Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Mašininio mokymo metodų taikymas įtartinų transakcijų aptikimui

Baigiamasis magistro projektas

Matas Stonys

Projekto autorius

Asist. Dr. Vygandas Vaitkus

Vadovas

Kaunas, 2025

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Mašininio mokymo metodų taikymas įtartinų transakcijų aptikimui

Baigiamasis magistro projektas

Valdymo technologijos (6211EX014)

Matas Stonys

Projekto autorius

Asist. Dr. Vygandas Vaitkus

Vadovas

Asist. Dr. Andrius Knyš

Recenzentas

Kaunas, 2025

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Matas Stonys

Mašininio mokymo metodų taikymas įtartinų transakcijų aptikimui

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdamas kitų asmenų autoriaus ar kitų teisių, laikydamasis Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs;

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalintas iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Matas Stonys

Patvirtinta elektroniniu būdu

Stonys, Matas. Mašininio mokymo metodų taikymas įtartinų transakcijų aptikimui. Magistro baigiamasis projektas / vadovas asist. dr. Vygandas Vaitkus; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Elektronikos inžinerija (Inžinerijos mokslai).

Reikšminiai žodžiai: pinigų plovimo prevencija, neprižiūrimasis mašininis mokymasis, anomalijų aptikimas, kolektyviniai algoritmai.

Kaunas, 2025. 45 psl.

Santrauka

Šiame darbe aprašomas tyrimas apie mašininių modelių naudojimą įtartinų transakcijų aptikimui. Tyrimui atlikti naudojama IBM korporacijos *AMLworld* simuliatoriaus sugeneruota sintetinė duomenų bazė, atitinkanti realius pinigų plovimo scenarijus. Tyrimo tikslas palyginti neprižiūrimojo mašininio mokymosi modelius įtartinų transakcijų atpažinime.

Tyrime išskiriami statiniai transakcijų laiko požymiai, dinaminiai transakcijų dažnio požymiai, grupiniai požymiai (suskirsčius vartotojus į grupes pagal jų mokėjimo vidurkius). Požymiai atrenkami pagal svarbą ir koreliaciją su mokėjimo nelegalumo žyma. Atrinkti požymiai apdorojami *PCA* požymių redukcijos technika, balansuojami naudojant *SMOTE* algoritmą. Apdoroti požymiai klasifikuojami ekstra gradientinio stiprinimo, logistinės regresijos metodais.

Tiriamas alternatyvus požiūris į įtartinų transakcijų aptikimą, darant prielaidą, kad legalios transakcijos yra panašios ir su panašumo požymiais bus sutraukiamos į vieną ar kelis klasterius. Duomenų objektai skirstomi į klasterius naudojant *DBSCAN* klasterizavimo algoritmą, naudojant skirtingus adaptavimo parametrus. Klasterių išskirtys (*angl. outliers*) priimamos kaip nelegalios transakcijos. Taip pat taikomas ir anomalijų aptikimo algoritmas *Isolation forest*, išskiriantis anomalijas be klasterizavimo.

Klaidingai teisingų rezultatų sumažinimui naudojamas ansamblio algoritmas, kuris suteikia balsavimo teisę kiekvienam algoritmui ir nustato jo balsavimo svorį pagal *F1* kokybinį įvertį. Atlikus tyrimą nustatyta, kad *PCA* ir *SMOTE* algoritmai duoda teigiamus rezultatus pagerinant klasifikatorių spėjimą.

Stonys, Matas. Application of machine learning methods for suspicious transaction detection. Master's Final Degree Project / supervisor assist. Dr. Vygandas Vaitkus; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): Electronics Engineering (Engineering Science)

Keywords: anti-money laundering (AML), unsupervised machine learning, anomaly detection, ensemble algorithms.

Kaunas, 2025. 45 p.

Summary

This thesis describes research about machine learning models usage for suspicious transactions detection. The research was done using IBM corporation ‘s synthetic database generated with *AMLworld* simulator, which resembles real money laundering scenarios. The goal of research is to compare unsupervised machine learning models in suspicious transaction detection.

In the research, static time-based features are separated as well as dynamic frequency-based transactions and group features by dividing users to groups by their payment averages. Features are chosen by importance and correlation with money laundering label from dataset. Chosen features are preprocessed using *PCA* dimensionality reduction technique, data is balanced using *SMOTE* algorithm. Processed features are classified using gradient boosting, logistic regression classifiers.

An alternative approach to the research is also investigated, assuming that legal transactions are similar and with similar features they can be assembled to one or several clusters. Data objects are assigned to clusters using *DBSCAN* clustering algorithm, while using different fine-tuning parameters. Outliers are assumed to be illegal transactions in this approach. There is also used anomaly detection algorithm *Isolation Forest* which separate anomalies without clustering.

For reduction of false positive results an ensemble of algorithms is used which gives voting right to every algorithm and the weight of the vote is determined by the *F1* qualitative score. After conducting research, the conclusion is made that *PCA* and *SMOTE* algorithms give improvements on classification prediction.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Įvadas

Šių dienų pasaulyje, vis didėjanti bankinių transakcijų apimtis kelia didelius iššūkius aptinkant neteisėtą veiklą, kuri gali būti susijusi su teroristinių organizacijų finansavimu, mokesčių vengimu ir kitomis nelegaliomis veiklomis. Tradiciniai metodai ne visada yra pakankamai veiksmingi norint aptikti nuolat tobulėjančius nusikaltėlių metodus. Iš šių problemų kyla poreikis turėti našesnius, greičiau pritaikomus ir veiksmingesnius nelegalių transakcijų, kitaip dar vadinamo pinigų plovimo, aptikimo būdus.

Nusikaltėlių taktikos taikomos įvairios - jos nuolat kinta ir yra pritaikomos prie valstybės ir banko saugumo priemonių. Siekiant nuslėpti neteisėtą lėšų kilmę ir jų galutinį gavėją, taikomi įvairūs metodai. Dažniausiai pasitaikantys metodai yra: struktūrizavimas (*smurfinimas*) - didelių sumų išskaidymas į mažesnes, siekiant nepatekti į akiratį; sluoksniavimas - pinigų perkėlimas per ne vieną sudėtingesnę transakciją, įtraukiant ofšorines paskyras, kitas kompanijas, tam kad būtų neatsekama pinigų kilmė (Hearty, 2024). Populiarėjant kripto valiutoms, atsiranda ir įvairių unikalių pinigų plovimų būdų naudojant *blockchain*[1] technologiją (Trozze, Davies, & Kleinberg, 2023).

Naudojant tokius pinigų plovimo būdus, tuo pačiu stipriai nykstant grynųjų pinigų srautams ir atsirandant dideliems kiekiams kredito kortelių transakcijų, bankams tenka apdoroti neįtikėtino dydžio transakcijų kiekius kasdien. Rankiniu būdu to padaryti neįmanoma, tad pasitelkiami įvairūs skaitmeniniai būdai ir algoritmai tą atlikti. (Cabirta, 2019)

Dabartiniai įtartinų transakcijų aptikimo būdai naudojami bankuose, turi nemažų trūkumų, kalbant apie aptikimo tikslumą ir progresyvumą. Tradicinės taisyklių sistemos tampa nebeveiksmingos, nusikaltėliams skaidant lėšas, išsiaiškinant taisykles ar paprasčiausiai pervedant mažesnius kiekius, kad nesuveiktų saugumo sistemos. Taip pat tokios sistemos sukelia nemažai problemų verslams ar fiziniams asmenims operuojantiems didesnėmis sumomis, neretai susiduriant su užšaldytomis transakcijomis, kol jos būna peržiūrimos banko darbuotojų. Kai taikomi nesudėtingi statistiniai metodai, jie neturi galimybės aptikti progresuojančių sudėtingų schemų. Jie tinkami atpažinti tik momentinei transakcijai, kuri yra visiškai netipinė. Galiausiai, dauguma egzistuojančių metodų, nėra tinkami pritaikyti realiu laiku vykstančiam atpažinimui.

Darbo tikslas: palyginti neprižiūrimojo mašininio mokymosi modelius įtartinų transakcijų atpažinime.

Darbo uždaviniai:

1. nustatyti ar atrinktais požymiais pasiekiamas tinkamas klasifikavimo tikslumas;
2. nustatyti kokią įtaką klasifikavimo algoritmų tikslumui daro požymių dimensijų redukcijos ir imties balansavimo algoritmai;
3. įvertinti modelių greitaveiką naudojant požymių dimensijų redukciją;
4. palyginti kolektyvinius (algoritmų ansamblio) metodus su pavieniais algoritmais;
5. ištirti hipotezę, kad klasterių išskirtys gali būti prilyginamos nelegalioms transakcijoms.

[1] Blockchain – decentralizuota vieša transakcijų saugojimo sistema

1. Teorinė dalis

Šiame skyriuje apžvelgiama užsienio literatūra ir įvairūs atlikti tyrimai, siekiant išsiaiškinti kovos su nelegaliomis transakcijomis temos aktualumą, pasiekiamus duomenis tyrimui su mašininio mokymo metodais atlikti, seniau naudotas ir dabar populiarias technologijas šioje srityje. Gilinamasi į algoritmų pritaikymą ir galimas tyrimo kryptis ir metodus jo atlikimui.

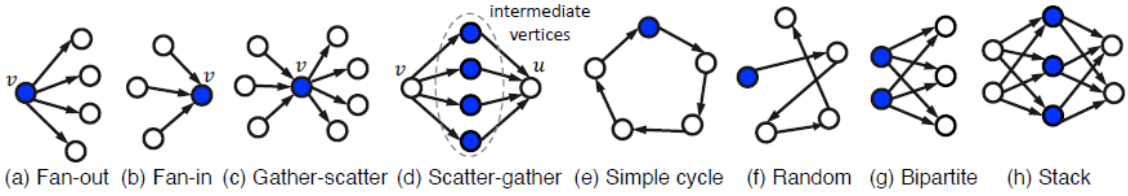
1.1. Temos aktualumas

Naudojant dabartines sistemas, dėl jų trūkumų, 2023 metais buvo skaičiuojama kad nelegalių pinigų metinė pasaulinė apyvarta gali būti iki 2-5% BVP ir gali siekti nuo 800 mln. iki 2 mlrd. JAV dolerių. Tuo tarpu iš viso šio kiekio aptinkama tik apie 1 % sumos. (Kulkarni, 2023)

Mašininio mokymo metodų pritaikymas įtartinų transakcijų aptikime suteikia galimybę lengviau pašalinti pagrindinius tradicinių kovos su pinigų plovimu (toliau - AML[2]) sistemų trūkumus. Dažniausiai literatūroje minimi trys kritiniai trūkumai, dėl ko tradicinės sistemos nebėra efektyvios.

Senosios taisyklėmis grindžiamos aptikimo sistemos, kurios taiko fiksuotas ribas transakcijos sumai ir kategorizuoja transakcijas pagal tipą (pvz., dažni tarptautiniai pervedimai), identifikuoja daugelį transakcijų kaip įtartinas iš kurių 95-99% yra klaidingai teigiamos. Jos apdorojamos analitikų, kurie sugaišta didelę dalį savo darbo laiko tikrindami tokius įspėjimus, dėl kurių nereikia imtis veiksmų (*angl. Non-Actionable alerts*) (Guidehouse, 2023).

Atsirandant naujoms pinigų plovimo taktikoms, tokioms kaip sluoksniavimas ir smurfingas (Lucinity, 2024), dabartiniuose modeliuose esančios statinės ribos nėra pritaikomos prie kintančių sąlygų ir apie vis mažiau sudėtingų schemų aptinkamos dabartiniais modeliais. Žinomos populiarios pinigų plovimo tipų schemos atvaizduotos 1 pav.



1 pav. Pinigų plovimo tipai (apskritimai - vartotojai, rodyklės – pervedimai)

(Altman, et al., 2024)

Nesusietos tarpusavyje sistemos taip pat trukdo efektyviam nelegalių transakcijų atpažinimui. (Elucidate team, 2024)

2022-aisiais įmonių ir valstybių patiriami nuostoliai dėl nelegalių lėšų siekė iki 41 mrld. JAV dolerių (Stripe, 2023), todėl reguliuojančiosios institucijos, tokios kaip centriniai bankai, remiasi ES DORA[3] ar panašiomis sistemomis ir skatina bankus taikyti mašininiu mokymu ar kitus efektyvesnius sprendimus ir taiko nuobaudas įstaigoms, kurių modeliai veikia neefektyviai. (Europos Sąjunga, 2022).

[2] AML – Kova su pinigų plovimu (*angl. Anti-Money Laundering*)
[3] DORA – Skaitmeninės veiklos atsparumo aktas (*angl. Digital Operational Resilience Act*)

Mašininis mokymas (toliau - ML[4]) leidžia sumažinti senųjų sistemų spragas pasitelkiant adaptyvų anomalijų aptikimą, kuris efektyviau nustato naujas tipologijas (modelis, apibūdinantis nusikalstamos veiklos struktūrą, pavyzdžiui smurfingas ar sluoksniavimas) (Doppalapudi, et al., 2022), tinklo grafų analizę, atskleidžiant daugiau netikrų banko paskyrų (*angl. mule accounts*) integruotose FRAML[5] sistemose, dinaminį rizikos vertinimą, kuris sumažina klaidingai teigiamų atvejų skaičių ir išlaiko panašų aptikimo tikslumą. (Guidehouse, 2023)

Technologiniai pokyčiai šioje sferoje turi būti įgyvendinami neatidėliojant. Finansų įstaigoms kasmet taikomos didesnės baudos už pinigų plovimo prevencijos pažeidimus, vien Europoje 2024 m. sumokėta 219 mln. JAV dolerių baudų, o Šiaurės Amerika užima lyderės vietą – sumoka apie 95 % pasaulinių baudų ir 2024 m. jos siekė 4,33 mlrd. JAV dolerių. (fenergo, 2024)

Tuo tarpu nusikaltėliai, taip pat pritaiko tobulėjančias technologijas ir vis dažniau naudodamiesi generatyviniu dirbtiniu intelektu, naudoja įmantresnius būdus apgauti žmones ir nuslėpti lėšų kilmę. Vis populiarėjantys DI[6] generuojami tekstai, nuotraukos, balso įrašai ir vaizdo klipai, kuriuos sunku atskirti nuo realybės, tampa dažnas jų įrankis. (FBI Alert, 2024) Perėjimas prie ML grįsto stebėjimo, tiek finansų srityje stebint transakcijas, tiek elektroninėje viešoje erdvėje, suteiktų galimybę lengviau pastebėti DI generuojamą turinį. Tyrimai rodo, kad ML diegimas finansų įstaigose gali reikšmingai sumažinti metines finansinių nusikaltimų prevencijos išlaidas, kurios siekia apie 214 mlrd. JAV dolerių (LexisNexis Risk Solutions, 2021), taip pat efektyviau kovoti su sparčiai augančiu giliosios dirbtinės tapatybės sukčiavimu (*angl. deepfake-assisted identity fraud*). (Preis, 2025)

1.2. Duomenų pasiekiamumas

AML tyrimuose dažnai naudojamos sintetinės duomenų bazės, nes realūs finansiniai duomenys yra retai prieinami dėl privatumo ir konfidencialumo. IBM sintetinė AML duomenų bazė (Altman, et al., 2024), sukurta naudojant *AMLworld* simuliaciją, yra viena iš pagrindinių, modeliuojanti virtualią ekonomiką su bankais, individais ir įmonėmis, įskaitant pinigų plovimo veiklas, kaip kontrabanda ir turto prievartavimas, su iki 180 mln. operacijų tam tikrose konfigūracijose. Duomenų bazė atitinka realius sandorius ir turi etiketes su informacija ar tai yra nelegali transakcija. Taip išsprendžiama žymų nebuvimo problemą, kuri riboja realių duomenų panaudojimą šioje srityje, kai daugelis neteisėtų veiklų lieka nepastebėtos ir nėra pažymimos kaip nelegalios.

Kitos duomenų bazės, kaip IEEE-CIS[7] (590 540 operacijų, 3,5 % sukčiavimo atvejų) ir Kreditinių kortelių sukčiavimo[8] (284 807 operacijos, 0,17 % sukčiavimo atvejų), yra skirtos kortelių sukčiavimo tyrimams, tuo tarpu AML srities tyrimai apima platesnį spektrą, kur kortelių mokėjimai yra tik viena iš daugelio mokėjimo formų.

Realių finansinių transakcijų duomenų bazių gavimas yra sudėtingas dėl duomenų apsaugos ir riboto jų prieinamumo. Egzistuojantys tyrimai taip pat yra konfidencialūs ir duomenys neskelbiami. Atlikus tyrimą su realiais duomenimis, palyginimo galimybės su kitais tyrimais labai ribotos.

[4] ML – Mašininis mokymas(is) (*angl. Machine Learning*)
[5] FRAML – Sukčiavimo prevencija ir kova su pinigų plovimu (*angl. FRaud prevention + Anti-Money Laundering*)
[6] DI – Dirbtinis intelektas
[7] https://www.kaggle.com/competitions/ieee-fraud-detection/
[8] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Egzistuoja ir kitos sintetinių duomenų bazių alternatyvos, kaip J.P. Morgan DI ir Amazon FDB duomenų bazės, kurios taip pat nemažai naudojamos, tačiau populiariausia lieka IBM AML duomenų bazė.

Remiantis nagrinėtomis duomenų bazėmis, pastebima, kad transakcijų duomenys yra panašūs visose. Pavyzdžiui IBM AML sintetinės transakcijos turi šiuos pradinius požymius:

- Laiko žyma: transakcijos data ir laikas
- Siuntėjo ir gavėjo banko ID
- Siuntėjo ir gavėjo sąskaitų ID
- Transakcijos suma
- Mokėjimų valiuta
- Mokėjimų tipas (čekis, pavedimas, kreditinė kortelė, kt.)
- Etiketė žyminti transakcijos legalumą

1.3. Požymių inžinerija

Ankstesniame skyrelyje minėti požymiai atveria galimybę naudojant požymių inžineriją išgauti tiek statinius (pvz., sąskaitos atributus), tiek dinaminius (pvz., operacijų modelius) išvestinius požymius. Duomenų bazės dydis ir etiketės („Ar plovimas“) daro ją tinkamą nenustatyto mokymosi tyrimams, kur etiketės ignoruojamos mokymo metu ir naudojamos validacijai. Šioje sekcijoje aptariami požymiai, kurie naudojami kituose AML srities tyrimuose apmokant mašininio mokymosi modelius.

FATF korporacija 2020m. išleistoje atskaitoje išskiria požymius, kurie nustatyti iš daugiau nei 100 tyrimų kaip indikuojantys didelę riziką (FATF, 2020). Šia ataskaita formuojant požymius mašininio mokymosi metodams remiasi ir tyrimai, pasiekiantys gerus rezultatus įtartinų transakcijų aptikime. (Fan, et al., 2025)

1.3.1. Pradiniai požymiai

Pradiniai požymiai nėra itin informatyvūs patys iš savęs tokiame tyrime, nes pagrindiniai transakcijos meta-duomenys[9] nesuteikia beveik jokios informacijos apie transakcijos legalumą, jos tikslą ar vartotojo įpročius. Tačiau naudojantis pradinių požymių visuma, galima įvertinti vartotojo, vartotojų grupės elgsenas ir kaip konkreti transakcija atrodo bendroje transakcijų visumoje. Iš pradinių požymių vedami statiniai ir dinaminiai požymiai kurie atitinkamai atspindi ne tik pačios transakcijos ypatumus, bet ir vartotojo ar vartotojų grupės transakcijų ypatumus ilgalaikėje perspektyvoje.

AML srityje pastebima, kad požymiai skirstomi į keletą kategorijų, kurių kiekviena atskleidžia skirtingą informaciją apie transakciją. Pagrindinės požymių skirstymo kategorijos yra: pradiniai, statiniai, dinaminiai. Dinaminiai požymiai dažnai skirstomi į mažesnes kategorijas pagal tai, kokią informaciją atspindi.

1.3.2. Statiniai požymiai

Statiniai požymiai yra išvestiniai tiesiogiai iš konkrečios transakcijos duomenų ir nereikalauja istorinių duomenų. Dėmesys sutelkiamas į transakcijos charakteristikas, tokias kaip standartizuota suma (suvienodinama valiuta taikant valiutų kursus), laiko požymiai (valanda paroje, savaitės diena, savaitgalio žyma). Kai kurie kategoriniai požymiai koduojami skaičiais, kad būtų galimas

[9] Meta-duomenys (*angl. metadata*) – duomenys, suteikiantys daugiau informacijos apie duomenis – autorius, laikas, gavėjas ir kt.

panaudojimas tolimesniuose skaičiavimuose bei mašininio mokymosi modeliuose, Pavyzdžiui savaitės diena žymima skaičiais nuo 0 – pirmadienis iki 6 – sekmadienis, tuomet savaitės dienos vertei esant 5 arba daugiau, išvedama dvejetainė žyma savaitgalis (0 – ne, 1 – taip). Tokiu principu iš pradinių duomenų išskiriami įvairūs statiniai požymiai aprašantys transakcijos specifiką – informacija apie naudojamą valiutų kombinaciją, ar transakcija atliekama tame pačiame banke, koks yra mokėjimo tipas bei kt.

1.3.3. Dinaminiai požymiai

Dinaminiams požymiams sukurti reikalinga platesnė informacija, nei viena transakcija. Dinamikai stebėti reikalingas istorinių duomenų kaupimas. Naudojant transakcijų archyvą stebima lėšų kilmė, jų kelias ir galutinis vartotojas. Tokiu būdu gali būti kuriamas vartotojo, vartotojų grupės profilis bei sudaromas vartotojų tinklo struktūra.

Naudojant dinaminius požymius lyginama konkreti transakcija su vartotojo ar vartotojo grupės elgesio modeliu. Lyginant transakciją su tipine vartotojo ar su grupės, kuriai priklauso vartotojas elgsena, vertinamas netipinis elgesys ir generuojami požymiai žymintys nukrypimus nuo tipinės vartotojo ar grupės charakteristikos (Fan, et al., 2025).

Stebint vartotojų tinklo struktūrą ir ją analizuojant, nustatoma lėšų kilmė, jų kelias. Šie požymiai yra svarbūs nustatant sudėtingas pinigų plovimo schemas tokias kaip sluoksniavimas ar smurfingas. Dinaminiai požymiai skirstomi į keletą kategorijų, kurios suteikia skirtingą informaciją bei gali būti skaičiuojamos pasitelkiant skirtingas technikas.

Paskyros lygio požymiai

Šio tipo požymiais nagrinėjami su paskyra bei transakcijos laiku ir dažniu susiję požymiai. Skaičiuojama transakcijų suma per dieną (arba kitą laiko periodą), tuomet vedamas to laiko periodo vidurkis, mediana, skaičiuojamas praėjęs laikas nuo paskutinės transakcijos. Taip pat naudojant statinius požymius tokius kaip savaitės diena ar valanda, tikrinama ar transakcija atliekama vartotojui įprastu metu.

Paskyros lygio požymiai apima ir valiutų, mokėjimų tipų, mokėjimo sumų, banko ir paskyros sąsajų požymius. Iš šių požymių išvedamos reikšmės, nurodančios ar naudojama tokia pati valiuta mokėjimą siunčiant ir gaunant, ar mokėjimas atliekamas tame pačiame banke, toje pačioje paskyroje. Tokie požymiai leidžia mašininio mokymo modeliams turėti daugiau informacijos apie riziką.

Tinklo požymiai

Tinklo požymiai atspindi vartotojo transakcijų sąveiką su kitais vartotojais. A. F. Colladon tyrime (Colladon & Remondi, 2017) pastebima, kad naudojami tokie požymiai kaip į sąskaitą gautų ir išeinančių operacijų skaičius parodantis lėšų surinkimą arba paskirstymą. Skaičiuojamas rodiklis „tarpiškumo centriškumas“ (*angl. betweenness centrality*), pagal kurį nustatomas kaip dažnai sąskaita yra trumpiausiame kelyje tarp kitų sąskaitų. *Pagerank* rodiklis vertina paskyros įtaką transakcijų erdvėje, kuriuo galima išskirti pagrindinius veikėjus. Galiausiai naudojant tokius algoritmus kaip *Louvain*, nustatomos narystės bendruomenėje ir išryškinamos dažnai sandorius atliekančių sąskaitų grupės. Visi šie rodikliai rodo kad vartotojai tinkle gali būti susiję daugiau nei įprastai.

Grafomis paremti požymiai

Šie patys sudėtingiausi požymiai, neša daug reikšmingos informacijos apie lėšų kilmę ir jų kelią nuo pradinio iki galutinio kliento. Nagrinėtame tyrime (Altman, et al., 2024), naudojant grafų algoritmus atpažįstami išskaidymo-sutelkimo pinigų plovimo algoritmai (*angl. scatter-gather*) (pavaizduota 1 pav.), sekant 6 valandų slenkantį laiko langą. Taip pat laikini ciklai (*angl. temporal cycle*) gali būti atpažįstami su slenkančiu vienos dienos laiko tarpu, kai lėšos grįžta pas pradinį jų siuntėją.

Tokie požymiai yra itin svarbūs atpažįstant sudėtingus pinigų plovimo algoritmus, tačiau šiems požymiams išgauti reikalingi galingi skaičiavimo resursai. Taip pat dinaminiai požymiai duoda didelės naudos, kai turimas ilgas periodas duomenų istorijos iš kurio galima sukurti vartotojo profilį ir aprašyti jo elgseną. Tačiau, jei istorija yra trumpa arba jos nėra, dinaminiai požymiai nėra itin naudingi, nes vartotojo elgsena neapibrėžta.

1.4. Vartotojų segmentavimas

Dėl kiekvienos paskyros naudotojo skirtingų ypatumų, dažnai sutinkamas vartotojo profilio kūrimas, kuris aprašo vartotojo metrikas. Jame aprašomi požymiai, apibūdinantys vartotoją, tokie kaip – mokėjimų dažnis per dieną, laikas tarp transakcijų, vidutinė mokėjimo suma, vertinamas paskyros aktyvumas (Fan, et al., 2025). Tokie požymiai skaičiuojami kaip vidurkis arba mediana, jei duomenų imtis nedidelė ir iš duomenų nėra pašalinti ekstremumai.

Išskyrus vartotojų požymius, pagal juos vartotojai gali būti skirstomi į grupes. Vartotojų grupavimas naudingas tuomet, kai duomenų imtis yra maža arba vartotojų mokėjimai reti ir sudėtinga apskaičiuoti vieno vartotojo dinamiką. Tokiu atveju imant grupę vartotojų, kurių elgsena panaši, skaičiuojami tokie patys grupės požymiai, kaip ir vartotojo ir vartotojams bei transakcijoms priskiriant žymą, kokiai grupei jos priklauso.

1.5. Požymių kodavimas

Dalis požymių yra išreiškiami kategorinėmis reikšmėmis arba skaičiais, kurie neturi sąsajos. Tokie požymiai nėra tinkami naudoti mašininio mokymosi modeliuose. Šiai problemai spręsti naudojami kodavimo (*angl. encoding*) algoritmai tokie kaip *one-hot*, *label*, arba *ordinal* (Pedregosa, et al., 2011).

Kategoriniai duomenys skaidomi į dvejetainius arba skaitinius stulpelius. Naudojant *one*-hot kodavimą, kiekvienai unikaliai vertei požymio stulpelyje sukuriami atskiri stulpeliai, turintys dvejetainę reikšmę 0 arba 1. Tuo tarpu *label* ir *ordinal* tipo kodavimas pasižymi kategorinės reikšmės keitimu į skaitinę išlaikant tą pačią požymių dimensiją. Duomenys stulpelyje konvertuojami iš kategorinės reikšmės į skaitinę, kiekvienai unikaliai reikšmei priskiriant kitą skaitinę reikšmę.

Skirtumas tarp *ordinal* ir *label* kodavimo algoritmų yra tas, kad *ordinal* kodavimo metu papildomai atsižvelgiama į duomenų hierarchiją ir gali būti išlaikomas svarbos eiliškumas konvertuojant duomenis, pavyzdžiui reikšmės „retas“, „vidutinis“, „dažnas“ gali būti atitinkamai koduojamas skaičiais 1, 2 ir 3, išlaikant svarbą jog kuo didesnis skaičius, tuo dažnesnis reiškinys.

Duomenų kodavimas leidžia mašininio mokymosi algoritmas tiksliau įvertinti reikšmę ir sąsajas tarp požymių, taip tiksliau apskaičiuojant svorius klasifikatoriaus apmokymo metu ar tiksliau klasterizuojant duomenis. Tačiau kiekvienas iš šių kodavimų turi savų trūkumų ir tinka ne visiems algoritmams. *One-hot* kodavimas tinkamas naudoti su algoritmais, kurie nevertina hierarchinės verčių

sąsajos, tokiems kaip loginė regresija, neuroniniai tinklai ar k-artimiausių kaimynų. Jis sutinkamas ir AML srities tyrimuose (Fan, et al., 2025). Tuo tarpu *label* kodavimas tinkamiausias medžio topologija grįstiems modeliams. (Sharma, 2024)

1.6. Požymių redukcijos algoritmai

Nagrinėjant transakciją ir jos meta duomenis turime apie 10 požymių. Įtraukiant statinius ir dinaminius požymius, jų skaičius gali padidėti nuo kelių iki keliolikos kartų. Tokį kiekį požymių apdoroti yra neefektyvu. Tokį kiekį duomenų apdoroti klasterizavimo ar klasifikavimo algoritmu užima daug skaičiavimo resursų ir laiko, taip pat panašūs požymiai perkrauna sistemą savo svoriais ir daro didelę įtaką, kai tas nėra reikalinga. Šiaip problemai spręsti pasitelkiamos dvi technologijos – požymių pašalinimas ir požymių redukcija.

1.6.1. Požymių apdorojimas

Skaičiuojant įvairius parametrus tarp požymių, natūralu, kad kai kurie požymiai kinta priklausomai vienas nuo kito. Tokie požymiai neša vienodą informaciją ir dubliuoja vienas kito informaciją. Tokiu atveju naudojamos įvairios technikos atpažinti koreliuojantiems požymiams ir juos pašalinti.

Medium tinklaraštyje (Yemulwar, 2019) pateikiami keli požymių atrinkimo būdai:

- Filtravimo metodas – naudojant koreliacijos koeficientus (pvz. Pearson‘o) eliminuojami požymiai, turintys didžiausią koreliacijos koeficientą tarpusavyje;
- Apvalkalo metodas – naudojant mašininio mokymo algoritmą požymiai po vieną pridedami (turint vieną) arba šalinami (turint visus), kol algoritmo rezultatai pasiekia piką ir pradeda prastėti.
- Integruotas metodas – iteracinis metodas, kai modelis apmokomas daug kartų ir požymių, kurie neduoda gerų rezultatų, svoriai yra mažinami iki kol jie pasiekia 0 ir yra pašalinami.

Požymius atrinkti aktualu tuomet, kai jų yra itin daug ir daugelio požymių koreliacija yra 1 arba labai arti 1. Jei požymių nėra daug ir koreliacija nėra lygi vienetui, jų eliminavimas nėra būtinas. Kai kurie požymiai nepaisant didelės koreliacijos neša šiek tiek skirtingą informaciją, kuri išskiriama kitais būdais, pavyzdžiui dimensijų redukcijos algoritmu.

1.6.2. Požymių standartizavimas

Duomenų standartizavimas naudojamas duomenų nešamos informacijos svarbai suvienodinti. Kadangi vieni požymiai, tokie kaip mokėta suma, gali stipriai skirtis tarp transakcijų (pavyzdžiui minėtoje IBM AML duomenų bazėje[10] mokėtina suma svyruoja nuo 0.01 iki 1046302363293.48), tuo tarpu dvejetainiai požymiai turi tik vertes 0 ir 1, atliekant matematinius veiksmus su jais, tokių požymių svoriai itin skiriasi, nors svarba gali būti ir vienoda. (Jaadi, Whitfield, & Sawtell-Rickson, Data Standardization: How to Do It and Why It Matters, 2025) Kad jų nešamos informacijos svarba būtų vienoda, jiems taikomas standartizavimas.

[10]https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml/data?select=HI-Small_Trans.csv

$$z = \frac{x - u}{s}; \quad (1)$$

Standartizavimas – tai visų reikšmių transformavimas, kuomet duomenų vidurkis tampa nuliu, o standartinis nuokrypis – 1. Standartizuotus duomenis galima tarpusavyje palyginti naudojant tą pačią skalę. Reikšmių standartizavimui, suskaičiuojamas požymio verčių vidurkis u ir jų standartinis nuokrypis s, tuomet turint x vertę pagal 1 formulę apskaičiuojama standartizuota vertė z. (Pedregosa, et al., 2011)

1.6.3. Duomenų balansavimas

Naudojant nesubalansuotus duomenis mašininio mokymo modelių apmokymui, dažnu atveju nėra pasiekiama gerų rezultatų. Modelis negeba teisingai klasifikuoti duomenų tai klasei, apie kurią mokymosi metu turėjo mažai informacijos. Šiai problemai spręsti naudojami duomenų balansavimo metodai. (Jensen & Iosifidis, 2023) Duomenų balansavimui gali būti naudojami skirtingi metodai, priklausomai nuo to kaip norima duomenis balansuoti. Tipiniai būdai yra pavyzdžių perteklinis įtraukimas (*angl. oversampling*) arba imties retinimas (*angl. undersampling*).

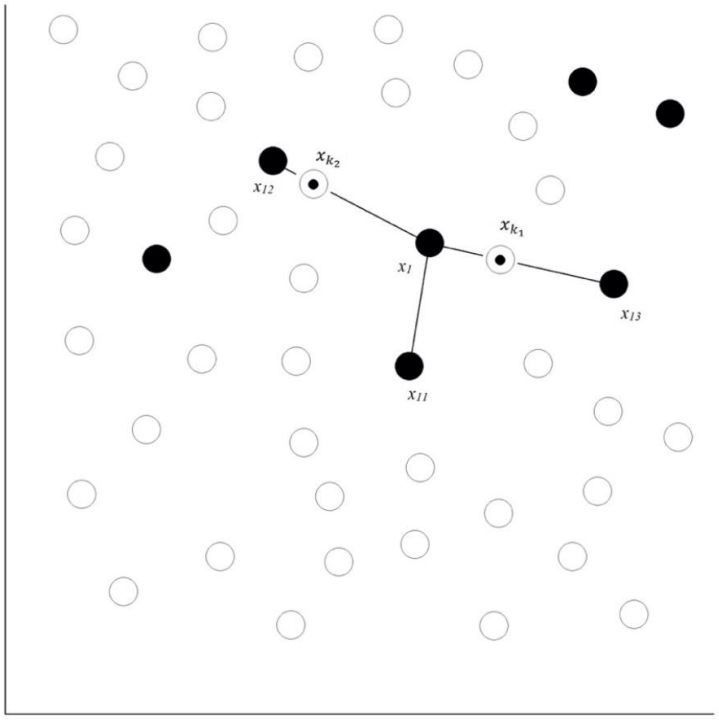
Duomenys gali būti atsitiktiniu būdu įtraukiami dubliuojami arba eliminuojami atitinkamai naudojant ROS (*angl. Random Over-Sampling*) arba RUS (*angl. Random Under-Sampling*) metodus. Taip pat gali būti naudojami ir kiti metodai, tokie kaip SMOTE ar ADASYN, kurie generuoja sintetinius duomenis mažiausioje klasėje. (Viloria, Lezama, & Mercado-Caruzo, 2020)

Sintetinė mažumos imties tankinimo technika (SMOTE)

Sintetinė mažumos imties tankinimo technika – SMOTE (*angl. Synthetic Minority Over-sampling technique*), tai metodas generuojantis sintetinius duomenų objektus duomenų klasių balansavimui. SMOTE algoritmas veikia k-artimiausių kaimynų algoritmo pagrindu (apie k-NN algoritmą žr. skiltį „Klasifikavimo metodai“). Sintetinių duomenų kiekis yra nustatomas iš anksto ir atspindi duomenų disbalanso lygį.

$$x_k = x_i + \lambda(x_i - x_{ij}); \quad (2)$$

Priimant, kad k nurodo artimiausių taškų kiekį, N – sintetinių duomenų kiekį, $x_i, i = 1, \dots, n_s$, - mažumos imties duomenis, o A visą mažumos imtį ($A \ni x_i$), tuomet tarp kiekvieno x_i imtyje A apskaičiuojamas euklidinis atstumas ir randami artimiausi x_i k-artimiausi kaimynai. Artimiausi kaimynai aprašomi S_{ik} imtimi. N sintetinių duomenų objektų yra sugeneruojami atsitiktiniu būdu, kurie žymimi $x_{ij}, (j = 1, \dots, N)$. Kai $\lambda = [0\,, 1]$, tuomet sintetiniu būdu sugeneruojami taškai apskaičiuojami pagal 2 formulę. (Brandt & Lanzén, 2020) Pagal šią formulę apskaičiuoto pavyzdžio iliustracija pateikiama 2 paveikslėlyje.



2 pav. SMOTE algoritmo iliustracija nesubalansuotuose duomenyse 2D erdvėje, kai juodi taškai yra mažumos klasė ir x_{k_1}, x_{k_2} sintetiniai duomenys yra atsitiktiniu atstumu tiesioje linijoje tarp artimiausių kaimyninių taškų
(Brandt & Lanzén, 2020)

Adaptyvus sintetinis mėginių kūrimo požiūris (ADASYN)

ADASYN (*angl. Adaptive Synthetic sampling approach*) – algoritmas, sukurtas SMOTE pagrindu, generuojantis sintetinius duomenų mėginius mažiausios imties klasei. Šie mėginiai taip pat yra išdėstyti požymių erdvėje tiesėje tarp dviejų artimiausių imties taškų, tačiau pagrindinis skirtumas tarp SMOTE ir ADASYN algoritmų yra tas, jog pastarasis generuoja sintetinius duomenis panašesnius į tuos duomenų objektus, kurie turi didesnės klasės požymių ir yra algoritmams yra sudėtingiau mokymo metu išmokti atpažinti šiuos taškus. Tokiu būdu skiriamas mažesnis dėmesys taškams, kurie lengvai atpažįstami ir daugiau tiems, kuriuos atpažinti sudėtinga. (Brandt & Lanzén, 2020)

1.6.4. Požymių redukcijos technikos

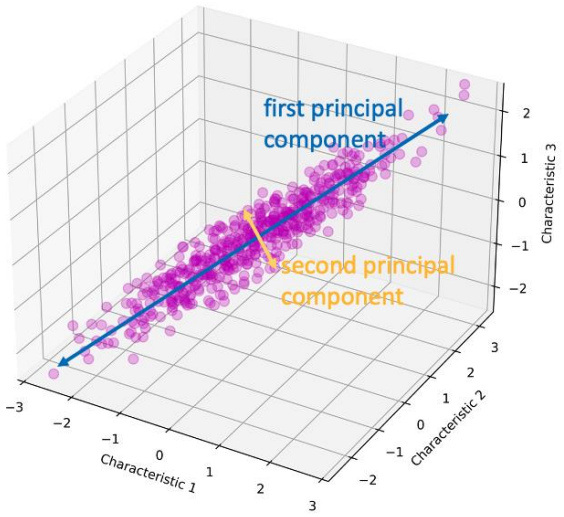
Pašalinus požymius, kurie koreliuoja tarpusavyje, lieka nemažai požymių, nešančių skirtingą informaciją. Kai požymių kiekis yra didelis, juos sudėtinga efektyviai apdoroti mašininio mokymosi metodais, dėl to naudojami požymių dimensijos redukcijos algoritmai. DRT[11] algoritmai transformuoja daugelio dimensijų požymių erdvę į mažesnių dimensijų, dažnu atveju iki 2D ar 3D erdvės išlaikant kritiškai svarbią informaciją. Tokiu būdu pagerinamas mašininio mokymosi klasifikavimo algoritmų efektyvumas ir greitaveika, bei atsiranda galimybė duomenis lengvai vizualizuoti.

[11] DRT – Dimensijų redukcijos technikos (*angl. Dimensionality Reduction Techniques*)

AML tyrimuose dažnai naudojamas tiesinis PCA[12] algoritmas ir netiesiniai metodai t-SNE[13] ir UMAP[14]. Pastarieji du dažnai naudojami, nes redukuojant požymius jais, vis dar įmanoma užfiksuoti sudėtingus sandorių modelius. t-SNE algoritmas leidžia sudaryti požymių erdvės žemėlapius dvimatėje arba trimatėje erdvėje, glaudžiai sugrupuodamas panašias transakcijas ir vizualizuoja klasterius, kurie rodo pinigų plovimo modelius pavyzdžiui greitus lėšų pervedimus arba žiedines operacijas.

Principinių komponenčių analizė

Principinių komponenčių analizė (*angl. principal component analysis*) (toliau – PCA) – tai klasikinis metodas, naudojamas duomenų analizėje duomenų transformacijai. Naudojantis PCA statistiniu metodu sumažinamos daugiamačių duomenų dimensijos. Metodas veikia didžiausios dispersijos krypties paieškos principu. Didžiausią dispersiją turinti kryptis vadinama pirmąja komponente, dažnai žymima PC1, toliau seka kitos komponentės atitinkamai turinčios vis mažesnę dispersiją, kaip pavaizduota 3D požymių erdvėje 2 komponenčių iliustracijoje 3 paveikslėlyje (Umbražiūnaitė, 2005)



3 pav. Principinių komponenčių išsidėstymas požymių erdvėje (pavyzdys) (Follette, 2023)

Principinių komponenčių esmė, kad visos komponentės turi kirsti centrinį duomenų tašką bei būti ortogonali prieš tai buvusioms komponentėms (nekoreliuoti su jomis). Esminė šio metodo idėja – sumažinti duomenų dimensiją kiek galima labiau, tuo pačiu išlaikant maksimalią duomenų dispersiją. Dažniausiai komponenčių skaičius parenkamas pagal suminę komponenčių dispersiją, pasirinkus minimalų suminį dispersijos lygį, pavyzdžiui 90 % arba 95 %.

PCA metodo taikymui duomenys pirmiausia standartizuojami pagal *z*-standartizavimą. Tuomet skaičiuojama simetrinė kovariacijos matrica $p \times p$ kurios dydis p atitinka duomenų dimensijų skaičių. Galiausiai suskaičiuojami kovariacijos matricos nuosavieji vektoriai (*angl. eigenvector*), kurie nurodo kryptis ašių, kuriose yra daugiausiai informacijos. Šias ašis vadiname principinėmis

[12] PCA – Principinė komponenčių analizė (*angl. Principal Component Analysis*)
[13] t-SNE – t-skirstomasis stochastinis kaimyninis įterpimas (*angl. t-Distributed Stochastic Neighbor Embedding*)
[14] UMAP – vientisas daugiamatis aproksimavimas ir projekcija (*angl. Uniform Manifold Approximation and Projection*)

komponentėmis. (Jaadi, Whitfield, & Pierre, Principal Component Analysis (PCA): A Step-by-Step Explanation, 2024)

1.7. Klasifikavimo metodai

Klasifikavimo metodai plačiai naudojami siekiant išskirti įtartinas transakcijas iš didelio transakcijų kiekio. Klasifikatoriai – tai prižiūrimojo mašininio mokymosi modeliai, pagal įvesties duomenis priskiriantys kategorijas. Naudodami apmokymo duomenų rinkinį, kurį sudaro požymiai ir duomenų klasės, šie algoritmai sugeba aptikti duomenyse slypinčias struktūras ir apibrėžti funkcijas, nurodančias duomenų klasifikavimo šabloną. Klasifikatoriui pateikus naujus požymius be klasių informacijos, algoritmas šiems duomenims priskiria labiausiai tikėtiną klasę.

Logistinė regresija

Logistinė regresija (angl. *logistic regression*) – vienas iš paprasčiausių klasifikavimo modelių. Nors logistinė regresija yra tiesinės regresijos tipas, prognozuojantis skaitines, o ne diskretines, reikšmes, dažnai šis algoritmas priskiriamas klasifikavimo metodams. Tai paprastas ir nedaug resursų naudojantis algoritmas, tinkamas tiek binariniam, tiek kelių klasių klasifikavimui. Jis yra gana populiarus ir plačiai sutinkamas AML srities tyrimuose (Harris, Pyndiura, Sturrock, & Christensen, 2021).

Paprastosios tiesinės regresijos modelis yra aprašomas lygtimi:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n = x\beta, \text{ kur } \beta = [\beta_0\ \beta_1 \dots \beta_n] \text{ ir } x = [1\ x_1 \dots x_n] \quad (3)$$

$x_1, x_2, \dots, x_n$ yra nepriklausomi kintamieji, kurie, parinkus tinkamus svertinius koeficientus $\beta_1, \beta_2, \dots, \beta_n$ bei laisvąjį narį β_0, leidžia nuspėti priklausomąjį kintamąjį y.

Logistinės regresijos atveju, šis rezultatas paverčiamas tikimybe naudojant logistinę funkciją, kuri leidžia bet kokią skaitinę reikšmę suspausti į intervalą $[0, 1]$:

$$P(y = 1|x) = \frac{1}{1 + e^{-x\beta}}; \quad (4)$$

Binarinio klasifikavimo atveju, yra pasirenkamas ribinis slenkstis ir pagal jį ši tikimybė yra naudojama priskirti duomenis vienai iš dviejų klasių. Dažniausiai naudojamas slenkstis yra 0.5, tačiau jis gali būti keičiamas priklausomai nuo konkretaus taikymo konteksto ir norimo jautrumo bei specifiškumo

Gradientinis stiprinimas

Gradientinis stiprinimas (angl. *gradient boosting*) – kolektyvinio mašininio mokymosi metodas, kuris kuria stiprų modelį jungdamas daug silpnų paprastų modelių (dažniausiai mažų sprendimų medžių). Šie modeliai apmokomi vienas po kito, ir kiekvienas iš jų atsižvelgia į praeito modelio padarytas klaidas. Šis algoritmas vadinamas gradientiniu, kadangi kiekviename žingsnyje naudojama klaidų funkcijos išvestinė (gradientas) ir pagal jį koreguojamas modelis, taip judant link minimalaus klaidų kiekio. Įvairūs gradientinio stiprinimo algoritmai, pavyzdžiui *AdaBoost*, plačiai taikomi ir AML srityje. (Venčkauskas, Grigaliūnas, Pocius, Brūzgienė, & Romanovs, 2025)

Gradientinio stiprinimo modelių kompleksiškumas priklauso nuo iteracijų skaičiaus. Kiekvienos iteracijos metu konstruojamas paprastas binarinis klasifikatorius $G_i: \mathbb{R}^p \to \{-1, 1\}$, įvertinamas jo tikslumas α_i ir perskaičiuojami duomenų aibės objektų svoriai $w_i = [w_{i1}\ w_{i2} \dots w_{ip}]$ taip, kad neteisingai

suklasifikuotų objektų svoris padidėtų. Sekantis modelis G_{i+1} apmokomas naudojant šiuos įvesties duomenų svorius, taip padidinant neteisingai suklasifikuotų objektų svarbą. Galutinis modelis išvedamas pagal svertinį visų tarpinių modelių vidurkį:

$$G(X) = sign\left(\sum \alpha_i G_i(X)\right); \quad (5)$$

Gradientinio stiprinimo metodai pasižymi keletu stiprybių. Pavyzdžiui, dėl didelio įvairių medžių kiekio, šie algoritmai retai persimoko (angl. *overfit*). Didelis iteracijų skaičius leidžia išgauti beveik tobulą apmokymo tikslumą, be to, toliau tęsiant modelio mokymą testavimo tikslumas gali toliau augti.

Atsitiktinis miškas

Atsitiktinis miškas (angl. *random forest*) priklauso maišytinių (angl. *bagging*, trumpinys iš *bootstrap aggregating*) ansamblinių mokymosi metodų grupei. Šie metodai apjungia daug modelių, kurie apmokomi naudojant atsitiktinius pakartotinius imties pogrupius (angl. *boostrap samples*). Įprastai maišytiniai metodai taikomi modeliams su dideliu šališkumu (angl. *bias*) ir maža variacija, kuriuos apjungus modelių šališkumas sumažėja, o variacija išlieka nežymi. Galiausiai, objektui priskiriama visų modelių dažniausiai nuspėjama klasė:

$$G(X) = \frac{1}{B}\sum_{b=1}^{B} G_b\,(X); \quad (6)$$

Čia B – apmokomų modelių skaičius, G_b – modelis, apmokytas naudojant atsitiktinį pakartotinį (angl. *bootstrapped*) imties pogrupį.

Atsitiktinis miškas apjungia daug pilnų (visiškai išsišakojusių) sprendimo medžių (angl. *full decision tree*), sugeneruotų su papildomu atsitiktinumu, kuris leidžia sumažinti jų panašumą ir tarpusavio priklausomybę (Breiman, 2001). Tarkime, kad pilną duomenų imtį sudaro n objektų ir p požymių. Kiekvienam medžiui apmokyti yra naudojamas duomenų imties pogrupis, sugeneruotas atsitiktinai pasirinkus n objektų su pakartojimais. Vadinasi, į šį pogrupį nepateks $\sim 1/e$ duomenų objektų. Be to, kiekvienam medžiui leidžiama naudoti tik $m \ll p$ požymių. Apmokymo metu, medžiams leidžiama augti iki galo, t. y., be jokių gylio ir maksimalaus lapų dydžio apribojimų. Įprastai tokie medžiai yra permokyti, tačiau, juos sujungus į ansamblį, bendras modelis pasižymi geru apibendrinimu ir aukštu testavimo tikslumu.

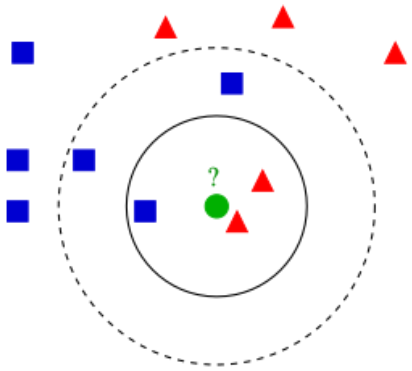
Ekstremalus gradientinis stiprinimas

Ekstremalus gradientinis stiprinimas (*angl. extreme gradient boosting*) toliau – XGBoost, tai gradientinio stiprinimo algoritmo atmaina, turinti didesnį našumą. Šis gradientinio stiprinimo algoritmo variantas skiriasi nuo originalaus, nes turi reguliarizaciją, kuri apsaugo nuo duomenų permokymo, įvedant L1/L2 baudas kiekvieno medžio svoriams ir šališkumo poslinkiams. Taip pat šis algoritmas geba apdoroti retus duomenų rinkinius naudojant svertinio kvantilinio eskizo (*angl. weighted quantile sketch*) algoritmą. Be šių papildomų funkcijų, *XGBoost* taip pat palaiko ir paralelų skaičiavimą. Tokiu būdu išnaudojama daugelio branduolių architektūrą ir pagreitinamas modelio veikimas. (Simplilearn, 2025)

K-artimiausių kaimynų

K-artimiausių kaimynų (*angl. k-Nearest Neighbours*), toliau – k-NN, tai neparametrizuojamas modelis, kuris klasifikuoja duomenis pagal artimiausią kaimyninį duomenų tašką požymių erdvėje. Šis algoritmas dažniausiai naudojamas klasifikavimo užduotims atlikti. Tai vienas paprasčiausių algoritmų, kuris priskiria duomenis klasei pagal tai, kurios klasės duomenų požymių erdvėje yra daugiausia.

Parametru k apibrėžiamas kaimynų skaičius, tad jei $k = 3$, kaip 4 pav. apibrėžta ištisa linija, žaliu apskritimu pažymėtas duomuo bus priskiriamas raudonų trikampių klasei, nes į nurodytą kaimynų skaičių jų papuola daugiausia, tuo tarpu, jei $k = 5$, kaip apibrėžta punktyrine linija, nustatoma klasė bus mėlynų kvadratų.



4 pav. k-NN klasifikavimo pavyzdys

Atstumai iki kaimyninių objektų gali būti nustatomi naudojant skirtingus atstumo nustatymo metodus. Kuomet x, y – duomenų objektai, o i – požymio indeksas, tai euklidinis atstumas nustatomas naudojantis 7 formule, Manheteno atstumas, 8 formule, Minkovskio – 9. Tai populiariausi atstumo nustatymo metodai. (IBM, be datos)

$$Euklidinis\ atstumas = d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}; \tag{7}$$

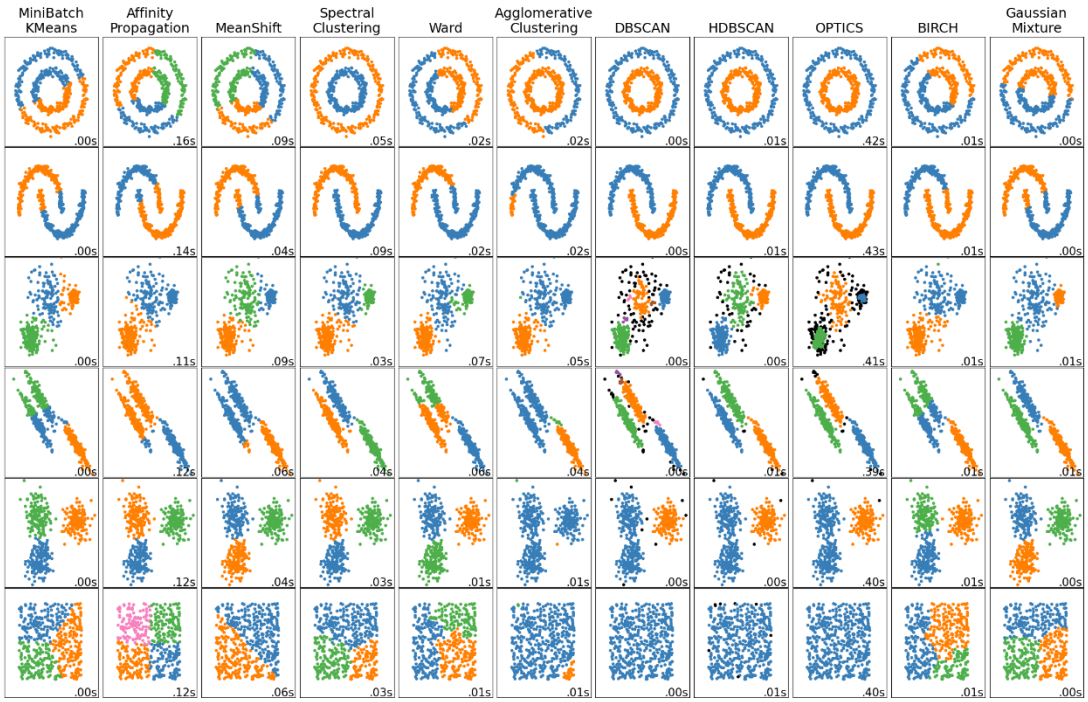
$$Manheteno\ atstumas = d(x,y) = \sqrt{\sum_{i=1}^{m}|x_i - y_i|}; \tag{8}$$

$$Minkowskio\ atstumas = d(x,y) = \left(\sum_{i=1}^{n}|x_i - y_i|\right)^{\frac{1}{p}}; \tag{9}$$

Tokie prižiūrimojo mokymo modeliai dažnai sutinkami AML srityje. Deja jiems apmokyti reikia daug sužymėtų duomenų, kurie nėra lengvai prieinami, kiekvienai institucijai kainuoja daug resursų juos susikurti, o tikslumas yra pakankamai mažas, nes anot (Kulkarni, 2023) aptinkama tik apie 1% visų nelegalių transakcijų. Sužymėjus tik tokį kiekį transakcijų, klasifikatoriai mokomi atpažinti tik 1% nelegalių transakcijų, traktuojant 99% procentus kaip legalias.

1.8. Klasterizavimo metodai

Klasterizavimas – tai mašininio mokymosi metodas, naudojamas panašiems duomenims grupuoti į grupes, kitaip vadinamas – klasteriais. Klasterizavimo tikslas – rasti duomenims būdingą struktūrą pagal jų požymius ir taip suskirstyti juos į poaibius pagal kokius nors parametrus. Klasterizavimo algoritmai priskiriami neprižiūrimajam mokymuisi, kadangi jie nesiremia iš anksto sugrupuotais duomenimis kad apsimokytų. Dėl šios priežasties, klasterizavimo algoritmai yra efektyvūs atrandant naujus ryšius tarp duomenų, kurie iki tol nebuvo žinomi. (Hastie, Tibshirani, & Friedman, 2009) Kaip atvaizduota 5 pav., tie patys duomenys naudojant skirtingus klasterizavimo algoritmus, skirstomi į klases labai skirtingai.



5 pav. Duomenų rinkinio klasterizavimas naudojant skirtingus algoritmus (Pedregosa, et al., 2011)

Klasterizavimo algoritmai pagal duomenų objektų priskyrimą klasteriams skirstomi į išskirtinius (*angl. exclusive*), persidengiančiuosius (*angl. overlapping*), hierarchinius (*angl. hierarchical*) ir tikimybinius (*angl. probabilistic*).

Išskyrus duomenis į klasterius nesunkiai galima toliau dirbant su duomenimis atskleisti pagrindines sąsajas tarp jų, išskirti nukrypimus, triukšmą, anomalijas. Būtent anomalijų aptikimas duomenyse yra ypatingai naudingas norint identifikuoti įtartinas transakcijas. Atlikus klasterizavimą ir nustačius minimalų klasterių dydį, visi klasteriai mažesni nei nustatyta riba, arba išskirtys (taškai, nepapuolantys į klasterius, *angl.* outliers), traktuojami kaip anomalijos ir pripažįstami nelegaliomis transakcijomis.

Tokio tipo modelius įmanoma panaudoti neturint pradinių apmokymų duomenų. Žinant, kad pasiskirstymas tarp legalių ir nelegalių transakcijų nėra vienodas (nelegalių transakcijų pasaulyje yra žymiai mažiau), atrinkus tinkamus požymius, panašios transakcijos grupuojamos kartu ir lieka pavienės išskirtinės – dažniausiai sugrupuotos transakcijos yra legalios, o išskirtos – ne.

Žemiau pateikti populiarūs klasterizavimo algoritmai naudojami šiame ir kituose panašaus tipo tyrimuose.

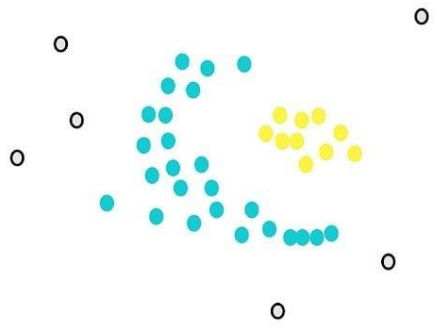
k-vidurkiai

k-vidurkiai (*angl. k-means*) – neprižiūrimojo mokymosi klasterizavimo algoritmas, paremtas vektorių kvantizavimu, kuris skirsto duomenų objektus į grupes ar klasterius, siekiant minimizuoti variabilumą tarp klasterių. Naudojant šį algoritmą, objektas gali priklausyti tik vienam klasteriui. Parinkus klasterių skaičių k, atsitiktiniu būdu parenkamas toks pats kiekis centroidų pagal atstumą (dažniausiai naudojamas euklidinis atstumas, bet gali būti taikomi ir kiti metodai atstumo skaičiavimui). Centroidai yra atnaujinami kaip kiekvieno klasterio taškų vidurkiai ir tai tęsiasi iki kol centroidai stabilizuojasi arba praeina nustatytas skaičius iteracijų. Taškai, nepapuolantys į klasterius, traktuojami kaip anomalijos arba išskirtys. Nepaisant to, kad šiam algoritmui privalumo nurodyti klasterių skaičių ir jam veikiant daroma prielaida, kad klasteriai yra sferiniai, jis yra dažnai naudojamas AML srities tyrimuose.

Tankiu pagrįstas erdvinis programų su triukšmu klasterizavimas (DBSCAN)

Tankiu pagrįstas erdvinis programų su triukšmu klasterizavimas - *(angl. Density-Based Spatial Clustering of Applications with Noise)* (toliau – DBSCAN) – tai duomenų tankiu grindžiamas klasterizavimo algoritmas, kuris klasterių skaičių nustato pagal duomenų išsibarstymo požymių erdvėje tankį. Klasteriai gali susiformuoti bet kokios formos, o išskirtys, nepapuolančios į jokį klasterį traktuojamos kaip anomalijos. (Yang, Lian, Li, Chen, & Li, 2014). DBSCAN algoritmas yra tinkamas, kai turimi duomenys požymių erdvėje yra išsidėstę grupėse panašiu tankiu.

DBSCAN algoritmo veikimas apibrėžiamas parametru ε, nurodančiu atstumą iki duomenų objektų, kurių esant daugiau kaip m, taškas skaitomas pagrindiniu (*angl. core point*). Pagrindiniai taškai sudaro duomenų objektų imtį skirstomą į klasterius, tuo tarpu nepatenkantys taškai į šią imtį – priskiriami anomalijoms. Nustatytoje duomenų imtyje taškai, kurie yra išsidėstę tankesniuose regionuose išskiriami į skirtingus klasterius (6 pav.)



6 pav. Duomenų rinkinio klasterizavimas naudojant DBSCAN
(Yenigün, 2024)

Hierarchinis tankiu pagrįstas erdvinis programų su triukšmu klasterizavimas (HDBSCAN)

Hierarchinis tankiu pagrįstas erdvinis programų su triukšmu klasterizavimas - *(angl. Hierarchical Density-Based Spatial Clustering of Applications with Noise)* (toliau – HDBSCAN) – tai duomenų klasterizavimo algoritmas, paremtas DBSCAN veikimu. HDBSCAN yra pranašesnis už DBSCAN,

nes taiko kintančią ε vertę ir geba klasterizuoti kintančio tankio duomenis. Kaip atvaizduota x pav. pastebima, kad HDBSCAN algoritmas palieka mažiau išskirčių, kai jos gali būti racionaliai įtraukiamos į klasterį.

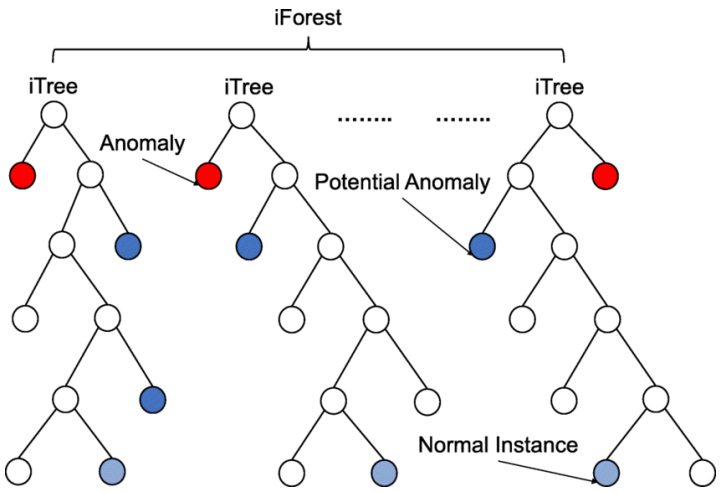
***SOM (Self-Organizing Maps)* (Susiorganizuojantys žemėlapiai)** – Tai neuroniniai tinklai, projektuojantys daugiau nei 2 dimensijų duomenis į mažesnės apimties (mažesnių dimensijų) tinklelį (dažniausiai 2D). Anomalijos nustatomos kaip taškai kurie yra toli nuo bet kurio kito taško tinklelyje, arba kurių kaimynystėje esančių taškų tankis yra mažas. Tai ypač naudingas metodas vizualizuojant didelių dimensijų duomenis.

1.9. Anomalijų aptikimas

Kai kurie algoritmai specifikuojasi anomalijų identifikavime arba yra našūs šiai užduočiai, bei sugeba unikaliu būdu išskirti anomalijas. Kadangi nelegalios transakcijos yra retos visoje duomenų imtyje, jas įmanoma identifikuoti anomalijų išskyrimo būdu.

Atskyrimo miškas

Atskyrimo miškas (*angl. Isolation forest*) – šis algoritmas skirtas anomalijų identifikavimui. Jis išskiria anomalijas, neklasterizuodamas duomenų. Atsitiktinai suskirstytiems duomenims naudojant sprendimų medžius, anomalijas lengviau išskirti, nes duomenis reikia suskirstyti į mažiau dalių, nei skirstant į klasterius. Anomalijos išskiriamos pagal vidutinį kelio ilgį iki taško, kurio reikia tašką priskirti anomalijai. Struktūrinė šio algoritmo schema pavaizduota 7 pav. Algoritmas veiksmingas didelių dimensijų duomenimis ir nėra labai jautrus parametrų derinimui. (Regaya, Fadli, & Amira, 2021)



7 pav. Atskyrimo miško algoritmo veikimo reprezentacija (Regaya, Fadli, & Amira, 2021)

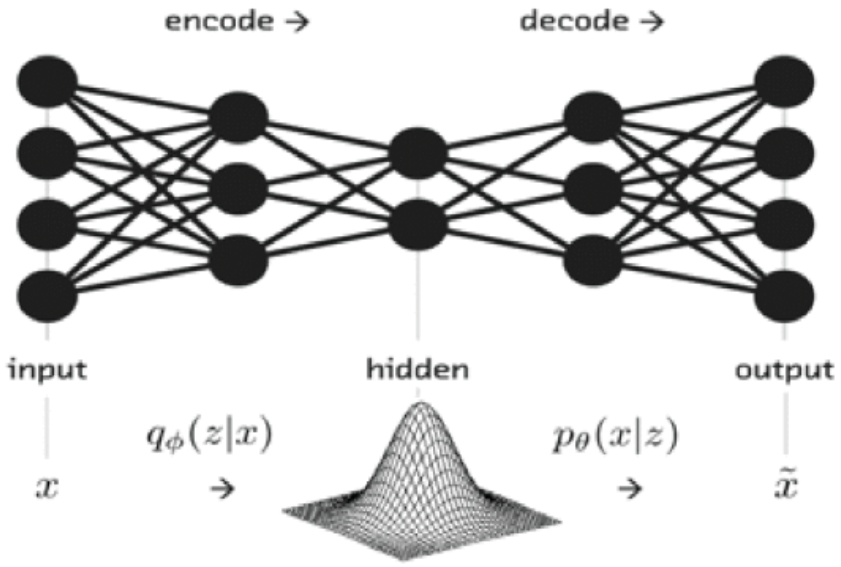
1.10. Kiti algoritmai

Variaciniai autoenkoderiai

Variaciniai autoenkoderiai (*angl. Variational Autoencoders*) – tai neuroniniais tinklais paremtas algoritmas, susidedantis iš dviejų dalių – enkoderio ir dekoderio (8 pav.). Autoenkoderiai veikia principu, kad enkoderiu iš duomenų yra išskiriamos tik pačios svarbiausios savybės į paslėptąjį (*angl. latent*) sluoksnį iš kurių dekoderis vėliau bando jas atstatyti į pilną formą. Kai duomenys panašūs,

atstatymas vyksta gerai ir palyginus juos su pradiniais duomenimis, atstatymo paklaida (*angl. reconstruction error*) yra maža. Tuo tarpu anomalijos, kadangi nėra panašios į visus duomenis, turi didelę atstatymo paklaidą. Pagal atstatymo paklaidos iš anksto nustatytą maksimalią ribą, taškai traktuojami kaip normalūs arba išskirtys. (Chen, Soliman, Nazir, & Shorfuzzaman, 2021)

Autoenkoderis, kaip minima šaltinyje (Jensen & Iosifidis, 2023) gali būti naudojamas ne tik klasifikavimui. Šis algoritmas gali būti pritaikomas taip pat ir dimensijų redukcijai ar sintetinių duomenų generavimui siekiant subalansuoti duomenų klases.



8 pav. Variacinio autoenkoderio struktūros reprezentacija
(Chen, Soliman, Nazir, & Shorfuzzaman, 2021)

1.11. Kokybiniai įverčiai

Algoritmų įvertinimui reikalinga pasirinkti kokybinius įverčius, kurie atspindėtų jų efektyvumą ir našumą įvairiose srityse. Vertinami klasifikavimo įverčiai, atspindintys modelio tinkamumą duomenims klasifikuoti, taip pat klasterizavimo kokybės įverčiai, nusakantys klasterio kokybę. Šie parametrai yra būtini algoritmų našumo vertinimui ir tiksliam jų parametrų adaptavimui ir optimalių rezultatų pasiekimui.

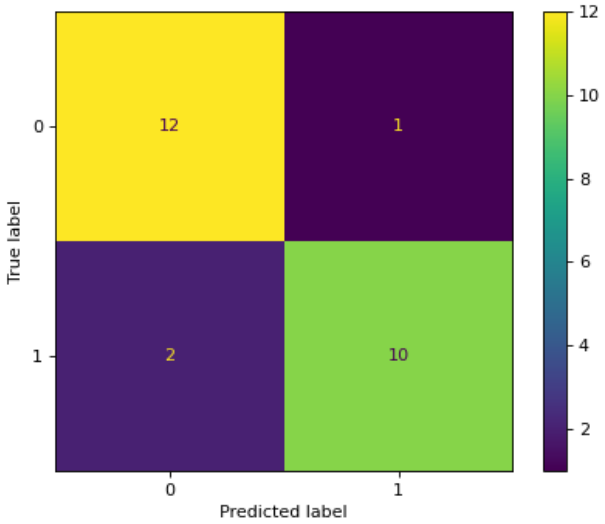
Tokie įverčiai gali būti gaunami iš neatitikimų matricos (*angl. confusion matrix*) (9 pav.), kurią sudaro naudoto modelio prognozavimo tikslumą apibūdinantys rezultatai - tikro teigiamo (TP[15]), tikro neigiamo (TN[16]), klaidingai teigiamo (FP[17]) ir klaidingai neigiamo (FN[18]) prognozių skaičiai. Šios keturios reikšmės naudojamos kitiems kokybiniams įverčiams apskaičiuoti.

Žemiau pateikiami dažniausiai naudojami ir populiariausi kokybiniai rodikliai modelio našumo įvertinimui ir formulės (1 - 4) jiems apskaičiuoti. (Tigerschiold, 2022)

[15] TP – True Positive (angl.)
[16] TN – True Negative (angl.)
[17] FP – False Positive (angl.)
[18] FN – False Negative (angl.)

Atsiminimas (*recall*): nurodo kiek iš visų reikšmių, kurios turėjo būti *tiesa* iš tikrųjų buvo atspėtos kaip *tiesa*.

$$Recall = \frac{TP}{TP + FN}; \tag{10}$$



9 pav. Neatitikimų matrica
(Pedregosa, et al., 2011)

Tikslumas (*accuracy*): nurodo kiek reikšmių buvo atspėtos teisingai iš visų prognozuotų.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}; \tag{11}$$

Tikslumas (*precision*): nurodo kiek iš prognozuotų reikšmių *tiesa* iš tikrųjų yra *tiesa*.

$$Precision = \frac{TP}{TP + FP}; \tag{12}$$

F1-įvertis (*F1-Score*): nurodo modelio kompromiso tarp tikslumo ir atsiminimo efektyvumą.

$$F1 = 2 * \frac{Precision \cdot Recall}{Precision + Recall}; \tag{13}$$

Kadangi AML srityje duomenys dažniausiai nėra subalansuoti, juos klasifikuojančių algoritmų našumui vertinti aktualu naudoti kokybinius įverčius, kurie yra pritaikyti nesubalansuotiems duomenims. Keletas iš tokių požymių – subalansuotas tikslumas, plotas po ROC ir PR kreivėmis. (Jensen & Iosifidis, 2023)

Subalansuotas tikslumas (*balanced accuracy*): nurodo virdukį tarp teisingų teigiamų ir teisingų neigiamų spėjimų, atspindi realesnę situaciją esant nesubalansuotiems duomenims.

$$Balanced\ accuracy = \frac{\left(\frac{TP}{TP + FP} + \frac{TN}{TN + FN}\right)}{2} \tag{14}$$

AUC-ROC (*Area Under the Receiver Operating Characteristic Curve*): kompromisas tarp atsiminimo (TPR, recall) ir klaidingai teigiamų rezultatų (FPR) pagal visas klasifikavimo ribas.

$$AUC - ROC = \int_0^1 Recall\left(\frac{FP}{FP+TN}\right) d\left(\frac{FP}{FP+TN}\right); \quad (15)$$

AUC-PR (*Area Under the Precision-Recall Curve*): kompromisas tarp tikslumo (precision) ir atsiminimo (recall) pagal ribas, daugiausia dėmesio skiriant teigiamos klasės rezultatams.

$$AUC - PR = \int_0^1 Precision(Recall) dRecall; \quad (16)$$

MCC - Matthew koreliacijos koeficientas (*Matthews Correlation Coefficient*): Matuoja koreliaciją tarp stebimų ir prognozuojamų klasifikacijų, atsparus nesubalansuotiems duomenų rinkiniams.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}; \quad (17)$$

Visi kokybiniai rodikliai vertinami skalėje nuo 0 iki 1. Siekiamybė, kad modelis turėtų įvertinimą visose srityse kuo arčiau 1. Kaip matome iš įverčių, tikslumas ir atsiminimas yra priešingi vienas kitam rodikliai ir vienam didėjant, kitas mažėja. Dažnu atveju svarstoma kas yra svarbiau – prognozuoti daugiau teisingų verčių, bet tarp jų turėti daug neteisingai identifikuotų, ar atspėti mažiau verčių, bet tiksliau (mažiau neteisingų). Apibendrinantis rodiklis, nurodantis kompromisą tarp rodiklių yra F1 įvertis. Kuo mažiau yra bendrai atpažintų klaidingų duomenų, tuo šis įvertis didesnis. Pagal F1 įvertį galima vertinti modelio tinkamumą duomenims atpažinti. Šis įvertis dažnai sutinkamas mašininio mokymosi srities tyrimuose. (Bi, Trinh, & Fan, 2024)

1.12. AML srities tyrimų rezultatai ir gairės tolimesniems tyrimams

AML srities tyrimų apžvalga (Chen, et al., 2017) nagrinėja įvairius AML srities tyrimus, naudojančius mašininį mokymą. Šiuose tyrimuose atrenkami nesudėtingi statiniai požymiai, tokie kaip mokėjimų dažnis ir vertės įvairiais intervalais. Aptariami ir (Le-Khac, Markos, & Kechadi) tyrimai, kuriuose aptariamas pusiau prižiūrimojo mokymo požiūris ir išbandomos algoritmų kombinacijos, tokios kaip klasterizavimo ir daugiasluoksnio perceptrono (MLP – *angl. Multi-Layer Perceptron*), k-vidurkių ir dirbtinio neuroninio tinklo. Nors rezultatai nėra itin geri, teigiama, kad pasiektas 0.5% atpažinimo rezultatas, tai pateikiama kaip geras rezultatas su perspektyva tolimesniam tyrinėjimui.

Kitame tyrime, (Fan, et al., 2025) pateikiamas CRP-AML modelis. Naudojant IT-AML duomenų bazę, pasiekiamas iki 83 % F1 įvertis.

Apžvelgus populiarius AML srities tyrimus, pastebimos įžvalgos, kad siūloma įtraukti alternatyvius klasterizavimo algoritmus, tokius kaip DBSCAN ir K-means. (Bakry, Alsharkawy, Farag, & Raslan, 2024) Apibendrinama, kad daugelyje tyrimų naudojami prižiūrimojo mokymosi metodai su mažomis duombazėmis, tuo tarpu neprižiūrimasis mokymas taikomas tik didelėms duomenų bazėms. (Jensen & Iosifidis, 2023)

2. Metodologija

Šioje dalyje aptariama metodika tyrimams atlikti, teorija, hipotezės atliekamiems tyrimams. Aptariama naudojama programinė įranga, duomenų bazės, kita programinė įranga, papildiniai ar algoritmai naudojami tyrimui atlikti.

Šio tyrimo metu, naudojant sintetinių transakcijų duomenų bazę, siekiama išsiaiškinti kokie maksimalūs kokybiniai įverčiai gali būti pasiekiami naudojant plačiai naudojamus klasifikavimo algoritmus. Naudojama riboto laiko duomenų bazė ir nesudėtingi požymiai, nešantys mažai informacijos apie vartotojo elgsenos pokytį, siekiant įvertinti galimybę atpažinti nelegalias transakcijas nekaupiant ilgos transakcijų istorijos.

Taikant įvairias duomenų apdorojimo metodikas, siekiama nustatyti kokie įrankiai leidžia pasiekti geresnę greitaveiką ir aukštesnius kokybinius rodiklius, Taikant spėjimų ansamblio metodą su balsavimo svoriais tarp algoritmų, siekiama išsiaiškinti kokie algoritmai sąveikauja tarpusavyje geriausiai ir pagerina kokybinius rodiklius, nei veikdami atskirai.

Dėl realių duomenų specifikos, kurie neturi pinigų plovimo žymos, siekiama įvertinti ir alternatyvų požiūrį į įtartinų transakcijų aptikimą. Daroma prielaida, kad kiekvienas vartotojas ir jų grupė turi sau būdingus mokėjimo įpročius – naudoja tą pačią valiutą, tuos pačius mokėjimo būdus, turi tipinį laiką, kuomet atlieka transakcijas. Išskiriant požymius, kurie atspindi nuokrypius nuo vartotojo būdingų transakcijų, spėjama, kad stipriausiai išsiskiria tos transakcijos, kurios yra nelegalios.

Šiam alternatyviam požiūriui įvertinti naudojami klasterizavimo metodai sugrupuojantys panašias transakcijas į tankius klasterius, siekiant patikrinti ar taškai paliekami kaip išskirtys arba maži klasteriai turi didesnį nelegalių transakcijų tankį, nei tankūs klasteriai.

2.1. Naudojama įranga

Šiam tyrimui atlikti naudojamas kompiuterinė įranga, kurios parametrai pateikti žemiau. Programiniam kodui įgyvendinti naudojama *python* programavimo kalba ir *PyCharm* programavimo aplinka. Duomenims apdoroti naudojamos *pandas, numpy* bibliotekos, leidžiančios apdoroti duomenis dvimačiu formatu, mašininio mokymo algoritmai naudojami pasitelkiant *scikit-learn* paketą (Pedregosa, et al., 2011), o duomenų atvaizdavimui naudojama *matplotlib* biblioteka. Duomenų patogiam interpretavimui kiekvieno sėkmingo programos įvykdymo metu, generuojamas *.html* formato failas su visa reikalinga tyrimo informacija (paveiksliukai, lentelės), kurį galima lengvai dalintis ar atidaryti bet kokioje interneto naršyklėje.

Aktualūs kompiuterinės įrangos parametrai:

- Procesorius - Intel Core i7-10875H @2.3 GHz
- RAM atmintis – 16GB SODIMM
- Kietasis diskas – Western Digital WDC PC SN730 1TB
- Operacinė sistema – Windows 11 Pro 64-bit

2.2. Duomenų bazė

Šiam tyrimui atlikti pasirinkta IBM Pinigų Plovimo Prevencijos (AML) duomenų bazė, kuri yra prieinama per Kaggle platformą, ir yra pagrindinis duomenų šaltinis kuriant neprižiūrimo mašininio mokymosi modelius, skirtus įtartiniems finansiniams sandoriams aptikti. Sukurta naudojant AMLworld daugelio agentų simuliatorių, ši duomenų bazė modeliuoja virtualią ekonomiką, kurioje agentai, atspindintys individus ir įmones, atlieka transakcijas, tokias kaip banko pervedimai ir pirkimai (Altman, et al., 2024).

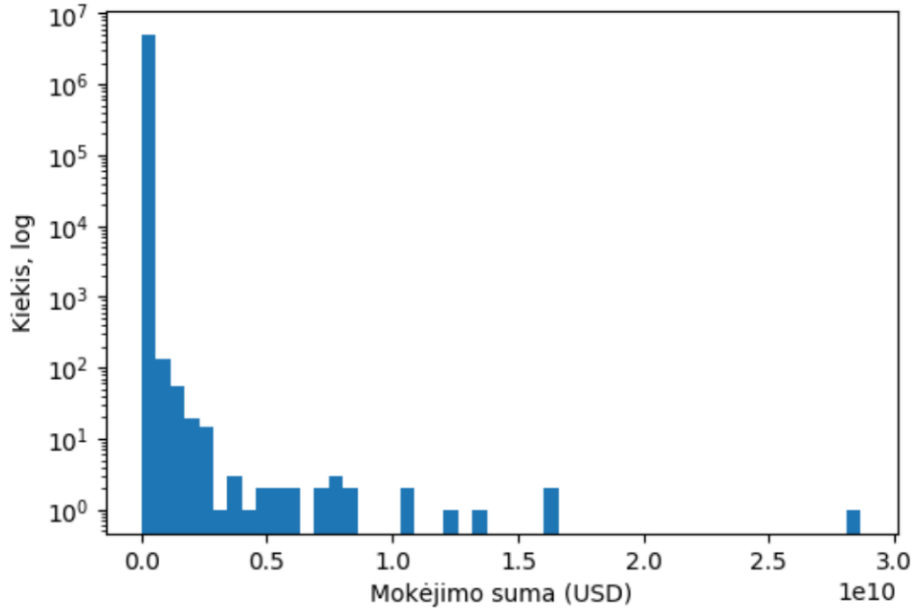
Žvelgiant į duomenų bazės struktūrą, jos generavimas remiasi AMLworld simuliatoriaus gebėjimu imituoti tiek teisėtas, tiek nelegalias transakcijas, generuojant *.csv* formato failus su požymiais, tokiais kaip transakcijos identifikacijos numeris (toliau – ID), transakcijos suma, jos tipas, laiko žyma ir etiketė, žyminti "ar nelegali" transakcija (10 pav.). Šis duomenų bazės dizainas atspindi realaus pasaulio AML srities iššūkius – nedaug duomenų apie transakciją, nesubalansuotą transakcijų pasiskirstymą tarp legalių ir nelegalių transakcijų, kai nelegalūs sandoriai yra itin reti, atkartojant finansinių nusikaltimų mažą paplitimą tikrose bankų sistemose.

Timestamp	From Bank	Account	To Bank	Account	Amount Received	Receiving Currency	Amount Paid	Payment Currency	Payment Format	Is Laundering
22/09/01 00:20	10	8000EBD30	10	8000EBD30	3697.34	US Dollar	3697.34	US Dollar	Reinvestment	0
22/09/01 00:20	3208	8000F4580	1	8000F5340	0.01	US Dollar	0.01	US Dollar	Cheque	0
22/09/01 00:00	3209	8000F4670	3209	8000F4670	14675.57	US Dollar	14675.57	US Dollar	Reinvestment	0
22/09/01 00:02	12	8000F5030	12	8000F5030	2806.97	US Dollar	2806.97	US Dollar	Reinvestment	0
22/09/01 00:06	10	8000F5200	10	8000F5200	36682.97	US Dollar	36682.97	US Dollar	Reinvestment	0
22/09/01 00:03	1	8000F5AD0	1	8000F5AD0	6162.44	US Dollar	6162.44	US Dollar	Reinvestment	0
22/09/01 00:08	1	8000EBAC0	1	8000EBAC0	14.26	US Dollar	14.26	US Dollar	Reinvestment	0
22/09/01 00:16	1	8000EC1E0	1	8000EC1E0	11.86	US Dollar	11.86	US Dollar	Reinvestment	0
22/09/01 00:26	12	8000EC280	2439	8017BF800	7.66	US Dollar	7.66	US Dollar	Credit Card	0
22/09/01 00:21	1	8000EDEC0	211050	80AEF5310	383.71	US Dollar	383.71	US Dollar	Credit Card	0
22/09/01 00:04	1	8000F4510	11813	8011305D0	9.82	US Dollar	9.82	US Dollar	Credit Card	0
22/09/01 00:04	1	8000F47F0	1	8000F47F0	9.38	US Dollar	9.38	US Dollar	Reinvestment	0
22/09/01 00:08	1	8000F4FE0	245335	812ED62E0	4.01	US Dollar	4.01	US Dollar	Credit Card	0
22/09/01 00:17	10	80012FD90	36056	812ED6380	106.7	US Dollar	106.7	US Dollar	Credit Card	0
22/09/01 00:11	12	80012FE00	13037	805B34210	0.54	US Dollar	0.54	US Dollar	Credit Card	0
22/09/01 00:09	1	80012FE50	1	80012FE50	3944232.29	US Dollar	3944232.29	US Dollar	Reinvestment	0
22/09/01 00:11	10	80012FEA0	10	80012FEA0	10020.68	US Dollar	10020.68	US Dollar	Reinvestment	0
22/09/01 00:00	1420	8005DFEB0	1420	8005DFEB0	897.37	US Dollar	897.37	US Dollar	Reinvestment	0
22/09/01 00:28	1665	8005E18F0	1665	8005E18F0	157.57	US Dollar	157.57	US Dollar	Reinvestment	0
22/09/01 00:22	1665	8005E24C0	1665	8005E24C0	52.75	US Dollar	52.75	US Dollar	Reinvestment	0

10 pav. IBM sintetinės duomenų bazės iškarpa

Duomenų bazė yra suskirstyta į 6 variacijas – trijų dydžių (maža, vidutinė, didelė) pagal transakcijų kiekį ir pagal nelegalių transakcijų tankį į didelio ir mažo tankio. Duomenų bazėse duomenų kiekis svyruoja nuo 5 iki 180 mln. transakcijų.

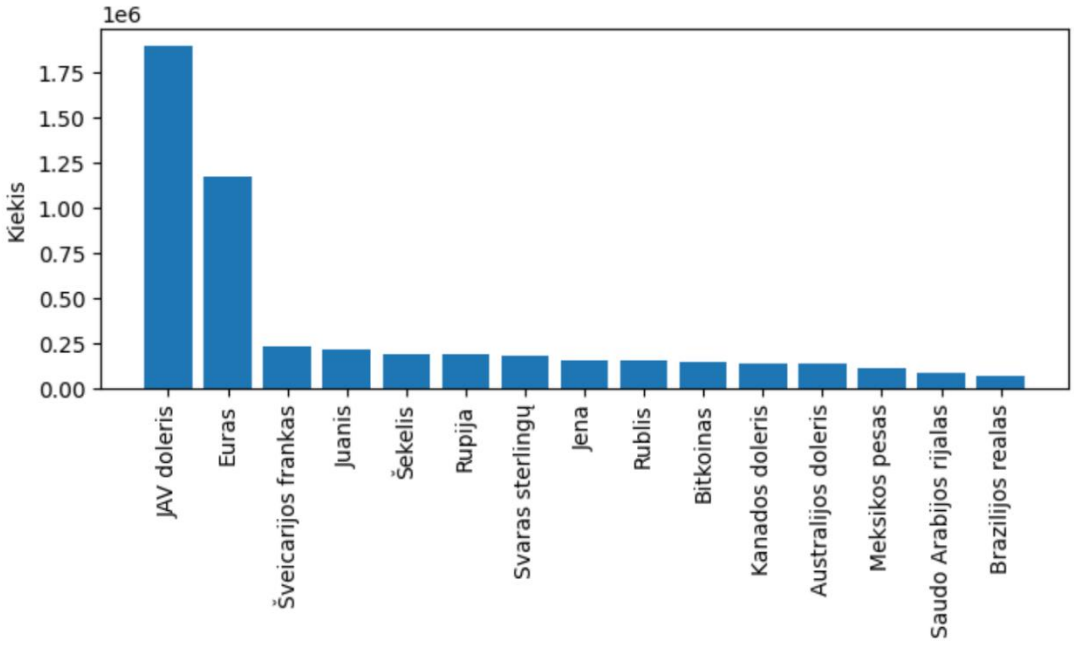
Kadangi transakcijų sumų ir nelegalių transakcijų kiekio pasiskirstymas tarp mažos, vidutinės ir didelės duomenų bazės yra vienodas, bei tyrimu siekiama įvertinti maksimalius modelių rezultatus naudojant minimaliai duomenų, naudojama yra maža, didelio nelegalių transakcijų tankio duomenų bazė, kuri turi virš 5 mln. įrašų ir nelegalių transakcijų kiekis joje siekia apie 0.1 %. Transakcijų pasiskirstymas logaritminėje skalėje pagal siunčiamų lėšų kiekį atvaizduojamas 11 pav.



11 pav. Transakcijų pasiskirstymas pagal siunčiamų lėšų kiekį

Valiutų konversija

Duomenų bazėje pateiktos transakcijos yra įvairių valiutų. (12 pav.) Kad skaičius būtų galima palyginti ir požymiai nebūtų iškreipti, siunčiamų ir gaunamų lėšų kiekiai konvertuojami į JAV dolerius.



12 pav. Transakcijų mokėjimo sumų pasiskirstymas pagal valiutą

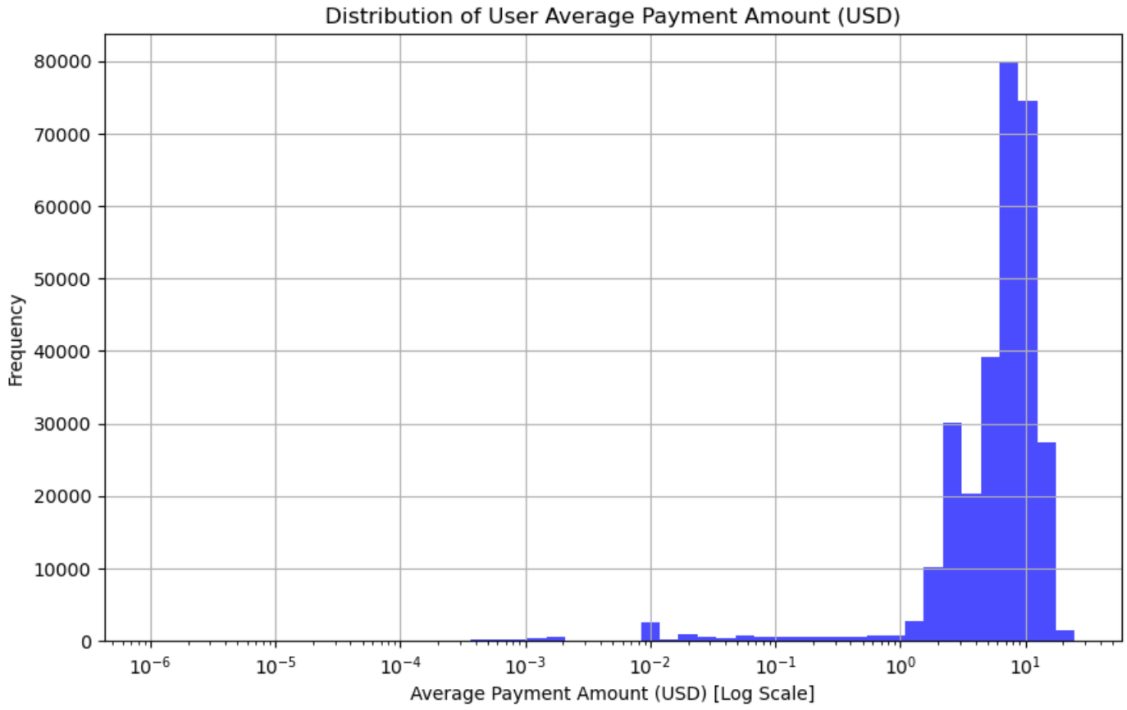
Kadangi visų duomenų bazių variacijų transakcijos atliktos 2022 m. rugsėjo mėn. – 2023 m. sausio mėn. datomis, keitimui naudojamas 2022 m. rugsėjo mėn. 1 d. valiutų kursas, kuris tinka tiek mažiausiai duomenų bazei (iki 2022 m. rugsėjo mėn. 18 d.), tiek didžiausiai (iki 2023 m. sausio mėn.). Žemiau lentelėje pateikiami valiutų kursai (XE, 2022), (yahoo!finance, 2022).

1 lentelė. Valiutų kursai 2022 m. rugsėjo mėn. 1 d.

Valiuta	**Kursas į 1 JAV dolerį (= 1 USD)**
Bitcoin (yahoo!finance, 2022)	0,0005
Euras	1,00539
Australijos doleris	1,47606
Kinijos juanis	6,90672
Indijos rupija	79,66131
Meksikos pesas	20,23564
Japonijos jena	139,94473
Didžiosios Britanijos svaras	0,86696
Rusijos rublis	60,19367
Kanados doleris	1,31641
Šveicarijos frankas	0,98387
Brazilijos realas	5,23102
Saudi Arabijos rialas	3,75
Izraelio šekelis	3,39583
Australijos doleris	1,47606

2.3. Vartotojų grupių profiliai

Duomenų bazės statistika rodo, kad parametrai tokie kaip mokėjimo suma (13pav.), transakcijų dažnis tarp vartotojų stipriai skiriasi. Atsižvelgiant į tai, kad skirtingi vartotojai, gali turėti skirtingus įpročius, vartotojus pravartu skirstyti į grupes ir rodiklius skaičiuoti kiekvienam vartotojui. Deja, dėl duomenų bazės ypatumų, jai esant trumpo periodo, nemaža dalis vartotojų transakcijas daro itin retai ir jų į duomenų imtį papuola vos 1 ar 2 transakcijos. Dėl šios priežasties nuspręsta skaidyti vartotojus į grupes ir skaičiuoti grupinius parametrus vietoje kiekvieno vartotojo.



13 pav. Vartotojų pasiskirstymas pagal mokėjimų sumos vidurkį logaritminėje skalėje

Vartotojų skirstymui į grupes skaičiuojamas vartotojo pervedamų transakcijų per dieną mediana, Pagal tai vartotojai skirstomi į 5 grupes. Grupių ribos yra statinės ir nekintamos – apibrėžtos atitinkamai iki 1, 2, 5, 10 ir begalybės transakcijų per dieną medianos. Tokiu būdu vartotojams turint kintamą parametrą o riboms liekant statinėms, atsižvelgiama į vartotojo dinamiką. Jei per skaičiuojamą periodą vartotojas atliko transakcijas dažniau ar rečiau, jis patenka į kitą grupę ir yra lyginamas su kitais vartotojais. Transakcijų imčiai iš duomenų bazės esant 5 mln., vartotojų pasiskirstymas tarp grupių ir grupių parametrai atvaizduoti 14 pav.

	Account	Frequency median	Frequency mean	Amount median	Amount mean	Amount std	Total transactions
0	100428660	18661.0	16574.625	1069.830000	325020.938360	8.703549e+06	132597
1	1004286A8	11332.0	10110.375	854.583793	271747.386088	6.383430e+06	80883
2	1004286F0	2061.0	1838.875	913.062930	194164.324532	2.835806e+06	14711
3	100428738	1524.0	1351.250	786.105915	146203.580001	3.301891e+06	10810
4	100428780	1928.0	1705.125	914.775567	298075.753865	5.948548e+06	13641
5	1004287C8	1549.0	1386.875	1321.214174	128530.075149	1.606570e+06	11095
6	100428810	1821.0	1618.375	1017.140352	355407.030993	6.653933e+06	12947
7	100428858	1205.0	1079.625	1214.925441	199507.050573	3.195354e+06	8637
8	1004288A0	1367.0	1212.000	1293.233337	183882.728406	2.135907e+06	9696
9	1004288E8	1022.0	929.250	1141.593743	321962.128708	5.662906e+06	7434

	Group	Amount Paid USD Median	Amount Paid USD Std	Amount Paid USD Mean	Account Count	Total Transaction Count
0	1	658.325000	5.858560e+07	499252.373342	277054	445254
1	2	754.913722	2.739815e+07	359959.028528	138570	806136
2	3	1022.730000	1.292229e+07	369787.249215	55741	1098049
3	4	921.100000	2.267798e+07	387350.501941	18795	911645
4	5	966.429415	6.786061e+06	273326.886224	4719	801592

14 pav. Vartotojų (viršuje) ir vartotojų grupių (apačioje) parametrai

Vartotojai skaidomi į grupes naudojant kelis skirtingus metodus. Skaidymas į grupes pirmuoju būdu vykdomas naudojant tolygų paskyrų pasiskirstymą tarp grupių vieno iš skaičiuotų parametrų atžvilgiu. Kitu būdu naudojamas *BayesianGaussianMixture* klasterizavimo modelis, sutraukiantis paskyras į klasterius naudojant klasterizavimo modelį. Optimalus klasterių kiekis parenkamas atliekant skirtingas klasterizavimo iteracijas su skirtingu klasteriu kiekiu ir skaičiuojant *Calinski-Harabasz* įvertį, nurodantį klasterizavimo kokybę.

2.4. Požymiai

Iš esamų pirminių transakcijų požymių išskaičiuojami statiniai ir dinaminiai požymiai, kurie kituose tyrimuose davė teigiamų rezultatų. Žemiau lentelėje pateikti pirminiai ir apskaičiuoti požymiai, jų žymėjimas tyrime (lentelėse, grafikuose), bei jų reikšmė.

2 lentelė. Atrinkti transakcijų požymiai tyrimui.

Požymio pavadinimas	Pavadinimo vertimas	Požymio reikšmė
Same bank	Tas pats bankas	Mokėjimas atliekamas tame pačiame banke
Same account	Ta pati paskyra	Mokėjimas atliekamas toje pačioje paskyroje
Same currency	Ta pati valiuta	Mokėtojas ir gavėjas naudoja tą pačią valiutą
Round amount	Apvali suma	Mokėjimo suma yra be centų (suapvalinta)
Amount Received	Gaunama suma	Suma kurią gauna gavėjas
Amount Paid	Mokėjimo suma	Suma kurią moka mokėtojas
Receiving Currency	Gaunama valiuta	Valiuta, kuria gauna mokėjimą gavėjas
Payment Currency	Mokėjimo valiuta	Valiuta, kuria moka mokėtojas
hour_of_day	valanda dienoje	valanda, kurią atlikta transakcija
Daytime	paros metas	Transakcijos atlikimo paros metas (Naktis, rytas, diena, vakaras)

is_non_typical_hour	netipinė valanda	Žyma, rodanti kad transakcija atlikta vartotojui neįprastą valandą
day_of_week	diena savaitėje	savaitės diena, kurią atlikta transakcija
is_weekend	savaitgalis	žyma, rodanti ar transakcija atlikta savaitgalį
Payment Format_	Mokėjimo tipas - ...	žyma, rodanti, kad mokėjimas yra ACH[19]/Bitcoin[20]/... kt. tipo
is_non_typical_day	netipinė diena	Žyma, rodanti kad transakcija atlikta vartotojui neįprastą dieną
Group	Grupė	žyma, nurodanti kuriai grupei vartotojas priklauso
Z Score paid	Mokėjimo sumos Z įvertis	Mokėjimo sumos Z įvertis skaičiuojamas iš grupės mokėjimo vidurkio ir standartinio nuokrypio
Relative deviation paid	Santykinis nuokrypis nuo grupės vidurkio	Sumos santykinis nuokrypis nuo vartotojų grupės mokėjimų sumos vidurkio

Šiame tyrime naudojant duomenų bazę, kurios transakcijų imtis apima 17 dienų laikotarpį. Dėl šios priežasties neskaičiuojami bendruomeniškumo (*angl. community*) požymiai, kurie aktualūs esant ilgesniam laiko diapazonui.

Apskaičiuojami kategoriniai požymiai – vartotojo grupė, transakcijos valanda, diena, valiuta, naudojant *OneHotEncoding* konvertuojami į dvejetainius *true/false* požymius duomenų standartizavimui.

2.5. Požymių apdorojimas

Maksimaliam duomenų optimizavimui prieš apmokant algoritmus atpažinti įtartinas transakcijas, duomenys apdorojami. Pirmiausia naudojant *scikit-learn* bibliotekos funkciją *train_test_split()* duomenų imtis išskaidoma į treniravimo ir testavimo imtis, naudojant atitinkamai 80/20 % proporcijas. Duomenys iš imties parenkami atsitiktinai, bet nustatant *random_state* parametrą fiksuotą, kad duomenys nekistų viso tyrimo metu. Tuomet duomenys standartizuojami paketo *scikit-learn* funkcija *StandardScaler*. Standartizuoti duomenys balansuojami *imbalance-learn* paketo *SMOTE* algoritmu, požymiai redukuojami *scikit-learn* paketo PCA duomenų redukcijos algoritmu.

2.6. Požymių redukcija

Iš visų požymių (16 pav.), tolimesniam naudojimui išrinkti požymiai toliau apdorojami pasirinktu dimensijų redukcijos algoritmu, siekiant pagreitinti klasterizavimo ir klasifikavimo algoritmų veikimus, bei eliminuoti požymius, dubliuojančius tą pačią informaciją.

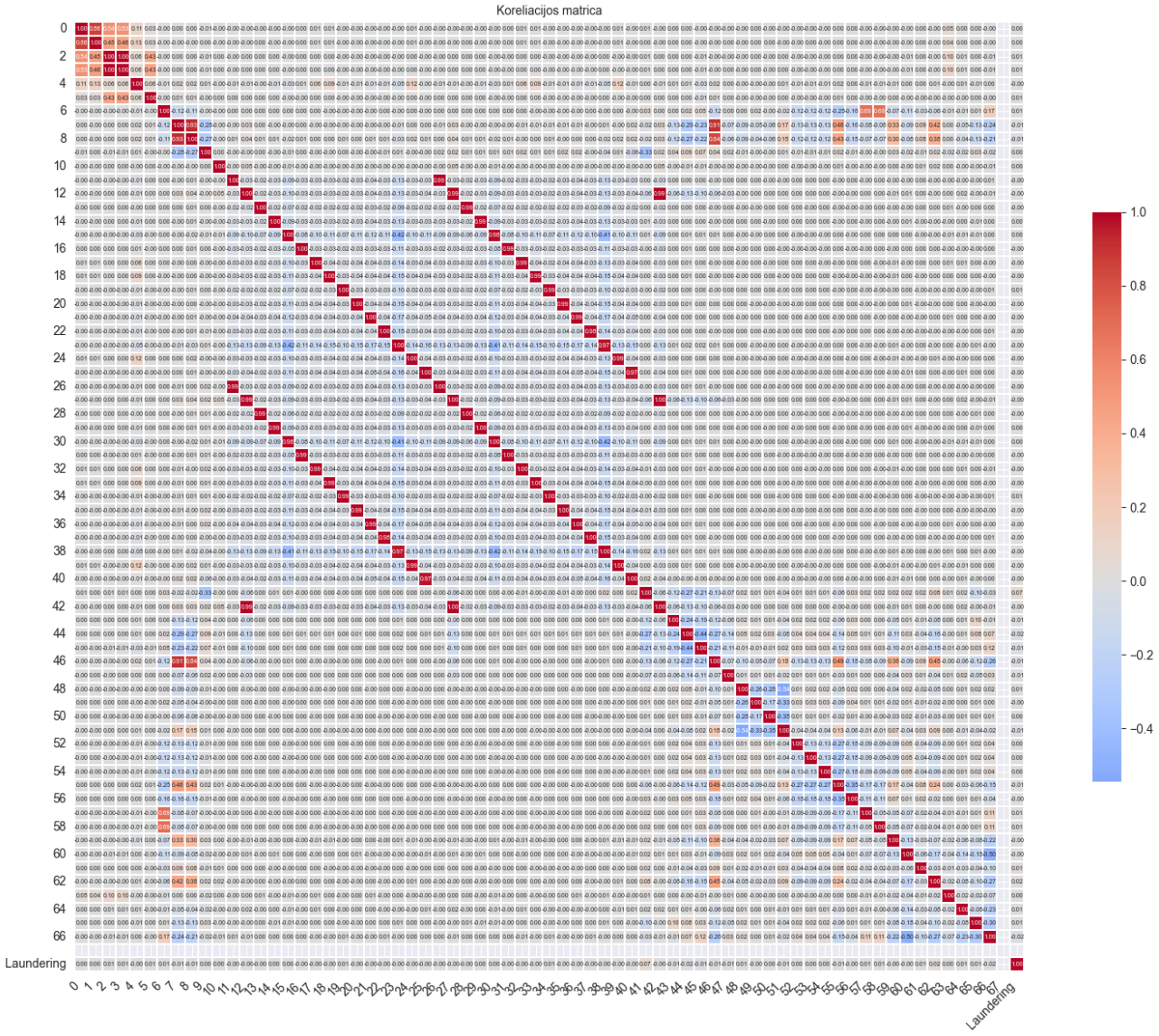
Požymių redukcijai siekiama parinkti tokį, komponenčių skaičių, kad suminis variabilumas nebūtų mažesnis nei 95 %. Tokiu būdu neprarandama svarbi informacija ir pagerinamas mašininio mokymosi algoritmų našumas. 15 pav. atvaizduojamas *PCA* algoritmo taikymas ir suminio variabilumo skaičiavimas *python* programiniu kodu. Požymiams pritaikius *PCA* redukciją, principinių komponenčių koreliacija pateikiama 17 pav.

[19] ACH – (*angl. Automated Clearing House*)

[20] Bitcoin – viena populiariausių kriptovaliutų

```python
reducer = PCA(n_components=pca_n_components, random_state=42)
X_train_reduced = reducer.fit_transform(X_train)
X_test_reduced = reducer.transform(X_test)
print("Explained Variance Ratio:", reducer.explained_variance_ratio_)
print("Cumulative Explained Variance:", np.cumsum(reducer.explained_variance_ratio_))
# Reconstruction error
X_train_reconstructed = reducer.inverse_transform(X_train_reduced)
reconstruction_error = np.mean((X_train - X_train_reconstructed) ** 2)
print(f"Reconstruction Error (MSE): {reconstruction_error:.4f}")
```

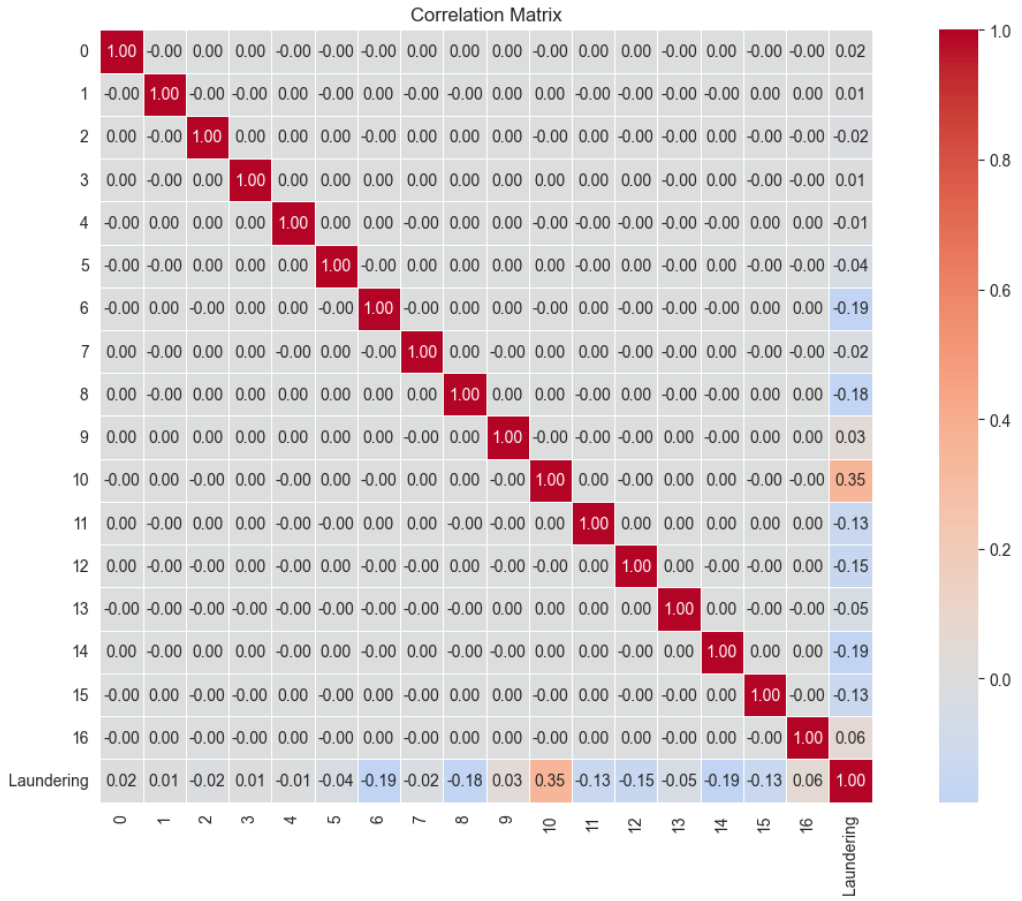
15 pav. PCA algoritmo realizavimas treniravimo ir testavimo duomenų imtims, bei variabilumo ir atstatymo paklaidos skaičiavimai naudojant *python* programinį kodą



16 pav. Atrinktų požymių koreliacijos lentelė (66 požymiai, 5 000 000 transakcijų duomenų imtis)

Požymių redukcijos algoritmų parametrams parinkti tyrimo metu atliekama analizė, kurios metu atsižvelgiama į algoritmo veikimo laiką, priklausomai nuo duomenų kiekio, modelių kokybinius

įverčius su redukuotais duomenimis, komponenčių suminį variabilumą ir atstatymo paklaidą. Parenkamas algoritmas ir komponenčių skaičius duodantis optimalius rezultatus.



17 pav. Atrinktų redukuotų požymių koreliacijos lentelė su pinigų plovimo žyma

2.7. Duomenų klasterizavimas

Pagal požymius transakcijos skirstomos į klasterius skirtingais metodais, siekiant kuo labiau sugrupuoti panašias transakcijas ir aptikti tas, kurios labiausiai išsiskiria. Transakcijų klasterizavimui naudojami DBSCAN, HDBSCAN, SOM, K-Means algoritmai. Klasterizavimo metu transakcijos sugrupuojamos į klasterius, kuriose siekiama sutelkti kuo panašesnes transakcijas. Kadangi nelegalių transakcijų kiekis yra itin mažas (0.004 %), siekiama kad klasteriuose atsidurtų legalios transakcijos, o tos kurios yra nelegalios, liks už jų ribų. Tokiu būdu už klasterių ribų esančios išskirtys (*angl. outliers*) traktuojamos kaip anomalijos, arba šio tyrimo kontekste - nelegalios transakcijos, kurios nėra panašios į kitas.

2.8. Anomalijų aptikimas

Anomalijų aptikimui naudojami klasterizavimo algoritmai ir tiesiogiai anomalijų aptikimui specifikuoti algoritmai. Vienas iš būdų kaip aptinkamos anomalijos, tai klasterizuojant transakcijas į grupes, visos išskirtys nepatenkančios į klasterius – priimamos kaip anomalijos.

Taip pat naudojami ir kiti unikalūs metodai anomalijų aptikimui. Vienas iš jų - *isolation forest* algoritmas, kuris duomenų neklasterizuodamas išskiria anomalijas iš visos duomenų imties. Kadangi šis algoritmas duomenų neklasterizuoja, jis veikia itin greitai. Kitas unikalus metodas – variaciniai autoenkoderiai. Šis algoritmas bandydamas atkurti taškus iš suspaustos informacijos, traktuoja taškus kaip anomalijas, jeigu jų atstatymo paklaida yra didesnė nei leistina riba.

2.9. Algoritmų ansamblis

Taip pat geresniam rezultatui pasiekti naudojamas ir ansamblio metodas. Kiekvienas klasifikavimo algoritmas turi savo spėjimą kas yra anomalija. Kadangi kiekvienas algoritmas jau yra įvertinamas kokybiniais įverčiais, klasifikatoriams suteikiami svoriai balsavime pagal AUC-PR įvertį. Tokiu būdu algoritmai, kurie turi geriausią AUC-PR įvertinimą, turi didžiausią svorį balsavime. Anomalijomis traktuojamos tos transakcijos, kurios po balsavimo turi didesnę riziką nei iš anksto nustatyta riba.

2.10. Kokybiniai įverčiai

Algoritmų kokybei įvertinti naudojami anksčiau aprašyti kokybiniai įverčiai. Duomenims esant nesubalansuotiems, *Accuracy* parametras nėra tikslus, tad jis nėra naudojamas. Atliekant tyrimą pastebėta, jog labiausiai modelio tinkamumą atspindi *F1* įvertis, taip pat ir *precision*, *recall* parametrai.

Lengvesniam algoritmo veikimo suvokimui taip pat išvedami ir *True Positive (TP)*, *False Positive (FP)*, *False Negative (FN)* parametrai. *True Negative (TN)* parametras nenaudojamas, nes esant nesubalansuotiems duomenims jo skaičius neatspindi situacijos, bei esant poreikiui jis gali būti lengvai išskaičiuojamas iš likusių trijų pateiktų parametrų.

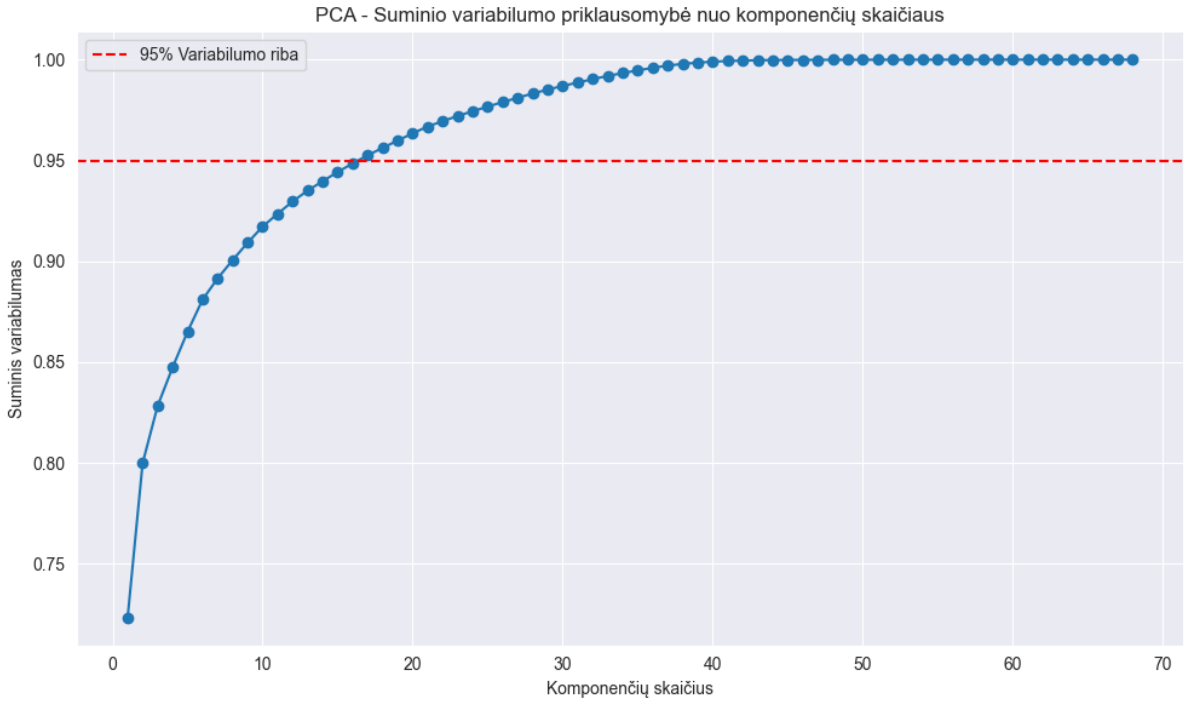
3. Tiriamoji dalis

Šiame skyriuje grindžiami duomenų apdorojimo parametrų pasirinkimai pagal kokybinius įverčius – duomenų redukcijos komponenčių kiekio parinkimas, duomenų perteklinio įtraukimo parametrai, apžvelgiami klasifikavimo bei klasterizavimo algoritmų rezultatai, pateikiami palyginimai kaip algoritmų rezultatai kinta duomenis papildomai apdorojus.

Be tradicinio klasifikavimo metodų, taip pat aprašomi alternatyvaus požiūrio į tyrimą, siekiant išskirti visas nelegalias transakcijas kaip klasterių išskirtis, rezultatai, aptariama kokius rezultatus pasiekia algoritmai individualiai, kokius rezultatus duoda jų jungimas į ansamblį.

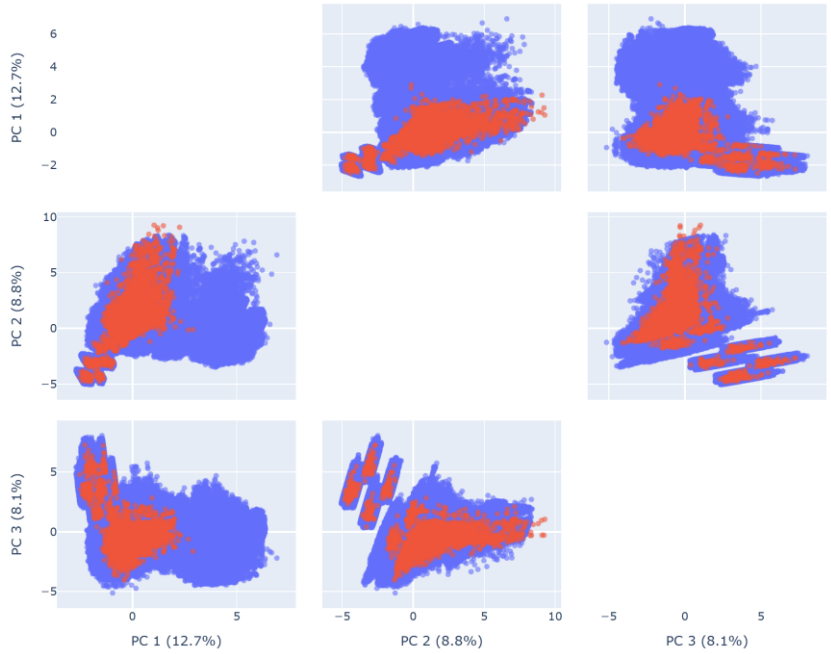
3.1. Požymių redukcijos komponenčių parinkimas

Požymių redukcijai naudojamas PCA algoritmas, kuriuo siekiama maksimaliai suspausti duomenis prarandant kuo mažiau informacijos. Priimama, kad šiame tyrime leistini 5% nuostoliai, tad komponenčių suminis variabilumas turėtų nebūti žemesnis nei 0.95. Atliekant 68 dimensijų požymių erdvės skaidymą į principines komponentes, nustatyta, kad norint turėti bent 95% duomenų atkuriamumą, reikalinga naudoti mažiausiai 17 principinių komponenčių. (18 pav.)

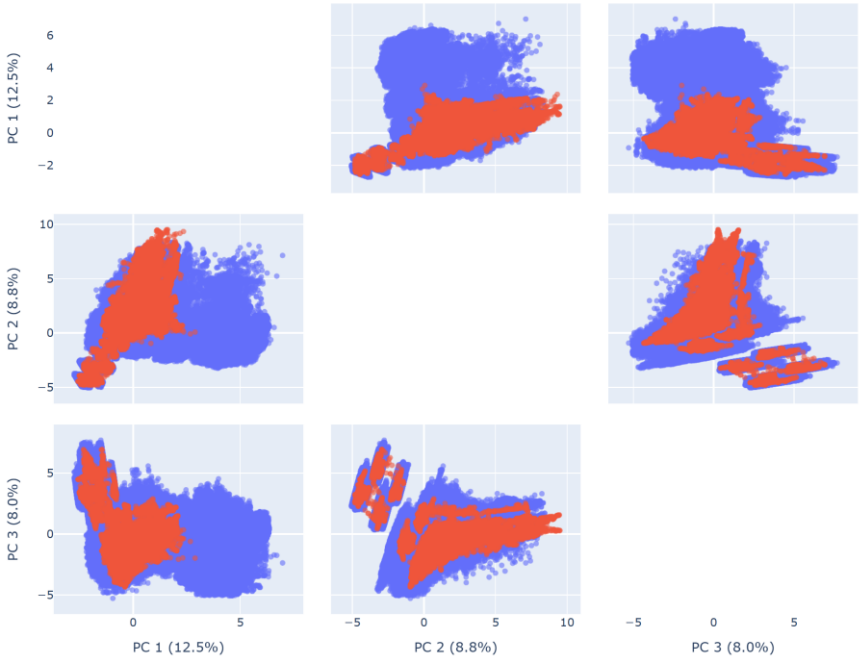


18 pav. Principinių komponenčių suminio variabilumo priklausomybė nuo komponenčių skaičiaus

Redukavus 68 dimensijų požymių erdvę į 17 principinių komponenčių, pirmų trijų komponenčių nešama informacija sudaro apie 35,4 % visų požymių nešamos informacijos ir yra atvaizduojama 19 pav. Sekančiame paveikslėlyje 20 atvaizduojama kaip požymiai principinėse komponentėse persiskirsto balansuojant duomenis SMOTE perpildymo (*angl. over*-sampling) algoritmu ir treniravimo duomenų imtyje padidinus nelegalių transakcijų kiekį 10 kartų.



19 pav. Principinių komponenčių pirmų trijų komponenčių (PC1, PC2, PC3) atvaizdavimas požymių erdvėje

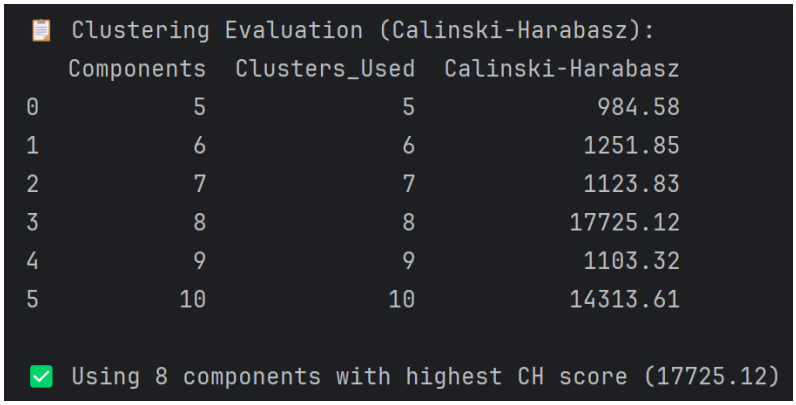


20 pav. Principinių komponenčių pirmų trijų komponenčių (PC1, PC2, PC3) atvaizdavimas požymių erdvėje naudojant SMOTE duomenų balansavimą.

3.2. Vartotojų grupavimas

Vartotojų grupavimas atliekamas dviem parašytais būdais – išskiriant tolygiai pagal vartotojų parametrus ir naudojant klasterizavimo metodą. Vartotojų skirstymas į tolygias grupes pagal mokėjimo sumos ar dienos transakcijos medianas ir vidurkius pasirodė neefektyvus, dėl to priimtas sprendimas naudoti *BayesianGaussianMixture* klasterizavimo metodą, kuris efektyviai apdoroja didelius duomenų kiekius, gali duomenis paskirstyti į klasterius, kurie nėra simetriškos formos,

nepalieka išskirčių už klasterių ribų. Klasteriams sugeneruoti naudotas kodas, skaičiuojantis *Calinski-Harabasz* įvertį, kurio didesnė vertė nurodo geresnį klasterizavimą. Atlikus klasterizavimą, priimta naudoti duomenis sugrupuotus į 8 klasterius. (21 pav.)



21 pav. Principinių komponenčių pirmų trijų komponenčių (PC1, PC2, PC3) atvaizdavimas požymių erdvėje naudojant SMOTE duomenų balansavimą.

3.3. Klasifikavimo rezultatai

Tyrimas atliekamas su pilnu duomenų rinkiniu, turinčiu 5 mln. transakcijų, siekiant palyginti algoritmų našumus ir papildomo duomenų apdorojimą daromą įtaką. Pirmiausia duomenų rinkinys klasifikuojamas 68 požymius standartizavus be papildomų apdorojimo algoritmų tipiniais klasifikatoriais. Gauti duomenys pateikiami 3 lentelėje.

3 lentelė. Klasifikavimo rezultatai naudojant 5 mln. transakcijų imtį su standartizuotais požymiais

Modelis	*Precision*	*Recall*	*F1 score*	Laikas (s)
Logistic regression(C1.0, max1000)	0	0	0	54,9
XGBoost(n100, lr0.1, mdpth3)	0	0	0	52,93
XGBoost(n100, lr0.25, mdpth3)	0,7	0,00390	0,00775	43,03
XGBoost(n100, lr0.75, mdpth5)	0,66667	0,01781	0,03469	45,6
Random forest(n100, mdpth10)	0	0	0	841,13
Isolation forest(n50, cont0.01, trh1)	0,00394	0,02226	0,00669	48,08
Isolation forest (n100, cont0.01, trh1)	0,00433	0,02449	0,00736	52,73
Isolation forest (n100, cont0.05, trh5)	0,00469	0,13244	0,00905	49,63

Lyginant rezultatus tarp modelių, pastebima, kad duomenų imtis su neapdorota didele dvejetainių požymių imtimi neduoda gerų rezultatų ir tipiniams klasifikavimo modeliams tokių duomenų klasifikavimas yra beveik neįmanomas. Visi duomenys buvo priskirti vienai klasei ir tik kai kuriose algoritmų parametrų konfigūracijose keli duomenų objektai buvo prognozuojami kaip nelegalūs. Tuo tarpu *Isolation forest* algoritmas, nors ir su prastais parametrais, bet su tokiais duomenimis dirba ir pateikia rezultatus.

Vėliau tas pats duomenų rinkinys klasifikuojamas atlikus duomenų balansavimą SMOTE algoritmu, padidinant treniravimo duomenyse nelegalių transakcijų kiekį 100 kartų ir pritaikius PCA požymių dimensijų redukciją iki 17 principinių komponenčių. Gauti rezultatai atvaizduojami 4 - 6 lentelėse.

4 lentelė. Klasifikavimo rezultatai – algoritmas *XGBoost*, su PCA ir SMOTE apdorojimu

Modelis	*Precision*	*Recall*	*F1 score*	Laikas (s)	TP	FP	FN
XGBoost(n50, lr0.05, mdpth3)	0,08329	0,33278	0,13323	5,94	598	6582	1199
XGBoost(n200, lr0.2, mdpth5)	0,07453	0,48915	0,07717	27,05	867	11869	930
XGBoost(n150, lr0.2, mdpth5)	0,07232	0,46132	0,08154	21,75	931	13601	866
XGBoost(n200, lr0.25, mdpth5)	0,07224	0,50640	0,03644	26,02	998	14746	799
XGBoost(n200, lr0.15, mdpth5)	0,07066	0,45186	0,08967	3185,29	959	14103	838

5 lentelė. Klasifikavimo rezultatai – algoritmas *Logistic Regression*, su PCA ir SMOTE apdorojimu

Modelis	*Precision*	*Recall*	*F1 score*	Laikas (s)	TP	FP	FN
LR(l1, fit_intercept=true)	0,04082	0,38342	0,07378	72.31	689	16192	1108
LR(l2, fit_intercept=true)	0,03925	0,39343	0,07138	16.79	707	17305	1090
LR(l1, fit_intercept=false)	0,00577	0,95103	0,01147	325,43	1709	294533	88
LR(l2, fit_intercept=false)	0,00636	0,94547	0,01263	19,95	1699	265505	98

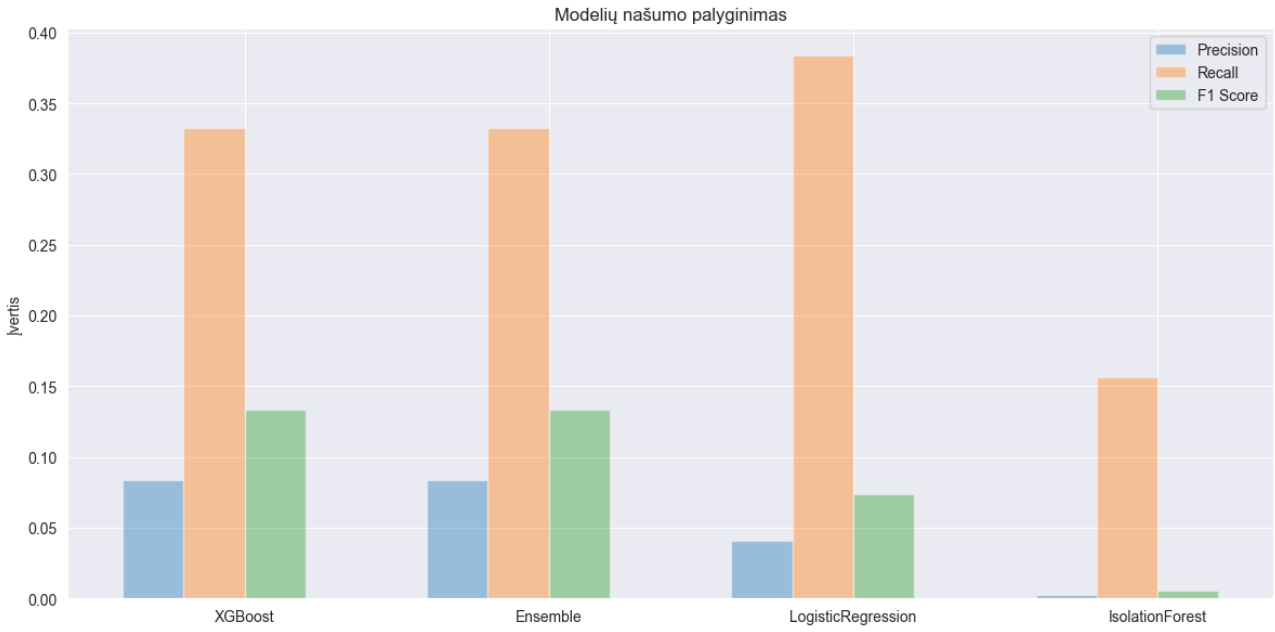
6 lentelė. Klasifikavimo rezultatai – algoritmas Isolation forest, su PCA ir SMOTE apdorojimu

Modelis	*Precision*	*Recall*	*F1 score*	Laikas (s)	TP	FP	FN
IF (n50, max 0.8, cont0.05)	0,00277	0,15637	0,00544	132,82	281	101258	1516
IF (n50, max 0.8, cont0.1)	0,00277	0,15637	0,00544	142,74	281	101258	1516
IF (n100, max 0.8, cont0.05)	0,00232	0,13133	0,00457	281,9	236	101326	1561
IF (n100, max 0.8, cont0.1)	0,00232	0,13133	0,00457	235,85	236	101326	1561
IF (n50, max 0.6, cont0.05,)	0,00209	0,11853	0,00412	133,94	213	101351	1584

Apdorojus duomenis SMOTE ir PCA algoritmais, klasifikatoriai geba duomenis klasifikuoti ir prognozuoti klasę. Tai įrodo pirmiausia PCA algoritmo naudą tipiniams klasifikatoriams, kuomet visi požymiai modeliuojami požymių erdvėje ir modeliui pateikiamos skaitinės vertės nešančios informaciją vietoje dvejetainių. Taip pat įrodo SMOTE algoritmo naudą, padidinant nelegalių transakcijų imtį ir taip leidžiant modelius labiau apmokyti apie tą duomenų klasę. Apibendrinti rezultatai grafiškai pateikiami histogramoje 22 pav.

Svarbu paminėti, kad pritaikius PCA analizę pagerėjo kai kurių klasifikatorių kokybiniai įverčiai ir pasikeitė jų greitaveika. *Logistical Regression, XGBoost* algoritmų duomenų apsimokymo ir prognozavimo laikas sutrumpėjo 2-3 kartus, rezultatai pagerėjo. Tuo tarpu *Isolation Forest* algoritmo rezultatai suprastėjo ir veikimas pailgėjo.

Ensemble metodas dėl kitų algoritmų mažo F1 įverčio, perėmė XGBoost spėjimus ir jų rezultatai identiški.



22 pav. Algoritmų vaizdinis kokybinių rodiklių palyginimas taikant SMOTE ir PCA apdorojimą

3.4. Klasterizavimo rezultatai

Klasterizavimo tyrimas taip pat atliekamas su pilnu duomenų rinkiniu, turinčiu 5 mln. transakcijų, siekiant palyginti algoritmų našumus ir papildomo duomenų apdorojimą daromą įtaką. Hipotezei patvirtinti klasterizavimas atliekamas tik su SMOTE ir PCA apdorotais duomenimis, bei DBSCAN algoritmu. Gauti duomenys pateikiami 5 lentelėje.

7 lentelė. Klasterizavimo rezultatai naudojant 5 mln. transakcijų imtį su standartizuotais požymiais

Modelis	*Precision*	*Recall*	*F1 score*	Laikas (s)
DBSCAN (eps 0.25, min 5)	0,01490	0,68394	0,02916	3408,87
DBSCAN (eps 0.5, min 5)	0,01233	0,69449	0,02422	3203,17
DBSCAN (eps 0.75, min 5)	0,01212	0,62322	0,02030	3168,62

Klasterizavimo rezultatai nėra stipriai išsiskiriantys savo tikslumu, tačiau didelis duomenų atsiminimas (*recall*) leidžia daryti prielaidą, kad klasterizavimo algoritmai gali būti naudingi įtartinų transakcijų aptikimui gerinti. Atsižvelgiant į gautus rezultatus, reikalinga išbandyti kitus algoritmus, pritaikyti įvairesnes algoritmo parametrų kombinacijas. Taip pat įžvelgiama galimybė klasterizavimo algoritmą naudoti duomenų imties susiaurinimui prieš klasifikavimą.

Išvados

1. Siekiant išgauti tikslesnius rezultatus, reikalinga įtraukti požymius, kurie nusakytų vartotojų tarpusavio sąsają ir nagrinėtų srautus. Grafomis remiamais algoritmais išskirti požymiai, atsižvelgiant į kitus atliktus tyrimus, leistų pasiekti geresnių rezultatų.
2. Nustatyta, kad duomenims taikant SMOTE algoritmą jų balansavimui bei apdorojant požymius PCA duomenų redukcijos technika išlaikant 95 % suminį variabilumą, iš individualiai veikiančių algoritmų *XGBoost* pasiekė geriausią *F1* įvertį (0,13323). Toks rezultatas nėra itin geras lyginant su kituose tyrimuose pasiekiamais rezultatais, tačiau atsižvelgiant į tyrimo apribojimus, rezultatas yra tenkinamas ir yra tinkamas įtartinų transakcijų išskyrimui, jei jo tikslumas tenkina naudotoją, arba transakcijų imties susiaurinimui
3. Nustatyta, kad naudojant PCA algoritmą požymių erdvės dimensijų redukavimui, algoritmų *Logistic Regression, XGBoost, Isolation forest* greitaveika kai kuriose jų parametrų konfigūracijose pagerėjo iki 3 kartų, tačiau pagerėjimas nėra vienodas tarp visų algoritmų.
4. Išsiaiškinta, kad ansamblio algoritmas balsavimo metodu šiai užduočiai atlikti su klasifikatoriais dalinai gerina efektyvumą. Kai kuriais bandytais atvejais kolektyvinis algoritmų sprendimas sumažino klaidingai nustatytų teisingų atvejų skaičių, bet tuo pačiu šiek tiek sumažindamas ir bendrai aptiktų įtartinų transakcijų kiekį, tačiau parinkus geriausiai veikiančius klasifikatorius, dėl mažo kitų algoritmų F1 įverčio, ansamblio metodas perėmė geriausio *XGBoost* algoritmo prognozes.
5. Dalinai patvirtinta hipotezė, kad klasterizuojant duomenis klasterių išskirtys gali būti prilyginamos anomalijoms. Tyrimo metu naudojant DBSCAN algoritmą pasiektas didelis atsiminimas, tačiau tikslumas yra mažas. Norint taikyti šį metodą individualiai reikalingas papildomas duomenų apdorojimas, tačiau šį metodą, dėl jo didelio atsiminimo, pravartu taikyti duomenų imties siaurinimui ir tikslinimui prieš klasterizavimą.

Literatūros sąrašas

1. Altman, E., Blanuša, J., von Niederhäusern, L., Egressy, B., Anghel, A., & Atasu, K. (2024). Realistic Synthetic Financial Transactions for Anti-Money Laundering Models. Paimta 2024 m. rugsėjo 12 d. iš https://arxiv.org/abs/2306.16424
2. Bakry, A. N., Alsharkawy, A. S., Farag, M. S., & Raslan, K. R. (2024). Combating Financial Crimes with Unsupervised Learning Techniques: Clustering and Dimensionality Reduction for Anti-Money Laundering. *Al-Azhar Bulletin of Science, 35*. doi:10.58675/2636-3305.1664
3. Bi, W., Trinh, T. K., & Fan, S. (2024 m. lapkritis). Machine Learning-Based Pattern Recognition for Anti-Money Laundering in Banking Systems. *Journal of Advanced Computing Systems (JACS), 4,* 30-41. doi:10.69987/JACS.2024.41103
4. Brandt, J., & Lanzén, E. (2020). A Comparative Review of SMOTE and ADASYN in Imbalanced Data Classification. Uppsala. Paimta 2024 m. gegužės 14 d. iš https://www.diva-portal.org/smash/get/diva2:1519153/FULLTEXT01.pdf
5. Breiman, L. (2001). Random forests. *Machine Learning*, 5-32. Paimta 2025 m. gegužės 14 d. iš https://doi.org/10.1023/A:1010933404324
6. Cabirta, A. (2019 m. 01 04 d.). *Anti-money laundering challenges in the financial sector*. (BBVA) Paimta 2025 m. 01 10 d. iš https://www.bbva.com/en/anti-money-laundering-challenges-in-the-financial-sector/
7. Chen, Z., Soliman, W. M., Nazir, A., & Shorfuzzaman, M. (2021). Variational Autoencoders and Wasserstein Generative Adversarial Networks for Improving the Anti-Money Laundering Process. *IEEE Access, 9,* 83762-83785. doi:10.1109/ACCESS.2021.3086359
8. Chen, Z., Van Khoa, L. D., Teoh, E. N., Nazir, A., Karuppiah, E. K., & Lam, K. S. (2017). Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review. *Springer Nature 2018*. Nuskaityta iš https://link.springer.com/article/10.1007/s10115-017-1144-z
9. Colladon, F. A., & Remondi, E. (2017). Using social network analysis to prevent money laundering. *Expert Systems with Applications, 67*(0957-4174), 49-58. doi:10.1016/j.eswa.2016.09.029
10. contributors, W. (2025 m. balandžio 30 d.). *Cluster analysis*, 1288045616. (T. F. Wikipedia, Prodiuseris) Paimta 2025 m. gegužės 10 d. iš Wikipedia: https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=1288045616
11. (2024). *Criminals Use Generative Artificial Intelligence to Facilitate Financial Fraud.* Public Service Announcement, Federal Bureau of Investigation. Paimta 2025 m. balandžio 27 d. iš https://www.ic3.gov/PSA/2024/PSA241203
12. Doppalapudi, P., Kumar, P., Murphy, A., Rougeaux, C., Stearns, R., Werner, S., & Zhang, S. (2022 m. spalio 7 d.). The fight against money laundering: Machine learning is a game changer. Paimta 2025 m. balandžio 26 d. iš https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/the-fight-against-money-laundering-machine-learning-is-a-game-changer
13. Elucidate team. (2024 m. gruodžio 16 d.). 5 strategies to simplify complex data systems in financial crime risk operations. Paimta 2025 m. balandžio 26 d. iš https://www.elucidate.co/blog/5-strategies-to-simplify-complex-data-systems-in-financial-crime-risk-operations

14. Europos Sąjunga. (2022 m. gruodžio 14 d.). EUROPOS PARLAMENTO IR TARYBOS REGLAMENTAS (ES) 2022/2554 dėl skaitmeninės veiklos atsparumo finansų sektoriuje, kuriuo iš dalies keičiami reglamentai (EB) Nr. 1060/2009, (ES) Nr. 648/2012, (ES) Nr. 600/2014, (ES) Nr. 909/2014 ir (ES) 2016/1011. Paimta 2025 m. balandžio 26 d. iš https://eur-lex.europa.eu/legal-content/LT/TXT/HTML/?uri=CELEX:32022R2554#enc_1
15. Fan, J., Shar, L. K., Zhang, R., Liu, Z., Yang, W., Niyato, D., . . . Lam, K.-Y. (2025). Deep Learning Approaches for Anti-Money Deep Learning Approaches for Anti-Money Framework, and Directions. *IEEE*. Nuskaityta iš https://arxiv.org/abs/2503.10058
16. FATF. (2020). *Money Laundering and Terrorist Financing Red Flag Indicators Associated with Virtual Assets.* Paryžius: FATF. Paimta 2024 m. lapkričio 26 d. iš https://www.fatf-gafi.org/content/dam/fatf-gafi/reports/Virtual-Assets-Red-Flag-Indicators.pdf#page=6.50
17. fenergo. (2024). *AML Enfrocement Action in 2024.* Fenergo. Paimta 2025 m. balandžio 27 d. iš https://www.fenergo.com/aml-report
18. Follette, K. B. (2023). An Introduction to High Contrast Differential Imaging of Exoplanets and Disks. doi:10.48550/arXiv.2308.01354
19. Guidehouse. (2023). Why Consider Machine Learning For Transaction Monitoring? *Guidehouse*. Paimta 2025 m. Balandžio 6 d. iš https://guidehouse.com/insights/financial-crimes/2023/consider-machine-learning-for-transaction-monitoring
20. Harris, D., Pyndiura, K., Sturrock, S., & Christensen, R. (2021). Using real-world transaction data to identify money laundering: Leveraging traditional regression and machine learning techniques. *STEM Fellowship Journal*. doi:10.17975/sfj-2021-006
21. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Nuskaityta iš https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf#page=2.34
22. Hearty, N. (2024 m. 04 16 d.). *Common Money Laundering Techniques Explained*. Paimta 2025 m. 01 19 d. iš Rahman Ravelli: https://www.rahmanravelli.co.uk/expertise/anti-money-laundering-investigations/articles/common-money-laundering-techniques-explained/
23. IBM. (be datos). *What is the KNN algorithm?* Nuskaityta iš IBM: https://www.ibm.com/think/topics/knn
24. Jaadi, Z., Whitfield, B., & Pierre, S. (2024 m. vasario 23 d.). *Principal Component Analysis (PCA): A Step-by-Step Explanation*. Paimta 2025 m. gegužės 14 d. iš builtin: https://builtin.com/data-science/step-step-explanation-principal-component-analysis
25. Jaadi, Z., Whitfield, B., & Sawtell-Rickson, J. (2025 m. sausio 9 d.). *Data Standardization: How to Do It and Why It Matters*. Nuskaityta iš builtin: https://builtin.com/data-science/when-and-why-standardize-your-data
26. Jensen, R. I., & Iosifidis, A. (2023). Fighting Money Laundering With Statistics and Machine Learning. *IEEE Access*. doi:10.1109/ACCESS.2023.3239549
27. Kulkarni, S. (2023). *Machine learning for Anti-money laundering (ML for AML).* KPMG. Nuskaityta iš https://kpmg.com/be/en/home/insights/2023/08/lh-machine-learning-for-anti-money-laundering.html
28. Le-Khac, N.-A., Markos, S., & Kechadi, M.-T. (be datos). Towards a New Data Mining-Based Approach for Anti-Money Laundering in an International Investment. *Digital Forensics and Cyber Crime*. Paimta 2024 m. 04 27 d. iš https://link.springer.com/chapter/10.1007/978-3-642-11534-9_8

29. LexisNexis Risk Solutions. (2021). *Global Spend on Financial Crime Compliance at Financial Institutions Reaches $213.9 Billion USD According to LexisNexis Risk Solutions Study*. Paimta 2025 m. balandžio 27 d. iš https://risk.lexisnexis.com/global/en/about-us/press-room/press-release/20210609-tcoc-global-study
30. Lucinity. (2024 m. spalio 7 d.). Integrating Modern AML Software with Legacy Systems: Challenges and Solutions. Paimta 2025 m. balandžio 26 d. iš https://lucinity.com/blog/integrating-modern-aml-software-with-legacy-systems-challenges-and-solutions
31. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research, 12*, 2825-2830. Nuskaityta iš https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
32. Preis, A. (2025 m. balandžio 7 d.). *Combating Deepfakes in Financial Service*. Nuskaityta iš PingIdentity: https://www.pingidentity.com/en/resources/blog/post/combat-deepfakes-financial-services.html
33. Regaya, Y., Fadli, F., & Amira, A. (2021). Point-Denoise: Unsupervised outlier detection for 3D point clouds enhancement. Nuskaityta iš https://doi.org/10.1007/s11042-021-10924-x
34. Sharma, S. (2024 m. gruodžio 9 d.). *One Hot Encoding vs Label Encoding*. Paimta 2025 m. gegužės 13 d. iš Geeks for geeks: https://www.geeksforgeeks.org/one-hot-encoding-vs-label-encoding/
35. Simplilearn. (2025 m. gegužės 4 d.). *What is XGBoost? An Introduction to XGBoost Algorithm in Machine Learning*. Nuskaityta iš simplilearn: https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article
36. Stripe. (2023 m. Birželio 27 d.). *How machine learning works for payment fraud detection and prevention*. Paimta 2025 m. Balandžio 6 d. iš Stripe: https://stripe.com/en-lt/resources/more/how-machine-learning-works-for-payment-fraud-detection-and-prevention
37. Tigerschiold, T. (2022 m. lapkričio 7 d.). *What is Accuracy, Precision, Recall and F1 Score?* Paimta 2025 m. sausio 12 d. iš Labelf: https://www.labelf.ai/blog/what-is-accuracy-precision-recall-and-f1-score.
38. Trozze, A., Davies, T., & Kleinberg, B. (2023). Of degens and defrauders: Using open-source investigative tools to. *Elsevier, 46*(2666-2817). doi:10.1016/j.fsidi.2023.301575
39. Umbražiūnaitė, J. (2005). Pagrindinių komponenčių metodo realizacijos neuroniniais tinklais tyrimas. *Magistro darbas*. Paimta 2025 m. gegužės 14 d. iš https://portalcris.vdu.lt/server/api/core/bitstreams/0fc1e083-90b6-4d44-bb6e-07984e7f0162/content
40. Venčkauskas, A., Grigaliūnas, Š., Pocius, L., Brūzgienė, R., & Romanovs, A. (2025). Machine Learning in Money Laundering Detection Over Blockchain Technology. *IEEE Access*, 7555-7573. doi:10.1109/ACCESS.2024.3452003
41. Viloria, A., Lezama, O. B., & Mercado-Caruzo, N. (2020). Unbalanced data processing using oversampling: Machine Learning. *Procedia Computer Science, 175*, 108-113. Nuskaityta iš https://www.sciencedirect.com/science/article/pii/S1877050920316975
42. XE. (2022 m. rugsėjo 1 d.). *Currency table: USD - US Dollar*. (XE) Paimta 2024 m. spalio 14 d. iš https://www.xe.com/currencytables/?from=USD&date=2022-09-01#table-section

43. yahoo!finance. (2022 m. rugsėjo 1 d.). *yahoo!finance*. Paimta 2024 m. spalio 14 d. iš https://finance.yahoo.com/quote/BTC-USD/history/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAANjcaRpBz_vmnYxeDFmFFiR4a_42fUJORM1NBKzhPwHHUYzU0Bugw9BeVAnUHUXVyFm9ao_pr5LBRkHbcoY2pY6gdrVUun3YzHqgJTZ3TDWkPySq3vOd98Xl1aJ5o
44. Yang, Y., Lian, B., Li, L., Chen, C., & Li, P. (2014). DBSCAN clustering algorithm applied to identify suspicious financial transactions. doi:10.1109/CyberC.2014.89
45. Yemulwar, S. (2019 m. rugsėjo 27 d.). *Medium*. Paimta 2025 m. gegužės 3 d. iš Feature Selection Techniques: https://medium.com/analytics-vidhya/feature-selection-techniques-2614b3b7efcd
46. Yenigün, O. (2024 m. kovo 11 d.). *DBSCAN Clustering Algorithm Demystified*. Nuskaityta iš builtin: https://builtin.com/articles/dbscan

Priedai

1 priedas. Programinis kodas duomenų apdorojimui Jupyter programavimo aplinkoje

```
#%%
import pandas as pd
import os
import numpy as np
import time
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.decomposition import PCA
from sympy.physics.units import amount

from evaluation_plotting import generate_plots
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
import plotly.express as px



start_time = time.time()
print("Starting data reading and preprocessing")
print("Loading dataset")
aml = pd.read_csv(os.path.join(os.getcwd(), "res", "HI-Small_Trans.csv"),
                  dtype={"Amount Received": "float", "Amount Paid": "float"}, na_values=[" (null)"])
print("Processing data")

# Renaming Laundering column
aml = aml.rename(columns={'Is Laundering': 'Laundering'})

# Converting timestamp
aml['Timestamp'] = pd.to_datetime(aml['Timestamp'], format="%Y/%m/%d %H:%M")

#%% md

#%% md
Payment amount conversion to USD dollars
#%%
exchange_rates = {
      'Bitcoin': 0.00005, 'US Dollar': 1, 'Euro': 1.00539, 'sic': 1.47606, 'Yuan': 6.90672,
      'Rupee': 79.66131, 'Mexican Peso': 20.23564, 'Yen': 139.94473, 'UK Pound': 0.86696, 'Ruble':
60.19367,
      'Canadian Dollar': 1.31641, 'Swiss Franc': 0.98387, 'Brazil Real': 5.23102, 'Saudi Riyal': 3.75,
      'Shekel': 3.39583, 'Australian Dollar': 1.47606
   }

aml["Amount Received USD"] = aml.apply(
   lambda row: row["Amount Received"] / exchange_rates[row["Receiving Currency"]],
```

```
    axis=1
)

aml["Amount Paid USD"] = aml.apply(
    lambda row: row["Amount Paid"] / exchange_rates[row["Payment Currency"]],
    axis=1
)

#%% md
Static features from timestamp
#%%
aml['Hour'] = aml['Timestamp'].dt.hour
aml['Weekday'] = aml['Timestamp'].dt.dayofweek
aml['Weekend'] = (aml['Weekday'] >= 5).astype(int)
aml["Same account"] = aml["Account"] == aml["Account.1"]
aml["Same bank"] = aml["From Bank"] == aml["To Bank"]
aml["Same currency"] = aml["Receiving Currency"] == aml["Payment Currency"]

#%%
conditions = [
    (aml['Hour'] >= 0) & (aml['Hour'] < 6),
    (aml['Hour'] >= 6) & (aml['Hour'] < 12),
    (aml['Hour'] >= 12) & (aml['Hour'] < 18),
    (aml['Hour'] >= 18) & (aml['Hour'] <= 23)
]

choices = ['Night', 'Morning', 'Day', 'Evening']

aml['Daytime'] = np.select(conditions, choices, default='Unknown')
aml
#%% md
One hot encoding of payment format
#%%
unique_values_daytime = aml["Daytime"].unique()
onehot = OneHotEncoder(sparse=False, handle_unknown='ignore')
onehot.fit(pd.DataFrame(unique_values_daytime, columns=["Daytime"]))
daytime_encoded = onehot.transform(aml[["Daytime"]])
daytime_df = pd.DataFrame(daytime_encoded, columns=[f"Daytime {cat}" for cat in
onehot.categories_[0]])
#%% md
One hot encoding of payment format
#%%
unique_values_format = aml["Payment Format"].unique()
onehot = OneHotEncoder(sparse=False, handle_unknown='ignore')
onehot.fit(pd.DataFrame(unique_values_format, columns=["Payment Format"]))
payment_format_encoded = onehot.transform(aml[["Payment Format"]])
```

```
payment_format_df = pd.DataFrame(payment_format_encoded, columns=[f"PF {cat}" for cat in onehot.categories_[0]])

#%% md
One hot encoding of currencies
#%%
unique_values_currencies = aml[["Receiving Currency", "Payment Currency"]].stack().drop_duplicates().tolist()
onehot = OneHotEncoder(sparse=False, handle_unknown='ignore')
onehot.fit(pd.DataFrame(unique_values_currencies, columns=["Currency"]))
receiving_encoded = onehot.transform(aml[["Receiving Currency"]].rename(columns={"Receiving Currency": "Currency"}))
payment_encoded = onehot.transform(aml[["Payment Currency"]].rename(columns={"Payment Currency": "Currency"}))
receiving_df = pd.DataFrame(receiving_encoded, columns=[f"Currency rec. {cat}" for cat in onehot.categories_[0]])
payment_df = pd.DataFrame(payment_encoded, columns=[f"Currency pay. {cat}" for cat in onehot.categories_[0]])

#%% md
Merge encoded parameters to one dataframe and remove unnecessary columns
#%%
aml_encoded = pd.concat([aml.drop(columns=["Receiving Currency", "Payment Currency", "Payment Format", "Daytime"]),
                         receiving_df.astype(bool), payment_df.astype(bool), payment_format_df.astype(bool), daytime_df.astype(bool)], axis=1)
aml_encoded
#%% md
Sort data by the time and split data to train/test splits
#%%
aml_encoded = aml_encoded.sort_values(by="Timestamp")

aml_encoded_y = aml_encoded[["Laundering"]]
aml_encoded_x = aml_encoded.drop("Laundering", axis=1)

split_index = int(len(aml)*0.8)

# Copy training data for group parameters calculation
aml_copy = aml_encoded_x.iloc[:split_index].copy()
#%% md
Dividing users to groups
#%%
aml_copy["Day"] = aml_copy["Timestamp"].dt.date

# Step 1: Transactions per account per day
```

```
txn_counts = aml_copy.groupby(["Account", "Day"]).size().reset_index(name="Transactions per Day")

# Step 2: Median frequency per account
freq_median = txn_counts.groupby("Account")["Transactions per Day"].median().reset_index()
freq_median.rename(columns={"Transactions per Day": "Frequency median"}, inplace=True)

# Step 3: Mean frequency per account
freq_mean = txn_counts.groupby("Account")["Transactions per Day"].mean().reset_index()
freq_mean.rename(columns={"Transactions per Day": "Frequency mean"}, inplace=True)

# Step 4: Median amount per account
amount_median = aml_copy.groupby("Account")["Amount Paid USD"].median().reset_index()
amount_median.rename(columns={"Amount Paid USD": "Amount median"}, inplace=True)

# Step 5: Merge all into user_profile
user_profile = freq_median.merge(freq_mean, on="Account", how="left")
user_profile = user_profile.merge(amount_median, on="Account", how="left")
user_profile
#%%
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.scatterplot(data=user_profile, x="Frequency median", y="Amount median")

plt.yscale("log")
plt.xlim(0, 40)  # Limit x-axis from 1 to 100
plt.xlabel("Frequency Median (Transactions per Day)")
plt.ylabel("Amount Median (USD)")
plt.title("Frequency vs Payment Amount Median per Account")
plt.grid(True)
plt.tight_layout()
plt.show()
#%%
bins = [0, 1, 2, 5, 10, float("inf")]
labels = [1, 2, 3, 4, 5]
user_profile["Group"] = pd.cut(user_profile["Frequency median"], bins=bins, labels=labels)
user_profile
#%% md
Group attributes
#%%
# Temporary merge to bring "Group" into aml_copy
temp_merged = aml_copy.merge(user_profile[["Account", "Group"]], on="Account", how="left")

# Group-level median of Amount Paid USD
```

```
group_profile = temp_merged.groupby("Group")["Amount Paid USD"].median().reset_index()
group_profile.rename(columns={"Amount Paid USD": "Amount Paid USD Median"}, inplace=True)

# Group-level mean of Amount Paid USD
group_mean = temp_merged.groupby("Group")["Amount Paid USD"].mean().reset_index()
group_mean.rename(columns={"Amount Paid USD": "Amount Paid USD Mean"}, inplace=True)

# Group-level std of Amount Paid USD
group_std = temp_merged.groupby("Group")["Amount Paid USD"].std().reset_index()
group_std.rename(columns={"Amount Paid USD": "Amount Paid USD Std"}, inplace=True)

# Unique account count per group (from user_profile)
group_counts = user_profile.groupby("Group")["Account"].nunique().reset_index()
group_counts.rename(columns={"Account": "Account Count"}, inplace=True)

# Total transaction count per group (from temp_merged)
txn_counts = temp_merged.groupby("Group")["Account"].count().reset_index()
txn_counts.rename(columns={"Account": "Total Transaction Count"}, inplace=True)

# Merge all metrics into group_profile
group_profile = (
    group_profile
    .merge(group_std, on="Group", how="left")
    .merge(group_mean, on="Group", how="left")
    .merge(group_counts, on="Group", how="left")
    .merge(txn_counts, on="Group", how="left")
)

del temp_merged, group_std, group_mean, group_counts, txn_counts

group_profile
#%% md
Transaction features for all transactions (train + test split)
#%%
aml_enriched = aml_encoded_x.merge(user_profile[["Account", "Group"]], on="Account", how="left")

# Merge group stats (mean and std) into aml_enriched
group_stats = group_profile[["Group", "Amount Paid USD Mean", "Amount Paid USD Std"]]
aml_enriched = aml_enriched.merge(group_stats, on="Group", how="left")

# Calculate Z-score of "Amount Paid"
aml_enriched["Z Score Paid"] = (
    (aml_enriched["Amount Paid"] - aml_enriched["Amount Paid USD Mean"]) /
    aml_enriched["Amount Paid USD Std"]
```

```
)

aml_enriched["Relative deviation paid"] = (
   abs(aml_enriched["Amount Paid USD"] - aml_enriched["Amount Paid USD Mean"]) / aml_enriched["Amount Paid USD Mean"]
)

aml_enriched
#%% md
Remove unused features
#%%
aml_cleared = aml_enriched.drop(columns=["Timestamp", "Amount Paid USD Mean", "Amount Paid USD Std", "From Bank", "Account", "To Bank", "Account.1"])
aml_cleared
#%% md
One hot encoding for last features
#%%
unique_values_hour = aml_cleared["Hour"].unique()
onehot = OneHotEncoder(sparse=False, handle_unknown='ignore')
onehot.fit(pd.DataFrame(unique_values_hour, columns=["Hour"]))
hour_encoded = onehot.transform(aml_cleared[["Hour"]])
hour_df = pd.DataFrame(hour_encoded, columns=[f"Hour {cat}" for cat in onehot.categories_[0]])

unique_values_weekday = aml_cleared["Weekday"].unique()
onehot = OneHotEncoder(sparse=False, handle_unknown='ignore')
onehot.fit(pd.DataFrame(unique_values_weekday, columns=["Weekday"]))
weekday_encoded = onehot.transform(aml_cleared[["Weekday"]])
weekday_df = pd.DataFrame(weekday_encoded, columns=[f"Weekday {cat}" for cat in onehot.categories_[0]])

unique_values_user_group = aml_cleared["Group"].unique()
onehot = OneHotEncoder(sparse=False, handle_unknown='ignore')
onehot.fit(pd.DataFrame(unique_values_user_group, columns=["Group"]))
user_group_encoded = onehot.transform(aml_cleared[["Group"]])
user_group_df = pd.DataFrame(user_group_encoded, columns=[f"Group {cat}" for cat in onehot.categories_[0]])

aml_onehot = pd.concat([aml_cleared.drop(columns=["Hour", "Weekday", "Group"]),
                        hour_df.astype(bool), weekday_df.astype(bool), user_group_df.astype(bool)], axis=1)

del unique_values_hour, unique_values_weekday, unique_values_user_group, hour_df, weekday_df, user_group_df

aml_onehot
#%% md
```

```
Data split to train/test
#%%
split_index = int(len(aml)*0.8)

# Copy training data for group parameters calculation
aml_train_x = aml_onehot.iloc[:split_index]
aml_test_x = aml_onehot.iloc[split_index:]
aml_train_y = aml_encoded_y.iloc[:split_index]
aml_test_y = aml_encoded_y.iloc[split_index:]
#%% md
Write to memory and free up RAM
#%%
# Write to .csv (to be done)

# Free up memory
import gc

del aml_cleared, aml_encoded, aml_enriched, aml_onehot, aml_copy, group_profile, user_profile

gc.collect()
#%% md
Data standardizing
#%%
from sklearn.compose import ColumnTransformer

columns_to_scale = ["Amount Received", "Amount Paid", "Amount Received USD", "Amount Paid
USD", "Z Score Paid", "Relative deviation paid"]

ct = ColumnTransformer(
    transformers=[('scaler', StandardScaler(), columns_to_scale)],
    remainder='passthrough'
)
aml_train_x_scaled = ct.fit_transform(aml_train_x)
aml_test_x_scaled = ct.transform(aml_test_x)

aml_test_x_scaled
#%% md
SMOTE
#%%
minority_count = sum(aml_train_y["Laundering"] == 1)
target_minority_count = minority_count * 100
smote = SMOTE(random_state=42, sampling_strategy={1: target_minority_count})
aml_train_x_balanced,      aml_train_y_balanced      =      smote.fit_resample(aml_train_x_scaled,
aml_train_y)

print(sum(aml_train_y_balanced["Laundering"] == 1))
```

```
print(minority_count)
#%% md
PCA
#%%
pca = PCA(n_components = 88)
pca.fit_transform(aml_train_x_balanced)
cumulative_evr = np.cumsum(pca.explained_variance_ratio_)
# Components needed for >95% variance
n_components_95 = np.argmax(cumulative_evr >= 0.95) + 1  # +1 for 1-based indexing
n_components_95
#%%
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_evr) + 1), cumulative_evr, marker='o', linestyle='-')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('PCA - Cumulative Explained Variance')
plt.grid(True)
plt.axhline(y=0.95, color='r', linestyle='--', label='95% Variance Threshold')
plt.legend()
plt.tight_layout()
plt.show()
#%%
aml_test_x_scaled = np.nan_to_num(aml_test_x_scaled, nan=0.0, posinf=0.0, neginf=0.0)
pca = PCA(n_components = n_components_95)
aml_train_x_pca = pca.fit_transform(aml_train_x_balanced)
aml_test_x_pca = pca.transform(aml_test_x_scaled)

aml_test_x_pca
#%% md
Export to .csv after SMOTE and PCA
#%%
def print_df(df, file):
    data_dir = os.path.join(os.getcwd(), 'data')
    os.makedirs(data_dir, exist_ok=True)
    path = os.path.join(data_dir, file)
    try:
        df.to_csv(path, index=False, header=True, sep=',')
    except PermissionError as e:
        print(f"PermissionError: Unable to write to {path}. Ensure the file is not open in another application and check OneDrive sync status. Error: {e}")
        raise

aml_train_x_scaled_df = pd.DataFrame(aml_train_x_scaled)
```

```
aml_test_x_scaled_df = pd.DataFrame(aml_test_x_scaled)
aml_train_x_pca_df = pd.DataFrame(aml_train_x_pca)
aml_test_x_pca_df = pd.DataFrame(aml_test_x_pca)

aml_train_x_scaled_df["Laundering"] = aml_train_y.values
aml_test_x_scaled_df["Laundering"] = aml_test_y.values
aml_train_x_pca_df["Laundering"] = aml_train_y_balanced.values
aml_test_x_pca_df["Laundering"] = aml_test_y.values

print_df(aml_train_x_scaled_df, "aml_train_scaled.csv")
print_df(aml_test_x_scaled_df, "aml_test_scaled.csv")
print_df(aml_train_x_pca_df, "aml_train_balanced.csv")
print_df(aml_test_x_pca_df, "aml_test_balanced.csv")
```