Ohjaamaton anomalioiden tunnistus linja-autovuorojen ajotiedoista

Juhana Heino

Pro gradu -tutkielma
Syyskuu 2022

Tarkastajat:
FT Joni Virta
VTT Henri Nyberg

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Juhana Heino: Ohjaamaton anomalioiden tunnistus linja-autovuorojen ajotiedoista
Pro gradu -tutkielma, 50 s.
Tilastotiede
Syyskuu 2022

Työn tarkoituksena on tutustua ohjaamattomiin anomaliantunnistusmenetelmiin ensin kirjallisuudesta ja sitten käyttäen kerättyä soveltuvaa aineistoa. Aineisto kerätään Helsingin Seudun Liikenteen (HSL) MQTT-rajapinnasta ja aineisto koostuu linja-autolinjan numero 20 reitistä ja noin 2000 vuorosta.

Koska kerätty aineisto on raakadataa suoraan HSL:n API:sta, työhön oleellisena osana kuuluu myös aineiston käsittely sopivaan muotoon. Aineisto tiivistetään, esisuodatetaan ja kootaan soveltuvaksi rakenteeksi.

Menetelmät, joita sovelletaan ovat Z-score, Mahalanobiksen etäisyys, K-means, lähinaapurimenetelmä, hierarkkinen klusterointi, DBSCAN ja LOF. Kaikista käydään teoriaa soveltuvissa määrin läpi, jonka jälkeen jokaista käytetään työn aineistoon käytännössä. Näistä syntyy mallijoukko (ensemble), joka löytää aineistosta anomaliavuoroja.

Asiasanat: ohjaamattomat menetelmät, anomalioiden tunnistus, joukkoliikenne, mallijoukot

Omistettu äidille, isälle, Emmille ja Seelalle.

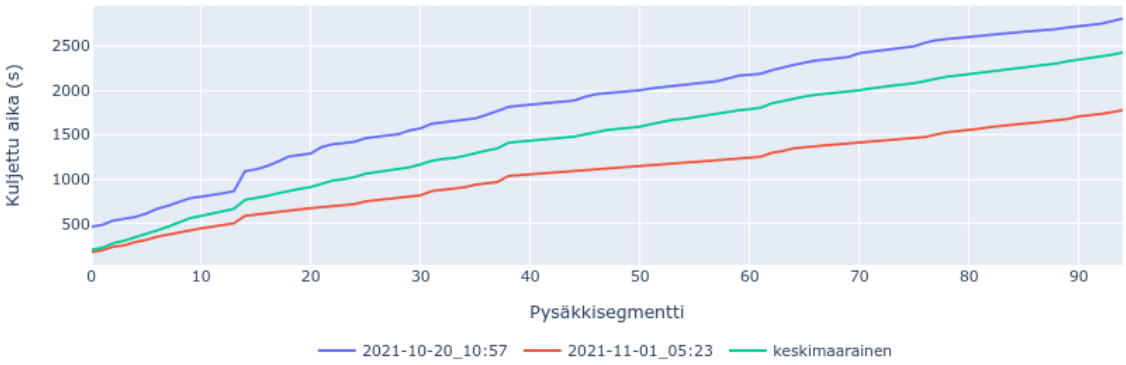
Sisällys

1 Johdanto

Työn tarkoitus on tehdä katsaus erilaisiin ohjaamattomiin (eng. unsupervised) anomalioiden tunnistusmenetelmiin sekä sitten soveltaa erilaisia menetelmiä kerättyyn aineistoon. Aineistona työssä käytetään linja-autojen aikasarjamuotoista dataa, joka sisältää mm. sijainti- ja nopeustietoa, mutta myös muuta metadataa, kuten vuoron tietoja. Aineisto on kerätty HSL:n MQTT-rajapinnasta ja sitä on noin yhden kuukauden verran. Aineiston koko on suodattamattomana noin 2400 ajettua vuoroa yhdeltä linjalta yhteen suuntaan. Seurattu linja on Helsingin bussilinja 20.

Työssä aiheena on anomaliat ja niiden tunnistaminen työn aineistosta. Yksi tapa määritellä anomalia on kutsua anomalioiksi aineiston havaintoja, jotka poikkeavat aineiston tyypillisestä käyttäytymisestä. Työssä tällaista tunnistamista kutsutaan anomalioiden tunnistamiseksi, vaikka samaa asiaa voisi kuvata monella muullakin nimellä, sillä termistö ei ole kovinkaan vakiintunut. Työn kontekstissa anomaliat ovat siis havaintoja, jotka jollain vuorolla poikkeavat muitten havaintojen massasta.

Työssä ei etsitä mitään tietystä syystä aiheutunutta anomaliaa, vaan pyritään käsittelemään aineistoa massana ja katsomaan, mitä se voisi kertoa. Esimerkiksi vaikka kuvan 1 tilanteessa, jossa nähdään kahden vuoron eteneminen reitillä verrattuna kaikkien vuorojen keskiarvoiseen etenemiseen. Molemmat vuorot poikkeavat keskiarvosta, toinen etenee nopeammin ja toinen hitaammin. Lisäksi toisessa vuorossa alkupäässä näkyy toista selvempi askelma. Kysymykseksi nouseekin, mikä näistä on anomalia vai onko kaikki? On paikallaan miettiä, onko normaalia nopeampi tai hitaampi kuitenkin tasaisen ajon vuoro anomalia. Vai onko esimerkiksi vain selvän viivästävän tapahtuman sisältävä vuoro anomalia? Työn päätavoite on siis tämänkaltaisten poikkeamien etsiminen useampiulotteisesti isosta massasta aineistoa.



Kuva 1: Kuvassa on kaksi todellista vuoroa sekä vertailuksi laskennallinen keskiarvoinen vuoro. Kuvassa näkyy vuoron eteneminen reitillä muodostetuista segmenteistä toiseen. Alimpana on nopea viivästyksetön vuoro, keskellä keskiarvoinen ja ylimpänä vuoro, joka on anomalia. Tässä anomalia on tapahtunut 7. ja 8. pysäkin välillä, pysäkkivälin loppupuolella.

Työn tavoite koostuu monesta osasta. Teoreettisessa osassa pääasia on anomalioiden tunnistamismenetelmien esittely ja ongelmakenttään tutustuminen. Koska anomaliatyyppejä sekä menetelmiä on lukemattomia, tässä työssä keskitytään oh-

jaamattomiin menetelmiin sekä menetelmiin, jotka mahdollisesti sopisivat käytössä olevaan aineistotyyppiin. Ohjaamattomat menetelmät valikoituivat tarkasteluun siksi, että käytössä oleva aineisto on luonteeltaan sellaista, jossa ei ole tietoa siitä, mikä havainto on poikkeava ja mikä ei. Tämä tekee myös ongelmasta hankalamman verrattuna siihen, jos aineisto sisältäisi tiedon poikkeamista. Tällöin voitaisiin käyttää ohjattuja menetelmiä ja uusista havainnoista anomalioiden tunnistaminen olisi helpompaa.

Empiirisessä osassa pääosassa on aineiston kerääminen ja sen käsittely analysoitavaan muotoon sekä lopulta aineiston analysointi esitettyjä menetelmiä hyväksi käyttäen. On siis tarkoitus kokeilla erilaisia aineistoon sopivia menetelmiä ja tunnistaa kerätystä linja-autoaineistosta anomalioita ohjaamattomilla menetelmillä, tutustua mahdollisiin visualisointitapoihin sekä pohtia syitä löydetyille anomalioille teoriaosan perusteella. Lisäksi myös selvittää osaavatko menetelmät löytää anomalioita aineistosta ja onko olemassa jotain sopivia menetelmiä tällaisten ohjaamattomien menetelmien vertailuun. Lopuksi otetaan vielä pieni katsanto siihen, millaisilla tavoilla saatuja tuloksia voi visualisoida.

Täysin samankaltaisia tutkimuksia ei kirjallisuudesta löytynyt, mutta osittain aihepiiriin osuvia tutkimuksia on tehty muun muassa syväoppimisen keinoin tutkimalla reittejä (Zhang et al., [1]) sekä kuljettajien ajokäyttäytymistä (Wang et al., [2]). Liikenneanomalioita on tutkittu myös, esimerkiksi joukkoistetuista aineistosta etsimällä (Kong et al. [3]) sekä taksien GPS-dataa hyväksi käyttäen (Kuang et al [4]). Kumpikaan ei kuitenkaan suoraan soveltanut tämän työn aihepiiriä, näissä etsittiin enemmän yleisen liikenteen anomalioita eikä juuri linja-autovuorojen anomalioita. Aineiston käsittelyyn liittyen löytyi myös tutkimus, jossa selvitettiin eri mittaisten reittien samanmittaistamista (Praczyk et al. [5]). Merenkulun lisäksi myös esimerkiksi lentokoneiden reittien anomalioiden tunnistamista on tutkittu rahtireittien ennustamisen näkökulmasta (Di Ciccio et al. [6]).

2 Anomalioiden tunnistaminen

2.1 Anomalian määritelmä

Anomalioiden tunnistaminen (eng. anomaly detection) on joukko menetelmiä, joilla pyritään tunnistamaan havaintoja, jotka poikkeavat muusta aineistosta [7]. Ne voidaan määritellä myös merkittäväksi poikkeamaksi tavanomaisesta tai normaalista, eli esimerkiksi keskiarvoisesta havainnosta [8]. Anomalioiden tunnistaminen on monin tavoin merkittävä kokoelma työkaluja, joille on käyttöä lähes alalla kuin alalla, melkein missä tahansa missä kerätään tai tuotetaan dataa. Niitä käytetään mm. petoksien estämisessä [9][10][11][12], tietoverkkojen tunkeutumisten tunnistamisessa [13][14] ja esimerkiksi vaikka tehtaissa laitteiden ja koneiden mekaanisten ongelmien tunnistamisessa [15][16] tai tuotannon laatuongelmien [17][18] esille tuonnissa. Valitsemalla sopiva menetelmä voidaan pyrkiä tunnistamaan poikkeavia havaintoja lähes millaisesta tahansa aineistosta.

Joskus anomalioiden tunnistamista voidaan kutsua myös poikkeavien havaintojen tunnistamiseksi (eng. outlier detection), harvinaisten tapahtumien tunnistamiseksi (eng. rare event detection) tai uutuuden tunnistamiseksi (eng. novelty detection). Monasti näillä kuitenkin tarkoitetaan samaa asiaa, eikä näiden eroista ole mitään yleistä sovittua määritelmää. Näistä harvinaisten tapahtumien tunnistaminen ja uutuuden tunnistus ovat selvemmin luonteeltaan erilaisia. Näistä harvinaisten tapahtumien tunnistamisessa koitetaan tunnistaa hyvin harvoin esiintyviä tapahtumia, joita ovat esimerkiksi tsunamit tai teollisuusonnettomuudet, joissa siis aineisto on hyvin epätasapainoinen ja toisaalta yleensä harvinaisella tapahtumalla on isot vaikutukset kokonaisuuteen. Uutuuden tunnistuksessa taas pyritään yleensä löytämään havaintoja, joita ei vielä koskaan ole havaittu. Huomionarvoista on kuitenkin se, että näillekin käytetyt menetelmät ovat ainakin osin päällekkäisiä anomalioiden tunnistamisessa käytettyjen kanssa.

Erityisesti yleisemmät termit anomalioiden tunnistaminen sekä poikkeavien havaintojen tunnistaminen esiintyvät usein synonyymeinä ja voidaankin sanoa, että lähes aina näillä tarkoitetaan samaa asiaa ja usein termien ero onkin enemmän filosofinen keskustelu. Nimiasiaan liittyvän epäselvyyden on esitetty aiheuttaneen jopa samanlaisia eri osapuolten tekemiä tutkimuksia, joissa on tutkittu samaa asiaa eri nimillä ja näin hidastettu ja vaikeutettu tutkimusta [19]. Standardointia ja asian yleistä sopimista ja määrittelyä onkin peräänkuulutettu.

Voidaan kuitenkin määritellä esimerkiksi, että anomalia on havainto, joka ei sovi jonkin prosessin odotettuun käyttäytymiseen, kun taas poikkeava havainto on sellainen, joka poikkeaa merkittävästi muista havainnoista ottamatta kantaa ne generoineeseen prosessiin [8]. Toisin sanoen, anomalia on suhteessa johonkin populaatiojakaumaan, kun taas poikkeava havainto on suhteessa otosjakaumaan. Toinen mahdollinen määritelmä on se, että anomalia on havainto, jonka on tuottanut jokin toinen prosessi, kun taas poikkeava havainto olisi saman prosessin tuottama havainto, joka vain sattuu olemaan kaukana keskiarvosta. Tässä työssä käytetään anomaliaa poikkeavan havainnon synonyyminä.

Anomalia voi esiintyä monin eri tavoin. Se voi olla yksittäinen havainto, joukko

havaintoja tai ryhmä peräkkäisiä havaintoja. Viimeksi mainittua kutsutaan kollektiiviseksi anomaliaksi (eng. collective anomaly) ja nämä johtuvat yleensä jostain poikkeaman aiheuttamasta tapahtumasta [7]. Ongelmakentän laajuuden takia on siis tärkeä panostaa riittävästi aikaa ja vaivaa oikeanlaisen menetelmän valitsemiseen. Tunnistus toimii parhaiten, kun siihen käytetään oikealle aineisto- ja anomaliatyypille tarkoitettua menetelmää, eli esimerkiksi yksiulotteiselle ja moniulotteiselle aineistolle tarvitaan omat menetelmänsä kuten vaikka aikasarjatyyppiselle, spatiaaliselle tai hierarkkisellekin aineistolle.

Aineiston poikkeamat ovat myös usein määrittelykysymys ja ainakin jonkin verran subjektiivisia, eli riippuu tilanteesta, minkälainen poikkeama tarvitaan siihen, että havainnon voidaan sanoa olevan poikkeava. Teknisellä tasolla tämän valinta on myös tyypillisesti menetelmien säätöparametri (eng. tuning parameter), jolla kontrolloidaan väärien ja oikeiden positiivisten määrää ja joskus tämä voi olla jopa tiedossa, kuten vaikka jossain teollisuusprosessissa, jossa tuotteen laatumäärittelyissä on ilmoitettu hyväksyttävä poikkeama. Normaalia aineistossa esiintyvää kohinaa ei tietenkään haluta todeta anomaliaksi vaan kiinnostuksen kohteena ovat enemmänkin nimenomaan merkittävästi poikkeavat havainnot [7]. Usein anomalioiden tunnistamisessa auttaa huomattavasti, kun aineiston analysoinnissa on käytössä osaamista aineiston kuvaamien ilmiöiden suhteen. Tällöin on helpompi esimerkiksi arvioida menetelmän toimintaa ja valita sopivampia menetelmiä, kun aineiston luonne on tiedossa.

Luonteeltaan anomalioiden tunnistaminen voi nopeasti vaikuttaa kahden luokan luokitteluongelmalta, johon sopisi ohjatussa tapauksessa mikä tahansa luokittelumenetelmä, kun taas ohjaamattomassa tapauksessa mikä tahansa klusterointimenetelmä. Aineistot ovat kuitenkin usein epätasapainoisia (luokkiin kuuluvia havaintoja on radikaalisti erisuuri määrä), jolloin esimerkiksi monet neuroverkot ja muut viimeisimmän tutkimuksen kehittämät koneoppimismenetelmät eivät useinkaan toimi luotettavasti ja tuottavat suuret määrät vääriä negatiivisia [8]. Tästä syystä anomalioiden tunnistamisessa tulisi käyttää tähän tarkoitukseen kehitettyjä ja hyväksi todettuja menetelmiä.

Kuten oikeastaan kaikissa muissakin tilastotieteen osa-alueissa, myös anomalioiden tunnistamisessa pätee sääntö, jonka mukaan yksinkertaisin riittävän suorituskykyinen malli on paras malli. Se on usein myös tulkittavuudeltaan paras, joka on myös anomalioiden tunnistamisessa erityisen tärkeä ominaisuus mallille.

Määritellään anomalia-termin lisäksi myös termi normaalihavainto, jolla tässä työssä tarkoitetaan havaintoa, joka ei ole anomalia.

2.2 Anomaliat linja-autolinjoilla

Vaikka työssä anomalioiden tunnistusmenetelmät ovatkin rajattu ohjaamattomiin menetelmiin, on silti hyvä pohtia minkälaisia anomalioita linja-autoihin liittyen voisi olla olemassa ja mitä niistä linja-autojen paikkatietodatasta voitaisiin löytää.

Paikkatietodatan perusteella on tietysti helppo nähdä aikataulussa pysyminen, eli kuinka paljon linja-auto on myöhässä tai edellä aikataulua missäkin reittipistees-

sä ja tämä lieneekin ensimmäiseksi mieleen tuleva ja myös tärkein linja-autoihin liittyvä anomalia. Käytössä oleva aineisto antaa myös lasketun myöhässäoloarvon jokaiselle datapisteelle, joten tämän suhteen myös ohjattuun oppimiseen perustuvat menetelmät voisivat olla mahdollisia. Paikkatietodatan perusteella voisi myös etsiä anomalioita normaalia suuremmista kiihtyvyyksistä tai jarrutuksista, jotka voisivat kertoa poikkeustilanteista. Myös vaikka harhaanajon tai kiertoreitin aiheuttama poikkeava sijainti (kuten kuvassa 2) olisi varsin helppo tunnistaa anomaliaksi, kuten myös poikkeavat nopeudet, esimerkiksi tien suurinta sallittua nopeutta suuremmat nopeudet tai selvät alinopeudet reitin osalla, jossa ei pitäisi olla pysähdyksiä.

Kuva 2: Selkeä esimerkki sijaintiin liittyvästä anomaliasta. Oikeanpuoleisen reitin loppuosa ei ole kiertänyt ollenkaan vastaavaa lenkkiä, joka reitille kuuluu normaalisti. Tämänkaltaiset anomaliat suodatettiin aineistosta pois jo ennen varsinaista anomaliantunnistusta, sillä nämä on helppo tunnistaa muutenkin.

Myös ajon taloudellisuuteen voitaisiin etsiä apuja anomalioiden tunnistamisella, esimerkiksi isot tai hyviksi todettujen kiihtyvyyksien väliltä poikkeavat arvot voisivat olla merkki epätaloudellisesta ajotavasta.

Monet näistä ovat tietysti suoraan datasta näkyviä ja analysoitavissa jopa yksinkertaisilla if-lausekkeilla, varsinkin kun tiedetään tarkalleen mitä haetaan, eli löytämiseen ei välttämättä tilastollisia menetelmiä tarvittaisi. Kiinnostava aspekti kuitenkin on myös se, tapahtuuko samassa kohdassa esimerkiksi aina sama ylinopeus vai ovatko jotkut vuorot pahempia tämän suhteen. Eli isosta aineistosta voidaan muodostaa erilaisille suureille jakaumia ja verrata yksittäisiä mittauksia niihin ja tehdä sitten päätelmiä anomalioista.

Linja-autoihin liittyviä anomalioita voidaan ajatella myös hieman yleisemmin, ilman suoraa yhteyttä paikkatietoaineiston mahdollistamiin kysymyksiin. Anomalioita voitaisiin koittaa etsiä matkustajien määrissä eri kohdissa, teknisten ongelmien ennakoivassa paikantamisessa tai esimerkiksi liikennevaloetuuksien ongelmien etsimisessä. Vieläkin laajemmin katsottuna voisi olla mahdollista myös etsiä parempia ajotapoja vaikka erilaisiin sääoloihin, analysoimalla vain huonossa säässä (esim. tiheä lumisade) ajettuja reittejä ja etsimällä näistä hyviksi esimerkeiksi reittejä, joissa on eniten anomalioita, joilla on positiivinen vaikutus lopputulokseen. Esimerkiksi vaikka tilanteessa, jossa kuljettajien välinen hajonta on saatu pienenemään työprosesseja parantamalla siihen pisteeseen, että anomalioita on vähän ja vuorot ajetaan aina lähes keskimääräisesti. Tällöin näitä positiivisen vaikutuksen tuovia anomalioita voidaan hyödyntää ohjaamaan kaikkien toimintatapoja paremmaksi, eli selvitetään sen syyt ja pyritään kopioimaan ne myös muiden toimintatapoihin. Jos vuoro-

kohtaisessa aineistossa olisi vielä lisäksi kuljettajatieto, aineistosta voitaisiin myös koittaa hakea kuljettajia, joilla on eniten positiivisen vaikutuksen anomalioita, eli jotka ajaisivat vaikka keskimääräistä taloudellisemmin.

Anomalioiden tunnistamisella voikin olla paljon saatavia etuja. Hyvän anomalioiden tunnistamisen kautta voitaisiin mahdollisesti esimerkiksi parantaa aikataulutusta tai pienentää kuluja esimerkiksi polttoaineen kulutuksen hallinnalla ja ennakoivalla huollolla. Myös kuljettajien koulutuksessa siitä olisi hyötyä. Nykytrendin mukaisesti näistä voisi myös koostaa esimerkiksi näkymiä (dashboardeja), joissa anomalioita tunnistettaisiin reaaliajassa ja visualisoitaisiin sopivalla hyödyllisellä tavalla. Tällöin ongelmiin voitaisiin puuttua mahdollisimman nopeasti. Tämä vaatisi kuitenkin menetelmiltäkin sen, että ne toimisivat tämänkaltaisessa reaaliaikaisessa havainto-havainnolta -aineistonsyötössä, eivätkä vaatisi syötettävältä aineistolta sitä, että se syötetään esimerkiksi kokonaisten vuorojen tai päivien sarjoissa (eng. batch).

3 Työn menetelmät

3.1 Vaatimukset työn menetelmille

Saatavilla oleva aineisto ei sisällä eksplisiittistä tietoa siitä, mikä havainto on anomalia ja mikä ei. Tästä syystä työssä käydään läpi pääasiassa ohjaamattomia menetelmiä ja niiden teoriaa. Menetelmien runsaudesta johtuen esiteltävien menetelmien valinnassa pyritään ottamaan huomioon käyttökelpoisuus työn kokeellisen osan aineiston kanssa. Aineistossa on havaintojen välinen aikariippuvuus, joka pyritään ottamaan huomioon menetelmiä valittaessa.

Kun menetelmiä aletaan käymään läpi, ne voidaan jakaa erilaisin tavoin. Tässä työssä menetelmät jaetaan useampaan eri pääryhmään, jotka ovat: probabilistiset ja tilastolliseen malliin perustuvat menetelmät, läheisyyteen perustuvat menetelmät sekä monimutkaisemmat menetelmät, jotka sisältävät mm. koneoppimiseen perustuvat menetelmät. Kaikkia näitä yhdistelevät ensemble-menetelmät ovat myös kiinnostava alue, mutta niitä ei käsitellä tässä työssä.

Jos aineisto olisi muuttujamääriltään suurempi, voitaisiin saada hyötyä mallinvalinnasta ja sen työkaluista. Ne voisivat helpottaa esimerkiksi mallin tulkintaa, tai keventää laskenta-ajan vaatimuksia. Tässä työssä mallin valintaa ei kuitenkaan käsitellä.

Menetelmien tulee olla myös täysin ohjaamattomia. Anomalioita voisi hakea semi-ohjaamattomasti siten, että harjoitusaineisto olisi koko aineisto, josta on poistettu anomaliat. Tällöin ne pitää siis olla aineistosta tiedossa. Tämä tapa ei sovellu työn aineistoon, sillä anomaliat eivät ole tiedossa.

Anomalioiden tunnistamiseen on kehitetty tässä työssä käsiteltävien autoenkooderien lisäksi paljon muitakin mielenkiintoisia koneoppimisen ja syväoppimisen malleja. Näitä ei kuitenkaan aiheen rajauksen takia käsitellä.

3.2 Tilastolliset menetelmät

Tilastollisiin menetelmiin voidaan lukea tilastolliset häntätödennäköisyydet ja tilastolliset testit. Jakaumariippumattomat häntätodennäköisyydet ovat hyödyllisiä, jos aineiston jakaumasta ei voida tai haluta tehdä oletuksia. Tällöin anomalioita voitaisiin tutkia myös häntäepäyhtälöillä (eng. tail inequalities), esimerkiksi Markovin epäyhtälöllä tai Chebychevin [20] epäyhtälöllä. Näissä menetelmissä on oletusten heikkouden takia kuitenkin paljon epätarkkuutta. Näitä menetelmiä ei tässä työssä käsitellä.

Intuitiivisesti yksinkertainen tapa anomalioiden tunnistamiseen on erilaisten tilastollisten testien käyttäminen. Perusajatuksena näissä on, että aineiston oletetaan noudattavan jotain tiettyä jakaumaa. Tällöin oletetaan, että tämä jakauma on generoinut kaikki aineiston pisteet eli jakauman parametrit voidaan estimoida aineistosta. Anomalioiksi voidaan sitten olettaa pisteet, joilla on alhainen todennäköisyys sille, että määritelty jakauma on lasketuilla parametreilla tuottanut havainnon. Tämän menetelmän käytössä tärkeää on määritellä se kuinka epätodennäköinen ha-

vainnon pitää olla ollakseen anomalia ja lisäksi erittäin tärkeää on valita jakauma oikein.

Erilaisia tilastollisia jakaumia ja niihin sopivia tilastollisia testejä on paljon. Erityisesti testejä löytyy eri jakaumille, muuttujien erilaisille määrille (ulottuvuuksille) ja niitä on lisäksi parametrisiä ja ei-parametrisiä. Yksinkertaisten jakaumien lisäksi voidaan käyttää myös sekoitemalleja ja niille sopivia testejä. Tilastollisten mallien etuna on myös se, että niitä voidaan soveltaa lähes minkä tyyppiselle aineistolle tahansa, niitä voidaan käyttää niin jatkuville kuin vaikka kategorisillekin muuttujille.

Tilastolliset testit ovat pääasiassa yksinkertaisia, tulkittavia ja helposti sovellettavia. Niillä on kuitenkin myös haittapuolensa, esimerkiksi jos aineiston jakaumaa ei tiedetä tai se osoittautuu vääräksi, tilastollisten testien perusteella tehty anomalioiden tunnistus ei toimi oikein. Haittapuolena on myös se, että mitä suuremmaksi mallin parametrien määrä kasvaa, sitä helpommin riskiksi tulee ylisovittaminen.

Yksi yksinkertainen esimerkki yksiulotteisen tapauksen tilastollisesta testistä on käyttää Z-arvoja, jotka voidaan kääntää helposti todennäköisyydeksi sille, että havainto on normaalijakaumasta

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}},$$

jossa f_X on tiheysfunktio satunnaismuuttujalle X, μ on jakauman odotusarvo, σ on keskihajonta ja x on havainto. Kun havaintojen määrä on suuri, voidaan keskihajonta ja odotusarvo estimoida aineistosta suurella tarkkuudella. Tällöin havainnon Z-arvo voidaan laskea lausekkeella

$$z_i = \frac{x_i - \mu}{\sigma}.$$

Z-arvo voidaan muuntaa todennäköisyydeksi käyttämällä standardinormaalijakauman kertymäfunktiota. Todennäköisyys kertoo siis sen, kuinka todennäköisesti keskiarvosta näin paljon poikkeava havainto on tullut normaalijakaumasta. Tätä tietoa voidaan käyttää anomalioiden tunnistamisessa, sillä mitä epätodennäköisemmin havainto on jakaumasta, sitä todennäköisemmin se on anomalia. Tällä menetelmällä voidaan kuitenkin tunnistaa vain ns. ääriarvoanomalioita.

Z-arvot eivät kuitenkaan ole ihanteellinen menetelmä moniulotteisille jakaumille. Tällöin yksi vaihtoehto on mallintaa aineisto moniulotteisena normaalijakaumana ja käyttää hyväksi Mahalanobiksen etäisyyttä ja sen ominaisuuksia. Tällöin d-ulotteisen normaalijakauman tiheysfunktio voidaan kirjoittaa muodossa

$$f_X(\boldsymbol{x}) = \frac{1}{\sqrt{|\boldsymbol{\Sigma}|}(2\pi)^{(d/2)}} \exp\left[-\frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{\mu}})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \bar{\boldsymbol{\mu}})^T\right],$$

jossa $\bar{\boldsymbol{\mu}}$ on d-ulotteinen jakauman keskiarvo, $\boldsymbol{\Sigma}$ on kovarianssimatriisi, $|\boldsymbol{\Sigma}|$ sen determinantti, ja $\boldsymbol{x}$ on d-ulotteinen havaintovektori.

Tästä nähdään, että eksponentissa on puolet Mahalanobiksen etäisyyden neliöstä, joka voidaan kirjoittaa muodossa

$$d_{Mahalanobis}(\boldsymbol{x}, \bar{\boldsymbol{\mu}}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x} - \bar{\boldsymbol{\mu}})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \bar{\boldsymbol{\mu}})^T}.$$

Mahalanobiksen etäisyys on käytännöllinen, koska se normalisoi aineiston muuttujien korrelaatioiden avulla, mikä erottaa sen euklidisestä etäisyydestä. Jos aineisto muunnetaan standardoiduiksi pääkomponenteiksi, Mahalanobiksen etäisyys on identtinen euklidisen etäisyyden kanssa.

Mahalanobiksen etäisyys on siitä hyödyllinen, että multinormaalijakautuneelle havainnolle se noudattaa χ^2-jakaumaa parametrilla d, eli aineiston dimensioiden määrällä. Tämä perustuu siihen, että standardoidusta normaalijakaumasta otettujen riippumattomien satunnaismuuttujien neliöiden summa noudattaa χ^2-jakaumaa, eli kyseinen jakauma d vapausasteella määritellään nimenomaan d:n riippumattoman standardinormaalijakaumasta otetun satunnaismuuttujan neliöiden summana, eli $\sum_{i=1}^{d} Z_i^2 \sim \chi^2(d)$. Tällöin anomalioiksi voidaan määritellä esimerkiksi kaikki ne pisteet, joille $d_{Mahalanobis} > \chi^2(0.975)$, joka vastaa (yksiulotteisessa jakaumassa) noin kolmen keskihajonnan etäisyyttä keskiarvosta.

Tämä menetelmä on yksinkertainen ja toimiva ja jos unohdetaan χ^2-jakauman käyttö leikkauspisteen valinnassa, saadaan jakaumariippumaton menetelmä, joka voidaan ajatella semiparametrisena menetelmänä, koska se vaatii keskiarvo- ja kovarianssimatriisiparametrien estimoinnin. Tällaisenaan se sopii erinomaisesti ohjaamattomaksi anomalioiden tunnistamismenetelmäksi. Se ei myöskään ole pelkän etäisyyden perusteella toimiva menetelmä, sillä se ottaa huomioon myös muuttujien välisiä riippuvuuksia ja toimiikin usein vähintään yhtä hyvin kuin monimutkaisemmatkin menetelmät [21], erityisesti robustina versiona. Sillä on myös etuja verrattuna monimutkaisempiin etäisyyteen perustuviin menetelmiin erityisesti tarkkuudessa, laskennan kompleksisuudessa ja parametrisoinnissa [7]. Se ei kuitenkaan ole täysin ongelmaton. Dimension kasvaessa Mahalanobiksen etäisyyden arvot lähenevät toisiaan ja anomaliat katoavat helpommin muun aineiston sekaan. Lisäksi Mahalanobiksen etäisyyden laskemisessa käytetään normaalijakauman parametrien estimaattoreina otoskeskiarvoa ja otoskovarianssimatriisia, jotka ovat varsin herkkiä poikkeaville havainnoille. Koska nämä parametrit lasketaan aineistosta, jossa nämä poikkeavat havainnot ovat mukana, ne vaikuttavat parametriestimaatteihin ja siten myös etäisyyksiin, jolloin anomaliat taas hukkuvat helpommin muun aineiston sekaan.

Mahalanobiksen etäisyydestä on myös robusti versio, joka ei ole niin herkkä anomalioille. Tämä on toivottava ominaisuus erityisesti silloin, kun sitä käytetään ohjaamattomaan anomalioiden tunnistamiseen, koska robusti menetelmä pienentää anomalioiden vaikutusta etäisyyteen. Yllä kuvatussa Mahalanobiksen etäisyydessä kovarianssimatriisi estimoidaan suurimman uskottavuuden menetelmällä (eng. maximum likelihood; MLE), mutta sen tuottama estimaattori on varsin herkkä aineiston anomalioille. Tämän takia myös itse etäisyydet ovat herkkiä anomalioille. Näissä tapauksissa olisi parempi käyttää kovarianssille robustia estimaattoria, jolloin saadaan kovarianssimatriisille parempi estimaattori, joka seuraisi paremmin todellisia havaintoja. Yksi vaihtoehto robustiksi menetelmäksi on pienimmän kovarianssin determinanttiestimaattori [22][23][24] (eng. minimum covariance determinant estimator; MCD). MCD:tä voidaan käyttää estimointiin aineistoissa, joissa on jopa $\frac{n-k-1}{2}$ anomaliaa ja sen idea on löytää $\frac{n+k+1}{2}$ havaintoa, joiden empiirisellä kovarianssimatriisilla on pienin determinantti. Laudekkeissa n on havaintojen määrä ja k on selit-

tävien muuttujien määrä. Tällöin saadaan aikaan anomaliaton alijoukko havaintoja, joille sitten voi laskea estimaatin keskiarvolle ja kovarianssimatriisille. Näin ollen, etäisyys voidaan merkitä muodossa

$$d_{RobustiMaha}(\boldsymbol{x}, \hat{\boldsymbol{\mu}}_{MCD}, \hat{\boldsymbol{\Sigma}}_{MCD}) = \sqrt{(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{MCD})\hat{\boldsymbol{\Sigma}}_{MCD}^{-1}(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_{MCD})^T},$$

joka vastaa muuten yllä esiteltyä tavanomaista Mahalanobiksen etäisyyttä, mutta estimoidut parametrit ovat MCD:llä estimoitu. MCD-estimointiprosessia ei tässä työssä tarkemmin esitellä, mutta sen eksakti tulos on laskennallisesti raskas, sillä sitä varten tulee laskea kaikkien $\binom{n}{h}$ alijoukon kovarianssimatriisi, jossa n on havaintojen määrä ja h on alijoukon koko. Tähän on kuitenkin kehitetty laskentaa nopeuttavia menetelmiä, kuten FAST-MCD, joka myöskään ei kuulu työn piiriin. MCD:kään ei ole ongelmaton, sillä sen harha kasvaa dimension kasvaessa [25].

3.2.1 Sekoitemallit

Sekoitemalleiksi [7] kutsutaan malleja, joissa oletetaan, että aineisto on peräisin jostakin k:sta parametrisesta jakaumasta. Esimerkiksi vaikka k:sta normaalijakaumasta tai multinormaalijakaumasta, joilla on omat parametrinsa eli odotusarvo ja varianssi tai kovarianssimatriisi. Jakaumien parametrit ovat kuitenkin tuntemattomia eikä myöskään tiedetä mihin jakaumaan mikäkin aineiston piste kuuluu.

Sekoitemallin teko aloitetaan olettamalla mallin muoto, esimerkiksi normaalijakaumien sekoite ja haluttu jakaumien määrä. Jakaumien määrän valinta voidaan tehdä kokeilemalla useampaa määrää ja valitsemalla näistä paras esimerkiksi BIC-arvon perusteella. Tämän jälkeen mallin parametrit estimoidaan iteratiivisella EM-algoritmilla (eng. expectation maximation). Tämän jälkeen voidaan laskea aineiston jokaisen pisteen todennäköisyys sopia sekoitemallin jokaiseen jakaumaan, jolloin anomaliat saavat pienen todennäköisyyden ja normaalihavainnot suuren [26][27]. On huomattava myös, että tavanomaisessa muodossaan sekoitemallit eivät ole ihanteellinen valinta ohjaamattomaan anomalioiden tunnistamiseen. [28] Tämä johtuu siitä, että anomalioista voi tulla oma anomaliajakaumansa, joka pitäisi tunnistaa. Tällöin pienin todennäköisyys ei enää kerro välttämättä anomaliasta.

Oletetaan malli M ja sille jakaumat G_i. Saadaan siis, että

$$f_{piste}(X_j|M) = \sum_{i=1}^{k} \alpha_i f^i(X_j),$$

jossa M on malli eli jakaumien G_i parametrit, α_i ($\sum_{i=1}^{k} \alpha_i = 1$) on prioritodennäköisyys jakaumalle i ja f^i on jakauman G_i tiheysfunktio ja X_j on js havainto. Prioritodennäköisyys voidaan nähdä painona, jolla kyseisen jakauman tiheysfunktion arvo painotetaan. Tällöin aineiston D kaikkien N havainnon yhteistiheysfunktio, saadaan kertomalla kaikki tiheysfunktiot yhteen eli

$$f_{aineisto}(D|M) = \prod_{j=1}^{N} f_{piste}(X_j|M).$$

Tätä vastaava logaritminen uskottavuusfunktio on

$$L(D|M) = \log \left[\prod_{j=1}^{N} f_{piste}(X_j|M) \right] = \sum_{j=1}^{N} \log \left[\sum_{i=1}^{k} \alpha_i f^i(X_j) \right].$$

Jos tiedetään minkä sekoitemallin pisteen tuotti mikäkin sekoitemallin jakauma, mallin optimiparametrit on helpompi määrittää erikseen jokaiselle sekoitteen komponentille. Toisaalta jokaisen pisteen todennäköisyys riippuu näistä parametreistä. Tämän takia parametrien estimointiin tarvitaan jokin iteratiivinen menetelmä, joka tässä tapauksessa on EM-algoritmi, jossa mennään askel askeleelta kohti optimiparametrejä. EM-algoritmia [29] ei tässä työssä esitetä. Tämän jälkeen voidaan laskea jokaiselle pisteelle x sen sopivuus malleihin $\alpha_i f^i(x)$. Jos sopivuus on huono, havainto on anomalia eli jos piste on kaukana kaikista jakaumista, se on todennäköisesti anomalia.

Menetelmän yksi haittapuoli on se, että sekoitemallin luomisessa pitää joko tietää tai arvioida mallin sisältävien jakaumien määrä. Lisäksi tämäkin on parametrisena mallina herkkä poikkeaville havainnoille. Myös, kuten jo mainittiin, jos menetelmää käytetään ohjaamattomana, jokin mallin jakaumista voi myös olla ns. anomaliajakauma, joka jollain tavalla pitäisi tunnistaa joukosta. Tähän on kehitetty erilaisia robusteja tapoja, joista yksi on asettaa tasajakauma yhdeksi jakaumista, joka sitten keräisi muiden jakaumien ulkopuoliset havainnot anomaliajakaumaksi [30].

3.3 Läheisyyteen perustuvat mallit

Läheisyyteen perustuvien mallien idea on se, että aineistossa anomaliat ovat havaintoja, jotka ovat avaruudessa selvästi erillään muista havainnoista, esimerkiksi jonkin etäisyysfunktion perusteella. Mallit ovat pääsääntöisesti ohjaamattomia ja ne ovat hyvin yleisesti anomalioiden tunnistamiseen käytettyjä malleja. Ne voidaan jakaa karkeasti kolmeen pääryhmään [8]:

1. Klusterointimenetelmät
2. Etäisyyteen perustuvat menetelmät
3. Tiheyteen perustuvat menetelmät

Klusteroinnissa ja tiheyteen perustuvissa menetelmissä etsitään aineiston tiheät alueet, jolloin anomaliat ovat havaintoja, jotka ovat näiden tiheiden alueiden ulkopuolella ja erityisesti kaukana näistä tiheistä alueista. Tiheällä alueella tarkoitetaan jotain avaruuden kohtaa tai aluetta, jossa havaintoja on lukumääräisesti enemmän kuin muissa. Klusterointi- ja tiheyteen perustuvat menetelmät ovat samankaltaisia, mutta eroavat siten, että klusterointi jakaa havaintopisteet toisistaan, kun taas tiheyteen perustuvat menetelmät jakavat havaintoavaruuden osiin.

Useissa klusterointimenetelmissä ensimmäinen askel on käyttää jotain klusterointialgoritmia, jotta saadaan selville aineiston tiheät alueet. Toisessa vaiheessa käytetään jotain soveltuvaa menetelmää selvittämään aineiston pisteiden sopivuus näihin

tiheisiin alueisiin. Esimerkiksi K-means-klusteroinnissa toisen vaiheen menetelmä voisi olla yksinkertainen etäisyys lähimpään klusterikeskukseen (eng. centroid), jolloin tästä saadaan sitten jokaiselle havainnolle arvio anomalia-asteesta. Klusteroinnissa, menetelmästä riippuen, havainto voidaan määritellä anomaliaksi monin eri tavoin. Anomalia voi olla havainto, joka muodostaa oman klusterinsa, kuuluu kooltaan pieneen klusteriin, on pienessä klusterissa ison klusterin lähellä tai joku näiden yhdistelmä. K-Means-klusteroinnin lisäksi esimerkkejä klusterointimenetelmistä ovat mm. lähinaapurimenetelmä (käsitelty kappaleessa 3.3.2), sumea klusterointi [8] (eng. fuzzy clustering) ja kokoava klusterointi [8] (eng. agglomerative clustering).

Tiheyteen perustuvissa menetelmissä taas sijainti avaruudessa kertoo anomalia-asteen, ei niinkään suora suhde muihin havaintoihin. Tiheyteen perustuvat menetelmät siis klusteroivat otoksen perusteella luodun populaatiojakauman. Esimerkkejä tiheyteen perustuvista menetelmistä ovat mm. kappaleessa 3.2.1 käsitelty sekoitetiheyden estimointi (eng. mixture density estimation) ja LOF (local outlier factor), joka myös käsitellään työssä myöhemmin.

Etäisyyteen perustuvissa menetelmissä lasketaan jokaisen pisteen etäisyys jollain etäisyysfunktiolla k:hon sitä lähellä olevaan havaintoon. Tällöin niitä havaintoja voidaan pitää anomalioina, joilla on suuret etäisyydet lähinaapureihin. Etäisyyteen perustuvat menetelmät ovat usein myös laskennallisesti tiheyteen ja klustereihin perustuvia menetelmiä raskaampia. Esimerkkejä etäisyyden laskutavoista ovat mm. etäisyys kaikkiin pisteisiin, etäisyys lähinaapuriin ja keskietäisyys k:hon lähinaapuriin [7].

3.3.1 K-means-klusterointi

K-means-klusterointi on tunnettu klusterointimenetelmä [8]. Menetelmässä määritellään haluttu klustereiden määrä k, asetetaan klusterikeskukset satunnaisien havaintojen mukaan ja liitetään jokainen havainto siihen klusteriin, johon havainnon etäisyys on lyhin. Tämän jälkeen klusterien keskukset päivitetään klusteriin liitettyjen havaintojen keskiarvoiksi ja toistetaan klustereihin asettaminen ja uuden klusterikeskuksen asetus. Iterointia jatketaan, kunnes klusterikeskuksiin liitettyjen päivityksessä muuttuvien havaintojen määrä on pienempi kuin jokin asetettu kynnysmäärä θ.

Menetelmä on yksinkertainen ja sillä on yksi säätöparametri, eli haluttu klustereiden määrä. Tämän asettamisessa voidaan käyttää erilaisia menetelmiä, voidaan esimerkiksi minimoida klusterien sisäisen ja klusterien välisen keskimääräisen etäisyyden suhdetta. Voidaan käyttää myös silhuettikertoimen [31][32] menetelmää (eng. silhouette coefficient), jolla voidaan verrata erilaisia k:n arvoja laskemalla silhuettiarvo jokaiselle havainnolle ja sitten vertaamalla arvoa käyttäen sitä miten samankaltainen havainto on klusteriinsa ja muihin klustereihin verrattuna. Silhuettiarvo havainnolle i lasketaan

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}},$$

jossa $a(i)$ on havainnon i keskietäisyys kaikista sen oman klusterin muista havainnoista ja $b(i)$ on havainnon i lyhin etäisyys muihin kuin omaan klusteriin. Silhuet-

tikerroin on sitten jokaisen klusterin siluettiarvojen keskiarvon maksimi yli koko aineiston, eli

$$SC = \max_{\kappa} \widetilde{s}(\kappa),$$

jossa $\widetilde{s}(\kappa)$ on klusterin κ silhuettiarvojen $s(i)$ keskiarvo ja SC on siis tämän joukon maksimiarvo. Silhuettiarvot ovat välillä $[-1, 1]$ ja jos suurimmalla osalla havainnoista on suuri positiivinen arvo, klusterointi on hyvä, kun taas jos moni havainto saa alhaisen tai negatiivisen arvon, klusterointi ei ole hyvä ja klustereita saattaa olla liian vähän tai liian paljon.

Formaalimmin K-means-menetelmä voidaan esittää klusterien sisäisten neliösummien minimoinnin muodossa, eli

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} ||\boldsymbol{x} - \boldsymbol{\mu}_i||^2,$$

jossa vektorit $\boldsymbol{x}$ ovat d-ulotteisia havaintoja, jotka halutaan luokitella k joukkoon, eli $S = \{S_1, S_2, \ldots, S_k\}$. Kaavassa $\boldsymbol{\mu}_i$ on joukon S_i keskipiste. Menetelmä voitaisiin esittää myös muodossa, jossa minimoidaan saman klusterin parittaisten etäisyyksien keskiarvo.

Klusteroinnin jälkeen pitää vielä määrittää havainnot, jotka voidaan asettaa anomalioiksi. Yleisesti tähän voidaan käyttää yhdistelmää kolmesta eri säännöstä, eli 1. anomalioiksi määritetään pienet klusterit, joissa on maksimissaan y prosenttia kaikista havainnoista tai y havaintoa, 2. anomalioiksi asetetaan havainnot, jotka eivät kuulu mihinkään klusteriin (ts. muodostavat omat kulusterinsa), ja 3. anomalioiksi asetetaan havainnot, jotka ovat kauempana kuin joku tietty etäisyys klusterin keskuksesta, esimerkiksi yli kahden klusterin sisäisen keskihajonnan päässä.

Menetelmästä on kymmeniä erilaisia muunnoksia ja jatkokehiteltyjä versioita, esimerkiksi k-medians-klusterointi [33], K-means++ [34] ja Otsun menetelmä [35]. Tässä työssä käytetään kuitenkin yksinkertaisinta perusversiota [36][7].

3.3.2 Lähinaapurimenetelmät

Kun lähinaapurimenetelmää käytetään anomalioiden tunnistamisessa, oletetaan, että normaalihavainnot esiintyvät tiheissä naapurustoissa ja anomaliat sitten vähemmän tiheissä naapurustoissa ja yleensä vielä kaukana tiheistä naapurustoista. Naapurustojen määrittämisessä käytetään jotain sopivaa etäisyysmittaa, esimerkiksi vaikka euklidistä etäisyyttä. Pitää siis laskea jokaisen havainnon etäisyys jokaisesta muusta havainnosta. Lasketaan siis esimerkiksi kahden d-ulotteisen havainnon $\boldsymbol{x}$ ja $\boldsymbol{x}'$ euklidinen etäisyys $d(\boldsymbol{x}, \boldsymbol{x}')$ eli

$$d(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2 + \cdots + (x_d - x_d')^2}.$$

Tämä toistetaan kaikille N havainnosta muodostuvalle $\binom{N}{2}$ parille. Kuten nähdään, mitä isompi aineisto on, sitä raskaampia lähinaapurimenetelmät ovat laskennallisesti. Niistä on tehty monenlaisia versioita, joissa laskenta-aikaa on pyritty optimoimaan [7].

Yksinkertaisimmillaan anomaliaksi voidaan määrittää esimerkiksi a kappaletta havaintoja, joilla etäisyyksien summa kaikkiin muihin on pisin. Toinen tapa on laskea yhteenlaskettu etäisyys esimerkiksi k lähimpään naapuriin ja asettaa taas a suurimman mitan havaintoa anomalioiksi. Muitakin tapoja on, mutta niitä ei käsitellä tässä työssä [7].

3.3.3 Hierarkkinen klusterointi

Hierarkkisella klusteroinnilla [37] ja lähinaapurimenetelmillä on jonkin verran yhteistä, sillä molemmat perustuvat havaintojen keskinäisiin etäisyyksiin. Hierarkkisessa klusteroinnissa lasketaan havaintojen välisistä etäisyyksistä matriisi, jonka perusteella kaikki havainnot yhdistetään klustereiksi askel kerrallaan, kunnes lopputuloksena on kaikki havainnot sisältävä yksi iso klusteri. Tätä kutsutaan kokoavaksi (eng. agglomerative) klusteroinniksi. Klusterointi voidaan myös tehdä hajoittavasti (eng. divisive) siten, että aloitetaan yhdestä isosta klusterista ja havaintoja irrotetaan pienemmiksi klustereiksi, kunnes kaikki ovat oma klusterinsa.

Hierarkkisen klusteroinnin tuloksena voidaanm muodostaa myös kuvaaja, jota kutsutaan dendrogrammiksi. Se on hierarkkinen kuvaus havaintojen rakenteesta.

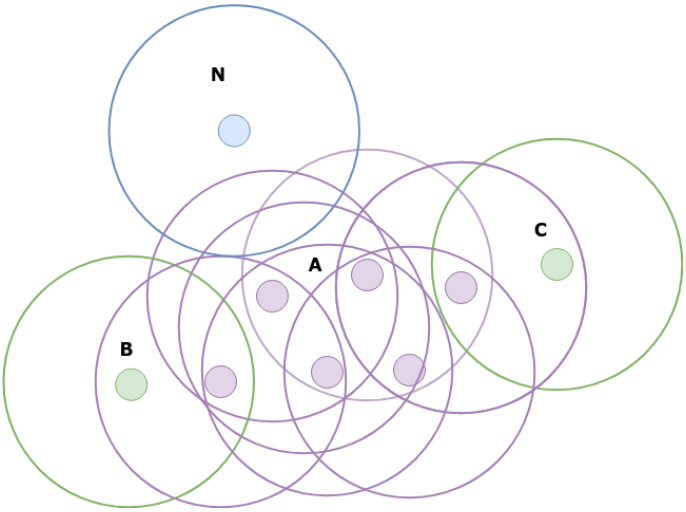
3.3.4 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) on klusterointimenetelmä, joka on suunniteltu löytämään nopeasti minkä tahansa muotoisia klustereita (myös asymmetrisia). Menetelmän etuna on myös se, että sen tarvitsee käydä aineisto läpi vain kerran. Tämä luku perustuu lähteisiin [8] ja [38].

Keskeinen ajatus menetelmässä on ns. ydinhavainnot (eng. core points). Piste p on ydinpiste, jos vähintään θ pistettä on siitä etäisyydellä r (mukaanlukien piste p). Tällöin piste q on suoraan saavutettavissa pisteestä p, jos piste q on maksimissaan etäisyyden θ päässä ydinpisteestä p. Ydinhavaintojen läheisyydessä olevat klustereiden pisteet ovat ns. rajapisteitä (eng. border points). Rajapisteet eivät ole ydinpisteitä, koska ne eivät jatka klusteria eteenpäin. Menetelmä on kuvattu kuvassa 3.

Piste q on saavutettavissa pisteestä p, jos sille on polku $p_1, \ldots, p_n$, jossa $p_1 = p$ ja $p_n = q$ sekä jokainen p_{i+1} on suoraan saavutettavissa pisteestä p_i. Tämä tarkoittaa siis sitä, että alkupisteen sekä kaikkien polulla olevien pisteiden (paitsi q) pitää olla ydinpisteitä. Jos piste p on ydinpiste, se muodostaa klusterin kaikista pisteistä (ydin- ja rajapisteistä) jotka ovat saavutettavissa siitä. Jokainen klusteri sisältää myös ainakin yhden ydinpisteen. Kaikki pisteet, jotka eivät ole saavutettavissa mistään muusta pisteestä, ovat anomalioita tai kohinapisteitä, eli ovat etäisyyttä r kauempana klusterin ydinpisteistä [38].

DBSCANin haittapuolena on se, että sillä on ongelmia aineistoissa, joissa on eri tiheyksisiä klustereita.



Kuva 3: DBSCAN-menetelmän toimintaperiaate: Violetit ovat ydinhavaintoja, koska havainnon ympärille piirretyn r-säteisen ympyrän sisällä on vähintään θ kappaletta haviantoja (esimerkissä $\theta = 4$). B ja C ovat rajapisteitä, eli pisteitä, jotka kuuluvat ympyröiden perusteella ydinhavaintojen muodostamaan klusteriin, mutta eivät enää jatka sitä (eli niiden ympyrän sisällä on alle θ kappaletta havaintoja. Lisäksi ulkopuolella on anomalia N, joka ei ympyränsä mukaan kuulu klusteriin.

3.3.5 LOF

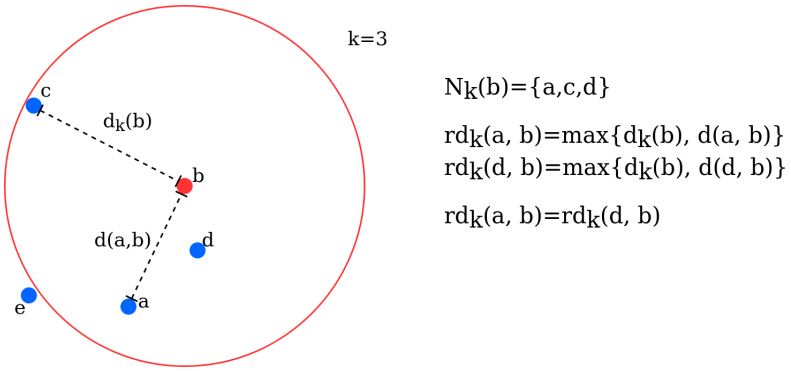
LOF (local outlier factor) -menetelmässä anomaliat tunnistetaan paikallisen tiheyden avulla, sillä anomalioiden tapauksessa paikallinen tiheys on yleensä pienempää kuin normaalien havaintojen. Menetelmässä valitaan k lähinaapuria ja lasketaan havainnon etäisyys näihin. Sen jälkeen lasketaan pisteelle lokaali tiheys, josta voidaan laskea anomalia-arvo jokaiselle pisteelle. Se muodostetaan tyypillisestä suhteellisesta etäisyydestä, joka siihen on naapurihavainnoista. Yleisesti havainto on anomalia, jos LOF-arvoksi tulee enemmän kuin 1. Menetelmä toimii hyvin ilmiselvien anomalioiden havaitsemisessa, mutta toimii heikommin, kun anomaliat ovat lähellä normaaleja havaintoja tai jos anomalioitakin on isompi joukko. Tämä luku perustuu lähteisiin [39] ja [8].

Formalisoidaan seuraavaksi edellisen kappaleen kuvaus. Olkoon $\mathrm{d}_k(a)$ pisteen a etäisyys sen k:nteen lähinaapuriin. Näistä muodostuu k:n lähinaapurihavainnon muodostama joukko $\mathrm{N}_k(a)$, joka saattaa sisältää enemmän kuin k havaintoa, jos joihinkin havaintoihin on yhtä pitkä matka. Saavutettavuusetäisyys (eng. reachability distance) pisteestä a pisteeseen b:hen, eli $\mathrm{rd}_k(a, b)$, voidaan sitten kirjoittaa muodossa

$$\mathrm{rd}_k(a, b) = \max\{\mathrm{d}_k(b), \mathrm{d}(a, b)\},$$

jossa $\mathrm{d}(a, b)$ on etäisyys pisteiden a ja b välillä, joka voi olla melkein mikä tahansa etäisyysmetriikka, esimerkiksi Manhattan-etäisyys, euklidinen etäisyys tai Minkowskin etäisyys. Tällöin saavutettavuusetäisyys pisteestä a pisteeseen b on siis pisteiden välinen todellinen etäisyys tai $\mathrm{d}_k(b)$, jos se on suurempi. Tämä tarkoittaa sitä, että pisteitä, jotka kuuluvat b:n k:n lähinaapurin joukkoon, voidaan pitää yhtä kaukaisina

suhteessa muuhun aineistoon. Periaate on esitettynä kuvassa 4.



Kuva 4: Punaisen ympyrän sisällä on pisteen b k lähinaapuria. Kuvasta näkee myös esimerkkinä etäisyyksien $\mathrm{rd}_k(a,b)$ ja $\mathrm{rd}_k(d,b)$ muodostuminen.

Paikallinen saavutettavuustiheys $\mathrm{lrd}_k(a)$ (eng. local reachability density) voidaan lopulta laskea

$$\mathrm{lrd}_k(a) = \left(\frac{\sum_{b \in \mathrm{N}_k(a)} \mathrm{rd}_k(a,b)}{|\mathrm{N}_k(a)|} \right)^{-1},$$

joka on pisteen a keskimääräisen saavutettavuusetäisyyden käänteisluku, tarkemmin se etäisyys, jolla piste a voidaan saavuttaa naapureistaan. Voidaan ajatella, että tämä paikallinen saavutettavuusetäisyys kertoo, miten kauas pitää matkustaa tietystä pisteestä, jotta saavutetaan seuraava pisteiden klusteri. Mitä pienempi paikallinen saavutettavuus on, sitä vähemmän tiheä alue on ja sitä pidemmälle pitää siis liikkua.

Lopuksi näitä paikallisia saavutettavuustiheyksiä verrataan naapureihin eli lasketaan $\mathrm{LOF}_k(a)$ kaavalla

$$\mathrm{LOF}_k(a) = \frac{\sum_{b \in \mathrm{N}_k(a)} \frac{\mathrm{ldr}_k(b)}{\mathrm{ldr}_k(a)}}{|\mathrm{N}_k(a)|} = \frac{\sum_{b \in \mathrm{N}_k(a)} \mathrm{ldr}_k(b)}{|\mathrm{N}_k(a)| \mathrm{ldr}_k(a)},$$

joka on siis pisteen keskimääräinen saavutettavuustiheys (eng. average local reachability density) jaettuna pisteen omalla lokaalilla saavutettavuustiheydellä. LOF on siis käytännössä pisteen a naapurien keskimääräisen ldr_k-arvon suhde pisteen a arvoon $\mathrm{ldr}_k(a)$. Jos tämä suhde on suuri (yli 1), pisteen a tiheys on keskimäärin pienempi kuin naapuriensa tiheys, jolloin tästä pisteestä täytyy siirtyä pidempiä matkoja seuraavaan pisteeseen tai seuraavaan pisteiden klusteriin verrattuna matkaan, jonka pisteen a naapureiden pitää siirtyä omiin naapureihinsa. Eli, jos LOF-arvo on pienempi kuin 1, tiheys on korkeampi ja jos se on suurempi kuin 1, tiheys on alempi. Jos arvo on noin 1, voidaan pisteen sanoa olevan naapureidensa kaltainen. Yleisesti ykköstä suurempia arvoja voidaan pitää anomalioina, mutta tarkka arvon valinta jää aina mallintajan päätettäväksi.

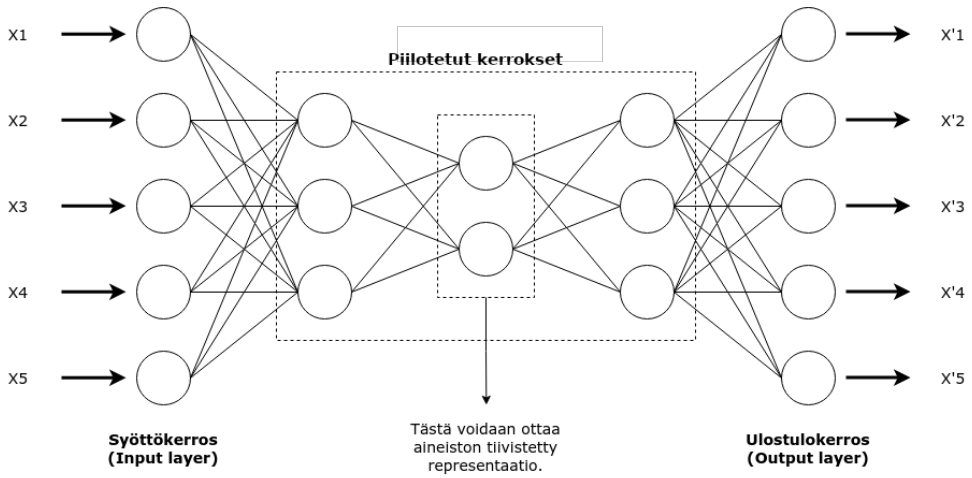
3.4 Muut menetelmät

3.4.1 Autoenkooderi

Autoenkooderi on menetelmä anomalioiden tunnistamiseen. Yksinkertaistettuna se rakentaa mallin aineistosta kaksivaiheisesti siten, että ensin se tiivistää aineiston enkoodaus-vaiheessa ja sitten laajentaa sen uudelleen dekoodaus-vaiheessa. Ajatus on siis se, että jos malli voi oppia funktion, joka tiivistää aineiston, silloin se voi myös rakentaa sen uudelleen. Autoenkoodereita käytetään optimaalisesti ohjattuna siten, että malli harjoitetaan vain normaalilla aineistolla, josta on poistettu anomaliat (semi-ohjattu), mutta se toimii myös ohjaamattomana, jos anomalioiden osuus on riittävän pieni. Luku perustuu lähteeseen [7].

Autoenkooderit ovat olleet viime vuosina varsin yleinen tutkimuskohde ja niistä on tehty myös eteenpäin kehitettyjä menetelmiä, lisäämällä verkoston kerroksien määrää. Niistä on tehty myös muunlaisia muunnoksia, kuten todennäköisyyksiin perustuva muuttuva autoenkooderi (VAE, variational autoencoder) [40]. VAE:t kehitettiin tekstin ja kuvien generoimiseen, mutta ovat löytäneet hyvin sovelluksia myös anomalioiden tunnistamisesta.

Autoenkooderi on siis syväoppimismalli, jossa on kaksi komponenttia. Enkooderi oppii alemman dimension esitystavan aineistosta ja dekooderi koittaa sitten luoda datan uudelleen alkuperäiseen dimensioon käyttäen enkooderin pienen dimension esitystä. Prosessi on esitetty kuvassa 5. Autoenkooderia voidaankin ajatella jonkinlaisena pakkaustapana, sillä hyvin suunniteltu ja harjoitettu autoenkooderi oppii aineistosta tärkeän sisällön ja jättää epäoleelliset huomiotta. Autoenkooderin oppiminen tapahtuu pyrkimällä minimoimaan uudelleenrakennusvirhettä (reproduction error) alkuperäisen aineiston havainnon ja dekooderista tulevan uudelleenrakennetun havainnon välillä.



Kuva 5: Autoenkooderissa vasemman puolen piilotetut kerrokset (joita käytännössä on aina enemmän kuin 1) tiivistävät informaatiota ja välituloksena voidaan ulos ottaa tiivistetty aineiston representaatio. Autoenkooderin oikea puoli sitten pyrkii rakentamaan sisään syötetyn aineiston uudelleen. Anomalia voidaan tunnistaa uudelleenrakennusvirheen avulla.

Anomalioiden tunnistamiseen autoenkooderia voi soveltaa siten, että kun malli on harjoitettu valmiiksi, sen läpi syötetty normaali anomaliaton havainto $\boldsymbol{x}$ antaa pienen uudelleenrakennusvirheen uudelleenmuodostettuna $\boldsymbol{x}'$, kun taas anomalian sisältävä havainto antaa suuren uudelleenrakennusvirheen re eli

$$\text{re} = \sum_{i=1}^{d} (x_i - x_i')^2.$$

Menetelmän taustalla on matriisin tekijöihin jakaminen (eng. matrix factorization), jota voidaan myös pitää enkoodaus- ja dekoodausmenetelmänä. Faktorisaatio $D \approx UV^T$ tiivistää matriisin D alemman asteen faktoreiksi U ja V, ja edelleen tulo UV^T palauttaa uudelleenrakennetun matriisin $D' = UV^T$.

Sellaisenaan autoenkooderi on kuitenkin huono ohjaamattoman opetuksen malli, sillä myös sen kohdalla opetusaineistossa olevat anomaliat heikentävät mallin estimointia. Varsinkin syvistä autoenkoodereista on kehitetty useita robusteja menetelmiä, esimerkiksi kohinaa poistava autoenkooderi [41] (eng. denoising autoencoder) ja korrelaatioentropian maksimoiva enkooderi [42] (eng. maximum correntropy autoencoder). Lisäksi on myös Zhoun et al. kehittämä malli, joka käyttää hyväksi robustia pääkomponenttianalyysia [43]. Tässä mallissa autoenkooderiin lisätään suodatuskerros, joka poistaa anomaliaksi havaitut vaikeammin uudelleenrakennettavat osat aineistosta, jolloin mallin sovittaminen onnistuu paremmin. Tämä voidaan tehdä joko l_1- tai $l_{2,1}$-regularisaatiolla. Menetelmää ei tässä työssä kuvata tarkemmin.

Autoenkooderi voidaan rakentaa myös LSTM-muotoon (long short-term memory), jolloin se ei ole enää suoraviivainen eteenpäin syöttävä (eng. feed forward) neuroverkko, vaan siinä on kerrosten välillä takaisinkytkentöjä. Tämänkaltainen neuroverkkorakenne sopii hyvin jatkuvaan aineistoon, esimerkiksi aikasarjoille, videolle tai äänelle. Monimutkaisuutensa vuoksi tätäkään ei tässä työssä kuitenkaan käsitellä sen enempää.

3.5 Menetelmistä

Edellä tehty katsaus erilaisiin menetelmiin oli laaja ja tutkimuksessa näkyi, miten paljon erilaisia menetelmiä on kehitetty erilaisiin tarpeisiin. Näistä menetelmistä esittelyyn valittiin vain pieni osa ja on hankala sanoa, ovatko nämä menetelmät ideaaleja työn aineistolle, vai onko olemassa vielä parempia. Toisaalta johonkin on vedettävä raja. Anomalioiden tunnistus on erittäin vilkas tutkimusala. Parannettuja sekä uusia menetelmiä kehitetään jatkuvasti ja lähivuodet tuovat varmasti mukanaan uusia mielenkiintoisia menetelmiä erilaisiin tarpeisiin.

Lisäksi kaikkia mainittuja, sekä monia muita menetelmiä voisi käyttää soppivana yhdistelmänä. Nämä ns. Outlier ensemble -menetelmät [7] ovat mainitsemisen arvoinen ryhmä. Kuten monessa muussakin sovelluksessa, myös anomalioiden tunnistamisessa saadaan usein hyötyä käyttämällä ensemble-malleja, jotka ovat siis käytännössä kokoelma erilaisia toisistaan poikkeavia malleja. Näillä voidaan saada mallia robustimmaksi, ja eri malleja voidaan käyttää esimerkiksi peräkkäin, jolloin

ensimmäinen malli antaa tulokseksi, jotain josta seuraava malli sitten jatkaa. Mallit voivat olla myös rinnakkain, jolloin mallit saavat saman aineiston ja lopputulos on esimerkiksi näiden mallien antama keskiarvo. Erilaiset ensemble-mallit ovat mielenkiintoinen osa anomalioiden tunnistamista, mutta erittäin laajana alueena näitä ei käsitellä tässä työssä.

4 Empiirinen osa

4.1 Aineisto ja menetelmät

4.1.1 Aineisto

Työssä käytetty aineisto on kerätty HSL:n (Helsingin seudun liikenne) ylläpitämästä avoimesta MQTT-rajapinnasta [44]. MQTT on kevyt julkaise-tilaa -protokolla (eng. publish-subscribe), jota käytetään mm. välittämään viestejä erilaisten laitteiden välillä yleensä TCP/IP-protokollan yli. MQTT ja vastaavat protokollat ovatkin usein käytettyjä nimenomaan IoT-maailmassa. Työssä tiedon keräämiseen käytetty HSL:n HFP API [45] tarjoaa siis tällaisen julkaise-tilaa -rajapinnan reaaliaikaisiin ajoneuvojen sijaintitietoihin. MQTT:n kaltaisessa protokollassa tilattavia päivityksiä kutsutaan aiheiksi (eng. topic) ja asiakas (eng. client) määrittelee mitkä aiheet haluavat palvelimen välittäjän (eng. broker) itselleen toimittavan. Aiheet esitetään yleensä hierarkkisesti ja kuvataan kauttaviivoin, eli esimerkiksi järjestelmässä X olevan koneen A lämpötilasensorin 1 mittaukset tilattaisiin aiheesta

`/järjestelmä_X/kone_A/lämpötila_1.`

Tällöin siis palvelin toimittaisi asiakkaalle aina viestillä uuden arvon, kun mittauksen arvo muuttuu. Juuri tämä on myös MQTT:n ja muiden vastaavien julkaise-tilaa -protokollien idea.

Työn aineisto kerättiin itse tehdyllä Python-ohjelmalla, joka on annettuna liitteessä 6.1. Tämä otti yhteyden HSL:n MQTT-palvelimeen käyttäen paho-mqtt -kirjastoa, tilasi halutut aiheet ja tallensi sitten välittäjältä saadut viestit sqlite-tietokantaan noin yhden kuukauden ajan. Työn aineistoksi valittiin HSL:n linja-autolinja 20, joka kulkee Eiran, Kampin ja Munkkivuoren väliä. Valintaperusteena käytettiin riittävän taajaa vuoroväliä sekä reitin sopivaa pituutta. Vuoron piti olla myös yksi niistä vuoroista, jotka ovat Helsingin alueella, sillä kaikista HSL:n busseista järjestelmään ei tule kattavaa tietoa. Koska reitit ovat kaksisuuntaisia, aineistoksi valittiin vain toinen suunta, eli Eira-Kamppi-Munkkivuori. Aineistoa kerättiin noin yhden kuukauden ajan ja sitä kertyi tietokantaan noin 5,6 miljoonaa riviä. Tiedosto oli kooltaan 1186 megatavua.

Käytetyn HFP API:n (high frequency positioning application programming interface) aiherakenne on

```
/<prefix>/<version>/<journey_type>/<temporal_type>/<event_type>/
<transport_mode>/<operator_id>/<vehicle_number>/<route_id>/
<direction_id>/<headsign>/<start_time>/<next_stop>/
<geohash_level>/<geohash>/<sid>,
```

jonka tarkempi sisältö ja eri osien vaihtoehdot on kuvattu API:n dokumentaatiossa [45]. Tämän työn aineistoa varten tilattu MQTT-aihe oli

`/hfp/v2/journey/ongoing/vp/bus/+/+/1020/1/#,`

jossa protokollan mukaan tilataan menossa olevien (/journey/ongoing/) bussimatkojen positioeventit (/vp/bus/) kaikille operaattoreille ja autonumeroille (/+/+/) reitillä 1020 ja sen alireitillä (/1020/1/), joka siis on bussi 20 Munkkivuoreen. Plus (+) ja risuaita (#) -merkit ovat jokerimerkkejä (eng. wildcard), jossa plus-merkki vastaa yhden aiheosan kaikkia vaihtoehtoja ja risuaita-merkki korvaa yhden tai useamman aiheosan, ja tässä tapauksessa se tarkoittaa "kaikki loput voivat olla mitä tahansa".

4.1.2 Aineiston kuvailu ja käsittely

Käytetty aineisto on tyypiltään aikasarjaa, mutta vuoroja tarkastellessa sitä voidaan pitää funktionaalisena datana [46] eli joukkona erillisiä, mutta toisiaan vastaavia aikasarjoja. Lisähaastetta aineiston käsittelyyn tuo se, että nämä aikasarjat käsittelevät samaa linja-autoreittiä, mutta sisältävät silti eri määriä havaintoja. Tämä johtuu siitä, että havaintoja tulee noin yksi per sekunti ja eri vuorot ajavat reitin eri ajassa. Tästä syystä aineisto pyrittiin muokkaamaan niin, että kaikki vuorot olisivat saman mittaisia ja siten helpommin analysoitavissa.

Kerätty aineisto sisälsi 5 611 362 riviä, joissa oli tietoa 2324 vuorosta. Aineisto sisältää 24 saraketta, jotka ovat kuvattuna taulukossa 1. Taulukon kuvausta tarkemmat määrittelyt näille löytyy HSL:n dokumentaatiosta [45]. Alussa ja lopussa oli muutama vajaa vuoro, jotka siivottiin pois. Lisäksi aineisto sisälsi vajaita vuoroja, esimerkiksi kiireellisemmän ajan jälkeen loppuneita vuoroja eli kesken reitin varikolle lähteneitä sekä muista syistä vajaiksi jääneitä. Myös nämä sekä joitain muita ilmiselvästi vääriä vuoroja pyrittiin suodattamaan aineistosta pois. Tämä tehtiin, koska nämä on helppo tunnistaa aineistosta jo keräysvaiheessa ilman varsinaisia anomalioiden tunnistamismenetelmiä ja toisaalta ilman niitä aineistosta saatiin hieman homogeenisempi ja siten myös tarkoituksenmukaisemmin analysoitava.

Aineiston siivoukseen etsittiin erilaisia menetelmiä, joista kuitenkin monet olivat aineiston laajuuden takia ajankäytöllisesti vaikeita, sillä ne olivat laskentaintensiivisiä ja aineiston läpikäynti olisi monilla menetelmillä kestänyt viikkoja. Haluttiin verrata kaksiulotteisia reittejä toisiinsa niin, että voitaisiin lopuksi suodattaa pois ne, jotka eivät selvästi vastaa normaalia reittiä, eli että esimerkiksi täysin vajavaiset ja jostain väärästä pisteestä alkavat reitit saadaan helposti pois. Tätäkin vaikeutti se, että ajetuilla reiteillä aineistopisteitä tuli kerran sekunnissa ja eri aikoihin reitin ajo vei eri määrän aikaa. Suora piste-pisteeseen vertaaminen ei esimerkiksi ollut mahdollista. Reittien vertaamiseen päädyttiin käyttämään Fréchet-etäisyyttä (myös eng. coupling distance) ja nimenomaan sen optimoitua diskreettiä Fréchet-etäisyyttä [47] (tarkemmin FastDiscreteFrechetMatrix [48]), jolla pystyttiin vertaamaan eri mittaisia reittejä toisiinsa. Menetelmä vertaa kahden käyrän samankaltaisuutta ottaen huomioon sijainnin lisäksi järjestystä. Diskreetti versio on menetelmän laajennus, joka on suunniteltu polygonikäyrille, jolloin se on huomattavasti vähemmän laskentaintensiivinen. Aluksi tarkoitus oli tehdä matriisi, jossa kaikkia reittejä olisi verrattu kaikkiin, mutta tämäkin olisi kestänyt CPU-intensiivisyytensä vuoksi viikkokausia. Päädyttiin valitsemaan käsin yksi mahdollisimman siisti mallireitti, johon sitten kaikkia muita reittejä verrattiin ja eniten eroavat poistettiin aineistosta suoraan. Nämä olivat pääasiassa laajoja GPS-virheitä tai tilanteita, joissa kuljettaja

Sarakkeen nimi	**Sarakkeen kuvaus**
id	Aineiston rivin id
topic	Rivin tuottanut aihetilaus
desi	Linja-auton vuorotunnus, aineistossa 20
dir	Reittisuunta, aineistossa 1
oper	Reitin operoinut yritys
veh	Käytetyn linja-auton numero
tst	Aikaleima, UTC-aika
tsi	Aika UNIX-muodossa
spd	Linja-auton nopeus
hdg	Linja-auton suunta
lat	Linja-auton leveyspiiri
long	Linja-auton pituuspiiri
acc	Linja-auton kiihtyvyys
dl	Estimoitu ero aikatauluun sekunteina (positiivinen kokonaisluku, jos jäljessä aikataulua ja negatiivinen, jos edellä)
odo	Vuoron matkamittarilukema
drst	Linja-auton ovien tila: 0 kaikki kiinni, 1 vähintään yksi auki
oday	Vuoron ajopäivä (alkamishetken mukaan)
jrn	Sisäistä tietoa, ei hyödyllinen tälle työlle
line	Sisäistä tietoa, ei hyödyllinen tälle työlle
start	Vuoron aikataulun mukainen aloitusaika
loc	Sijaintitiedon lähde: GPS, ODO (laskettu matkamittarin avulla)
stop	Pysäkkikoodi pysäkille jolla linja-auto on havaintohetkellä
route	Reitin reitti-id
occu	Matkustajien määrä (ei käytössä linja-autoilla)

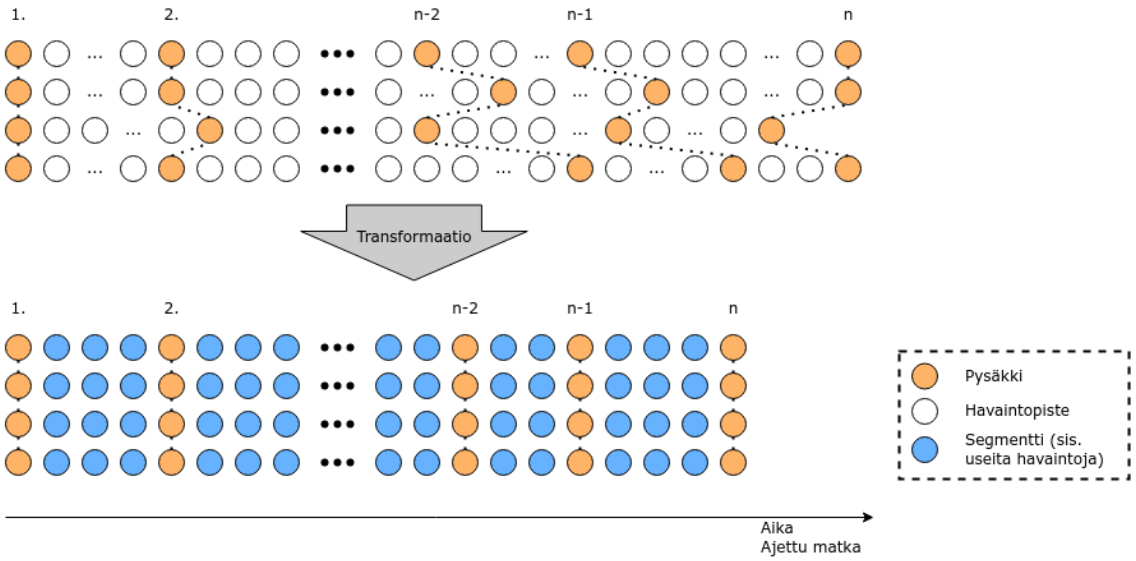
Taulukko 1: Aineiston sarakkeet ja sarakkeiden kuvaukset.

oli unohtanut käynnistää tai sammuttaa loggauksen, tai se oli ollut epäkunnossa.

Aineistossa oli jokaisella vuorolla yli toista tuhatta reittipistettä (keskimäärin noin 2400). Ennen kuin Fréchet-etäisyyttä pystyttiin käyttämään soveliaassa ajassa, piti tehdä lisää optimointia, eli aineiston reittipisteiden määrää piti vielä vähentää. Tässä menetelmäksi valikoitui Geopandas-kirjastossa oleva GeoSeries.simplify -metodi [49]. Aineiston reittien leveys- ja pituuspiirikoordinaatit muutettiin ensin GeoPandasin GeoSeries-muotoon käyttäen oikeaa koordinaattireferenssisysteemiä (eng. Coordinate Reference System, CRS), joka HSL:n aineistossa oli WGS 84. GeoSeries on datatyyppi, joka on suunniteltu sisältämään maantieteellisiä geometriaob-

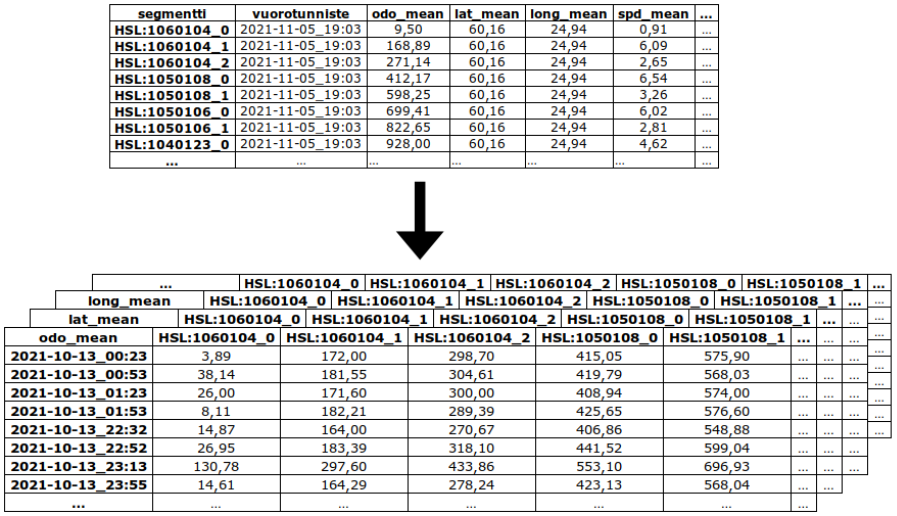
jekteja. Tämän jälkeen jokaiselle reitille ajettiin em. simplify-metodi, joka palauttaa yksinkertaistetun GeoSeries-objektin. Metodi käyttää algoritmia, joka jakaa alkuperäisten koordinaattipisteiden välissä olevan janan pienempiin osiin ja yhdistää sitten näiden osien päätepisteet suoralla viivalla. Sitten algoritmi poistaa kaikki pisteet, joiden etäisyys tähän suoraan viivaan on määriteltyä toleranssia (esim. 100 metriä) pienempi. Tällöin saadaan siis poistettua aineistosta ns. turhia koordinaattipisteitä, esimerkiksi niitä joissa on oltu pitkään liikennevaloissa eli koordinaatit ovat lähellä toisiaan mutta eivät sisällä sinänsä reitin kannalta arvokasta informaatiota. Tämän työn aineiston kanssa toleranssina käytettiin yhtä metriä, joka on sopiva kompromissi tarkkuuden ja yksinkertaistetun aineiston koon välillä.

Tämän jälkeen aineisto muokattiin muotoon, jossa jokainen ajettu reitti sisältää saman verran havaintopisteitä. Tämän voi tehdä kahdella eri tavalla, ajan tai matkan suhteen. Koska molemmat tavat tuovat mahdollisesti erilaisen näkymän aineistoon ja molemmat tavat toteutettiin. Käytännössä aineiston muokkaus toteutettiin niin, että jaettiin jokainen pysäkkiväli valitun mittaisiin segmentteihin joko ajan tai matkan suhteen. Lopputuloksena tuli siis kaksi vaihtoehtoista aineistoa. Tämän jälkeen segmenttien arvot aggregoitiin sarakkeelle soveltuvalla tavalla, esimerkiksi summana (esim. kulunut aika tai ajettu matka) tai keskiarvona (esim. nopeus tai sijaintikoordinaatti). Tällöin näistä segmenteistä saatiin jokaiselle pysäkkivälille yhtä monta havaintopistettä. Tätä tehdessä huomattiin myös, että aineistossa olevaan pysäkkitietoon ei välttämättä voinut jokaisessa tapauksessa luottaa, joten rakennettiin vielä pysäkkien koordinaattien perusteella metodi, joka paikansi vuoron aineistoriveistä aina sen havainnon, joka oli lähimpänä kutakin pysäkkiä. Tätä havaintoa pidettiin aina pysäkkitietona, joiden välille segmenttijaot sitten rakennettiin. Prosessi löytyy kuvattuna kuvasta 6. Rajallisen ajan takia ajan suhteen työn kokeissa käytettiin lopulta vain matkan suhteen tehtyä aineistoa.



Kuva 6: Aineiston reittien muokkaus yhtä pitkäksi. Eri vuoroissa pysäkkien välillä on eri määriä havaintoja, tavoite on koostaa pysäkkien välit segmenteiksi, joita jokaisen vuoron tietyssä pysäkkivälissä olisi yhtä monta.

Transformaation jälkeen aineiston jokainen ajettu reitti sisältää saman määrän segmenttejä, eli se saadaan yhdenmukaistettua ja voidaan analysoida käyttäen laajempaa skaalaa menetelmiä. Kun aineisto pivotoidaan (selventävä esitys kuvassa 7), voidaan nyt muodostaa jokaiselle mittaussarakkeelle muoto, jossa analysointiin voidaan käyttää laajemmin erilaisia anomalioiden tunnistamismenetelmiä. Saatu aineisto on tavallaan kolmiulotteinen, jossa havaitut dimensiot ovat vuoron ja segmentin lisäksi kolmas ulottuvuus, joka kerää aggregoidut mittaukset.



segmentti	vuorotunniste	odo_mean	lat_mean	long_mean	spd_mean	...
HSL:1060104_0	2021-11-05_19:03	9,50	60,16	24,94	0,91	...
HSL:1060104_1	2021-11-05_19:03	168,89	60,16	24,94	6,09	...
HSL:1060104_2	2021-11-05_19:03	271,14	60,16	24,94	2,65	...
HSL:1050108_0	2021-11-05_19:03	412,17	60,16	24,94	6,54	...
HSL:1050108_1	2021-11-05_19:03	598,25	60,16	24,94	3,26	...
HSL:1050106_0	2021-11-05_19:03	699,41	60,16	24,94	6,02	...
HSL:1050106_1	2021-11-05_19:03	822,65	60,16	24,94	2,81	...
HSL:1040123_0	2021-11-05_19:03	928,00	60,16	24,94	4,62	...
...	...	...	...	...	...	...

...	HSL:1060104_0	HSL:1060104_1	HSL:1060104_2	HSL:1050108_0	HSL:1050108_1	...
long_mean	HSL:1060104_0	HSL:1060104_1	HSL:1060104_2	HSL:1050108_0	HSL:1050108_1	...
lat_mean	HSL:1060104_0	HSL:1060104_1	HSL:1060104_2	HSL:1050108_0	HSL:1050108_1	...
odo_mean	HSL:1060104_0	HSL:1060104_1	HSL:1060104_2	HSL:1050108_0	HSL:1050108_1	...
2021-10-13_00:23	3,89	172,00	298,70	415,05	575,90	...
2021-10-13_00:53	38,14	181,55	304,61	419,79	568,03	...
2021-10-13_01:23	26,00	171,60	300,00	408,94	574,00	...
2021-10-13_01:53	8,11	182,21	289,39	425,65	576,60	...
2021-10-13_22:32	14,87	164,00	270,67	406,86	548,88	...
2021-10-13_22:52	26,95	183,39	318,10	441,52	599,04	...
2021-10-13_23:13	130,78	297,60	433,86	553,10	696,93	...
2021-10-13_23:55	14,61	164,29	278,24	423,13	568,04	...
...	...	...	...	...	...	...

Kuva 7: Aineiston muokkaus lopulliseen muotoon. Yllä segmenttikohtaisesti aggregoitu aineisto, joka pivotoidaan alla olevaan muotoon per jokainen aggregoitu sarake. Lopputulos on tavallaan kolmiulotteinen matriisi, jossa x-akseli on vuoro, y-akseli on reittisegmentti ja z-akseli on aggregoidut mittaukset.

Transformaation jälkeen lopullisessa muodossa olevia aineistoja on kaksi, “matka” ja “aika”. Aineisto-objekti sisältää myös molemmille jonkin verran metadataa, eli dataa joka on transformaatiossa poistunut varsinaisesta aineistotaulukosta. Aineiston matkaosassa on lopulta 95 segmenttiä, eli pysäkkien välit jaettiin segmentteihin noin 100 m välein. Aika-osassa taas on 65 segmenttiä, tässä pysäkkivälit jaettiin osiin 60 sekunnin välein. Kokonaisuudessa pysäkkejä reitillä on 25, mutta aineistossa ensimmäistä pysäkkiä (Eira; HSL:1060101) ei ole lainkaan, koska aineisto muodostettiin siten, että pysäkki- ja segmenttitieto on aina “se mihin päädyttiin”, eli lähtöpysäkki ei aineistossa silloin näy. Vuoroja lopullisessa aineistossa on aineiston siivoamisen jälkeen 2199. Segmenttivälit 100 m ja 60 s valittiin, jotta segmenttejä tulee riittäväksi arvioitu määrä.

Muokatussa aineistossa oli 70 aggregoitua mittaussaraketta ja ne ovat kuvattuna liitteessä 6.2.

4.2 Menetelmät

4.2.1 Laskuympäristö

Kaikki menetelmien ajot ja analyysit suoritettiin käyttäen työkaluina Pythonin 3.10.4 versiota, Jupyter Labin työskentely-ympäristöä sekä sopivia kirjastoja scikit-learnin ja Scipyn kirjastokokoelmista. Tarvittaessa käytettiin muita soveltuvia kirjastoja ja jos niitä käytettiin jossain oleellisen tärkeässä roolissa, se mainitaan erikseen.

4.2.2 Tunnistusmenetelmien valinta

Menetelmien valintaan vaikuttaa aineiston muoto ja tyyppi, sekä myös anomaliat, joita halutaan tunnistaa. Koska työssä ollaan kiinnostuneita kaikista anomalioista ja havaitut muuttujat sisältävää ohjattuun tunnistamiseen tarvittavaa opetusaineistoa ei ole, anomalioita tunnistetaan ohjaamattomasti. Tarkoituksena on siis testata muutamaa soveltuvaa menetelmää aineistoon.

Menetelmiksi valittiin esiteltyjen menetelmien joukosta tilastollisiin menetelmiin perustuvat Z-arvo sekä Mahalanobiksen etäisyys. Sekoitemallia ei valittu, koska sen ei nähty soveltuvan aineistolle riittävän hyvin ollakseen sopiva. Läheisyyteen perustuvista malleista valittiin K-means, lähinaapurimenetelmä sekä hierarkkinen klusterointi. Tiheyteen perustuvista valittiin DBSCAN ja LOF.

Tarkoituksena on vertailla muutamaa menetelmää aineistoon, sekä kertoa miten menetelmiä käytettiin ja minkälaisia tuloksia menetelmillä tuli. Koitetaan myös määritellä jonkinlaista hyvää tapaa menetelmien tulosten keskinäiseen vertailuun.

Ohjaamattomien anomaliantunnistusmenetelmien tulosten arviointi on vaikeaa. Anomaliat ovat jo määritelmänsä mukaan harvinaisia ja entistä hankalammaksi arvioinnin tekee se, jos menetelmät ovat ohjaamattomia, eli ns. oikeaa vastausta ei ole vertailukohdaksi. Menetelmien hyvyyttä pyritään vertaamaan niiden tulosten samankaltaisuudella, eli oletetaan, että jos menetelmät määrittävät samat vuorot tai segmentit anomalioiksi, ne voisivat olla anomalioita.

Kun anomalioita tunnistetaan, tulokset voivat olla yhdessä kolmesta ryhmästä: oikea tunnistus, väärä positiivinen tai väärä negatiivinen. Oikean tunnistuksen tapauksessa menetelmä tunnistaa anomalian oikein, väärässä positiivisessa menetelmä luokittelee anomaliaksi pisteen, joka ei tällainen ole, ja väärässä negatiivisessa olemassa ollut anomalia jää tunnistamatta.

4.2.3 Aineiston hienosäätö

Aineistoksi valittiin siis rajallisen ajan takia matkan perusteella segmentteihin jaettu vaihtoehto. Ennen kuin varsinaisia anomaliantunnistusmenetelmiä päästään käyttämään, se muokataan vielä Pandas-kirjaston Dataframe-muodosta tunnistusmenetelmille suoraan kelpaavaksi kolmiulotteiseksi Numpy-matriisiksi. Lisäksi siitä karsitaan vielä selittäviä tekijöitä, jotta esimerkiksi muuttujien seassa ei ole molempia mediaania ja keskiarvoa. Lopuksi jäljelle jäi 22 selittävää tekijää, eli yhteensä 48

pudotetaan pois ennen analyysien ajoa. Jäljelle jääneet sarakkeet kuvataan taulukossa 2. Aineisto on nyt siis kolmiulotteisessa Numpy-arrayssa, jossa x-suunnassa ovat vuorot, y-suunnassa pysäkkisegmentit ja z-suunnassa selittävät tekijät. Dimensioiltaan rakenne on lopulta $2199 \times 95 \times 22$ (N × A × C).

Sarakkeen nimi	**Sarakkeen kuvaus**
id_count	Segmentin aggregoitujen rivien määrä
acc_mean	Segmentin kiihtyvyys, keskiarvo
acc_max	Segmentin kiihtyvyys, maksimi
dl_mean	Segmentin aikataulupoikkeama, keskiarvo
dl_max	Segmentin aikataulupoikkeama, maksimi
matkamittari_mean	Segmentin kuljettu matka, keskiarvo
matkamittari_max	Segmentin kuljettu matka, maksimi
lat_mean	Segmentin leveyspiiri, keskiarvo
lat_max	Segmentin leveyspiiri, maksimi
long_mean	Segmentin pituuspiiri, keskiarvo
long_max	Segmentin pituus, maksimi
spd_min	Segmentin nopeus, minimi
spd_mean	Segmentin nopeus, keskiarvo
spd_max	Segmentin nopeus, maksimi
hdg_mean	Segmentin ajosuunta, keskiarvo
tsi_cum_mean	Segmentin ajoon käytetty aika, keskiarvo
tsi_mean	Segmentin UNIX-aikaleima, keskimääräinen (myöhemmin todettiin turhaksi)
seg_times_segmentti_matka_mean	Segmentin pituus (matka), keskimääräinen
seg_times_segmentti_aika_mean	Segmentin pituus (aika), keskimääräinen
tsi_diff_sum	Segmentin peräkkäisten aikaleimojen erotusten summa

Taulukko 2: Lopullisen kolmiulotteisen Numpy-aineistomatriisin sarakkeet ja niiden kuvaukset.

Koska mikään menetelmä ei tue tämän kaltaista kolmiulotteista rakennetta, luodaan vielä aineiston käsittelyyn menetelmä jolla voidaan lohkoa kolmiulotteisesta matriisista erilaisia "levyjä", esimerkiksi jokainen segmentti erikseen, jolloin otetaan x:n suunnassa kaikki, z:n suunnassa kaikki, mutta y-suunnassa vain yksi lohko.

4.2.4 Tulosten muoto

Aineistorakenteen koko on $2199 \times 95 \times 22$ ja Z-scorea lukuun ottamatta kaikki menetelmät toimivat vain kahdessa ulottuvuudessa ($2199 \times 1 \times 22$) ja Z-score vain yhdessä ($2199 \times 1 \times 1$). Analyysit tehdään pääsääntöisesti niin, että tarkastellaan vuorotellen jokaista yksittäistä pysäkkisegmenttiä ja suoritetaan anomalioiden etsintä erikseen jokaisen segmentin sisällä. Poikkeuksena Z-score, jossa etsintä tehdään

jokaisen segmentin jokaisen selittävän muuttujan sisällä. Tämä tarkoittaa sitä, että pitää päättää, millaisessa muodossa tulos halutaan.

Määritellään, että tässä työssä tärkein tulos on saada tieto siitä, onko koko vuoro kaikkine pysäkkisegmentteineen anomalia vai ei. Tämän lisäksi menetelmät kertovat kuitenkin myös, että onko joku tietty pysäkkiväli ollut anomalia vai ei. Nämä molemmat tulokset saadaan suoraviivaisesti käymällä menetelmien kanssa jokainen segmenttiväli erikseen läpi, ja Z-scoren tapauksessa myös muuttujat. Tällöin saadaan em. pysäkkisegmenttianomaliatieto ja tiivistämällä se vuorokohtaiseksi ja valitsemalla joku cutoff-arvo (kynnysarvo), saadaan vuorotulos. Tiivistäminen vuorokohtaiseksi tapahtuu aggregoimalla segmenttien anomaliatulos 2199×95 vektorista 2199×1 vektoriksi summaamalla segmenttien yli ja Z-scoren tapauksessa myös muuttujien yli. Tämä 2199×1 vektori saadaan binärisoitua, kun valitaan joku sopiva cutoff-arvo. Tällöin cutoff-arvon alittavat merkitään nollaksi, eli ne eivät ole anomalioita ja cutoff-arvon ylittävät merkitään ykköseksi, eli ne ovat anomalioita.

4.2.5 Oletettu anomalioiden määrä

Koska menetelmissä on jonkin verran parametri- ja cutoff-arvojen valintaa, pitää miettiä, kuinka paljon anomalioita aineistossa saattaisi olla. Lähdetään oletuksesta, että 2199 vuorosta noin 20 % voisi olla anomalioita. Tähän lukemaan päästään olettamalla, että vuorokaudessa on 2 ruuhka-aikaa ja ne kestävät noin 2 tuntia. Ruuhka-ajat lienevät myös suurin anomalioita aiheuttava syy, jolloin lienee perusteltua olettaa, että tämä näkyisi aineistossakin. Tällöin siis jokaisen menetelmän tulisi antaa noin 400 anomaliaa 2199 vuoron kokoisesta aineistosta. Käytetään tätä menetelmien parametrien säädössä sekä cutoff-arvojen valinnoissa.

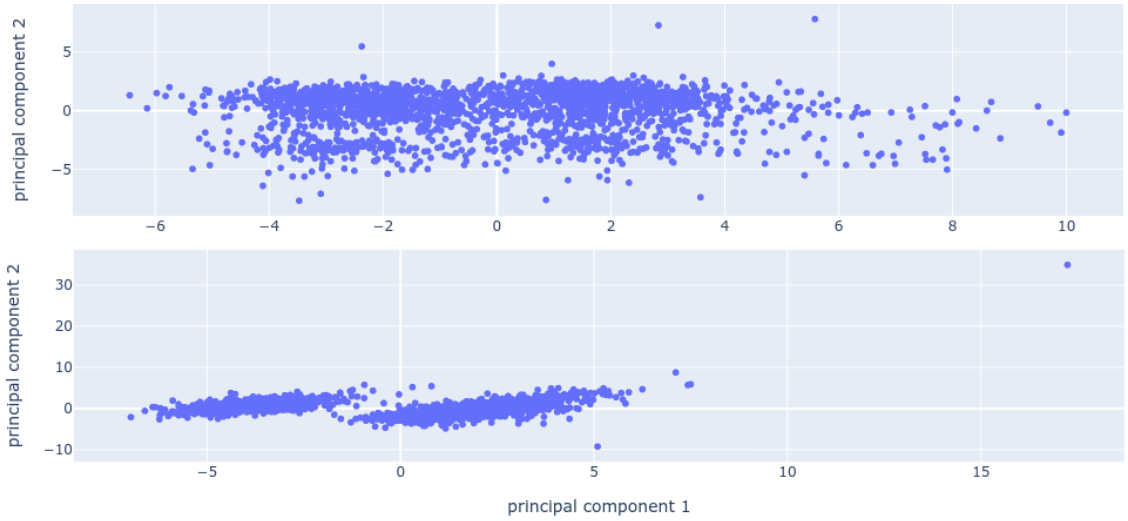
4.2.6 Aineiston pääkomponenttianalyysi

Aivan aluksi haluttiin yrittää visualisoida havaintoja, jotta saataisiin mahdollisesti tietoa aineiston rakenteesta, joka voisi auttaa analyysien tekemisessä tai tulosten tulkinnassa. Koska anomalian tunnistaminen on tavallaan luokittelua, valittiin visualisointitavaksi pääkomponenttianalyysi (PCA). Käytiin PCA:ta käyttäen läpi kaikki pysäkkisegmentit ja monesta oli erotettavissa kaksi ryhmää. Toisaalta myös, monesta segmentistä ryhmiä ei paljaalla silmällä pystynyt näkemään. Lisäksi kaksiulotteisilla kuvilla, eli kahdella ensimmäisellä pääkomponentilla, katettiin pysäkkien selittävien muuttujien varianssista keskimäärin vain noin 44 % ja parhaimmillaan 67 %.

Pääkomponenttianalyysi auttoi hieman hahmottamaan aineistoa, varsinkin ymmärtämään kaksiluokkaisuutta, joka aiheutuu hyvin suurella todennäköisyydellä joko arjen ja viikonloppujen eroista tai normaalin ja ruuhka-ajan eroista.

4.2.7 Z-testi

Z-testin toteutuksessa käytettiin scipy-kirjaston zscore-metodia. Tämä metodi ajettiin yksinkertaisesti jokaiselle aineistomatriisin x-suuntaiselle $2199 \times 1 \times 1$ sarak-



Kuva 8: Esimerkit tehdystä PCA-analyysista. Ylemmässä kuvassa on pysäkkivälin 1050106 ensimmäinen segmentti, joka on reitin alkupuolella. Selitetyn varianssin määrä on 48 % ja kuvassa ei ole nähtävissä kahta selkeästi erilaista ryhmää, toisaalta anomalioita voi olla näkyvissä. Alemmassa kuvassa on pysäkkivälin 1304157 kolmas segmentti. Tässä on nähtävissä varsin siististi kaksi klusteria, toisaalta myös anomalioita. Alemmassa kuvassa selitetyn varianssin määrä on 53 %.

keelle erikseen. Tuloksena saatiin dimensioiltaan aineistomatriisia vastaava Z-score-matriisi, jossa jokainen arvo oli nyt sen z-arvo tämän yksittäisen sarakkeen sisällä. Anomalioiksi luokiteltiin kaikki havainnot, joiden Z-scoren arvo oli välin (-4, 4) ulkopuolella. Tämän jälkeen välituloksena oli matriisi, joka oli edelleen kokoa $2199 \times 95 \times 22$, mutta nyt jokaiselle solulle on merkattu, oliko se anomalia (1) vai ei (0). Koska tulokseksi halutaan pysäkkisegmenttikohtaiset tulokset, sekä vuorokohtaiset tulokset, summataan ensin z-akselin suunnassa ja asetetaan sopivalla cutoff-arvolla pysäkkisegmentti anomaliaksi. Tästä jatketaan summaamalla myös y-suunta ja valitsemalla taas sopiva cutoff-arvo, jolloin saadaan lopulta vuorokohtaiset anomaliatiedot. Cutoff-arvon valinnassa käytetään aikaisemmin määriteltyä 20 % määrää.

4.2.8 Mahalanobiksen etäisyys

Mahalanobiksen etäisyys toistaa suurilta osin Z-score-menetelmän toimintatapaa. Ainut iso ero on se, että Mahalanobis käyttää kahta ulottuvuutta, eli jokainen pysäkkisegmentti käsitellään edelleen erikseen, mutta kaikki muuttujat huomioidaan samanaikaisesti. Saadaan tulokseksi 2199×95 matriisi, jossa jokainen solu on binäärinen anomaliatieto. Etäisyyden laskemiseen ja tunnistamiseen käytetään rakennettua metodia, joka laskee etäisyydet ja määrittää anomaliatiedon käyttäen khiin neliö -jakaumaa parametrilla 95 %. Lopulta vuorokohtaisen anomaliatiedon saamiseen laskettiin vielä 2199×95 pysäkkitulosmatriisin y-suuntainen rivisumma ja valittiin taas sopiva 20 % cutoff-arvo vuorojen osoittamiseen anomalioiksi vuorotulosmatrii-

siin (2199×1).

Menetelmää käyttäessä huomiona nousi esiin se, että käytössä oleva aineisto tuotti kovarianssimatriisiksi singulaarisen matriisin. Ongelma ratkaistiin vaihtamalla normaali kääntöfunktio Moore-Penrose-pseudo-käänteisfunktioksi.

4.2.9 K-means-klusterointi

Valittiin käytettäväksi kirjastoksi sklearn.cluster.KMeans. Käytettiin myös skaalaukseen StandardScaler-toimintoa skaalaamaan aineisto nollakeskiarvoiseksi ja yksikkövarianssiin. Ensimmäiseksi K-means-klusterointia tehdessä, valittiin arvo parametrille k. Oletus oli, että voitaisiin valita arvo 2, mutta tämä varmistettiin vielä muutamalla testillä ja näiden perusteella $k = 2$ oli hyvä ja sopiva valinta, muut parametrit pidettiin oletusarvoissa. K-means algoritmi suoritettiin taas jokaiselle pysäkille erikseen ja saaduista klustereista pienempi asetettiin aina anomaliaklusteriksi ja isompi normaaliklusteriksi. Saadusta pysäkkitulosmatriisista (2199×95) muodostettiin taas binäärinen vuorotulosmatriisi (2199×1) summaamalla vuorojen sarakkeet yhteen ja käyttämällä sopivaa 20 % mukaan laskettua cutoff-arvoa.

4.2.10 Lähinaapurimenetelmä

Valittiin käytettäväksi kirjastoksi sklearn.neighbors.NearestNeighbors. Myös lähinaapurimenetelmän kanssa käytettiin skaalaukseen StandardScaleria. Ensimmäiseksi K-means-klusterointia tehdessä haarukoitiin arvo lähinaapurien määrälle ja päädyttiin laskenta-ajan optimoimiseksi käyttämään määrää 0.2N. Tällä ajettiin menetelmä pysäkki kerrallaan ja laskettiin 20 % mukaan jokaiselle pysäkkisegmentille oma cutoff-arvo, eli haettiin arvo jolla kussakin segmentissä tunnistettiin noin 20 % vuoroista anomalioiksi. Tämän menetelmän kanssa sen ajateltiin tuovan ainakin näennäisesti lisää tarkkuutta, koska cutoff-arvo vaihteli runsaasti pysäkkisegmenteittäin.

Tämän jälkeen laskettiin jokaiselle vuorolle per pysäkkisegmentti lähinaapurit, otettiin etäisyyksien keskiarvo ja saatuja arvoja verrattiin cutoff-arvoon per pysäkkisegmentti. Cutoff-arvon yli menneet merkittiin binaarisesti anomalioiksi (1) ja normaalihavainnoiksi (0). Lopuksi tiivistettiin summaamalla pysäkkianomalia-arvot (2199×95) vielä vuorotulosmatriisiksi (2199×1) 20 % mukaan valitulla cutoff-arvolla.

4.2.11 Hierarkkinen klusterointi

Valittiin käytettäväksi kirjastoksi scipy.cluster.hierarchy. Myös tämän menetelmän kanssa käytettiin StandardScaleria. Ensimmäiseksi määriteltiin sopiva katkaisukohdan tason p-parametrin arvo. Tällä p-parametrilla tarkoitetaan scipy.cluster-paketin hierarchy.dendrogram menetelmän katkaisuparametria, joka määrittää maksimimäärän tasoja, jotka säilytetään katkaisussa. Tämä p-parametrin arvo valittiin siten, että hierarkkinen klusterointi leikattiin niin, että noin 20 % havainnoista on omassa

isoimmassa klusterissaan ja muut sitten ominaan (tai pienemmissä ryhmissä). Arvoksi valittiin $p = 200$. Näin saatiin pysäkkianomaliamatriisi ja taas tämä sopivalla cutoff-arvolla tiivistämällä saatiin vuorotulosmatriisi.

4.2.12 DBSCAN

Valittiin käytettäväksi kirjastoksi sklearn.cluster.DBSCAN. Myös tämän menetelmän kanssa käytettiin StandardScaleria. Ensimmäiseksi määriteltiin sopivat arvot parametreille eps, joka on kahden vierekkäisen samaan naapurustoon laskettavan havainnon minimietäisyys (r kappaleen 3.3.4 merkinnöillä), ja min_samples, joka määrittää ydinpisteet naapurustossa (θ kappaleen 3.3.4 merkinnöillä). Näidenkin määrittämisessä käytettiin taas 20 % määrää. Arvoiksi saatiin eps = 2.6 ja min_samples = 394. Algoritmi ajettiin taas pysäkkisegmenttikohtaisesti jokaiselle segmentille ja tulokset muunnettiin binaariseksi. Tästä muodostui pysäkkianomaliamatriisi ja taas sopivalla cutoff-arvolla tiivistämällä saatiin muodostettua vuorotulosmatriisi.

4.2.13 LOF

Valittiin käytettäväksi kirjastoksi sklearn.neighbors.LocalOutlierFactor. Myös tämän menetelmän kanssa käytettiin StandardScaleria. Ensimmäiseksi määriteltiin sopiva arvo parametrille n_neighbors, joka on käytetty lähinaapureiden määrä. Muut parametrit pidettiin oletusarvona. Parametrille saatiin arvoksi n_neighbors=180 ja sitä käyttäen muodostettiin pysäkkisegmenttikohtaisesti ajamalla pysäkkianomaliamatriisi. Lopuksi sopivalla cutoff-arvolla tiivistämällä saatiin muodostettua vuorotulosmatriisi.

4.3 Tulokset

Tuloksissa keskitytään enemmän anomaliaksi tunnistettuihin vuoroihin, mutta myös vuorosegmenttien anomalioita käsitellään lyhyesti. Tuloksia analysoidessa keskitytään siihen, merkitsevätkö menetelmät samoja vuoroja ja pysäkkisegmenttejä anomalioiksi.

4.3.1 Vuorotulokset

Kaikki menetelmät löysivät aineistosta noin 300-400 anomaliavuoroa, mikä on tietenkin luonnollista, koska katkopisteiden valinnalla tähän pyrittiin. Kuitenkin tuloksia tarkemmin katsoessa, arkivuorojen anomaliat olivat selvästi yliedustettuna kaikissa menetelmissä. Tämä voi selittyä sillä, että arkipäiviä on myös suunnilleen samassa suhteessa. Nyt jälkikäteen katsoessa, arki- ja viikonloppujen vuorot olisi kannattanut analysoida erikseen, jolloin tuloksista olisi varmasti tullut parempia. Ajanpuutteen vuoksi tämä piti kuitenkin rajata pois ja jättää tulevaisuudessa tutkittavaksi. Menetelmien tulokset ovat nähtävissä kuvan 9 taulukossa.

	Tunnistettuja anomaloita			Suhde	
	Yht.	Arki	Vkl	Arki	Vkl
Z-Score	**415**	333	82	80,2 %	19,8 %
Mahalanobis	**413**	344	69	83,3 %	16,7 %
K-means	**312**	265	47	84,9 %	15,1 %
Lähinaapuri	**409**	330	79	80,7 %	19,3 %
Hierark. Klusterointi	**439**	357	82	81,3 %	18,7 %
DBSCAN	**377**	295	82	78,2 %	21,8 %
LOF	**344**	273	71	79,4 %	20,6 %
Arjen ja viikonlopun suhde koko aineistossa				75,5 %	24,5 %

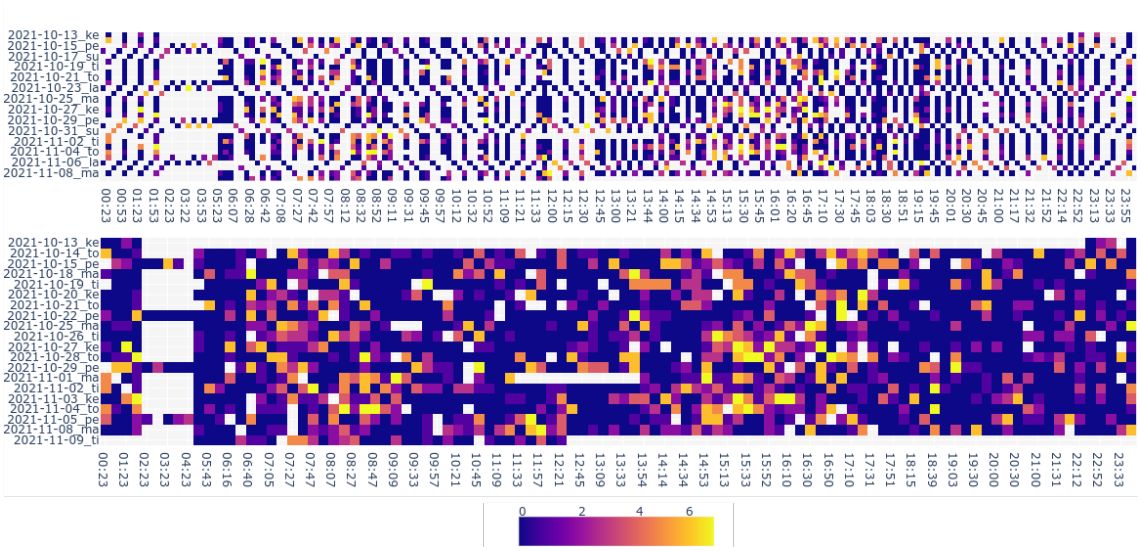
Kuva 9: Taulukko sisältää menetelmien löytämät anomaliavuorot jaoteltuna arjen- ja viikonlopun vuoroihin. Esitettynä on myös näiden suhde aineistossa. Tunnistettujen anomalioiden määrä saadaan käyttämällä summaamalla aggregoitua 2199×1 tulosvektoria, jossa jokainen vuoro on merkitty joko anomaliaksi (1) tai anomaliattomaksi vuoroksi (0). Tunnistettujen anomalioiden määrä on siis tämän 2199×1 vektorin summa, eli arvojen 1 määrä.

Tulokset voidaan esittää matriisina siten, että jokainen aineiston vuoro on pikseli kuvassa (kuva 10, yläosa). Lasketaan menetelmien anomalia-arvojen summa jokaiselle vuorolle, jolloin kaikkien menetelmien normaaliksi toteama havainto on arvoltaan 0 ja kaikkien menetelmien anomaliaksi tunnistama menetelmä on arvoltaan 7. Nämä voidaan vielä värikoodata, jolloin useamman menetelmän anomaliaksi tunnistama vuoro on kirkkaankeltainen ja anomaliattomaksi tunnistettu vuoro tummanvioletti. Kuvassa on joitain aukkoja puuttuvien vuorojen takia.

Nähdään selvästi, että menetelmät ovat tunnistaneet varsin hyvin ruuhka-aikojen anomalioita sillä aikaväleillä 06-08 sekä 15-17 on reilusti muuta aikaa enemmän anomalioita. Koska viikonlopun vuorot ovat hieman erilaisia, suodatetaan ne pois ja saadaan hieman selvempi kuva arkivuoroista (kuva 10, alaosa). Kuvien perusteella voidaan todeta, että menetelmät antavat ainakin yhdessä jotakuinkin järkevää tulosta.

Katsotaan tarkemmin, miten menetelmät tunnistivat eri vuorot anomaliavuoroiksi. Kuvassa 11 on jakauma tunnistamisten määrästä. 1280 vuoroa ei saanut miltään menetelmältä yhtään tunnistusta ja toisessa ääripäässä 42 vuoroa tunnistettiin anomaliaksi kaikkien seitsemän menetelmän perusteella. 222 vuoroa olivat sellaisia, jotka vähintään 5 menetelmää tunnistivat anomaliaksi. Jos mietitään, kuinka monen menetelmän tulisi vuoro merkata anomaliaksi, jotta sen voisi suuremmalla varmuudella todeta anomaliaksi, voidaan kysymystä lähestyä jo analyysivaiheessa käytettyä 20 % arvoa hyväksi käyttäen. Jotta 2199 vuorosta saataisiin merkittyä noin 20 prosenttia anomalioiksi, pitäisi anomaliavuoroiksi määrittää kaikki, jotka on tunnistettu anomaliaksi vähintään kolmen (451) tai neljän (332) menetelmän kanssa. Jonkinlaista varmuutta voidaan kuitenkin ajatella olevan ainakin siinä, että jopa 1280 vuoron kohdalla kaikki menetelmät olivat sitä mieltä, ettei kyseessä ollut anomalia.

Kuvassa 12 on taulukko, jossa kaikkia menetelmiä on verrattu keskenään. Tällä voidaan koittaa selvittää sitä, miten yhteneviä anomaliatuloksia menetelmät antavat. Kaiken kaikkiaan keskimäärin vain puolet kahden menetelmän anomaliaksi tunnistamista vuoroista on molempien menetelmien tunnistamia. Poikkeuksiakin kuitenkin on, eniten samoja vuoroja anomalioiksi tunnistivat lähinaapurimenetel-

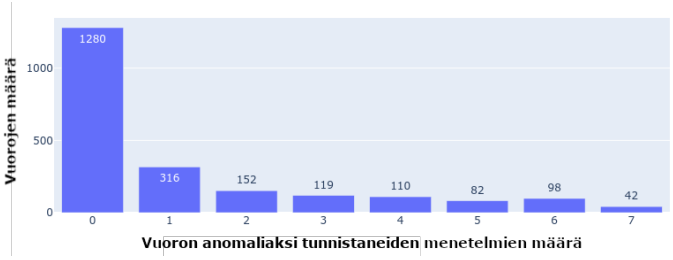


Kuva 10: Kuvassa x-akseli on vuorokaudenaika ja y-akseli on ajetun vuoron päivämäärä. Kuvassa on summa kaikkien vuorojen kaikkien menetelmien anomalia-arvoista. Yksi solu on yksi vuoro ja sen arvo kertoo, kuinka moni menetelmistä tunnisti kyseisen vuoron anomaliaksi. Maksimiarvo on 7 ja minimi on 0. Ylemmässä kuvassa on koko aineisto viikonloppuineen, alemmassa kuvassa pelkkä arki ilman viikonloppuja.

mä ja hierarkkinen klusterointi, yhteensä 365 samaa vuoroa, ja myös Lähinaapuri ja DBSCAN pääsivät kolmeen sataan samaan tunnistamiseen. Näille menetelmäpareille samoja tunnistuksia tuli siis yli 80 %. Toisaalta taas, toisesta ääripäästä huonoja parivertailuvastaavuuksia löytyi enemmän. Näitä olivat esimerkiksi Z-Score ja K-means, Mahalanobis ja K-means sekä K-means ja LOF. Nämä antoivat vain noin 100 samaa anomaliatunnistusta ja samojen osuus oli siis heikoimmillaan vain 20-30 % menetelmien löytämistä anomalioista.

Kuvasta 13 nähdään vielä tarkemmin, miten menetelmät käyttäytyvät. Kuvassa on ensimmäiset 250 vuoroa visualisoituna siten, että jokaisessa viidestä päällekkäisestä kuvan osasta on 50 vuoroa siten, että rivit ovat menetelmät ja sarakkeet vuoroja. Keltaiseksi merkityt ruudut ovat rivin menetelmän anomaliaksi asettamia vuoroja ja tummansiniset anomaliattomia vuoroja. Kuvassa menetelmät ylhäältä alaspäin ovat Z-score, Mahalanobis, K-means, lähinaapuri, hierarkkinen klusterointi, DBSCAN ja LOF.

Keskitytään seuraavaksi 42 kaikkien menetelmien löytämään vuoroon. Koitetaan löytää syy siihen, miksi nuo merkittiin anomaliaksi, eli selvitetään olivatko ne todella anomalioita muihin lähes 1300 anomaliattomaan vuoroon verrattuna. Jotta nähdään, millaisia anomalioita nuo kaikkien menetelmien löytämät 42 vuoroa ovat, katsotaan ensin niitä karttamuodossa. Kuvassa 14 vasemmalla on kaikkien 42 vuoron reitti piirretty samaan kuvaan. Näistä vain yksi poikkeaa reitiltään jonkin verran ja sekin on selvästi jonkinlainen GPS-virhe, sillä reittijälki on saman muotoinen kuin muutkin, mutta se on noin 50 metriä sivussa. Tämän näkee tarkemmin vielä kuvan 14 oikealta puolelta, jossa on yksi satunnaisesti valittu toinen reitti sekä tämä si-



Kuva 11: Vuorojen anomaliaksi tunnistumismäärät. Kuva kertoo sen, kuinka monta menetelmää tunnisti kuinkakin monta vuoroa anomaliaksi. Esimerkiksi 1280 vuoroa oli kaikkien menetelmien mielestä anomaliattomia, kun taas vain 42 vuoroa oli kaikkien menetelmien mukaan anomaliavuoroja.

vussa oleva reitti. Näiden anomalioiksi tunnistettujen 42 vuoron seasta siis vain yksi erottuu muista katsomalla pelkästään karttaan piirtyvää reittiä.

Pystytäänkö siis jotenkin muuten selvittämään, mistä syystä juuri nämä 42 valikoituivat anomaliaksi? Yksi vaihtoehto on katsoa, eroavatko anomalioiksi tunnistettujen vuorojen mittaukset jotenkin niistä joita ei tunnistettu anomalioiksi. Taas keskitytään kaikkien menetelmien tunnistamaan 42 vuoroon. Ensimmäisenä katsotaan vuoron ajoon käytetyn ajan ja matkamittarilukeman kuvaajaa, joka löytyy kuvasta 15. Jo pikaisella vilkaisullakin on selvää, että ylemmässä 42 anomaliavuoroa sisältävässä joukossa on suurempi hajonta kuin alempana olevassa 42 satunnaisesti valittua anomaliatonta vuoroa sisältävässä joukossa. Satunnainen valinta toistettiin myös varmuuden vuoksi useamman kerran ja joka kerralla tulos oli sama. Tämän perusteella voidaan siis sanoa, että kaikki menetelmät ovat ihan oikein tunnistanut noissa jotain poikkeuksellista.

Vahvistusta saadaan myös katsomalla aineiston muuttujaa, joka kertoo suoraan ainakin jonkinlaisista anomalioista ja mittaa laskennallista myöhästymisaikaa jokaisella ajanhetkellä. Kuvan 16 yläosassa on taas anomaliavuorot ja alaosassa anomaliattomat vuorot, edelleenkin 42 kappaletta kumpaakin. On selvää, että anomaliavuorot ovat olleet enemmän myöhässä sekä myös etuajassa. Anomaliattomissa vuoroissa kuvaajien nippu on paljon selvemmin kasassa. Eli voidaan todeta, että menetelmät olivat löytäneet vuoroja, jotka ovat muista poikkeavia.

Lisää vahvistusta voidaan hakea vielä vertaamalla 42 anomaliaksi tunnistetun vuoron ja 42 anomaliattoman vuoron muuttujakeskiarvoja toisiinsa. Kuvassa 17 on vertailtu anomaliallisten keskiarvon suhdetta anomaliattomien keskiarvoon. Nähdään selvästi, että myöhästymiseen suoraan verrannolliset muuttujat keskimääräinen- ja maksimimyöhästymisaika sekä matkaan käytetty aika ovat anomaliallisilla suhteellisesti paljon suurempia, kun taas myöhästymiseen käänteisesti liittyvät muuttujat kuten minimi, keskiarvo ja maksiminopeudet ovat pienempiä.

Edelliset tarkastelut tehtiin 42 anomaliaksi määritellyllä ja satunnaisesti valitulla 42 anomaliattomalla vuorolla. Nämä 42 anomaliatonta vuoroa olivat siis jokaisen menetelmän anomaliaksi merkkaamia vuoroja. Lyhyen testin perusteella, tulos ei merkittävästi muuttunut, vaikka mukaan otettiin myös yhteensä 5 ja 6 menetelmän anomalioiksi merkitsemät vuorot. Jos aineistoa tutkitaan edelleen tulevaisuudessa,

		Määrät			Samojen suhde		
A	B	A	B	Samoja	A:sta	B:stä	Keskiarvo
Z-Score	Mahalonobis	415	413	263	0,63	0,64	0,64
Z-Score	K-means	415	312	93	0,22	0,30	0,26
Z-Score	Lähinaapuri	415	409	195	0,47	0,48	0,47
Z-Score	Hier. klusterointi	415	439	235	0,57	0,54	0,55
Z-Score	DBSCAN	415	377	159	0,38	0,42	0,40
Z-Score	LOF	415	344	250	0,60	0,73	0,66
Mahalonobis	K-means	413	312	104	0,25	0,33	0,29
Mahalonobis	Lähinaapuri	413	409	244	0,59	0,60	0,59
Mahalonobis	Hier. klusterointi	413	439	286	0,69	0,65	0,67
Mahalonobis	DBSCAN	413	377	187	0,45	0,50	0,47
Mahalonobis	LOF	413	344	265	0,64	0,77	0,71
K-means	Lähinaapuri	312	409	120	0,38	0,29	0,34
K-means	Hier. klusterointi	312	439	123	0,39	0,28	0,34
K-means	DBSCAN	312	377	126	0,40	0,33	0,37
K-means	LOF	312	344	75	0,24	0,22	0,23
Lähinaapuri	Hier. klusterointi	409	439	365	0,89	0,83	0,86
Lähinaapuri	DBSCAN	409	377	300	0,73	0,80	0,76
Lähinaapuri	LOF	409	344	230	0,56	0,67	0,62
Hier. klusterointi	DBSCAN	439	377	286	0,65	0,76	0,71
Hier. klusterointi	LOF	439	344	264	0,60	0,77	0,68
DBSCAN	LOF	377	344	171	0,45	0,50	0,48

Kuva 12: Menetelmien samoiksi tunnistamien vuorojen parivertailu. Suhteessa eniten samoja vuoroja anomalioiksi tunnistivat lähinaapurimenetelmä ja hierarkkinen klusterointi. Myös esimerkiksi lähinaapurimenetelmä sekä DBSCAN vastasivat suhteellisen hyvin toisiaan. Z-Score ja K-means, Mahalanobis ja K-means sekä K-means ja LOF antoivat eniten toisistaan poikkeavia tuloksia.

voisi yksi vaihtoehto olla selvittää mikä olisi tässä sopiva määrä tunnistamisia.

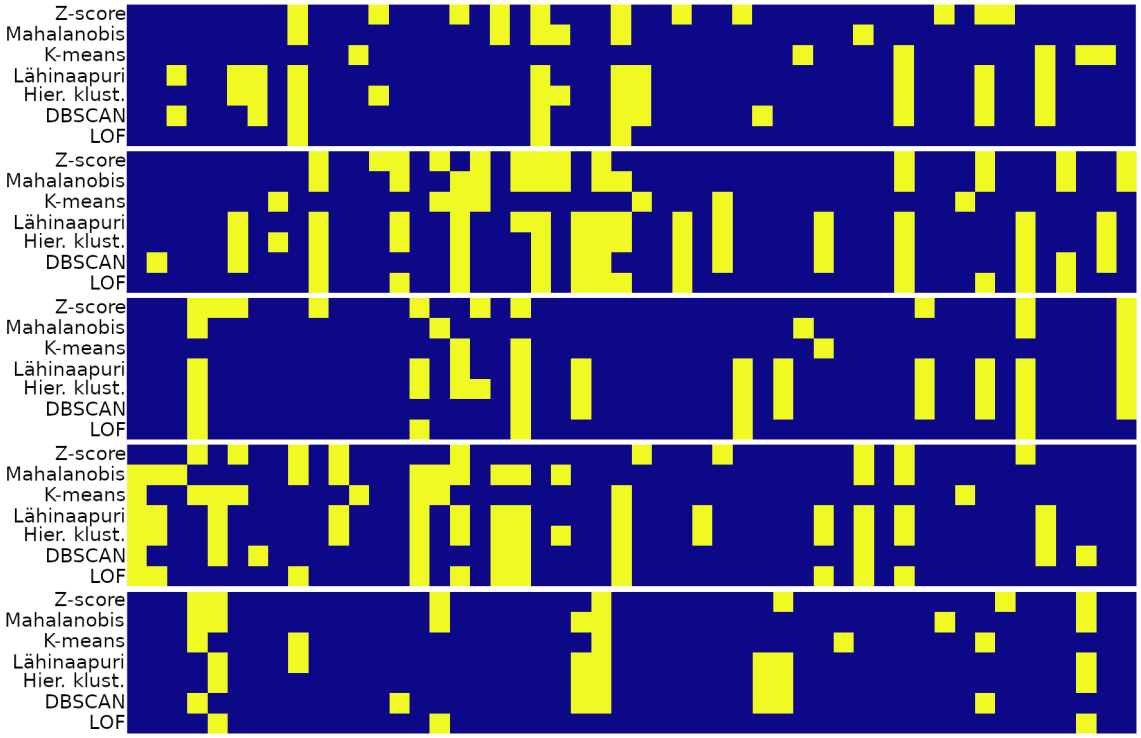
4.3.2 Pysäkkisegmenttitulokset

Käsitellään vielä lyhyesti pysäkkisegmenttien anomalioita, sillä käytetyt menetelmät tuottivat myös jokaisen vuoron jokaiselle pysäkkisegmentille anomalia-arvon. Kuvassa 18 on esitetty anomaliavuoroiksi tunnistettujen 42 vuoron segmentit (yllä) sekä satunnaisesti valittujen anomaliattomien vuorojen segmentit (alla). On luontevaa, että anomaliavuoroiksi tunnistetuilla vuoroilla on myös selvästi enemmän anomaliasegmenttejä kuin satunnaisesti valituilla. Tämä on luontevaa siksi, että käytetyn pysäkkisegmenttitulosten summaamalla tapahtuvan aggregoinnin myötä anomaliavuorot ovat vuoroja, joissa on eniten anomaliasegmenttejä. Myös tämä pysäkkisegmenttien tarkastelu vahvistaa menetelmien löytämien anomaliavuorojen tulosta.

Myös segmenttitasolla olisi helppo paneutua tuloksiin syvällisemmin, mutta se jääköön tulevien tutkimusten tehtäväksi.

4.4 Keskustelu ja pohdinta

Vaikka menetelmät valittiin pääasiassa mielenkiinnon vuoksi, oli silti mielenkiintoista nähdä, että menetelmät antoivat yllättävänkin hyviä tuloksia. Tavallaan hieman vahingossakin näistä rakennettiin ensemble. Tulevaisuuteen jätetään tutkittavaksi se, kuinka monen menetelmän täytyy pitää vuoroa anomaliana, jotta se kannattaa merkata anomaliaksi, sillä nyt tarkasteltiin vain kahta ääripäätä eli sitä kun mikään



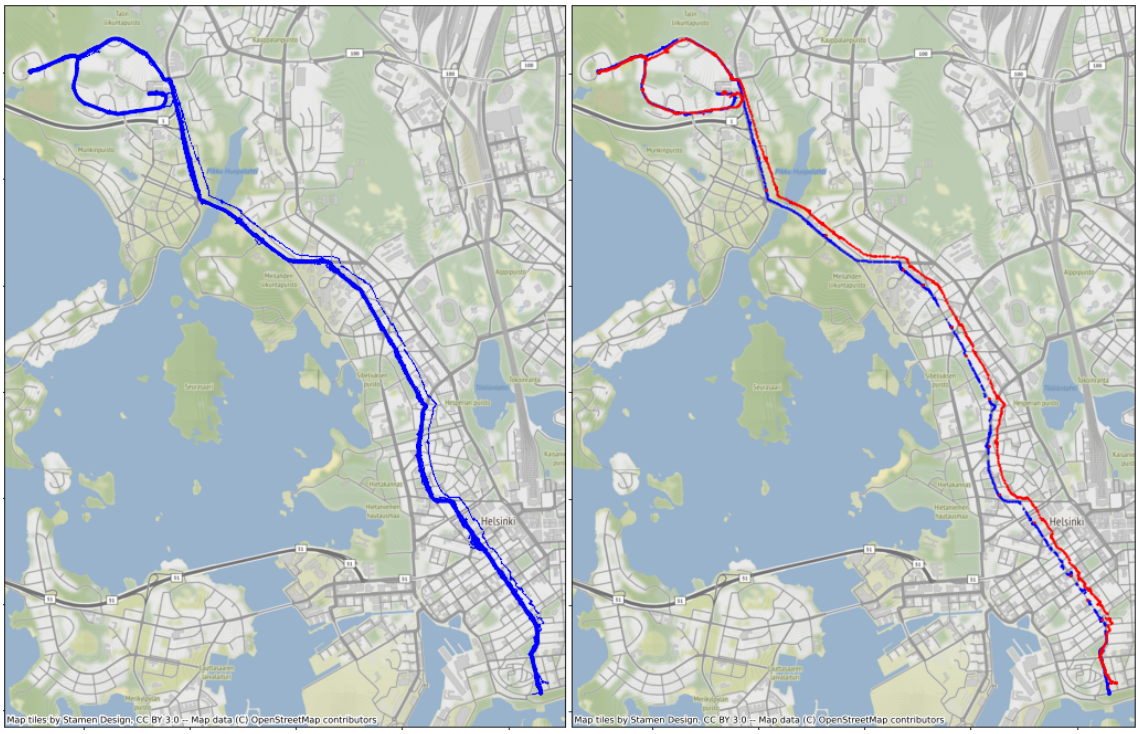
Kuva 13: Kuvassa on esitettynä aineiston ensimmäiset 250 vuoroa ja jokaisen menetelmän näille vuoroille antama anomalia-arvo. Keltainen on anomalia-arvo 1 ja sininen on 0. Kuvassa menetelmät ylhäältä alaspäin ovat Z-score, Mahalanobis, K-means, lähinaapuri, hierarkkinen klusterointi, DBSCAN ja LOF.

menetelmä ei merkannut vuoroa anomaliaksi ja sitä kun kaikki merkitsivät vuoron anomaliaksi.

Tarkkuutta olisi varmasti parantanut se, että arki- ja viikonloppuvuorot olisi analysoitu erikseen omina ryhminään. Ne ovat selvästi erilaisia luonteeltaan, arjessa on kaksi ruuhka-aikaa ja viikonloppuna ei, tai ainakin harvoin. Tätä oletusta ruuhka-ajoista käytettiin menetelmien tulosten kalibroinnissa ja cutoff-arvojen valinnassa, jolloin se vaikutti myös tuloksiin.

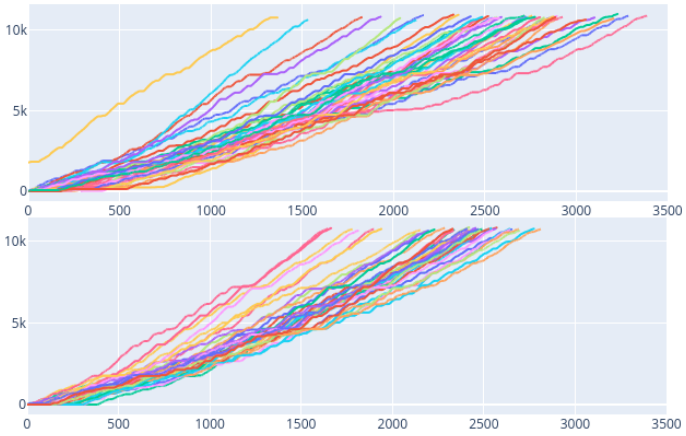
Lopulta, menetelmät tuntuivat löytävän eniten anomalioita, jotka liittyivät ajoaikaan. Ei liene yllätys, että linja-autot ovat joskus myöhässä ja voikin ajatella, että olisiko noiden anomalioiden löytämiseen ollut yksinkertaisempiakin menetelmiä. Toisaalta aineisto on laaja ja siihen löytäisi vielä monta erilaista näkökulmaa ja näillä sieltä voisi mahdollisesti löytää myös muunlaisia anomalioita. Alun perin tarkoitus oli löytää anomalioita, mutta myös näiden syitä, ja nimenomaan harvinaisempia syitä. Valitut menetelmät eivät tätä kuitenkaan tue riittävästi, että näistä voisi sanoa tarkemmin. Menetelmien käytön jälkeenkin ehkä selvin syy anomaliaksi merkkaamiseen oli myöhästyminen. Jatkotutkimuksissa voisi esimerkiksi käyttää selittävänä muuttujana vain nopeutta tai kiihtyvyyttä ja selvittää olisiko tällä vaikutusta tuloksiin.

Aineistoa myös suodatettiin ja gps-jälkiä tiivistettiin huomattavasti ennen mene-

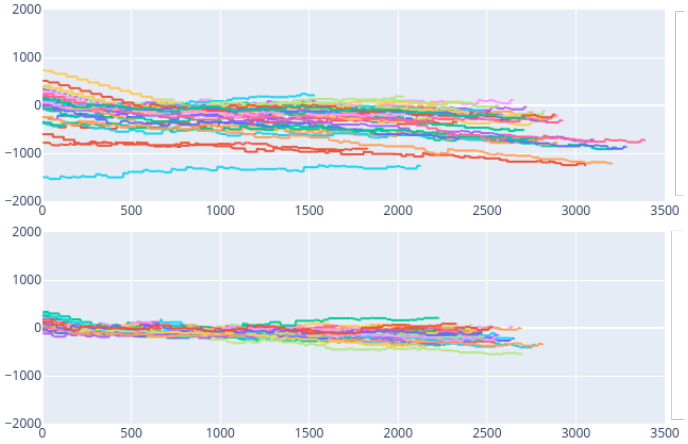


Kuva 14: Oikeanpuoleisessa kartassa on kaikki 42 jokaisen menetelmän anomaliaksi määrittämää vuoroa. Oikealla puolella on näiden joukosta löytynyt GPS-jäljeltään sivussa oleva, joka oli ainut joka pystyttiin reittijäljen perusteella huomaamaan anomaliaksi.

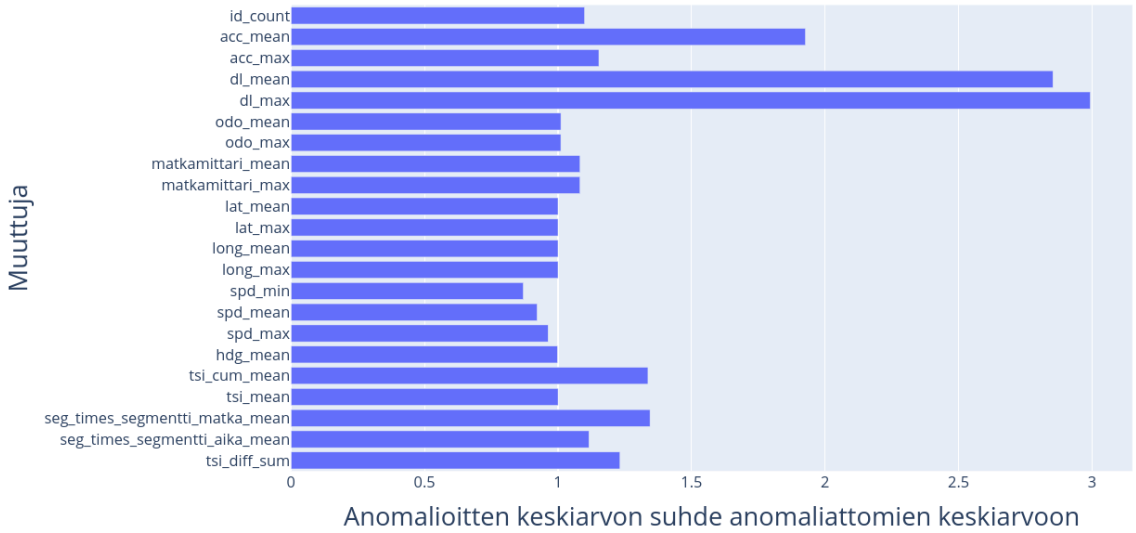
telmien käyttämistä laskenta-ajan säästämiseksi. Tämä olisi myös yksi testaamisen kohde, tulisiko tuloksista erilaisia, jos käytettäisiin koko aineistoa eli jokaisen vuoron jokaista mittapistettä. Toisaalta tässäkin tapauksessa tulisi löytää optimointeja, sillä tässä laajuudessa koko miljoonan rivin aineistoa ei pystyisi järkevässä ajassa käsittelemään siten kuin nyt tehtiin.



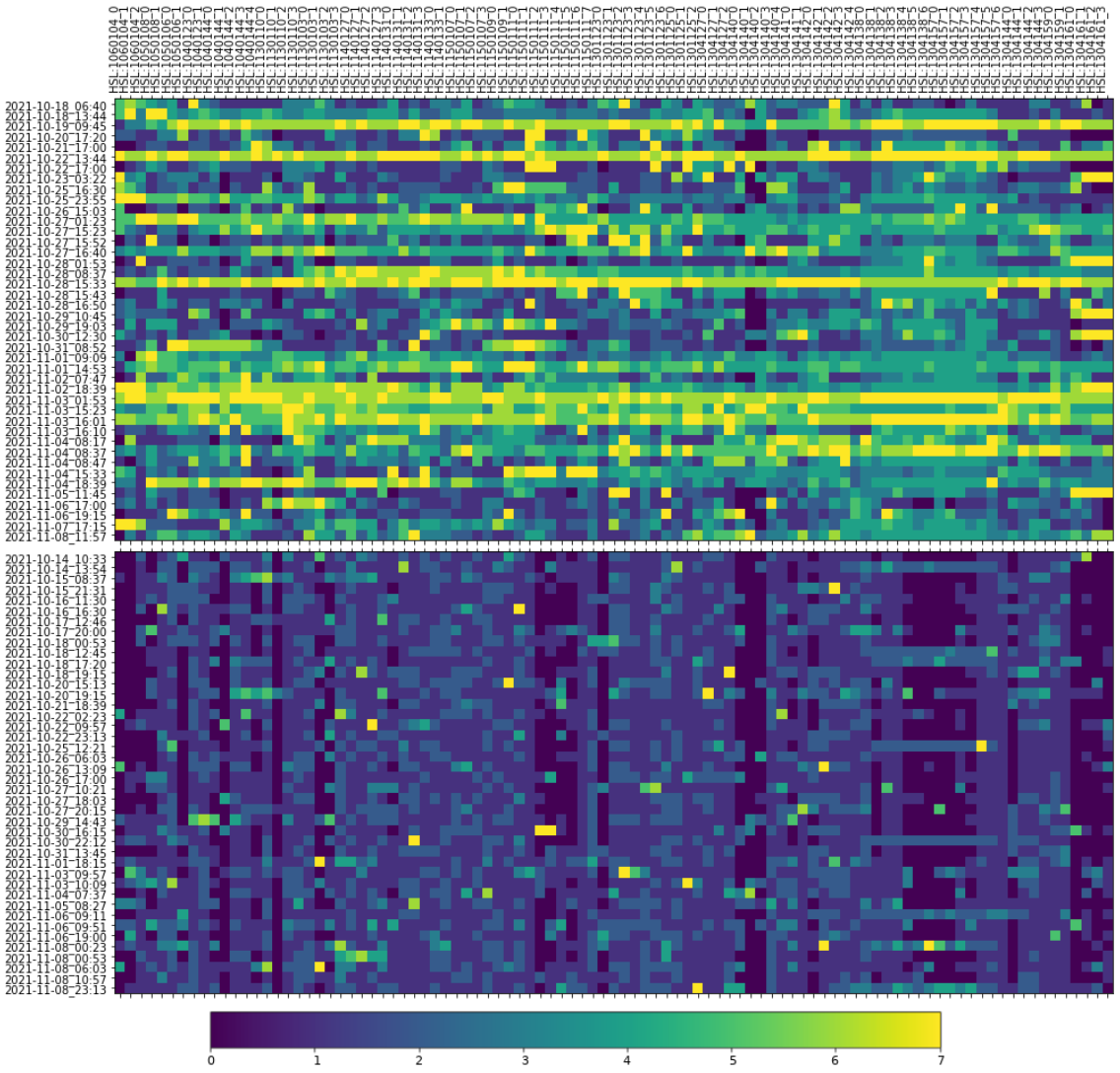
Kuva 15: Yllä olevassa kuvaajassa on 42 kappaletta anomaliaksi tunnistettuja vuoroja, kuvaaja on käytetyn ajan suhde matkamittarilukemaan. Alemmassa kuvassa on 42 anomaliatonta vuoroa. Anomaliavuorojen hajonta on selvästi suurempaa.



Kuva 16: Yllä olevassa kuvaajassa ovat 42 kappaletta anomaliaksi tunnistettuja vuoroja, kuvaajassa on esitettynä käytetyn ajan suhde annettuun viivästymiseen. Alemmassa kuvassa on 42 anomaliatonta vuoroa. Anomaliavuorojen hajonta on myös tässä selvästi suurempaa.



Kuva 17: Kuvassa on verrattu tiivistemuuttujien arvoja suhteessa toisiinsa anomaliallisten ja anomaliattomien muuttujien välillä. Kuvasta näkyy esimerkiksi, miten anomaliallisilla on ollut lähes kaksinkertainen viivästymisen keskiarvo (dl_{mean}) kun taas esimerkiksi nopeudet ovat olleet hitusen pienempiä (esimerkiksi spd_{min}).



Kuva 18: Kuvassa anomaliavuoroiksi tunnistettujen 42 vuoron segmentit yläpuolella, alla satunnaisesti valittujen anomaliattomien vuorojen segmentit. Anomaliavuoroiksi tunnistetuilla on selvästi myös enemmän anomalia-segmenttejä.

5 Johtopäätökset

Johtopäätöksenä tämän työn tuloksia voidaan todeta, että valitut menetelmät toimivat anomalian tunnistamisessa työn aineistossa. Menetelmien erilaisuuden vuoksi ne toimivat hyvin ensemblenä ja kaikkien menetelmien anomalioiksi merkitsemiä vuoroja löytyi useampi. Koska aineisto mahdollisti vain ohjaamattoman oppimisen, menetelmien tarkkuutta ei voitu objektiivisesti määrittää, mutta sitä koetettiin haarukoida muuta kautta. Tässä onnistuttiin sikäli, että voitiin todeta anomalioita löytyneen.

Kokeellisessa osassa aineiston käsittely oli tärkeä osa ja se ehdottomasti helpotti itse menetelmien käyttöä, varsinkin siitä syystä, että kaikkein selvimmin anomalioita olleet vuorot karsittiin pois jo ennen työn menetelmien kanssa työskentelemistä.

Aineisto on laaja ja varsin monipuolinen, joten sen parissa riittäisi jatkotutkittavaa moneltakin eri kantilta.

6 Liitteet

6.1 Koodi HSL:n API:n hyödyntämiseen

Huomautuksena koodiin liittyen: Koodissa on optio käyttää hyväksi säikeistystä, sillä oletettiin, että tietoa voi tulla liian paljon yhden prosessin hoidettavaksi. Tätä ei lopulta kuitenkaan tarvittu, yksi säie riitti tähän tarkoitukseen.

```
import paho.mqtt.client as mqtt
import sqlite3
from sqlite3 import Error
import json
from datetime import datetime
import _thread as thread, queue, time

safeprint = thread.allocate_lock()
dataQueue = queue.Queue()

database = "bussit_v3.sqlite"
HANDLERS = 1
handled = [None] * HANDLERS

sql_create_table = """ CREATE TABLE IF NOT EXISTS bussit (
                                        id INTEGER PRIMARY KEY
    AUTOINCREMENT,
                                        topic TEXT,
                                        event_type TEXT,
                                        desi INT,
                                        dir INT,
                                        oper INT,
                                        veh INT,
                                        tst TEXT,
                                        tsi INT,
                                        spd REAL,
                                        hdg REAL,
                                        lat REAL,
                                        long REAL,
                                        acc REAL,
                                        dl INT,
                                        odo INT,
                                        drst INT,
                                        oday TEXT,
                                        jrn INT,
                                        line INT,
                                        start TEXT,
                                        loc TEXT,
                                        stop TEXT,
                                        route INT,
                                        occu INT,
                                        seq INT,
                                        label TEXT,
                                        ttarr TEXT,
                                        ttdep TEXT,
                                        dr_type INT,
                                        tlp_requestid INT,
```

```
                                        tlp_requesttype TEXT,
                                        tlp_prioritylevel TEXT,
                                        tlp_reason TEXT,
                                        tlp_att_seq INT,
                                        tlp_decision TEXT,
                                        sid INT,
                                        signal_groupid INT,
                                        tlp_signalgroupnbr INT,
                                        tlp_line_configid INT,
                                        tlp_point_configid INT,
                                        tlp_frequency INT,
                                        tlp_protocol TEXT,
                                        block TEXT
                                    ); """


def create_connection(db_file):
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return conn


def create_table(conn, sql):
    try:
        c = conn.cursor()
        c.execute(sql)
    except Error as e:
        print(e)


def insert_data(conn, data):
    topic = data["topic"]
    payload = data["payload"]

    event_type = list(payload.keys())[0]

    values = payload[event_type]
    values["topic"] = topic
    values["event_type"] = event_type
    columns = ', '.join(values.keys())

    columns = columns.replace("-", "_")

    placeholders = ', '.join('?' * len(values))
    sql = 'INSERT INTO bussit ({}) VALUES ({})'.format(columns,
   placeholders)

    cur = conn.cursor()
    cur.execute(sql, tuple(values.values()))
    conn.commit()

```

```
102     return cur.lastrowid
103
104
105 def consumer(idnum, h):
106     conn = create_connection(database)
107
108     if conn is not None:
109         create_table(conn, sql_create_table)
110     else:
111         print("Error! cannot create the database connection.")
112
113     counter = 0
114     s = -999
115     while True:
116         counter += 1
117         try:
118             data = dataQueue.get()
119             s = dataQueue.qsize()
120             h[idnum] = counter
121
122             print(data)
123             insert_data(conn, data)
124         except queue.Empty as e:
125             if hasattr(e, 'message'):
126                 print(e.message)
127             else:
128                 print(e)
129         finally:
130             if counter % 250 == 0:
131                 with safeprint:
132                     print(f'consumer {idnum}, got => "data", queue
    size {s}, counter {counter}, timestamp {datetime.now().strftime
    ("%d.%m.%Y %H:%M:%S")}')
133
134
135 def on_connect(cli, userdata, flags, rc):
136     print("Connected with result code "+str(rc))
137     cli.subscribe("/hfp/v2/journey/ongoing/+/bus/+/+/1020/1/#")
138
139
140 def on_message(cli, userdata, msg):
141     data = msg.payload.decode()
142     data_as_json = json.loads(data)
143     dataQueue.put({"topic": msg.topic, "payload": data_as_json})
144
145
146 client = mqtt.Client()
147 client.on_connect = on_connect
148 client.on_message = on_message
149
150 for i in range(HANDLERS):
151     thread.start_new_thread(consumer, (i, handled))
152
153 client.connect("mqtt.hsl.fi", 1883, 60)
154
155 client.loop_forever()
```

6.2 Muunnetun aineiston sarakkeet

id_min	Segmentin aggregoitujen rivien määrä, minimi
id_max	Segmentin aggregoitujen rivien määrä, maksimi
id_count	Segmentin aggregoidut rivit, määrä
tst_first	Segmentin aikaleima, ensimmäinen
tst_last	Segmentin aikaleima, viimeinen
acc_min	Segmentin kiihtyvyys, minimi
acc_mean	Segmentin kiihtyvyys, keskiarvo
acc_median	Segmentin kiihtyvyys, mediaani
acc_max	Segmentin kiihtyvyys, maksimi
dl_min	Segmentin aikataulupoikkeama, minimi
dl_mean	Segmentin aikataulupoikkeama, keskiarvo
dl_median	Segmentin aikataulupoikkeama, mediaani
dl_max	Segmentin aikataulupoikkeama, maksimi
odo_min	Segmentin kuljettu matka (aineiston odo), minimi
odo_mean	Segmentin kuljettu matka (aineiston odo), keskiarvo
odo_median	Segmentin kuljettu matka (aineiston odo), mediaani
odo_max	Segmentin kuljettu matka (aineiston odo), maksimi
matkamittari_min	Segmentin kuljettu matka (laskettu koordinaateista), minimi
matkamittari_mean	Segmentin kuljettu matka (laskettu koordinaateista), keskiarvo
matkamittari_median	Segmentin kuljettu matka (laskettu koordinaateista), mediaani
matkamittari_max	Segmentin kuljettu matka (laskettu koordinaateista), maksimi
lat_min	Segmentin leveyspiiri, minimi
lat_mean	Segmentin leveyspiiri, keskiarvo
lat_median	Segmentin leveyspiiri, mediaani
lat_max	Segmentin leveyspiiri, maksimi
long_min	Segmentin pituuspiiri, minimi
long_mean	Segmentin pituuspiiri, keskiarvo
long_median	Segmentin pituuspiiri, mediaani
long_max	Segmentin pituuspiiri, maksimi
spd_min	Segmentin nopeus, minimi
spd_mean	Segmentin nopeus, keskiarvo
spd_median	Segmentin nopeus, mediaani
spd_max	Segmentin nopeus, maksimi

hdg_min	Segmentin ajosuunta, minimi
hdg_mean	Segmentin ajosuunta, keskiarvo
hdg_median	Segmentin ajosuunta, mediaani
hdg_max	Segmentin ajosuunta, maksimi
tsi_cum_min	Segmentin ajoon käytetty aika, minimi
tsi_cum_mean	Segmentin ajoon käytetty aika, keskiarvo
tsi_cum_median	Segmentin ajoon käytetty aika, mediaani
tsi_cum_max	Segmentin ajoon käytetty aika, maksimi
tsi_min	Segmentin UNIX-aikaleima, minimi
tsi_mean	Segmentin UNIX-aikaleima, keskiarvo
tsi_median	Segmentin UNIX-aikaleima, mediaani
tsi_max	Segmentin UNIX-aikaleima, maksimi
seg_times_segmentti_matka_min	Segmentin pituus (matka), minimi
seg_times_segmentti_matka_mean	Segmentin pituus (matka), keskiarvo
seg_times_segmentti_matka_median	Segmentin pituus (matka), mediaani
seg_times_segmentti_matka_max	Segmentin pituus (matka), maksimi
seg_times_segmentti_aika_min	Segmentin pituus (aika), minimi
seg_times_segmentti_aika_mean	Segmentin pituus (aika), keskiarvo
seg_times_segmentti_aika_median	Segmentin pituus (aika), mediaani
seg_times_segmentti_aika_max	Segmentin pituus (aika), maksimi
pysakki_dist_meters_min	Etäisyys segmentin pysäkistä, minimi
pysakki_dist_meters_mean	Etäisyys segmentin pysäkistä, keskiarvo
pysakki_dist_meters_median	Etäisyys segmentin pysäkistä, mediaani
pysakki_dist_meters_max	Etäisyys segmentin pysäkistä, maksimi
pysakkivali_matka_odo_min	Pysäkkivälin pituus (laskettu odo:sta), minimi
pysakkivali_matka_odo_mean	Pysäkkivälin pituus (laskettu odo:sta), keskiarvo
pysakkivali_matka_odo_median	Pysäkkivälin pituus (laskettu odo:sta), mediaani
pysakkivali_matka_odo_max	Pysäkkivälin pituus (laskettu odo:sta), maksimi
pysakkivali_matka_laskettu_min	Pysäkkivälin pituus (laskettu koordinaateilla), minimi
pysakkivali_matka_laskettu_mean	Pysäkkivälin pituus (laskettu koordinaateilla), keskiarvo
pysakkivali_matka_laskettu_median	Pysäkkivälin pituus (laskettu koordinaateilla), mediaani
pysakkivali_matka_laskettu_max	Pysäkkivälin pituus (laskettu koordinaateilla), maksimi
pysakkivali_aika_s_min	Pysäkkiväli sekunteina, minimi
pysakkivali_aika_s_mean	Pysäkkiväli sekunteina, keskiarvo
pysakkivali_aika_s_median	Pysäkkiväli sekunteina, mediaani
pysakkivali_aika_s_max	Pysäkkiväli sekunteina, maksimi

tsi_diff_sum	Segmentin peräkkäisten aikaleimojen erotusten summa, summa

Taulukko 3: Aggregoidun kolmiulotteisen aineistomatriisin selittävien muuttujien sarakkeet ja niiden kuvaukset.

Viitteet

1. Zhang, X., Zheng, Y., Zhao, Z., Liu, Y., Blumenstein, M. & Li, J. Deep Learning Detection of Anomalous Patterns from Bus Trajectories for Traffic Insight Analysis. *Knowledge-Based Systems* **217,** 106833. ISSN: 0950-7051 (huhtikuu 2021).
2. Wang, Z.-Y., Jin, B.-H., Ge, T. & Xue, T.-F. Detecting Anomalous Bus-Driving Behaviors from Trajectories. en. *Journal of Computer Science and Technology* **35,** 1047–1063. ISSN: 1860-4749 (lokakuu 2020).
3. Kong, X., Song, X., Xia, F., Guo, H., Wang, J. & Tolba, A. Lotad: Long-Term Traffic Anomaly Detection Based on Crowdsourced Bus Trajectory Data. *World Wide Web* **21,** 825–847. ISSN: 1386-145X, 1573-1413 (toukokuu 2018).
4. Kuang, W., An, S. & Jiang, H. Detecting Traffic Anomalies in Urban Areas Using Taxi GPS Data. *Mathematical Problems in Engineering* **2015,** 1–13. ISSN: 1024-123X, 1563-5147 (2015).
5. Praczyk, T. Ship Trajectory Anomaly Detection. *Intelligent Data Analysis* **23.** Publisher: IOS Press, 1021–1040. ISSN: 1088467X (syyskuu 2019).
6. Di Ciccio, C., van der Aa, H., Cabanillas, C., Mendling, J. & Prescher, J. Detecting Flight Trajectory Anomalies And Predicting Diversions In Freight Transportation. *Decision Support Systems* **88,** 1–17. ISSN: 0167-9236 (elokuu 2016).
7. Aggarwal, C. C. *Outlier Analysis* ISBN: 978-3-319-47577-6 (Springer International Publishing, Cham, 2017).
8. Mehrotra, K. G., Huang, H. & Mohan, C. K. *Anomaly Detection Principles and Algorithms* 1st ed. 2017. ISBN: 9783319675268 (Springer International Publishing : Imprint: Springer, Cham, 2017).
9. Asheesh Kumar Dwivedi, Ashish Kumar Rai & Ashish Kashyap. Fraud Detection in Credit Card Transactions using Anomaly Detection. *Turkish journal of computer and mathematics education* **12.** Place: Trabzon Publisher: Karadeniz Technical University Distance Education Research and Application Center, 837–846. ISSN: 1309-4653 (2021).
10. Hilal, W., Gadsden, S. A. & Yawney, J. Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances. *Expert Systems with Applications* **193,** 116429. ISSN: 0957-4174 (toukokuu 2022).
11. Vanhoeyveld, J., Martens, D. & Peeters, B. Value-Added Tax Fraud Detection with Scalable Anomaly Detection Techniques. *Applied Soft Computing* **86,** 105895. ISSN: 1568-4946 (tammikuu 2020).
12. Yongbum Kim & Kogan, A. Development of an Anomaly Detection Model for a Bank's Transitory Account System. *Journal of Information Systems* **28.** Publisher: American Accounting Association, 145–165. ISSN: 08887985 (2014).
13. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G. & Vázquez, E. Anomaly-based Network Intrusion Detection: Techniques, Systems And Challenges. *Computers & Security* **28,** 18–28. ISSN: 0167-4048 (helmikuu 2009).

14. Myers, D., Suriadi, S., Radke, K. & Foo, E. Anomaly Detection for Industrial Control Systems Using Process Mining. *Computers & Security* **78,** 103–125. ISSN: 01674048 (syyskuu 2018).

15. Carrasco, J., López, D., Aguilera-Martos, I., García-Gil, D., Markova, I., García-Barzana, M., Arias-Rodil, M., Luengo, J. & Herrera, F. Anomaly Detection in Predictive Maintenance: A New Evaluation Framework for Temporal Unsupervised Anomaly Detection Algorithms. *Neurocomputing* **462,** 440–452. ISSN: 09252312 (lokakuu 2021).

16. Givnan, S., Chalmers, C., Fergus, P., Ortega, S. & Whalley, T. Real-Time Predictive Maintenance Using Autoencoder Reconstruction and Anomaly Detection. *arXiv:2110.01447 [cs].* arXiv: 2110.01447 (lokakuu 2021).

17. Tonnaer, L., Li, J., Osin, V., Holenderski, M. & Menkovski, V. *Anomaly Detection for Visual Quality Control of 3D-Printed Products* teoksessa *2019 International Joint Conference on Neural Networks (IJCNN)* ISSN: 2161-4407 (heinäkuu 2019), 1–8.

18. Nieves Avendano, D., Caljouw, D., Deschrijver, D. & Van Hoecke, S. Anomaly Detection and Event Mining in Cold Forming Manufacturing Processes. *International Journal of Advanced Manufacturing Technology* **115.** Publisher: Springer Nature, 837–852. ISSN: 02683768 (heinäkuu 2021).

19. Carreño, A., Inza, I. & Lozano, J. A. Analyzing Rare Event, Anomaly, Novelty and Outlier Detection Terms Under the Supervised Classification Framework. *Artificial Intelligence Review* **53,** 3575–3594. ISSN: 0269-2821, 1573-7462 (kesäkuu 2020).

20. Amidan, B., Ferryman, T. & Cooley, S. *Data OutlierDdetection Using the Chebyshev Theorem* teoksessa *2005 IEEE Aerospace Conference* ISSN: 1095-323X (maaliskuu 2005), 3814–3819.

21. Chiranjit Das, Akhtar Rasool, Aditya Dubey & Nilay Khare. Analyzing the Performance of Anomaly Detection Algorithms. *(IJACSA) International Journal of Advanced Computer Science and Applications* **12,** 439–445 (2021).

22. Rousseeuw, P. J. Least Median of Squares Regression. *Journal of the American Statistical Association* **79,** 871–880. ISSN: 0162-1459 (joulukuu 1984).

23. *Robust Covariance Estimation and Mahalanobis Distances Relevance* `https://scikit-learn/stable/auto_examples/covariance/plot_mahalanobis_distances.html` (2022).

24. Hubert, M. & Debruyne, M. Minimum Covariance Determinant. *WIREs Computational Statistics* **2,** 36–43. ISSN: 1939-0068 (2010).

25. Li, X., Deng, S., Li, L. & Jiang, Y. Outlier Detection Based on Robust Mahalanobis Distance and Its Application. *Open Journal of Statistics* **9.** Number: 1 Publisher: Scientific Research Publishing, 15–26 (tammikuu 2019).

26. Blanco, R., Malagón, P., Briongos, S. & Moya, J. M. Teoksessa *Hybrid Artificial Intelligent Systems* (toim. Pérez García, H., Sánchez González, L., Castejón Limas, M., Quintián Pardo, H. & Corchado Rodríguez, E.) 648–659 (Springer International Publishing, Cham, 2019). ISBN: 978-3-030-29858-6 978-3-030-29859-3.

27. Li, L., Hansman, R. J., Palacios, R. & Welsch, R. Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring. *Transportation Research Part C: Emerging Technologies* **64,** 45–57. ISSN: 0968-090X (maaliskuu 2016).

28. Clark, K. *Outlier Detection in Gaussian Mixture Models* Accepted: 2020-10-15T13:51:27Z. Thesis (2020).

29. Moon, T. The Expectation-Maximization Algorithm. *IEEE Signal Processing Magazine* **13,** 47–60 (1996).

30. Angelis, A., De Angelis, G. & Carbone, P. Using Gaussian-Uniform Mixture Models for Robust Time-Interval Measurement. *IEEE Transactions on Instrumentation and Measurement* **64,** 3545–3554 (syyskuu 2015).

31. Rousseeuw, P. J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics* **20.** Publisher: North-Holland, 53–65. ISSN: 0377-0427 (marraskuu 1987).

32. Sarle, W. S. Review of Finding Groups in Data: An Introduction to Cluster Analysis. *Journal of the American Statistical Association* **86.** Publisher: American Statistical Association, Taylor & Francis, Ltd., 830–832. ISSN: 0162-1459 (1991).

33. Khachay, M. & Khachay, D. Attainable Accuracy Guarantee for the K-Medians Clustering in [0, 1]. *Optimization Letters* **13,** 1837–1853. ISSN: 1862-4472, 1862-4480 (marraskuu 2019).

34. Arthur, D. & Vassilvitskii, S. *K-Means++: The Advantages of Careful Seeding* teoksessa *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* event-place: New Orleans, Louisiana (Society for Industrial ja Applied Mathematics, USA, 2007), 1027–1035. ISBN: 978-0-89871-624-5.

35. Liu, D. & Yu, J. *Otsu Method and K-Means* teoksessa *2009 Ninth International Conference on Hybrid Intelligent Systems* **1** (elokuu 2009), 344–349.

36. Chandola, V., Banerjee, A. & Kumar, V. Anomaly Detection: A Survey. *ACM Computing Surveys* **41,** 1–58. ISSN: 0360-0300, 1557-7341 (heinäkuu 2009).

37. *Data Mining and Knowledge Discovery Handbook* (toim. Maimon, O. & Rokach, L.) ISBN: 978-0-387-24435-8 978-0-387-25465-4 (Springer, New York, 2005).

38. Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. *A Density-based Algorithm For Discovering Clusters In Large Spatial Databases With Noise* teoksessa (AAAI Press, 1996), 226–231.

39. Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. LOF: Identifying Density-based Local Outliers. *ACM SIGMOD Record* **29,** 93–104. ISSN: 0163-5808 (kesäkuu 2000).

40. Kingma, D. P. & Welling, M. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning* **12.** arXiv:1906.02691 [cs, stat], 307–392. ISSN: 1935-8237, 1935-8245 (2019).

41. Gehring, J., Miao, Y., Metze, F. & Waibel, A. *Extracting Deep Bottleneck Features Using Stacked Auto-encoders* teoksessa *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* ISSN: 2379-190X (toukokuu 2013), 3377–3381.

42. Qi, Y., Wang, Y., Zheng, X. & Wu, Z. *Robust Feature Learning by Stacked Autoencoder with Maximum Correntropy Criterion* teoksessa *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* ISSN: 2379-190X (toukokuu 2014), 6716–6720.

43. Zhou, C. & Paffenroth, R. C. *Anomaly Detection with Robust Deep Autoencoders* teoksessa *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, New York, NY, USA, elokuu 2017), 665–674. ISBN: 978-1-4503-4887-4.

44. Banks, A., Briggs, E., Borgendale, K. & Gupta, R. *MQTT Version 5.0* Oasis Standard. 2019. `https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html` (2021).

45. HSL. *High-frequency positioning* 2019. `https://digitransit.fi/en/developers/apis/4-realtime-api/vehicle-positions/` (2021).

46. Ramsay, J., Ramsay, J., Silverman, B. W. & Silverman, H. O. W. P. o. M. B. W. *Functional Data Analysis* ISBN: 978-0-387-40080-8 (Springer Science & Business Media, kesäkuu 2005).

47. Eiter, T. & Mannila, H. *Computing Discrete Fréchet Distance* tekninen raportti (1994).

48. Figueira, J. P. *Discrete-Frechet* original-date: 2019-08-03T19:22:12Z. Maaliskuu 2022. `https://github.com/joaofig/discrete-frechet` (2022).

49. *geopandas.GeoSeries.simplify — GeoPandas 0.10.2+0.g04d377f.dirty documentation* `https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoSeries.simplify.html` (2022).