



Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Anomalių aptikimo vandens valymo sistemoje efektyvumo tyrimas, taikant mašininio mokymo metodus

Baigiamasis magistro projektas

Mykolas Adomaitis

Projekto autorius

Lekt. prakt. Kęstutis Brazauskas

Vadovas

Kaunas, 2026



Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Anomalių aptikimo vandens valymo sistemoje efektyvumo tyrimas, taikant mašininio mokymo metodus

Baigiamasis magistro projektas

Valdymo technologijos (6211EX014)

Mykolas Adomaitis

Projekto autorius

Lekt. prakt. Kęstutis Brazauskas

Vadovas

Doc. dr. Arūnas Lipnickas

Recenzentas

Kaunas, 2026



Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Mykolas Adomaitis

Anomalijų aptikimo vandens valymo sistemoje efektyvumo tyrimas, taikant mašininio mokymo metodus

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) 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 sąrašė;
3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);
4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Mykolas Adomaitis

Patvirtinta elektroniniu būdu

Adomaitis, Mykolas. Anomalijų aptikimo vandens valymo sistemoje efektyvumo tyrimas, taikant mašininio mokymo metodus. Magistro baigiamasis projektas / vadovas lekt. prakt. Kęstutis Brazauskas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos inžinerija, inžinerijos mokslai.

Reikšminiai žodžiai: anomalijų aptikimas, mašininis mokymas, efektyvumo tyrimas.

Kaunas, 2026. 63 p.

Santrauka

Pramoninių sistemų veikimo metu svarbu, kad jos dirbtų kuo stabiliau. Tačiau tokiuose procesuose dėl tam tikrų vidinių ar išorinių veiksnių neretai atsiranda anomalijos, kurias dėl procesų kompleksiskumo bei duomenų gausos dažnai sunku nustatyti. Todėl jų aptikimui pradėti naudoti įvairūs mašininio mokymo metodai.

Šiame darbe realizuojami penki skirtingi mašininio mokymo modeliai, kurie apmokomi aptikti anomalijas vandens valymo sistemos duomenyse. Tyrimui naudojami „SWaT“ ir „WADI“ duomenų rinkiniai, surinkti iš realių vandens valymo sistemų, kurie prieš naudojant modelių apmokymui apdorojami. Naudojant optimizavimo metodus parenkami tinkamiausi modelių parametrai. Modelių efektyvumas vertinamas naudojant tiek standartines metrikas, tokias kaip „F1“ įvertis, tiek diapazono metrikas, kurios modelių efektyvumą vertina laiko intervalais, kas kaip nustatyta tyrime yra labiau tinkamas modelių efektyvumo vertinimo metodas pramoninių procesų duomenyse.

Atlikus reikalingus tirtų mašininio mokymo modelių efektyvumo nustatymo žingsnius, padarytos išvados apie tinkamiausio modelio anomalijų aptikimui vandens valymo sistemos duomenyse parinkimą.

Adomaitis, Mykolas. Investigation of Efficiency of Anomaly Detection Using Machine Learning Methods in Water Treatment System. Master's Final Degree Project / supervisor lect. asst. Kęstutis Brazauskas; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): electronics engineering, engineering sciences.

Keywords: anomaly detection, machine learning, investigation of efficiency.

Kaunas, 2026. 63.

Summary

Industrial control systems must be running as stable as possible. However, in such processes, due to certain internal or external factors, anomalies may occur. Such anomalies are difficult to detect because of the complexity of the processes and large volume of process data. Therefore, various machine learning methods are used for their detection.

In this work, five different machine learning models are implemented and trained to detect anomalies in water treatment system data. The study uses the “SWaT” and “WADI” datasets, collected from real water treatment systems, which are preprocessed before being used for model training. Optimization methods are applied to select the most suitable model parameters. Model performance is evaluated using both standard metrics, such as the F1 score, and range-based metrics, which assess model effectiveness over time intervals. As established in the study, this is a more appropriate method for evaluating performance in industrial process data.

After completing the necessary steps to evaluate the performance of the investigated machine learning models, conclusions about the selection of the most suitable model for anomaly detection in water treatment system data were made.

Turinys

Lentelių sąrašas	8
Paveikslų sąrašas	9
Santrumpų ir terminų sąrašas	10
Įvadas.....	12
1. Literatūros analizė.....	13
1.1. Pramoninės valdymo sistemos	13
1.2. Anomalijos pramoninių procesų duomenyse ir jų tipai.....	14
1.2.1. Taškinės anomalijos	15
1.2.2. Kontekstinės anomalijos.....	15
1.2.3. Kolektyvinės anomalijos	16
1.3. Anomalijų aptikimo metodai pramoninių procesų duomenyse.....	17
1.3.1. K artimiausių kaimynų (KNN) algoritmas	19
1.3.2. Atraminų vektorių mašinos (SVM) algoritmas	20
1.3.3. K-vidurkių algoritmas	21
1.3.4. Auto-enkoderio algoritmas.....	22
1.3.5. Variacinio auto-enkoderio algoritmas	23
1.3.6. Atsitiktinio miško algoritmas	24
1.3.7. Izoliacijos miško algoritmas.....	25
1.3.8. USAD algoritmas	25
1.3.9. Generatyviniai priešpriešiniai tinklai	27
1.4. Anomalijų aptikimo metodų vertinimo metrikos	30
1.5. Pramoninių procesų duomenų rinkiniai.....	32
1.5.1. Saugaus vandens valymo sistema SWaT	32
1.5.2. Vandens paskirstymo sistema WADI.....	32
1.5.3. Vandens paskirstymo sistema BATADAL	32
1.5.4. „Tennessee Eastman“ procesas	33
2. Metodinė dalis.....	34
2.1. Parinktų duomenų rinkinių specifikacijos	34
2.1.1. SWaT.....	34
2.1.2. WADI.....	36
2.2. Duomenų rinkinių apdorojimas.....	38
2.3. Realizuoti anomalijų aptikimo modeliai	41
2.3.1. Izoliacijos miško modelis.....	41
2.3.2. K-vidurkių modelis.....	42
2.3.3. Auto-enkoderio modelis	43
2.3.4. Variacinio auto-enkoderio modelis	45
2.3.5. USAD modelis.....	47
3. Tyrimo rezultatai.....	49
3.1. Pirminio modelių apmokymo rezultatai	49
3.2. Optimizuotų modelių efektyvumo rezultatai	49
3.3. Modelių efektyvumo rezultatai, naudojant diapazono metriką „Range-F _β “	56
Išvados	60
Literatūros sąrašas	61
Priedai.....	64

1	priedas. Duomenų rinkinių apdorojimo programinis kodas	64
2	priedas. Izoliacijos miško modelio programinis kodas	65
3	priedas. K-vidurkių modelio programinis kodas	65
4	priedas. Auto-enkoderio, bei variacinio auto-enkoderio modelių programinis kodas	66
5	priedas. USAD modelio programinis kodas	71

Lentelių sąrašas

1 lentelė. Anomalijų aptikimo modelių vertinimo metrikų matematinės formulės	31
2 lentelė. Jutikliai ir pavaros SWaT duomenų bazėje	35
3 lentelė. Jutikliai ir pavaros WADI duomenų bazėje	37
4 lentelė. Duomenų rinkinių imtys, po apdorojimo	39
5 lentelė. Izoliacijos miško modelio išvestys ir jų reikšmės	42
6 lentelė. Izoliacijos miško modelio pradiniai parametrai	42
7 lentelė. K-vidurkių modelio pradiniai parametrai	43
8 lentelė. Auto-enkoderio modelio pradiniai parametrai	44
9 lentelė. Variacinio auto-enkoderio modelio pradiniai parametrai.....	46
10 lentelė. USAD enkoderio struktūra	47
11 lentelė. USAD dekodečių struktūra	47
12 lentelė. USAD modelio pradiniai parametrai	48
13 lentelė. Pirminio modelių apmokymo rezultatai SWaT duomenų rinkinyje.....	49
14 lentelė. Pirminio modelių apmokymo rezultatai WADI duomenų rinkinyje	49
15 lentelė. Izoliacijos miško modelio parametrų reikšmės optimizavimo metu	50
16 lentelė. K-vidurkių modelio parametrų reikšmės optimizavimo metu.....	52
17 lentelė. Auto-enkoderio modelio parametrų reikšmės optimizavimo metu	53
18 lentelė. Variacinio auto-enkoderio modelio parametrų reikšmės optimizavimo metu	54
19 lentelė. USAD modelio parametrų reikšmės optimizavimo metu	55
20 lentelė. Optimizuotų modelių įverčiai SWaT duomenų rinkinyje	56
21 lentelė. Optimizuotų modelių įverčiai WADI duomenų rinkinyje.....	56
22 lentelė. Pagal diapazono metrikas optimizuotų modelių įverčiai SWaT duomenų rinkinyje	58
23 lentelė. Pagal diapazono metrikas optimizuotų modelių įverčiai WADI duomenų rinkinyje	58
24 lentelė. Modelių, optimizuotų pagal F_β įvertį parametrai	59

Paveikslų sąrašas

1 pav. Tipinės pramoninės valdymo sistemos veikimas	13
2 pav. Paprastas dviejų dimensijų duomenų rinkinio anomalijų pavyzdys	14
3 pav. Taškinės anomalijos pavyzdys temperatūros kitimo duomenyse	15
4 pav. Kontekstinės anomalijos pavyzdys temperatūros kitimo duomenyse.....	16
5 pav. Kolektyvinės anomalijos pavyzdys temperatūros kitimo duomenyse	17
6 pav. Vienmatės laiko eilutės duomenų pavyzdys atvaizduotas grafiškai	18
7 pav. Daugiamatės laiko eilutės duomenų pavyzdys atvaizduotas grafiškai.....	18
8 pav. Pavyzdinių duomenų K artimiausių kaimynų algoritmo atstumo atvaizdavimas, kai $k=3$	20
9 pav. SVM klasifikavimo hiperplokštuma	20
10 pav. K-vidurkių algoritmo klasterizavimo pavyzdys trimis etapais (a-c).....	22
11 pav. Standartinio auto-enkoderio modelio architektūra	23
12 pav. Dvejetainio sprendimų medžio schema.....	24
13 pav. USAD algoritmo architektūra	26
14 pav. Standartinio GAN architektūra.....	27
15 pav. Sąlyginio GAN architektūra	28
16 pav. AnoGAN modelio, papildyto enkoderiu, architektūra	29
17 pav. MAD-GAN modelio architektūra	29
18 pav. PI-GAN modelio architektūra	30
19 pav. SWaT (angl. <i>secure water treatment system</i>) sistema ir ją sudarantys procesai	35
20 pav. Atakų SWaT sistemos procese sąrašas.....	36
21 pav. WADI (angl. <i>water distribution system</i>) sistema ir ją sudarantys procesai	37
22 pav. Atakų WADI sistemos procese sąrašas	38
23 pav. Grafiškai atvaizduotų SWAT duomenų iškarpa po apdorojimo	40
24 pav. Grafiškai atvaizduotų WADI duomenų iškarpa po apdorojimo.....	40
25 pav. Izoliacijos miško modelio architektūra	41
26 pav. Naudojamo auto-enkoderio modelio architektūra.....	44
27 pav. Naudojamo variacinio auto-enkoderio modelio architektūra	46
28 pav. Naudojamo USAD modelio architektūra	48
29 pav. Izoliacijos miško modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu.....	50
30 pav. Principinių komponentų analizės variacijos grafikas	51
31 pav. Alkūnės metodo grafikas.....	52
32 pav. K-vidurkių modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu	52
33 pav. Auto-enkoderio modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu.....	53
34 pav. Variacinio AE modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu	54
35 pav. USAD modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu	55
36 pav. Modelių Range-F β įverčio kitimas modelių optimizavimo metu SWaT duomenyse	57
37 pav. Modelių Range-F β įverčio kitimas modelių optimizavimo metu WADI duomenyse	57

Santrumpų ir terminų sąrašas

Santrumpos:

SCADA – pramoninė valdymo sistema, skirta procesų stebėjimui ir valdymui realiu laiku, duomenų kaupimui ir analizei, valdymo ir gamybos linijos vizualizacijai;

DSC – tai kompiuterizuota pramoninių procesų valdymo platforma, kurioje valdymo funkcijos yra ne sutelktos viename centriniame kompiuteryje, o paskirstytos per daugybę autonominių valdiklių, išdėstytų visoje gamykloje;

PLC – pramoninis kompiuteris, naudojamas automatizuoti įvairius gamybos procesus, valdyti įrenginius;

KNN – K artimiausių kaimynų algoritmas, mašininio mokymosi metodas, dažnai naudojamas klasifikavimui ir regresijai;

USAD – neprižiūrimo anomalijų aptikimo modelis (angl. *UnSupervised Anomaly Detection*) yra greitas ir stabilus algoritmas, skirtas aptikti anomalijas daugiamatėse laiko eilutėse;

SWaT – saugaus vandens valymo sistema (angl. *secure water treatment system*), bandymų stendas;

WADI - vandens paskirstymo sistema (angl. *water distribution system*), bandymų stendas;

SVM - atraminių vektorių mašina (angl. *Support Vector Machine*), mašininio mokymosi algoritmas, naudojamas duomenims klasifikuoti;

GAN – generatyvinis priešpriešinis tinklas (angl. *Generative Adversarial Network*), mašininio mokymosi modelis;

WGAN – *Wasseršteino* GAN (angl. *Wasserstein GAN*), populiari generatyvinių priešpriešinių tinklų (GAN) atmaina;

AnoGAN – anomalijų GAN (angl. *Anomaly GAN*), dirbtinio intelekto metodas, skirtas anomalijoms aptikti;

MAD-GAN – daugiamačių laiko eilučių anomalijų aptikimo naudojant GAN modelis, pažangus mašininio mokymosi metodas, naudojamas anomalijoms aptikti;

LSTM-RNN – tai specialus pasikartojančių neuroninių tinklų tipas, sukurtas informacijai analizuoti ir atsiminti per ilgus laiko ar teksto intervalus;

PI-GAN – fizikiniais dėsniais paremtas generatyvinis priešpriešinis tinklas;

MCC - *Matthewso* koreliacijos koeficientas (angl. *Matthews correlation coefficient*);

BATADAL - viešai prieinamas daugiamatis laiko eilučių duomenų rinkinys, imituojantis realios vidutinio dydžio vandentiekio sistemos veiklą;

PCA – principinė komponentų analizė, metodas, naudojamas didelės apimties duomenims supaprastinti;

WCSS - vidinių klasterių kvadratų suma (angl. *Within-Cluster Sum of Squares*), kuri naudojama duomenų analizėje ir mašiniame mokyme.

Terminai:

Pramoninės valdymo sistemos – sistemos, kurios naudoja kompiuterines, ryšių ir valdymo technologijas pramonės gamybos procesams valdyti, stebėti, prižiūrėti realiu laiku.

Žmogaus-mašinos sąsaja – yra vartotojo skydelis ar ekranas, jungiantis žmogų su mašina, sistema ar įrenginiu. Tai pramoninėje automatikoje naudojami ekranai, leidžiantys stebėti procesus, valdyti įrenginius, diagnozuoti problemas ir peržiūrėti duomenis.

Neparametrinis algoritmas – yra statistinis metodas arba mašininio mokymosi algoritmas, kuris nekelia griežtų prielaidų apie duomenų pasiskirstymo formą (pvz., kad duomenys turi atitikti normalųjį skirstinį).

Perceptronas – yra paprasčiausias dirbtinio neuroninio tinklo modelis, naudojamas mašiniame mokyme dvejetai klasifikavimui (sprendimui „taip“ arba „ne“).

Hiperplokštuma – yra sprendimų riba, naudojama klasifikuoti duomenims į skirtingas kategorijas.

Klasterinė analizė – viena matematinės analizės formų – objektų skirstymas pagal jų savybes (panašumą) į grupes, arba klasterius.

Auto-enkoderis – dirbtinio intelekto neuroninis tinklas, naudojamas neženkliniams duomenims mokytis, suspausti (kodavimui) ir rekonstruoti (dekodavimui).

Latentinis – reiškia slaptą, išoriškai nepasireiškiantį, esantį potencialioje būsenoje procesą ar savybę, kuri gali pasireikšti tam tikromis sąlygomis.

Generatyvinis modelis – dirbtinio intelekto modelių tipas, skirtas ne tik analizuoti ar klasifikuoti esamus duomenis, bet ir kurti visiškai naują, originalų turinį.

Desorberis – tai pramoninis įrenginys arba prietaisas, skirtas atvirkštiniam sorbcijos procesui (desorbcijai) atlikti.

Ivadas

Pramoninės valdymo sistemos dažnai naudojamos didelės svarbos objektuose, tokiuose kaip gamybos procesai, elektrinės, transporto tinklai ar vandens valymo ir skirstymo sistemos. Šios sistemos dažniausiai sudarytos iš programinių ir fizinių komponentų. Jos realiu laiku renka duomenis iš sistemos jutiklių, taip stebint proceso veiklą ir duoda atitinkamus signalus pavaroms, taip valdant procesą ir išlaikant jį stabilumo ribose. Tokių sistemų sutrikimai gali sukelti didelio masto nepageidaujamus padarinius, todėl itin svarbu, kad jos būtų atidžiai prižiūrimos ir veiktų be sutrikimų. Nepageidaujami išoriniai ar vidiniai veiksniai gali sukelti proceso anomalijas, kas eigoje procesą gali išvesti iš stabilumo ribų.

Išpopuliarėjus mašininio mokymo bei dirbtinio intelekto metodams, jie buvo pradėti naudoti ir anomalijų aptikimui. Norint tokius metodus naudoti anomalijų aptikimui pramoniniuose procesuose, reikia parinkti tinkamą mašininio mokymo modelį, nustatyti tinkamus modelio parametrus, turėti kokybišką duomenų rinkinį modelio apmokymui bei testavimui. Nors literatūroje siūloma nemažai įvairių sprendimų anomalijų aptikimui, tačiau nėra vienareikšmio sprendimo, kokie modeliai anomalijas aptinka efektyviausiai. Modelių apmokymui tyrėjai naudoja įvairius duomenų rinkinius, tačiau vienos labiausiai anomalijų aptikimo uždaviniams taikomų sistemų – vandens valymo bei skirstymo sistemos. Išsiaiškinus kokie mašininio mokymo modeliai efektyviausiai veikia tokio tipo sistemų anomalijų aptikimui leistų palengvinti anomalijų nustatymo metodų pasirinkimą inžinieriams, dirbantiems su vandens valymo sistemomis.

Darbo tikslas – ištirti bei palyginti skirtingų mašininio mokymo metodų anomalijų aptikimo efektyvumą vandens valymo sistemos duomenyse, naudojant pramoniniams procesams tinkamas įverčių metrikas.

Darbo uždaviniai:

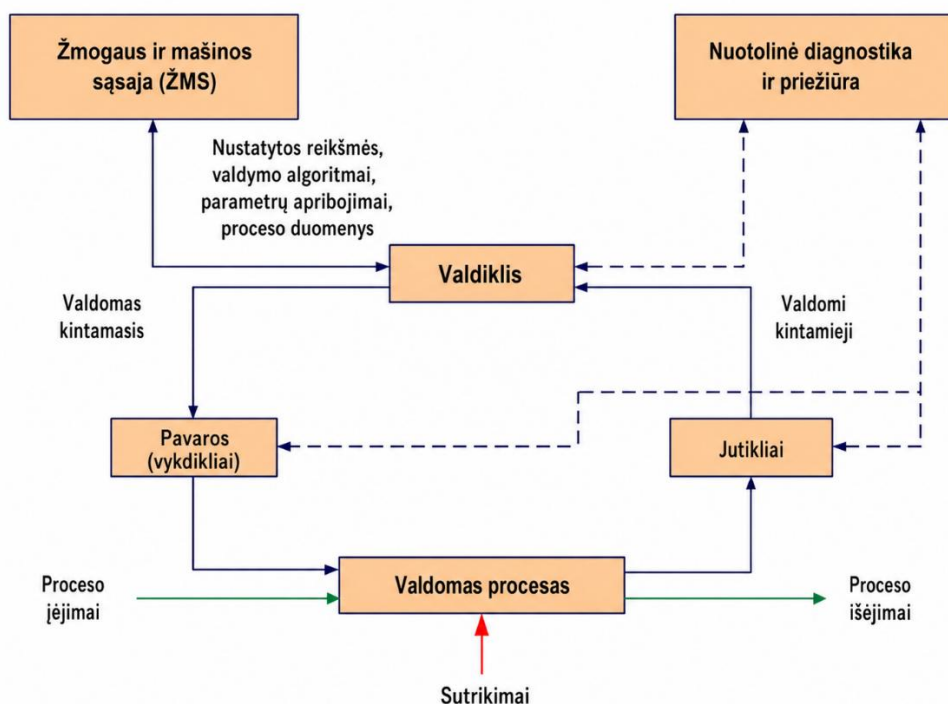
1. atlikti mokslinės literatūros analizę.
2. parinkti tyrimui tinkamus duomenų rinkinius modelių apmokymui, atitinkamai juos apdoroti.
3. realizuoti penkis anomalijų aptikimo modelius – izoliacinio miško modelį, K-vidurkių modelį, auto-enkoderio modelį, variacinio auto-enkoderio modelį, bei USAD modelį. Parinkti tinkamus modelių parametrus, modelius apmokyti.
4. ištirti anomalijų aptikimo modelių efektyvumą naudojant pramoniniams procesams tinkamas metrikas.
5. įvertinti gautus rezultatus bei pateikti išvadas.

1. Literatūros analizė

Šiame skyriuje apžvelgiami tyrimo įgyvendinimui reikalingi literatūros šaltiniai. Pateikiama informacija apie pramonines valdymo sistemas, anomalijų pramoninių procesų duomenyse tipus, anomalijų aptikimo metodus bei algoritmus, modelių vertinimo metrikas, anomalijų aptikimui naudojamų pramoninių procesų duomenų rinkinius.

1.1. Pramoninės valdymo sistemos

Pramoninės valdymo sistemos yra bendras terminas, kuris apibūdina keletą valdymo sistemų tipo, įskaitant prižiūravimo valdymo ir duomenų surinkimo (SCADA) sistemas, paskirstytojo valdymo sistemas (DCS) ir kitas valdymo sistemų konfigūracijas, tokias kaip programuojamus loginius valdiklius (PLC), dažnai sutinkamus pramoniniuose sektoriuose. Pramoninės valdymo sistemos susideda iš valdymo komponentų, kurie gali būti elektriniai, mechaniniai, hidrauliniai, pneumatiniai, kurie dirba visumoje, siekiant įvykdyti tam tikrą procesą. Procesu vadinama sistemos dalis, iš kurios turi išeiti tam tikra išvestis. Valdymo dalis sistemoje nusako norimą išvestį. Valdymas gali būti pilnai autonominis arba valdymo grandinėje turėti žmogų. Sistemos gali būti konfigūruojamos dirbti atviros grandinės, uždaros grandinės bei rankiniame režimuose. Atviros grandinės režime sistemos išvestis valdoma nustatytais parametrais. Uždaros grandinės režime išvesties reikšmė paveikia įvestį taip, kad būtų palaikoma nustatyta reikšmė. Rankiniame režime sistema pilnai valdoma žmogaus. Sistemos dalis, kuri atsakinga už nustatytų parametrų palaikymą yra vadinama valdikliu. Tipinė pramoninė valdymo sistema turi keletą valdymo grandinių, žmogaus-mašinos sąsajas, nuotolinės diagnostikos ir priežiūros įrankius, naudojant tinklų protokolus. Pramoninės valdymo sistemos valdo pramoninius procesus, įprastai naudojamus elektros, vandens ir nuotekų, naftos ir dujų, chemijos, transporto, medicinos, popieriaus, maisto ir gėrimų pramonėse [1].



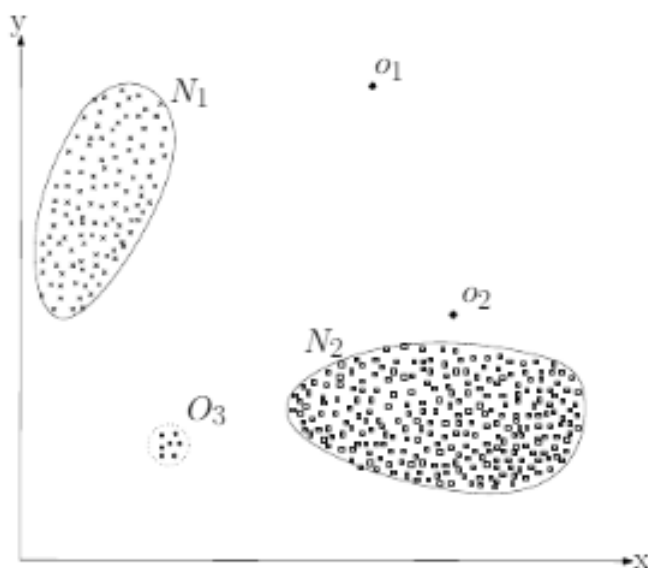
1 pav. Tipinės pramoninės valdymo sistemos veikimas [1]

Duomenų perdavimo tinklai pramoninėse valdymo sistemose turi didelę svarbą, kadangi jų pagalba galima nuotoliniu būdu stebėti bei valdyti procesus. Dėl šios priežasties, o taipogi ir augančių grėsmių kibernetiniam saugumui vis didinama sistemų apsauga ir patikimumas. Tokie incidentai, kaip 2015 metais Ukrainos elektros tinkluose įvykusi kibernetinė ataka, ar 2010 metais Irano atominėje elektrinėje įvykusios kibernetinės atakos parodo pramoninių valdymo sistemų tinklų pažeidžiamumą išoriniams veiksniams [2]. Tai parodo, kad pramoninių valdymo sistemų tinklų saugumą vis dar reikia stiprinti, ypač struktūriškai svarbiuose objektuose. Šie incidentai turėjo juntamą sociologinį bei ekonominį poveikį bei sutrikdė minėtų šalių elektros tiekimo tvarką.

Kenkėjiškos atakos gali būti ne tik išorinės, bet ir vidinės, kylančios iš pačios sistemos. Nemažai incidentų aptikimo įrankių turi stebėjimo sistemas, kurios aptinka parametrus, kurie nėra nustatytose ribose ar parametrus su iš anksto nustatytomis ribomis, iš kurių išėjus kyla tam tikra grėsmė. Tačiau ne visi pavojai ir pažeidžiamumai gali būti numatomi iš anksto [3]. Dėl šios priežasties svarbu sistemų saugai taikyti pažangius metodus, kurie padeda aptikti tiek išorines atakas tiek vidinius sistemos gedimus, klaidas ar netinkamą veikimą [2].

1.2. Anomalijos pramoninių procesų duomenyse ir jų tipai

Anomalijos yra duomenų struktūros, kurios nedera prie nustatyto normalaus duomenų kitimo. Paveikslėlyje žemiau pavaizduotos anomalijos paprastame dviejų dimensijų duomenų rinkinyje. Duomenys turi du normalius regionus N_1 ir N_2 , kadangi dauguma duomenų taškų yra šiuose regionuose. Taškai, kurie yra pakankamai nutolę nuo regionų, pavyzdžiui, taškai o_1 ir o_2 ir O_3 regiono taškai yra laikomi anomalijomis [4].



2 pav. Paprastas dviejų dimensijų duomenų rinkinio anomalijų pavyzdys [4]

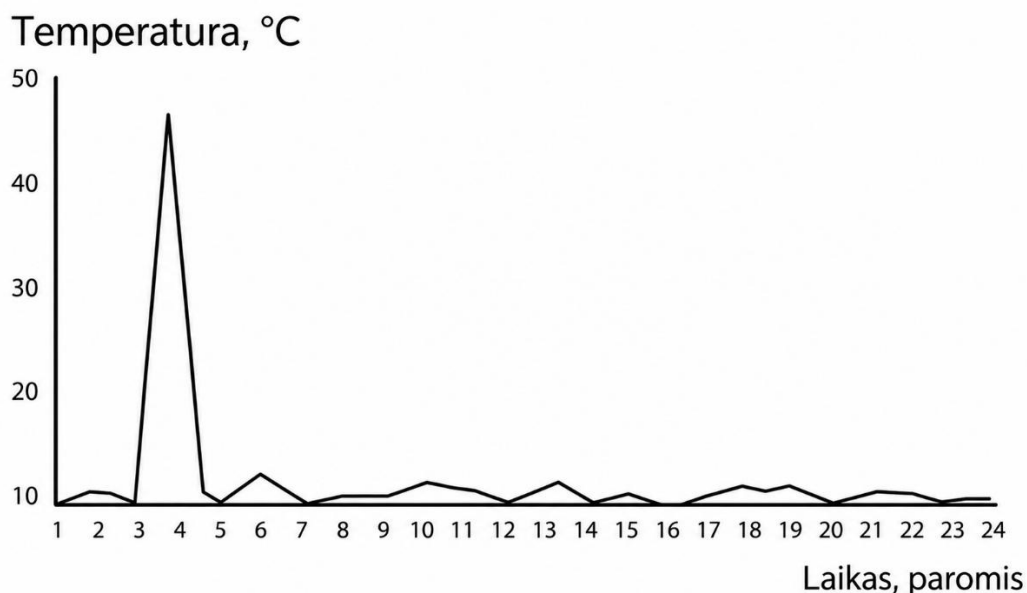
Pramoninėse valdymo sistemose anomalijos dažniausiai nurodo neįprastus jutiklių duomenis, pavarų veikimą ar sistemos būsenos modelius, neatitinkančius numatytų veikimo normų ar išorines atakas. Tokie duomenų nukrypimai dažnai gali signalizuoti apie sistemos įrangos gedimus, nestabilias proceso darbo sąlygas ar tam tikras kibernetines ar fizines atakas [5]. Priklausomai nuo duomenų nuokrypio pobūdžio ir konteksto, anomalijos pramoninių sistemų duomenyse paprastai skirstomos į tris pagrindines kategorijas: taškinės, kontekstinės ir kolektyvinės anomalijas.

1.2.1. Taškinės anomalijos

Taškinė anomalija nurodo duomenų tašką, kuris visumoje skiriasi nuo likusių duomenų. Šis anomalijų tipas taip pat literatūroje vadinamas kaip paprastoji anomalija ir yra lengviausiai aptinkama iš visų anomalijų tipų [4]. Tokias anomalijas įprastai galima nesunkiai aptikti paprastais metodais.

Antrame paveikslėlyje, taškai o_1 ir o_2 ir regiono O_3 taškai nepatenka nei į vieno iš normalių duomenų regionų ribas, todėl yra laikomi taškinėmis anomalijomis, kadangi skiriasi nuo normalių duomenų taškų.

Realus taškinės anomalijos pavyzdys būtų kreditinės kortelės apgavysčių aptikimas. Tarkime, kad duomenų rinkinys reiškia asmens kreditinės kortelės transakcijas. Darykime prielaidą, kad duomenis apibūdina tik vienas parametras – išleistų pinigų skaičius. Transakcija, kurios metu išleistas itin didelis pinigų skaičius lyginant su normaliomis asmens išleistų pinigų ribomis būtų taškinė anomalija [4].



3 pav. Taškinės anomalijos pavyzdys temperatūros kitimo duomenyse [6]

1.2.2. Kontekstinės anomalijos

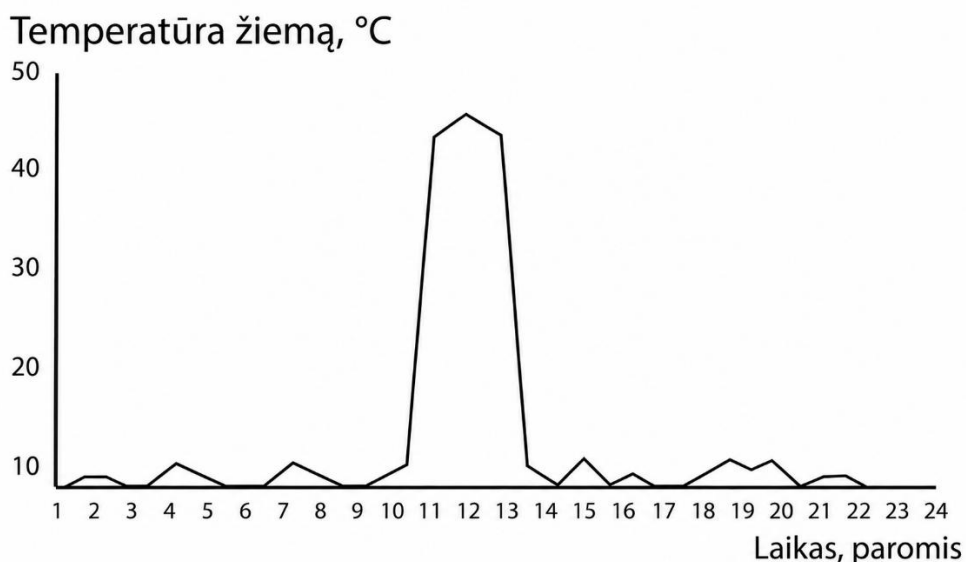
Kontekstinė anomalija yra taškas ar rinkinys duomenų taškų, kurie laikomi anomalijomis tam tikrame specifiniame kontekste ir galimai nebūtų laikomi anomalijomis kitokiame kontekste [6]. Kontekstas yra nustatomas duomenų rinkinio struktūros. Kiekvienas duomenų rinkinio taškas gali būti apibūdinamas dvejais parametru rinkiniais:

1. Kontekstiniai parametrai. Kontekstiniai parametrai naudojami apibūdinti kontekstą ar kaimynystę tam atvejui. Pavyzdžiui erdvinių duomenų rinkiniuose vietos plokštuma ir ilguma būtų kontekstiniai parametrai. Laiko eilučių duomenyse kontekstinis atributas yra laikas, kuris nurodo tam tikro atvejo poziciją visoje duomenų eilėje.
2. Elgesio parametrai. Elgesio parametrai apibūdina ne kontekstines to atvejo savybes. Pavyzdžiui, erdvinių duomenų rinkinyje kuris parodo vidutinį kritulių kiekį visame pasaulyje, kritulių kiekis bet kurioje pasaulio vietoje būtų elgesio parametras.

Anomalija nustatoma naudojant elgesio parametrų reikšmes tam tikrame kontekste. Vienu atveju duomenys gali būti kontekstinė anomalija tam tikrame kontekste, bet identiški duomenys gali būti laikomi normaliais duomenimis kitame kontekste. Tai yra svarbiausia savybė nustatant kontekstinius ir elgesio parametrus kontekstinių anomalijų aptikime [4].

Kontekstines anomalijas taip pat galima paaiškinti realiu pavyzdžiu iš kreditinių kortelių apgavysčių aptikimo srities. Kontekstinis parametras šiuo atveju galėtų būti pirkimo laikas. Tarkime asmuo dažniausiai kiekvieną savaitę išleidžia 100\$ išskyrus per Kalėdų savaitę, tada asmuo išleidžia 1000\$. Naujas 1000\$ vertės pirkinys liepos mėnesį būtų kontekstinė anomalija, kadangi tai neatitinka normalaus asmens elgesio laiko kontekste, nors tas pats pinigų kiekis išleistas Kalėdų savaitę būtų laikomas normaliu [4].

Pramoninių procesų duomenyse kontekstiniai parametrai galėtų būti paros laikas, gamybos etapas, sistemos veiklos režimas. Pavyzdžiui vandens skirstymo sistemoje didelio vandens srauto būtų galima tikėtis įprastomis vandens tiekimo valandomis, bet didelis vandens srautas atsiradęs per iš anksto suplanuotą techninės priežiūros laiką būtų kontekstinė anomalija. Tokioms anomalijoms nustatyti didelio dydžio duomenų rinkiniuose reikia pažangių metodų, tokių kaip mašininio mokymo modeliai.



4 pav. Kontekstinės anomalijos pavyzdys temperatūros kitimo duomenyse [6]

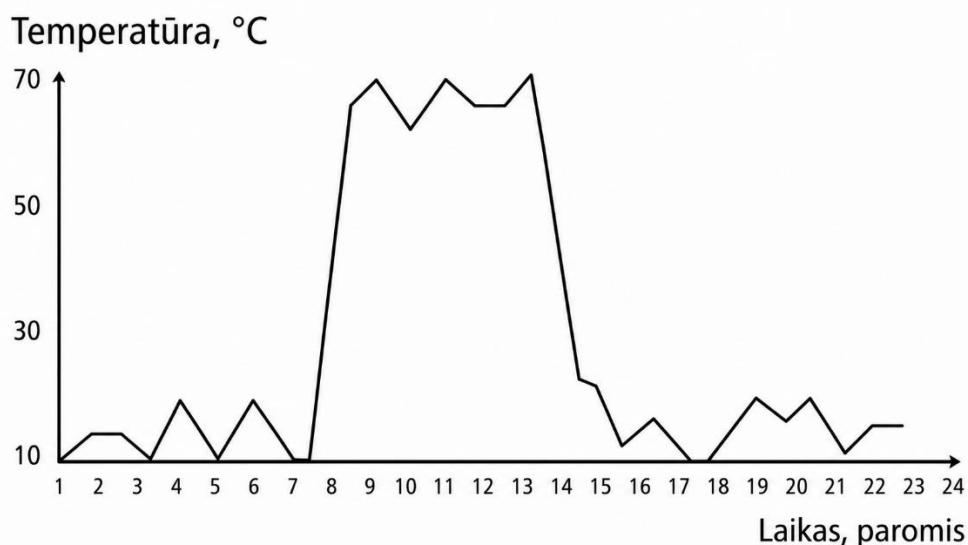
1.2.3. Kolektyvinės anomalijos

Kolektyvinė anomalija yra duomenų taškų, kurie yra laikomi anomalija grupėje, rinkinys, nors vienas tokių duomenų taškas nebūtų laikomas anomalija ne grupės ribose [6].

Pavyzdžiui jei žmogaus elektrokardiogramos išvestyje panaši žema vertė laikytųsi neįprastai ilgą laiką, tai būtų kolektyvinė anomalija, nors ta žema reikšmė iš savęs nėra anomalija, tačiau tos reikšmės grupė būtų anomalija.

Reikėtų paminėti, kad taškinės anomalijos gali atsirasti bet kokiame duomenų rinkinyje, o kolektyvinės anomalijos gali atsirasti tik tokiuose duomenų rinkiniuose, kur duomenų atvejai yra susiję. Kontekstinių anomalijų atsiradimas priklauso nuo kontekstinių parametrų duomenyse egzistavimo. Taškinė anomalija arba kolektyvinė anomalija taip pat gali būti kontekstinė anomalija

tam tikrame kontekste. Todėl taškinės anomalijos aptikimo uždavinys arba kolektyvinės anomalijos aptikimo uždavinys gali tapti kontekstinės anomalijos aptikimo užduotimi atsižvelgus į kontekstą [4].



5 pav. Kolektyvinės anomalijos pavyzdys temperatūros kitimo duomenyse [6]

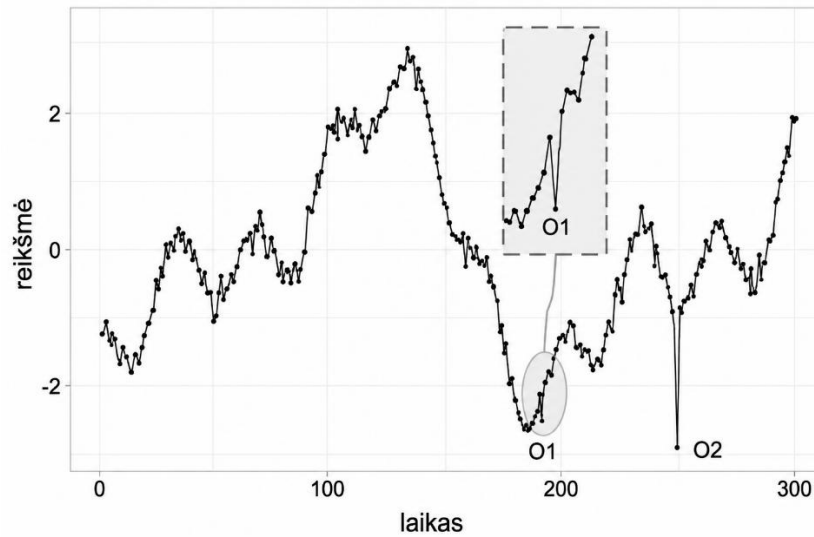
1.3. Anomalijų aptikimo metodai pramoninių procesų duomenyse

Anomalijų aptikimo metodai skiriasi, priklausomai nuo duomenų tipo. Pramoniniai procesai vyksta laiko erdvėje, todėl jų duomenys yra laiko eilutės. Laiko eilutės yra eilė duomenų taškų, nuosekliai sužymėtų pagal laiką. Dažniausias laiko eilučių tipas – seka stebėtų parametrų, sužymėtų einant laikui. Laiko eilutės dažnai skirstomos į vienmates (vienos dimensijos) ir daugiamates (kelių dimensijų) [7].

Kaip galima suprasti iš pavadinimo, vienmatė laiko eilutė yra duomenų seka, kurioje einant laikui kinta vienas kintamasis. Paprastas tokios laiko eilutės pavyzdys – drėgmės lygio pažymėjimas kiekvieną valandą per dieną. Laiko eilutė X , kurioje yra t laiko žymių gali būti užrašyta kaip tokia duomenų taškų seka:

$$X = (x_1, x_2, \dots, x_t), \quad (1)$$

čia x_i reiškia duomenis laiko žymoje $i \in T$, kai $T = \{1, 2, \dots, t\}$.



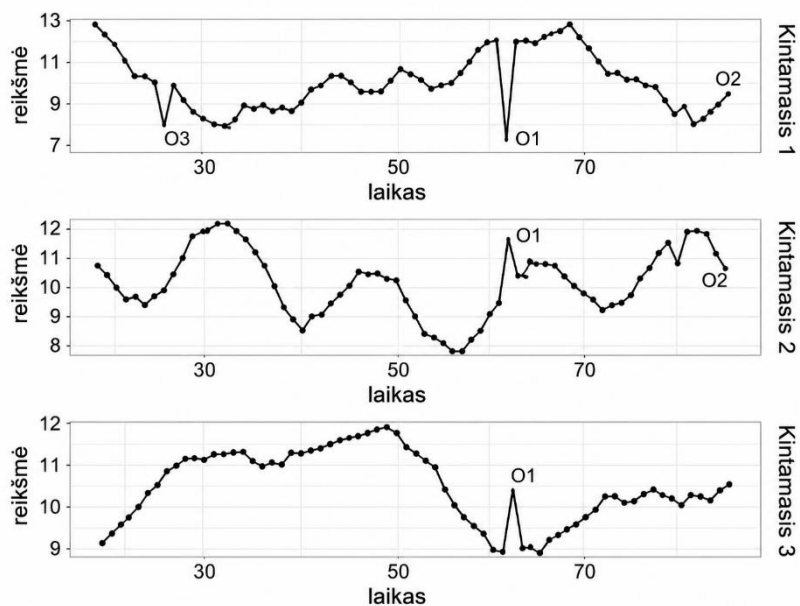
6 pav. Vienmatės laiko eilutės duomenų pavyzdys atvaizduotas grafiškai [8]

Skirtingai nei vienmatės laiko eilutės, daugiamatės laiko eilutės atvaizduoja keletą kintamųjų, kurie yra priklausomi nuo laiko ir kurių kiekvienas priklauso tiek nuo praeitų reikšmių, tiek nuo kitų kintamųjų (dimensijų), pagal jų priklausomybę [7]. Priklausomybės tarp skirtingų kintamųjų yra vadinamos erdvinėmis priklausomybėmis. Pavyzdžiui, oro slėgis ir temperatūra būtų žymimi kiekvieną dienos valandą kartu su vienmatės eilutės pavyzdyje minėtu drėgmės lygiu.

Daugiamatę laiko eilutę galima užrašyti kaip vektorių seką einant laikui, kur kiekvienas vektorius laiku i , X_i , sudaryta iš d dimensijų:

$$X = (X_1, X_2, \dots, X_t) = (x_1^1, x_1^2, \dots, x_1^d), (x_2^1, x_2^2, \dots, x_2^d), \dots, (x_t^1, x_t^2, \dots, x_t^d), \quad (2)$$

čia $X_i = (x_i^1, x_i^2, \dots, x_i^d)$ reiškia duomenų vektorių laiku i , kur kiekvienas x_i^j parodo stebėjimą laiku i dimensijoje j^{th} , kai $j = 1, 2, \dots, d$ ir d reiškia dimensijų skaičių.



7 pav. Daugiamatės laiko eilutės duomenų pavyzdys atvaizduotas grafiškai [8]

Vieni pirmųjų metodų, naudotų pramoninių procesų anomalijoms aptikti buvo statistiniai algoritmai. Aptinkant anomalijas statistiniais metodais, kuriamas statistinis modelis, apibūdinantis normalią duomenų elgseną. Tada gali būti atliekamas statistinis išvadų testas, siekiant nustatyti ar reikšmė priklauso sudarytam modeliui. Statistinis anomalijų aptikimas atliekamas naudojant keletą metodų, tokių kaip atstumu pagrįstus, parametrinius, ne parametrinius ir pusiau parametrinius metodus [9].

Kadangi pramoninių procesų, o ypač vandens valymo ir skirstymo sistemų duomenys beveik visada būna daugiamačiai, didelės apimties, sudėtingi, sudaryti iš daug jutiklių bei valdiklių, statistinių metodų anomalijų aptikimui tokiuose duomenyse naudojimas yra neefektyvus tiek laiko tiek tikslumo atžvilgiu. Todėl anomalijų aptikimui šiose sistemose naudojami mašininio mokymo bei dirbtinio intelekto metodai.

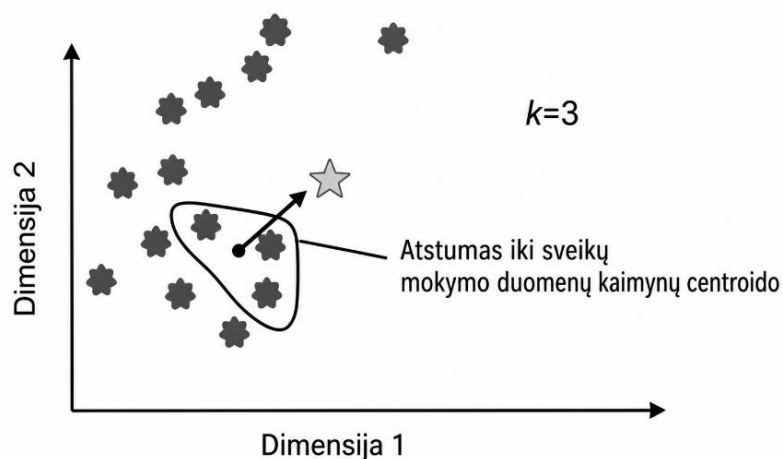
1.3.1. K artimiausių kaimynų (KNN) algoritmas

Vienas iš paprastesnių, tačiau nesudėtingiems duomenis kartais literatūroje taikomų mašininio mokymo metodų anomalijoms aptikti yra K artimiausių kaimynų algoritmas. KNN yra neparametrinis algoritmas, kuris negeba numatyti daugiamačio duomenų sąryšio ar pasiskirstymo. Šio algoritmo veikimas paremtas atstumo tarp duomenų taško ir jo artimiausio kaimyno apmokymo duomenų aibėje. Jei tiriamas taškas turi didesnę, nei nustatytą „atstumą“ nuo savo kaimynų, jis laikomas anomaliju. Šis atstumas skaičiuojamas naudojant *Euklido*, *Mahalanobio* ar kitus metodus [10].

Algoritmo veikimą galima apibūdinti penkiais pagrindiniais žingsniais:

1. Parenkami apmokymo duomenys, testavimo duomenys ir nustatoma K reikšmė.
2. Apmokymo duomenys modeliui pateikiami kaip įvesties duomenys, pažymimos apmokymo duomenų kategorijų klasės.
3. Pagal testavimo duomenis pasirinktu atstumo skaičiavimo metodu skaičiuojamas atstumas tarp kiekvieno testavimo duomenų taško ir apmokymo duomenų taško.
4. K apmokymo duomenų taškų, artimiausių kiekvienam testavimo duomenų taškui yra priskiriami artimiausiais kiekvieno testavimo duomenų taško kaimynais.
5. Nustatoma kiekvieno testavimo duomenų taško klasė, pagal dažniausiai pasikartojančias kategorijas prie K artimiausių kaimynų taškų [11].

Įprastai pramoninių procesų duomenyse šis algoritmas dažniausiai naudojamas taškinių anomalijų aptikimui.

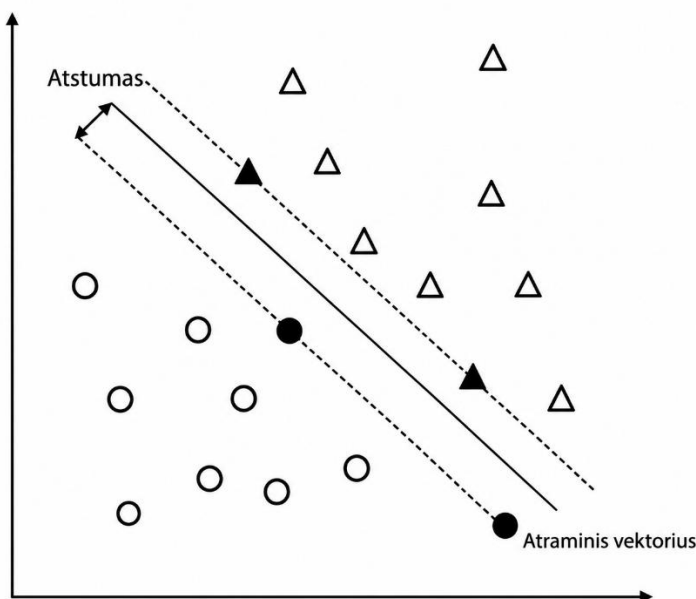


8 pav. Pavyzdinių duomenų K artimiausių kaimynų algoritmo atstumo atvaizdavimas, kai $k=3$ [10]

1.3.2. Atraminių vektorių mašinos (SVM) algoritmas

Atraminių vektorių mašinos įprastai yra skirtos naudoti dvejetainių skaitinių duomenų klasifikacijai, tačiau plačiai paplitę metodai, leidžiantys vietoj dvejetainės klasės taip pat naudoti įvairias kitas duomenų klases, kas leidžia šį algoritmą taikyti plačiau. Kitaip tariant SVM algoritmas yra panašus į perceptroninius neuroninius tinklus [12].

Šio algoritmo veikimo principas paremtas geriausio kelio arba sprendinio ribos sukūrimas, kuris gali padalinti n dimensijų erdvę į rinkinius, pagal kuriuos klasifikuojami nauji duomenų taškai. Kitaip sakant, SVM algoritmas bando sukurti unikalią atskiriamąją hiperplokštumą, kuri padaro didžiausią atskiriamąją ribą tarp dviejų klasių. Parametrų vektoriai, kurie šios atskiriamosios ribos ribose yra vadinami atraminiais vektoriais.



9 pav. SVM klasifikavimo hiperplokštuma [13]

Pagrindinė problema naudojant atraminių vektorių mašinas anomalijų aptikimui yra tai, kad šis mašininio mokymo modelis yra sukurtas pagrinde dvejoms klasėms, kurios turi būti atskiriamos sprendimų riba ir jos hiperplokštuma. Tačiau pramoninių procesų duomenyse, duomenys dažniausiai nėra sužymėti ir todėl modelis sudaromas su prielaida, kad visi duomenys priklauso normaliai klasei. Siekiant išspręsti šią problemą, daroma prielaida, kad branduoliu paremtas, transformuotas atvaizdas priklauso anomalijų klasei. Todėl klasifikavimo tarp normalių duomenų ir anomalijų kokybė, tam tikruose duomenyse bei atitinkama transformacija, stipriai priklauso nuo šios prielaidos teisingumo [14].

1.3.3. K-vidurkių algoritmas

K-vidurkių (angl. *K-means*) algoritmas yra laikomas vienu populiariausių neprižiūrimų mašininio mokymo metodų klasterinei analizei. Šio algoritmo apmokymui nereikia naudoti sužymėtų duomenų, todėl jis plačiai naudojamas įvairiuose srityse, tokiose kaip pramonė, transportas, medicina ir kitose srityse.

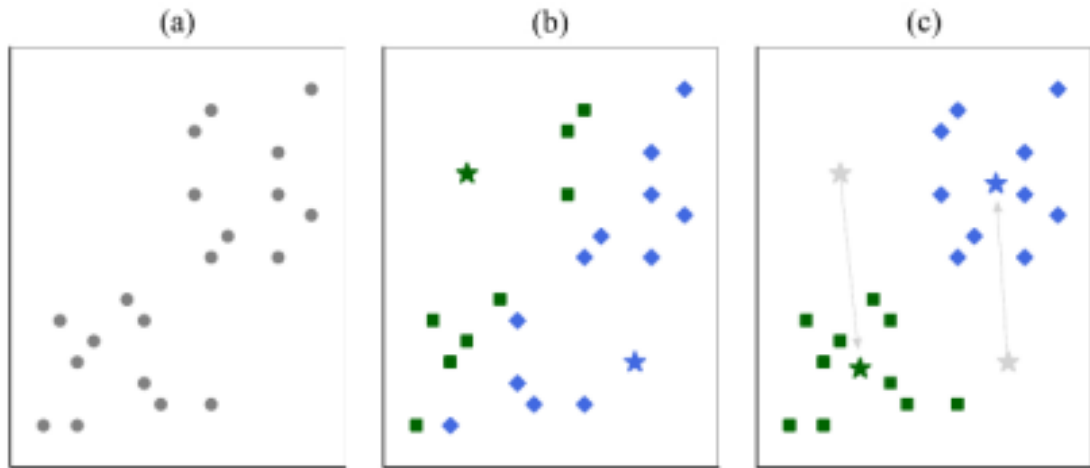
Klasterizavimas yra duomenų padalijimas į keletą skirtingų rinkinių. Kai klasterizavimas naudojamas duomenų klasifikavimui, Euklidinis atstumas ar kitas panašus skaičiavimo metodas yra naudojamas kaip matavimo kriterijus duomenų panašumui. Kuo didesnis duomenų panašumas, tuo mažesnis Euklidinis atstumas tarp duomenų. K-vidurkių klasterizavimo metodas veikia bandydamas suformuoti klasterių skaičių, kuris yra nurodomas vartotojo, kaip modelio parametras. Kiekvienas klasteris eina per visų klasterio elementų vidurkio reikšmę, vadinamą centroidu [15].

Tarkime, kad $X = \{x_1, \dots, x_n\}$ yra d dimensijų Euklidinės erdvės R^d duomenų rinkinys. Tuomet $A = \{a_1, \dots, a_c\}$ reiškia c klasterių centrus. Naudojant $Z = [Z_{ik}]_{n \times c}$ galima nustatyti klasterių priklausomybę, kur dvejetainis kintamasis Z_{ik} parodo ar tiriamas duomenų taškas x_i priklauso k klasteriui. K-vidurkių algoritmas siekia minimalizuoti tikslo funkciją $J(Z, A)$ perskaičiuojant klasterių centrus ir priklausomybes pagal dvi sąlygas, aprašytas formulėse žemiau:

$$a_k = \sum_{i=1}^n Z_{ik} x_{ij} / \sum_{i=1}^n Z_{ik}, \quad (3)$$

$$Z_{ik} = \begin{cases} 1, & \text{if } ||x_i - a_k||^2 = \min (||x_i - a_k||^2), 1 \leq k \leq c. \\ else & 0 \end{cases} \quad (4)$$

K-vidurkių algoritmo klasterizavimą galima atvaizduoti ir schematiškai. Algoritmas paveikslėlyje žemiau atvaizduojamas trimis žingsniais nuo a iki c, laikoma, kad modelyje nustatytas dviejų klasterių parametras. Veikimo pradžioje, schemas a dalyje duomenys nėra niekaip suskirstyti į klases. K-vidurkių algoritmas numato du taškus kaip būsimų klasių centrus, kurie parenkami atsitiktinai. Paveikslėlyje žali ir mėlyni žymekliai žymi modelio parinktus centrus b dalyje. Naudojant atstumo matavimą visų stebėjimų klasę galima nustatyti pagal atstumą iki naujų klasterių centrų. Toliau algoritmas perskaičiuoja centrus iki kol susidaro akivaizdūs duomenų klasteriai. Tai reiškia, kad modelis baigė klasterizavimą [16].



10 pav. K-vidurkių algoritmo klasterizavimo pavyzdys trimis etapais (a-c) [16]

1.3.4. Auto-enkoderio algoritmas

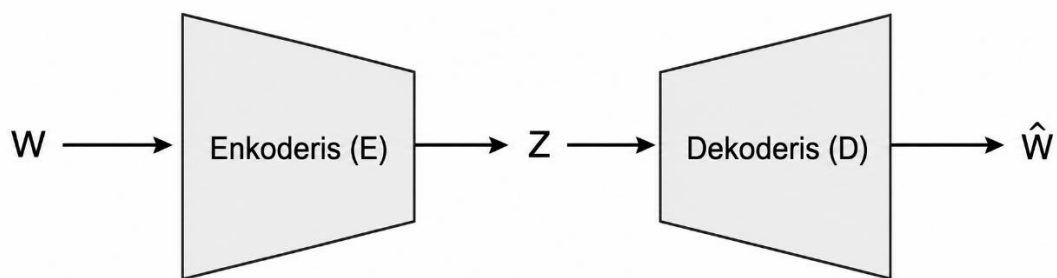
Auto-enkoderis yra neprižiūrimas dirbtinis neuroninis tinklas, kombinuojantis enkoderį E ir dekoderį D . Enkoderio dalis ima įvestis X ir priskiria jas latentinių kintamųjų Z rinkiniui, o dekoderis priskiria latentinius kintamuosius Z atgal į įvesčių erdvę kaip rekonstrukcijos R . Skirtumas tarp originalaus įvesties vektoriaus X ir rekonstrukcijos R yra vadinamas rekonstrukcijos paklaida. Todėl modelio apmokymo tikslas yra minimizuoti šią paklaidą. Tai galima išreikšti taip:

$$\partial_{AE} = ||X - AE(X)||_2, \quad (5)$$

čia $AE(X) = D(Z)$, $Z = E(X)$ ir $||\cdot||_2$ žymi $L2$ normą [17].

Klasikinis auto-enkoderis yra visiškai sujungtas neuroninis tinklas su tiesioginiu ryšiu, kuriame neuronų skaičius įvesties sluoksnyje ir išvesties sluoksnyje yra toks pat ir paslėptuosiuose sluoksniuose yra gerokai mažiau neuronų nei įvesties ir išvesties sluoksniuose. Auto-enkoderio algoritmas bando sukurti išvestį, kuri rekonstruoja jos įvestį su pašalintais triukšmais. Kadangi paslėptieji sluoksniai turi gerokai mažiau neuronų, siekiant atkurti įvestį kuo tiksliau, paslėptųjų sluoksnių svoriai užfiksuoja tik labiausiai išsiskiriančius originalių įvesties duomenų bruožus ir ignoruoja detalizuotą įvesties duomenų specifiką. Todėl normalius duomenys yra žymiai lengviau atkurti nei anomalijas [18].

Klasikiniai auto-enkoderiai dažniausiai yra naudojami ne nuosekliems duomenims. Siekiant atlikti anomalijų aptikimo uždavinį nuosekliuose duomenyse, tokiuose kaip laiko eilutės, taikomi papildomi metodai kartu su auto-enkoderio architektūra. Vieni populiariausių ir dažniausiai literatūroje anomalijų aptikimui pramoninių procesų duomenyse naudojamų priemonių auto-enkoderių papildymui – rekurentiniai neuroniniai tinklai, transformatorių sluoksniai, konvoliuciniai neuroniniai tinklai.



11 pav. Standartinio auto-enkoderio modelio architektūra [19]

1.3.5. Variacinio auto-enkoderio algoritmas

Variacinis auto-enkoderis yra neprižiūrimas giliojo mokymosi generatyvinis modelis, kuris gali modeliuoti apmokymo duomenų pasiskirstymą. Šis algoritmas paremtas *Bayeso* statistika ir yra sudarytas iš enkoderio, latentinio skirstinio, dekoderio. Jo veikimo principas paprastu, tokiu kaip Gauso, skirstiniu su žinomais parametrais ir charakteristikomis, kuris teoriškai gali tikti bet kokiam skirstiniui kombinuojant su neuroniniais tinklais [20].

Modelio darbą galima apibūdinti tokia seka: įvestis X perduodama enkoderiui ir gauna latentinės erdvės skirstinio parametrus. Latentinis kintamasis z gaunamas bandant atkartoti esamą skirstinį, tuomet z naudojamas generuoti rekonstruotus duomenis naudojant dekoderį. Po išvedimo, supaprastinimo ir variacinio išvedimo, variacinio auto-enkoderio nuostolių lygtis gali būti išreikšta taip:

$$L_{VAE}(E, D) = ||X - X_{recon}||_2 + \frac{1}{2}||z||_2 + \frac{1}{2} \sum_{i=1}^{zdim} [(\mu_i^2 + \sigma_i^2) - 1 - \log(\sigma_i^2)]. \quad (6)$$

Algoritmo eigoje yra žingsnis, kurio metu bandomas atkartoti ar, kitaip tariant, perparametrizuoti skirstinys. Dėl Gauso skirstinio superpozicijos duomenys po perskaičiavimo yra vienodi su atkartotu skirstiniu, pagal tam tikrus pasiskirstymo parametrus. Atkartoti duomenys, gauti iš standartinio normalaus skirstinio gali būti laikomi konstanta, o šios konstantos skaičiavimo procesas yra diferencijuojamas, todėl modelis gali atlikti atgalinį sklidimą (angl. *back propagation*) [20].

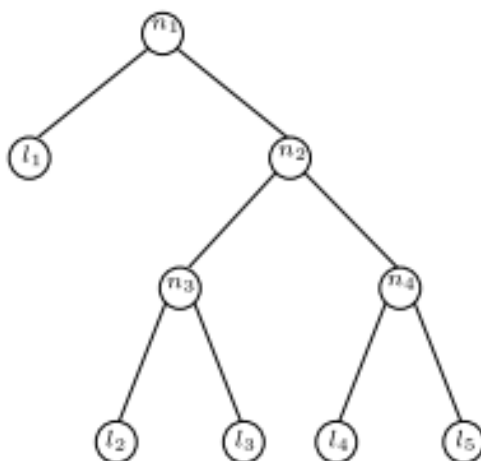
$$\varepsilon \sim N(0,1) \quad z = \mu + \varepsilon * \sigma \rightarrow z \sim N(\mu, \sigma). \quad (7)$$

Pagrindinis skirtumas tarp variacinio ir standartinio auto-enkoderio yra tai, kad variacinis auto-enkoderis yra stochastinis generatyvinis modelis, kuris gali pateikti sukalibruotas tikimybes, o auto-enkoderis yra deterministinis diskriminacinis modelis, neturintis tikimybinių pagrindo. Tai patvirtina variacinio auto-enkoderio modeliuojami skirstinio parametrai, kaip nurodyta aukščiau.

1.3.6. Atsitiktinio miško algoritmas

Atsitiktinio miško algoritmas yra procedūrinis mašininio mokymo algoritmas, paremtas kolektyvinių mokymusi, kuris yra lankstus ir gana lengvai naudojamas. Šis metodas kombinuoja kelis sprendimų medžius, kas išauga į sprendimų medžių „mišką“ ir duoda geresnius rezultatus.

Prieš gilinantis į atsitiktinio miško algoritmą svarbu suprasti sprendimų medžio veikimo principą. Standartinėje formoje, sprendimų medžiai yra sukurti vektoriniais duomenimis ir yra ne parametriniai modeliai, kurie rekursyviai dalina funkcijų erdvę į vis labiau homogeniškus poaibius, pagrįstus požymių reikšmėmis. Kiekvienas vidinis medžio mazgas atitinka sprendimą, pagrįsta tam tikru požymiu ar slenkstine verte. Kuriant sprendimų medžius, apmokymo duomenys turi būti padalinami į pogrupius, kiekviename mazge parenkant požymius ir slenkstines vertes, geriausiai padalinančius duomenis pagal „priemaišų“ kriterijų. Pagrindiniai sprendimų medžių pliusai yra jų gebėjimas modeliuoti sudėtingas, ne linijines elgsenas tarp požymių, minimalus išankstinio duomenų apdorojimo reikalavimas. Jie gali apdoroti tiek skaitmeninius, tiek kategorinius duomenis, trūkstamas reikšmes ir skirtingo mastelio duomenis [21].



12 pav. Dvejetainio sprendimų medžio schema [21]

Atsitiktinio miško algoritmas generuoja daugybę sprendimų medžių, apmokant kiekvieną medį pagal duomenų pradinės įkelties (angl. *bootstrap*) parametro imtį. Algoritmas taip pat kiekvieno medžio kūrimo metu, kiekvienoje galimoje medžio atskyrimo vietoje, atskyrimui naudoja tik atsitiktinę požymių dalį, o ne visus turimus požymius. Šis procesas suteikia miškui dar daugiau įvairovės, kadangi sumažina atskirų medžių tarpusavio koreliaciją [21].

Taigi duomenų rinkiniui su N stebėjimų ir f požymių, atsitiktinio miško algoritmas sukuria T medžių, kurių kiekvienas yra apmokomas naudojant duomenų pradinės įkelties imtį. Kiekviename medžio padalijime atsitiktine tvarka parenkamas m požymių pogrupis ($m < f$) ir iš jo randamas geriausias padalijimas. Klasifikavimo uždaviniuose galutinis spėjimas nustatomas balsų dauguma per visus medžius, o regresijos uždaviniuose tai yra visų spėjimų vidurkis [21].

Atsitiktinio miško privalumai, lyginant su standartiniais sprendimų medžiais:

1. Geresnė apibendrinimo kokybė. Apskaičiuojamas vidurkis iš didelio skaičiaus tarpusavyje nesusijusių medžių, taip mažinant pernelyg didelį prisitaikymą, taip gaunant geresnius rezultatus nei iš pavienių medžių.
2. Vidinis apibendrinimo paklaidos įvertinimas. Naudodamas OOB (angl. *out-of-bag*) imtis, neįtrauktas į tam tikro medžio pradinės įkelties imtį, algoritmas gali apskaičiuoti paklaidą nenaudojant atskiro validacijos duomenų rinkinio.
3. Atsitiktinio miško medžiai gali būti apmokomi nepriklausomai ir lygiagrečiai, todėl atsitiktinio miško modelis tinka didelio masto duomenų mokymuisi.

1.3.7. Izoliacijos miško algoritmas

Atsitiktinio miško algoritmai pagrįsti yra sukurti prižiūrimiems klasifikavimo ir regresijos uždaviniams, tačiau sprendimų medžiais grįsti modeliai taip pat yra efektyvūs neprižiūrimuose anomalijų aptikimo uždaviniuose. Izoliacijos miško algoritmas yra vienas iš algoritmų, specialiai sukurtų anomalijų aptikimui didelių dimensijų duomenyse. Skirtingai nei tradiciniai anomalijų aptikimo metodai, izoliacijos miško algoritmas veikia pasinaudodamas pastebėjimu, kad anomalijos duomenyse dažniausiai yra retos ir skiriasi nuo įprastų duomenų, todėl jas iš esmės yra lengviau izoliuoti suskirstytose erdvėse [21].

Šis algoritmas kuria medžių ansamblį apmokymo duomenims. Anomalijos yra tie duomenų rinkinio taškai, kurie turi trumpus vidutinius kelius iki minėtų medžių. Siekiant sudaryti izoliacijos miško modelį, visų pirma reikia parinkti izoliacinių medžių skaičių, kurių kiekvienas yra kuriamas tokiais žingsniais:

1. Atsitiktinai atrenkama n pavyzdžių iš apmokymo duomenų rinkinio.
2. Atsitiktinai atrenkama savybė, pagal kurią vykdomas dalijimas.
3. Atsitiktinai parenkama dalijimo reikšmė iš vienodo pasiskirstymo, apimančio antrame žingsnyje pasirinktos savybės minimaliąją ir maksimaliąją vertes.

Aukščiau aprašyti antras ir trečias žingsniai kartojami rekursyviai, kol visi n pavyzdžiai iš atsitiktinai atrinktos imties izoliuojami izoliacijos medžio lapų mazguose. Tokiu būdu anomalijoms izoliuoti lapiniuose mazguose reikės mažiau atsitiktinių padalijimų, o tai lems trumpesnį numatomą kelio ilgį nuo šaknies mazgo iki lapų mazgo [22].

Tarkime, kad duomenų rinkinyje yra n pavyzdžių ir $h(x)$ yra pavyzdžio x kelio ilgis. Kelio ilgio $h(x)$ reikšmė normalizuojama naudojant vidutinį kelio ilgį $c(n)$. Kadangi izoliaciniai medžiai turi tokią pat struktūrą kaip dvejetainės paieškos medžiai, $c(n)$ reikšmė yra tokia pati kaip nesėkmingos paieškos dvejetainės paieškos medyje kelio ilgis. Ši reikšmė apskaičiuojama naudojant lygtį, kurioje $H(i)$ yra harmoninis skaičius, kuris gali būti apskaičiuojamas pagal $\ln(i) + 0.5772156649$ (Eulerio konstanta):

$$c(n) = 2H(n - 1) - \left(\frac{2(n - 1)}{n} \right), \quad (8)$$

duomenų rinkiniui, kuris turi n pavyzdžių, anomalijų įvertiai kiekvienam duomenų taškui x yra apskaičiuojami šia formule:

$$s(x, n) = 2^{-\left(\frac{E(h(x))}{c(n)} \right)}, \quad (9)$$

čia $E(h(x))$ yra $h(x)$ vidurkis iš izoliacinių medžių. Kuo anomalijos įvertis artimesnis 1, tuo didesnė tikimybė, kad tas duomenų taškas yra anomalija. Taipogi, anomalijos įvertis artimas 0 reiškia, kad duomenys yra normalūs [22].

1.3.8. USAD algoritmas

USAD trumpinys reiškia neprižiūrimą anomalijų aptikimą (angl. *UnSupervised Anomaly Detection*). Šis algoritmas yra sudarytas iš auto-enkoderio architektūros, veikiančios dviejų etapų priešpriešinio apmokymo sistemoje. Viena vertus, tokia sistema leidžia apeiti auto-enkoderių modeliams būdingus apribojimus apmokant modelį kuris gali identifikuoti, net kuomet įvesties duomenys neturi anomalijų

ir taip atlikti kokybišką rekonstrukciją. Kita vertus auto-enkoderio architektūra leidžia užtikrinti stabilumą priešpriešinio apmokymo metu, kas pašalina generatyviniuose modeliuose pasitaikančią nesutapimo ir žlugimo problemą.

USAD modelis yra sudarytas iš trijų elementų: enkoderio tinklo E ir dviejų dekoderių tinklų D_1 ir D_2 . Kaip parodyta paveikslėlyje žemiau, šie trys elementai yra sujungti į architektūrą, sudarytą iš dviejų enkoderių AE_1 ir AE_2 , kurie turi tą patį enkoderio tinklą:

$$AE_1(W) = D_1(E(W)), AE_2(W) = D_2(E(W)). \quad (10)$$

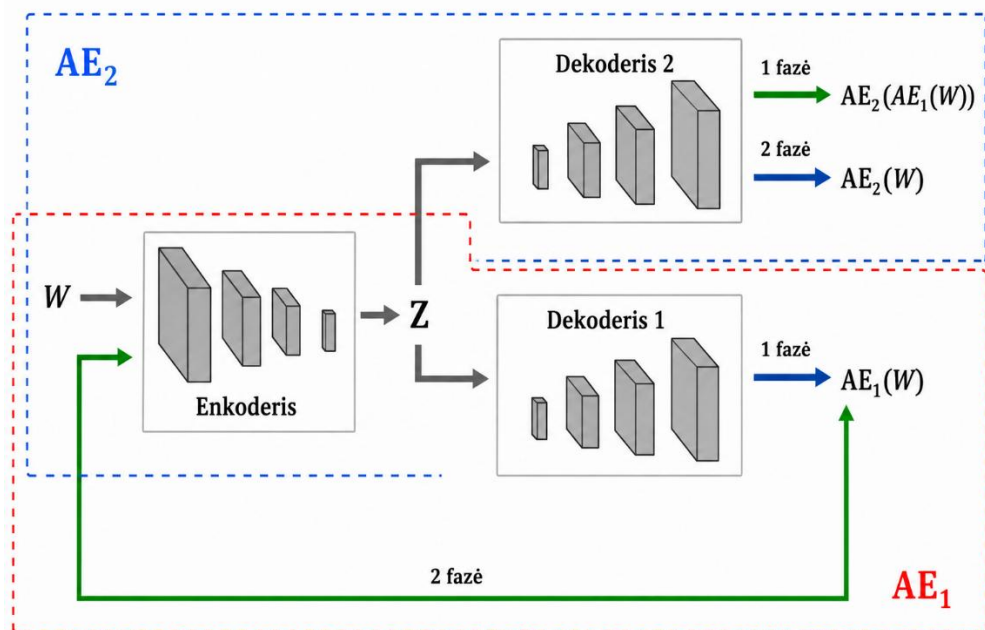
Architektūra, kuri užrašyta dešimtoje formulėje yra apmokoma dvejomis fazėmis. Pirmą du auto-enkoderiai apmokomi rekonstruoti normalių įėjimų langus W . Tuomet šie du auto-enkoderiai yra mokomi priešpriešiniu būdu, kur AE_1 siekia „apgauti“ AE_2 , o AE_2 siekia išmokyti kada duomenys yra tikri, o kada rekonstruoti [23].

Pirmajame apmokymo etape siekiama kiekvieną auto-enkoderį apmokyti atkurti įvesties duomenis. Įvesties duomenys W yra suspaudžiami enkoderio E į latentinę erdvę Z ir tada rekonstruojami kiekvieno enkoderio:

$$\delta_{AE_1} = ||W - AE_1(W)||_2, \quad (11)$$

$$\delta_{AE_2} = ||W - AE_2(W)||_2. \quad (12)$$

Antrame apmokymo etape apmokymo tikslas yra apmokyti AE_2 atskirti tikrus duomenis nuo duomenų kuriamų AE_1 ir apmokyti AE_1 apgauti AE_2 . Duomenys, kuriami AE_1 suspaudžiami enkoderio į Z erdvę ir tada rekonstruojami AE_2 . Naudojant priešpriešinę apmokymo konfigūraciją, AE_1 tikslas yra minimizuoti skirtumą tarp W ir AE_2 išėjimo. AE_2 tikslas yra šį skirtumą maksimizuoti. Pirmasis auto-enkoderis mokosi pagal tai ar sėkmingai geba apgauti antrąjį auto-enkoderį, o antrasis auto-enkoderis bando atskirti pirmojo auto-enkoderio kuriamus duomenis nuo tikrų duomenų.

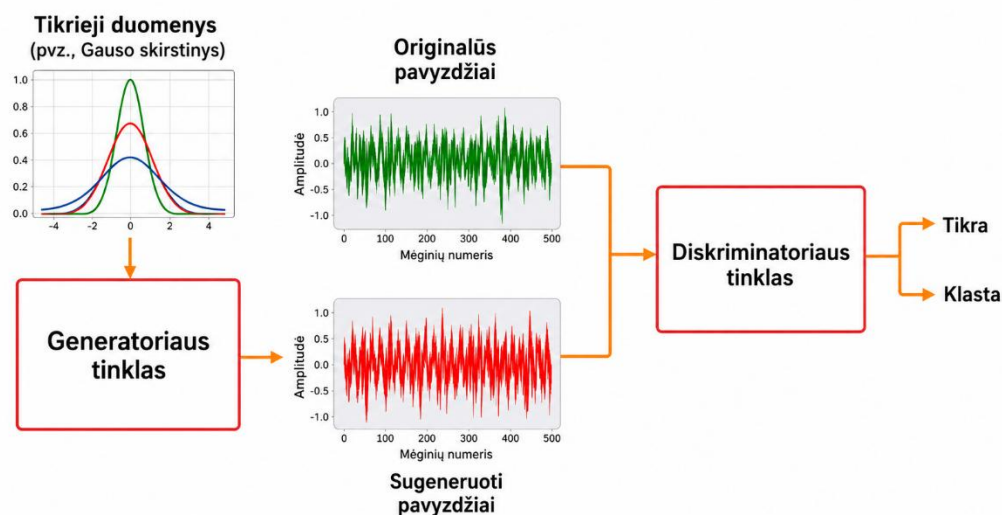


13 pav. USAD algoritmo architektūra [23]

1.3.9. Generatyviniai priešpriešiniai tinklai

Generatyviniai priešpriešiniai tinklai (GAN) - tai generatyvinių mašininio mokymo modelių klasė. Šie tinklai mokosi kurti duomenis, itin panašius į įvesties duomenų pasiskirstymą. Generatyviniai priešpriešiniai tinklai ypač tinka užduotims, kurioms reikia tikslumo ir papildomų duomenų generavimo. Šie modeliai pradėjo plisti ir anomalijų aptikimo pramoniniuose procesuose srityje, o iš to kilo daug šių tinklų variacijų. Jos apžvelgiamos žemiau.

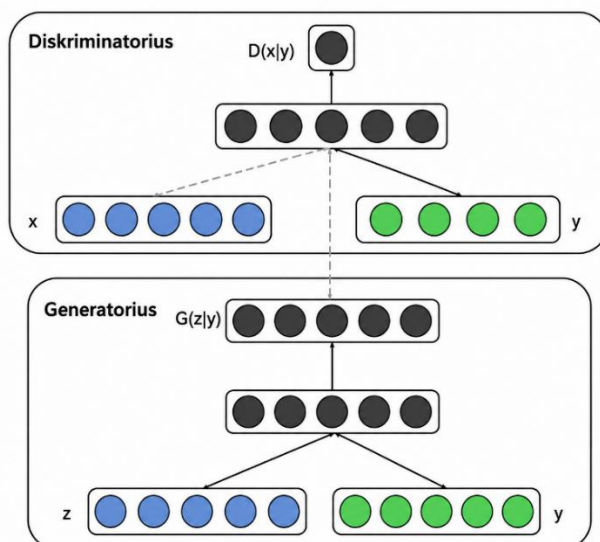
Standartinį generatyvinių priešpriešinių tinklų (GAN) modelį sudaro du pagrindiniai komponentai – generatoriaus tinklas ir diskriminatoriaus tinklas. Generatoriaus tinklas, pasinaudodamas atsitiktiniu triukšmo vektoriumi, paimtu iš tam tikro, pavyzdžiui, Gauso skirstinio, sukuria dirbtinius duomenis. Diskriminatorius apdoroja sugeneruotus duomenis ir išveda tikimybę, parodančią kiek sugeneruoti duomenys yra panašūs į tikrus duomenis [24]. Standartinio GAN architektūra pavaizduota nuotraukoje žemiau.



14 pav. Standartinio GAN architektūra [24]

Diskriminatorius yra mokomas maksimizuoti teisingo tikrų ir netikrų duomenų atpažinimo tikimybę, o generatorius bando minimizuoti diskriminatoriaus gebėjimą duomenis atskirti. Modelio apmokymas tęsiamas iki kol generatorius pradeda kurti išvestis, kurias diskriminatorius nesugeba atskirti nuo realių duomenų. Šių modelių apmokymas yra gan sudėtingas dėl tam tikrų, generatyviniams tinklams būdingų problemų, tokių kaip nesutapimas ir modelio griūtis. Šioms problemoms spręsti bei siekiant pagerinti priešpriešinių generatyvinių tinklų pritaikymą siauresnėms sritims, kuriamos GAN modelių variacijos.

Sąlyginiai GAN (angl. *conditional GAN*) modeliui pateikia pagalbinę informaciją, tokią kaip klasės, aplinkos parametrai, ribiniai laukai. Tokios papildomos sąlygos, perduodamos diskriminatoriui ir generatoriui, leidžia tiksliau sintetinti generuojamus rezultatus. Generatoriuje įėjimo triukšmas ir sąlyga yra apjungiami paslėptame atvaizde, o priešpriešinio mokymo architektūra suteikia didelę laisvę sprendžiant, kaip šis paslėptas atvaizdas turi būti sudarytas [25].



15 pav. Sąlyginio GAN architektūra [25]

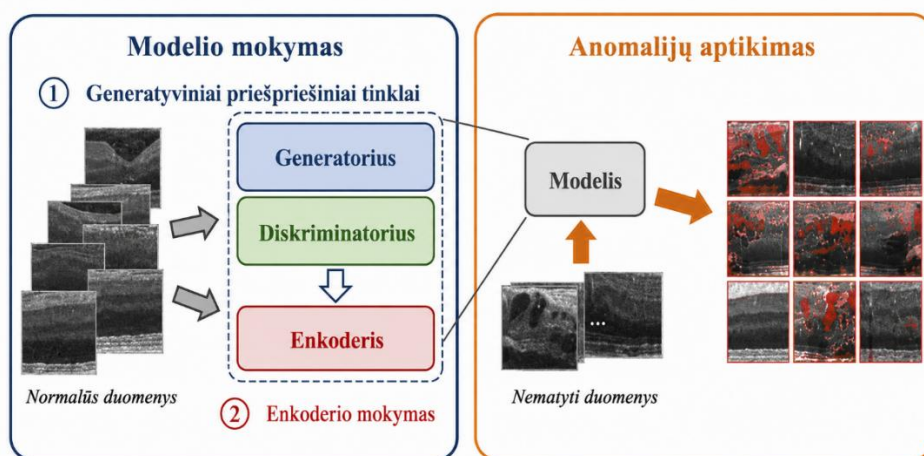
Dar vienas priešpriešinių generatyvinių tinklų tipas yra *Wasseršteino* GAN (WGAN). Šis modelis naudoja *Wasseršteino* atstumo sampratą, siekiant išmatuoti atstumą tarp realaus ir sugeneruotų duomenų skirstinių. Esminis WGAN skirtumas nuo standartinio GAN yra nuostolių funkcijos pakeitimas *Wasseršteino* nuostolių funkcija. Tai leidžia užtikrinti stabilesnį modelio apmokymo procesą, kadangi kaip minėta aukščiau, standartiniai GAN tinklai neretai sudėtingas dėl jiems būdingų problemų atsiradimo. Tai būtina norint naudoti GAN anomalijų aptikimo uždaviniams, kadangi jei generatorius negali atkurti visų duomenų pasiskirstymo pavyzdžių, tuomet trūkstami režimai gali būti traktuojami kaip anomalijos, o tai padidina klaidingai teigiamų rezultatų skaičių [26].

Tačiau tiek GAN, tiek WGAN modeliai turi savų pranašumų ir trūkumų ir yra tinkami skirtingiems taikymo scenarijams. Standartinius GAN dažniausiai yra lengviau suprasti ir įgyvendinti, taip pat egzistuoja daug jų modifikacijų bei pritaikymų įvairioms užduotims. Tuo tarpu WGAN išsiskiria gebėjimu sumažinti dažniausiai pasitaikančias standartiniuose GAN mokymo problemas, todėl jie naudingi situacijose, kuriose itin svarbus modelio stabilumas ir patikimas apmokymo procesas.

Kitas GAN modelių tipas, pritaikytas būtent anomalijų aptikimui yra AnoGAN modelis. Pagrindinis iššūkis su kuriuo susiduriama naudojant generatyvinius priešpriešinius tinklus anomalijų aptikimo uždaviniams yra tai, kad šiai architektūrai trūksta kodavimo mechanizmo iš duomenų erdvės atgal į latentinę erdvę z . AnoGAN sprendžia šią problemą naudojant optimizavimo procesą, kai pavyzdžiui x randamas atitinkamas latentinis vektorius z^* taip, kad sugeneruotas vaizdas $G(z^*)$ būtų labai panašus į x . Šis latentinis vektorius gaunamas minimizuojant nuostolių funkciją, jungiančią du narius: liekamuosius nuostolius ir diskriminacinius nuostolius [27].

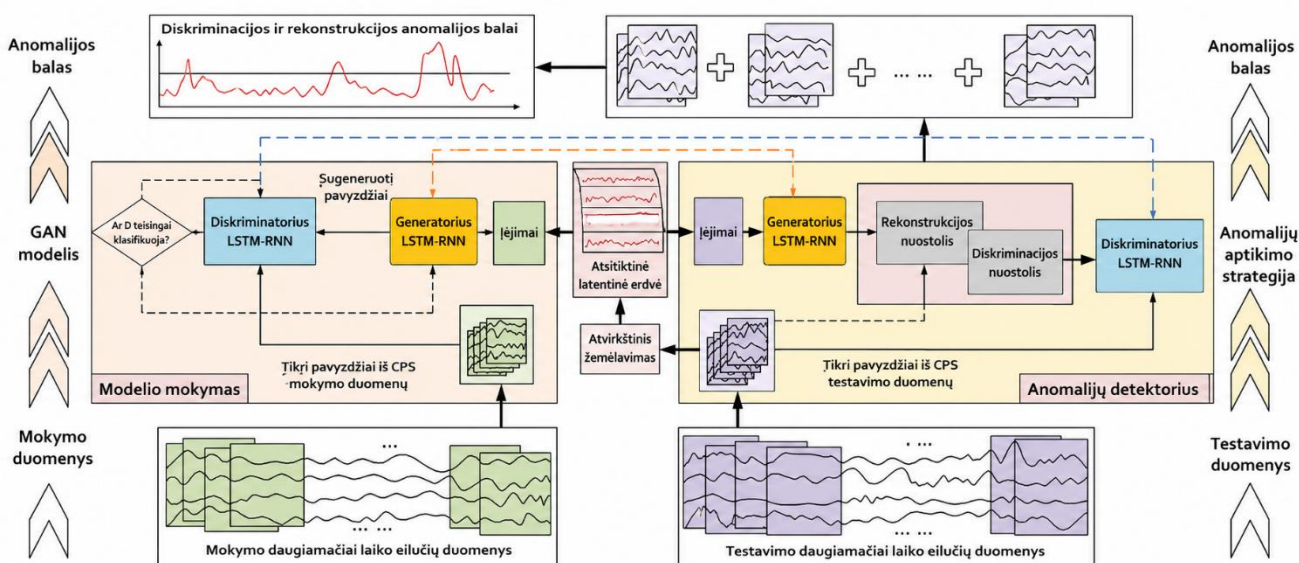
Vienas iš pagrindinių šio modelio privalumų lyginant su kitais generatyviniais tinklais – gebėjimas modeliuoti sudėtingus, didelės apimties duomenų skirstinius, kas leidžia modelį naudoti tokiose srityse kaip pramoniniai procesai, medicina, defektų aptikimas ir taip toliau. Tačiau šis modelis taip pat turi ir trūkumų. Vienas iš jų yra tai, kad optimizavimo procedūra, kurios metu ieškoma latentinės erdvės vektorių kiekvienam modelio ciklui, reikalauja daug resursų, kas gali trukdyti anomalijų aptikimo realiu laiku uždaviniams. Taip pat, anomalijų aptikimo efektyvumas ir duomenų rekonstrukcijos kokybė labai priklauso nuo modelio mokymo duomenų, bei apmokymo stabilumo.

Siekiant padaryti AnoGAN modelį efektyvesniu, į modelio architektūrą buvo pridėtas enkoderio tinklas, kas leidžia greičiau nustatyti anomalijų įvertį, be ciklinio optimizavimo. Taip atsirado tokie modeliai kaip fAnoGAN ir EfficientGAN [28]. Vis tik AnoGAN modelis išlieka labiau tinkamu anomalijų aptikimui vaizduose, ar kitokio tipo duomenyse, nei laiko eilutėse.



16 pav. AnoGAN modelio, papildyto enkoderiu, architektūra [28]

Dar vienas modelis, sukurtas anomalijų aptikimui naudojant generatyvinius priešpriešinius tinklus būtent laiko eilutėse yra MAD-GAN modelis. Siekiant apdoroti laiko eilučių duomenis, GAN generatorius ir diskriminatorius sukuriami kaip ilgalaikės-trumpalaikės atminties rekurentinių neuroninių tinklų (LSTM-RNN) modelius. Kaip parodyta nuotraukoje žemiau, taip pat kaip ir tipinėje GAN modelio struktūroje, generatorius generuoja dirbtinius duomenis, šiuo atveju laiko eilutes, naudodamas sekas iš atsitiktinės latentinės erdvės. Šie sugeneruoti duomenys perduodami diskriminatoriui, kuris bando jas atskirti nuo tikrų duomenų skirstinių. Vietoj to, kad kiekvienas duomenų srautas būtų traktuojamas atskirai, MAD-GAN modelio struktūra atsižvelgia į visą duomenų rinkinį vienu metu, kas leidžia modeliui nustatyti ryšius tarp kintamųjų. Daugiamatės laiko eilutės prieš perduodant diskriminatoriui padalijamos į dalis, naudojant slankiuosius langus.

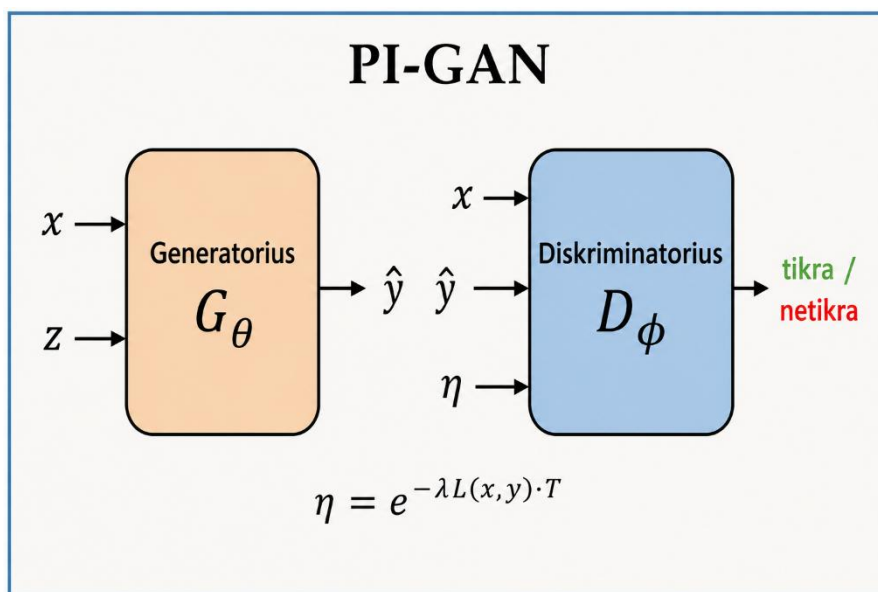


17 pav. MAD-GAN modelio architektūra [29]

Kaip ir standartiniame GAN modelyje, diskriminatoriaus ir generatoriaus parametrai yra atnaujinami pagal diskriminatoriaus išvestis, kas leidžia diskriminatorių apmokyti kuo taisykliau ir jautriau atskirti tiek realiais, tiek generuotais duomenų sekas, kol generatorius mokomas kurti kuo į realius duomenis panašesnes duomenų sekas. Gebėdamas generuoti realistiškus duomenis, generatorius išmoka paslėptus duomenų kintamųjų ryšius, kitaip sakant daugiamatį eilučių duomenų skirstinius. Šiame modelyje anomalijų aptikimui naudojamas tiek generatorius, tiek diskriminatorius. Generatorius naudojamas skaičiuojant rekonstrukcijos paklaidą tarp realiu ir sugeneruotų pavyzdžių, o diskriminatorius naudojamas laiko eilučių duomenų klasifikavimui ir apskaičiuojamas diskriminacijos nuotolis. Šios dvi paklaidos naudojamos apskaičiuojant anomalijų balą [29].

Pastaruoju metu išpopuliarėjo dar vienas priešpriešinių generatyvinių tinklų modelių tipas – fizikiniais dėsniais pagrįsti GAN, sutrumpinti literatūroje vadinami PI-GAN. Šie modeliai fizikinius dėsnius bei savybes, tokius kaip dalines diferencialines lygtis, nuostolių funkcijas, algebrinius apribojimus ar kitokias specifinės srities sąlygas, įtraukia į GAN modelių mokymosi procesą, taip apribojant modelių generuojamus duomenis ir juos patikslinant.

Šiuose modeliuose įvedamas naujas parametras – fizikinių savybių nuoseklumo balas, kuris indukuoja modelio spėjimo tikimybę atitikti nurodytas fizikines sąlygas. Šis parametras yra perduodamas diskriminatoriaus tinklui kaip papildoma sąlyga, todėl diskriminatorius ne tik bando atskirti realius duomenis nuo sugeneruotų duomenų, bet ir mokosi duomenų skirstinio pagal užduotas fizikines sąlygas [30].



18 pav. PI-GAN modelio architektūra [30]

1.4. Anomalijų aptikimo metodų vertinimo metrikos

Anomalijų aptikimo modelių našumas vertinamas specializuotomis metrikomis, kuriomis galima įvertinti modelių generuojamų rezultatų kokybę [31]. Pagrindinės anomalijų aptikimo modelių vertinimui naudojamos metrikos yra apskaičiuojamos pagal tam tikras formules, kurių kintamieji pagrindu yra šie elementai:

1. Teisingai neigiamas atvejis. Skaičius normalių duomenų taškų, aptiktų teisingai.
2. Klaidingai neigiamas atvejis. Skaičius anomalijų, aptiktų kaip normalūs duomenys.
3. Teisingai teigiamas atvejis. Skaičius anomalijų, aptiktų kaip anomalijos.
4. Klaidingai teigiamas atvejis. Skaičius normalių duomenų taškų, aptiktų kaip anomalijos.

Pagal šiuos kintamuosius modelių efektyvumui nustatyti skaičiuojamos šios pagrindinės metrikos:

1. Tikslumas (angl. *precision*). Tikslumu vertinamas tikrų aptiktų anomalijų ir visų atvejų, kurie buvo pažymėti kaip anomalijos, santykis. Ši metrika parodo kiek modelio pažymėtų anomalijų yra nustatytos teisingai. Atsižvelgiant į šį rodiklį galima stebėti modelio klaidingai aptinkamų anomalijų kiekį.
2. Atkūrimas (angl. *recall*). Atkūrimas parodo, kaip dažnai mašininio mokymosi modelis teisingai nustato teigiamus atvejus (tikruosius teigiamus atvejus) iš visų faktinių teigiamų atvejų duomenų rinkinyje.
3. F1 balas (angl. *F1 score*). F1 balas - tai metrika, mašiniame mokyme kombinuojanti tikslumo ir atkūrimo rodiklius, kaip harmoninį vidurkį. Dažnai naudojama kaip vertinimo metrika dvinariame ir daugiaklasiame klasifikavime.
4. Klaidingai teigiamų atvejų rodiklis (angl. *false positive rate*). Klaidingai teigiamų atvejų rodiklis yra metrika, kuria skaičiuojamas anomalijų aptikimo modelio neteisingų teigiamų spėjimų procentas. Jis matuojamas lyginant modelio spėtus teigiamus atvejus su tikrais neigiamais atvejais.
5. *Matthewso* koreliacijos koeficientas (angl. *Matthews correlation coefficient*). Šis algoritmas skirtas matuoti binarinės klasifikacijos kokybę. Lyginant šią metriką su dažniausiai anomalijų aptikimo modelių vertinimui naudojamu F1 balu, šis algoritmas skiria didesnę dėmesį tikriesiems neigiamiems atvejams, o tai leidžia geriau patikrinti modelio apmokymo lygį. MCC reikšmės gali būti nuo -1 iki 1. Kai MCC lygus -1, tai reiškia, kad modelis modelio reikšmės yra priešingos negu yra pagrindinė tiesa (angl. „*ground truth*“), tuo metu 1 reiškia, kad modelio reikšmės sutampa su pagrindine tiesa. 0 reikšmė MCC algoritme reiškia, kad nėra jokios koreliacijos tarp modelio rezultatų ir pagrindinės tiesos, kas reiškia, kad modelis gražina bet kokias reikšmes ir nėra gerai apmokytas.

Kaip minėta aukščiau, tikslumo, atkūrimo bei F1 balo metrikos skaičiuojamos naudojant tam tikras matematines formules, pateiktas lentelėje žemiau [32].

1 lentelė. Anomalijų aptikimo modelių vertinimo metrikų matematinės formulės.

Metrika	Matematinė formulė
Tikslumas	$\frac{\text{Teisingai teigiamas atvejis}}{\text{Teisingai teigiamas atvejis} + \text{Klaidingai teigiamas atvejis}}$
Atkūrimas	$\frac{\text{Teisingai teigiamas atvejis}}{\text{Teisingai teigiamas atvejis} + \text{Klaidingai neigiamas atvejis}}$
F1 balas	$2 * \left(\frac{\text{Tikslumas} * \text{Atkūrimas}}{\text{Tikslumas} + \text{Atkūrimas}} \right)$
Klaidingai teigiamų atvejų rodiklis	$\frac{\text{Klaidingai teigiamas atvejis}}{\text{Klaidingai teigiamas atvejis} + \text{Teisingai neigiamas atvejis}}$

1.5. Pramoninių procesų duomenų rinkiniai

Siekiant įvertinti ir plačiau išanalizuoti pramoninius procesus reikalingi duomenų rinkiniai, sudaryti iš proceso sistemoje veikiančių jutiklių, valdiklių ir kitokių reikalingų duomenų. Norint sudaryti tokius rinkinius reikia daug laiko, resursų bei prieigos prie realių, fizinių sistemų. Todėl įvairiems tyrimams ir bandymams neretai naudojamos jau sudarytos, viešai prieinamos pramoninių procesų duomenų bazės. Šiame skyriuje apžvelgtos kelios viešai prieinamos duomenų bazės, dažnai literatūroje naudojamų anomalijų aptikimo uždaviniams pramoniniuose procesuose atlikti.

1.5.1. Saugaus vandens valymo sistema SWaT

Saugaus vandens valymo sistema SWaT (angl. *secure water treatment system*) - vandens valymo bandymų stendas, sumažinta realių miesto vandens valymo įrenginių versija. SWaT duomenų rinkimo procesas truko 11 dienų, sistema veikė 24 valandas per parą. Per paskutines 4 dienas duomenų rinkimo proceso SWaT sistemoje buvo surengtos iš viso 36 atakos. Sistemoje atakuojami taškai apima jutiklius (vandens lygio jutiklius, srauto matuoklius ir pan.) ir pavaras (vožtuvus, siurblius ir pan.). Šios atakos bandymų stende buvo surengtos su skirtingais tikslais ir įvairia atakų trukme (nuo kelių minučių iki valandos) per paskutines keturias duomenų rinkimo dienas. Prieš surengiant sekančią ataką, vienais atvejais sistemai buvo leidžiama pasiekti įprastą veikimo būseną, o kitais atvejais atakos buvo vykdomos paeiliui. Vandens valymo procesas SWaT sistemoje susideda iš šešių posistemių, žymimų *P1–P6*. Pirmasis procesas skirtas neapdoroto vandens tiekimui ir saugojimui, *P2* – pirminiam valymui, kurio metu vertinama vandens kokybė. Nepageidaujamos medžiagos pašalinamos ultrafiltravimo ir atgaline plovimo procedūra *P3* procese. Likęs chloras sunaikinamas dechlorifikacijos procese (*P4*). Vėliau vanduo iš *P4* proceso pumpuojamas į atvirkštinės osmozės sistemą (*P5*), kur sumažinamas neorganinių priemaišų kiekis. Galiausiai *P6* procese vanduo saugomas, kol bus paruoštas tolimesniam skirstymui [29].

1.5.2. Vandens paskirstymo sistema WADI

Vandens paskirstymo sistema WADI (angl. *water distribution system*). Skirtingai nuo vandens valymo įrenginių, kurie paprastai būna įrengti saugioje vietoje, vandens paskirstymo sistema susideda iš daugybės vamzdinių, nusidriekiančių dideliame plote. Tai smarkiai padidina fizinių atakų vandens paskirstymo tinkle riziką. Vandens paskirstymo (WADI) bandymų platforma yra SWaT sistemos išplėtimas, kurioje naudojama dalis SWaT atvirkštinės osmozės filtrato ir neapdoroto vandens, siekiant sukurti išbaigtą ir realistišką vandens valymo, saugojimo ir paskirstymo tinklą. Vandens paskirstymo sistemoje yra trys valdymo procesai. Pirmasis procesas – neapdoroto vandens paėmimas iš SWaT, Singapūro Viešųjų paslaugų valdybos įvado arba grįžtamojo vandens WADI sistemoje ir neapdoroto vandens saugojimas dviejuose rezervuaruose. *P2* paskirsto vandenį iš dviejų pakeltų rezervuarų ir šešių vartotojų rezervuarų pagal iš anksto nustatytą paklausos modelį. Trečiajame procese vanduo yra perdirbamas ir grąžinamas į *P1*. WADI bandymų stende taip pat yra įrengtos cheminių medžiagų dozavimo sistemos, slėgio didinimo siurblys ir vožtuvai, matavimo prietaisai bei analizatoriai. Be atakų valdymo sistemoje imitavimo, WADI turi galimybes imituoti fizinių atakų, tokių kaip vandens nuotėkis ir tyčinis cheminių medžiagų įleidimas, situacijas. WADI duomenų rinkimo procesas truko 16 dienų nepertraukiamos sistemos veiklos, iš kurių 14 dienų duomenys buvo renkami įprastinėmis veiklos sąlygomis, o 2 dienas – taikant atakų scenarijus. Duomenų surinkimo metu buvo surinkti visi tinklo srautų, jutiklių ir pavarų duomenys [29].

1.5.3. Vandens paskirstymo sistema BATADAL

BATADAL - tai duomenų rinkinys, sudarytas iš simuliuotos vandens paskirstymo sistemos. Rinkinį sudaro keli duomenų rinkiniai, vienas sudarytas tik iš normalaus veikimo duomenų, kitas – su atakuojamos sistemos veikimo duomenimis, taip pat yra rinkinys skirtas testams, su sužymėtomis anomalijomis. Rinkinys yra paruoštas mašininio mokymo modeliams apmokyti, tačiau tai simuliuotas rinkinys, todėl duomenys yra nevisai realistiški, lyginant su fizinės sistemos duomenų rinkiniais. Šis duomenų rinkinys buvo sukurtas vykdant konkursą, kurio tikslas – palyginti algoritmų, naudojamų kibernetinėms atakoms vandens tiekimo sistemose aptikti, veikimą. Jis simuliuoja miesto vandens paskirstymo sistemą, pagrįstą realiu vidutinio dydžio tinklu, kurį sudaro 429 vamzdžiai, 388 jungtys, 7 talpyklos, 11 siurblių, išdėstytų 5 siurblinėse, 5 vožtuvų, vieno rezervuaro ir devynių programuojamų loginių valdiklių. Duomenų rinkinį sudaro 43 skaitmeniniai rodikliai, fiksuojami kartą per valandą, taigi kiekvienas požymių vektorius atitinka vieną valandą ir susideda iš 43 požymių. Šie požymiai yra SCADA sistemos rodmenys ir apima 7 požymius, atspindinčius vandens lygį rezervuaruose, 12 požymių, atspindinčių įleidimo ir išleidimo slėgį valdomame vožtuve bei visose siurblinėse, ir 24 požymius, atspindinčius jų srautą ir būseną [33].

1.5.4. „Tennessee Eastman“ procesas

„Tennessee Eastman“ procesas (angl. *Tennessee Eastman process*) taip pat yra duomenų rinkinys paremtas simuliacija. Šis rinkinys simuliuoja cheminio proceso valdymo sistemą, kuri reaguoja į išorinių veiksnių pokyčius. Šis duomenų rinkinys paremtas proceso modeliu, kuris yra dažnai naudojamas etalonas procesų automatizavimo srityje. Sistemą galima suskirstyti į penkis modulius - dviejų fazių cheminį reaktorių, kondensatorių, garų ir skysčių separatorių, desorberį ir boilerį. Be to, „Tennessee Eastman“ procesas apima 11 automatinių pneumatinio valdymo vožtuvų, du siurblius ir vieną kompresorių. Proceso signalizacijos sistema apibrėžia 81 žemo ir 81 aukšto pavojaus slenksčio signalus, taip pat penkis labai aukšto ir tris labai žemo pavojaus slenksčių signalus. Duomenų rinkinys apima 100 simuliacijų su 300 nurodytų situacijų su anomalijomis. Šios situacijos buvo suprojektuotos naudojant aštuonių skirtingų sutrikimų variacijas. Jos skiriasi jų trukmėmis, masteliais ir deriniais.. Signalizacijos sistema generuoja iš viso 7343 signalizacijos atvejus per visas 300 situacijų. Vienas iš šio duomenų rinkinio privalumų yra tai, kad visos sukeltos anomalijos yra aiškiai žinomos ir aprašytos, todėl galima taikyti išorinės validacijos rodiklį [34].

2. Metodinė dalis

Siekiant atlikti tyrimą, kuriame analizuojamas mašininio mokymo modelių anomalijų aptikimo efektyvumo palyginimas vandens valymo sistemoje, reikalingos priemonės, leidžiančios gauti rezultatus ir duomenis, reikalingus tolimesnei analizei tiriamojame darbo dalyje. Numatoma, kad tyrimui bus reikalingi realizuoti mašininio mokymo modeliai, tinkami anomalijų aptikimui pramoninių procesų duomenyse, duomenų rinkiniai, tinkami apmokyti mašininio mokymo modelius, taip pat aplinka, skirta mašininio mokymo modelių realizavimui, apmokymui, bei modelių efektyvumo vertinimo metrikų gavimui.

2.1. Parinktų duomenų rinkinių specifikacijos

Siekiant sukurti ir ištestuoti modelį, aptinkantį pramoninio proceso anomalijas, reikia parinkti procesą, kurio duomenys bus naudojami modeliui mokyti. Šiam tikslui dažnai naudojami jau paruošti duomenų rinkiniai, sudaryti tam tikro proceso veikimo metu, plačiau aprašyti 1.5 skyriuje. Šiuo atveju tyrimui parinktas saugaus vandens valymo sistemos SWaT (angl. *secure water treatment system*) duomenų rinkinys, kadangi šis rinkinys yra sudarytas iš realios, pilnai veikiančios vandens valymo įrenginio sistemos duomenų, taip pat nesunku gauti prieigą prie šio duomenų rinkinio, rinkinys, bei procesas yra aiškiai dokumentuotas ir aprašytas. Taip pat, siekiant tikslingiau palyginti mašininio mokymo modelių efektyvumą, parinktas ir antras duomenų rinkinys modelių apmokymui bei testavimui - vandens paskirstymo sistemos WADI (angl. *water distribution system*) duomenų rinkinys. Šis rinkinys sudarytas iš sistemos, kuri yra SWaT sistemos išplėtimas, todėl duomenų rinkinio sandara yra panaši, tačiau skiriasi proceso veikimas, duomenų skirstiniai. Abiejuose parinktuose duomenų rinkiniuose yra visi trys 1.2 skyriuje aprašyti anomalijų tipai (taškinės, kontekstinės bei kolektyvinės anomalijos). Tolimesniuose poskyriuose aprašomos SWaT bei WADI duomenų rinkinių specifikacijos.

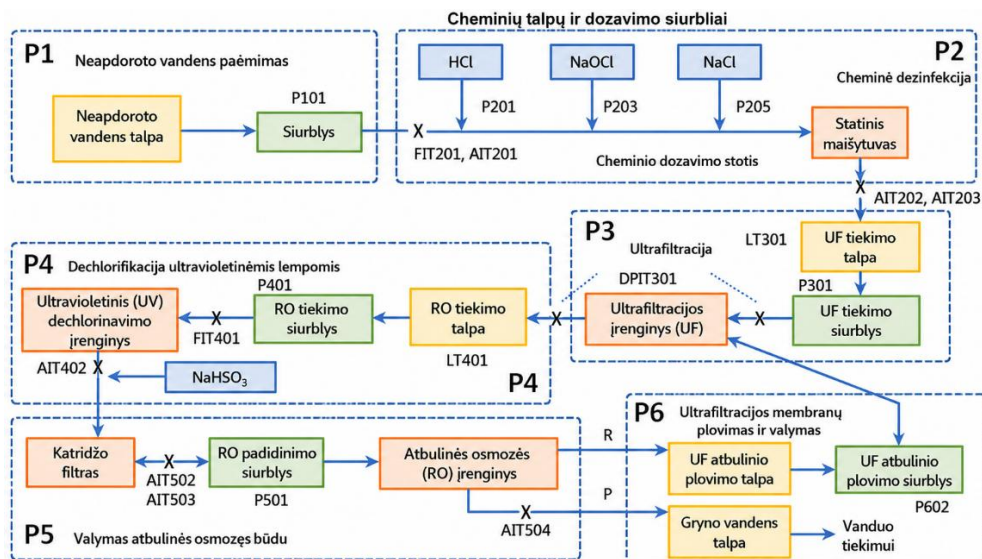
2.1.1. SWaT

SWaT duomenų rinkinys yra sudarytas iš modernaus šešių pakopų vandens valymo įrenginio, kuris apdoroja apytiksliai 19 litrų vandens per minutę duomenų. SWaT sistemos sandara atvaizduota 19 paveikslėlyje. Sistemą sudaro 6 procesai, kurie išvalo vandenį prieš jį tiekiant:

1. *P1* - neapdoroto vandens paėmimas.
2. *P2* - cheminė dezinfekcija.
3. *P3* - ultrafiltracija.
4. *P4* - dechlorifikacija naudojant ultravioletines lempas.
5. *P5* - valymas naudojant atbulinę osmozę.
6. *P6* - ultrafiltracijos membranos plovimas ir valymas.

Pirmasis procesas (*P1*) palaiko neapdoroto vandens talpos lygį, iš kurios vanduo tiekiamas kitiems procesams. Antrasis procesas (*P2*) apdoroja vandenį iki nustatytos kokybės ribos. Jei neapdorotas vanduo talpoje pirmo proceso metu nesiekia priimtinos kokybės normos (matuoja *AIT201*, *AIT202* ir *AIT203* jutikliai), pompos *P201*, *P203* ir *P205* atidaromos ir tiekiamas tam tikras chemikalų kiekis vandens kokybės gerinimui. Vandeniui pasiekus reikiamą kokybės normą, jis tiekiamas trečiajam

procesui (P3). Šio proceso metu šalinamos vandenyje likusios nepageidaujamos medžiagos, ultrafiltracijos (UF) metodu, naudojant filtravimo membraną. Ketvirtojo (P4) proceso metu vandenyje esantis chloras pašalinamas dechlorifikacijos metodu, naudojant ultravioletines lempas. Toliau šalinamos įvairios vandens priemaišos, dechlorifikuotas vanduo tiekiamas penktajam (P5) procesui, kuriame vyksta atbulinė osmozė. Galiausiai šeštajame procese (P6) vanduo tiekiamas vartojimui.



19 pav. SWaT (angl. *secure water treatment system*) sistema ir ją sudarantys procesai [35]

SWaT sistemą sudaro 25 jutikliai ir 26 pavaros. Kiekvienoje iš proceso dalių naudojami jutikliai, jų vertės perduodamos į programuojamą loginį valdiklį, kuris valdo procesų būsenas. Priklausomai nuo jutiklių reikšmių, programuojamas loginis valdiklis duoda signalą pavaroms. Pavyzdžiui, pirmo (P1) proceso metu jutiklis *LIT-101* matuoja vandens lygį talpoje. Programuojamas loginis valdiklis pagal jutiklio reikšmę nusprendžia kiek atidaryti ar uždaryti vožtuvą *MV-101*. Jei *LIT-101* reikšmė viršija nustatytą ribą, *MV-101* vožtuvas uždaromas, kas sustabdo neapdoroto vandens įtekėjimą į talpą.

Duomenų surinkimas SWaT sistemoje truko 11 dienų. Sistema visą šį laiką veikė nepertraukiamai, 24 valandas per parą. Per pirmas septynias duomenų surinkimo dienas sistema dirbo normaliomis sąlygomis (be trikdžių). Trikdžiai buvo paleisti per paskutines keturias duomenų rinkimo dienas. Trikdžiai sukuriama perrašant jutiklių reikšmes programuojamajame loginiame valdiklyje, kai kurios reikšmės keičiamos į netikras iš lėto, o kitos iš kart. Iš viso duomenų rinkinį sudaro 53 eilutės (1 laiko žymeį, 1 trikdžių žymeį, 51 jutiklių, pavarų reikšmėms) ir 912791 stulpeliai.

2 lentelė. Jutikliai ir pavaros SWaT duomenų bazėje.

Procesas	Jutikliai	Pavaros
P1	FIT101, LIT101	MV101, P101, P102
P2	FIT201, AIT201, AIT202, AIT 203	MV201, P201, P202, P203, P204, P205, P206
P3	FIT301, DPIT301, LIT301	MV301, MV302, MV302, MV303, MV304, P301, P302
P4	FIT401, AIT401, AIT402, LIT401	UV401, P401, P402, P403, P404

P5	FIT501, FIT502, FIT503, FIT504, AIT501, AIT502, AIT503, AIT504, PIT501, PIT502, PIT503	P501, P502
P6	FIT601	P601, P602, P603

Ataka	Aprašymas	Poveikis	Ar atakos tikslas pasiektas?
1	MV-101 yra atidarytas, nors turėtų būti uždarytas	Rezervuaro perpildymas	Taip
2	P-102 įjungtas, nors turėtų būti išjungtas	Vamzdžio trūkimas	Taip
3	LT-101 rodmuo padidintas 1 m per sekundę greičiu	Rezervuaro ištuštėjimas ir P-101 pažeidimas	Taip
4	MV-504 atidarytas, nors turėtų būti uždarytas	Poveikio nėra	Ne
5	AIT-202 rodmuo sumažintas žemiau norminės vertės	AIT-504 padidėjo, UV neįsijungė	Ne
6	LT-301 rodmuo padidintas virš maksimalios ribos	Rezervuaro ištuštėjimas ir P-101 pažeidimas	Taip
7	DPIT-301 rodmuo padidintas virš norminės vertės	Atbulinis procesas paleistas, sumažintas Tank-401 vandens lygis, padidintas Tank-301 vandens lygis	Taip
8-9	FIT-401 rodmuo sumažintas žemiau norminės vertės	UV procesas išjungtas, P-501 išsijungia	Taip
10	MV-304 uždarytas, nors turėtų būti atidarytas	MV-304 buvo uždarytas	Ne
11	MV-303 užstrigęs uždarytoje padėtyje	Poveikio nėra	Ne
12	LT-301 rodmuo sumažintas 1 m per sekundę greičiu	Rezervuaro perpildymas	Taip
13	MV-303 užstrigęs uždarytoje padėtyje	Sustabdyta 3 pakopos veikla	Taip
14-15	AIT-504 rodmuo padidintas virš norminės vertės	Poveikio nėra	Ne
16	MV-101 užstrigęs atidarytoje padėtyje, LT-101 rodmuo nustatytas į 0,7 m	Rezervuaro perpildymas	Taip
17	UV-401 išjungtas, AIT-502 rodmuo padidintas virš norminės vertės, P-501 įjungtas	Sumažintas FIT-502 išėjimo srautas	Ne
18	DPIT-301 rodmuo padidintas virš norminės vertės, MV-302 užstrigęs atidarytoje padėtyje, P-602 užstrigęs išjungtoje padėtyje	Sistemos užšalimas	Taip
19	P-203 ir P-205 įjungti, nors turėtų būti išjungti	Poveikio nėra	Ne
20	LT-401 rodmuo padidintas virš norminės vertės, P-205 užstrigęs įjungtoje padėtyje	Rezervuaro perpildymas	Taip
21	P-101 užstrigęs įjungtoje padėtyje, nors turėtų būti išjungtas, LT-301 rodmuo nustatytas į 0,8 m	P-101 ištuština Tank-101 ir rezervuaro perpildymas	Taip
22	P-302 užstrigęs įjungtoje padėtyje, o LT-401 rodmuo nustatytas į 0,6 m	Rezervuaro perpildymas	Taip
23	P-302 išjungtas, nors turėtų būti įjungtas	Ištekėjimas į Tank-401 sustoja	Taip
24	P-201, P-203 ir P-205 įjungti, nors turėtų būti išjungti	Poveikio nėra	Ne
25	P-101 ir MV-101 užstrigę įjungtoje padėtyje, nors turėtų būti išjungti, ir LT-101 rodmuo nustatytas į 0,7 m	Tank-101 ištuštėja ir rezervuaro perpildymas	Taip
26	LT-401 rodmuo sumažintas žemiau minimalaus lygio	Rezervuaro perpildymas	Taip
27	LT-301 rodmuo padidintas virš maksimalaus lygio	Rezervuaro ištuštėjimas ir P-302 pažeidimas	Taip
28	LT-101 rodmuo padidintas virš maksimalaus lygio	Rezervuaro ištuštėjimas ir P-101 pažeidimas	Taip
29	P-101 užstrigęs įjungtoje padėtyje, nors turėtų būti išjungtas	Atsarginis siurblys P-102 įjungtas	Ne
30	P-101 išstrigęs, nors turėtų būti įjungtas, ir P-102 užstrigęs išjungtoje padėtyje	Ištekėjimas sustoja	Taip
31	LT-101 rodmuo sumažintas žemiau minimalaus lygio	Rezervuaro perpildymas	Taip
32	P-501 išjungtas, nors turėtų būti įjungtas, ir FIT-502 rodmuo padidintas virš norminės vertės	P-501 neišsijungia, FIT-502 sumažėja, P-501 greitis padidėja	Ne
33	AIT-402 ir AIT-502 rodmenys nustatyti į 260	Poveikio nėra	Ne
34	FIT-401 ir AIT-501 rodmenys nustatyti į 0,5 ir 140 mV	Poveikio nėra	Ne
35	FIT-401 rodmuo nustatytas į nulį	P-402 neatidaromas	Ne
36	LT-301 rodmuo sumažintas 0,55 per sekundę greičiu	Rezervuaro vandens lygis mažėja	Ne

20 pav. Atakų SWaT sistemos procese sąrašas [36]

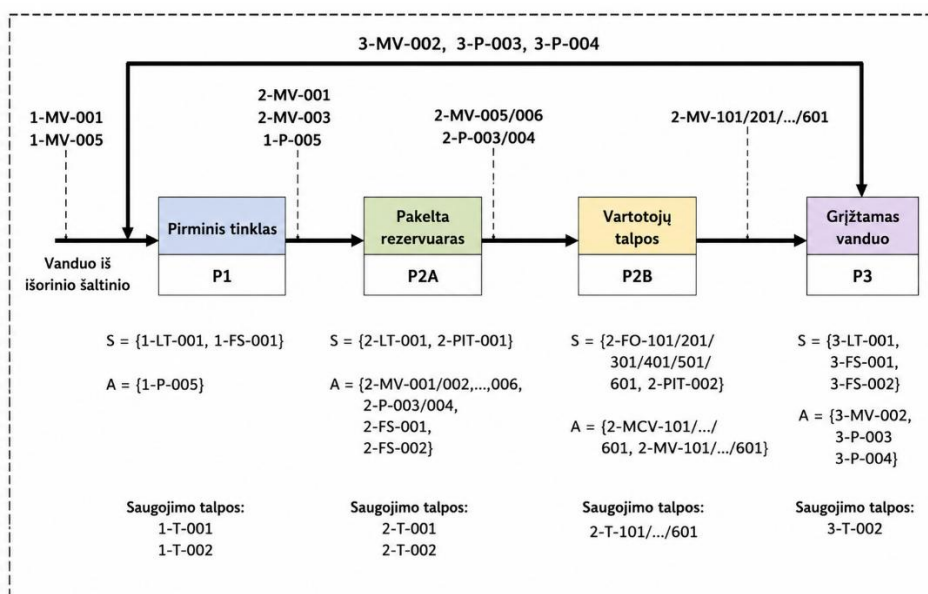
2.1.2. WADI

WADI duomenų rinkinys buvo parinktas antruoju rinkiniu anomalijų aptikimo modelių apmokymui ir vertinimui. WADI laikomas SWaT sistemos plėtinium, kadangi ši sistema yra fiziškai prijungta prie SWaT sistemos ir iš jos gauna išvalytą vandenį, kuris toliau yra skirstomas. WADI duomenys atspindi vandens paskirstymo tinklo veikimą, todėl jis tinka modelių veikimui vertinti esant skirtingoms pramoninių procesų charakteristikoms. WADI duomenų rinkinys buvo surinktas iš pilnai veikiančios vandens paskirstymo bandymų platformos, sukurtoje kibernetinio saugumo tyrimams kibernetinėse-fizinėse sistemose. Sistema tiekia apie 38 litrų filtruoto vandens per minutę ir atspindi sumažintą tikro miesto vandens paskirstymo tinklo versiją. WADI architektūra susideda iš trijų pagrindinių procesų, kuriuos kiekvieną valdo specialūs programuojami loginiai valdikliai ir nuotoliniai terminalai:

1. *P1-W* – pirminis tinklas.
2. *P2-W* – antrinis tinklas.
3. *P3-W* – grįžtamojo vandens tinklas.

Pirmajame procese (*P1-W*) iš SWaT sistemos gaunamas išvalytas vanduo surenkamas ir saugomas dviejuose neapdoroto vandens rezervuaruose, kurių kiekvieno talpa yra 2500 litrų. Vandens lygio jutikliais stebimas rezervuarų lygis, o analizatoriaus jutikliai matuoja vandens kokybės parametrus.

Siekiant užtikrinti priimtina vandens kokybę prieš jo paskirstymą, sistemoje įrengtos cheminių medžiagų dozavimo sistemos. Antrasis procesas ($P2-W$) atsakingas už vandens paskirstymą. Vanduo iš pirminių talpyklų pumpuojamas į du pakeltus rezervuarus, o tada paskirstomas į šešias vartotojų talpas pagal iš anksto sukurtus paklausos modelius, kurie imituoja realų vartotojų vandens naudojimą. Srauto jutikliai, slėgio jutikliai, slėgio skirtumo jutikliai ir srauto reguliatoriai naudojami paskirstymo procesui reguliuoti ir stebėti. Trečiasis procesas ($P3-W$) tvarko grįžtamojo vandens surinkimą atgal į sistemą. Kai vartotojų rezervuarai užpildomi, likęs vanduo grįžta į grįžtamojo vandens tinklą, kur jis surenkamas ir nuvedamas atgal į pirminį tinklą.



21 pav. WADI (angl. *water distribution system*) sistema ir ją sudarantys procesai [37]

WADI sistemą sudaro 69 jutikliai ir 54 pavaros, įskaitant analizatorius, srauto daviklius, lygio daviklius, slėgio jutiklius, siurblius, variklinius vožtuvus, elektromagnetinius vožtuvus ir proceso valdiklius.

Duomenų rinkinys buvo surinktas per 16 dienų nepertraukiamo veikimo laikotarpį. Per pirmąsias 14 dienų sistema veikė įprastomis sąlygomis be sutrikimų, o surinkti duomenys buvo sudėti į mokymo rinkinį. Per paskutines 2 dienas prieš sistemą buvo paleisti keli kibernetiniai-fiziniai atakų scenarijai, taip sukuriant testavimo rinkinį. Panašiai kaip ir SWaT duomenų rinkinyje, atakos buvo vykdomos manipuliuojant jutiklių matavimais, pavarų būsenomis arba valdiklių reikšmėmis, imituojuant realistiškus nuokrypius, nukreiptus į skirtingus proceso etapus. Iš viso WADI duomenų rinkinį sudaro 127 stulpeliai bei 1221372 eilutės.

3 lentelė. Jutikliai ir pavaros WADI duomenų bazėje.

Procesas	Jutikliai	Pavaros
P1-W	1_LT_001, 1_AIT_001, 1_AIT_002, 1_AIT_003, 1_AIT_004, 1_AIT_005, 1_LS_001, 1_LS_002	1_MV_001, 1_MV_002, 1_MV_003, 1_MV_004, 1_P_001, 1_P_002, 1_P_003, 1_P_004, 1_P_005, 1_P_006

P2-W	2_LT_001, 2_LT_002, 2_FIT_001, 2_FIT_002, 2_FIT_003, 2_PIT_001, 2_PIT_002, 2_PIT_003, 2_DPIT_001, 2A_AIT_001-004, 2B_AIT_001-004	2_MV_001, 2_MV_002, 2_MV_003, 2_MV_004, 2_MV_005, 2_MV_006, 2_MV_009, 2_MV_101, 2_MV_201, 2_MV_301, 2_MV_401, 2_MV_501, 2_MV_601, 2_MCV_101, 2_MCV_201, 2_MCV_301, 2_MCV_401, 2_MCV_501, 2_MCV_601, 2_SV_101, 2_SV_201, 2_SV_301, 2_SV_401, 2_SV_501, 2_SV_601, 2_P_001, 2_P_002, 2_P_003, 2_P_004
P3-W	3_LT_001, 3_FIT_001, 3_AIT_001, 3_AIT_002, 3_AIT_003, 3_AIT_004, 3_AIT_005, 3_LS_001	3_MV_001, 3_MV_002, 3_MV_003, 3_P_001, 3_P_002, 3_P_003, 3_P_004

Ataka	Aprašymas	Poveikis	Ar atakos tikslas pasiektas?
1	1-MV-001 yra atidarytas, nors turėtų būti uždarytas	Rezervuaro perpildymas	Ne
2	1-FIT-001 rodmuo yra pakeistas (sugadintas)	Pakeičiama vandens kokybė	Taip
3-4	1-AIT-001 rodmuo yra pakeistas (sugadintas)	Rezervuaro ištuštėjimas	Taip
5	2-MCV-101 iki 2-MCV-601 uždaryti, nors turėtų būti atidaryti	Trikdoma vandens paskirstymo proceso eiga	Taip
6	2-MCV-101 ir 2-MCV-201 atidaryti, nors turėtų būti uždaryti	Trikdoma vandens paskirstymo proceso eiga	Taip
7	1-AIT-002 rodmuo yra pakeistas (sugadintas) ir 2-MV-003 uždarytas, nors turėtų būti uždarytas	Pakeičiama vandens kokybė	Taip
8, 11, 12	2-MCV-007 atidarytas, nors turėtų būti uždarytas	Trikdoma vandens paskirstymo proceso eiga	Ne
9	1-P-006 įjungtas, nors turėtų būti išjungtas	Vamzdžio trūkimas	Taip
10	Sukeliamas 1-MV-001 pažeidimas ir siurblio siurbiamos neapdoroto vandens tekėjimas	Rezervuaro ištuštėjimas	Taip
13	Sumažinamas slėgio siurblio nustatymas	Trikdoma vandens paskirstymo proceso eiga	Taip
14	Nutraukiamas cheminių medžiagų dozavimo siurblių veikimas	Pakeičiama vandens kokybė	Taip
15	AIT-001 rodmuo yra pakeistas (sugadintas)	Rezervuaro perpildymas	Taip

22 pav. Atakų WADI sistemos procese sąrašas [36]

2.2. Duomenų rinkinių apdorojimas

Kadangi SWaT bei WADI yra tiesiogiai iš realių sistemų jutiklių bei pavarų reikšmių sudaryti duomenų rinkiniai, prieš šiuos duomenis pateikiant modelių apmokymui, reikia atsižvelgti į: triukšmus duomenyse, skirtingas jutiklių matavimo skales, dubliuojamus (atsarginius) jutiklius. Dėl šių priežasčių, duomenų apdorojimo etape, prieš duomenis naudojant modelių apmokymui buvo imtasi apdorojimo žingsnių. Duomenų rinkiniai buvo apdoroti naudojant programinį kodą, kuris nurodytas darbo prieduose.

SWaT duomenų rinkinys originaliai sudarytojų yra suskirstytas į dvi, apmokymo ir testavimo dalis, atitinkamai „SWaT_Dataset_Normal_v1.xlsx“ ir „SWaT_Dataset_Attack_v0.xlsx“. Apmokymo

duomenys neturi anomalijų, o testavimo duomenys yra su atakomis. Visų pirma duomenų rinkinių apdorojimo metu, tiek apmokymo tiek testavimo duomenų rinkiniuose buvo pašalinti anomalijų aptikimui nereikšmingi duomenys – pašalinta pirmoji duomenų rinkinio eilutė, kadangi ji nurodo formataavimo informaciją, o ne proceso matavimus. Taip pat, kadangi duomenys vėliau bus dalinami į slenkančius laiko langus, pašalinamas pirmas duomenų stulpelis su matavimų laiko duomenimis, bei paskutinė testavimo duomenų rinkinio eilutė, su anomalijų žymėjimais. Anomalijų žymėjimai buvo perkelti į atskirą failą, kur anomalijai priskirta 1 reikšmė, o normaliems duomenis 0 reikšmė. Taip pat, SWaT duomenų rinkinyje buvo pašalinti pavarų duomenys, kurie viso duomenų rinkinio metu nesikeitė – buvo konstantos. Tokie duomenys neturi įtakos duomenų skirstiniui ir tik papildomai apkrauna, bei klaidina mašininio mokymo modelius. Todėl atitinkamai buvo pašalinti *P102, P201, P202, P204, P206, P401, P403, P404, P502, P601, P603* pavarų – vandens pompų duomenys, kurios viso duomenų surinkimo metu buvo tik įjungtos arba išjungtos ir jų būsenos nesikeitė.

WADI duomenų rinkinys, kaip ir SWaT yra padalintas į apmokymo ir testavimo dalis, atitinkamai „WADI_14days_new.csv“ ir „WADI_attackdataLABEL.csv“. Panašiai kaip ir SWaT duomenų rinkinyje, pašalinti eilučių numeriai, datos ir laiko žymėjimai, kurie nėra reikšmingi sistemos proceso veikimui. Taip pat testavimo duomenų faile pašalintas paskutinis duomenų stulpelis su anomalijų žymekliais ir kaip ir SWaT duomenų rinkinyje, perkelti į atskirą anomalijų žymėjimo failą. Kadangi WADI yra didelis duomenų rinkinys su daug jutiklių ir pavarų, kaip nurodo duomenų rinkinio sudarytojai, dėl klaidų ir dingusių signalų duomenų surinkimo metu yra trūkstamų duomenų. Todėl apdorojimo metu pašalinti keturių jutiklių ir pavarų duomenys, atitinkamai - *2_LS_001_AL, 2_LS_002_AL, 2_P_001_STATUS* ir *2_P_002_STATUS*.

4 lentelė. Duomenų rinkinių imtys, po apdorojimo.

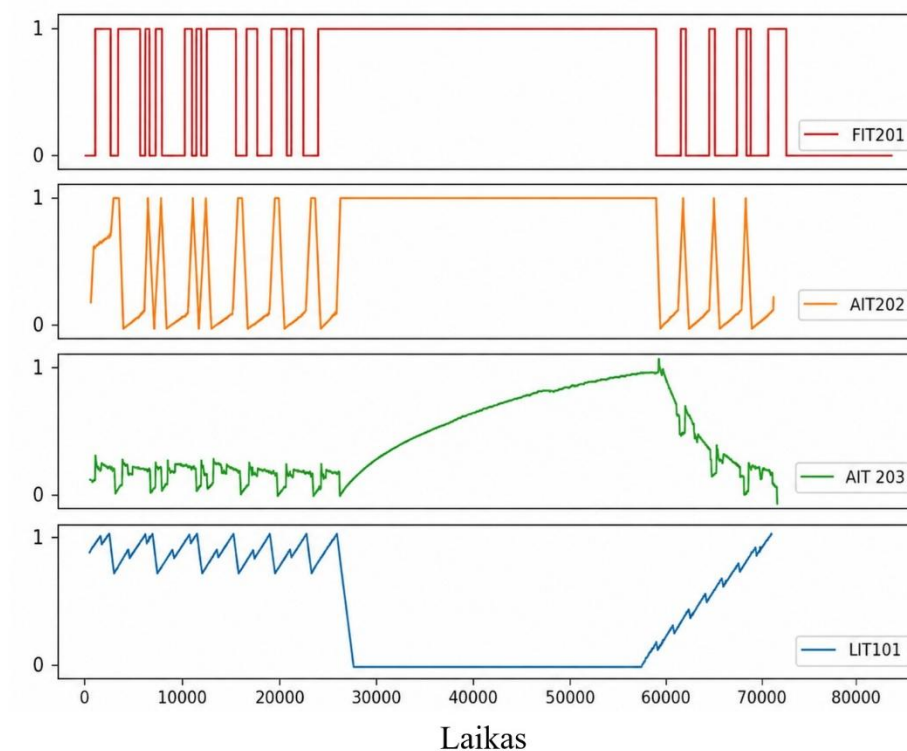
Duomenų rinkinys	Apmokymo duomenų imtis	Testavimo duomenų imtis	Jutiklių, pavarų skaičius	Anomalijų dalis (%)
SWaT	496800	449919	40	11.98
WADI	1048571	172801	123	5.99

Po duomenų valymo ir anomalijų žymių perkėlimo, jutiklių ir pavarų duomenys buvo normalizuoti, naudojant „MinMaxScaler“ iš „sklearn“ bibliotekos. Šis metodas normalizuoja visas jutiklių ir pavarų vertes į intervalą nuo 0 iki 1.

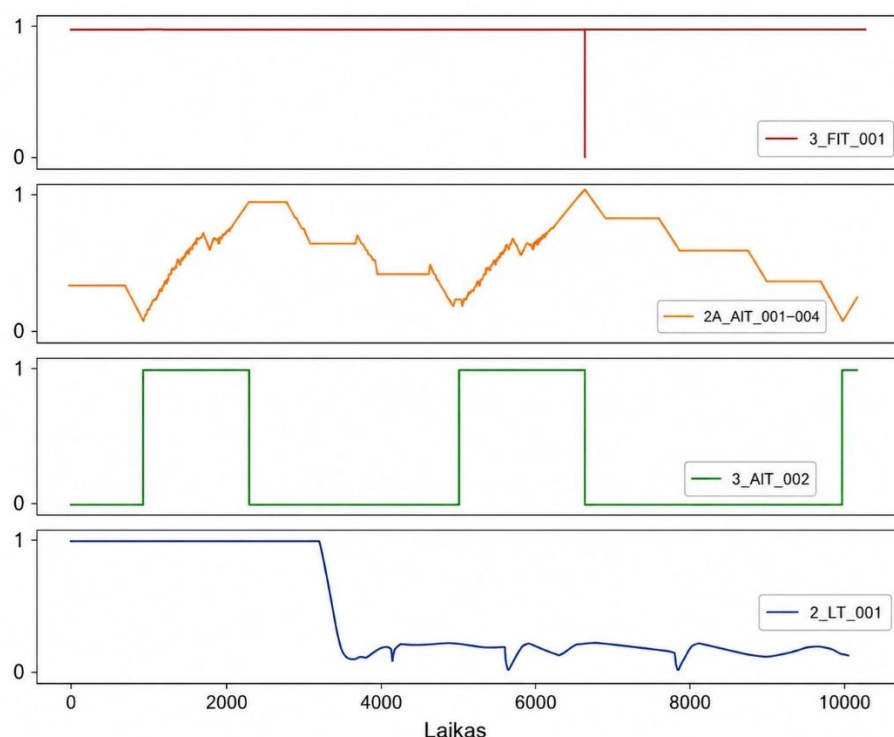
$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad (13)$$

čia x yra originali jutiklio ar pavaros duomenų vertė, x_{min} yra minimali vertė, o x_{max} yra maksimali vertė. Po normalizacijos proceso, visos jutiklių ir pavarų vertės yra $[0,1]$ intervale.

Po to, tiek apmokymo, tiek testavimo duomenų rinkiniai ir failai su anomalijų žymėjimais konvertuojami ir išsaugomi „NumPy“ masyvo formatu (.npy), siekiant užtikrinti efektyvų duomenų saugojimą ir spartesnę įkėlimą bei apdorojimą modelių mokymo bei optimizavimo metu.



23 pav. Grafiškai atvaizduotų SWAT duomenų iškarpa po apdorojimo



24 pav. Grafiškai atvaizduotų WADI duomenų iškarpa po apdorojimo

Kadangi, kaip nagrinėta literatūros analizės 1.2 skyriuje anomalijos pramoniniuose procesuose būna įvairių tipų, o ne vien taškinės, todėl duomenis mašininio mokymo modeliams reikia pateikti ne taškais, o suskirstytus į tam tikro ilgio sekų langus. Šis žingsnis įvykdomas modelių apmokymo metu, kadangi skirtingiems anomalijų aptikimo modeliams parenkami skirtingi sekų langų ilgiai, kurie nustatomi optimizavimo metu.

2.3. Realizuoti anomalijų aptikimo modeliai

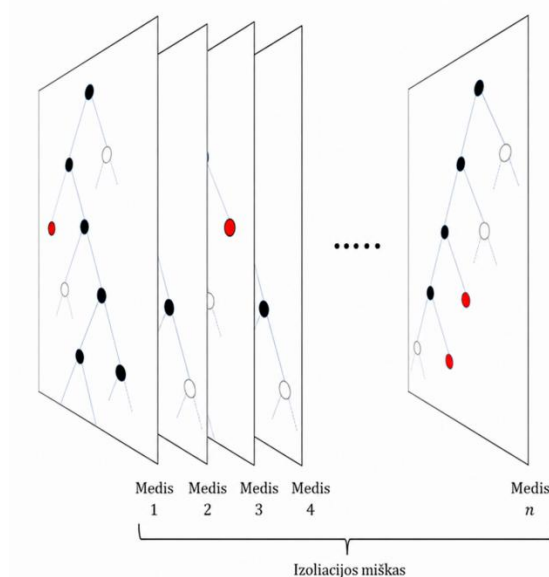
Literatūros analizės dalyje išanalizavus mašininio mokymo modelius, naudojamus anomalijų aptikimo srityje, parinkti penki tinkamiausi metodai anomalijų aptikimo uždaviniui pramoninio proceso, o tiksliau – vandens valymo ir skirstymo sistemos duomenyse. Atitinkamai realizuoti izoliacinio miško, K-vidurkių, auto-enkoderio, variacinio auto-enkoderio, ir USAD modeliai.

Minėtų anomalijų aptikimo modelių realizavimui, jų apmokymui, optimizavimui, o taip pat jų efektyvumo vertinimo metrikoms gauti naudojama „Microsoft Visual Studio Code“ aplinka, bei „Python“ programavimo kalba. Ši programa, leidžia redaguoti programinį kodą, dirbti su įvairiomis programavimo kalbomis ir duomenų analizės įrankiais. Ši programa turi automatinio kodo užbaigimo, klaidų diagnostikos funkcijas bei integruotą terminalą, kas leidžia patogiai vykdyti bei redaguoti „Python“ kodo dalis tiesiai iš terminalo lango. Naudojant šią aplinką bei įrašius „Python“ plėtinį nesunkiai galima integruoti bibliotekas, naudojamas mašininio mokymo modeliams realizuoti, tokias kaip „Numpy“, „Pandas“, „Scikit-learn“, „Tensorflow“, „Pytorch“. Modelių programiniai kodai pateikti priedų skyriuje.

2.3.1. Izoliacijos miško modelis

Kaip apžvelgta 1.3.7 skyriuje, izoliacijos miško modelis yra neprižiūrimas mašininio mokymo metodas, naudojamas aptikti anomalijas. Šis modelis yra ansamblinis regresorius, apimantis keletą izoliacijos medžių. Kiekvienas medis apmokomas naudojant atsitiktinius apmokymo duomenų sekų langus. Pagrindiniai izoliacijos miško modelio parametrai yra tokie:

1. Medžių skaičius ($n_{estimators}$).
2. Maksimalus stebėjimų skaičius (m_{max}), reiškiantis duomenų sekų, naudojamų kiekvienam medžiui apmokyti skaičių.
3. Maksimalus požymių skaičius (n_{max}), nustatantis parinktų požymių rinkinį, naudojamų apmokyti kiekvienam medžiui.



25 pav. Izoliacijos miško modelio architektūra

Naudojamas „Scikit-learn“ bibliotekos izoliacijos miško modelis išveda -1,0 ir 1 reikšmes, kurios atitinkamai reiškia:

5 lentelė. Izoliacijos miško modelio išvestys ir jų reikšmės.

Modelio išvestis	Reikšmė
-1	Didelė tikimybė, kad yra duomenys yra anomalija
1	Duomenys nėra anomalija
0	Duomenys yra netoli anomalijos ribos ir galimai yra anomalija.

Šiuo atveju dėl mažo 0 išvesčių skaičiaus ir metodo supaprastinamo vertinamos tik -1 ir 1 reikšmės.

Modelis naudoja izoliacijos principą, kad atskirtų anomalijos, kai kiekvienas izoliacijos medis atlieka rekursyvų dvejetainį padalijimą atsitiktiniam duomenų sekų pogrupiui, atsitiktinai parinkdamas padalijimo požymį ir jo padalijimo vertę (kuri įeina į nurodytą intervalą). Kiekvienas padalijimas sukuria mazgą, kuris gali būti vidinis, jei atitinkamuose padalijimo regionuose yra galimi tolesni padalijimai, arba išorinis mazgas, reiškiantis, kad tai yra paskutinis šakos mazgas, kai to regiono duomenų pogrupio dydis yra 1 arba pasiekiamas maksimalus medžio gylis. Sekos su anomalijomis skiriasi nuo įprastų duomenų sekų ir jas galima lengvai išskirti. Todėl tikėtina, kad jos bus arčiau medžių šaknies ir dėl to turės trumpesnę atstumą. Anomalijų aptikimo įvertis tam tikrai duomenų imčiai skaičiuojamas pagal (9) formulę.

Pradiniai modelio parametrai pateikti lentelėje žemiau.

6 lentelė. Izoliacijos miško modelio pradiniai parametrai.

Parametras	Reikšmė	Apibūdinimas
window_size	30	Duomenų sekos ilgis
n_estimators	100	Izoliacijos medžių skaičius
contamination	0.03	Numatomas anomalijų santykis
random_state	42	Atsitiktinių operacijų pakartojamumas
max_samples	256	Vienam medžiui naudojamas atsitiktinis pogrupio dydis
max_features	1	Atsitiktinai atrinktų požymių skaičius vienam medžiui

2.3.2. K-vidurkių modelis

Antrasis naudotas mašininio mokymo modelis anomalijoms aptikti – K-vidurkių modelis. Tai dar vienas neprižiūrimas modelis, neretai naudojamas duomenų grupavimui bei anomalijų aptikimui. Modelis siekia suskirstyti duotų duomenų sekas į K grupių, kurių skaičius nustatomas iš anksto, taip, kad kiekviena duomenų imtis priklausytų artimiausiam centroidui. Kaip ir izoliacinio miško modeliui, duomenys buvo padalinti į laiko eilutes.

Prieš apmokant modelį, būtent šiam modeliui, duomenims taikomas papildomas apdorojimo žingsnis – principinių komponentų analizė. Šis metodas taikytas siekiant išvengti su K-vidurkių modeliu siejamo „dimensijų prakeiksmo“, kai modelis apmokomas didelės apimties duomenų rinkiniais. Įvykus šiam fenomenui, ženkliai suprastėja modelio efektyvumas bei anomalijų aptikimo kokybė.

Šiame K-vidurkių modelyje anomalijos aptinkamos apskaičiuojant Euklidinį atstumą tarp kiekvienos duomenų sekos ir artimiausio centroido. Duomenų sekos, esančios toliausiai nuo centroidų, buvo laikomos turinčiomis anomalijų. Euklidinis atstumas apskaičiuojamas pagal šią formulę:

$$d(x, \mu) = \sqrt{\sum_{i=1}^n (x_i - \mu_i)^2}, \quad (14)$$

čia x – duomenų seka, μ – centroidas.

Pradinis anomalijų santykis (angl. *contamination*) naudotas 0.05, todėl 5% sekų, su apskaičiuotu didžiausiu atstumu iki centroidų, buvo pažymėtos kaip anomalijos. Naudojamas „Scikit-learn“ bibliotekos K-vidurkių modelis. Pradiniai modelio parametrai pateikti lentelėje žemiau.

7 lentelė. K-vidurkių modelio pradiniai parametrai.

Parametras	Reikšmė	Apibūdinimas
window_size	30	Duomenų sekos ilgis
n_clusters	3	Klasterių skaičius
contamination	0.05	Numatomas anomalijų santykis
random_state	42	Atsitiktinių operacijų pakartojamumas
n_init	10	Pradinių centroidų skaičius
max_iter	300	Maksimalus iteracijų skaičius
distance_metric	Euclidean	Naudojama atstumo metrika

2.3.3. Auto-enkoderio modelis

Toliau realizuotas auto-enkoderio modelis. Kaip nagrinėta 1.3.4 skyriuje, modelio veikimas pagrįstas neprižiūrimos rekonstrukcijos principu. Šiuo atveju modeliui naudota „PyTorch“ bibliotekos „TransformerBasicBottleneckScalingNoAct“ enkoderio bei dekoderio architektūra. Skirtingai nei rekurentiniai auto-enkoderiai, transformatoriniai auto-enkoderiai labiau tinkami anomalijų aptikimui pramoniniuose procesuose, nes duomenų įvestis atkuria tiksliau, kadangi dekoderis gali matyti enkoderio būseną, per tam tikrą žingsnių skaičių. Tačiau šis rekonstravimo palengvinimas gali reikšti tai, kad modelis neišmoks visų reikšmingų duomenų latentinio atvaizdavimo.

Dėl šios priežasties modelio architektūroje įvedamas ribojimas, dekoduoiant vieną fiksuotą atvaizdą:

$$z = \frac{1}{T} \sum_{i=1}^T h_i, \quad (15)$$

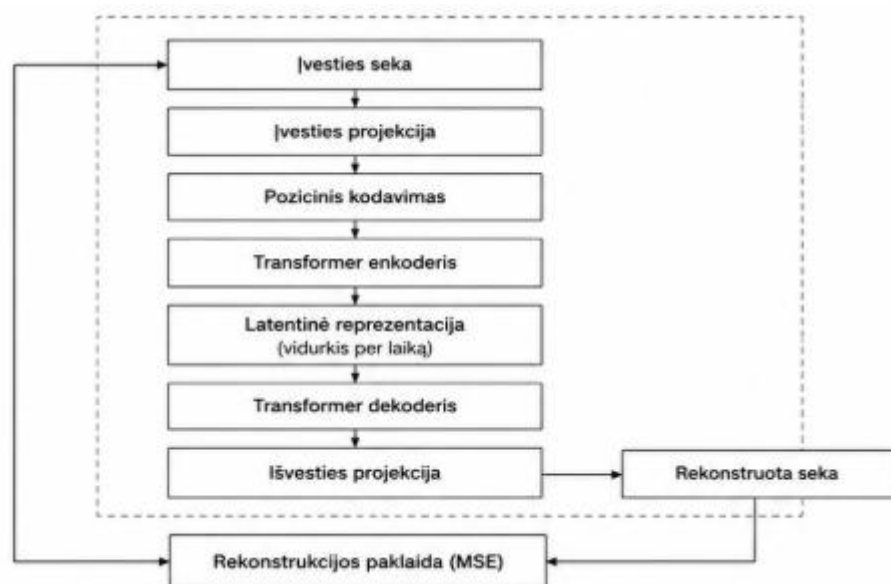
čia $[h_1, h_2, \dots, h_T] = f^{enc} \varphi([e_1, e_2, \dots, e_T])$ yra enkoderio išėjimai. Taigi, dekoderis turi tiesioginį priėjimą prie serijų atvaizdo z , o ne kontekstinės reprezentacijos h_i kiekviename enkoderio laiko žingsnyje.

Taip pat modelyje įvestiems sekoms naudojamas pozicinis kodavimas. Kadangi transformatoriniai modeliai neturi rekurentinių sluoksnių, jie savaime nesupranta reikšmių sekoje eiliškumo. Todėl kiekvienam įvesties elementui papildomai priskiriamas pozicinis vektorius.

Anomalijų aptikimas atliekamas naudojant rekonstrukcijos paklaidą. Kiekvienam sekų langui priskiriama rekonstrukcijos paklaida:

$$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2, \quad (16)$$

čia x_i – originali reikšmė, o $\hat{x}_i$ – rekonstruota reikšmė.



26 pav. Naudojamo auto-enkoderio modelio architektūra

8 lentelė. Auto-enkoderio modelio pradiniai parametrai.

Parametras	Reikšmė	Apibūdinimas
window_size	100	Duomenų sekos ilgis
batch_size	64	Vienu metu apdorojamų sekų langų skaičius
learning_rate	0.001	Modelio mokymosi greitis
epochs	60	Apmokymo epochų skaičius
weight_decay	0	Reguliarizacijos koeficientas
features	51-123	Naudojamų požymių skaičius
scale	16	Vidinės transformatorių dimensijos mastelio koeficientas

d_model	816	„Transformer“ vidinės reprezentacijos dimensija
nhead	51-123	„Multi-head attention“ skaičius
encoder_layers	1	Enkoderio sluoksnių skaičius
decoder_layers	1	Dekoderio sluoksnių skaičius
dim_feedforward	256	„Feed-forward“ tinklo dimensija
dropout	0.1	„Dropout“ reguliacijos koeficientas
positional_encoding	Sinusoidinis	Pozicinio sekų kodavimo metodas
loss_function	MSE	Rekonstrukcijos paklaidos funkcija
anomaly_score	Reconstruction Error	Svorių inicializavimo metodas
threshold_method	Validation Threshold	Slenkstis nustatomas naudojant validavimo duomenis

Modelio mokymo metu naudojamas „Adam“ optimizatorius.

2.3.4. Variacinio auto-enkoderio modelis

Šioje dalyje realizuotas variacinis auto-enkoderio modelis. Skirtingai nuo standartinio auto-enkoderio, variacinio auto-enkoderio modelis mokosi ne vienos fiksuotos latentinės reprezentacijos, bet tikimybinio latentinio pasiskirstymo, skirtų geriau modeliuoti normalių duomenų variacijas. Plačiau algoritmo veikimas nagrinėtas 1.3.5 skyriuje. Modelio įvesties duomenims taip pat naudoti duomenų sekų langai. Kaip ir auto-enkoderio modeliui, naudota „PyTorch“ biblioteka, o tiksliau „VAE_scale_no_act“ modelis.

Įvestiems sekoms naudojamas taip pat naudojamas pozicinis kodavimas. Modelio enkoderio tinklas generuoja latentines sekų reprezentacijas, kurios vėliau transformuojamos į vidurkio vektorių ir logaritminės dispersijos vektorių. Tokiu būdu aprašomas latentinių požymių tikimybinis skirstinys:

$$z \sim N(\mu, \sigma^2). \quad (17)$$

Latentinis vektorius sugeneruojamas, naudojant perparametrizavimo metodą:

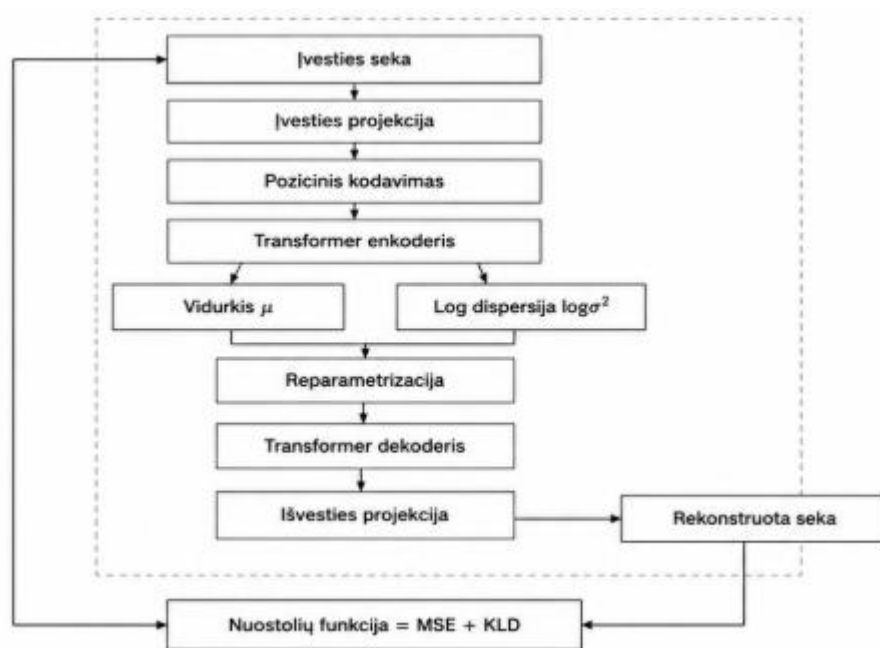
$$z = \mu + \epsilon * \sigma, \quad (18)$$

čia $\epsilon \sim N(0, I)$, $\sigma = e^{0.5 * \log var}$.

Dekoderio tinklas rekonstruoja originalią įvesties seką iš sugeneruoto latentinio vektoriaus. Kaip ir auto-enkoderio modelyje, rekonstrukcijos kokybė vertinama naudojant vidutinę kvadratinę paklaidą (16 formulė), tačiau šiame modelyje papildomai naudojamas *Kullbeka-Leiblerio* nuokrypis, kuris reguliuoja latentinę erdvę:

$$KLD = -\frac{1}{2} \sum (1 + \log(\sigma^2) - \mu^2 - \sigma^2). \quad (19)$$

Taigi, galutinė modelio nuostolių funkcija apskaičiuojama sudėjus vidutinę kvadratinę paklaidą ir *Kullbeka-Leiblerio* nuokrypį.



27 pav. Naudojamo variacinio auto-enkoderio modelio architektūra

9 lentelė. Variacinio auto-enkoderio modelio pradiniai parametrai.

Parametras	Reikšmė	Apibūdinimas
window_size	100	Duomenų sekos ilgis
batch_size	64	Vienu metu apdorojamų sekų langų skaičius
learning_rate	0.0005	Modelio mokymosi greitis
epochs	50	Apmokymo epochų skaičius
weight_decay	0	Reguliarizacijos koeficientas
features	51-123	Naudojamų požymių skaičius
scale	16	Vidinės transformatorių dimensijos mastelio koeficientas
d_model	816	„Transformer“ vidinės reprezentacijos dimensija
nhead	51-123	„Multi-head attention“ skaičius
encoder_layers	1	Enkoderio sluoksnių skaičius
decoder_layers	1	Dekoderio sluoksnių skaičius
dim_feedforward	256	„Feed-forward“ tinklo dimensija
dropout	0.1	„Dropout“ reguliacijos koeficientas
positional_encoding	Sinusoidinis	Pozicinio sekų kodavimo metodas
reconstruction_loss	MSE	Rekonstrukcijos paklaidos funkcija
anomaly_score	Reconstruction Error	Svorių inicializavimo metodas

threshold_method	Validation Threshold	Slenkstis nustatomas naudojant validavimo duomenis
reparametrization	Gaussian Sampling	Latentinio vektoriaus generavimo metodas
regularization_loss	KLD	Nuokrypio funkcija

Modelio mokymo metu naudojamas „Adam“ optimizatorius.

2.3.5. USAD modelis

USAD taip pat yra neprižiūrimas mašininio mokymo metodas, sudarytas iš dviejų dekoderių ir vieno bendro enkoderio (plačiau bendras modelio veikimas bei sandara apžvelgtas 1.3.8 skyriuje). Modelio enkoderiui ir dekoderiams naudojama pilnai sujungtų sluoksnių (angl. *fully connected layers*) architektūra, kuriuose atliekamas laipsniškas požymių dimensijos mažinimas. Tokiu būdu modelis išmoksta svarbiausius normalių procesų latentinius požymius. Enkoderio ir dekoderių struktūros nurodytos 10 ir 11 lentelėse. Abiem dekoderiams naudojama tokie patys sluoksniai ir struktūra.

10 lentelė. USAD enkoderio struktūra.

Sluoksnis	Iėjimo dimensija	Išėjimo dimensija	Aktyvacija
FC1	input_size	input_size/2	ReLU
FC2	input_size/2	input_size/4	ReLU
FC3	input_size/4	latent_size	ReLU

11 lentelė. USAD dekoderių struktūra.

Sluoksnis	Iėjimo dimensija	Išėjimo dimensija	Aktyvacija
FC1	latent_size	input_size/4	ReLU
FC2	input_size/4	input_size/2	ReLU
FC3	input_size/2	input_size	Sigmoid

Paskutiniame dekoderių sluoksnyje naudojama „Sigmoid“ aktyvacijos funkcija, kadangi visi SWaT ir WADI duomenų rinkinių duomenys apdorojimo etape buvo normalizuoti į intervalą [0;1]. Todėl šis sluoksnis užtikrina, kad modelio išvestys išliktų tokia pačia intervale kaip ir originalūs duomenys. Modelyje naudojami dekoderiai atlieka skirtingas funkcijas, pirmasis dekoderis rekonstruoja originalias duomenų sekas, o antrasis papildomai rekonstruoja pirmojo dekoderio rekonstrukcijas ir yra naudojamas priešpriešiniam mokymui. Todėl naudojamos dvi nuostolių funkcijos. Pirmasis auto-enkoderis minimizuoja rekonstrukcijos paklaidą:

$$L_{AE1} = \frac{1}{n} \left| x - \hat{x}_1 \right|^2 + \left(1 - \frac{1}{n} \right) \left| x - \hat{x}'_2 \right|^2, \quad (20)$$

o antrasis auto-enkoderis priešpriešinio principu didina skirtumą tarp rekonstrukcijų:

$$L_{AE2} = \frac{1}{n} \left| x - \hat{x}_2 \right|^2 + \left(1 - \frac{1}{n} \right) \left| x - \hat{x}'_2 \right|^2. \quad (21)$$

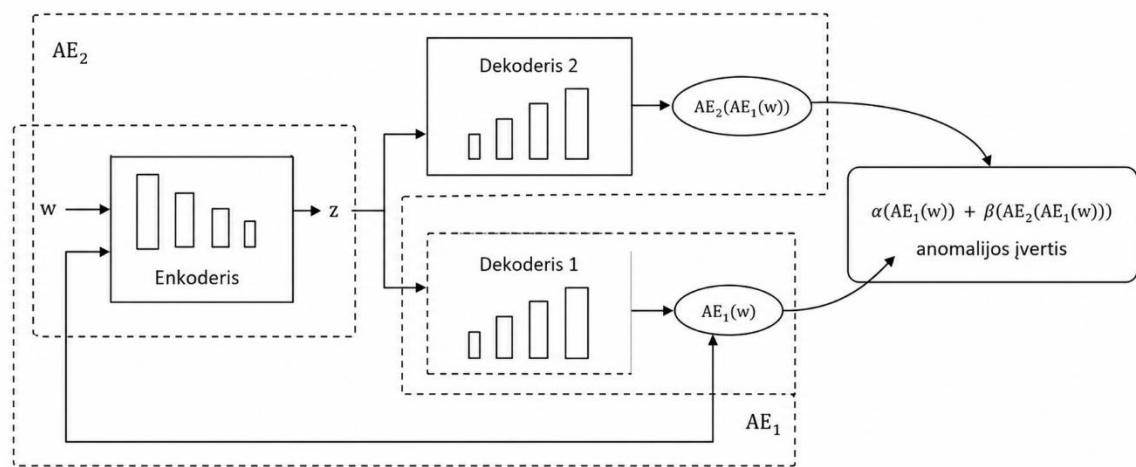
Čia x – originali duomenų seka, $\hat{x}_1$ - pirmojo dekoderio rekonstrukcija, $\hat{x}_2$ - antrojo dekoderio rekonstrukcija, $\hat{x}'_2$ - papildoma antrojo dekoderio rekonstrukcija, n – epochos numeris.

Modelio apmokymo metu naudojami du vienas nuo kito nepriklausomi „Adam“ optimizatoriai. Pirmasis optimizuoja enkoderio ir pirmojo dekoderio parametrus, antrasis taip pat optimizuoja enkoderio bei antrojo dekoderio parametrus.

Testavimo metu anomalijos balas apskaičiuojamas naudojant dviejų rekonstrukcijos paklaidų svertinį vidurkį:

$$Balas = \alpha * Vidurkis((x - AE1)^2) + \beta * Vidurkis((x - AE2)^2), \quad (22)$$

čia $AE1$ – pirmojo auto-enkoderio rekonstrukcija, $AE2$ – antrojo auto-enkoderio rekonstrukcija, α ir β – svorio koeficientai.



28 pav. Naudojamo USAD modelio architektūra

12 lentelė. USAD modelio pradiniai parametrai.

Parametras	Reikšmė	Apibūdinimas
window_size	12	Duomenų sekos ilgis
features	51-123	Požymių skaičius
input_size	612-1476	Įėjimo dimensija
batch_size	128	Vienu metu apdorojamų sekų langų skaičius
lr	0.02	Modelio mokymosi greitis
nepochs	80	Epochų skaičius
alpha	0.5	Alfa koeficientas
beta	0.5	Beta koeficientas

3. Tyrimo rezultatai

Šiame skyriuje pateikiami gauti modelių efektyvumo rezultatai SWaT bei WADI duomenų rinkiniuose, rezultatai po modelių parametrų optimizavimo, o taip pat modelių efektyvumo rezultatai, naudojant diapazono įverčius.

3.1. Pirminio modelių apmokymo rezultatai

Realizavus visus penkis parinktus mašininio mokymo modelius su pirminiais parametrais, jie buvo apmokyti apdorotais SWaT ir WADI duomenimis. Auto-enkoderio modelis mokomas 60 epochų, variacinio auto-enkoderio modelis apmokomas 50 epochų, o USAD – 80 epochų. Modelių anomalijų aptikimo efektyvumas vertinamas trimis literatūroje dažniausiai anomalijų aptikimo uždaviniais pramoniniuose procesuose vertinti naudojamomis metrikomis – tikslumu, atkūrimu ir F1 įverčiu (plačiau apžvelgta 1.4 skyriuje). Gauti rezultatai pateikiami 13 ir 14 lentelėse.

13 lentelė. Pirminio modelių apmokymo rezultatai SWaT duomenų rinkinyje.

	Izoliacijos miškas	K-vidurkių modelis	Auto-enkoderis	Variacinis auto- enkoderis	USAD
Tikslumas	0.85	0.66	0.65	0.45	0.69
Atkūrimas	0.44	0.48	0.59	0.38	0.75
F1 įvertis	0.58	0.55	0.62	0.41	0.73

14 lentelė. Pirminio modelių apmokymo rezultatai WADI duomenų rinkinyje.

	Izoliacijos miškas	K-vidurkių modelis	Auto-enkoderis	Variacinis auto- enkoderis	USAD
Tikslumas	0.78	0.58	0.60	0.42	0.63
Atkūrimas	0.36	0.41	0.56	0.34	0.68
F1 įvertis	0.49	0.48	0.58	0.37	0.65

Kaip galima matyti iš gautų rezultatų, visi modeliai geriau pasirodė aptinkant anomalijas SWaT duomenų rinkinyje. Tokių rezultatų buvo tikėtasi, kadangi WADI duomenų rinkinys yra gerokai didesnis, bei sudėtingesnis už SWaT (123 jutiklių ir pavarų duomenys WADI, vietoj 40 jutiklių ir pavarų duomenų SWaT). Geriausiai abiejuose duomenų rinkiniuose pasirodė USAD modelis, surinkęs didžiausią F1 įvertį (0.73 SWaT ir 0.65 WADI).

3.2. Optimizuotų modelių efektyvumo rezultatai

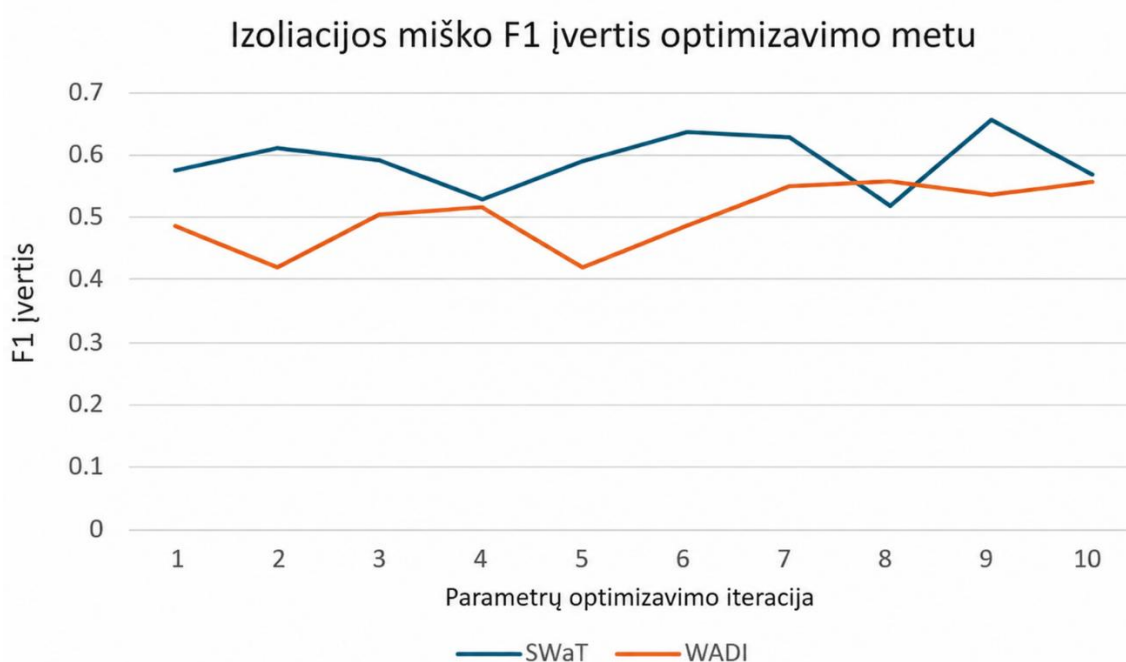
Tolimesnėje tyrimo eigoje buvo optimizuoti modelių parametrai. Kiekvienam modeliui buvo taikomi atitinkami optimizavimo metodai, nustatomi reikšmingi parametrai, bei keičiamos jų reikšmės kol gautos didžiausios F1 įverčio vertės tiek SWaT, tiek WADI duomenų rinkiniuose.

Izoliacijos miško modelis buvo optimizuojamas keičiant šiuos parametrus: medžių skaičių, kiekvienam medžiui naudojamų imčių skaičių, požymių pogrupio dydį, anomalijų proporciją ir slenkančio lango ilgį. Testuotos parametrų reikšmės pateiktos 15 lentelėje.

15 lentelė. Izoliacijos miško modelio parametrų reikšmės optimizavimo metu.

Parametras	Reikšmė	Apibūdinimas
window_size	10, 20, 30, 60	Duomenų sekos ilgis
n_estimators	100, 200, 300, 500	Izoliacijos medžių skaičius
contamination	0.03, 0.05, 0.06, 0.08, 0.10, 0.12	Numatomas anomalijų santykis
max_samples	256, 512, 1024, "auto"	Vienam medžiui naudojamas atsitiktinis pogrupio dydis
max_features	0.5, 0.7, 1.0	Atsitiktinai atrinktų požymių skaičius vienam medžiui

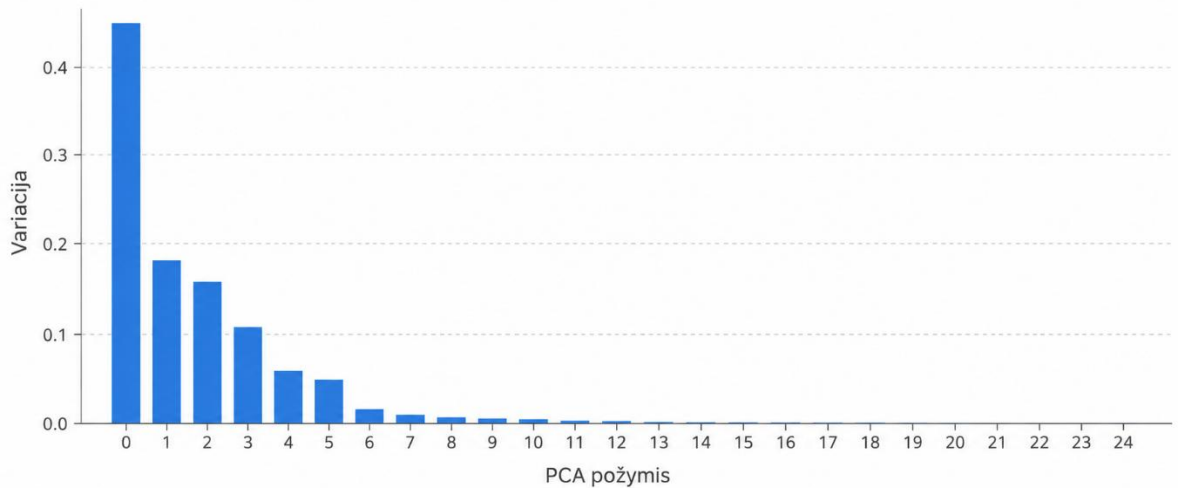
F1 įverčio kitimas modelio optimizavimo metu atvaizduotas 29 paveikslėlyje.



29 pav. Izoliacijos miško modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu

Kaip galima matyti iš paveikslėlio aukščiau, optimizavimo metu geriausias izoliacijos miško modelio F1 įvertis SWaT duomenims gautas 0.66, taikant *window_size*-30, *n_estimators*-100, *contamination*-0.10, *max_samples*-256, *max_features*-1 parametrus. WADI duomenų rinkiniui geriausias F1 įvertis gautas 0.56, taikant *window_size*-60, *n_estimators*-200, *contamination*-0.05, *max_samples*-256, *max_features*-1 parametrus.

Kadangi K-vidurkių modelio duomenims papildomai taikomas principinių komponentių analizės metodas, reikia optimizuoti ir jo parametrus. Tam taikomi tam tikri algoritmai. PCA metodas suskaičiuoja visas galimas principines komponentes ir jų variacijas. Iš to gaunamas variacijos grafikas:



30 pav. Principinių komponentių analizės variacijos grafikas

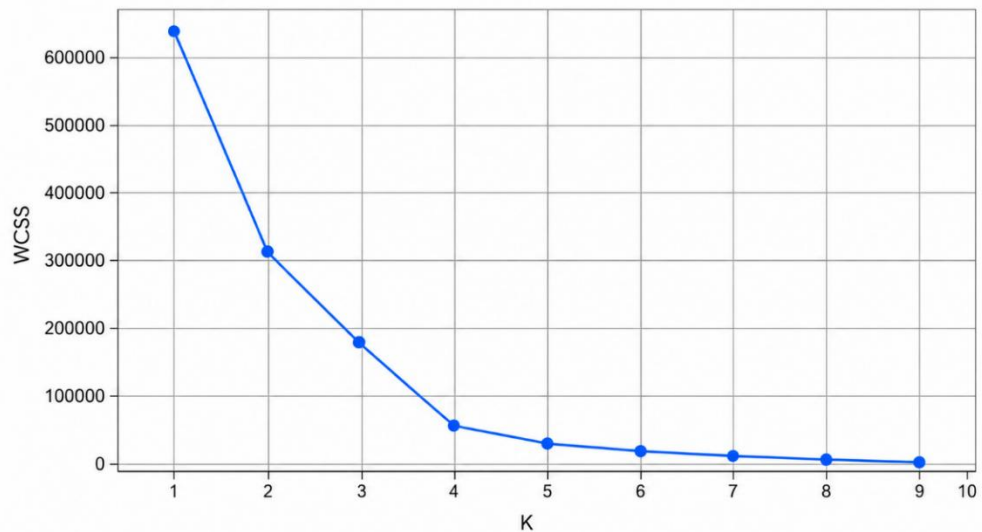
Iš šio grafiko nustatomos principinės komponentės, turinčios reikšmingą variaciją. Kaip matoma iš grafiko, variacinę, didesnę už 0.1 turi tik pirmos trys principinės komponentės, todėl komponentių skaičius parenkamas 3.

Pagal variacijos analizę gautus rezultatus duomenys toliau paverčiami į 3 dimensijų principinių komponentių erdvę ir taikomas „alkūnės metodas“ optimaliam K -vidurkių modelio parametro - K -klasterių skaičiui rasti. Tai yra vizualinis metodas rasti optimaliam K skaičiui, naudojant WCSS parametro (angl. *Within-Cluster Sum of Squares*) reikšmę, kuri nusako bendrą atstumų tarp duomenų taškų ir jų centroidų kvadratų sumą. WCSS gali būti apskaičiuotas pagal šią formulę:

$$WCSS = \sum_{i=1}^K \sum_{x_j \in C_i} ||x_j - \mu_i||^2, \quad (23)$$

čia K - klasterių skaičius, C_i - i -tojo klasterio duomenų imčių aibė, x_j – tam tikra duomenų imtis, μ_i – tam tikro klasterio centroidas.

Didėjant klasterių skaičiui K , WCSS reikšmė mažėja, nes centroidai artėja prie duomenų taškų. Tačiau tam tikru momentu WCSS reikšmės mažėjimas ženkliai sulėtėja. Ši vieta grafike vizualiai sudaro alkūnę, todėl metodas vadinamas „alkūnės metodu“. Optimalus klasterių skaičius parenkamas toje vietoje, kur papildomų klasterių pridėjimas nebesudaro reikšmingo WCSS sumažėjimo. Kiekvienai K reikšmei buvo apskaičiuojamas WCSS parametras ir braižomas alkūnės grafikas. Pagal gautą grafiką, atvaizduotą 30 paveikslėlyje, nustatyta, kad optimalus klasterių skaičius yra 4.



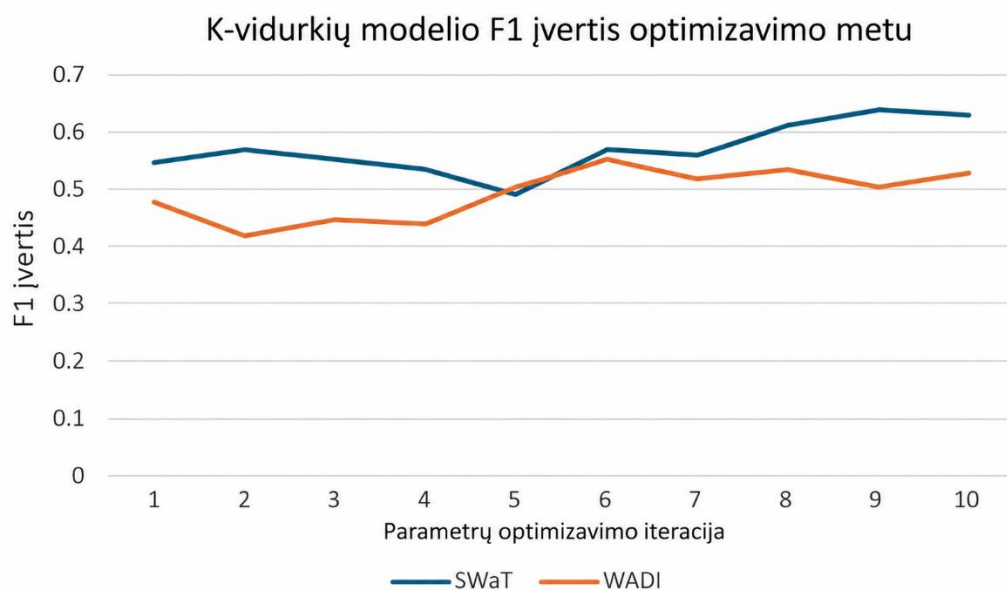
31 pav. Alkūnės metodo grafikas

K-vidurkių modelis toliau optimizuojamas keičiant šiuos parametrus: duomenų sekos ilgį, numatomų anomalijų santykį, pradinių centroidų skaičių, maksimalų iteracijų skaičių. Testuotos parametru reikšmės pateiktos 16 lentelėje.

16 lentelė. Izoliacijos miško modelio parametru reikšmės optimizavimo metu.

Parametras	Reikšmė	Apibūdinimas
window_size	10, 20, 30, 60	Duomenų sekos ilgis
contamination	0.03, 0.05, 0.06, 0.08, 0.10, 0.12	Numatomas anomalijų santykis
n_init	10, 20, 50	Pradinių centroidų skaičius
max_iter	300, 500, 1000	Maksimalus iteracijų skaičius

F1 įverčio kitimas modelio optimizavimo metu atvaizduotas 32 paveikslėlyje.



32 pav. K-vidurkių modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu

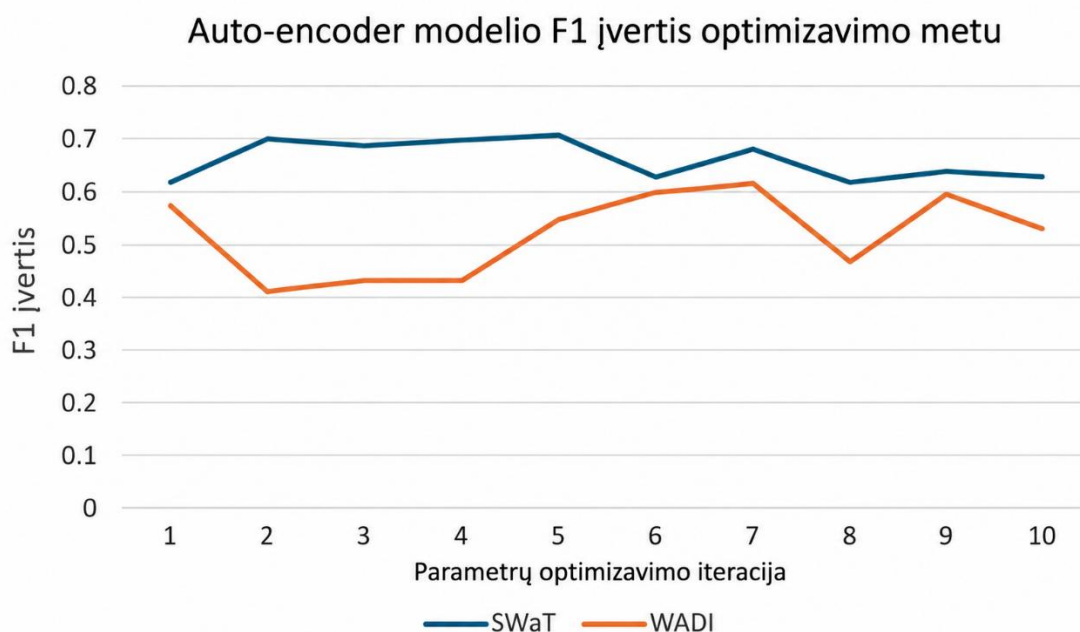
Optimizavimo metu geriausias K-means modelio F1 įvertis SWaT duomenims gautas 0.64, taikant *window_size*-30, *contamination*-0.12, *n_init*-20, *max_iter*-500 parametrus. WADI duomenų rinkiniui geriausias F1 įvertis gautas 0.55, taikant *window_size*-60, *contamination*-0.03, *n_init*-50, *max_iter*-500 parametrus.

Auto-enkoderio modelis buvo optimizuotas keičiant šiuos parametrus: duomenų sekos ilgį, modelio mokymosi greitį, apmokymo epochų skaičių, vidinės transformatorių dimensijos mastelio koeficientą, „Dropout“ reguliacijos koeficientą ir vienu metu apdorojamų sekų kiekį. Testuotos parametų reikšmės pateiktos 17 lentelėje.

17 lentelė. Auto-enkoderio modelio parametų reikšmės optimizavimo metu.

Parametras	Reikšmė	Apibūdinimas
window_size	30, 60, 100, 150	Duomenų sekos ilgis
learning_rate	0.0001, 0.0005, 0.001	Modelio mokymosi greitis
epochs	30, 60, 80, 100	Apmokymo epochų skaičius
scale	4, 8, 16	Vidinės transformatorių dimensijos mastelio koeficientas
dropout	0.05, 0.1, 0.2	„Dropout“ reguliacijos koeficientas
batch_size	32, 64, 128	Vienu metu apdorojamų sekų kiekis

F1 įverčio kitimas modelio optimizavimo metu atvaizduotas 33 paveikslėlyje.



33 pav. Auto-enkoderio modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu

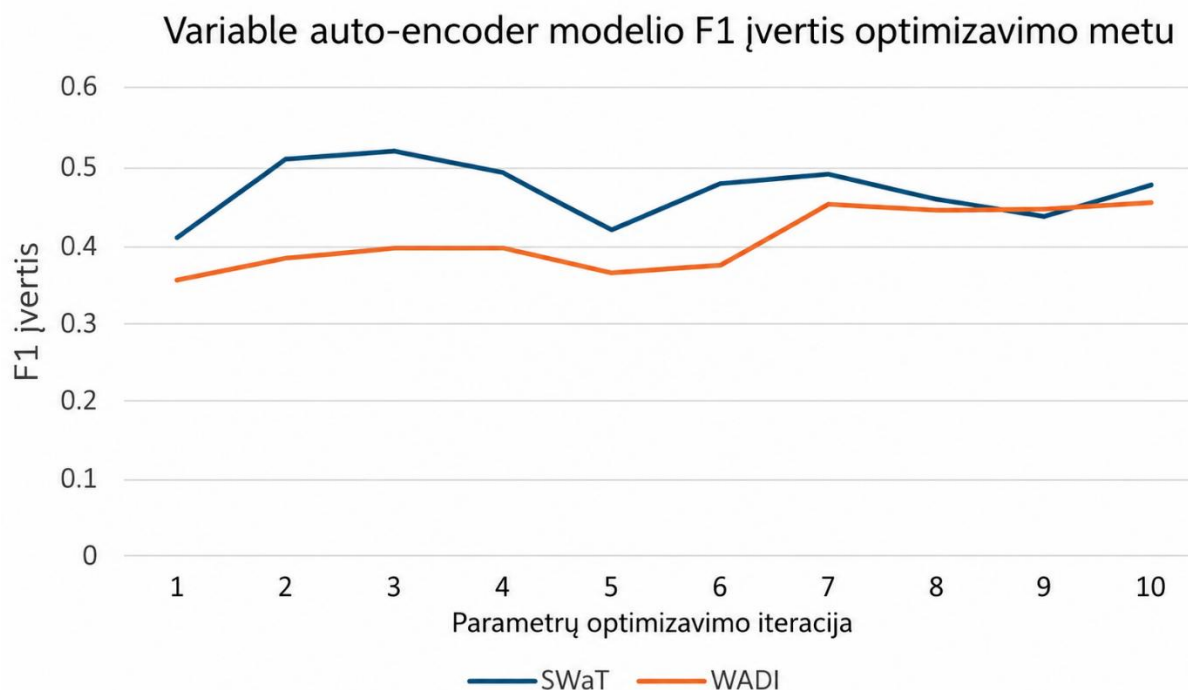
Optimizavimo metu geriausias auto-enkoderio modelio F1 įvertis SWaT duomenims gautas 0.71, taikant *window_size*-100, *learning_rate*-0.001, *epochs*-60, *scale*-16, *dropout*-0.1, *batch_size*-64 parametrus. WADI duomenų rinkiniui geriausias F1 įvertis gautas 0.62, taikant *window_size*-150, *learning_rate*-0.0005, *epochs*-100, *scale*-8, *dropout*-0.2, *batch_size*-32 parametrus.

Variacinio auto-enkoderio modelis optimizuotas parenkant šiuos parametrus: duomenų sekos ilgį, modelio mokymosi greitį, apmokymo epochų skaičių, vidinės transformatorių dimensijos mastelio koeficientą, „Dropout“ reguliacijos koeficientą ir vienu metu apdorojamų sekų kiekį, transformatoriaus vidinės reprezentacijos dydį, KLD įtakos koeficientą. Testuotos parametų reikšmės pateiktos 18 lentelėje.

18 lentelė. Variacinio auto-enkoderio modelio parametų reikšmės optimizavimo metu.

Parametras	Reikšmė	Apibūdinimas
window_size	30, 60, 100, 150	Duomenų sekos ilgis
learning_rate	0.0001, 0.0005, 0.001	Modelio mokymosi greitis
epochs	30, 60, 80, 100	Apmokymo epochų skaičius
scale	4, 8, 16	Vidinės transformatorių dimensijos mastelio koeficientas
dropout	0.05, 0.1, 0.2	„Dropout“ reguliacijos koeficientas
batch_size	32, 64, 128	Vienu metu apdorojamų sekų kiekis
d_model	1x, 8x, 16x	Transformatoriaus vidinės reprezentacijos dydis
KLD_weight	0.001, 0.01, 0.1, 1.0	KLD įtaka bendrai nuostolių funkcijai

F1 įverčio kitimas modelio optimizavimo metu atvaizduotas 34 paveikslyje.



34 pav. Variacinio auto-enkoderio modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu

Optimizavimo metu geriausias auto-enkoderio modelio F1 įvertis SWaT duomenims gautas 0.52, taikant *window_size*-100, *learning_rate*-0.0005, *epochs*-80, *scale*-16, *dropout*-0.1, *batch_size*-64, *d_model*-16x, *KLD_weight*-0.1 parametrus. WADI duomenų rinkiniui geriausias F1 įvertis gautas

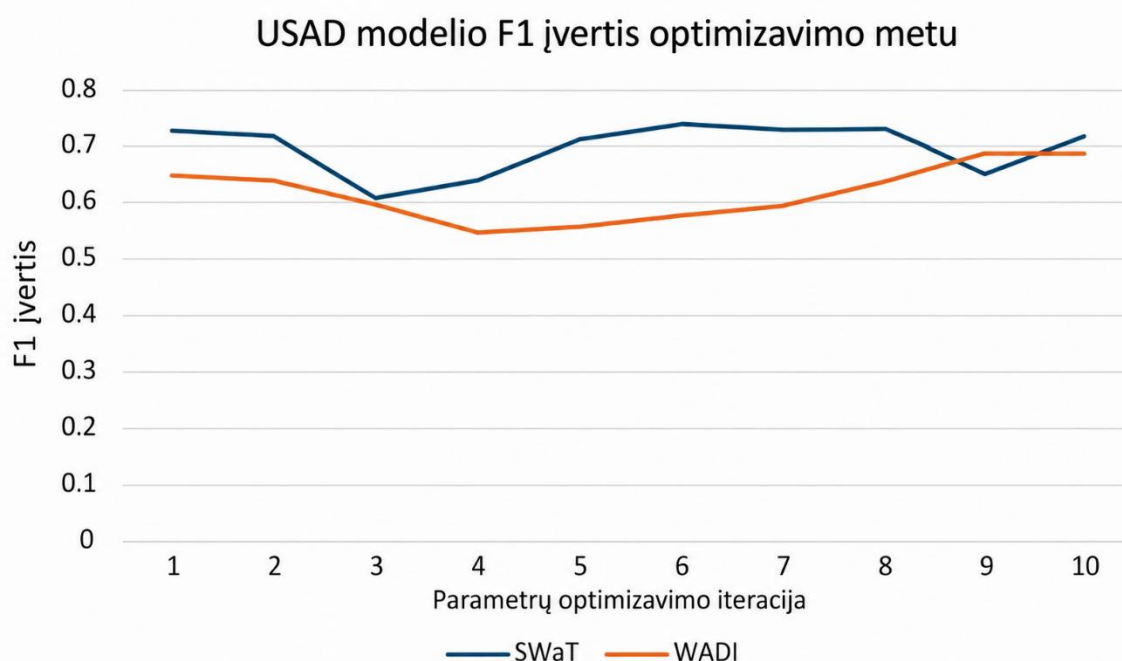
0.47, taikant *window_size*-150, *learning_rate*-0.0005, *epochs*-100, *scale*-8, *dropout*-0.2, *batch_size*-32, *d_model*-8x, *KLD_weight*-0.01 parametrus.

Toliau USAD modelis optimizuotas keičiant šiuos parametrus: duomenų sekos ilgį, vienu metu apdorojamų sekų langų skaičių, modelio mokymosi greitį, epochų skaičių, „Alfa“ koeficientą ir „Beta“ koeficientą. Testuotos parametų reikšmės pateiktos 19 lentelėje.

19 lentelė. USAD modelio parametų reikšmės optimizavimo metu.

Parametras	Reikšmė	Apibūdinimas
window_size	12, 20, 30, 60	Duomenų sekos ilgis
batch_size	64, 128, 256	Vienu metu apdorojamų sekų langų skaičius
lr	0.001, 0.005, 0.01, 0.02	Modelio mokymosi greitis
nepochs	80, 150, 300	Epochų skaičius
alpha	0.3, 0.5, 0.7	Alfa koeficientas
beta	0.3, 0.5, 0.7	Beta koeficientas

F1 įverčio kitimas modelio optimizavimo metu atvaizduotas 35 paveikslėlyje.



35 pav. USAD modelio F1 įverčio reikšmės kitimas modelio optimizavimo metu

Optimizavimo metu geriausias USAD modelio F1 įvertis SWaT duomenims gautas 0.74, taikant *window_size*-12, *batch_size*-128, *lr*-0.005, *nepochs*-150, *alpha*-0.5, *beta*-0.5 parametrus. WADI duomenų rinkiniui geriausias F1 įvertis gautas 0.69, taikant *window_size*-30, *batch_size*-128, *lr*-0.005, *nepochs*-150, *alpha*-0.3, *beta*-0.7 parametrus.

Taigi tyrimo metu buvo sėkmingai optimizuoti visų penkių mašininio mokymo modelių parametrai bei gauti nauji tikslumo, atkūrimo bei F1 įverčiai. Jie pateikti 20 ir 21 lentelėse.

20 lentelė. Optimizuotų modelių įverčiai SWaT duomenų rinkinyje.

	Izoliacijos miškas	K-vidurkių modelis	Auto-enkoderis	Variacinis auto- enkoderis	USAD
Tikslumas	0.78	0.70	0.74	0.58	0.71
Atkūrimas	0.58	0.59	0.68	0.47	0.77
F1 įvertis	0.66	0.64	0.71	0.52	0.74

21 lentelė. Optimizuotų modelių įverčiai WADI duomenų rinkinyje.

	Izoliacijos miškas	K-vidurkių modelis	Auto-enkoderis	Variacinis auto- enkoderis	USAD
Tikslumas	0.68	0.62	0.66	0.52	0.66
Atkūrimas	0.48	0.49	0.58	0.43	0.72
F1 įvertis	0.56	0.55	0.62	0.47	0.69

Kaip galima matyti iš lentelių aukščiau, po parametrų optimizavimo pagal F1 įvertį pagerėjo kone visų modelių tikslumo, atkūrimo ir F1 balai, išskyrus izoliacijos miško modelį, kurio tikslumas sumažėjo, bet padidėjo atkūrimas, dėl ko padidėjo ir F1 balas. Tai reiškia kad modelis aptiko daugiau anomalijų, bet ir daugiau normalių duomenų pažymėjo kaip anomalijas.

Nors po optimizavimo mažiausiai pasikeitė USAD modelio efektyvumo įverčiai, šis modelis vis tiek turėjo didžiausius F1 įverčius tiek prieš, tiek po parametrų optimizavimo.

3.3. Modelių efektyvumo rezultatai, naudojant diapazono metriką „Range-F_β“

Taškinės tikslumo, atkūrimo bei F1 įverčio metrikos, naudotos ankstesniuose skyriuose gali ne visuomet teisingai atspindėti modelio efektyvumą aptinkant anomalijas pramoninių procesų duomenyse. Taip yra todėl, nes taškiniai įverčiai vienodai vertina klaidingai teigiamus ir klaidingai neigiamus anomalijų aptikimo atvejus, nors kiekvieno iš šių atvejų svarba gali skirtis pagal procesą, vykstantį sistemoje. Taškinis F1 įvertis daugiau svorio suteikia ilgesnėms anomalijoms ir aukštas F1 įvertis gali vis tiek būti pasiektas, nors ir buvo praleistos kelios trumpos anomalijos.

Range-F_β įvertis skaičiuojamas naudojant diapazono tikslumo ir diapazono atkūrimo metrikas, bei β koeficientą, pagal šią formulę:

$$\text{Range} - F_{\beta} = \frac{(1 + \beta^2) * \text{Range} - \text{prec} * \text{Range} - \text{rec}}{(\beta^2 * R - \text{prec}) + R - \text{rec}}, \quad (24)$$

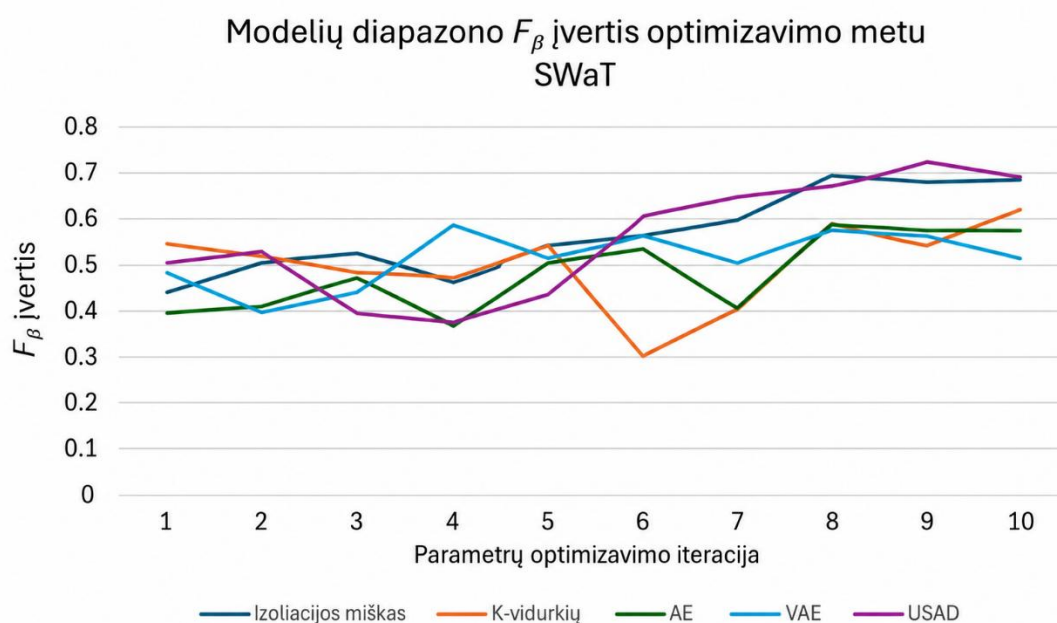
čia *Range-prec* – anomalijų intervalų tikslumas, *Range-rec* – anomalijų intervalų atkūrimas, β – tikslumo ir atkūrimo svarbos svoris.

Tikslumo ir atkūrimo svarbos svoris β leidžia metriką pritaikyti skirtingų procesų duomenims – vienoje sistemoje gali būti svarbu sumažinti klaidingų aliarmų skaičių, o kitais atvejais gali būti svarbiau aptikti kuo daugiau anomalijų net ir padidinant klaidingų aptikimų kiekį. Kai β koeficientas yra didesnis už 1, tuomet atkūrimas yra laikomas svarbesniu parametru, o kai mažesnis už 1, tuomet svarbesniu parametru laikomas tikslumas.

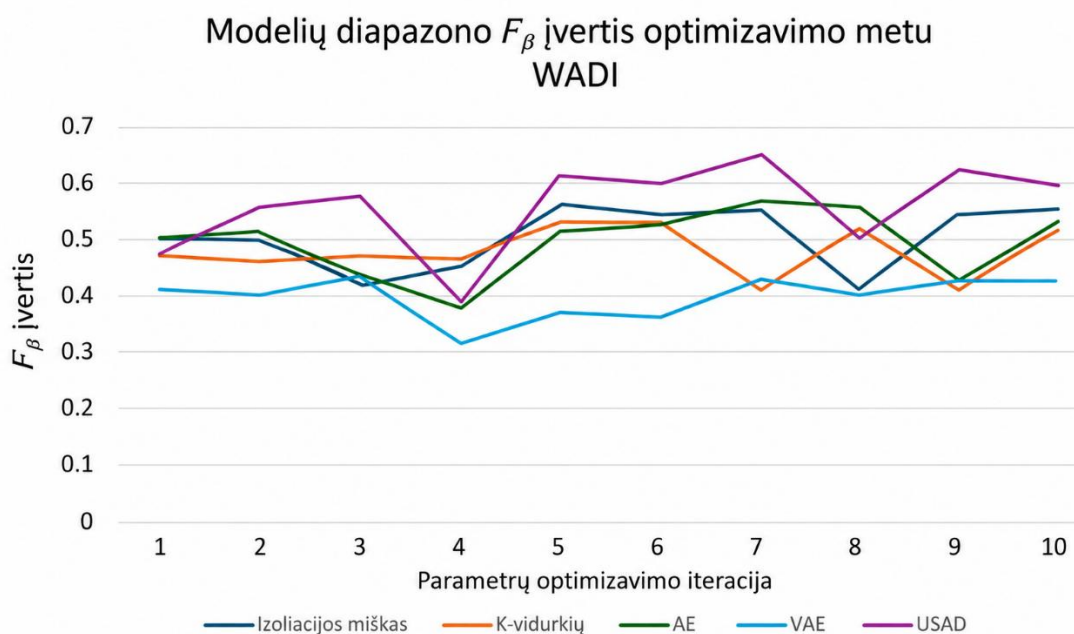
Šiuo atveju, tiek SWaT, tiek WADI duomenų rinkiniams β koeficientas parenkamas 3. Tai reiškia, kad modeliai optimizuojami aptikti daugiau anomalijų, bet tuo pačiu daugiau normalių duomenų sekų bus pažymima kaip anomalijos. Toks sprendimas priimamas todėl, nes vandens valymo sistemose dažniausiai svarbu aptikti kuo daugiau anomalijų, siekiant palaikyti procesą kuo stabilesnį.

Diapazono metrikų skaičiavimui naudojama „prts 1.0.0.03“ Python biblioteka.

Paveikslėliuose žemiau atvaizduotas modelių $Range-F_\beta$ įverčio kitimas optimizavimo metu SWaT ir WADI duomenų rinkiniuose.



36 pav. Modelių $Range-F_\beta$ įverčio kitimas modelių optimizavimo metu SWaT duomenyse



37 pav. Modelių $Range-F_\beta$ įverčio kitimas modelių optimizavimo metu WADI duomenyse

Pagal diapazono metrikas optimizuotų modelių įverčiai pateikti 22 ir 23 lentelėse.

22 lentelė. Pagal diapazono metrikas optimizuotų modelių įverčiai SWaT duomenų rinkinyje.

	Izoliacijos miškas	K-vidurkių modelis	Auto-enkoderis	Variacinis auto- enkoderis	USAD
Diapazono tikslumas	0.58	0.52	0.49	0.43	0.60
Diapazono atkūrimas	0.73	0.68	0.65	0.58	0.76
Diapazono F_β įvertis	0.69	0.62	0.59	0.52	0.72

23 lentelė. Pagal diapazono metrikas optimizuotų modelių įverčiai WADI duomenų rinkinyje.

	Izoliacijos miškas	K-vidurkių modelis	Auto-enkoderis	Variacinis auto- enkoderis	USAD
Diapazono tikslumas	0.48	0.45	0.50	0.36	0.57
Diapazono atkūrimas	0.63	0.60	0.64	0.50	0.71
Diapazono F_β įvertis	0.57	0.54	0.58	0.44	0.65

Kaip galima matyti iš gautų rezultatų, optimizavus modelius naudojant diapazono metrikas, vyrauja tos pačios tendencijos kaip ir modelius optimizavus standartinėmis, taškinėmis metrikomis. Visi modeliai šiek tiek prasčiau pasirodė WADI, nei SWaT duomenų rinkinyje, kaip jau aptarta, dėl duomenų rinkinio sudėtingumo. Taip pat geriausiai pasirodė USAD modelis, surinkęs 0.72 F_β įvertį SWaT duomenyse ir 0.65 F_β įvertį WADI duomenyse.

Lyginant diapazono metrikų įverčius su standartinių metrikų įverčiais, kone visi modeliai surinko šiek tiek žemesnius diapazono įverčius, išskyrus izoliacijos miško algoritmą, kuris SWaT rinkinyje surinko šiek tiek didesnę diapazono F_β įvertį (0.69), nei taškinį F1 įvertį (0.66).

Nors modelių efektyvumo įverčiai diapazono metrikomis yra mažesni, jie geriau atspindi modelių efektyvumą konkrečiame pramoniniame procese, su konkrečiomis sąlygomis, kurios reguliuojamos β koeficientu. Taip pat, kaip matoma 36 ir 37 paveikslėliuose, modelių optimizavimo metu F_β įverčio kitimo variacija yra gerokai didesnė, nei standartinio F1 įverčio variacija, kuri matoma 29-35 paveikslėliuose. Tai reiškia, kad modeliai, derinami naudojant F_β įvertį yra gerokai jautresni parametrų kitimui, kas leidžia tiksliau sureguliuoti modelių parametrus.

Lentelėje žemiau pateikti modelių parametrai, su kuriais modeliai optimizavimo metu pagal diapazono F_β įvertį parodė geriausius rezultatus.

24 lentelė. Modelių, optimizuotų pagal F_{β} įvertį parametrai.

	SWaT	WADI
Izoliacijos miškas	window_size=30, n_estimators=100, contamination=0.10, max_samples=256, max_features=1.0	window_size=60, n_estimators=200, contamination=0.06, max_samples=512, max_features=0.7
K-vidurkių modelis	window_size=30, contamination=0.12, n_init=20, max_iter=500	window_size=60, contamination=0.06, n_init=50, max_iter=500
Auto-enkoderis	window_size=100, learning_rate=0.001, epochs=60, scale=16, dropout=0.1, batch_size=64	window_size=150, learning_rate=0.0005, epochs=100, scale=8, dropout=0.2, batch_size=32
Variacinis auto- enkoderis	window_size=100, learning_rate=0.0005, epochs=80, scale=16, dropout=0.1, batch_size=64, KLD_weight=0.1	window_size=150, learning_rate=0.0005, epochs=100, scale=8, dropout=0.2, batch_size=32, KLD_weight=0.01
USAD	window_size=12, latent_size=40, batch_size=128, lrate=0.005, nepochs=150, alpha=0.5, beta=0.5	window_size=30, latent_size=64, batch_size=128, lrate=0.005, nepochs=150, alpha=0.3, beta=0.7

Išvados

1. Atlikus literatūros analizę išsiaiškinta, kad anomalijų aptikimui pramoninių procesų duomenyse dažniausiai taikomi neprižiūrimi mašininio mokymo modeliai, kurie nereikalauja mokymosi duomenų su sužymėtomis anomalijomis.
2. Parinkus vandens valymo ir skirstymo sistemos duomenų rinkinius nustatyta, kad duomenų rinkiniai prieš tolimesnį naudojimą turi būti apdoroti, pašalinant nereikalingus duomenis, triukšmus, taikant normalizavimą, bei išsaugant tinkamu formatu. Toks duomenų rinkinių apdorojimas leidžia pagerinti mašininio mokymo modelių apmokymo kokybę.
3. Realizavus penkis mašininio mokymo modelius – izoliacijos miško modelį, K-vidurkių modelį, auto-enkoderį, variacinį auto-enkoderį ir USAD modelį, suprasta, kad norint pagerinti modelių efektyvumo rodiklius, reikia optimizuoti modelių parametrus, kadangi atlikus parametrų derinimą visų modelių efektyvumo rodikliai ženkliai pagerėjo abiejuose tirtuose duomenų rinkiniuose. Atlikus parametrų optimizavimą didžiausią pokytį turėjo variacinio auto-enkoderio modelio F1 įvertis SWaT duomenyse – pakilo nuo 0.41 iki 0.52.
4. Nustatyta, kad visi naudoti mašininio mokymo modeliai prasčiau pasirodė WADI duomenų rinkinyje. Tai galima paaiškinti tuo, kad WADI yra gerokai didesnis ir sudėtingesnis nei SWaT duomenų rinkinys (123 jutiklių ir pavarų duomenys WADI, o 40 jutiklių ir pavarų duomenų SWaT), taip pat turi labiau triukšmingus signalus.
5. Diapazono metrikos pasirodė tinkamesnės modelių efektyvumo vertinimui pramoniniuose duomenyse, kadangi jos geriau atspindi modelių efektyvumą konkrečiame pramoniniame procese, su konkrečiomis sąlygomis, nei standartinės tikslumo, atkūrimo ir F1 įverčio metrikos. Nors optimizuojant modelių parametrus pagal diapazono F_β įvertį gauti šiek tiek žemesni modelių efektyvumo įverčiai (išskyrus izoliacijos miško modelį, kuris surinko 0.03 didesnę F_β įvertį, nei F1 įvertį), gautas F_β įverčių grafikas parametrų derinimo metu rodo didesnę F_β įverčio kitimą, kas reiškia, kad modeliai jautriau reaguoja į parametrų kitimą naudojant diapazono metrikas, kas leidžia tiksliau optimizuoti modelius.
6. Geriausius rezultatus naudojant tiek standartines, tiek diapazono metrikas efektyvumo vertinimui abiejuose duomenų rinkiniuose parodė USAD modelis (0.74 F1 įvertis ir 0.72 F_β įvertis SWaT duomenų rinkinyje). Iš to galima spręsti, kad iš tirtų modelių šis modelis yra tinkamiausias anomalijų aptikimui vandens apdorojimo ir skirstymo sistemose.

Literatūros sąrašas

1. STOUFFER, K. – LIGHTMAN, S. – PILLITTERI, V. – ABRAMS, M. – HAHN, A. Guide to industrial control systems (ICS) security. In *NIST Special Publication 800-82, Revision 2 Initial Public Draft*. Gaithersburg, MD: National Institute of Standards and Technology, 2014.
2. LEE, R. M. – ASSANTE, M. J. – CONWAY, T. Analysis of the cyber attack on the Ukrainian power grid: Defense use case. In *Electricity Information Sharing and Analysis Center (E-ISAC); SANS Institute*. 2016-03-18.
3. MCCARTHY, J. – POWELL, M. – STOUFFER, K. – TANG, C. – ZIMMERMAN, T. – BARKER, W. – OGUNYALE, T. – WYNNE, D. – WILTBERGER, J. Securing manufacturing industrial control systems: Behavioral anomaly detection. In *NIST Interagency Report 8219*. Gaithersburg, MD: National Institute of Standards and Technology, 2020.
4. CHANDOLA, V. - BANERJEE, A. - KUMAR, V. Anomaly detection: A survey. In *ACM Computing Surveys (CSUR)*. 2009. Vol. 41, no. 3, p. 1–58.
5. CHANDOLA, V. - BANERJEE, A. - KUMAR, V. Anomaly detection for discrete sequences: A survey. In *IEEE Transactions on Knowledge and Data Engineering*. 2010. Vol. 24, no. 5, p. 823–839.
6. KHETTAF, D. – DJENOURI, D. – REZAEIFAR, Z. – DJENOURI, Y. Deep learning for collective anomaly detection. In *ACM Computing Surveys*. 2026, Vol. 58, no. 8, Article 214, p. 1–34.
7. ZAMANZADEH DARBAN, Z. – WEBB, G. I. – PAN, S. – AGGARWAL, C. – SALEHI, M. Deep learning for time series anomaly detection: A survey. In *ACM Computing Surveys*. 2024, Vol. 57, no. 1, Article 15, p. 1–42.
8. BLÁZQUEZ-GARCÍA, A. – CONDE, A. – MORI, U. – LOZANO, J. A. A review on outlier/anomaly detection in time series data. In *ACM Computing Surveys*. 2021, Vol. 54, no. 3, Article 56, p. 1–33.
9. NASSIF, A.B. - ABU TALIB, M. - NASIR, Q. - DAKALBAB, F.M. Machine learning for anomaly detection: A systematic review. In *IEEE Access*. 2021. Vol. 9, p. 78658–78700.
10. AHMED, M. - MAHMOOD, A.N. - HU, J. A survey of network anomaly detection techniques. In *Journal of Network and Computer Applications*. 2016. Vol. 60, p. 19–31.
11. CUI, L. – ZHANG, Q. – SHI, Y. – YANG, L. – WANG, Y. – WANG, J. – BAI, C. A method for satellite time series anomaly detection based on fast-DTW and improved-KNN. In *Chinese Journal of Aeronautics*. 2023, Vol. 36, no. 2, p. 149–159.
12. HU, W. – LIAO, Y. – VEMURI, V. R. Robust anomaly detection using support vector machines. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*. 2003.
13. MOUNCE, S. R. – MOUNCE, R. B. – BOXALL, J. B. Novelty detection for time series data analysis in water distribution systems using support vector machines. In *Journal of Hydroinformatics*. 2011, Vol. 13, no. 4, p. 672–686.
14. AGGARWAL, C. C. *Data mining: The textbook*. Cham: Springer International Publishing, 2015.
15. ZHANG, J. – ZHANG, H. – DING, S. – ZHANG, X. Power consumption predicting and anomaly detection based on Transformer and K-means. In *Journal of Physics: Conference Series*. 2022.

16. WESTA, N. – SCHLEGL, T. – DEUSE, J. Unsupervised anomaly detection in unbalanced time series data from screw driving processes using k-means clustering. In *Procedia CIRP*. 2023. Vol. 120, p. 1185–1190.
17. AUDIBERT, J. – MICHIARDI, P. – GUYARD, F. – MARTI, S. – ZULUAGA, M. A. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20)*. Virtual Event, USA: ACM, 2020.
18. KIEU, T. – YANG, B. – GUO, C. – JENSEN, C. S. Outlier detection for time series with recurrent autoencoder ensembles. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2018.
19. AUDIBERT, J. – TAVENARD, R. – LORENZI, M. – ZULUAGA, M. A. – MICHIARDI, P. – FILIPPONE, M. Unsupervised anomaly detection in time-series. Sorbonne Université, PhD thesis, 2021.
20. ZHANG, C. – LI, S. – ZHANG, H. – CHEN, Y. VELC: A new variational autoencoder based model for time series anomaly detection. In *Lecture Notes in Computer Science*. 2020.
21. AZZARI, A. On mining time series data with random forest models: Perspectives from classification, anomaly detection, and distance measures. Doctoral dissertation. Verona: University of Verona; Politecnico di Torino, 2026.
22. LASKAR, M. T. R. – HUANG, J. X. – SMETANA, V. – STEWART, C. – POUW, K. – AN, A. – CHAN, S. – LIU, L. Extending isolation forest for anomaly detection in big data via k-means. In *ACM Transactions on Cyber-Physical Systems*. 2021, Vol. 5, no. 4, Article 41.
23. AUDIBERT, J. – MICHIARDI, P. – GUYARD, F. – MARTI, S. – ZULUAGA, M. A. USAD: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20)*. Virtual Event, USA: ACM, 2020.
24. JIANG, W. – HONG, Y. – ZHOU, B. – HE, X. – CHENG, C. A GAN-based anomaly detection approach for imbalanced industrial time series. In *IEEE Access*. 2019. Vol. 7, p. 143608–143619.
25. MIRZA, M. – OSINDERO, S. Conditional Generative Adversarial Nets. In arXiv preprint arXiv:1411.1784. 2014.
26. HALOUI, I. – GUPTA, J. S. – FEUILLARD, V. Anomaly detection with Wasserstein GAN. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*. Munich, Germany: Springer, 2019. p. 3–15.
27. SCHLEGL, T. – SEEBÖCK, P. – WALDSTEIN, S.M. – SCHMIDT-ERFURTH, U. – LANGS, G. AnoGAN: Deep anomaly detection for MRI images. In *Medical Image Analysis*. 2017. p. 1–13.
28. SCHLEGL, T. – SEEBÖCK, P. – WALDSTEIN, S. M. – LANGS, G. – SCHMIDT-ERFURTH, U. f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. In *Medical Image Analysis*. 2019. Vol. 54, p. 30–44.
29. LI, D. – et al. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*. Cham: Springer International Publishing, 2019.
30. DAW, A. – MARUF, M. – KARPATNE, A.. PID-GAN: A GAN Framework based on a Physics-informed Discriminator for Uncertainty Quantification with Physics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21)*. Virtual Event, Singapore: Association for Computing Machinery, 2021. p. 237–247.

31. KOVÁCS, G. – SEBESTYÉN, G. – HANGAN, A.. *Evaluation metrics for anomaly detection algorithms in time-series*. In *Acta Universitatis Sapientiae, Informatica*. 2019. Vol. 11, no. 2, p. 113–130.
32. MALAIYA, R. K. – KWON, D. – SUH, S. C. – KIM, H. – KIM, I. – KIM, J.. *An Empirical Evaluation of Deep Learning for Network Anomaly Detection*. In *IEEE Access*. 2019. Vol. 7, p. 140806–140817.
33. STOJANOVIĆ, B. – NEUSCHMIED, H. – WINTER, M. – KLEB, U.. *Enhanced Anomaly Detection for Cyber-Attack Detection in Smart Water Distribution Systems*. In *Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES 2022)*. Vienna: Association for Computing Machinery, 2022. 7 p.
34. MANCA, G. – DIX, M. – FAY, A.. *Clustering of Similar Historical Alarm Subsequences in Industrial Control Systems Using Alarm Series and Characteristic Coactivations*. In *IEEE Access*. 2021. Vol. 9, p. 154965–154983.
35. MATHUR, A.P. - TIPPENHAUER, N.O. SWaT: A Water Treatment Testbed for Research and Training on ICS Security. In 2016 IEEE International Conference on Industrial Technology (ICIT). 2016.
36. ELNOUR, M. – MESKIN, N. – KHAN, K. – JAIN, R.. *A Dual-Isolation-Forests-based Attack Detection Framework for Industrial Control Systems*. In *IEEE Access*. 2020. Vol. 8, p. 78585–78596.
37. AHMED, C. M. – PALLETI, V. R. – MATHUR, A. P.. *WADI: A Water Distribution Testbed for Research in the Design of Secure Cyber Physical Systems*. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWATER 2017)*. Pittsburgh: Association for Computing Machinery, 2017. p. 25–28.

1 priedas. Duomenų rinkinių apdorojimo programinis kodas

```

1  import ast
2  import csv
3  import os
4  import sys
5  from pickle import dump
6  import json
7
8  import argparse
9
10 import numpy as np
11 import pandas as pd
12 from sklearn.preprocessing import MinMaxScaler
13
14
15
16 def load_as_np(category, filename, dataset, dataset_folder, output_folder):
17     temp = np.genfromtxt(os.path.join(dataset_folder, category, filename),
18                         dtype=np.float32,
19                         delimiter=',')
20     return temp
21
22
23 def load_data(dataset, base_dir, output_folder, json_folder):
24
25     if dataset == 'SWaT':
26         dataset_folder = os.path.join(base_dir, 'SWaT/Physical')
27         normal_data = pd.read_excel(os.path.join(dataset_folder, 'SWaT_Dataset_Normal_v1.xlsx'))
28         normal_data = normal_data.iloc[1:, 1:-1].to_numpy()
29         normal_data = MinMaxScaler().fit_transform(normal_data).clip(0, 1)
30         np.save(os.path.join(output_folder, dataset + "_train.npy"), normal_data)
31
32         abnormal_data = pd.read_excel(os.path.join(dataset_folder, 'SWaT_Dataset_Attack_v0.xlsx'))
33         abnormal_label = abnormal_data.iloc[1:, -1] == 'Attack'
34         abnormal_label = abnormal_label.to_numpy().astype(int)
35
36         abnormal_data = abnormal_data.iloc[1:, 1:-1].to_numpy()
37         abnormal_data = MinMaxScaler().fit_transform(abnormal_data).clip(0, 1)
38         np.save(os.path.join(output_folder, dataset + "_test.npy"), abnormal_data)
39         np.save(os.path.join(output_folder, dataset + "_test_label.npy"), abnormal_label)
40
41     elif dataset == 'WADI':
42         normal_data = pd.read_csv(os.path.join(base_dir, 'WADI/WADI.A2_19 Nov 2019/WADI_14days_new.csv'))
43         normal_data = normal_data.dropna(axis='columns', how='all').dropna()
44         normal_data = normal_data.iloc[:, 3:].to_numpy()
45         normal_data = MinMaxScaler().fit_transform(normal_data).clip(0, 1)
46         np.save(os.path.join(output_folder, dataset + "_train.npy"), normal_data)
47
48         abnormal_data = pd.read_csv(os.path.join(base_dir, 'WADI/WADI.A2_19 Nov 2019/WADI_attackdataLABEL.csv'), header=1)
49         abnormal_data = abnormal_data.dropna(axis='columns', how='all').dropna()
50         abnormal_label = abnormal_data.iloc[:, -1] == -1
51         abnormal_label = abnormal_label.to_numpy().astype(int)
52
53         abnormal_data = abnormal_data.iloc[:, 3:-1].to_numpy()
54         abnormal_data = MinMaxScaler().fit_transform(abnormal_data).clip(0, 1)
55         np.save(os.path.join(output_folder, dataset + "_test.npy"), abnormal_data)
56         np.save(os.path.join(output_folder, dataset + "_test_label.npy"), abnormal_label)
57
58
59
60
61 if __name__ == '__main__':
62     parser = argparse.ArgumentParser()
63     parser.add_argument("--dataset", required=True, type=str,
64                         help="Name of dataset; SWaT/WADI")
65     parser.add_argument("--data_dir", required=True, type=str,
66                         help="Directory of raw data")
67     parser.add_argument("--out_dir", default=None, type=str,
68                         help="Directory of the processed data")
69     parser.add_argument("--json_dir", default=None, type=str,
70                         help="Directory of the json files for the processed data")
71     options = parser.parse_args()
72
73     datasets = ['SWaT', 'WADI']
74
75     if options.dataset in datasets:
76         base_dir = options.data_dir
77
78         if options.out_dir == None:
79             output_folder = os.path.join(base_dir, 'processed')
80         else:
81             output_folder = options.out_dir
82         if not os.path.exists(output_folder):
83             os.mkdir(output_folder)
84
85         if options.json_dir == None:
86             json_folder = os.path.join(base_dir, 'json')
87         else:
88             json_folder = options.json_dir
89         if not os.path.exists(json_folder):
90             os.mkdir(json_folder)
91
92         load_data(options.dataset, base_dir, output_folder, json_folder)

```

2 priedas. Izoliacijos miško modelio programinis kodas

```
1 import numpy as np
2 import pandas as pd
3 from sklearn.ensemble import IsolationForest
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import classification_report
6
7 PATH = "C:\\Users\\User\\Desktop\\Datasets"
8
9 df = pd.read_csv(PATH + '\\SWaT_Dataset_Normal_v1.csv', parse_dates=['Timestamp'])
10 float64_cols = df.select_dtypes(include=['float64']).columns
11 df[float64_cols] = df[float64_cols].astype('float32')
12 int64_cols = df.select_dtypes(include=['int64']).columns
13 df[int64_cols] = df[int64_cols].astype('int32')
14
15 df_copy = df.drop(['Timestamp'], axis=1)
16 df_copy['isAttack'] = np.where(df_copy['Machine_Status'] == 'Attack', -1, 1)
17 df_copy = df_copy.drop(['Machine_Status'], axis=1)
18
19 X = df_copy.drop(['isAttack'], axis=1)
20 y = df_copy['isAttack']
21
22 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
23
24 contamination = 0.03
25 model = IsolationForest(random_state=42, n_jobs=-1, contamination=contamination)
26 model.fit(X_train)
27
28 y_pred = model.predict(X_test)
29
30 target_names = ['normal', 'attack']
31 report = classification_report(y_test.to_numpy(), y_pred, target_names=target_names, zero_division=1)
32 print(report)
33
34 unique, counts = np.unique(y_pred, return_counts=True)
35 print(dict(zip(unique, counts)))
```

3 priedas. K-vidurkių modelio programinis kodas

```
1 import pandas as pd
2 import numpy as np
3 import os
4 import matplotlib.pyplot as plt
5 from sklearn.decomposition import PCA
6 from sklearn.cluster import KMeans
7 from sklearn.metrics import classification_report
8 from sklearn.metrics.pairwise import cosine_distances
9
10 PATH = "C:\\Users\\User\\Desktop\\Datasets\\SWaT_Dataset_Normal_v1.csv"
11 PLOT = True
12 ANOMALY_FRACTION = 0.05
13 PCA_COMPONENTS = 3
14 N_CLUSTERS = 3
15 RANDOM_STATE = 42
16
17 df = pd.read_csv(PATH, parse_dates=['Timestamp'])
18 float_cols = df.select_dtypes(include=['float64']).columns
19 int_cols = df.select_dtypes(include=['int64']).columns
20 df[float_cols] = df[float_cols].astype('float32')
21 df[int_cols] = df[int_cols].astype('int32')
22
23 df.loc[df['Machine_Status'] == 'Normal', 'Machine_Status'] = 0
24 df.loc[df['Machine_Status'] == 'Attack', 'Machine_Status'] = 1
25 df['Machine_Status'] = df['Machine_Status'].astype(int)
26 df.dropna(subset=['Machine_Status'], inplace=True)
27 df.set_index('Timestamp', inplace=True)
28 df = df.loc[df.index.year == 2015]
29
30 num_cols = [col for col in df.columns if df[col].nunique() > 10]
31 df_numeric = df[num_cols]
32
33 pca = PCA(n_components=PCA_COMPONENTS)
34 pComponents = pca.fit_transform(df_numeric)
35 principal_df = pd.DataFrame(data=pComponents, columns=[f'pca{i+1}' for i in range(PCA_COMPONENTS)], index=df.index)
36
37 kmeans = KMeans(n_clusters=N_CLUSTERS, random_state=RANDOM_STATE)
38 kmeans.fit(principal_df.values)
39 labels = kmeans.predict(principal_df.values)
40 principal_df['cluster'] = labels
```

```

42 def getDistanceByPoint(data, model, distance_type='euclidean'):
43     distance = []
44     for i in range(len(data)):
45         Xa = np.array(data.iloc[i][:-1])
46         Xb = model.cluster_centers_[model.labels_[i]]
47         if distance_type == 'euclidean':
48             distance.append(np.linalg.norm(Xa - Xb))
49         elif distance_type == 'cosine':
50             distance.append(cosine_distances([Xa], [Xb])[0][0])
51     return pd.Series(distance, index=data.index)
52
53 distance = getDistanceByPoint(principal_df, kmeans, distance_type='euclidean')
54 number_of_outliers = int(ANOMALY_FRACTION * len(distance))
55 threshold = distance.nlargest(number_of_outliers).min()
56 principal_df['anomaly'] = (distance >= threshold).astype(int)
57 df['anomaly'] = principal_df['anomaly']
58
59 predictions = df[['Machine_Status', 'anomaly']]
60 print(classification_report(predictions['Machine_Status'], predictions['anomaly']))
61
62 if PLOT:
63     attack = df[df['anomaly'] == 1]
64     plt.figure(figsize=(40,3))
65     plt.plot(df['FIT101'], color='blue', label='Normal')
66     plt.plot(attack['FIT101'], linestyle='none', marker='X', color='red', markersize=12, label='Anomaly')
67     plt.xlabel('Date and Time')
68     plt.ylabel('Sensor Reading')
69     plt.title('FIT101 Anomalies')
70     plt.legend(loc='best')
71     if not os.path.exists('plots'):
72         os.makedirs('plots')
73     plt.savefig('plots/anomaly.png', bbox_inches='tight')
74     plt.show()

```

4 priedas. Auto-enkoderio, bei variacinio auto-enkoderio modelių programinis kodas

```

1  t math
2  t numpy as np
3  t torch
4  t torch.nn as nn
5  torch.utils.data import Dataset, DataLoader
6  torch.optim import Adam
7
8
9  TimeSeriesWindowDataset(Dataset):
10 def __init__(self, data_path, window_size=100, stride=2, labels_path=None):
11     data = np.load(data_path).astype(np.float32)
12
13     self.data = torch.tensor(data, dtype=torch.float32)
14     self.window_size = window_size
15     self.stride = stride
16
17     if labels_path is not None:
18         labels = np.load(labels_path).astype(np.float32)
19         self.labels = torch.tensor(labels, dtype=torch.float32)
20     else:
21         self.labels = None
22
23     self.indices = list(range(0, len(self.data) - window_size + 1, stride))
24
25 def __len__(self):
26     return len(self.indices)
27
28 def __getitem__(self, idx):
29     start = self.indices[idx]
30     end = start + self.window_size
31
32     x = self.data[start:end]
33
34     if self.labels is not None:
35         y = self.labels[start:end]
36         return x, y
37
38     return x, torch.zeros(self.window_size)
39
40
41 PositionalEncoding(nn.Module):
42 def __init__(self, d_model, dropout=0.1, max_len=5000, batch_first=True):
43     super().__init__()
44
45     self.dropout = nn.Dropout(p=dropout)
46     self.batch_first = batch_first

```

```

47     pe = torch.zeros(max_len, d_model)
48
49
50     position = torch.arange(0, max_len).float().unsqueeze(1)
51
52     div_term = torch.exp(
53         torch.arange(0, d_model).float() * (-math.log(10000.0) / d_model)
54     )
55
56     pe += torch.sin(position * div_term)
57     pe += torch.cos(position * div_term)
58
59     if batch_first:
60         pe = pe.unsqueeze(0)
61     else:
62         pe = pe.unsqueeze(0).transpose(0, 1)
63
64     self.register_buffer("pe", pe)
65
66     def forward(self, x):
67         if self.batch_first:
68             x = x + self.pe[:, :x.size(1), :]
69         else:
70             x = x + self.pe[:x.size(0), :, :]
71
72         return self.dropout(x)
73
74
75     class TransformerAE(nn.Module):
76         def __init__(self, features, window_size=100, scale=16):
77             super().__init__()
78
79             self.features = features
80             self.window_size = window_size
81             self.scale = scale
82             self.d_model = features * scale
83
84             self.input_layer = nn.Linear(features, self.d_model)
85             self.output_layer = nn.Linear(self.d_model, features)
86
87             self.pos_encoder = PositionalEncoding(
88                 d_model=self.d_model,
89                 dropout=0.1,
90                 max_len=window_size,
91                 batch_first=True
92             )
93
94             encoder_layer = nn.TransformerEncoderLayer(
95                 d_model=self.d_model,
96                 nhead=features,
97                 dim_feedforward=256,
98                 dropout=0.1,
99                 batch_first=True
100             )
101
102             decoder_layer = nn.TransformerDecoderLayer(
103                 d_model=self.d_model,
104                 nhead=features,
105                 dim_feedforward=256,
106                 dropout=0.1,
107                 batch_first=True
108             )
109
110             self.encoder = nn.TransformerEncoder(
111                 encoder_layer,
112                 num_layers=1
113             )
114
115             self.decoder = nn.TransformerDecoder(
116                 decoder_layer,
117                 num_layers=1
118             )
119
120         def forward(self, src):
121             model_dim = self.d_model
122
123             src_emb = self.input_layer(src)
124             src_emb = src_emb * math.sqrt(model_dim)
125             src_emb = self.pos_encoder(src_emb)
126
127             memory = self.encoder(src_emb)
128
129             z = torch.mean(memory, dim=1, keepdim=True)

```

```

131         tgt_emb = self.input_layer(src)
132         tgt_emb = tgt_emb * math.sqrt(model_dim)
133
134         out = self.decoder(tgt_emb, z)
135         out = self.output_layer(out)
136
137         return out
138
139
140 class TransformerVAE(nn.Module):
141     def __init__(self, features, window_size=100, scale=16):
142         super().__init__()
143
144         self.features = features
145         self.window_size = window_size
146         self.scale = scale
147         self.d_model = features * scale
148
149         self.input_layer = nn.Linear(features, self.d_model)
150         self.output_layer = nn.Linear(self.d_model, features)
151
152         self.pos_encoder = PositionalEncoding(
153             d_model=self.d_model,
154             dropout=0.1,
155             max_len=window_size,
156             batch_first=True
157         )
158
159         encoder_layer = nn.TransformerEncoderLayer(
160             d_model=self.d_model,
161             nhead=features,
162             dim_feedforward=256,
163             dropout=0.1,
164             batch_first=True
165         )
166
167         decoder_layer = nn.TransformerDecoderLayer(
168             d_model=self.d_model,
169             nhead=features,
170             dim_feedforward=256,
171             dropout=0.1,
172             batch_first=True
173         )
174
175         self.encoder = nn.TransformerEncoder(
176             encoder_layer,
177             num_layers=1
178         )
179
180         self.decoder = nn.TransformerDecoder(
181             decoder_layer,
182             num_layers=1
183         )
184
185         self.mean = nn.Linear(
186             window_size * self.d_model,
187             self.d_model
188         )
189
190         self.logvar = nn.Linear(
191             window_size * self.d_model,
192             self.d_model
193         )
194
195     def reparameterize(self, mean, logvar):
196         std = torch.exp(0.5 * logvar)
197         eps = torch.randn_like(std)
198
199         return mean + eps * std
200
201     def forward(self, src):
202         batch_size = src.size(0)
203         model_dim = self.d_model
204
205         src_emb = self.input_layer(src)
206         src_emb = src_emb * math.sqrt(model_dim)
207         src_emb = self.pos_encoder(src_emb)
208
209         memory = self.encoder(src_emb)
210
211         hidden = memory.reshape(batch_size, -1)
212
213         mean = self.mean(hidden)
214         logvar = self.logvar(hidden)
215
216         z = self.reparameterize(mean, logvar)
217         z = z.reshape(batch_size, 1, -1)

```

```

219         tgt_emb = self.input_layer(src)
220         tgt_emb = tgt_emb * math.sqrt(model_dim)
221
222         out = self.decoder(tgt_emb, z)
223         out = self.output_layer(out)
224
225         return out, mean, logvar
226
227
228     def vae_loss(reconstruction, original, mean, logvar):
229         batch_size = original.size(0)
230
231         mse = nn.functional.mse_loss(
232             reconstruction.reshape(batch_size, -1),
233             original.reshape(batch_size, -1),
234             reduction="sum"
235         )
236
237         kld = -0.5 * torch.sum(
238             1 + logvar - mean.pow(2) - logvar.exp()
239         )
240
241         return mse + kld, mse, kld
242
243
244     def initialize_weights(model):
245         for p in model.parameters():
246             if p.dim() > 1:
247                 nn.init.xavier_uniform_(p)
248
249
250     def train_ae(model, train_loader, val_loader, epochs, learning_rate, device):
251         model.to(device)
252
253         initialize_weights(model)
254
255         optimizer = Adam(
256             model.parameters(),
257             lr=learning_rate,
258             weight_decay=0
259         )
260
261         criterion = nn.MSELoss()
262
263         for epoch in range(1, epochs + 1):
264             model.train()
265
266             train_loss = 0.0
267
268             for x, _ in train_loader:
269                 x = x.to(device)
270
271                 optimizer.zero_grad()
272
273                 reconstruction = model(x)
274
275                 loss = criterion(reconstruction, x)
276
277                 loss.backward()
278
279                 optimizer.step()
280
281                 train_loss += loss.item()
282
283             train_loss /= len(train_loader)
284
285             model.eval()
286
287             val_loss = 0.0
288
289             with torch.no_grad():
290                 for x, _ in val_loader:
291                     x = x.to(device)
292
293                     reconstruction = model(x)
294
295                     loss = criterion(reconstruction, x)
296
297                     val_loss += loss.item()
298
299             val_loss /= len(val_loader)
300
301             print(
302                 f"AE Epoch {epoch}/{epochs} | "
303                 f"Train loss: {train_loss:.6f} | "
304                 f"Val loss: {val_loss:.6f}"
305             )

```

```

308 def train_vae(model, train_loader, val_loader, epochs, learning_rate, device):
309
310     initialize_weights(model)
311
312     optimizer = Adam(
313         model.parameters(),
314         lr=learning_rate,
315         weight_decay=0
316     )
317
318     criterion = nn.MSELoss()
319
320     for epoch in range(1, epochs + 1):
321         model.train()
322
323         train_loss = 0.0
324         train_mse = 0.0
325         train_kld = 0.0
326
327         for x, _ in train_loader:
328             x = x.to(device)
329
330             optimizer.zero_grad()
331
332             reconstruction, mean, logvar = model(x)
333
334             loss, mse, kld = vae_loss(
335                 reconstruction,
336                 x,
337                 mean,
338                 logvar
339             )
340
341             loss.backward()
342
343             optimizer.step()
344
345             train_loss += loss.item()
346             train_mse += mse.item()
347             train_kld += kld.item()
348
349         train_loss /= len(train_loader)
350         train_mse /= len(train_loader)
351         train_kld /= len(train_loader)
352
353         model.eval()
354         val_loss = 0.0
355
356         with torch.no_grad():
357             for x, _ in val_loader:
358                 x = x.to(device)
359
360                 reconstruction, _, _ = model(x)
361
362                 loss = criterion(reconstruction, x)
363
364                 val_loss += loss.item()
365
366         val_loss /= len(val_loader)
367
368         print(
369             f"VAE Epoch {epoch}/{epochs} | "
370             f"Train loss: {train_loss:.6f} | "
371             f"MSE: {train_mse:.6f} | "
372             f"KLD: {train_kld:.6f} | "
373             f"Val loss: {val_loss:.6f}"
374         )
375
376     if __name__ == "__main__":
377         device = "cuda" if torch.cuda.is_available() else "cpu"
378
379         train_dataset = TimeSeriesWindowDataset(
380             "./data/train_norm.npy",
381             window_size=100,
382             stride=7
383         )
384
385         val_dataset = TimeSeriesWindowDataset(
386             "./data/validation_norm.npy",
387             window_size=100,
388             stride=100
389         )
390
391         train_loader = DataLoader(
392             train_dataset,
393             batch_size=64,
394             shuffle=True,
395             num_workers=4
396         )

```

```

401     val_loader = DataLoader(
402         val_dataset,
403         batch_size=64,
404         shuffle=False
405     )
406
407     ae_model = TransformerAE(
408         features=51,
409         window_size=100,
410         scale=16
411     )
412
413     train_ae(
414         model=ae_model,
415         train_loader=train_loader,
416         val_loader=val_loader,
417         epochs=60,
418         learning_rate=1e-3,
419         device=device
420     )
421
422     torch.save(
423         ae_model.state_dict(),
424         "transformer_ae.pt"
425     )
426
427     vae_model = TransformerVAE(
428         features=51,
429         window_size=100,
430         scale=16
431     )
432
433     train_vae(
434         model=vae_model,
435         train_loader=train_loader,
436         val_loader=val_loader,
437         epochs=50,
438         learning_rate=5e-4,
439         device=device
440     )
441
442     torch.save(
443         vae_model.state_dict(),
444         "transformer_vae.pt"
445     )

```

5 priedas. USAD modelio programinis kodas

```

1  import os
2  import pickle
3  import numpy as np
4  import torch
5  import torch.nn as nn
6  import torch.optim as optim
7  from sklearn.preprocessing import MinMaxScaler
8  from sklearn.metrics import precision_score, recall_score, f1_score, roc_curve, roc_auc_score
9  from torch.utils.data import DataLoader, TensorDataset
10
11
12  class Encoder(nn.Module):
13      def __init__(self, input_size, latent_size=40):
14          super().__init__()
15
16          self.model = nn.Sequential(
17              nn.Linear(input_size, input_size // 2),
18              nn.ReLU(inplace=True),
19              nn.Linear(input_size // 2, input_size // 4),
20              nn.ReLU(inplace=True),
21              nn.Linear(input_size // 4, latent_size),
22              nn.ReLU(inplace=True)
23          )
24
25      def forward(self, x):
26          return self.model(x)
27
28
29  class Decoder(nn.Module):
30      def __init__(self, input_size, latent_size=40):
31          super().__init__()
32
33          self.model = nn.Sequential(
34              nn.Linear(latent_size, input_size // 4),
35              nn.ReLU(inplace=True),
36              nn.Linear(input_size // 4, input_size // 2),
37              nn.ReLU(inplace=True),
38              nn.Linear(input_size // 2, input_size),
39              nn.Sigmoid()
40          )
41
42      def forward(self, x):
43          return self.model(x)

```

```

46 class USAD(nn.Module):
47     def __init__(self, input_size, latent_size=40):
48         super().__init__()
49
50         self.encoder = Encoder(input_size, latent_size)
51         self.decoder1 = Decoder(input_size, latent_size)
52         self.decoder2 = Decoder(input_size, latent_size)
53
54     def forward(self, x):
55         z = self.encoder(x)
56
57         w1_prime = self.decoder1(z)
58         w2_prime = self.decoder2(z)
59
60         w2_double_prime = self.decoder2(
61             self.encoder(w1_prime)
62         )
63
64         return w1_prime, w2_prime, w2_double_prime
65
66
67 class USADLoss(nn.Module):
68     def __init__(self, mode="AE1"):
69         super().__init__()
70
71         if mode not in ["AE1", "AE2"]:
72             raise ValueError("mode must be 'AE1' or 'AE2'")
73
74         self.mode = mode
75
76     def forward(self, x, y, z, epoch):
77         if self.mode == "AE1":
78             return (
79                 (1 / epoch) * torch.mean((x - y) ** 2)
80                 + (1 - 1 / epoch) * torch.mean((x - z) ** 2)
81             )
82
83         return (
84             (1 / epoch) * torch.mean((x - y) ** 2)
85             - (1 - 1 / epoch) * torch.mean((x - z) ** 2)
86         )
87
88
89 class USADTrainer:
90     def __init__(self, model, learning_rate=0.02, alpha=0.5, beta=0.5, device="cpu"):
91         self.model = model.to(device)
92         self.device = device
93
94         self.alpha = alpha
95         self.beta = beta
96
97         self.optimizer1 = optim.Adam(
98             list(self.model.encoder.parameters())
99             + list(self.model.decoder1.parameters()),
100             lr=learning_rate
101         )
102
103         self.optimizer2 = optim.Adam(
104             list(self.model.encoder.parameters())
105             + list(self.model.decoder2.parameters()),
106             lr=learning_rate
107         )
108
109         self.loss1 = USADLoss(mode="AE1")
110         self.loss2 = USADLoss(mode="AE2")
111
112     def train_batch(self, x, epoch):
113         self.model.train()
114
115         x = x.to(self.device)
116
117         z = self.model.encoder(x)
118
119         w1_prime = self.model.decoder1(z)
120         w2_prime = self.model.decoder2(z)
121
122         w2_double_prime = self.model.decoder2(
123             self.model.encoder(w1_prime)
124         )
125
126         error1 = self.loss1(
127             x,
128             w1_prime,
129             w2_double_prime,
130             epoch
131         )

```

```

133         error2 = self.loss2(
134             x,
135             w2_prime,
136             w2_double_prime,
137             epoch
138         )
139
140         error1.backward(retain_graph=True)
141         error2.backward()
142
143         self.optimizer1.step()
144         self.optimizer2.step()
145
146         self.optimizer1.zero_grad()
147         self.optimizer2.zero_grad()
148
149         return error1.item(), error2.item()
150
151     @torch.no_grad()
152     def validate_batch(self, x, epoch):
153         self.model.eval()
154
155         x = x.to(self.device)
156
157         z = self.model.encoder(x)
158
159         w1_prime = self.model.decoder1(z)
160         w2_prime = self.model.decoder2(z)
161
162         w2_double_prime = self.model.decoder2(
163             self.model.encoder(w1_prime)
164         )
165
166         error1 = self.loss1(
167             x,
168             w1_prime,
169             w2_double_prime,
170             epoch
171         )
172
173         error2 = self.loss2(
174             x,
175             w2_prime,
176             w2_double_prime,
177             epoch
178         )
179
180         return error1.item(), error2.item()
181
182     @torch.no_grad()
183     def anomaly_score(self, x):
184         self.model.eval()
185
186         x = x.to(self.device)
187
188         ae1 = self.model.decoder1(
189             self.model.encoder(x)
190         )
191
192         ae2 = self.model.decoder2(
193             self.model.encoder(ae1)
194         )
195
196         score = (
197             self.alpha * torch.mean((x - ae1) ** 2, dim=1)
198             + self.beta * torch.mean((x - ae2) ** 2, dim=1)
199         )
200
201         return score.detach().cpu().numpy()
202
203
204     def load_pickle(path):
205         with open(path, "rb") as file:
206             return pickle.load(file)
207
208

```

```

256 def evaluate_model(y_true, scores, threshold):
257     y_pred = scores > threshold
258
259     precision = precision_score(y_true, y_pred, zero_division=0)
260     recall = recall_score(y_true, y_pred, zero_division=0)
261     f1 = f1_score(y_true, y_pred, zero_division=0)
262     auc = roc_auc_score(y_true, scores)
263
264     return precision, recall, f1, auc
265
266
267 def get_scores(trainer, loader):
268     scores = []
269
270     for (x_batch,) in loader:
271         batch_scores = trainer.anomaly_score(x_batch)
272         scores.extend(batch_scores.tolist())
273
274     return np.asarray(scores)
275
276
277 def train_usad(
278     trainer,
279     train_loader,
280     val_loader,
281     test_loader,
282     y_test,
283     epochs=80,
284     save_path="usad_model.pth"
285 ):
286     for epoch in range(1, epochs + 1):
287         train_loss1 = []
288         train_loss2 = []
289
290         for (x_batch,) in train_loader:
291             error1, error2 = trainer.train_batch(
292                 x_batch,
293                 epoch
294             )
295
296             train_loss1.append(error1)
297             train_loss2.append(error2)
298
299         valid_loss1 = []
300         valid_loss2 = []
301
302         for (x_batch,) in val_loader:
303             error1, error2 = trainer.validate_batch(
304                 x_batch,
305                 epoch
306             )
307
308             valid_loss1.append(error1)
309             valid_loss2.append(error2)
310
311         train_total = np.mean(train_loss1) + np.mean(train_loss2)
312         valid_total = np.mean(valid_loss1) + np.mean(valid_loss2)
313
314         scores = get_scores(trainer, test_loader)
315         threshold = get_threshold_from_roc(y_test, scores)
316
317         precision, recall, f1, auc = evaluate_model(
318             y_test,
319             scores,
320             threshold
321         )
322
323         print(
324             f"Epoch {epoch}/{epochs} | "
325             f"Train loss: {train_total:.6f} | "
326             f"Valid loss: {valid_total:.6f} | "
327             f"Threshold: {threshold:.6f} | "
328             f"Precision: {precision:.4f} | "
329             f"Recall: {recall:.4f} | "
330             f"F1: {f1:.4f} | "
331             f"AUC: {auc:.4f}"
332         )
333
334     torch.save(
335         {
336             "encoder": trainer.model.encoder.state_dict(),
337             "decoder1": trainer.model.decoder1.state_dict(),
338             "decoder2": trainer.model.decoder2.state_dict()
339         },
340         save_path
341     )

```

```

344 if __name__ == "__main__":
345     DATASET = "SWaT"
346
347     PROCESSED_DIR = "./processed"
348
349     WINDOW_LENGTH = 12
350     LATENT_SIZE = 40
351     BATCH_SIZE = 128
352     LEARNING_RATE = 0.02
353     EPOCHS = 80
354
355     ALPHA = 0.5
356     BETA = 0.5
357
358     FEATURES = 51
359
360     TRAIN_PATH = os.path.join(
361         PROCESSED_DIR,
362         f"{DATASET}_train.pkl"
363     )
364
365     TEST_PATH = os.path.join(
366         PROCESSED_DIR,
367         f"{DATASET}_test.pkl"
368     )
369
370     LABEL_PATH = os.path.join(
371         PROCESSED_DIR,
372         f"{DATASET}_test_label.pkl"
373     )
374
375     device = torch.device(
376         "cuda:0" if torch.cuda.is_available() else "cpu"
377     )
378
379     train_data = load_pickle(TRAIN_PATH).reshape(-1, FEATURES)
380     test_data = load_pickle(TEST_PATH).reshape(-1, FEATURES)
381     test_labels = load_pickle(LABEL_PATH).reshape(-1)
382
383     train_data = preprocess(train_data)
384     test_data = preprocess(test_data)
385
386     train_windows = create_windows(
387         train_data,
388         WINDOW_LENGTH
389     )
390
391     test_windows = create_windows(
392         test_data,
393         WINDOW_LENGTH
394     )
395
396     y_test = create_window_labels(
397         test_labels,
398         WINDOW_LENGTH
399     )
400
401     train_windows = flatten_windows(train_windows)
402     test_windows = flatten_windows(test_windows)
403
404     train_split_index = round(0.8 * len(train_windows))
405
406     x_train = train_windows[:train_split_index]
407     x_valid = train_windows[train_split_index:]
408
409     input_size = WINDOW_LENGTH * FEATURES
410
411     train_loader = DataLoader(
412         TensorDataset(torch.from_numpy(x_train).float()),
413         batch_size=BATCH_SIZE,
414         shuffle=False,
415         num_workers=0
416     )
417
418     val_loader = DataLoader(
419         TensorDataset(torch.from_numpy(x_valid).float()),
420         batch_size=BATCH_SIZE,
421         shuffle=False,
422         num_workers=0
423     )
424
425     test_loader = DataLoader(
426         TensorDataset(torch.from_numpy(test_windows).float()),
427         batch_size=BATCH_SIZE,
428         shuffle=False,
429         num_workers=0
430     )
431
432     model = USAD(
433         input_size=input_size,
434         latent_size=LATENT_SIZE
435     )

```

```
436
437     trainer = USADTrainer(
438         model=model,
439         learning_rate=LEARNING_RATE,
440         alpha=ALPHA,
441         beta=BETA,
442         device=device
443     )
444
445     train_usad(
446         trainer=trainer,
447         train_loader=train_loader,
448         val_loader=val_loader,
449         test_loader=test_loader,
450         y_test=y_test,
451         epochs=EPOCHS,
452         save_path=f"{DATASET}_USAD.pth"
453     )
```