jamk.fi

Poikkeamantunnistus Microsoft Anomaly Detectorilla

Joonas Pöyhönen

Opinnäytetyö
Joulukuu 2020
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), tieto- ja viestintätekniikan ala

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Pöyhönen, Joonas	Opinnäytetyö, AMK	Joulukuu 2020
	Sivumäärä 59	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Poikkeamantunnistus Microsoft Anomaly Detectorilla

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Manninen Pasi, Rantala Ari

Toimeksiantaja(t)
Iotas Oy

Tiivistelmä

Hajautettujen järjestelmien ja esineiden internetin myötä järjestelmien suorituskyvyn hallinta ja seuraaminen on muuttunut työlääksi. Suurien kohdemäärien seuraaminen perinteisillä sääntöpohjaisilla ratkaisuilla ei takaa muuttuvassa ympäristössä ongelmien tarkkaa paikannusta poikkeustilanteissa. Poikkeaman tunnistaminen voi auttaa osoittamaan, missä ongelma esiintyy ja tehostamaan juurisyyn selvittämistä.

Tavoitteena oli perehtyä Microsoftin Anomaly Detector -nimiseen poikkeamantunnistus palveluun ensin rakentamalla tietoperusta palvelun käyttämistä teknikoista ja ominaisuuksista ja sitten rakentamalla työympäristö, jossa kerättiin aikasarja-aineistoja Anomaly Detectorin poikkeaman tunnistamista varten. Poikkeaman tunnistamisesta saaduilla tuloksilla yritettiin selvittää soveltuuko Anomaly Detector toimeksiantajan määrittämiin käyttötarkoituksiin ja saadaanko vastaukset esitettyihin kysymyksiin.

Työympäristö toteutettiin käyttäen C#-ohjelmointikieltä Visual Studio -ohjelmointiympäristössä ja sovellukset luotiin sykedatan keräämistä varten, sen esikäsittelyyn ja lähettämiseen Anomaly Detectoriin tutkittavaksi ja lopulta tulosten esittämiseen graafisessa käyttöliittymässä.

Tulokset osoittivat Anomaly Detectorin soveltuvan määritettyihin käyttötarkoituksiin ja ne myös täsmäsivät suhteellisen hyvin odotettuihin tuloksiin Microsoftin dokumenttien lupausten perusteella. Tuloksien pohjalta palvelun integroiminen toimeksiantajan järjestelmiin on täysin mahdollista ja pohditut jatkokehitysmahdollisuudet antoivat syyn lisätutkimuksille.

Avainsanat (asiasanat)

Aikasarja, Anomaly Detector, Koneoppiminen, Microsoft, Poikkeamantunnistus

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Pöyhönen, Joonas	Bachelor's thesis	December 2020
		Language of publication: Finnish
	Number of pages: 59	Permission for web publication: x

Title of publication

Anomaly detection with Microsoft Anomaly Detector

Degree programme

Information Technology

Supervisor(s)

Manninen Pasi, Rantala Ari

Assigned by

Iotas Oy

Abstract

With the distributed systems and internet of things, managing and monitoring the performance of systems has become laborious. Tracking large numbers of things with traditional rule-based solutions does not guarantee accurate pinpointing of problems in anomalous situations in a changing environment. Anomaly detection can help indicate where the problem occurs and improve the investigation of the root cause.

The goal was to become familiar with Microsoft's Anomaly Detector service, first by building a knowledge base of the technologies and features used by the service and then by building a work environment that collected time series data for Anomaly Detector. The results obtained from the Anomaly Detector were used to determine whether the Anomaly Detector is suitable for the uses specified by the client and whether the questions regarding Anomaly Detector can be answered.

The work environment was implemented using C# programming language with the development environment Visual Studio and applications were created to collect heart rate data, pre-process and send the data to Anomaly Detector for analysis, and finally to present the results in a graphical user interface.

The results showed that Anomaly Detector was suitable for the client's specified uses and matched the relatively well-expected results based on the promises of Microsoft documents. Based on the results, the integration of the service into the client's systems is entirely possible, and the considered further development opportunities gave rise to further research.

Keywords/tags (subjects)

Anomaly detection, Anomaly Detector, Machine learning, Microsoft, Time series

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

Sanasto

Aikasarja
Aikasarja on aineisto, joka sisältää mitattujen arvojen lisäksi myös mittausajankohdan.

Aliverkkotunnus
Aliverkkotunnus (engl. Subdomain) on ensisijaisen verkkotunnuksen laajennus, joka sijaitsee ennen ensisijaista verkkotunnusta pisteellä eroteltuna.

Anomaly
Poikkeama, eli normaalista poikkeava datailmentymä annetussa aineistossa

IDE
Lyhenne sanoista Integrated development environment, eli ohjelmointikehitysympäristö

JSON
Lyhenne sanoista JavaScript Object Notation. JSON on kevyt tekstipohjainen formaatti, joka on pääasiassa käytetty datan siirtämiseen verkkosovelluksissa

.NET
.NET, eli dotnet on Microsoftin kehittämä ohjelmistokehys.

POST-pyyntö
POST-pyyntöä käytetään selaimissa tai WWW-palvelimissa tiedonsiirtoon, esimerkiksi REST-rajapintoja kutsuessa. (ks. REST-rajapinta)

Päätepiste
Päätepiste on URL-osoitteen lopussa oleva polku haluttuun toimintoon.

REST-rajapinta
REST-rajapinta on ohjelmointirajapinta, jota yleensä käsitellään HTTP-protokollan yli tulevilla pyynnöillä.

1 Johdanto

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Iotas Oy. Iotas Oy on Jyväskylässä sijaitseva yritys, joka perustettiin vuonna 2016. Päätoimiala on ohjelmistojen suunnittelu, valmistus ja konsultointi. Tällä hetkellä Iotas tarjoaa pääasiassa vain konsultointia yrityksille, mutta kehityksessä on myös älykäs turva- ja hyvinvointiranneke ratkaisu, joka on suunnattu vanhusten kotihoitoa varten. (Iotas 2020.)

1.2 Työnanto

Työnantona oli rakentaa toimiva työympäristö poikkeamien tunnistamista ja analysointia varten. Poikkeamien tunnistamiseen työnantaja valitsi Microsoft Anomaly Detector -nimisen palvelun.

Työn teoriavaiheessa perehdyttiin poikkeaman tunnistamiseen yleisellä tasolla käymällä läpi eri poikkeaman tunnistamisessa käytettyjä tekniikoita ja strategioita. Työn kannalta tärkeä teoriakohde oli myös aikasarjat, koska Anomaly Detectorin toiminta perustuu aikasarja-aineistojen tutkimiseen. Myöhemmin työssä tutustuttiin Microsoftin Anomaly Detectoriin ja siinä käytettyihin teknologioihin käymällä läpi Anomaly Detectorin ominaisuudet ja selvittämällä, kuinka Anomaly Detectorin tunnistamisessa käytetyt parametrit toimivat ja kuinka se havaitsee poikkeamia syötetyistä aineistoista.

Työn toteutusvaiheessa rakennettiin aiemmin mainittu työympäristö itse valitulla ohjelmointikielellä, joka sisälsi tarvittavat sovellukset sykedatan keräämistä, käsittelyä, analysointia ja esittämistä varten. Saatuja tuloksia käytettiin arvioimaan, että soveltuuko Anomaly Detector Iotaksen käyttötarkoituksiin ja tuoko se yritykselle lisäarvoa.

2 Poikkeamantunnistus

2.1 Yleistä

Poikkeamantunnistus (engl. Anomaly detection tai Outlier detection) on prosessi, jolla etsitään poikkeavuuksia annetusta aineistosta (engl. dataset). Poikkeamat ovat pisteitä tai pistekokoelmia aineistoissa, jotka eivät noudata odotettua toimintatapaa samalla tavalla kuin muut aineiston pisteet. Aineistoissa, joissa toimintatapaa tai kausiluonteisuutta ei tiedetä, poikkeamat tarkistetaan tyypillisesti katsomalla pisteet, jotka eroavat merkittävästi muista aineiston pisteistä. (Kotu & Deshpande 2018, 447.)

Piste, tai datapiste on annetun aineiston yksittäinen kohta, mikä sisältää tutkittavan arvon. Aikasarjoissa piste saattaa sisältää myös aikaleiman mitatun pisteen ajanhetkeltä. Jos piste eroaa merkittävästi aikaisemmista pisteistä, eikä noudata odotettua toimintatapaa, niin se on todennäköisesti poikkeama. (Chandola, Banerjee & Kumar 2009, 1.) Kausiluonteisessa aineistossa poikkeava piste ei aina erotu merkittävästi joukosta pikaisesti katsottuna, koska poikkeamat voidaan havaita vertaamalla tarkasteltua pistettä edellisen jakson vastaavaan arvoon tietyissä algoritmeissa, mikä tietyssä asiayhteydessä ei vaadi suurta eroa. Esimerkiksi kello 15:00 mitattua arvoa voidaan verrata edellisen päivän kello 15:00 mitattuun arvoon. (Kotu & Deshpande 2018, 401.)

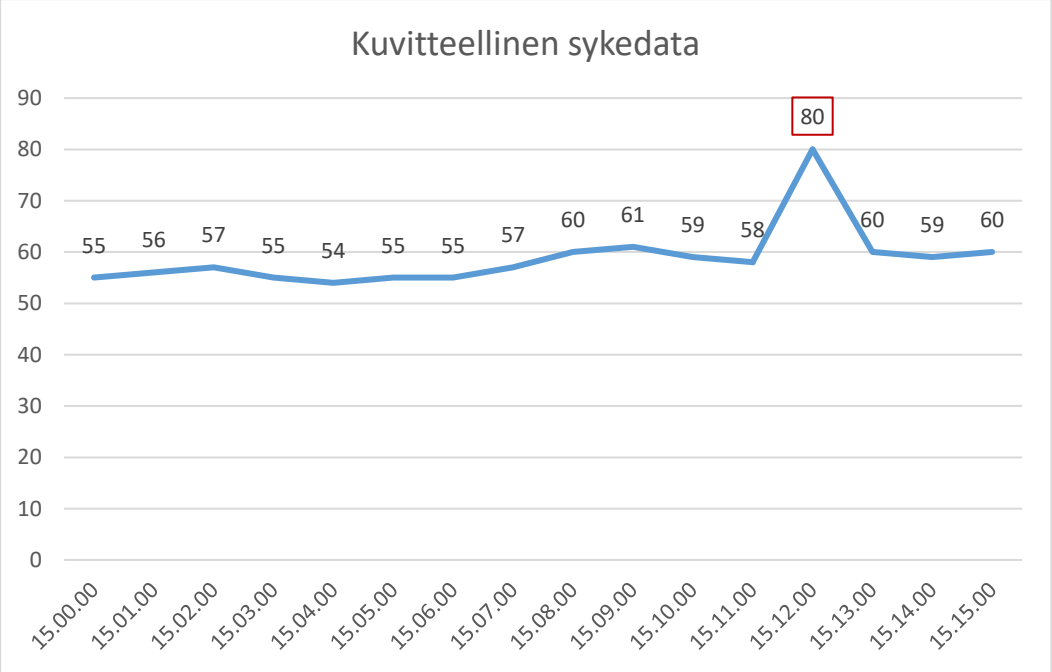
Poikkeaman tunnistamisen havaitsemismenetelmät perustuvat aikaisempien tietojen malleihin ja odotettuihin arvoihin. Ensisijainen oletus normaalista käyttäytymisestä aineistossa on stationaarinen, eli toisin sanoen prosessit, jotka tuottavat aineistoa ei uskota muuttuvan merkittävästi. Siksi tilastot, jotka ovat aiemmin kuvanneet järjestelmää tietyllä tavalla, kuvaavat järjestelmää myös tulevaisuudessa samalla tavalla. Jos aineisto muuttuu ajan kuluessa, niin sitä voidaan tilastoissa kuvata pitkällä aikavälillä. (Mehrotra, Kishan, Mohan, Chilukuri, Huang & HuaMing 2017, kindle sijainti 610.)

2.2 Erilaiset poikkeamatyypit

2.2.1 Pistepoikkeama

Pistepoikkeama tai globaali poikkeama (engl. Point anomaly tai Global anomaly) on yksittäinen piste aineistossa, joka ei noudata mitään tunnettua toimintatapaa ja eroaa koko aineiston muista pisteistä. Pisteen arvo voi siis olla huomattavasti suurempi tai pienempi normaalista. (Cohen 2019.)

Kuvio 1:n viivakaavio käyttää aineistona satunnaista sykedataa. Sykkeet ovat hajautettu siten, että yksi datapiste on mitattu syke tietyllä ajanhetkellä ja jokainen piste on mitattu minuutin välein. Tässä kaaviossa on globaali poikkeama, jonka piste on 80 ajanhetkellä 15:12.



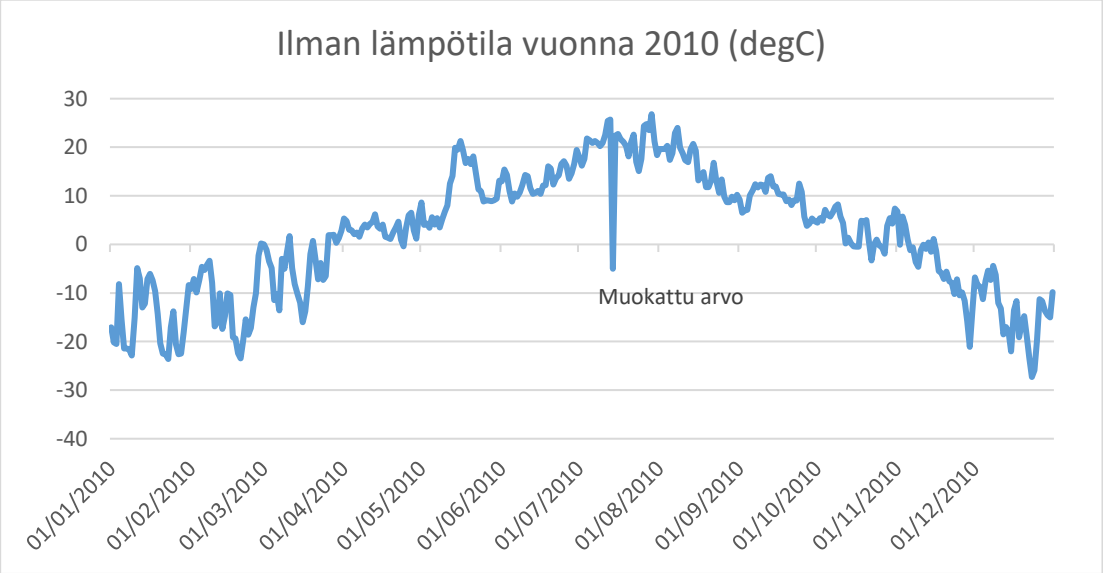
Kuvio 1. Viivakaavio satunnaisella sykedatalla

2.2.2 Asiayhteydestä riippuva poikkeama

Asiayhteydestä riippuva poikkeama (engl. Contextual outlier) on piste, jonka arvo ei täsmää odotettua arvoa tietyssä kontekstissa. Suuressa aineistossa voi olla paljon pisteitä, jotka toistuvat tietyissä jaksoissa ja näitä jaksoja voidaan kutsua kontekstiksi. (Cohen 2019.)

Esimerkiksi sijoittamalla Suomessa mitatut lämpötilat usealta vuodelta kuvaajaan paljastaa, että Suomessa on normaalisti lämmin kesällä ja kylmä talvella. Kuvaajassa on silloin paljon pisteitä, jotka käyvät pakkasen puolella ja myös hellerajan yläpuolella. Mutta jos kesällä yksi piste on miinuksen puolella, niin vaikka tämä ei globaalissa kontekstissa tarkoittaisi poikkeamaa, niin se ei noudattaisi vuosikohtaista kausiluonteisuutta, joka on säähavaintojen kohdalla tyypillinen konteksti ja siksi se lasketaan asiayhteydestä riippuvaksi poikkeamaksi.

Kuviossa 2 esiintyy ilman lämpötila kaikille päiville vuodelta 2010. Lämpötila on mitattu Jyväskylän lentoasemalla. Heinäkuun yhdelle mitatulle arvolle on asetettu arvoksi -5 ja tämä on selkeä asiayhteydestä riippuva poikkeama, koska se ei noudata aineiston kesän arvoja samassa kontekstissa. Mutta tämä piste ei ole pistepoikkeama, koska se kuuluu odotettuihin arvoihin alku- ja loppuvuodesta.

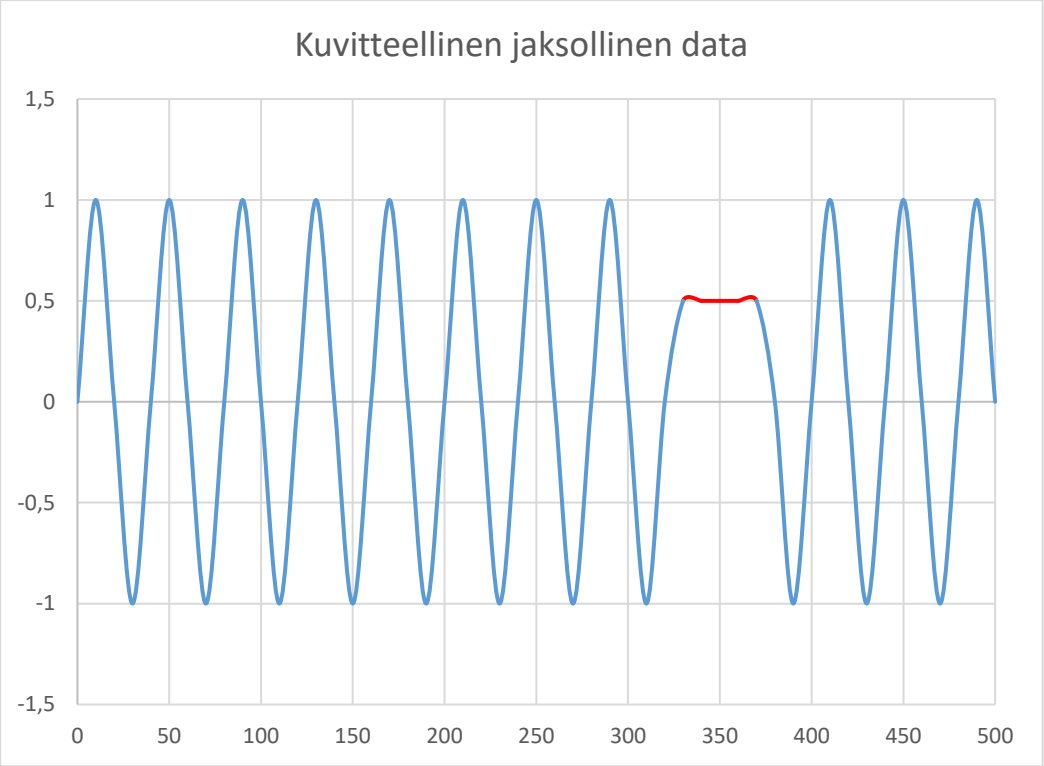


Kuvio 2. Ilmatieteenlaitoksen säähavainnot vuodelta 2010 yhdellä muokatulla arvolla (Ilmatieteenlaitos.)

2.2.3 Yhteispoikkeamat

Yhteispoikkeama (engl. Collective outlier) on ilmiö, jossa useampi piste eroaa peräkkäin merkittävästi aineiston muista pisteistä, mutta niitä ei voida luokitella piste- tai asiayhteydestä riippuvaksi poikkeamaksi kokoelmana. (Cohen 2019.)

Kuviossa 3 kuvataan jotain jaksollista ilmiötä, joka poikkeaa normaalista useassa eri pisteessä kohdissa 330–370. Poikkeama ei ole pistepoikkeama, koska mikään peräkkäinen piste ei itsessään ole aineiston ulkopuolella verrattuna aikaisempiin pisteisiin. Poikkeaman peräkkäiset pisteet eivät myöskään noudata mitään aikaisempaa toimintatapaa ja siksi se ei voi olla asiayhteydestä riippuvainen poikkeama.



Kuvio 3. Kaavio jaksollisella datalla, jossa on punaisella yhteispoikkeama

2.3 Poikkeamien havaitsemismenetelmät

2.3.1 Valvomaton poikkeamien havaitseminen

Valvomaton poikkeamien havaitseminen (engl. Unsupervised anomaly detection) ottaa vastaan aineiston, jossa ei kerrota mitä datapisteet edustavat ja täten käytetty tai käytetyt algoritmit joutuvat mallintamaan datapisteet etsimällä eniten aineistoista poikkeavat pisteet (Kotu & Deshpande 2018, 10.)

2.3.2 Valvottu poikkeamien havaitseminen

Valvottu poikkeamien havaitseminen (engl. Supervised anomaly detection) vaatii ennestään koulutusaineiston, johon on merkitty jo valmiiksi normaaleja ja poikkeavia datapisteitä. Valvotussa poikkeamien havaitsemisessa verrataan siis uutta aineistoa käytettyyn koulutusaineistoon, jonka perusteella havaitaan yhteneväisyydet ja pystytään ennustamaan tulevat arvot. (Kotu & Deshpande 2018, 10.)

2.3.3 Puolivalvottu poikkeamien havaitseminen

Puolivalvottu poikkeamien havaitseminen (engl. Semi-supervised anomaly detection) on kahden ylemmän havaitsemismenetelmän keskitie. Puolivalvotussa poikkeamien havaitsemisessa analysoitava aineisto on yleensä ainakin puoliksi merkitty, joten siinä näkyy mahdolliset poikkeavuudet. Mutta toisen puoliskon datapisteet eivät ole merkittyjä, joten niitä täytyy verrata jo valmiiksi merkattuihin arvoihin. (Li & Liang 2019, 1.)

2.4 Poikkeamien havaitsemisessa käytettyjä strategioita

2.4.1 Sääntöpohjainen strategia

Sääntöpohjainen strategia on yleisesti sovellettavissa monessa eri osa-alueessa. Tyypillinen sääntöpohjaisen strategian ilmentymä on sääntöihin perustuva poikkeamien havaitsemisjärjestelmä, joka sisältää joukon sääntöjä, joukon tosiasioita ja tulkin sääntöjen ja tosiseikkojen soveltamiseksi. Tässä strategiassa säännöt on opittava,

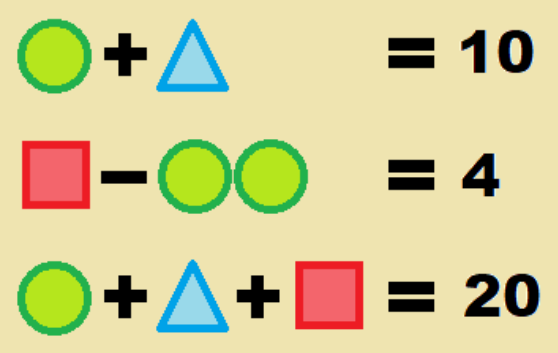
jotta poikkeamien havaitseminen olisi mahdollisimman tarkkaa. Säännöt ovat yleensä asiantuntevien ihmisten suunnittelemia, joilta löytyy vahva tieto tietyssä sovelluksessa. On myös mahdollista, että säännöt ovat opittu jo valmiiksi merkitystä aineistosta luokitteluun perustuvilla menetelmillä, kuten päätöspuulla ja tukivektorikoneella. (Huang 2018, 36.)

Esimerkiksi kuvio 4 esittää seuraavat säännöt:

- Jos ympyrään lisätään kolmio, niin tulos on 10
- Jos neliöstä vähennetään kaksi ympyrää, niin tulos on 4
- Jos ympyrä, kolmio ja neliö lasketaan yhteen, niin tulos on 20

Ja tosiseikat kuviossa ovat seuraavat:

- Ympyrä on 3
- Kolmio on 7
- Neliö on 10



Kuvio 4. Yksinkertainen matemaattinen pulma

2.4.2 Tapauspohjainen strategia

Tapauspohjainen strategia etsii kohdetapauksen kaltaisia tapauksia auttamaan poikkeamien tunnistamisessa. Ensi silmäyksellä tämä strategia vaikuttaa perustuvan lähinaapurimenetelmiin, jotka määrittävät datapisteiden epänormaalisuuden lähimpien naapureiden perusteella, mutta tapauspohjainen strategia on kuitenkin tehokkaampi

sovelluksissa, joissa kohdeobjekti on monimutkainen ja poikkeavuuksien havaitsemista koskevien sääntöjen päätteleminen on hankalaa. Monimutkaisissa sovelluksissa, kuten verkon analysoimisessa tapauspohjainen strategia on ensimmäinen vaihe, joka parantaa huomattavasti poikkeamien havaitsemista. (Huang 2018, 15.)

Esimerkki tapauspohjaisen strategian hyvästä käyttökohteesta on käyttäjäprofiiliaineisto, koska se on melko monimutkainen siinä mielessä, että se sisältää monen tyyppistä dataa, esimerkiksi kuvainformaatiota, kategorista tietoa ja numeerista tietoa. Tämän seurauksena poikkeamien havaitsemista koskevien sääntöjen yhteenveto tulee hankalaksi ja tehottomaksi. Tapauspohjainen strategia on siis sopivampi, koska siinä keskitytään kohdetapauksen vastaavien tapausten tunnistamiseen ja vähennetään merkittävästi analysoitavien tietojen kokoa. (Mts. 15.)

2.4.3 Odotuspohjainen strategia

Odotuspohjainen strategia liittyy odotetun normaalikonseptin hyödyntämiseen poikkeamien määrittämiseksi. Toisin sanoen, aina kun aineistossa oleva esiintymä on odotusten ulkopuolella, sitä pidetään poikkeamana. (Huang 2018, 17.)

Datan todennäköisyysanalyysissä analysoidaan ja muotoillaan normaalidatan jakaumat, jotta voidaan määrittää datan esiintymän poikkeavuuden todennäköisyys. Tosin datan arvioinnissa odotetut datapisteet lasketaan kuitenkin suoraan normaalitiedoista niiden todennäköisyydestä riippumatta. Nämä kaksi muotoa voidaan myös yhtenäistää antamaan todennäköisyysarvio normaaliuden käsitteestä. Näiden lomakkeiden valinta riippuu suuresti sovelluksista. (Mts. 18.)

Konkreettisena esimerkkinä aikasarjojen ennakoinnissa poikkeamien havaitsemiseksi tulevaisuuden arvon todennäköisyysarviointi esittää odotetun arvon lisäksi myös varianssin siten, että saadaan ennusteen luotettavuus. (Mts. 18.)

2.4.4 Ominaisuuspohjainen strategia

Ominaisuuspohjainen strategia on suhteellisen uusi havaitsemisstrategia, joka perustuu datan esiintymien piilevien ominaisuuksien havaitsemiseen. Tämä ominaisuuspohjainen strategia olettaa, että järjestelmässä, joka luo aineiston on aina staattisia tai vakaita ominaisuuksia. Ominaisuuksien tunnistamisen avulla niitä voidaan käyttää kriteerinä erottamaan normaalit ja epänormaalit datapisteet. Esimerkiksi epänormaaleilla datapisteillä ei ole tiettyjä ominaisuuksia, mikä viittaa siihen, että taustajärjestelmä ei luo niitä. Piilevien ominaisuuksien löytämiseen on ehdotettu useita menetelmiä, joista kaksi edustavaa esimerkkiä ovat 1) ominaisuuksien korrelaatio, joka nimenomaisesti löytää vakaan sisäisen suhteen dataominaisuuksien välille; ja 2) datan pakkaus, joka mittaa implisiittisesti dataan piilotetun normaalin tiedon määrää. Siksi niin kauan kuin datan ilmentymä rikkoo vakaata sisäistä suhdetta tai normaalia tietomäärää, se määritellään poikkeamaksi. (Huang 2018, 19–20.)

2.5 Poikkeamien tunnistamistekniikkoja ja algoritmeja

2.5.1 Klusterointi

Klusterointitekniikkoja käytetään samankaltaisen datanilmentyminen ryhmittelyssä klustereiksi. Klusterointi on ensisijaisesti valvomaton tekniikka ja se voidaan jakaa kolmeen eri kategoriaan, tai oletukseen, joista ensimmäinen on:

Oletus: *Normaalit datailmentymät kuuluvat aineistossa olevaan klusteriin, kun taas poikkeamat eivät kuulu mihinkään klusteriin*

Edellä olevaan oletukseen perustuvat tekniikat soveltavat tunnettua klusterointiin perustuvaa algoritmia aineistoon ja ilmoittavat epätavallisiksi kaikki datailmentymät, jotka eivät kuulu mihinkään klusteriin. Tähän oletukseen perustuvia klusterointialgoritmeja on seuraavanlaisia: *DBSCAN*, *ROCK* ja *SNN Clustering***.** *FindOut* on myös mainitsemisen arvoinen, koska se poistaa aineistosta kaikki klusterit, jolloin jäljelle jää vain datailmentymiä, jotka merkitään poikkeamiksi. (Chandola, Banerjee & Kumar, 2009 30.)

Oletus: *Normaalit datailmentymät ovat lähellä lähintä klusterikeskiötä, kun taas poikkeamat ovat kaukana lähimmästä klusterikeskiöstään*

Edellä olevaan oletukseen perustuvat tekniikat koostuvat kahdesta vaiheesta. Ensimmäisessä vaiheessa data on ryhmitelty klusterointialgoritmilla. Toisessa vaiheessa jokaista datailmentymää kohden mitataan sen etäisyys lähimpään klusterikeskiöön ja se merkitään ilmentymän poikkeavuus pisteeksi. (Mts. 31.)

On lukuisia poikkeamien tunnistamistekniikoita, jotka seuraavat tätä kaksivaiheista lähestymistapaa, kuten: *Self-Organizing Maps* (SOM), *K-means Clustering* ja *Expectation Maximization*. (Mts. 31.)

On myös hyvä mainita, että toiseen oletukseen perustuvat algoritmit voivat toimia myös puolivalvotussa tilassa, jossa koulutusaineisto on klusteroitu ja koulutusaineistoon kuuluvat datailmentyvät verrataan klustereihin poikkeavuus pisteiden saamiseksi testi-datailmentymää varten. Mutta jos poikkeamat aineistossa muodostavat klustereita itsekseen, niin yllä mainitut tekniikat eivät pysty tunnistamaan tällaisia poikkeamia. Tämän ratkaisemiseksi on olemassa kolmas oletukseen perustuva tekniikka: (Mts. 31.)

Oletus: *Normaalit datailmentymät kuuluvat suuriin ja tiheisiin klustereihin, kun taas poikkeamat joko kuuluvat pieniin tai harvoihin klustereihin*

Tekniikat, jotka perustuvat kolmanteen oletukseen ilmoittavat klustereihin kuuluvat datailmentymät, joiden koko ja/tai tiheys on alle kynnyksen poikkeavana. Kolmannesta tekniikasta on muutamia variaatioita, kuten *Cluster-Based Local Outlier* (Find-CBLOF), *k-d tree* ja *CD-tree*. (Mts. 31–32.)

K-means-klusterointi

K-Means-klusterointi (K-Means clustering) on yksi käytetyimmistä ja yksinkertaisimmista klusterointialgoritmeista. Tässä tekniikassa käyttäjä määrittää joukon klustereita (*k*), jotka ryhmitellään aineistoon. Aikaisemmin tekstissä todettiin, että k-

means:in käyttäminen poikkeaman tunnistamisessa tapahtuu kaksivaiheisena oletuksena, eli ensin k-means klusterointialgoritmia käytetään datapisteiden ryhmittelemiseksi klustereihin ja sitten mitataan datapisteiden etäisyys lähimpään klusterikeskiöön, josta saadaan poikkeavuus piste. (Kotu & Deshpande 2018, 226.)

K-means toimii käytännössä seuraavalla periaatteella kaksiulotteisessa aineistossa:

Vaihe 1.1: Keskipisteiden määrittäminen

Valitse n-määrä keskipisteitä aineiston datapisteistä. Keskipisteiden määrä kertoo kuinka monta klusteria aineistossa tulee olemaan (Mts. 226.)

Silmämääräisesti valitessa kannattaa arvioida kuinka monta klusteria aineistoissa tulisi olla katsomalla datapisteiden tiheyttä ja etäisyyttä toisiin. Yksittäisiin datapisteisiin klustereiden ulkopuolella ei kannata kiinnittää liikaa huomiota.

Vaihe 1.2: Etäisyyksien laskeminen

Datapisteet määritetään niitä lähimpään keskipisteeseen laskemalla etäisyys. Euklidinen etäisyys on yleisin menetelmä etäisyyden laskemiseksi kahden pisteen välillä. Yhtälö euklidisen etäisyyden laskemiseksi:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

jossa *x* on datapisteen x-akseli ja *y* on y-akseli. (Mts. 230.)

Vaihe 1.3: Keskipisteiden uudelleen laskeminen

Kun datapisteet on määritetty niitä lähimpiin klustereihin, niin klustereille voi laskea uuden keskipisteen kaavalla:

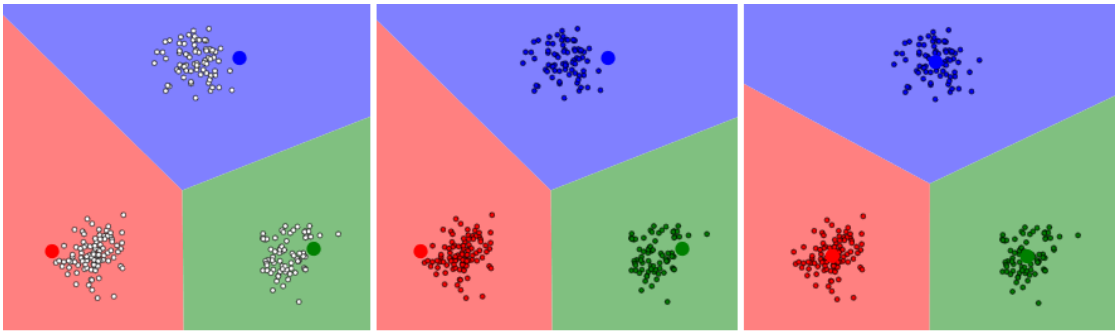
$$\mu_i = \frac{1}{j_i} \sum_{x \in c_i} X$$

jossa μ_i on klusterin keskipiste, j_i on datapiste, c_i on klusteri ja X on tässä tapauksessa kaksiulotteinen vektori, jossa on kaikki datapisteet. Uusi keskipiste klusterissa on siis kaikkien datapisteiden keskiarvo. (Mts. 231.)

Vaihe 1.4: Toista kunnes keskipisteet eivät liiku aineistossa

Vaiheet 1.2 ja 1.3 toistetaan niin kauan, kunnes klustereiden keskipisteet eivät enää muutu. (Mts. 232.)

Kuvio 5 havainnollistaa klusteroinnin eri vaiheet. Kuvassa keskipisteiden laskeminen onnistui alusta loppuun ensimmäisen iteraation aikana.



Kuvio 5. K-means algoritmin eri vaiheet visualisoituna (Harris 2014.)

2.5.2 Tilastolliset tekniikat

Minkä tahansa tilastollisen poikkeaman tunnistamismenetelmän periaate on, että poikkeama on havainto, jonka epäillään olevan osittain tai kokonaan merkityksetön, koska sitä ei synny oletetun stokastisen mallin avulla. Tilastollisten poikkeamien tunnistamistekniikat perustuvat seuraavaan keskeiseen oletukseen:

Oletus: *Normaalit datailmentymät esiintyvät stokastisen mallin suurilla todennäköisyysalueilla, kun taas poikkeamat esiintyvät stokastisen mallin pienen todennäköisyyden alueilla*

Tilastolliset tekniikat sopivat tilastomallin annettuihin tietoihin ja soveltavat sitten tilastollista päättelykoetta sen selvittämiseksi, kuuluuko näkymättömät ilmentymät tähän malliin vai eivät. Ilmentymät, joilla on matala todennäköisyys generoitua opitusta mallista sovelletun testitilaston perusteella, merkitään poikkeamaksi. Tilastolliset tekniikat voidaan jakaa kahteen eri poikkeamantunnistustekniikkaan: parametriset ja ei-parametriset tekniikat. (Chandola, Banerjee & Kumar, 2009 33.)

3 Aikasarja-analyysi

3.1 Johdanto aikasarja-analyysiin

Tämän opinnäytetyön tutkittavana kohteena on aineistot, jotka sisältävät tutkittavan datapisteen lisäksi aikaleiman, joten kaikki analysoiminen ja poikkeamantunnistus tapahtuu käyttäen aikasarja-analyysia ja ennustamista Anomaly Detectorin kautta.

Aikasarja-analyysin (engl. Time series analysis) käyttäminen yhdessä perinteisten poikkeamantunnistusmenetelmien kanssa on yleistynyt viime vuosina, koska suoratoiston kasvun yhteydessä on avautunut mahdollisuus reaaliaikaiseen datan keräämiseen ja analysoimiseen valvomattomilla havaitsemismenetelmillä. Varhainen poikkeamien havaitseminen on arvokasta, mutta se voi olla vaikea toteuttaa luotettavasti käytännössä. Sovellusrajoitukset edellyttävät järjestelmiä käsittelemään tietoja reaaliajassa, eikä erissä kuten tavanomaisessa poikkeaman tunnistamisessa. Tämän takia suoratoistetussa datassa suositaan jatkuvasti oppivia algoritmeja. (Ahmad, Lavin, Purdy & Agha 2017.)

Aikasarja-analyysin lisäksi on aikasarjaennustaminen (engl. Time series forecasting), jolla ennustetaan aikasarja-aineiston tulevaisuuden arvo aikaisempien havaintojen ja muiden syötteiden perusteella. Aikasarjaennuste on yksi vanhimmista tiedossa olevista ennakoivista analyysitekniikoista. Sitä käytetään laajalti kaikissa organisaatioissa ja sillä on syvälliset tilastolliset perusteet. (Kotu & Deshpande 2018, 401.)

Aikasarjassa, jossa on selkeitä trendejä tai kausiluonteisuutta, aika vaikuttaa arvoon. Aikasarjaa kutsutaan stationaariseksi, kun aikasarjojen arvo ei ole riippuvainen ajasta. Esimerkiksi satunnainen valkoinen kohina on stationaarinen aikasarja. Päivittäiset säähavainnot samassa paikassa eivät ole stationaarisia, koska suuntaus on kausiluonteinen, ja aika vaikuttaa siihen. Stationaarisia aikasarjoja ei voida ennustaa millään keinolla, koska ne ovat täysin satunnaisia. (Mts. 421.)

3.2 Aikasarjaennustaminen ARIMA-mallilla

ARIMA (Autoregressive Integrated Moving Average) on suosittu malli aikasarjojen ennustamiseen. ARIMA:n toiminta on monivaiheinen ja perustuu useaan ”rakennuspalikkaan”, mutta toteutus on kuitenkin aika suoraviivaista. (Kotu & Deshpande 2018, 418.)

Korrelaatiolla mitataan kuinka muuttujat ovat riippuvaisia toisistaan tai jos heillä on lineaarinen suhde toisiinsa. Tämä tapahtuu vertaamalla aikasarja-aineiston arvoja edellisiin arvoihin ja tällä yritetään etsiä muuttujia, joilla on vahva korrelaatio keskenään. Taulukossa 1 sarakkeen ”lämpötila” lisäksi on viivesarakkeet, jotka näyttävät edellisen vuosineljänneksen lämpötilan toisella sarakkeella seuraavalla vuosineljänneksellä. Eli aloitustilassa ensimmäisen mittauksen kohdalla viiveitä ei vielä ole merkitty, mutta toisen mittauskerran jälkeen ensimmäinen mitattu arvo lisätään viiveeksi 1. Kolmannella mittauskerralla toinen mitattu arvo päätyy viiveeksi 1 ja edellinen viive siirtyy yhdellä eteenpäin. Taulukosta voi huomata jo osittaista korrelaatiota mitattujen vuosien välillä. Lämpötilat aikaväliltä 2016Q1 ja 2016Q4 eivät eroa juurikaan viivesarjoista, jotka ovat merkitty samalla värillä taulukkoon. Tämä on kausiluonteinen havainto, mutta tässä asiayhteydessä sitä voidaan kutsua autokorrelaatioksi, koska joka neljännes viivesarja korreloi vuotuisen kausiluonteisuuden kanssa. (Mts. 419–420.)

Taulukko 1. Keskilämpötila neljännesvuotta kohden ja viiveet niiden perusteella. Luotu ilmatieteenlaitoksen säähavainnoista vuosilta 2015-2017 (Ilmatieteenlaitos.)

Neljännesvuosi	Lämpötila	Viive-1	Viive-2	Viive-3	Viive-4	Viive-5	Viive-6
2015 Q1	-3	?	?	?	?	?	?
2015 Q2	8.1	-3	?	?	?	?	?
2015 Q3	13.73	8.1	-3	?	?	?	?
2015 Q4	2.27	13.73	8.1	-3	?	?	?
2016 Q1	-6.27	2.27	13.73	8.1	-3	?	?
2016 Q2	10.07	-6.27	2.27	13.73	8.1	-3	?
2016 Q3	13.6	10.07	-6.27	2.27	13.73	8.1	-3
2016 Q4	-1	13.6	10.07	-6.27	2.27	13.73	8.1
2017 Q1	-3.9	-1	13.6	10.07	-6.27	2.27	13.73
2017 Q2	6.63	-3.9	-1	13.6	10.07	-6.27	2.27
2017 Q3	12.5	6.63	-3.9	-1	13.6	10.07	-6.27
2017 Q4	1	12.5	6.63	-3.9	-1	13.6	10.07

Autoregressiiviset mallit (AR)

Autoregressiiviset mallit ovat regressiomalleja, joita sovelletaan viiveisiin, jotka on luotu käyttämällä alkuperäisiä aikasarjoja. Palautettu tulos useilla lineaarisilla regressioilla on lineaarinen yhdistelmä useista tulomuuttujista. Autoregressiomallien tapauksessa lähtö on tuleva datapiste ja se voidaan ilmaista lineaarisena yhdistelmänä aikaisemmille *p*-datapisteille. *p* on viiveikkuna. Autoregressiivinen malli voidaan merkitä yhtälöksi:

$$y_t = l + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_p y_{t-p} + e$$

jossa y_t on laskettu arvo ajalle t, y_{t-1} on edellinen arvo, eli viivearvo. l on aineiston taso, α_1 on kerroin välillä -1–1 ja e on kohina-arvo jokaisen ajan kohdalla. (Mts. 420–421.)

Erottelu (I)

Epästationaarinen aikasarja voidaan muuntaa stationaariseksi aikasarjaksi tekniikalla, jota kutsutaan erotteluksi (engl. Differencing). Ero-sarja on muutos sarjan peräkkäisten datapisteiden välillä. (Mts. 422.)

$$y'_t = y_t - y_{t-1}$$

Tätä kutsutaan ensimmäisen asteen erotteluksi (engl. First order differencing). Joissakin tapauksissa erottelua tulee käyttää useamman kerran, jos aikasarja pysyy epästationaarisena. Silloin tarvitaan toisen asteen erottelua. Toisen asteen erotus on muutos kahden peräkkäisen datapisteen välillä ensimmäisessä järjestyksessä erotetussa aikasarjassa. (Mts. 422.)

Kausiluonteinen erottelu tarkoittaa muutosta saman jakson välillä kahdessa eri vuodenajassa. Yhtälö tämän laskemiseen muistuttaa ensimmäisen asteen erottelua:

$$y'_t = y_t - y_{t-m}$$

jossa m on vuodenajan pituus. Taulukko 1:sen esimerkissä m olisi 4. (Mts. 422.)

Virheen liukuva keskiarvo (MA)

Lisäksi regression luomiseen aikaisempien *p*-datapisteiden perusteella on mahdollista luoda regressioyhtälö, johon sisältyy ennustevirheitä aikaisemmasta datasta ja sitä voi käyttää ennustajana:

$$y_t = l + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$$

jossa e_i on datapisteen *i* ennustevirhe. Tämä käy järkeen edellisille datapisteille, mutta ei datapisteelle *t,* koska sitä ei olla ennustettu vielä. Siksi e_t oletetaan valkoiseksi kohinaksi. y_t:n regressioyhtälö voidaan ymmärtää menneiden q-ennusteiden virheiden painotettuna (θ) liukuvana keskiarvona. (Mts. 423.)

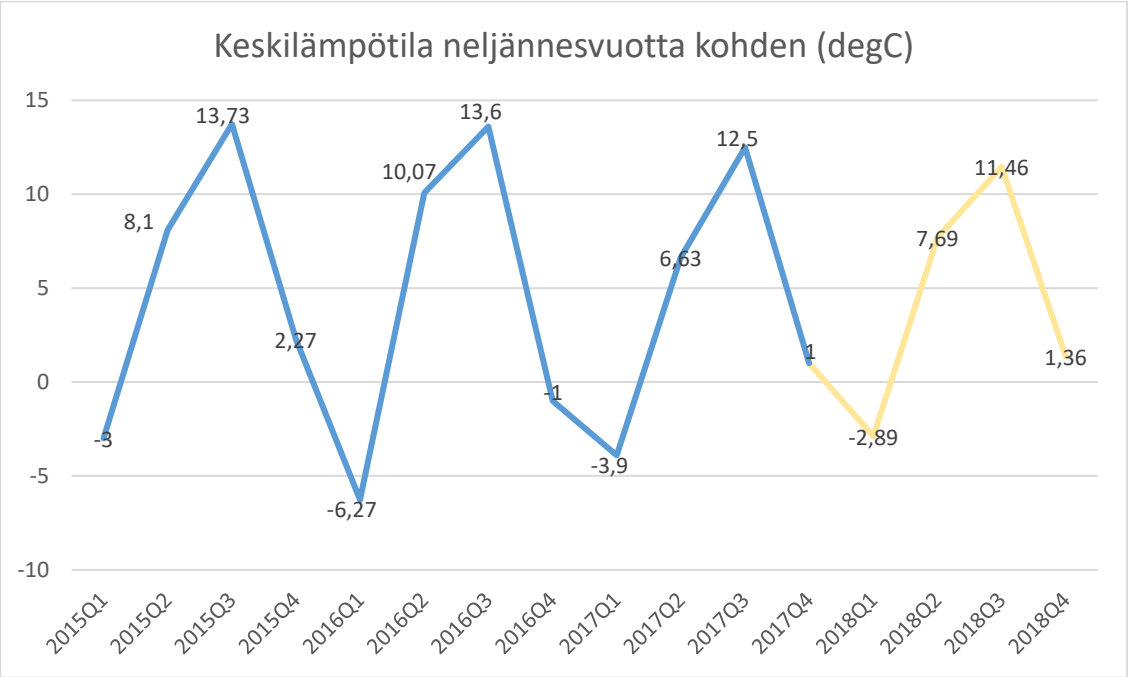
(AR)+(I)+(MA)

ARIMA-malli on yhdistelmä erotellusta autoregressiivisestä mallista liikkuvan keskiarvo -mallin kanssa. Nämä rakennuspalikat yhdistämällä saadaan kaava:

$$y'_t = I + \alpha_1 y'_{t-1} + \alpha_2 y'_{t-2} + \cdots + \alpha_p y'_{t-p} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$$

AR-osa osoittaa, että aikasarja on regressoitunut omaan aikaisempaan dataan. MA-osa osoittaa, että ennustevirhe on lineaarinen yhdistelmä aikaisemmista vastaavista virheistä. I-osa osoittaa, että data-arvot on korvattu *d*-asteen erottelulla stationaarisen datan saamiseksi, mikä on ARIMA-mallin vaatimus. (Mts. 423.)

Taulukko 1:sen arvojen syöttäminen ARIMA-malliin mahdollistaa lämpötilojen ennustamisen tulevaisuuteen. Kuviossa 6 keltaisella on merkitty ennuste vuodelle 2018.



Kuvio 6. ARIMA-mallin ennuste ilmatieteenlaitoksen keskilämpötiloista vuodelle 2018

4 Microsoft Anomaly Detector

4.1 Microsoft Anomaly Detectorin esittely

Anomaly Detector on Microsoftin vuonna 2019 julkaisema poikkeamantunnistus järjestelmä, joka on osa Azure Cognitive Service:ä. (Anomaly Detector is now available 2019.)

Anomaly Detector suunniteltiin skenaarioihin, joissa on tarvetta operatiiviseen tai reaaliaikaiseen aikasarja-aineistojen monitorointiin, kuten yrityksen reaaliaikainen suoritusindikaattorin seuraaminen ja IoT-monitorointi. Anomaly Detectorin vahvuus on sen helppokäyttöisyys ja nopea integroiminen käyttäjien omiin alustoihin. (Xing 2019.)

Anomaly Detector on täysin pilvessä toimiva järjestelmä, jonka käyttäminen tapahtuu kutsumalla käyttäjälle määritettyjä REST-rajapintoja. Palvelun käyttäminen edellyttää Microsoft-tiliä ja aktiivisen Azure-tilauksen Cognitive Service:n käyttämistä varten. Anomaly Detector on tyypiltään *stateless*, eli se ei muista käyttäjän aikaisemmin lähetettyjä aineistoja, eikä myöskään käytä niitä hyväksi kouluttaakseen Anomaly Detectoria myöhemmissä aineistoissa, joten kaikki poikkeamantunnistus tehdään yhden lähetetyn aikasarja-aineiston perusteella. (Best practices for using the Anomaly Detector API 2019.)

4.2 Anomaly Detectorissa käytetyt REST-rajapinnat

4.2.1 Anomaly Detector Azure Cognitive Service:ssa

Anomaly Detectorin käyttäminen tapahtuu kutsumalla REST-rajapintoja itse valitsemalla ohjelmointi- tai skriptauskielellä. Nämä REST-rajapinnat avautuvat käyttäjälle, kun Azure-tilaus on aktiivinen ja Azure Cognitive Serviceen on luotu oma resurssi Anomaly Detectoria varten. Azure Cognitive Service luo käyttäjälle oman aliverkkotunnuksen ja tästä muodostuu päätepiste, jonka kautta pääsee käsiksi Anomaly Detectorin rajapintoihin lähettämällä POST-pyyntöjä. Rajapintoja on tällä hetkellä kolme eri kappaletta: 1) sarjatunnistaminen (engl. Batch detection); 2) viimeisen datapisteen tunnistaminen (engl. Detection on the latest data point); ja 3) uusien datatrendien tunnistaminen (engl. Change point detection). (Quickstart: Detect anomalies in your time series data 2020.)

4.2.2 Sarjatunnistaminen

Käyttämällä tätä rajapintaa Anomaly Detector analysoi lähetetyn aikasarja-aineiston kokonaan ja lähettää vastauksessa löydetyt poikkeamat ja kaikki odotetut arvot jokaista datapistettä kohden. Microsoft suosittelee käyttämään sarjatunnistamista, jos aineisto sisältää kausiluonteisuutta satunnaisilla poikkeavuuksilla, tai jos aineiston datapisteet noudattavat enimmäkseen tasaista trendiä. (Best practices for using the Anomaly Detector API 2019; Detect Entire Series 2020.)

4.2.3 Viimeisen datapisteen tunnistaminen

Viimeisen datapisteen tunnistamisessa käytetty rajapinta vastaanottaa aineiston samalla tavalla kuin sarjatunnistamisessa, mutta Anomaly Detector lähettää vastauksessa nimensä mukaisesti vain tulokset viimeisestä datapisteestä. Jos aineiston 100 datapistettä halutaan tarkistaa käyttämällä tätä rajapintaa, niin silloin Anomaly Detectoriin on myös lähetettävä 100 kutsua. Jokaisen datapisteen tarkistuttaminen erikseen myös mahdollistaa sen, että Anomaly Detector voi valita sopivan algoritmin jokaista pistettä kohden, mikä ei onnistu sarjatunnistamisen kohdalla, koska kaikki aineiston pisteet tarkistetaan kerrallaan. Tämän rajapinnan kohdeyleisö on reaaliaikaista monitorointia tarvitsevat käyttäjät, tai sellaiset aikasarja-aineistot, jotka eivät noudata sarjatunnistamiseen suositeltuja käytäntöjä. (Best practices for using the Anomaly Detector API 2019; Detect Last Point 2020.)

4.2.4 Uusien datatrendien tunnistaminen

Uusien datatrendien tunnistaminen on viimeisin Microsoftin lisäämä rajapinta ja tässä vaiheessa siitä on todella vähän materiaalia löydettävissä verkossa. Tämä rajapinta eroaa hieman aikaisemmista ottamalla ylimääräisiä parametreja lähtevään kutsuun, mutta itse aikasarja-aineisto on samassa muodossa. Vastauksessa ei myöskään ilmoiteta poikkeamia, vaan pelkästään ne ajanhetket, kun aikasarja-trendi muuttui. (Detect Change Point 2020.)

4.3 Datan esikäsittely ja lähettäminen

4.3.1 Aineiston käsittely ennen lähettämistä

Ennen kuin aineisto lähetetään Anomaly Detectorin rajapintoihin on tärkeää tarkistaa lähtevän datan yhtenäisyys ja muoto. Aineisto tulee muuttaa tarpeeksi yksinkertaiseksi, jotta jokaisen datapisteen voi esittää numeerisena arvoja ja jokaisella datapisteellä pitää olla myös aikaleima. Datapisteiden aikaleimojen välillä oleva aika tulisi olla mahdollisimman tasainen, eli aikaleimat näkyisivät kuvaajana lineaarisena kasvuna. Jos aikasarja on täysin satunnainen, niin olisi suositeltavaa yrittää koota aikasarja-aineisto tiettyihin aikayksikköihin. (Best practices for using the Anomaly Detector API 2019.)

Puuttuvat datapisteet ovat yleisiä tasaisesti jakautuneissa aikasarja-aineistoissa, etenkin niissä, joissa on tiheä näytteenottoväli. Jos aineistosta puuttuu yli 10 % odotetuista datapisteistä, niin siinä tapauksessa olisi suositeltavaa täyttää aukkoja esimerkiksi lineaarisella interpoloinnilla tai liukuvalla keskiarvolla edellisten ja myöhempien datapisteiden perusteella. (Mt.)

Lähtevään kutsuun tulee määrittää aikasarja-aineisto muotoon, jossa on lista mitattuja arvoja aikaleimoilla ja parametrit, jotka helpottavat Anomaly Detectoria tulkitsemaan aineistoa. (Mt.)

Kuviossa 7 on muutettu kolmen datapisteen aineisto JSON-muotoon, jota Anomaly Detector osaa käsitellä. Näiden JSON-kenttien toiminnot ovat seuraavat:

- ***series*** on taulukko, joka ottaa vastaan aikaleiman *timestamp* universaaliaikana ja datapisteen *value* numeerisena arvona, joka voi sisältää desimaaleja. Jos *period* on määritetty, niin parhaat tulokset saadaan määrittämällä taulukon kooksi 4 * *period* + 1 (Detect Entire Series 2020; Mt.)
- ***maxAnomalyRatio*** kertoo Anomaly Detectorille, kuinka monta prosenttia datapisteistä voidaan ilmoittaa poikkeamana vastauksessa. Esimerkiksi jos aineistossa on 100 datapistettä ja *maxAnomalyRatio* on 0,1, niin silloin vastauk-

sessa voi olla korkeintaan 10 poikkeamaa. *maxAnomalyRatio* ei ole pakko lisätä kutsuun, mutta lisätessä sen pitää olla 0:n ja 0,5:n välillä (Mt.; Find anomalies for the entire series in batch 2019.)

- ***period*** on tärkeä lisätä kutsuun, jos aineistossa on kausiluonteisuutta ja jos tämän toistuvuus tiedetään karkeasti. Esimerkiksi jos toistuvuus tapahtuu seitsemän päivän välein ja aineiston kaikki datapisteet on merkitty tunnin välein, niin *period* tulisi olla 168. Luku saadaan kertomalla kaikki päivän tunnit seitsemällä päivällä. Parhaimmassa tapauksessa poikkeaman tunnistamisen viive saattaa tippua jopa 50 %, jos *period* on määritetty kutsuun. Mutta *period* on vain suositus ja sitä ei ole pakko lisätä kutsuun. (Detect Entire Series 2020.)
- ***sensitivity*** on kokonaisluku välillä 0–99 ja se kertoo Anomaly Detector:ille kuinka herkästi normaalista poikkeavat datapisteet tulisi merkitä poikkeamiksi. Suurempi arvo tarkoittaa sitä, että poikkeamia hyväksytään enemmän, kun taas pienemmällä arvolla poikkeavuuksia ei aina hyväksytä niin herkästi. Normaalisti Microsoftin dokumenteissa käytetään herkkyyttä, mikä on välillä 80–90. *sensitivity* ei ole pakollinen kenttä. (Mt.)
- ***granularity***:lla vahvistetaan, että päteekö aineiston aikaleimojen välillä oleva aika merkittyyn arvoon. *granularity* on tekstimuodossa ja se voi olla jokin seuraavista: *daily*, *hourly*, *minutely*, *monthly*, *secondly*, *weekly* tai *yearly.* Jos datapisteet on mitattu tai generoitu aina minuutin välein, niin silloin *granularity* on myös *minutely*. (Mt.)
- ***customInterval***:ia käytetään asettamaan aineiston aikaleimojen välillä esiintyvä epätyypillinen aikaväli. Esimerkiksi jos datapisteet on mitattu viiden tunnin välein, niin silloin *customInterval* on 5 ja *granularity* merkitään tekstiarvoksi *hourly*. (Mt.)


```
{
  "series": [
    {
      "timestamp": "2020-04-02T00:00:00Z",
      "value": 70
    },
    {
      "timestamp": "2020-04-02T00:01:00Z",
      "value": 68
    },
    {
      "timestamp": "2020-04-02T00:02:00Z",
      "value": 71
    }
  ],
  "maxAnomalyRatio": 0.1,
  "period": 0,
  "sensitivity": 90,
  "granularity": "minutely",
  "customInterval": 1
}
```


Kuvio 7. Anomaly detectoriin lähtevä kutsu JSON-muodossa

4.3.2 Datan lähettäminen

Ennen lähettämistä tulee tarkistaa, että onko lähtevässä datassa sallittu määrä datapisteitä. Datapisteitä pitää olla vähintään 12 ja korkeintaan 8640. (Find anomalies for the entire series in batch 2019.)

Data lähetetään kuvio 7:n tapaan JSONina Microsoft-tilille määritettyyn päätepisteeseen ja perään laitetaan haluttu rajapinta kuvio 8:n esitetyssä muodossa. POST-pyynnön otsikkotietoihin pitää määrittää Anomaly Detector API -avain, joka luodaan automaattisesti käyttäjälle, kun Anomaly Detector otetaan käyttöön. (Quickstart: Detect anomalies in your time series data 2020.)

Batch detection	`/anomalydetector/v1.0/timeseries/entire/detect`
Detection on the latest data point	`/anomalydetector/v1.0/timeseries/last/detect`
Change point detection	`/anomalydetector/v1.0/timeseries/changepoint/detect`

Kuvio 8. Anomaly Detectorin rajapinnat (Quickstart: Detect anomalies in your time series data 2020.)

4.4 Datan vastaanottaminen ja tulkinta

4.4.1 Datan vastaanottaminen

Anomaly Detectorin vastauksessa on JSON-muotoinen teksti, joka on sarja- ja viimeisen datapisteen tunnistamisessa muuten sama, mutta sarjatunnistamisen vastauksessa arvot ovat taulukossa jokaista mitattua datapistettä kohden, kun taas jälkimmäisen rajapinnan vastaus sisältää vain yhdet arvot viimeiselle mittaukselle. Ennen seuraavaa vaihetta kannattaa tarkistaa, että sisältääkö vastaus JSON-objektin nimellä *code*, sillä se viittaa epäonnistuneeseen operaatioon. (Quickstart: Detect anomalies in your time series data 2020.)

4.4.2 Vastauksen tulkinta

Vastaus Anomaly Detectorilta viimeistä datapistettä tunnistaessa on muodoltaan seuraavanlainen:

- ***expectedValue*** on odotettu desimaaliarvo mitatulle datapisteelle, jonka Anomaly Detector päättelee syötetyn aineiston datapisteiden perusteella. *expectedValue* harvemmin täsmää mitattua arvoa, mutta pieni eroavuus ei haittaa, eikä yleensä myöskään tee datapisteestä automaattisesti poikkeamaa. (Detect Last Point 2020.)
- ***isAnomaly*** on tyyppiä boolean ja se ilmoittaa, että oliko kyseinen datapiste poikkeama vai ei. (Mt.)
- ***isNegativeAnomaly*** on tyyppiä boolean ja se on *true* vain, jos datapiste todettiin poikkeamaksi ja sen arvo oli pienempi kuin odotettu arvo. (Mt.)
- ***isPositiveAnomaly*** on tyyppiä boolean ja se on *true* vain, jos datapiste todettiin poikkeamaksi ja sen arvo oli suurempi kuin odotettu arvo. (Mt.)
- ***lowerMargin*** on desimaaliarvo, jolla Anomaly Detector laskee alarajan datapisteelle. Jos mitattu datapiste on pienempi kuin alaraja, niin se on poikkeama. (Mt.)
- ***upperMargin*** on desimaaliarvo, jolla Anomaly Detector laskee ylärajan datapisteelle. Jos mitattu datapiste on suurempi kuin yläraja, niin se on poikkeama. (Mt.)
- ***period*** on kokonaisluku ja se ilmoittaa, että löysikö Anomaly Detector syötetystä aineistoista toistuvuutta ja kuinka monen datapisteen välein toistuvuus tapahtuu. Jos toistuvuutta ei löydetty, niin arvo on 0. (Mt.)
- ***suggestedWindow*** on Anomaly Detectorin suosittelema määrä datapisteitä viimeisen datapisteen tunnistamista varten. Jos *period* ei ole 0, niin yleensä *suggestedWindow* on neljä kertaa se + 1.Havainto perustuu Microsoftin itse suosittelemaan tapaan määrittää arvo *period* lähteviin kutsuihin, eli neljä kertaa määritetty *period*. Tämä kenttä ei tule vastaukseen sarjatunnistamisessa. (Detect Last Point 2020; Best practices for using the Anomaly Detector API 2019.)

Kuviossa 9 on generoitu aikasarja-aineisto, jossa on Anomaly Detectorin pienin sallittu määrä datapisteitä. Aineistolle on annettu kuvan parametrit ja se on lähetetty viimeisen datapisteen tunnistamisen -rajapintaan. Palautettu vastaus näkyy kuvan oikealla puolella ja siinä on todettu viimeisimmän datapisteen olevan positiivinen poikkeama.

timestamp	value
2020-04-01T09:01:00Z	71
2020-04-01T09:02:00Z	68
2020-04-01T09:03:00Z	67
2020-04-01T09:04:00Z	68
2020-04-01T09:05:00Z	73
2020-04-01T09:06:00Z	70
2020-04-01T09:07:00Z	69
2020-04-01T09:08:00Z	67
2020-04-01T09:09:00Z	71
2020-04-01T09:10:00Z	71
2020-04-01T09:11:00Z	73
2020-04-01T09:12:00Z	87

maxAnomalyRatio	0,10
period	0
sensitivity	90
granularity	minutely
customInterval	1

/anomalydetector/v1.0/timeseries/last/detect


```
{
  "expectedValue": 69.6,
  "isAnomaly": true,
  "isNegativeAnomaly": false,
  "isPositiveAnomaly": true,
  "lowerMargin": 1.7399999999999949,
  "period": 0,
  "suggestedWindow": 1441,
  "upperMargin": 1.7399999999999949
}
```


Kuvio 9. Aikasarja-aineisto 12 datapisteellä ja vastaus viimeisen datapisteen tunnistamisen rajapinnasta

4.5 Anomaly Detectorin parametrien vaikutus

Edellisillä sivuilla käsiteltiin Anomaly Detectoriin lähteviä ja vastaanotettuja parametreja rajapintakutsuissa, mutta niiden vaikutus itse poikkeaman tunnistamisessa on toinen tutkimisen arvoinen kohde.

Herkkyyden määrittelemällä lähtevään kutsuun voi vaikuttaa poikkeamien ylä- ja alarajoihin. Ylärajan laskeminen tapahtuu seuraavalla laskutoimituksella:

$$U_B = E_V + (100 - S) * U_M$$

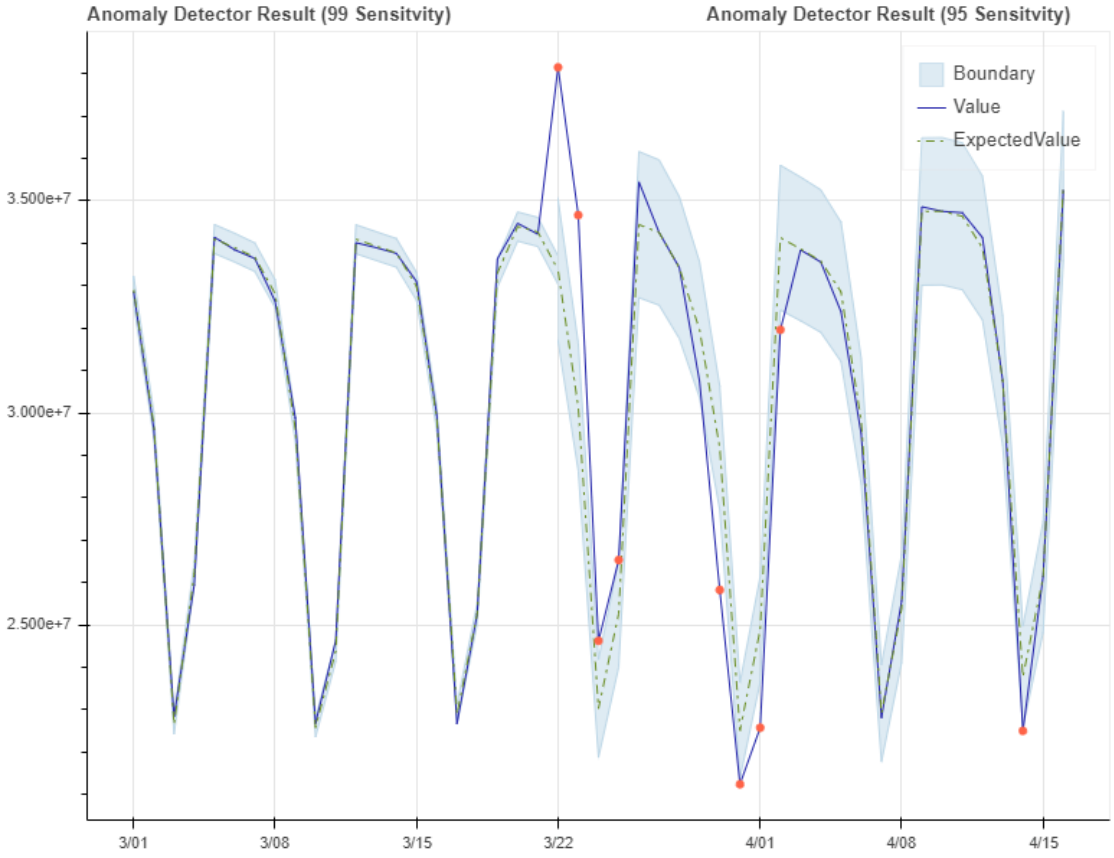
jossa U_B on yläraja (*UpperBoundary*), E_V on odotettu arvo (*ExpectedValue*), S on herkkyys (*Sensitivity*) ja U_M on ylämarginaali (*UpperMargin*). (Detect Last Point 2020.)

Vastaavasti alarajan voi laskea laskutoimituksella:

$$L_B = E_V - (100 - S) * L_M$$

jossa L_B on alaraja (*LowerBoundary*) ja L_M on alamarginaali (*LowerMargin*). (Mt.)

Kuvio 10 havainnollistaa herkkyyden vaikutusta Anomaly Detectorin toiminnassa käyttäen kahta identtistä aineistoa. Kuvan vasemmalla puolella herkkyysparametri on asetettu arvoksi 99 ja kuvan oikealla puolella se on 95. Vaaleansininen alue kuvassa kuvastaa ylä- ja alarajaa, joidenka sisällä ollessa mitattu arvo ei ole poikkeama. Pieni muutos herkkyyteen on vaikuttanut merkittävästi tämän alueen kokoon, mikä mahdollisesti selittää miksi Microsoftin dokumenteissa toistuu korkea herkkyys esimerkeissä.



Kuvio 10. Microsoftin itse tuottama aineisto kahdella eri herkkyydellä ja havaitut poikkeamat. Kuvat on muokattu yhteen (How to: Use the Anomaly Detector API on your time series data 2019.)

4.6 Anomaly Detectorissa käytettyjä algoritmeja

4.6.1 Microsoftin mainitsemat algoritmit

Microsoft ei ole paljastanut ulkopuolisille paljoa käyttämistään algoritmeista, mutta seuraavat algoritmit on mainittu Microsoftin blogissa: (Introducing Azure Anomaly Detector API 2019.)

- Fourier Transformation
- Extreme Studentized Deviate (ESD)
- STL Decomposition
- Dynamic Threshold
- Z-score detector
- SR-CNN

Näistä ainoastaan SR-CNN on kuvailtu yksityiskohtaisesti, sillä Anomaly Detectorin kehittäjätiimi julkaisi tutkielman algoritmista (Ren, Xu, Wang, Yi, Huang, Kou, Xing, Yang, Tong & Zhang 2019.).

4.6.2 Rajallisesti tiedetyt algoritmit

Fourier Transformation

Fourier transform (suomeksi Fourier-muunnos) on monikäyttöinen matemaattinen työkalu differentiaaliyhtälöiden ratkaisemiseen tai signaalien analysoimiseen. Fourier-muunnos hajottaa syötetyn signaalin taajuuskomponentteihin, joka mahdollistaa mm. kohinan poistamisen signaalista. (Mila 2019.)

Extreme Studentized Deviate

Extreme Studentized Deviate on tilastollinen testi, joka havaitsee poikkeavuuksia annetuista datapisteistä käyttämällä Grubbs'n testiä *k*-kertaa, jossa *k* on oletettu määrä poikkeamia aineistossa. (Zaiontz N.d.)

STL Decomposition

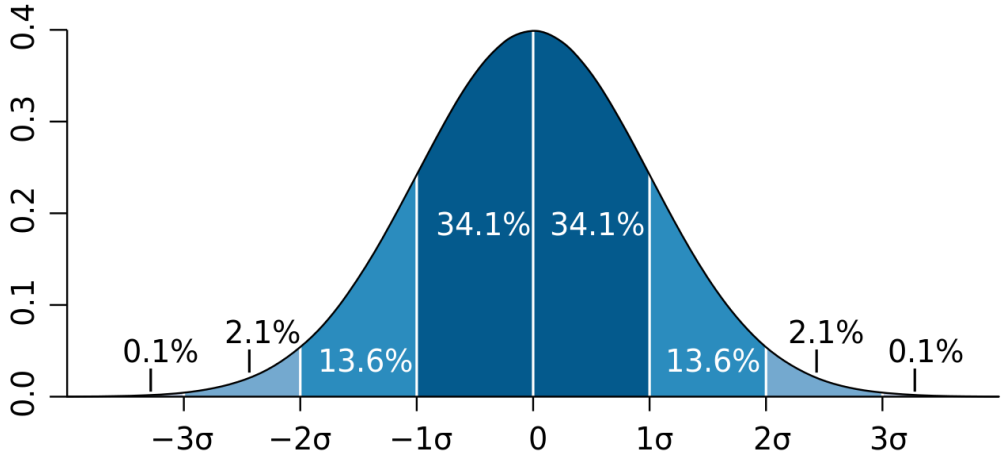
STL, eli “Seasonal and Trend decomposition using Loess” on menetelmä aikasarjojen hajottamiseen komponentteihin, jotka ovat tyypillisesti aikasarjan trendi, kausiluonteisuus ja kahden edellisen komponentin poistamisen jäljiltä jäänyt kohinakomponentti. (Hyndman & Athanasopoulos 2018, luku 6.6.)

Dynamic Threshold

Dynamic Threshold:lla tarkoitetaan tilastollisessa analyysissa muutoksen havaitsemisena dynaamisen järjestelmän ominaisuuksissa tietyn ajan kuluessa. Anomaly Detectorin kohdalla dynaaminen järjestelmä kuvastaa rajapintoihin saapunutta aikasarja-aineistoa sillä hetkellä. (Kalmuk, Granichin, Granichina & Ding 2016.)

Z-score detector

Z-score, eli Z-piste on vakiopiste, joka kertoo datapisteen etäisyyden muiden datapisteiden keskiarvosta ja onko sen keskiarvo suurempi vai pienempi muihin nähden. Toisin sanoen Z-piste kertoo, että kuinka monta standardipoikkeamaa datapiste on keskiarvosta. (McLeod 2019.) Kuviossa 11 on tyypillinen esitys standardipoikkeamista.



Kuvio 11. Standardipoikkeamat tai keskihajoamat normaalijakautumassa (Toews 2005.)

4.6.3 SR-CNN

Microsoft on antanut ymmärtää, että Anomaly Detector on rakennettu SR-CNN-algoritmin ympärille, sillä tämä on ainoa algoritmi, jota Microsoft on aktiivisesti mainostanut blogeissaan ja julkaisemassaan tutkielmassa koskien poikkeaman tunnistamista Microsoftin palveluissa. (Introducing Azure Anomaly Detector API 2019.)

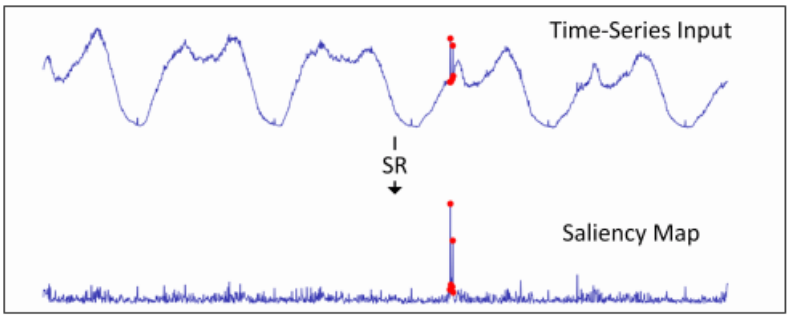
SR-CNN, eli Spectral Residual and Convolutional Neutral Network (suomeksi Spektraalinen jäännös- ja konvoluutiohermoverkko) on Anomaly Detectorin kehitystiimin ehdoittama algoritmi, joka perustuu spektraalijäämiin ja konvoluutiohermoverkkoon. Ehdoitetussa algoritmissa yritetään integroida SR-malli salituditunnistamisen (engl. Saliency detection) alueesta aikasarjojen poikkeaman tunnistamiseen ja algoritmiin yhdistetään myös CNN suorituskyvyn parantamiseksi. (Ren ym. 2019.)

Spektrinen jäännösalgoritmi (SR)

Spektrinen jäännösalgoritmi koostuu kolmesta päävaiheesta: (Mt. luku 4.1.)

1. Fourier-muunnos log-amplitudispektrin saamiseksi
2. Spektrijäännöksen laskeminen
3. Käänteinen Fourier-muunnos, joka muuttaa sekvenssin takaisin spatiaaliseksi alueeksi

Kuviossa 12 on esimerkki alkuperäisestä aikasarjasta ja vastaavasta salitudikartasta SR-käsittelyn jälkeen. ”Innovaatiopiste” (näkyy punaisella) salitudikartassa on paljon merkittävämpi kuin alkuperäisessä syötteessä.



Kuvio 12. Salitudikartta aikasarjasta SR-algoritmilla (Mt. Figure 4.)

Kun salitudikartta on tiedossa, niin tulosekvenssi *O(x)* voidaan laskea seuraavasti:

$$O(x_i) = \begin{cases} 1, & jos\ \dfrac{S(x_i) - \overline{S(x_i)}}{S(x_i)} > \tau, \\ 0, & muuten, \end{cases}$$

jossa x_i on satunnainen piste sarjassa x ja $S(x_i)$ on vastaava piste salitudikartassa. $\overline{S(x_i)}$ on edellisten z-pisteiden paikallinen keskiarvo $S(x_i)$:stä. (Mt.)

Algoritmin odotetaan löytävän poikkeamat matalalla viiveellä sekvenssin liukuvassa ikkunassa, kun otetaan huomioon datapisteet x_1, x_2.., x_n, jossa x_n on viimeisin piste. Mutta SR-algoritmi toimii paremmin, jos kohdepiste sijaitsee liukuvan ikkunan keskellä. Siksi x_n:n jälkeen lisätään useita arvioituja pisteitä, ennen kuin sekvenssi syötetään SR-malliin. Ennuste datapisteelle x_{n+1} lasketaan kaavoilla:

$$\bar{g} = \frac{1}{m}\sum_{i=1}^{m} g(x_n, x_{n-1})$$

$$x_{n+1} = x_{n-m+1} + \bar{g} * m$$

jossa $g(x_i, x_j)$ osoittaa pisteen x_i ja x_j välisen suoran kaltevuutta ja $\bar{g}$ kuvastaa edellisten pisteiden keskimääräistä kaltevuutta. m on tarkasteltujen edellisten pisteiden määrä ja kuvion 12 esimerkissä se on asetettu arvoksi 5. Ensimmäisellä arvioidulla datapisteellä on ratkaiseva merkitys ja siksi piste x_{n+1} vain kopioidaan *K*-kertaa ja lisätään sekvenssin loppuun. (Mt.)

SR-CNN arkkitehtuuri

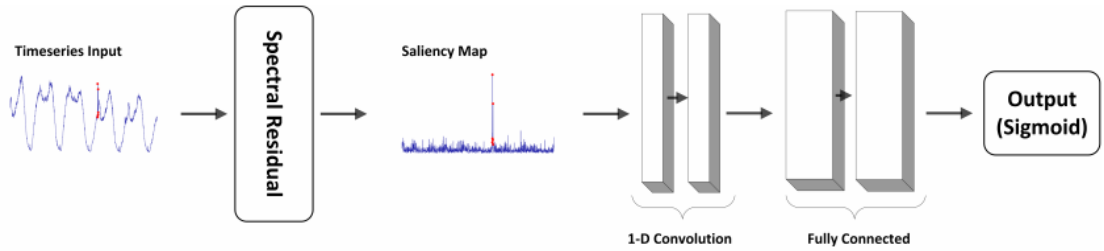
Alkuperäinen SR-menetelmä käyttää yhtä kynnystä salitudikartassa poikkeavuuskohtien havaitsemiseksi, mutta tämä sääntö on kuitenkin sen verran naiivi, että on parempi etsiä kehittyneempiä päätöksentekosääntöjä. Ratkaisu on kouluttaa ehdollinen malli (engl. Conditional model) hyvin suunnitelluista synteettisistä tiedoista poikkeamien tunnistajaksi. Synteettiset tiedot voidaan tuottaa lisäämällä poikkeavuuspis-

teitä salitudikarttojen kokoelmaan, jotka eivät sisälly arvioituun dataan. Lisätyt pisteet on merkitty poikkeamiksi, kun taas toiset on merkitty normaaliksi. Konkreettisesti tämä tarkoittaa sitä, että aikasarja-aineistosta valitaan satunnaisesti useita pisteitä, lisätty piste lasketaan korvaamaan alkuperäinen piste ja lopulta saadaan salitudikartta. Poikkeavuuspisteiden arvot lasketaan seuraavasti:

$$x = (\bar{x} + mean)(1 + var) * r + x$$

jossa $\bar{x}$ on paikallinen keskiarvo edeltävistä pisteistä, $mean$ tarkoittaa liukuvan ikkunan kaikkien pisteiden keskiarvoa ja var liukuvan ikkunan pisteiden varianssia. $r \sim \mathrm{N}(0, 1)$ valitaan satunnaisesti. (Mt. luku 4.2.)

CNN on yleisesti käytetty valvottu malli salituditunnistamisessa, mutta Anomaly Detectorin kohdalla syötetyssä datassa ei ole tarpeeksi merkittyä tietoja, niin CNN:ää käytetään salitudikartoilla raa'an datan sijasta, mikä tekee poikkeamien merkinnän ongelmasta huomattavasti helpompaa. Käytännössä tämä tarkoittaa sitä, että harjoitteludatana toimii syötetyt aikasarjat, joissa on synteettisiä poikkeavuuksia. SR-CNN:n arkkitehtuuri on rakennettu kuvion 13 mukaisesti: verkko koostuu kahdesta 1-D-konvoluutiokerroksesta (suodattimen koko on yhtä suuri kuin liukuvan ikkunan koko ω) ja kahdesta täysin yhdistetystä kerroksesta. Ensimmäisen konvoluutiokerroksen kanavan koko on yhtä suuri kuin ω samalla kun kanavan koko kaksinkertaistetaan toisessa konvoluutiokerroksessa. Kaksi täyttä liitettyä kerrosta on pinottu ennen Sigmoid-lähtöä. Ristientropia (engl. Cross entropy) hyväksytään häviöfunktioksi ja SGD-optimoijaa käytetään koulutusprosessissa. (Mt.)



Kuvio 13. SR-CNN arkkitehtuuri (Mt. Figure 5)

5 Työn tutkittava kohde ja suunnittelu

5.1 Tutkittava kohde

Työn tutkittavana kohteena on analysoida älykellolla mitattuja yöllisiä sykearvoja ranteesta Anomaly Detectorin avulla ja tämän lisäksi selvittää, että soveltuuko Anomaly Detector toimeksiantajan käyttötarkoituksiin. Hyvinvointitietojen keräys ja analysointi on aikasarjojen poikkeaman tunnistamisessa todella yleistä, sillä käyttökohteita on paljon niin lääke- ja liikuntatieteen puolella, mutta tutkittavan kohteen valinta perustui täysin aikaisempiin kokemuksiin hyvinvointitietojen keräämisestä älykelloilla.

Yöllisillä sykearvoilla selvitetään omaa unenlaatua käyttämällä aineistona usean eri yön mittauksia ja sitten syöttämällä ne Anomaly Detectoriin. Työssä yritetään etsiä vastauksia seuraaviin kysymyksiin:

- Onko sykearvoissa kausiluonteisuutta tai selkeitä trendejä?
- Vaikuttaako levottomuus poikkeamien määrään?
- Näkyykö unisyklit tai niistä poistuminen tuloksissa?
- Kuinka Anomaly Detectorin parametrit ja aineiston esikäsittely vaikuttaa lopputulokseen?
- Kuinka eri rajapintojen vastaukset eroavat toisistaan?

5.2 Ohjelmointiympäristö

5.2.1 Visual Studio ohjelmointiympäristönä

Työn kirjoittaminen tapahtuu Visual Studio 2019 Professional -ohjelmointiympäristössä, koska kaikki työhän valitut ohjelmistokehitysalustat kuuluvat Microsoftin .NET ohjelmistokomponenttikirjaston alaisuuteen. Ohjelmointikieleksi on valittu C#.

Visual Studio on Microsoftin kehittämä ohjelmistonkehitysympäristö Windows- ja Mac-tietokoneille. Microsoftin .NET-alusta mahdollistaa mm. graafisten käyttöliittymien, verkkosivujen, konsoli-, Unity- ja mobiilisovelluksien luomisen. (Develop .NET applications n.d.)

5.2.2 Ohjelmistokehitysalustat

Tizen

Sykearvojen keräämisessä käytetty Samsungin älykello toimii Tizen-käyttöjärjestelmällä. Tizen on erittäin joustava käyttöjärjestelmä, koska se on rakennettu Linuxin päälle. Samsung käyttää itse Tizeniä televisioissa, älykelloissa, kameroissa ja jopa jääkaapeissaan. (Introduction to Tizen n.d.)

Tizen sovelluksien kehittäminen tehdään joko Tizenin omassa Tizen Studiossa, tai Visual Studiossa, mutta työhön valittu ohjelmointikieli ei sovellu Tizen Studioniin, koska Tizenin oma IDE on rakennettu Eclipse-ohjelmointiympäristöön ja täten tarvitsee toimiakseen JDK:n (Java Development Kit) (Tizen Studio n.d.).

Työssä käytetään Tizen Service App -palvelusovellusta, joka pyörii älykellon taustalla ilman minkäänlaista käyttöliittymää.

.NET Core Console Application

Aineistojen lähettäminen Anomaly Detectorin rajapintoihin tapahtuu monialustaisessa komentokehote-sovelluksessa. Samassa sovelluksessa myös aineistot muutetaan tuettuun JSON-muotoon ennen lähettämistä.

WPF

WPF (Windows Presentation Foundation) on ohjelmistokehitysalusta työpöytäsovelluksien rakentamiseen graafisilla käyttöliittymillä. WPF käyttää XAML-merkintäkieltä käyttöliittymien luomisessa. (Windows Presentation Foundation n.d.)

Työssä käytetään WPF-käyttöliittymäsovellusta kaavioiden piirtämiseen, jolloin ei tarvitse manuaalisesti syöttää aineistoja ja Anomaly Detectorin vastauksia Exceliin. Kaavioiden piirtämisen apuna käytetään LiveCharts nuget-pakettia.

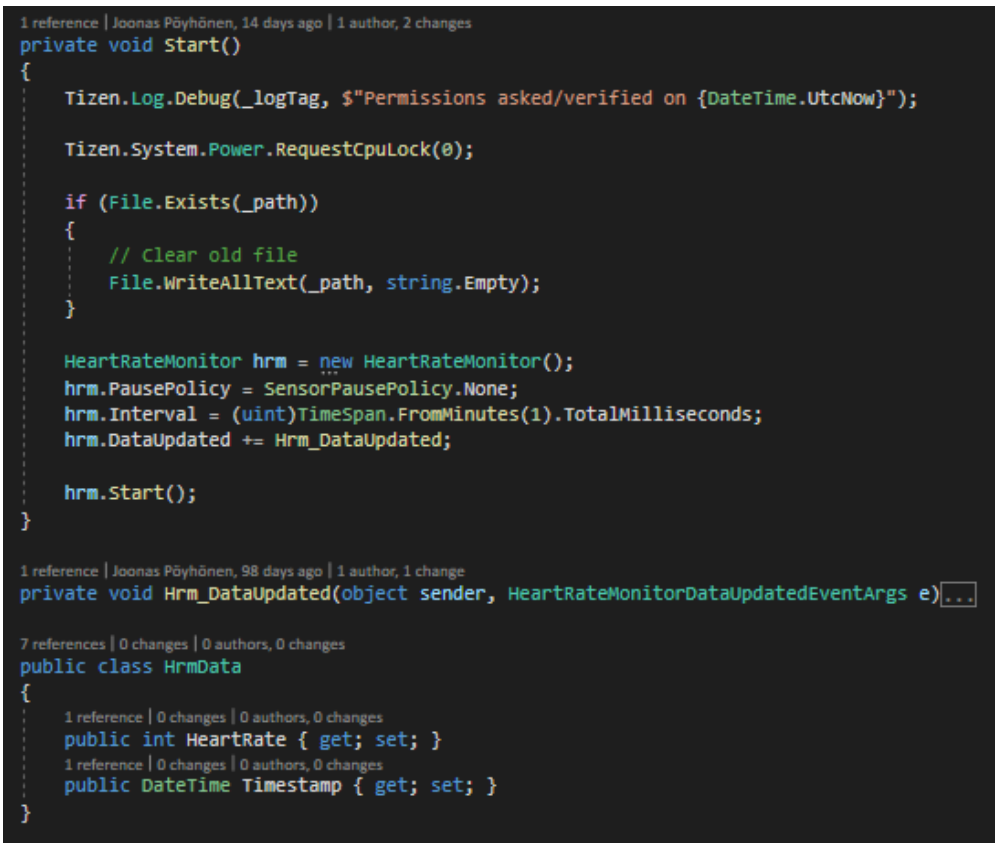
6 Työn toteutus ja tulokset

6.1 .NET työympäristöjen pystyttäminen

6.1.1 Tizen palvelusovellus

Palvelusovellusta varten luotiin uusi Tizen Service App -projekti Visual Studiossa. Työssä tarvittava koodipuoli oli tehty todella yksinkertaiseksi, koska sen ainoa toiminto oli kerätä älykellon sykemittarilla tietyn ajan välein sykedataa ja tallentaa ne tiedostoon.

Kuviossa 14 on tiivistetty sovelluksen logiikka. Palvelusovelluksen taustalla pitäminen edellyttää prosessorin lukitsemista, koska muuten sovellus menee nukkumaan ja tiedostoon tallentaminen ei onnistu. _path-muuttujaan on tallennettu polku- ja tiedostonimi sykedataa sisältävälle JSON-tiedostolle ja se tyhjennetään ennen kuin sykeanturi aloittaa mittaamisen. *HeartRateMonitor* on Tizen-järjestelmän käyttämä sykemittari ja sen käyttäminen edellyttää uuden ilmentymän luokasta, koska antureita voi olla enemmän kuin 1. Sykemittariin asetetaan *PausePolicy*-arvoksi 0, jolloin mittari toimii myös näytön ollessa pois päältä. Sykemittarin mittausväli on 1 minuutti ja seuraavalla rivillä määritetään takaisinkutsufunktio sykemittarin tapahtumankäsittelijään, jossa kuvion ulkopuolella sykearvo tallennetaan tiedostoon. Lopuksi sykemittari käsketään menemään päälle. Kuviossa näkyy myös *HrmData*-luokka, jonka yksi ilmentymä vastaa yhtä datapistettä aineistoissa. Tämän luokan ilmentymät tallennetaan _path-muuttujan osoittamaan JSON-tiedostoon ja myöhemmin käsitellään ennen Anomaly Detectoriin lähettämistä.



Kuvio 14. Tizen palvelusovelluksessa käytetty koodi sykemittarin alustamiseen

6.1.2 Komentokehote-sovellus

Komentokehote-sovellus toteutettiin .NET Core Console App -projektina. Sovelluksen lähetys- ja vastauslogiikka oli suurin piirtein tehty vastaamaan Microsoftin C# esimerkkiä (Quickstart: Detect anomalies in your time series data 2020), mutta sekaan oli lisäilty tarpeelliset koodit sykedatan esikäsittelyyn, aineiston muuttamiseen Anomaly Detectorin vaatimaan JSON-muotoon ja lopulta POST-pyynnön vastauksen käsittelyyn ja tallentamiseen JSON-tiedostoon.

Kuvio 15:ssa näkyy karkeasti kaikki askeleet sykedatan esikäsittelyssä. Esikäsittelyfunktio sisältää kolme vaihtoehtoista parametria, jotka käyttävät vakioarvoa, jos parametria ei syötetä. Aikasarjan aikaväli, eli *datasetGranularity* ja aikavälin arvo numerona *datasetInterval* tarkistetaan piilotetussa koodissa syötetystä sykedatasta. Tarvittaessa sykedataan lisätään puuttuvat datapisteet lineaarisella interpolaatiolla (ks. liite 1). Puuttuviksi datapisteiksi lasketaan myös nolla-arvot, eli mittauskerrat, joista

anturi ei saanut kunnollista lukemaa. Jos sykedatasta löytyy identtisiä arvoja puuttuvien datapisteiden lisäämisen jälkeen, niin ne hoidetaan seuraavaksi. *AnomalyApiRequest*-luokan ilmentymä *model* vastaa kuvion 7 JSON:ia, kun se lähetetään Anomaly Detectorin rajapintoihin. Funktion lopussa lisätään vielä juuri luotuun ilmentymään kaikki sykedatan datapisteet ja aikaleimat muutetaan oikeaan muotoon.

```
1 reference
public static AnomalyApiRequest ConvertRawDataWithDefaultSettings(List<HeartRateData> heartRateData,
                                                                  bool fillEmptyDatapoints = true,
                                                                  int customInterval = 1,
                                                                  int sensitivity = 90,
                                                                  int period = 0)
{
    AnomalyGranularity datasetGranularity = AnomalyGranularity.Minutely;
    int datasetInterval = 0;

    Get granularity from dataset

    if (fillEmptyDatapoints)
    {
        // Optionally fill empty data points
        FillEmptyDataPoints(ref heartRateData);
    }
    else
    {
        // Remove empty data points
        RemoveEmptyDatapoints(ref heartRateData);
    }

    // Remove all duplicate timestamp data points
    heartRateData = heartRateData.Distinct().ToList();

    string granularityName = Enum.GetName(typeof(AnomalyGranularity), datasetGranularity).ToLower();

    AnomalyApiRequest model = new AnomalyApiRequest
    {
        Granularity = granularityName,
        Period = period,
        CustomInterval = datasetInterval,
        MaxAnomalyRatio = 0.25,
        Sensitivity = sensitivity,
    };

    foreach (var data in heartRateData)
    {
        model.Series.Add(new AnomalyApiRequest.Point
        {
            Timestamp = DateTimeHelper.ConvertDateTimeToIso8601(data.Timestamp),
            Value = data.HeartRate
        });
    }

    return model;
}
```

Kuvio 15. Sydedatan esikäsittely

Esikäsittelyn jälkeen *AnomalyApiRequest*-luokan ilmentymä lähetetään kuviossa 16 Anomaly Detectorin sarjatunnistamisen rajapintaan ja vastaus tallennetaan tiedostoon siten, että tiedostoon menee JSON-pyyntö ja sen lisäksi rajapinnasta tullut vastaus.

```
2 references
public static void DetectAnomaliesBatch(AnomalyApiRequest requestData)
{
    if (requestData == null)
    {
        Console.WriteLine("requestData is null");
        return;
    }

    string data = requestData.Serialize();

    Console.WriteLine("Detecting anomalies as a batch");

    //construct the Anomaly-api request
    var result = Request(
        azureEndPoint,
        batchDetectionUrl,
        anomalyApiKey,
        data).Result;

    dynamic jsonObj = JsonConvert.DeserializeObject(result);
    Console.WriteLine(jsonObj);

    if (jsonObj["code"] != null)
    {
        Console.WriteLine($"Detection failed. ErrorCode:{jsonObj["code"]}," +
                          $" ErrorMessage:{jsonObj["message"]}");
    }
    else
    {
        // deserialize data to AnomalyBatchResponse
        AnomalyBatchResponse batchResponse = result.Deserialize<AnomalyBatchResponse>();

        if (batchResponse == null)
        {
            Console.WriteLine("Error, batchResponse is null");
            return;
        }

        string path = Program.GetParentOfPath(Directory.GetCurrentDirectory(), 5) +
                      Path.DirectorySeparatorChar + "datasetit" +
                      Path.DirectorySeparatorChar + "processed" + Path.DirectorySeparatorChar;

        DateTime date = requestData.Series.First().Timestamp;

        File.WriteAllText(path + Constants.AnomalyDataFile($"batch-{date:ddMMyyyy_HH.mm}"),
                          new AnomalyDataFile(requestData, batchResponse).Serialize());
    }
}
```

Kuvio 16. Aineiston lähettäminen ja vastauksen tallentaminen

6.1.3 Graafinen käyttöliittymä kaavioille

WPF-sovellus luotiin myös .NET Core -sovelluksena. Sen toimintaperiaate oli yksinkertaistettuna kysyä käyttäjältä joko raakaa sykeaineistotiedostoa tai jo kertaalleen esikäsiteltyä ja Anomaly Detectorissa käytettyä aineistoa, joka sisältää poikkeamat, odotetut sykearvot, ylä- ja alamarginaalit ja muut vastauksessa tulevat arvot. Aineistot sitten piirrettiin näytölle viivakaavion muodossa käyttäen LiveCharts nuget-pakettia.

Kuviossa 17 on LiveCharts-kaavio XAML-merkintäkielellä kirjoitettuna. XAMLiin on sidottu (engl. Binding) merkintäkielen ”code-behind”-tiedostosta tarvittavat tiedot, jotka halutaan näyttää LiveCharts-kaaviossa. Liitteessä 2 on näytetty tarkemmin, kuinka tiedot on sidottu C#-ohjelmointikieltä käyttäen.

```
<lvc:CartesianChart x:Name="cartesianChart"
                    Series="{Binding SeriesCollection}"
                    Zoom="X"
                    LegendLocation="Bottom"
                    DisableAnimations="True"
                    Hoverable="True"
                    ScrollMode="X"
                    ScrollHorizontalFrom="{Binding From, Mode=TwoWay}"
                    ScrollHorizontalTo="{Binding To, Mode=TwoWay}"
                    ScrollBarFill="#25303030">
    <lvc:CartesianChart.AxisY>
        <lvc:Axis Title="Syke"
                  MinValue="{Binding MinValue}"
                  MaxValue="{Binding MaxValue}"
                  LabelFormatter="{Binding YFormatter}"
                  FontSize="15"
                  Position="LeftBottom">
        </lvc:Axis>
    </lvc:CartesianChart.AxisY>
    <lvc:CartesianChart.AxisX>
        <lvc:Axis Title="Aikaleima"
                  Labels="{Binding Labels}"
                  FontSize="15"
                  LabelsRotation="3">
            <lvc:Axis.Separator >
                <lvc:Separator Step="60"></lvc:Separator>
            </lvc:Axis.Separator>
        </lvc:Axis>
    </lvc:CartesianChart.AxisX>
</lvc:CartesianChart>
```

Kuvio 17. WPF-sovelluksessa käytetty XAML kaavion piirtämiseen

Series ottaa vastaan kokoelman mitattuja pisteitä ja piirtää ne viivoina LiveCharts-kaavioon. Poikkeamia sisältävien aineistojen kohdalla kaavioon tulee seuraavat tiedot:

- Laskettu yläraja normaalia toimintatapaa noudattaville datapisteille
- Laskettu alaraja normaalia toimintatapaa noudattaville datapisteille
- Mitatut sykkeet
- Odotetut sykkeet Anomaly Detectorin ennusteesta
- Poikkeamat

MaxValue ja *MinValue* on kaavion y-akselin suurin ja pienin mahdollinen arvo lasketuna aineiston mittausten ja rajojen perusteella, jolloin kaavio pysyisi järkevän kokoisena. *Labels* on x-akselin mittausajanhetket aikaleimoina.

6.2 Poikkeamien analysointi ja tulosten tulkinta

6.2.1 Testimenetelmät

Työssä tehdyt testit tehtiin käyttämällä sykedataa, jotka oli mitattu lokakuussa 2020 kahden viikon ajan öisin välillä 00:00 – 07:00 (± 1 tunti). Mitatut sykkeet jaettiin omiksi aineistoiksi ja nimettiin mittauspäivämäärän mukaisesti. Nämä aineistot syötettiin sitten komentokehote-sovelluksessa olevien käsittelyjen läpi ja lopulta ne lähetettiin Anomaly Detectoriin tarkistettavaksi. Jokainen aineisto kävi sarja- ja viimeisen datapisteen tunnistamisen rajapinnoissa, jolloin kahden eri rajapinnan toimintaa voisi vertailla. Jälkimmäisen rajapinnan kohdalla kaikki aineiston datapisteet syötettiin ensimmäisen 12 pisteen jälkeen yksi kerrallaan tunnistettavaksi, koska kyseinen rajapinta ei käsittele kuin yhden datapisteen kerrallaan ja poikkeamantunnistus ei tapahtunut reaaliaikaisesti työn aikana. Aineistoja oli myös yhdistelty toisiinsa, jolloin niistä voisi löytää mahdollista toistuvuutta helpommin, kun sykedataa on enemmän kuin yhdeltä yöltä.

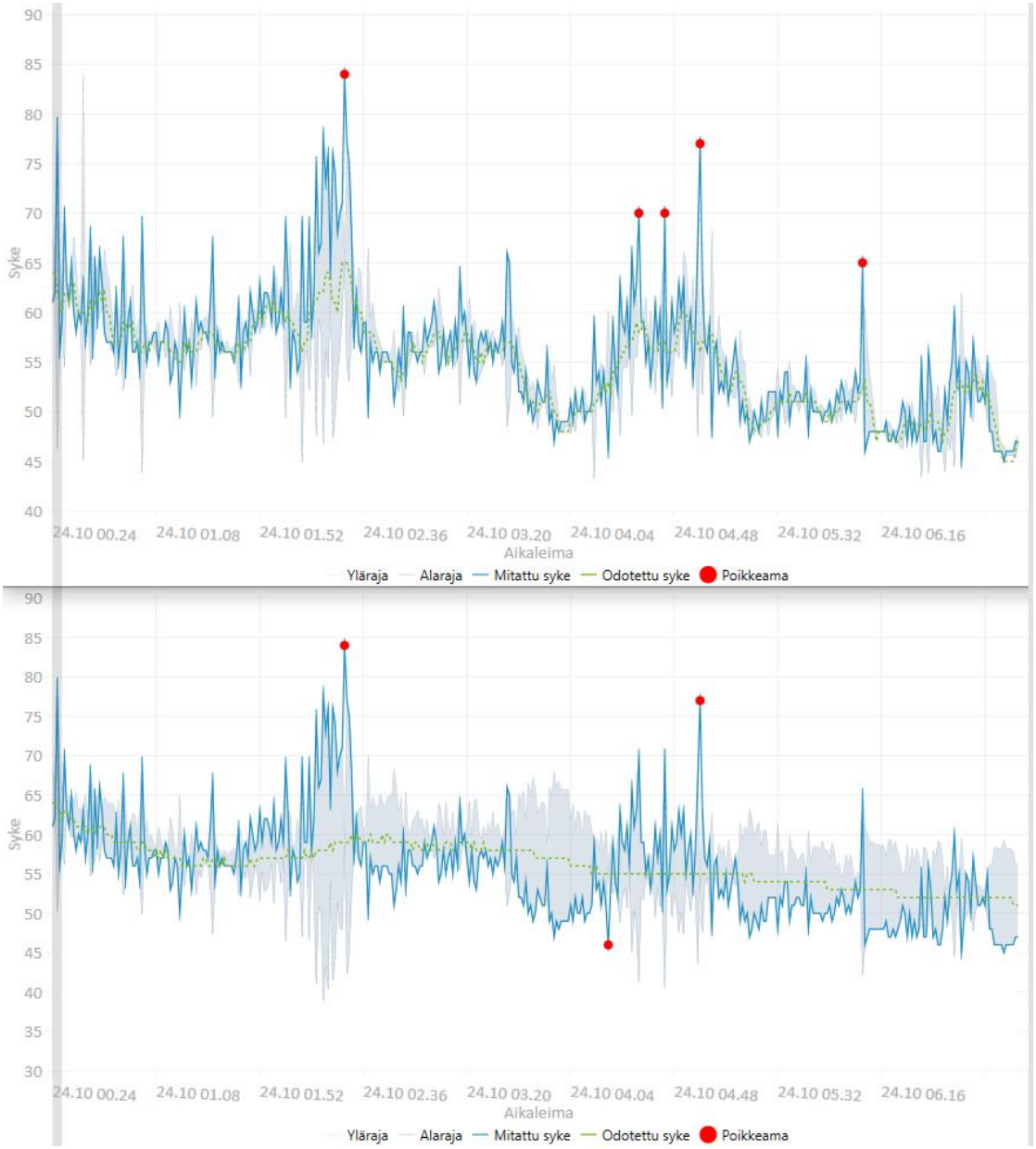
6.2.2 Yksittäiset aineistot

Yksittäiset aineistot koostuivat vain yhden yön mittaustuloksista ja niissä käytetyt poikkeaman tunnistamisen parametrit valiutuivat testauksien perusteella seuraavasti:

- *sensitivity*: 99
- *period*: 0
- *custominterval*: 1
- *granularity*: minutely
- *maxAnomalyRatio*: 0.25

Herkkyysparametri oli valittu korkeimmaksi mahdolliseksi, koska sykedatan satunnaisuuden takia poikkeamien ylä- ja alarajojen määrittäminen oli hankalaa Anomaly Detectorille, koska yksittäisissä aineistoissa on vaikeampi selvittää toistuvuus ja tämän takia jaksoparametri olikin jätetty tyhjäksi. Datapisteiden aikaväli oli minuutti, koska se on pienin mahdollinen arvo, jota Anomaly Detector tukee ja yksittäisten aineistojen kohdalla datapisteitä ei ollut muutenkaan niin paljon, että se olisi ongelma.

Kuviossa 18 näkyy yhden yön mittaustulokset ja yläpuolella on syötetty aineisto sarjatunnistamisen rajapintaan ja alapuolella viimeisen datapisteen tunnistamisen rajapintaan. Koska jaksoparametria ei olla määritetty, niin Anomaly Detector joutuu itse päättelemään mahdolliset toistuvuudet aineistossa, mutta kertaakaan testatessa ei vastauksen mukana tullut Anomaly Detectorin itse havaittua jaksoarvoa. Teoriassa ilman jaksoja poikkeamantunnistus tehdään globaalissa kontekstissa, jolloin kaikki löydetyt poikkeamat ovat globaaleja poikkeamia (ks. luku 2.2.2), mutta ylemmässä kaaviossa 05:32 ja 06:16 välillä on poikkeama, joka ei ole globaali verrattaessa aineiston aikaisempiin datapisteisiin. Tämä viittaa siihen, että Anomaly Detector on kumminkin löytänyt jonkin verran jaksollisuutta, mutta sitä ei vaan aina ilmoiteta vastauksen mukana. Alemmassa kaaviossa poikkeamia on löydetty yhtä suppeasti kuin sarjatunnistamisen tapauksessa, mutta samaan aikaan vihreällä katkoviivalla piirretty viiva on huomattavasti tasaisempi ja loppua kohden lähes suora. Anomaly Detector on onnistunut päättelemään tasaisen trendin sykearvojen kohdalla ja edes satunnaiset piikit mitatuissa sykkeissä ei vaikuta siihen suuresti.



Kuvio 18. 24.10 mitatut sykedatat ilman jaksoparametria kahdesta eri rajapinnasta

Anomaly Detector onnistui havaitsemaan vain suurimmat piikit yksittäisen aineiston kohdalla. Tämä kertoo vain siitä, kuinka kriittinen jaksoparametri on, jos datapisteissä on suuria eroavaisuuksia ja ilman käyttäjän itse määrittämää jaksollisuutta Anomaly Detector joutuu käsittelemään koko aineistoa yhtenä jaksona. Jos Anomaly Detector itsestään havaitsee selkeän toistuvuuden, niin se kyllä palautetaan vastauksen mukana. Kuviossa 19 on samaan tapaan syötetty yhden yön tulokset tarkistettavaksi, eli ylemmässä kaaviossa on käytetty sarjatunnistamista ja alempana viimeisen

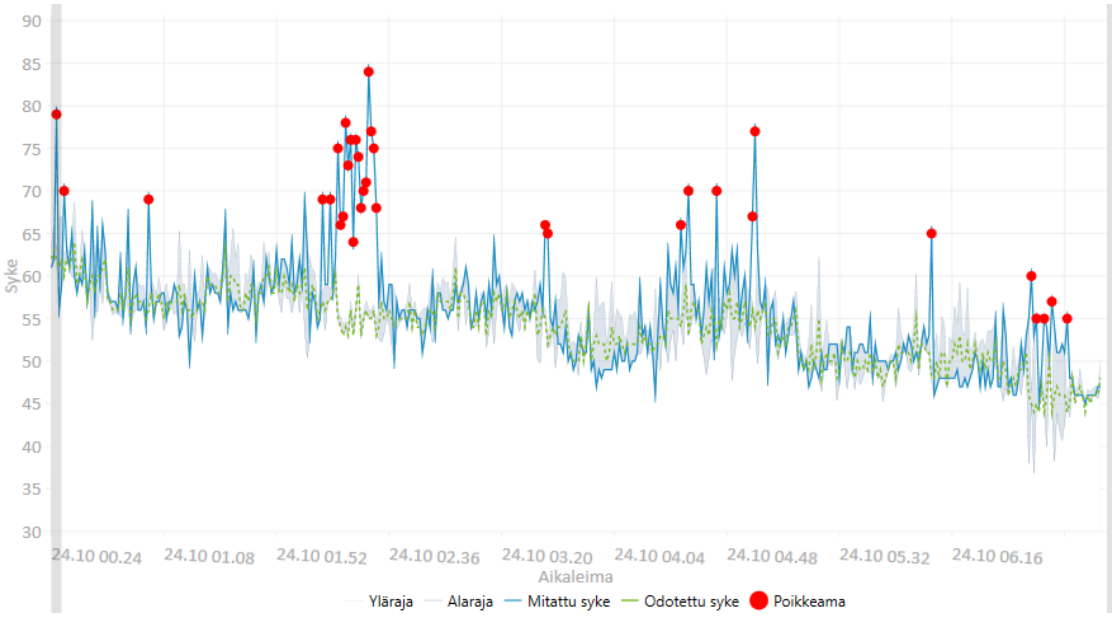
datapisteen tunnistamista. Tähän mennessä ainoastaan viimeisen pisteen tunnistuksessa on löydetty negatiivisia poikkeamia, mutta muuten toiminta kummankin eri rajapinnan välillä on ollut aika lähellä toisiaan poikkeamien osalta. Samaan tapaan myös alemmassa kaaviossa on löydetty tasainen trendi odotetuista sykearvoista.

Kuvio 19. 25.10 mitatut sykedatat ilman jaksoparametria kahdesta eri rajapinnasta

Kaikki yksittäiset aineistot noudattivat samaa kaavaa poikkeaman tunnistamisen jälkeen, eli poikkeamia löydettiin vain ääripäistä ja viimeisen datapisteen tunnistamisen

kohdalla trendi tasaantui loppua kohden. Testausmielessä kuviossa 18 näkyvä aineisto syötettiin toisen kerran sarjatunnistamisen rajapintaan, mutta tällä kertaa jaksoparametri asetettiin mukaan. Jaksoparametriksi asetettiin luku 90, joka on saatu jakamalla odotetun unisyklin pituus (90 minuuttia) aineiston aikavälillä (1 minuutti). Unisyklin kesto on yleensä 90–12 minuuttia aikuisilla, jos uni ei ole katkonaista (Uni n.d.), mutta tämän testin kohdalla jaksoparametrin tarkkuudella ei ole suurta merkitystä.

Kuviossa 20 näkyy, kuinka paljon poikkeamia ilmestyi lisäämällä jaksoparametrin mukaan, vaikka syötetty jaksoparametri ei välttämättä olisikaan tarkka tai edes lähellä totuutta. Parametrin lisäys antoi Anomaly Detectorille viitekehyksen, johon nojata, kun datapisteitä vertailtiin toisiinsa. Suuri osa löydetyistä poikkeamista voitaisiin laskea asiayhteydestä riippuvaisiksi poikkeamiksi, kun toistuvuus on tiedetty. Mutta esimerkiksi n. 02:10 aamuyöllä alkavat mittaukset seuraavan 20 minuutin ajan ovat kaikki yhteispoikkeamia, koska vastaavaa toimintaa ei esiinny muissa aineiston jaksoissa.



Kuvio 20. 24.10 mitatut sykedatat jaksoparametrilla, joka vastaa yhden unisyklin pituutta

6.2.3 Yhdistetyt aineistot

Jokainen yhdistetty aineisto koostui neljän yön mittaustuloksista ja lopuksi yhdistetyt aineistot yhdistettiin vielä kertaalleen yhdeksi suureksi aineistoksi. Anomaly Detectoriin lähtevät parametrit muuttuivat yhdistettyjen aineistojen kohdalla hieman, sillä nyt jaksoparametri *period* oli helppo määrittää, koska mittaukset olivat usealta yöltä. Samalla datapisteiden aikaväli vaihtui minuutista viiteen minuuttiin, koska datapisteiden määrä kasvoi huomattavasti suuremmaksi ja aikavälin kasvattaminen antoi mahdollisuuden laskea keskiarvon aina jokaista viittä datapistettä kohden, mikä laski kohinan määrää aineistossa. Myös aineistojen esikäsittelyyn lisättiin uusi vaihe, jossa jokainen aineisto rajattiin kuuteen tuntiin. Tämän vaiheen jälkeen jokainen aineisto sisälsi identtisen määrän datapisteitä. Muutoksien jälkeen uusi jaksoparametri voitiin laskea kertomalla yksittäisen aineiston kokonaisaika yhden tunnin mittauskerroilla, josta tuli jaksoparametri 72. Herkkyysparametrin tiputtaminen 99 -> 95 ei testitulosten perusteella vaikuttanut lopputulokseen, joten se ja muut ei-mainitut parametrit pysyivät samana kuin yksittäisten aineistojen kohdalla.

Kaikki poikkeaman tunnistaminen on yhdistettyjen aineistojen kohdalla tehty sarjatunnistamisen rajapinnan kautta, eikä viimeisen datapisteen tunnistamista olla käytetty.

Kuviossa 21 on yhdistetty neljät yölliset mittaustulokset ja ylemmässä kaaviossa mukaan on laitettu jaksoparametri, kun taas alemmassa jaksoparametri on poistettu. Kun toistuvuus on tiedossa, niin Anomaly Detector pystyy suorittamaan poikkeaman tunnistamisen vertaamalla datapisteitä muihin jaksoihin, eli tässä tapauksessa muiden öiden mittaustuloksiin. Kuten yksittäisten aineistojen kohdalla, ilman jaksoparametria poikkeamia on havaittu vain muutama alemmassa kaaviossa, koska Anomaly Detector on joutunut päättelemään itse toistuvuuden, jota ei kumminkaan vastaukseen päätynyt.

Kuvio 21.Ensimmäisen neljän yön mittaustulokset

Kuviossa 22 näkyy todella selkeä kausiluonteinen ilmiö viimeisen kolmen yön kohdalla mitattujen ja odotettujen sykearvojen kohdalla ja Anomaly Detector on myös osannut huomioida tämän jaksoparametrin ansiosta. 28 ja 29 päivän välillä näkyvät yhteispoikkeamat ovat kaikki mitattu aamulla ja vastaavaa ei muissa jaksoissa tapahdu, vaan syke on aina tippunut aamuun mennessä ja vasta seuraavana yönä mittausten aloittaessa on syke ollut alkuhetkellä korkea. Syy tälle ilmiölle saattaa löytyä aineiston tiedostonimestä, sillä kyseisenä yönä uni oli levotonta ja herääminen kesti pitkään aamulla, jonka takia tiedostonimeen on merkitty ”levoton yö”.

Kuvio 22. Viimeisen neljän yön mittaustulokset

Lopulta kuviossa 23 kaikki työssä käytetyt aineistot on yhdistetty toisiinsa ja syötetty Anomaly Detectoriin samoilla parametreilla, kuin aikaisemmin. Kaaviossa on kahdeksan päivän edestä mittauksia, jolloin myös jaksoja on kahdeksan ja datapisteitä on yhteensä 576 + 1. Kausiluonteisuus on edellistäkin kuviota selkeämpi tässä ja sen selvittäminen on myös helpompaa, kun dataa on enemmän käytössä. Silmämääräisesti katsottuna Anomaly Detectorin valitsemat poikkeamat ovat järkeen pitäviä, sillä kaikki sykkeiden nousut ja laskut tapahtuvat noudattaen samaa logiikkaa joka yö ja

ne kohdat, joissa tämä toimintatapa ei toteudu on Anomaly Detectorin toimesta havaittu ja merkitty poikkeamaksi. Kausiluonteisuus ei aina tarkoita sitä, että kuvaajan täytyisi piirtää identtistä viivaa jokaisen jakson kohdalle, vaan trendin pitää olla selkeä ja se voi olla stationaarinen, laskeva tai nouseva. Alemmasta kaaviosta puuttuu tässäkin tapauksessa jaksoparametri ja tulokset ovat kanssa mitäänsanomattomat.

Kuvio 23. Kahdeksan yön mittaustulokset yhdistettynä

7 Pohdinta

7.1 Työn tavoitteet

Työn tavoitteena oli selvittää, kuinka Microsoftin Anomaly Detector soveltuu toimeksiantajan asettamiin käyttötarkoituksiin ja samalla perehtyä poikkeaman tunnistamiseen ja itse Anomaly Detectorissa käytettyihin teknologioihin. Ohjelmointiympäristön rakentaminen oli myös kriittinen vaihe työssä, jotta aineistojen kerääminen ja analysointi olisi mahdollisimman ketterää.

Työn tutkittavan kohteen toiminta yritettiin selvittää käyttämällä itse kerättyä sykedataa, jota pyrittiin analysoimaan käyttämällä ensin Microsoftin suosittelemia toimenpiteitä aineiston esikäsittelyssä ja lopulta syöttämällä aineiston joko sarja- tai viimeisen datapisteen tunnistamisen rajapintoihin Anomaly Detectorin päätepisteessä. Poikkeaman tunnistaminen jaettiin toteutuksessa kahteen osaan, eli yksittäisiin ja yhdistettyihin sykeaineistoihin ja näille aineistoille määritettiin testauksen ja Microsoftin suosituksien mukaisesti sopivat parametrit, joita Anomaly Detector käytti hyväksi poikkeamien havaitsemisessa.

7.2 Tulosten tarkastelu

Saadut tulokset täsmäsivät suhteellisen hyvin odotettuja tuloksia Microsoftin dokumenttien lupausten perusteella. Ensimmäiset työssä esitetyt tulokset tehtiin ilman *period*-jaksoparametria ja sen pois jättäminen teki poikkeaman tunnistamisesta huomattavasti epäluotettavampaa, koska ilman tiedettyä toistuvuutta on hankala havaita missä asiayhteydessä mitäkin datapistettä pitäisi verrata muihin aineiston pisteisiin. Anomaly Detectorin yksi ominaisuus on tämän toistuvuuden etsiminen aineistosta, jos sitä ei olla itse määritetty jaksoparametrina, mutta työn aikana sitä ei löydetty kertaakaan automaattisesti. Siksi yksittäisten aineistojen kohdalla on hankala arvoida, että suoriutuiko Anomaly Detector onnistuneesti poikkeamien ja odotettujen datapisteiden havaitsemisessa. Silmämääräiset havainnot ovat myös vaihtelevia tämän kohdalla ja ensimmäisissä esitetyissä aineistoissa on tiettyjä pisteitä, jotka

voisi olettaa poikkeamiksi, vaikka ne eivät kaaviossa välttämättä esiintyisi poikkeamina. Tämä on tosin haaste, joka tulee väkisinkin vastaan, jos poikkeaman tunnistaminen tehdään lyhyen aikavälin sisällä ja siihen nähden Anomaly Detectorin tulokset olivat hyväksyttävät.

Kausiluonteisuuden ja trendien havaitseminen oli yksi esitetty kysymys työn aikana ja se onnistui oikein hyvin Anomaly Detectorilta yhdistettyjen aineistojen kohdalla, kun jaksoparametri oli määritetty. Jokaisen yön kohdalta löytyi selkeä trendi, jota Anomaly Detector pystyi käyttämään viitekehyksenä poikkeamien etsimistä varten. Pieni satunnaisuus sykkeessä ei myöskään rikkonut kausiluonteisuutta ja se todettiinkin tekstissä erään levottoman yön sykeaineiston kohdalla.

Sarja- ja viimeisen datapisteen tunnistamisen rajapintojen eroja ei välttämättä onnistuttu tekemään läpikohtaisesti työn aikana, vaikka kummatkin rajapinnat olivat yksittäisten aineistojen tapauksessa vertailussa. Jälkimmäinen tunnistamistekniikka ei ollut mukana ollenkaan yhdistetyissä aineistoissa, koska kyseinen tekniikka on tarkoitettu pääasiassa reaaliaikaista monitorointia varten tai jos kausiluonteisuus ei ole tiedossa aineistossa.

Unisyklien tutkiminen mittaustuloksista ei mahtunut kunnolla lopulliseen työhön mukaan aikarajoituksien ja muiden haasteiden takia ja sitä käytettiin vain testimielessä yhden aineiston kohdalla, kun jaksoparametri määritettiin keskimääräisen unisyklin pituuden perusteella.

Lopullinen vaikutelma Anomaly Detectorista on enimmäkseen positiivinen saatujen tuloksien perusteella ja sen integroiminen toimeksiantajan käyttötarkoituksien mukaisesti on täysin mahdollista tämänhetkisillä tiedoilla.

7.3 Haasteet

Anomaly Detector on vielä varhaisessa vaiheessa ja se toi omat haasteensa opinnäytetyötä kirjoittaessa ja rajapintoja testatessa työn aikana. Rajapintakutsujen para-

metreissa oli jonkin verran puutoksia dokumentaatiossa ja asiat sai selvittää itse kantapään kautta. Anomaly Detectorissa käytetyissä teknologioissa ja algoritmeissa oli myös paljon aukkoja dokumentaation puolella, mutta tämä saattoi olla myös Microsoftin oma päätös olla julkaisematta kaikkea tietoa Anomaly Detectorin sisäisestä toiminnasta.

Yksi haaste sykedataa kerätessä oli tarkan unirytmin ylläpitäminen ja hyvän unenlaadun varmistaminen terveellisillä elämäntavoilla, mutta se ei aina onnistunut sen hetkisen elämäntilanteen mukaisesti. Sykeaineistoja sai myös esikäsitellä varsinkin yhdistettyjen aineistojen kohdalla huomattavasti enemmän kuin normaalisti, koska Anomaly Detector ei hyväksy aineistoja, joissa on paljon aukkoja aikaleimojen välillä. Työssä käytetyt sykedatat oli mitattu vain öisin, joten jos usean eri aineiston yhdisti toisiinsa, niin tämä tarkoitti suuria aukkoja jokaisen yön välillä ja se vaati jonkin verran soveltamista, että Anomaly Detector hyväksyisi aineistot.

7.4 Jatkokehitys

Jatkokehitykseen olisi hyvä ottaa mukaan enemmän yksittäisen datapisteen tunnistamisen tutkimista, koska sillä on paljon potentiaalia reaaliaikaisten käyttötapauksien puolella. Varsinkin jaksoparametrin toiminnan selvittäminen viimeisen datapisteen rajapinnan kohdalla on tärkeää, jos parametrin tuoma muutos on yhtä suuri kuin sarjatunnistamisen tapauksessa. Jatkokehityksessä olisi myös hyvä testata Anomaly Detectorin toimintaa erityyppisillä aineistoilla ja myös pitkää aikaväliä ajatellen.

Microsoft oli lisännyt kolmannen rajapinnan Anomaly Detectoriin opinnäytetyön aloittamisen jälkeen ja sen toimintaa ei ehditty testaamaan. Sen tutkiminen voisi olla hyödyllistä jatkon kannalta.

Lähteet

Ahmad, S., Lavin, A., Purdy, S., Agha, Z. 2017. Unsupervised real-time anomaly detection for streaming data. Viitattu 16.9.2020. https://www.sciencedirect.com/science/article/pii/S0925231217309864

Anomaly Detector is now available. 2019. Päivitys Microsoftin www-sivuilla 2.4.2019. Viitattu 30.9.2020. https://azure.microsoft.com/en-in/updates/anomaly-detector-is-now-available/

Best practices for using the Anomaly Detector API. 2019. Dokumentaatio Microsoftin www-sivuilla. 3.26.2019. Viitattu 9.10.2020 https://docs.microsoft.com/en-us/azure/cognitive-services/anomaly-detector/concepts/anomaly-detection-best-practices

Chandola, V., Banerjee, A., Kumar, V. 2009. Anomaly Detection: A Survey. Research paper, University of Minnesota. Viitattu 26.9.2020. http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf

Cohen, I. 2019. A Quick Guide to the Different Types of Outliers. Kirjoitus Anodotin blogissa 31.7.2019. Viitattu 5.9.2020. https://www.anodot.com/blog/quick-guide-different-types-outliers/

Detect Change Point. 2020. Dokumentaatio Microsoftin www-sivuilla. 31.8.2020. Viitattu 10.10.2020. https://docs.microsoft.com/en-us/rest/api/cognitiveservices/anomalydetector/detectchangepoint/detectchangepoint

Detect Entire Series. 2020. Dokumentaatio Microsoftin www-sivuilla. 31.8.2020. Viitattu 9.10.2020. https://docs.microsoft.com/en-us/rest/api/cognitiveservices/anomalydetector/detectentireseries/detectentireseries

Detect Last Point. 2020. Dokumentaatio Microsoftin www-sivuilla. 31.8.2020. Viitattu 10.10.2020. https://docs.microsoft.com/en-us/rest/api/cognitiveservices/anomalydetector/detectlastpoint/detectlastpoint

Develop .NET applications. N.d. .NET yleiskatsaus Microsoftin www-sivuilla. Viitattu 31.10.2020. https://visualstudio.microsoft.com/vs/features/net-development/

Find anomalies for the entire series in batch. 2019. Dokumentaatio Microsoftin Cognitive Services www-sivuilla. Viitattu 8.10.2020. https://westus2.dev.cognitive.microsoft.com/docs/services/AnomalyDetector/operations/post-timeseries-entire-detect

Harris, N. 2014. Visualizing K-Means Clustering. K-means työkalu blogissa 19.1.2014. Viitattu 26.9.2020. https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

How to: Use the Anomaly Detector API on your time series data. 2019. Dokumentaatio Microsoftin www-sivuilla. 1.10.2019. Viitattu 18.10.2020. https://docs.microsoft.com/en-us/azure/cognitive-services/anomaly-detector/how-to/identify-anomalies

Huang, C. 2018. Featured Anomaly Detection Methods and Applications. Thesis, university. PhD in Computer Science, University of Exeter. Viitattu 16.9.2020. https://ore.exeter.ac.uk/repository/handle/10871/34351

Hyndman, R.J., & Athanasopoulos, G. 2018. Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2

Ilmatieteenlaitos. Havaintojen lataus paikassa Jyväskylän lentoasema vuosille 2010 ja 2015–2017. Viitattu 5.9.2020. https://www.ilmatieteenlaitos.fi/havaintojen-lataus

Introduction to Tizen. N.d. Dokumentaatio Tizen Docs www-sivuilla. Viitattu 31.10.2020. https://docs.tizen.org/platform/what-is-tizen/overview/

Iotas. 2020. Iotas Oy. Viitattu 1.6.2020. https://www.iotas.fi/Home/Company

Kalmuk, A., Granichin, O., Granichina, O., Ding, M. 2016. A Dynamic Threshold Based Algorithm for Change Detection in Autonomous Systems. Tutkielma. Viitattu 28.10.2020. https://www.sciencedirect.com/science/article/pii/S2405896316312290

Kotu, V. & Deshpande, B. 2018. Data Science (Second Edition). Burlington: Morgan Kaufmann

Li, Y., Liang, D. 2019. Safe semi-supervised learning: a brief introduction. Nanjing: Nanjing University. Viitattu 9.9.2020. http://www.lamda.nju.edu.cn/liyf/paper/FCS19-SafeSSL.pdf

McLeod, S. A. 2019. Z-score: definition, calculation and interpretation. Kirjoitus Simply Psychology www-sivuilla. 17.5.2019. Viitattu 30.10.2020. https://www.simplypsychology.org/z-score.html

Mehrotra, Kishan G., Mohan, Chilukuri K., Huang, HuaMing. 2017. Anomaly Detection Principles and Algorithms (Terrorism, Security, and Computation). Springer International Publishing. Kindle Edition.

Mila, T. 2019. What is the Fourier Transform?. Artikkeli Siemensin www-sivuilla. 29.8.2019. Viitattu 29.10.2020. https://community.sw.siemens.com/s/article/what-is-the-fourier-transform

Quickstart: Detect anomalies in your time series data. 2020. Dokumentaatio Microsoftin www-sivuilla. 9.3.2020. Viitattu 8.10.2020. https://docs.microsoft.com/en-us/azure/cognitive-services/anomaly-detector/quickstarts/detect-data-anomalies-csharp

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., Zhang, Q. 2019. Time-Series Anomaly Detection Service at Microsoft. Tutkimusartikkeli. Viitattu 30.10.2020. https://arxiv.org/abs/1906.03821

Tizen Studio. N.D. Dokumentaatio Tizen Docs www-sivuilla. Viitattu 31.10.2020. https://docs.tizen.org/application/tizen-studio/

Toews, M. W. 2005. Standard deviation diagram. Viitattu 29.10.2020. https://commons.wikimedia.org/w/index.php?curid=1903871

Uni. N.d. Artikkeli terveyskirkon www-sivuilla. Viitattu 19.11.2020. https://www.terveysverkko.fi/tietopankki/terveysliikunta/uni/

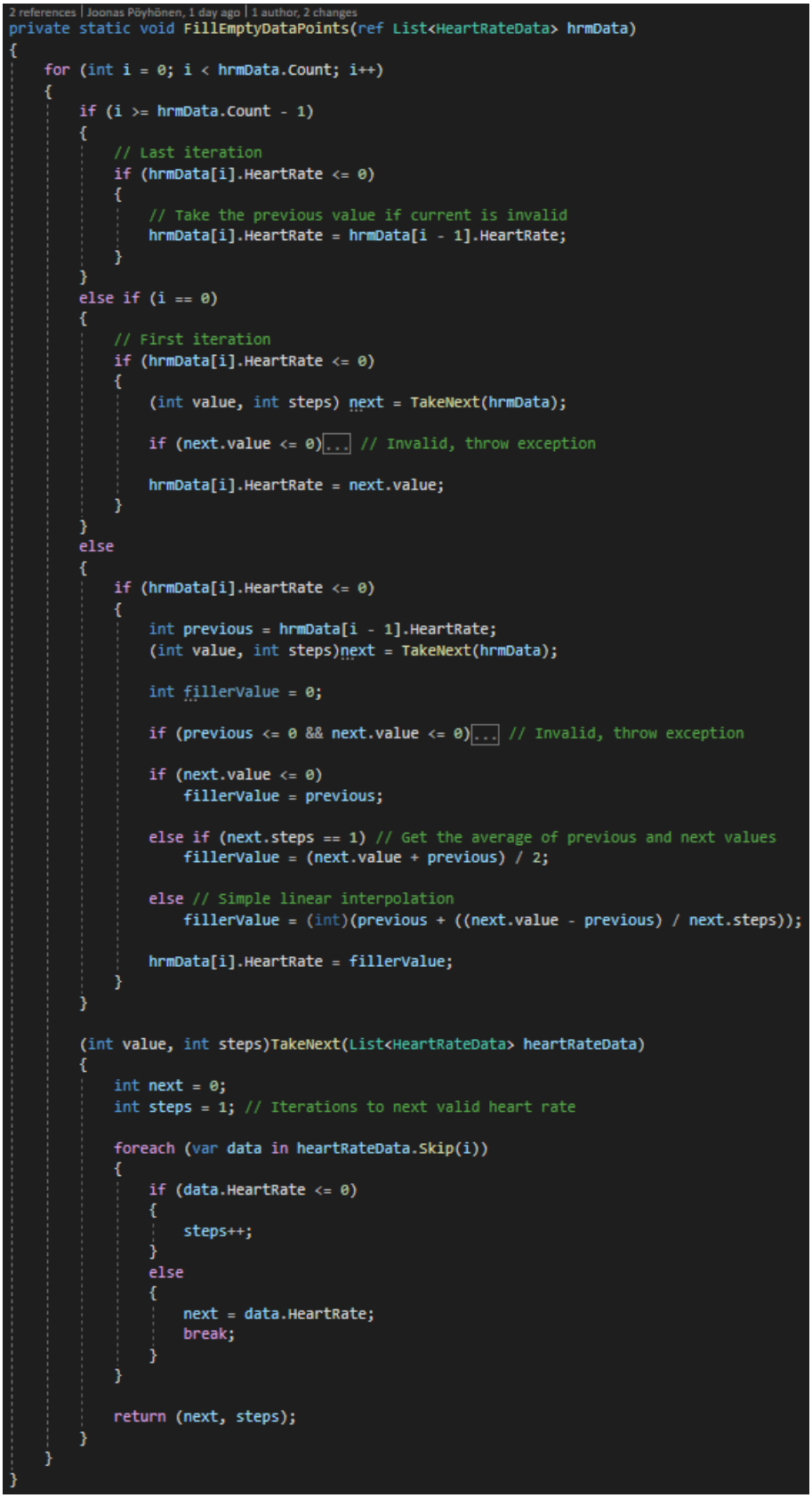
Windows Presentation Foundation. N.d. WPF yleiskatsaus Microsoftin www-sivuilla. Viitattu 31.10.2020. https://visualstudio.microsoft.com/vs/features/wpf/

Xing, T. 2019. Introducing Azure Anomaly Detector API. Kirjoitus Microsoftin blogissa. 30.4.2019. Viitattu 30.9.2020. https://techcommunity.microsoft.com/t5/ai-customer-engineering-team/introducing-azure-anomaly-detector-api/ba-p/490162

Zaiontz, C. N.d. Generalized Extreme Studentized Deviate Test. Kirjoitus Real-statistics -blogissa. Viitattu 27.10.2020. https://www.real-statistics.com/students-t-distribution/identifying-outliers-using-t-distribution/generalized-extreme-studentized-deviate-test/

Liitteet

Liite 1. Puuttuvien datapisteiden lisääminen



```
2 references | Joonas Pöyhönen, 1 day ago | 1 author, 2 changes
private static void FillEmptyDataPoints(ref List<HeartRateData> hrmData)
{
    for (int i = 0; i < hrmData.Count; i++)
    {
        if (i >= hrmData.Count - 1)
        {
            // Last iteration
            if (hrmData[i].HeartRate <= 0)
            {
                // Take the previous value if current is invalid
                hrmData[i].HeartRate = hrmData[i - 1].HeartRate;
            }
        }
        else if (i == 0)
        {
            // First iteration
            if (hrmData[i].HeartRate <= 0)
            {
                (int value, int steps) next = TakeNext(hrmData);

                if (next.value <= 0)... // Invalid, throw exception

                hrmData[i].HeartRate = next.value;
            }
        }
        else
        {
            if (hrmData[i].HeartRate <= 0)
            {
                int previous = hrmData[i - 1].HeartRate;
                (int value, int steps)next = TakeNext(hrmData);

                int fillerValue = 0;

                if (previous <= 0 && next.value <= 0)... // Invalid, throw exception

                if (next.value <= 0)
                    fillerValue = previous;

                else if (next.steps == 1) // Get the average of previous and next values
                    fillerValue = (next.value + previous) / 2;

                else // Simple linear interpolation
                    fillerValue = (int)(previous + ((next.value - previous) / next.steps));

                hrmData[i].HeartRate = fillerValue;
            }
        }

        (int value, int steps)TakeNext(List<HeartRateData> heartRateData)
        {
            int next = 0;
            int steps = 1; // Iterations to next valid heart rate

            foreach (var data in heartRateData.Skip(i))
            {
                if (data.HeartRate <= 0)
                {
                    steps++;
                }
                else
                {
                    next = data.HeartRate;
                    break;
                }
            }

            return (next, steps);
        }
    }
}
```

Liite 2. XAML "code-behind" -alustaminen kaaviolle

```
SeriesCollection = new SeriesCollection
{
    new LineSeries
    {
        Title = "Yläraja",
        Values = new ChartValues<double>(_graphData.UpperBoundaryValues),
        PointGeometry = null,
        StrokeThickness = 0.2,
        LineSmoothness = 0.2,
        Fill = new SolidColorBrush(Color.FromArgb(50, 136, 158, 185)), // Light gray/blue
        Stroke = new SolidColorBrush(Color.FromRgb(136, 158, 185)), // Light gray/blue
    },
    new LineSeries
    {
        Title = "Alaraja",
        Values = new ChartValues<double>(_graphData.LowerBoundaryValues),
        PointGeometry = null,
        StrokeThickness = 0.3,
        LineSmoothness = 0.2,
        Fill = Brushes.White,
        Stroke = new SolidColorBrush(Color.FromRgb(136, 158, 185)), // Light gray/blue
    },
    new LineSeries
    {
        Title = "Mitattu syke",
        Values = new ChartValues<int>(_graphData.MeasuredValues),
        PointGeometry = null,
        StrokeThickness = 1.2,
        LineSmoothness = 0,
        Fill = Brushes.Transparent,
        Stroke = new SolidColorBrush(Color.FromRgb(28, 142, 196)), // Blue
    },
    new LineSeries
    {
        Title = "Odotettu syke",
        Values = new ChartValues<int>(_graphData.ExpectedValues),
        StrokeDashArray = new DoubleCollection { 2 },
        PointGeometry = null,
        StrokeThickness = 1.2,
        LineSmoothness = 0,
        Fill = Brushes.Transparent,
        Stroke = new SolidColorBrush(Color.FromRgb(121, 186, 33)), // Green
    },
    new LineSeries
    {
        Title = "Poikkeama",
        Values = new ChartValues<double>(anomalyPoints),
        PointGeometry = Geometry.Parse("M 0,0 A 180,180 180 1 1 1,1 Z"), // Circle
        PointGeometrySize = 8,
        PointForeground = Brushes.Red,
        StrokeThickness = 0,
        LineSmoothness = 1,
        Fill = Brushes.Transparent,
    }
};

Labels.AddRange(_graphData.Timestamps.Select(d => d.ToString("dd.MM HH:mm")));
YFormatter = value => value.ToString();
MinValue = _graphData.MinValue;
MaxValue = _graphData.MaxValue;
```