Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Laiko eilučių detekcija taikant autoenkoderiu grįstą požymių inžineriją

Baigiamasis magistro studijų projektas

Aurelijus Leonavičius

Projekto autorius

prof. dr. Evaldas Vaičiukynas

Vadovas

dr. Paulius Danėnas

Vadovas

Kaunas, 2022

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Laiko eilučių detekcija taikant autoenkoderiu grįstą požymių inžineriją

Baigiamasis magistro studijų projektas

Didžiųjų verslo duomenų analitika (6213AX001)

Aurelijus Leonavičius

Projekto autorius

prof. dr.
Evaldas Vaičiukynas
Vadovas

dr.
Paulius Danėnas
Vadovas

prof. dr.
Robertas Alzbutas
Recenzentas

prof. dr.
Rytis Krušinskas
Recenzentas

Kaunas, 2022

Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Aurelijus Leonavičius

Laiko eilučių detekcija taikant autoenkoderiu grįstą požymių inžineriją

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 sąraše;

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.

Aurelijus Leonavičius

Patvirtinta elektroniniu būdu

Leonavičius, Aurelijus. Laiko eilučių detekcija taikant autoenkoderiu grįstą požymių inžineriją. Magistro studijų baigiamasis projektas / vadovas prof. dr. Evaldas Vaičiukynas; Kauno technologijos universitetas, Informatikos fakultetas / vadovas dr. Paulius Danėnas; Kauno technologijos universitetas, Informatikos fakultetas

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika (Matematikos mokslai).

Reikšminiai žodžiai: laiko eilučių analizė, požymių inžinerija, autoenkoderis, detekcija, catch22

Kaunas, 2022. 76 p.

Santrauka

Magistro baigiamojo projekto tema - laiko eilučių detekcija taikant autoenkoderiu grįstą požymių inžineriją. Pagrindinė užduotis - autoenkoderiu gautų požymių pritaikymas detekcijos uždaviniui spręsti. Literatūros apžvalgoje pateikta požymių pritaikymo tikslingumo ir metodai. Aprašyti metodai kuriuos taikant spręstas uždavinys. Tyrimo dalyje aprašyti du išbandyti autoenkoderių tipai, atlikta jų latentinių erdvių vizualizacija. Transformavus laiko eilutes į latentinę erdvę gauti požymiai naudoti detekcijos modeliui sudaryti. Detekcijos modelis sudarytas gradientinio tobulinimo metodo principu. Modelio sudarymui ir ištestavimui naudojami 42 skirtingi laiko eilučių rinkiniai, kuriuos iš viso sudaro 45 807 laiko eilutės. Norint įvertinti sudarytojo modelio rezultatus detekcija atlikta pritaikant catch22 požymių rinkinį. Palyginus dviejų požymių rinkinių veikimą sudarytas naujas jungtinis požymių rinkinys kurio AUC įvertis yra 0,9416, truputi nusileidžiantis catch22 rinkiniui kurio AUC įvertis yra 0,9484, tačiau dėka šio modifikavimo pavyko daugiau nei dvigubai padidinti per laiko vienetą apdorojamų laiko eilučių skaičių.

Leonavičius, Aurelijus. Time series detection using autoencoder-based feature engineering. Master's Final Degree Project / supervisor prof. dr. Evaldas Vaičiukynas; Faculty of Informatics, Kaunas University of Technology / supervisor dr. Paulius Danėnas; Faculty of Informatics, Kaunas University of Technology

Study field and area (study field group): Applied Mathematics (Mathematical Science).

Keywords: time series analysis, feature engineering, autoencoder, detection, catch22.

Kaunas, 2022. 76 pages.

Summary

Master’s final degree project topic - Time series detection using autoencoder-based feature engineering. Main task – explore the use of features extracted from autoencoder for timeseries detection. The need for features and methods of conventional methods of extraction are discussed in the literature review section. Theory behind techniques used is more thoroughly explored in methodology section. Two slightly different architecture types of autoencoders are explored and evaluated, some latent space visualizations are provided. Timeseries when transformed to their latent space representation are used in training the detection model. Detection model is based on gradient boosting technique. To build the model 42 different sets of timeseries was used, totaling a total of 45 807 unique time series. To evaluate the model’s performance the features extracted from latent space have been compared with catch22 feature set. A joint autoencoder-catch22 model has been created, which has an AUC score equal to 0.9416, which is slightly worse than catch22 model alone which has an AUC score of 0.9484, but due to the modification made computational speed of the detection task has been more than doubled.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

LSTM – Long Short Term Memory (*liet.* Ilgos trumpos atminties (modelis))

t-SNE – t-distributed stochastic neighbor embedding (*liet.* t pasiskirstymo stochastinis kaimynų transformavimas)

UMAP – Uniform Manifold Approximation and Projection (*liet.* Tolydžios daugdaros aproksimavimas ir projektavimas)

PCA – Principal Component Analysis (*liet.* Principinių komponenčių analizė)

Terminai:

Autoenkoderis – dirbtinio neuroninio tinklo tipas skirtas dimensionalumo mažinimui taikant neprižiūrimo mokymosi metodus, transformuojant duomenis į modelio latentinę erdvę.

Latentinė erdvė – Transformuotų požymių į dažniausiai mažesnio dimensionalumo daugdara kurioje atstumas tarp dviejų duomenų taškų reprezentacijų gali būti interpretuojamas kaip dviejų taškų panašumo matas.

Požymis – Laiko eilutę apibendrinanti skaitinė statistika.

Klasifikavimas – objektų suskirstymas į tam tikras diskrečias grupes.

Detekcija – objektų suskirstymas į vieną iš dviejų grupių.

Konvoliucija – tiesinėje algebroje taikomas metodas, kuriuo matrica slenkančio lango principu skaliariškai sudauginama su tam tikra branduolio funkcija.

Aktyvacijos funkcija – dirbtiniuose neuroniniuose tinkluose taikoma netiesinė funkcija kuri pritaikoma kiekvieno neurono įvesties signalų svorinei sumai.

Įvadas

Klasifikavimas, arba siauresne prasme detekcija, yra viena iš problemų aktualių laiko analizės srityje. Spartus kompiuterinių resursų augimas leido sekti didžiulius kiekius įvairių laiko eilučių, todėl laikui bėgant vis labiau ir labiau augo laiko eilučių analizės aktualumas. Didžiosios internetinių technologijų įmonės kaip Google ar Amazon ir kitos kaupia didžiulius kiekius informacijos laiko eilutėmis, t. y. tinklalapių paspaudimai, paieškų skaičius, pajamos, vartotojų skaičius skirtingose platformose ir t.t.

Verslus laiko eilutės gali dominti dėl daugelio aspektų, pavyzdžiui techninių parametrų stebėjimas norint nustatyti ar stebimas procesas nenukrypsta nuo suprojektuotų veikimo normų, ar procesas yra arti gedimo, klaidos, ribos. Internetinėje aplinkoje stebint naudotojo požymius aptikti nesąžiningą naudojimąsi paslaugomis, pvz. įtarti kad tinklalapį varto ne žmogus, o automatizuotai renkami duomenys. Taip pat finansuose aptikti sukčiavimo atvejus, arba medicinoje patvirtinti arba atmesti tam tikro negalavimo hipotezę.

Laiko eilučių analitikoje, pvz. klasifikavimo ar prognozavimo uždaviniuose, kuomet yra naudojamas mašininio mokymosi meta-modelis yra aktualu dinamika aprašantys požymiai. Hyndman‘as [1] R pakete *tsfeatures* yra įgyvendinę daugelio pasiteisinusių požymių skaičiavimo formules kurios sėkmingai buvo naudojamos [3, 4] prognostinių modelių sudarymui.

Tyrimo tikslas duomenimis grįsta požymių reprezentacija laiko eilutėms, tinkanti detekcijos ir kitiems uždaviniams.

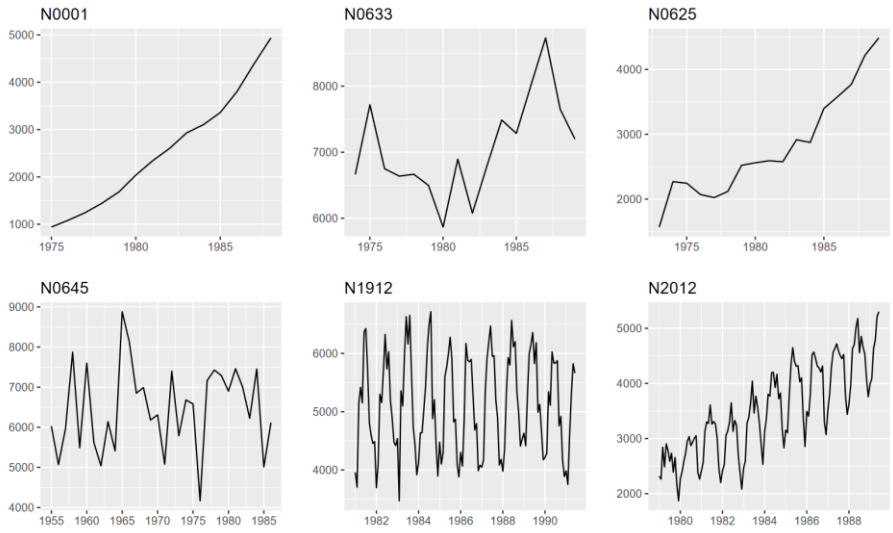
Darbo uždaviniai:

1. Sudaryti mokslinės literatūros apžvalgą analizuojančią požymių svarbą laiko eilučių kontekste.
2. Pasigilinti į tinkamas šiam uždaviniui autoenkoderių architektūras.
3. Sudaryti autoenkoderiu grįstą modelį apmokymui naudojant didelį skaičių laiko eilučių.
4. Gautą modelį pritaikyti laiko eilučių detekcijos uždaviniui spręsti, įvertinti jo rezultatus.
5. Palyginti detekcijos rezultatus taikant catch22 požymių rinkinį.
6. Darbe sudaryto autoenkoderio pritaikymas požymių rinkinio patobulinimui.

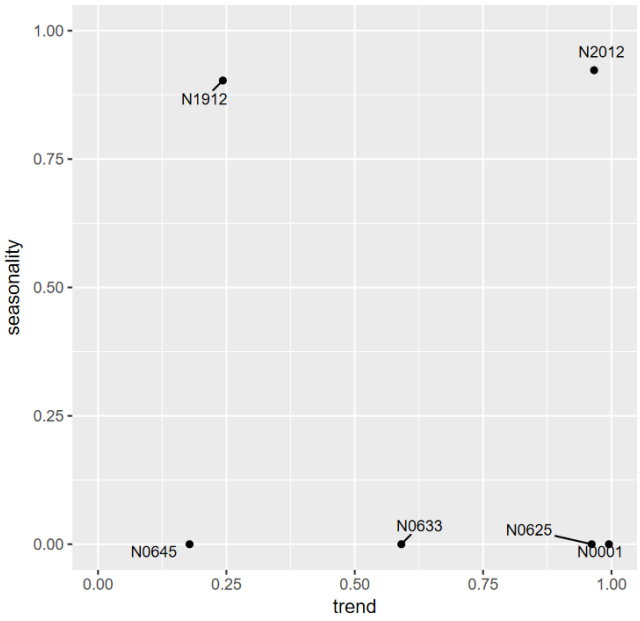
1. Literatūros apžvalga

1.1. Laiko eilučių požymiai

Talagala, [2] siūlo užuot dirbus su laiko eilučių stebėjimais tiesiogiai, jas analizuoti remiantis iš jų gauta požymių aibe. Laiko eilutės požymis yra bet kokia pamatuotina laiko eilutės charakteristika. Pavyzdys galetų būti 6 laiko eilutės iš M3 varžybų [4] (1 pav.) ir išskirti du jų požymiai (2 pav.) – sezoniškumo (angl. seasonality) ir krypties (angl. trend) stiprumai. Sezoniškumo ir krypties stiprumas yra suskaičiuoti metodais pasiūlytais Wang‘o [5]. Laiko eilutės kurių požymiai 2 pav. yra apatiniame dešiniajame kvadrante būdingas stiprus kryptiškumas, bet silpnas sezoniškumas, o viršutiniame dešiniame kvadrante yra laiko eilutės, šiuo atveju tik viena, turinti stiprų tiek sezoniškumą tiek kryptiškumą. Kiti galimi laiko požymiai yra autokoreliacija, periodiškumas, Furjė dekompozicijos rezultatai.



1 pav. Laiko eilučių atvaizdavimas laiko domene [4]



2 pav. Laiko eilučių požymiai [4]

1.2. Požymių reikšmė parenkant prognozavimo modelius

Reid‘as [6] vienas pirmųjų pastebėjo, kad atskirų metodų prognozavimo tikslumas kinta priklausomai nuo laiko eilučių požymių. Dėl šio efekto norint surasti optimalų modelį laiko eilutei tirti (prognozuoti, klasifikuoti) yra naudinga į modeliavimą vienaip ar kitaip įtraukti laiko eilučių požymius.

Yra išrasta daugybė prognozavimo metodų kurie grindžiami specifinėmis charakteristikomis kurios būdingos specifinėms, atskiroms disciplinoms. Pvz. GARCH modeliai buvo sukurti, kad gebėtų apskaičiuoti finansinių laiko eilučių nestabilumą bei jas prognozuoti, ETS modeliai gerai aproksimuoja kryptiškumą (angl. *trend*) ir sezoniškumą (angl. *seasonality*) kuris būdingas ketvirtiniam ar mėnesiniam įmonių finansiniams straipsniams, pvz., apyvartai, grynajam pelnui. Teisingai parinktas laiko eilučių požymių rinkinys turėtų padėti parinkti geriausią modelį ar modelių rinkinį specifinei laiko eilutei prognozuoti [2].

Jau kuris laikas tyrėjai bando sukurti modelius kurie remtųsi laiko eilučių požymiais [3, 5]. Neseniai buvo pasiūlytas automatizuotas metodas atrinkti laiko eilutes, padaugintas iš svorinių koeficientų prognozavimo metodų rinkiniui sudaryti [3]. Metodas yra dviejų žingsnių, pirma naudojantis laiko eilučių rinkiniu yra apmokomas meta-modelis ir priskiriami svoriai įvairiems galimiems prognozavimo metodams minimizuojant vidutinę prognozavimo paklaidą. Meta-modelio įeities parametrai yra laiko eilučių požymiai gaunami matematiškai kiekvienai laiko eilutei atskirai. Antruoju žingsniu yra atliekamas prognozavimas su prieš tai suskaičiuotais svoriniais koeficientais modelių rinkiniui. Šis metodas M4 varžybose pasiekė antrąją vietą [8].

1.3. Požymių atradimas ne laiko eilučių kontekste

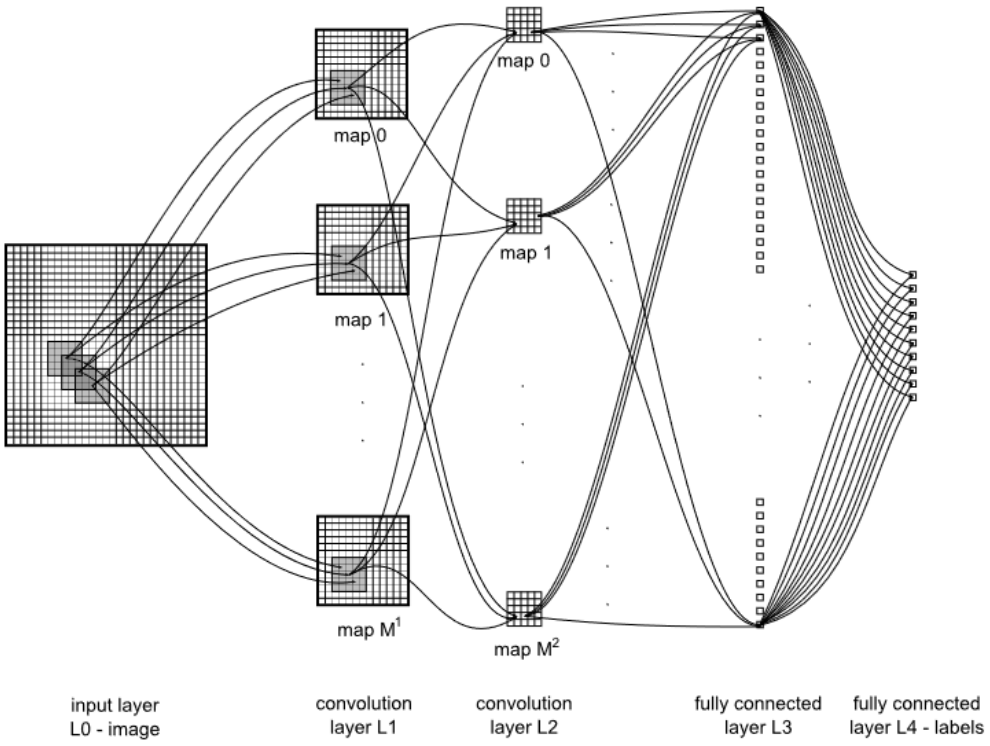
Iš dalies automatinis požymių atradimas yra jau kelis dešimtmečius naudojamas vaizdų atpažinimo domene. Čia sėkmingai yra naudojami konvoliuciniai neuroniniai tinklai (CNN) įvairių objektų nuotraukoms atpažinti ir klasifikuoti.

Konvoliuciniai neuroniniai tinklai yra taip vadinami dėl jų architektūros. Tokio tipo neuroninio tinklo struktūra: pirmas yra pasirinktinas paveikslėlio išankstinio apdorojimo sluoksnis, kur naudojami iš anksto nustatyti filtrai kurie nekinta apmokymo metu, jais gali būti pažymimi objektų kraštai, spalvų gradientinis filtras. Toliau naudojamas konvoliucinis (arba sąsūkos) sluoksnis ar sluoksniai pagal kuriuos šis neuroninių tinklų tipas ir yra pavadintas. Jų paskirtis yra aptikti atskirus bruožus kuriuos apmokymo metu bandoma identifikuoti kaip reikšmingus pateiktoms mokymo duomenų klasėms klasifikuoti. Toliau naudojamas surinkimo sluoksnis kuriame konvoliucijų išeities duomenys yra sutraukiami galiausiai rezultatus pateikiant klasifikavimo sluoksniui. Atlikus šį veiksmą gaunamas galutinis modelio prašomas klasifikavimo atsakymas [9].

Tajbakhsh‘as [10] rašo, jog pagrindiniai iššūkiai norint sukurti naują konvoliucinių neuroninių tinklų modelį yra didelio žymėtų duomenų rinkinio poreikis, kur priklausomai nuo domeno, šiuo atveju medicininio, gali būti sunku ar neįmanoma šių duomenų gauti. Net ir turint didelį kiekį duomenų modelio apmokymas yra sunkus skaičiavimų prasme, reikia didelio kiekio kompiuterinių resursų, be kurių apmokymas truktų neįtikėtinai ilgai. Galiausiai konvoliucinių neuroninių tinklų apmokymas yra labai jautrus persimokymui, tad galima tikėtis jog reikės architektūros ir mokymosi parametrų optimizavimo, kad būtų užtikrinta jog visi tinklo sluoksniai yra apmokomi panašiu greičiu. Kaip to

alternatyva yra pateikiamas siūlymas naudoti iš anksto apmokytus neuroninius tinklus kitai aplikacijai, modelį tik papildant.

Azizpour‘as [11] teigia jog, kad apmokyto modelio adaptavimo kitam domenui sėkmė priklauso nuo atstumo, arba panašumo, tarp duomenų kuriais buvo modelis apmokytas ir duomenų kuriems modelį bandoma adaptuoti.



3 pav. Konvoliucinio neuroninio tinklo struktūra [10]

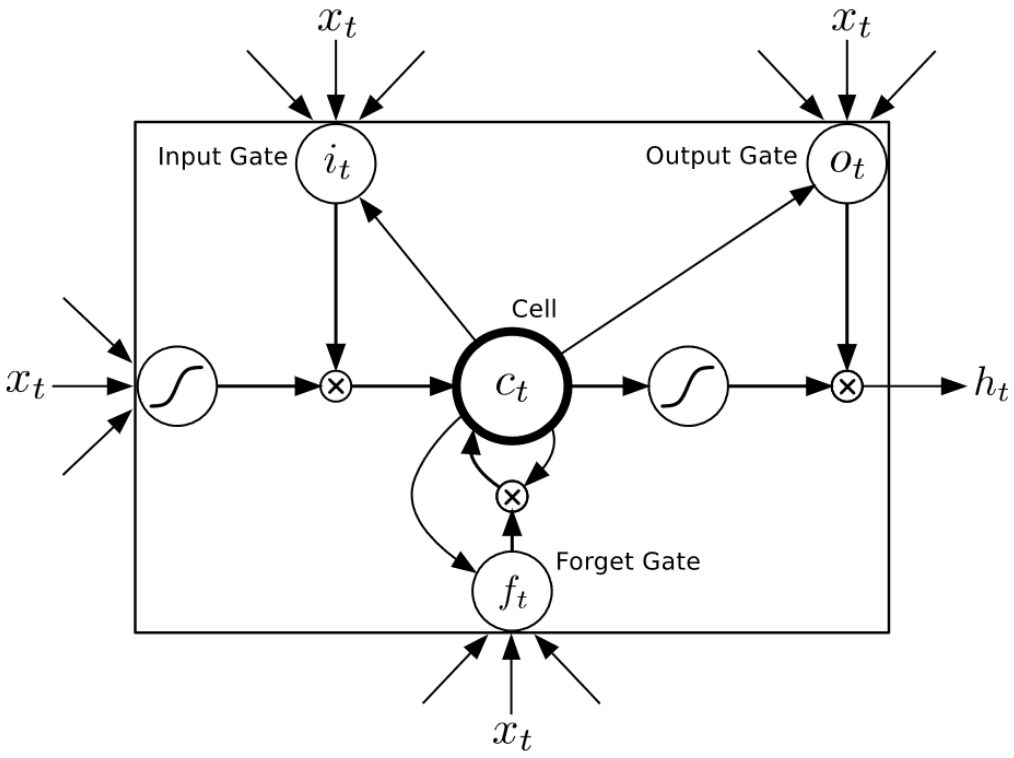
1.4. Neuroniniai tinklai laiko eilutėms

Įprastinių tiesioginės sklaidos neuroninių tinklų modeliams būdinga nepriklausomumo laike prielaida, todėl sekvenciniams duomenims, tokiems kaip laiko eilutės, jie nėra taikomi, vienas iš sprendimo būdų yra rekurentinių neuroninių tinklų naudojimas.

Rekurentiniai neuroniniai tinklai yra tinkami naudoti duomenų sekoms, kadangi geba aptikti duomenų priklausomybę laike dėl to, jog jų architektūroje yra įgyvendinta atmintis, t.y. jie geba „atsiminti“ prieš tai ėjusius duomenis tuo momentu kai jiems yra pateikiamas naujas duomuo [12]. Kiekvieno žingsnio metu yra nuskaitomas duomenų $x = (x_1, x_2, \dots, x_T)$ vektorius ir po kiekvieno žingsnio modelio viduje yra suformuojama paslėpta būsena h_t, kuri kito žingsnio metu paduodama atgal kaip įeities vektorius šalia naujų duomenų.

Šio tipo modeliai yra neginčijamai efektyvūs sekvenciniams duomenims modeliuoti, bet turi ir apribojimų dėl savo atminties. Jiems yra būdinga nykstančio gradiento problema, kuri kyla jeigu reikalaujamas išvesties signalas žingsnyje t yra priklausomas nuo daug ankstesnio žingsnio. Šiai problemai spręsti buvo pasiūlyti LSTM modeliai [13]. Graves‘as [14] šiuos modelius aiškina, kaip rekurentinių neuroninių tinklų architektūros patobulinimą informacijai išsaugoti ilgesnį laiko tarpą. Šaltinio straipsnio [14] parašymo metu LSTM modeliai buvo lyderiaujantis modelių tipas tarp

daugelio su sekos tipo duomenimis susijusių uždavinių, pavyzdžiui, tarp kalbos ar rašysenos atpažinimo užduočių.



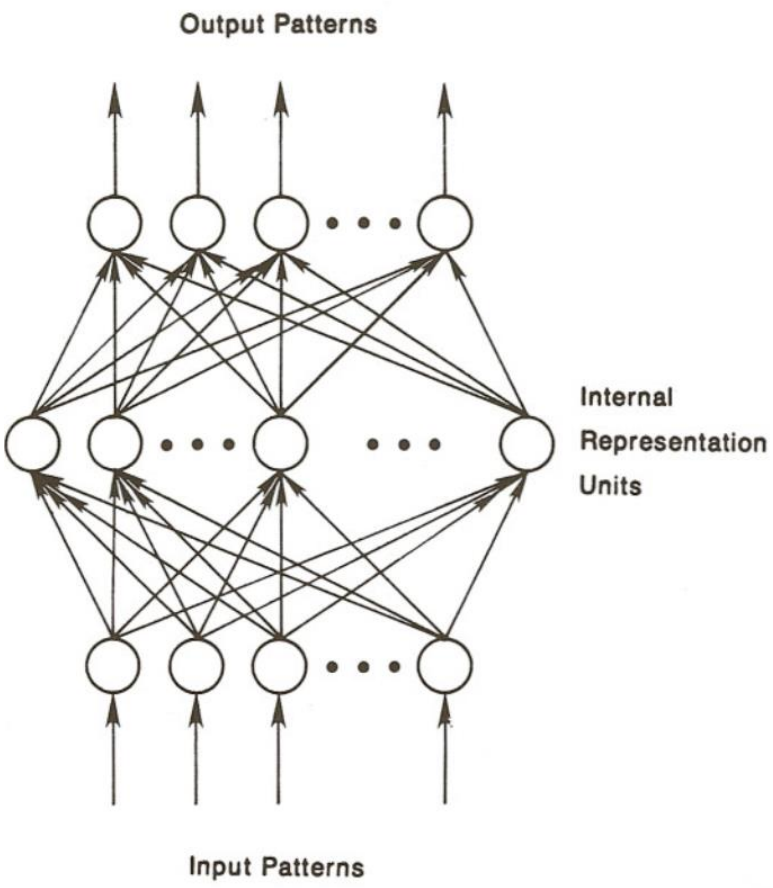
4 pav. LSTM modelio celės architektūra [14]

1.5. Autoenkoderiai ir seq2seq modeliai

Autoenkoderių paskirtis yra duomenų dimensionalumo mažinimas automatiškai surandant pagrindines mokymuisi naudoto duomenų rinkinio charakteristikas. To yra pasiekiama kaip tikslo funkciją nurodant tuos pačius duomenis, priverčiant optimizavimo algoritmus konverguoti ties sprendiniu kuris duotus duomenis efektyviausiai suspaudžia, palikdamas tik esmines, daugiausia informacijos turinčias duomenų rinkinio charakteristikas pagal kurias pasiekiamas didžiausias atkūrimo tikslumas. Tai yra vienas iš seq2seq modelių šeimai priklausančių modelių. Šiam modeliui pateikiama duomenų seka ir modelis sugeneruoja naują (pageidautina identišką pateiktajai) duomenų seką.

Rumelhart‘as [15] pirmasis savo knygoje aprašė neuroninių tinklų modelį kuris vėliau pavadintas autoenkoderiu. Iš esmės autoenkoderio modelį sudaro dvi dalys, tai yra enkoderis ir dekoderis. Enkoderio užduotis yra surasti iš anksto nežinomus požymius iš duomenų rinkinio pagal kuriuos dekoderis galėtų atlikti priešingą veiksmą ir naudojantis surastais požymiais kuo tiksliau atkurti pradinius duomenis. Taip duomenys yra priverčiami pereiti pro butelio kaklelį, kas gali būti interpretuojama kaip dimensionalumo mažinimas, kitaip tariant duomenys yra suspaudžiami, arba transformuojami, į latentinę erdvę.

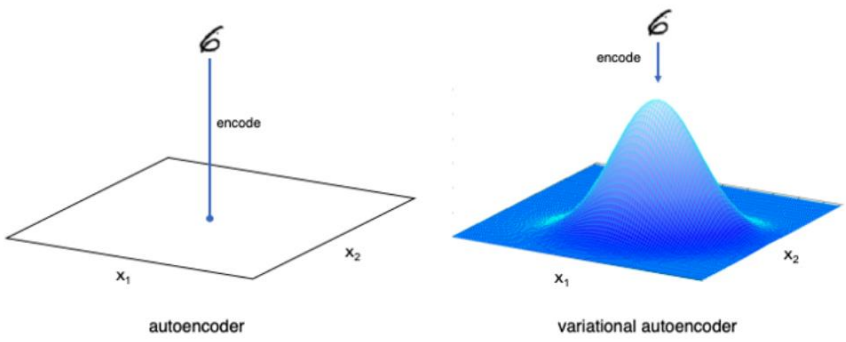
Naudojant autoenkoderius reikia atkreipti dėmesį, kad didinant autoenkoderio laisvės laipsnių skaičių bei jo netiesiškumą, siekiant išgauti kuo geresnę atkuriamų duomenų kokybę, duomenys taps sunkiau interpretuojami, nes latentinė erdvė taps vis mažiau reguliari. Tai yra du latentinės erdvės taškai esantys netoli vienas kito gali duoti du labai skirtingus rezultatus.



5 pav. Rumelhart modelio struktūra [15]

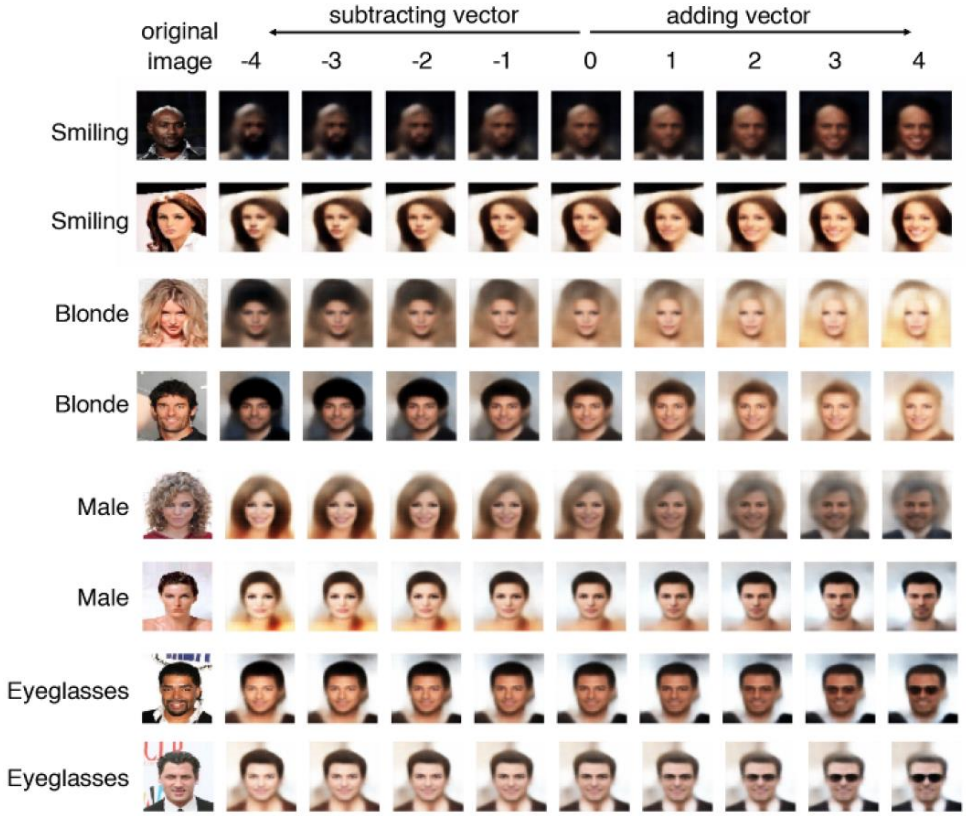
1.6. Variaciniai autoenkoderiai

Kingma [16] pasiūlė sprendimą autoenkoderių latentinės erdvės nereguliarumui sutvarkyti. Autorius pasiūlė būdą reguliarizuoti latentinę erdvę pritaikant tikimybinius modelius. Įprastame autoenkoderyje kiekvienas duomenų taškas būtų susiejamas su vienu specifiniu tašku latentinėje erdvėje, tuo tarpu variaciniuose autoenkoderiuose susiejimas atliekamas kiek kitaip, kiekvienas duomenų taškas gali būti susiejamas su bet kuriuo latentinės erdvės tašku, pagal tikimybę pagal normalųjį skirstinį. Tas priverčia optimizavimo algoritmą optimizuoti enkoderį taip, jog dviejų latentinės erdvės taškų panašumas būtų atvirkščiai proporcingas atstumui tarp tų taškų.



6 pav. Autoenkoderio ir variacinio autoenkoderio duomenų taško pasiskirstymas latentinėje erdvėje [17]

Foster‘is [17] savo knygoje demonstruoja kaip latentinės erdvės vektorius interpretuoti veidams generuoti. Pasirenkant du taškus erdvėje ir fiksuotu žingsniu „einant“ lantentinės erdvės atkarpa bei generuojant veidus yra pasiekiamas tolygus perėjimas nuo vieno asmens veido iki kito. Arba suradus norimo atributo vektoriaus kryptį ir keičiant latentinės erdvės reikšmes šio vektoriaus kryptimi galima priversti modelį atlikti norimą pokytį veidui, pvz. uždėti akinius, pakeisti akių spalvą, lytį ir t. t.. Todėl variacinių autoenkoderių modeliai yra tinkami naujam turiniui generuoti bei požymiams tirti.



7 pav. Portretų modifikavimas taikant variacinį autoenkoderį [17]

1.7. Autoenkoderiai anomalijų aptikimui

Zimek‘as [18] enciklopedijoje anomalijų aptikimas yra apibrėžiamas kaip tikslas surasti duomenų taškų, kurie dėl duomenų užterštumo, žmonių ar sisteminių klaidų yra pakankamai nutolę nuo teisingų duomenų taškų pavyzdžių. Statistiškai buvimas „neįprastu“ yra įvertinamas pagal tai kiek duomenų taškas ar seka neatitinka modelio sudaryto išskirčių pasiskirstymo.

Pereira [12] sėkmingai pritaikė požymių inžinerija naudojantis autoenkoderiais anomalijų aptikimui atsinaujinančios energetikos sektoriuje, analizuojant elektros gamintojų išmaniųjų skaitliukų surinktus duomenis. Pagrindinė autoenkoderių taikymo anomalijų aptikimui mintis yra vietoje anomalijų modeliavimo, autoenkoderiai modeliuoja normalią sistemos būklę. Po apmokymo autoenkoderis geba rekonstruoti normaliai veikiančios sistemos duomenų seką, tačiau anomalijų ne. Taigi lyginant modelio rekonstruotas ir tikras duomenų sekas galima vertinti jų skirtumą ir matuoti skaitinį anomalijos įvertį.

Pats įgyvendinimas atliktas pritaikant variacinius autoenkoderius su Bi-LSTM architektūra. Kadangi pritaikyti variaciniai autoenkoderiai tampa įmanoma skaičiuoti atkūrimo tikimybės vertes pagal An [19] pasiūlytą metodiką pritaikant Monte Karlo integravimą. Teigiama, kad tai yra geresnis anomalijų

įvertinimo matas, nei atkūrimo paklaida taikoma kituose autoenkoderiais ar principinėmis komponentėmis grįstais anomalijų aptikimo metodais.

Latentinės erdvės atvaizdavime matoma, kad saulės jėgainių apskaitos sugeneruoti duomenys priklausomai nuo paros meto latentinėje erdvėje atsiduria panašiose vietose ir stebima ciklinė trajektorija, su 1 dienos periodiškumu. Galima sakyti modelis išmoko informaciją apie duomenų periodiškumą net atskirai to nenurodžius.

8 pav. Pereira anomalijų aptikimo modelio latentinės erdvės vizualizavimas 2D t-SNE metodu (kairėje) ir pirmomis dvejomis principinėmis komponentėmis (dešinėje) [12]

1.8. Autoenkoderių panaudojimas požymių atradimui

Autoenkoderių pritaikomumą požymių inžinerijoje galima rasti tiek kibernetinėje apsaugoje [20], tiek energijos sąnaudų prognozavime [21], tiek gamybinėje pramonėje [22]. Automatinę požymių inžineriją prasminga pritaikyti srityse, kuriose dėl nuolatinių pokyčių, nėra įmanoma rasti vieno rinkinio ekspertinėmis įžvalgomis grįstų požymių [23].

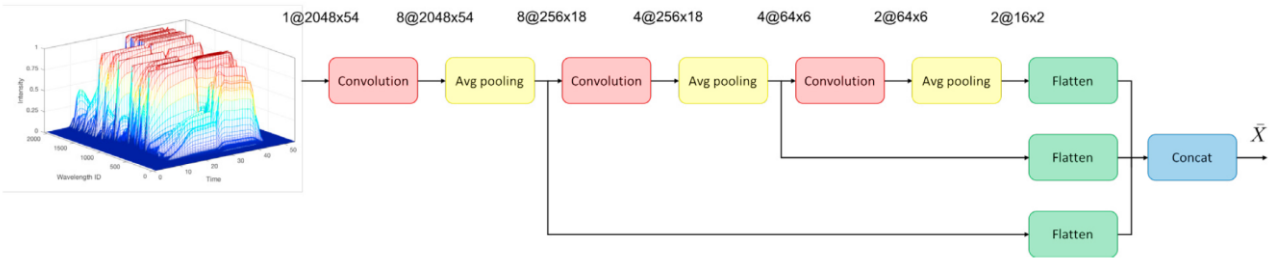
Maggipinto‘sas [22] atliko automatinę požymių inžineriją puslaidininkinių įtaisų gamybos industrijoje. Konkrečiai virtualioje metrologijoje, kur lustų fizinė kokybė yra įprastai vertinama atliekant netiesioginius stebėjimus ir taikant įvairius statistinius metodus.

Šioje industrijoje dažniausiai taikomas rankinis požymių sudarymas, kuris palyginus su automatiniu, trunka ilgai, reikalauja technologinio proceso ekspertų įžvalgų, bei dėl puslaidininkių industrijos specifikos kompleksiškumo sudaryti požymiai yra prastai generalizuojami.

Maggipinto‘sas [22] darbe požymių inžinerija atlikta naudojantis optinės emisijos spektroskopijos (OES) duomenimis. OES duomenų dvidimensiška bei fiksuoto dydžio prigimtis, tyrėjam savaime siūlė taikyti kompiuterinio matymo įkvėptas metodologijos, konkrečiai konvoliucinius neuroninius tinklus kurie yra vieni sėkmingiausių šio domeno modelių.

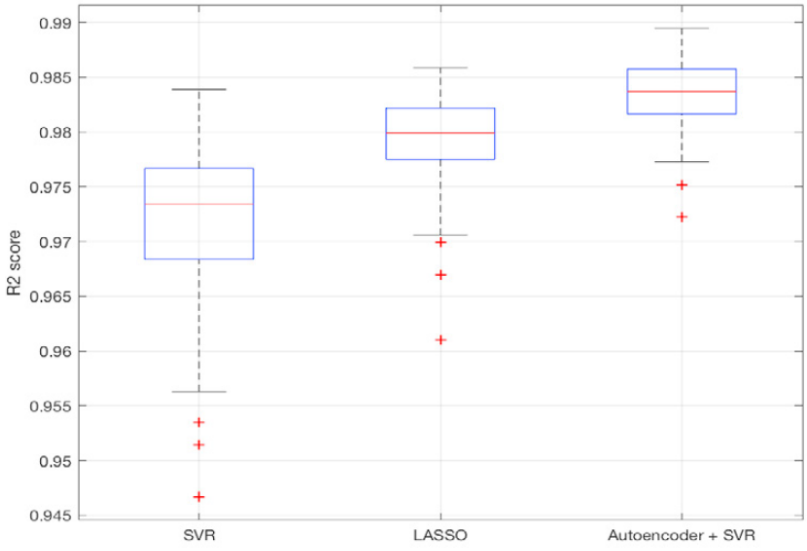
Enkoderio dalyje naudoti trys konvoliuciniai sluoksniai, tarp jų taikant vidurkių surinkimo sluoksnius, kas užtikrina konvoliucijų surastų požymių tolydumą ir tolesnį tinkamumą regresinei analizei. Gautas požymių vektorius tris kart sumažino duomenų dimensiją iki p = 38464, nuo matuotų 2014 bangos ilgių x 54 laiko žingsniais. Surinkti požymiai, norint įvertinti modelio efektyvumą,

atraminių vektorių regresijos metodu naudoti lustų graviravimo tempo (vieno iš esminių virtualios metrologijos pagalba atliekamų technologinių stebėjimų) prognozavimui.



9 pav. Maggipinto požymių inžinerijos metodo struktūra [22]

Pasiūlytas metodas lygintas su dažniausiai taikomais metodais, konkrečiai principinių komponenčių ir statistinių momentų skaičiavimo metodais. Modelio rezultatai pranoko visus lygintus tradicinius metodus, kas įrodo autoenkoderių potencialą požymių inžinerijoje.



10 pav. Maggipinto lyginamų metodų R^2 rezultatų palyginimas [22]

Panašų metodą taikė Fan‘as [21] pastatų energijos sąnaudų prognozavime. Moderniuose pastatuose, taikant išmaniąsias pastatų valdymo technologijas per pastarąjį dešimtmetį buvo surinkti dideli kiekiai duomenų, kas sudarė sąlygas prognozuoti energijos sąnaudas nesiremiant vien tik ekspertų įžvalgomis, bet atlikti ir duomenimis grįsta analizę bei prognozę.

Tam pasiekti buvo išbandyti pilnai sujungtų neuronų architektūros autoenkoderis, konvoliucinis autoenkoderis bei GAN (Generative Adversarial Network) tipo modeliai. Jais sudarytais požymių rinkiniais atliktas prognozavimas ir lygintas su ekspertų bei statistiniais metodais sudarytų požymių rinkiniais remtu prognozavimu. Šiuo atveju nustatyta, jog bendrai tiksliausia prognozavimą siūlo GAN modeliai (14% mažesnis RMSE įvertis lyginant su ekspertiniu požymių rinkiniu), tačiau pilnai sujungtais autoenkoderiais bei konvoliuciniais autoenkoderiais grįsti metodai taip pat ženkliai sumažina paklaidos įvertį (7% ir 8% atitinkamai mažesnis RMSE).

1.9. Laiko eilučių požymių rinkiniai

1.9.1. Catch22 požymių rinkinys

Metodo palyginimui naudojama Lubba [24] sudarytas catch22 požymių rinkinys. Tyrėjai iš 7658 požymių esančių hctsa rinkinyje atrado 22 požymius kuriuos taikant klasifikavimo uždavinio tikslumas lyginant su viso požymių rinkinio taikymu sumažėjo vos 6,5% testuojant su 147000 laiko eilučių (71,7% lyginant su 77,2%). Sumažinus požymių skaičių, taip pat yra teigiama kad skaičiavimų trukmė sutrumpėjo tūkstanteriopai. Visas catch22 požymių rinkinys yra aprašytas **1 lentelė**. Laiko eilučių požymių įvertinimui naudotas UCR/UEA (University of East Anglia and University of California) laiko eilučių klasifikavimo duomenų bazė.

Kadangi beveik visos laiko eilutės esančios šioje duomenų bazėje yra z normalizuotos, buvo iš karto pašalinti 766 laiko eilučių požymiai netinkantys tokio tipo duomenims. Yra pažymima, jog tam tikriems duomenų rinkiniams šie pašalinti požymiai gali būti vertingi ir smarkiai pagerinti uždavinio rezultatus, tad tam tikrais atvejais šie atrinkti požymiai gali nebūti geriausias pasirinkimas [24].

Taip pat pašalinti požymiai kurių skaičiavimo metu gaunamos specialiosios reikšmės. Požymiai kurie 80% duomenų rinkinių bent vienai laiko eilutei nesugebėdavo suskaičiuoti tikslios reikšmės buvo pašalinti iš požymių rinkinio. Taip buvo pašalintos 2101 laiko eilutės ir dirbama buvo su 4791 laiko eilute [24].

Gan didelis išankstinis požymių rinkinio susiaurinimas yra pasekmė griežto tyrėjų pasirinkto atrankos metodo, ieškančio patikimai realias reikšmes suskaičiuojančių plačiam laiko eilučių spektrui metodų, iškart atmetant mažiau universalius požymius (pavyzdžiui reikalaujančius minimalaus skaičiaus stebėjimų, tik teigiamų stebėjimų reikšmių, arba netinkamų monotoniškoms laiko eilutėms).

Skirtingai nuo tipinių požymių atrankos algoritmų kurie ieško požymių kombinacijų kurias taikant kartu galima pasiekti geriausių rezultatų, tyrėjai pasirinko požymius atrinkti pagal tai kiek jie gerai individualiai tinkami duomenims tirti ir papildantys vienas kitą.

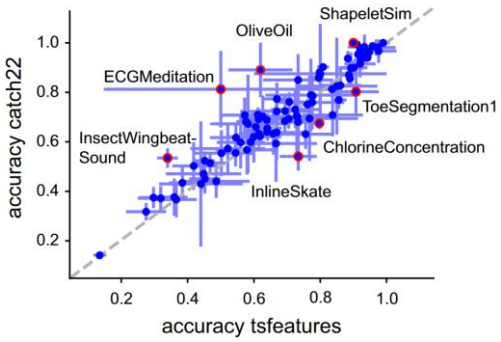
1 lentelė Catch22 požymių rinkinys [24]

Hctsa požymio pavadinimas	**Paaiškinimas angl.**
Pasiskirstymas	
DN_HistogramMode_5	Z transformuotos laiko eilutės moda (5 intervalų histograma)
DN_HistogramMode_10	Z transformuotos laiko eilutės moda (10 intervalų histograma)
Paprastos laiko statistikos	
SB_BinaryStats_mean_longstretch1	Ilgiausias reikšmių esančių virš vidurkio periodas
DN_OutlierInclude_p_001_mdrmd	Laiko intervalas tarp pasikartojančių išskirčių virš vidurkio
DN_OutlierInclude_n_001_mdrmd	Laiko intervalas tarp pasikartojančių išskirčių žemiau vidurkio

Tiesinė autokoreliacija	
CO_f1ecac	Pirma autokoreliacijos funkcijos 1/e kirtimo vieta
CO_FirstMin_ac	Pirmas autokoreliacinės funkcijos minimumas
SP_Summaries_welch_rect_area_5_1	Furjė galios spektro žemiausių penktadalio dažnių suminė galia
SP_Summaries_welch_rect_centroid	Furjė galios spektro centroidas
FC_LocalSimple_mean3_stderr	Vidutinė paklaida prognozuojant 3 paskutinių dydžių slenkančio vidurkio principu
Netiesinė autokoreliacija	
CO_trev_1_num	Laiko apgrežiamumo statistika
CO_HistogramAMI_even_2_5	Savitarpio informacija, m = 2, τ = 5
IN_AutoMutualInfoStats_40_gaussian_fmmi	Pirmas savitarpio informacijos funkcijos minimumas
Skirtumai	
MD_hrv_classic_pnn40	Pasikartojančių skirtumų viršijančių 0.04 proporcija
SB_BinaryStats_diff_longstretch0	Ilgiausias funkcijos mažėjimo periodas
SB_MotifThree_quantile_hh	Šanono entropija diskrizuotai laiko eilutei (dviem iš paskos einantiems intervalams)
FC_LocalSimple_mean1_tauresrat	Koreliacijos ilgio pokytis apskaičiavus laiko eilutės skirtumus
CO_Embed2_Dist_tau_d_expfit_meandiff	Eksponentinės funkcijos klaida prognozuojant tolimesnes laiko eilutės reikšmes
Svyravimų analizė	
SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1	Trumpesnių laiko intervalų svyravimas kurie yra proporcingi linkmės eliminavimo fliuktuacinės analizės metodo rezultatams
SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1	Trumpesnių laiko intervalų svyravimas kurie yra proporcingi su R/S analizės rezultatais
Kiti	
SB_TransitionMatrix_3ac_sumdiagcov	3 simbolių abėcėlės perėjimo kovariacinės matricos pėdsakas
PD_PeriodicityWang_th0_01	Periodiškumo matas

1.9.2. tsfeatures požymių rinkinys

Hyndman‘o [1] sudarytą tsfeatures požymių rinkinį sudaro 30 laiko eilučių požymių. Tai yra daugelio mokslininkų laiko eilutėm tirti siūlomų požymių rinkinys aprašančių entropiją, autokoreliacijas, sezoniškumą ir t.t.. Šį požymių rinkinį palyginimui naudojo catch22 autoriai.



11 pav. catch22 ir tsfeatures požymių klasifikavimo uždavinio tikslumo palyginimas [24]

Šį požymių rinkinį lyginant su Catch22 požymių rinkiniu UCR/UCA duomenų bazės laiko eilutėms analizuoti pasiekiami labai panašūs rezultatai. Skaičiuojant tikslumą subalansuotoms klasėms viso duomenų rinkinio laiko eilutėms abiejų rinkinių tikslumo rezultatai buvo labai panašūs, Pearson koreliacijos koeficientas r = 0,93. Vidutinis tikslumas taikant Catch22 duomenų rinkinį buvo nežymiai didesnis (71,2%) lyginant su tsfeatures duomenų rinkinių (69,4%) [24].

1.9.3. Kiti požymių rinkiniai

Christ‘as [25] sudarė metodą tinkamų, prasmingų, laiko eilučių požymių suradimui ankstyvoje mašininio mokymo proceso stadijoje, kas, be ekspertinių įžvalgų, ar iteracinių metodų ieškant optimalaus požymių rinkinio, pateikia prasmingų tam tikrai laiko eilutei požymių rinkinį. Iš viso metodo požymių aibę sudaro 794 požymių, iš kurių kaip prasmingi atrenkama tik dalis.

Nun‘as [26] sukurtas FATS (Feature Analysis for Time Series) požymių rinkinys yra kurtas su mintimi pritaikyti požymių suradimui astronominiuose duomenyse, tačiau yra pritaikomas ir kitose srityse. Požymių rinkinį sudaro 64 požymiai.

Dempster‘is [27] išleido požymių sudarymo metodą pavadintą ROCKET (**R**and**O**m **C**onvolutional **KE**rnel **T**ransform). Daugelis kitų laiko eilučių klasifikavimo metodų su geriausiais tikslumo įverčiais reikalauja daug kompiuterinių resursų atlikti skaičiavimams net ir nedideliems duomenų rinkiniams ir yra nepanaudojami kelis kartus, modelius būtina kas kartą apmokyti iš naujo. Šis metodas paremtas tiesiniu klasifikatoriumi su atsitiktiniais konvoliuciniais branduoliais pasiekia geriausius tikslumo įverčius nereikalaudamas didelių kompiuterinių resursų. Tyrėjai teigia, jog šiuo metodu įmanoma apmokyti ir išbandyti su visais 85 duomenų rinkiniais UCR archyve per greičiau nei 2 valandas.

Facebook [28] išleido Kats (Kits to Analyze Time Series), tai yra Python programavimo aplinkai skirta biblioteka visokeriopai laiko eilučių analizei – nuo požymių radimo iki mašininio mokymo modelių sudarymo. Bibliotekos TSFeatures modulis geba skaičiuoti 65 skirtingus laiko eilutės požymius su aiškiomis statistinėmis reikšmėmis, kuriuos vėliau tos pačios bibliotekos aplinkoje galima pritaikyti mašininio mokymo klasifikavimo, regresijos uždaviniams. Darbo rašymo metu dėl šios bibliotekos naujumo dar nėra išleista ją pritaikiusių mokslinių straipsnių.

1.10. Laiko eilučių požymių rinkinių panaudojimas

Praeitame skyriuje minėti laiko eilučių rinkiniai daugelyje straipsnių naudojami įvairioms paskirtims, tokioms kaip klasifikavimas, prognozavimas, siūlomo kito metodo lyginimui, sintetinių duomenų generavimui.

2 lentelė laiko eilučių požymių bibliotekų pritaikymas mokslinėje literatūroje

Autoriai	Straipsnio pavadinimas	Naudotas požymių rinkinys	Požymių pritaikymas
Angus Dempster, François Petitjean, Geoffrey I. Webb	ROCKET: exceptionally fast and accurate time seriesclassification using random convolutional kernels	Catch22	Palyginimas su kitu siūlomu požymių parinkimo metodu (ROCKET)
Olivier Tessier-Larivière, Luke Y. Prince, Pascal Fortier-Poisson, Lorenz Wernisch, Oliver Armitage, Emil Hewage, Guillaume Lajoie, Blake A. Richards	PNS-GAN: Conditional Generation of Peripheral Nerve Signals in theWavelet Domain via Adversarial Networks	Catch22	Sugeneruotų sintetinių ir tikrų vilnelių atstumo mato apskaičiavimas catch22 požymių erdvėje.
Mahdi Abolghasemi, Eric Beh, Garth Tarr, Richard Gerlach	Demand forecasting in supply chain: The impact of demand volatility in the presence of promotion	Tsfeatures	Paklausos prognozavimas
Shaohui Ma, Robert Fildes	Retail sales forecasting with meta-learning	Tsfeatures	Pardavimų prognozavimas
Jean Paul, Latyr Faye	Nouvelle méthode de prédiction d'échec de modèles prédictifs et de détection de changements de régimes dans les séries chronologiques	Tsfeatures	Laiko momento nuo kurio modelis praranda savo prognozavimo gebėjimą aptikimas.

Abolghasemi, M., Hyndman, R., Spiliotis, E., & Bergmeir, C	Model selection in reconciling hierarchical time series	Tsfeatures	Prognozavimo modelio parinkimo uždavinys
Xiaoqian Wang, Yanfei Kang, Fotios Petropoulos, Feng Li	The uncertainty estimation of feature-based forecast combinations	Tsfeatures	Prognozavimo modelių rinkinio svorinių įverčių parinkimas prognozuojant modelių neapibrėžtumą.
Yanfei Kang, Rob J. Hyndman, Feng Li	GRATIS: GeneRAting TIme Series with diverse and controllable characteristics	Tsfeatures	Sintetinių laiko eilučių generavimas su nurodytais požymiais
Maximilian Christ, Andreas W. Kempa-Liehr, Michael Feindt	Distributed and parallel time series feature extraction forindustrial big data applications	Tsfresh	Tsfresh bibliotekos autorių demonstracija parodanti metodo efektyvumą klasifikavimo ir klasterizavimo uždaviniams pramoniniame domene.
Vera Schroeder, Ethan D. Evans, You-Chi Mason Wu, Constantin-Christian A. Voll, Benjamin R. McDonald, Suchol Savagatrup, and Timothy M. Swager	Chemiresistive Sensor Array and Machine Learning Classification of Food	Tsfresh	Kompleksinių kvapų klasifikavimas naudojantis cheminių sensorių duomenimis.
Martin Rätz,Amir Pasha Javadi,Marc Baranski,Konstantin Finkbeiner,Dirk Müller	Automated data-driven modeling of building energy systems via machine learning algorithms	Tsfresh	Pastatų energetinių sistemų veikimo prognozavimas, siekiant tobulinti inžinerinio projektavimo kokybę.
Shih-Yuan Yu, Arnav Vaibhav Malawade, Sujit Rokka Chhetri, Mohammad Abdullah Al Faruque	Sabotage Attack Detection for Additive Manufacturing Systems	Tsfresh	Sabotažo aptikimas industrinėse 3D spausdinimo vis.
Donato Tiano, Angela Bonifati, Raymond Ng	Feature-driven Time Series Clustering	Tsfresh	Laiko eilučių klasterizavimas pagal požymius

Karim Pichara, Pavlos Protopapas, Daniel León	Meta-Classification for Variable Stars	FATS	Kintamųjų žvaigždžių klasifikavimas pagal šviesos kreivių požymius
Anthony Brunel, Johanna Pasquet, Jérôome Pasquet, Nancy Rodriguez, Frédéric Comby, Dominique Fouchez, Marc Chaumont	A CNN adapted to time series for the classification of Supernovae	FATS	Klasifikavimo uždavinys I.a tipo supernovai atskirti nuo kitų supernovų pagal šviesos kreivės požymius.

Pagal Wolpert'o [29] „No free lunch theorem" neegzistuoja toks optimizavimo metodas kuris tiktų visoms įmanomoms klasėms optimaliai. Bet kokios metodo geresnės savybės tam tikrai klasei, bus atsvertos prastesnėmis savybėmis kitose klasėse. Tą galima apibendrinti ir laiko eilutėms, kad neegzistuoja toks prognozavimo algoritmas, kuris visoms įmanomoms laiko eilučių klasėms prognozavimą atliktų geriausiai. Todėl daugelis autorių bando surasti metodą, pagal kurį laiko eilutes būtų galima išskirti į atskiras klases ir taip supaprastinti geriausio algoritmo iš aibės prognozavimo algoritmų parinkimo uždavinį. 1 lentelėje keliuose minėtuose straipsniuose bandoma pritaikyti laiko eilučių požymius algoritmų parinkimui. Suradus laiko eilučių požymius tampa daug lengviau laiko eilutes tarpusavyje lyginti ir grupuoti, žinant algoritmus kurie gerai veikė panašioms laiko eilutėms, galima iš anksto pasakyti kurie gerai veiks su dar neišbandytomis laiko eilutėmis.

Tačiau ne visada yra prasminga remtis maksimaliai dideliu laiko eilutės požymių sąrašu, tiek dėl skaičiavimų trukmės, tiek dėl egzistuojančių specifinių domenui įrankių, kuriuose su srities ekspertų įžvalgomis yra atrinkti požymiai turintys prasmę analizuojamoje sferoje. Kaip pavyzdys to yra Nun'o [26] sudarytas požymių rinkinys kuris yra pritaikytas astronominės kilmės šviesos analizei. Šiame požymių rinkinyje yra 64 požymiai, lyginant su tsfresh bibliotekos siūlomais 784 yra ženklus supaprastinimas dėl kurio tiek kompiuteriniai resursai yra naudojami efektyviau, tiek gauti rezultatai srities tyrėjų yra interpretuojami lengviau, kas gali būti svarbiau nei modelio klasifikavimo ar prognozavimo tikslumas.

Atskirose sferose ne visada egzistuoja pakankamai viešai prieinamų laiko eilučių rinkinių, tad tyrėjams surinkti pakankamai duomenų klasifikavimui ar prognozavimui tėra du būdai. Vienas iš jų yra brangus laiko ir finansų prasme tradicinis realaus pasaulio duomenų rinkimo būdas, arba kitas – generuoti sintetines laiko eilutes pritaikytas tyrėjų poreikiams. Kitas dažnas poreikis sintetinėms laiko eilutėms yra bendrinių laiko eilučių analizės metodų įvertinimas. Muñoz'as [30] teigia, jog nepaisant šiuo metu egzistuojančių didžiulių laiko eilučių rinkinių (UCI repository, M3, M4 varžybų ir kiti rinkiniai) tie rinkiniai yra gana homogeniški ir laiko eilučių analizės tyrėjai yra kritikuojami dėl per didelio aprašomų algoritmų priderinimo būtent tiems rinkiniams. Apibrėžus kokiais požymiais pasižyminčių laiko eilučių tyrėjams trūksta yra įmanoma pagal juos generuoti naujas laiko eilutes potencialiai užpildysiančias spragas prieinamuose duomenyse [31].

1.11. Kiti laiko eilučių klasifikavimo algoritmai

1.11.1. Bag of Symbolic Fourier Approximation Symbols (BOSS)

Šis algoritmas priklauso žodynu grįstų klasifikatorių pritaikytų laiko eilutėms grupei. Šiuo metodu yra surandami dažniausiai pasikartojančios sekos laiko eilutėse ir pritaikant jų santykinius dažnius atliekamas klasifikavimas. Būtent šis metodas buvo išrinktas kaip geriausias žodynu grįstas metodas iš visų šio tipo metodų [32].

Žodynu grįstas mokymasis dažnai yra pritaikomas signalų apdorojimo, kompiuterinio matymo bei garso apdorojimo užduotyse. Pritaikant tas pačias idėjas metodas naudojamas ir laiko eilučių klasifikavimo srityje daugeliui problemų. Buvo nustatyta, jog žodynu grįsti metodai aptinka visiškai kitus atskiriamuosius bruožus tarp laiko eilučių, nei kiti laiko eilučių klasifikavimo algoritmų tipai ir buvo įrodyta jog BOSS požymių ansamblio prijungimas prie HIVE-COTE požymių ansamblio smarkiai pagerina klasifikavimo tikslumą [32]. BOSS ansamblis sudaromas tinkleliu grįsta paieška įvertinant aibę parametrų ir paliekant tuos klasifikatorius kurių tikslumas yra ne mažesnis nei 92% geriausios kombinacijos, įvertinimui taikant kryžminį patikrinimą.

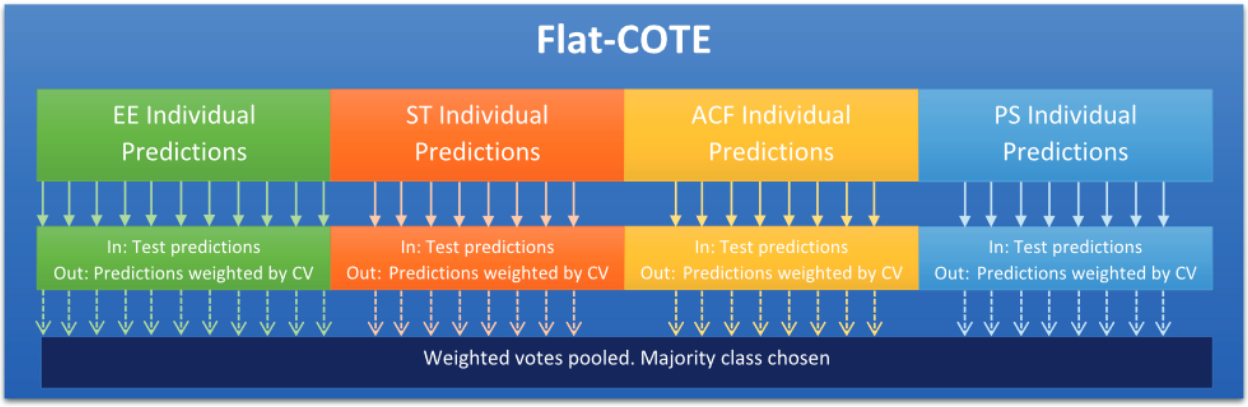
Tačiau BOSS ansamblis turi trūkumų. Kryžminio patikrinimo poreikis kiekvienai parametrų kombinacijai reiškia tai, jog modelio parametrų skaičius yra smarkiai ribojamas kompiuterinių resursų. Poreikis atmintyje išsaugoti griežtai neapibrėžtą kiekį bazinių klasifikatorių sukuria didelius atminties resursų reikalavimus. Galiausia sudaromos labai didelės histogramos ir poreikis jas saugoti kiekvienam baziniam klasifikatoriui dar labiau padidina reikalavimus kompiuterinei atminčiai.

BOSS klasifikatorius veikia slenkančio lango principu. Užduodamas parametras p, kuris reiškia normalizuoti ar ne lange esančias vertes. Tuomet kiekviename žingsnyje pritaikoma diskreti Furjė transformacija, ignoruojant pirmąjį koeficientą jeigu $p = True$, paliekami yra tik pusė surastų koeficientų. Gautas koeficientų sąrašas yra padalinami į α galimų verčių grupių pritaikant daugelio koeficientų grupavimo (Multiple Coefficient Binning, MCB) algoritmą. Jeigu tarp greta esančių langų aptinkamas tas pats žodis, tai tas žodis yra laikomas kaip matytas tik vieną kartą. Klasifikuojant naudojama speciali algoritmo atstumo metrika, kuri pritaikyta kartu su artimiausių kaimynų klasifikatoriumi naudojama sudaryti galutiniam modeliui. Bazinis klasifikatorius turi keturis parametrus kurie visi tinklelio principu išbandomi. Lango ilgis $w \in \{10..m\}$, žodžių ilgis$l \in \{16, 14, 12, 10, 8\}$, ar naudojamas normalizavimas $p \in \{true, false\}$ ir žodyno dydis $\alpha \in \{4\}$, m yra laiko eilutės ilgis.

1.11.2. Collective of Transformation-Based Ensembles (COTE)

Šis modelis pritaikė dvi idėjas kurios leido sukurti tikslesnį laiko eilučių klasifikavimo modelį. Pirmoji yra, kad paprasčiausias būdas kaip laiko eilučių klasifikavimo domene pasiekti geresnių rezultatų yra laiko eilučių transformavimas į alternatyvią erdvę, kur požymiai leidžiantys laiko eilutes atskirti į dvi ar daugiau grupių tampa lengviau pastebimi. Antroji idėja, yra kad tikslumas gali būti pagerinamas apjungiant daugelį prognostinių modelių į ansamblį.

Kaip metodo straipsnyje rašo autoriai [33], modelis sukuria ansamblį iš 35 klasifikatorių su skirtingais svoriniais koeficientais. Kiekvienai laiko eilutei klasifikatoriai sudaromi iš 4 skirtingų erdvių. Tai laiko erdvė, formelių erdvės, autokoreliacinė erdvė ir galios spektrinio tankio erdvė. Kiekvieno klasifikatoriaus svoris yra suskaičiuojamas pritaikant kryžminį validavimą apmokymo duomenims.



12 pav. COTE klasifikatoriaus grafinis vizualizavimas.[34]

1.11.3. Random Interval Spectral Ensemble (RISE)

Random Interval Spectral Ensemble (RISE) yra medžiais grįstas klasifikavimo modelis. RISE modelis laiko eilutes įvertina pasirinkdamas atsitiktinius laiko eilutės intervalus, kiekvienam iš jų suskaičiuojant baigtinį rinkinį Furjė, autokoreliacinių bei dalinai autokoreliacinių požymių. Pagrindinė modelio kritika yra jo veikimo sparta įvertinta $O(nm^2)$, kur m yra laiko eilutės ilgis, o n yra laiko eilučių skaičius apmokymo imtyje. Dėl šios priežasties šis metodas nėra tinkamas taikyti srityse kur dažnos yra ilgos laiko eilutės, pvz. garso duomenų klasifikavimui, kur taikomi metodai transformuojantys laiko eilutes į spektrinius duomenis [35].

Algorithm 1. BuildRISE(Training data *train*, number of classifiers r, minimum interval length *minLen*)

Let $\mathbf{F} \leftarrow < F_1 \dots F_r >$ be the trees in the forest.
Let m be the length of series in *train*
$wholeSeriesFeatures \leftarrow$ getSpectralFeatures(*train*)
buildRandomTreeClassifier($\mathbf{F}_1$,*wholeSeriesFeatures*)
for $i \leftarrow 2$ to r **do**
    $startPos \leftarrow$ randBetween$(1, m - minLen)$
    $endPos \leftarrow$ randBetween$(startPos + minLen, m)$
    train $\leftarrow$ removeAttributesOutsideOfRange(*train*,*startPos*,*endPos*)
    intervalFeatures $\leftarrow$ getSpectralFeatures(*train*)
    buildRandomTreeClassifier($\mathbf{F}_i$,*intervalFeatures*)

13 pav. RISE modelio sudarymo algoritmas [35]

1.12. Apibendrinimas

Laiko eilučių analizė atliekama laiko eilučių požymiams yra plačiai taikomas analizės metodas, kuriuo dažnai yra pasiekiami geri rezultatai, tačiau remiantis vien rankiniu būdų sudarytais požymiais susiduriama su problemomis kylančiomis dėl didelio egzistuojančio sugalvotų požymių skaičiaus. Pavyzdžiui hctsa požymių rinkinį sudaro virš 7 tūkstančių individualiai skaičiuojamų požymių, tad norint taikyti juos visus kompiuterinių resursų prasme tai būtų labai brangu. Dėl šios priežasties atliktas rankinis požymių atrinkimas, sudarytas požymių poaibis catch22, parenkantis vos 22 požymius iš viso hctsa požymių rinkinio.

Norint išvengti dirbtinio prisirišimo prie baigtinio požymių rinkinio, požymių rinkinys gali būti sudaromas autoenkoderio pagalba. Tai yra metodas kuriuo optimizuojamas pradinės laiko eilutės atkūrimas pirmiausia ją suspaudžiant iki nedidelio skaičiaus skaitinių reikšmių „butelio kakliuke“. Optimizuojant šiuo metodu „butelio kakliuke“ gaunamo vektoriaus reikšmes galima laikyti kaip tankiausią informacijos apie laiko eilutę išraišką. Gautą išraišką galima taikyti įvairiems su laiko eilutėmis susijusiais detekcijos, klasifikavimo, prognozavimo uždaviniams įvairiuose domenuose, nuo medicininio, iki pramoninio, iki finansinio kaip aprašyta 2 lentelėje.

Tyrimo tikslas duomenimis grįsta požymių reprezentacija laiko eilutėms, tinkanti detekcijos ir kitiems uždaviniams.

Darbo uždaviniai:

1. Sudaryti mokslinės literatūros apžvalgą analizuojančią požymių svarbą laiko eilučių kontekste.
2. Pasigilinti į tinkamas šiam uždaviniui autoenkoderių architektūras.
3. Sudaryti autoenkoderiu grįstą modelį apmokymui naudojant didelį skaičių laiko eilučių.
4. Gautą modelį pritaikyti laiko eilučių detekcijos uždaviniui spręsti, įvertinti jo rezultatus.
5. Palyginti detekcijos rezultatus taikant catch22 požymių rinkinį.
6. Darbe sudaryto autoenkoderio pritaikymas požymių rinkinio patobulinimui.

2. Metodologija

2.1. Duomenų rinkiniai ir programinė įranga

Modeliui sudaryti ir išbandyti taikytas duomenų rinkinys atrinktas Bagnall‘o [36] su tikslu palyginti naujausius algoritmus taikomus laiko eilučių klasifikavimui. Laiko eilutės atrinktos iš UCR duomenų rinkinio. Kartu su duomenimis yra apibrėžtos tikslios apmokymo ir testavimo imtys, dėl ko modelių palyginimas tampa lengvesnis. Tačiau toks duomenų pateikimas turi reikšmingą problemą, jog dėl atsitiktinių procesų tampa įmanoma modeliams būti permokytiems ir gerai atitikti tik būtent šią mokymo ir testavimo imtį, todėl autoriaus yra siūlomą praplėsti testavimą kryžminio patikrinimo metodu, kartojant apmokymą testavimą kelis kartus su kaskart atsitiktinai parenkama testavimo ir apmokymo imtimi.

Darbui pasirinktas apribojimas rinktis tik tuos duomenų rinkinius, kurie atitinka detekcijos uždavinio apibrėžimą, reiškiantį jog užduotis yra laiko eilučių klasifikavimą į tik dvi galimas klases. Taikant šį apribojimą parinkti 42 duomenų rinkiniai, aprašyti 3 lentelė, kurie kartu apima 45807 laiko eilutes.

Laiko eilutės nėra apribotos vienu domenu, bet apima daugelį įvairių sričių įskaitant ir medicininius, ekonominius, inžinierinius duomenis. Kas leidžia manyti, kad sudarytas modelis bus lankstesnis naujiems panaudojimo atvejams.

Duomenyse egzistuoja nemažai trūkstamų reikšmių, su kuriomis buvo tvarkomasi tiesinio interpoliavimo metodu, t.y. trūkstamos reikšmės užpildamos tarpinėmis reikšmėmis tarp dviejų žinomų taškų.

Kadangi ne visi metodai yra tinkami skirtingo ilgio laiko eilutėms, tai prieš pradedant modeliavimą laiko eilučių ilgius reikėjo suvienodinti. Tas buvo atliekama suvienodinant visų laiko eilučių ilgį iki ilgiausios rinkinyje laiko eilutės ilgio, trūkstamas reikšmes pakeičiant *0*. Vėliau duomenys buvo standartizuojami, apskaičiuojant keliais standartiniais nuokrypiais kiekvienas duomenų taškas yra nutolęs nuo laiko eilutės vidurkio, pagal formulę:

$$z = \frac{x - u}{s}$$

Kur x yra duomenų taško realioji vertė, u laiko eilutės vidutinė reikšmė, s laiko eilutės duomenų taškų standartinis nuokrypis.

Tolesniems giliojo mokymo modeliams sudaryti buvo taikomas *pyTorch* paketas Python programavimo kalbai. Duomenys buvo standartizuojami *sklearn* paketo pagalba, ir naudojami *pandas* pakete aprašytomis struktūromis. Suformuotos latentinės erdvės vizualizacijoms naudojamas *matplotlib* ir *seaborn* paketai. Principinių komponenčių ir t-SNE metodai dimensionalumo mažinimo metodai atliekamas jau minėto *sklearn* paketo pagalba, UMAP dimensionalumo mažinimui pasitelkiamas paketas sudarytas pačių metodo autorių *UMAP*.

3 lentelė Modeliui sudaryti taikytos laiko eilutės

Duomenų rinkinys	**Laiko eilučių skaičius**	**Trumpas apibūdinimas**
BeetleFly	40	Vabalo nuo musės atskyrimas pagal kontūrą transformuotą į viendimensę išraišką
BirdChicken	40	Vištos nuo kitų paukščių atskyrimas pagal kontūrą transformuotą į viendimensę išraišką.
Chinatown	363	Savaitgalio ir darbo dienos pėsčiųjų srauto atskyrimas.
Coffee	56	Robusta ir Arabica kavos atskyrimas pagal spektrografinius duomenis.
Computers	500	Skirtingų namų ūkių energijos sunaudojimo duomenys, atskiriant ar naudojamas stacionarus ar nešiojamas kompiuteris.
DistalPhalanxOutlineCorrect	876	Asmens amžiaus grupės nustatymas pagal galinio pirštakaulio formą.
DodgerLoopGame	288	Eismo intensyvumo matavimas netoli sporto stadiono, siekiant atskirti dienas kuomet vyksta varžybos.
DodgerLoopWeekend	288	Eismo intensyvumo matavimas netoli sporto stadiono, siekiant atskirti savaitgalius nuo darbo dienų.
ECG200	200	Elektrokardiograma pavienių širdies dūžių siekiant atskirti miokardo infarkto paveiktą širdies ritmą nuo normalaus.
ECGFiveDays	884	Skirtingomis dienomis pamatuotų elektrokardiogramų duomenų atskyrimas (skirtumas 5 dienos).
Earthquakes	461	Žemės drebėjimų duomenys Richterio skalėje bandant atpažinti atvejus po kurių pakartotiniai žemės drebėjimai nėra stebimi

FordA	4941	Automobilio subsistemos nominalaus ir gedimo paveikto darbo atpažinimas pagal garsą.
FordB	4446	Automobilio subsistemos nominalaus ir gedimo paveikto darbo atpažinimas pagal garsą triukšmingesnėje aplinkoje.
FreezerRegularTrain	3000	Šaldiklio virtuvėje ir garaže atskyrimas pagal elektros energijos suvartojimo profilį.
FreezerSmallTrain	2878	Šaldiklio virtuvėje ir garaže atskyrimas pagal elektros energijos suvartojimo profilį.
GunPoint	200	Ginklo išsitraukimo iš dėklo judesio kontūro atskyrimas pagal tai ar yra ištraukiamas tikras ginklas, ar tik atliekamas judesys imituojant.
GunPointAgeSpan	451	Ginklo išsitraukimo iš dėklo judesio kontūro atskyrimas pagal tai ar yra ištraukiamas tikras ginklas, ar tik atliekamas judesys imituojant.
GunPointOldVersusYoung	451	Ginklo išsitraukimo iš dėklo judesio kontūro atskyrimas pagal aktoriaus amžiaus grupę.
Ham	451	Ispaniško ir Prancūziško sauso vytinimo kumpio atskyrimas pagal spektrogramą.
HandOutlines	1370	Rankos kaulų kontūrų duomenys, siekiant atskirti laiko eilutes į grupes pagal amžiaus kategoriją.
Herring	128	Šiaurės jūroje ir Temzės upėje pagautų silkių atskyrimas pagal otolitų kontūrus.
HouseTwenty	159	Bendro namų ūkio elektros profilio atskyrimas, nuo skalbimo ir džiovinimo mašinų kontūro suvartojamos elektros energijos.

ItalyPowerDemand	1096	Italijos elektros suvartojimo atskyrimas pagal metų periodą (spalis-kovas, balandis-rugsėjis)
Lightning2	121	Elektros išlydžių spektrogramų atskyrimas į dvi grupes.
MiddlePhalanxOutlineCorrect	891	Asmens amžiaus grupės nustatymas pagal vidurinio pirštakaulio formą.
MoteStrain	1272	Drėgmės ir temperatūros sensorių atskyrimas pagal jų parodymus.
PhalangesOutlinesCorrect	2658	Asmens amžiaus grupės nustatymas pagal pirštakaulių formą.
PowerCons	360	Namų ūkių elektros energijos suvartojimo profiliai atskiriant šiltąjį ir šaltąjį sezonus.
ProximalPhalanxOutlineCorrect	891	Asmens amžiaus grupės nustatymas pagal pamatinio pirštakaulio formą.
SemgHandGenderCh2	900	Rankų elektromiografiniai duomenys atliekant suspaudimo judesį, atskiriant vyrus ir moteris.
ShapeletSim	200	Triukšmė paslėptų fomų atskyrimas į dvi grupes.
SonyAIBORobotSurface1	621	Paviršiaus kuriuo eina robotas (cemento ir kilimo) atskyrimas pagal X ašies akselerometro duomenis.
SonyAIBORobotSurface2	980	Paviršiaus kuriuo eina robotas (cemento ir kilimo/pievos) atskyrimas pagal X ašies akselerometro duomenis.
Strawberry	983	Braškių atskyrimas nuo kitų vaisių ir uogų pagal spektrografas.
ToeSegmentation1	268	Normalaus ir apsunkinto (simuliuojant traumas, šlubavimą) ėjimo atskyrimas pagal X ašies matavimus.
ToeSegmentation2	166	Normalaus ir apsunkinto (simuliuojant traumas, šlubavimą)

		ėjimo atskyrimas pagal y ašies matavimus.
TwoLeadECG	1162	Dviejų skirtingų profilių elektrokardiogramų atskyrimas.
Wafer	7164	Lustų atskyrimas į brokuotus ir gerus pagal vieno iš kelių sensorių duomenis.
Wine	111	Vyno klasifikavimas pagal spektrografinius duomenis.
WormsTwoClass	258	Caenorhabditis elegans kirminų klasifikavimas į laukinius ir mutavusius pagal jų judesio profilį.
Yoga	3300	Aktorių lyties nustatymas pagal jų kontūrą keičiant Jogos pozas.

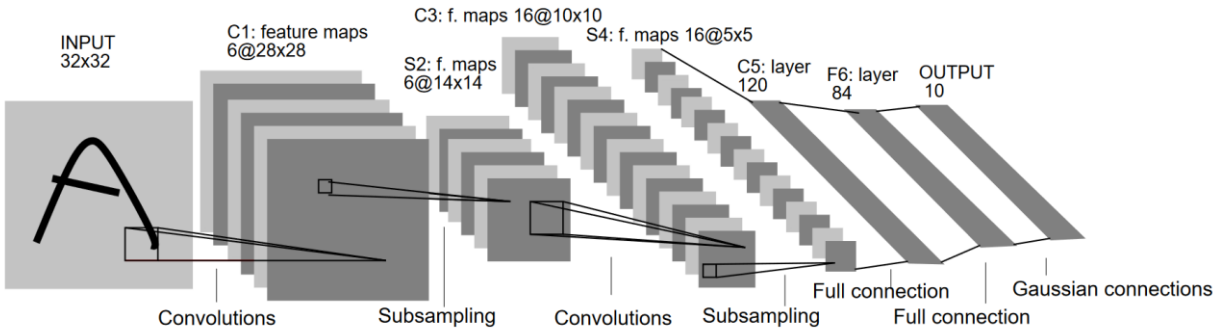
2.2. Neuroniniai tinklai

2.2.1 ir 2.2.2 aprašomi darbe naudotos neuroninių tinklų tipai ir pagrindiniai jų parametrai kurie buvo darbo metu parenkami. Skyrelyje 2.2.3 aprašomas mokymosi greičio parametras, koks darbe jis buvo taikomas bei jam atlikta modifikacija nuo įprastai naudojamų.

2.2.1. Konvoliuciniai neuroniniai tinklai

Konvoliuciniai neuroniniai tinklai yra vienas dažniausiai pritaikomų neuroninių tinklų tipas. Konvoliucinių neuroninių tinklų pavadinimas siejamas su matematinės operacijos – sąsukos (angl. convolution) – pavadinimu. Konvoliucinius neuroninius tinklus sudaro keli skirtingo tipo sluoksniai: konvoliuciniai sluoksniai, netiesiniai sluoksniai, apjungimo (angl. pooling) sluoksniai bei pilnai sujungti sluoksniai. Apmokomi parametrai yra tik konvoliuciniuose ir pilnai sujungtuose sluoksniuose, apjungimo bei netiesiniai sluoksniai jokių apmokomų parametrų neturi. Šių tinklų stiprioji pusė yra dėsningumų atpažinime, todėl šio tipo tinklai yra plačiausiai taikomi vaizdinio tipo duomenims tirti, pvz. kompiuterinio matymo uždaviniams spręsti. Taip pat turi platų pritaikymą natūralios kalbos apdorojime [37].

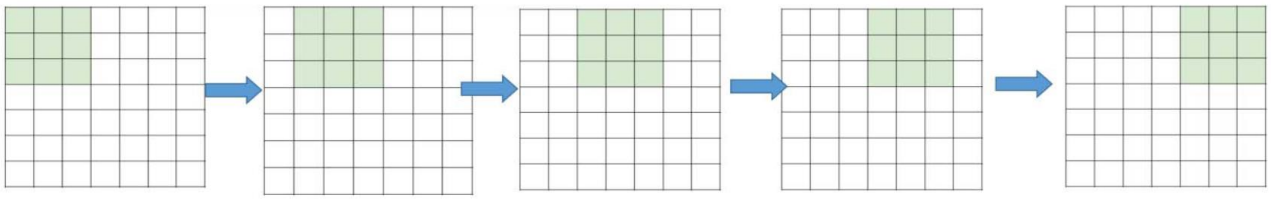
Konvoliuciniai sluoksniai apjungia tris modelių architektūrines idėjas: lokalaus jautrumo laukai, nekintantys svoriniai koeficientai, bei imties ėmimas daliai erdvės. Tipinis pavyzdinis konvoliucinis neuroninis tinklas formų atpažinimui yra LeNet-5, matomas paveiksliuke žemiau.



14 pav. LeNet-5 architektūra taikoma simbolių atpažinimui [38]

Kiekvienas konvoliucinio sluoksnio elementas apdoroja informaciją tik apie dalį kaimynystėje tarpusavyje esančių duomenų taškų prieš tai buvusiame sluoksnyje. Kadangi modelis jautrus tik lokaliems požymiams, tai išgaunami tik elementarūs požymiai, tokie kaip kraštinės, kampai, pabaigos. Šie požymiai tolesniuose sluoksniuose yra apjungiami tam, kad išgauti aukštesnio lygmens požymius. Kadangi konvoliucijos ir kartu elementariųjų požymių išgavimas vyksta slenkančio lango principu, tai tas pats požymių detektorius aptinkantis požymį vienoje erdvės vietoje, tą taip pat gerai darys ir kitose erdvės vietose. Aptikus požymį jo tiksli vieta nėra svarbi, svarbus tik jo santykinė pozicija su kitais požymiais. Kaip pavyzdį galima paminėti ranka rašytų skaičių identifikavimą, jeigu žinome kad viršuje kairėje yra apytiksliai horizontalios linijos pabaiga, o apačioje dešinėje apytiksliai vertikalios linijos pabaiga, tuomet galime manyti kad tai yra skaičius 7. Tiksli šių požymių vieta ne tik kad nėra svarbi, bet jos išsaugojimas yra net galimai žalingas, nes tiksli šių požymių pozicija bus kintanti tarp skirtingų rašytinių skaičių pavyzdžių.

Kadangi slenkantis langas yra esminė šių modelių veikimo dalis, tai svarbu skirti dėmesio į lango žingsnio (angl. stride) dydžio parametrą. Didinant žingsnį mažinamas gautų požymių persidengimas, kadangi langas juos greičiau „peršoka“, ir kiekvienas individualus taškas dalyvauja mažesniame skaičiuje konvoliucijų. Tačiau reikia būti atsargiems, kadangi žingsnį padidinus daugiau nei pats lango dydis dalį požymių įmanoma iš viso praleisti [37].



15 pav. Vienetinio žingsnio slenkančio lango vizualizacija. (Albawi, Mohammed, & Al-Zawi, 2017)

Kadangi slenkantis langas visada yra didesnis nei 1 elementas, bei žingsnis nebūtinai yra vienetinis problema tampa kraštinės reikšmės. Kadangi požymiai aptinkami slenkančio lango filtrui slenkant paviršiumi, kraštuose esantys požymiai reikšmės neturi šanso būti pastebėti. Taip pat dėl žingsnio dydžio galimas variantas, jog filtras negalės atlikti paskutinio žingsnio nes atlikus papildomą žingsnį bus žengta už matricos ribų. Šiai problemai spręsti yra žinomas vienas labai paprastas, bet efektyvus metodas – nulinio pamušalo taikymas. Tai paprastas metodas, kuriuo matricos šonuose surašomos nulinės reikšmės.

0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

16 pav. nulinis pamušalas.

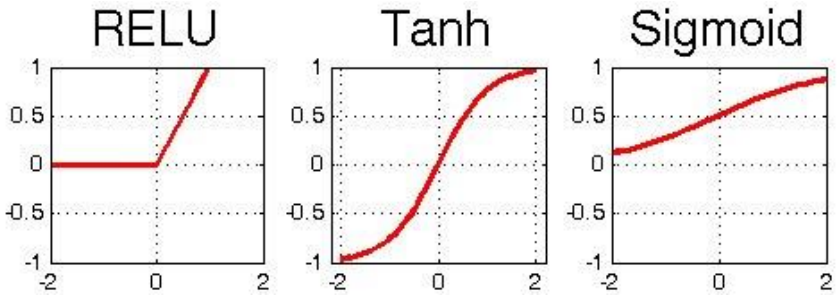
Suskaičiavus konvoliucinių sluoksnių reikšmes kitas etapas yra netiesiškumo įvedimas. Netiesiškumas gautus rezultatus pakeičia įvairiomis prasmėmis, geba sumenkinti vienas reikšmes, arba išryškinti kitas. Dažniausiai taikomos netiesinės funkcijos yra:

- Hiperbolinio tangento funkcija (1 lygtis)
- Sigmoidės funkcija (2 lygtis)
- ReLU funkcija (3 lygtis)

$$\tanh\left(\frac{x}{2}\right) = \frac{\sinh x}{\cosh x + 1} \tag{1}$$

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{2}$$

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0. & x \leq 0 \end{cases} \tag{3}$$



17 pav. netiesinių funkcijų vizualizavimas

Kaip rašo Albawi‘s [37] iki ReLU funkcijos pastebėjimo ir įvertinimo dažniausiai buvo taikomos hiperbolinio tangento ir sigmoidės funkcijos, tačiau dabar dažniausiai galima pamatyti taikomą ReLU funkciją, dėl kelių jos savybių:

1. Paprastesnis apibrėžimas tiek funkcijoje, tiek gradiente (4 lygtis)
2. Dėl to jog sigmoidė bei tanh yra funkcijos su įsodrinimu giliuose tinkluose signalas pradeda nykti, kas srityje plačiai žinoma kaip nykstančio gradiento problema. Taip yra todėl, jog minėtų funkcijų gradientas yra labai artimas nuliui visur, išskyrus patį funkcijos centrą. ReLU šios problemos neturi, kadangi jos gradientas yra pastovus visoms teigiamoms reikšmėms.

3. ReLU funkcija gradiente geba išduoti tikrą nulinę reikšmę, lyginant su sigmoide ir hiperboliniu tangentu kurių gradientų reikšmės visoje funkcijos reikšminėje srityje yra ne nulinės. Tai galimai padeda tinklo mokymuisi.

$$\frac{d}{dx}ReLU(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \quad (4)$$

Apjungimo operacija atliekama kiekvienam konvoliuciniam sluoksniui individualiai. Pagrindinė šios operacijos mintis yra rezultatų rezoliucijos mažinimas siekiant dar labiau sumažinti duomenų kompleksiškumą, analogiškai rezoliucijos mažinimui vaizdų apdorojimo srityje. Apjungimo operacija vykdoma kaip ir konvoliucijos operacijos slenkančio lango principu, tačiau pati operacija yra gerokai paprastesnė, dažniausiai naudojama apjungimo operacija yra maksimalios reikšmės lange radimas, kad iš viso lango reikšmių paliekama tik viena – pati didžiausia, dar galima sutikti vidurkio apjungimo operacija kurią taikant gražinama vidutinė lango reikšmė.

Pilnai sujungtų sluoksnių architektūra yra panaši tradicinių neuroninių tinklų architektūrai, kur kiekvienas mazgas yra sujungtas su visais prieš tai buvusio sluoksnio ir su visais kito sluoksnio elementais. Pagrindinė problema kurią šis sluoksnis sukelia yra labai didelis optimizuojamų parametrų skaičiaus išauginimas, kadangi kiekvienas mazgas jungiasi su kiekvienu prieš tai sluoksnio išvestyje esančiu tašku kur kiekvienam sujungimui reikalingas suskaičiuoti svorinis koeficientas, bei analogiški koeficientai pilnai sujungto modelio išvestyje.

Konvoliucinio neuroninio tinklo architektūra kuri buvo naudojama darbe aprašyta lentelėje žemiau. Tinklą galima suskaidyti į tris dalis: enkoderį, latentinę erdvę ir dekoderį, enkoderį sudaro du konvoliuciniai sluoksniai po jų sekančiomis netiesinėmis ReLU funkcijomis, latentinėje erdvėje informacija yra suspaudžiama iki 16 realių skaičių reikšmių, iš kurių dekoderyje duomenys yra kiek įmanoma geriau atkuriami, optimizuojant mažiausią absoliutinį kvadratinį nuokrypį nuo pradinės laiko eilutės.

4 lentelė Darbe naudojamo konvoliucinio neuroninio tinklo architektūra

ID		Sluoksniai	Parametrai		
1	Enkoderis	Conv1d	Kernel_size = 7	Stride = 2	Padding = 3
2		ReLU			
3		Conv1d	Kernel_size = 7	Stride = 2	Padding = 3
4		ReLU			
5		Conv1d	Kernel_size = 7	Stride = 2	Padding = 3
6		ReLU			
7	Latentinė erdvė	Pilnai sujungtas	Features = 16		
8	Dekoderis	ConvTranspose1d	Kernel_size = 7	Stride = 2	Padding = 3
9		ReLU			
10		ConvTranspose1d	Kernel_size = 7	Stride = 2	Padding = 3
11		ReLU			
12		ConvTranspose1d	Kernel_size = 7	Stride = 2	Padding = 3
13		ReLU			
14		ConvTranspose1d	Kernel_size = 7	Stride = 1	Padding = 2
15		Pilnai sujungtas	Out_features = laiko_eilutės_ilgis		

2.2.2. Mokymosi greitis

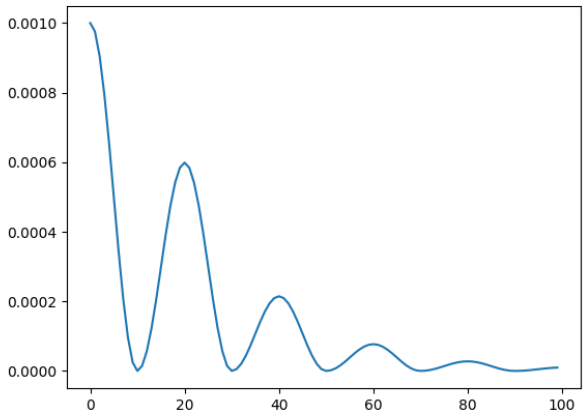
Daugelio mašininio mokymosi metodų tikslas yra tikslo funkcijos $f(x)$ optimizavimas keičiant parametrus x. Tai dažniausiai atliekama iteraciniu procesu, kurio metu kiekvienoje iteracijoje yra esi parametrai yra modifikuojami per Δx. Parametrų atnaujinimą po kiekvienos iteracijos t labai paprastai galima aprašyti formule:

$$x_{t+1} = x_t + \Delta x_t \tag{5}$$

Kad būtų nustatytas teisingas mokymosi greitis (absoliutinė Δx reikšmę) dažnai taikomas derinimo metodas, kurio tikslas ranka parinkti didžiausią mokymosi greitį ties kuriuo tikslo funkcija vis dar konverguoja. Parinkus per didelį mokymosi greitį modelis gali niekada nekonverguoti, tad dažnai

tinkamo mokymosi greičio ieškojimas tampa mažai mokslu pagrįstas, o spėliojimu ir ankstesne patirtimi paremtas.

Gradientui tikslo funkcijos paviršiuje artėjant prie minimumo parametrų vertės gali pradėti svyruoti apie minimumą, taip nustodamos link jo artėti. Taikytini yra metodai kurie keičia mokymosi greitį modelio apmokymo proceso eigoje, tai atliekama arba padidinant greitį tada kada galima, ar siekiant sulėtinti mokymosi greitį artėjant prie lokalaus minimumo. Darbe yra taikomas sulėtinimo metodas modifikuojant kosinuso pjūklinį tvarkaraštį prie jo pridedant slopinimo koeficientą (18 pav.). Mokymosi greitis yra atnaujinimas kiekvienos mokymosi epochos pabaigoje.



18 pav. Mokymosi greičio kitimas pagal modifikuota kosinuso pjūklinį tvarkaraštį

2.3. Daugiadimensės erdvės vizualizavimas

2.3.1. Principinių komponenčių metodas

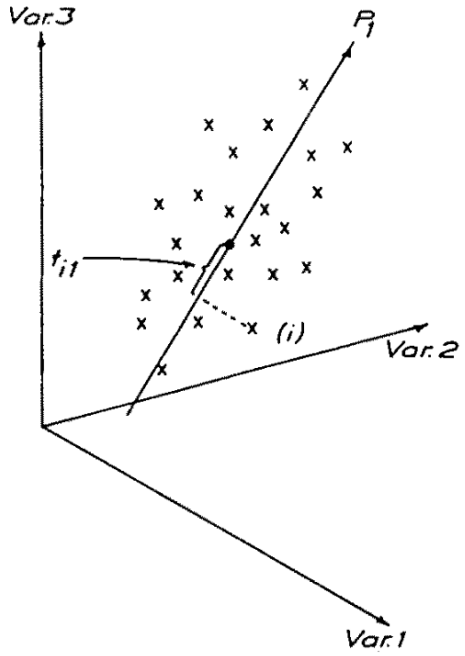
Principinių komponenčių analizė (PCA) yra ko gero dažniausiai taikomas dimensionalumo mažinimo metodas dėl savo paprastumo ir lengvos interpretacijos. PCA duomenų matricoje suranda vyraujančius dėsningumus, arba paprastai tariant surandama ašis (arba ašys) kuriomis galima paaiškinti didžiausia kiekį duomenų variacijos.

PCA gali būti taikoma bet kokiai duomenų matricai, jeigu ši yra teisingai transformuota bei yra suvienodintos pirminės duomenų matricos ašių skalės [39]. Tai yra rekomenduojamas kaip pirmasis žingsnis atliekant bet kokią daugiamatę analizę, tam kad greitai ir paprastai būtų pastebėta duomenų struktūra, esančios išskirtys.

Principinių komponenčių metodo viena stipriųjų pusių, kad komponentes galima suskaičiuoti analitiškai, jokio tipo optimizavimas nėra reikalingas, gautas rezultatas visada yra optimalus mažiausių kvadratų nuokrypio kriterijui. Analitinis principinių komponenčių ieškojimo algoritmas:

1. Suskaičiuojamas duomenų vidurkis kiekvienai duomenų ašiai
2. Suskaičiuojama kovariacinė matrica visam duomenų rinkiniui pagal ašis, pavyzdžiui tridimensiu atveju su ašimis x, y, z $\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$
3. Surandami tikriniai vektoriai ir jų tikrinės reikšmės

4. Tikriniai vektoriai išrūšiuojami pagal tikrines reikšmės mažėjimo tvarka ir parenkami k pirmų tikrinių vektorių
5. Duomenys projektuojami į k-dimensę erdvę.

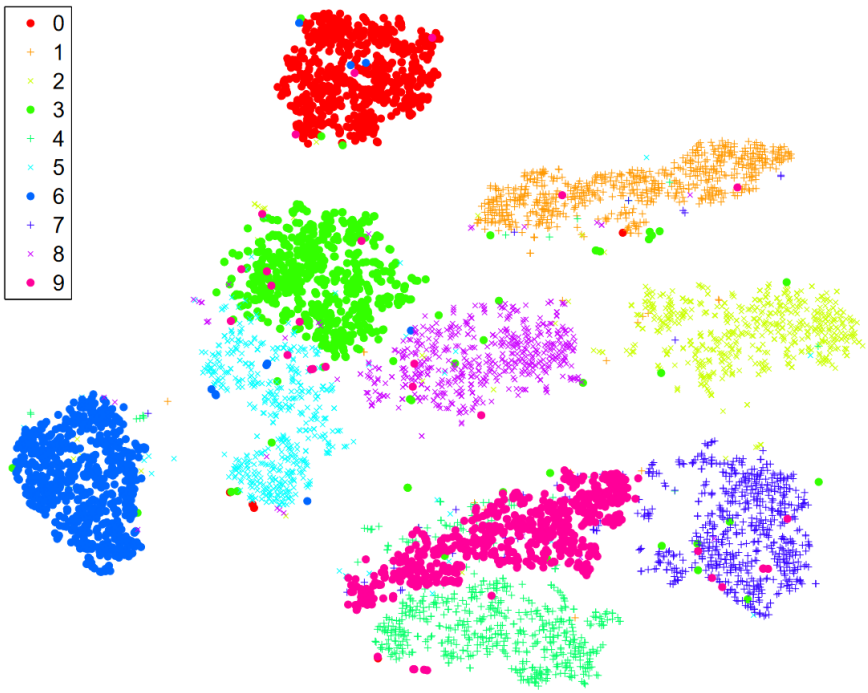


19 pav. Matricos X duomenų išsidėstymas tridimensėje erdvėje su tiesia linija kertančia duomenis taip, jog ji geriausia aproksimuotų visus duomenų taškus. Tai atitinka vienos komponentės principinių komponenčių analizės modelį [39].

2.3.2. t-SNE

t-SNE kaip ir visų dimensionalumo mažinimo metodų pagrindinis tikslas yra daugiadimensės erdvės suspaudimas į mažesnio dimensionalumo erdvę išsaugant kiek įmanoma daugiau informacijos. Skirtingai nuo principinių komponenčių metodo, šiuo metodu informacija kurią siekiama išsaugoti yra duomenų taškų klasteriai, o ne tiksli duomenų taškų vieta. Pirmą kartą šį metodą aprašė van der Maaten‘as [40] ir kaip teigiama autoriaus, siūlomas metodas daug tiksliau atskyrė duomenis į atitinkamus klasterius negu visi kiti išbandyti metodai.

Pagrindinė problema kuri buvo išspręsta taikant šį metodą yra ankstesnių metodų polinkis visus duomenis sugrupuoti į vieną grupę, niekaip prasmingai jų neatskiriant. Kodėl daugiadimensės erdvės atstumo matas skaičiuojant poromis dažnai neduoda prasmingo rezultato, paaiškinama pavyzdžiu 10 dimensijų erdvės pavyzdžiu. Tokioje erdvėje yra galima padėti 11 taškų kurie visi vienas nuo kito būtų nutolę per vienodą atstumą, kas yra niekaip neįmanoma atvaizduoti 2 dimensijų vaizde, su tuo susijęs ir labai skirtingas atstumų tarp taškų poromis skirstinys nuo to kokio tikėtumėmės dviejų dimensijų atveju [40]. Siūloma daugiadimensės erdvės atstumų metriką paversti į tikimybes naudojantis Gauso skirstinį, o žemo dimensionalumo skirstinyje atstumo metrikai taikyti vieno laisvės laipsnio Stjudento t skirstinį su palyginus Gauso skirstiniui storesnėmis uodegomis. Tas leidžia vidutinį atstumą tarp taškų aukštoje dimensijoje daug tiksliau modeliuoti projekcijoje.



20 pav. t-SNE vizualizacijos pavyzdys pritaikius metodą MNIST ranka rašytų skaitmenų rinkiniui [40]

Šis algoritmas yra iteracinis, nėra įmanoma rasti sprendinio vienu žingsniu. Supaprastintai žingsnius galima aprašyti taip:

1. Suskaičiuojama atstumo matrica tarp visų taškų esančių aukšto dimensionalumo duomenų rinkinyje poromis ir gauta metrika transformuojama pagal Gauso skirstinį, standartinį nuokrypį nustatant pagal taškų tankį erdvėje apie modeliuojamą tašką.

2. Taškai išmėtomi atsitiktinai norimo dimensionalumo erdvėje.

3. Suskaičiuojama atstumo matrica tarp visų taškų esančių mažesnio dimensionalumo erdvėje ir rezultatai transformuojami pagal Stjudento t skirstinį su vienu laisvės laipsniu.

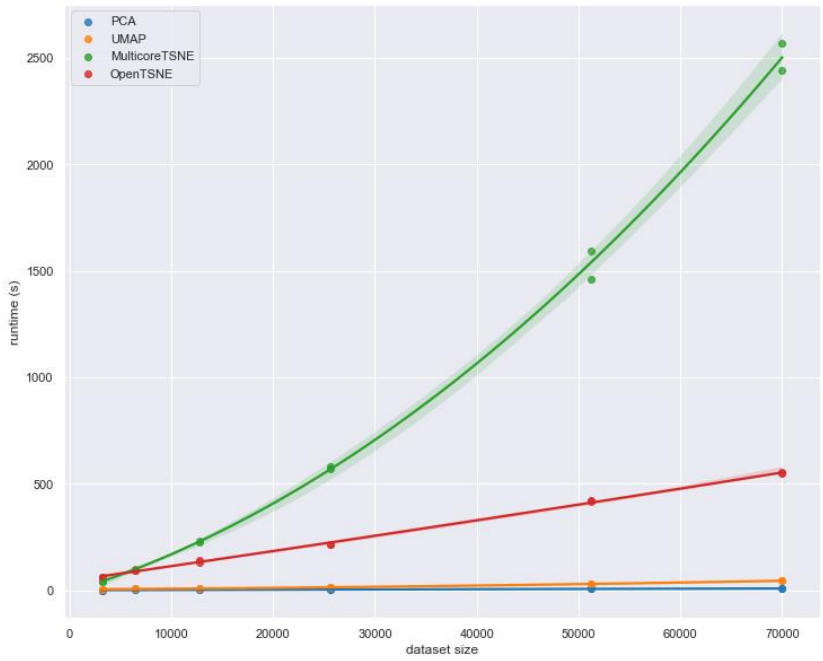
4. Trečiame punkte gauta matrica lyginama su pirmame punkte gauta matrica ir formuojamoje žemesnio dimensionalumo matricoje taškai yra perstumiami taip, jog panašumo matricos tarpusavyje kiek įmanoma labiau suvienodėtų, arba didžiausio gradiento kryptimi.

5. Jeigu iteracijų skaičius nepasiekė apibrėžto skaičiaus, grįžtama į trečią punktą.

2.3.3. UMAP

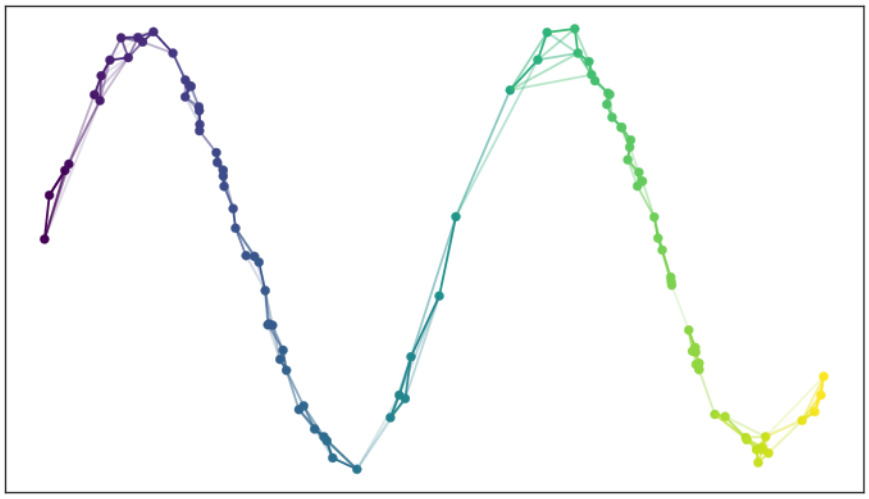
UMAP (Uniform Manifold Approximation and Projection) yra gana naujas netiesinis dimesionalumo mažinimo metodas pirmą kartą aprašytas McInnes‘o [41], pagal galimybes prilyginamas t-SNE metodui. UMAP veikimo principas yra aukštos dimensijos grafo suformavimas ir jo perkėlimas į mažesnę dimensiją išlaikant topologiją nepakitusią.

Metodo didžiausia stiprybė lyginant su t-SNE yra, tai jog šis metodas nėra iteracinis, tad jo skaičiavimų trukmė yra gerokai mažesnė. Autorių palyginime (**21 pav.**) aiškiai galima matyti UMAP pranašumą prieš t-SNE metodą, tačiau dėl kiek didesnio savo paprastumo PCA metodas vis tiek išlieka greičiausias.



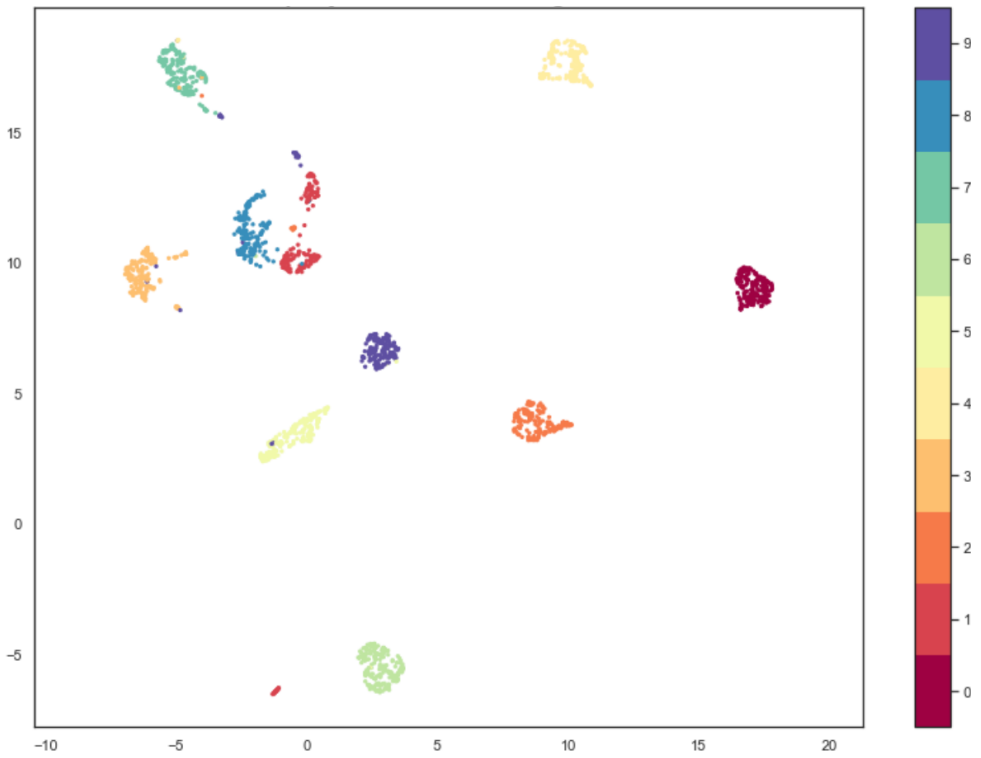
21 pav. UMAP, PCA ir t-SNE metodų greitaveikos palyginamas keičiant duomenų rinkinio dydį. [41]

Grafo formavimo tikslas yra suformuoti grafą kuriame visi duomenų taškai būtų sujungti. Tai yra atliekama apskaičiuojant apie kiekvieną tašką jo lokalumo metriką apsibrėžus k artimiausių kaimynų, pagal Rymano geometrijoje taikomus metodu. Taškai kurie papuola į kito lokalumo erdvę yra sujungiami ir jungties stiprumas nustatomas pagal lokalumo metriką pagal kurią jie buvo sujungti, arba kitaip tariant taškai esantys labiau nutolę vienas nuo kitu bus sujungti silpnesne jungtimi. Gauto grafo pavyzdys yra 22 pav.



22 pav. Suformuotas grafas sujungus taškus pagal jų lokalumo metriką [41]

Gautas grafas yra perkeliamas į žemesnio dimensionalumo erdvę išlaikant topografiją ir taškus išdėliojant taip, kad taškai su stipriomis jungtimis būtų atvaizduojami arti vienas kito, o taškai sujungti silpnomis jungtimis atvaizduojami toliau vienas kito. Verta paminėti, kad gauta žemesnės dimensijos vizualizacija nėra unikali, daugelyje vietų grafo dalis, ar visas grafas, gali būti pasukamas, tad algoritmą kartojant kelis kartus bus gaunama vis kitaip atrodanti vizualizacija.



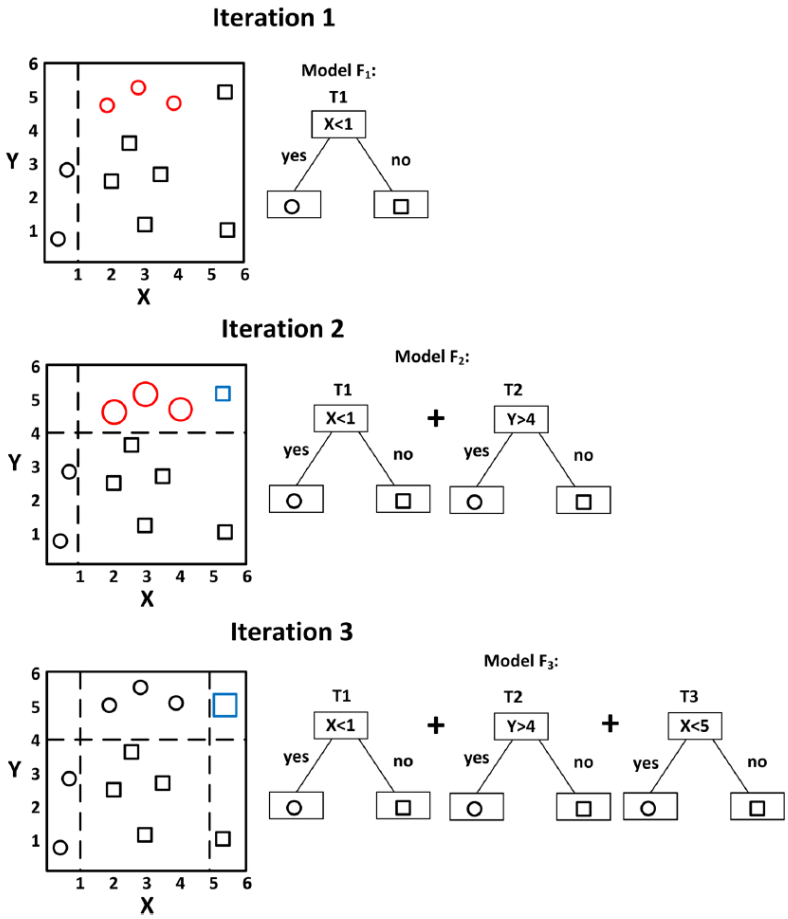
23 pav. UMAP vizualizacija MNIST ranka rašytų skaitmenų rinkiniui [41]

2.4. Gradientinio tobulinimo klasifikavimo (detekcijos) metodas

Gradientinio tobulinimo (angl. gradient boosting) modeliai yra vieni galingiausių prognozavimo algoritmų. Boosting (tobulinimo) modelio dalies pagrindinė užduotis yra silpną mokinį padaryti geresniu. Kearns‘as [42] pavadino tai „hipotezių tobulinimo problema“, ir iš praktinės pusės apibrėžė kaip efektyvų būdą iš blogų hipotezių pereiti prie labai gerų hipotezių. Bloga, arba silpna, hipotezė gali būti suprantama kaip hipotezė kuri yra vos geresnė už atsitiktinį spėjimą.

Pirmasis funkcionalus algoritmas pritaikęs šią idėja buvo AdaBoost (Adaptive Boosting), šis algoritmas pagrįstas grubių ir gana netikslių euristikų apjungimu, tam jog jos kartu, arba ansamblyje, sudarytų vieną bendrą stiprų prognostinį vienetą [43]. AdaBoost visiems prognozuojamiems taškams suteikia svorinį įvertinimą, sunkiai prognozuojamiems atvejams skiriant didesnį svorį. Dėl to formuojant naujas silpnas euristikas yra labiau atsižvelgiama į būtent tuos taškus kuriuos iki šiol modeliui sunkiai sekėsi prognozuoti.

Friedman‘as [44] patobulino AdaBoost algoritmą, perfrazavęs tobulinimą, kaip skaitinio optimizavimo uždavinį, kuriuo siekiama minimizuoti kainos, arba klaidos, funkciją, silpnų mokinių generavimui taikant gradientinio nusileidimo tipo procedūrą. Šio patobulinimo dėka tapo galima naudoti bet kokias diferencijuojamas klaidos funkcijas, todėl modelį tapo įmanoma taikyti ir regresijos, bei kelių klasių klasifikavimo uždaviniams spręsti.



24 pav. gradientinio tobulinimo modelio formavimo vizualizacija [45]

Gradientinio tobulinimo algoritmas vieną po kito formuoja sprendinių medžius.

1. Suformuojamas geriausias sprendimų medis pagal užduotą klaidos funkciją.
2. Įvertinama suformuoto modelio liekana pagal klaidos funkcija.
3. Formuojamas naujas medis, kuris prognozuoja modelio liekanas.
4. Jeigu nėra įvykdyta modelio generavimo stabdymo sąlyga, grįžtama į antrą žingsnį.

Gauto modelio prognozė gaunama sumuojant visų individualių suformuotų medžių prognozes.

$$D(x) = d_{1_{medis}}(x) + d_{2_{medis}}(x) \dots \quad (6)$$

Apmokytas modelis prognozuojamą rezultatą pateikia įverčiu kuris yra realus skaičius tarp 0 ir 1. Papildomai yra naudojamas slenksčio parinkimas virš kurio laiko eilutei prognozuojama klasė bus laikoma „1“, o kitais atvejais laiko eilutė bus laikoma „0“ klasės. Slenkstis yra parenkamas taip jog būtų maksimizuojamas teisingų „1“ klasės klasifikuotų elementų skaičius ir minimizuojamas klaidingai kaip „1“ klasė klasifikuotų elementų skaičius, pagal 7 formulę, kur TPR yra tikrų „1“ reikšmių skaičius prie tam tikros slenkstinės vertės, o FPR yra klaidingų „1“ reikšmių skaičius prie tam tikros slenkstinės vertės.

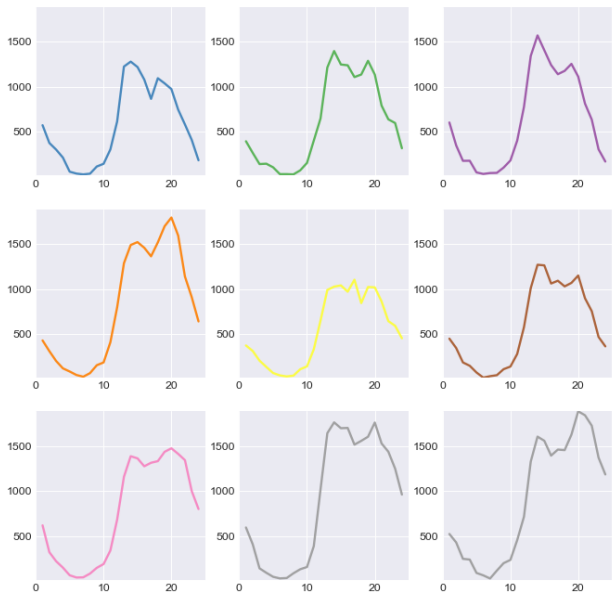
$$\min\{TPR - FPR\} \quad (7)$$

3. Rezultatai

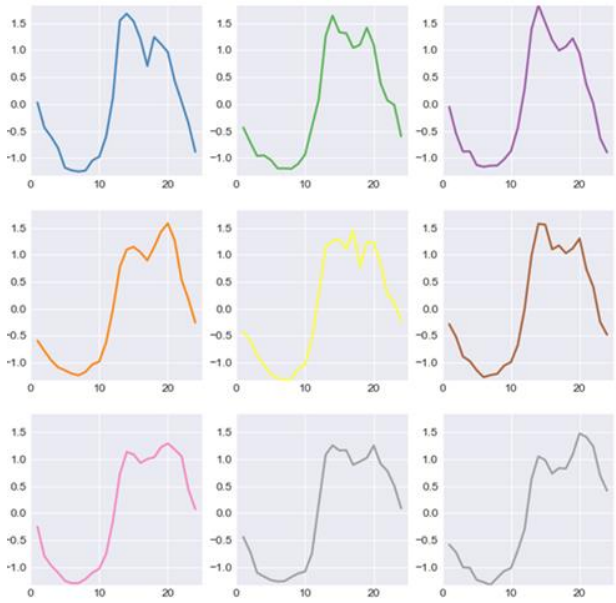
3.1. Autoenkoderio sudarymas

Šioje dalyje aprašyti sudaryti autoenkoderiai, jų tyrimas ir aptarti rezultatai. Autoenkoderis sudaromas naudojantis *pyTorch* biblioteka, o sudaryti požymiai įvertinami taikant dimensionalumo mažinimo metodus. Autoenkoderio sudarymui taikyti visi turimi duomenys, t.y. visos 45 807 laiko eilutės iš 42 duomenų rinkinių. Sudaryti kiek skirtingų architektūrų autoenkoderiai, pavaizduoti ir aptarti skirtumai tarp jų formuojamų latentinių erdvių.

Atliktas duomenų išankstinis apdorojimas siekiant pašalinti trūkstamas reikšmes bei siekiant suvienodinti skirtingų laiko eilučių mastelį. Siekiant išvengti persimokymo, taikomas duomenų padalinimas į apmokymo ir validavimo duomenų rinkinius. Validavimo duomenų rinkinys modelių parametrų optimizavimui nėra naudojamas, šiais duomenimis modelis yra tik išbandomas kiekvienos epochos gale išsaugant modelį kurio klaidos funkcija skaičiuojant validavimo duomenims yra mažiausia. Skaičiavimus atlikus apibrėžtą epochų skaičių kaip geriausias priimamas modelis, kurio klaida validavimo duomenims yra mažiausia ir tolesniems skaičiavimams būtent šis modelis ir naudojamas.



25 pav. Netransformuotos laiko eilutės

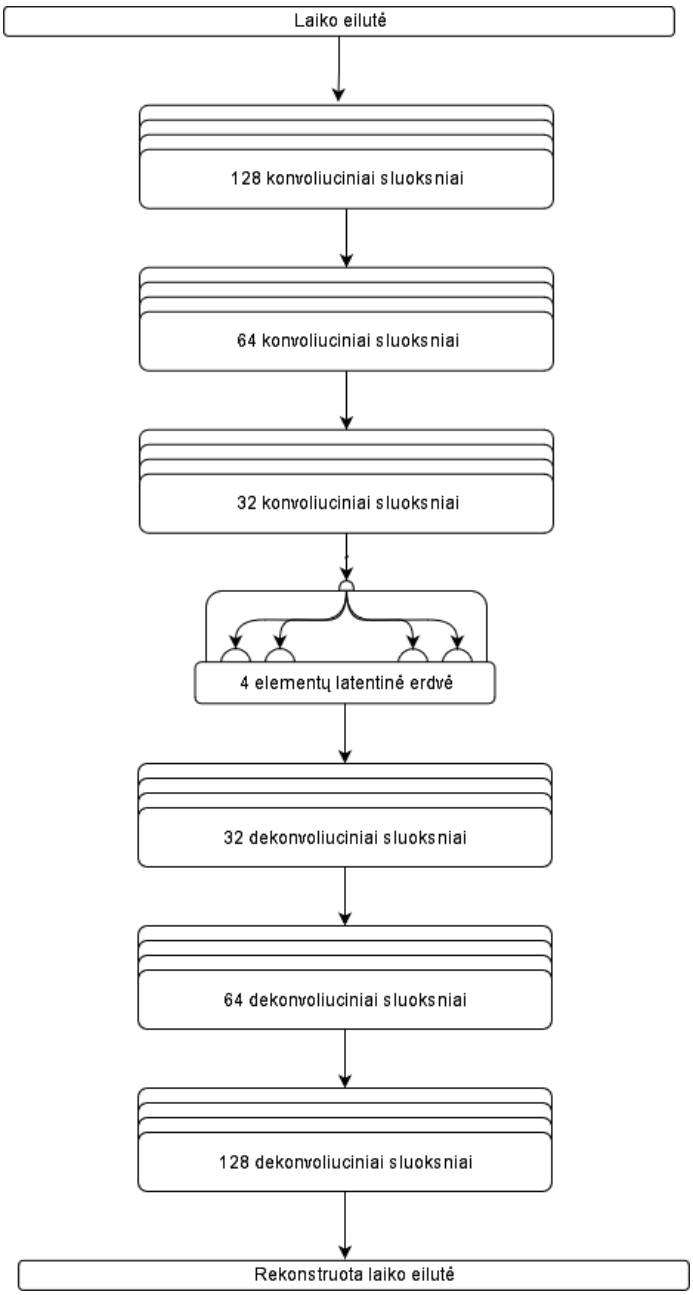


26 pav. Transformuotos laiko eilutės

3.1.1. Konvoliucinis autoenkoderis

Modelio architektūra apibendrintai pavaizduota 27 pav. Modelį galima išskirti į 7 dalis, tai enkoderį sudarančios trys konvoliucinių operacijų sluoksniai po 128, 64, 32 sluoksnius tinklui gilėjant, po kiekvieno sluoksnio suformuotoms matricoms pritaikoma ReLU aktyvacijos funkcija. Enkoderio suskaičiuoti rezultatai pilnai sujungiami su butelio kakliuke esančiu vektoriu, kuris atitinka modelio latentinę erdvę. Latentinė erdvė pilnai sujungiama su dekoderiu, kurį sudaro 3 sluoksniai dekonvoliucinių operacijų po 32, 64, 128 sluoksnius. Paskutiniai sluoksniai pilnai sujungiami su vektoriumi, kuris atitinka rekonstruotos laiko eilutės rezultatus. Visų konvoliucinių sluoksnių branduolio dydis parinktas lygus 7, o žingsnis lygus 2.

Modelį sudaro 7 780 785 optimizuojami parametrai.

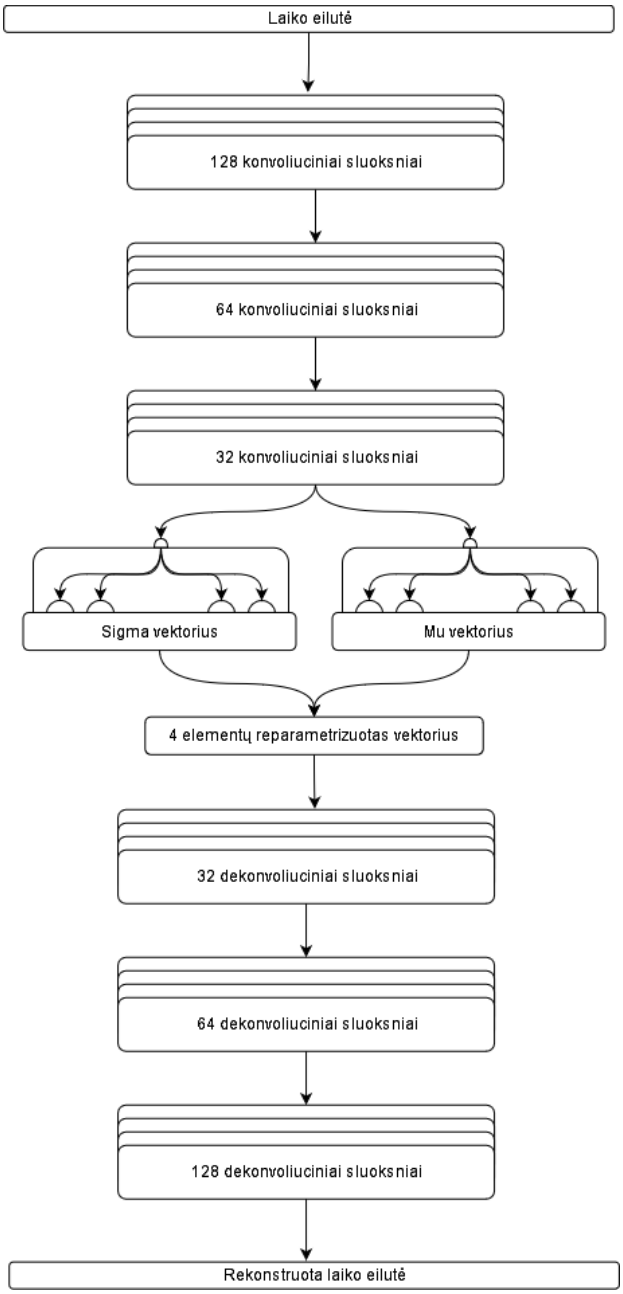


27 pav. Konvoliucinio autoenkoderio architektūra

3.1.2. Variacinis konvoliucinis autoenkoderis

Modelio architektūra apibendrintai pavaizduota **28 pav.** Modelio architektūra labai panaši į 3.1.1. punkte aprašyto modelio architektūra, pakeista tik kaip yra skaičiuojama modelio latentinė erdvė. Butelio kakliuke duomenys iš pradžių yra išskiriami į du vienodo ilgio vektorius μ ir σ, σ vektoriui įvedamas neapibrėžtumas ir šie du vektoriai yra sudedami, gaunant naują, tokio pat ilgio kaip ir μ ir σ vektoriai, vektorių.

Modelį sudaro 7 911 667 optimizuojami parametrai, kas yra 15% daugiau negu 3.1.1. punkte aprašyto modelio.



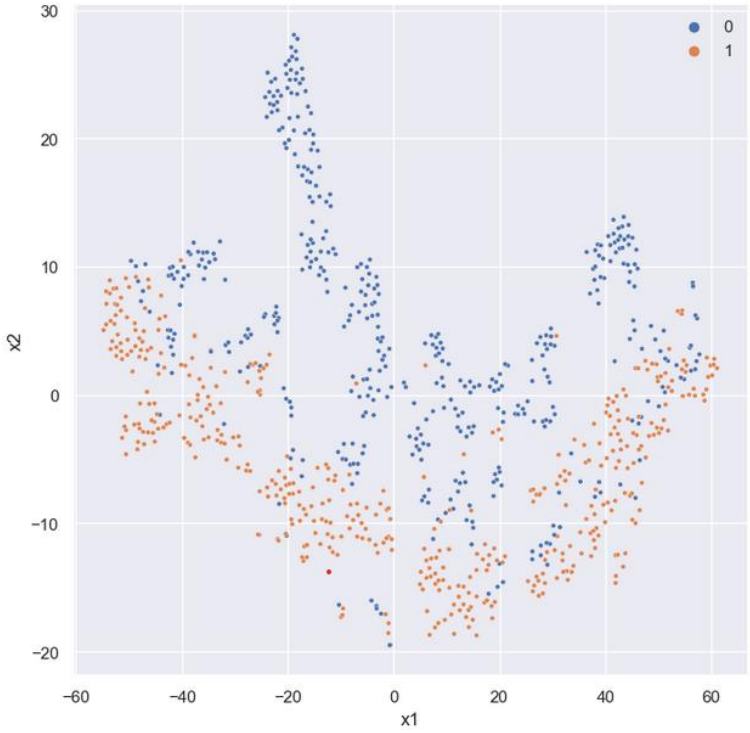
28 pav. Variacinio konvoliucinio autoenkoderio architektūra

3.1.3. Latentinių erdvių vizualizavimas

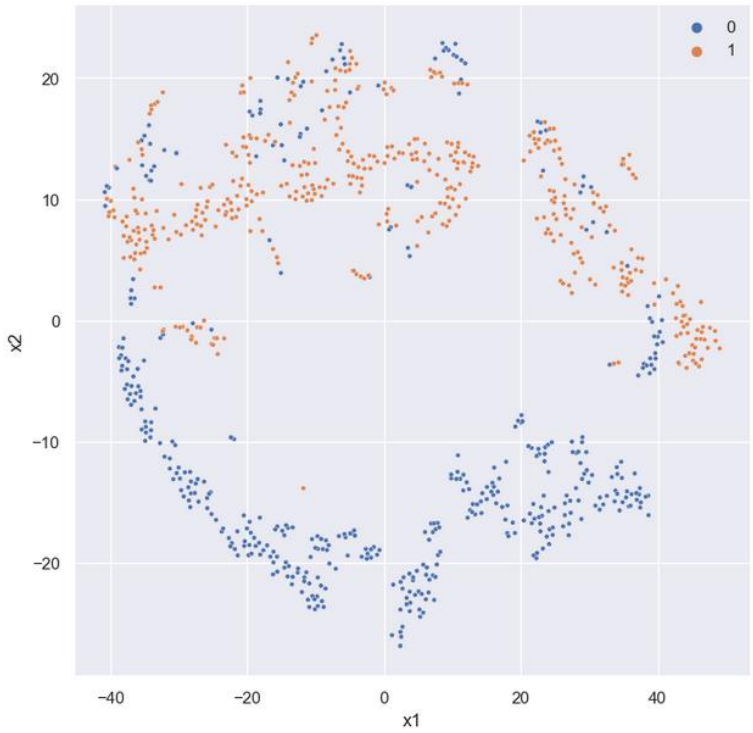
Latentinės erdvės vizualiniam tyrimui rezultatų dimensionalumas yra sumažinimas iki dviejų. Taikomi įvairūs metodai siekiant išlaikyti kuo daugiau informacijos apie latentinės erdvės kokybę, tiriami trimis metodais: t-SNE, UMAP ir principinių komponenčių metodais. Vizualizacijose skirtingomis spalvomis pažymėtos duomenų rinkinių klasės.

Aptarimui atrinkti 4 rinkiniai, tai yra ECGFiveDays, Wafer, PhalangesOutlineCorrect ir FordA.

ECGFiveDays duomenų taškų vizualizaciją t-SNE metodu autoenkoderio latentinėje erdvėje galima matyti jog dvi klasės yra atskiriamos labai gerai tiek konvoliuciniu autoenkoderiu (29 pav.), tiek variaciniu konvoliuciniu autoenkoderiu (30 pav.) kas leidžia numatyti gerą klasifikavimo rezultatą taikant šiuos požymius. Taip pat galima pastebėti jog taikant variacinį konvoliucinį autoenkoderį kaip ir galima būtų tikėtis šis klases atskyrė toliau vieną nuo kitos. Visos šio duomenų rinkinio latentinės erdvės vizualizacijos yra 1 ir 2 prieduose.

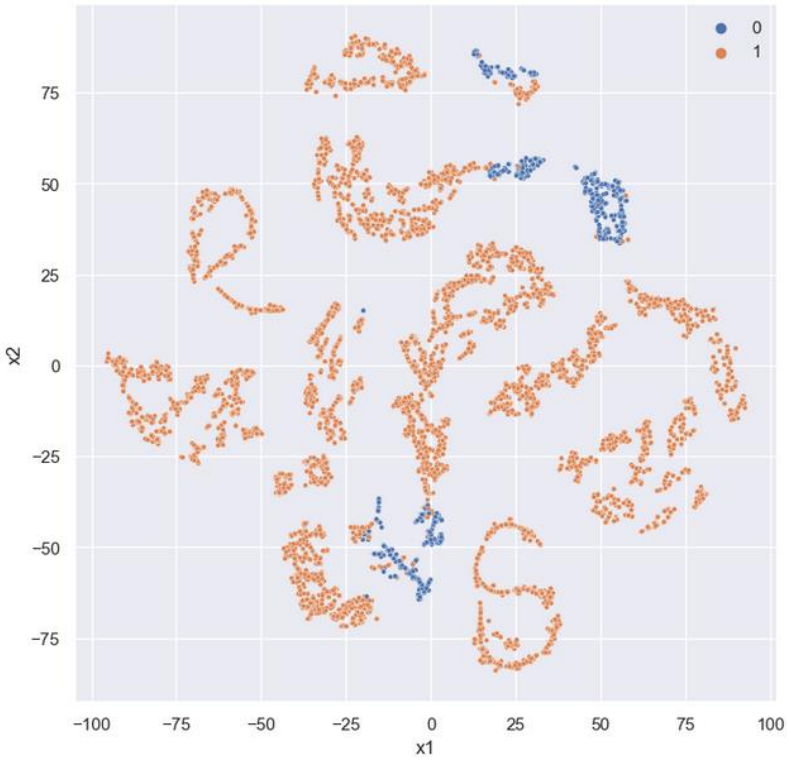


29 pav. ECGFiveDays duomenų vizualizavimas konvoliucinio autoenkoderio latentinėje erdvėje

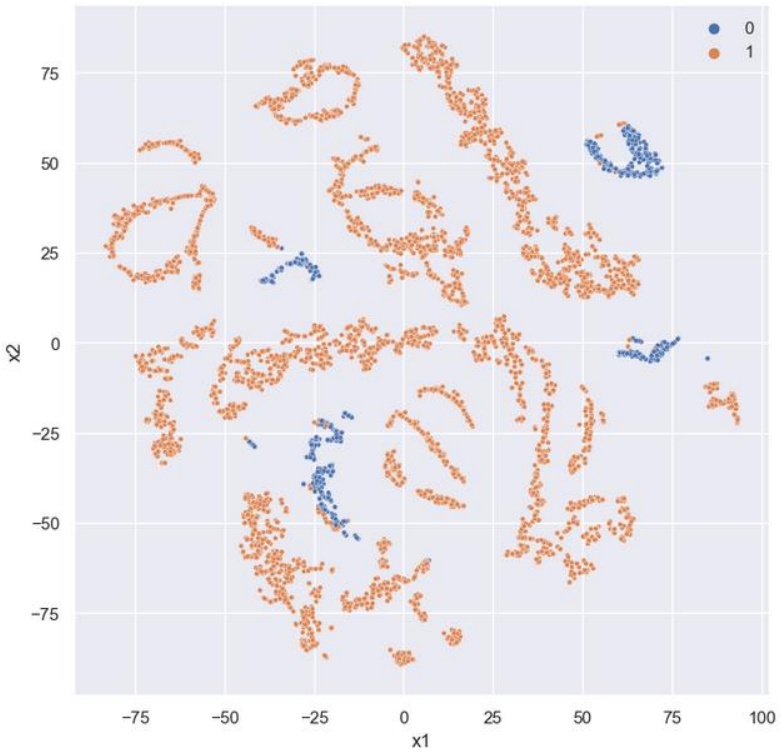


30 pav. ECGFiveDays duomenų vizualizavimas variacinio konvoliucinio autoenkoderio latentinėje erdvėje

Wafer duomenų rinkinio klasės taip pat gana aiškiai yra atskiriamos abiejų tipų autoenkoderių, tačiau skirtingų autoenkoderių atskyrimo kokybės skirtumo akimi įvertinti nebėra taip lengva, galima sakyti jie yra panašūs. Taip pat ryškus panašumas, jog abu autoenkoderiai suformavo po keturis gerai atskirtus nulinės klasės klasterius. Visos šio duomenų rinkinio latentinės erdvės vizualizacijos yra 3 ir 4 prieduose.

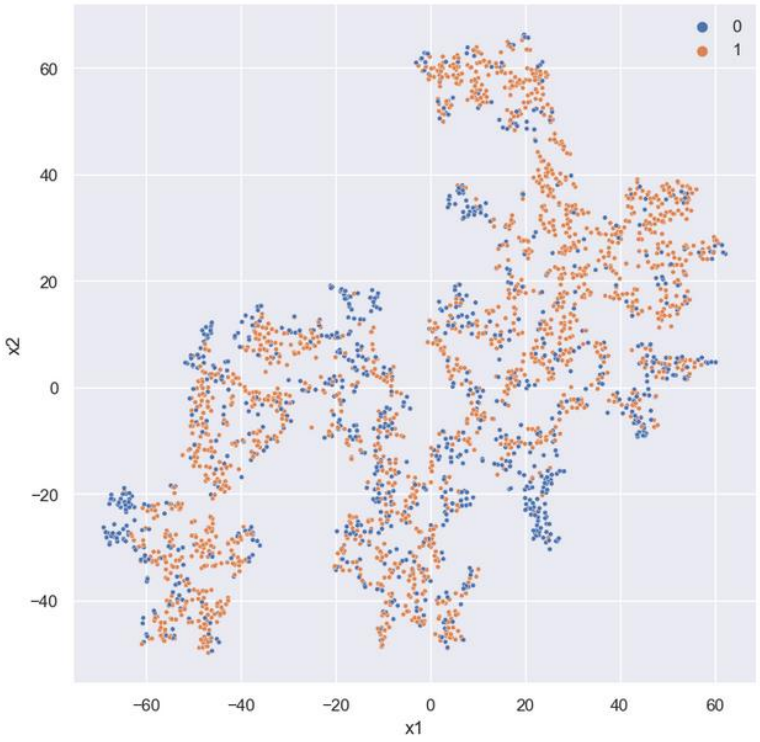


31 pav. Wafer duomenų vizualizavimas konvoliucinio autoenkoderio latentinėje erdvėje

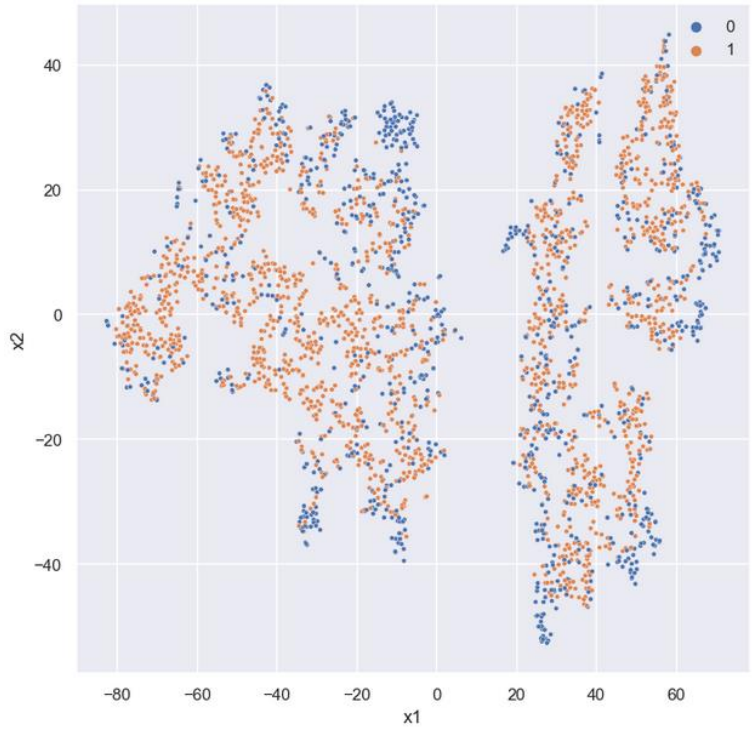


32 pav. Wafer duomenų vizualizavimas variacinio konvoliucinio autoenkoderio latentinėje erdvėje

PhalangesOutlineCorrect duomenų rinkinio apdorojimas autoenkoderiais nesuteikė tokio gero atskyrimo kaip pastarųjų dviejų aptartų modelio, dėl ko galima numanyti ir gerokai prastesnius detekcijos rezultatus. Nors duomenyse vis dar matoma struktūra, tačiau tarp dviejų klasių stebimas labai ryškus persidengimas. Visos šio duomenų rinkinio latentinės erdvės vizualizacijos yra 5 ir 6 prieduose.

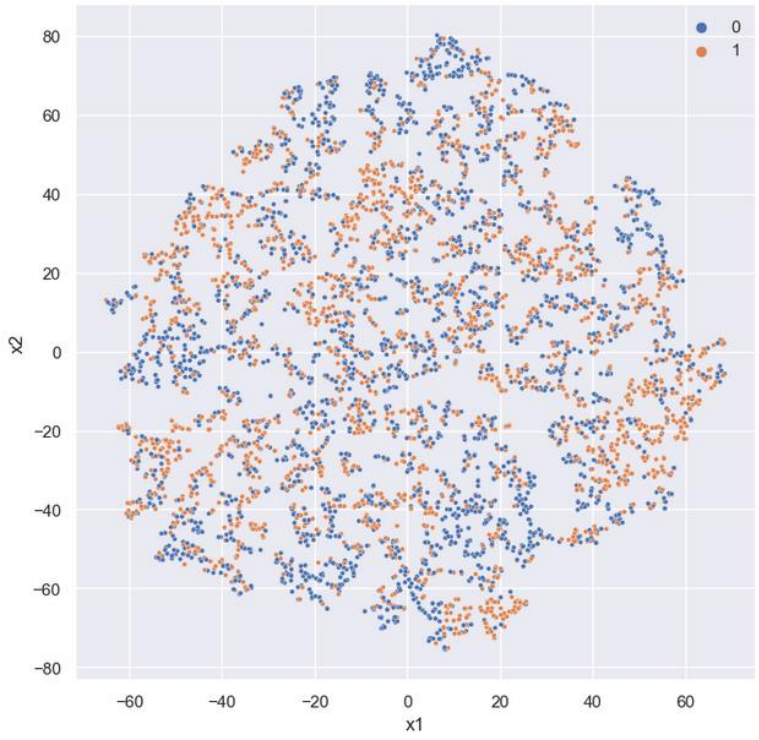


33 pav. PhalangesOutlineCorrect duomenų vizualizavimas konvoliucinio autoenkoderio latentinėje erdvėje

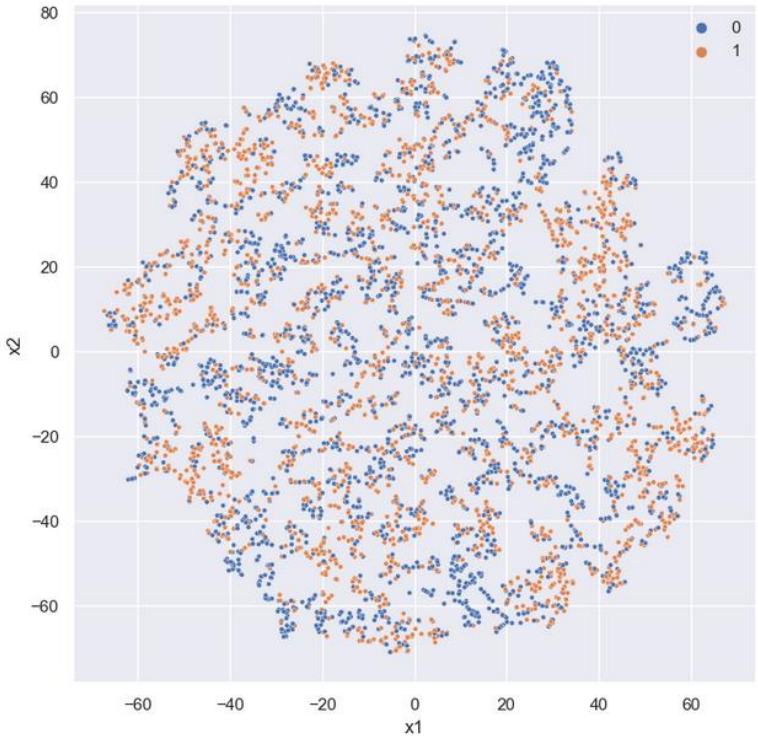


34 pav. PhalangesOutlineCorrect duomenų vizualizavimas variacinio konvoliucinio autoenkoderio latentinėje erdvėje

FordA duomenų rinkinys apdorotas autoenkoderiu duomenims nesuteikė jokios struktūros ir duomenyse matomas vien triukšmas. Galima numatyti, jog suformuotais požymiu rinkiniais atliktas klasifikavimas neatneš jokių kokybiškų rezultatų. Visos šio duomenų rinkinio latentinės erdvės vizualizacijos yra 7 ir 8 prieduose.



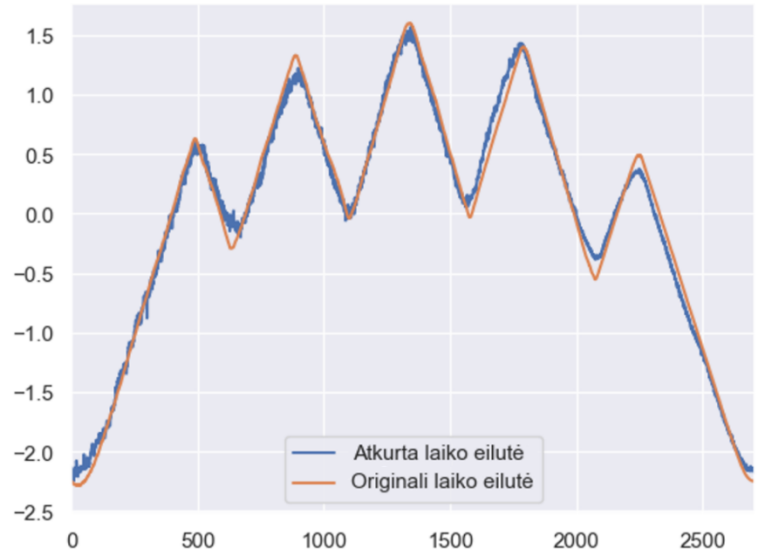
35 pav. FordA duomenų vizualizavimas konvoliucinio autoenkoderio latentinėje erdvėje



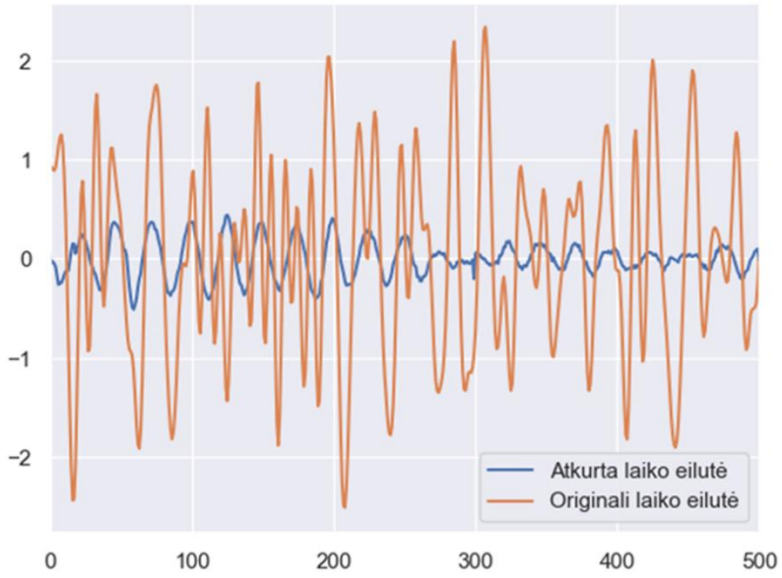
36 pav. FordA duomenų vizualizavimas variacinio konvoliucinio autoenkoderio latentinėje erdvėje

3.1.4. Laiko eilučių atkūrimas

Kadangi autoenkoderis optimizuotas laiko eilučių atkūrimui, galima vizualiai įvertinti apmokymo kokybę patikrinant su turimais duomenimis. Atkūrimo kokybė smarkiai priklauso ir nuo su laiko eilučių rinkiniu sudarytos latentinės erdvės kokybės. Duomenų rinkinių iš kurių taikant autoenkoderį nepavyko išgauti informacijos ir jie neturi prasmingos reprezentacijos autoenkoderio latentinėje erdvėje laiko eilutės nebus sėkmingai atkurtos iš latentinėje erdvėje esančių duomenų. Laiko eilutės atkurtos iš vektoriaus kurio ilgis yra lygus 4.



37 pav. Variaciniu konvoliuciniu autoenkoderiu originali ir atkurta laiko eilutės iš HandOutlines duomenų rinkinio



38 pav. Variaciniu konvoliuciniu autoenkoderiu originali ir atkurta laiko eilutės iš ShapeletSim duomenų rinkinio

3.2. Klasifikavimas

Laiko eilutes transformavus į jų požymių rinkinį tampa įmanoma atlikti klasifikavimo, arba detekcijos, uždavinius. Klasių detekcijos uždaviniui atlikti visais atvejais pasirinktas XGBoost metodas, vienas iš aibės medžiais grįstų metodų kurie yra rekomenduojami kaip baziniai modeliai įvertinimams [46]. Hperparametrų optimizavimas atliktas siekiant surasti optimalų mokymosi greitį, maksimalų medžių dydį bei minimalų svorinį koeficientą priklausomiems medžiams.

3.2.1. Detekcija taikant Catch22 požymių rinkinį

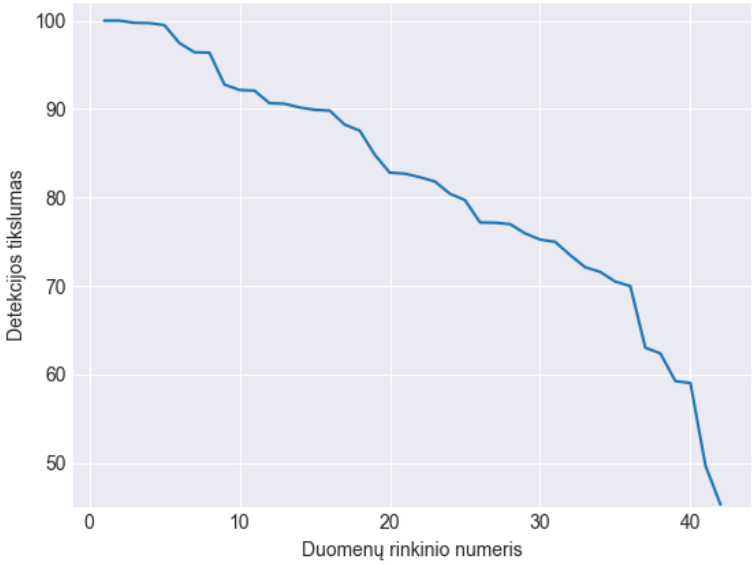
Kaip bazinis požymių modelis palyginimui pasirinktas Catch22 požymių rinkinys. Taikant juo sugeneruotus laiko eilučių požymius atlikta laiko eilučių klasifikacija. Detalūs rezultatai pavaizduoti 9 priede.

40 pav. pavaizduoti po 5 tiksliausiai ir prasčiausiai klasifikuojamų duomenų rinkinių taikant vien Catch22 požymių rinkinį. Laiko eilučių rinkinius ShapeletSim, GunPointOldVersusYoung taikant šį požymių rinkinį, pagal duomenų autorių apibrėžtas apmokymo, testavimo imtis, galima atskirti į dvi klases idealiai, arti idealių rezultatų taip pat yra FreezerSmallTrain, FreezerRegularTrain, tačiau tikslumas labai priklauso nuo duomenų rinkinio ir vertinant prasčiausiai klasifikuotus duomenų rinkinius galima pastebėti jog ne visų tikslumas siekia ten 50%.

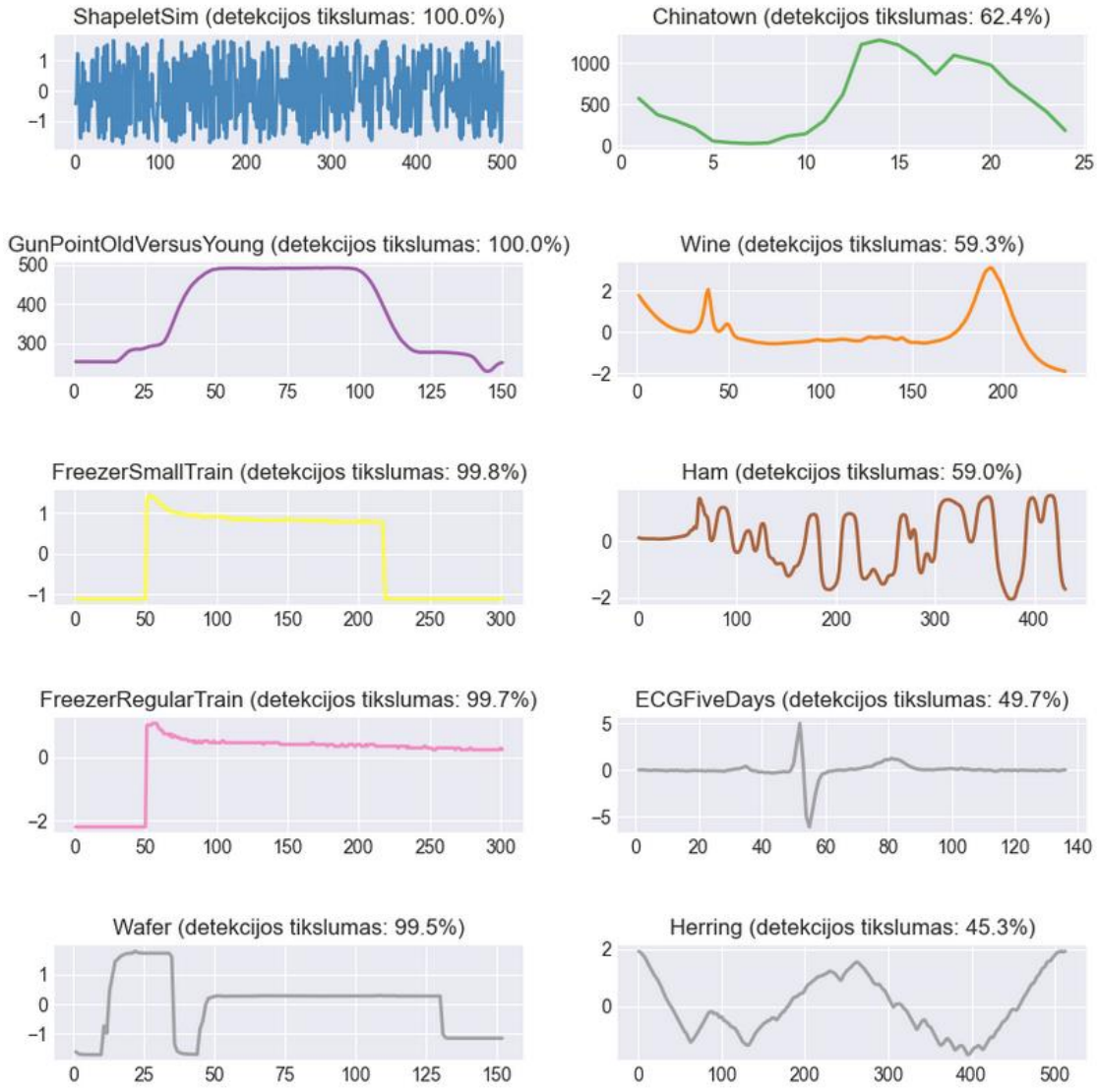
Tiek tarp tiksliausiai, tiek tarp mažiausiai tikslių laiko eilučių yra įvairių ilgių, triukšmingumo laiko eilučių, tad kaip ir galima būtų tikėtis pagal autorių aprašytus rezultatus šis požymių rinkinys yra tinkamas naudoti plačiai aibei įvairių tipų laiko eilučių [24].

Apibendrintas visų rezultatų įvertinimas grafiku pateiktas **39 pav.** pavaizduotu grafiku.

Pamatuota šio modelio greitaveika. Analizuojant visus duomenų rinkinius kartu vidutinis duomenų apdorojimo tempas yra 98,45 laiko eilutės per sekundę.



39 pav. Apibendrintas visų duomenų rinkinių detekcijos rezultatų grafikas taikant Catch22 požymių rinkinį



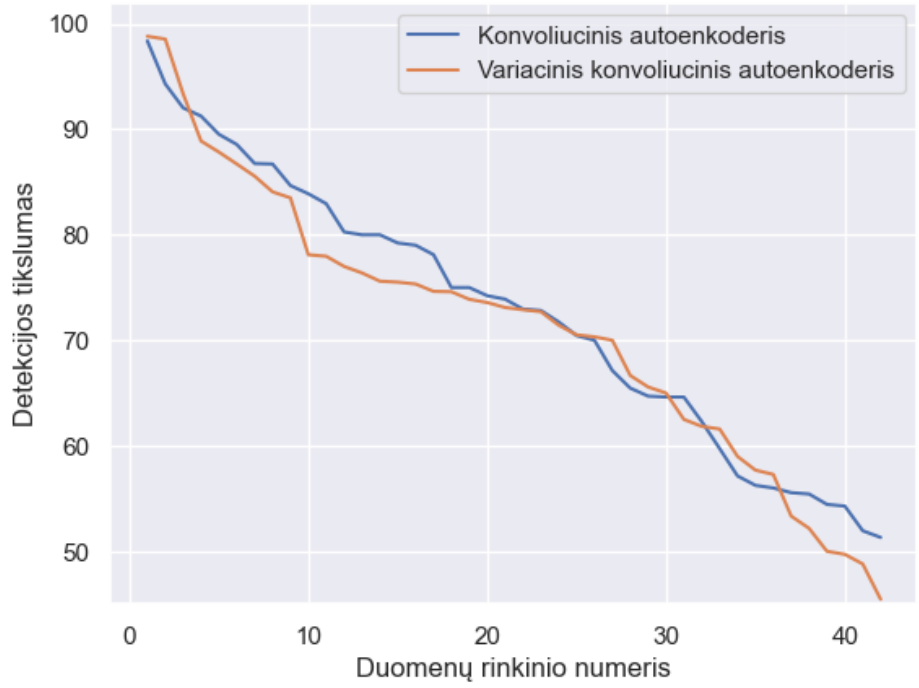
40 pav. 5 geriausiai (kairėje) ir prasčiausiai (dešinėje) įvertintų duomenų rinkinių pavyzdžiai

3.2.2. Detekcija taikant autoenkoderiu sudarytu požymiu rinkiniu

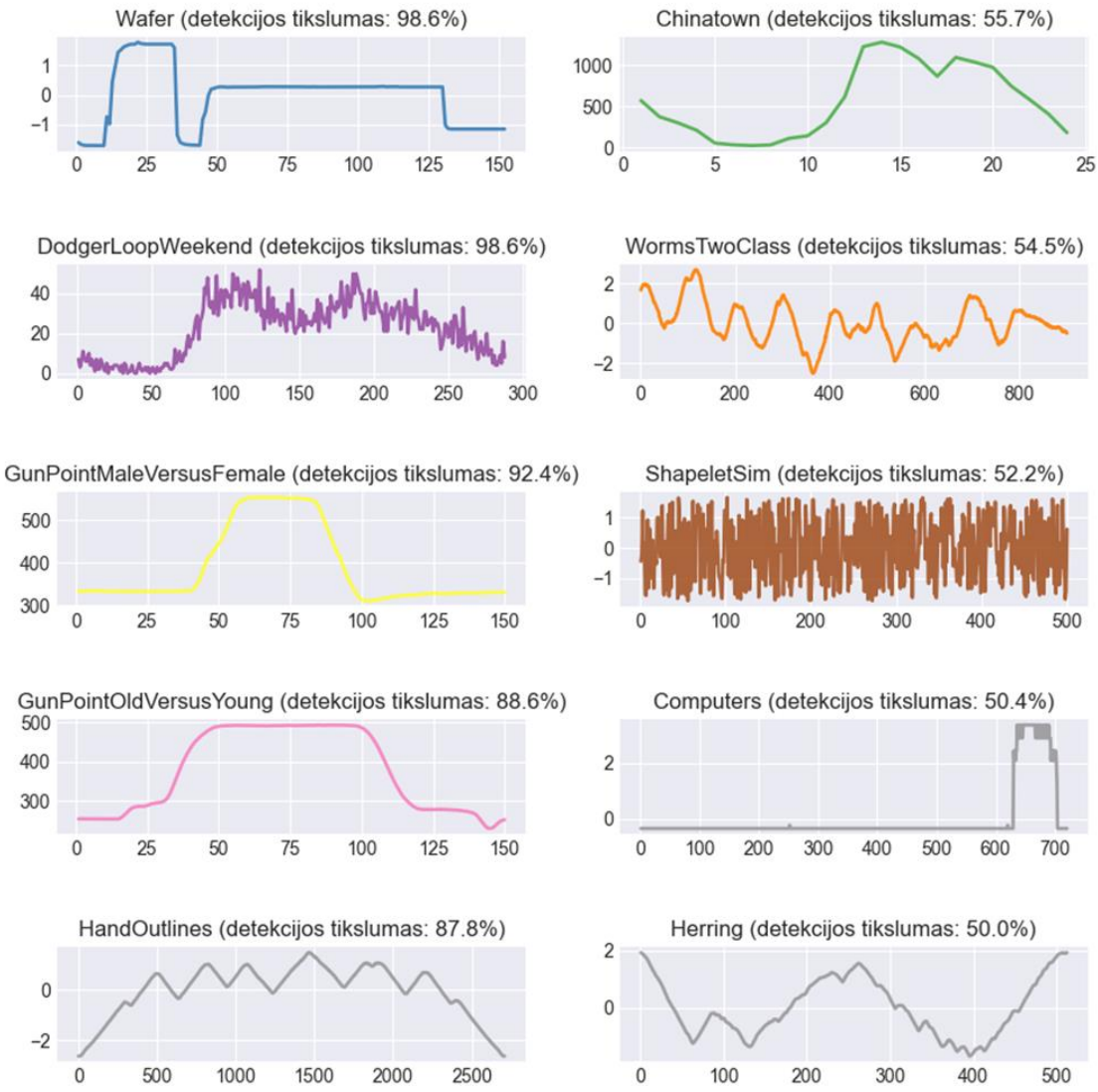
Detekcijai naudotas autoenkoderiu sudarytas požymių rinkinys, kurį sudaro vos 4 elementų ilgio vektorius. Požymiai sudaryti su abiejų tipų autoenkoderiais ir atlikti įvertinimai. Bendrą modelių tikslumo įvertinimą galima matyti 41 pav.

Taikant abu šiuos požymių rinkinius nustatyta, kad rezultatai tarp skirtingų modelio architektūrų drastiškai vienas nuo kito nesiskiria, skaičiuojant kartu tarp visų 42 duomenų rinkinių variacinio konvoliucinio autoenkoderio vidutinis tikslumo įvertis yra 72,43%, konvoliucinio autoenkoderio 71,50%.

42 pav. pavaizduoti po 5 geriausiai ir prasčiausiai klasifikuotus laiko eilučių rinkinius. Kaip ir taikant Catch22 požymių rinkinį, taip ir šiais automatinės požymių inžinerijos metodais sudarytais požymiais ne visos laiko eilutės yra klasifikuojamos gerai. Šie požymių rinkiniai gerai apibendrina pačias laiko eilutės formas, tačiau prastai atspindi požymius apie triukšmo pobūdį. Tokioms laiko eilutėms kaip DodgerLoopWeekend kur laiko eilutės signalas atrodo pakankamai triukšmingas, tačiau egzistuoja pakankamai tiksli laiko eilutės forma sudarytas požymių rinkinys veikia gerai. Tačiau laiko eilutėms kur informacija apie klases yra paslėpta požymiuose į kuriuos autoenkoderiai nekreipia dėmesio, arba triukšmo lygis yra per didelis, autoenkoderiais sudarytu požymiu rinkiniu nėra įmanoma kokybiškai atlikti klasifikavimo. ShapeletSim duomenų rinkinio atveju, nuo 100% klasifikavimo tikslumo taikant Catch22 požymių rinkinį, tikslumas nukrito iki 52.2% taikant variacinio konvoliucinio autoenkoderio sudarytus požymius. Chinatown duomenų rinkinys čia išsiskiria, kadangi pagal autorių apibrėžtą apmokymo ir testavimo imtį, kuri apibrėžta kaip 20 specifinių laiko eilučių apmokymo imtyje ir net 345 laiko eilutės testavimo imtyje, duomenų gautų šiais požymių rinkiniais nepakanka gerai atskirti skirtingų klasių rezultatus. Taikant labiau tradicinį 80/20 duomenų padalinimą šios laiko eilutės klasifikavimo tikslumas išauga iki 98% testavimo imtyje.



41 pav. Variacinio konvoliucinio ir konvoliucinio autoenkoderių tikslumo įverčių grafikas



42 pav. Variacinio konvoliucinio autoenkoderio geriausių (kairėje) ir prasčiausių (dešinėje) detekcijos rezultatų rinkinių pavyzdžiai

Pamatuota šio modelio greitaveika. Analizuojant visus duomenų rinkinius kartu variacinio konvoliucinio autoenkoderio vidutinis duomenų apdorojimo greitis yra 298,02 laiko eilutės per sekundę, o konvoliucinio autoenkoderio 298,51 laiko eilutės per sekundę. Abu šie įverčiai yra labai panašūs, ir lenkia tiesiog Catch22 požymių rinkinio greitaveiką daugiau nei 3 kartus.

3.2.3. Detekcija taikant apjungtą požymių rinkinį

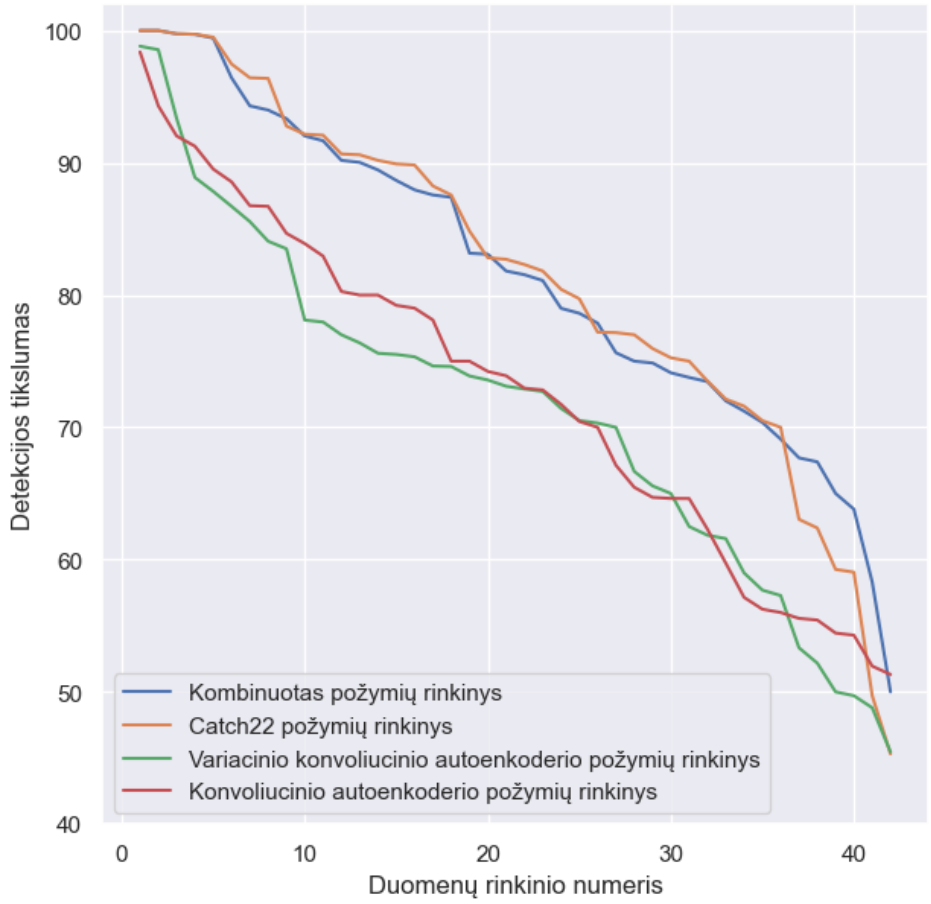
Autoenkoderiu sudaryti požymių rinkiniai yra gerai tinkami tam tikriems duomenų rinkiniams, tačiau jie nėra tiek universalūs kiek Catch22 požymių rinkinys, tačiau jo greitaveika yra 3 kartus didesnė. Todėl šie požymių rinkiniai yra sujungti, siekiant pasisavinti geriausias abiejų tipų modelių savybes.

Siekiama pašalinti tuos Catch22 požymius, kurių informacija yra užkoduoda 4-iuose autoenkoderio sudarytuose požymiuose. Modelių požymių sujungimas yra atliktas žingsniais:

1. Sudarytas apjungtas visų požymių rinkinys, t.y. 22 Catch22 požymiai ir 4 autoenkoderiais sudaryti požymiai.
2. Apskaičiuojamas vidutinis detekcijos tikslumas visiems duomenų rinkiniams su esamu požymių rinkiniu
3. Paeiliui po vieną pašalinami ir grąžinami Catch22 požymiai ir suskaičiuojami vidutiniai tikslumai
4. Požymis kurio pašalinimas turėjo mažiausią įtaką vidutiniams detekcijos rezultatams yra pašalinamas iš požymių rinkinio
5. Jeigu dar yra daugiau nei 1 nepašalintas požymis grįžtama į antrą žingsnį.

Taikant šį metodą pašalinti 11 iš 22 požymių:

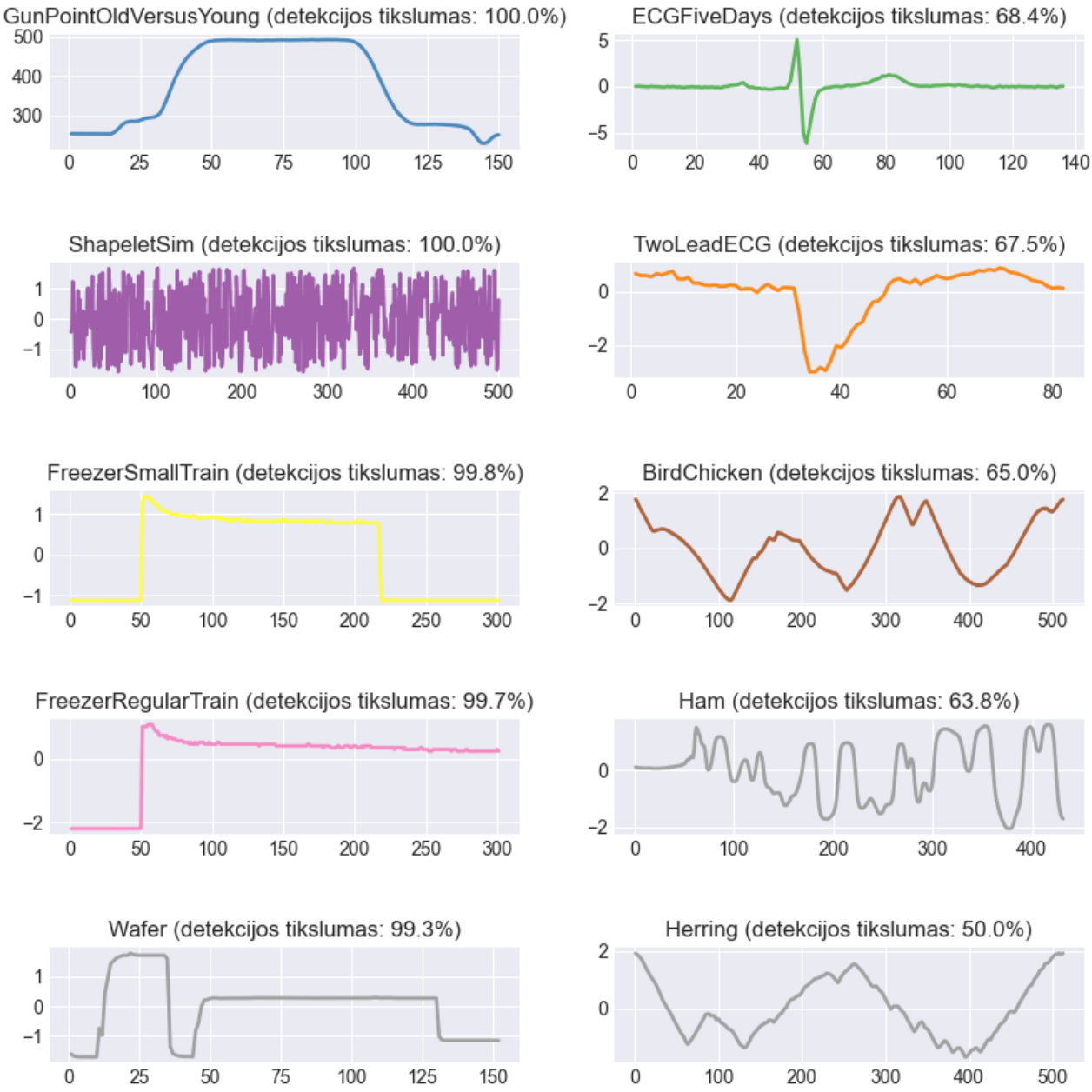
- DN_HistogramMode_5
- DN_HistogramMode_10
- SB_BinaryStats_diff_longstretch0
- DN_OutlierInclude_p_001_mdrmd
- DN_OutlierInclude_n_001_mdrmd
- CO_f1ecac
- CO_FirstMin_ac
- SP_Summaries_welch_rect_centroid.
- FC_LocalSimple_mean3_stderr
- SB_TransitionMatrix_3ac_sumdiagcov
- SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1



43 pav. Skirtingų požymių rinkinių tikslumo įvertinimo palyginimas

Galutinį požymių rinkinį sudaro 15 elementų. Bendrą įvairių modelių tikslumo palyginimą galima matyti 43 pav. Gautas naujas požymių rinkinys nenusileidžia Catch22 požymių rinkiniui, nors bendras požymių skaičius buvo sumažintas net beveik trečdaliu. Negana to, tam tikriems duomenų rinkiniams pavyko netgi pagerinti detekcijos rezultatus, pvz. ECGFiveDays detekcijos rezultatas lyginant su detekcija taikant tik Catch22 požymių rinkinį pagerėjo nuo 49,7% tikslumo iki 69,1% tikslumo. O informaciją kurios trūko norint idealiai nustatyti klases GunPointOldVersusYoung ir ShapeletSim autoenkoderiams yra gaunama dėka likusių Catch22 požymių. Tačiau liko ir prastai klasifikuojamų duomenų, Herring duomenų rinkinio detekcijos kokybei požymių apjungimas įtakos neturėjo. Taip pat verta paminėti, jog kombinuotas požymių rinkinys kiek pagerino rezultatus prasčiausius rezultatus taikant catch22 požymių rinkinį.

Catch22 požymių rinkinio sudarymo greitaveika taikant šį metodą taip pat yra smarkiai pagerinama. Laiko eilučių apdorojimo greitis išaugo nuo vidutiniškai 98,25 apdorojamų laiko eilučių per sekundę iki 199.36 apdorojamų laiko eilučių per sekundę.

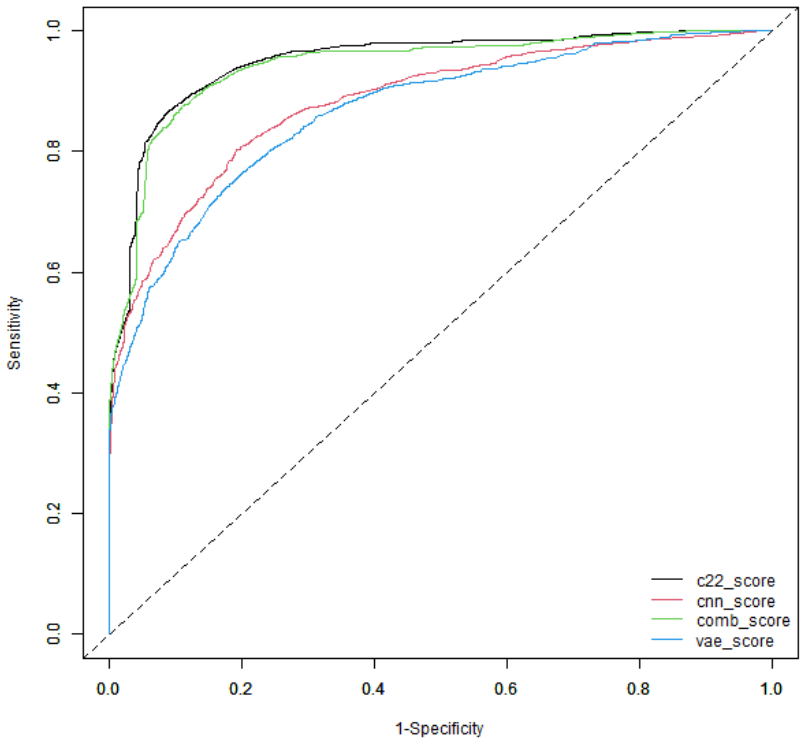


44 pav. Apjungto požymių rinkinio geriausių (kairėje) ir prasčiausių (dešinėje) detekcijos rezultatų rinkinių pavyzdžiai

3.3. Modelių ROC kreivės

Papildomas modelių palyginimas atliktas *easyROC* programiniu paketu. Sudarant ROC kreives naudotos visų rinkinių laiko eilutės kartu. Palyginimas atliktas principu „kiekvienas su kiekvienu“, vertinant AUC charakteristikas. Hipotezę dėl AUC įverčių vienodumo visais atvejais galima atmesti ir priimti jog visi plotai esantys po kreivėmis yra skirtingi ($p < 0,01$). Šis palyginimas šiam atvejui nėra labai reikšmingas dėl didelio skaičiaus testuojamų laiko eilučių, tad bet koks mažas skirtumas tarp galutinio įverčio bus vertinamas kaip statistiškai reikšmingu. Tačiau vertinant rezultatus (

je), galima pastebėti, kad Catch22 ir kombinuoto modelio AUC įverčiai skiriasi labai neženkliai, vos per 0,0068, kas daugeliu panaudojimo atveju leidžia šiuos modelius laikyti lygiaverčiais. Variacinio konvoliucinio ir konvoliucinių autoenkoderių modeliai pavieniui kaip ir kituose palyginimuose (43 pav.) yra gerokai prastesni nei kombinuotas arba catch22 modeliai.



45 pav. Modelių ROC kreivės

5 lentelė modelių ROC kreivių palyginimas

I modelis	**J modelis**	**AUC(I)**	**AUC(J)**	**|I – J|**	**SE(|I – J|)**	**p vertės (patikslintos)**
Variacinis AE	Konvoliucinis AE	0,8661	0,8815	0,0154	0,0028	0,0000
Variacinis AE	Catch22	0,8661	0,9484	0,0823	0,0024	0,0000
Variacinis AE	Kombinuotas	0,8661	0,9416	0,0755	0,0024	0,0000
Konvoliucinis AE	Catch22	0,8815	0,9484	0,0669	0,0023	0,0000
Konvoliucinis AE	Kombinuotas	0,8815	0,9416	0,0601	0,0023	0,0000
Catch22	Kombinuotas	0,9484	0,9416	0,0068	0,0018	0,0013

3.4. Atvejo analizė. Wafer duomenų rinkinys.

Puslaidininkinės mikroelektronikos gamyboje taikomi sudėtingi procesai kurių metu įvairių medžiagos sluoksniais yra uždedamos ant silikoninio lusto ir graviruojamos pagal tam tikrą šabloną, tam, kad būtų suformuota norima elektros grandinė. Dažniausiai graviravimas yra atliekamas fotolitografijos principu, kurio metu naudojant labai didelio tikslumo ultravioletinę spinduliuotę graviruojama norima struktūra ant lusto paviršiaus. Proceso metu paviršius būna papildomai padengiamas fotorezistine medžiaga, kurią vėliau reikia pašalinti, tas dažniausiai daroma patalpinant lustą į vakuuminę kamerą kuri vėliau užpildoma reaktyvia plazma. Visas šis procesas apima daugiau nei 250 žingsnių ir visuose iš jų yra galimybė lustui būti sugadintam, tai yra galutinis lusto produktas veiks prasčiau nei numatyta jo parametruose, sumažės jo patikimumas, arba jis iš viso neveiks. Lustų kurių parametrai surinkti į darbe naudotą duomenų rinkinį, gamybos proceso metu parametrai yra stebimi 450 kokybės kontrolės sensorių. Srities ekspertų buvo atrinkti 6 kritiniai kokybę stebintys sensoriai kurių duomenimis šis duomenų rinkinys yra pagrįstas.

Tikslių arba apytikslių kaštų kuriuos lustus gaminančios įmonės patiria dėl neišvengiamo broko nėra, tačiau natūrali išlaidų optimizavimo užduotis yra šių kaštų, kartu ir broko, minimizavimas. Tačiau broko visiškai nepanaikinus ir neaptikus broko prieš galutinį produktą atiduodant klientui yra sulaukiama papildomų kaštų susijusiu su techniniu, garantiniu aptarnavimu, kas lemia išaugusią produkto kainą ir kartu mažesnį įmonės gaminamų produktų konkurencingumą ir galimą reputacinę žalą, todėl ankstyvas broko aptikimas yra ne mažiau aktuali užduotis gamybinėms įmonėms.

6 ir 7 lentelėse pateikiami detekcijos rezultatai pagal rinkinio autorių apibrėžtą mokymosi ir testavimo imtį, nestandartiškai mokymuisi skirta 1000 laiko eilučių, o testavimui 6164, kas apytiksliai atitinka 14:86 laiko eilučių imčių santykį, smarkiai besiskiriantį nuo standartinio 80:20 padalinimo. Naudojant labiau įprastą laiko eilučių padalinimo santykį galutinis detekcijos tikslumas tampa gerokai tikslesnis (99,9%), tačiau vertinimui ir tolesniam komentavimui naudojami rezultatai gauti pagal autorių apibrėžtą duomenų padalijimą.

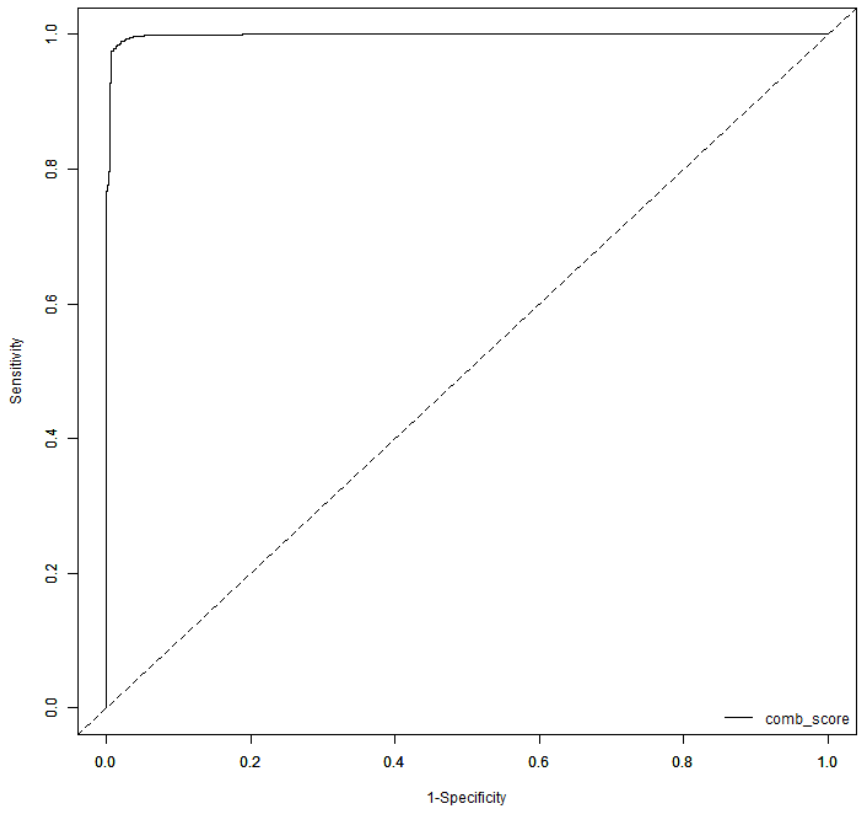
Taikant požymiais grįstus metodus gaunami geri tikslumai. Kombinuoto modelio atveju teisingai yra aptinkama 95% brokuotų lustų, tuo tarpu 99,9% nebrokuotų lustų yra teisingai nustatomi kaip geri. Labiau vertinant broko aptikimą galimas slenksčio modifikavimas, darbe slenkstis parinktas toks jog būtų maksimizuojamas skirtumas tarp teisingai kaip gerų nustatytų lustų skaičiaus ir antro tipo klaidų skaičiaus, tai yra maksimizuojamas 6 lentelės dešinėje pusėje esančių skaičių skirtumas pagal (7) formulę. Labai geri detekcijos rezultatai atsispindi ir 46 pav. ROC kreivėje, kurios AUC įvertis yra lygus 0.9984

6 lentelė Kombinuoto modelio sumaišymo matrica Wafer duomenų rinkiniui

	Prognozuojamas brokuotas lustas	Prognozuojamas geras lustas
Brokuotas lustas	659	38
Geras lustas	6	5461

7 lentelė Skirtingų modelių Wafer duomenų rinkiniui tikslumo įverčiai

Variacinis konvoliucinis autoenkoderis	Konvoliucinis autoenkoderis	Catch22	Kombinuotas
0.988157	0.983777	0.994971	0.992862



46 pav. Kombinuoto modelio ROC kreivė Wafer duomenų rinkiniui.

Išvados

1. Požymių taikymas užduotims susijusioms su laiko eilutėmis dažniausiai remiasi požymių rinkiniais sudarytais ekspertinės analizės būdu iš sąrašo galimų, žmogaus sugalvotų, požymių. Automatinis požymių generavimas yra perspektyvus būdas išvengti dirbtinio apsiribojimo tik tam tikra iš anksto apibrėžta aibe ir leidžia generuoti aktualius požymius automatiškai be ekspertinių įžvalgų.
2. Konvoliucinių neuroninių tinklų taikymas leido sudėtingas laiko eilučių „formas“ sutraukti vos iki 4 skaitinių požymių. Tačiau laiko eilutėms pasižyminčiomis aukšta entropija šiuo metodu sugeneruoti požymiai nėra prasmingi, kadangi nėra jokios pakankamai gerai apibrėžtos laiko eilutės formos.
3. Vieną kartą apmokytas autoenkoderis su plačiu spektru duomenų rinkinių yra tinkamas visoms laiko eilutėms tirti ir modelio nereikia kas kartą iš naujo apmokyti.
4. Konvoliucinio ir variacinio konvoliucinio autoenkoderių detekcijos rezultatai yra panašūs. Tam tikrais atvejais kuomet konvoliucinio autoenkoderio klasės yra atskiriamos gerai, kaip ECGFiveDays atveju, variacinio autoenkoderio jos gali būti atskirtos geriau. Tačiau apskritai variacinio autoenkoderio taikymas tik retu atveju leidžia nors kiek ženkliau padidinti tikslumą, o dėl variaciniui autoenkoderiui būdingo neapibrėžtumo ir autoenkoderiui negebant surasti geros duomenų reprezentacijos neapibrėžtumas gali lemti sunkesnį tolesnio, detekcijos, modelio apmokymą ir dėl to prastesnius rezultatus. Autoenkoderio apmokymo kokybei smarkią įtaką turėjo mokymosi greičio keitimo planas. Taikant modifikuota kosinuso pjūklinį tvarkaraštį su slopinimo koeficientu modelis konverguoja patikimai. Taikant kitus išbandytus tvarkaraščius, dažnu atveju modelio apmokymą tekdavo kartoti kelis kartus kol modeliui „pasisekdavo“ konverguoti. Apmokius modelį su visomis laiko eilutėmis gauta laiko eilučių informacijos reprezentacija labai tankiame formate, lyginant su kitais požymių rinkiniais tai yra bene tankiausiai informaciją suspaudžiantis metodas, dėl ko tai geras metodas norint mažinti požymių dimensionalumą, kovoti su „dimensionalumo prakeiksmu“.
5. Autoenkoderiais sugeneruoti požymių rinkiniai pagal detekcijos rezultatų kokybę daugeliu atveju nusileidžia catch22 požymių rinkiniui. Tačiau modelio pranašumas yra jo greitaveikoje, vidutiniškai jis yra 3 kartus greitis lyginant su catch22 požymių pritaikymu detekcijoje.
6. Autoenkoderiais grįsti požymių rinkiniai sėkmingai apjungti su catch22 požymių rinkiniu pašalinant požymius kuriuose esanti informacija yra perteikiama autoenkoderiu grįstu modeliu. Pašalinus 11 iš 22 catch22 požymių ir prijungus vos 4 autoenkoderiu sudarytus požymius detekcijos tikslumas žymiai nepasikeitė, tačiau kombinuoto modelio apdorojamų laiko eilučių skaičius per laiko vienetą išaugo dvigubai, lyginant su vien catch22 požymių rinkinio taikymu.

Literatūros sąrašas

1. HYNDMAN, R. ir kt. *tsfeatures: Time Series Feature Extraction* [interaktyvus]. . 2018. .

2. TALAGALA, T.S. ir kt. Meta-learning how to forecast time series. In *Monash Econometrics and Business Statistics Working Papers* . 2018. Vol. 6, no. 18, p. 16. .

3. MONTERO-MANSO, P. ir kt. FFORMA: Feature-based forecast model averaging. In *International Journal of Forecasting* . 2020. Vol. 36, no. 1, p. 86–92. .

4. MAKRIDAKIS, S. - HIBON, M. The M3-Competition: results, conclusions and implications. In *International Journal of Forecasting* . 2000. Vol. 16, no. 4, p. 451–476. .

5. WANG, X. ir kt. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. In *Neurocomputing* . 2009. Vol. 72, no. 10, p. 2581–2594. .

6. REID, D. A comparison of forecasting techniques on economic time series. In *Forecasting in Action. Operational Research Society and the Society for Long Range Planning* . 1972. .

7. COLLOPY, F. - ARMSTRONG, J.S. Rule-Based Forecasting: Development and Validation of an Expert Systems Approach to Combining Time Series Extrapolations. In *Management Science* . 1992. Vol. 38, no. 10, p. 1394–1414. .

8. MAKRIDAKIS, S. ir kt. The M4 Competition: Results, findings, conclusion and way forward. In *International Journal of Forecasting* . 2018. Vol. 34, no. 4, p. 802–808. .

9. CIREŞAN, D.C. ir kt. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two* . Barcelona, Catalonia, Spain: AAAI Press, 2011. p. 1237–1242. [žiūrėta 2022-04-09]. .

10. TAJBAKHSH, N. ir kt. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? In [interaktyvus]. 2017. [žiūrėta 2022-04-09]. . Prieiga per internetą: <https://arxiv.org/abs/1706.00712>.

11. AZIZPOUR, H. ir kt. From generic to specific deep representations for visual recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [interaktyvus]. Boston, MA, USA: IEEE, 2015. p. 36–45. [žiūrėta 2022-04-09]. Prieiga per internetą: <http://ieeexplore.ieee.org/document/7301270/>.

12. PEREIRA, J. - SILVEIRA, M. Unsupervised Anomaly Detection in Energy Time Series Data Using Variational Recurrent Autoencoders with Attention. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* [interaktyvus]. Orlando, FL: IEEE, 2018. p. 1275–1282. [žiūrėta 2022-04-09]. Prieiga per internetą: <https://ieeexplore.ieee.org/document/8614232/>.

13. HOCHREITER, S. - SCHMIDHUBER, J. Long Short-Term Memory. In *Neural Computation* . 1997. Vol. 9, no. 8, p. 1735–1780. .

14. GRAVES, A. ir kt. Speech Recognition with Deep Recurrent Neural Networks. In *arXiv:1303.5778 [cs]* [interaktyvus]. 2013. [žiūrėta 2022-04-09]. . Prieiga per internetą: <http://arxiv.org/abs/1303.5778>.

15. RUMELHART, D.E. ir kt. Learning representations by back-propagating errors. In *nature* . 1986. Vol. 323, no. 6088, p. 533–536. .

16. KINGMA, D.P. - WELLING, M. Auto-Encoding Variational Bayes. In [interaktyvus]. 2013. [žiūrėta 2022-04-09]. . Prieiga per internetą: <https://arxiv.org/abs/1312.6114>.

17. FOSTER, D. *Generative deep learning: teaching machines to paint, write, compose, and play*. . First edition. Ed. Sebastopol, CA: O'Reilly Media, Inc, 2019. 308 p. ISBN 978-1-4920-4194-8.

18. ZIMEK, A. - SCHUBERT, E. Outlier Detection. In LIU, L. - ÖZSU, M.T.Sud. *Encyclopedia of Database Systems* [interaktyvus]. New York, NY: Springer New York, 2017. p. 1–5. [žiūrėta 2022-04-09]. ISBN 978-1-4899-7993-3Prieiga per internetą: <http://link.springer.com/10.1007/978-1-4899-7993-3_80719-1>.

19. AN, J. - CHO, S. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. In . 2015. .

20. YOUSEFI-AZAR, M. ir kt. Autoencoder-based feature learning for cyber security applications. In *2017 International Joint Conference on Neural Networks (IJCNN)* [interaktyvus]. Anchorage, AK, USA: IEEE, 2017. p. 3854–3861. [žiūrėta 2022-04-09]. Prieiga per internetą: <http://ieeexplore.ieee.org/document/7966342/>.

21. FAN, C. ir kt. Deep learning-based feature engineering methods for improved building energy prediction. In *Applied Energy* . 2019. Vol. 240, p. 35–45. .

22. MAGGIPINTO, M. ir kt. A Convolutional Autoencoder Approach for Feature Extraction in Virtual Metrology. In *Procedia Manufacturing* . 2018. Vol. 17, p. 126–133. .

23. NG, A. - OTHERS Sparse autoencoder. In *CS294A Lecture notes* . 2011. Vol. 72, no. 2011, p. 1–19. .

24. LUBBA, C.H. ir kt. catch22: CAnonical Time-series CHaracteristics. In [interaktyvus]. 2019. [žiūrėta 2022-04-09]. . Prieiga per internetą: <https://arxiv.org/abs/1901.10200>.

25. CHRIST, M. ir kt. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). In *Neurocomputing* . 2018. Vol. 307, p. 72–77. .

26. NUN, I. ir kt. FATS: Feature Analysis for Time Series. In [interaktyvus]. 2015. [žiūrėta 2022-04-09]. . Prieiga per internetą: <https://arxiv.org/abs/1506.00010>.

27. DEMPSTER, A. ir kt. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. In [interaktyvus]. 2019. [žiūrėta 2022-04-09]. . Prieiga per internetą: <https://arxiv.org/abs/1910.13051>.

28. FACEBOOK Kats. In [interaktyvus]. [žiūrėta 2022-04-09]. Prieiga per internetą: <https://facebookresearch.github.io/Kats/>.

29. WOLPERT, D.H. - MACREADY, W.G. No free lunch theorems for optimization. In *IEEE Transactions on Evolutionary Computation* . 1997. Vol. 1, no. 1, p. 67–82. .

30. MUÑOZ, M.A. ir kt. Instance spaces for machine learning classification. In *Machine Learning* . 2018. Vol. 107, no. 1, p. 109–147. .

31. KANG, Y. ir kt. GRATIS: GeneRAting TIme Series with diverse and controllable characteristics. In [interaktyvus]. 2019. [žiūrėta 2022-04-09]. . Prieiga per internetą: <https://arxiv.org/abs/1903.02787>.

32. MIDDLEHURST, M. ir kt. Scalable Dictionary Classifiers for Time Series Classification. In YIN, H. ir kt.Sud. *Intelligent Data Engineering and Automated Learning – IDEAL 2019* [interaktyvus]. Cham: Springer International Publishing, 2019. p. 11–19. [žiūrėta 2022-05-02]. ISBN 978-3-030-33606-6Prieiga per internetą: <http://link.springer.com/10.1007/978-3-030-33607-3_2>.

33. BAGNALL, A. ir kt. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. In *IEEE Transactions on Knowledge and Data Engineering* . 2015. Vol. 27, no. 9, p. 2522–2535. .

34. LINES, J. ir kt. Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. In *ACM Transactions on Knowledge Discovery from Data* . 2018. Vol. 12, no. 5, p. 1–35. .

35. FLYNN, M. ir kt. The Contract Random Interval Spectral Ensemble (c-RISE): The Effect of Contracting a Classifier on Accuracy. In PÉREZ GARCÍA, H. ir kt.Sud. *Hybrid Artificial Intelligent Systems* [interaktyvus]. Cham: Springer International Publishing, 2019. p. 381–392. [žiūrėta 2022-05-03]. ISBN 978-3-030-29858-6Prieiga per internetą: <http://link.springer.com/10.1007/978-3-030-29859-3_33>.

36. BAGNALL, A. ir kt. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. In *Data Mining and Knowledge Discovery* . 2017. Vol. 31, no. 3, p. 606–660. .

37. ALBAWI, S. ir kt. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* . 2017. p. 1–6. .

38. LECUN, Y. ir kt. Object Recognition with Gradient-Based Learning. In FORSYTH, D.A. ir kt. *Shape, Contour and Grouping in Computer Vision* [interaktyvus]. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. p. 319–345. [žiūrėta 2022-04-09]. ISBN 978-3-540-66722-3Prieiga per internetą: <http://link.springer.com/10.1007/3-540-46805-6_19>.

39. WOLD, S. ir kt. Principal component analysis. In *Chemometrics and Intelligent Laboratory Systems* . 1987. Vol. 2, no. 1–3, p. 37–52. .

40. MAATEN, L. Van der - HINTON, G. Visualizing Data using t-SNE. In *Journal of Machine Learning Research* . 2008. Vol. 9, no. 86, p. 2579–2605. .

41. MCINNES, L. ir kt. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. In [interaktyvus]. 2018. [žiūrėta 2022-04-09]. . Prieiga per internetą: <https://arxiv.org/abs/1802.03426>.

42. KEARNS, M. Thoughts on Hypothesis Boosting. In . 1988. .

43. FREUND, Y. - SCHAPIRE, R.E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *Journal of Computer and System Sciences* . 1997. Vol. 55, no. 1, p. 119–139. .

44. FRIEDMAN, J.H. Greedy function approximation: A gradient boosting machine. In *The Annals of Statistics* [interaktyvus]. 2001. Vol. 29, no. 5. [žiūrėta 2022-05-02]. . Prieiga per

internetą: <https://projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full>.

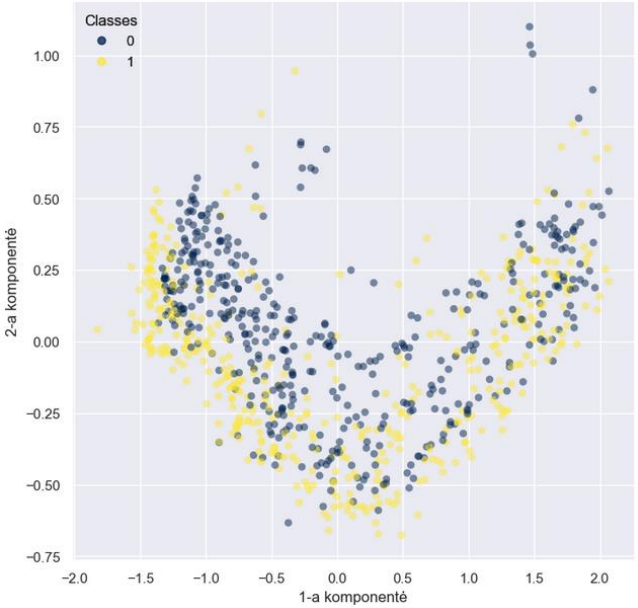
45. ZHANG, Z. ir kt. Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning. In *Scientific Reports* . 2018. Vol. 8, no. 1, p. 10628. .

46. MIENYE, I.D. ir kt. Prediction performance of improved decision tree-based algorithms: a review. In *Procedia Manufacturing* . 2019. Vol. 35, p. 698–703. .

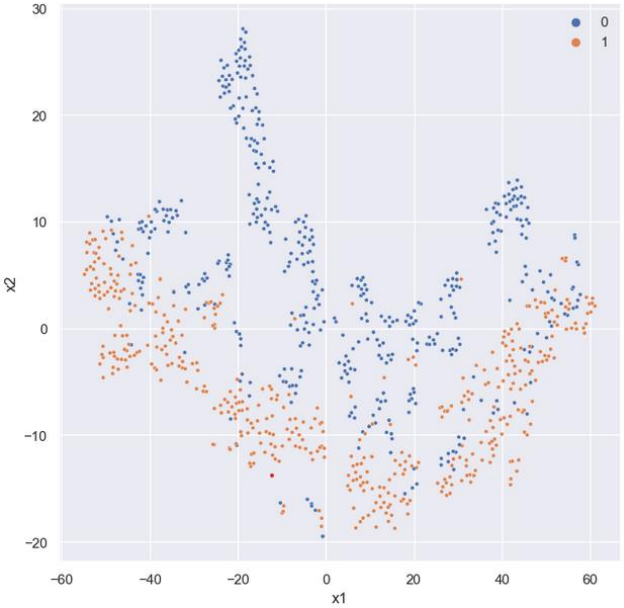
47. OLSZEWSKI, R.T. *Generalized feature extraction for structural pattern recognition in time-series data*. . [s.l.]: Carnegie Mellon University, 2001. ISBN 0-493-53871-2.

Priedai

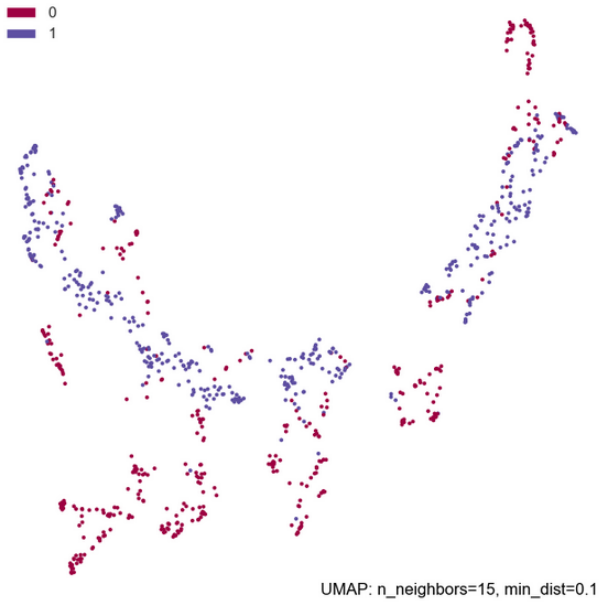
1 Priedas. ECGFiveDays duomenų konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija

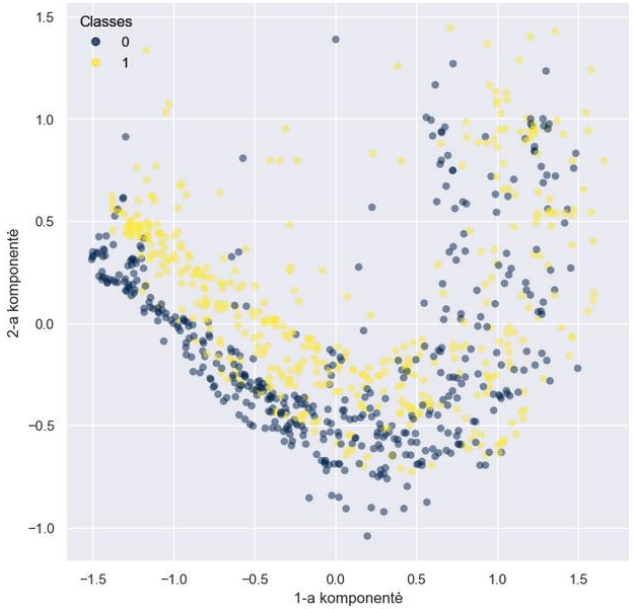


t-SNE vizualizacija

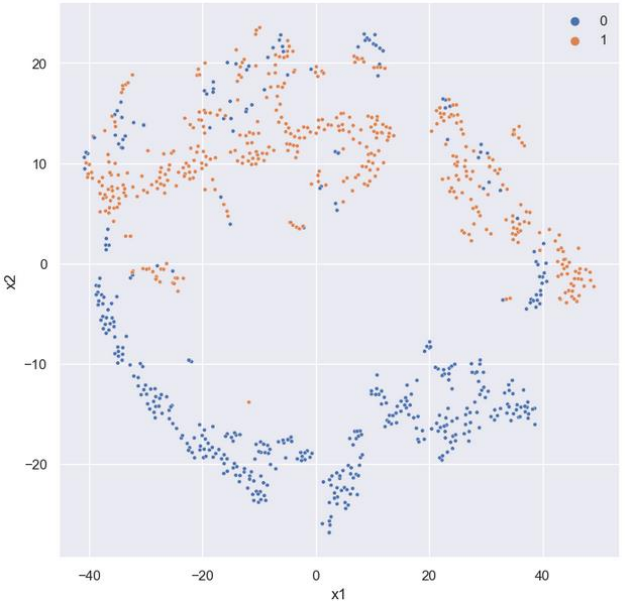


UMAP vizualizacija

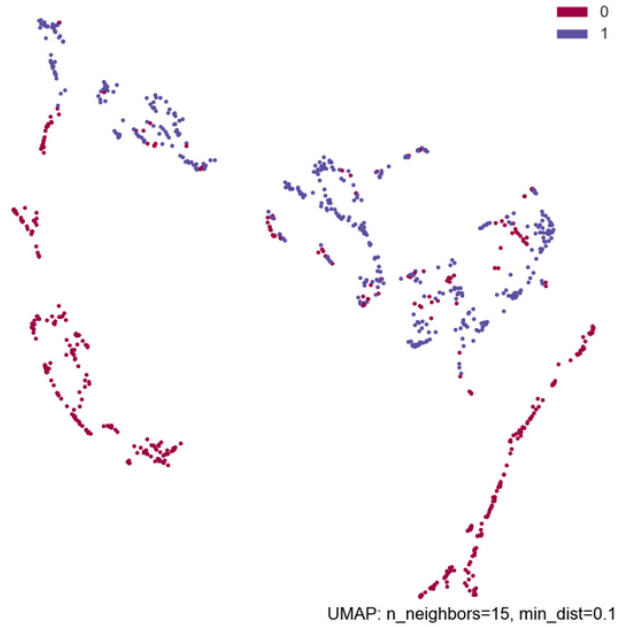
2 Priedas. ECGFiveDays duomenų variacinio konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija

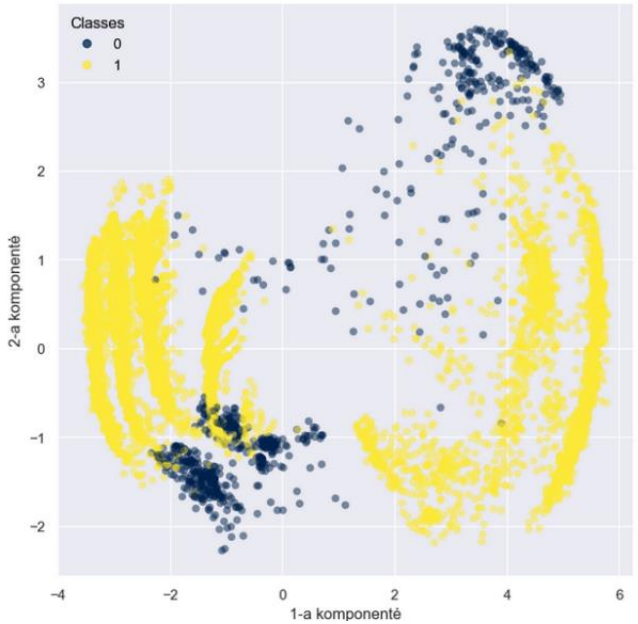


t-SNE vizualizacija

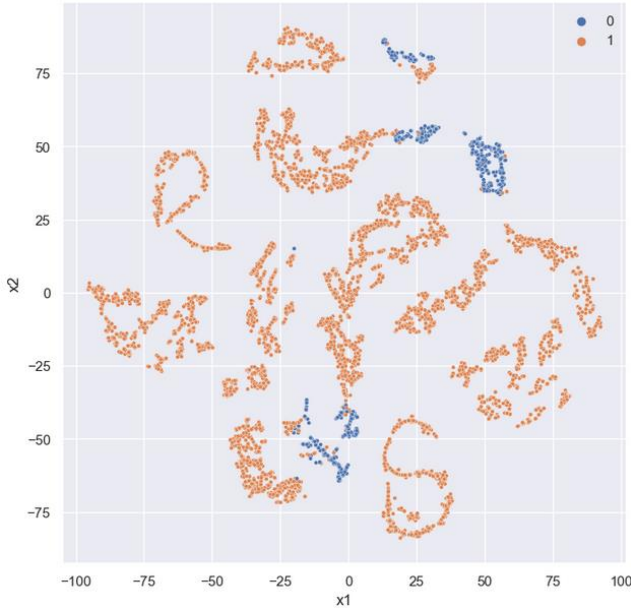


UMAP vizualizacija

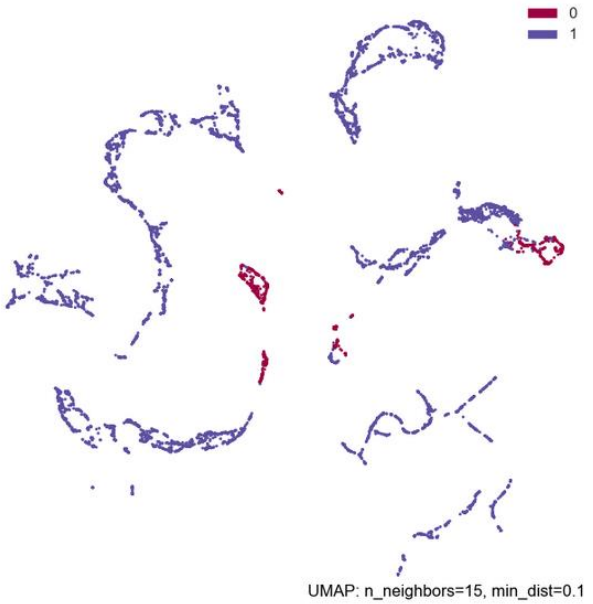
3 Priedas. Wafer duomenų konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija

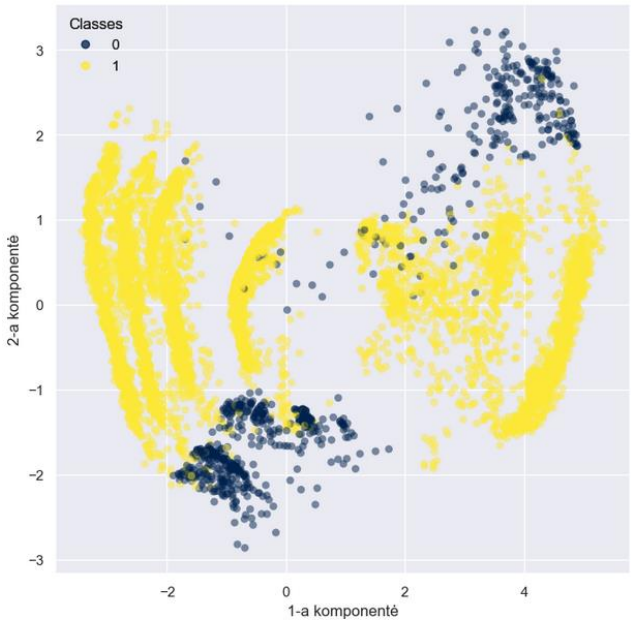


t-SNE vizualizacija

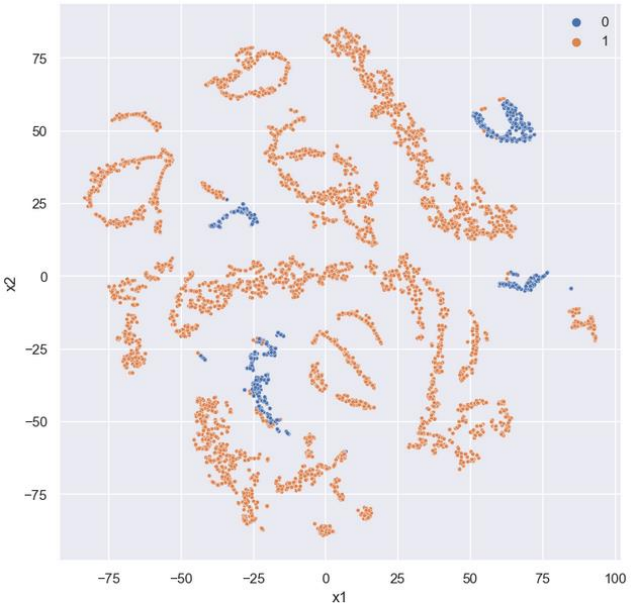


UMAP vizualizacija

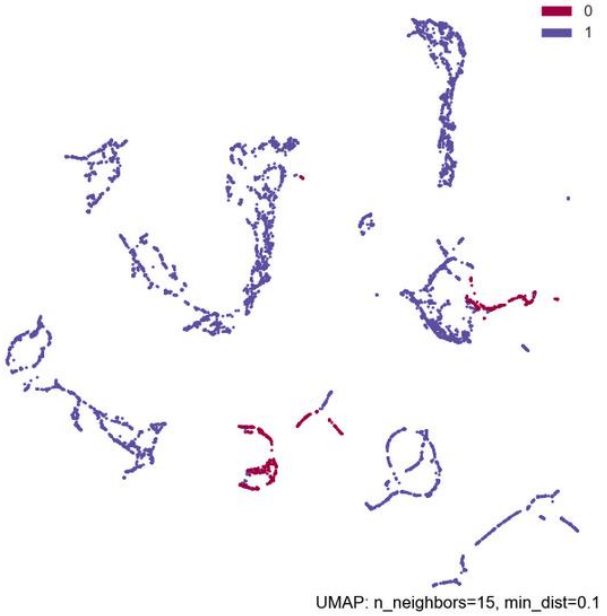
4 Priedas. Wafer duomenų variacinio konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija

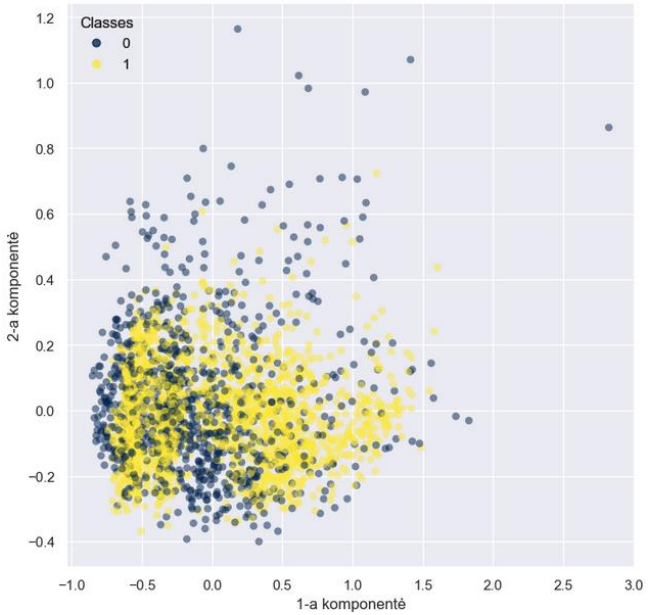


t-SNE vizualizacija

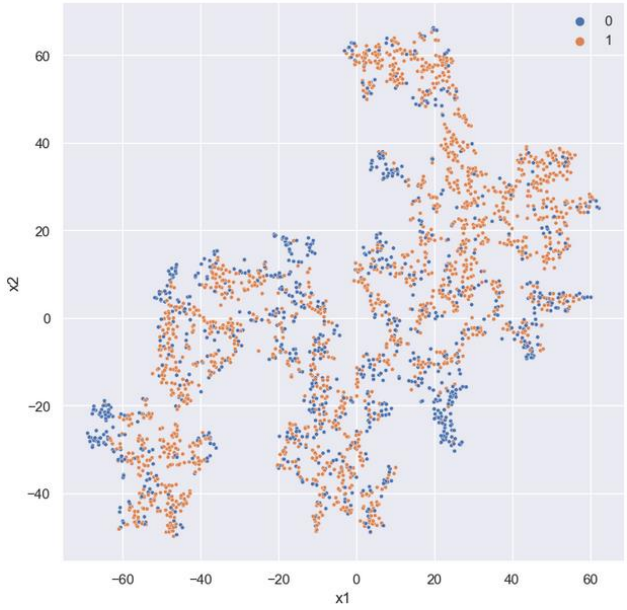


UMAP vizualizacija

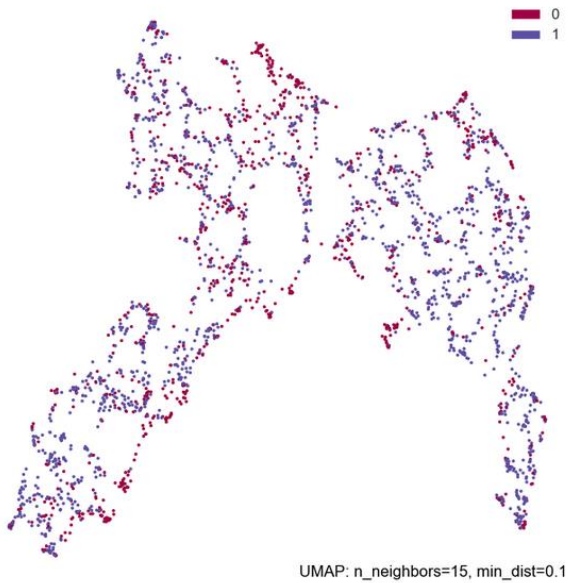
5 Priedas. PhalangesOutlineCorrect duomenų konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija

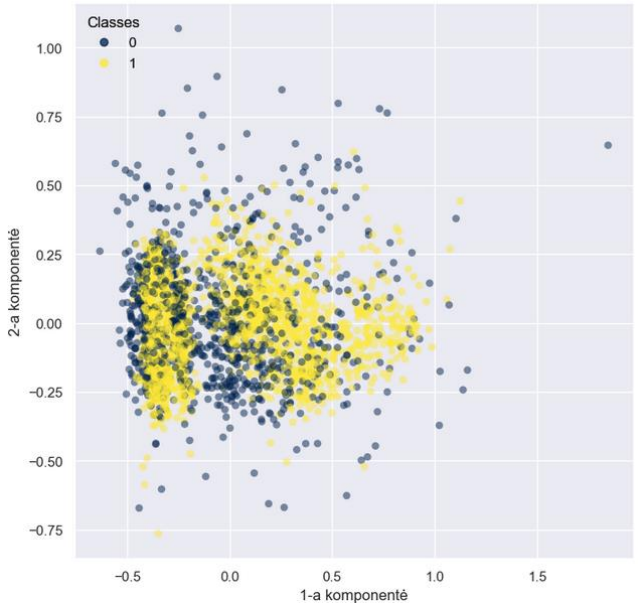


t-SNE vizualizacija

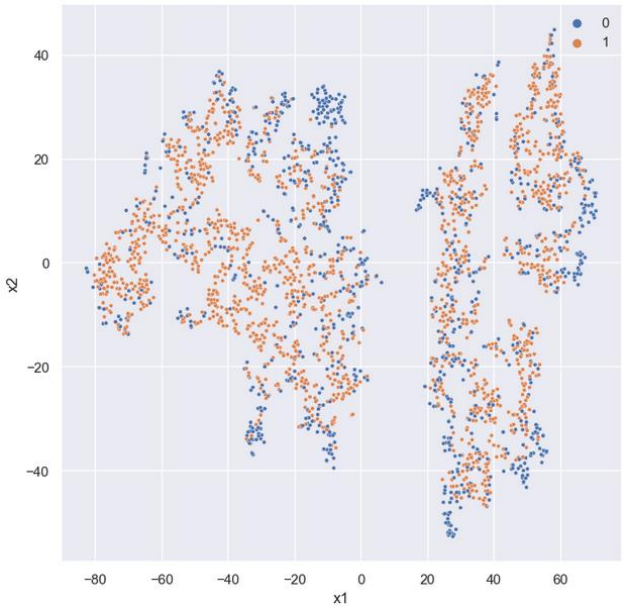


UMAP vizualizacija

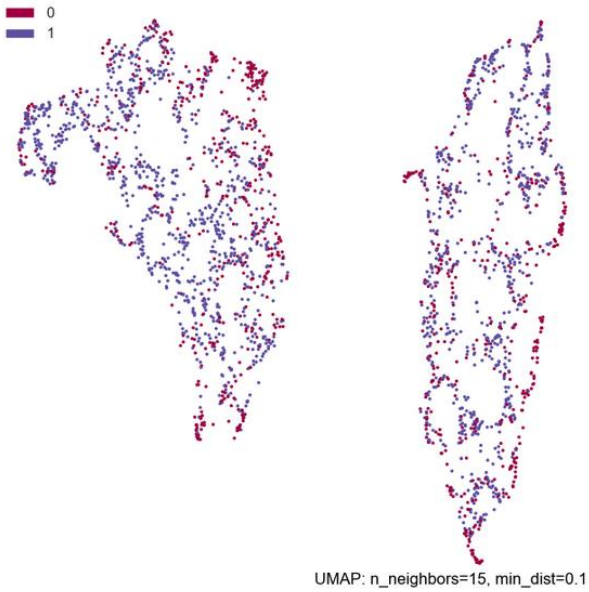
6 Priedas. PhalangesOutlineCorrect duomenų variacinio konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija

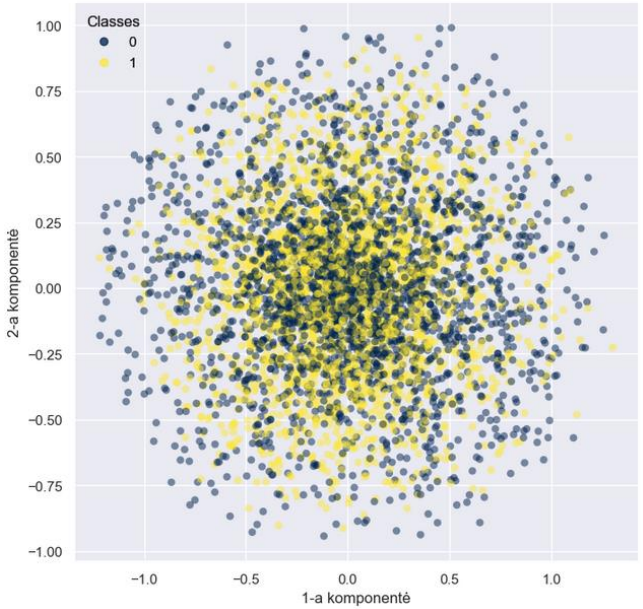


t-SNE vizualizacija

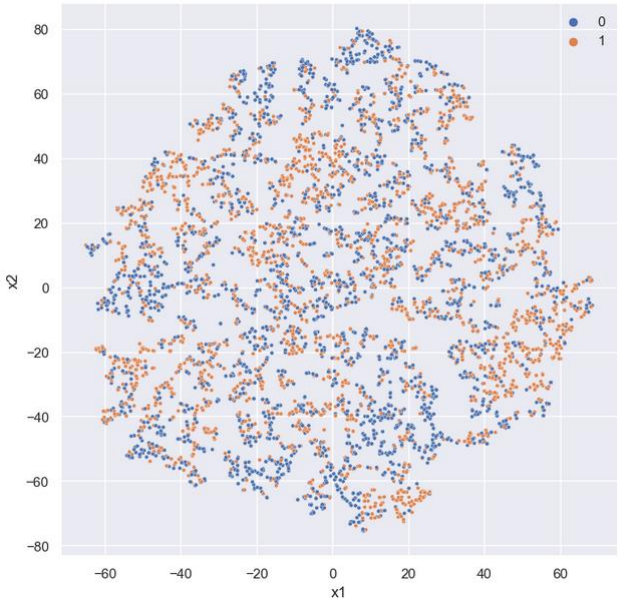


UMAP vizualizacija

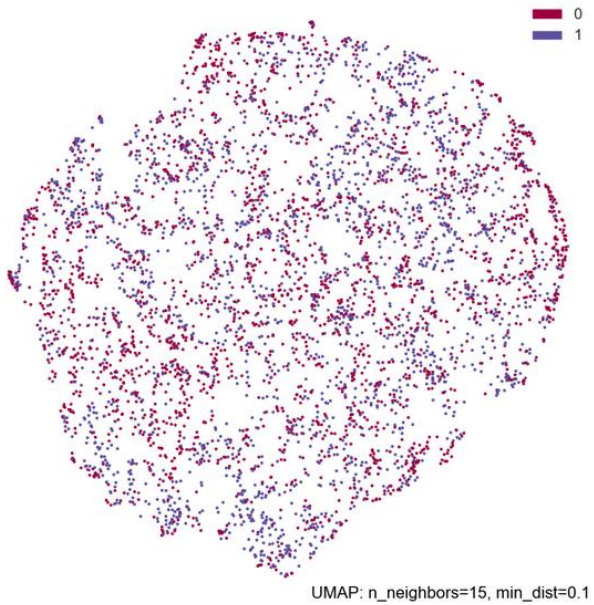
7 Priedas. FordA duomenų konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija

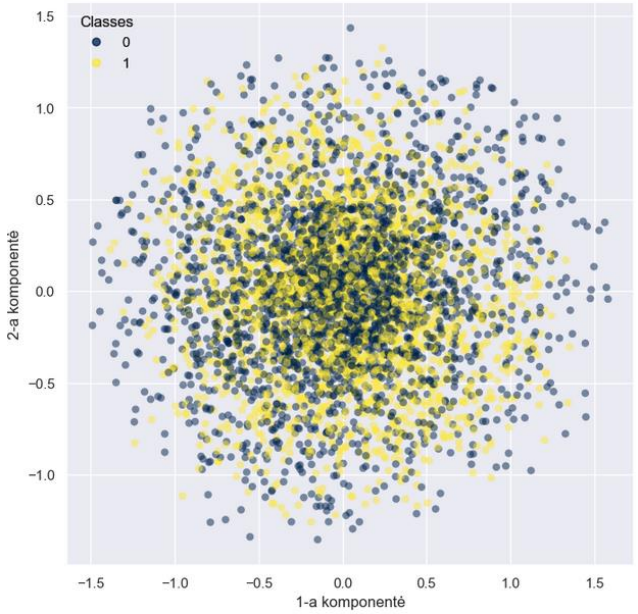


t-SNE vizualizacija

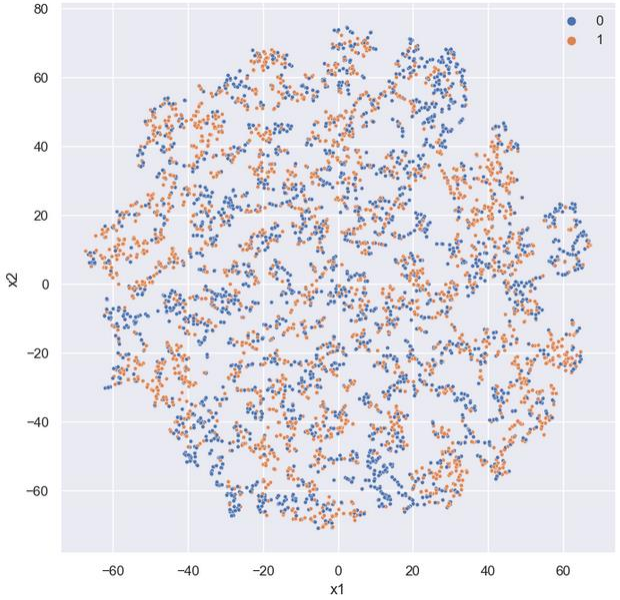


UMAP vizualizacija

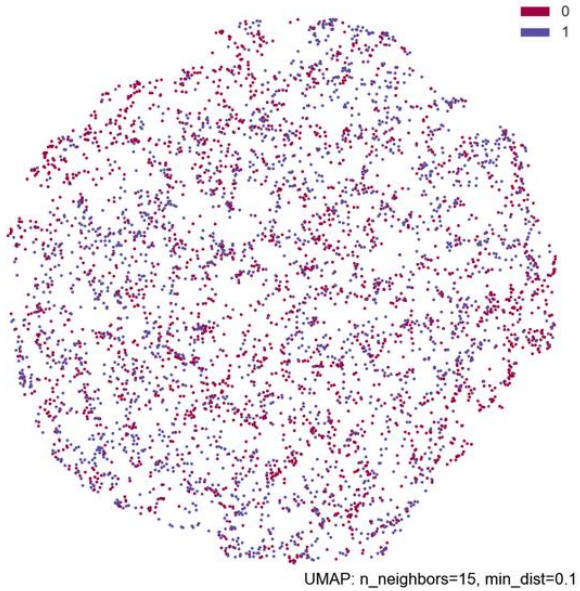
8 Priedas. FordA duomenų variacinio konvoliucinio autoenkoderio latentinės erdvės vizualizacijos



PCA vizualizacija



t-SNE vizualizacija



UMAP vizualizacija

9 Priedas. Visų modelių detekcijos tikslumo lentelė

Duomenų rinkinys	Variacinis konvoliucinis autoenkoderis	Konvoliucinis autoenkoderis	Catch22	Apjungtas
BeetleFly	0.650000	0.700000	0.750000	0.750000
BirdChicken	0.700000	0.750000	0.700000	0.650000
Chinatown	0.763848	0.912536	0.623907	0.853090
Coffee	0.714286	0.750000	0.964286	0.964286
Computers	0.488000	0.560000	0.716000	0.720000
DistalPhalanxOutlineCorrect	0.746377	0.728261	0.804348	0.786232
DodgerLoopGame	0.521739	0.717391	0.630435	0.673913
DodgerLoopWeekend	0.985507	0.920290	0.963768	0.920290
ECG200	0.770000	0.800000	0.770000	0.790000
ECGFiveDays	0.497096	0.571429	0.497096	0.691057
Earthquakes	0.589928	0.654676	0.705036	0.712230
FordA	0.615909	0.646212	0.906061	0.879545
FordB	0.572840	0.554321	0.771605	0.734568
FreezerRegularTrain	0.840702	0.885614	0.997193	0.997193
FreezerSmallTrain	0.735789	0.829474	0.997544	0.997544
GunPoint	0.753333	0.800000	0.906667	0.886667
GunPointAgeSpan	0.867089	0.867089	0.920886	0.943038
GunPointMaleVersusFemale	0.933544	0.943038	0.974684	0.939873
GunPointOldVersusYoung	0.888889	0.895238	1.000000	1.000000
Ham	0.533333	0.542857	0.590476	0.638095
HandOutlines	0.878378	0.802703	0.875676	0.875676
Herring	0.625000	0.562500	0.453125	0.500000
HouseTwenty	0.731092	0.647059	0.899160	0.873950
ItalyPowerDemand	0.755102	0.790087	0.848397	0.830904
Lightning2	0.655738	0.622951	0.721311	0.737705
MiddlePhalanxOutlineCorrect	0.738832	0.704467	0.752577	0.810997
MoteStrain	0.746006	0.781150	0.759585	0.778754
PhalangesOutlinesCorrect	0.705128	0.671329	0.797203	0.756410
PowerCons	0.855556	0.838889	0.927778	0.933333
ProximalPhalanxOutlineCorrect	0.756014	0.742268	0.828179	0.831615
SemgHandGenderCh2	0.835000	0.846667	0.898333	0.916667
ShapeletSim	0.500000	0.544444	1.000000	1.000000
SonyAIBORobotSurface1	0.727121	0.597338	0.901830	0.901830
SonyAIBORobotSurface2	0.779643	0.792235	0.882476	0.900315
Strawberry	0.781081	0.867568	0.921622	0.894595
ToeSegmentation1	0.618421	0.513158	0.771930	0.741228
ToeSegmentation2	0.576923	0.646154	0.823077	0.815385
TwoLeadECG	0.703248	0.729587	0.827041	0.676910
Wafer	0.988157	0.983777	0.994971	0.992862
Wine	0.666667	0.555556	0.592593	0.703704
WormsTwoClass	0.454545	0.519481	0.818182	0.818182
Yoga	0.729000	0.739000	0.735000	0.748667