



Kandidatutkielma

Tietojenkäsittelytieteen kandiohjelma

Päätöspuut datavirtojen louhinnassa

Toni Rämö

23.12.2022

MATEMAATTIS-LUONNONTIETEELLINEN TIEDEKUNTA
HELSINGIN YLIOPISTO

Yhteystiedot

PL 68 (Pietari Kalmin katu 5)
00014 Helsingin yliopisto

Sähköpostiosoite: info@cs.helsinki.fi
URL: <http://www.cs.helsinki.fi/>

Tiedekunta — Fakultet — Faculty		Koulutusohjelma — Utbildningsprogram — Study programme	
Matemaattis-luonnontieteellinen tiedekunta		Tietojenkäsittelytieteen kandiohjelma	
Tekijä — Författare — Author			
Toni Rämö			
Työn nimi — Arbetets titel — Title			
Päätöspuut datavirtojen louhinnassa			
Ohjaajat — Handledare — Supervisors			
professori Nikolaj Tatti			
Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages	
Kandidutkielma	23.12.2022	24 sivua	
Tiivistelmä — Referat — Abstract			
<p>Dataa tuotetaan yhä kasvavia määriä mitä moninaisemmista lähteistä, kuten verkkoliikenteestä, teollisuuden prosesseista, pankkitoiminnasta, liikenteenohjauksesta, terveydenhoidosta — oikeastaan kaikilta yhteiskunnan toimialoilta, joissa on tietotekniikkaa läsnä. Näistä lähteistä peräisin olevaa, kuten oikeastaan mitä tahansa dataa, jota tuotetaan yli ajan voidaan kutsua datavirroiksi tai niiden otoksiksi. Ne tarjoavat ison potentiaalin tiedonlouhinnalle, siis toiminnalle, jolla pyritään löytämään hyödyllistä tietoa tietomassoista. Toisaalta volyymien kasvaessa se on yhä vaativampaa. Ideaalisti yhtään dataa ei jätettäisi analysoimatta.</p> <p>Tässä työssä tutkitaan, miten datavirtojen louhinta onnistuu luokittelun näkökulmasta tarkastelemalla ja vertailemalla viittä erilaista tarkoitukseen kehitettyä päätöspuualgoritmia. Tärkeimpänä näistä voidaan pitää Hoeffding-puuta, johon neljä muuta perustuvat. Se kykenee kasvattamaan päätöspuuta tehokkaasti asteittain saavuttaen hyviä ennustustarkkuuksia isoilla aineistoilla. Sen toteutusta kutsutaan todella nopeaksi päätöspuuksi (VFDT).</p> <p>Muut tarkastellut algoritmit pyrkivät parantamaan luokittimen toimintaa erilaisista näkökulmista. Konseptimukautuvan VFDT:n tavoitteena on ylläpitää ennustustarkkuus konseptinvaihteluiden ilmetessä eli tilanteissa, joissa datan taustalla olevat jakaumat muuttuvat. Epävarmuutta käsittelevä konseptimukautuva VFDT taas pystyy käsittelemään epävarmaa dataa. Toisaalta tiukka VFDT on suunniteltu minimoimaan tarvittavaa muistia. Hoeffding-ainavalmistus puolestaan pyrkii entistä parempaan tilastolliseen nopeuteen.</p> <p>Algoritmien välinen vertailu osoittaa, että Hoeffding-puu on tiukkaa VFDT:tä lukuunottamatta sekä tila- että aikavaativuudeltaan muita pienempi. Kaikissa tapauksissa tilavaativuus riippuu puun koosta, joten käsiteltyjen näytteiden määrä vaikuttaa tilan tarpeeseen. Laskennallisesta näkökulmasta puun koolla on vaikutusta vaativuuteen vain osassa algoritmeista.</p> <p>ACM Computing Classification System (CCS) Computing methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by classification</p>			
Avainsanat — Nyckelord — Keywords			
algoritmit, päätöspuu, Hoeffding-puu, luokittelu, datavirrat, koneoppiminen, tiedonlouhinta			
Säilytyspaikka — Förvaringsställe — Where deposited			
Helsingin yliopiston kirjasto			
Muita tietoja — övriga uppgifter — Additional information			

Sisällys

1	Johdanto	1
2	Taustatiedot	2
2.1	Päätöspuu	2
2.2	Datavirrat luokittelun kohteena	5
2.3	Datavirtojen luokittelun sovelluksia	7
3	Algoritmien esittely	8
3.1	Hoeffding-puu ja todella nopea päätöspuu	8
3.2	Konseptimukautuva VFDT	11
3.3	Epävarmuutta käsittelevä ja konseptimukautuva VFDT	12
3.4	Tiukka VFDT	13
3.5	Hoeffding-ainavalmis-puu ja äärimmäisen nopea päätöspuu	15
4	Algoritmien vaativuudet	16
4.1	Annetut vaativuudet ja niiden yhtenäistäminen	16
4.1.1	Hoeffding-puu	16
4.1.2	Konseptimukautuva VFDT	17
4.1.3	Epävarmuutta käsittelevä ja konseptimukautuva VFDT	17
4.1.4	Tiukka VFDT	17
4.1.5	Hoeffding-ainavalmis-puu	18
4.2	Vaativuuksien vertailu	18
5	Yhteenveto	21
	Lähteet	22

1 Johdanto

Dataa tuotetaan nykyään alati kasvavia määriä, mitä moninaisemmista lähteistä, kuten verkkoliikenteestä, teollisuuden prosesseista, sosiaalisesta mediasta, erilaisista anturiverkoista, liikenteenvalvonnasta, pankkitoiminnasta ja terveydenhoidosta — oikeastaan kaikilta yhteiskunnan toimialoilta, joissa on tietotekniikkaa läsnä (Bifet et al., 2010; Bahri et al., 2021). Lähteet tuottavat usein moniulotteisia, hajanaisia ja hetkellisiä havaintoja. Tällaista, kuten itse asiassa mitä tahansa dataa, jota tuotetaan yli ajan, voidaan pitää datavirtana tai otoksena siitä.

Datavirtojen määrän ja volyymien kasvaessa organisaatioilla on käytössään yhä suuremmat tietokannat, jotka kasvavat rajoituksetta useilla miljoonilla tietueilla päivittäin (Domingos ja Hulten, 2000). Hyödyllisen tiedon löytäminen näistä tietomassoista, siis tiedonlouhinta, vaatii yhä tehokkaampia menetelmiä. Ihanteellisesti yhtään arvokasta tietoa ei menisi hukkaan. Kaikkea dataa ei kuitenkaan ole käytännöllistä, saati edes mahdollista, tallentaa myöhempää käyttöä varten. Näin ollen louhinnan on tapahduttava nopeammin kuin, mitä dataa syntyy samalla kuitenkin tuottaen luotettavia tuloksia.

Tässä työssä tutkitaan, miten haaste on ratkaistu ennustavan tiedonlouhintamenetelmän, luokittelun näkökulmasta. Datavirtojen luokittelun toteutukset pohjautuvat tyypillisesti vakiintuneeseen päätöspuualgoritmiin, Hoeffding-puuhun (Domingos ja Hulten, 2000), jota työssä tarkastellaan. Tämän lisäksi tutkitaan neljää muuta ominaisuuksiltaan poikkeavaa Hoeffding-puuhun perustuvaa algoritmia. Tutkimuksessa havainnoidaan tarkasteltavien algoritmien tila- ja aikavaativuuksia ja niiden välisiä eroja. Vertailun perusteella odotetusti ominaisuuksiltaan monimutkaisemmat algoritmit ovat laskennallisesti vaativampia. Tämä pätee myös tilan tarpeeseen, joskaan erot tilavaativuuksissa eivät ole niin merkittäviä. Molemmissa tapauksissa vaativuuksien kokoluokkien erot liittyvät tuotettavien päätöspuiden kokoon. Puiden koot taas riippuvat käsiteltyjen näytteiden määrästä.

Työ etenee siten, että aluksi kappaleessa 2 annetaan oleelliset taustatiedot: esitellään päätöspuut (luku 2.1), kerrotaan, mitä datavirtojen luokittelu vaatii (luku 2.2) ja annetaan esimerkkejä datavirtojen louhinnan ja luokittelun sovelluskohteista (luku 2.3). Seuraavaksi kappaleessa 3 esitellään tarkasteluun valitut algoritmit. Tämän jälkeen kappaleessa 4 tarkastellaan algoritmien tila- ja aikavaativuuksia (luku 4.1) ja suoritetaan niiden vertailu (luku 4.2). Lopuksi kappale 5 yhteenvetää tutkimuksen sisällön.

2 Taustatiedot

Tässä kappaleessa annetaan taustatiedot päätöspuihin ja datavirtojen luokitteluun liittyen. Luvussa 2.1 kerrotaan, mikä on päätöspuu, miten se toimii ja mitä asioita sen muodostamiseen liittyy. Seuraavassa luvussa 2.2 kuvataan datavirtojen luonnetta ja, mitä erityispiirteitä niiden luokitteluun liittyy. Lopuksi luvussa 2.3 annetaan esimerkkejä erilaisista datavirtoihin liittyvistä luokittelun sovelluskohteista.

2.1 Päätöspuu







Päätöspuu on vakiintunut ja laajasti tutkittu menetelmä tiedon luokitteluun. Itse asiassa sitä soveltavat algoritmit, kuten C4.5 (kts. Quinlan, 1993), ovat suosituimpien tiedonlouhintatapojen joukossa (Wu et al., 2008). Yhä edelleen päätöspuualgoritmit ovat vertailukelpoisia muiden uudempienkin luokittelumenetelmien rinnalla (Zhang et al., 2017). Vakiintunut asema selittynee menetelmän monilla hyödyllisillä ominaisuuksilla, joihin luokittelee de Villen (2013) mukaan muun muassa joustavuus ja helppokäyttöisyys sekä tulosten laatu ja tulkittavuus. Toisaalta päätöspuille on ominaista niiden ei-toivottu epästabiilisuus: pienikin muutos opetusaineistossa voi muuttaa muodostettavaa päätöspuuta merkittävästi (Li ja Belford, 2002).

Päätöspuu koostuu testi- ja lehtisolmuista, joista ensimmäisiä voidaan kutsua myös päätösolmuiksi tai sisäsolmuiksi (Quinlan, 1993, 1996; Manapragada et al., 2018). Testisolmussa testataan luokiteltavan syötteen tiettyä piirrettä ja päätetään testin tuloksen perusteella, mihin puun haaraan luokittelu etenee. Testattavia piirteitä voi tuki olla yhdellä kertaa enemmänkin (Hernández et al., 2021), mutta tässä työssä keskitytään algoritmeihin, joissa näitä on vain yksi kerrallaan. Testisolmua seuraa aina alipuu eli puun on haarauduttava siitä vähintään kahteen uuteen solmuun. Sen sijaan lehtisolmu päättää aina luokittelun. Se ennustaa siihen liitetyn luokkatunnisteen perusteella, mihin luokkaan syöte kuuluu.















Luokitteluongelma voidaan esittää formaalisti Domingosin ja Hultenin (2000) tavoin seuraavasti:

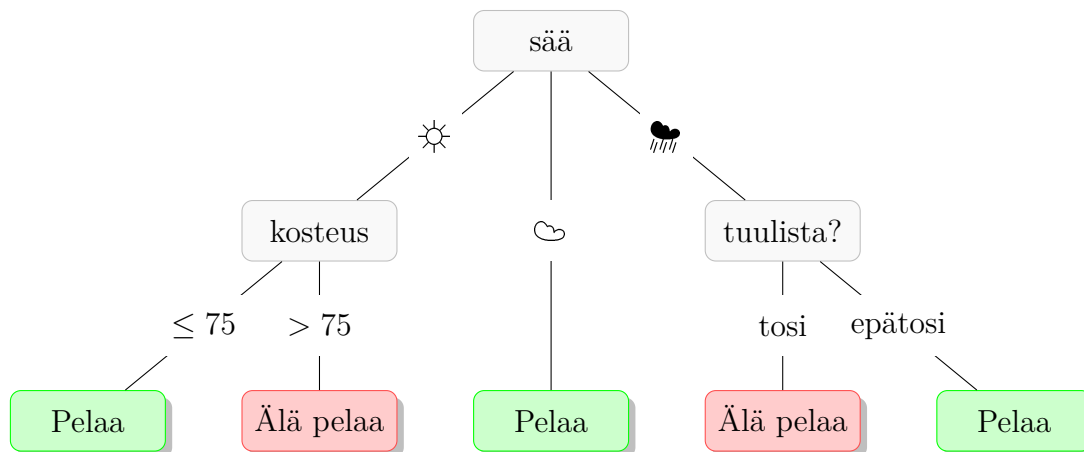
”Annettuna on joukko N näytettä muodossa (\mathbf{x}, y) , jossa y on diskreetti luokkatunniste ja \mathbf{x} on vektori, joka koostuu d symbolisesta tai numeerisesta piir-

teestä. [...] Luokan tunniste $y = DT(\mathbf{x})$ tapaukselle \mathbf{x} saadaan kuljettamalla näytettä puun juuresta lehteen asti testaamalla sopivaa piirrettä kulloisessakin puun solmussa ja seuraamalla piirteen arvoa vastaavaa haaraa.”

Havainnollistetaan päätöspuuhun perustuvaa luokittelua Quinlanin (1986, 1993) laatiman esimerkin avulla. Kuvassa 2.1 on päätöspuu, joka toteuttaa luokittelun taulukon 2.1 opetusaineiston mukaisesti. Sillä voi ennustaa luokkia opetusaineiston mukaisille syötteille, jotka koostuvat arvoista piirteille *sää* (,  tai ), *lämpötila*, *kosteus* ($0 \dots 100\%$) ja *tuulista?* (tosi, epätosi). Käytännössä tämä voisi esittää tilannetta, jossa henkilö pohtii pelaisiko esimerkiksi golfia tietyissä sääolosuhteissa ja mahdolliset päätelmät eli tässä tapauksessa luokat ovat ”pelaa” tai ”älä pelaa”. Puussa on kolme testisolmua *sää*, *kosteus* ja *tuulista?*. Näistä ensimmäinen on myös puun juuri, josta luokittelu lähtee liikkeelle. Kussakin testisolmussa testataan nimenmukaista piirrettä ja kuljetaan piirteen arvoa vastaavaa haaraa seuraavaan solmuun. Siten esimerkiksi, jos *sää* = , kuljetaan solmuun *kosteus*. Luokittelu etenee kunnes päädytään yhteen puun viidestä lehtisolmusta *Pelaa* tai *Älä pelaa*. Näin ollen, jos syöte on vaikkapa muotoa $\mathbf{x} = (\text{, 24^\circ\text{C}, 50\%, \text{epätosi})$, niin $DT(\mathbf{x}) = \text{Pelaa}$. Toisaalta $DT(\text{, 18^\circ\text{C}, 90\%, \text{tosi}) = \text{Älä pelaa}$.

Taulukko 2.1: Quinlan (1993) mukainen opetusaineisto (lämpötila muutettu Celsius-asteeksi)

\mathbf{x}				y
Sää	Lämpötila ($^\circ\text{C}$)	Kosteus (%)	Tuulista?	Luokka
	24	70	tosi	Pelaa
	27	90	tosi	Älä pelaa
	29	85	epätosi	Älä pelaa
	22	95	epätosi	Älä pelaa
	21	70	epätosi	Pelaa
	22	90	tosi	Pelaa
	28	78	epätosi	Pelaa
	18	65	tosi	Pelaa
	27	75	epätosi	Pelaa
	22	80	tosi	Älä pelaa
	18	70	tosi	Älä pelaa
	24	80	epätosi	Pelaa
	20	80	epätosi	Pelaa
	21	96	epätosi	Pelaa



Kuva 2.1: Taulukon 2.1 aineiston mukainen päätöspuu. Mukailtu Quinlanin (1986, 1993) vastaavista esimerkeistä.

Päätöspuun muodostaminen tapahtuu käyttämällä opetusdataa, minkä vuoksi sitä kutsutaan päätöspuuinduktioksi (kuten Quinlan, 1986). Tämä viittaa siihen, että ennustava malli, siis tässä yhteydessä *luokitin*, muodostetaan ennaltaluokiteltujen syötteiden, näytteiden avulla. Toisin sanoen, kyseessä on *ohjattu* koneoppimismenetelmä (Rokach, 2015). Menetelmiä puun muodostamiseen löytyy lukuisia, joista tunnetuimpien joukossa on edellämainittu C4.5. Rokach (2015) lukee näiden joukkoon myös C4.5 edeltäjän ID3:n (Quinlan, 1986) sekä algoritmit CART (Breiman et al., 1984), CHAID (Kass, 1980) ja QUEST (Loh ja Shih, 1997). Näitä voidaan pitää eräajettavina (eng. batch mode) menetelminä, joissa kaikki opetusaineisto oletetaan olevan kerralla saatavilla. Ne eivät sovellu sellaiseenaan datavirtojen luokitteluun luvussa 2.2 määriteltujen ehtojen puitteissa, joten niiden yksityiskohtainen käsittely ei kuulu tähän työhön.

Yleisesti ottaen puun muodostaminen koostuu kasvatus- ja karsintavaiheesta (Rokach, 2015). Yleisin tapa puun kasvattamiseen on ”ylhäältä-alas”-lähestymistapa, jossa puu muodostetaan rekursiivisesti juuresta alkaen ja haarautetaan alipuihin ennalta määritellyn jakokriteerin mukaan. Jakokriteerin avulla päätetään, millä ehdoin ja minkä piirteen perusteella aineisto kannattaa jakaa osajoukkoihin, jotta saadaan paras luokitteluhuoty. Paremmuuden arviointi tehdään jonkin heuristisen hyvyysfunktion kuten C4.5:ssä käytetyn tiedonlisäyksen (eng. information gain) tai CARTista tutun Gini-indeksin avulla (kts. Quinlan, 1993; Breiman et al., 1984). Muodostaminen alipuussa lopetetaan, jos valittu pysähtymiskriteeri täyttyy. Tällöin viimeiseksi solmuksi jää lehtisolmu pysähtymiskriteerin määrittelemällä luokkatunnisteella. Pysähtymiskriteeri voi testata muun muassa sitä, onko alipuuhun valikoituneen aineiston kaikilla näytteillä sama luokka tai onko puun enimmäissyvyys saavutettu (kts. Rokach, 2015, luku 3.6). Yleistävä ja seikkaperäinen esi-

tys päätöspuun muodostamiselle löytyy muun muassa (Hernández et al., 2021) mukaillen (Rokach, 2015) esitettyä mallia.

Tarkastellaan vielä luokitteluesimerkkiä (taulukko 2.1 ja kuva 2.1). Voidaan havaita, että mielenkiintoisesti piirrettä *lämpötila* ei käytetty päätöspuussa lainkaan, vaikka se esiintyy opetusnäytteissä. Yhtä hyvin sekin olisi voinut olla mukana, sillä mikä tahansa muukin opetusaineiston mukaisen luokittelun toteuttava puu olisi mahdollinen. Tämän käyttäminen tosin voisi johtaa paljon monimutkaisempaan puuhun (kts. Quinlan, 1986 kuva 3). Yleensä yksinkertaisempi puu on monimutkaista helpommin tulkittava ja todennäköisesti toteuttaa yleispätevämmän luokittimen (Quinlan, 1986, 1999). Toisaalta liian pienikään puun ei pidä olla, jottei hyödyllistä luokittelutietoja jää käyttämättä ja ennustustarkkuus jää heikoksi (Breiman et al., 1984). Näitä ilmiöitä kutsutaan koneoppimisessa mallin yli- ja alisovittamiseksi (Rokach, 2015). Ylisovittamista voidaan hillitä karsimalla päätöspuuta.

On hyvä huomata, että, kuten Domingosin ja Hultenin (2000) määrittelyssä yllä, päätöspuilla viitataan yleensä menetelmään, joka pyrkii ennustamaan annetulle syötteelle tiettyä luokkaa. Kuitenkin Breiman et al. (1984) osoittaa, että puut soveltuvat myös regressio-ongelmien tutkimiseen. Tällöin luokan tunnisteen y on numeerinen. Puun käyttötarkoituksen erottelemiseksi voidaan näin ollen käyttää nimityksiä luokittelupuun (eng. classification tree) ja regressiopuun (eng. regression tree). Tästä juontaa nimensä myös Breimanin ja muiden päätöspuun menetelmä CART (Classification and Regression Trees), joka soveltuu sekä luokittelu- että regressio-ongelmien analysointiin. Lisää aiheesta ja erilaisista puista on kirjoittanut muun muassa Czapkowski ja Kretowski (2016). Tässä työssä keskitytään luokittelupuihin ja puhuttaessa päätöspuusta viitataan jatkossa niihin.

2.2 Datavirrat luokittelun kohteena

Stefanowski ja Brzezinski (2017) listaavat neljä datavirroille tyypillistä ominaisuutta. Ensimmäkin dataa virtaa jatkuvasti, näyte toisen perään. Toiseksi datamäärät ovat valtavia, jopa äärettömän kokoisia. Kolmanneksi datan virtausnopeus on todella iso suhteessa käytettävissä olevaan prosessointikykyyn. Neljänneksi data on altis muutoksille. Toisin sanoen, aineiston taustalla olevat jakaumat voivat muuttua ajan yli. Tällöin puhutaan konseptinvaihtelusta (eng. concept drift).

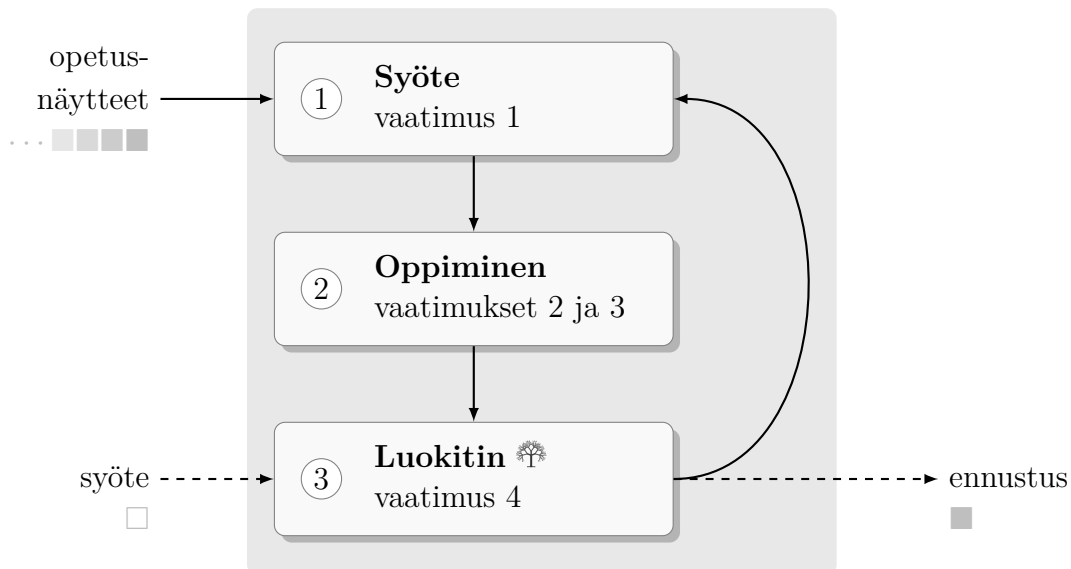
Nämä ominaisuudet asettavat perinteisiin eräajettaviin menetelmiin verrattuna erityiset vaatimukset niin käytettävälle muistille, prosessointiajalle kuin luokittelun mukautumiskyvylle. Haasteina siis on, että muisti riittää mahdollisesti jopa äärettömän datamäärää kä-

siteltäessä, algoritmien tulisi suoriutua nopeammin kuin, mitä näytteitä vastaanotetaan ja toisaalta samalla luokittimen tulisi säilyttää tarkuutensa datan luonteen muuttuessa. Haasteita on toki muitakin ja listaa voisi jatkaa esimerkiksi Bahri et al. (2021) katsauksen tavoin.

Bifet et al. (2010) muotoilee luokitteluympäristöjen, jotka käyttävät datavirtoja, tärkeimmät vaatimukset seuraavasti:

1. kyky käsitellä yksi näyte kerrallaan ja lukea se korkeintaan kerran
2. rajallinen muistin käyttö
3. suoriutuminen rajallisessa ajassa
4. valmius ennustaa milloin tahansa

Lainaten heidän esitystapaansa datavirtoja käyttävä luokittelualgoritmi noudattaa tyypillisesti kuvassa 2.2 esitettyä sykliä. Aluksi (kohta 1) algoritmi saa syötteenä seuraavan saatavilla olevan näytteen vaatimuksen 1 mukaisesti. Seuraavaksi (kohta 2) algoritmi käsittelee näytteen samalla päivittäen tietorakenteensa. Tämä tapahtuu ylittämättä käytettävissä olevaa muistia ja mahdollisimman nopeasti, siten suoriutuen vaatimuksista 2 ja 3. Päivitettään luokittimen (kohta 3) algoritmi on valmis vastaanottamaan seuraavan näytteen (paluu kohtaan 1) ja tarpeen tullen ennustamaan luokkia syötteille, joita ei ole ennen nähty (kohta 3) kuten vaatimuksessa 4 on määritelty.



Kuva 2.2: Datavirran luokittelusykli. Mukailtu Bifet et al. (2010) kuvasta 1.

2.3 Datavirtojen luokittelun sovelluksia

Datavirtojen louhinnan sovelluskohteet voidaan jakaa yleisellä tasolla kolmeen ryhmään (Žliobaitė et al., 2016; Stefanowski ja Brzezinski, 2017):

1. seurantaan ja valvontaan,
2. tiedonhallintaan sekä
3. analytiikkaan ja diagnostiikkaan.

Seuranta ja valvonta liittyy pääasiassa sovelluksiin, joissa pyritään havaitsemaan epänormaaleja tilanteita. Ne voivat liittyä esimerkiksi liikenteenohjaukseen, rahavirtoihin, laadunvalvontaan tai tietoverkkoihin. Tiedonhallintaan lukeutuu muun muassa erilaiset personointiin ja profilointiin liittyvät toimenpiteet kuten tuotesuosittelu, haun personointi, asiakasprofilointi, sähköpostien luokittelu ja suodatus ja rikosten ennustaminen. Analytiikan ja diagnostiikan sovellukset pitävät sisällään erityisesti ennustamiseen liittyviä tehtäviä kuten kysynnän ennustamisen, luottokelpoisuuden arvioinnin ja lääkeresitenssin kehittymisen ennustamisen.

Jokainen ryhmä eroaa niin datan määrältään kuin tyypiltäänkin. Seurannan ja valvonnan kohteissa datan määrät ovat tyypillisesti suurempia kuin muissa ryhmissä ja niissä havainnoitavat muutokset äkillisesti ilmeneviä. Tiedonhallinnan sovelluksissa data on yleensä suhteellista ja muutokset ilmenevät asteittain. Analytiikan ja diagnostiikan tapauksissa käsitellään usein aikasarjoja ja ilmiöt ovat luonteeltaan toistuvia.

Sovelluksia voidaan tilanteesta riippuen tarkastella muun muassa luokittelu-, regressio- tai ryvästämisongelmina. Joskus saatetaan tarvita useampaakin menettelytapaa samaan aiheeseen. Tyypillisesti luokittelua käytetään diagnosointiin ja päätöksentekoon liittyvissä sovelluksissa kuten lääketutkimuksessa, roskapostin tunnistamisessa tai vaikkapa uutisten luokittelussa. Luvussa 3.1 esiteltävän Hoeffding-puun kehittäjät Domingos ja Hulten (2000) soveltavat luokittelualgoritmiaan Washingtonin yliopiston kampukselta peräisin olevien sivupyyntöjen luokitteluun. Tämä mahdollistaa esimerkiksi verkon välimuistituksen optimointia ennustamalla viimeisimpien pyyntöjen perusteella, mitä isäntiä tai sivuja tullaan lähitulevaisuudessa kutsumaan.

3 Algoritmien esittely

Datavirtojen luokitteluun soveltuvia päätöspuualgoritmeja löytyy useita. Ne pohjautuvat pääasiassa luvussa 3.1 esiteltävään Hoeffding-puuhun. Motivaatiot algoritmien takana vaihtelevat ja niihin lukeutuvat muun muassa seuraavat tavoitteet. Tavoitteina voi olla

1. kyky mukautua konseptinvaihteluihin,
2. mallintaa epävarmalla luokitteluaineistolla,
3. minimoida tarvittavan muistin koko tai
4. maksimoida tilastollinen nopeus
(minimoida luokitteluun tarvittavien näytteiden määrä).

Tässä kappaleessa tarkastellaan luokittelualgoritmeja, joiden tavoitteina on ratkaista yksi tai useampi yllä mainituista teemoista. Aluksi luvussa 3.1 esitellään Hoeffding-puu ja tämän toteutus todella nopea päätöspuu (VFDT). Hoeffding-puu käydään läpi muita perusteellisemmin, sillä se toimii perustana muille algoritmeille. Seuraavaksi luvussa 3.2 tarkastellaan konseptimukautuvaa VFDT:tä ja luvussa 3.3 tämän johdannaista algoritmia epävarmuutta käsittelevää ja konseptimukautuvaa VFDT:tä. Tämän jälkeen luvussa 3.4 kerrotaan tiukasta VFDT:stä. Lopuksi luvussa 3.5 esitellään Hoeffding-ainavalmis-puu eli äärimmäisen nopea päätöspuu.

3.1 Hoeffding-puu ja todella nopea päätöspuu

Hoeffding-puuta (eng. Hoeffding tree) tai siihen pohjautuvaa todella nopeaa päätöspuuta (eng. very fast decision tree) voidaan pitää vakiintuneena datavirtojen luokittelumenetelmänä (Manapragada et al., 2018). Sen on kehittänyt Domingos ja Hulten (2000). Toisin kuin klassiset algoritmit kuten C4.5 ja CART, Hoeffding-puu-algoritmi ei edellytä, että muistissa olisi kaikki oppimiseen käytettävät näytteet yhdellä kertaa. Sen sijaan se kasvattaa puuta asteittain mahdollisimman pienellä näytemäärällä kerrallaan samalla kuitenkin pyrkien edellisten tarkkuutteen. Keskeinen idea — josta se on nimensäkin saanut — on hyödyntää Hoeffdingin epäyhtälöä (eng. Hoeffding bound). Sitä soveltamalla voidaan

tietyllä todennäköisyydellä valita lehtisolmujen jakoperusteeksi sama piirre kuin, mitä valittaisiin, jos käytössä olisi ääretön määrä näytteitä. Siten suurilla otoskoilla menetelmän tarkkuus lähestyy eräajettavia menetelmiä.

Hoeffdingin epäyhtälön käyttö voidaan Domingosia ja Hultenia (2000) lainaten määritellä seuraavalla tavalla. Olkoon satunnaismuuttuja r , jonka vaihteluvälin suuruus on R . R riippuu valittavasta heuristisesta hyvyysfunktioista $G(\cdot)$, mikä voi olla esimerkiksi luvussa 2.1 mainittu tiedonlisäys tai Gini-indeksi. Olkoon lisäksi itsenäisiä havaintoja muuttujasta r n kappaletta, ja niiden keskiarvo \bar{r} . Tällöin Hoeffdingin epäyhtälön mukaan todennäköisyydellä $1 - \delta$ muuttujan r todellinen keskiarvo on vähintään $\bar{r} - \epsilon$, jossa

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}. \quad (3.1)$$

Oletetaan, että piirteellä X_a ja n näytteellä saadaan paras keskiarvo \bar{G} ja piirteellä X_b vastaavasti toiseksi paras \bar{G} . Merkitään $\Delta\bar{G} = \bar{G}(X_a) - \bar{G}(X_b)$. Tällöin Hoeffdingin epäyhtälön perusteella todellinen $\Delta G \geq \Delta\bar{G} > \epsilon$ todennäköisyydellä $1 - \delta$. Siten samalla todennäköisyydellä X_a on paras valinta jakoperusteeksi. Käytännössä tätä tietoa sovelletaan siis niin, että puu haarautetaan eli kulloinkin tarkesteltava lehtisolmu muutetaan testisolmuksi piirteen X_a mukaan, jos $\bar{G}(X_a) - \bar{G}(X_b) > \epsilon$. Tämän lisäksi jaon toteuttamisen ehtona on, että $X_a \neq X_\emptyset$, jossa X_\emptyset on ”tyhjä” piirre vastaten tilannetta, jossa jakoa ei tehdä minkään piirteen mukaan. Toisin sanoen, jako tehdään vain, jos sen oletetaan (todennäköisyydellä $1 - \delta$) parantavan luokittelua valitun heuristiikan perusteella. Tätä kutsutaan esikarsinnaksi.

Hoeffding-puun pseudokoodi on esitetty algoritmissa 1. Koodi koostuu kahdesta pääosasta. Aluksi alustetaan päätöspuu (rivit 2-7). Tämän jälkeen voidaan aloittaa oppiminen näyte kerrallaan (riviltä 8 alkaen). Oppiminen etenee siten, että aluksi opetusnäyte luokitellaan kasvatettavaa Hoeffding-puuta käyttäen johonkin sen lehdistä (rivi 9). Tämän jälkeen päivitetään lehteä vastaavat tunnusluvut (rivit 10-11) ja lehden luokkatunniste, jos tarpeen (rivi 12). Sitten tarkistetaan pysähtymiskriteeri eli kuuluvatko kaikki lehteen päätyneet näytteet samaan luokkaan (rivi 13). Jos näin on, poimitaan uusi näyte ja aloitetaan oppimissykli alusta (paluu riville 8). Jos pysähtymiskriteeri ei täyty, jatketaan laskemalla tarvittavat lukuarvot jakokriteerin arviointia varten (rivit 14-17). Seuraavaksi, tarkistetaan jakokriteeri ($\Delta\bar{G} > \epsilon$ ja $X_a \neq X_\emptyset$) ja haarautetaan puu solmun kohdalta jakokriteerin mukaisella piirteellä, mikäli kriteerin ehdot täyttyvät (rivit 18-24).

Kirjallisuudessa saatetaan käyttää todella nopeaa päätöspuuta synonyyminä Hoeffding-puulle, mutta artikkelissaan Domingos ja Hulten (2000) määrittelevät sen pikemminkin

Algoritmi 1 Domingosin ja Hultenin (2000, taulukko 1) mukainen Hoeffding-puu-algoritmi suomennettuna.

Syötteet: S on näytteiden sarja, \mathbf{X} on diskreettien piirteiden joukko, G on hyvyysfunktio ja δ on $1 - P(\text{"valitaan paras piirre tarkasteltavan lehtisolmun jakoperusteeksi"})$.

Tuloste: HT on päätöspuu.

- 1: **Proseduuri** HOEFFDING-PUU(S, \mathbf{X}, G, δ)
 - ▷ Puun alustaminen:
 - 2: Olkoon HT puu, joka koostuu yhdestä lehdestä l_1 (puun juuri).
 - 3: Olkoon $X_1 = \mathbf{X} \cup \{X_\emptyset\}$.
 - 4: Olkoon $\overline{G}_1(X_\emptyset) \overline{G}$, joka saadaan ennustamalla yleisin luokka sarjassa S .
 - 5: **Jokaiselle** luokalle y_k
 - 6: **Jokaisen** luokan $X_i \in \mathbf{X}$ arvolla x_{ij}
 - 7: Olkoon $n_{ijk}(l_1) = 0$.
 - ▷ Oppiminen näyte kerrallaan:
 - 8: **Jokaiselle** näytteelle (\mathbf{x}, y_k) sarjassa S
 - 9: Luokittele näyte lehteen l käyttäen puuta HT .
 - 10: **Jokaisella** x_{ij} vektorissa \mathbf{x} , jolla $X_i \in \mathbf{X}_l$
 - 11: Kasvata $n_{ijk}(l)$.
 - 12: Valitse lehden l tunnisteeksi luokka, josta on eniten havaintoja lehdessä l .
 - 13: **Jos** lehdessä l toistaiseksi havaitut näytteet eivät kuulu samaan luokkaan, **niin**
 - 14: Laske $\overline{G}_l(X_i)$ jokaiselle piirteelle $X_i \in X_l - \{X_\emptyset\}$ tunnusluvulla $n_{ijk}(l)$.
 - 15: Olkoon X_a piirre, jolla saadaan paras \overline{G}_l .
 - 16: Olkoon X_b piirre, jolla saadaan toiseksi paras \overline{G}_l .
 - 17: Laske ϵ kaavalla 3.1.
 - 18: **Jos** $\overline{G}_l(X_a) - \overline{G}_l(X_b) > \epsilon$ ja $X_a \neq X_\emptyset$, **niin**
 - 19: Korvaa lehti l testisolmulla, joka jakaa puun piirteellä X_a .
 - 20: **Jokaiselle** jaon haaralle
 - 21: Lisää uusi lehti l_m ja olkoon $\mathbf{X}_m = \mathbf{X} - \{X_\emptyset\}$.
 - 22: Olkoon $\overline{G}_m(X_\emptyset) \overline{G}$, joka saadaan ennustamalla yleisin luokka l_m :ssä.
 - 23: **Jokaiselle** luokalle y_k ja piirteen $X_i \in \mathbf{X}_m - \{X_\emptyset\}$ arvolle x_{ij}
 - 24: Olkoon $n_{ijk}(l_m) = 0$.
 - 25: Palauta HT .
-

algoritmin parannelluksi toteutukseksi, VFDT-järjestelmäksi. Siinä on mukana lisäominaisuuksia. Sen lisäksi, että käyttäjä voi asettaa δ arvon ja valita käytetäänkö funktiona G tiedonlisäystä vai Gini-indeksiä, mukana on muitakin käyttäjän määriteltävissä olevia parametreja. Muun muassa menetelmä ottaa vastaan syötteenä muuttujan τ arvon, joka sallii jaon, jos $\Delta\overline{G} < \epsilon < \tau$. Tämä nopeuttaa laskentaa jakotilanteissa, joissa on käytännössä tasatilanne kahden eri piirteen välillä. Lisäksi käyttäjä voi asettaa n_{min} , joka vähentää laskentatarvetta viivyttämällä jaon arviointia siihen asti kunnes n_{min} näytettä on vastaanotettu. (Käytännössä tämä pienentää δ arvoa). Menetelmää on täydennetty myös muun muassa muistinhallinnalla ja mahdollisuudella käynnistää algoritmi ennaltaopetetulla puulla.

Hoeffding-puussa ja todella nopeassa päätöspuussa on tiettyjä rajoituksia, mistä johtuen niiden pohjalta on kehitetty muun muassa tässä työssä esitettäviä variantteja. Ensinnäkin niissä oletetaan näytteiden noudattavan muuttumatonta jakaumaa eli ne eivät havaitse mahdollisia konseptinvaihteluita. Tätä varten niistä on jatkokehitetty luvussa 3.2 esiteltävä konseptimukautuva VFDT. Toiseksi Hoeffding-puussa ja todella nopeassa päätöspuussa oletetaan opetusdatan olevan aina tunnettua. Ne eivät siis tue epävarmaa luokittelua luaineistoa toisin kuin luvun 3.3 epävarmuutta käsittelevä ja konseptimukautuva VFDT. Kolmanneksi puut voivat kasvaa induktion aikana merkittävästi vaatien paljon muistia. Tämä saattaa olla haaste sovelluksissa, joissa resursseja on niukasti. Tätä voi rajoittaa tiukentamalla jaon ehtoja kuten tiukkassa VFDT:ssä (luku 3.4). Neljänneksi puun haarauttamista voi entisestään nopeuttaa tilastollisesti kuitenkin heikentämättä olennaisesti luokittimen tarkkuutta kuten luvun 3.5 Hoeffding-ainavalmis-puu osoittaa.

On jopa esitetty kritiikkiä siitä, että Hoeffdingin epäyhtälöä sovellettaisiin väärin, eikä se olisi edes paras mahdollinen menetelmä tarkoitukseen (Rutkowski et al., 2013; De Rosa ja Cesa-Bianchi, 2015; Ortiz Díaz et al., 2020). Toisaalta Ortiz Díaz et al. (2020) mukaan Hoeffding-puu saavuttaa silti käytännössä hyviä ennustustarkkuuksia, sillä se jakaa lehtisolmuja nopeasti ja kasvattaa siten isoja puita. Tällöin mahdolliset mukaan tulevat huonot jaot eivät välttämättä olennaisesti haittaa lopputulosta.

3.2 Konseptimukautuva todella nopea päätöspuu

Kuten luvussa 2.2 todetaan, konseptinvaihtelu on datavirroille ominaista. Tätä esiintyy varsinkin isoissa kuukausien tai jopa vuosien aikana koottavissa tietokannoissa. (Hulten et al., 2001). Tämä siitä syystä, että pitkällä aikavälillä datan syntyyn ja käsittelyyn liit-

tyvät mekanismit voivat vaihdella joskus hyvinkin olennaisesti. Taustalla asiayhteydestä riippuen voivat olla esimerkiksi muuttuvat taloudelliset tekijät tai sääolosuhteet, juhlapyhät, uuden tuotteen esittely tai mainoskamppanja, tietomurto tai vaikkapa huonosti kalibroitu mittalaite. Vastatakseen tähän, Hulten et al. (2001) ovat jatkokehittäneet edellämainitusta Hoeffding-puusta ja tämän toteutuksesta, VFDT:stä, konseptimukautuvan VFDT-luokittelijan (eng. concept-adapting VFDT, CVFDT).

Konseptimukautuva VFDT mukautuu konseptinvaihteluihin ylläpitämällä liukuvaa ikkunaa tuoreimmasta näytteestä alkaen. Tämän ulkopuolelle jäävät tulkitaan liian vanhoiksi ja jätetään huomioimatta — siis unohdetaan. Sen sijaan, että puu tulisi muodostaa alusta jokaisen näytteen kohdalla, luokitin pitää yllä kunkin solmun kohdalla tunnuslukuja, jotka kertovat kuinka moni solmuun päätyneistä näytteistä ovat ajantasaisia ja päivittää vain osia puusta niiden perusteella.

Solmujen jakokriteeri on VFDT:n mukainen sillä erotuksella, että tasatilanteita varten asetettua ehtoa on tiukennettu. Jaon edellytyksenä on, että $(\Delta \overline{G} > \epsilon)$ tai $(\epsilon < \tau$ ja $\Delta \overline{G} \geq \tau/2)$. Algoritmi käy läpi säännöllisesti puun solmuja ja jos se havaitsee, että jokin olemassa olevista testisolmuista ei enää täytä jaon ehtoja, se alkaa kasvattaa vaihtoehtoja alipuuta kyseisestä solmusta. Olemassa olevaa alipuuta ei kuitenkaan korvata heti vaihtoehtoisella puulla, vaan vasta, kun tietyn aikavälein ajettava testi osoittaa uudemman alipuun vanhempaa tarkemmaksi.

3.3 Epävarmuutta käsittelevä ja konseptimukautuva todella nopea päätöspuu

Käytännön sovelluksissa on tyypillistä, että luokitteluun käytettävä aineisto on epävarmaa (Liang et al., 2010). Tämä voi johtua esimerkiksi yhteyskatkoksista tai epätarkoista mitauksista. Joskus epävarmuutta halutaan tietoisesti lisätä aineistoon yksityisyyden suojan vuoksi. Useat datavirtoihin kehitetyt luokittelualgoritmit kuitenkin olettavat opetusdatan olevan varmaa. Liang et al. (2010) esittelevät vaihtoehtoisen lähestymistavan epävarmuutta käsittelevällä ja konseptimukautuvalla VFDT –algoritmillaan (eng. uncertainty handling and concept-adapting VFDT). Tämä voidaan lyhentää epävarmuutta käsitteleväksi CVFDT:ksi.

Epävarmuutta käsittelevän CVFDT:n perustana on kaksi muuta päätöspuualgoritmia: päätöspuu epävarmalle datalle (eng. decision tree for uncertain data) (Qin et al., 2009) ja

edellämainittu konseptimukautuva VFDT. Tässä yhteydessä riittää tarkastella kuinka algoritmi soveltaa ensimmäistä, sillä logiikka noudattelee muutoin jälkimmäisenä mainittua (kts. edeltävä luku 3.2).

Päätöspuu epävarmalle datalle on suunniteltu staattisen datan luokitteluun, mutta sen tapaa käsitellä epävarmaa dataa on sovellettu epävarmuutta käsittelevässä CVFDT:ssä. Idena on, että syötteet käsitellään todennäköisyysvektorein: opetusnäytteet ovat muotoa (X, y) ja tuntemattoman näytteen ennustukset $y = DT(X)$, joissa epävarmojen kategoristen piirteiden joukko $X = \{X_1, X_2, \dots, X_d\}$. Kunkin $X_i \in X$ määrittelyjoukko $\text{dom}(X_i) = \{v_1, v_2, \dots, v_m\}$, jossa X_i noudattaa todennäköisyysjakaumaa. Näin ollen X_i voidaan esittää todennäköisyysvektorina $\mathbf{p} = (p_1, p_2, \dots, p_m)$, jolla todennäköisyys $P(X_i = v_j) = p_j$ ja $\sum_{j=1}^m p_j = 1$.

Toinen olennainen seikka liittyy todennäköisyyksien käsittelyyn. Algoritmissa näytteet jaetaan kunkin testisolmun kohdalla jaon perusteena olevan piirteen mukaan osanäytteiksi. Jokaiselle osanäytteelle annetaan painoarvo, joka määräytyy tarkasteltavan piirteen arvon todennäköisyyden ja edeltäneiden painoarvojen mukaan. Näytteen paino tietyssä solmussa siis käytännössä ilmaisee, mikä on todennäköisyys sille, että näyte päättyy kyseiseen solmuun. Painoja käytetään niin kutsutun todennäköisyyskardinaliteetin (eng. probabilistic cardinality) laskemiseen. Se saadaan laskemalla tarkasteltavan näytejoukon painojen summa tarkasteltavassa solmussa.

Todennäköisyyskardinaliteetti on kyseessä olevan algoritmin kannalta olennainen suure. Sitä käytetään muun muassa tarkasteltavien alkiodien määrän n korvaamiseen Hoeffdingin epäyhtälössä ja syötteen ennusteen päättelemiseen. Suuretta soveltamalla saadaan laskettua myös algoritmin käyttämän hyvyysfunktion arvo. Tässä tapauksessa käytetään epävarmaa tiedonlisäystä (eng. uncertain informatin gain), mikä on aiemmin mainitun tiedonlisäyksen muunnos.

3.4 Tiukka todella nopea päätöspuu

Tiukka VFDT on suunniteltu toimimaan VFDT:tä rajoitetummalla muistin käytöllä (Turrisi da Costa et al., 2018). Tämä saavutetaan hillitsemällä puuta kasvamasta tarpeettoman suureksi tiukennetulla jakokriteerillä. Samalla puun ollessa pienempi saatetaan saada etuja myös lyhyemmän laskenta-ajan muodossa.

Algoritmi perustuu kolmeen oletukseen:

1. Lehtisolmun jako tulee olla mahdollista vain, jos solmun oletettuun luokkaan liittyy riittävän pieni epävarmuus tarkasteluhetken ja aiempien tunnuslukujen perusteella.
2. Kaikilla lehtisolmuilla tulee olla yhtenevä määrä alkioita, jotta ne voidaan jakaa uusiksi solmuiksi.
3. Jakoon käytettävän piirteen on oltava ollut mahdollisimman epärelevantti aiemmissa jaoissa.

Se käyttää seuraavaa funktiota, jota tarvitaan alla esiteltävien ehtojen laskemiseen.

$$\varphi(x, X) = \begin{cases} \text{Tosi,} & \text{jos } x \geq \bar{X} - \sigma(X) \\ \text{Epätosi} & \text{muulloin} \end{cases},$$

jossa X on havaitut arvot, \bar{X} niiden keskiarvo, $\sigma(X)$ varianssi ja x uusi havainto.

Funktiota ja oletuksia soveltamalla, algoritmi noudattaa neljää VFDT:hen nähden jakoa tiukentavaa ehtoa kunkin lehden l kohdalla:

1. $\varphi(H_l, \{H_{l_0}, H_{l_1}, \dots, H_{l_L}\})$, jossa ensimmäinen parametri on lehden l entropia ja jälkimmäinen kaikkien senhetkisten lehtien L entropioiden joukko, l mukaan lukien. (Oletus 1);
2. $\varphi(H_l, \{H_{s_0}, H_{s_1}, \dots, H_{s_S}\})$, jossa jälkimmäinen parametri vastaa kaikkia entropioita, jotka on laskettu S kerralla, kun jokin lehdistä on täyttänyt VFDT:n jakoehdot. (Oletus 1);
3. $\varphi(G_l, \{G_{s_0}, G_{s_1}, \dots, G_{s_S}\})$, jossa G_l on parhaalla jakopiirteellä saatava $G(\cdot)$ ja jälkimmäinen parametri kaikkien niiden $G(\cdot)$ joukko, jotka on laskettu kaikilla S kerralla, kun jokin lehdistä on täyttänyt VFDT:n jakoehdot. (Oletus 3);
4. $n_l \geq \overline{\{n_{s_0}, n_{s_1}, \dots, n_{s_S}\}}$, jossa n_l on lehdessä l nähtyjen näytteiden määrä ja jälkimmäinen parametri on niiden havaittujen näytteiden lukumäärien keskiarvo, joita on ollut lehdillä S kerralla, kun jokin lehdistä on täyttänyt VFDT:n jakoehdot. (Oletus 2).

Algoritmista on itseassa variantti SVFDT-II, joka hyödyntää edellisten lisäksi ohitusmekanismia, joka mahdollistaa puun nopeamman kasvun. Keskitymme tarkastelussa vain versioon ilman ohitusmekanismia.

3.5 Hoeffding-ainavalmis-puu ja äärimmäisen nopea päätöspuu

Hoeffding-ainavalmis-puu pohjautuu Hoeffding-puuhun, mutta se onnistuu kehittäjiensä Manapragada et al. (2018) mukaan nopeuttamaan oppimista (tilastollisesti) huomattavasti vain ”pienellä laskennallisella lisäkustannuksella”. Lisäksi puu sietää luonnostaan konseptinvaihteluita, vaikkei sitä ole siihen varsinaisesti suunniteltukaan. Sen toteutusta kutsutaan äärimmäisen nopeaksi päätöspuuksi (eng. *extremely fast decision tree*) viitaten tämän kykyyn luoda ainavalmiin luokittimen Hoeffding-puun toteutusta, todella nopeaa päätöspuuta vielä pienemmällä näytemäärällä.

Hoeffding-ainavalmis-puun toiminta perustuu Hoeffding-puun tavoin Hoeffdingin epäyhtälön soveltamiseen. Lehtisolmujen jakokriteerinä käytetään edelleen ehtoja $\overline{G}(X_a) - \overline{G}(X_b) > \epsilon$ ja $X_a \neq X_\emptyset$. Tässä tapauksessa X_a on hyvyysfunktion G perusteella keskimäärin paras piirre kuten Hoeffding-puussa, mutta $X_b = X_\emptyset$. Eli vertailua ei tehdä toiseksi parhaaseen piirteeseen vaan ainoastaan tilanteeseen, jossa solmua ei jaettaisi ollenkaan. Tämän ansiosta puu kasvaa Hoeffding-puuta nopeammin ja luokitin on aiemmin käytettävissä ennustamiseen.

Hoeffding-ainavalmis-puu on siis jakovaiheessa Hoeffding-puuta lyhyempi kriteerissään. Tätä kompensoidakseen se käy läpi jo luotuja testisolmuja ja tarvittaessa päivittää puuta huomatessaan, että parempi lopputulos saadaan vaihtamalla testattavaa piirrettä. Tämä toki lisää laskennallista vaativuutta, mutta parantaa puun tarkkuutta. Tästä seuraa myös edellä mainittu luokittimen kyky sietää konseptinvaihteluita.

Mielenkiintoisesti Hoeffding-ainavalmis-puusta on jatkokehitetty viime aikoina yhä uudempiä versioita, mikä viittaa kasvaneeseen kiinnostukseen algoritmia kohtaan. Muunnoksiin lukeutuu muun muassa vihernopeutettu Hoeffding-puu (eng. *green accelerated Hoeffding tree*), tehostettu Hoeffding-ainavalmis-puu (eng. *enchanced Hoeffding tree*) sekä hybridi äärimmäisen nopea päätöspuu (eng. *hybrid extremely fast decision tree*) (Benlarch et al., 2021; Bahloul et al., 2022; Garcia-Martin et al., 2022).

4 Algoritmien vaativuudet

Edellä tarkasteltiin viittä erilaista päätöspuualgoritmia datavirtojen luokitteluun. Neljä jälkimmäistä pohjautuivat ensimmäisenä esiteltyyn Hoeffding-puuhun. Tässä kappaleessa tarkastellaan näiden tila- ja aikavaativuuksia (luku 4.1) ja vertaillaan vaativuuksia keskenään sekä pohditaan lisäominaisuuksien tuomaa hintaa Hoeffding-puuhun verrattuna (luku 4.2). Tässä yhteydessä tilavaativuus vastaa algoritmin ylläpitämiseen vaadittavaa tilaa. Aikavaativuudella taas tarkastellaan tilannetta, jossa luokitin päivitetään datavirrasta poimitulla näytteellä.

4.1 Annetut vaativuudet ja niiden yhtenäistäminen

Tässä luvussa tarkastellaan kappaleessa 3 esitettyjen algoritmien tila- ja aikavaativuuksia. Lähteissä ilmoitettujen vaativuuksien merkintätavat vaihtelevat kirjoittajan mukaan, joten luvussa yhtenäistetään vaativuudet vertailukelpoisiksi.

4.1.1 Hoeffding-puu

Tilavaativuus: Domingos ja Hulten (2000) ilmoittavat Hoeffding-puun tilavaativuudeksi $O(dvc)$ puun lehteä kohden eli yhteensä $O(ldvc)$, jossa l on lehtien määrä puussa tarkasteluhetkellä, d piirteiden määrä, v suurin mahdollisten piirrekohtaisten arvojen määrä ja c luokkien määrä.

Aikavaativuus: Domingos ja Hulten (2000) toteavat Hoeffding-puun aikavaativuuden olevan vakio (pahimmillaan suhteessa piirteiden määrään). Manapragada et al. (2018) ilmaisevat asian tarkemmin ja heidän mukaansa Hoeffding-puun pääoperaatiot eli lehtien tunnuslukujen ylläpitäminen ja jakojen arviointi ovat molemmat vaativuudeltaan $O(dvc)$. Koska nämä ovat sekä peräkkäisiä että vaativimpia operaatioita (kts. Domingos ja Hulten, 2000, taulukko 1), voidaan päätellä aikavaativuuden olevan $O(dvc)$. Huomaa, että käsitelyn alussa tapahtuvaan näytteen luokitteluun kuluva aikaa ei huomioida tässä, sillä sen oletetaan olevan verrattain pieni muihin operaatioihin nähden.

4.1.2 Konseptimukautuva todella nopea päätöspuu

Tilavaativuus: Hulten et al. (2001) päättelivät konseptimukautuvan VFDT:n tilavaativuuden olevan $O(n_c dvc)$, jossa n_c on solmujen määrä yhteensä luokittimen pääpuussa ja tämän vaihtoehtoisissa alipuissa.

Aikavaativuus: Konseptimukautuva VFDT pitää luokitinta ajantasaisena ajassa $O(h_c dvc)$, jossa h_c on pisimmän mahdollisen näytteen kulkeman reitin pituus (eli puun korkeus) kerrottuna vaihtoehtoisten alipuiden lukumäärällä (Hulten et al., 2001).

4.1.3 Epävarmuutta käsittelevä ja konseptimukautuva todella nopea päätöspuu

Tilavaativuus: Liang et al. (2010) mukaan epävarmuutta käsittelevän CVFDT:n tilavaativuus on $O(n_u dvc)$, jossa n_u ilmaisee solmujen määrää yhteensä varsinaisessa puussa ja tämän vaihtoehtoissa alipuissa.

Aikavaativuus: Epävarmuutta käsittelevä CVFDT pitää luokitinta ajantasalla vaativuudella $O(n_u dvc)$ (Liang et al., 2010).

4.1.4 Tiukka todella nopea päätöspuu

Turrise da Costa et al. (2018) ilmoittavat tila- ja aikavaativuudet luvussa 3.4 mainittujen ehtojen 1-4 tasolla, siis mitkä ovat näiden lisäkustannukset todella nopean päätöspuun opimiseen verrattuna. Pseudokoodin perusteella (Turrise da Costa et al., 2018, algoritmit 1 ja 2) tiukka VFDT näyttäisi mukailevan Hoeffding-puuta (Domingos ja Hulten, 2000, taulukko 1) VFDT:n laajennetuin parametrein, joten voidaan soveltaa algoritmille annettuja vaativuuksia luvussa 4.1.1 esitettyjen vaativuuksien kanssa.

Tilavaativuus: Tiukkassa VFDT:ssä ehtojen 2-4 laskemiseen tarvitaan muistia lisää $O(1)$. Sen sijaan ehto 1 kustantaa $O(l_s)$, jossa l_s on puun lehtien enimmäismäärä opimisen aikana. Näin ollen tilavaativuus on $O(l_s dvc + l_s + 3)$ eli $O(l_s dvc)$.

Aikavaativuus: Todella nopeaa päätöspuuta täydentävästä ehdosta 1 koituu kerrallaan $O(l_s)$ kustannus ja ehdoista 2-4 vakiokustannus (Turrise da Costa et al., 2018). Koska ehdot ajetaan peräkkäin, tiukan VFDT:n vaatima aika on enimmillään $O(dvc + l_s + 3)$. Olettaen, että $dvc > l_s$, aikavaativuus tiivistyy muotoon $O(dvc)$.

4.1.5 Hoeffding-ainavalmis-puu

Tilavaativuus: Manapragada et al. (2018) mukaan Hoeffding-ainavalmis-puu vaatii solmujen tunnuslukujen tallentamiseen muistia $O(dvc)$ solmua kohden. Koska solmuja voi olla yhteensä korkeintaan $(1 - v^d)/(1 - v)$, on tilavaativuus korkeintaan $O(v^{d-1}dvc)$. Vaihtoehtoisesti tämä voidaan ilmoittaa muodossa $O(n_h dvc)$, jossa n_h on puun solmujen kokonaismäärä.

Aikavaativuus: Manapragada et al. (2018) ilmoittavat algoritmin aikavaativuuden kahden pääoperaation tarkkuudella: (i) näytteen käsittelyyn liittyvien solmukohtaisten tunnuslukujen päivittäminen ja (ii) jakoarvioinnit lehdessä ja matkalla lehteen kohdattujen testisolmujen kohdalla. Ensimmäinen operaatio vie aikaa $O(dvc)$ solmua kohden eli yhteensä $O(h_h dvc)$, jossa h_h on puun enimmäiskorkeus. Vastaavasti toinen operaatio vie aikaa $O(h_h dvc)$. Koska kyseessä on peräkkäiset operaatiot, voidaan päätellä, että oppiminen näytettä kohden vaatii Hoeffding-ainavalmis-puulla aikaa $O(h_h dvc)$.

4.2 Vaativuuksien vertailu

Taulukkoon 4.1 on koottu edellä esitettyjen algoritmien tila- ja aikavaativuudet. Lisäksi taulukossa on ilmaistu kunkin Hoeffding-puuhun pohjautuvan algoritmin tila- ja aikavaativuuden suhde Hoeffding-puun vaativuuksiin. Tämä ilmaisee lisätyn toiminnallisuuden mahdollisen kustannuksen sekä vaaditun tilan että laskennallisen ajan näkökulmasta. Kuten nähdään, kaikkien algoritmien tila- ja aikavaativuudet riippuvat piirteiden ja niiden mahdollisten arvojen sekä luokkien määrästä. Tämä on siis yhteinen elementti kaikille. Erot taas tulevat siinä, kuinka paljon luokittimien tulee ylläpitää erilaisia tunnuslukuja ja, mitä operaatioita vaaditaan luokittelureittien testisolmuissa ja lehdissä.

Tilavaativuuksien osalta voidaan todeta, että algoritmista riippumatta vaadittava tila on vähintään suhteessa lehtien määrään kuten Hoeffding-puussa ja tiukassa VFDT:ssä. Ylläpidettävien tunnuslukujen vuoksi muut algoritmit tarvitsevat enemmän tilaa vaativuuden ollessa suhteessa kaikkien luokittimien solmujen määrään pelkkien lehtien sijaan. Toisaalta Manapragada et al. (2018) mukaan lehtien määrän tilavaativuus on oikeastaan $O(\text{solmujen määrä})$, joten käytännössä erot ilmenevät siinä kuinka paljon kukin algoritmi tuottaa solmuja. Tämä taas riippuu käsiteltyjen näytteiden määrästä. Sen selvittämiseksi, missä suhteessa puut näytteillä kasvavat, olisi hyödyllistä käyttää myös muita työkaluja kuten kokeellista tutkimusta päättelyn tueksi. Algoritmien kasvumekanismeja

Taulukko 4.1: Algoritmien tila- ja aikavaativuudet sekä niiden suhde Hoeffding-puun vaativuuksiin. Tilavaativuus vastaa koko induktion aikaista vaativuutta, aikavaativuus taas luokittimen päivittämiseen kuluva enimmäisaikaa käsiteltäessä yhtä näytettä.

Algoritmi	Tilavaativuus $O(\cdot)$	Aikavaativuus $O(\cdot)$
Hoeffding-puu	$ldvc$	dvc
Konseptimukautuva VFDT	$n_c dvc$	$h_c dvc$
Epävarmuutta käsittelevä CVFDT	$n_u dvc$	$n_u dvc$
Tiukka VFDT	$l_s dvc$	dvc
Hoeffding-ainavalmis-puu	$n_h dvc$	$h_h dvc$
Suhde Hoeffding-puuhun		
Konseptimukautuva VFDT	n_c/l	h_c
Epävarmuutta käsittelevä CVFDT	n_u/l	n_u
Tiukka VFDT	l_s/l	1
Hoeffding-ainavalmis-puu	n_h/l	h_h

tarkastelemalla voidaan kuitenkin saada erojen suuruusluokasta suuntaa-antavaa tietoa. Esimerkiksi, koska tiukka VFDT noudattaa tarkoituksellisesti tiukempia jakoehtoja, kasvaa puu Hoeffding-puuta todennäköisesti maltillisemmin eli $l_s < l$, jolloin suhde $l_s/l < 1$. Toisaalta, koska sekä konseptimukautuva VFDT että epävarmuutta käsittelevä CVFDT tuottavat myös vaihtoehtoisia alipuita, on todennäköistä, että näiden tilavaativuudet ovat muita suuremmat. Vastaavasti Hoeffding-ainavalmis-puu kasvaa muita nopeammin, joten se voi kasvaa ainakin Hoeffding-puuta ja tiukkaa VFDT:tä solmumäärältään suuremmaksi. Tosin sen puuta päivittävä ominaisuus saattaa ennen pitkää kompensoida tätä.

Aikavaativuuksissa on enemmän vaihtelua. Hoeffding-puun ja tiukan VFDT:n luokittimien päivittämisen aikavaativuudet eivät riipu puun koosta, minkä vuoksi vastaanotettujen näytteiden määrä ei vaikuta näissä olennaisesti oppimiseen kuluvaan aikaan. Käytännössä puun koko toki vaikuttaa vähintäänkin siihen, kuinka pitkän matkan näytteen on kuljettava juuresta lehteen, mutta sen laskennallinen vaikutus on oletettu pienemmäksi muihin operaatioihin nähden. Odotetusti muut, toiminnoiltaan monimutkaisemmat algoritmit ovat laskennallisesti kahta edellämainittua vaativampia, sillä nämä suorittavat toistuvia operaatioita lehtien lisäksi myös testisolmuissa. Erot ovat kertaluokaltaan suhteessa puun kokoon (korkeuteen tai solmujen määrään). Siten näiden aikavaativuudet riippuvat lopulta myös näytteiden määrästä. Tässäkin tapauksessa pelkästään vaativuuksiin vai-

kuttavien muuttujien perusteella ei voi tehdä tyhjentävää vertailua. Ilmeisesti kuitenkin $h_c < n_u$ (Liang et al., 2010), joten epävarmuutta käsittelevää CVFDT on todennäköisesti konseptimukautuvaa VFDT:tä laskennallisesti vaativampi päivittää. Toisaalta, koska Hoeffding-ainavalmis-puu ei muodosta vaihtoehtoisia puita, on mahdollista, että $h_h < h_c$. Tosin, mikäli Hoeffding-ainavalmis-puu kasvaa konseptimukautuvaa VFDT:tä nopeammin eikä datavirrassa esiinny konseptinvaihtelua, voi tilanne olla toinenkin.

Havainnot voidaan tiivistää seuraavasti. Tilavaativuus riippuu kaikilla algoritmeilla piirteiden ja luokkien lisäksi puun koosta ja siten käsiteltyjen näytteiden määrästä. Tosin näiden suhteissa on eroja. Aikavaativuuksissa puun koko on tekijänä muissa paitsi Hoeffding-puussa ja tiukassa VFDT:ssä. Odotettavasti ominaisuuksien lisääminen Hoeffding-puuhun lisää useimmiten algoritmin aikavaativuutta. Myös tilavaativuus on isompi algoritmeissa, joissa laskentaa on lisätty — ellei algoritmin tarkoitus ole ollut nimenomaan rajoittaa tätä. Tila- ja aikavaativuus eivät mittareina kuitenkaan yksinään kerro koko totuutta lisätoimintojen mahdollisista kustannuksista puiden koon ollessa molemmissa tapauksissa algoritmit toisistaan erottava tekijä. Se, kuinka paljon puu kasvaa tietyllä näytemäärällä, riippuu muun muassa opetusaineistosta, algoritmien käyttämistä jakokriteereistä ja tavoista ylläpitää puuta. Näin ollen vertailua voisi täydentää kokeellisella tutkimuksella todellisten suhdelukujen selvittämiseksi. Tätä toki tarjotaan algoritmit esittelevissä julkaisuissa, mutta tulokset eivät ole vertailukelpoisia keskenään, sillä aineistot ja testitavat vaihtelevat tutkimusten välillä.

5 Yhteenveto

Tässä työssä esiteltiin päätöspuut datan luokittimina, kerrottiin, mitä datavirtoihin sovellattavalta luokittimelta vaaditaan ja annettiin esimerkkejä datavirtojen luokittelun sovelluskohteista. Tämän jälkeen tarkasteltiin vakiintunutta ja tehokasta datavirtojen luokittelualgoritmia, Hoeffding-puuta ja neljää muuta tähän pohjautuvaa algoritmia: konseptimukautuvaa VFDT:tä, epävarmuutta käsittelevää CVFDT:tä, tiukkaa VFDT:tä ja Hoeffding-ainavalmis-puuta. Jokainen valittu Hoeffding-puun muunnos pyrkii ratkaisemaan jonkin osaongelman ja siten parantamaan luokittimen toimintaa tietyissä tilanteissa. Ongelmat liittyvät konseptinvaihteluihin, epävarmaan dataan, käytettävään muistiin ja tilastolliseen nopeuteen. Lopuksi algoritmien tila- ja aikavaativuuksia vertailtiin keskenään ja pohdittiin lisäominaisuuksien kustannuksia yksinkertaisempaan Hoeffding-puuhun verrattuna.

Vertailussa havaittiin, että algoritmien tilavaativuuksiin vaikuttaa piirteiden ja luokkien lisäksi puun koko eri suhtein riippuen ylläpidettävistä tunnusluvuista ja muista mahdollisista tiedoista. Puun koko taas riippuu käsiteltyjen näytteiden määrästä, joten kaikissa tapauksissa voidaan olettaa näytteiden määrän vaikuttavan tilan tarpeeseen. Se, missä suhteessa näytteet puita kasvattavat eri algoritmeilla liittyy kunkin algoritmin jakokriteeriin ja mahdollisiin menetelmiin päivittää ja karsia puuta.

Aikavaativuuksissa havaittiin enemmän eroja ja monimutkaisemmat algoritmit olivat odotetusti laskennallisesti vaativimpia. Myös näissä erot liittyivät puun kokoon. Toisin kuin muissa, yksinkertaisemmissa Hoeffding-puussa ja tiukassa VFDT:ssä puun koolla ja siten näytteiden määrällä ei todettu olevan olennaista vaikutusta laskenta-aikaan aikavaativuuksiin liittyvien havaintojen perusteella.

Lisäksi huomattiin, että vaativuuksien suhdetta ja suuruusjärjestystä on haastavaa päätellä tyhjentävästi pelkästään käytetyin teoreettisin keinoin. Siksi mahdolliset jatkotutkimukset voisivat keskittyä tila- ja aikavaativuuksien vertailuun kokeellisin menetelmin. Näissä voisi selvittää, miten vaativuuksien suhteet käyttäytyvät erilaisilla opetusaineistoilla. Tarkastelua voisi laajentaa muihinkin Hoeffding-puun muunnoksiin. Lisäksi algoritmeja olisi mielenkiintoista vertailla myös tarkkuuden näkökulmasta, mitä ei tässä työssä otettu huomioon.

Lähteet

- Bahloul, S. N., Abderrahim, O., Amar, A. I. B. ja Bouhedadja, M. Y. (2022). "Improvement of Data Stream Decision Trees". *International Journal of Data Warehousing and Mining* 18.1, s. 1–17. DOI: <http://doi.org/10.4018/IJDWM.290889>.
- Bahri, M., Bifet, A., Gama, J., Gomes, H. M. ja Maniu, S. (2021). "Data stream analysis: Foundations, major tasks and tools". *WIREs Data Mining and Knowledge Discovery* 11.3, e1405. DOI: <https://doi.org/10.1002/widm.1405>. eprint: <https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1405>. URL: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1405>.
- Benllarch, M., Benhaddi, M. ja El Hadaï, S. (2021). "Enhanced Hoeffding Anytime Tree: A Real-time Algorithm for Early Prediction of Heart Disease". *International Journal on Artificial Intelligence Tools* 30.03. DOI: [10.1142/S021821302150010X](https://doi.org/10.1142/S021821302150010X). eprint: <https://doi.org/10.1142/S021821302150010X>. URL: <https://doi.org/10.1142/S021821302150010X>.
- Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T. ja Seidl, T. (2010). "MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering". Teoksessa: *Proceedings of the First Workshop on Applications of Pattern Analysis*. Toim. T. Diethe, N. Cristianini ja J. Shawe-Taylor. Vol. 11. Proceedings of Machine Learning Research. Cumberland Lodge, Windsor, UK: PMLR, s. 44–50. URL: <https://proceedings.mlr.press/v11/bifet10a.html>.
- Breiman, L., Friedman, J. H., Olshen, R. A. ja Stone, C. J. (1984). *Classification and regression trees*. eng. Belmont (CA): Wadsworth. ISBN: 0-534-98054-6.
- Czajkowski, M. ja Kretowski, M. (2016). "The role of decision tree representation in regression problems—An evolutionary perspective". *Applied soft computing* 48, s. 458–475.
- De Rosa, R. ja Cesa-Bianchi, N. (2015). "Splitting with confidence in decision trees with application to stream mining". Teoksessa: *2015 International Joint Conference on Neural Networks (IJCNN)*, s. 1–8. DOI: [10.1109/IJCNN.2015.7280392](https://doi.org/10.1109/IJCNN.2015.7280392).
- De Ville, B. (2013). "Decision trees". *Wiley Interdisciplinary Reviews: Computational Statistics* 5.6, s. 448–455.
- Domingos, P. ja Hulten, G. (2000). "Mining High-Speed Data Streams". Teoksessa: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '00. Boston, Massachusetts, USA: Association for Computing Mac-

- hinery, s. 71–80. ISBN: 1581132336. DOI: [10.1145/347090.347107](https://doi.org/10.1145/347090.347107). URL: <https://doi-org.libproxy.helsinki.fi/10.1145/347090.347107>.
- Garcia-Martin, E., Bifet, A., Lavesson, N., König, R. ja Linusson, H. (2022). *Green Accelerated Hoeffding Tree*. DOI: [10.48550/ARXIV.2205.03184](https://arxiv.org/abs/2205.03184). URL: <https://arxiv.org/abs/2205.03184>.
- Hernández, V. A. S., Monroy, R., Medina-Pérez, M. A., Loyola-González, O. ja Herrera, F. (2021). "A Practical Tutorial for Decision Tree Induction: Evaluation Measures for Candidate Splits and Opportunities". eng. *ACM computing surveys* 54.1, s. 1–38. ISSN: 0360-0300.
- Hulten, G., Spencer, L. ja Domingos, P. (2001). "Mining Time-Changing Data Streams". Teoksessa: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '01. San Francisco, California: Association for Computing Machinery, s. 97–106. ISBN: 158113391X. DOI: [10.1145/502512.502529](https://doi-org.libproxy.helsinki.fi/10.1145/502512.502529). URL: <https://doi-org.libproxy.helsinki.fi/10.1145/502512.502529>.
- Kass, G. V. (1980). "An Exploratory Technique for Investigating Large Quantities of Categorical Data". eng. *Applied Statistics* 29.2, s. 119–127. ISSN: 0035-9254.
- Li, R.-H. ja Belford, G. G. (2002). "Instability of Decision Tree Classification Algorithms". Teoksessa: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '02. Edmonton, Alberta, Canada: Association for Computing Machinery, s. 570–575. ISBN: 158113567X. DOI: [10.1145/775047.775131](https://doi.org/10.1145/775047.775131). URL: <https://doi.org/10.1145/775047.775131>.
- Liang, C., Zhang, Y. ja Song, Q. (2010). "Decision Tree for Dynamic and Uncertain Data Streams". Teoksessa: *Proceedings of 2nd Asian Conference on Machine Learning*. Toim. M. Sugiyama ja Q. Yang. Vol. 13. Proceedings of Machine Learning Research. Tokyo, Japan: PMLR, s. 209–224. URL: <https://proceedings.mlr.press/v13/liang10a.html>.
- Loh, W.-Y. ja Shih, Y.-S. (1997). "Split selection methods for classification trees". *Statistica Sinica* 4, s. 815–840. ISSN: 10170405, 19968507.
- Manapragada, C., Webb, G. I. ja Salehi, M. (2018). "Extremely Fast Decision Tree". Teoksessa: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery, s. 1953–1962. ISBN: 9781450355520. DOI: [10.1145/3219819.3220005](https://doi.org/10.1145/3219819.3220005). URL: <https://doi.org/10.1145/3219819.3220005>.
- Ortiz Díaz, A. A., Frías Blanco, I. I., Palomino Mariño, L. M. ja Baldo, F. (2020). "An Online Tree-Based Approach for Mining Non-Stationary High-Speed Data Streams".

- Revista de Informática Teórica e Aplicada* 27.1, s. 36–47. DOI: [10.22456/2175-2745.90822](https://doi.org/10.22456/2175-2745.90822).
- Qin, B., Xia, Y. ja Li, F. (2009). "DTU: A Decision Tree for Uncertain Data". Teoksessa: s. 4–15. ISBN: 978-3-642-01306-5. DOI: [10.1007/978-3-642-01307-2_4](https://doi.org/10.1007/978-3-642-01307-2_4).
- Quinlan, J. R. (1986). "Induction of decision trees". *Machine learning* 1.1, s. 81–106.
- (1993). *C4.5 : programs for machine learning*. eng. San Mateo (CA): Morgan Kaufmann. ISBN: 1-55860-238-0.
- (1996). "Learning decision tree classifiers". eng. *ACM computing surveys* 28.1, s. 71–72. ISSN: 0360-0300.
- (1999). "Simplifying decision trees". eng. *International journal of human-computer studies* 51.2, s. 497–510. ISSN: 1071-5819.
- Rokach, L. (2015). *Data mining with decision trees : theory and applications*. eng. 2nd edition. Series in Machine Perception and Artificial Intelligence ; vol. 81. Singapore ; World Scientific Pub. Co. ISBN: 9789814590082.
- Rutkowski, L., Pietruczuk, L., Duda, P. ja Jaworski, M. (2013). "Decision Trees for Mining Data Streams Based on the McDiarmid's Bound". *IEEE Transactions on Knowledge and Data Engineering* 25.6, s. 1272–1279. DOI: [10.1109/TKDE.2012.66](https://doi.org/10.1109/TKDE.2012.66).
- Stefanowski, J. ja Brzezinski, D. (2017). "Stream Classification". Teoksessa: *Encyclopedia of Machine Learning and Data Mining*. Toim. C. Sammut ja G. I. Webb. Boston, MA: Springer US, s. 1191–1199. ISBN: 978-1-4899-7687-1. DOI: [10.1007/978-1-4899-7687-1_908](https://doi.org/10.1007/978-1-4899-7687-1_908). URL: https://doi.org/10.1007/978-1-4899-7687-1_908.
- Turrisi da Costa, V., Carvalho, A. de ja Barbon Junior, S. (2018). "Strict Very Fast Decision Tree: A memory conservative algorithm for data stream mining". *Pattern Recognition Letters*. DOI: [10.1016/j.patrec.2018.09.004](https://doi.org/10.1016/j.patrec.2018.09.004).
- Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S. et al. (2008). "Top 10 algorithms in data mining". *Knowledge and information systems* 14.1, s. 1–37.
- Zhang, C., Liu, C., Zhang, X. ja Alpanidis, G. (2017). "An up-to-date comparison of state-of-the-art classification algorithms". *Expert Systems with Applications* 82, s. 128–150.
- Žliobaitė, I., Pechenizkiy, M. ja Gama, J. (2016). "An Overview of Concept Drift Applications". Teoksessa: *Big Data Analysis: New Algorithms for a New Society*. Toim. N. Japkowicz ja J. Stefanowski. Cham: Springer International Publishing, s. 91–114. ISBN: 978-3-319-26989-4. DOI: [10.1007/978-3-319-26989-4_4](https://doi.org/10.1007/978-3-319-26989-4_4). URL: https://doi.org/10.1007/978-3-319-26989-4_4.