Aikasarjojen luokittelu koneoppimismenetelmiä käyttäen

Pro Gradu -tutkielma
Tuomas Hautamäki
Matemaattisten tieteiden tutkimusyksikkö
Oulun yliopisto
Kevät 2023

Sisällys

Tiivistelmä

Tässä tutkielmassa perehdytään aikasarjojen luokitteluun koneoppimismenetelmiä käyttäen. Menetelmiksi valikoitui K-lähinaapurimenetelmä, jonka etäisyysmittoina käytetään euklidista etäisyyttä, dynaamista aikavääristymää ja dynaamisen aikavääristymän derivaatallista sekä painotettua versiota, sekä aikasarjametsä, joka pohjautuu satunnaismetsään. Lisäksi tutkitaan, onko aineiston muuntamisella shapelet-muunnoksella vaikutusta luokittelun tarkkuuteen, kun luokittelijana käytetään satunnaismetsää.

K-lähinaapurimenetelmää, jonka etäisyysmittana käytetään dynaamista aikavääristymää, pidetään aikasarjojen luokittelussa oletusmenetelmänä, johon muita menetelmiä verrataan. Satunnaismetsä on puolestaan suosittu ja useimmiten hyvin toimiva menetelmä muiden kuin aikasarja-aineistojen luokittelussa, joten nämä kaksi menetelmää valittiin tutkielmaan näillä perusteilla. Aineiston esikäsittelyllä sekä erilaisilla muunnoksilla on vaikutusta luokittelijoiden toimintaan, joten shapelet-muunnos valittiin tutkielmaan edustamaan tätä osaa aikasarjojen luokittelusta.

Käytännön luokittelutestien aineistona tutkielmassa käytettiin Iso-Britanniassa kerättyä sähkölaitteiden sähkönkulutuksesta koottua aikasarja-aineistoa, joka sisältää seitsemän erilaisen kodinkoneen sähkönkulutuksesta muodostettuja tasamittaisia, 96 havaintoa sisältäviä aikasarjoja. Aineiston valintakriteereinä olivat riittävän suuri koko, jotta koneoppimismenetelmien malleilla on riittävästi opetusmateriaalia, moniluokkaisuus sekä aikasarjojen kohtuullinen pituus.

Parhaiten toiminut menetelmä oli shapelet-muunnos, jonka kanssa käytettiin satunnaismetsäluokittelijaa. Menetelmän normaali tarkkuus oli noin 73 prosenttia ja tasapainotettu tarkkuus noin 63 prosenttia. Aikasarjametsän molemmat tarkkuudet olivat noin 4 prosenttiyksikköä matalammat kuin shapelet-muunnoksen avulla saavutetut, mutta aikasarjametsä oli ajallisesti tehokkain menetelmä. Mallin sovitus opetusaineistoon ja testiaineiston luokittelu kesti yhteensä vain noin 30 sekuntia. Shapelet-muunnoksen sovittaminen kesti nopeimmillaankin noin 50 minuuttia, ja parhaiten toimineen muunnoksen sovittaminen vaati melkein vuorokauden. K-lähinaapurimenetelmän tarkkuudet jäivät heikoimmaksi, ja luokitteluun kuluneet ajat olivat euklidisen etäisyyden vajaasta minuutista painotetun dynaamisen aikavääristymän 76 minuuttiin.

1 Johdanto

Aikasarja on tietystä havaintoyksiköstä tietyllä tarkkuudella mitatuista havainnoista muodostuva järjestyksellinen lukujono. Esimerkiksi tunnin välein tehdyt lämpötilamittaukset muodostavat aikasarjan, jossa järjestyksen määrää aika ja tarkkuus on yksi tunti. Tällaista yhtä ilmiötä kuvaavaa aikasarjaa kutsutaan yhden muuttujan aikasarjaksi. Kun puolestaan halutaan kuvata jotain laajempaa ilmiötä, esimerkiksi säätilaa, tarvitaan havaintoja muistakin säähän liittyvistä ilmiöistä. Lämpötilan lisäksi pitää mitata myös esimerkiksi ilmanpainetta sekä -kosteutta, jolloin näistä saadaan monimuuttuja-aikasarja.

Aikasarjoja syntyy siis kaikesta, mitä mitataan säännöllisesti ja useammin kuin kerran. Nopeasti yleistynyt trendi on ollut erilaiset ihmisen aktiivisuutta ja elintoimintoja mittaavat puettavat laitteet, kuten kellot ja sormukset (Niinistö, 2020). Kyseiset laitteet mittaavat parhaimmillaan ihmisen liikkeitä ja elintoimintoja kaksikymmentäneljä tuntia vuorokaudessa vuoden jokaisena vuorokautena, jolloin erilaisia aikasarjoja syntyy valtavasti. Lisäksi laitteet keräävät yleensä tarkempaa tietoa, kun ne asetetaan urheilutilaan, jolloin ne voivat kerätä esimerkiksi sykettä, GPS:ään perustuvaa nopeus- sekä paikkatietoa ja ilmanpainemittariin perustuvaa korkeustietoa sekunnin tarkkuudella.

Pörssi- ja osakekurssit sekä korkojen muutokset muodostavat myös aikasarjoja, joita voi olla kiinnostavaa analysoida ja luokitella (Liang ym., 2022). Niistä voidaan havaita esimerkiksi markkinoihin vaikuttavia tapahtumia ja tehdä niiden perusteella vastatoimia. Pankit voivat myös käyttää aikasarjojen luokittelua höydyksi tunnistamaan epäilyttävää toimintaa pankkiliikenteessä.

Sähkön kulutuksen seuraaminen on näin energiakriisin (Popkostova, 2022) aikana varsin tärkeää, ja kotitalouksien sähkönkulutuksesta muodostuukin aikasarjoja erilaisilla mittausväleillä aina yhdestä tunnista yhteen vuoteen, joita kuluttuja voi itse seurata sähköyhtiönsä palveluiden kautta. Sähkön kulutusta on mahdollista mitata ja seurata myös reaaliaikaisesti erillisellä laitteella jopa sekunnin tarkkuudella. Tällaisen tarkasti mitatun datan luokittelu eri sähkölaitetyyppeihin voisi olla kuluttajalle hyödyllistä, koska käyttäjä näkisi mistä sähkön kulutus muodostuu, ja tätä kautta sähkön kulutusta olisi helpompi vähentää.

Tässä tutkielmassa perehdytään kuinka erilaiset koneoppimismenetelmät so-

veltuvat aikasarjojen luokitteluun. Luokitteluaineistona käytetään erilaisten kodin sähkölaitteiden sähkönkulutuksesta muodostettua aikasarja-aineistoa. Menetelminä käytetään K-lähinaapurimenetelmää, aikasarjoille sovellettua satunnaismetsää eli aikasarjametsää sekä shapelet-muunnosta, jonka kanssa käytetään satunnaismetsäluokittelijaa (Deng ym., 2013). K-lähinaapurimenetelmän kanssa käytetään neljää eri etäisyysmittaa: euklidista etäisyyttä, dynaamista aikavääristymää ja dynaamisen aikavääristymän derivaatallista sekä painotettua versiota.

Tutkielmassa käytetyt koneoppimismenetelmät ovat ohjatun oppimisen (*supervised learning*) menetelmiä, koska kyse on aikasarjojen luokittelusta, jolloin aineistoissa on jokaisen havainnon luokka tiedossa. Jos aineistossa ei olisi todellisia luokkia tiedossa, voitaisiin aikasarjoja klusteroida eli tutkia muodostavatko aineiston havainnot selkeitä, toisistaan erottuvia rypäitä. Näitä menetelmiä kutsutaan ohjaamattoman oppimisen (*unsupervised learning*) menetelmiksi.

Tutkielman rakenne on jaettu siten, että luvussa 2 kuvataan mitä aikasarjaluokittelu on, kuvataan tutkielmassa käytetyt tilastolliset koneoppimismenetelmät ja niiden teoriaa, käydään läpi luokittelijoiden hyperparametrien hienosäätö ja käydään läpi luokittelijoiden toimivuuden tarkasteluun sekä hienosäätöön tarvittavat mittarit. Luvussa 3 esitellään tutkielmassa käytetty aineisto ja luvussa 4 puolestaan aineiston luokittelusta saadut tulokset eri menetelmillä. Lopuksi luvussa 5 pohditaan menetelmien toimivuutta sekä mahdollisia jatkotutkimuksia aiheen ympärillä.

2 Menetelmät

Aikasarjojen luokitteluun löytyy pelkästään Pythonin *sktime*-paketista 40 erilaista luokittelijaa (Löning ym., 2019). Näiden lisäksi aikasarjojen luokitteluun voidaan käyttää myös neuroverkkojen luokittelijoita, joista 4 löytyy tällä hetkellä myös *sktime*-paketista. Tässä tutkielmassa tarkasteltavien ja käytettävien luokittelijoiden valinta on tehty pääasiassa niiden suosion sekä aseman perusteella.

K-lähinaapurimenetelmää, jonka etäisyysmittana käytetään dynaamista aikavääristymää, pidetään aikasarjaluokittelun oletusmenetelmänä, johon muita menetelmiä verrataan (Bagnall & Lines, 2014). Satunnaismetsät puolestaan ovat suosittuja menetelmiä luokitteluongelmissa, koska niiden suorituskyky on hyvä, ja niiden mallit ovat helpompia sovittaa aineistoon sekä mallien hyperparametrit ovat helpompia hienosäätä, kuin verrattuna esimerkiksi tehostettujen (*boosting*) menetelmien malleihin (Hastie ym., 2009).

Kolmas tutkielmaan valikoitu menetelmä, shapelet-muunnos, ei ole luokittelija, vaan kyseessä on menetelmä, joka muuntaa aineiston muotoon, josta luokittelijan on helpompi tunnistaa luokat toisistaan. Varsinaisena luokittelijana shapelet-muunnoksen kanssa tässä tutkielmassa käytetään satunnaismetsää, ja tarkoitus on tutkia parantaako muunnos luokittelijan suorituskykyä ilman muunnosta käytettyyn aikasarjametsään verrattuna.

2.1 Aikasarjojen luokittelu

Aikasarja (*time series*) x_i, jossa i viittaa havaintoon aineistossa $\mathbf{D}$, määritellään joukkona järjestettyjä havaintoja

$$\mathrm{x}_i = (x_{i_1}, \ldots, x_{i_m}),$$

jossa järjestys on yleensä aikajärjestys ja m on havaintojen määrä aikasarjassa (Bagnall & Lines, 2014). Havaintojen määrä voi olla eri aikasarjojen välillä, mutta tässä tutkielmassa aikasarjat oletetaan yhtä pitkiksi. Aikasarja voidaan muodostaa myös muusta kuin ajallisesti järjestetystä datasta: esimerkiksi kaksiulotteisia kuvia voidaan käsitellä aikasarjoina, joissa pikselien sijainnit määrittävät järjestyksen. Lisäksi tutkielmassa oletetaan, että havaintojen välinen aika on kiinteä, esimerkiksi 2 minuuttia. Luokittelua varten aikasarjaan liitetään lisäksi luokan nimiö (*label*) y_i.

Aikasarjoista x_i ja aikasarjan luokan nimiöistä y_i muodostetaan aineisto

$$\mathbf{D} = \{(\mathrm{x}_1, y_1), \ldots, (\mathrm{x}_n, y_n)\},$$

jossa n on aikasarjojen määrä aineistossa (Bagnall & Lines, 2014). On tärkeää huomata, että aikasarjojen yhteydessä on kahdenlaisia havaintoja: aikasarjojen sisällä olevat yksittäiset havainnot eli mittaukset sekä aineiston havainnot, joita edustavat kokonaiset aikasarjat. Aineisto jaetaan opetus- ja testiaineistoihin, joista opetusaineistoa käytetään aikasarjaluokittelijan opettamiseen, ja testiaineistoa puolestaan käytetään luokittelijan toimivuuden testaamiseen. Yleensä opetusaineiston koko on noin 50–80 prosenttia koko aineistosta, jolloin testiaineistoon jää noin 20–50 prosenttia aikasarjoista. Opetusaineistosta voidaan vielä erottaa esimerkiksi 25 prosenttia erilliseksi validointiaineistoksi, jota käytetään eri luokittelijoiden hyperparametrien hienosäätöön, jos aineiston koko tämän sallii. Tällöin testiaineistoa käytetään vain lopulliseksi valitun luokittelijan toimivuuden testaamiseen ja tarvittavien mittarien laskemiseen. Hyperparametrit ovat mallin tai algoritmin parametreja, joita ei voida päivittää oppimisprosessissa, vaan ne tulee määrittää ennen mallin opetusta, koska ne määrittävät koneoppimismallin arkkitehtuurin (Yang ym., 2020). Luokittelijoiden hyperparametrien hienosäätö on kuvattu tarkemmin luvussa 2.5. Aikasarjojen luokittelu ei siis poikkea muista luokitteluongelmista kuin sen suhteen, että miten yksittäisten aikasarjojen samanlaisuutta mitataan.

Hills ym. (2014) jakavat aikasarjoja erottelevan samanlaisuuden kolmeen eri kategoriaan sen mukaan, mihin aikasarjan ominaisuuteen samanlaisuus perustuu:

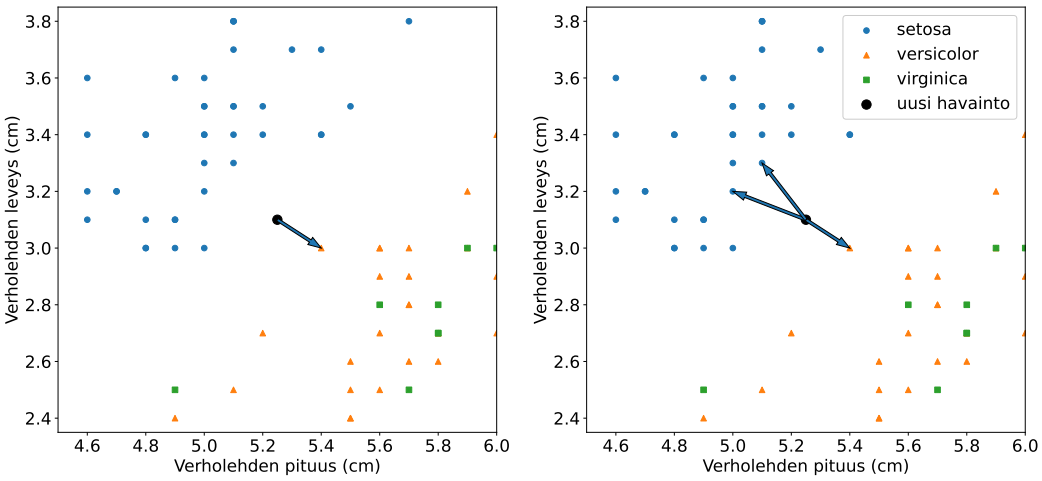
- samanlaisuus ajan suhteen (*similarity in time*),
- samanlaisuus muutoksen suhteen (*similarity in change*) ja
- samanlaisuus muodon suhteen (*similarity in shape*).

Toisaalta luokittelu voidaan jakaa myös joko havaintoperusteiseksi (*instance-based*) tai piirreperusteiseksi (*feature-based*) sen mukaan, perustuuko luokittelu koko aikasarjasta muodostettavaan metriikkaan, esimerkiksi etäisyysmittaan, vai aikasarjasta muodostettaviin piirteisiin (*feature*), esimerkiksi keskiarvoon, jotka yleensä lasketaan tietyn mittaisille intervalleille (Deng ym., 2013). Tässä tutkielmassa perehdytään luokittelijoihin, jotka käyttävät aikasarjojen samanlaisuutta ajan suhteen ja samanlaisuutta muodon suhteen hyväkseen sekä havainto- että piirreperusteisiin luokittelijoihin.

2.2 K-lähinaapurimenetelmä

K-lähinaapurinmenetelmän (*k-Nearest-Neighbor, kNN* tai *k-NN*) luokittelijat ovat muistiin perustuvia (*memory-based*), eivätkä ne vaadi mallin sovittamista (Hastie ym., 2009). Bagnall & Lines (2014) käyttävät K-lähinaapurimenetelmästä myös nimitystä laiska luokittelija (*lazy classifier*), joka viittaa siihen että varsinainen työ tehdään luokitteluvaiheessa. Kun uusi havainto $\mathbf{x}_0$ halutaan luokitella, etsitään sille k lähintä pistettä opetusaineistosta ja se luokitellaan siihen luokkaan, joka on eniten edustettuna k:n pisteen joukossa (Hastie ym., 2009). Tasatilanteissa luokka määrätään satunnaisesti (Hastie ym., 2009).

Kuvassa 1 havainnollistetaan kurjenmiekka-aineistolla, kuinka uuden havainnon luokittelu riippuu naapurien määrästä k. Vasemmanpuoleisessa kuvaajassa naapurien määrä on 1 eli uuden havainnon luokka määräytyy aina lähimmän naapurin mukaan. Tässä tapauksessa uusi havainto luokiteltaisiin versicolor-lajiin. Oikeanpuoleisessa kuvaajassa on puolestaan käytetty naapurien määrää 3, jolloin uusi havainto luokiteltaisiinkin setosa-lajin edustajaksi.



Kuva 1: K-lähinaapurimenetelmä havainnollistettuna kurjenmiekka-aineistolla kahdella eri k:n arvolla, kun luokitteluun käytetään euklidista etäisyyttä ja muuttujina verholehden pituutta ja leveyttä. Vasemmalla $k = 1$ ja oikealla $k = 3$. Nuolet osoittavat uutta havaintoa lähinnä olevat naapurit. Euklidiset etäisyydet lähimpiin naapureihin ovat 0,033 cm, 0,063 cm ja 0,073 cm.

K-lähinaapurimenetelmissä naapurien määrän lisäksi merkittävä tekijä luokittelijan toimivuudessa on käytetty etäisyysmitta. Euklidinen etäisyys on toimiva mitta, kun eri muuttujat ovat samalla asteikolla ja havainnoista

on mitattu yksittäisiä piirteitä, kuten edellisessä esimerkissä kurjenmiekka-aineistossa olevat terä- ja verholehtien pituudet sekä leveydet. Euklidista etäisyyttä voidaan käyttää myös aikasarjojen samanlaisuuden mittaamiseen, mutta käytettävissä on useita muitakin mittoja, joista monet toimivat paremmin kuin euklidinen etäisyys. Luvussa 2.2.1 esitellään euklidinen etäisyys ja luvussa 2.2.2 dynaaminen aikavääristymä, joka on aikasarjojen luokittelussa hyväksi havaittu mitta. Lisäksi luvussa 2.2.3 esitellään dynaamisen aikavääristymän derivaatallinen versio ja luvussa 2.2.4 sen painotettu versio.

2.2.1 Euklidinen etäisyys

Euklidinen etäisyys kahden n-ulotteisen pisteen $\mathbf{x} = (x_1, \ldots, x_n)$ ja $\mathbf{y} = (y_1, \ldots, y_n)$ välillä määritellään

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{1}$$

(Kubat, 2017).

Yhtälöä 1 voidaan käyttää myös kahden aikasarjan etäisyyden mittaamiseen siten, että koordinaattipisteiden asemesta laskenta tapahtuu aikasarjan järjestyksen (esimerkiksi ajan) mukaisesti aikasarjojen havaintojen yli. Toisin sanoen kahden aikasarjan x_1 ja x_2 välinen euklidinen etäisyys on

$$d_E(\mathrm{x}_1, \mathrm{x}_2) = \sqrt{\sum_{i=1}^{m} (x_{1_i} - x_{2_i})^2}, \tag{2}$$

jossa m on aikasarjan havaintojen määrä. Muun muassa Lines & Bagnall (2015) jättävät yhtälöstä 2 neliöjuuren pois eli kyseessä on euklidisen etäisyyden neliö. Jos $d_E(\mathrm{x}_1, \mathrm{x}_2) = 0$, aikasarjat x_1 ja x_2 ovat täsmälleen samat eli $\mathrm{x}_1 = \mathrm{x}_2$, ja muulloin aikasarjat poikkeavat toisistaan. Euklidinen etäisyys toimii hyvänä etäisyysmittarina aikasarjojen luokittelussa silloin, kun samanlaisuus on ajan suhteen (*similarity in time*) eli havaittava muoto aikasarjassa on ajallisesti samassa kohdassa.

2.2.2 Dynaaminen aikavääristymä

Aina havaittavat tapahtumat tai muutokset aikasarjoissa eivät tapahdu ajallisesti täsmälleen samalla hetkellä. Tällöin tärkeämpää on tunnistaa tapahtuman muoto, eikä tapahtuman ajankohdalla ole niin suurta merkitystä. Dynaaminen aikavääristymä (*Dynamic Time Warping*, jatkossa DTW) on ke-

hitetty tähän ongelmaan, eli tunnistamaan samanlaisuutta muodon suhteen (*similarity in shape*) (Lines & Bagnall, 2015).

Oletetaan, että halutaan mitata etäisyyttä kahden aikasarjan x_1 ja x_2 välillä. Olkoon $M(\mathrm{x}_1,\ \mathrm{x}_2)$ kooltaan $m \times m$ etäisyysmatriisi aikasarjojen x_1 ja x_2 välillä, jossa $M_{i,j} = (x_{1_i} - x_{2_j})^2$. Vääristymäpolku (*warping path*)

$$P = \langle (e_1,\ f_1), \ldots, (e_s,\ f_s) \rangle$$

on joukko pisteitä eli indeksipareja, jotka määrittävät polun matriisin M läpi. Näin ollen esimerkiksi polku matriisin diagonaalin yli määrittää euklidisen etäisyyden $d_E(\mathrm{x}_1,\ \mathrm{x}_2) = \sum_{i=1}^{m}(x_{1_i} - x_{2_i})^2$. Kelvollisen vääristymäpolun tulee täyttää ehdot $(e_1,\ f_1) = (1,\ 1)$ ja $(e_s,\ f_s) = (m,\ m)$ siten, että $0 \leq e_{i+1} - e_i \leq 1$ ja $0 \leq f_{i+1} - f_i \leq 1$ kaikilla $i < m$. (Lines & Bagnall, 2015).

Toisin sanoen polun tulee lähteä matriisin ensimmäisestä solusta vasemmasta yläkulmasta ja päätyä matriisin viimeiseen soluun oikeaan alakulmaan. Polku ei saa hypätä matriisin solujen yli eikä se saa myöskään palata takaisinpäin, koska kahden peräkkäisen indeksi erotus tulee olla joko 0 tai 1. Indeksi voi siis pysyä samana tai kasvaa yhdellä, mutta ei pienentyä.

DTW etäisyys aikasarjojen välillä on polku läpi matriisin M, joka minimoi kokonaisetäisyyden sallitun vääristymän rajoitteiden mukaisesti. Olkoon $p_i = M_{\mathrm{x}_{1e_i},\mathrm{x}_{2f_i}}$ havaintojen välinen etäisyys aikasarjan x_1 sijainnissa e_i ja aikasarjan x_2 sijainnissa f_i i:nnelle pisteparille ehdotetulla vääristymäpolulla P. Minkä tahansa polun P etäisyys on

$$D_P(\mathrm{x}_1,\ \mathrm{x}_2) = \sum_{i=1}^{s} p_i.$$

Jos $\mathcal{P}$ on kaikkien mahdollisten polkujen muodosta avaruus, vääristymäpolku P^* on polku, jolla on pienin etäisyys

$$P^* = \min_{P \in \mathcal{P}}(D_P(\mathrm{x}_1,\ \mathrm{x}_2)),$$

ja näin ollen DTW etäisyys aikasarjojen välillä on

$$D_{P^*}(\mathrm{x}_1,\ \mathrm{x}_2) = \sum_{i=1}^{k} p_i.$$

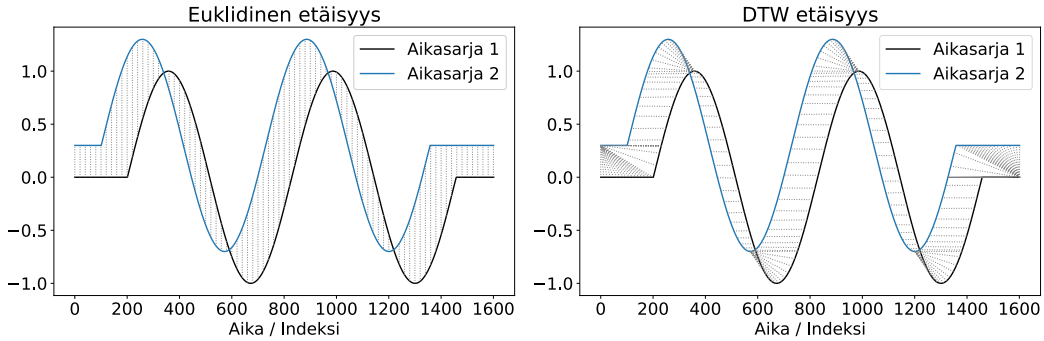
(Lines & Bagnall, 2015).

Optimaalinen polku P^* voidaan löytää täsmällisesti käyttämällä dynaamista ohjelmointia (*dynamic programming*). Tämä saattaa olla ajallisesti vaativa tehtävä, joten yleensä sallitun vääristymän määrää rajoitetaan. Rajoitus on yhtäläistä sille, että kaikille indeksipareille ehdotetulla polulla asetetaan suurin hyväksytty etäisyys. Jos vääristymäikkuna (*warping window*) r on sallitun vääristymän osuus aikasarjan kokonaispituudesta m, on optimaalinen polku rajoitettu siten, että

$$|e_i - f_i| \leq r \cdot m \quad \forall\,(e_i,\ f_i) \in P^*.$$

(Lines & Bagnall, 2015).

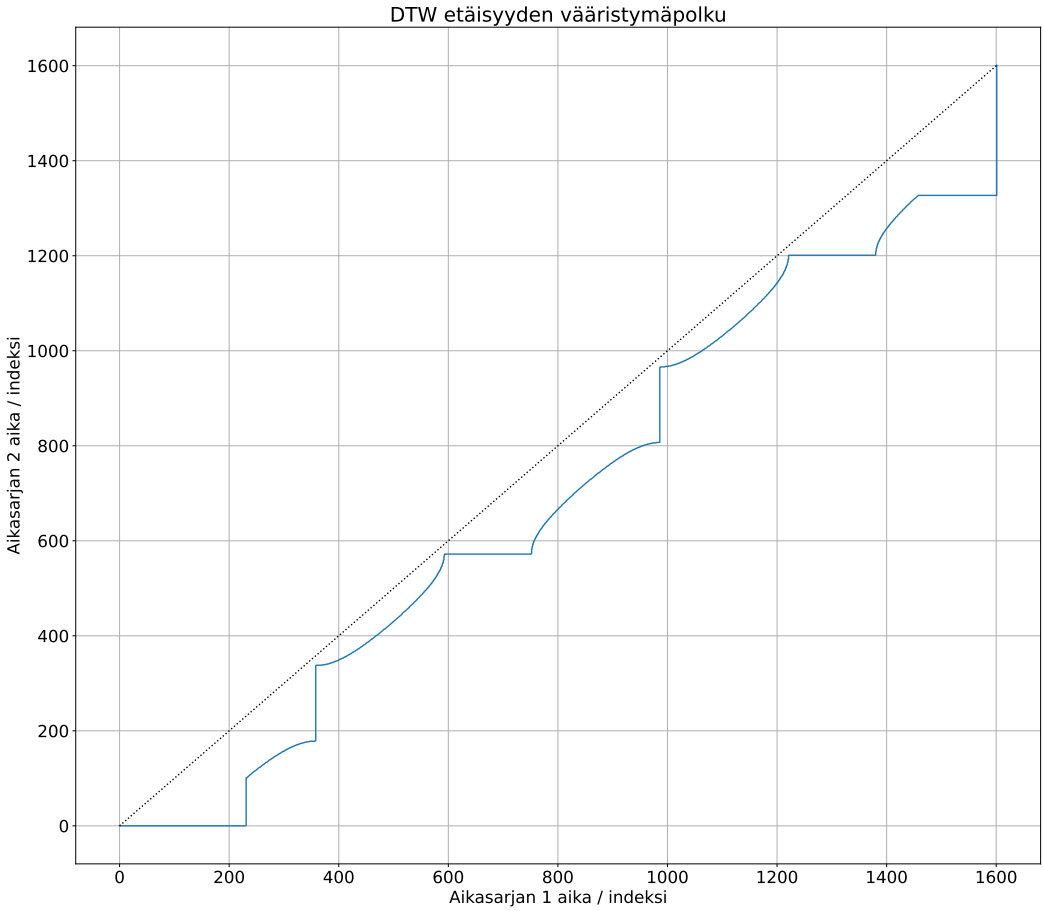
Kuvassa 2 on tehty yksinkertainen vertailu euklidisen etäisyyden ja DTW etäisyyden välillä. Aikasarjat ovat siniaaltoja, joiden alussa ja lopussa on tasaiset jaksot. Euklidinen etäisyys mittaa etäisyyden aina aikasarjojen välillä samassa aikapisteessä, kun DTW etäisyys pyrkii löytämään aikasarjoista samanlaisen muodon sekä lyhimmän mahdollisen etäisyyden niiden välillä. Aikapisteet, joiden välillä mittaus tehdään, voivat poiketa aikasarjojen välillä huomattavasti, kuten DTW etäisyyden kuvaajasta on havaittavissa.



Kuva 2: Vertailu euklidisen etäisyyden ja DTW etäisyyden erosta. Kuvassa on käytetty kahta siniaaltoa, joista toista on aikaistettu 100 aikayksikön verran ja siirretty 0,3 yksikköä korkeammalle, jotta erot olisivat selvemmin näkyvillä. Vertailtavat pisteet on merkitty pisteviivalla aikasarjojen välillä. DTW etäisyyden kuvaaja on tuotettu käyttäen *dtw-python*-pakettia (Giorgino, 2009).

Kuvassa 3 on puolestaan esitetty kuvan 2 aikasarjojen DTW etäisyyden vääristymäpolku. Kuvaajasta nähdään, että alussa vain aikasarjan 1 indeksi kasvaa ja aikasarjan 2 indeksi pysyy nollassa. Aikasarjan 2 indeksi alkaa kasvaa vasta kun aikasarjan 1 siniaalto-osuus alkaa. Sama ilmiö on nähtävissä ku-

vassa 2 kun pisteviivoilla merkityt vertailtavat pisteet pysyvät ylemmän aikasarjan osalta alkuun paikallaan. Kuvaajaan on piirretty lisäksi euklidisen etäisyyden ”vääristymäpolku”, joka kulkee kuvaajan lävistäjällä ja on siis täysin suora. Välillä DTW:n vääristymäpolku käy lähellä lävistäjää, mutta poikkeaa siitä kuitenkin selkeästi.



Kuva 3: Kuvan 2 aikasarjojen vääristymäpolku. Euklidisen etäisyyden ”vääristymäpolku” kulkee kuvaajan lävistäjällä ja on merkitty pisteviivalla. Kuvaaja on tuotettu käyttäen *dtw-python*-pakettia (Giorgino, 2009).

2.2.3 Derivaatallinen dynaaminen aikavääristymä

Dynaamisen aikavääristymän derivaatallinen versio (*Derivative Dynamic Time Warping*, jatkossa DDTW) on saanut nimensä siitä, että aikasarjoille tehdään muunnos ensimmäisen asteen erotuksiksi, joka on yksinkertainen numeerinen menetelmä ensimmäisen asteen derivaatalle (Lines & Bagnall,

2015). Ajatus tällaisen mitan käyttöönotossa on ollut ehkäistä singulariteettien syntyminen eli tilanne, jossa yksittäinen aikasarjan piste kartoittuu isoon osajoukkoon toisessa aikasarjassa (Lines & Bagnall, 2015). Kuvassa 2 tämä ilmiö on havaittavissa normaalilla DTW etäisyydellä aikasarjojen alussa ja lopussa.

Merkitään aikasarjan $\mathrm{x}_1 = (x_{1_1}, \ldots, x_{1_m})$ erotusaikasarjaa

$$\mathrm{x}'_1 = (x'_{1_2}, \ldots, x'_{1_{m-1}}),$$

jossa x'_{1_i} on määritetty aikasarjan havaintojen $x_{1_{i-1}}$ ja x_{1_i} välisen kulmakertoimen sekä havaintojen $x_{1_{i-1}}$ ja $x_{1_{i+1}}$ välisen kulmakertoimen keskiarvona, eli

$$x'_{1_i} = \frac{(x_{1_i} - x_{1_{i-1}}) + ((x_{1_{i+1}} - x_{1_{i-1}})/2)}{2}, \quad 1 < i < m.$$

DDTW on suunniteltu lieventämään kohinaa, joka saattaa vaikuttaa haitallisesti tavalliseen dynaamiseen aikavääristymään ja on usein käytetty yhtäaikaisesti DTW:n kanssa aikasarjojen samanlaisuuden mittaamiseen. (Lines & Bagnall, 2015).

2.2.4 Painotettu dynaaminen aikavääristymä

Painotettu dynaaminen aikavääristymä (*Weighted Dynamic Time Warping*, jatkossa WDTW) lisää multiplikatiivisen, sakottavan painon, joka perustuu vääristymäetäisyyteen vääristymäpolun pisteiden välillä. Se toisin sanoen suosii pientä vääristymää ja sakottaa suurta vääristymää. Etäisyysmatriisia M muodostettaessa otetaan käyttöön paino $w_{|i-j|}$ siten, että

$$M_{i,j} = w_{|i-j|}(x_{1_i} - x_{2_j})^2.$$

Painofunktiona voidaan käyttää esimerkiksi logistista funktiota, eli vääristymä, jonka pisteiden etäisyys on $|i - j|$ määrää painon

$$w(|i-j|) = \frac{w_{max}}{1 + e^{-g(|i-j|-m/2)}},$$

jossa w_{max} on painon yläraja (asetettu 1:een), m on aikasarjan pituus ja g on parametri, joka säätää painon määrää suurilla vääristymillä. Mitä suurempi g on, sitä suurempi on vääristymän paino eli sakko. (Lines & Bagnall, 2015).

2.3 Aikasarjametsä

Aikasarjametsä (*Time Series Forest, TSF*) on aikasarjojen luokitteluun sovitettu muunnos satunnaismetsästä (*Random Forest, RF*). Aikasarjametsä on intervallipiirteisiin (*interval features*) perustuva luokittelija, ja intervalli tarkoittaa tässä tapauksessa osaa aikasarjasta, esimerkiksi väliä ajanhetkestä 50 ajanhetkeen 100. Intervalleille voidaan laskea monia erilaisia piirteitä, kuten esimerkiksi keskiarvo ja keskihajonta. Piirteet määritellään seuraavasti: Olkoon K piirteiden määrä ja $f_k(\cdot)$, $k = 1, \ldots, K$ olkoon k:s piirre. Tällöin $f_k(t_1, t_2)$, jossa $1 \leq t_1 \leq t_2 \leq m$, tarkoittaa k:nnetta piirrettä laskettuna aikapisteiden t_1 ja t_2 määrittämässä intervallissa. Nyt jos esimerkiksi keskiarvo olisi ensimmäinen laskettava piirre aikasarjalle x_1, merkitään se

$$f_1(t_1, t_2) = \frac{\sum_{i=t_1}^{t_2} x_{1_i}}{t_2 - t_1 + 1}.$$

(Deng ym., 2013).

2.3.1 Jakokriteeri

Aikasarjapuu (*time series tree*) on aikasarjametsän peruskomponentti. Jakokriteeriä (*splitting criterion*) käytetään tunnistamaan paras tapa jakaa puun solmu (*node*) uusiksi alisolmuiksi. Ehdolla oleva jako S aikasarjapuun solmussa testaa ehdon

$$f_k(t_1, t_2) \leq \tau, \tag{3}$$

jossa τ on kynnysarvo. Havainnot, jotka täyttävät ehdon, menevät vasempaan alisolmuun, ja muut havainnot menevät oikeaan alisolmuun. (Deng ym., 2013).

Olkoon $\{f_k^n(t_1, t_2), n \in 1, \ldots, N\}$ joukko piirteen $f_k(t_1, t_2)$ arvoja laskettuna kaikille opetushavainnoille solmussa. Jotta yhtälön 3 kynnysarvolle τ saataisiin hyvä arvo, yksi menetelmä on järjestää kaikkien opetushavaintojen piirteiden arvot ja tämän jälkeen valita paras kynnysarvo peräkkäisten arvojen keskipisteistä, mutta tämä on voi olla laskennallisesti raskas menetelmä. Deng ym. (2013) ehdottavat ongelman kiertämiseksi seuraavaa menetelmää: Määritetään mahdolliset kynnysarvot tietylle piirteelle $f_k(t_1, t_2)$ siten, että väli $[\min_{n=1}^{N}(f_k^n(t_1, t_2)),\ \max_{n=1}^{N}(f_k^n(t_1, t_2))]$ jaetaan tasapitkiin intervalleihin. Ehdolla olevien kynnysarvojen määrää merkitään κ:lla ja on kiinteä, esimerkiksi 20. Tämän jälkeen paras kynnysarvo valitaan ehdolla olevista arvoista. Tällä menetelmällä arvojen järjestämistä ei tarvitse tehdä ja tarvittavien testien määrä on vain κ. Jakokriteeriä määrittää parhaan jaon, joka merkitään

$$S^* : f_*(t_1^*, t_2^*) \leq \tau^*.$$

(Deng ym., 2013).

Deng ym. (2013) ottavat käyttöön entropiahyödyn (*entropy gain*) ja etäisyysmitan yhdistelmän jakokriteerinä, josta he käyttävät nimitystä *Entrance* (*entropy* ja *distance* sanojen yhdistelmä). Entropiahyöty on yleisesti käytetty jakokriteeri puumalleissa. Merkitään havaintojen luokittaisia osuuksia puun solmuissa γ_c:llä, jossa $c = 1, \ldots, C$ on havainnon luokka ja C on luokkien kokonaismäärä. Solmun entropia määritetään

$$Entropy = -\sum_{c=1}^{C} \gamma_c \log \gamma_c. \tag{4}$$

Jaon entropiahyöty $\Delta Entropy$ on alisolmujen entropian painotetun summan ja alisolmujen vanhemman entropian erotus. Alisolmun paino on kyseessä olevaan solmuun määritettyjen havaintojen osuus. Yleistetty kaava entropiahyödylle on esitetty yhtälössä 5, jossa P on alisolmujen vanhempi (*parent*), C on alisolmu (*child*), ja $|\cdot|$ on joukon kardinaliteetti eli koko. (Deng ym., 2013).

$$\Delta Entropy = Entropy(P) - \sum_{solmu \in \{\text{vasen, oikea}\}} \frac{|C_{solmu}|}{|P|} \cdot Entropy(C_{solmu}). \tag{5}$$

Aikasarjojen luokittelussa ehdolla olevien jakojen määrä voi olla suuri ja on yleistä, että usealla ehdolla olevalla jaolla on sama entropiahyöty. Tämän vuoksi Deng ym. (2013) käyttävät lisäksi mittaa nimeltä $Margin$, joka laskee etäisyyden ehdolla olevan kynnysarvon ja sitä lähinnä olevan piirteen arvon välillä. Jaon $f_k(t_1, t_2) \leq \tau$ $Margin$ lasketaan

$$Margin = \min_{n=1,\ldots,N} |f_k^n(t_1, t_2) - \tau|,$$

jossa $f_k^n(t_1, t_2)$ on piirteen $f_k(t_1, t_2)$ arvo n:nnelle havainnolle solmussa. Seuraavaksi määritetään uusi jakokriteeri $Entrance$, joka määritellään entropiahyödyn ja $Margin$:in yhdistelmänä

$$Entrance = \Delta Entropy + \alpha \cdot Margin,$$

jossa α on jokin pieni luku, sillä sen rooli mallissa on vain estää tasatilanteet, joita voi esiintyä entropiahyödyllä yksistään. Vaihtoehtoisesti entropiahyödyn ja $Margin$:in arvot voidaan tallentaa ja käyttää $Margin$:ia päättämään tasatilanteet, jos kahdella jaolla on sama entropiahyöty. (Deng ym., 2013).

Luonnollisesti jako, jolla on suurin *Entrance*, tulisi valita solmun jaoksi. $Margin$ ja *Entrance* ovat kuitenkin herkkiä piirteiden mitta-asteikolle. Siispä Deng ym. (2013) käyttävät seuraavaa strategiaa, jos piirteet ovat eri mitta-asteikoilla: Jokaiselle piirteelle f_k valitaan jako, jolla on suurin *Entrance*-hyöty, ja eri piirteiden parhaista jaoista valitaan se, jolla on suurin entropiahyöty $\Delta Entropy$. Jos useammalla piirteellä on sama suurin entropiahyöty, valitaan paras jako satunnaisesti niiden joukosta. (Deng ym., 2013).

2.3.2 Aikasarjapuu ja aikasarjametsä

Aikasarjapuun muodostaminen noudattaa rekursiivista, ylhäältä-alas-strategiaa, joka on samanlainen kuin tavanomaisilla päätöspuualgoritmeilla, mutta käyttää *Entrance*-hyötyä jakokriteerinä. Lisäksi satunnaismetsässä käytettyä satunnaisotantastrategiaa hyödynnetään aikasarjametsässä. Satunnaismetsä testaa $\sqrt{p}$ piirrettä satunnaisesti jokaiselle solmulle, kun kaikkien piirteiden määrä on p. Jokaiselle aikasarjapuun solmulle otetaan satunnaisotos intervallien kokoja, jonka kompleksisuus on $O(\sqrt{m})$, ja satunnaisotos alkupisteitä, jonka kompleksisuus on myös $O(\sqrt{m})$. Tällä saadaan piirreavaruuden kompleksisuus pienennettyä kokoon $O(m)$. (Deng ym., 2013).

Algoritmi 1 kuvaa satunnaisotoksen algoritmin. Ensinnä arvotaan $\sqrt{m}$ kokoinen otos intervallien pituuksia. Tämän jälkeen jokaiselle intervallin pituudelle arvotaan $\sqrt{m}$ kappaletta alkupisteitä ja muodostetaan alkupisteitä vastaavat loppupisteet lisäämällä alkupisteisiin intervallin pituus. Lopuksi palautetaan alku- ja loppupisteiden joukot.

Algoritmi 2 kuvaa, kuinka aikasarjapuu muodostetaan. *Otos* funktiolla poimitaan ensinnä satunnaisotos intervallien alku- ja loppupisteitä. Tämän jälkeen lasketaan jokaiselle piirteelle ehdokaskynnykset ja asetetaan tarvittaville asteriskilla merkityille muuttujille alkuarvot. Muuttujat ovat sitä varten, että niihin voidaan myöhemmässä vaiheessa tallettaa arvoja. Seuraavaksi lähdetään käymään läpi kaikki intervallit, niille lasketaan intervallipiirteet ja jokaisen piirteen ehdokaskynnykset, joille lasketaan $\Delta Entropy$ ja *Entrance*. Jos *Entrance* on suurempi kuin $Entrance^*$, asetetaan asteriskilla merkittyihin muuttujiin ilman asteriskia olevien muuttujien arvot eli kyseisen for-silmukan muuttujien arvot. For-silmukoiden päätyttyä tarkistetaan, onko $\Delta Entropy$ nolla ja jos on, niin kyseinen solmu merkitään lehtisolmuksi eli solmuksi, jota ei enää jaeta alisolmuiksi, sekä palataan algoritmista pois. Jos $\Delta Entropy$ ei ollut nolla, aineisto jaetaan vasempaan ja oikeaan alisolmuun kynnysarvon mukaisesti, ja algoritmia kutsutaan rekursiivisesti jaetuilla aineistoilla.

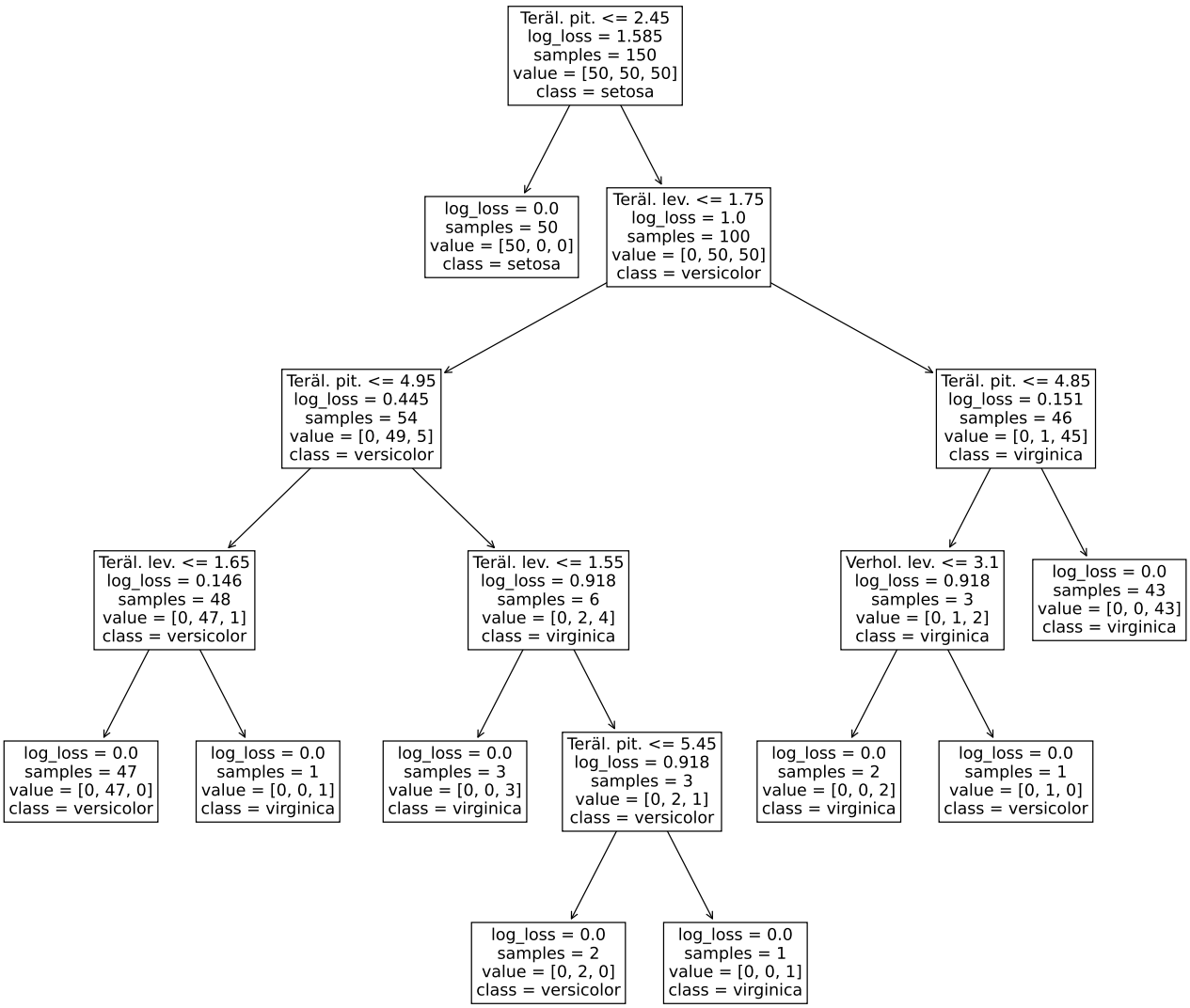
Algoritmi 1 Deng ym. (2013) kuvaama funktio *otos*(), joka palauttaa satunnaisotoksen intervalleja (T_1, T_2), jossa T_1 on joukko intervallien alkuaikoja ja T_2 on joukko intervallien loppuaikoja. Funktio $satOtosIlmanPalaut(joukko, otoskoko)$ valitsee satunnaisesti *otoskoko* parametrin osoittaman määrän alkioita joukosta *joukko* ilman palautusta.

Require: m (aikasarjan pituus)
  $T_1 \leftarrow \emptyset$, $T_2 \leftarrow \emptyset$
  $W \leftarrow satOtosIlmanPalaut(\{1, \ldots, m\}, \sqrt{m})$
  for w joukosta W **do**
    $T_1 \leftarrow satOtosIlmanPalaut(\{1, \ldots, m - w + 1\}, \sqrt{m - w + 1})$
    for t_1 joukosta T_1 **do**
      $T_2 \leftarrow T_2 \cup (t_1 + w - 1)$
    end for
  end for
  return (T_1, T_2)

Aikasarjametsä on kokoelma aikasarjapuita, ja aikasarjametsä luokittelee testattavan havainnon luokkaan, johon enemmistö kaikista aikasarjapuista kyseisen havainnon äänestää (*vote*) eli luokittelee (Deng ym., 2013). Kuvassa 4 on havainnollistettu, miltä päätöspuu voi näyttää. Puu on muodostettu kurjenmiekka-aineistosta yksinkertaisella päätöspuuluokittelijalla (*decision tree classifier*), jonka jakokriteerinä on käytetty informaatiohyötyä. Kuvasta nähdään, että esimerkiksi juurisolmussa (ylin solmu) on käytetty terälehden pituutta muuttujana, jonka mukaan luokittelu tehdään, ja kynnysarvo on ollut 2,45 senttimetriä. Havainnot, joiden terälehden pituus on pienempi kuin kynnysarvo, menevät vasempaan alisolmuun ja muut puolestaan oikeaan alisolmuun. Vasen alisolmu on tässä tapauksessa myös lehtisolmu, eli sitä ei enää jaeta useammiksi alisolmuiksi, ja sinne on luokiteltu kaikki 50 setosalajin havaintoa. Oikeaan alisolmuun jää kaikki versicolor- ja virginica-lajien havainnot, joten se pitää jakaa uudelleen, ja prosessi toistetaan niin kauan, että kaikki havainnot on saatu luokiteltua.

Algoritmi 2 Deng ym. (2013) kuvaama funktio *puu*() aikasarjapuun muodostamiseen. Algoritmin yksinkertaistamiseksi oletetaan, että eri tyyppisten piirteiden arvoasteikko on sama, jotta *Entrance*:a voidaan vertailla suoraan.

Require: D (luokiteltava aikasarja-aineisto)
  $(T_1, T_2) \leftarrow otos(m)$
  Laske $Kynnys_k$, eli joukko ehdokaskynnyksiä jokaiselle piirteelle k
  $Entrance^* \leftarrow 0$
  $\Delta Entropy^* \leftarrow 0$
  $t_1^* \leftarrow 0$
  $t_2^* \leftarrow 0$
  $\tau^* \leftarrow 0$
  $f_* \leftarrow \emptyset$
  for (t_1, t_2) joukoista (T_1, T_2) **do**
    for k menee $1 : K$ **do**
      for τ joukosta $Kynnys_k$ **do**
        Laske $\Delta Entropy$ ja $Entrance$ piirteelle $f_k(t_1, t_2) \leq \tau$
        if $Entrance > Entrance^*$ **then**
          $Entrance^* \leftarrow Entrance$
          $\Delta Entropy^* \leftarrow \Delta Entropy$
          $t_1^* \leftarrow t_1$
          $t_2^* \leftarrow t_2$
          $\tau^* \leftarrow \tau$
          $f_* \leftarrow f_k$
        end if
      end for
    end for
  end for
  if $\Delta Entropy = 0$ **then**
    Merkitse solmu lehtisolmuksi ja poistu funktiosta
  end if
  $data_{vasen} \leftarrow$ aikasarjat joille $f_*(t_1^*, t_2^*) \leq \tau^*$
  $data_{oikea} \leftarrow$ aikasarjat joille $f_*(t_1^*, t_2^*) > \tau^*$
  $puu(data_{vasen})$
  $puu(data_{oikea})$



Kuva 4: Esimerkki päätöspuusta, joka on muodostettu kurjenmiekka-aineistosta. Jakokriteerinä on käytetty informaatiohyötyä. Solmun ylin rivi on muuttuja, jonka perusteella jako on tehty sekä käytetty kynnysarvo, jos kyseessä ei ole lehtisolmu (solmu, joka ei haaraudu enää). Toisella rivillä on solmun entropia, joka tässä tapauksessa on ilmoitettu *log_loss* terminä. Kolmantena on solmussa olevien havaintojen määrä. Neljäntenä on tieto, kuinka solmussa olevat havainnot ovat jakautuneet eri luokkiin, ja viimeisenä on luokka, johon kyseinen (lehti)solmu havainnot luokittelee.

2.4 Shapelet-muunnos

Shapelet sanalle ei ole virallista suomennosta, eikä sen määritelmää löydy myöskään Oxfordin sanakirjasta. Hills ym. (2014) kuvaavat shapeletin olevan aikasarjan alijakso, joka mahdollistaa aikasarjaluokittelun pohjautuen paikalliseen, vaiheriippumattomaan samanlaisuuteen muodon suhteen. Shapelet sanaa käytetään tässä tutkielmassa kuin se olisi oikea suomen kielen sana.

Shapeleteihin perustuva luokittelu käyttää samanlaisuutta shapeletin ja aikasarjan välillä erottelevana piirteenä. Tämän lähestymistavan yksi hyvä puoli on siinä, että shapeletit ovat konkreettisia aikasarjan osia ja saattavat tarjota ymmärrystä ongelman tilasta. Toisin sanoen shapeletien voidaan ajatella olevan aikasarjan alijaksoja, jotka kuvaavat kyseisen aikasarjan luokkaa mahdollisimman hyvin. (Hills ym., 2014).

2.4.1 Shapeleteista

Jokaista alijaksoa jokaisessa aikasarjassa aineistossa **D** kutsutaan ehdokkaaksi. Shapeletit etsitään ehdokkaiden joukosta tyhjentävällä haulla minimi- ja maksimipituuksien välillä ja ne normalisoidaan riippumattomasti. Normalisointi tehdään sen takia, että olemme kiinnostuneita tunnistamaan paikallisia muodon samanlaisuuksia, joten niiden pitää olla muuttumattomia mittaasteikolle ja vaihesiirrolle. Shapeletien haussa on kolme pääkomponenttia: ehdokkaiden muodostaminen, samanlaisuuden mittaaminen shapeletin ja aikasarjan välillä, ja shapeletin laadun mittaus valitulla mittarilla. (Hills ym., 2014).

Algoritmi 3 kuvaa geneerisesti, kuinka shapeletin etsintä tapahtuu. Algoritmi käy for-silmukalla läpi kaikki pituudet l minimin ja maksimin välillä, generoi ehdokas-shapeletit aineistosta **D** pituudella l, käy for-silmukalla kaikki ehdokas-shapeletit läpi ja laskee niille *laatu* muuttujan arvon. Jos kyseinen laatu on suurempi kuin *paras* muuttujaan asetettu arvo, saa *paras* arvokseen *laatu* muuttujan arvon, ja *parasShapelet* muuttujaan talletetaan kyseinen ehdokas-shapelet. Lopuksi algoritmi palauttaa parhaan shapeletin.

2.4.2 Ehdokkaiden generointi ja etäisyyksien laskenta

Aikasarja, pituudeltaan m, sisältää $(m-l)+1$ erillistä l:n pituista ehdolla olevaa shapeletia. Merkitään aikasarjan x_i kaikkien normalisoitujen l:n pituisten alisarjojen joukkoa $W_{i,l}$, ja koko aineistolle **D** merkitään

$$W_l = \{W_{1,l} \cup W_{2,l} \cup \cdots \cup W_{n,l}\}.$$

Algoritmi 3 Geneerinen funktio *shapeletinValinta()* parhaan shapeletin etsintään (Hills ym., 2014).

Require: **D**, min, max
  $paras \leftarrow 0$
  $parasShapelet \leftarrow \emptyset$
  for l menee $min : max$ **do**
    $W_l \leftarrow generoiKandidaatit(\mathbf{D}, l)$
    for alijakso S joukossa W_l **do**
      $D_S \leftarrow etsiEtaisyydet(S, \mathbf{T})$
      $laatu \leftarrow arvioKandidaatti(S, D_S)$
      if $laatu > paras$ **then**
        $paras \leftarrow laatu$
        $parasShapelet \leftarrow S$
      end if
    end for
  end for
  return $parasShapelet$

Kaikkien ehdokkaiden joukko aineistolle **D** on

$$\mathbf{W} = \{W_{min} \cup W_{min+1} \cup \cdots \cup W_{max-1} \cup W_{max}\},$$

jossa $min \geq 3$ ja $max \leq m$. Joukossa **W** on $|\mathbf{W}| = \sum_{l=min}^{max} n(m - l + 1)$ ehdolla olevaa shapeletia. (Hills ym., 2014).

Etäisyys aikasarjan x_i ja l:n pituisen alisarjan S välillä on pienin etäisyys S:n ja kaikkien aikasarjan x_i normalisoitujen l:n pituisten alisarjojen välillä

$$d_{S,i} = \min_{R \in W_{i,l}} d_E(S, R),$$

jossa Euklidinen etäisyys d_E on määritetty ilman neliöjuurta. Etäisyydet kaikkien ehdolla olevien shapeletien S ja aikasarjojen välillä aineistossa **D** lasketaan ja niistä muodostetaan n pituinen lista

$$D_S = (d_{S,1}, d_{S,2}, \ldots, d_{S,n}).$$

(Hills ym., 2014).

2.4.3 Shapeletin laadun arviointi

Algoritmi 3 sisältää funktion shapeletin laadun arviointiin. Shapeletien laatu perustuu siihen, kuinka hyvin luokat C erotetaan toisistaan etäisyysjoukolla

D_S. Tavanomainen lähestymistapa on käyttää informaatiohyötyä (*information gain, IG*) shapeletien laadun määrittämiseen. Joukko D_S järjestetään ja informaatiohyöty jokaiselle mahdolliselle shapeletin S jakopisteelle sp arvioidaan. Kelvollinen jakopiste on keskiarvo minkä tahansa kahden peräkkäisen etäisyyden välillä joukossa D_S. Jokaiselle mahdolliselle jakopisteelle lasketaan informaatiohyöty osioimalla kaikki alkiot, joille pätee $D_S < sp$, joukkoon A_S ja kaikki muut alkiot joukkoon B_S. Informaatiohyöty IG jakopisteelle sp lasketaan kaavalla

$$IG(D_S, sp) = H(D_S) - \left(\frac{|A_S|}{|D_S|} H(A_S) + \frac{|B_S|}{|D_S|} H(B_S) \right), \tag{6}$$

jossa $|\cdot|$ on joukon kardinaliteetti eli koko, ja $H(\cdot)$ on joukon entropia, joka lasketaan yhtälöllä 4. Informaatiohyöty shapeletille S lasketaan hakemalla maksimi jakopisteiden sp informaatiohyödyistä

$$IG_S = \max_{sp \in D_S} IG(D_S, sp).$$

(Hills ym., 2014). Yhtälöstä 6 voidaan havaita, että Deng ym. (2013) käyttämä entropiahyöty, eli yhtälö 5, on itse asiassa informaatiohyöty.

Informaatiohyödyn laskeminen vaatii listan D_S järjestämisen ja tämän jälkeen kaikkien jakopisteiden arvioinnin, josta aiheutuu ylimääräinen kompleksisuus $O(n \log n)$ jokaiselle shapeletille, mutta tämä on käytännössä triviaalia verrattuna listan D_S laskemiseen, jonka kompleksisuus on $O(nml)$. Hills ym. (2014) osoittavat artikkelissaan, että esimerkiksi F-suure on merkittävästi nopeampi laskea ja on paremmin erotteleva kuin informaatiohyöty aikasarjojen luokittelussa.

2.4.4 Tekniikoita haun nopeuttamiseksi

Hills ym. (2014) esittävät kolme erilaista menetelmää shapeletien haun nopeuttamiseksi: shapeletin S ja aikasarjan x_i etäisyyden laskennan aikainen lopetus, aikasarjojen välisten etäisyystunnuslukujen laskeminen etukäteen ja shapeletin laadun arvioinnin aikainen lopetus. Lisäksi Bostrom & Bagnall (2017) esittelevät neljännen menetelmän, binääriset shapeletit, jota myös *sktime*-paketti käyttää.

Etäisyyden laskemisen aikainen lopetus Shapeletin ja aikasarjan välisen etäisyyden laskemisen aikainen lopetus on mahdollista, koska $d_{S,i}$ on minimi alijaksojen etäisyyksistä S:n ja x_i:n välillä, joten yksittäiset laskennat voidaan lopettaa, jos ne ovat suurempia kuin paras siihen mennessä löydetty arvo. (Hills ym., 2014).

Algoritmi 4 kuvaa, kuinka aikainen lopetus käytännössä toimii: Alisarja S ja aikasarja x normalisoidaan, normalisoidun alisarjan indeksit järjestetään, ja lasketaan ensimmäinen etäisyys normalisoimattoman alisarjan ja normalisoidun aikasarjan välillä, joka asetetaan tässä vaiheessa parhaaksi etäisyydeksi. Tämän jälkeen käydään for-silmukalla läpi kaikki indeksit 1:stä $l - m$:ään, jossa l on alisarjan pituus ja m on aikasarjan pituus. Toisin sanoen käydään läpi kaikki aikasarjan x l:n pituiset alisarjat. Seuraavaksi lasketaan juoksevat summat, joita käytetään seuraavassa vaiheessa normalisoinnin parametreina. While-silmukassa lasketaan alisarjan S uudelleenjärjestettyä etäisyyttä d aikasarjaan x, joka samalla normalisoidaan, niin kauan kuin while-silmukan iteraattori j on pienempi kuin alisarjan pituus l tai etäisyys d on pienempi kuin paras etäisyys b. Jos while-silmukan jälkeen j on alisarjan pituus l ja etäisyys d on pienempi kuin paras etäisyys b, asetetaan parhaaksi etäisyydeksi d. Lopuksi palautetaan paras eli pienin etäisyys b.

Etäisyystunnuslukujen laskeminen etukäteen Ratkaisu tarvittavien laskutoimitusten vähentämiseen on laskea etäisyystunnuslukuja etukäteen. Jokaista alisarjaa verrataan toisiinsa, mistä aiheutuu paljon päällekkäisiä laskutoimituksia. Esimerkiksi kun verrataan alisarjoja, joiden alkupisteet ovat a ja b, ovat monet laskennat jo suoritettu aiemmin alisarjoille, joiden alkupisteet ovat $a - 1$ ja $b - 1$. Ratkaisuna tähän ongelmaan on, että jokaiselle parille aikasarjoja (x_i, x_j) lasketaan kumulatiivinen summa, neliösumma ja ristitulo etukäteen, jolloin etäisyys alisarjojen välillä voidaan laskea vakioidussa ajassa $O(n^2m^3)$. Ristitulon etukäteen laskeminen kaikkien aikasarjojen välillä tosin vaatii muistia $O(n^2m^2)$, joka on liikaa useimpien ongelmien kohdalla. Tämän vuoksi onkin järkevämpää laskea tunnusluvut ennen jokaisen aikasarjan skannausta, joka vähentää muistin tarpeen $O(nm^2)$, mutta kasvattaa suoritusaikaa. (Hills ym., 2014).

Laadun arvioinnin aikainen lopetus Tämä nopeutusmenetelmä toimii siten, että informaatiohyödylle IG haetaan yläraja olettamalla optimistisin arvon asetus sen jälkeen kun jokainen etäisyys $d_{S,i}$ on laskettu. Jos kyseinen yläraja on alle tähän mennessä löydetyn parhaan arvon, joukon D_S:n laskenta voidaan keskeyttää. Tällä on potentiaalisesti suuri nopeusetu, kun heikot shapeletit voidaan hylätä, joskin parhaan jaon ja ylärajan laskeminen jokaiselle uudelle etäisyydelle $d_{S,i}$ aiheuttaa hieman ylimääräistä suoritusaikaa. Moniluokkaisissa ongelmissa oikea yläraja voidaan löytää vain käymällä läpi asetetut jaot jokaiselle mahdolliselle luokalle, mikä voi lisätä suoritusaikaa huomattavasti. (Hills ym., 2014).

Algoritmi 4 Samanlaisuushaku online-normalisoinnilla ja uudelleenjärjestetyllä aikaisella lopetuksella (Hills ym., 2014).

Require: Aikasarja x $= (x_i, \ldots, x_m)$ ja alisarja $S = (s_i, \ldots, s_l)$, jossa $l < m$.
  $S' \leftarrow normalisoi(S,\ 1,\ l)$
  (A_i on i:nneksi suurimman absoluuttisen arvon indeksi S':ssa)
  $A \leftarrow jarjestaIndeksit(S')$
  $F \leftarrow normalisoi(x,\ 1,\ l)$
  $p \leftarrow 0$ (säilöö juoksevan summan)
  $q \leftarrow l$ (säilöö neliöiden juoksevan summan)
  $b \leftarrow etaisyys(S,\ F)$ (etsi ensimmäinen etäisyys ja aseta parhaaksi)
  (Skannaa läpi kaikkien alisarjojen)
  for i menee $1 : (m - l)$ **do**
    (päivitä juoksevat summat)
    $p \leftarrow p - x_i,\ q \leftarrow q - x_i^2,\ p \leftarrow p + x_{i+l}$
    $q \leftarrow q + x_{i+l}^2,\ \bar{t} \leftarrow \frac{p}{l},\ s \leftarrow \frac{q}{l} - \bar{t}^2$
    $j \leftarrow 1,\ d \leftarrow 0$
    (Etäisyys S:n ja $(x_{i+1}, \ldots, x_{i+l+1})$ välillä aikaisella lopetuksella)
    while $j \leq l$ ja $d < b$ **do**
      (uudelleenjärjestetty online-normalisointi)
      $d \leftarrow d + \left(S_{A_j} - \frac{x_{i+A_j} - \bar{t}}{s}\right)^2$
      $j \leftarrow j + 1$
    end while
    if $j = 1$ ja $d < b$ **then**
      $b \leftarrow d$
    end if
  end for
  (Minimietäisyys S:n ja kaikkien l:n pituisten x:n alisarjojen välillä)
  return b

Binääriset shapeletit Shapeletien laadunarvioinnin vakiomenetelmä mittaa kuinka hyvin shapelet jakaa kaikki luokat. Bostrom & Bagnall (2017) esittävät kolme potentiaalista tästä johtuvaa ongelmaa moniluokkaisten ongelmien luokittelussa: Ensimmäinen ongelma on se, että hyödyllistä, yhtä luokkaa koskevaa informaatiota, voidaan menettää. Toinen ongelma ilmenee, jos jokin luokista on helpompi luokitella kuin muut. Tällöin on hyvin todennäköistä, että shapeleteja löytyy enemmän helposta luokasta kuin muista luokista. Kolmas ongelma on, ettei etäisyyden laskennan aikainen lopetus ole käyttökelpoinen moniluokkaisten ongelmien tapauksessa.

Bostrom & Bagnall (2017) määrittävät binäärisen shapeletin siten, että sen laadun arviointi mittaa, kuinka hyvin se jakaa aikasarjan luokan, johon shapelet kuuluu, kaikista muista luokista. Tämän pitäisi lieventää ensimmäistä ongelmaa. Toisen ongelman lievittämiseksi he esittävät, että jokaiselle luokalle määritetään suurin mahdollinen shapeletien määrä k/C, jossa k on suurin mahdollinen etsittävien shapeletien määrä, ja C on luokkien määrä. Viimeisen ongelman lieventämiseksi esitetään uusi shapeletin etäisyyden mittaustapa, jossa mittaus aloitetaan aina sen indeksin kohdalta, josta shapeletin alku on löytynyt alkuperäisessä aikasarjassa. Mittaus tehdään kyseisestä pisteestä kumpaankin suuntaan aikasarjassa, jota vasten etäisyys mitataan. Jos lyhin etäisyys löytyy esimerkiksi hieman aloitusindeksiä myöhemmin, voidaan todennäköisesti aloitusindeksiä aiemmin tehtävät mittaukset lopettaa aikaisin. Jos mittaus olisi aloitettu ensimmäisestä indeksistä lähtien, ei tämä olisi mahdollista.

Algoritmi 5 kuvaa funktion, jolla binääriset shapeletit valitaan. Geneeriseen, algoritmin 3 kuvaamaan, shapeletin valinta funktioon verrattuna tässä algoritmissa haetaan k kappaletta parhaita shapeleteja ja muuttujan *osuus* osoittama määrä parhaimman laadun omaavia shapeleteja jokaiselle luokalle. Itsensä kanssa samankaltaiset shapeletit poistetaan, jolla saadaan pienennettyä lopullista shapeletien määrää. Lisäksi algoritmin $etsiEtaisyydet()$ funktiossa käytetään edellisessä kappaleessa kuvattua etäisyyden mittausta aikaisella lopetuksella.

2.4.5 Muunnosprosessi

Muunnosprosessi on tärkeää erottaa varsinaisesta luokittelijasta. Hills ym. (2014) esittämä muunnosprosessi käsittää kolme erillistä vaihetta: Ensinnä algoritmi suorittaa yksinkertaisen aineiston skannauksen, jolla se poimii k parasta shapeletia, jossa k on shapeletien maksimimäärä, paljonko niitä tallennetaan etsinnän aikana. Toiseksi k:n shapeletin joukkoa voidaan pienentää

Algoritmi 5 Bostrom & Bagnall (2017) esittämä funktio binääristen shapeletien valintaan $binaariShapeletinValinta()$.

Require: **D**, min, max, k
  $luokkienMaara \leftarrow laskeLuokkienJakauma(\mathbf{D})$
  $kShapeletitLista \leftarrow \emptyset$
  $osuus \leftarrow \frac{k}{luokkienMaara}$
  for x_i joukossa **D** **do**
    $shapeletit \leftarrow \emptyset$
    for l menee $min : max$ **do**
      $W_{i,l} \leftarrow generoiKandidaatit(x_i, l)$
      for alisarja S joukossa $W_{i,l}$ **do**
        $D_S \leftarrow etsiEtaisyydet(S, \mathbf{D})$
        $laatu \leftarrow arvioKandidaatti(S, D_S)$
        $shapeletit.lisaa(S, laatu)$
      end for
    end for
    $lajitteleLaadunMukaan(shapeletit)$
    $poistaSamankaltaiset(shapeletit)$
    $kShapeletit \leftarrow x_i.luokka$
    $kShapeletit \leftarrow yhdista(osuus, kShapeletit, shapeletit)$
    $kShapeletitLista.lisaa(kShapeletit, x_i.luokka)$
  end for
  return $kShapeletitLista$

joko hylkäämällä shapeletit tietyn raja-arvon alapuolelta, esimerkiksi vähentämällä joukko 256 shapeletista 10 shapeletiin, tai klusteroimalla shapeletit. Lopuksi luodaan uusi muunnettu aineisto, jossa jokainen muuttuja edustaa shapeletia ja muuttujan arvo on etäisyys shapeletin ja alkuperäisen aikasarjan välillä. Aineiston muuntaminen tällä menetelmällä erottaa shapeletien etsinnän luokittelijasta ja mahdollistaa muunnetun aineiston käyttämisen minkä tahansa luokittelijan kanssa. Luokittelijan ei tarvitse olla aikasarjoille sovitettu, vaan luokittelijaksi soveltuu mikä tahansa luokitteleva koneoppimismenetelmä. (Hills ym., 2014).

2.5 Luokittelijoiden hyperparametrien hienosäätö

Luokittelijoilla on yleensä yksi tai useampi hyperparametri, jotka tulisi hienosäätää siten, että luokittelija tuottaa parhaimmat mahdolliset tulokset. Kyseiset parametrit tulee asettaa koneoppimismallille ennen sen opettamista, eikä malli voi oppia niitä opetusprosessin yhteydessä. Esimerkiksi K-lähinaapurimenetelmän luokittelijalla hienosäädettävä hyperparametri on malliparametri k eli lähimpien naapurien lukumäärä, ja aikasarjametsällä puolestaan käytettävien puiden määrä. Shapelet-muunnoksella on lisäksi algoritmiparametreja, jotka vaikuttavat sekä luokittelun toimivuuteen että suoritusaikaan, joita ovat esimerkiksi shapelet-otoksen koko (*n_ shapelet_ samples*) sekä eräkoko (*batch size*).

Jos opetusaineisto on suuri, voidaan siitä erottaa esimerkiksi 25 prosenttia havainnoista erilliseksi validointiaineistoksi, jota vasten luokittelijan toimivuus eri hyperparametrien arvoilla testataan (Hastie ym., 2009). Kun parhaat hyperparametrien arvot ovat löytyneet, tehdään lopullinen mallin arviointi testiaineistoa käyttäen (Hastie ym., 2009). Validointiaineistoa erotettaessa, kuten myös seuraavissa kappaleissa esiteltävissä menetelmissä, tulee aineiston luokkien jakaumat ottaa huomioon ja käyttää tarvittaessa ositettua (*stratified*) otantaa, jolloin jokaisesta luokasta poimitaan suhteellisesti sama määrä havaintoja (Kubat, 2017). Aikasarjojen luokittelussa aineiston havaintoja eli aikasarjoja käsitellään kokonaisina, jolloin yksi aikasarja kuuluu aina kokonaisuudessaan joko opetus-, validointi- tai testiaineistoon.

Aina aineiston koko ei kuitenkaan ole riittävä, jotta erillistä validointiaineistoa voisi siitä erottaa. Tällöin hyperparametrien hienosäätö voidaan tehdä ristiinvalidoinnilla (*cross-validation, CV*). Pienelle aineistolle voidaan tehdä yksi-pois ristiinvalidointi (*leave-one-out CV, LOOCV*), jossa käydään koko opetusaineisto läpi siten, että yhdellä kierroksella jätetään aina yksi havainto pois, jota vasten testaus, eli luokittelu, tehdään. Haluttujen, luvussa 2.6.2

esiteltyjen mittarien arvot lasketaan jokaisella kierroksella, joille lasketaan lopuksi keskiarvot. Lopulliseen malliin voidaan valita hyperparametrit ristiinvalidoinnin mallista, joka tuotti esimerkiksi korkeimman tarkkuuden tai pienimmän virheosuuden. (Hastie ym., 2009).

Isommille aineistoille yksi-pois ristiinvalidointi ei ole laskennallisesti järkevää, joten tällöin käytetään K-kertaista ristiinvalidointia (*K-fold CV*). Tässä menetelmässä opetusaineisto jaetaan esimerkiksi 5 tai 10 samankokoiseen osaan ja yksi näistä osista jätetään aina yhdellä kierroksella pois mallin opetuksesta, jota vasten tehdään jälleen testaus ja mittarien laskenta. Lopuksi lasketaan mittarien keskiarvot. Mittarien keskiarvojen käyttäminen ehkäisee sitä, että opetusaineiston jako vaikuttaisi hienosäädön lopputulokseen, joka voi olla mahdollista yksittäistä validointiaineistoa käytettäessä. (Hastie ym., 2009).

2.6 Eri menetelmien toimivuuden tarkastelu

Luokittelussa voidaan käyttää useita erilaisia mittareita eri menetelmien toimivuuden ja paremmuuden mittaamiseen. Tässä luvussa avataan yleisesti käytettyjä termejä sekä lyhenteitä ja kuvataan tutkielmassa käytetyt mittarit. Luvun lähteenä on käytetty (Kubat, 2017) lukua 11.

2.6.1 Termit ja lyhenteet

Jotta kappaleessa 2.6.2 esitettyjen eri mittarien laskukaavat olisi helpompi ymmärtää, on taulukkoon 1 koottu tarvittavat termit niiden englanninja suomenkielisillä selitteillä sekä niiden kaavoissa käytetyt lyhenteet. *True positive* ja *true negative* termien juuret ovat todennäköisesti lääketieteessä, jossa usein testien tulos on joko positiivinen tai negatiivinen. Näitä termejä kuitenkin käytetään kaksiluokkaisissa (binäärisissä) ongelmissa, vaikka luokat eivät olisikaan ”positiivinen” ja ”negatiivinen”. Useampiluokkaisissa ongelmissa nämä eivät kuitenkaan ole enää käyttökelpoiset termit, vaan oikein luokiteltuja voitaisiin merkitä esimerkiksi termeillä *true class 1* (TC_1), *true class 2* (TC_2),..., *true class C* (TC_C), jossa C on luokkien määrä. Vastaavasti väärin luokitellut voitaisiin merkitä *false class 1,2* ($FC_{1,2}$), *false class 2,1* ($FC_{2,1}$),..., *false class C,C-1* ($FC_{C,C-1}$), joissa ensimmäinen indeksi kertoo todellisen luokan ja jälkimmäinen indeksi mihin luokkaan havainto on virheellisesti luokiteltu. Numeroiden tilalle voidaan asettaa luokkien todelliset nimet.

Kuvassa 5 on esitetty kaksiluokkaisen ongelman luokittelutaulukko (harmaal-

Lyhenne	Selite	
	englanniksi	suomeksi
TP	true positive	oikein positiiviseksi luokiteltujen havaintojen lukumäärä
TN	true negative	oikein negatiiviseksi luokiteltujen havaintojen lukumäärä
FP	false positive	väärin positiiviseksi luokiteltujen havaintojen lukumäärä
FN	false negative	väärin negatiiviseksi luokiteltujen havaintojen lukumäärä
Acc	accuracy	tarkkuus
Se	sensitivity / recall	sensitiivisyys, herkkyys
Sp	specificity	spesifisyys
PPV	positive predictive value / precision	positiivinen ennustearvo
F_β	F-score	F-mitta

Taulukko 1: Luokittelumittareissa käytettyjä lyhenteitä, mittareiden lyhenteitä sekä niiden selitteet.

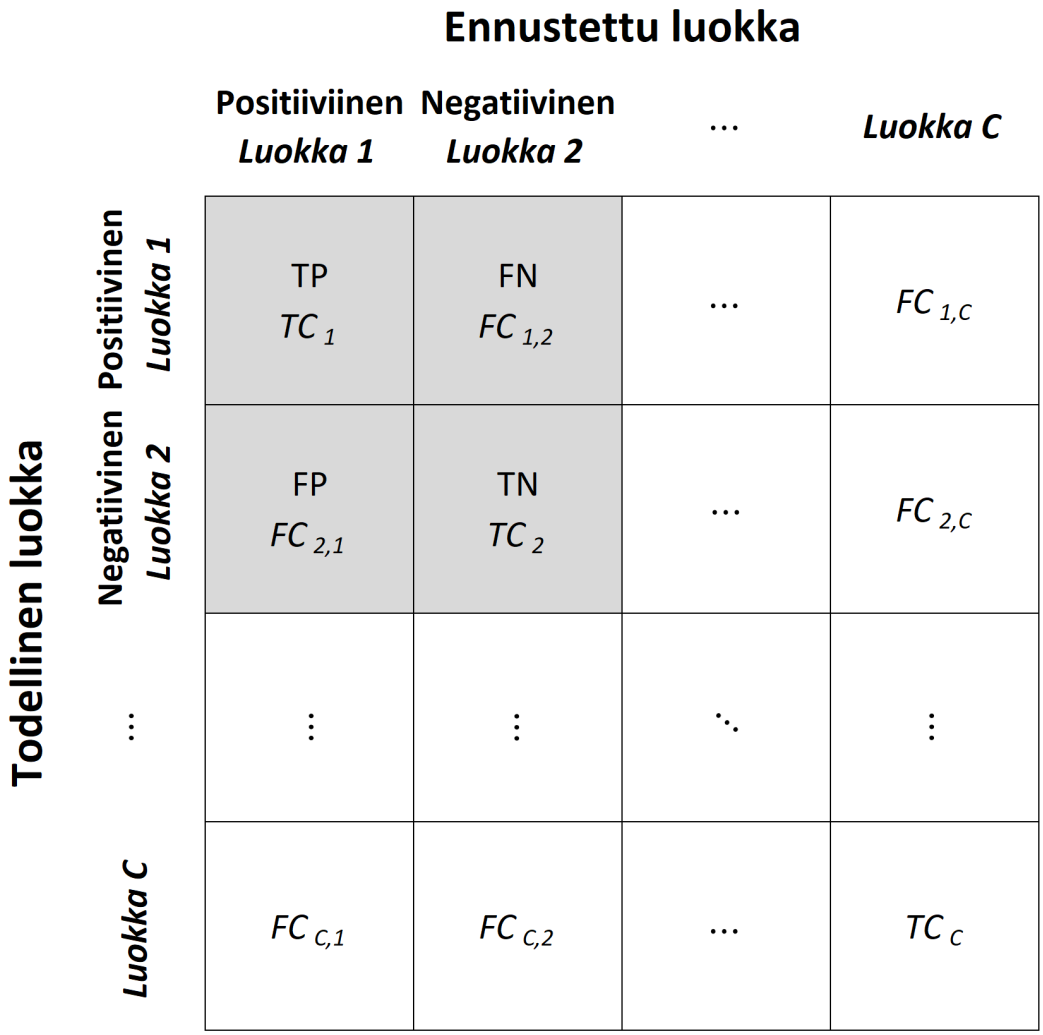
la pohjalla olevat solut ja pystyyn ladottu teksti) sekä yleistettynä useamman luokan ongelmaan (luokkien nimet kursiivilla). Taulukon rivit edustavat todellista luokkaa ja taulukon sarakkeet puolestaan mihin luokkaan menetelmä ennustaa havainnot. Taulukon diagonaalilta löytyy aina kunkin luokan oikein luokiteltujen havaintojen määrä.

2.6.2 Mittarit

Yleisesti käytetty mittari on tarkkuus, joka kertoo oikein luokiteltujen havaintojen osuuden kaikista luokiteltavista havainnoista eli

$$\begin{aligned} \text{Acc} &= \frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{n} \\ &= \frac{TC_1+TC_2+\cdots+TC_C}{n}, \quad n = \text{havaintojen lukumäärä.} \end{aligned} \tag{7}$$

Tarkkuuden lisäksi usein käytetään virheosuutta (*error rate*), joka on $1-\text{Acc}$ eli virheellisesti luokiteltujen osuus kaikista luokiteltavista. Tarkkuus tai virheosuus eivät ole hyviä mittareita silloin, jos aineisto on epätasapainossa eri luokkien välillä. Toisin sanoen jotkin luokista ovat hyvin isoja ja jotkin puolestaan hyvin pieniä. Tällöin pienten luokkien tarkkuus voi olla esimerkiksi



Kuva 5: Esimerkki luokittelutaulukosta (*confusion matrix*) sekä binäärisen, eli kaksiluokkaisen, ongelman tapauksessa (harmaat solut) että yleistettynä useampiluokkaisen ongelman tapaukseen.

lähellä 0 prosenttia, mutta kokonaistarkkuus lähellä 100 prosenttia, jos isojen luokkien tarkkuus on korkea.

Moniluokkaisten ongelmien tapauksessa kuvan 5 luokittelutaulukko voidaan normalisoida jakamalla rivien lukumäärät rivisummilla, eli todellisten luokkien havaintomäärillä, jolloin luokkakohtaiset tarkkuudet löytyvät taulukon diagonaalilta. Luokkakohtainen tarkkuus on

$$\mathrm{Acc}_c = \frac{TC_c}{FC_{c,1} + \cdots + FC_{c,c-1} + FC_{c,c+1} + \cdots + FC_{c,C}}.$$

Tarkkuudesta on olemassa myös tasapainotettu versio (*balanced accuracy*), joka korjaa luokkien kokoeroista johtuvaa vääristymää normaalissa tarkkuudessa. Löning ym. (2019) määrittävät tasapainotetun tarkkuuden luokkakohtaisten tarkkuuksien keskiarvona eli

$$\mathrm{Acc}_{\text{tasapainotettu}} = \frac{1}{C} \sum_{c=1}^{C} \mathrm{Acc}_c, \quad c = 1, \ldots, C,$$

jossa C on luokkien kokonaismäärä. Tällöin jokaista luokkaa kohdellaan samansuuruisena, ja tasapainoisen aineiston tapauksessa tasapainotettu tarkkuus palautuu yhtälön 7 tarkkuudeksi. Tässä tutkielmassa yhtälön 7 tarkkuudesta käytetään myös nimitystä kokonaistarkkuus viittaamaan siihen, että se on koko aineiston tarkkuus sekä normaali tarkkuus erottamaan sen tasapainotetusta tarkkuudesta.

Muita taulukossa 1 esitettyjä mittareita sensitiivisyys, spesifisyys, positiivinen ennustearvo ja F-mitta ei voida suoraan laskea moniluokkaiselle aineistolle, vaan mittarit tulee laskea luokkakohtaisesti. Jos koko aineistolle halutaan yksi yhteinen arvo, tulee se laskea luokkakohtaisista arvoista laskemalla esimerkiksi keskiarvo. Seuraavaksi esitellään laskukaavat näille mittareille käyttämällä kaksiluokkaisen ongelman termistöä. Kaavat, spesifisyyttä lukuun ottamatta, yleistyvät moniluokkaisille ongelmille siten, että TP on luokan oikein luokiteltujen havaintojen lukumäärä, jolle mittari halutaan laskea, FN on luokan rivisumma luokittelutaulukosta, josta on poistettu oikein luokiteltujen määrä, ja FP puolestaan on luokan sarakesumma luokittelutaulukosta, josta on poistettu oikein luokiteltujen määrä.

Sensitiivisyys on oikein positiiviseksi luokiteltujen osuus kaikista positiivisista havainnoista eli

$$\mathrm{Se} = \frac{TP}{TP + FN},$$

ja se kuvaa luokittelijan kyvykkyyttä luokitella positiiviset havainnot oikein. Sensitiivisyydestä käytetään myös suomenkielistä termiä herkkyys tai koneoppimismenetelmien yhteydessä englanninkielistä termiä *recall*. Spesifisyys on puolestaan oikein negatiivisiksi luokiteltujen osuus kaikista negatiivisista havainnoista eli

$$\mathrm{Sp} = \frac{TN}{TN + FP},$$

ja se kuvaa luokittelijan kyvykkyyttä luokitella negatiiviset havainnot oikein.

Positiivisesta ennustearvosta käytetään usein myös englanninkielistä termiä *precision* varsinkin koneoppimismenetelmien yhteydessä. Se on oikein positiiviseksi luokiteltujen osuus kaikista positiiviseksi luokitelluista havainnoista eli

$$\mathrm{PPV} = \frac{TP}{TP + FP},$$

ja se kuvaa – nimensä mukaisesti – luokittelijan kyvykkyyttä ennustaa todelliset positiiviset arvot positiivisiksi.

F-mitta on sensitiivisyyden ja positiivisen ennustearvon painotettu harmoninen keskiarvo ja se lasketaan kaavalla

$$F_\beta = \frac{(\beta^2 + 1) \times \mathrm{PPV} \times \mathrm{Se}}{\beta^2 \times \mathrm{PPV} + \mathrm{Se}}. \tag{8}$$

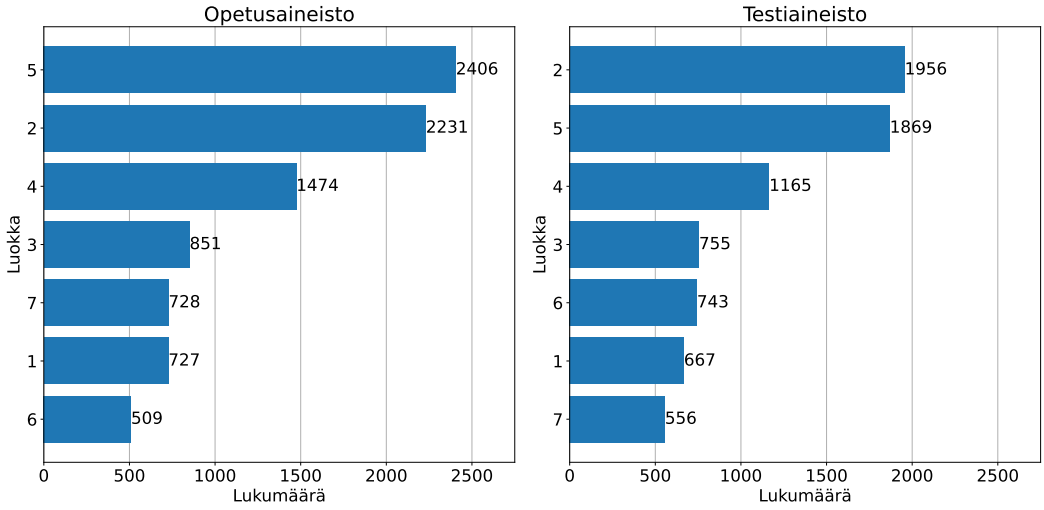
Kun β on suurempaa kuin 1, yhtälö 8 painottaa sensitiivisyyttä ja vastaavasti kun β on pienempää kuin 1, yhtälö 8 painottaa positiivista ennustearvoa. Usein käytetään β:n arvoa 1, koska se on neutraali, eikä laskennan tekevä henkilö välttämättä tiedä kumpi parametreista on tärkeämpi. Tällöin yhtälö 8 supistuu muotoon

$$F_1 = \frac{2\mathrm{PPV} \times \mathrm{Se}}{\mathrm{PPV} + \mathrm{Se}}.$$

3 Aineisto

Tässä tutkielmassa käytetty aineisto on hankittu Itä-Anglian (Norwich, Iso-Britannia) ja Riversiden (Kalifornia, Yhdysvallat) yliopistojen aikasarjojen luokitteluun keskittyvältä internetsivustolta. Sivusto sisältää kaikkiaan 128 erilaista aikasarja-aineistoa, joista suurin osa on yhden muuttujan aineistoja, mutta sivusto sisältää myös noin 30 monimuuttuja-aineistoa. Kaikki aineistot ovat vapaasti ladattavissa sivuston kautta. Sivusto sisältää myös referenssituloksia eri menetelmillä sekä tietoa eri algoritmeista ja menetelmistä. (Bagnall ym.).

Tutkielmassa käytettäväksi aineistoksi valikoitui *ElectricDevices* niminen aineisto, joka sisältää erilaisten sähkölaitteiden sähkönkulutuksesta muodostettuja aikasarjoja. Valintaperusteina olivat aineiston riittävän suuri koko, jotta koneoppimismalleille on riittävästi opetusaineistoa, sekä moniluokkaisuus. Lisäksi aikasarjat ovat kohtuullisen lyhyitä, josta on ajallinen hyöty mallien sovittamisessa ja aineiston luokittelussa. Aineisto on alun perin kerätty Iso-Britannian hallituksen tukemaa tutkimusta *Powering the Nation* varten, jossa on tutkittu ihmisten sähkönkulutuskäyttäytymistä, jotta kotitalouksien hiilijalanjälkeä saataisiin pienennettyä (Bagnall ym.).



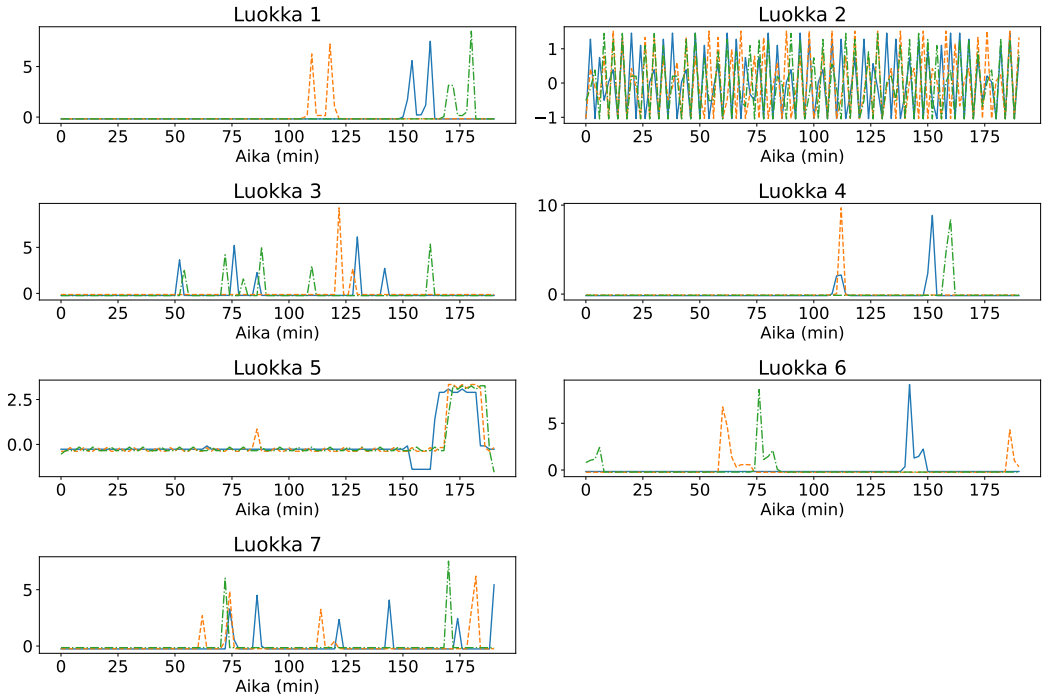
Kuva 6: ElectricDevices-aineiston opetus- ja testiaineistojen yksiulotteiset jakaumat.

Aineistossa on kaikkiaan 16 637 havaintoyksikköä, jotka ovat jaettu valmiiksi opetus- ja testiaineistoihin siten, että opetusaineistossa on 8926 havaintoyksikköä ja testiaineistossa on 7711 havaintoyksikköä. Kunkin havaintoyksikön

eli aikasarjan pituus on 96 havaintoa ja havaintojen väli on 2 minuuttia. Havaintoyksiköt on standardoitu riippumattomasti, eli jokaisen aikasarjan keskiarvo on 0 ja varianssi on 1.

Havaintoyksiköt jakautuvat seitsemään eri luokkaan eli erilaiseen sähkölaitteeseen. Mahdollisia laitteita ovat esimerkiksi vedenkeitin, mikroaaltouuni, astianpesukone ja tietokone. Aineistossa luokat on nimetty vain numeroilla 1–7, eikä oikeita luokkien nimiä ole tiedossa. Opetus- ja testiaineistojen yksiulotteiset jakaumat on esitetty kuvassa 6, joista on havaittavissa, että luokkien välillä on jonkin verran epätasapainoa. Pienimmissäkin luokissa on kuitenkin yli 500 havaintoa, joten opetusaineistolle ei ole tehty keinotekoista tasapainotusta, vaan sitä on käytetty sellaisenaan.

Eri luokkien aikasarjat on havainnollistettu kuvassa 7. Jokaisesta luokasta on piirretty kolme eri aikasarjaa eri väreillä ja viivatyypillä. Luokka 2 näyttäisi poikkeavan muista luokista selkeimmin ollen tasaisesti hyppelevää noin -1 ja 1 välillä. Muissa luokissa on selkeämmin tietyssä kohdassa ilmenevä piikki tai useammassa kohdassa oleva piikki, mutta kuten kuvasta voidaan havaita, useimmiten nämä piikit eivät ole samassa ajankohdassa eri mittauksilla.



Kuva 7: Esimerkit ElectricDevices-aineiston eri luokkien aikasarjoista.

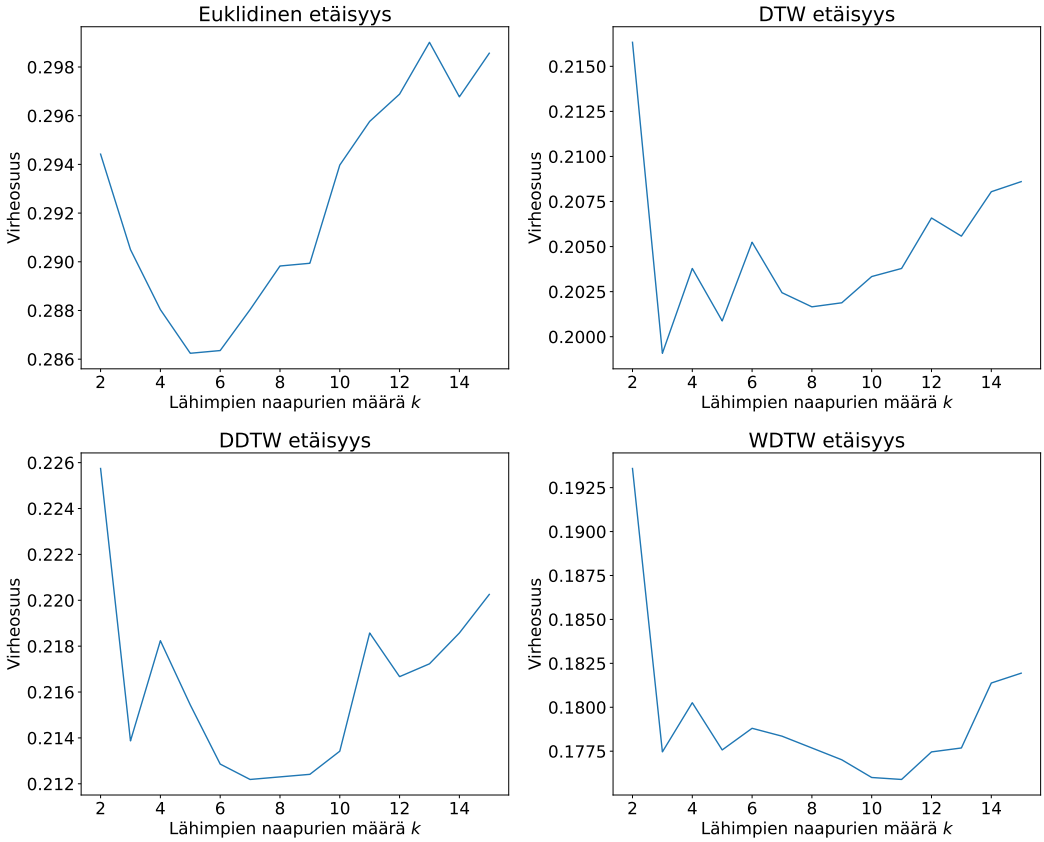
4 Tulokset

Kaikki tämän tutkielman laskelmat toteutettiin Pythonin versiolla 3.9.7 (Python Software Foundation). Kuvaajat piirrettiin *matplotlib*-paketin versiolla 3.6.1 (Hunter, 2007), lukuun ottamatta kuva 5, joka on piirretty Microsoft Excelillä. Aikasarjaluokittelijat *KNeighborsTimeSeriesClassifier* ja *TimeSeriesForestClassifier* sekä shapelet-muunnos *RandomShapeletTransform* ovat *sktime*-paketin Python-versiosta 0.17.2 (Löning ym., 2019). Shapelet-muunnoksen kanssa käytettiin *RandomForestClassifier* satunnaismetsäluokittelijaa, joka on *scikit-learn*-paketin versiosta 1.2.2 (Pedregosa ym., 2011). Laskentaan käytettiin Hetznerin Helsingin konesalista vuokrattua virtuaalikonetta, jossa oli käytössä 8 dedikoitua virtuaalisuoritinta (AMD Milan Epyc 7003), 32 gigatavua työmuistia ja käyttöjärjestelmänä Ubuntu Server 22.04 LTS (Hetzner Online GmbH).

Luokittelijoiden hyperparametrit hienosäädettiin käyttäen *scikit-learn*-paketin version 1.2.2 (Pedregosa ym., 2011) *GridSearchCV* funktiota, jolle annetaan säädettävien parametrien arvot hilana. Funktio suorittaa hilan kaikille parametrikombinaatioille ristiinvalidoinnin halutulla ristiinvalidointistrategialla. Tutkielmassa käytettiin ositettua 5-kertaista ristiinvalidointia, jossa jokaisen luokan suhteelliset osuudet ovat likimain samat, eli jokaisen luokan opetuksessa käytettävä osuus on noin 80 prosenttia ja validointiin käytettävä osuus noin 20 prosenttia. Lisäksi aineisto sekoitettiin ennen jakamista ristiinvalidointia varten, jotta aikasarjat jakautuisivat mahdollisimman satunnaisesti jokaiseen jakoon. Jotta jokaisella luokittelijalla olisi käytössä samalla tavalla sekoitettu ja toistettavissa oleva aineisto, *StratifiedKFold* funktion *random_state* parametriksi, eli satunnaislukugeneraattorin siemenluvuksi, asetettiin 2023. Hilahaussa käytettiin tietokoneen kaikkia suorittimia eri mallien sovittamiseen eri osittteilla, ja itse mallien sovittamisissa ja validointiaineiston luokittelussa käytettiin yhtä suoritinta. Tällä tavoin suoritusajat olivat yhtenevämmät eri osittteiden välillä. Lopulliset mallit sovitettiin käyttäen moniprosessointia tai säikeistettyä laskentaa, jos luokittelijat sitä tukivat.

4.1 K-lähinaapurimenetelmä

Aineiston luokittelu suoritettiin K-lähinaapuriluokittelijalla käyttäen luvussa 2.2 esiteltyjä etäisyysmittoja, ja lähimpien naapurien määrä k haettiin väliltä 2–15. Taulukkoon 2 on koottu hilahakuun sekä ristiinvalidointiin kuluneet ajat. K-lähinaapuriluokittelijan suoritusajat eivät ole riippuvaisia k:n arvosta, joten taulukkoon on laskettu keskiarvot suoritusajoista. Kuvassa 8 on esitetty ristiinvalidoinnissa saadut virheosuudet eri k:n arvoilla sekä eri



Kuva 8: K-lähinaapuriluokittelijan ristiinvalidoinnin virheosuudet eri k:n arvoilla sekä eri etäisyysmitoilla.

etäisyysmitoilla. Lopullinen luokittelu testiaineistolla on tehty käyttäen k:n arvoja, jotka antoivat ristiinvalidoinnissa pienimmät virheosuudet eli suurimmat tarkkuudet. Lopulliset luokittelun tulokset testiaineistolla sekä paras k:n arvo kullekin etäisyysmitalle on koottu taulukkoon 3.

Taulukosta 2 nähdään, että K-lähinaapurimenetelmä on todellakin niin kutsuttu malliton menetelmä, koska mallin sovittamiseen kulunut aika oli noin 6 sekuntia riippumatta etäisyysmitasta tai k:n arvosta. Mallin sovittaminen tarkoittaakin K-lähinaapuriluokittelijan osalta etäisyysmatriisin valmistelemista varsinaista luokittelua varten. Itse luokittelu kesti huomattavasti kauemmin riippuen käytetystä etäisyysmitasta. Euklidinen etäisyys oli ylivoimaisesti nopein etäisyysmitta, koska se on laskettavissa yksinkertaisilla operaatioilla, eikä vaadi minkäänlaista minimin etsimistä kuten dynaaminen aikavääristymä. Parhaan k:n arvon etsintä euklidisella etäisyydellä kesti vain hieman vajaa 3 minuuttia, kun dynaamisella aikavääristymällä aikaa kului

Etäisyysmitta	Keskimääräinen suoritusaika per osite (min)		Kokonais-aika (min)
	Mallin sovitus	Luokittelu	
euklidinen etäisyys	0,09	0,20	2,9
DTW etäisyys	0,09	15,8	143
DDTW etäisyys	0,09	15,2	140
WDTW etäisyys	0,09	17,3	158

Taulukko 2: K-lähinaapuriluokittelijan ristiinvalidoinnin keskimääräiset suoritusajat per osite sekä ristiinvalidointiin kulunut kokonaisaika.

2,3–2,6 tuntia.

Euklidinen etäisyys on siis huomattavan nopea verrattuna muihin etäisyysmittoihin, mutta toisaalta sen tarkkuus oli huonoimmasta päästä, kuten taulukosta 3 voidaan nähdä. Koko testiaineiston luokitteluun kului aikaa vain 47 sekuntia. Dynaaminen aikavääristymä ja sen eri versiot tuottivat pääasiassa paremman tarkkuuden kuin euklidinen etäisyys, mutta samalla testiaineiston luokitteluun kulunut aika nousi yli tuntiin. Parhaiten suoriutunut etäisyysmitta oli dynaamisen aikavääristymän painotettu versio, jonka tarkkuus ylsi 64,4 prosenttiin. Tarkasteltaessa tasapainotettuja tarkkuuksia, voidaan huomata, että euklidisella etäisyydellä ja DDTW etäisyydellä ne olivat käytännössä samat, vaikka DDTW etäisyyden normaali tarkkuus oli hieman heikompi. WDTW etäisyyden tasapainotettu tarkkuus oli selkeästi korkein ollen 5 prosenttiyksikköä korkeampi kuin DTW etäisyydellä, vaikka normaalissa tarkkuudessa eroa oli hieman vähemmän, 3,4 prosenttiyksikköä. Luokkakohtaisista arvoista lasketut F_1-mitat olivat hyvin lähellä tasapainotettuja tarkkuuksia, ollen 0,2–1,3 prosenttia sitä korkeampia.

Etäisyys-mitta	k	Luokittelun suoritusaika (min)	Tarkkuus		F_1-mitta
			Normaali	Tasapainotettu	
euklidinen	5	0,78	0,583	0,491	0,493
DTW	3	65,6	0,610	0,532	0,542
DDTW	7	61,9	0,569	0,490	0,502
WDTW	11	76,0	0,644	0,582	0,595

Taulukko 3: K-lähinaapuriluokittelijan lopulliset lähimpien naapurien määrät k, luokitteluun kuluneet ajat, tarkkuudet sekä F_1-mitta (keskiarvo luokkakohtaisista arvoista).

Kuvassa 9 on esitetty K-lähinaapurimenetelmän testiaineiston luokittelun

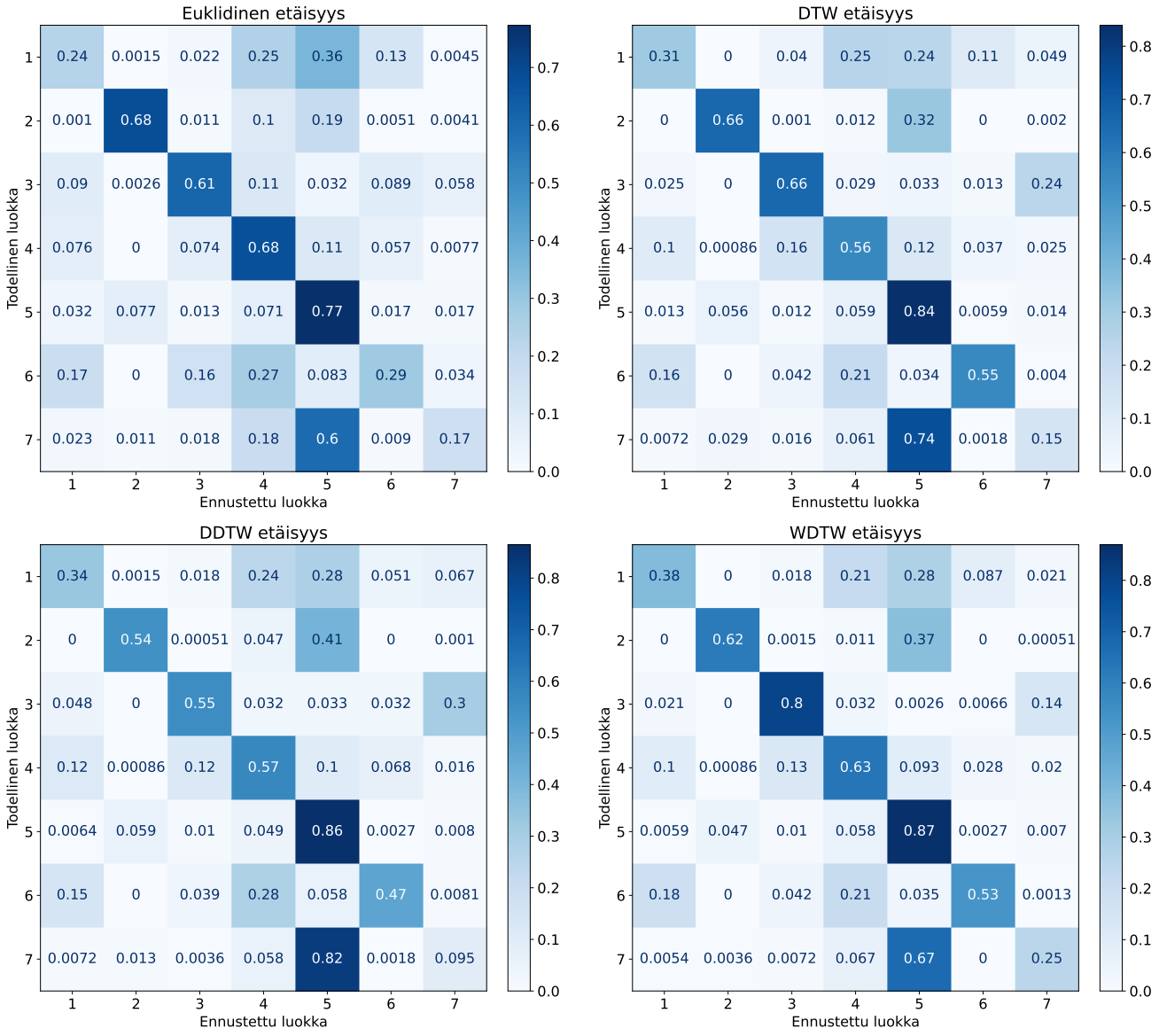
luokittelutaulukot eri etäisyysmitoilla ja parhailla k:n arvoilla. Luokittelutaulukoista nähdään, että varsinkin luokkien 1 ja 7 luokittelu oli heikkoa etäisyysmitasta riippumatta. Esimerkiksi DDTW etäisyyttä käytettäessä luokan 7 tarkkuus oli vain 9,5 prosenttia ja parhaimmillaankin se oli vain 25 prosenttia WDTW etäisyydellä. Luokasta 7 oli jopa 60–82 prosenttia luokiteltu luokkaan 5, vaikka kuvasta 7 silmämääräisesti tarkasteltuna luokkien välillä on melko selkeä ero. Euklidisella etäisyydellä myös luokan 6 tarkkuus oli heikko, vain 29 prosenttia, kun muilla etäisyysmitoilla se oli 50 prosentin paikkeilla. Luokan 5 luokittelu onnistui parhaiten jokaisella menetelmällä ja sen tarkkuus oli 77 prosentista 87 prosenttiin. WDTW etäisyydellä myös luokan 3 luokittelu onnistui kohtalaisen hyvin tarkkuuden ollessa 80 prosenttia. Muiden luokkien tarkkuudet jäivät poikkeuksetta alle 70 prosenttiin jokaisella etäisyysmitalla.

4.2 Aikasarjametsä

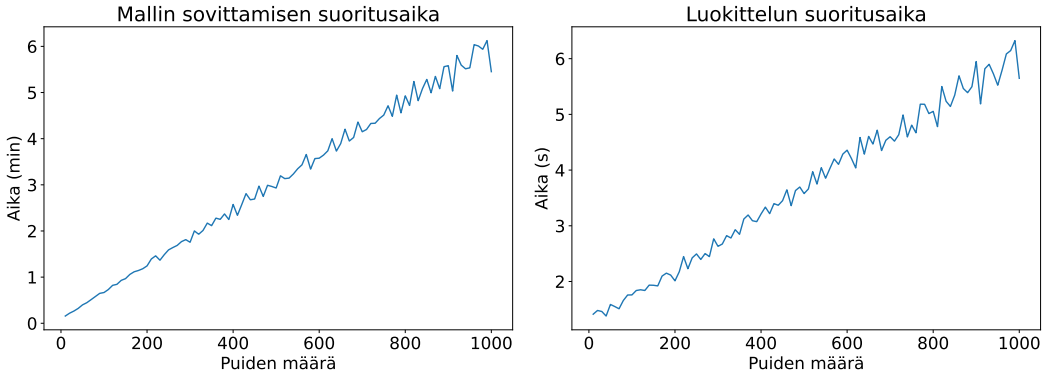
Sktime-paketin aikasarjametsäluokittelija on riisuttu versio *scikit-learn*-paketin satunnaismetsäluokittelijasta. Säädettäviä hyperparametreja on vain kaksi: puiden määrä sekä intervallin minimipituus. Paras arvo puiden määrälle haettiin 10–1000 puun väliltä 10 puun askelissa eli yhteensä 100:sta eri arvosta. Intervallin minimipituus pidettiin oletuksena eli 3 havainnossa. Luokittelija laskee jokaiselle intervallille keskiarvon, keskihajonnan sekä kulmakertoimen, joista muodostetaan uusi aineisto varsinaista luokittelua varten (Löning ym., 2019). Toistettavuuden varmistamiseksi *random_state* parametriksi, eli satunnaislukugeneraattorin siemenluvuksi, asetettiin 202305.

Hilahaku ristiinvalidoinnilla kesti kokonaisuudessaan noin 3,6 tuntia. Aikasarjametsäluokittelijan mallin sovittamiseen sekä aineiston luokitteluun kuluva aika on jokseenkin lineaarisesti riippuvainen mallissa käytettävien puiden määrästä, joten kuvassa 10 on esitetty näihin kuluneet keskimääräiset ajat ristiinvalidoinnissa. Kuvasta nähdään, että pisimmillään mallin sovittaminen kesti noin 6 minuuttia, kun puiden määrä on lähellä tuhatta puuta. Satunnaismetsäluokittelijalle tyypillisesti aineiston luokittelu puolestaan on erittäin nopeaa ja pisimmilläänkin validointiaineiston luokittelu kesti vain hieman yli 6 sekuntia.

Kuvassa 11 vasemmalla on esitetty ristiinvalidoinnin virheosuudet ja oikealla validaatiokäyrä, josta voidaan tulkita mallin ali- tai ylisovittumista. Malli on alisovittunut silloin, kun tarkkuus sekä validointiaineistolla että opetusaineistolla on matala, ja vastaavasti ylisovittunut silloin, kun tarkkuus opetusaineistolla on korkea, mutta validointiaineistolla matala. Tässä tutkielmassa

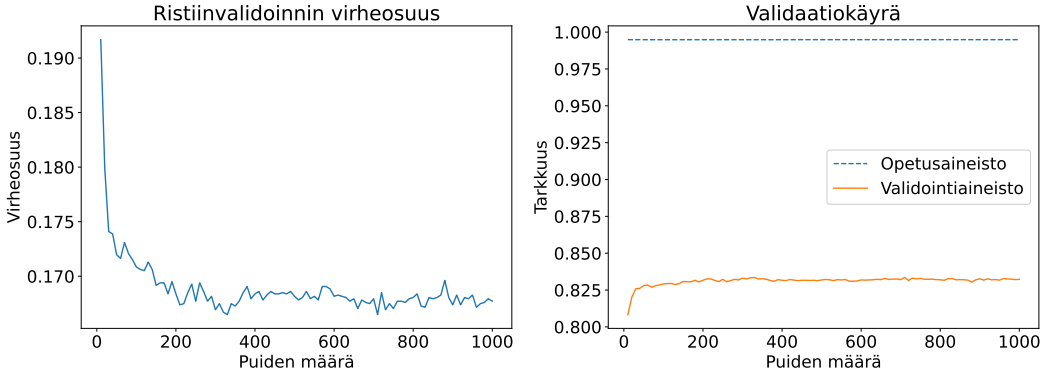


Kuva 9: K-lähinaapuriluokittelijan testiaineiston luokittelutaulukot eri etäisyysmitoilla ja taulukon 3 k:n arvoilla.



Kuva 10: Vasemmalla aikasarjametsäluokittelijan mallin sovittamiseen kulunut keskimääräinen aika eri puiden määrällä per osite. Oikealla validointiaineiston luokitteluun kulunut keskimääräinen aika eri puiden määrällä per osite.

käytetyllä aineistolla validointiaineiston tarkkuus nousee melko nopeasti noin 83 prosentin paikkeille, eikä puumäärään lisääminen enää paranna tarkkuutta. Ei siis ole järkevää käyttää suurta puumäärää, vaikka satunnaismetsäluokittelijat eivät ole herkkiä ylisovittumaan.



Kuva 11: Vasemmalla aikasarjametsäluokittelijan ristiinvalidoinnin virheosuudet eri puiden määrällä. Oikealla aikasarjametsäluokittelijan validaatiokäyrä, josta voidaan tulkita mallin ali- tai ylisovittumista.

Taulukkoon 4 on koottu ristiinvalidoinnin parhaimmalla puumäärällä sovitetun mallin tulokset testiaineistolla. Malli on sovitettu 10:llä eri satunnaislukugeneraattorin siemenluvulla siten, että ensimmäisellä mallilla se oli 202305 ja sitä kasvatettiin 25:llä per malli. Lopuksi tuloksista laskettiin keskiarvot. Aikasarjametsäluokittelija tukee moniprosessointia eli se pystyy hyödyntämään tietokoneen kaikkia suorittimia laskentaan. Mallin sovittamiseen ku-

Puiden	Suoritusaika (s)		Tarkkuus		F_1-
määrä (kpl)	Mallin sovitus	Luokittelu	Normaali	Tasap.	mitta
330	25,0	6,1	0,693	0,589	0,598

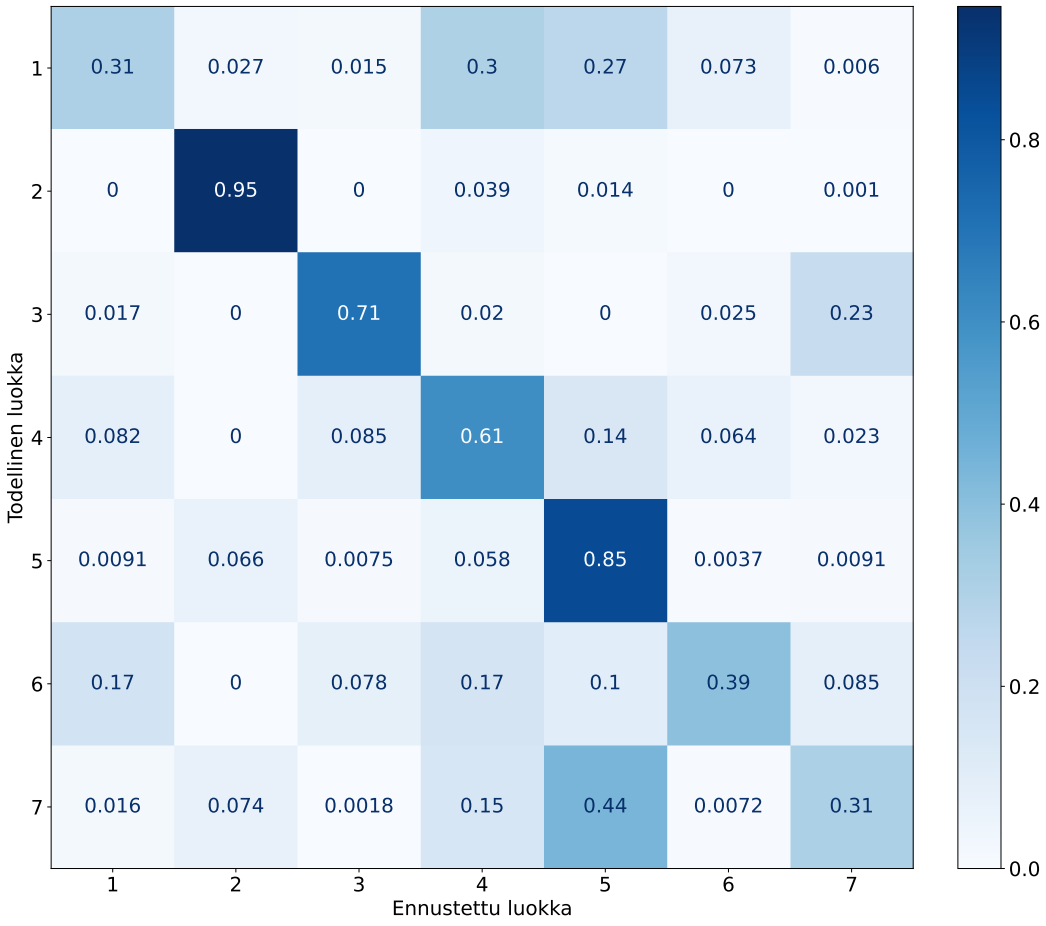
Taulukko 4: Aikasarjametsäluokittelijan lopullisen mallin puiden määrä, mallin sovittamiseen ja testiaineiston luokitteluun kuluneet ajat, luokittelun tarkkuudet, sekä F_1-mitta (keskiarvo luokkakohtaisista arvoista). Tulokset on laskettu 10 eri sovituskerran keskiarvoina, ja keskihajonnat olivat tarkkuuksille ja F_1-mitalle 0,0013–0,0017.

luikin aikaa vain 25 sekuntia, ja koko testiaineistolla luokitteluun kului aikaa noin 6 sekuntia, eli menetelmä on ajallisesti erittäin tehokas. Tarkkuus parani WDTW etäisyyttä käyttävän K-lähinaapuriluokittelijan 64,4 prosentista 69,3 prosenttiin, eli noin 5 prosenttiyksikköä. Toisaalta tasapainotettu tarkkuus ei käytännössä parantunut ollenkaan, vaan oli vain 0,7 prosenttiyksikköä korkeampi. F_1-mitta oli jälleen lähellä tasapainotettua tarkkuutta, ollen 0,9 prosenttiyksikköä sitä korkeampi. Tarkkuuksien ja F_1-mitan keskihajonnat olivat matalat eli eri siemenluvuilla sovitettujen mallien välillä ei ole suuria eroja.

Tarkasteltaessa kuvan 12 luokittelutaulukkoa, voidaan havaita, että aikasarjametsäkään ei pysty parantamaan luokkien 1, 6 ja 7 tarkkuutta, vaan ne olivat edelleen varsin matalat 31–39 prosenttia. Toisaalta aikasarjametsä onnistui parantamaan luokan 2 tarkkuuden 95 prosenttiin, eli se oli 27–41 prosenttiyksikköä korkeampi kuin K-lähinaapurimenetelmällä. Luokkien 3, 4 ja 5 tarkkuudet olivat samalla tasolla K-lähinaapurimenetelmän kanssa. Myös aikasarjametsä luokitteli suuren osan, eli 44 prosenttia, luokan 7 havainnoista luokkaan 5.

4.3 Shapelet-muunnos

Shapelet-muunnos on käytännössä mahdoton hienosäätää pitkien suoritusaikojen takia, joten muunnos suoritettiin aineistolle taulukon 5 parametreilla siten, että shapelet-otoksen kokoa kasvatettiin 200:sta 10 000:een ja shapeletien maksimimäärä oli aina puolet otoksen koosta. Taulukkoon on myös koottu todellinen shapeletien määrä muunnoksen jälkeen ja muunnoksen sovittamiseen opetusaineistoon sekä opetusaineiston muuntamiseen kuluneet ajat. Satunnaislukugeneraattorin siemenlukuna, eli *random_state* parametrina, käytettiin lukua 2023, jotta lopputulokset ovat toistettavissa. Muille shapelet-muunnoksen parametreille käytettiin oletusarvoja, eikä muunnok-



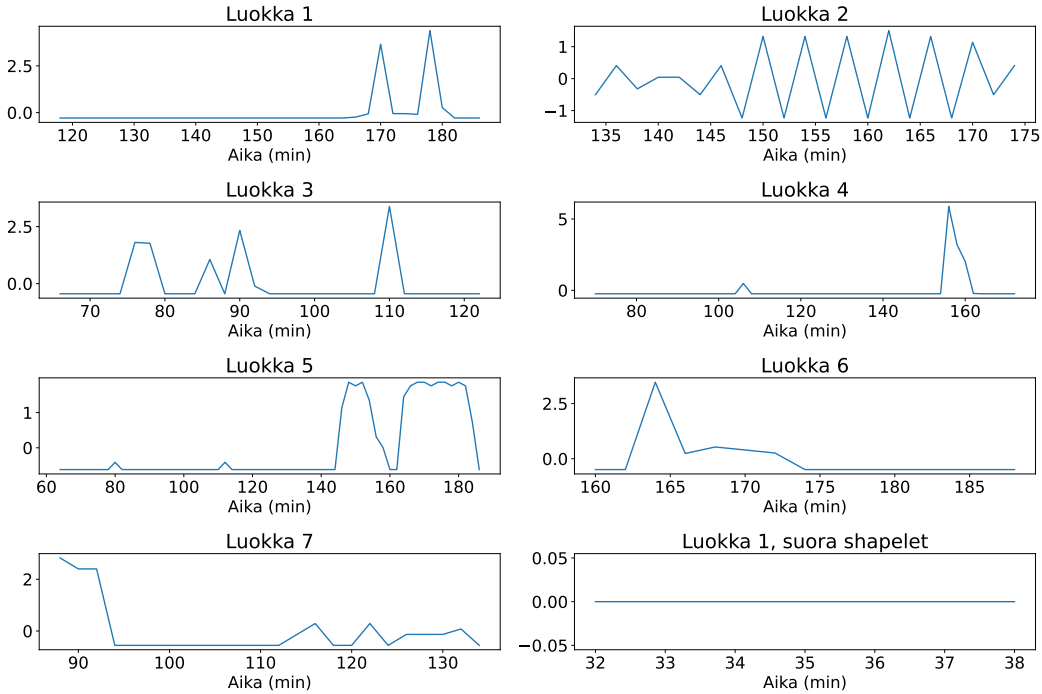
Kuva 12: Aikasarjametsäluokittelijan luokittelutaulukko testiaineistolla taulukon 4 puiden määrällä. Satunnaislukugeneraattorin siemenlukuna käytettiin lukua 202405, jolla luokittelun tarkkuudet ja F_1-mitta olivat lähimpänä keskiarvoja.

	Algoritmin parametrit			Shapeletien	Suoritusajat (min)	
Muunnos #	Shapelet-otoskoko	Shapeletien maksimimäärä	Eräkoko	todellinen määrä	Muunnoksen sovitus	Aineiston muuntaminen
1	200	100	100	19	49,5	0,08
2	400	200	100	49	121,5	0,21
3	600	300	100	114	183,0	0,49
4	800	400	100	154	240,8	0,62
5	1000	500	200	251	328,9	1,0
6	1500	750	200	510	558,0	2,0
7	2000	1000	300	650	686,5	2,6
8	2500	1250	300	971	835,2	3,9
9	3000	1500	400	1141	952,1	4,5
10	3500	1750	400	1283	1069	5,4
11	4000	2000	500	1409	1289	5,9
12	4500	2250	500	1605	1356	6,7
13	5000	2500	500	1798	1515	7,6
14	7500	3750	1000	3023	2014	12,9
15	10000	5000	2000	3595	2632	15,4

Taulukko 5: Shapelet-muunnoksessa käytetyt parametrit (shapelet-otoskoko (*n_shapelet_samples*), shapeletien maksimimäärä (*max_shapelets*) ja eräkoko (*batch_size*)), shapeletien todellinen määrä sovittamisen jälkeen sekä muunnoksen opetusaineistoon sovittamiseen ja opetusaineiston muuntamiseen kuluneet ajat.

seen käytettävää aikaa rajoitettu.

Taulukosta 5 voidaan nähdä, että todellinen shapeletien määrä jäi aina reilusti alle maksimimäärän etenkin pienillä ja suurilla maksimimäärillä. Tämä johtuu siitä, että itsensä kanssa samankaltaiset shapeletit poistetaan muunnoksesta ja toisaalta kaikista luokista ei välttämättä löydy riittävän laadukkaita shapeleteja, jotta niitä otettaisiin muunnokseen mukaan. Muunnokseen sovittamiseen kulunut aika oli lyhimmillään hieman vajaa 50 minuuttia ja pisimmillään lähes 44 tuntia. Opetusaineiston muuntaminen sujui huomattavasti nopeammin ja kesti hitaimmillaankin vain noin 15 minuuttia. Testiaineiston muuntamiseen kuluneita aikoja ei ole taulukkoon 5 kirjattu, mutta ne olivat hieman lyhyempiä, koska testiaineistossa oli noin 1200 havaintoa vähemmän.



Kuva 13: Esimerkit millaisia shapelet-muunnoksen löytämät shapeletit voivat kullekin luokalle olla. Shapeletit ovat muunnoksesta numero 6, lukuun ottamatta luokan 7 shapelet, joka on muunnoksesta numero 8.

Kuvaan 13 on piirretty esimerkit jokaisen luokan shapelet-muunnoksen löytämistä shapeleteista. Kuvasta voidaan havaita, että shapeletit voivat olla hyvin monen mittaisia: esimerkkien shapeletien pituudet ovatkin 4–52 havaintoa. Aineiston aikasarjat sisältävät melko pitkiä tasaisia jaksoja ja monet shapeletit löytyivät hieman yllättäen näistä jaksoista. Kuvan 13 oikeassa alakulmassa on luokasta 1 löytynyt neljän havainnon mittainen suora shapelet, jonka laatu eli informaatiohyöty, oli muunnoksen numero 6 korkein. Se on siis shapelet-muunnoksen näkökulmasta sellainen shapelet, että se erottaa luokan 1 muista luokista parhaiten.

Shapelet-muunnetun aineiston kanssa käytettiin *scikit-learn*-paketin satunnaismetsäluokittelijaa, koska muunnoksen jälkeen aineisto ei ole enää aikasarjamuodossa eikä aikasarjametsäluokittelijaa ole siten tarkoituksenmukaista käyttää. Muunnetussa aineistossa yksi piirre, eli muuttuja, on aina etäisyys yhden löydetyn shapeletin ja aikasarjan välillä. Näin ollen muuttujien järjestyksellä ei ole merkitystä. Satunnaismetsäluokittelijan puiden määrä hienosäädettiin jokaiselle muunnetulle aineistolle hilahaulla 10–500 puun väliltä 10 puun askelissa, kun muille parametreille käytettiin oletusarvoja. Satunnaislu-

kugeneraattorin siemenluvuksi asetettiin 152023, jotta tulokset ovat toistettavissa. Lopuksi tehtiin testiaineiston luokittelu käyttäen hilahaulla löydettyjä satunnaismetsäluokittelijan parhaita puumääriä. Hilahakujen suoritusajat olivat muunnoksen numero 1 noin 2,2 minuutista muunnoksen numero 15 noin 47 minuuttiin.

Taulukkoon 6 on koottu eri muunnoksilla saavutetut tulokset sekä satunnaismetsäluokittelijoiden mallien sovittamiseen että testiaineiston luokitteluun kuluneet ajat. Taulukon tulokset ovat 10 eri mallin sovituksen ja luokittelun tuloksista lasketut keskiarvot siten, että ensimmäisen mallin satunnaislukugeneraattorin siemenluku oli 152023 ja sitä kasvatettiin 25:llä per malli. Satunnaismetsäluokittelija tukee säikeistettyä laskentaa, joten se voi hyödyntää tietokoneen kaikkia suorittimia laskentaan. Mallien sovittamiset sujuivatkin erittäin nopeasti ja suoritusajat olivat reilusta sekunnista vajaaseen puoleen minuuttiin. Luokitteluun kulunut aika oli pisimmilläänkin vain hieman päälle 2 sekunnin kymmenesosaa.

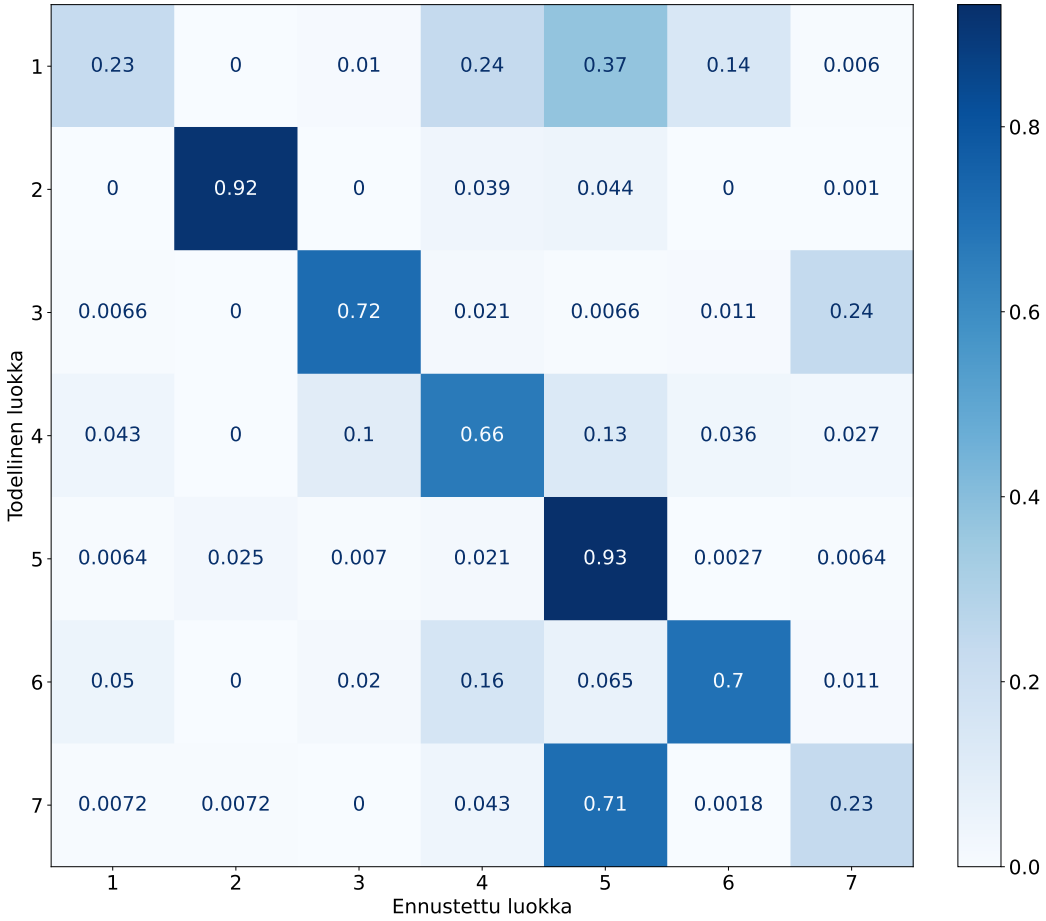
Kahden ensimmäisen muunnoksen tarkkuudet ja F_1-mitta olivat selvästi muita muunnoksia heikommat, mikä ei yllätä, koska ensimmäisen muunnoksen shapeletien määrä oli vain 19 ja toisenkin vain 49. Kolmannen ja neljännen muunnoksen tarkkuudet ja F_1-mitta olivat puolestaan aikasarjametsäluokittelijan kanssa samalla tasolla, ja viidennestä muunnoksesta eteenpäin ne olivat jonkin verran korkeammat. Parhaat tulokset tuotti muunnos numero 12, jonka shapeletien määrä oli 1605. Normaali tarkkuus oli 73,3 prosenttia ja tasapainotettu tarkkuus 62,7, eli kumpikin oli noin 4 prosenttiyksikköä satunnaismetsäluokittelijan tuloksia korkeampia. WDTW etäisyyttä käyttävään K-lähinaapuriluokittelijaan verrattuna normaali tarkkuus oli noin 9 prosenttiyksikkö korkeampi, mutta toisaalta tasapainotetun tarkkuuden ja F_1-mitan erot olivat samalla tasolla kuin satunnaismetsäluokittelijaan verrattaessa. Tarkkuuksien ja F_1-mitan keskihajonnat olivat myös shapelet-muunnoksen kanssa matalat.

Kuvan 14 luokittelutaulukosta nähdään, että luokat 1 ja 7 olivat myös shapelet-muunnoksen kanssa vaikeimmat luokitella ja kummankin luokan tarkkuus oli vain 23 prosenttia. Luokasta 7 oli jopa 71 prosenttia luokiteltu jälleen luokkaan 5. Luokan 2 tarkkuus laski 3 prosenttiyksikköä 92 prosenttiin verrattuna aikasarjametsään, mutta ero ei ole mitenkään merkittävä. Luokan 5 tarkkuus oli käytetyistä menetelmistä korkein ja nousi 93 prosenttiin, eli 8 prosenttiyksikköä aikasarjametsään verrattuna ja 6 prosenttiyksikköä WDTW etäisyyttä käyttävään K-lähinaapurimenetelmään verrattuna. Luokan 6 tarkkuus oli 70 prosenttia ja samalla käytetyistä menetelmistä korkein

Muunnos #	Puiden määrä (kpl)	Suoritusaika (s) Mallin sovitus	Luokittelu	Tarkkuus Normaali	Tasap.	F_1-mitta
1	420	1,3	0,12	0,559	0,478	0,488
2	470	1,8	0,14	0,607	0,514	0,527
3	420	2,3	0,12	0,694	0,590	0,602
4	230	1,6	0,07	0,683	0,574	0,584
5	340	3,3	0,10	0,711	0,610	0,623
6	280	4,6	0,09	0,721	0,621	0,632
7	360	7,0	0,12	0,723	0,623	0,635
8	480	12,2	0,16	0,728	0,624	0,633
9	480	13,3	0,17	0,729	0,624	0,633
10	270	8,0	0,10	0,732	0,627	0,635
11	260	8,2	0,10	0,731	0,626	0,635
12	420	14,7	0,17	**0,733**	**0,627**	**0,636**
13	480	18,0	0,20	0,733	0,626	0,634
14	240	12,2	0,15	0,733	0,625	0,634
15	480	26,5	0,23	0,729	0,623	0,632

Taulukko 6: Shapelet-muunnoksien kanssa käytettyjen satunnaismetsäluokittelijoiden lopullisten mallien puiden määrä, mallien sovittamiseen ja testiaineiston luokitteluun kuluneet ajat, luokittelun tarkkuudet, sekä F_1-mitta (keskiarvo luokkakohtaisista arvoista). Tulokset on laskettu 10 eri sovituskerran keskiarvoina, ja keskihajonnat normaalilla tarkkuudella olivat 0,0013–0,0027, tasapainotetulla tarkkuudella 0,0012–0,0031 ja F_1-mitalla 0,0015–0,0034. Parhaimmat tulokset ovat lihavoituna.

myös tälle luokalle. Luokan 3 tarkkuus oli samalla tasolla aikasarjametsän kanssa ja 8 prosenttiyksikköä matalampi kuin WDTW etäisyyttä käyttävällä K-lähinaapurimenetelmällä. Luokan 4 tarkkuus puolestaan osui aikasarjametsän ja euklidista etäisyyttä käyttävän K-lähinaapurimenetelmän väliin ollen 66 prosenttia.



Kuva 14: Shapelet-muunnoksen parhaiten suoriutuneen muunnoksen (numero 12) ja aikasarjametsäluokittelijan (puiden lukumäärä 420) luokittelutaulukko. Satunnaislukugeneraattorin siemenlukuna käytettiin lukua 202380, jolla luokittelun tarkkuudet ja F_1-mitta olivat lähimpänä keskiarvoja.

5 Pohdinta

Tutkielman tavoitteena oli perehtyä aikasarjojen luokittelun koneoppimismenetelmiin sekä niiden toimivuuteen empiirisellä aineistolla. Menetelmiksi valittiin kolme toisistaan poikkeavaa menetelmää, joista varsinkin K-lähinaapurimenetelmä ja aikasarjametsä (satunnaismetsän muunnos aikasarjoille) ovat hyvin yleisesti käytettyjä menetelmiä. Kolmantena menetelmänä oli shapelet-muunnos, joka muuntaa aineiston siten, että se pyrkii löytämään aikasarjasta paikallisia muotoja, joiden etäisyydet aikasarjojen kanssa muodostavat muunnetun aineiston. Sen kanssa voidaan käyttää mitä tahansa luokittelijaa, esimerkiksi satunnaismetsää.

Empiiriset tulokset erilaisten sähkölaitteiden kulutustietoja sisältävällä aineistolla näyttivät, että tutkielmaan valituilla menetelmillä päästiin parhaimmillaan noin 73 prosentin kokonaistarkkuuteen, ja yksittäisen luokan osalta jopa 95 prosentin tarkkuuteen. Toisaalta huonoimmin ennustettujen luokkien tarkkuudet olivat vain noin 30 prosenttia parhaitenkin suoriutuneella menetelmällä. Tasapainotettu tarkkuus, joka korjaa luokkien kokoeroista johtuvaa harhaa normaalissa tarkkuudessa, oli parhaimmillaan noin 63 prosenttia. Kokonaistarkkuus ei ole kovinkaan korkea, jos tarkoitus olisi ennustaa luotettavasti laitteen tyyppi pelkästä sähkön kulutuksesta useisiin eri laitetyyppeihin. Yksittäisen, selkeästi muista erottuvan laitetyypin luokittelu sen sijaan onnistuisi jo näilläkin menetelmillä hyvin.

Parhaat tarkkuudet saavutettiin shapelet-muunnoksella satunnaismetsäluokittelijan kanssa, mutta ero aikasarjametsän tarkkuuksiin oli melko pieni, noin 4 prosenttiyksikköä. Etenkin kun huomioidaan muunnoksen sovittamiseen kulunut aika, joka oli parhaat tarkkuudet antaneella muunnoksella 22,6 tuntia, ei shapelet-muunnos välttämättä sovi kaikkien ongelmien ratkaisemiseen. Toisaalta muunnoksen sovittamisen jälkeen aineiston muuntaminen on kohtuullisen nopeaa, ja satunnaismetsän mallin sovittaminen sekä aineiston luokittelu on erittäin nopeaa. Aikasarjametsän mallin sovittaminen opetusaineistoon ja testiaineiston luokittelu kesti sen sijaan kokonaisuudessaan vain noin 30 sekuntia. K-lähinaapurimenetelmän luokittelijat eivät luo mallia, jonka avulla uusia havaintoja voitaisiin luokitella. Tämän vuoksi koko opetusaineisto täytyy sopia laskennan suorittavan laitteen muistiin eikä menetelmä käytännössä sovellu puettaviin laitteisiin, kuten älykelloihin tai sormuksiin, joissa on rajoitettu muistimäärä ja laskentateho.

Aineiston luokittelu osoittautui haastavaksi tehtäväksi, eikä yksinkertaisilla luokittelijoilla päästy korkeaan kokonaistarkkuuteen. Bostrom & Bagnall

(2017) ovat kuitenkin päässeet tällä aineistolla shapelet-muunnosta käyttäen peräti 89,5 prosentin tarkkuuteen, kun käytössä on ollut kahdeksasta eri luokittelijasta – K-lähinaapuri-, naiivi Bayes-, C4.5 päätöspuu-, tukivektorikone- (lineaarisella ja kvadraattisella kantafunktioytimellä), satunnaismetsä-, kiertometsä- (*rotation forest*) ja Bayes-verkkoluokittelijasta – koostuva kokonaisuus. Neuroverkoilla Ismail Fawaz ym. (2019) ovat päässeet parhaimmillaan 72,9 prosentin tarkkuuteen ResNet-luokittelijaa käyttäen. Toisaalta huonoiten suoriutuneen neuroverkon, t-LeNet-luokittelijan, tarkkuus on ollut vain 24 prosenttia, eli vain 10 prosenttiyksikköä korkeampi kuin satunnaisesti eri luokkiin jakamalla.

Kuten Bostrom & Bagnall (2017) todistavat, on tutkielmassa käytetyllä aineistolla mahdollista päästä melko korkeaan kokonaistarkkuuteen, kun käytetään useista eri luokittelijoista koostuvaa kokonaisuutta ja aineiston shapelet-muunnosta. Voisi siis olla mielenkiintoista tutkia, saako heikosti ennustettujen luokkien tarkkuutta parannettua vastaavanlaisilla koostemenetelmillä (*ensemble methods*), joissa opetetaan useita erilaisia luokittelijoita, ja havainnon lopullinen luokka määrätään sen mukaan, mihin luokkaan enemmistö luokittelijoista on havainnon ennustanut. Koostemenetelmien heikkoutena on laskennan viemä aika, joten toinen kiinnostava tutkimuskohde voisi olla menetelmien tehostaminen myös ajallisesti.

Tutkielmassa käytetty aineisto oli kerätty suhteellisen harvalla 2 minuutin näytevälillä. Joidenkin laitteiden, esimerkiksi mikroaaltouunin tai vedenkeittimen, käyttötapa saattaa kuitenkin olla sellainen, ettei niiden kulutus ehdi näkyä 2 minuutin aikaikkunassa. Kolmas tulevaisuuden tutkimuskohde voisikin olla tarkemmin mitatun kulutusdatan kerääminen ja analysointi, että voidaanko sen avulla saavuttaa parempia luokittelutuloksia.

Lähdeluettelo

Bagnall, A. & Lines, J. (2014). Technical report CMP-C14-01: An experimental evaluation of nearest neighbour time series classification. *arXiv preprint arXiv:1406.4757*. https://doi.org/10.48550/arXiv.1406.4757

Bagnall, A., Lines, J., Vickers, W. & Keogh, E. (julkaisuaika tuntematon). *The UEA & UCR time series classification repository*. Haettu 1.4.2023 osoitteesta https://www.timeseriesclassification.com

Bostrom, A. & Bagnall, A. (2017). Binary shapelet transform for multiclass time series classification. Teoksessa A. Hameurlain, J. Küng, R. Wagner, S. Madria & T. Hara (toim.), *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXII: Special Issue on Big Data Analytics and Knowledge Discovery* (1st ed.) (s. 24–46). Springer. https://doi.org/10.1007/978-3-662-55608-5

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences, 239*, 142–153. https://doi.org/10.1016/j.ins.2013.02.030

Giorgino, T. (2009). Computing and visualizing dynamic time warping alignments in R: the dtw package. *Journal of Statistical Software, 31*(7), 1–24. https://doi.org/10.18637/jss.v031.i07

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

Hetzner Online GmbH (julkaisuaika tuntematon). *Truly thrifty cloud hosting*. Haettu 3.5.2023 osoitteesta https://www.hetzner.com/cloud

Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2014). Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery, 28*(4), 851–881. https://doi.org/10.1007/s10618-013-0322-1

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. & Muller, P. A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery, 33*(4), 917–963. https://doi.org/10.1007/s10618-019-00619-1

Kubat, M. (2017). *An Introduction to Machine Learning* (Vol. 2). Springer. https://doi.org/10.1007/978-3-030-81935-4

Liang, M., Wang, X., & Wu, S. (2022). Improving stock trend prediction through financial time series classification and temporal correlation analysis based on aligning change point. *Soft Computing, 27*(7), 3655-–3672. https://doi.org/10.1007/s00500-022-07630-7

Lines, J. & Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery, 29*(3), 565–592. https://doi.org/10.1007/s10618-014-0361-2

Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., & Király, F. (2019). Sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*. https://doi.org/10.48550/arXiv.1909.07872

Niinistö, M. (18.7.2020). *Oman liikkumisen ja terveyden seuraaminen on nyt trendikästä – mutta mitä syke, askelmäärä ja uni kertovat voinnistasi? Kysyimme asiantuntijoilta.* Yle Uutiset | yle.fi. https://yle.fi/a/3-11436540

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M. Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Popkostova, Y. (2022). Europe's energy crisis conundrum. *European Union Institute for Security Studies.*

Python Software Foundation (julkaisuaika tuntematon). *About Python™ | Python.org*. Haettu 1.4.2023 osoitteesta https://www.python.org

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing, 415*, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061