Raport științific și tehnic

privind implementarea proiectului 287PED/2020 “Graphomaly” în 2021

0. Introducere

Proiectul Graphomaly are ca scop producerea unei biblioteci de programe pentru *detecția anomaliilor în grafuri* ce modelează tranzacții financiare; prin anomalii înțelegem în primul rând posibile tranzacții frauduloase cu caracter complex, cum ar fi spălarea de bani, care se manifestă prin structuri tipice în graful de tranzacții.

Anul 2021, ocupând mai mult de jumătate din durata proiectului, a adus rezultate semnificative care ne apropie de încheierea cu succes a proiectului. Prezentăm în acest raport:

1. Algoritmii dezvoltați, cu accent pe cei specifici grafurilor de tranzacții
2. Starea implementării software
3. Rezultate experimentale pe date reale și simulate
4. Publicațiile realizate până în prezent
5. Modul de îndeplinire a planului de realizare

1. Algoritmi dezvoltați

Algoritmii pe care i-am dezvoltat până acum se încadrează în următoarele categorii generale:

1. Extragere de trăsături din graful asociat tranzacțiilor, urmată de aplicarea unor algoritmi generali de detecție de anomalii.
2. Similar, folosind direct atribute din lista de tranzacții și derivate ale lor.
3. Algoritmi de detecție de anomalii specifici.

Ei vor fi descriși mai jos, în această ordine.

1.1 Extragere de trăsături din graful de tranzacții

Graful de tranzacții construit pe baza listei de tranzacții dintr-o perioadă dată (de exemplu câteva luni) are ca noduri ID-urile conturilor implicate în tranzacții; mai multe conturi cu același deținător sunt grupate sub același ID atunci când este posibil. Un arc al grafului este caracterizat de

- nodul sursă (ordonator)
- nodul destinație (beneficiar)
- suma totală a tranzacțiilor între cele două entități, în lei
- numărul tranzacțiilor
- suma medie a unei tranzacții, ce rezultă imediat din aceste informații

Din acest graf se pot extrage mai multe feluri de trăsături. Am obținut rezultate bune cu subgrafuri de tip egonet și cu trăsături de tip random walk.

1.1.1 Trăsături extrase din egonet

Egonetul asociat unui nod *k* (numit rădăcină) al unui graf *G* este subgraful *E(k)* al lui *G* generat de nodul *k* și toți vecinii lui direcți; noțiunea se poate extinde la o vecinătate mai mare (de ordin 2, 3, etc.), dar beneficiile sunt minore în cazul datelor noastre, în timp ce complexitatea crește foarte mult. Pentru un graf orientat, definim similar out-egonetul $E_o(k)$ și in-egonetul $E_i(k)$, formate din vecinii pe arce de ieșire, respectiv intrare, ale nodului *k*. Noțiunea de egonet a fost inițial utilizată în [AMF10] pentru detecția de anomalii.

Pentru un nod *k* și egonetul asociat *E(k)* am calculat mai multe trăsături directe, între care cele mai relevante sunt:

- gradul (I/O, de intrare și de ieșire)
- suma totală (I/O)
- numărul de tranzacții (I/O)
- numărul de noduri al egonetului asociat
- numărul de arce al egonetului asociat

Din acestea, rezultă ușor trăsături derivate:

- suma medie (I/O)
- densitatea de arce în egonet

O inovație în acest proiect este definirea și utilizarea *egonetului redus* (egored), definit ca graful rămas din egonet după eliminarea nodurilor care au arce (în egonet) doar între ele și rădăcina egonetului. Pentru egored putem defini aceleași trăsături ca pentru egonet, cu observația că, prin creșterea densității arcelor, putem observa mai ușor structuri de tip clică sau chiar inel. Invers, structurile de tip stea pot dispărea, eventual parțial, ceea ce din nou e o diferență semnificativă față de egonet. Într-o rețea de tranzacții financiare așa cum este ea vizibilă de o bancă, nodurile eliminate reprezentă deseori fie ID-uri cu puține operații, fie ID-urile unor clienți ai altor bănci, ale căror alte tranzacții nu mai pot fi urmărite de banca în cauză.

Pe lângă trăsăturile definite identic cu cele ale unui egonet, pentru egored putem defini o serie de trăsături derivate:

- gradul relativ (I/O), adică raportul dintre gradul rădăcinii în egored și cel al rădăcinii în egonet; desigur, este un număr subunitar
- suma totală relativă (I/O) în raport cu egonetul
- suma medie relativă (I/O) în raport cu egonetul
- numărul de noduri relativ
- numărul de arce relativ

Datorită scalării naturale în intervalul [0,1], aceste trăsături sunt mai utile în detecția de anomalii în comparație cu valorile absolute extrase din egored.

1.1.2 Trăsături de tip random walk

Calculul egoneților este o operație destul de costisitoare pentru a fi făcută în timp real. De aceea algoritmi online pe acest principiu pot funcționa doar de pe-o zi pe alta, refacerea tuturor

egoneților afectați de operațiile unei zile de lucru și recalcularea trăsăturilor respective putând dura ore. De aceea ne-am orientat și către trăsături de tip random walk, care pot fi permanent recalculate imediat ce un nod a fost afectat (prin apariția/dispariția unui arc sau modificarea sumei totale a unui arc).

Asociem unui nod k trăsături obținute din construcția unui căi aleatoare pornind din k sau terminându-se în k, cu lungime cel mult ℓ. Nodul următor din cale este ales aleator dintre vecinii (in sau out, după caz) nodului curent. Dacă se revine în nodul k, atunci calea este încheiată chiar dacă nu a fost atinsă lungimea maximă; aceasta este de fapt o informație foarte relevantă, deoarece este vorba de sume care se întorc la plătitor, situație specifică spălării de bani (dar care poate avea și explicații legitime). De asemenea, calea poate fi mai scurtă dacă se ajunge într-un nod fără vecini pe ieșire. Toate informațiile se mediază pentru un număr fixat de căi, N_w, care pornesc sau se termină în rădăcina k (cele două cazuri sunt tratate distinct.)

Trăsăturile utilizate sunt valorile mediate pentru:

- Suma pe primul arc din cale, dacă acesta pornește din rădăcină, sau ultimul arc, dacă se termină în rădăcină. Aceasta este un substitut pentru suma totală (I/O).
- Suma care parcurge întreaga cale, adică cea mai mică sumă a unei tranzacții (arc) de pe acea cale. Aceasta poate fi privită ca o valoare medie tipică a fluxului de bani din rețea, obținută prin eșantionare.
- Suma care revine în rădăcină. Acesta este un eveniment mai degrabă atipic.

O altă valoare, de data aceasta nemediată, este

- Maximul sumelor care revin în rădăcină, pe ansamblul celor N_w căi aleatoare. Aceasta poate fi mai relevantă decât media pe cele N_w căi datorită posibilelor probabilități mici ale găsirii unei astfel de căi.

Menționăm că random walk a fost folosit și în alte metode de detecție de anomalii, dar nu în modul descris mai sus. De exemplu, în [WGZ18] se construiește un graf de similaritate a datelor, în care random walk este folosit pentru a detecta nodurile mai puțin vizitate; deci datele nu sunt nativ în formă de graf iar scopul este detectarea nodurilor izolate care potențial reprezintă anomalii, situație clar diferită de a noastră. Primele rezultate de acest tip pot fi găsite în [MT08].

1.2. Extragere de trăsături avansate din lista de tranzacții

În multe lucrări anterioare, pe același subiect sau subiecte conexe [CC19], a fost validat faptul că extinderea pe lăţime a setului de date, cu trăsături suplimentare calculate pe baza celor existente în setul inițial de tranzacții, aduce beneficii cu privire la performanţa obţinută atât cu metode supervizate, cât şi nesupervizate. În cele ce urmează, descriem tipurile și caracteristicile trăsăturilor noi, calculate pe baza celor existente şi în funcție de unitatea de timp în care au fost înregistrate (timestamp), iar în secțiunea Sec.3.1 vom detalia îmbunătățirea adusă faţă de algoritmii aplicaţi pe setul de date fără aceste trăsături avansate.

1.2.1. Trăsături extrase din marcajul de timp (dată, oră, minut, etc.)

Data exactă (timestamp) la care a fost înregistrată o tranzacţie este procesată segmentat, pentru a contribui la nivelul trăsăturilor îmbogățite. Din această înregistrare calendaristică, extragem următoarele trăsături noi, caracteristice fiecărei tranzacții:

- Ziua din an;
- Ziua din lună;
- Ziua din săptămână;
- Anul;
- Luna;
- Ora;
- Minutele din oră;
- Minutele din zi.

1.2.2. Trăsături istorice de grup, din marcajul de timp

În cadrul acestui grup de trăsături, cream două noi atribute ce calculează durata de la tranzacţia anterioară a aceluiași client (utilizator), în minute, respectiv ore.
Precum în cazul trăsăturilor statistice de grup, un anumit client face parte din două grupuri relevante:

1. Grupul tranzacțiilor efectuate - unde clientul este ordonator/emitent, și
2. Grupul tranzacțiilor primite - unde clientul este beneficiar/destinatar.

Duratele extrase se calculează pe rând, în cadrul aceluiași grup. De exemplu, vom calcula numărul de minute petrecute între data tranzacţiei curente, efectuată de clientul X, şi data ultimei tranzacții efectuată de același client.

1.2.3. Trăsături statistice de grup

Aceste trăsături sunt calculate peste valorile aferente *sumei tranzacţiei*, relativ la alte tranzacții dintr-o anumită zi ale aceluiași client (utilizator), excluzând ziua din care face parte tranzacția curentă, raportate la intervale de timp diferite, precum:

- 1D - ziua precedentă;
- 7D - cu 7 zile în urmă;
- 30D - cu 30 zile în urmă;
- 60D - cu 60 zile în urmă;
- 90D - cu 90 zile în urmă;
- 1M - aceeaşi zi a lunii precedente (e.g., 2 martie, faţă de 2 aprilie - data curentă);
- 2M - aceeaşi zi acum 2 luni;
- 3m - aceeaşi zi acum 3 luni.

Trăsăturile alese și folosite aici, peste un grup definit ca mai sus (i.e., sumele aferente tranzacțiilor unui anumit utilizator în cea mai recentă perioada dată) sunt:

- Minim;
- Maxim;
- Media;
- Deviaţia standard;

- Percentila 25;
- Mediana (percentila 50);
- Percentila 75.

Un anumit client face parte din două grupuri pentru aceeași perioadă, peste care se aplică setul de trăsături statistice, si anume:

1. Grupul tranzacțiilor efectuate - unde clientul este ordonator/emitent, și
2. Grupul tranzacțiilor primite - unde clientul este beneficiar/destinatar.

1.2.4. Trăsături istorice

Acest tip de trăsături reprezintă diferenţa dintre valorile a două trăsături existente: de regulă, suma tranzacţiei curente și o măsură statistică a tranzacțiilor efectuate într-o zi dintr-un interval ales faţă de ziua tranzacţiei curente. Pentru această ultimă categorie, trăsăturile sunt precalculate ca trăsături statistice de grup (vezi secțiunea anterioară).
La nivel de interval, selectăm pe rand fiecare opţiune dintre cele descrise mai sus (1D, 7D, 30D, etc.), iar la nivel de masura statistică aplicăm, pe rând, o opţiune dintre: minim, maxim şi medie.
De exemplu, având suma tranzacţiei curente și valoarea mediei peste toate tranzacţiile zilei cu 30 zile în urmă, vom obține o nouă valoare de forma:

$$transaction.amount\_local \; - \; 30day\_avg.amount\_local$$

1.3. Detecție de anomalii

Odată extrase trăsăturile, se pot aplica orice algoritmi de detecție de anomalii. Deși am obținut rezultate foarte bune cu Isolation Forest [LTZ08], am investigat și propus alte metode. Descriem aici metodele supervizate.

Procese Gaussiene. Problemele de detecție a anomaliilor pot fi rezolvate cu algoritmi de procese Gaussiene [WiRa06] în contextul unei învățări supervizate. Problema principală poate fi divizată în probleme de regresie și de clasificare. Ieșirile clasificării sunt etichete de clasă discrete, în timp ce regresia calculează predicția pe un spațiu continuu. Un caz particular de clasificare este clasificarea binară unde ambele metode pot fi aplicate. În acest caz, problema de regresie este rezolvată ca o problemă de regresie logistică în care eticheta clasei este determinată prin calculul semnului pentru predicția regresiei.

Modelele de regresie ale proceselor gaussiene sunt calculate prin definirea unui spațiu de funcții în care predicția este făcută prin proiectarea intrării prin spațiul funcțiilor. Prin natura sa ne așteptăm ca fiecare colecție finită de semnale aleatoare să aibă o distribuție normală multivariată. Cu toate acestea, distribuția normală multivariată ar trebui să respecte distribuția normală a datelor care nu prezintă anomalii. Mai mult, înainte de a calcula ponderile de regresie, semnalele sunt proiectate într-un spațiu neliniar prin utilizarea funcțiilor nucleu. Această funcție definește majoritatea proprietăților pentru modelele GP. Pentru a studia asemănarea dintre semnale, am folosit nucleul RBF. Antrenarea se face în două etape: optimizarea hiper parametrilor pentru nucleul dat și rezolvarea problemei de regresie.

Predicția anomaliilor se face prin calculul mediei și varianței distribuției normale multivariate pentru datele de test. Folosind valorile medii estimate de modelul GPR, obținem o separare logistică prin calculul semnului valorilor medii.

Pentru a demonstra utilitatea algoritmului propus, am dezvoltat o nouă metodă în formatul PyOD Toolbox. Pentru dezvoltarea algoritmului de procese gaussiene am folosit biblioteca GPyTorch [GPBWW18]. Deoarece procesele Gaussiene sunt algoritmi supervizați, toate seturile de date au fost împărțite în 60% pentru antrenament și 40% pentru testare. Pentru cele mai multe dintre seturile de date utilizate ca benchmark de metodele existente din PyOD, am obținut rezultate competitive în comparație cu acestea, conform așteptărilor de a avea rezultate superioare cu metode supervizate.

DL cu reprezentări rare uniforme. În [IRP21] este propusă o metodă nouă de detecție de anomalii folosind învățarea dicționarelor (DL - Dictionary Learning). Ea se bazează pe reprezentarea exclusiv a semnalelor normale, căutând un dicționar în care acestea să fie reprezentate de un grup de atomi comuni, ceilalți atomi fiind folosiți foarte rar. În acest scop se introduce un termen de regularizare care încurajează gruparea coeficienților nenuli pe liniile matricelor de reprezentare. Funcția folosită pentru regularizare poate fi convexă (suma normelor euclidiene ale liniilor) sau nu (norma 0), în ambele cazuri fiind deduse soluții exacte pentru problemele de optimizare asociate fazei de actualizare a dicționarului. Algoritmul rezultat este deci rapid, similar ca viteză cu K-SVD [AEB06].

Rezultatele pe benchmark-ul PyOD sunt din nou clar mai bune decât cele ale metodelor din bibliotecă, un factor fiind și caracterul supervizat al metodei noastre. De menționat că pentru antrenare sunt folosite 90% din semnalele normale, iar pentru testare 10% din semnalele normale și toate cele anormale.

Metode de vot și metode de tip ensemble pentru a identifica combinațiile optime. Biblioteca Graphomaly (sec.2.2) conține o implementare pentru analiza rezultatelor, ce permite comparații între diferitele metode de detecție de anomalii. Scopul este identificarea tranzacțiilor pentru care există consens între algoritmi, și stabilirea gradului de încredere în rezultatele obținute.

Datorită specificului diferit al seturilor de date utilizate (Libra, BRD, sintetice), acest consens se poate evalua la nivel de

- tranzacție (arc în graf)
- entitate financiară (nod în graf).

Metodele disponibile în biblioteca PyOD se încadrează în cinci clase distincte: modele liniare, bazate pe proximitate, probabilistice, rețele neurale și metode de tip *ensemble*. Analiza rezultatelor permite evaluarea consensului per categorie de algoritmi, pentru cazul în care se testează cel puțin două metode din fiecare clasă. Utilitatea acestui tip de evaluări stă în posibilitatea identificării tipului potrivit de metode pentru setul de date utilizat. Spre exemplu, atât

testele realizate pe datele Libra, cât și pe datele BRD au rezultat într-un consens ridicat în cadrul categoriei de metode liniare, din care s-au testat algoritmii Principal Component Analysis (PCA), Minimum Covariance Determinant (MCD), One-Class SVM (OCSVM) și Deviation-based Outlier Detection (LMDD). La extrema cealaltă, cel mai scăzut consens pentru ambele baze de date le-au reprezentat metodele Ensembles, din care s-au testat Isolation Forest, Locally Selective Combination of Parallel Outlier Ensembles (LSCP), Extreme Boosting Based Outlier Detection (XGBOD) și Lightweight On-line Detector of Anomalies (LODA).

Funcția de analiză a rezultatelor generează un raport ce conține numărul de anomalii identificate de fiecare metodă, consensul total între metodele testate (numărul de anomalii identificate de toate metodele), consensul per categorie, cât și listele de anomalii identificate de cel puțin 2 metode.

De asemenea, biblioteca PyOD conține metode de votare bazate pe scorurile obținute de algoritmii de identificare de anomalii, ce pot fi extinse și la metode ce nu sunt incluse în PyOD. Funcțiile returnează un nou set de scoruri, calculate integrând după diferite criterii scorurile metodelor testate. Lista de anomalii bazată pe aceste scoruri este inclusă în raport.

2. Implementări software și biblioteca Graphomaly

Descriem aici rezultatele obținute în privința implementării algoritmilor, proprii sau nu, care vor constitui produsul software final al acestui proiect.

2.1. Biblioteca Dictionary Learning

Biblioteca Dictionary Learning, realizată în mare parte în anul trecut, a fost acum completată, verificată și documentată. Pachetul Python este numit dictlearn și este public disponibil la https://gitlab.com/unibuc/graphomaly/dictionary-learning/. Documentația aferentă poate fi găsită la https://unibuc.gitlab.io/graphomaly/dictionary-learning/index.html.

Biblioteca va fi actualizată cu alte funcții de învățare a dicționarelor și va avea un caracter deschis. Interfața este realizată în format sklearn [SciLe11], astfel că biblioteca poate fi apelată direct de utilizatorul tipic de software pentru machine learning. Avem în vedere același gen de interfață pentru toate programele realizate.

2.2. Biblioteca Graphomaly

Biblioteca Graphomaly, dedicată detecției de anomalii în date de tip graf, este într-un stadiu avansat. În acest moment, ea conține mai multe funcții de preprocesare, extragere de trăsături, definire de trăsături derivate, detecție de anomalii și calcul de indicatori de performanță. Și ea va

avea un caracter deschis, structura funcțională permițând adăugarea de funcții noi specializate pentru anumite etape. Descrierea bibliotecii este lăsată pentru etapa finală a proiectului; aici dăm doar câteva detalii relevante pentru faza curentă.

Extragerea trăsăturilor propuse de tip egonet sau random walk, descrise în Sec.1, a fost implementată în Python, folosind printre altele și pachetul Networkx. Câteva observații:

- Extragerea unui egonet orientat (in-egonet sau out-egonet) este foarte rapidă în raport cu extragerea unui egonet neorientat folosind funcția ego_graph din Networkx; este vorba doar despre egonet de ordin 1, pentru ordine mai mari complexitatea fiind oricum nepractică. De aceea, am folosit o implementare care s-a dovedit mult mai rapidă, construind o listă de noduri conținând rădăcina, succesorii și predecesorii ei, apoi extrăgând subgraful corespunzător acelor noduri. Câștigul de viteză este semnificativ, făcând calculul trăsăturilor fezabil pentru grafuri cu sute de mii de noduri și milioane de arce.
- Implementarea este flexibilă, lăsând loc și pentru crearea de trăsături din atribute sumabile asociate arcelor; sumele acestora se asociază nodurilor adiacente.
- Deocamdată am implementat random walk cu probabilități egale pentru arcele unui nod; extensia la probabilități depinzând de sumele tranzacțiilor este tehnic ușor posibilă, dar trebuie investigat ce relații între probabilități și sume sunt mai relevante, deoarece simpla proporționalitate poate produce probabilități foarte mici.

În ceea ce privește detecția anomaliilor la nivel de tranzacții, am extins funcţionalitatea bibliotecii pentru a cuprinde partea de preprocesare: crearea trăsăturilor avansate din lista de tranzacții, descrise in Sec. 1.2..

De asemenea, întrucât implementările anterioare (e.g, PyOD) de arhitecturi de reţele adânci testate de noi aveau câteva neajunsuri (salvarea modelului antrenat și a parametrilor acestuia nu era completă, iar arhitectura era greu de ajustat pentru diferite scenarii), am recurs la implementarea modelelor de Auto-Encoder [LCL14] şi Auto-Encoder Variaţional [KW13], ambele parametrizabile, prin intermediul bibliotecii Python pentru rețele neurale, Tensorflow [AAB16].

În scopul demersurilor pentru publicarea bibliotecii Graphomaly, am refactorizat masiv codul, adăugând fișiere de configurare și teste unitare, menţinând un flux cât mai coerent al etapelor către obținerea unor rezultate pe un set de date oarecare, ce se supune caracteristicilor acestei probleme.

De asemenea, în cadrul Graphomaly au fost folosite și biblioteci Python existente pentru învățare automată și detecție de anomalii:

- Python Outlier Detection (PyOD) [ZNL19]
- Accelerating Large-scale Unsupervised Heterogeneous Outlier Detection (SUOD) [ZHC21]
- Skyline real time anomaly detection system (Skyline) [GitSky]
- Automated Time-series Outlier Detection System (TODS) [LZW21]
- Python Streaming Anomaly Detection (PySAD) [YK20]

3. Rezultate experimentale

Având în vedere specificul tranzacțiilor bancare, în care anomaliile (tranzacțiile frauduloase) sunt rare și verificarea unei tranzacții de către un expert este consumatoare de timp și deci costisitoare, este esențial ca anomaliile adevărate să aibă un scor mare și deci să fie plasate cât mai la începutul listei de anomalii detectate (ordonate conform scorurilor specifice algoritmului folosit). Vom evalua rezultatele pe următorii indicatori principali:

- TPR (true positive rate = număr anomalii detectate / număr total anomalii) pentru primele 0.1%, 0.2%, 0.5% și 1% din totalul ID-urilor analizate. Acestea sunt niște borne oarecum arbitrare, dar arată performanța metodelor în zona critică interesantă.
- TPR AUC 1% (TPR Area under Curve) în 1% din totalul ID-urilor analizate. Acesta este un indicator integrator, care permite compararea globală a mai multor metode folosind o singură valoare; îl vom nota AUC_1%. Ne limităm la 1% și deoarece pe ansamblul tuturor ID-urilor majoritatea metodelor dau rezultate aproape de valoarea optimă 1.

Desigur, graficul TPR este și el relevant și este la baza informațiilor de mai sus.

Menționăm că aproape toate rezultatele sunt obținute cu funcții integrate în biblioteca Graphomaly.

3.1 Rezultate pe date de la Libra Internet Bank

În cadrul colaborării cu Libra Internet Bank (numită Libra în acest raport), ne-au fost puse la dispoziție tranzacțiile din anii 2020 și 2021; informațiile despre clienți au fost anonimizate. Vom prezenta aici rezultatele pe un set de date cuprinzând tranzacțiile bancare între clienți Libra sau între aceștia și clienți ai altor bănci, pe o durată de 3 luni în cursul anului 2021; au fost excluse tranzacțiile cu cardul. Numim Libra_3months acest set de date. Iată câteva caracteristici ale sale:

- Număr de tranzacţii iniţiale: 4558805
- Număr de tranzacții rămase în graf: 1819529
- Număr ID-uri distincte: 385100
- Număr tranzacții cu alerte: 517. Aceste tranzacții sunt marcate, conform unei proceduri interne a băncii, bazată pe un set de reguli rezultate din experiență, pentru investigație ulterioară de către un expert uman.
- Număr de tranzacții cu raportări: 11. Aceste tranzacții au un caracter potențial fraudulos, determinat în urma investigației expertului uman, și sunt raportate ONPCSB (Oficiul Național de Prevenire și Combatere a Spălării Banilor).

3.1.1 Rezultate obţinute din procesarea la nivel de noduri din graf

Graful rezultat din aceste tranzacții are următoarele caracteristici:

- Număr de noduri (ID-uri): 385100
- Număr de arce (perechi ordonate de noduri care tranzacționează, adică numărul de tranzacții între ele este cel puțin 1) : 597165
- Număr ID-uri distincte cu alerte: 600. Deoarece un ID poate fi implicat în mai multe tranzacții suspicioase, numărul de alerte pentru un ID poate fi mai mare ca 1; atunci când

calculăm TPR, asociem ID-ului o pondere egală cu numărul de alerte. Numărul maxim de alerte asociate unui singur ID este 22.
- Număr ID-uri cu raportări: 15. Aceeași observație ca mai sus. Numărul maxim de raportări asociate unui ID este 3.

Prezentăm aici doar rezultatele obținute de câte o configurație de trăsături de fiecare tip:
- Egored - trăsături de tip egonet (grad I/O, sume I/O, sume medii I/O, densitate de arce în egonet) și egonet redus (aceleași trăsături, dar gradele și sumele în valoare relativă față de egonet.
- RW - trăsături de tip random walk, pe lângă grade, sume și sume medii (al căror calcul este banal), obținute prin medierea a 100 de treceri pentru fiecare nod.
- Oddball [AMF10] calculează mai multe statistici; raportăm aici rezultate ale unor statistici diferite pentru alerte (egonet-in-degree vs egonet-in-weight) și raportări (edges vs nodes, on egonets), cele care au dat cele mai bune rezultate; majoritatea statisticilor din Oddball dau rezultate foarte slabe
- EL - metoda din [Ell19], dar utilizând doar trăsăturile care se pot calcula relativ rapid (GAW, GAW10, GAW20, Standard degree); pentru celelalte trăsături timpul de calcul este extrem de lung: deja pentru câteva zeci de mii de noduri timpul este de ordinul zilelor.

Rezultatele se găsesc în tabelul de mai jos. Pentru Egored și RW, după extragerea trăsăturilor, detecția de anomalii a fost efectuată cu Isolation Forest; rezultatele raportate sunt medii pentru 10 rulări. După cum se vede, abordarea noastră de tip egonet redus dă rezultatele cele mai bune atât pentru alerte cât și pentru raportări la AUC_1% și la aproape toate bornele. Abordarea de tip random walk are rezultate apropiate. Oddball este mult inferioară, în special la alerte, iar EL are rezultate rezonabil de bune, dar slabe pentru borna 0.1%.

Config	Alerte					Raportări				
	AUC 1%	TPR 0.1%	TPR 0.2%	TPR 0.5%	TPR 1%	AUC 1%	TPR 0.1%	TPR 0.2%	TPR 0.5%	TPR 1%
Egored	0.6042	0.4045	0.5143	0.6596	0.7446	0.6032	0.2409	0.3864	0.6909	0.8091
RW	0.5792	0.3333	0.4563	0.6335	0.7550	0.5403	0.2364	0.3773	0.5000	0.8227
Oddball	0.2504	0.0445	0.0948	0.2514	0.4671	0.4183	0.1364	0.2727	0.5454	0.5909
EL	0.5561	0.1982	0.4265	0.6286	0.7843	0.4250	0.1362	0.3636	0.5000	0.5909

Timpii de execuție pe un laptop cu procesor i7 cu 6 nuclee și 16 GB memorie RAM sunt: aproximativ o oră pentru extragerea trăsăturilor Egored și RW (plus aproape un minut pentru fiecare execuție Isolation Forest), aproximativ 2 ore pentru Oddball și 10 minute pentru varianta

simplificată a EL. Deși RW are un timp de execuție comparabil cu Egored, reamintim că extragerea trăsăturilor RW se pretează foarte bine unui algoritm online, adaptarea lor fiind foarte simplă atunci când se adaugă noduri sau arce noi în graf.

3.1.2. Rezultate obţinute din procesarea la nivel de tranzacții

Aplicând trei algoritmi pentru detecția nesupervizată de anomalii: Isolation Forest (IF) - din biblioteca PyOD, Auto-Encoder (AE) şi Auto-Encoder Variaţional (VAE), am comparat rezultatele obţinute pe setul de date de 3 luni descris mai sus, cu trăsături tranzacţionale extrase şi procesate direct din setul de date iniţial, cu setul de date extins cu trăsături avansate descrise în Sec. 1.2.. Rezultatele sunt prezentate mai jos.

3.1.2.1. Rezultate obţinute la nivel de tranzacții, cu trăsături iniţiale procesate

Am aplicat cei 3 algoritmi peste setul de tranzacții inițiale, obținând cele mai bune rezultate prezentate în primele 3 linii din tabelul de mai jos.

3.1.2.2. Rezultate obţinute la nivel de tranzacții, cu trăsături extinse cu cele avansate

Pentru a putea calcula cat mai multe valori pentru trăsăturile extinse, avansate, din cele descrise mai sus (raportate la intervale de 60 sau 90 de zile), menţionăm că am extins, doar pentru aceasta procesare, setul de date cu 3 luni anterioare perioadei evaluate. Prin urmare, algoritmii evaluați pe setul extins cu trăsături avansate au beneficiat de mai multă informație istorică decât restul algoritmilor prezentați. Este foarte probabil ca o bună parte din câștigul în performanță să fie, deci, datorat informației adiționale din trecut, pe lângă capacitatea de sinteză a datelor surprinse de aceste trăsături, în felul în care sunt definite. Aceste rezultate sunt redate în ultimele 3 linii din tabelul de mai jos.

Algoritmii din ambele scenarii sunt antrenaţi și evaluaţi pe aceleași 3 luni specificate mai sus, pentru o analiză uniformă.

Menționăm că aceste rezultate nu pot fi comparate direct cu cele obținute după extragerea trăsăturilor din graf, deoarece analiza, în acest caz, se face la nivel de tranzacţie, iar în cazul analizei de graf, la nivel de nod/ID, reprezentând un anumit client. Mai mult, în urma procesării la nivel de graf, se ajunge la un set de tranzacții de circa 2.5 ori mai mic faţă de setul inițial, analizat în tabelul următor.

Config	Alerte		Raportări	
	AUC 1%	TPR 1%	AUC 1%	TPR 1%
IF	**0.2036**	**0.3927**	**0.4874**	**0.9091**
AE	0.1178	0.2018	0.2574	0.3636

VAE	0.1103	0.1873	0.2191	0.2727
IF_adv	**0.3518**	**0.4473**	0.2293	0.2727
AE_adv	0.2248	0.3727	**0.2846**	**0.3636**
VAE_adv	0.2182	0.3673	0.2823	**0.3636**

Observăm că Isolation Forest are, în general, o performanţă mai bună faţă de metodele de deep learning (AE, VAE), în ambele scenarii de evaluare (cu/fără trăsături avansate) pentru alerte, cu mai mult de 10% creştere în valoare absolută atât pentru AUC1%, cât și pentru TPR1%. Totuşi, IF obține rezultate mai slabe cu 6% AUC1% şi 9% TPR1% faţă de AE, în valoare absolută, pentru scenariul cu trăsături avansate, în cazul raportărilor.

În ceea ce privește extragerea de trăsături avansate, aceasta se dovedeşte extrem de utilă atunci când încercăm să surprindem tranzacțiile alertate, obținând rezultate cu minim 10% mai bune pe fiecare algoritm în parte. Singura excepție este cazul evaluării IF pe tranzacții cu trăsături de bază, în detecţia de raportări, unde IF ajunge la o performanţă de 0.9091 TPR1%, următorul cel mai bun scor obținut fiind doar de 0.3636 TPR1%.

3.1.3. Rezultate obţinute prin aplicarea unor metode de vot

Pe acelasi set de date, folosind scorurile algoritmilor de mai sus (IF, AE şi VAE), am aplicat algoritmi de vot existenţi în biblioteca PyOD şi adaptaţi în framework-ul bibliotecii curente (Average, Max, Median, AOM, MOA). Scorurile sunt raportate separat, cu/fără trăsături avansate:

Config	Alerte		Raportări	
	AUC 1%	TPR 1%	AUC 1%	TPR 1%
Average	0.1204	0.2255	0.2547	0.3636
Max	**0.1444**	**0.3473**	**0.3195**	**0.7273**
Median	0.1143	0.1945	0.2505	0.3636
AOM*	0.1204	0.2255	0.2547	0.3636
MOA*	**0.1444**	**0.3473**	**0.3195**	**0.7273**
Average_adv	0.2595	0.3945	0.2919	**0.3636**
Max_adv	**0.2979**	**0.4400**	**0.2941**	**0.3636**
Median_adv	0.2223	0.3655	0.2829	**0.3636**
AOM_adv*	0.2595	0.3945	0.2919	**0.3636**

MOA_adv*	**0.2979**	**0.4400**	**0.2941**	**0.3636**

*AOM şi MOA aplicate peste un grup de dimensiune 3 devin echivalente cu Average, respectiv Maxim. În viitor, avem în vedere aplicarea acestor metrici pentru grupuri mai mari.

Comparativ cu scorurile obţinute anterior, performanța obținută prin metodele de vot nu este mai bună pentru setul de date cu trăsături inițiale, rezultatul obţinut de Isolation Forest fiind superior atât în cazul alertelor, cât și al tranzacțiilor.
În schimb, pentru setul de date extins cu trăsăturile avansate, observam o ușoară îmbunătățire, prin vot Maxim, a predicției raportărilor, pentru AUC1%, de la 0.2846 (pentru AE_adv) la 0.2941 (pentru Max_adv). Acest lucru se datorează, probabil, incompatibilităţii de detecție a anomaliilor: diferite tipuri anomalii sunt detectate de diferiți algoritmi. Prin urmare, în viitor este de dorit aplicarea metodelor de vot peste algoritmi sau grupuri de algoritmi similari (e.g., mai multe rulări cu hiper-parametri diferiți pentru reţele, peste aceleași date), precum și găsirea unor metode complexe de vot (e.g., votul pentru IF să conteze mai mult în anumite situații - eventual cu segmentarea datelor nesupervizat în subseturi pe care IF are, în general, o predicție mai buna).

3.2. Rezultate pe date simulate

Prezentăm aici rezultatele pentru un singur graf simulat.

Datele sintetice sunt generate utilizând un model de graf multidirecțional bloc stochastic cu 50000 de noduri și 438091 arce. Graful conține un număr de 5000 de module, ce simulează structuri de tip comunități. Caracteristicile grafului sunt obținute prin setarea unui set de parametri ce descriu probabilitățile de existență a arcelor între 2 noduri aparținând aceluiași modul, respectiv a 2 noduri ce aparțin unor module diferite. În plus, între oricare 2 noduri din același modul pot exista maxim 6 arce, iar între noduri din module diferite maxim 3 arce.

În acest graf sunt introduse în locații aleatoare un număr de 25 de subgrafuri reprezentând structuri anormale, toate de dimensiune 10 noduri:

- 15 structuri de tip clică
- 5 structuri de tip stea
- 5 structuri de tip inel

Fiecare nod este descris de 5 atribute numerice având o distribuție normală și 2 atribute categoriale. Arcele sunt descrise de 40 de atribute numerice și 6 atribute categoriale. În plus, fiecare arc (ce reprezintă o tranzacție) conține un atribut suplimentar ce reprezintă suma tranzacționată și unul ce reprezintă momentul de timp la care se realizează aceasta. Sumele tranzacționate respectă o distribuție log-normală și sunt limitate în intervalul [1, 1000000]. Tranzacțiile din întreg graful se înscriu într-o perioadă de 10 luni.

Ultimele două atribute sunt stabilite în mod diferit pentru sub-grafurile anormale. Toate tranzacțiile dintr-o structură anormală sunt realizate într-un interval de 8 ore, iar varianța pentru sumele tranzacționate între 2 noduri este 0.1.

Așa cum era de așteptat, având în vedere ponderea mare a clicilor între anomalii, metoda OddBall are rezultatele cele mai bune, obținând un AUC_1% de 0.8079 pentru statistica „nodes vs edges”; alte statistici dau în schimb rezultate mult mai slabe, a doua fiind „egonet-weight vs egonet-max-weight”, cu doar 0.3498. În funcție de trăsăturile folosite, metoda noastră dă rezultate care variază între 0.7733 pentru trăsături de tip random walk și 0.8034 atunci când sunt incluse și trăsături de tip Egored. Obținem așadar valori foarte apropiate de OddBall. În schimb, rezultatele EL (varianta simplificată) sunt dezamăgitoare, cu AUC_1% = 0.046.

În ceea ce priveşte aplicarea IF și a metodelor de deep-learning, peste setul de date sintetice, raportăm un scor AUC_1% de 0.7645 pentru IF, 0.6552 pentru AE şi, respectiv, 0.6648 pentru VAE, peste trăsăturile extrase din egoneți. Deşi puţin mai slabe decât cea mai bună performanţă raportată mai sus, aceste rezultate sunt promiţătoare, având în vedere că atât IF, cât şi cele două arhitecturi de reţele nu sunt adaptate pentru interpretarea și procesarea, la nivel de graf, a datelor, ci utilizează doar trăsăturile extrase din graf în etapa de preprocesare.

Așadar, între algoritmii comparați, metoda noastră are rezultate foarte bune atât pe datele Libra cât și pe cele simulate.

3.3 Rezultate pe date de la BRD

BRD a furnizat o bază de date sintetică de test realizată plecând de la specificul evenimentelor anormale identificate în cadrul băncii. Baza de date are un conținut anonimizat și este destinată unui studiu explorator al posibilităților de analiză a anomaliilor pe graf, fiind ne-etichetată. Aceasta conține următoarele câmpuri: PLAY (codifică direcția arcelor între entitățile care tranzacționează), KID (desemnează entitatea care tranzacționează), BOX (desemnează resursa tranzacționată), VBOX (valoare numerică ce caracterizează câmpul BOX), VTOYS (valoare numerică), DAY, MONTH, HOUR, MINUTE. Baza de date conține 379341 tranzacții individuale.

Testele pe această bază de date au urmat două abordări. Prima a constat în evaluarea unui set de metode din biblioteca PyOD, cu verificarea consensului între algoritmi și returnarea unei liste scurte de tranzacții suspecte.

Cea de-a doua abordare a presupus construcția unor grafuri egonet de ordinul 2 și extragerea de trăsături privind conectivitatea acestuia, urmată de o metodă de clustering nesupervizată. Figura 1 prezintă structura unui astfel de graf, în care, pe lângă vecinii direcți ai unui nod (nodurile cu care acesta are tranzacții într-un anumit interval de timp), sunt incluse și nodurile cu care interacționează aceștia.

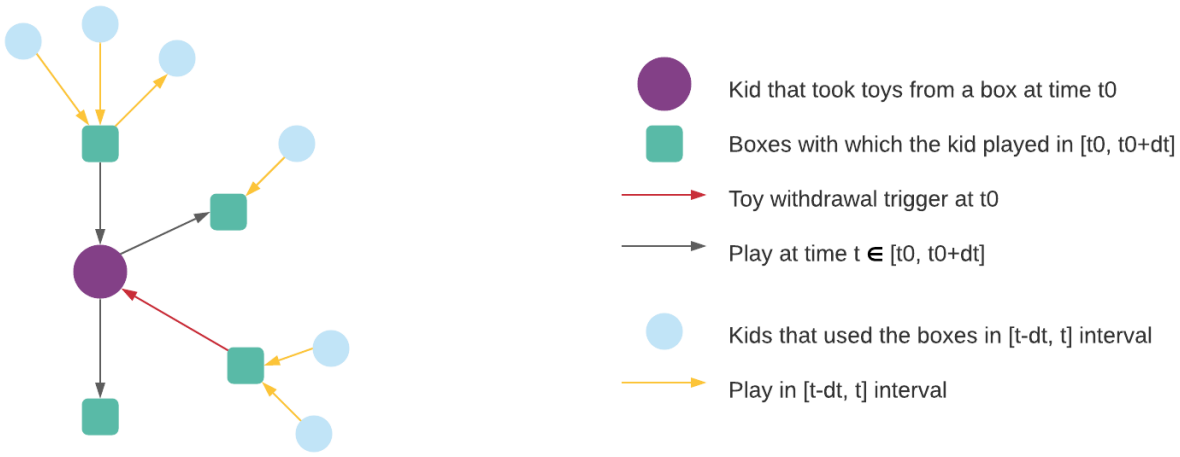

Figura 1. Exemplu de egonet de ordin 2

Graful este generat pentru fiecare tranzacție PLAY = -1 și se obține următorul set de trăsături:

- Egonet de ordinul 1
  - la nivel de arce (PLAYS)
    - numărul de arce în fiecare direcție
    - media și deviația standard ale variabilei VTOYS pentru fiecare direcție
    - media și deviația standard a diferențelor între tranzacția curentă și restul tranzacțiilor cu nodurile vecine
  - la nivel de entități tranzacționate (BOXES)
    - media și deviația standard a variabilei BOX
    - media și deviația standard a variabilei VBOX
- Egonet de ordinul 2
  - la nivel de arce (PLAYS)
    - numărul de arce în fiecare direcție
    - media și deviația standard ale variabilei VTOYS pentru fiecare direcție
    - media și deviația standard a diferențelor între tranzacția curentă și restul tranzacțiilor cu nodurile vecine
  - la nivel de noduri (KID)
    - media și deviația standard a variabilei KID

Cel mai mic cluster obținut aplicând algoritmului KMeans (cu 30 de clustere) pe acest set de date are dimensiune 342 (tranzacții), și reprezintă posibile anomalii.

4. Publicații

Publicațiile rezultate până acum din activitatea la acest proiect sunt:

[IlDu21] D.Ilie-Ablachim, B.Dumitrescu, "Classification with Incoherent Kernel Dictionary Learning", Int. Conf. Control Systems and Computer Science (CSCS), Bucharest, May 2021.

[MiDu21] F.I.Miertoiu, B.Dumitrescu, "Shape Parameter and Sparse Representation Recovery under Generalized Gaussian Noise", European Signal Processing Conference (EUSIPCO), Dublin, Ireland, Aug. 2021.

[RuIr21] C. Rusu, P. Irofti, “Efficient and Parallel Separable Dictionary Learning,” 2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS), Beijing, China, Dec. 2021 (https://arxiv.org/abs/2007.03800)

[PaIr21] A. Pătrașcu, P.Irofti, “Computational complexity of Inexact Proximal Point Algorithm for Convex Optimization under Holderian Growth”, submitted to Journal of Machine Learning Research, Aug. 2021 (https://arxiv.org/abs/2108.04482)

[IRSP21] Paul Irofti, Luis Romero-Ben, Florin Stoican, Vicenç Puig, “Data-driven Leak Localization in Water Distribution Networks via Dictionary Learning and Graph-based Interpolation”, submitted to 2022 American Control Conference (ACC) (https://arxiv.org/abs/2110.06372)

[IRP21] Paul Irofti, Cristian Rusu, Andrei Pătrașcu, “Dictionary Learning with Uniform Sparse Representations for Anomaly Detection”, submitted to 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

Pe lângă acestea, este în curs de redactare un articol pe baza rezultatelor despre egored și random walk prezentate în acest raport, care se situează în zona contribuțiilor dorite în propunerea de proiect. Acest articol va fi trimis la o revistă.

Prezentăm acum în câteva cuvinte conținutul articolelor realizate în acest an.

În [IlDu21] am extins algoritmii de învățare a dicționarelor (Dictionary Learning - DL) care optimizează incoerența la probleme neliniare cu nucleu (kernel DL). Avantajul este buna condiționare a dicționarului. Pentru problemele de clasificare abordate, un avantaj surprinzător este reducerea substanțială a timpului de clasificare în cazul în care dimensiunea semnalelor este mai mare decât numărul de semnale dintr-o clasă; în acest caz dicționarul asociat nucleului are dimensiunea “mică”, cea a numărului de semnale din clasă. Mai important, calitatea clasificării nu este afectată, în condițiile în care timpul de antrenare este de 10-100 de ori mai scurt.

Lucrarea [MiDu21] este conexă problemelor DL și se referă exclusiv la etapa de reprezentare rară. Am abordat aici cazul zgomotului necunoscut, despre care se știe doar apartenența la distribuția gaussiană generalizată (DGG). Elementul de originalitate este estimarea automată a factorului de formă ce caracterizează DGG. Se pot obține astfel reprezentări cu calitate

superioară cazului în care se utilizează algoritmi dedicați zgomotului gaussian (de exemplu, OMP), celui laplacian sau celui DGG, dar cu factor de formă fixat [WLL16].

În [RuIr21] sunt propuși algoritmi cu grad ridicat de paralelism pentru forma separabilă a DL. Ei se bazează pe optimizarea succesivă a dicționarelor, apelând la probleme de tip cele mai mici pătrate, ca în MOD [EAH99]. Un caz particular interesant este cel al dicționarelor ortogonale, pentru care accelerația are valori remarcabile.

În [PaIr21] se analizează complexitatea Algoritmilor de Punct Proximal (APP) cu Informație Inexactă pentru minimizarea funcțiilor convexe cu creștere de tip Hölder. Algoritmii APP, în varianta lor originală, necesită la fiecare iterație soluția unei subprobleme regularizate, de regulă dificil de obținut. Varianta inexactă sugerată în [PaIr21] dezvoltă o schemă implementabilă a acestor algoritmi bazată pe restartare. Aplicațiile prezentate acoperă metoda de învățare GraphSVM, care profită de inter-relațiile dintre atributele unui set de date, structurate sub formă de graf neorientat. Testele confirmă eficiența algoritmilor dezvoltați comparativ cu cei existenți. Deoarece extensiile stochastice APP au fost deja analizate în literatură, de asemenea concluzionăm că extensia schemelor de GraphSVM din [PaIr21] la varianta lor stochastică sau online este oarecum directă.

În [IRSP21] am pornit offline de la seturi de date nominale provenite din câteva noduri ale unui graf (cu aplicație la rețelele de distribuție de apă) asupra cărora am folosit metode de interpolare pentru a ajunge la un model nominal. Acest model este folosit apoi online asupra seturilor noi de date pentru a identifica anomaliile drept abateri de la modelul interpolat nominal.

În [IRP21] sunt propuși algoritmi de detecție de anomalii utilizând o nouă formă a problemei DL, în care se urmărește o selecție a atomilor (numită reprezentare uniformă) care sunt utilizați pentru reprezentarea semnalelor normale. Este deci o metodă parțial supervizată.

5. Îndeplinirea planului de realizare

Discuție despre planul de realizare și implementarea lui. Enumerăm mai jos activitățile și principalele rezultate obținute în cadrul lor.

Act. 2.1. Proiectare și implementare programe detecție anomalii cu DL. Redactare articol.

Lucrarea [RuIr21] prezintă algoritmi separabili de învățare de dicționar. De asemenea, în [IRP21] sunt prezentați trei algoritmi noi de detecție de anomalii folosind tehnici de antrenare de dicționar. O altă lucrare, bazată pe eșantionarea semnalelor utilizate în faza de reprezentare rară, urmată de actualizarea dicționarului folosind doar semnalele cu erori mici, este în curs de elaborare. Sec.1 prezintă alte detalii despre detecția de anomalii.

Act. 2.2. Adaptare și implementare metode generice de detectie de anomalii. Combinare metode

Rezultatele sunt descrise în Sec.1.3. Rezultate ale unor metode de vot sunt prezentate ín Sec. 3.1.3.

De asemenea, metodele SVM, PCA, Isolation Forrest apar în PyOD, TODS și SUOD ce au fost integrate în biblioteca Graphomaly. În cadrul lucrării [IRP21], în secțiunea experimentală, apar teste comparative ce utilizează biblioteca Graphomaly.

Act. 2.3. Proiectare algoritmi online și distribuiți de detecție de anomalii

Algoritmii de extragere de trăsături de tip random walk descriși în Sec.1.1.2 sunt prin natura lor distribuiți, fiecare parcurgere a grafului putând fi efectuată în paralel cu altele cu noduri de plecare diferite. De asemenea, algoritmii au forme online evidente; trăsăturile calculate cu mediere pot fi actualizate utilizând un factor de uitare; cele de tip max pot fi calculate pe baza unei liste (scurte) de valori.

De asemenea, algoritmii din [IRSP21] și [PaIr21] au fie caracter online, fie extensie directă la varianta online.

Act. 2.4. Testare și validare

Rezultatele sunt descrise în Sec.3. Am folosit date de la Libra Internet Bank și BRD, precum și date simulate.

Act. 2.5. Diseminare

Activitatea de diseminare prin publicații este descrisă în Sec.4. În plus, echipa UB a susținut mai multe prelegeri pe această temă în cadrul seminariilor științifice din cadrul Centrului de Cercetare pentru Logică, Optimizare și Securitate (LOS), la care au fost invitați participanți de la mai multe universități naționale și internaționale.

6. Alte informații despre proiect

Site-ul proiectului este http://graphomaly.upb.ro.

Cooperarea între parteneri și cu partenerii externi a fost constructivă. Am avut întâlniri online bi-săptămânale la care au participat majoritatea membrilor echipei și cel puțin un reprezentant al Libra Internet Bank. Datele puse la dispoziție de partenerii externi, în special Libra, au fost intens analizate și au constituit un excelent studiu de caz pentru care am alocat poate mai mult timp decât trebuia, în detrimentul generalității. Cu toate acestea, programele scrise sunt orientate către liste de tranzacții generale, care sunt reduse la o formă esențială și universală de graf.

7. Concluzii

Cu patru luni înainte de terminarea proiectului putem aprecia ca mulțumitor stadiul în care ne aflăm. Mai putem redacta 2-3 articole pe baza rezultatelor obținute până acum. O parte din software este deja public (dictlearn), iar biblioteca Graphomaly este în curs de finalizare a versiunii inițiale. Continuând în același ritm, putem încheia cu rezultate peste cele scontate la începutul proiectului.

Bibliografie

[AAB16] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[AEB06] M. Aharon, M. Elad, A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” IEEE Trans. Signal Proc., vol. 54, no. 11, pp. 4311–4322, 2006.

[AMF10] L. Akoglu, M. McGlohon, C. Faloutsos.“Oddball: Spotting anomalies in weighted graphs”. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 410–421, 2010.

[CC19] Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.

[DI18] B.Dumitrescu, P.Irofti - Dictionary Learning Algorithms and Applications, Springer, 2018.

[Ell19] A. Elliott et al. Anomaly detection in networks with application to financial transaction networks. *arXiv:1901.00402*, 2019.

[EAH99] K. Engan, S. O. Aase, and J. H. Husoy, “Method of optimal directions for frame design,” in IEEE Int. Conf. Acoustics Speech Signal Proc., 1999, vol. 5, pp. 2443–2446.

[GPBWW18] Gardner, Jacob R., Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration." arXiv preprint arXiv:1809.11165 (2018).

[GitSky] https://github.com/earthgecko/skyline

[KW13] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[LCL14] Liou, C.-Y., Cheng, W.-C., Liou, J.-W., & Liou, D.-R. (2014). Autoencoder for words. Neurocomputing, 139, 84–96.

[LZW21] K.-H. Lai, D. Zha, G. Wang, J. Xu, Y. Zhao, D. Kumar, Y. Chen, P. Zumkhawaka, M. Wan, D. Martinez and X. Hu (2021), *TODS: An Automated Time Series Outlier Detection System,* Proceedings of the AAAI Conference on Artificial Intelligence, 35(18), 16060-16062.

[LTZ08] F.T. Liu, K.M. Ting, Z.H. Zhou, Isolation Forest. In *8th IEEE International Conference on Data Mining*, pp. 413-422, Dec. 2008.

[MT08] H. D. K. Moonesinghe and P.-N. Tan, ‘‘Outrank: A graph-based outlier detection framework using random walk,’’ Int. J. Artif. Intell. Tools, vol. 17, no. 1, pp. 19–36, 2008.

[SciLe11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas. Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825-2830, 2011.

[WGZ18] C.Wang, H.Gao, Z.Liu, Y.Fu. “A new outlier detection model using random walk on local information graph”, *IEEE Access*, *6*, pp.75531-75544, 2018.

[WLL16] F. Wen, P. Liu, Y. Liu, R. C. Qiu, and W. Yu, “Robust sparse recovery in impulsive noise via ℓp-ℓ1 optimization,” IEEE Transactions on Signal Processing, vol. 65, no. 1, pp. 105–118, 2016.

[WiRa06] Williams, Christopher K., and Carl Edward Rasmussen. Gaussian processes for machine learning. Vol. 2, no. 3. Cambridge, MA: MIT Press, 2006.

[YK20] F.S. Yilmaz and S. Kozat (2020), *PySAD: A Streaming Anomaly Detection Framework in Python*, arXiv preprint arXiv:2009.02572.

[ZNL19] Y. Zhao, Z. Nasrullah and Z. Li, (2019). *PyOD: A Python Toolbox for Scalable Outlier Detection*. Journal of machine learning research (JMLR), 20(96), pp.1-7.

[ZHC21] Y. Zhao, X. Hu, C. Cheng, C. Wang, C. Wan, W. Wang, J. Yang, H. Bai, Z. Li, C. Xiao, Y. Wang, Z. Qiao, J. Sun, L. Akoglu (2021), *SUOD: Accelerating Large-Scale Unsupervised Heterogeneous Outlier Detection*, Proceedings of Machine Learning and Systems 3.

Director proiect,
Prof. Bogdan Dumitrescu