TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Brandon Loorits

Patsientide enim levinud ravitrajektooride leidmine DTW meetodil

Bakalaureusetöö (9 EAP)

Juhendaja: Raivo Kolde, PhD
Kaasjuhendaja: Markus Haug, BSc

Tartu 2022

Patsientide enim levinud ravitrajektooride leidmine DTW meetodil

Lühikokkuvõte:

Lõputöö eesmärk on luua töövoog, mis aitab leida kasutajal enim levinud ravitrajektoorid teatud haigusega seotud patsientide kohordil. Välja pakutud töövoog koosneb 7 osast - andmete soovitud kujule viimine, sarnasusmaatriksi arvutamine dünaamilise ajadeformatsiooni meetodil, klasterdamine, siluetianalüüs, trajektooride korrigeerimine, tulemustrajektooride loomine ja visualiseerimine. Lõputöös pakutakse välja töövoog, mis potentsiaalselt aitab leida kasutajal enim levinud ravitrajektoorid automaatselt. Antud töövoog kasutab ravitrajektooride sarnasuse määramiseks dünaamilist ajadeformatsiooni, klasterdamise meetodina hierarhilist aglomeratiivset klasterdamist ning klasterdamise hindamiseks siluetianalüüsi. Töövoo tulemused visualiseeritakse kui ka prinditakse väljundina.

Võtmesõnad: Python, ravitrajektoorid, statistiline andmeanalüüs

CERCS: P170 Arvutiteadus, süsteemid

Finding the most common treatment trajectories of patients based on the DTW method

Abstract:

The goal of this thesis is to produce a workflow to help user to find most common treatment trajectories of some specific disease patients in cohort. Proposed workflow consists of 7 parts - converting the data to the required shape, calculating the similarity matrix by the method of dynamic time warping, clustering, silhouette analysis, correction of trajectories, creation and visualization of result trajectories. The proposed workflow in the thesis potentially helps the user to find the most common treatment trajectories automatically. This workflow uses dynamic time warping to determine the similarity of treatment trajectories, hierarchical agglomerative clustering as the clustering method, and silhouette analysis to evaluate clustering. Workflow results are visualized and printed as output as well.

Keywords: Python, treatment trajectories, statistical data analysis

CERCS: P170 Computer science, systems

Sisukord

Sissejuhatus

Aegridade analüüsi kasutatakse paljudes rakendustes nagu näiteks aktsiaturgude analüüs, eelarveanalüüs, majanduse ja müügi ennustamine. Lõputöös käsitleb autor aegridade analüüsi algoritmi, milleks on dünaamiline ajadeformatsioon. Dünaamiline ajadeformatsioon on laialt levinud aegridade analüüsi algoritm, mille abil on võimalik tuvastada üksteisele sarnaseid trajektoore. Peamiselt kasutatakse dünaamilist ajadeformatsiooni kõnetuvastuse, muusikalise töötluse ja aegridade klasterdamise eesmärgil. Kuivõrd lõputöö autori jaoks oli oluline leida patsientide enim levinud ravitrajektoorid, siis selle leidmiseks kasutas lõputöö autor dünaamilist ajadeformatsiooni kaugusmeetriku arvutamiseks klasterdamisalgoritmide jaoks.

Varasemalt on uuritud, kas dünaamilist ajadeformatsiooni on võimalik rakendada enim levinud haigustrajektooride leidmiseks. Koostatud töödes on kasutatud RHK 9-10 diagnooside koode ja need on olnud edukad. Samas pole uuritud, kas selle algoritmi abil on võimalik leida ka enim levinud ravitrajektoore. Kuivõrd haigustrajektoorid on osahulk ravitrajetooridest, püstitas lõputöö autor hüpoteesi: kuna dünaamilise ajadeformatsiooni abil on leitud enim levinud haigustrajektoorid, siis töötab dünaamiline ajadeformatsioon ka enim levinud ravitrajektooride leidmiseks.

Lõputöö eesmärgiks on pakkuda välja töövoog, mis aitaks kasutajal automaatselt leida teatud patsientide kohordi ravi andmestikul enim levinud ravitrajektoorid. Töövoo poolt väljastatud andmeid on võimalik kasutada statistika tegemiseks, kaasuvate haiguste leidmiseks, trajektoori läbiva patsiendi tuleviku prognoosimiseks ja modelleerimiseks.

Lõputöö koosneb kolmest peatükist. Esimeses peatükis antakse üldine ülevaade töös kasutatavatest meetoditest, mõistetest, andmestikust ja implementeeritavast töövoost. Teises peatükis kirjeldatakse täpsemalt töövoo implementeerimisel kasutatud peamisi meetodeid ning kuidas ja miks neid implementatsioonis kasutati. Kolmandas peatükis analüüsitakse välja pakutud töövoo tulemusi ja täpsusi nii klasterdamisel kui ka mudeli valikul siluetikoefitsendi alusel ning lõputöö autor pakub välja võimalusi edasiarenduseks. Lisadesse on lisatud GitHubi keskkonna repositoorium töövoo koodiga (vt Lisa I) ja litsents (vt Lisa II).

1. Taustainfo

Selles peatükis annab lõputöö autor ülevaate lõputöös kasutatud terminitest, mõistetest ning andmestikust. Lisaks kirjeldatakse, millised on töövoo erinevad osad ja mis on iga töövoo osa eesmärk.

1.1. Aegread

Järgnev selgitus põhineb Raul Kangro aine "Aegridade analüüs" esimese loengu slaididel [1]. Aegridasid ja nende analüüsi kasutatakse erinevates valdkondades - majandus, keskkond, tehnilised protsessid. Tegureid, mida hinnatakse aegridade alusel on näiteks aktsiate hinnad, maksulaekumised, sademete hulk, temperatuur, rahvaarv, sisend-väljund andmed, mõõdetud signaalid jne. Aegridade analüüsi peamiseks eesmärgiks on hetkeolukorra analüüs ja tuleviku ennustamine.

Aegrida on erinevatele ajavahemikele vastav väärtuste kogum, mis on ajas muutuv ja mille väärtuste suurus on juhuslikest teguritest sõltuv. Ajavahemikud, mida aegrea korral kasutatakse peavad olema järjestatud. Aegread võivad olla nii pideva ajaga kui ka diskreetse ajaga. Pideva ajaga aegrea korral toimuvad mõõtmised pidevalt. Diskreetse ajaga aegrea korral tehakse mõõtmisi teatud aja tagant [2]. Antud lõputöös käsitletakse aegridasid ravitrajektooridena ning ravitrajektooride puhul on tegemist diskreetse ajaga aegridadega.

Wilfredo Palma on enda raamatus selgitanud, aegrea definitsiooni [3]. Wilfredo Palma selgituse järgi tähistatakse aegrida kujul $\{y_t\}$, kus t tähistab mõõtmise hetke ja $t \in Z$, kuid praktikas on enamasti ainult lõplik piiratud hulk andmeid. Sellest asjaolust sõltuvalt tähistatakse aegridu ka kujul $\{y_1, y_2, ..., y_n\}$, kus n ajavahemiku pikkus.

1.2. Aegrea komponendid

Aegrida koosneb erinevatest komponentidest nagu nt y_1 või y_4 jne. Aegrea komponenti nimetatakse üksikmõõtmiseks ja see on aegrea liige või aegrea olek. Selleks, et aegridasid täpsemalt kirjeldada, tuleb seda analüüsida osade kaupa, kuna erinevatel osadel on erinev iseloom [4]. Aegrea juhusliku suuruse ja selle erinevate osade juhuslike suuruste muutumisel võib olla erinevaid põhjuseid. Põhjusteks võivad olla [2]:

- Ümbritseva keskkonna, energiaturu hindade tõus, sõja mõju tõttu või aktsiate hindade tõus/langus ettevõtte juhtkonna plaanide tõttu, nimetatakse trendiks;
- Muutused võivad toimuda ka vastavalt päevale, kuule või aastale, näiteks mänguasjade müük jõulukuul või Tartu linnarataste laenutamise sageduse tõus nädalavahetustel, taolisi muutusi nimetatakse sesoonseteks muutusteks;
- Maksulaekumisi, restoranide külastajate arvu jms muutusi võivad mõjutada tööpäevade arv nädalas (kas riiklikud pühad on nädalavahetusel või nädala sees). Selliseid muutusi nimetatakse jooksva aasta kalendrist sõltuvateks muutusteks;
- Muutused, mis toimuvad lühikeste perioodide vältel, nimetatakse ebaregulaarseteks muutusteks.

Majanduses toimuvad pikkade perioodide vältel erinevad aeglased tõusude ja languste tsüklid, mis on enamasti erineva pikkusega. Selliseid muutusi on keeruline, kui mitte võimatu andmete põhjal eristada. Taolisi aegrea osasid, mille aeglane muutus toimub pikaajaliselt, nimetatakse trend-tsükliteks [2]. Lisaks eristatakse aegridades ka juhuslikkuse komponenti ehk müra [4].

1.3. Ravitrajektoorid

K. Künnapuu jt. [5] on enda artiklis välja toonud, et elektrooniliste terviseandmete kasutamine uuringutes on kasvanud. See annab võimaluse koostada suuremahulisi uuringuid erinevatest haigustest. On mitmeid artikleid, kus on uuritud ajaliselt järjestatud haiguste esinemist ehk haigustrajektoore. Uuringutes kasutatud andmete põhjal on tehtud häid iseloomustusi, samuti pakuvad ka artikli autorid töös välja raamistiku, mis tuvastab ajaliselt järjestatud kliinilisi sündmusi.

Haigustrajektoore kirjeldatakse kui ajaliselt järjestatud haiguste esinemist, siis ravitrajektoore kirjeldatakse kui ajaliselt järjestatud ravi esinemisi. Bernice A. Pescosolido [6] on enda artiklis kirjeldanud patsiendi trajektoori kui sündmuste ja ravi muutuste esinemisi. Lisaks toob ta välja asjaolu, et patsiendi trajektoori eksisteerimiseks peab esinema patsiendil haigus. Lõputöös käsitletakse patsiendi trajektoori kui ravitrajektoori. Kuivõrd haigustrajektoorid ja ravitrajektoorid on sarnased, siis lõputöös püütakse rakendada enim levinud trajektooride leidmist ka ravitrajektooridele.

Lõputöö eesmärgiks on luua töövoog, mis leiab enim levinud ravitrajektoorid. Enim levinud ravitrajektooride leidmine annab võimaluse neid modelleerida, ravitrajektoori läbiva patsiendi edasise ravi prognoosida ja kasutada statistika jaoks.

Bernice A. Pescosolido [6] artiklis on välja toodud, et trajektooride loomine elektrooniliste terviseandmete põhjal ja nende uurimine on kasulik, sest see annab võimaluse muuta tervishoiusüsteemi paremaks, sealhulgas parandada patsientide ravi ja tervishoiu organisatsioonide ravisüsteemi. Näiteks kui patsiendile on korduvalt välja kirjutatud ravimeid või on patsient tihedalt külastanud arsti, siis ravitrajektoore teades on võimalik ära hoida inimese sattumine haiglasse.

1.4. Andmestik

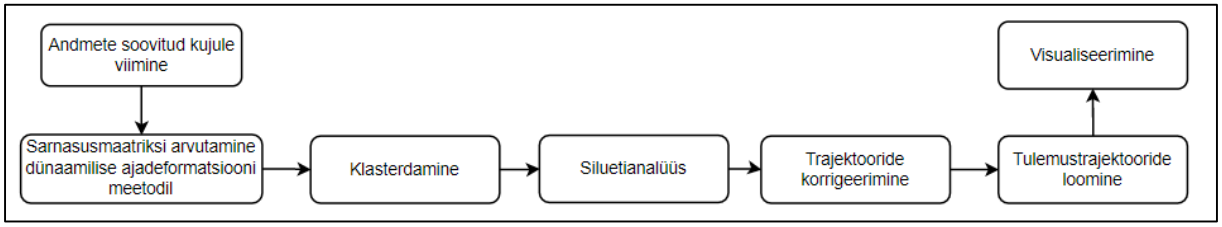
Andmed, mida kasutati on Eesti Haigekassa digitaalsed terviseandmed. Lõputöös kasutatud andmestik on koostatud pseudonümiseeritud astma patsientide andmete põhjal. Kõik andmed, mida kasutati töövoo koostamisel on anonümiseeritud ja neid ei ole võimalik kindlate inimestega seostada. Lõputöö eesmärgi saavutamiseks kasutatud andmestik, mis oli juhendaja poolt pakutud koosnes 147 466 reast andmetest 4 tunnusega. Tunnusteks olid patsiendi ID, seisund, seisundi algus- ja lõppkuupäev.

Algandmestikus olid 23 024 patsiendi andmed. Igal patsiendil oli erineval hulgal seisundeid, millest koostati hiljem ravitrajektoorid. Pikima ravitrajektoori pikkus oli 9 seisundit ja lühima pikkus 1 seisund. Andmestikust eemaldati patsiendid, kellele oli määratud ravimigrupp ainult ühe korra, sest lõputöö eesmärgi saavutamiseks olid vajalikud ainult need andmed, mille ravi seisundeid oli vähemalt 2.

Hiljem lisati andmestikule erineval hulgal ka samade tunnustega müra. Müra lisamise eesmärgiks on modelleerida tavapäraselt eksisteerivaid meditsiiniandmeid, kus sisaldub tihti sündmuseid, mis pole uuritava trajektooriga seotud. Müraks olid kõik ravimid, mis ei ole astma ravimid. Tulemuste analüüsimise eesmärgil kitsendati hiljem andmestik alamhulgaks ning võeti arvutustesse ainult 1000 ravitrajektoori dünaamilise ajadeformatsiooni sarnasusmaatriksi ajalise keerukuse tõttu [7]. Kitsendatud andmestik oli pakutud juhendaja poolt.

1.5. Töövoog

Lõputöös välja pakutud töövoo sisendiks on teatud haigusega seotud andmestik ning selle väljundiks eeldatavalt enim levinud ravitrajektoorid. Välja pakutud lahendus kasutab tulemuste leidmiseks pikka loogikaahelat, mille igal osal on alameesmärk lõppeesmärgi saavutamiseks. Loogikaahela 7 etappi nähtuvad joonisel 1. Lõppeesmärgiks on luua töövoog ravitrajektooride analüüsiks, mille abil oleks võimalik ravitrajektoori läbiva patsiendi edasist kulgu ennustada, modelleerida haigusega seotud patsientide ravitrajektoore ja kasutada statistikaks.



Joonis 1 Töövoog - 7 etapist koosnev loogikaahel.

Töövoo kasutamisel on võimalik sisestada erinevaid parameetreid tulemuste parandamiseks. Lisaks tuleb sisestada ka andmefaili nimi, sarnasusmaatriksi faili nimi ning kausta failitee. Andmefaili ja sarnasusmaatriksi failitüüp peab olema CSV formaadis. Etteantud kausta salvestatakse samuti kõik visualisatsioonid ja sarnasusmaatriks. Töövoo käivitamiseks on vajalik, et andmefail eksisteeriks etteantud kaustas. Tulemuste võrdlemiseks müraga ja ilma on vajalik esmalt töövoog käivitada algandmetega. Eeltoodu on vajalik, et töövoog saaks koostada algandmete alusel loodud esinemissageduste tabeli, mille põhjal saab valideerida välja pakutud töövoo tulemusi müraga andmete kasutamisel.

Töövoos kasutatud algoritmide keerukuse tõttu on loodud kaks python-faili, mis aitavad kiirendada tööaega. Esimene fail loob tslearn paketi abil dünaamilise ajadeformatsiooni algoritmi alusel trajektooride sarnasusmaatriksi, mille arvutamine on kõige ajakulukam. Teine fail rakendab sellel erinevaid statistilise andmeanalüüsi meetodeid. Uue andmestiku korral tuleb esmalt käivitada koodifail, mis loob sarnasuse maatriksi ning edasi saab kasutada ainult teist faili, mis kasutab eelnevalt loodud sarnasusmaatriksit ning rakendab sellel töövoogu.

2. Implementatsioon

Järgnevas peatükis kirjeldab lõputöö autor välja pakutava töövoo implentatsiooni ja selgitab täpsemalt selle osasid. Töövoog on kirjutatud Python programmeerimiskeeles ning selle implementeerimiseks on kasutatud mitmeid pakette. Standardteegist kasutati math, itertools, collections, statistics pakette. Standardteegi välised paketid, mida kasutati olid - pandas [8], numpy [9], matplotlib [10], tslearn [11], plotly [12], sklearn [13]. Töövoog on jaotatud 7 alamosaks, mille igal etapil on oma eesmärk.

Töövoog jaguneb:

1. andmete töötlemine;
2. DTW sarnasusmaatriksi koostamine;
3. klasterdamine;
4. mudeli valik;
5. trajektooride korrigeerimine;
6. tulemustrajektoori väljund;
7. visualiseerimine.

2.1. Andmete töötlemine

Tunnused ehk atribuudid on andmete puhul vahendid, mille alusel on võimalik erinevat tüüpi andmed kategoriseerida ja selle tunnuse alla koondada. Käesolevas lõputöös kasutatud algandmestiku suuruseks oli 147 466 rida 4 tunnusega. Analüüsi tulemuste jaoks tehti algandmestikule kitsendusi. Lõputöö eesmärgi tulemuste analüüsimiseks genereeris juhendaja algandmete põhjal uue andmestiku. Antud andmestikus tehti kitsendus, kus võeti viis enim levinud ravitrajektoori algandmestikust. Järgnevalt kirjeldab lõputöö autor analüüsi jaoks algandmetel genereeritud andmestikku. Algandmete põhjal genereeritud andmestikus oli 5018 rida 4 tunnusega - patsiendi ID, seisund, seisundi algus- ja lõppkuupäev.

Tunnused:

- patsiendi ID;
- seisund;
- seisundi alguskuupäev;
- seisundi lõppkuupäev.

Tunnus patsiendi ID on anonüümne ja seda ei ole võimalik inimestega seostada. Patsiendi ID alusel on võimalik patsiente ja nende ravitrajektoore eristada. Kõik patsiendi ID-d on arvulised väärtused. Patsientide unikaalsed ID-sid genereeritud andmestikus oli 1000.

Tunnus seisund kirjeldab hetke kui patsiendile määrati teatud astma haiguse raviks mõni ravim ravimigrupist. Astma ravimigruppe oli algandmestikus 11 ja selle põhjal genereeritud kitsendustega andmestikus 4, millele hiljem müra lisades tekitati juurde uusi ravimeid, mis pole astma raviga seotud.

Ravimigrupid genereeritud andmestikus:

1. LABA&ICS;
2. ICS;
3. SABA;
4. Systemic glucocorticoids.

Tunnus seisundi alguskuupäev annab infot, selle kohta, millal ravimigrupp patsiendile välja kirjutati. Seisundi alguskuupäev on kirjas formaadis AAAA-KK-PP, kus AAAA on aasta, KK on kuu ja PP on päev arvudes. Algandmestikul oli kõige varasem seisundi alguskuupäev 2012-01-01 ja kõige hilisem 2016-12-31 ehk algandmestikus oli andmeid viie aasta pikkuse perioodi kohta. Genereeritud andmestikus oli kõige varasem seisundi alguskuupäev 2012-04-30 ja kõige hilisem 2016-06-13 ehk uues andmestikus oli andmeid u nelja aasta pikkuse perioodi kohta.

Patsiendi_ID	Seisund	Seisundi_Algus
1	LABA&ICS	2012-03-01
1	ICS	2012-05-14
4	ICS	2015-09-11
4	SABA	2015-09-11
5	SABA	2012-07-02
5	LABA&ICS	2012-10-24
14	LABA	2012-05-21
14	Systemic glucocortisoids	2012-08-16
14	SABA	2013-04-10
30	LABA&ICS	2013-10-17

Joonis 2 Näide eeltöödeldud andmetest.

Algandmestikus esinesid kõik seisundid ühe ravitrajekoori piires ainult ühekordselt, kuid genereeritud andmestikus mitte. Andmestikule müra juurde lisamisel võisid seisundid esineda ravitrajektooris rohkem kui üks kord. Seisundi esinemisel rohkem kui üks kord, võisid samad

seisundid olla üksteise järel (nt LABA&ICS, LABA&ICS, SABA) või erinevates kohtades (nt LABA&ICS, SABA, LABA&ICS). Kuna enim levinud ravitrajektooride leidmiseks oli vaja leida seisundite muutused, siis mitmekordsed esinemised taandati ühekordseks.

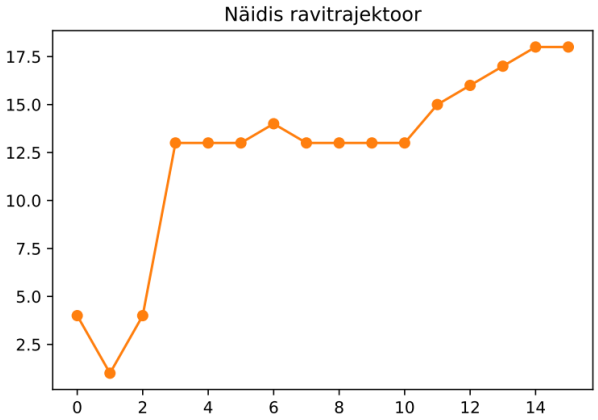
2.1.2. Ravitrajektooride loomine

Lõputöös kasutatud algandmestikus oli info 18 548 patsiendi kohta ja seega 18 548 ravitrajektoori, sest igale patsiendile vastas üks ravitrajektoor. Genereeritud andmestikus, mida kasutati tulemuste analüüsimiseks oli info 1000 patsiendi kohta ja seega 1000 ravitrajektoori. Pikim algandmestikul koostatud ravitrajektoor oli pikkusega 4 seisundit ja lühim 2 seisundit, samuti uues andmestikus.

Ravitrajektoor koostati võttes aluseks patsiendi ID ja lisades sellele raviseisundid, vastavalt ajateljele, kuidas raviseisundid sõltuvalt ravi alguskuupäevast muutusid. Ravitrajektoor salvestati Pythoni programmeerimiskeele poolt pakutavasse sõnastikku, mille võtmeks oli patsiendi ID ja mille väärtuseks oli teine sõnastik. Teise sõnastiku üheks võtmeks oli seisundid, mille väärtusteks olid ravimigrupi eelnevalt määratud arvud. Teiseks sõnastiku võtmeks olid alguskuupäevad, millele vastasid ravimigrupi väljakirjutamise alguskuupäevad eelnevalt kirjeldatud arvulises formaadis nagu nähtub joonisel 3. Trajektoore on graafikul võimalik seega visualiseerida joongraafikuna nagu nähtub joonisel 4.

```
10929: {'seisundi_algus': ['2012-04-27', '2014-03-18'], 'arv_väärtus': [4, 1]}
```

Joonis 3 Näide ravitrajektoorist sõnastikuna.



Joonis 4 Näide ravitrajektoorist joongraafikuna müratasemega 300.

2.1.3. Valideerimise tabel

Lisaks koostati algandmetest genereeritud andmete põhjal valideerimise tabel. Tabeli järgi valideeriti lõputöö autori poolt välja pakutud töövoo töökindlus ning täpsus enim levinud ravitrajektooride leidmiseks. Valideerimise tabel andis võimaluse hiljem leida tulemustrajektoori indeks, mis kirjeldas kui levinud trajektoor genereeritud andmetel oli. Selleks võeti kõik ravitrajektoorid ja koondati andmed ühte tabelisse, kus on selgelt näha kui palju vastava trajektooriga ravitrajektoore andmestikus leidus, ning kui levinud need andmestikus olid.

	Trajektoorid	Loendus
1	[LABA&ICS, ICS]	2029
2	[LABA&ICS, SABA]	1891
3	[LABA&ICS, Systemic glucocorticoids]	1214
4	[LABA&ICS, ICS, SABA]	515
5	[LABA&ICS, ICS, Systemic glucocorticoids]	420

Joonis 5 Viis enim levinud ravitrajektoori valideerimise tabelis algandmetest genereertiud andmete põhjal.

2.1.4. Seisundite sõnastik

Pythoni programmeerimiskeele poolt pakutava sõnastikuna koostas lõputöö autor ka seisundite sõnastiku, mille abil määrati igale vastavale ravimigrupile ehk võtmele diskreetne arvuline väärtus, et hiljem kasutada neid dünaamilise ajadeformatsiooni algoritmiga, kuna dünaamilist ajadeformatsiooni ei kasutata nominaalsete tunnustega. Pärast dünaamilise ajadeformatsiooni ja erinevate meetodite rakendamist leitud enim levinud trajektoorid teisendatakse numbriliselt kujult tagasi sõnalisele kujule seisundite sõnastiku abil.

```
{'Xanthines': 0, 'ICS': 1, 'SABA': 2, 'Systemic glucocorticoids': 3, 'LABA&ICS': 4, 'LABA': 5,
 'SAMA': 6, 'LTRA': 7, 'SABA&SAMA': 8, 'LAMA': 9, 'LABA&LAMA': 10}
```

Joonis 6 Seisundite sõnastik kõigi astma ravimigruppidega.

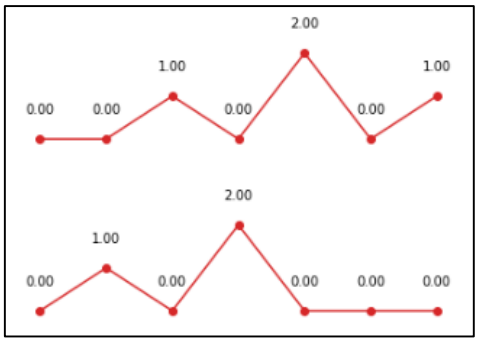
2.2. Dünaamiline ajadeformatsioon

Robert Giegerich on enda artiklis välja toonud, et tõenäoliselt kõige populaarsem programmeerimise meetod bioinformaatikas on just dünaamiline programmeerimine [15]. Lõputöö autor kasutab lõputöös dünaamilise programmeerimise ühte varianti, milleks on dünaamiline ajadeformatsioon. Dünaamilist ajadeformatsiooni kasutatakse peamiselt heli

tuvastamise jaoks, kuid autor kasutab lõputöös seda enim levinud ravitrajektooride leidmiseks. Dünaamiline ajadeformatsioon on laialdaselt kasutusel erinevates valdkondades nagu kõnetuvastus [16], bioinformaatika [17] ja aegridade klasterdamine [18]. Lõputöö autor kasutab töös dünaamilist ajadeformatsiooni klasterdamise eesmärgil.

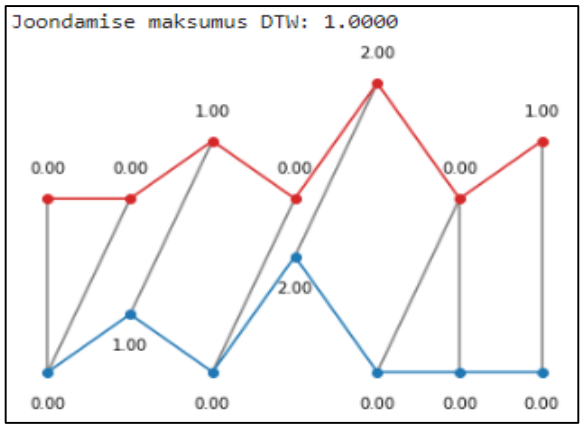
2.2.1 Mõiste

Dünaamiline ajadeformatsioon (ingl *Dynamic Time Warping*), edaspidi DTW, on algoritm, mis aitab efektiivselt leida kahe erineva aegrea sarnasused, olenemata sellest, millal need sündmused ajaliselt on toimunud [19]. Näiteks kui aegridade mingid osad on väga sarnased, aga need toimuvad aegrea ajatelje suhtes erinevatel ajahetkedel. Jooniselt 7 nähtuvad aegread on mingis osas identsed (aegridadel väärtuste jada 0, 1, 0, 2, 0), aga toimuvad erinevatel ajahetkedel.

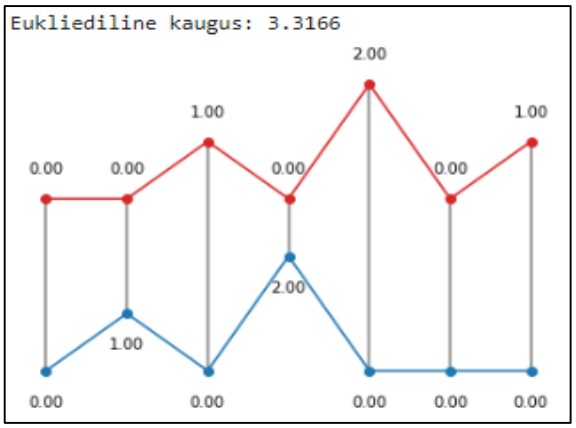


Joonis 7 Näide kahest erinevast ravitrajektoorist.

Joonis 9 kirjeldab, kuidas toimib dünaamiline ajadeformatsioon ning jooniselt 8 nähtub, kuidas toimis sarnasuse mõõtmine Eukleidilise kauguse mõõtmise korral. Nende kahe joonise abil on võimalik hästi aru saada, kuidas erinevad üksteisest Eukleidilise kauguse mõõdik ja dünaamilise ajadeformatsiooni sarnasuse mõõdik. Kui võrrelda kahte aegrida, siis näeme, et dünaamilise ajadeformatsiooni sarnasuse mõõdikut ei mõjuta nii palju absoluutne paigutus ajaskui Eukleidilist. Algoritmi eesmärgiks on leida kahel erineval aegreal ühiseid mustreid ja vastuseks anda mõõdik, mis näitab kui sarnased aegread, olenemata ajahetkest, millal need toimusid, üksteisega on. Lõputöös käsitletakse aegridadena ravitrajektoore.



Joonis 8 Näide dünaamilise ajadeformatsiooni sarnasuse mõõdiku leidmisest.



Joonis 9 Näide eukleidilse kauguse mõõdiku leidmisest.

2.2.2. Algoritm

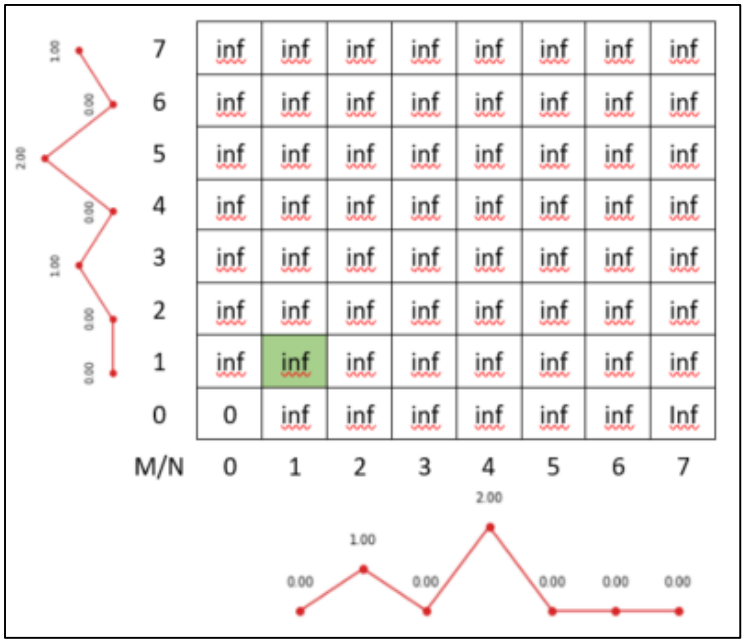
Dünaamilise ajadeformatsiooni algoritm töötab järgmiselt. Eeldame, et on antud kaks aegrida $X = (x_1, x_2, x_3,\ldots..,x_N)$, N kuulub naturaalarvude ja $Y = (y_1, y_2, y_3,\ldots..,y_M)$ M kuulub naturaalarvude hulka, kus N ja M on aegridade pikkused [19] (näite puhul nii N kui ka M pikkusega 7).

Algoritmi rakendamiseks on vajalikud eeldused [20]:

1. Piirtingimused – eeldus on vajalik, et optimaalne tee algaks ja lõppeks maatriksi diagonaalil asuvatel vastassuunalistel nurkadel $w_1 = (1,1)$ ja $w_K = (m, n)$.
2. Järjepidevus – eeldus piirab optimaalse tee lubatud sammude arvu naaber asukohtadeni (kaasa arvatud diagonaalselt). Kui antud $w_k = (a, b)$ siis $w_{k-1} = (a', b')$, kus $a - a' \leq 1$ ja $b - b' \leq 1$.
3. Monotoonsus – eeldus määrab, et maatriksi punktid oleksid vastavalt ajateljele monotoonselt paigutatud. Kui antud $w_k = (a, b)$, siis $w_{k-1} = (a', b')$, kus $a - a' \geq 0$ ja $b - b' \geq 0$.

Järgnev algoritmi selgitus põhineb tslearn paketi koodil [11]. Kui eeldused kehtivad, siis kahe aegrea joondamiseks luuakse M+1 x N+1 maatriks. Maatriksi iga element täidetakse väärtusega lõpmatus (∞) ja maatriksi algpunkti määratakse väärtus 0. Algoritmi rakendatakse alates reast n_1 ja veerust m_1 nagu nähtub jooniselt 10. Kõik ülejäänud maatriksi punktide väärtused

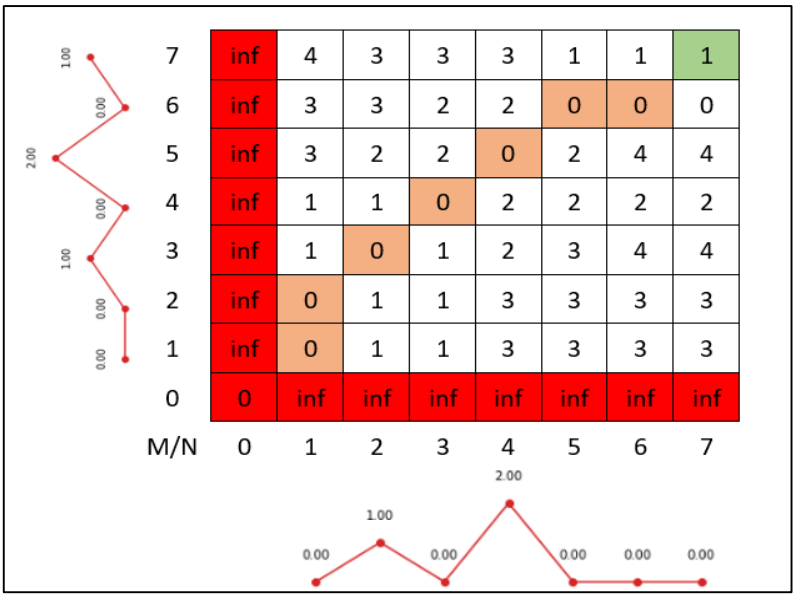
arvutatakse valemiga D(n, m) = d(x_n, y_m) + min{ D(n-1,m-1) , D(n-1,m), D(n,m-1) }, kus esimene pool valemist võrdub Eukleidilise kaugusega kahe punkti x_n ja x_m vahel, mida arvutatakse valemiga d(x_n, y_m) = $(x_n-y_m)^2$ ja teine pool valemist algoritmi poolt eelnevalt arvutatud sarnasuse mõõdikust.



Joonis 10 Sarnasusmaatriksi algseisund.

D(n-1, m-1) eeldab maatriksis liikumist diagonaalselt ehk mõlema aegrea vastavad punktid on vastavuses. D(n-1, m) eeldab maatriksis liikumist vertikaalselt ehk lisatakse punkt teisele trajektoorile, millega trajektoori võrreldakse. D(n, m-1) eeldab maatriksis liikumist horisontaalselt ehk eemaldatakse punkt teiselt trajektoorilt, millega trajektoori võrreldakse. Valemi teine pool valibki minimaalse väärtuse enda naaberkohtadest maatriksis ehk valitakse, kas punktid on vastavuses, eemaldatakse või lisatakse punkt [11].

Kui kogu maatriks on väärtustatud, siis eemaldatakse esimene veerg ja esimene rida. Selleks, et leida kahe aegrea sarnasus vaadatakse maatriksi (n, m) kohal olevat väärtust. Selleks, et leida kuidas kaks aegrida on omavahel vastavuses vaadatakse optimaalset teed, mis leitakse liikudes maatriksis kohalt (m, n) järgmistele kohtadele valides minimaalne väärtus naaberkohtadest (m-1, n-1 või m-1, n või m, n-1) [11]. Näidis aegridade sarnasusmaatriksi algseisundist on nähtavad joonisel 10 ja sarnasusmaatriksi lõppseisundist koos väärtustega, optimaalse teega ja sarnasuse mõõdikuga on nähtavad jooniselt 11.



Joonis 11 Sarnasusmaatriksi lõppseisund.

2.2.3. DTW rakendamine

Dünaamilise ajadeformatsiooni rakendamine ravitrajektooridele annab võimaluse eelnevalt kirjeldatud algoritmi alusel leida sarnasuse mõõdikud, mille arvutamisel ei ole tähtis nende täpne toimumise hetk, vaid nende järjestus ajaliselt. Otsitakse kahes trajektooris sarnaseid mustreid ning nende osasid. Seetõttu rakendatakse ka lõputöös ravitrajektooridele dünaamilise ajadeformatsiooni algoritmi, sest enim levinud ravitrajketooride leidmiseks ei ole vaja arvesse võtta nende seisundite muutumise täpset ajahetke vaid nende ajalist järjestust.

Lõputöös kasutas autor dünaamilise ajadeformatsiooni implementatsioonil Pythoni tarkvara paketti tslearn [11], mis pakub masinõppe tööriistu aegridade analüüsiks. 2017. aastal loodud pakett on koostatud scikit-learn, numpy ja scipy moodulite põhjal, mis on levinud masinõppe paketid. Täpsemalt kasutatakse lõputöös tslearni funktsiooni cdist_dtw, mis leiab erinevate pikkustega ravitrajektooride sarnasusmaatriksi eelnevas peatükis kirjeldatud algoritmi abil.

Tslearni paketi funktsiooni cdist_dtw sisendiks on list, mille sisuks on ravitrajektoorid listi kujul. Ravitrajektooride list koosneb ainult seisundite arvulistest väärtustest ajalises järjestuses. Funktsiooni sisendiks peavad olema listid, mis on sama dimensiooniga, kuid nende pikkus ei pea olema sama. Funktsioon tagastab sarnasusmaatriksi 2-dimensionaalse numpy järjendina, kus iga list kirjeldab ühte rida maatriksist.

Dünaamilise ajadeformatsiooni ajaline keerukus on O(nm), kus n ja m on trajektooride pikkused [19]. Kuna maatriksi arvutamine on kõige ajakulukam osa töövoost, siis leidis lõputöö autor, et tööaja vähendamiseks on kasulik eelnevalt arvutada sarnasusmaatriks ja salvestada see CSV failina. Töövoog loeb sarnasusmaatriksi failist sisse ja rakendab sellele erinevaid andmeanalüüsi statistilisi meetodeid. Dünaamilise ajadeformatsiooni rakendamine trajektooride sarnasuse mõõdiku leidmiseks ning sarnasusmaatriksi eelnevalt välja arvutamine annab ajakulu eelise, kui töövoogu proovida käivitada erinevate parameetritega. Sellisel juhul ei pea kasutaja ootama iga kord sarnasusmaatriksi arvutamise taga, vaid saab tulemusi analüüsida kiiremini.

2.3. Klasterdamine

Järgnev lõik põhineb Karl-Oskar Masingu aine "loomuliku keele töötlus" õppematerjalidel [22]. Masinõppes kasutatakse nii juhendatud kui ka juhendamata õppimist. Klasterdamine on juhendamata õppimise näide, mis liigitub tsentroidipõhiseks, tiheduspõhiseks, jaotuspõhiseks, hierarhiliseks, graafipõhiseks klasterdamiseks jne. Juhendamata õppimise eesmärgiks on leida andmetes korrapära. Juhendatud õppimise korral on olemas märgendus ja on võimalik hinnata lahenduse viga, kuid juhendamata õppimise korral mitte.

Järgnev selgitus põhineb T. Soni artiklil [23]. Klasterdamine on statistilises andmeanalüüsis levinud meetod, mida kasutatakse erinevates valdkondades nagu bioinformaatika, mustri tuvastamine, masinõpe jne. Klasterdamise eesmärgiks on liigitamata elementide struktureerimine, liigitades ühte klastrisse kõige sarnasemad elemendid ning mis oleksid teiste klastrite elementidest võimalikult erinevad. Liigitamine toimub teatud kauguse mõõdiku alusel.

2.3.1 Hierarhiline klasterdamine

Hierarhiline klasterdamine jaotub aglomeratiivseks ehk alt-üles liikuvaks ja jagatavaks ehk ülevalt-alla liikuvaks. Aglomeratiivse klasterdamise puhul liigutakse alt üles ja iga element märgistatakse eraldi klastrina ning uued klastrid saadakse nende ühendamisel uuteks suurematekst klastriteks. Jagatava klasterdamise puhul moodustatakse esmalt üks suur klaster ning seejärel jagatakse klaster väiksemateks klastriteks [23]. Kõige populaarsemaks hierarhilise klasterdamise meetodidest on aglomeratiivne, mida kasutatakse ka lõputöös [22].

Aglomeratiivse ehk alt-üles klasterdamise korral tuleb valida klastrite vaheliste kauguse funktsioon. Aglomeratiivse klastrite vahelise kauguse funktsioonid on [13]:

1. Ward - minimaliseerib kõigi klastrite vahelise kauguste ruutude summa, sarnaneb k-keskmiste klasterdamisele, kuid kasutab aglomeratiivse klasterdamise põhimõtteid;
2. Maksimaalne ehk täielik - minimaliseerib maksimaalse kauguse klastri elementide paaride vahel;
3. Keskmine - minimaliseerib kõigi klastri elementide vahelise keskmise kauguse;
4. Üksik - minimaliseerib klastri elementide lähimate paaride kauguse.

Hierarhilise klasterdamise visualiseerimiseks kasutatakse puudiagrammi (ingl *dendrogram*), mis kirjeldab, kuidas klastrid on tekkinud. Puu struktuuriga diagrammil kujutatakse klastreid tagurpidi U kujuliste ühendustega, kus tipp näitab klastrit ja alumised servad, milliste klastrite moodustamisel klaster saadud on. Servade pikkus näitab klastrite vahelisi kaugusi [13].

2.3.2 Hierarhilise klasterdamise rakendamine

Järgmiseks osaks töövoost on klasterdamise mudelite treenimine. Klasterdamise eesmärgiks on grupeerida sarnased elemendid ühte klastrisse, mis tähendab, et sarnasuse mõiste peab olema üheselt defineeritud. Töövoos kasutatakse sarnasuse mõõdikuna dünaamilise ajadeformatsiooni väljundina antud sarnasuse mõõdikut. Klasterdamiseks kasutab lõputöö autor sklearn paketi võimalusi. Täpsemalt funktsiooni AgglomerativeClustering, mis klasterdab ette antud trajektoorid dünaamilise ajadeformatsiooni sarnasuse mõõdiku alusel gruppidesse. Seega on eesmärgiks leida struktuur erinevate ravitrajektooride vahel.

Aglomeratiivset klasterdamist on lõputöös kasutatud, sest selle klasterdamise meetodil märgitakse algselt element eraldi klastrina ning ühendatakse suurematekс klastriteks kaugus funktsiooni alusel. Klasterdamise kauguse funktsioonina kasutati üksik funktsiooni, mis valib minimaalse kauguse klastri elementide paaride kaugustest. Klasterdamise funktsiooni treeniti eelnevalt arvutatud dünaamilise ajadeformatsiooni sarnasusmaatriksi alusel. Klasterdamine annab võimaluse leida sarnased ravitrajektoorid ning tänu sellele hiljem eemaldada müra. Valides igas klastris enim levinud trajektoorid ning jättes välja vähe levinud. Klasterdamise mudeleid loodi lõputöös mitmeid, mis salvestati listi, et hiljem leida valideerimise abil parima klasterdamise tulemuse andnud mudel.

2.4. Siluetianalüüs

Siluetianalüüsi puhul saab kasutada siluetikoefitsenti, mis annab hinnangu klastrisisese tugevuse ja klastrite eraldatuse kohta. Siluetianalüüsi osa on ka siluetidiagrammid. Siluetikoefitsent annab hinnangu klasterdamisele arvulise väärtusena, kuid siluetidiagramm visuaalse aimduse klasterdamise efektiivsuse kohta.

S. Aranganayagi jt. [24] on kirjeldanud siluetikoefitsenti enda artiklis. Eeltoodud artikli järgi on siluetikoefitsent (ingl *Silhouette score*) mõõdik, mida kasutatakse siluetianalüüsi puhul klasterdamise kvaliteedi hindamiseks. Mõõdik annab hinnangu, kui hästi on klassifitseeritud info klastrisiseselt ja kui eraldatud klastrid üksteisest on. Siluetikoefitsendi väärtused jäävad vahemikku [-1, 1].

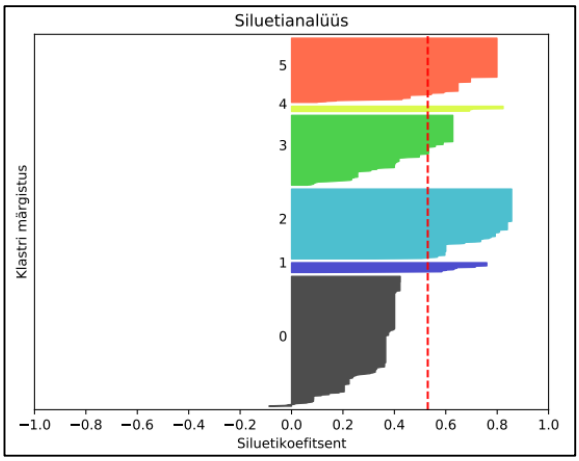
Järgnev siluetikoefitsendi selgitus põhineb Riivo Kikase bakalaureusetööl [25]. Siluetikoefitsent arvutatakse algoritmi abil, mis aitab leida klastris C_i iga elemendi x_i keskmise kauguse kõigist teistest elementidest selles klastris, mis valemis avaldub kujul a_i. Lisaks arvutatakse algoritmi abil elemendi x_i keskmine kaugus lähima klastri C_j, kuhu x_i ise ei kuulu, kõigi x_j elementide vahel. Valemi kohaselt arvutatakse iga elemendi siluetikoefitsent s_i keskmiste kauguste b_i ja a_i vahe ning jagatakse maksimaalse keskmise kaugusega a_i-st või b_i-st.

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

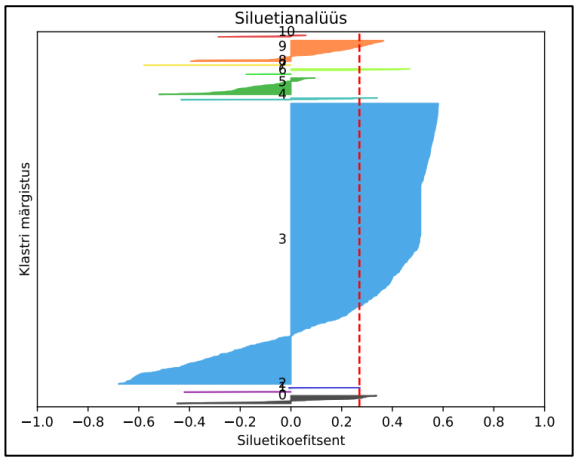
Kui on olukord, kus mõõdik annab väärtuse -1 lähedale, siis see tähendab, et klastrisisene keskmine kaugus a_i on väiksem kui lähima klastri keskmine kaugus b_i, mis tähendab, et klastrite elemendid on klastrite piiril või elemendile on antud vale klastri märgistus. Kui on olukord, kus mõõdik annab väärtuse 1, siis tähendab, et klasterdamine on toimunud hästi. Teisisõnu, mida lähemal 1-le on mõõdiku väärtus, seda paremini on klasterdamine toimunud ning saame olla kindlad, et klastrid on piisavalt hästi eristatud. Klastri keskmise siluetikoefitsendi saab leida elementide siluetikoefitsentide aritmeetilise keskmise leidmisega. Klasterdamise siluetikoefitsent on võimalik leida aritmeetilise keskmise leidmisega klastrite siluetikoefitsentidest.

Siluetidiagrammide analüüsi selgitus põhineb B. D. Pena jt. artiklil [26]. Siluetianalüüsi puhul saame kasutada ka siluetidiagramme, mis annavad aimu, kui võrdselt on klastrid moodustunud. Kui siluetikoefitsent annab arv väärtuse, siis diagrammide puhul on võimalik visuaalselt hinnata, kui võrdselt on klastrid tekkinud ning kui suured on siluetikoefitsendid iga elemendi puhul. Selleks, et hinnata klastrite õnnestumist tuleks hinnata visuaalselt:

- kui võrdselt klastrid on jaotunud;
- klastri siseselt oleks kõigil elementidel siluetikoefitsent positiivne;
- klastri elemendid oleks võimalikult sarnase siluetikoefitsendiga.

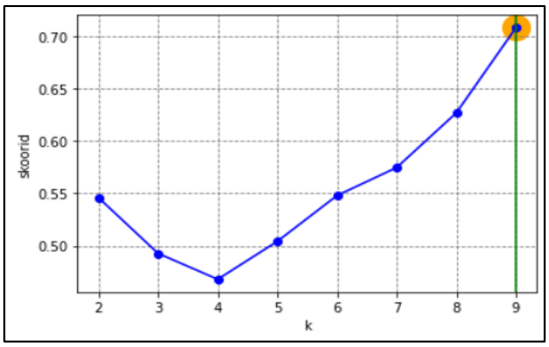


Joonis 13 Siluetidiagramm hea klasterdamise korral. Punane ristjoon näitab siluetikoefitsendi arv väärtust.



Joonis 12 Siluetidiagramm halva klasterdamise korral. Punane ristjoon näitab siluetikoefitsendi arv väärtust.

Mudeli valik välja pakutavas töövoos enim levinud ravitrajektooride leidmiseks tehakse siluetikoefitsendi väärtuse alusel. Selleks, et leida parim klastrite arv salvestatakse erinevate klastrite arvuga mudelid ühte listi. Listi läbi itereerimisel arvutatakse kõigi mudelite siluetikoefitsent ning valitakse nende seast parima siluetikoefitsendiga mudel. Lisaks visualiseeritakse kõigi mudelite tulemus ka graafikul, et oleks võimalik näha siluetikoefitsendi muutusi vastavalt klastrite arvule nagu nähtub jooniselt 14.



Joonis 14 Joondiagramm siluetikoefitsendi arvulise väärtuse muutumisest erinevate klastrite arvu korral.

2.5. Trajektooride korrigeerimine

Trajektooride korrigeerimine algab klastritest erindite eemaldamisega, mis eeldatavalt on klastrisse sattunud valesti klasterdamise teel. Igas klastris eemaldatakse trajektoorid, mis on väiksemad kui kasutaja poolt määratud osakaalu protsent. Osakaal leitakse trajektoori esinemiste arv klastris jagatud kogu klastri trajektooride arvuga. Trajektoorid salvestatakse pandas andme raamistikku ja seejärel kontrollitakse tingimust, et trajektooride osakaalu protsent oleks suurem kui määratud osakaalu protsent. Ühesugused trajektoorid grupeeritakse.

Lõputöö juhendaja soovitusel leiab lõputöö autor kõigi ühes klastris asuvate trajektooride ühisosa. Ühisosa tähendab, et leitakse kõik sellised seisundid, mis esinevad kõigis klastris olevates trajektoorides. Ühisosa leidmiseks leitakse esmalt klastris olevate trajektooride kõik unikaalsed seisundid, et kontrollida, kas need unikaalsed seisundid esinevad ka kõigis klastrisisestes trajektoorides. Kui seisund esineb kõigis klastri trajektoorides, siis lisatakse need ühisossa.

Ühisosa leidmine on etapp töövoos, mille tulemuse alusel on võimalik klastris olevaid trajektoore korrigeerida. Korrigeerimine tähendab, et kõigis klastrisse kuuluvates trajektoorides jäetakse alles ainult sellised seisundid, mis kuuluvad ka ühisossa. Eelduse kohaselt peaksid ühes klastris olema trajektoorid, kus on astma ravimigrupid ja nende vahel mürana teisi ravimeid. Kuna klastrid on moodustatud dünaamilise ajadeformatsiooni maatriksi alusel, siis eeldatavalt satuvad sellised trajektoorid ühte klastrisse. Ühisosa võtmine aitab kaotada astma ravimigruppide vahele jäävad müra seisundid ning jäävad alles ainult levinud ravitrajektoorid valideerimistabelist.

Saadud ühisosa tulemustrajektooril kontrollitakse, et seisundid ei esineks topelt, kuna need loetakse ühiseks seisundi muutuseks algses valideerimise tabelis. Topelt esinemine ei ole vajalik, sest vaadeldakse ainult seisundi muutusi ning samast seisundist samasse seisundisse liikumine ei ole seisundi muutus.

2.6. Tulemustrajektooride koostamine

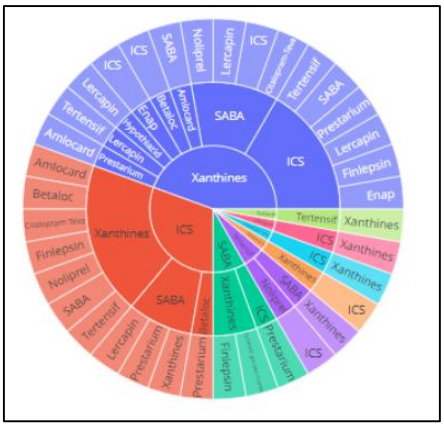
Järgmiseks etapiks lõputöö autori poolt välja pakutavas töövoos on tulemustrajektooride koostamine, kus esindava seisundi järjestusse valitakse trajektooridest tulenev suurima osakaaluga seisund. Suurima osakaaluga seisundi leidmiseks igas etapis lisatakse esmalt kõik korrigeeritud trajektoorid pandas paketi andmeraamistikku.

Seisundi osakaale võrreldakse vastavalt ajateljele. Näiteks kui ühisosa elementide arv oleks 2, siis otsitakse kõigi klastris olevate ravitrajektooride esimeste punktide seisundid ning loendatakse kõik seisundid kokku. Järgmisena kõigi trajektooride teiste punktide seisundid ning loendatakse need kokku.

Suurima osakaaluga seisund lisatakse tulemustrajektoori. Tulemustrajektoor salvestatakse listina tulemuste listi, mille järgi on võimalik hiljem leida kõik tulemustrajektoorid, mis koostati. Suurima osakaaluga seisundite valik implementeeriti lõputöö autori poolt juhendaja soovitusel.

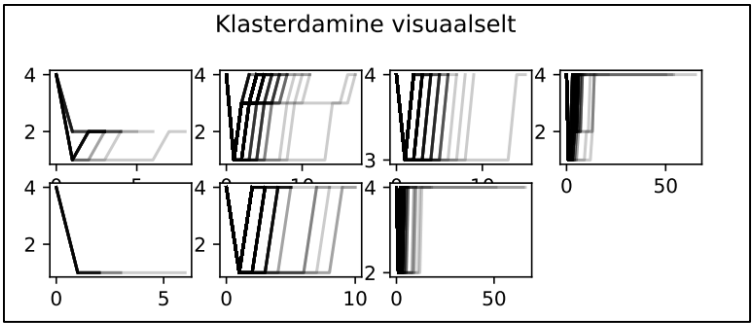
2.7. Tulemuste visualiseerimine

Kõigis ravitrajektoori punktides valitakse kõige suurema osakaaluga seisund tulemustrajektoori, kuid jäetakse alles ka kõik teised seisundid, et kasutada neid visualiseerimiseks. Visualiseerimine annab kasutajale parema arusaamise, klastris olevate ravitrajektooride jagunemisest ja nende osakaaludest vastavalt esimeses etapis, teises etapis jne nagu nähtub jooniselt 15.



Joonis 15 Sunburst diagram - kirjeldab ravitrajektooride jagunemist klastris, Kõige sisemises ringis ravitrajektooride esimene seisund, teises ringis teine seisund jne. Suurima osakaaluga - suurima alaga.

Tulemustrajektoori ei märgita klastrit kirjeldavas diagrammis, kuid prinditakse väljundina välja. Kõik visualisatsioonid salvestatakse HTML ning PDF failina etteantud kausta. Lisaks visualiseeritakse iga n klastrite arvu korral siluetidiagramm ja hierarhilise klasterdamise puudiagramm. Visualiseeritakse ka klastrite sisu trajektooridena, kus diagrammil on näha klastri enim esinenud ravitrajektoorid tumedama joonena. Mida tumedam joon, seda suurema esinemiste arvuga ravitrajektoor on, nagu nähtub jooniselt 16.



Joonis 16 Klastri siseste trajektooride jaotuvus.

Väljundis esitletakse kõiki tulemustrajektoore listi kujul, parimat klastrite arvu, mitu klastrit kõigist loodud klastritest andis õige trajektoori, mis esines valideerimistabelis ning mitu trajektoori valideerimistabelist protsentuaalselt leiti. Prinditakse välja ka õige trajektoori väljastanud klastrite arv ning vale trajektoori väljastanud klastrite arv. Kasutajal on võimalik käivitada töövoog ka selliselt, et prinditakse võrdluseks välja kõigi n klastrite tulemused.

3. Analüüs

Järgnevas peatükis kirjeldab lõputöö autor välja pakutud töövoo tulemusi ja pakub välja lahendusi tulemuste parandamiseks. Esimeses alajaotuses analüüsitakse erinevate parameetrite mõju töövoo tulemustele, sh müra sisalduse mõju andmestikus ja siluetikoefitsendi järgi mudeli valiku edukust. Teises alajaotuses pakub lõputöö autor välja edasiarenduse võimalusi.

3.1. Tulemused

Tulemuste analüüsimiseks tehti algandmetele kitsendusi ning võeti alamhulk algandmetest. Selleks, et tulemusi analüüsida teatud mõõdikute alusel, defineeriti kaks täpsuse mõõdikut. Esimene täpsuse mõõdik (täpsus1) näitab protsentuaalselt, mitu klastrit koostatud klastritest andis õige tulemuse. Teine täpsuse mõõdik (täpsus2) näitab protsentuaalselt, mitu trajektoori leiti valideerimise tabelist. Selleks, et tulemusi võrrelda on võimalik kasutajal sisestada erinevaid sisendparameetreid näiteks, kui suur peab olema klastris esineva trajektoori osakaal ja mitme klastrini tulemusi võrreldakse.

Töövoole lisati funktsioon, mis annab võimaluse kasutajal analüüsida tulemusi. Funktsiooni väljundiks on parim klastrite arv, leitud müra klastrite arv (valesti leitud trajektoor), õigesti leitud klastrite arv, tulemustrajektooride arv, mis esinevad valideerimistabelis, täpsus1 ja täpsus2. Lisaks koostas lõputöö autor ka analüüsi eesmärgil eraldi python-faili, mille alusel valideerida siluetikoefitsendi järgi klastrite arvu valikut. Failis olev töövoog ei tee valikut siluetikoefitsendi järgi, vaid kasutajal on võimalik leida tulemused soovitud klastrite arvu vahemikus.

3.1.1 Siluetikoefitsendi järgi klastrite arvu valik

Tulemuste analüüsi esimeseks alajaotuseks on tulemuste analüüs erineva müratasemega andmestiku puhul. Klastrite arvu määramiseks kasutatakse kõrgeima siluetkoefitsendi väärtuse andnud klastrite arvu, kuid klastrite arvu alampiiriks on määratud 5, sest kui otsime algtabelist viite enim levinud ravitrajektoori, peab olema klastrite arv vähemalt 5, et saaks tekkida 5 tulemustrajektoori. Tulemused tabelis on kirjutatud vastavalt etteantud osakaaludele 5/15/25/50. Kui trajektoori osakaal on alla etteantud osakaalu eemaldatakse see trajektoor klastrist. Tulemuste analüüsist saab järeldada, kuidas mõjutavad täpsusi müra sisaldus andmestikus ning kuidas mõjutab osakaalu erinevalt määramine tulemusi.

Tabel 1 Tulemuste tabel erinevate müratasemete ja erinevate osakaalude korral.

Müratasemе %	Parim klastrite arv	Täspsus1 %	Täpsus2 %
0	7/7/7/7	100/100/100/28.6	100/100/100/40
10	24/24/24/24	41.67/41.67/41.67/25.0	100/100/100/100
25	6/6/6/6	16.67/16.67/16.67/0	20/20/20/0
50	5/5/5/5	20/20/20/0	60/60/60/0
75	5/5/5/5	0/0/0/0	0/0/0/0
100	5/5/5/5	0/0/0/0	0/0/0/0
200	5/5/5/5	40/40/40/20	60/60/60/60
300	8/8/8/8	25/25/25/12,5	60/60/60/60

Tabel 1 järgi saab järeldada, et osakaalu muutmine ei mõjuta tulemusi nii palju. Ainult osakaalu 50 puhul muudab tulemusi kehvemaks, mis tähendab, et sellise osakaalu puhul ei teki klastrites nii palju esinevaid trajektoore, mille esinemise osakaal oleks suurem kui 50. Kuna osakaalu muutmine tulemusi suurel määral ei muuda võib järeldada, et klasterdamise algoritm töötab sarnaste trajektooride koondamisel piisavalt hästi ja ei teki erindeid suuremal hulgal. Tabelist 1 nähtub, et osakaalu tõstmine liiga kõrgeks halvendab tulemusi.

3.1.2 Tulemused n klastrite arvu korral

Tulemuste analüüsi teiseks jaotuseks on tulemuste võrdlemine erinevate n klastrite arvu korral. Selle analüüsi eesmärgiks on hinnata, kui hästi sobib valideerimiseks siluetikoefitsendi järgi klastrite arvu valimine. All olevad joonised kirjeldavad täpsusi erinevatel müratasemetel ja klastrite arv on vahemikus 20. Vahemiku alguseks on valitud klastrite arv 5. Kui valideerimise tabelis on 5 enim levinud ravitrajektoori ning iga klaster annab ühe tulemustrajektoori, mis on klastrit esindav trajektoor, siis on mõistlik alustada testimist klastrite arvuga 5. Vahemiku lõpuks on valitud 20 võrra suurem klastrite arv.

Joonis 17 Täpsuste tulemused osakaalu suurusega 15 ja erinevate klastrite arvu ning erinevate müratasemete korral.

Kõige paremini on klasterdamise algoritm töötanud enim levinud ravitrajektooride leidmisel, kui täpsus1 ja täpsus2 on võrdsed. Kui mõlemad on võrdsed ja mõlemad täpsused on lähedal 100%-le, siis on välja pakutud töövoog saanud väga hästi enim levinud trajektooride leidmisega hakkama. Näiteks müratasemе 0% juures, alates seitsmendast klastrist. Kui täpsus2 on väga

kõrge ja täpsus1 väga madal on olukord, kus klasterdamisel on tekkinud palju müraseid klastreid ja vähe täpseid klastreid, kuid kõikides leitud täpsetes klastrites on leitud enim levinud ravitrajektoorid, mis esindavad klastrit. Järelikult ei ole klasterdaja saanud klasterdamisega hästi hakkama.

Hall tulp diagrammil kirjeldab kui hästi on teatud klastri arvul enim levinud ravitrajektoorid üles leitud. Nagu diagrammidelt nähtub, müratasemega 0%, 10%, 75%, 100% ja 300% on töövoog leidnud üles kõik enim levinud trajektoorid valideerimise tabelist. Kehvem tulemus on müratasemе 25% ja 200% juures, kus täpsuseks on 80%, ja kõige kehvem müratasemе 50% juures, kus täpsuseks on 60%. Kuivõrd lõputöö eesmärgiks oli leida enim levinud ravitrajektoorid, siis võib järeldada, et töövoog on enamikel juhtudel leidnud üles kõik enim levinud ravitrajektoorid. Kõikidel juhtudel üle poolte trajektooridest.

Diagrammidest nähtub, et siluetikoefitsendi järgi mudeli valik ei tööta enam nii hästi kui hakatakse suuremal määral müra juurde lisama. Näiteks müratasemе 75% ja 100% juures on siluetikoefitsendi väärtus suurim viie klastri korral, kuid nagu diagrammidelt ja tabelist nähtub, siis sellise klastrite arvu korral ei leia töövoog ühtegi enim levinud ravitrajektoori. Samuti müratasemete 25%, 200% ja 300% siluetikoefitsendi väärtuse järgi mudeli valikut tehes on enim levinud trajektooride leidmise tulemused kehvemad kui etteantud klastrite arvu korral. Müratasemе 0% ja 10% leidis töövoog kõik enim levinud ravitrajektooridest, mis ühtivad ka siluetikoefitsendi väärtuse järgi mudeli valikut tehes.

3.2. Edasiarendus

Kuna sarnasusmaatriksi arvutamine vajab palju ressursse suurel andmehulgal, siis pakub lõputöö autor välja kasutada parema jõudlusega arvuteid või servereid. Võimaluseks oleks kasutada ka dünaamilise ajadeformatsiooni edasiarendusi. Lõputöö autor kasutas Tartu Ülikooli Rocket serverit arvutuste tegemiseks, kuid need ei andnud tulemusi täielikel andmetel 24h tunni jooksul vahemäluga 124 gigabaiti ja protsessorite arvuga 8. Seega tuli analüüsida tulemusi väiksematel andmehulkadel. Kui kasutada tervet andmestikku, siis aitaks see parandada töövoo tulemusi, kuna siis eristuks klastrid paremini suurema esinemiste arvu tõttu.

Lisaks pakub lõputöö autor välja edasiarendusena katsetada erinevaid klasterdamise meetodeid, kuna tulemused on tugevalt sõltuvuses klasterdamise efektiivsusest. Kui klasterdamise meetod

moodustab klastrid valesti või lisab ühte klastrisse erinevad trajektoorid, siis võib tekkida olukord, kus ilmnevad ühisosa võtmisel tihti tühihulgad, kuna klasterdamise alusel loodud klastrid sisaldavad liiga erinevaid ravitrajektoore.

Klasterdamise efektiivsuse hindamiseks oleks võimalik katsetada erinevaid valideerimise meetodeid. Kuna lõputöös kasutatud kõrgeima siluetikoefitsendi järgi mudeli valimine ei andnud piisavalt häid tulemusi, siis pakub lõputöö autor välja proovida erinevaid valideerimise võimalusi. Kuna valideerimise ja klasterdamise meetodeid on erinevaid, siis erinevatele klasterdajatele sobivad vastavad valideerimise meetodid. Nende testimisel ja analüüsimisel on võimalik leida parim klasterdamise ja valideerimise meetod.

Lõputöö eesmärgiks oli luua töövoog, mis aitaks leida teatud patsientide kohordi ravi andmestikul enim levinud ravitrajektoorid ilma mitmete spetsialistide kaasamiseta. Kuivõrd väljatöötatud töövoo kasutamine eeldab teatud määral teadmisi statistilistest meetoditest ja Python programmeerimise keele tundmist, siis pakub lõputöö autor välja edasiarendusena töövoo kasutajasõbralikumaks muutmise.

Kokkuvõte

Trajektooride loomine elektrooniliste terviseandmete põhjal ja nende uurimine on kasulik, sest see annab võimaluse muuta tervishoiusüsteemi paremaks, sealhulgas parandada patsientide ravi ja tervishoiu organisatsioonide ravisüsteemi [6]. Elektrooniliste terviseandmete kasutus uuringutes on kasvanud ning neid on edukalt kasutatud erinevate haigustrajektooride kirjeldamisel [5].

Kuivõrd haigustrajektoorid ja ravitrajektoorid on sarnased, siis pakkus lõputöö autor välja töövoo, mille abil leida enim levinud ravitrajektoorid dünaamilise ajadeformatsiooni meetodil. Selle alusel püstitati ka lõputöö hüpotees - kuna dünaamilise ajadeformatsiooni abil on leitud enim levinud haigustrajektoorid, siis töötab dünaamiline ajadeformatsioon ka enim levinud ravitrajektooride leidmiseks.

Lõputöös implementeeriti töövoog, mis koosnes 7 peamisest osast - andmete vajalikule kujule viimine, sarnasusmaatriksi arvutamine dünaamilise ajadeformatsiooni algoritmi alusel, klasterdamine, siluetianalüüs, trajektooride korrigeerimine, tulemustrajektooride koostamine ja visualiseerimine.

Kuivõrd dünaamilise ajadeformatsiooni sarnasusmaatriksi arvutamise ajaline keerukus on suur, siis kasutati lõputöös algandmete alamhulka tulemuste analüüsimiseks. Töövoo testimise eesmärgil prooviti ka paremate ressurssidega ülikooli serverit Rocket, mis aitaks kiirendada sarnasusmaatriksi arvutamist, kuid ei andnud piisavaid tulemusi. Seega kasutati algandmete põhjal genereeritud väiksemat andmehulka. Lõputöös pakuti tulemuste paremaks analüüsimise võimaluseks suurematel andmehulkadel kasutada paremate ressurssidega arvutit või dünaamilise ajadeformatsiooni edasiarendusi. Sellisel juhul eristuks klasterdamisel selgemini enim levinud trajektoorid.

Töövoog kasutab siluetikoefitsendi väärtust klastrite arvu valiku tegemiseks. Siluetikoefitsendi väärtuse järgi mudeli valimine andis väiksemate müratasemete korral soovitud tulemused, kuid suuremate müratasemete korral mitte. Seetõttu pakuti lõputöös välja edasiarendusena erinevate valideerimise ja klasterdamise meetodide testimine, mis aitaks tõhusamalt üles leida enim levinud ravitrajektoorid.

Kuivõrd töövoog ei leidnud enim levinud ravitrajektoore üles siluetikoefitsendi väärtuse järgi mudeli valikul kõrge müratasemе korral, leidis välja pakutud töövoog siiski üles enim levinud ravitrajektoorid eelnevalt sisestatud klastrite arvu korral. Näiteks leidis töövoog teatud klastrite arvu ja erinevate müratasemete korral vähemalt üle poolte enim levinud ravitrajektooridest ning enamikel juhtudel kõik enim levinud ravitrajektoorid. Saab öelda, et välja pakutud töövoog oli edukas enim levinud ravitrajektooride leidmisel kõrgemate müratasemete korral, kui ei kasutatatud siluetikoefitsendi järgi mudeli valikut.

Kasutatud kirjandus

[1] Kangro R. Aegridade analüüs. 2016. slaid 3. https://courses.ms.ut.ee/MTMS.01.023/2016_fall/uploads/Main/aegread_slides1.pdf (01.05.2022).

[2] Kangro R. Aegridade analüüs. Tartu. 2011, lk 6 https://kodu.ut.ee/~rkangro/aegread/2011/aegread.pdf (01.05.2022).

[3] Palma W. Time series analysis. John Wiley & Sons. 2016, p. 2.

[4] Tooding L.-M. Aegrea esmasanalüüs, 2020. https://samm.ut.ee/aegrea-esmasanal%C3%BC%C3%BCs (27.04.2022).

[5] Künnapuu K., Ioannou S., Ligi K., Kolde R., Laur S. jt. Trajectories: a framework for detecting temporal clinical event sequences from health data standardized to the Observational Medical Outcomes Partnership (OMOP) Common Data Model. *JAMIA Open*, 2022, Vol 5, issue 1. https://doi.org/10.1093/jamiaopen/ooac021 (27.04.2022).

[6] Pescosolido B. A. Patient trajectories. *The Wiley Blackwell Encyclopedia of Health, Illness, Behavior, and Society*. USA: John Wiley & Sons, 2014, pp. 1770-1777.

[7] Müller M. Information retrieval for music and motion. Springer, 2007, pp. 69–84; pp. 20–21.

[8] The Pandas Development Team. Pandas-dev/pandas: Pandas. 2020. https://doi.org/10.5281/zenodo.3509134 (01.05.2022).

[9] Harris C. R., K. Jarrod Millman K. J., Walt S. J. Array programming with NumPy. *Nature*. 2020, Vol 585, No. 7825, pp. 357–362. https://doi.org/10.1038/s41586-020-2649-2 (01.05.2022).

[10] Hunter J.D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. 2007, Vol 9, No. 3, pp. 90–95.

[11] Tavenard R., Faouzi J., Vandewiele G. jt. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*. 2020 , Vol 21, No. 118, pp. 1–6. http://jmlr.org/papers/v21/20-091.html (01.05.2022).

[12] Inc. P. T. Collaborative data science. 2015. https://plot.ly (01.05.2022).

[13] Pedregosa F., Varoquaux G., Gramfort A. jt. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, Vol 12, pp. 2825-2830.

[14] Virtanen P., Gommers R., Oliphant T. E. jt. Scipy: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, Vol 17, pp. 261-272.

[15] Giegerich R. A systematic approach to dynamic programming in bioinformatics. *Bioinformatics*. Germany: Faculty of Technology Bielefeld University, 2000, Vol 16, issue 8, pp. 665–677. https://doi.org/10.1093/bioinformatics/16.8.665 (01.05.2022).

[16] Amin T. B., Mahmood I. Speech Recognition using Dynamic Time Warping. *2008 2nd International Conference on Advances in Space Technologies*. IEEE, 2008.

[17] Aach J., Church G. M. Aligning gene expression time series with time warping algorithms. *Bioinformatics.* USA, 2001, Vol 17, issue 6, pp. 495-508.

[18] Debregeas A., Hebrail G. Interactive Interpretation of Kohonen Maps Applied to Curves. France: *KDD*, 1998.

[19] Senin P. Dynamic Time Warping Algorithm Review. Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA, 2008.

[20] Keogh E. J., Pazzani M. J. Derivative Dynamic Time Warping. *Proceedings of the 2001 SIAM international conference on data mining*. 2001, p. 3.

[21] Keogh E. J., Pazzani M. J. Scaling up dynamic time warping for datamining applications. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. USA: Association for Computing Machinery, 2000.

[22] Masing K.-O. Loomuliku keele töötlus Pythonis. 2015. https://courses.cs.ut.ee/MTAT.03.311/2016_fall/uploads/Main/Loeng9pdf (05.05.2022).

[23] Madhulatha T. S. An overview on Clustering Methods. *IOSR Journal of Engineering.* 2021, Vol 2(4) pp. 719-725.

[24] Aranganayagi S., Thangavel K. Clustering Categorical Data Using Silhouette Coefficient as a Relocating Measure. *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*. IEEE, 2007, pp. 13-17.

[25] Kikas, Riivo. 2009. "Veebisessioonide klasterdamine kasutaja käitumise alusel", TÜ arvutiteaduse instituudi bakalaureusetöö. https://kt.era.ee/supervision/Kikas2009.pdf

[26] Pena B. D., Blakely L., Reno M. J. Parameter Tuning Analysis for Phase Identification Algorithms in Distribution System Model Calibration. 2021, pp. 1-6.

Lisad

I. Githubi repositoorium

Lõputöös välja pakutud töövoo kood on kloonitud isiklikku repositooriumisse, millele pääseb ligi lingil:
https://github.com/BrandonLoorits/FindingMostCommonTreatmentTrajectoriesWithDTW

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Brandon Loorits,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Patsientide enim levinud ravitrajektooride leidmine DTW meetodil**, mille juhendajateks on **Raivo Kolde ja Markus Haug**, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Brandon Loorits
10.05.2022