	0.8904	0.8969
	25						0.4868	**0.9533**	**0.9533**	**0.9551**
	50	0.9891	**0.9911**	0.9906	0.9893	0.9894	0.9175	0.9880	0.9880	0.9880
	75						0.9870	**0.9967**	**0.9967**	**0.9965**
	80		0.9921	0.9914	0.9893	0.9894		**0.9977**	**0.9976**	**0.9974**
	100						0.9983	**0.9992**	**0.9992**	**0.9990**
IOF/ROL	10		**0.3704**	**0.3779**	0.3632	0.3670		0.1578	0.1004	0.1939
	25							0.4330	0.3802	**0.4718**
	50	0.4892	0.5101	0.5025	0.5046	0.5189		**0.7909**	0.7762	**0.8125**
	75							0.8784	**0.9019**	**0.8964**
	80		0.5495	0.5287	0.5569	0.5678		0.8874	**0.9107**	**0.9026**
	100							0.9180	**0.9318**	0.9199

sunt: RC-Reduced (utilizând cele 4 caracteristici discutate mai sus), RC-3*100-cycles (utilizând 3 caracteristici suplimentare bazate pe cele trei 100 de cicluri) și RC-11*30-cycles (utilizând unsprezece caracteristici suplimentare bazate pe cele unsprezece 30 de cicluri). Mai multe experimente pot fi găsite în teză.

5.2 Adoptarea unificării: o abordare cuprinzătoare a prioritizării moderne a cazurilor de testare

Conținutul acestei secțiuni este obținut din lucrarea originală [19].

Testarea de regresie este esențială pentru sistemele software care suferă modificări pentru a asigura funcționalitatea și pentru a identifica potențialele probleme. Este crucial să se verifice că modificările, cum ar fi corecțiile de bug-uri sau îmbunătățirile, nu afectează componentele funcționale existente ale sistemului. Prioritizarea Cazurilor de Testare (TCP) este o strategie utilizată în testarea de regresie care implică reordonarea cazurilor de testare pentru a detecta defectele devreme cu un cost minim de execuție.

Metodele TCP actuale au investigat diverse abordări, inclusiv criterii de acoperire bazate pe cod sursă, condiții bazate pe risc și condiții bazate pe cerințe. Cu toate acestea, din câte știm, în prezent nu există o reprezentare cuprinzătoare a TCP care să integreze eficient toți acești factori influenți. Abordarea noastră își propune să umple acest gol prin propunerea unei perspective cuprinzătoare a problemei TCP care integrează numeroase aspecte într-un framework unificat: informații de trasabilitate, context și informații despre caracteristici.

Pentru a valida abordarea noastră, folosim un set de date sintetic care ilustrează șase scenarii, fiecare cu combinații variate de cazuri de testare, defecte, cerințe, cicluri de execuție și informații despre codul sursă. Trei metode, Random, Greedy și Clustering (clasificare nesupervizată), sunt utilizate pentru a compara rezultatele obținute sub diverse bugete de execuție. Rezultatele experimentelor arată că metoda Clustering depășește constant Random și Greedy în diverse scenarii și bugete.

Secțiunea începe prin schițarea contextului investigației, împreună cu motivația, fundalul TCP și abordările de ultimă generație. Apoi prezentăm abordarea noastră nouă și unificată pentru TCP modern (așa cum se vede și în Figura 5.3). În final, oferim studiul de caz cu setul de date sintetic

și cele 6 scenarii pentru construirea soluțiilor TCP. Mai multe detalii pot fi găsite în teză.

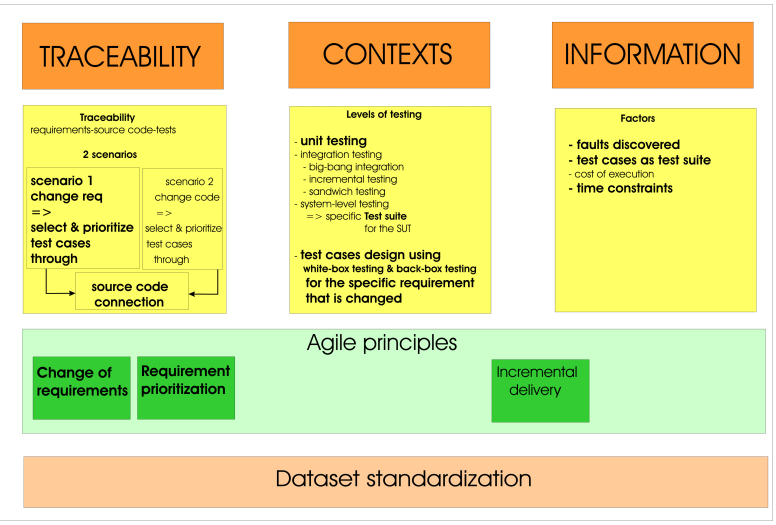

Figura 5.3: Prezentare generală a abordării.

Metrica utilizată pentru evaluare este NAPFD [9], însă pentru fiecare scenariu, informații diferite sunt încorporate în formulă. De exemplu, în scenariul bazat pe cerințe, defectele considerate sunt cele legate de cerințele care sunt schimbate.

Soluțiile TCP obținute pentru fiecare metodă utilizată sunt prezentate în Tabela 5.2. Experimentele au fost efectuate cu diverse configurații de buget de timp, de la 10 la 100. Majoritatea celor mai bune rezultate obținute sunt pentru metoda Clustering pentru aproape toate bugetele. Cele mai bune rezultate sunt îngroșate în Tabela 5.2.

Tabela 5.2: Valorile NAPFD pentru abordările Random, Greedy și Clustering pentru diverse valori de buget.

Scenario	S1						S2						S3					
Budget	10	25	50	75	80	100	10	25	50	75	80	100	10	25	50	75	80	100
Random	13.54	25.25	39.29	51.8	53.93	64.46	8.64	13.47	21.2	28.16	29.59	36.04	9.95	17.78	27.15	35.51	36.96	43.53
Greedy	21.43	35.71	64.29	**77.27**	79.17	**85.29**	**21.43**	**28.57**	**47.62**	50.79	51.43	53.78	**28.57**	**42.86**	56.12	60.71	61.69	**65.13**
Clustering	**42.86**	**57.14**	**72.22**	**77.27**	**80.77**	**85.29**	**21.43**	**28.57**	45.71	**51.95**	**52.75**	**53.78**	**28.57**	**42.86**	**58.04**	**62.5**	**63.19**	**65.13**
Scenario	S4						S5						S6					
Budget	10	25	50	75	80	100	10	25	50	75	80	100	10	25	50	75	80	100
Random	8.29	15.77	24.62	32.48	33.73	39.1	5.7	9.91	17.74	24.25	25.66	32.01	3.48	6.54	11.94	17.1	18.05	23.14
Greedy	21.43	**42.86**	**50**	**51.95**	**52.38**	**53.78**	21.43	**42.86**	50	51.95	52.38	**53.78**	**21.43**	37.5	40.18	40.91	41.07	**41.6**
Clustering	**37.5**	**42.86**	**50**	**52.04**	52.04	52.94	**28.57**	**42.86**	**51.43**	**53.06**	**53.06**	**53.78**	**21.43**	**38.57**	**40.71**	**41.21**	**41.33**	**41.6**

Soluțiile obținute pentru Greedy și Clustering sunt furnizate la acest link împreună cu setul de date [17].

Soluția Greedy este: [10, 11, 5, 1, 9, 12, 13, 6, 3, 2, 15, 16, 7, 17, 4, 8, 14]. Pentru soluția Clustering, clusterele sunt, de asemenea, evidențiate: [1, 5], [11], [10, 12], [13, 9], [15], [14, 4, 8, 17, 7], [6, 16, 2, 3]. Analiza relevă că, deși soluția Greedy identifică corect cazurile de testare mai devreme decât soluția Clustering, aceasta omite durata acestor teste. În contrast, soluția Clustering încorporează toate informațiile relevante, inclusiv durata testelor, oferind astfel o soluție mai cuprinzătoare și echilibrată. Mai multe informații despre rezultate pot fi găsite în teză.

5.3 Concluzii

În testarea de regresie, Prioritizarea Cazurilor de Testare este una dintre strategiile care poate fi utilizată, care își propune să ordoneze cazurile de testare pe baza unor criterii bine definite. Abordarea propusă RoughTCP utilizează un algoritm de clasificare nesupervizată bazat pe mulțimi

rough pentru a grupa și clasa cazurile de testare pe baza tiparelor lor intrinseci în informațiile despre defecte, durată și cicluri de integrare continuă.

Abordarea a fost validată riguros utilizând date din trei proiecte industriale distincte, fiecare furnizând informații din mediile lor de integrare continuă, cum ar fi durata, defectele sau informațiile de la 350 de cicluri. Rezultatele generale arată că abordarea RoughTCP depășește soluțiile de ultimă generație existente.

Acest capitol a propus, de asemenea, o unificare cuprinzătoare a diferitelor caracteristici și date care pot fi necesare în timpul unui proces TCP. După cum am discutat, în prezent nu există o definiție formală a TCP-ului care să acopere în mod adecvat diferiții factori influenți. Din studiul nostru de caz și validare, putem afirma empiric că având seturi de date cuprinzătoare și o soluție completă pentru acestea, crește performanța și precizia generală a procesului.

Lucrările viitoare pot considera alte proiecte pentru a valida RoughTCP. Caracteristicile proiectelor pot fi, de asemenea, luate în considerare în timpul clasificării nesupervizate *rough*. Un alt aspect important în contextul testării de regresie este legat de trasabilitatea între cerințe și cazurile de testare, astfel caracteristicile care o encapsulează pot fi de interes pentru investigațiile viitoare. Pentru unificare, viziunea noastră este să umplem golul existent cu un framework TCP unificat care să integreze adecvat numeroase perspective: cerințe, context și informații despre cod. Framework-ul ar trebui să poată asista atât la construirea seturilor de date, cât și la dezvoltarea soluțiilor pentru prioritizarea cazurilor de testare.

Capitolul 6

Concluzii

Această teză a prezentat cercetările noastre asupra algoritmilor teoretici de învățare automată, precum și aplicarea acestora în diferite contexte, inclusiv testarea software-ului.

În primul rând, discutăm despre obiectivele menționate în Capitolul 1. Am început cercetarea cu Obiectivul 1 și am studiat atât teoriile mulțimilor *fuzzy*, cât și cele ale mulțimilor *rough*, deoarece ambele sunt potrivite pentru contexte incerte. Ne-am decis să aplicăm mulțimile *rough*, deoarece sunt mai ușor de aplicat și adaptat pentru toate tipurile de date. Obiectivul 2 și Obiectivul 3 au fost realizate prin proiectarea și implementarea ABARC (descris în Secțiunea 3.1). Obiectivul 4, Obiectivul 5 și Obiectivul 6 se referă la evaluarea noilor noastre metode de clasificare nesupervizată bazate pe mulțimile *rough*. Am utilizat metodologii standard pentru evaluarea obișnuită a rezultatelor de clasificare, dar pentru rezultatele *rough* nu am găsit nicio metodologie existentă de evaluare, așa că am propus o metodologie nouă și am implementat-o pentru a evalua rezultatele generate de metodele de clasificare nesupervizată rough (prezentată în Capitolul 3). Obiectivul 7 a fost împlinit prin aplicarea metodelor noastre de clasificare nesupervizată *rough* pe diverse seturi de date standard dar și seturi de date industriale din testarea software (explicat în Secțiunea 5.1), precum și pe alte date din alte domenii. Obiectivul 8 specifică cercetarea unui domeniu diferit, testarea software-ului, în care metodele noastre noi s-au dovedit a fi utile. Am cercetat acest domeniu și am găsit niște cazuri de utilizare pentru metodele noastre propuse în prioritizarea cazurilor de testare (TCP). Obiectivul 9 descrie modul nostru de a contribui la domeniul TCP dintr-o perspectivă mai teoretică, am propus o abordare nouă și unificată pentru TCP, care ar combina toate informațiile cunoscute pentru a oferi o testare de regresie mai precisă (mai multe detalii pot fi găsite în Secțiunea 5.2). Obiectivul 10 se referă tot la cercetarea domeniului TCP, mai exact la analizarea metodelor de învățare de ultimă generație. După analiza noastră, am propus noi modele bazate pe rețele neuronale pentru TCP, așa cum este menționat în Obiectivul 11 (mai multe informații pot fi găsite în Secțiunea 4.1). Obiectivul 12 a fost realizat prin construirea soluției MixTCP în contextul TCP în CI (descrisă în Secțiunea 4.2). În final, Obiectivul 13 menționează aplicarea noilor noastre abordări de clasificare nesupervizată *rough* în testarea software și a fost realizat prin aplicarea acestora pe seturi de date industriale în contextul TCP (Secțiunea 5.1) și utilizându-le pentru un experiment în cadrul abordării unificate menționate mai sus (Secțiunea 5.2).

În primul rând, am dezvoltat metode inovatoare de clasificare nesupervizată aglomerativ care utilizează teoria mulțimilor *rough* pentru a identifica date hibride, inclusiv outlieri și instanțe *rough* care prezintă caracteristici aliniate cu multiple grupuri. Această abordare îmbunătățește analiza datelor prin oferirea unor perspective mai profunde asupra seturilor de date complexe, potențial dezvăluind noi clase sau anomalii. Scalabilitatea algoritmului nostru este asigurată de agenți

software care permit procesearea eficientă pe calculatoare moderne, chiar și pentru seturi de date mari. Analiza comparativă cu alte tehnici de clasificare a arătat că algoritmul nostru depășește majoritatea în termeni de acuratețe, entropie și Index Rand Ajustat, mai ales după eliminarea outlierilor.

În plus, am introdus o metodologie obiectivă pentru evaluarea calității datelor hibride identificate, oferind o evaluare mai fiabilă comparativ cu măsurile tradiționale de calitate a grupurilor. Această nouă metodologie de validare s-a dovedit eficientă în analiza noastră comparativă, demonstrând performanța robustă a abordării noastre.

Cercetarea noastră explorează, de asemenea, impactul diferitelor măsuri de similaritate asupra rezultatelor clasificării nesupervizate, în special în ceea ce privește instanțele *rough*. Având în vedere provocarea reprezentată de lipsa informațiilor specifice despre regiunile de suprapunere în seturile de date standard, am propus o metodologie inovatoare pentru identificarea potențialelor instanțe *rough*, care ar putea servi ca instrument de benchmarking.

Am realizat și o evaluare cuprinzătoare a rezultatelor noastre de clasificare, inclusiv a celor *rough*. Am luat în considerare metrici de evaluare interne, externe și *rough*. Compararea cu lucrările similare arată că metodele noastre sunt cele mai bune în majoritatea situațiilor.

În contextul integrării continue, unde testele sunt efectuate pentru fiecare caracteristică nouă sau corecție de bug, alegerea strategiei potrivite pentru Prioritizarea Cazurilor de Testare (TCP) este crucială pentru a detecta defectele devreme și eficient.

Investigația noastră asupra modelelor bazate pe rețele neuronale, în special abordarea NEUTRON integrată în MixTCP pentru aplicații reale, arată capacități superioare de prioritizare, depășind permutările aleatorii și rivalizând cu alte metode prin considerarea atât a duratei testului, cât și a ratelor de defecte.

Aplicarea reală a MixTCP a îmbunătățit semnificativ procesul TCP, și conduce la detectarea mai rapidă a defectelor și utilizarea mai eficientă a resurselor, ceea ce, la rândul său, accelerează timpii de lansare a software-ului și îmbunătățește fiabilitatea produsului.

În plus, am introdus metoda RoughTCP, care utilizează un algoritm de clasificare nesupervizată bazat pe mulțimile *rough* pentru TCP în testarea de regresie. Această metodă prioritizează cazurile de testare pe baza tiparelor lor legate de defecte, durată și cicluri de integrare continuă. Validată cu date din trei proiecte industriale, RoughTCP depășește soluțiile existente în performanță.

Lucrările noastre subliniază, de asemenea, necesitatea unei abordări unificate a TCP, integrând diverse date și caracteristici pentru a îmbunătăți eficacitatea procesului. Această strategie cuprinzătoare, validată prin studii de caz, arată o îmbunătățire semnificativă a performanței și preciziei TCP, subliniind importanța unei soluții complete și versatile pentru îmbunătățirea calității și eficienței dezvoltării software.

În ceea ce privește direcțiile viitoare, dorim să realizăm experimente suplimentare cu seturi de date care prezintă regiuni de suprapunere mai extinse și grupuri de forme variate. Ne propunem să explorăm efectul outlierilor asupra constatărilor noastre și să examinăm influența diverselor metrici de distanță asupra acestor rezultate.

În continuare, intenționăm să extindem cercetarea noastră asupra NEUTRON și RoughTCP aplicând aceste metode la o gamă mai largă de proiecte și scenarii mai complexe care includ diverse cicluri și un număr crescut de cazuri de testare care identifică defecte identice. Acest lucru implică, de asemenea, examinarea relației dintre schimbările în codul sursă sau cerințe și rezultatele cazurilor de testare. De asemenea, dorim să îmbunătățim adaptabilitatea MixTCP în diferite

medii de dezvoltare prin integrarea diverselor metodologii avansate TCP.

În final, cercetarea noastră viitoare va include potențial o evaluare a caracteristicilor specifice proiectelor care ar putea influența rezultatele clasificării nesupervizate *rough*. În contextul testării de regresie, abilitatea de a urmări conexiunea între cerințe și cazurile de testare este crucială, și caracteristicile care surprind această relație ar putea fi valoroase pentru studii ulterioare. Obiectivul nostru final este să dezvoltăm un framework TCP cuprinzător care să integreze perfect multiple dimensiuni, precum cerințele, contextul și informațiile despre cod, astfel acoperind golul existent în domeniu. Acest framework ar sprijini crearea de seturi de date și dezvoltarea de soluții pentru prioritizarea cazurilor de testare, oferind o abordare holistică pentru îmbunătățirea proceselor de testare a software-ului.

Bibliografie

[1] ALPAYDIN, E. *Machine learning*. MIT press, 2021.

[2] ELBAUM, S., ROTHERMEL, G., AND PENIX, J. Techniques for Improving Regression Testing in Continuous Integration Development Environments. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (New York, NY, USA, 2014), FSE 2014, Association for Computing Machinery, p. 235–245.

[3] GARCIA-DIAS, R., VIEIRA, S., PINAYA, W. H. L., AND MECHELLI, A. Clustering analysis. In *machine learning*. Elsevier, 2020, pp. 227–247.

[4] GAROUSI, V., RAINER, A., LAUVÅS JR, P., AND ARCURI, A. Software-testing education: A systematic literature mapping. *Journal of Systems and Software 165* (2020), 110570.

[5] GĂCEANU, R. D., SZEDERJESI-DRAGOMIR, A., AND VESCAN, A. Leveraging Rough Sets for Enhanced Test Case Prioritization in a Continuous Integration Context. In *7th Workshop on Validation, Analysis and Evolution of Software Tests (VST 2024), at the 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2024)* (Rovaniemi, Finland, 12 march 2024), IEEE. *accepted for publication*.

[6] GĂCEANU, R. D., SZEDERJESI-DRAGOMIR, A., POP, H. F., AND SÂRBU, C. ABARC: An agent-based rough sets clustering algorithm. *Intelligent Systems with Applications 16* (2022), 200117.

[7] KHATIBSYARBINI, M., ISA, M. A., JAWAWI, D. N., AND TUMENG, R. Test case prioritization approaches in regression testing: A systematic literature review. *Information and Software Technology 93* (2018), 74–93.

[8] LIMA, J. A. P., AND VERGILIO, S. R. A Multi-Armed Bandit Approach for Test Case Prioritization in Continuous Integration Environments. *IEEE Transactions on Software Engineering 48*, 2 (2022), 453–465.

[9] QU, X., COHEN, M., AND WOOLF, K. Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization. In *2007 IEEE International Conference on Software Maintenance* (Los Alamitos, CA, USA, oct 2007), IEEE Computer Society.

[10] SHAHIN, M., BABAR, M. A., AND ZHU, L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE access 5* (2017), 3909–3943.

[11] SHARMA, R., SHARMA, K., AND KHANNA, A. Study of supervised learning and unsupervised learning. *International Journal for Research in Applied Science and Engineering Technology 8*, 6 (2020), 588–593.

[12] SPIEKER, H., GOTLIEB, A., MARIJAN, D., AND MOSSIGE, M. Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis* (New York, NY, USA, 2017), ISSTA 2017, Association for Computing Machinery, p. 12–22.

[13] SZEDERJESI-DRAGOMIR, A. A Comprehensive Evaluation of Rough Sets Clustering in Uncertainty Driven Contexts. *Studia Universitatis Babeș-Bolyai Informatica 69*, 1 (2024), 41–56.

[14] SZEDERJESI-DRAGOMIR, A., GĂCEANU, R. D., POP, H. F., AND SÂRBU, C. A Comparison Study of Similarity Measures in Rough Sets Clustering. In *Proceedings of the 2019 IEEE 15th International Scientific Conference on Informatics (INFORMATICS 2019). Editors: William Steingartner, Štefan Korečko, Anikó Szakál* (Poprad, Slovakia, Nov 20 – 22, 2019), IEEE Press, pp. 399 – 405.

[15] SZEDERJESI-DRAGOMIR, A., GĂCEANU, R. D., POP, H. F., AND SÂRBU, C. Experiments on Rough Sets Clustering with Various Similarity Measures. *IPSI BGD TRANSACTIONS ON INTERNET RESEARCH 16* (2020), 75–83.

[16] SZEDERJESI-DRAGOMIR., A., GĂCEANU., R., AND VESCAN., A. Industrial Validation of a Neural Network Model Using the Novel MixTCP Tool. In *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2024* (Angers, France, 29 april 2024), INSTICC, SciTePress, pp. 110–119.

[17] VESCAN, A., GACEANU, R., AND SZEDERJESI-DRAGOMIR, A. Dataset and Results for Embracing Unification: A Comprehensive Approach to Modern Testcase Prioritization. `https://figshare.com/s/250342cc522867910bc4`, Accessed: 2024-03-09.

[18] VESCAN, A., GĂCEANU, R. D., AND SZEDERJESI-DRAGOMIR, A. Neural Network-Based Test Case Prioritization in Continuous Integration. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)* (sep 2023), IEEE, IEEE Computer Society, pp. 68–77.

[19] VESCAN., A., GĂCEANU., R., AND SZEDERJESI-DRAGOMIR., A. Embracing Unification: a Comprehensive Approach to Modern Test Case Prioritization. In *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2024* (Angers, France, 29 april 2024), INSTICC, SciTePress, pp. 396–405.

[20] YOO, S., AND HARMAN, M. Regression testing minimization, selection and prioritization: a survey. *Software testing, verification and reliability 22*, 2 (2012), 67–120.

[21] Mix. `https://hexdocs.pm/mix/Mix.html`. Accessed: 2024-05-15.