# Rajoitteilla ohjattu gradienttivahvistus-menetelmä

FM-tutkielma
Teemu Savunen
Matemaattisten tieteiden tutkimusyksikkö
Oulun yliopisto
Kevät 2021

# Tiivistelmä

Tutkielmassa tarkastellaan tekoälyyn rinnastettavien mallintamismenetelmien - valintapuiden - toimintaa vertaamalla asiantuntijoiden tietoa hyödyntävän valintapuun ja tavanomaisen valintapuun tarkkuutta toisiinsa. Lisäksi tutkielmassa käsitellään valintapuihin liitettyä mallintamisen tarkkuutta parantavaa algoritmia, joka perustuu jyrkimmän laskun menetelmään. Tavoitteena tutkielmassa on selvittää mainittujen valintapuiden toimivuutta terästen kristallisoitumislämpötilojen ennustamisessa.

Tavanomainen valintapuun toiminta perustuu puun ennustamien arvojen virheiden minimoimiseen käyttämällä virheen muodostamaa funktiota. Valintapuut koulutetaan ennustamaan mahdollisimman oikeita arvoja mallintamisen kohteena olevasta systeemistä kerätyn datan avulla. Valintapuiden muokattavuuden johdosta niiden mallintamisen tarkkuutta voidaan parantaa yhdistämällä niihin erilaisia tehostamisalgoritmeja. Tutkielmassa tarkasteltava asiantuntijan tietoon pohjautuva valintapuumenetelmä perustuu virheen minimoimisen lisäksi asiantuntijan tiedosta muodostettuihin rajoitteisiin, jotka ohjaavat puuta rakentumaan haluttuun suuntaan toimimalla ylimääräisenä sakkona, joka lisätään virheeseen silloin kun puu ei noudata annettua rajoitetta.

Mallinnettaessa kristallisoitumislämpötiloja käytetyn aineiston avulla, ei tuloksissa havaittu merkittävää eroa eri menetelmien kesken. Asiantuntijatiedon saatavuus työtä tehden oli kuitenkin rajoitettu, joten algoritmia ohjaavien rajoitteiden muodostaminen oli epävarmaa. Valintapuut ovat toimiva mallintamisen menetelmä kyseisen aineiston kohdalla johtuen aineiston monipuolisuudesta ja laajuudesta. Jatkoa ajatellen tutkielma antaa hyvän pohjan sekä toisenlaisten valintapuumenetelmien vertailuun, että entistä parempien menetelmien ja algoritmien löytämiseen.

# Sisältö

# Johdanto

Tekoälyn hyödyntäminen erilaisten ongelmien ratkaisemiseksi on yleistynyt viime vuosien aikana. Matemaattisten mallien luomisessa ongelmana on usein niiden monimutkaisuus, sillä monet mallinnettavat ilmiöt saattavat riippua useista, jopa kymmenistä, eri muuttujista. Tämän lisäksi monet näistä muuttujista voivat olla riippuvaisia toisistaan, mikä tekee mallintamisesta entistä hankalampaa. Matemaattisen mallin löytäminen tällaiselle monimutkaiselle systeemille analysoimalla systeemiä ja rakentamalla teoria sen toiminnasta on erittäin työlästä ja haastavaa. Tekoälyä hyödyntävien mallintamismenetelmien vahvuus tuntemattomien monimutkaisten systeemien mallintamisessa on niiden kyky tarkastella systeemistä kerätyn datan muuttujien vaikutusta selvitettäviin ominaisuuksiin sekä myös muuttujien välisiä suhteita, samalla kun se pyrkii luomaan mallin tai joukon malleja tutkittavasta ilmiöstä. Lisäksi mallintaminen tapahtuu iteratiivisesti, jolloin luodaan aina aikaisempia parempia malleja, jotta saavutettaisiin mahdollisimman tarkka malli systeemistä.

Yksi yleisimpiä dataa sellaisenaan hyödyntäviä mallintamismenetelmiä, mitä mallien luomisessa käytetään ovat valintapuut. Ne ovat yksinkertaisia, nopeita kouluttaa, eivätkä ne ole rajoitettu vain yhdentyyppisten muuttujien käsittelyyn, vaan muuttujat voivat saada sekä kokonaisluku- että reaalilukuarvoja. Valintapuita voidaan rakentaa ja kouluttaa monipuolisesti, jolloin moni puun parametri on käyttäjän säädettävissä.

Valintapuiden mallintamisen tarkkuutta voidaan parantaa yhdistämällä aiemmin tunnettuja tai uusia algoritmeja puiden koulutusvaiheeseen. Tässä tutkielmassa tarkastellaan yhtä suhteellisen uutta algoritmia hyödyntävää valintapuuta ja sovelletaan sitä erääseen metallurgian datajoukkoon. Algoritmi hyödyntää asiantuntijoiden antamaa informaatiota tai "ohjeita" systeemin muuttujien välisistä riippuvuuksista, joka toimii kannustavana rajoitteena ohjaten puun rakentumista. Tarkoituksena on verrata kyseisen knowledge intensive gradient boosting (KiGB) valintapuun mallien tarkkuutta tavallisen gradient boosting (GB) valintapuun mallien tarkkuuteen. Ennen tätä käydään läpi valintapuihin ja niihin liitettyjen algoritmien toimintaa, sekä valintapuiden kouluttamiseen liittyvää teoriaa.

# 1 Valintapuut ja niiden kouluttaminen

Tässä luvussa käsitellään valvottua oppimista ja yhtä sen yleisintä mallintamismenetelmää: valintapuita. Tarkoituksena on luoda selkeä kuva tutkimuksessa käytettävän menetelmän toiminnasta, sen käytöstä, vahvuuksista ja heikkouksista.

## 1.1 Valvottu oppiminen

Valvotun oppimisen tarkoituksena on etsiä paras mahdollinen funktio $f$, joka kuvaa muuttujien suhdetta tutkittavan järjestelmän jonkin tai joidenkin selvitettävien arvojen suhteen. Tarkastelemalla näiden muuttujien eli syötearvojen suhdetta selvitettäviin arvoihin eli ulostuloarvoihin, selvitetään niiden perusteella haluttua ilmiötä kuvaava funktio. Valvottua oppimista hyödyntäviä mallintamismenetelmiä on useita, mutta tässä tutkimuksessa keskitytään vain yhteen näistä eli valintapuihin. [1].

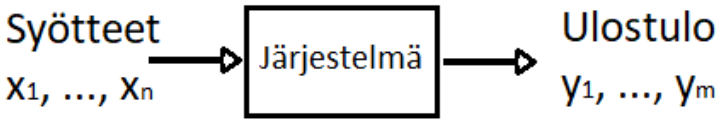
Valvotun oppimisen avulla ratkaistavat ongelmat voidaan jakaa luokitteluja regressio-ongelmiin. Sovellettavia osa-alueita valvotulle oppimiselle löytyy multimedian, biologian, tekniikan ja yhteiskunnallisilta alueilta. Esimerkiksi numeroiden tai ihmisten tunnistamiseen kuvista, geneettisten tautien etsimiseen DNA-seulonnassa, automaattiseen tekstin kääntämiseen, puheen muokkaamiseen tekstimuotoon sekä markkinahintojen ennustamisessa voidaan hyödyntää valvottua oppimista. [1].

### 1.1.1 Toiminta

Funktion $f$ etsimisessä käytetään syötevektoreista $x = (x_1, x_2, ..., x_n)$ ja niitä vastaavista ulostulovektoreista $y = (y_1, y_2, ..., y_m)$ koostuvia niin kutsuttuja näytepareja. Funktioon sijoitettujen syötevektorien arvojen tulisi palauttaa ulostulovektorin arvot. Funktion selvittämiseen käytetään ainoastaan mainittuja näytepareja, joita kerätään datana tutkittavasta järjestelmästä. Parhaan funktion etsinnästä tulee lopulta optimointiongelma, jossa funktion syötearvojen perusteella palautetut ulostuloarvojen $y'$ erotus kerätyn datan vastaavista näyteparien ulostuloarvoista $y$ tulisi olla mahdollisimman lähellä lukua 0. Toisin sanoen

$$|y - y'| = |y - f(x)| \approx 0 \quad \text{tai} \quad \{0, ..., 0\}$$

.
Tätä kutsutaan virhe- tai tappiofunktioksi, ja on oleellinen osa valvottua oppimista. Tappiofunktioita on useita erilaisia ja niiden valinta riippuu optimoitavasta ongelmasta. [1].




Kuva 1: Järjestelmän tuottamat ulostuloarvot syötearvojen perusteella

### 1.1.2 Tappiofunktio

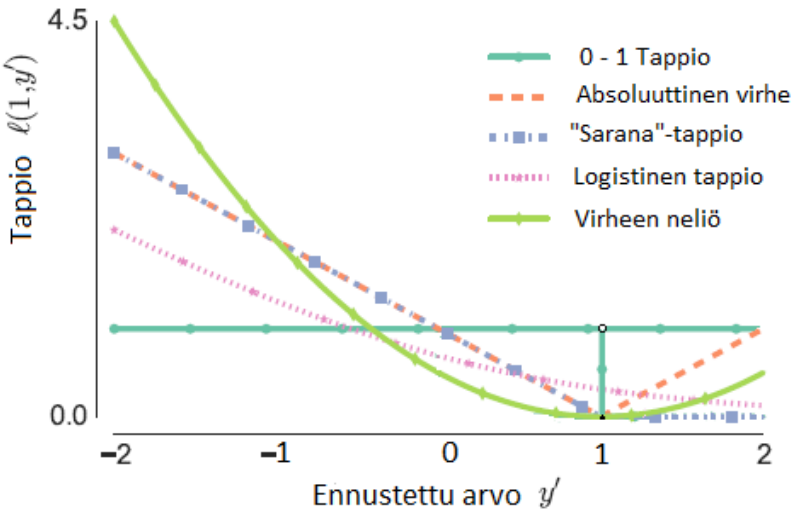
Tappiofunktion valintaan oleellisesti vaikuttaa valvotun oppimisongelman laatu. Regressiossa ($y \in \mathbb{R}^m$) käytetään usein virheen neliötä, lukuunottamatta tilanteita, joissa halutaan välttää poikkeavien arvojen vääristävää vaikutusta. Tällöin suositaan muiden virheiden käyttöä, kuten esimerkiksi absoluuttista virhettä. Luokittelussa ($y = 0, ..., k-1$) puolestaan virhe tavallisimmin ilmoitetaan keskimääräisenä $0-1$ tappiona, jota kutsutaan virhesuhteeksi. Ongelmana tällaisen funktion optimoimisessa on kuitenkin se, ettei kyseessä ole konveksi, jatkuva funktio, vaan diskreetti funktio, jonka optimoiminen on laskennallisesti hyvin haastavaa. Ongelman helpottamiseksi käytetään sen sijaan niin kutsuttua 'sarana' – tappiota tai logistista tappiota. Näissä tapauksissa ennustettaville arvoille $f(x)$ annetaan kynnysarvot, joiden ylittäessään tai alittaessaan tappio lisätään. [1].

| Regressio tappio | |
|---|---|
| Virheen neliö | $l(y, y') = \frac{1}{2}(y - y')^2$ |
| Absoluuttinen virhe | $l(y, y') = \lvert y - y' \rvert$ |
| Binäärinen luokittelu tappio | |
| $0-1$ tappio | $l(y, y') = 1(y \neq y')$ |
| Sarana-tappio | $l(y, y') = \max(0, 1 - yy')$ |
| Logistinen tappio | $l(y, y') = \log(1 + \exp(-2yy'))$ |

Taulukko 1: Tappiofunktioita

Kuvan 2 kuvaajissa on esitetty tappiofunktioiden antamat arvot $l(1, y')$. Nähdään, että $0-1$ tappio on vakiofunktio, joka on epäjatkuva kohdassa $y' = 1$. Sarana tappio on puolestaan lineaarinen aina, kun $y' \leq 1$ ja on vakio, kun $y' \geq 1$. Logistinen tappio rankaisee kaikilla virheillä ja on ainoastaan nolla, kun malli antaa oikean arvon äärettömyydessä. [1].

Jos tarkastellaan regressiossa käytettäviä tappiofunktioita voidaan huomata, että jokaisesta oikeellisesta arvosta $y$ poikkeavasta arvosta rangaistaan eri määrällä riippuen siitä, kuinka kaukana ennustettu arvo on oikeasta




Kuva 2: Tappio suhteessa oikeelliseen arvoon $y = 1$ (käännetty [1])

arvosta. Luokittelussa puolestaan ajatellaan, että virheestä syntyvä tappio riippuu vain siitä, onko ennustettu arvo luokiteltu oikein. Ongelmissa, joissa on tarkoituksena valvotun oppimisen avulla luokitella syötteiden perusteella, eivät regressio-ongelmissa käytetyt tappiofunktiot välttämättä kyseisestä syystä sovellu hyvin. [1].

## 1.2 Valintapuut

Valintapuut ovat eräs valvotun oppimisen menetelmä mallintaa syöte-avaruuden arvot ulostulo-avaruuden arvoihin. Ne ovat hierarkisesti rakennettu ja koostuvat testi- ja lehtisolmuista. Valintapuu alkaa latvasolmusta, joka testisolmuna ohjaa näytteiden jakautumista puussa niiden muuttujien arvojen perusteella. Lopulta näytteet saapuvat lehtisolmuihin, joissa niihin liitetään niille laskettu ulostuloarvo eli ennuste. [1].

Tätä menetelmää voidaan verrata ihmisten loogisiin ja organisoituihin peräkkäisesti järjestettyihin kyllä/ei kysymyksiin. Esimerkkinä tästä menetelmästä on lääkärin tekemä lääketieteellinen diagnoosi potilaan terveydellisestä tilasta [1]. Lääkäri esittää potilaalle joukon kysymyksiä ja vastausten perusteella lääkäri voi jatkaa jatkokysymyksillä riippuen potilaan antamista vastauksista. Menetelmän kysymykset ovat verrattavissa valintapuun testisolmuihin, ja vastaukset näyteparin syötearvoihin, jotka kuvaavat poti-

laan terveydentilan ominaisuuksia tai arvoja. Kysymysten ei tarvitse olla kyllä/ei tyyppisiä, vaan ne voivat olla luokittelevia, kuten esimerkiksi kysymys: "Kuinka usein harrastat liikuntaa?, voidaan jakaa useampaan luokkaan, joiden valitsemisella on tietynlainen vaikutus lopulliseen diagnoosiin. Kun syötteenä toimii puolestaan reaaliluku - kuten esimerkiksi lämpötila - kysymys voidaan esittää muodossa: "Onko potilaan lämpötila alle 37 astetta?". Tällöin testisolmu jakaa näytteen eri syötteet kahteen lapsisolmuun, sen mukaan onko näytteen lämpötila-arvo yli vai alle 37 astetta. Kyseiset lapsisolmut voivat jälleen toimia testisolmuina, joissa näytteet jaetaan lämpötila-arvon perusteella tai jonkin muun syötteen muuttujan arvon perusteella. Tätä hierarkista menetelmää voidaan jatkaa niin pitkälle kuin on nähty tarpeelliseksi ja se päättyy aina yhteen lehtisolmuun, johon on sidottu tietty lopullinen arvo eli näytteen ennustettu ulostuloarvo.

Valintapuut ovat suosittuja koneoppimisen malleja muun muassa seuraavista syistä:

1. Hierarkinen malli ottaa huomioon mahdolliset epälineaarisuudet ja ehdolliset riippuvuudet syötteiden ja ulostulojen välillä.

2. Valintapuun kasvatus on laskennallisesti nopeaa.

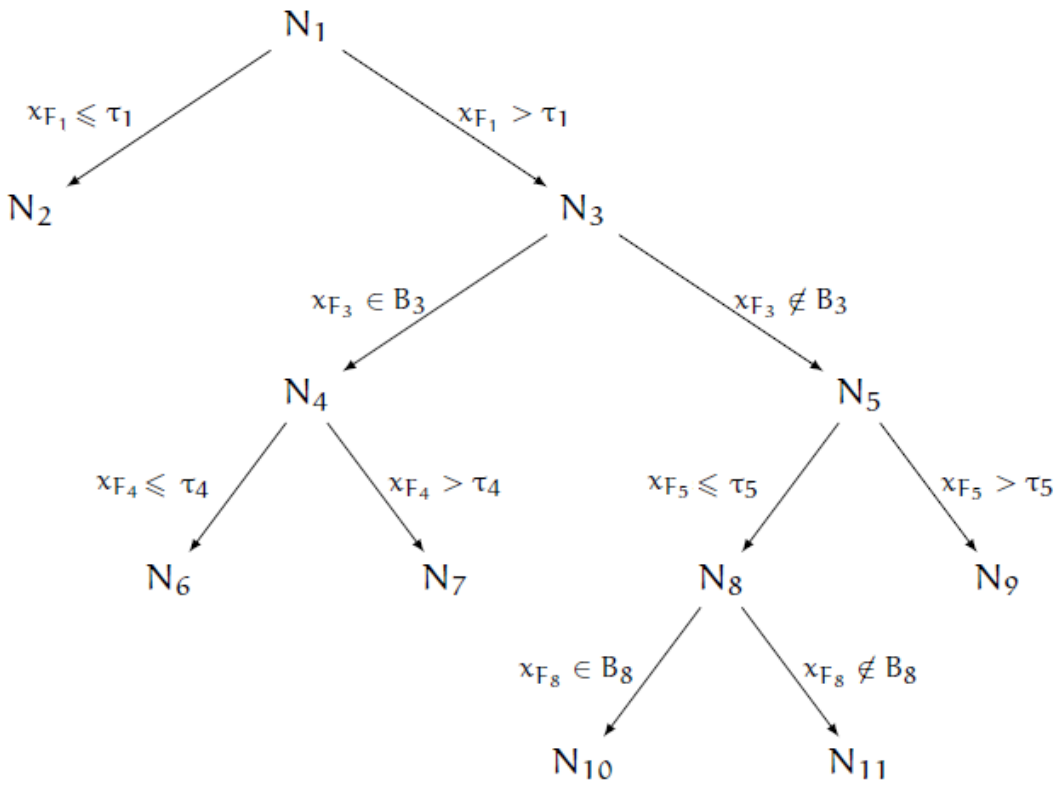
3. Menetelmän toiminta on riippumaton datan laadusta, toisin sanoen se voi koostua binäärisistä (kyllä/ei), luokittelevista ja numeerisista syötemuuttujista.

4. Valintapuut ovat tulkittavissa olevia malleja, jolloin niiden rakenteesta voidaan päätellä eri syötteiden vaikutusta ulostuloarvoihin.

5. Valintapuun koulutusalgoritmia voidaan muokata siten, että se kykenee käsittelemään puuttuvia syötearvoja. [1].

### 1.2.1 Valintapuun rakenne

Yleinen valintapuun rakenteen malli, jota käytetään on binäärinen. Tällöin yhdestä testisolmusta lähtee kaksi haaraa, johon näytteen syötteet jaetaan testisolmun ehdon mukaan, vasen lapsisolmu ja oikea lapsisolmu. Testisolmuja kutsutaan myös sisäisiksi solmuiksi tai haarasolmuiksi. Lehtisolmujen toisia nimityksiä ovat muun muassa ulkoiset solmut ja päätesolmut. Testisolmulla $N_t$ on siis vasen ja oikea lapsi. Lisäksi sillä on jakosääntö $s_t$, joka testaa sen kuuluuko näyte vasemmalle vai oikealle lapselle. Jatkuvien arvojen syötteille jakosääntö on yleisesti muotoa $s_t(x) = x_{Ft} \leq \tau_t$, mikä testaa onko syötearvo $x_{Ft}$ pienempi tai yhtä suuri kuin vakio $\tau_t$. Jos kyseessä on




Kuva 3: Binäärinen valintapuu, joka koostuu viidestä testisolmusta ja kuudesta lehtisolmusta. Lähde: [1].

binäärinen tai luokittelusyöte, jakosääntö on muotoa $s_t(x) = x_{Ft} \in B_t$, mikä testaa sen kuuluuko syötearvo $x_{Ft}$ osajoukkoon $B_t$. [1].

Kuvan 3 valintapuu ensimmäisenä jakaa syöte-avaruuden kahteen joukkoon $A_2 = \{x : x_{F1} \leq \tau_1\}$ ja $A_3 = \{x : x_{F1} > \tau_1\}$ latvasolmussa $N_1$, jolla on kaksi lasta: vasen lapsi $N_2$ ja oikea lapsi $N_3$. $N_2$ on lehtisolmu ja $N_3$ testisolmu, joka edelleen jakaa syöte-avaruuden luokittelevan joukon $B_3$ perusteella osiin $A_4 = \{x \in A_3, x_{F3} \in B_3\}$ ja $A_5 = \{x \in A_3, x_{F3} \notin B_3\}$, missä $A_3 = A_4 \cup A_5$. Syöte-avaruus jaetaan vielä kolmella muulla testisolmulla, joista kaksi tapahtuvat reaaliarvoisen jakosäännön perusteella $(N_4, N_5)$ ja yksi luokittelevan jakosäännön perusteella $N_8$. Loput solmuista $N_6$, $N_7$, $N_9$, $N_{10}$ ja $N_{11}$ ovat lehtisolmuja.

Testattaessa valintapuu ennustaa ennalta tuntemattoman näytteen ulostulon seuraamalla rakennetun puun testisolmujen jakosääntöjen logiikkaa. Tämä rekursiivinen prosessi alkaa latvasolmusta, josta näyte kulkee testisol-

mujen jakosääntöjen mukaan puuta alas. Näytteen kulku jatkuu, kunnes se saapuu lehtisolmuun, jossa kyseiselle lehtisolmulle liitetty arvo toimii ennustettuna arvona. [1]. Valintapuun mallia $f : \mathbb{X} \to \mathbb{Y}$ voidaan kuvata indikaattorifunktioiden $1(x \in A_t)$ summana jokaisesta puun $|T|$ solmusta, missä $|T|$ on puun kaikkien solmujen joukko. Lisäksi merkitään lehtisolmujen joukkoa $T_l \subset |T|$.

**Määritelmä 1.1** (Indikaattorifunktio)**.** Indikaattorifunktio palauttaa arvon 1, jos indikaattorifunktion muuttuja täyttää sille määritellyn ehdon. Ehtona pidetään sitä, kuuluuko muuttujan arvo tiettyyn joukkoon $A$. Tapauksessa, jossa muuttujaan liitetty ehto ei täyty, indikaattorifunktio palauttaa arvon 0.

$$1(x \in A) = \begin{cases} 1 & ,\text{kun } x \in A \\ 0 & ,\text{kun } x \notin A \end{cases} \tag{1}$$

Syötemuuttujan $x$, puun solmun indeksin $t$ sekä indikaattorifunktioiden avulla voidaan muodostaa valintapuun malli $f$.

$$f(x) = \sum_{t=1}^{|T|} \beta_t 1(x \in A_t) 1(t \in T_l), \tag{2}$$

missä $\beta_t$ on solmuun $N_t$ liitetty ennuste. Laskennallinen monimutkaisuus ennaltaan tuntemattoman näytteen ennustamiseen on täten riippuvainen kuljetun reitin syvyydestä. [1].

**Algoritmi 1** Ennusta näytteen x arvo valintapuulla [1].

**function** PUU_ENNUSTAA(puu, x)
$t \leftarrow$ puun latvasolmun indeksi
**while** solmu $N_t$ ei ole lehtisolmu **do**
  **if** Solmun $N_t$ jakosääntö $s_t(x)$ on totta **then**
    $t \leftarrow$ Vasemman lapsisolmun indeksi
  **else**
    $t \leftarrow$ Oikean lapsisolmun indeksi
  **end if**
**end while**
**palauta** arvo $\beta_t$

Kyseisen binäärisen valintapuun malli toimii myös useampi haaraisella valintapuun rakenteella. Jokaisella puun sisäisellä solmulla voi olla enemmän

kuin kaksi lasta, käyttämällä useampaan haaraan jakavaa jakosääntöä. Tätä voidaan kuitenkin verrata peräkkäisiin binäärisiin jakoihin. Useampi haaraisten jakojen ongelmana on opetusdatan nopea sirpaloituminen valintapuun kasvattamisessa, mikä heikentää puun yleistämiskykyä. [1].

### 1.2.2 Valintapuun kasvattaminen

Valintapuiden kasvattaminen tapahtuu syöte-ulostulo näytepareilla. Puun kasvattaminen aloitetaan latvasolmusta, missä syöte-avaruus jaetaan rekursiivisesti jakosääntöjen avulla, kunnes lopulta saavutetaan päätöskriteeri, joka voi olla puun maksimi syvyys tai minimi määrä näytteitä solmussa. Jokaiselle uudelle rakennetulle testisolmulle etsitään paras mahdollinen jakosääntö, joka jakaa koulutukseen käytettävät näytteet kahteen joukkoon. Osituksen tarkoitus on luoda "puhtaampia" näytejoukkoja, missä samantyyppiset näytteet löytyisivät samasta joukosta. Tämä kasvatusprosessi voidaan määritellä kolmella ominaisuudella:

* Jakosäännön etsintä algoritmi.
* Haarautumisen päätöskriteeri, jolla lopetetaan näytteiden jakaminen puussa.
* Lehtisolmujen ulostuloarvon määrittäminen. [1].

Yhdistettäessä kaikki nämä tekijät yhteen saadaan muodostettua seuraava puun kasvatus algoritmi.

**Algoritmi 2** Kasvata valintapuu käyttäen näytejoukkoa $L = \{(x^i, y^i) \in \mathbb{X} \times \mathbb{Y}\}_{i=1}^{n}$ [1].

**function** KASVATA_PUU(L)
$q$ = TyhjäJono() (pitää kirjaa kullekin solmulle kuuluvista näytejoukoista)
Alusta puun rakenne latvasolmulla ($N_1$)
q.LisääJonoon($(1, L)$) (lisää jonoon 1. solmun indeksin ja näytteet).
**while** $q$ ei ole tyhjä **do**
  $(t, L_t) \leftarrow$ q.PoistaJonosta() (poistaa ja sijoittaa viimeisimmän lisäyksen arvot).
  **if** Solmu $N_t$ tyydyttää yhden päätös kriteereistä **then**
    Merkitään solmu $t$ *lehdeksi*, jolle kuuluu näytteet $L_t$
  **else**
    Etsi paras jakosääntö $s_t$ näytteille $L_t$.
    Jaa $L_t$ joukkoihin $L_{t,r}$ ja $L_{t,l}$ jakosäännöllä $s_t$.
    Merkitse solmu $t$ *testisolmuksi*, jolla on jakosääntö $s_t$.
    q.LisääJonoon($(2t, L_{t,l})$) (lisää vasemman lapsen indeksin näytteineen).

q.LisääJonoon$((2t+1, L_{t,r}))$ (lisää oikean lapsen indeksin näytteineen).
**end if**
**end while**
**palauta** Kasvatettu puu.

Puuta kasvattaessa me ositamme rekursiivisesti syöte-avaruuden $\mathbb{X}$ ja näytejoukon $L = \{(x^i, y^i) \in \mathbb{X} \times \mathbb{Y}\}_{i=1}^{n}$. Jokaisessa testisolmussa $t$ näytejoukko $L_t$ jaetaan kahteen pienempään osajoukkoon $L_{t,l}$ ja $L_{t,r}$ binäärisen jakosäännön $s_t : \mathbb{X} \to \{0, 1\}$ avulla. Tästä jakosäännöstä herää kaksi kysymystä: mikä on käytettävissä olevien binääristen tai akselittaisten jakosääntöjen joukko $\Omega(L_t)$ näytejoukon $L_t$ kohdalla ja kuinka valita tästä joukosta paras jakosääntö, joka tekee testisolmun jälkeläisten näytejoukoista "puhtaampia" kuin itse testisolmun? [1].

Muuttujaan $x_j \in X_j$ liittyvä perhe $Q(x_j)$ jakosääntöjä koostuu kaikista mahdollisista osajoukoista $X'$:

$$Q(x_j) = \{s_t(x) \equiv 1(x_j \in X') : X' \subset X_j\} \tag{3}$$

Jakosääntöperheen koko kasvaa eksponentiaalisesti kaikkien mahdollisten arvojen kasvaessa ($|Q(x_j)| = 2^{|X_j|-1}$) [1].

Jos muuttujan $x_j$ mahdolliset arvot ovat järjestetty, jakosääntöperheen koko voidaan pienentää lineaariseen lukuun jakosääntöjä ($|Q(x_j)| = |X_j|-1$):

$$Q(x_j) = \Big\{s_t(x) \equiv 1(x_j \leq \tau) : \tau \in X_j\Big\}. \tag{4}$$

Numeerisella reaaliarvoisella muuttujalla $x_j \in \mathbb{R}$, mahdollisten jakosääntöjen määrä on ääretön. Tästä huolimatta, opetusdatan näytejoukon koko on aina äärellinen. Valitun jakosäännön $s_t$ tulisi jakaa näytejoukko $L_t$ siten, että seuraavat ehdot pätevät: näytejoukot $L_{t,r}$ ja $L_{t,l}$ eivät ole tyhjiä ($L_{t,r} \neq \emptyset, L_{t,l} \neq \emptyset$) muodostaen leikkauksen $L_{t,l} \cap L_{t,r} = \emptyset$) ja yhdisteen ($L_t = L_{t,l} \cup L_{t,r}$) alkuperäisesti näytejoukosta $L_t$. Puuta jatkettaessa testisolmun $t$ vasemmalla ja oikealla lapsella jakosääntö $s_t$ valitaan kaikkien mahdollisten jakosääntöjen joukosta $\Omega(L_t)$. [1].

$$\begin{aligned} s_t \in \Omega(L_t) = \Big\{ s : s \in & \bigcup_{j \in \{1,\ldots,n\}} Q(x_j), \\ & L_{t,l} = \{(x, y) \in L_t : s(x) = 1\}, \\ & L_{t,r} = \{(x, y) \in L_t : s(x) = 0\}, \\ & L_{t,l} \neq \emptyset, L_{t,r} \neq \emptyset \Big\} \end{aligned} \tag{5}$$

### 1.2.3 Jakosäännön valinta

Tarkoituksena on valita "paras" mahdollinen paikallinen jakosääntö $s_t$ testisolmulle $t$, joka johtaisi hyvään yleistämiskykyyn. Tähän kykyyn liittyvää virhettä on kuitenkin mahdotonta minimoida. Virhe voitaisiin liittää myös opetusdatajoukkoon, mutta tällä tavalla luotu puu on triviaali ja sen kyky yleistää on huono. Näiden vaihtoehtojen sijaan puun kasvattaminen tehdään "ahneesti" maksimoimalla niin kutsutun epäpuhtausfunktion $I : (X \times Y) \times ... \times (X \times Y) \to \mathbb{R}$ vähentäminen. [1]. Matemaattisesti epäpuhtauden vähentäminen $\Delta I$ määritetään jakamalla näytejoukko $L_t$ kahteen osajoukkoon $(L_{t,r}, L_{t,l})$ seuraavasti

$$\Delta I(L_t, L_{t,l}, L_{t,r}) = I(L_t) - \frac{|L_{t,l}|}{|L_t|} I(L_{t,l}) - \frac{|L_{t,r}|}{|L_t|} I(L_{t,r}). \tag{6}$$

Jakosäännön valintaan liittyvä ongelma (katso Algoritmi 2) voidaan esittää täten

$$s_t = \arg \max_{s \in \Omega(L_t)} \Delta I(L_t, L_{t,l}, L_{t,r}). \tag{7}$$

Epäpuhtaus $I$ on pienimmillään, kun kaikilla näytteillä on sama ulostuloarvo. Tällöin solmun sanotaan olevan "puhdas". Koska epäpuhtauden vähentäminen on additiivinen, paras jako solmussa $t$ paikallisesti lasketulle epäpuhtauden vähennykselle on paras jako solmulle myös koko puun yhteenlasketulle epäpuhtaudelle. Jäljelle jäävä puun $T$ epäpuhtaus $Imp(T)$ on kaikkien lehtisolmujen epäpuhtauksien summa. [1].

$$Imp(T) = \sum_{t \in T} p(t) I(L_t) 1(t \in T_l), \tag{8}$$

missä $p(t) = |L_t|/|L|$ on solmuun $t$ saapuvien näytteiden osuus. Jos ajatellaan, että lehtisolmun $t$ muutettaessa testisolmuksi, jolla on jakosääntö $s_t$ ja vasen $t_l$ ja oikea $t_r$ lapsisolmu luo uuden puun $T'$. Yhteenlaskettu epäpuhtaus vähenee alkuperäisestä puusta $T$ uuteen isompaan puuhun $T'$ verrattuna ja epäpuhtauden vähennys on

$$\begin{aligned} Imp(T) - Imp(T') &= p(t) I(L_t) - p(t_l) I(L_{t,l}) - p(t_r) I(L_{t,r}) \\ &= p(t) \Delta I(L_t, L_{t,l}, L_{t,r}). \end{aligned} \tag{9}$$

Täten puun kasvatus on toistuva prosessi, jossa puun epäpuhtautta pyritään

vähentämään niin nopeasti kuin mahdollista valitsemalla juuri oikeat paikalliset jakosäännöt. [1].

Tapauksissa, joissa näytteiden jakaminen tapahtuu luokittelun perusteella, solmun sanotaan olevan puhdas, kun kaikilla näytteillä on sama luokka. Annettaessa näytejoukko $L_t$, joka saapuu solmuun $t$, tarkoittaa $p(y = l|t) = \frac{1}{|L_t|} \sum_{(x,y)\in L_t} 1(y = l)$ sitä osuutta näytteistä, joilla on luokka $l$ ja saapuvat solmuun $t$. Solmu on puhdas silloin, kun $p(y = l|t)$ on 1 luokalle $l$ ja 0 kaikille muille. Solmun epäpuhtaus tulisi kasvaa aina, kun näytteet, joilla on eri luokat sekoittuvat keskenään. Näiden perusteella epäpuhtauden $I$ luokittelutapauksissa tulee noudattaa seuraavia ehtoja.

1. $I$ saavuttaa minimin vain silloin, kun solmu on puhdas $p(y = l|t) = 1$ ja $p(y = m|t) = 0 \ \ \forall m \in \{1, ..., l-1, l+1, ..., k\}$;
2. $I$ saavuttaa maksimin vain silloin, kun $p(y = l|t) = 1/k \ \ \forall l \in \{1, ..., k\}$:
3. $I$ on symmetrinen funktio suhteessa luokkien suhteisiin $p(y = 1|t), ...,$ $p(y = k|t)$, jotta mitään luokkaa ei suosittaisi. [1].

Yksi funktio, joka toteuttaa nämä kolme ehtoa on väärin luokittelun virhesuhde:

$$\text{Error rate}(L_t) = 1 - \max_{l\in\{1,...,k\}} p(y = l|t). \tag{10}$$

Tämä funktio ei toimi sellaisenaan. Käytännössä monella jakosäännöllä on usein sama virhesuhteen vähentäminen, varsinkin moniluokkaisissa tapauksissa, missä vain suurimman luokan näytteiden määrällä on väliä. [1].

Otetaan esimerkkinä seuraavanlainen luokkien jaon muodostama ongelma, missä on kyse binäärisestä luokittelusta, jossa näytteinä toimii 500 negatiivista ja 500 positiivista näytettä. Ensimmäinen jakosääntö jakaa 125 positiivista ja 375 negatiivista näytettä vasemmalle lapselle ja 375 positiivista ja 125 negatiivista näytettä oikealle lapselle. [1]. Väärin luokittelun virhesuhteen vähennys lasketaan tällöin kaavan (6) mukaisesti:

$$\frac{500}{1000} - 2\frac{500}{1000}(1 - \frac{375}{500}) = 0,25.$$

Seuraavaksi tarkastellaan toista jakosääntöä, joka johtaa puhtaaseen lapsisolmuun, mihin menee 250 positiivista näytettä ja toiseen solmuun puolestaan 250 positiivista ja 500 negatiivista näytettä. Tämän jakosäännön epäpuhtauden vähentämisen arvo on yhtä suuri. [1].

$$\frac{500}{1000} - \frac{750}{1000}(1 - \frac{500}{750}) - \frac{250}{1000}(1 - \frac{250}{250}) = 0,25.$$

Väärin luokittelu ei täten rankaise tarpeeksi solmun puhtautta, sillä tämä on lineaarisesti riippuvainen suurimman luokan näyteosuuteen. Tämä ongelma voidaan ratkaista lisäämällä luokittelun epäpuhtausfunktioon neljäs ehto:

4. $I$ täytyy olla kovera suhteessa luokkasuhteeseen $p(y = l|t)$. [1].

Neljäs ehto sirpaloittaa epäpuhtaiden vähentämiseen liittyviä arvoja, jolloin jakosäännöt eivät tuota yhtä suuria epäpuhtauden vähentämisen arvoja niin useasti. Samalla se vähentää puun epävakautta opetusdatajoukon suhteen. [1].

Kaksi sopivampaa luokittelun epäpuhtauden mittakriteeriä, jotka toteuttavat nämä kaikki neljä ehtoa ovat Gini indeksi, tilastollisen hajonnan mitta, ja entropian mitta, informaatioteorian mitta [1].

$$\text{Gini}(L_t) = \sum_{l=1}^{k} p(y = l|t)(1 - p(y = l|t)) \tag{11}$$

$$\text{Entropy}(L_t) = -\sum_{l=1}^{k} p(y = l|t) \log p(y = l|t) \tag{12}$$

Minimoimalla Gini indeksi, samalla minimoidaan luokkien hajonta. Kun jakosääntö valitaan entropian mitan perusteella, minimoidaan kohteen ennustettamattomuutta. [1].

Palataan aiempiin luokkien jaon esimerkkeihin ja lasketaan niiden epäpuhtauden vähennys käyttäen Gini indeksiä ja entropiaa. Ensimmäisellä jaolla olisi Gini indeksin arvo 0,0625, kun taas toisella jaolla, jossa toinen solmu on puhdas, se olisi $\approx$ 0,083. Laskettu entropian mitan arvo ensimmäisen jaon kohdalla olisi $\approx$ 0,131 ja toisen jaon puolestaan $\approx$ 0,216. Näiden mittakriteerien perusteella valittaisiin esimerkeistä jälkimmäinen molempien mittojen tapauksissa. [1].

Tarkastellaan seuraavaksi jaon epäpuhtautta regression kohdalla. Tällöin solmun ajatellaan olevan puhdas, kun puusta ulostulevan arvon ja oikeellisen arvon hajonta on 0. Regressiossa epäpuhtauden kriteeri määritetään vain silloin nollaksi, kun kaikki ulostuloarvot ja oikeat arvot ovat samoja. Yleinen hajonnan mitta, jota regressiopuiden kasvattamisessa käytetään on empiirinen varianssi. [1].

$$\text{Variance}(L_t) = \frac{1}{|L_t|} \sum_{(x,y)\in L_t} (y - \bar{y})^2, \text{ missä } \bar{y} = \frac{1}{|L_t|} \sum_{(x,y)\in L_t} y \tag{13}$$

Varianssin minimoimiseksi etsitään jakosääntöä minimoimalla virheen neliötä $l(y, y') = \frac{1}{2}(y - y')^2$. Huomioon otettavaa on se, että Gini indeksi ja empiirinen varianssi johtavat samaan epäpuhtauden mittaan binääristen luokittelujen ja moniluokittelujen kohdalla, kun ulostuloluokat merkitään numeerisilla muuttujilla $\{0, 1\}$. [1].

### 1.2.4 Lehtien ulostuloarvojen määrittäminen ja kasvattamisen päättäminen

Kun puun kasvattaminen pysäytetään päätöskriteerin perusteella, syntyneelle uudelle lehdelle täytyy asettaa jokin ulostuloarvo (katso Algoritmi 2). Merkitään tätä vakiolla $\beta_t$, joka valitaan siten, että minimoidaan tappiofunktio $l$ kaikkien näytteiden, jotka saavuttavat kyseisen lehtisolmun $t$, suhteen. [1].

$$\beta_t = \arg\min_{\beta} \sum_{(x,y) \in L_t} l(y, \beta). \tag{14}$$

Regression tapauksissa vakion $\beta_t$ etsinnässä käytetään virheen neliötä ja sen minimointia:

$$\beta_t = \arg\min_{\beta} \sum_{(x,y) \in L_t} \frac{1}{2}(y - \beta_t)^2. \tag{15}$$

Kun tämä funktio derivoidaan ja merkitään nollaksi, saadaan:

$$\sum_{(x,y) \in L_t} (y - \beta_t) = 0 \;\Leftrightarrow\; \beta_t = \frac{1}{|L_t|} \sum_{(x,y) \in L_t} y. \tag{16}$$

Lehden ennustaman vakion ulostuloarvo virheen neliön minimoinnissa on siis kaikkien lehteen $t$ saapuvien näytteiden oikeiden ulostuloarvojen keskiarvo. [1].

Luokittelun tapauksissa vakion $\beta_t$ etsiminen tapahtuu 0 – 1 tappiofunktiolla ($l_{0-1}(y, y') = 1(y \neq y')$). Tällöin vakio $\beta_t$ on yleisimmän luokan arvo. Valintapuu voi toimia myös luokkien todennäköisyyksien estimaattorina ennustamalla lehtisolmuun $t$ saapuvien näytteiden, joilla on sama luokka, osuutta kaikista kyseiseen lehtisolmuun päätyvistä näytteistä. [1].

Lehden ennusteen ei tarvitse noudattaa tätä kyseistä vakioarvon mallia, vaan valintapuiden lehtien ulostuloarvo voi noudattaa muita malleja, jotka lisäävät valintapuun mallintamisen tarkkuutta laskenta-ajan ja uusien hyperparametrien kustannuksella [1]. Tätä käsitellään seuraavassa gradient boosting luvussa.

Puun kasvattamisen päättäminen tapahtuu haarautumisen päätöskriteerin avulla. Tämän päätöskriteerin tehtävänä on välttää puun ylisovittamista opetusdataan pysäyttämällä puun kasvattaminen ajoissa. Kriteerit perustuvat joko puun rakenteellisiin ominaisuusiin tai itse käytettyyn dataan. Kriteeri voi liittyä suoraan myös jakosääntöön, esimerkiksi vaatimalla minimimäärän näytteitä solmun vasemmalle ja oikealle lapsisolmulle. [1].

Puun rakenteellisuuteen perustuvat päätöskriteerit säätävät puun kasvua tarkasti rajoittamalla puun monimutkaisuutta esimerkiksi seuraavien ominaisuuksien perusteella:

- haarojen syvyydet tai
- solmujen kokonaismäärä [1].

Näistä jälkimmäisessä tapauksessa järjestys, jossa puun solmut jaetaan, vaikuttaa lopputulokseen ja tämä järjestys voidaan valita siten, että epäpuhtauden vähentäminen on maksimoitu [1].

Dataan perustuvat päätöskriteerit pysäyttävät puun kasvun, jos jokin laskettu tilastollinen ominaisuus jakosäännön dataan ei toteuta jotain tiettyä rajoitetta kuten:

- solmuun saapuvien näytteiden määrää,
- vasempaan ja oikeaan lapsisolmuun saapuvien näytteiden määrää jakamisen jälkeen,
- epäpuhtauden vähennystä tai
- jonkin tilastollisen merkitsevyystestin p-arvoa, kuten esimerkiksi Chineliötestin tai ulostuloarvon ja jaon riippumattomuuden testaamisen. [1].

### 1.2.5 Valintapuun tulkinta

Yksi valintapuumallin vahvuuksista on niiden tulkittavuus. Tarkasteltaessa valintapuita voimme muuntaa tämän mallin joukoksi toisiansa poissulkevia luokittelu- tai regressio- eli reaaliarvosääntöjä. Nämä säännöt saadaan seurattaessa polkua lehtisolmusta latvasolmuun. Seuraavana on muutettu Kuvan 4 valintapuu kolmeksi joukoksi jako- tai ennustussääntöjä, yksi jokaiselle iris-kukan lajille:

1. Jos terälehden leveys $\leq$ 0,7 cm, niin Setosa.

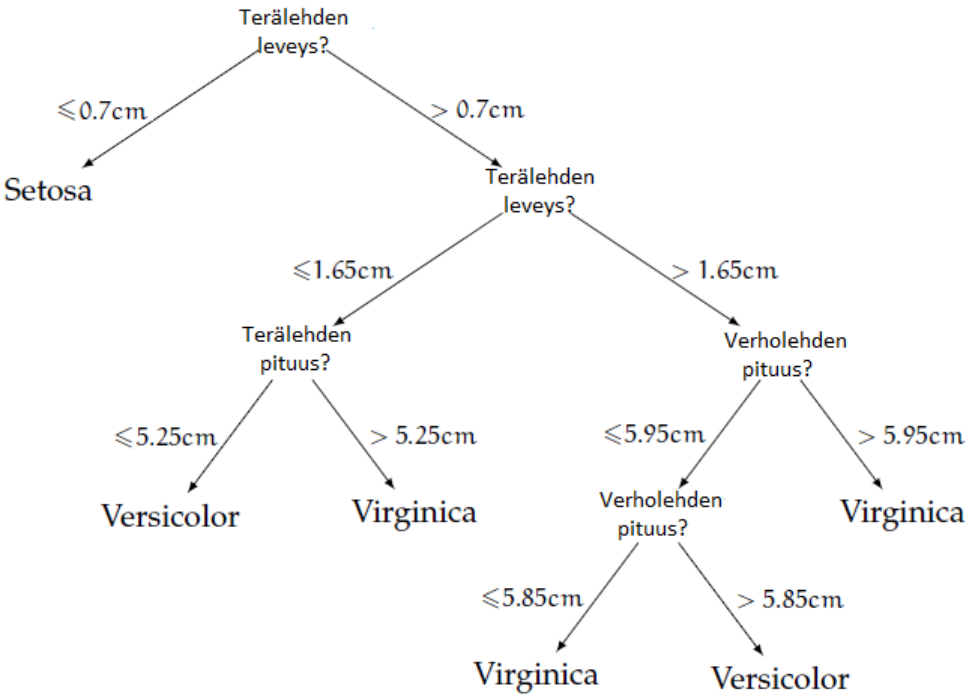
2. Jos terälehden leveys $> 0{,}7$ cm ja terälehden leveys $\leq 1{,}65$ cm ja terälehden pituus $\leq 5{,}25$ cm, niin Versicolor.

3. Jos terälehden leveys $> 0{,}7$ cm ja terälehden leveys $> 1{,}65$ cm ja verholehden pituus $\leq 5{,}95$ cm ja verholehden pituus $> 5{,}85$ cm, niin Versicolor.

4. Jos terälehden leveys $> 0{,}7$ cm ja terälehden leveys $\leq 1{,}65$ cm ja terälehden pituus $> 5{,}25$ cm, niin Virginica.

5. Jos terälehden leveys $> 0{,}7$ cm ja terälehden leveys $> 1{,}65$ cm ja verholehden pituus $\leq 5{,}95$ cm ja verholehden pituus $\leq$, niin Virginica.

6. Jos terälehden leveys $> 0{,}7$ cm ja terälehden leveys $> 1{,}65$ cm ja verholehden pituus $> 5{,}95$ cm, niin Virginica. [1].



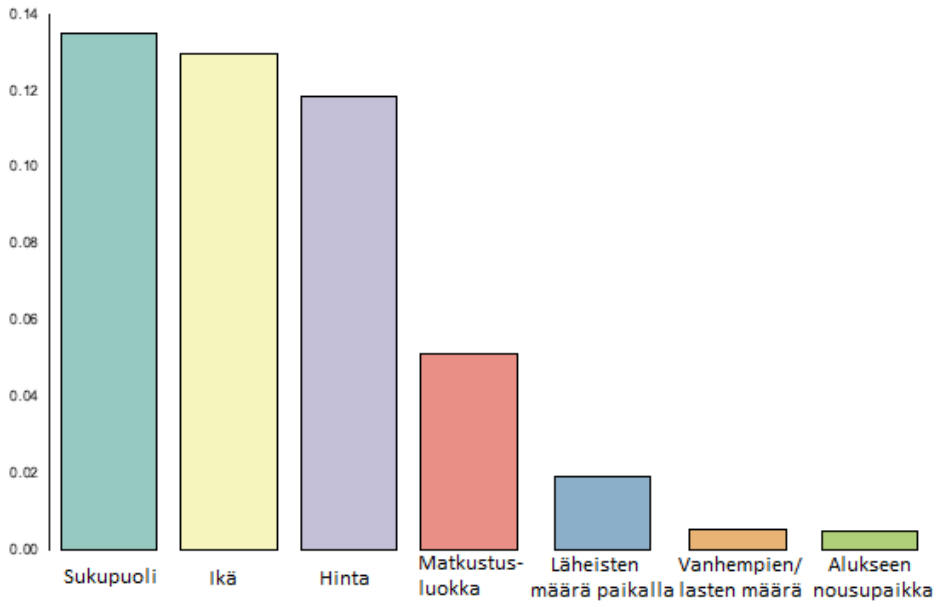

Kuva 4: Valintapuu on tulkittava joukko sääntöjä, jotka on organisoitu hierarkisesti. Kukkalajien (Versicolor, Virginica, Setosa) tunnistamiseen käytettävä valintapuu neljän eri syötemuuttujan (terälehden pituus ja leveys, verholehden pituus ja leveys) perusteella. (Käännetty [1]).

Huomioon otettavaa on binäärinen hierarkinen rakenne, jossa jotkin säännöt ovat tarpeettoman toistavia ja niitä voidaan yksinkertaistaa. Esimerkiksi

6. säännössä voidaan yhdistää säännöt "Terälehden leveys $>$ 0,7 cm ja terälehden leveys $>$ 1,65 cm" säännöksi "Terälehden leveys $>$ 1,65 cm". [1].

Valintapuun malli myös osoittaa, mitkä syötemuuttujat $x_1, ..., x_n$ ovat ulostulomuuttujan tai -muuttujien $y$ ennustamisessa tärkeitä. Tätä syötemuuttujien tärkeyttä voidaan mitata keskimääräisellä epäpuhtauden vähentämisellä (MDI, mean decrease of impurity) syötemuuttujan $x_j$ suhteen. Funktiona esitettynä MDI yhteenlaskee kaikkien valintapuun $T$ solmujen, missä muuttuja $x_j$ toimii jakosääntönä, epäpuhtauden vähennykset painotettuna kyseiseen solmuun saapuvien näytteiden $|L_t|/|L|$ suhteella. [1].

$$\mathrm{MDI}(x_j) = \sum_{t \in T : x_j \text{ on jakosääntö } s_t} \frac{|L_t|}{|L|} \Delta I(L_t, L_{t,l}, L_{t,r}). \tag{17}$$




Kuva 5: Keskimääräinen epäpuhtauden vähennys jokaiselle syötemuuttujalle Titanic tietojoukolle valintapuun tarkkuuden optimoimiseksi. (Käännetty [1])

Funktio MDI arvostelee syötemuuttujat niiden puun kasvattamisessa olevan merkittävyyden perusteella, kuten Kuvassa 5 on havainnollistettu. Se ottaa huomioon muuttujien korrelaatiot, monimuuttuja ja ei-lineaariset vaikutukset. Valintapuita hyödynnetään usein esikäsittelyn työkaluina valittaessa syötemuuttujista vain kaikkein tärkeimmät muuttujat. [1].

# 2 Gradient boosting

Tämä luku käsittelee jyrkimmän laskun menetelmää ja miten sitä voidaan hyödyntää valintapuissa entistä tehokkaampien mallien luomisessa.

Esitetty valintapuun malli, jossa lehtisolmut ennustavat tietyn vakioarvon on esimerkki dataan pohjautuvasta yhdestä vahvasta ennustusmallista. Tämä ei välttämättä toimi halutulla tavalla, jos tarkoituksena on ennustaa hyvin monipuolista ja riippuvuuksiltaan monimutkaista dataa. Parempi lähestymistapa olisi muodostaa useamman ennustusmallin kokonaisuus tällaiselle valvotun oppimisen tehtävälle. Gradient boosting koneissa (GBM) oppimisen menetelmänä on sovittaa peräkkäin uusia malleja, jotka tuottavat entistä parempia ennusteita ulostulo- tai vastemuuttujasta. Pääidea tämän algoritmin takana on rakentaa uudet "perusoppijat" siten, että ne korreloisivat koko oppijajoukon tappiofunktion negatiivisen gradientin kanssa mahdollisimman paljon. [2].

## 2.1 Funktion estimointi

Tarkastellaan valvotun oppimisen kannalta ongelmaa, jossa tulisi arvioida funktiota ja sen ennustuskykyä. Kun käytettävissä on datajoukko $(x, y)_{i=1}^{N}$, missä $x = (x_1, ..., x_n)$ merkitsevät syötemuuttujia ja $y$ puolestaan vastemuuttujaa. Tarkoituksena on rekonstruoida tuntematon funktionaalinen riippuvuus $f : x \to y$, estimaatista $\widehat{f}(x)$ siten, että valittu tappiofunktio $\Psi(y, f)$ minimoituu. [2].

$$\widehat{f}(x) = \arg\min_{f(x)} \Psi(y, f(x)) \tag{18}$$

Tässä vaiheessa ei tehdä oletuksia todellisen funktionaalisen riippuvuuden $f(x)$ tai estimaattifunktion muodosta. Jos estimointiongelma kirjoitetaan uudelleen odotusarvojen perusteella, tätä vastaava kaava minimoisi tappiofunktion vastemuuttujan $E_y(\Psi[y, f(x)])$, joka riippuu havaitun datan syötemuuttujista $x$ seuraavasti:

$$\widehat{f}(x) = \arg\min_{f(x)} \underbrace{E_x[\overbrace{E_y(\Psi[y, f(x)])}^{\text{odotettu tappio } y} | x]}_{\text{koko datajoukon odotusarvo}} \tag{19}$$

Vastemuuttuja voi tulla eri arvojoukoista ja tämä luonnollisesti vaikuttaa tappiofunktion valintaan. Jos vastemuuttuja on binäärinen $y \in \{0, 1\}$, tappiofunktiona toimii jokin luokittelun tappiofunktioista. Jos taas vastemuuttuja saa reaaliarvoja $y \in \mathbb{R}$, käytetään klassista virheen neliön tappiofunktiota. [2]. Muitakin tappiofunktioita on mahdollista käyttää, kuten esimerkiksi

Huber-tappio, Laplace $L_1$ tappiota tai itse suunniteltua tappiota, mutta näiden valinta riippuu tutkijasta itsestään ja tässä tutkimuksessa keskitytään vain aikaisemmin esitettyihin tappiofunktioihin.

Tehdäksemme funktion estimoinnista käsiteltävissä olevan rajoitetaan funktion etsintäavaruus parametriseksi perheeksi funktioita $f(x, \theta)$. Näin funktion optimointi ongelma saadaan muutettua parametrin estimoinniksi:

$$\widehat{f}(x) = f(x, \widehat{\theta}),$$

$$\widehat{\theta} = \arg\min_{\theta} E_x[E_y(\Psi[y, f(x, \theta)])|x]. \tag{20}$$

Suljetussa muodossa olevat ratkaisut parametrien estimoinneille eivät ole tyypillisesti mahdollisia, joten estimointi tapahtuu iteratiivisten menetelmien avulla. [2].

## 2.2 Parametrien estimointi

Parametrien estimoinnille, määrälle $M$ iteraatioita, voidaan esittää summakaava [2]:

$$\widehat{\theta} = \sum_{i=1}^{M} \widehat{\theta}_i. \tag{21}$$

Yksinkertaisin ja eniten käytetty parametrin estimoinnin menetelmä on jyrkimmän laskun menetelmä (gradient descent). Kun käytettävissä on $N$ kappaletta datapisteitä $(x, y)_{i=1}^{N}$ tarkoituksena on vähentää empiiristä tappiofunktiota $J(\theta)$ tarkasteltavan datan suhteen. [2]:

$$J(\theta) = \sum_{i=1}^{N} \Psi(y_i, f(x_i, \widehat{\theta})). \tag{22}$$

Perinteisen jyrkimmän laskun optimointimenetelmä perustuu jatkuviin parannuksiin tappiofunktion gradientin $\nabla J(\theta)$ suuntaan. Koska parametrien estimointi $\widehat{\theta}$ on esitetty askeleittain, merkitään $\widehat{\theta}_t$ on estimaatti askeleella $t$. Yläindeksi $\widehat{\theta}^t$ merkitsee estimaattijoukkoa summana ensimmäisestä askeleesta askeleeseen $t$. Jyrkimmän laskun optimointimenetelmä toimii seuraavasti:

1. Alusta parametrien estimaatti $\widehat{\theta}_0$

2. Jokaiselle askeleelle $t$: hanki yhdistetty parametrin estimaatti $\widehat{\theta}^t$ kaikista edellisistä askeleista:

$$\widehat{\theta}^t = \sum_{i=0}^{t-1} \widehat{\theta}_i$$

3. Laske tappiofunktion gradientti $\nabla J(\theta)$, hankitulle parametrien estimaattisummalle:

$$\nabla J(\theta) = \{\nabla J(\theta_i)\} = \left[\frac{\partial J(\theta)}{\partial J(\theta_i)}\right]_{\theta=\widehat{\theta}^t} \tag{23}$$

4. Laske uusi parametrien estimaatti $\widehat{\theta}_t$ askeleelle $t$:

$$\widehat{\theta}_t \leftarrow -\nabla J(\theta) \tag{24}$$

5. Lisää uusi estimaatti $\widehat{\theta}_t$ summaan (21). [2].

## 2.3 Optimointi funktioavaruudessa

Suurin ero boosting menetelmien ja tavanomaisten koneoppimisen tekniikoiden välillä on se, että optimointi tapahtuu funktioavaruudessa. Tällöin funktion estimaatti $\widehat{f}$ parametrisoidaan muodossa:

$$\widehat{f}(x) = \widehat{f}^M(x) = \sum_{i=0}^{M} \widehat{f}_i(x). \tag{25}$$

Tässä $M$ on iteraatioiden määrä, $\widehat{f}_0$ alustava veikkaus ja $\{\widehat{f}_i\}_{i=1}^M$ ovat funktioiden lisäykset, joita kutsutaan myös "boostauksiksi". [2].

Jotta tämän ongelman lähestyminen funktioiden kannalta olisi toimiva käytännössä, eri funktioiden perheet parametrisoidaan. Seuraavaksi määritetään parametrisoidut "perusoppija" funktiot $h(x, \theta)$ erottaaksemme ne kokonaisista funktioiden estimaattijoukosta $\widehat{f}(x)$. Valitaan erilaisten perusoppijoiden joukosta jokin malli, tässä tapauksessa valintapuu. Laaditaan "ahne vaiheittainen" lähestymistapa funktion lisäyksiin perusoppijoilla. Tähän tarkoitukseen täytyy valita optimaalinen askelkoko $\rho$ jokaiselle iteraatiolle. [2]. Funktion estimaatille iteraatiolla $t$, optimointisääntö on täten:

$$\widehat{f}_t = \widehat{f}_{t-1} + \rho_t h(x, \theta_t) \text{ , missä} \tag{26}$$

$$(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \sum_{i=1}^{N} \Psi(y_i, \widehat{f}_{t-1}) + \rho h(x_i, \theta). \tag{27}$$

## 2.4 Gradient boost algoritmi

Tappiofunktion sekä perusoppijan valinta voi tapahtua mielivaltaisesti tarvittaessa. Käytettäessä mielivaltaista tappiofunktiota $\Psi(y, f)$ ja/tai perusoppijaa $h(x, \theta)$, parametrien estimaatin ratkaisuja on käytännössä vaikea löytää. Tästä johtuen määritellään uusi funktio $h(x, \theta_t)$, joka valitaan siten, että se on mahdollisimman yhdensuuntainen tutkittavan datan negatiivisen gradientin $\{g_t(x_i)\}_{i=1}^N$ kanssa. [2].

$$g_t(x) = E_y \left[ \frac{\partial \Psi(y, f(x))}{\partial f(x)} | x \right]_{f(x)=\widehat{f}^{t-1}(x)} \tag{28}$$

Sen sijaan, että etsittäisiin yleistä ratkaisua askeleittaiselle boostaukselle funktioavaruudessa, valitaan uuden funktion lisäys yksinkertaisesti eniten korreloivan $-g_t(x)$:n kanssa. Tämä tekee mahdolliseksi erittäin vaikean optimointiongelman (27) muuntamisen tutuksi pienimmän neliösumman minimoimiseksi:

$$(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \sum_{i=1}^{N} \left[ -g_t(x_i) + \rho h(x_i, \theta) \right]^2 . \tag{29}$$

Lopulta laaditaan gradient boosting algoritmi, jonka alun perin esitti Jerome H. Friedman vuonna 2001. Algoritmin tarkka muoto vastaavien kaavojen kanssa tulee riippumaan valituista tappiofunktiosta $\Psi(y, f)$ ja perusoppijasta $h(x, \theta)$ suuresti. [2].

**Algoritmi 3** Friedmanin Gradient Boost algoritmi

Alusta $\widehat{f}_0$ vakiolla
**for** $t = 1$ to $M$ **do**
Laske negatiivinen gradientti $g_t(x)$
Sovita uusi perusoppijan funktio $h(x, \theta)$
Etsi paras gradientin laskun askelkoko $\rho_t$:

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{N} \Psi \left[ y_i, \widehat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t) \right]$$

Päivitä funktion estimaatti: $\widehat{f}_t \leftarrow \widehat{f}_{t-1} + \rho_t h(x, \theta_t)$
**end for**

# 3 Knowledge intensive gradient boosting

Tässä teoriaosuuden viimeisessä luvussa käsitellään sovellettavan ja vertailtavan KiGB-menetelmän toiminta pääpiirteittäin. Menetelmä perustuu laadullisten rajoitteiden, nimenomaisesti monotonisten vaikutusten, käyttöön.

Lähdetään tarkastelemaan oppimisen ongelmaa hyödyntäen laadullisia rajoitteita, kuten monotonisuuksia ja synergioita eli yhteisvaikutuksia. Monissa oikean elämän osa-alueissa, esimerkiksi lääketieteessä tai logistiikassa, tällaisen tiedon käyttö on hyvin tyypillistä ja helppo saada. Esimerkiksi lääkäri voi neuvoa: "Kun A1C luku kasvaa, niin sydänkohtauksen riski kasvaa". Tai logistiikan asiantuntija puolestaan kertoo meille: "Kun lähtöpaikan ja määränpään välimatka kasvaa, niin kuljetuksen hinta kasvaa". Tämän kaltaisen neuvon tai *ohjeen* huomiotta jättäminen mallia opetettaessa ei ole tehokasta. Aikaisempia tutkimuksia asiantuntijan tiedon liittämisestä koneoppimisen malleihin on useita. Merkittävin hyöty kyseisen tiedon antamien rajoitteiden käytöstä on huomattu tapauksissa, joissa tarkasteltava data on vääristynyttä tai harvaa. Lisäksi suuriulotteisen datan tapauksessa koneoppimisen mallejen toiminta voi potentiaalisesti parantua. [3].

## 3.1 Laadulliset rajoitteet

Laadullisten rajoitteiden tarkoituksena on rajoittaa ja säädellä malliin syötetyn datan vaikuttamista itse opittavan mallin luomiseen. Laadullisten rajoitteiden ja muuttujien välisten vaikutusten käyttöä koneoppimisen malleissa on tehty jo vuodesta 1990 lähtien ja on tutkittu ja sovellettu laajalti eri aloilla, kuten esimerkiksi rahoitus-, asunto-, lääketiedealalla ja konenäön ongelmissa. Vallitsevin ja olennaisin laadullisen rajoitteiden tyyppi on monotonisuus. [3].

**Määritelmä 3.1** (Monotoninen vaikutus)**.** Jollain muuttujalla $x$ on monotoninen vaikutus jollekin muuttujalle $y$, jos muuttujan $x$ kasvu johtaa muuttujan $y$ arvon stokastiseen kasvuun (tai vähentymiseen). Tätä merkitään $x \prec^{Q+} y$ (tai $x \prec^{Q-} y$).

Monotonisten vaikutteiden on todistettu olevan tehokkaita puihin perustuvissa malleissa käytettyinä. Oikeassa elämässä tapahtuvien ilmiöiden mallintaminen tiukasti monotonisten funktioiden avulla on ongelmallista, sillä kaikkien vaikuttavien tekijöiden ja niiden välisten mahdollisten riippuvuuksien löytäminen on epäkäytännöllistä. Tarkastellaan kyseistä ongelmaa seuraavan esimerkin kautta. Tyypillisesti logistiikan alalla pitää paikkansa, että "lähtöpaikan ja määränpään välimatkan kasvaessa, kuljetuksen hinta kasvaa", silti voi kuljetuspalvelun tarjoaja veloittaa alhaisemman hinnan jois-

sain pitkän matkan kuljetuksissa. Syitä tähän voi olla useita, kuten kuljettaja on palaamassa yrityksen paikkakunnalle, sopiva pysäköintipaikka lähellä, seuraava aikataulutettu haku. Tästä johtuen esitetään heikosti monotoninen funktio, joka ottaa huomioon sekä datan että ohjeen. Tämä menettelytapa ei takaa monotonisuutta täysin, mutta kokeelliset tulokset osoittavat sen toimivan paremmin kuin vahvaa monotonisuutta hyödyntävät menetelmät. [3].

## 3.2 Menetelmän esittely

Knowledge intensive gradient boosting (KiGB) hyödyntää asiantuntijoilta saatuja laadullisia vaikutteita parantaakseen oppimista funktionaalisella gradient boosting -menetelmällä. Jokaisen iteraation aikana KiGB liittää ohjeen antamat vaikutteet boostauksen aikana. [3].

Tarkastellaan boostausta regression tapauksessa. Tällöin minimoitava tappiofunktio on yleisesti virheen neliö:

$$\arg\min_{\psi} \sum_{i=1}^{N} (y_i - \psi(x_i))^2. \tag{30}$$

Huomataan, ettei tällainen tavanomainen funktio ota huomioon asiantuntijan monotonisia vaikutteita eli ohjeita. Nämä vaikutteet voisivat mahdollisesti tuoda sellaista tietoa, joka puuttuu itse datasta. KiGB -menetelmä esittää tavan hyödyntää tätä informaatiota. Oletetaan, että puu $(\psi_t)$ haarautuu solmussa $n$ muuttujan $a$ perusteella. Jos on olemassa monotoninen vaikute $a \prec^{Q+} y$, niin solmun $n$ vasemman lapsisolmun odotetut arvot ($a_i \leq$ jakosääntö) tulisi olla oikean lapsisolmun odotettuja arvoja pienempiä ($a_i >$ jakosääntö). Toisin merkittynä $\mathbb{E}_{\psi_t}[n_L] \leq \mathbb{E}_{\psi_t}[n_R]$, missä $n_L$ on vasemmalle lapsisolmulle asetetut näytteet ja $n_R$ puolestaan oikealle lapsisolmulle asetetut näytteet solmun $n$ kohdalla. KiGB-menetelmä käyttää tätä oletusta *pehmeänä* rajoitteena lehtisolmujen tuottamille arvoille ja lisää rajoitteen rikkomisesta syntyvän sakon tappiofunktion virheeseen. Lisäksi on huomioitava asiantuntijan antamien ohjeiden paikkansapitävyys ja datassa esiintyvän vääristyneen datan määrä, eli ohjeisiin perustuvan rajoitteen sakon suuruus tulee olla säädettävissä. [3].

## 3.3 Menetelmän toimintatapa

Sisällytetään monotoniset vaikutteet ohjeiden rajoitteiden, jotka ovat muotoa $\mathbb{E}_{\psi_t}[n_L] - \mathbb{E}_{\psi_t}[n_R] \leq \varepsilon$, avulla. Tämä kuvaa $\varepsilon$-marginaali rajoitetta solmulle $n$

regressiopuussa, mikä takaa monotonisuuden toteutumisen huomioiden sietorajan $\varepsilon$. Seuraavana otetaan uusi apumuuttuja $\zeta_n = (\mathbb{E}_{\psi_t}[n_L] - \mathbb{E}_{\psi_t}[n_R] - \varepsilon)$ käyttöön, jotta voidaan mitata monotonisen rajoitteen rikkomista jokaisessa solmussa. Muokataan alkuperäistä neliön virheen tappiofunktiota siten, että se ottaa huomioon ohjeen rikkomisesta ($\zeta_n > 0$) syntyvän virheen ($(\zeta_n)^2$):

$$\arg\min_{\psi_t} \underbrace{\sum_{i=1}^{N}(y_i - \psi_t(x_i))^2}_{\text{tappiofunktio suhteessa dataan}} + \underbrace{\frac{\lambda}{2}\sum_{n\in N(x_c)} \max(\zeta_n \cdot |\zeta_n|, 0)}_{\text{tappiofunktio suhteessa ohjeeseen}}, \tag{31}$$

missä $N(x_c)$ on kaikkien testisolmujen ja latvasolmun, jotka haarautuvat monotonisen vaikutteen alla olevan muuttujan $x_c$ perusteella, joukko. Hyperparametri $\lambda$ on sakon kerroin, joka toimii rajoitteen tärkeyden mittana kyseisen mallintamisen ongelmassa. Sekä $\lambda$ että sietoraja $\varepsilon$ ovat mallin käyttäjän valittavissa. Tappiofunktio suhteessa rajoitteisiin on tyypiltään saranatappio ja se aktivoituu vain silloin, kun rajoite ($\zeta_n > 0$) rikotaan. [3].

Muokatun tappiofunktion avulla, hyperparametrit jokaiselle lehtisolmulle $l \in \psi_t$ voidaan esittää seuraavasti:

$$\psi_t^l(x) = \underbrace{\frac{1}{|l|}\sum_{i=1}^{N} y_i \cdot 1(x_i \in l)}_{\text{keskiarvo}} + \underbrace{\frac{\lambda}{2}\sum_{n\in N(x_c)} \mathbb{I}(\zeta_n > 0) \cdot \left(\frac{1(l \in n_R)}{|n_R|} - \frac{1(l \in n_L)}{|n_L|}\right)}_{\text{ohjeen rajoitteen rikkomisen virhe}}, \tag{32}$$

missä $1(l \in n_R)$ osoittaa sen kuuluuko lehti $l$ solmun $n$ oikealle lapselle vai ei, $|l|$ on näytteiden määrä lehtisolmussa $l$ ja $\mathbb{I}(\zeta_n > 0)$ rajoitteen tappiofunktio. [3].

Pehmeän rajoitteen rikottua solmulle $n$ lisätään sakko $\lambda \cdot \zeta_n$ korjauksena kaikkien kyseiseen solmuun liitettyihin lehtisolmuihin. Olennaista on huomata, että sakko, joka lisätään jokaiseen lapsisolmuun, on kääntäen verrannollinen näytteiden lukumäärään kyseisessä lapsisolmussa. Tämä tarkoittaa sitä, että ohjeen vaikutus pienenee, mitä enemmän dataa on käytettävissä. Puu rakennetaan ensin selvittämällä jakosäännön muuttuja suhteessa virheen neliön tappiofunktioon ja tämän jälkeen lehtien arvot päivitetään muutoksen mukaisesti. [3].

Kertoimen $\lambda$ korkeat arvot pakottavat muutokset agressiivisesti rajoitteiden pohjalta, kun rajoite rikotaan, ja samalla itse datan vaikutus vähenee. Vastaavasti pienet $\lambda$ arvot tekevät rajotteista löysempiä, jolloin puut riippuvat enemmän datasta. Tapauksessa $\lambda = 0$ käytössä on tavallinen valintapuun

oppimisen menetelmä, jonka toiminta perustuu täysin käytössä olevaan dataan. Negatiiviset $\varepsilon$ arvot noudattavat tiukkoja rajoitteen marginaaleja, kun taas positiiviset arvot sallivat päällekkäisiä marginaaleja. [3].

## 3.4 Algoritmi

Seuraavassa Algoritmissa 4 esitetään KiGB oppimisprosessi, kun kyseessä on regressio-ongelma. Prosessi aloitetaan alustamalla estimaatti keskiarvolla optimoidakseen virheen neliön keskiarvoa (MSE) ja iteratiivisesti lisäten mallin, joka on sovitettu datan ja ohjeisiin perustuvien pehmeiden rajoitteiden mukaan. Iteroivassa vaiheessa lasketaan funktionaalinen gradientti suhteessa dataan, sovitetaan puu laskettuun gradienttiin ottaen huomioon yhtälön (32) keskiarvon. Tämän jälkeen jokaiselle puun lehdelle (regressio) selvitetään rajoitteisiin liittyvät sakot (suhteessa monotonisiin muuttujiin $x_c$, kertoimeen $\lambda$ ja marginaaliin $\varepsilon$) ja lisätään ne lehtiin. Lopulta muodostettu puu lisätään nykyiseen malliin. Päivitetyt lehtien arvot ($\psi_m^l$) ohjaavat gradientteja ($\tilde{y}$) seuraavassa iteraatiossa ja auttavat nopeamman konvergoitumisen saavuttamisessa. Huomiona, vaikka algoritmi ottaa parametrina syötteen $M$, puiden lukumäärän, niin on mahdollista käyttää muitakin kriteereitä algoritmin päättämiselle, kuten mallien joukon rakentamisen todennäköisyydessä tapahtuvaa muutosta. Tällöin gradient boosting -algoritmin parametriton luonne voidaan säilyttää. [3].

**Algoritmi 4** Knowledge intensive gradient boosting regressiopuu [3]

Alusta estimaatti $\psi(x) = \psi_0(x) = \text{keskiarvo}(y)$
**for** $m = 1$ **to** $M$ **do**
  Laske gradientti $\tilde{y} = y - \psi(x)$
  Opi seuraava puu $\psi_m(x) = \text{puu}(\tilde{y}, x)$
  **for** $l$ **in** $\psi_m$ **do**
    $\psi_m^l(x) = \psi_m^l(x) + \text{sakko}^l(x_c, \lambda, \varepsilon)$
  **end for**
  Päivitä funktio $\psi(x) = \psi(x) + \psi_m(x)$
**end for**
Palauta $\psi(x)$

# 4 Sovellutus: Seosmetallien kristallisoitumislämpötilan ennustaminen

## 4.1 Alustus

Tässä luvussa käsitellään tutkielmassa käsiteltävän knowledge intensive gradient boosting -menetelmän soveltamista seosmetallien kristallisoitumislämpötilojen ennustamiseen eri jäähdytysnopeuksilla. Menetelmässä koulutetaan ja testataan rakennettua valintapuuta, jonka hyperparametrit on valittu siten, että ennustettujen arvojen virhe todellisiin arvoihin verrattuna olisi mahdollisimman pieni.

Metallurginen data koostuu 1272 rivistä eri seosmetallien (*Steel*) alkuaineiden osuuksia, käytetystä jäähdytysnopeudesta ($R$) sekä kristallisoitumislämpötilasta ($Texp$). Data on jaettu siten, että 20% datasta on valittu satunnaisesti testaamista varten ja loput 80% on kerätty kouluttamista varten. Yksi valintapuiden käytön hyödyistä on datan käyttö suoraan sellaisenaan puiden kouluttamisessa ja testaamisessa.

Ohjelmassa koulutetaan knowledge intensive gradient boosting -menetelmään (KiGB) perustuva valintapuu, jonka muodostamista voidaan ohjata annettuihin ohjeisiin perustuvien pehmeiden rajoitteiden avulla. Ohjeena on käytetty hiilen ja jäähdytysnopeuden käänteistä vaikutusta kristallisoitumislämpötilaan niiden kasvaessa, sillä näiden kahden muuttujan on katsottu vaikuttavan kyseiseen lämpötilaan eniten. Lisäksi ohjelmassa koulutetaan tavallinen gradient boosting -menetelmään (GB) perustuva valintapuu, jolloin sen testaamisessa ilmenevää virhettä voidaan verrata ohjeistetun valintapuun virheeseen.

Testaamisessa valintapuiden avulla ennustetaan kristallisoitumislämpötiloja aineiden osuuksien sekä jäähdytysnopeuden perusteella. Ennustettuja arvoja verrataan testausdatan oikeellisiin arvoihin ja muodostetaan molemmille valintapuille virhe näiden arvojen avulla. Tässä sovellutuksessa on käytetty virheen neliöiden keskiarvoa (mean squared error).

Kouluttamisen ja testaamisen jälkeen ohjeistetun valintapuun avulla muodostetaan annetulle seosmetallille kuvaaja, joka esittää sen kristallisoitumislämpötilaa jäähdytysnopeuksien funktiona. Ennustetut lämpötila-arvot eivät kuvaa lämpötila-arvoja täydellisesti, joten kuvaajaan on lisätty ei-lineaarinen regressiomalli ennustetuista lämpötila-arvoista eri jäähdytysnopeuksilla. Useiden seosmetallien tapauksissa jäähdytysnopeuden kasvu aiheuttaa kristallisoitumislämpötilan käänteisexponentiaalisen vähenemisen, jolloin ennustetut lämpötila-arvot noudattavat seuraavanlaista riippuvuutta jäähdytysnopeudesta.

$$y = a \cdot e^{-b \cdot (x-c)} + d. \tag{33}$$

Tässä $y$ on lämpötila ja $x$ jäähdytysnopeus. Puolestaan kertoimet $a$, $b$, $c$ ja $d$ valitaan parhaan mahdollisen regressiomallin sovittamiseksi ennustettujen arvojen välille. Voimme huomata, että jäähdytysnopeuden lähestyessä ääretöntä kristallisoitumislämpötila saavuttaa raja-arvon $d$. Näin muodostetun regressiomallin avulla on huomattavasti helpompi lukea lämpötila-arvo eri jäähdytysnopeuksien perusteella, minkä voi huomata Luvussa 4.5 olevan Kuva 8:n kuvaajasta.

## 4.2 Datan valmistelu

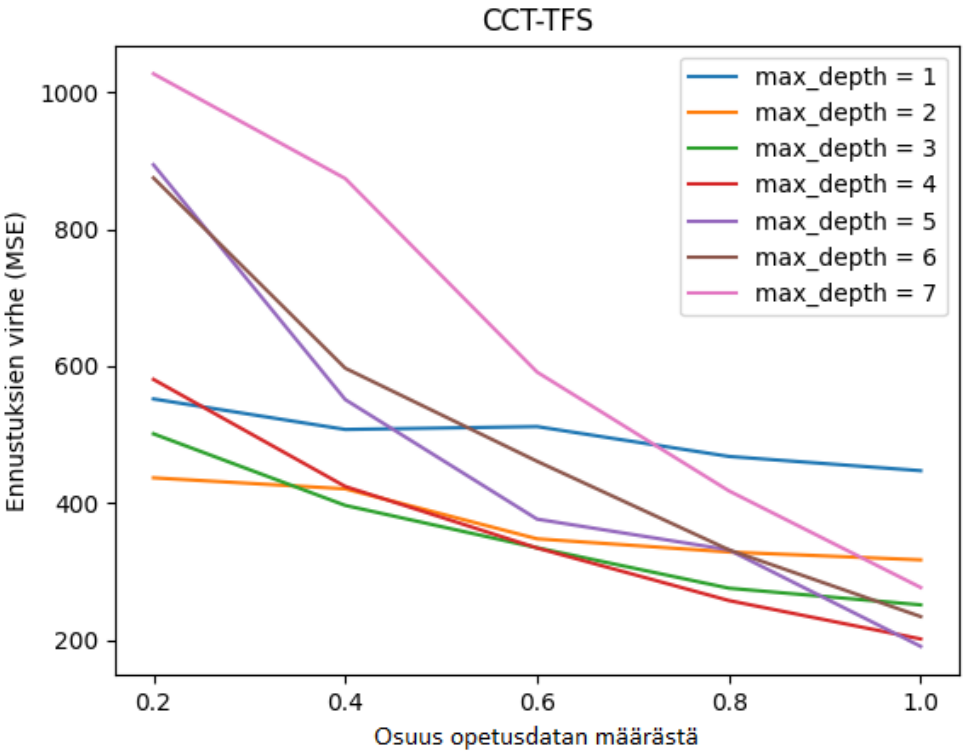
Datajoukon rivit koostuvat 13 sarakkeesta, joista 9 ovat eri metallien ja alkuaineiden osuuksia kyseisessä seosmetallissa. Yksi sarakkeista luokittelee koostumukseltaan erilaiset seosmetallit tietyllä koodilla (esimerkiksi, GA01, CM20). Sarakkeista löytyy myös jäähdytysnopeus ja kristallisoitumislämpötila.

Näistä sarakkeista kaksi on jätetty pois, joista ensimmäisenä on metallityypin luokittelusarake. Toisena pois jätettiin alkuaine boorin osuus metallissa, sillä sen osuus suurimmassa osassa dataa on olematon. Lopulta koulutusvaiheeseen kuuluvaan dataan kuuluu alkuaineiden hiili, kupari, mangaani, alumiini, kromi, nikkeli, pii ja molybdeeni osuudet sekä tutkittavan seosmetallin jäähdytysnopeus. Yhden sarakkeen merkitys datassa jäi tuntematto-maksi eikä sitä sen vuoksi otettu huomioon.

## 4.3 Hyperparametrien valinta

Hyperparametrien valinta on osa valintapuiden muodostamista, ja niiden säätäminen voi auttaa virheen pienentämisessä huomattavasti. Tämän tutkimuksen valintapuiden säädettävät hyperparametrit ovat: $max\_depth$, $learning\_rate$, $random\_state$, $\lambda$ ja $\varepsilon$. Parametri $max\_depth$ vaikuttaa puiden solmujen lukumäärään rajoittamalla peräkkäisten solmujen maksimimäärää eli puun syvyyttä. $Learning\_rate$ puolestaan kertoo kuinka vahvasti kouluttaminen tapahtuu ja vaikuttaa jokaisen puun virheeseen ja täten gradientin laskuvaiheeseen. Hyperparametri $random\_state$ tuottaa kokonaisluvullaan tietyn satunnaislukugeneraattorin, jonka mukaan puu alustetaan. Hyperparametri $\lambda$ on kerroin virheelle, mikä lisätään puulle, joka ei noudata annettuja ohjeiden pehmeitä rajoitteita solmujen kohdalla ja $\varepsilon$ on solmusta lähtevän vasemman ja oikean lehden ehtoarvojen erotuksen marginaali, mikä vaikuttaa rajoitteen rikkomisesta johtuvan virheen lisäämiseen itse virhefunktioon.

Tarkasteltaessa KiGB-menetelmän puun ennustamien arvojen virhettä eri puun syvyyksillä, $max\_depth$, opetusdatan määrän muuttujana, nähdään miten optimaalisen puun syvyyden valinnassa tulisi huomioida käytössä olevan opetusdatan määrä. Kuvassa 6 on havainnollistettu opetusdatan määrän vaikutus puun ennustamaan virheeseen eri puun syvyyksillä.




Kuva 6: Puun syvyyden vaikutus virheeseen opetusdatan mukaan

Tässä osuus 1.0 opetusdatan määrästä on 1018 kappaletta. Kuvaajasta nähdään, että opetusdatan määrän kasvaessa ennustettavien arvojen virhe laskee sitä voimakkaammin, mitä syvempää puuta käytetään. Toisaalta mitä syvempi puu, sitä monimutkaisempi ja siten myös alttiimpi se on suuremmille virheille.

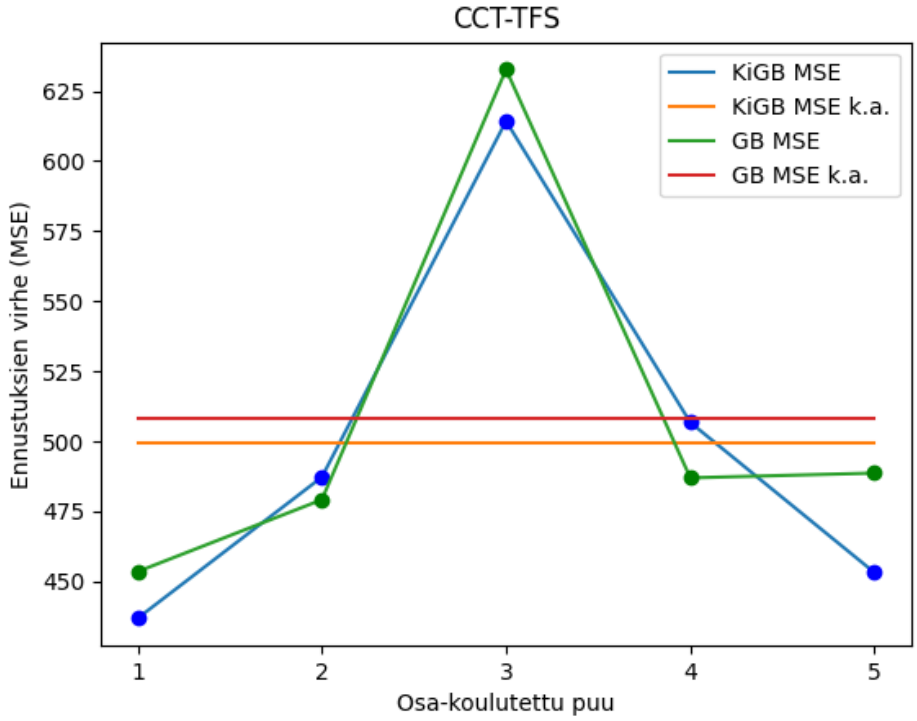
Kokeilemalla ja vertailemalla eri parametrin arvoja on lopulta saatu optimaalisten hyperparametrien arvoiksi puiden vertailua varten: $max\_depth = 2$; $learning\_rate = 0,3$; $random\_state = 15$; $\lambda = 1,2$ ja $\varepsilon = -4,8$.

Koulutettaessa ja ennustettaessa KiGB-menetelmän puulla eri seosmetallien kristallisoitumislämpötiloja jäähdytysnopeuden muuttujana, mistä ohjelma piirtää kuvaajan, on hyperparametrissa $max\_depth$ käytetty arvoa 5,

sillä kyseinen puu on koulutettu täydellä 80%:n opetusdatalla.

## 4.4 Gradient boosting -menetelmien vertailu

Koska valintapuiden kouluttamiseen vaikuttaa itse data, jolla niitä koulutetaan, on käytössä oleva 80%:n data jaettu viiteen yhtä suureen koulutuserään. Molemmat KiGB ja GB -menetelmän puut koulutetaan erikseen kaikilla näillä erillä, testataan kouluttamisen jälkeen ja talletetaan ennustevirheiden arvot. Näiden koulutuserien virheet on esitetty Kuvassa 7.

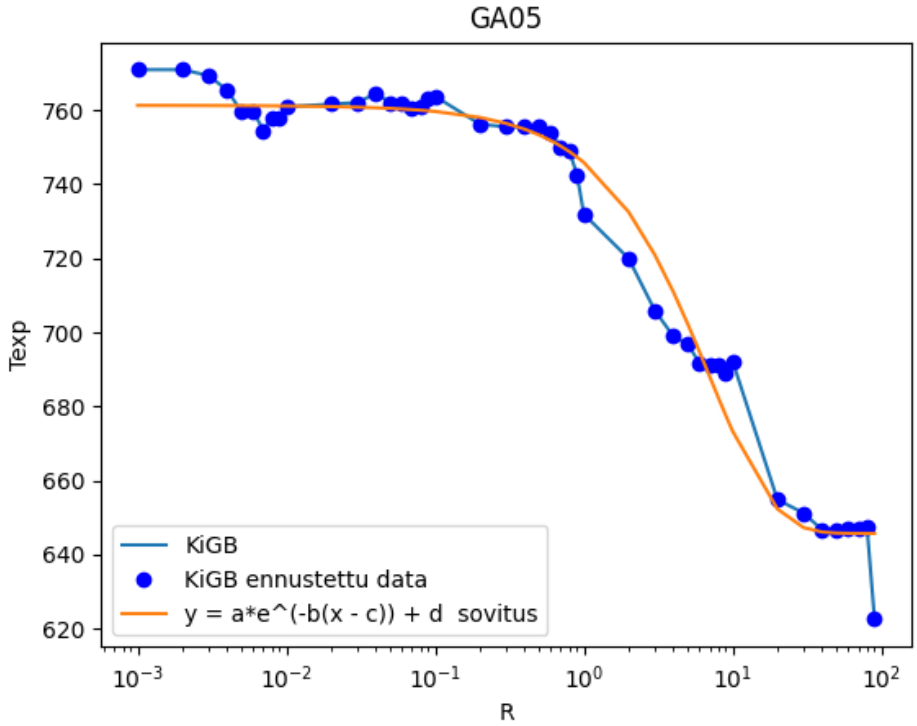



Kuva 7: Koulutettavien puiden vertailu

Suorat viivat kuvaavat niitä vastaavilla menetelmillä koulutettujen virheiden keskiarvoja. Kuvaajasta voidaan nähdä, että KiGB-menetelmän puu on keskimääräisesti ennustanut hieman paremmin verrattuna tavalliseen GB-menetelmän puuhun. Ohjelman tulostamat tarkemmat virheiden keskiarvot ovat KiGB: 499,7 ja GB: 508,2.

## 4.5 Kristallisoitumislämpötilan ennustaminen KiGB puulla

Ennustettaessa eri seosmetalleille kristallisoitumislämpötilaa jäähdytysnopeuden muuttujana on käytetty KiGB-menetelmän puuta ja sen kouluttamisessa koko 80%:n opetusdataa. Vertailun vuoksi on koulutettu myös tavallisen GB-menetelmän puu ja koulutettu sekin samalla opetusdatalla. Molemmat puut testattiin 20%:n testausdatalla ja virheiksi saatiin KiGB: 190,8 ja GB: 200,2. Lisäksi keskimääräisen absoluuttisten virheiden suhteelliset arvot ovat KiGB: 1,48% ja GB: 1,53%. Ohjelmaan voidaan syöttää seosmetallin tunnuskoodi, jolloin puiden kouluttamisen ja testaamisen lisäksi ohjelma ennustaa KiGB-menetelmän puulla kyseisen seosmetallin kristallisoitumislämpötilan eri jäähdytysnopeuksille (ks. Kuva 8).




Kuva 8: Erään seosmetallin (GA05) ennustetut kristallisoitumislämpötilat jäähdytysnopeuden muuttujana

Tämä kuvaaja on esimerkki yhden seosmetallin tapauksesta. Jäähdytysnopeutta edustavan vaaka-akselin asteikko on logaritminen, kun taas ennustettavan kristallisoitumislämpötilan asteikko on lineaarinen. Ennustusdatan

pisteisiin on sovitettu ei-lineaarisen regressiomallin käyrä, joka noudattaa luvun 4 yhtälön (33) riippuvuutta. Huomioon otettavaa on kuitenkin se, ettei kyseinen lämpötilan riippuvuus jäähdytysnopeudesta ole samanlainen jokaisella seosmetallilla.

# 5 Tulosten analysointi

Virhettä tarkastellessa on huomioitava, sen suuruuden johtuvan neliöön korottamisesta. Jos katsotaan aiemmin mainittuja suhteellisia virheitä, ovat ne kyseisen kristallisoitumislämpötilan (700-800 astetta) tarkastelun kannalta noin 11-12 astetta molempien puiden tapauksessa. Keskimääräinen virheen neliö kuitenkin osoittaa sen, että ohjeistettu valintapuu antaa hieman parempia tuloksia, kuin vertauksen kohteena oleva tavallinen valintapuu.

Ero näiden kahden eri menetelmän puilla ei ole suuri. Annetut ohjeet ja niihin perustuvat rajoitteet eivät parantaneet puun kykyä ennustaa merkittävästi, eikä siten kyseinen KiGB-menetelmä sovi sellaisenaan kyseiseen metallurgisen datan mallintamiseen. Syynä on rajoitteiden monotoninen vaikutus virheeseen, joka ohjaa puun rakentumista. Todellisuudessa, jollekin syötteelle annettu monotoninen rajoite ei pidä paikkaansa kaikilla näytepareilla, vaan näyteparin syötteiden muuttujien, esimerkiksi metallien osuuksien, arvot vaikuttavat keskenään toisiinsa ja siten myös lopulliseen kristallisoitumislämpötilan arvoon. Rajoitteista saatava hyöty tulee esille silloin, kun opetusdatan määrä on joko vähäistä tai puutteellista.

Valintapuiden tarkkuuteen vaikuttaa hyperparametrien lisäksi myös opetusdatan määrä - mitä enemmän opetusdataa käytetään, sitä tarkempia tuloksia ne kykenevät ennustamaan. Datan tulisi olla mahdollisimman monipuolista, jotta valintapuu kykenee ennustamaan kristallisoitumislämpötiloja monipuolisesti eri metallityyppien kohdalla. Toisaalta, jos koulutusvaiheessa käytetään liian paljon ja hyvin yksipuolista dataa voidaan törmätä "overfitting" -ilmiöön, jossa valintapuu kykenee ennustamaan vain opetusdataa vastaavia ulostuloarvoja eikä uusia opetusdatasta eroavan datan ulostuloarvoja, jolloin sen yleistämiskyky on heikko.

Vaikka koulutettujen valintapuiden ennustukset eivät antaisikaan riittävän tarkkoja arvoja, on niiden käyttö yksinkertaista ja kouluttaminen tapahtuu nopeasti. Ohjeistettua valintapuuta voitaisiin jatkossa verrata muihin olemassa oleviin valintapuihin kuin pelkästään tässä tutkimuksessa olevaan tavalliseen valintapuuhun.

# 6 Yhteenveto

Verrattaessa KiGB-menetelmän valintapuuta tavalliseen valintapuuhun tutkittavan datan mallintamisen yhteydessä, ei huomata merkittävää eroa menetelmien tarkkuuksien välillä. Kyseiseen aiempaan tunnettuun tietoon perustuvat puun rakentamista ohjaavat rajoitteet eivät täten paranna valintapuun mallin tarkkuutta kyseisen systeemin kohdalla. Tutkittavan algoritmin hyöty valintapuun yhteydessä tulee esille sitä näkyvämmin, mitä vähemmän dataa on käytettävissä. Valintapuiden soveltaminen kyseisessä tapauksessa ei sovi sellaisenaan mallintamiseen. Huomioon otettavaa on kuitenkin menetelmän moninaisuus mallintamisessa. Sitä voitaisiin käyttää esikäsittelyn työkaluna, joka analysoi datassa käytettävien muuttujien oleellisuuden suhteessa tarkasteltavaan ja mallinnettavaan muuttujan arvoon. Valintapuut ovat muokattavissa olevia mallintamisen menetelmiä, joten kyseistä algoritmia voitaisiin tulevaisuudessa muokata ennustamaan nykyistä tehokkaammin.

# Lähdeluettelo

[1] J. Arnaud (2017): *Exploiting random projections and sparsity with random forests and gradient boosting methods*. Väitöskirja, Liègen yliopisto.

[2] A. Natekin & A. Knoll (2013): *Gradient boosting machines, a tutorial*. Frontiers in Neurorobotics 7: 21.

[3] H. Kokel, P. Odom, S. Yang & S. Natarajan (2020): *A unified framework for knowledge intensive gradient boosting: Leveraging human experts for noisy sparse domains*. EAAI-20 Proceedings Vol. 34 No. 04: AAAI-20 Technical Tracks 4.