Tampereen yliopisto

Toni Grönberg

KATSAUS KONEOPPIMISEEN

Koneoppimismallin kehitysprosessi

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Toukokuu 2023

TIIVISTELMÄ

Toni Grönberg: Katsaus koneoppimiseen
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2023

Koneoppimismallin kehittämisen tavoitteena on rakentaa matemaattinen malli, joka kykenee edistykselliseen ongelmanratkaisuun ja pystyy tekemään tarkkoja päätelmiä uuden tai tuntemattoman datan perusteella. Tässä tutkielmassa esitellään koneoppimismallin kehitysprosessin eri vaiheet datan siivoamisesta mallin arviointiin. Tutkielman tavoitteena on tutustua koneoppimismalli kehitysprosessiin ja selvittää, kuinka koneoppimismalli koulutetaan.

Työ on kirjallisuuskatsaus, joka jakautuu kahteen osaan. Ensimmäisessä osassa esitellään tutkielman kannalta oleellisia koneoppimisen algoritmeja, sekä koneoppimismallin eri tyyppejä. Osan tarkoitus on esitellä tyypit, joihin koneoppimismallit yleisimmin jaotellaan, sekä tarkastella koneoppimismallien tunnetuimpia algoritmeja. Koneoppimismallit voidaan jaotella tyyppeihin monella eri tapaa, mutta yleisimmin mallit jaotellaan käytettävän datajoukon perusteella. Koneoppimisalgoritmit voidaan koneoppimismallin tapaa jaotella eri tyyppeihin tai luokkiin. Näistä luokista tarkastellaan tarkemmin tapausperustaisia algoritmeja. Tutkielman toisessa osassa perehdytään itse koneoppimismallin kehittämisen vaiheisiin suoraviivaisessa järjestyksessä.

Koneoppimismallin kehittäminen on monivaiheinen prosessi, jonka jokainen vaihe vaikuttaa lopullisen mallin tarkkuuteen ja tehokkuuteen. Tutkielmassa koneoppimismallin kehitysprosessi jaotellaan datan valmisteluun, mallin valintaan ja kouluttamiseen, sekä koulutetun mallin arviointiin ja optimointiin. Datanjoukon keruuta tutkielmassa ei käsitellä. Kerätty raaka datajoukko ei suoraan sovi käytettäväksi koneoppimismallia kouluttaessa, vaan raaka data tulee valmistella. Malli, jonka kouluttamiseen käytetään raakaa, jäsentämätöntä, tai muuten huonokuntoista dataa ei pysty tekemään tarkkoja päätelmiä, ja malli, joka käyttää datan ja ongelman tyypille sopimattomia algoritmeja, voi johtaa pitkiin prosessointi aikoihin ja resurssien tuhlaukseen. Työn lopussa on lyhyt yhteenveto koneoppimismallin kehitysprosessista.

Avainsanat: koneoppiminen, koneoppimismalli, ominaisuuksien suunnittelu, datan valmistelu

SISÄLLYSLUETTELO

1 Johdanto 1

2 Tutkimusmenetelmä 2

3 Koneoppimismallin tyypit ja algoritmit 2

3.1 Koneoppimismallien jaottelu 3

3.2 Ohjatun oppimisen algoritmit 3

3.2.1 K-lähimmän naapurin menetelmä 4

3.2.2 Tukivektorikone 6

3.3 Ohjaamattoman oppimisen algoritmit 7

3.3.1 K-keskiarvon ryvästysmenetelmä 8

4 Koneoppimismallin kehitysprosessi 9

4.1 Datan valmistelu 10

4.1.1 Datan puhdistaminen 11

4.1.2 Ominaisuuksien suunnittelu 12

4.2 Koneoppimismallin valinta ja kouluttaminen 13

4.2.1 Koneoppimismallin tyypin valinta 13

4.2.2 Koneoppimismallin kouluttaminen 13

4.3 Koneoppimismallin optimointi ja arviointi 14

5 Yhteenveto 15

Lähdeluettelo 16

1 Johdanto

Tekoälystä (artificial intelligence) on tullut yksi modernin yhteiskunnan suurimmista rakennuspalikoista. Kaikki nykyajan *älykkäät järjestelmät* (intelligent systems), kuten itseohjautuvat autot, kuvantunnistus, sekä asiantuntijajärjestelmät (expert systems) ovat riippuvaisia tekoälyn jatkuvasta kehityksestä. Viime aikoina, tekoälyllä varustetut chat-botit ja kuvan generointimallit, kuten ChatGPT, sekä DALE-2 ovat olleet jatkuvan keskustelun alla niin sosiaalisessa mediassa, kuin eri uutiskanavilla.

Koneoppiminen (machine learning) on yksi tekoälyn opinaloista (subfield) (Russell & Norvig, 2021), ja se tutkii tietokoneen oppimista. Koneoppimisen pääpiirteenä on ajatus tietokoneesta, joka ongelmakohtaista dataa tarkkailemalla oppii ratkaisemaan dataan liittyvät ongelmat. Yksinkertaistettuna, koneoppimisen tarkoituksena on opettaa tietokoneelle, kuinka ratkoa tietty ongelma annetusta syötteestä, ilman erillistä ohjelmointia (Fan ym., 2020; Janiesch ym., 2021).

Oletetaan että tarkoituksena on luoda järjestelmä, joka pystyy vastaamaan kysymykseen, onko kuvassa kissa vai koira. Tällaista järjestelmää kutsutaan malliksi, tai tarkemmin *koneoppimismalliksi* (machine learning model). Koneoppimismalli on prediktiivinen malli (predictive model), joka yhdistää *koneoppimisalgoritmin* (machine learning algoritghm) ja *oppimisdatajoukon* (learning dataset) (Fan ym., 2021). Koneoppimismallin kehitysprosessi (development process) on monivaiheinen prosessi, jonka avulla malli koulutetaan vastaamaan esitettyyn kysymykseen tai ongelmaan mahdollisimman tarkasti. Koulutettu malli pystyy siis antamaan ennusteita, vastauksia, ja suosituksia, jotka auttavat ratkaisemaan *vaativia ongelmia* (advanced problems) eri aloilla, kuten lääketieteessä ja rahoitusalalla (Baloglu ym., 2022; Fan ym., 2020; Janiesch ym., 2021).

Tämän tutkielman tavoitteena on tutkia koneoppimismallin kehitysprosessia ja samalla vastata tutkimuskysymykseen ”kuinka tekoäly koulutetaan koneoppimismallin avulla”. Tutkielman kolmannessa luvussa tarkastellaan koneoppimismallin ja -algoritmien jaottelua eri tyyppeihin (tai luokkiin) ja tutustua yleisimpiin koneoppimisalgoritmeihin. Luvussa neljä käydään läpi koneoppimismallin kehitysprosessi palapalalta, jakamalla se datan valmisteluun, mallin valintaan ja kouluttamiseen, sekä mallin optimointiin ja arviointiin. Viidennessä luvussa on lyhyt yhteen veto tutkielmasta.

2 Tutkimusmenetelmä

Tämä tutkielma on kirjallisuuskatsaus koneoppimiseen ja koneoppimismallin kehitysprosessiin. Kaikki tutkielmassa tarkasteltu aineisto sivuaa tutkielman aihepiiriä vaihtelevalla tarkkuudella. Teoksia on haettu pääosin Tampereen yliopiston Andor hakupalvelimen tietokannoista. Tarkasteltuja teoksia on täydennetty ProQuest, sekä IEEE Xplore tietokannoista. Aineiston haussa käytettiin hakusanoja: ”Machine learning” ja ”Artificial intelligence”. Hakusanoja täydennettiin myös hakusanoilla ”Applications”, ”Algorithm”, ja ”Feature engineering”. Tarkasteltuja teoksia on myös täydennetty tekemällä hakuja hakusanoilla löytyneiden teosten lähdeluetteloista.

Hakusanoilla löytyneitä lähteitä on karsittu rajaamalla lähteet englanninkielisiin kirjoihin, artikkeleihin, sekä konferenssi julkaisuihin, jotka käsittelevät koneoppimisprosessin eri vaiheita ja menetelmiä. Kirjallisuuskatsaukset, sekä teokset, jotka käsittelevät vain *syväoppimista* (deep learning) on rajattu pois hauista. Koska löytyneitä lähteitä oli vielä todella paljon, tarkennettiin rajausta rajaamalla lähteet vain viimeisen viiden vuoden aikana (2018–2022) tehtyihin julkaisuihin.

3 Koneoppimismallin tyypit ja algoritmit

Koneoppimismallin yksi olennaisimmista osista on mallin kouluttamisessa käytetyt koneoppimisalgoritmit. Algoritmien tarkoitus on havaita kuvioita (patterns) ja suhteita syötteeksi annetusta tietojoukosta, jolloin koulutettu malli voi tehdä tarkkoja päätelmiä tehtävätyyppiin perustuen (Janiesch ym., 2021). Esimerkiksi, jos mallin kouluttamiseen on käytetty kuvia kissoista ja koirista pystyy malli päättelemään (tietyllä tarkkuudella), onko sille syötteenä annetussa kuvassa kissa vai koira.

Mahdollisia algoritmeja koneoppimismallin kouluttamiselle on useita ja jokaista algoritmia voidaan käyttää monien eri ongelmien ratkaisemiseen. Koska algoritmien kirjo on niin suuri, jaetaan ne usein eri luokkiin tai tyyppeihin oppimistehtävän perusteella. Näitä luokkia ovat *regressiomallit* (regression models), *tapausperustaiset algoritmit* (instance-based algorithms), *päätöspuut* (decision trees), *Bayesin menetelmät* (Bayesian methods), ja *keinotekoiset neuroverkot* (artificial neural networks) (Janiesch ym., 2021). Kaikkien näiden eri luokkien tarkasteleminen ei olisi tämän tutkielman pituuden kannalta järkevää, joten tarkempaan tarkasteluun tutkielmassa on valittu vain tapausperustaiset algoritmit.

Suurin eroavaisuus koneoppimisalgoritmien välillä on niiden tapa löytää suhteita syöte datasta (Baloglu ym., 2022). Esimerkiksi, *tukivektorikone* (support vector machine) pyrkii luomaan erottelevan *hypertason* (hyperplane) dataluokkien välille (Ghosh ym.,

2019), kun taas *k-lähimmän naapurin menetelmä* (k-nearest neigbors) luokittelee datan datapisteiden lähimpien ”naapureiden” perusteella (Russell & Norvig, 2021).
Koneoppimismallin koulutukseen on olennaista valita käytettävät koneoppimisalgoritmit tarkasti, sillä algoritmien tehokkuus ja tarkkuus vaihtelee koulutusdatan määrän, rakenteen, sekä käytettävissä olevan laitteiston perusteella. Vaikka koneoppimismallin käytössä oleva algoritmi sopisi kyseisen ongelman ratkaisemiseen, sen tarkkuus voi kärsiä pienen datamäärän takia. Vastaavasti jos datajoukko on todella suuri, voi mallin koulutus ja tulosten saanti olla tehottoman hidasta (El Mrabet ym., 2021).

3.1 Koneoppimismallien jaottelu

Koneoppiminen ja samalla koneoppimismallit jaetaan yleisimmin kahteen eri tyyppiin käsiteltävän datajoukon perusteella. Nämä tyypit ovat *ohjattu oppiminen* (supervised learning), sekä *ohjaamaton oppiminen* (unsupervised learning) (Janiesch ym., 2021). Koneoppiminen voidaan jakaa datan perusteella myös *vahvistusoppimiseen* (reinforcement learning), sekä *puoliohjattuun oppimiseen* (semi-supervised learning) (Russell & Norvig, 2021), mutta näitä ei käsitellä tässä tutkielmassa. Käsiteltävä datajoukko ei kuitenkaan ole ainoa tapa luokitella koneoppimismalleja, vaan ne voidaan luokitella myös käytettävän algoritmin tai ratkaistavan *koneoppimisongelman* (machine learning problem) perusteella.

Jokaisella tyypillä on omat käyttötarkoituksensa. Ohjatun oppimisen malleja käytetään erilaisten ennusteiden ja päätelmien tekemiseen syötedatasta. Nämä päätelmät voiva olla hyvin triviaaleja kuten edellä mainittu päättely kissasta ja koirasta, tai paljon monipuolisempia, kuten aktiivisten käyttäjien määrän ennustamiseen alustalla tietyssä ajassa (Janiesch ym., 2021). Ohjaamattoman oppimisen malleja käytetään pääosin datanpisteiden ryhmittelyyn yhteisten ominaisuuksien perusteella, toisin sanoen *ryvästymiseen* (clustering), sekä *korkea dimensionaalisen datan* (high-dimensional data) projisointiin pienemmässä dimensionaalisuudessa, eli *dimensioiden vähentämiseen* (dimensionality reduction) (Janiesch ym., 2021). Seuraavaksi tarkastellaan tarkemmin ohjatun oppimisen mallien, sekä ohjaamattoman oppimisen mallien käyttämiä algoritmeja.

3.2 Ohjatun oppimisen algoritmit

Ohjatun oppimisen malleja käytetään, kun käytössä oleva data on merkittyä. *Merkitty data* (labeled data) sisältää merkintöjä datan sisällöstä ja oletetusta *tulosteesta* (output). Nämä merkinnät toimivat valmiina vastauksina syötteelle, joiden avulla kalibroidaan koneoppimismallin eri parametreja mallin koulutusprosessin aikana (Janiesch ym., 2021). Esimerkiksi kissa ja koira kuvista koostuvan datajoukon syötteen mukana on merkintä,

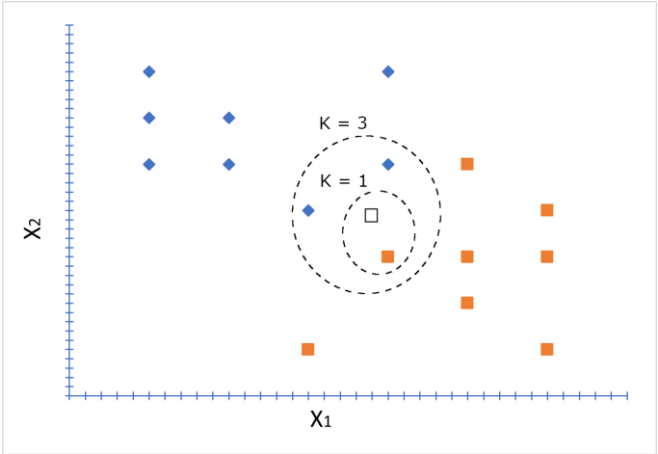
onko kuvassa kissa vai koira. Tätä datajoukkoa käyttämällä koulutettu malli pystyy myöhemmin tarkasti arvioimaan uusista tai merkitsemättömistä kuvista, onko kuvassa kissa vai koira.

Kuten aikaisemmin mainittiin, koneoppimismallit voidaan jakaa myös koneoppimisongelman perusteella. Näin ollen, ohjattu oppiminen mallit voidaan tarkemmin jakaa *regressiomalleihin* (regression models) ja *luokittelumalleihin* (classification models). Luokittelumallien oppimisongelman tuloste on rajallinen arvojoukko, kuten [kissa, koira] tai [positiivinen, negatiivinen]. Regressiomallin oppimisongelman tuloste on puolestaan numeerinen (Russell & Norvig, 2021). Ohjatun oppimisen algoritmeista tarkempaan tarkasteluun on valittu k-lähimmän naapurin menetelmä, sekä tukivektorikone.

3.2.1 K-lähimmän naapurin menetelmä

K-lähimmän naapurin menetelmä (KNN) on epäparametrinen (non-parametric) ja tapausperustainen oppimismenetelmä, jonka Cover ja Hart esittelivät ensimmäistä kertaa vuonna 1968 (Chen, 2020). KNN algoritmi on yksi yksinkertaisimmista ohjatun oppimisen algoritmeista ja sitä voidaan käyttää sekä luokittelu- että regressiomalleissa.

KNN algoritmin toiminta perustuu sen kykyyn määritellä luokka tai *ominaisuus arvo* (property value) syötteenä saadulle datapisteelle, vertailemalla datapisteen etäisyyttä sen *naapureista* (neighbors). Luokittelumalleissa datapisteelle etsitään sen K lähintä naapuria, joiden yleisin luokka (tai tuloste) määrittelee datapisteen luokan (Chen, 2020). Tilanteessa $K = 1$, datapiste saa saman luokan kuin sen lähin naapuri, ja vastaavasti tilanteessa $K = 3$, jos naapurit luokitellaan [kissa, kissa, koira], saa datapiste luokan kissa. Regressiomalleissa datapisteen arvoksi voidaan laskea sen K lähimmän naapurin keskiarvo, mediaani tai lineaarinen regressio (Russell & Norvig, 2021). Kuva 1 havainnollistaa kuinka K:n arvo muutos vaikuttaa datapisteen määrittelyyn. Jos $K = 1$ määritellään datapiste neliöksi, ja jos $K = 3$, määritellään datapiste ruuduksi.



Kuva 1. Havainnollistava kuva uuden datapisteen luokitteluun eri K:n arvoilla. Muuttujat X_1, X_2 esittävät pisteiden eri ominaisuuksia. Esimerkiksi korvien pituutta ja kynsien terävyyttä.

Oletetaan että käytössä on datajoukko, jossa on *N* määrä datapareja x ja y, missä y on x:n ominaisuus arvo tai luokka. Tällöin dataparit voidaan merkitä muodossa $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$. Oletetaan myös että x: *ll*ä on *M* määrän eri ominaisuuksia. Jos nämä ominaisuudet ovat samankaltaisia, kuten pituus ja paino, syötteen $\bar{x}$ arvon $\bar{y}$ määrittämiseksi, käytetään usein Euklidista etäisyyden kaavaa d_e laskemaan syötteen $\bar{x}$ ja datapisteiden x_N välinen etäisyys (Russell & Norvig, 2021).

$$d_e = \sqrt{\sum_{m=1}^{M} (\bar{x}_m - x_{nm})^2} \quad (1)$$

Euklidisen etäisyyden kaava (Chen, 2020)

Jos datapisteiden ominaisuudet x_m eroavat toisistaan, kuten ikä ja sukupuoli, Euklidisen etäisyyden sijasta käytetään usein Manhattan etäisyyden kaavaa d_m.

$$d_m = \sqrt{\sum_{m=1}^{M} |\bar{x}_m - x_{nm}|} \quad (2)$$

Manhattan etäisyyden kaava (Chen, 2020)

Edellä olevista kaavoista voidaan johtaa arvon $\bar{x}$ ja datapisteiden x_N välisen etäisyyden yleistetty kaava L^p, jota kutsutaan nimellä Minkowskin etäisyys tai L^p normi (Russell & Norvig, 2021). Minkowskin etäisyyden kaava, kun p = 1, vastaa Manhattan etäisyyttä, ja kun p = 2, vastaa kaava Euklidista etäisyyttä.

$$L^p(x_j, x_q) = \left(\sum_i |x_{j,i} - x_{q,i}|^p\right)^{1/p} \quad (3)$$

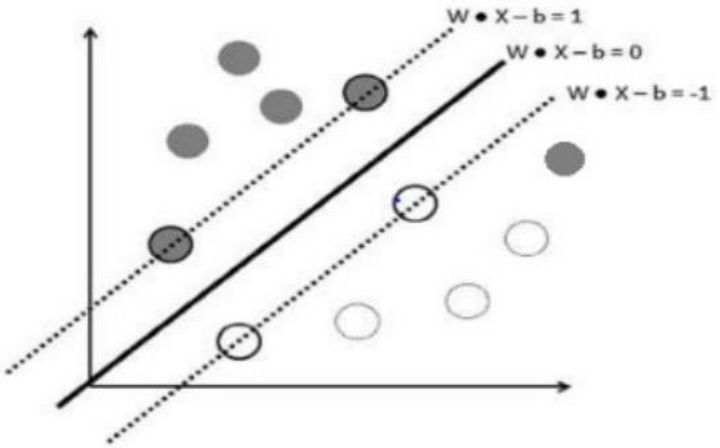
Minkowskin etäisyyden kaava (Russell & Norvig, 2021)

3.2.2 Tukivektorikone

Toinen tutkielmassa tarkasteltavista ohjatun oppimisen algoritmeista on tukivektorikone (SVM). 2000-luvun alussa tukivektorikoneilla varustetut koneoppimismallit olivat suosituimpia niin sanottuja *suoraan hyllyltä* (off-the-shelf) malleja (Russell & Norvig, 2021). Tukivektoreita on myös onnistuttu soveltamaan ohjaamattoman oppimisen malleissa (Ghosh ym., 2019)

SVM on epäparametrinen algoritmi, jota KNN algoritmin tapaan voidaan käyttää sekä luokittelu- että regressiomalleissa. Sen tarkoituksena on erotella datajoukko kahteen luokkaan luomalla hypertason näiden luokkien välille. Hypertaso on *päätösraja* (decision boundary), joka jakaa datapisteet kahteen eri luokkaan riippuen siitä, millä puolella hypertasoa piste sijaitsee. Hypertasoja luokkien välille on rajaton määrä, tämän takia tukivektorikoneen tarkoitus onkin maksimoida luokkien pisteiden välisen marginaalin suurus ja näin löytää optimaalinen hypertaso.

Hypertaso muodostetaan luokkien datapisteiden rajoille muodostettujen yhdensuuntaisten *tukivektorien* (support vektors) avulla. Tukivektorit ovat vektoreita dataluokkien rajoilla, jotka rajoittavat datapisteiden välisen marginaalin suuruutta ja samalla erottelevat eri luokkien datapisteet. Datapisteiden, ja näin ollen myös tukivektorien välinen marginaali on eri luokkien lähimpien pisteiden maksimaalinen etäisyys (Ghosh ym., 2019). Kuvassa 2 mustien ja valkoisten pisteiden välille on muodostettu hypertaso $\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$ tukivektorien $\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = \boldsymbol{1}$ ja $\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = \boldsymbol{-1}$ avulla.



Kuva 2. Havainnollistava kuva hypertason $\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$ muodostamisesta. Tukivektorit $\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = \boldsymbol{1}$ ja $\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = \boldsymbol{-1}$ muodostetaan datapisteiden rajalle, niin että ne erottelevat datapisteet eri luokkiin. Hypertaso muodostetaan niin että se on yhtä kaukana molemmista tukivektoreista (Ghosh ym., 2019).

Oletetaan että käytössä on datajoukko, jossa on *N* määrä datapareja muodossa $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, missä $y_N = \pm 1$ osoittaa datapisteen x_N luokan. Tällöin hypertaso dataluokkien välillä on muotoa:

$$wx + b = 0 \quad (4)$$

(Ghosh ym., 2019)

Oletetaan myös, että datapisteet ovat oikein luokiteltu, jos $wx + b \geq 0$, kun $y_N = 1$ ja jos $wx + b \leq 0$, kun $y_N = -1$. Saadaan yhteinen rajoite, joka on muotoa:

$$y_N(wx + b) \geq 0 \quad (5)$$

(Chen, 2020)

Etäisyys hypertason ja datapisteen x välillä on muotoa $\frac{|wx+b|}{\|w\|}$, ja marginaali on muotoa $\frac{2}{\|w\|}$ (Ghosh ym., 2019). Suurin mahdollinen marginaali voidaan löytää minimoimalla $\|w\|$. Russellin ja Norvigin (2021) mukaan, optimaalisen hypertason voi selvittää ratkaisemalla alla oleva neliöllinen ääriarvotehtävä (quadratic optimization task), jonka johtamista ei tässä tutkielmassa käydä.

$$\arg\max_{a} \sum_{j} a_j - \frac{1}{2} \sum_{j,k} a_j a_k y_j y_k (x_j \cdot x_k) \quad (6)$$

Duaali esitys optimaalisen hypertason löytämiseen (Russell & Norvig, 2021)

3.3 Ohjaamattoman oppimisen algoritmit

Ohjaamatonta oppimista käytetään, kun käytössä oleva data ei ole merkittyä. Toisin kuin merkitty data, *merkitsemätön data* (unlabeled data) ei sisällä entuudestaan tuttuja merkintöjä eikä tietoa oletetusta tulosteesta. Esimerkiksi, merkitsemätön data voi olla joukko datapisteitä eri ominaisuuksista, kuten korvien pituus ja kynsien terävyys, mutta näitä ominaisuuksia ei ole valmiiksi luokiteltu kissoiksi tai koiriksi. Näin ollen, ohjaamattoman oppimisen tavoitteena on kouluttaa malli, joka pystyy tällaisesta tuntemattomasta ja luokittelemattomasta datasta etsimään ja löytämään kuvioita ja rakenteita, joiden avulla data voidaan jakaa samankaltaisiin joukkoihin. Tarkastellaan seuraavaksi ohjaamattoman oppimisen algoritmia *k*-keskiarvon ryvästysmenetelmä.

3.3.1 K-keskiarvon ryvästysmenetelmä

Yksi tunnetuimmista ja käytetyimmistä ohjaamattoman oppimisen menetelmistä on *k-keskiarvon ryvästysmenetelmä* (k-means clustering method) (Sinaga & Yang, 2020). *K*-keskiarvon ryvästysmenetelmä on menetelmä, jonka avulla koulutusdata pyritään ryhmittelemään *K* ryppääseen niin, että jokainen datapiste kuuluu vain yhteen ryppäistä. Näiden ryppäiden tarkoitus on jaotella datapisteet niiden ominaisuuksien mukaan yhtenäisiin kokonaisuuksiin, jolloin samaan ryppääseen kuuluvat datapisteet ovat keskenään samankaltaisia (similar) ja muihin ryppäisiin verrattuna toisen kaltaisia (dissimilar). Esimerkiksi optimaalisessa tilanteessa, kissoista ja koirista koostuva data ryhmiteltäisiin ryppäisiin a_1 ja a_2, joista a_1 koostuu vain kissoista ja a_2 koostuu vain koirista.

K-keskiarvon ryvästysmenetelmä mittaa pisteiden samankaltaisuutta niiden etäisyyden perusteella. Aluksi datan joukkoon luodaan satunnaisesti *K* määrä *painopisteitä* (centroids). Tämän jälkeen lasketaan datapisteiden etäisyys kaikkiin painopisteisiin, jonka jälkeen näitä etäisyyksiä verrataan keskenään. Jos datapiste *x* on lähimpänä painopistettä *a*, kuuluu datapiste *x* ryppääseen *a*, $x \in a$. Etäisyyksien laskemiseen data- ja painopisteiden välillä käytetään usein euklidisen etäisyyden kaavaa (1).

Oletetaan että $X = \{x_1, x_2, \dots, x_n\}$ on datajoukko *d*-ulotteisessa Euklidisessa avaruudessa $\mathbb{R}^d$. Olkoon $A = \{a_1, a_2, \dots, a_c\}$ ryppäiden c painopisteet. Olkoon myös $z = [z_{ik}]_{n \times c}$, jossa z_{ik} on binäärinen muuttuja, joka osoittaa, kuuluuko datapiste x_i *k*:nneen ryppääseen, $k = 1, \dots, c$. Tällöin *k-keskiarvon kohdefunktiota* (k-means objective function) kuvastaa alla oleva yhtälö (7) (Sinaga & Yang, 2020).

$$J(z, A) = \Sigma_{i=1}^{n} \Sigma_{k=1}^{c} z_{ik} \|x_i - a_k\|^2 \quad (7)$$

K-keskiarvon kohdefunktio (Sinaga & Yang, 2020)

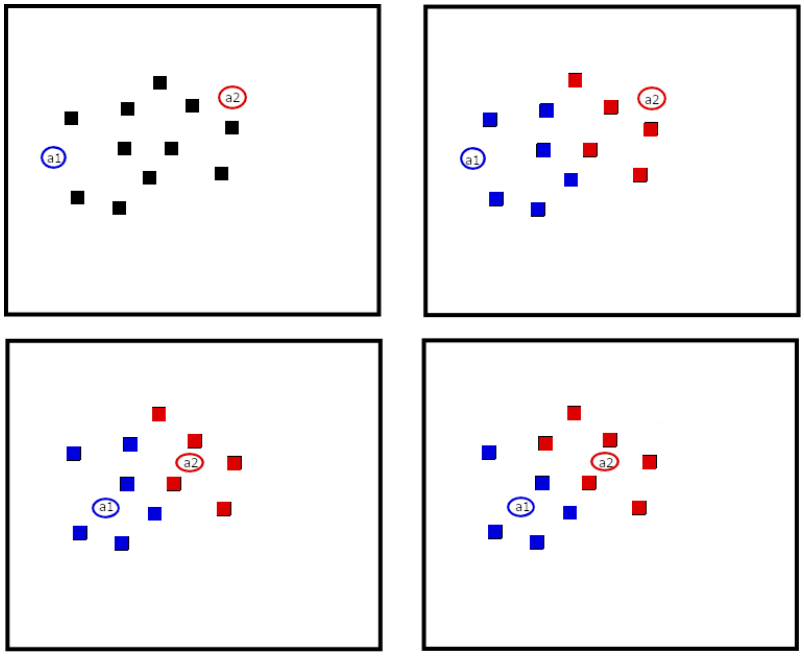
K-keskiarvon ryvästysmenetelmä koostuu päältä osin kahdesta vaiheesta: datapisteet ryhmitellään ryppäisiin lähimmän painopisteet perusteella, jonka jälkeen ryppäiden painopisteet päivitetään (Chen, 2020). Toistamalla edellä olevia vaiheita, kunnes painopisteet eivät enää muutu iteraatioiden välillä, saadaan *k*-keskiarvon kohdefunktio $J(z, A)$ minimoitua. Ryppäiden painopisteet, sekä datapisteiden ryhmät päivitetään seuraavasti:

$$a_k = \frac{\Sigma_{i=1}^{n} z_{ik} x_{ij}}{\Sigma_{i=1}^{n} z_{ik}} \quad (8)\ (Sinaga\ ym., 2020)$$

$$z_{ik} = \begin{cases} 1, jos\ \|x_i - a_k\|^2 = \min\limits_{1 \le k \le c} \|x_i - a_k\|^2 \\ 0, muuten. \end{cases} \quad (9)\ (Sinaga\ ym., 2020)$$

missä $\|x_i - a_k\|^2$ on euklidinen etäisyys datapisteen x_i ja painopisteen a_k välillä.

Kuvassa 3 hahmotellaan *k*-keskiarvon ryvästysmenetelmän painopisteiden päivityksen vaikuttamista ryppäiden koostumukseen. Alkutilanteessa *K* arvolla 2 luodaan kaksi satunnaista painopistettä a_1 ja a_2. Tämän jälkeen datapisteet ryhmitellään, ja painopisteet päivitetään. Alla olevasta kuvasta huomataan, kuinka painopisteiden päivityksen jälkeen data- ja painopisteiden etäisyydet muuttuvat, mikä kuvan tilanteessa johtaa joidenkin datapisteiden ryppäiden muuttamiseen.



Kuva 3. Havainnollistava kuva K-keskiarvon ryvästysmenetelmän eri iteraatiovaiheista.

Vaikka *k*-keskiarvon ryvästysmenetelmä kuuluu ohjaamattomaan oppimiseen, Sinagan ja Yangin (2020) mukaan tämä ei pidä täysin paikkaansa. *K*-keskiarvon ryvästysmenetelmän tarkkuus ja tehokkuus riippuu paljon menetelmän alustavista tekijöistä, sekä etukäteen määriteltyjen ryppäiden määrästä. Näin ollen, *k*-keskiarvon ryvästysmenetelmä ei ole täysin ohjaamatonta. Sinagan ja Yangin (2020) ehdottama *ohjaamaton k-keskiarvon ryvästys algoritmi* (Unsupervused k-means clustering algorithm) on yksi tapa ratkaista nämä riippuvuudet.

4 Koneoppimismallin kehitysprosessi

Koneoppimismallin kehittämisen tavoitteena on rakentaa malli, joka pystyy tarkasti ja tehokkaasti tekemään päätelmiä ja ennusteita annetun syötedatan perusteella. Näin ollen, mallin tarkoitus on ratkaista toimialakohtaisia ongelmia kohtuullisessa ajassa minimoiden virheelliset tulosteet. Nykyään, koneoppimismalleja on implementoitu onnistuneesti

useilla eri toimialoilla, kuten sähköisillä markkinoilla (Janiesch ym., 2021), ja lääketieteessä (Baloglu ym., 2022). Koska mallin käyttötarkoitus ja sille syötetty data ovat toimialakohtaisia, tulee se räätälöidä tilannekohtaisesti.

Koneoppimismallin kehittäminen koostuu useasta eri vaiheesta. Nämä vaiheet voidaan karkeasti erotella: *datan valmisteluun* (data preparation), *mallin valintaan ja kouluttamiseen* (model selection and training), sekä *mallin arviointi* (model assestment) (Janiesch ym., 2021; Russell & Norvig, 2021). Seuraavaksi tutustutaan tarkemmin koneoppimismallin kehitysprosessin erivaiheisiin, ja niiden vaikutuksiin lopullisessa mallissa.

4.1 Datan valmistelu

Käytössä olevan datan määrä, laatu ja tyyppi vaikuttavat suoraan koneoppimismallin laatuun. Mitä paremmin käytettävä data kuvastaa ratkaistavaa ongelmaa, sitä parempi koulutettu malli tulee olemaan (Ozdemir & Susarla, 2018; El Mrabet ym., 2021). Valmistelematon *raaka data* (raw data) ei usein ole käyttökelvollista koneoppimismallin kouluttamisessa. Raaka data usein sisältää *virheellisiä*, *poikkeavia* (outliers) tai *puuttuvia arvoja*, mitkä ovat haitallisia koneoppimismallia kouluttaessa. Samalla, raaka data voi olla hyvin *yksipuoleista* (biased), jolloin datajoukko esimerkiksi kissoista ja koirista, sisältäisi suurimmaksi osaksi vain kissoja. Jos malli koulutetaan tällaisella yksipuoleisella datalla, oppii se ennakkoluuloisesti vastaamaan jokaiseen kysymykseen kissa, sillä se on suurimman osan ajasta oikeassa.

Ozdemirin ja Susarlan (2018) mukaan suurin osa ajasta koneoppimimallin kehittämisestä kuluu datan valmisteluun. Data valmistellaan usein manuaalisesti, mikä on erittäin työläs, ja aikaavievä prosessi. Datan valmistelun tarkoitus on muuttaa käytössä oleva datajoukko koneoppimismallille sopivaan muotoon. Samalla datajoukkoa pyritään muokkaamaan niin, että se paremmin kuvastaa käsiteltävää ongelmaa. Näin valmistellulla datalla voidaan parantaa koulutetun koneoppimismallin tarkkuutta ja tehokkuutta.

Datanvalmistelu pitää sisällään datan esikäsittelyn: *datan keräämisestä* ja *puhdistamisesta* (data cleaning) *ominaisuuksien suunnitteluun* (feature engineering) (Ozdemir & Susarla, 2018; Fan ym., 2020). Datan puhdistaminen usein sisällytetään ominaisuuksien suunnitteluun, mutta tässä tutkielmassa ne ovat eroteltu selkeämmän kokonaiskuvan saamiseksi. Datan keräämistä ei käsitellä tässä tutkielmassa. Olettaen että koneoppimismallin kehittämisen käytettävä data on valmiiksi kerätty, tarkastellaan seuraavaksi datan puhdistamista ja ominaisuuksien suunnittelua.

4.1.1 Datan puhdistaminen

Kuten aikaisemmin todettiin, raaka datajoukko pitää usein sisällään poikkeavaa, puuttuvaa, tai muutoin sellaisenaan käyttökelvotonta dataa. Tällaisen datan muodostuminen voi johtua useista eri tekijöistä, kuten virheellisistä mittauksista tai vaurioituneesta laitteistosta. Datan puhdistamisen tarkoitus on muuttaa nämä ongelmalliset datapisteet koneoppimismallille hyödylliseen muotoon.

Puuttuva data on yksi yleisimmistä ongelmista datajoukossa. Usein tämä tarkoittaa sitä, että datapisteistä puuttuu jotakin oleellista, kuten potilas tietojen tapauksessa potilaan nimi. Puuttuva data on ongelmallista, sillä suurin osa koneoppisalgoritmeista ei kykene prosessoimaan datapisteitä, joista puuttuu arvoja (Ozdemir & Susarla, 2018). Yleisimpiä tapoja prosessoida puuttuva data on joko poistaa kaikki datapisteet joista puuttuu dataa, tai *imputoida* (impute) puuttuvat tiedot.

Puuttuvien datapisteiden poistaminen on tehokas tapa saada datajoukosta täydellinen (complete). Tällöin mistään datajoukon datapisteistä ei puutu arvoja, mikä estäisi koneoppimisalgoritmien toiminnan. Kuitenkin datapisteiden suora poistaminen pienentää datajoukon kokoa, millä voi olla negatiivisia vaikutuksia koneoppimismallin kouluttamisessa. Etenkin, jos suurimmasta osasta datajoukkoa puuttuu joitakin arvoja, ei datapisteiden poistamien välttämättä johda parhaimpaan lopputulokseen.

Toinen tapa ratkaista puuttuvat arvot on datan imputointi. Imputoinnin tarkoituksena on täyttää (fill in) puuttuvat arvot, jolloin kaikkia datapisteitä voidaan käyttää mallin kouluttamiseen. Puuttuvat arvot imputoidaan käyttäen apuna valmiina olevaa dataa. Imputoidessa usein käytetään valmiiden datapisteiden keskiarvoa, sillä 0 tai “null“ arvon käyttäminen voi vaikuttaa koulutetun mallin tarkkuuteen ei-toivotulla tavalla.

Toinen yleinen ongelma, mikä tulee ottaa huomioon dataa valmistellessa, on poikkeavat arvot. Monet koneoppimismallien algoritmit, kuten aiemmin esitelty *k*-lähimmän naapurin menetelmä, käsittelevät kaikkia datajoukon pisteitä samalla tavalla, mikä tekee niistä alttiita poikkeavalle datalle. Näin ollen, poikkeavilla datapisteillä on suuri vaikutus koneoppimismallin parametreihin (Russell & Norvig, 2021). Dataa tarkastellessa visuaalisesti, poikkeava data ilmenee yksittäisinä pisteinä kaukana muusta datajoukosta. Esimerkiksi tarkasteltaessa tuotteiden hintoja, poikkeavat arvot (tuotteet) esiintyvät paljon halvempana tai kalliimpana muiden joukossa.

Datajoukkoa *normalisoimalla* (normalization) tai *standardoimalla* (standardization), voidaan datajoukon datapisteet muuttaa muotoon, mikä vähentää poikkeavien arvojen vaikutusta. Tällainen datan käsittely voidaan suorittaa usealla eri tavalla. Esimerkiksi *minimi-maksimi normalisointi* (min-max normalization) muuntaa datapisteet välille [0, 1], kun taas *z-pisteen standardointi* (z-score standardization) muuntaa datapisteet niin, että niiden keskiarvo on nolla ja keskihajonta yksi (Ozdemir & Susarla, 2018). Poikkeavien arvojen muuttamisen lisäksi, datajoukko normalisointia ja standardointia voidaan käyttää

jos datapisteiden vaihteluväli on suuri. Näin muuntamalla datajoukko suppeampaan muotoo, pienennetään suurten vaihteluvälien vaikutusta koneoppimismallin kouluttamiseen.

4.1.2 Ominaisuuksien suunnittelu

Toisin kuin datan puhdistaminen, ominaisuuksien suunnittelu ei keskity virheellisen datan korjaamiseen, vaan datanjoukon *attribuuttien* (attribute) muuntamiseen *ominaisuuksiksi* (features), jotka paremmin kuvastavat käsiteltävää ongelmaa. Ominaisuudella tarkoitetaan kaikkia datan attribuutteja, joilla on hyödyllinen vaikutus koneoppimismallin kehitykselle (Ozdemir & Susarla, 2018). Koneoppismallin tehokkuus on riippuvainen tällaisista hyvin suunnitelluista ominaisuuksista. Näin ollen, ominaisuuksien suunnittelun tavoitteena on parantaa koneoppimismallin tarkkuutta ja tehokkuutta, poistamalla ja muuntamalla tarpeettomia muuttujia, sekä lisäämällä tarkentavia ominaisuuksia. Ozdemir ja Susarla (2018) erottelevat ominaisuuksien suunnittelun useaan eri kategoriaan, joista vain yleisimpiä käsitellään tässä tutkielmassa.

Yksi ominaisuuksien suunnittelun kategoria on *ominaisuuksien valinta* (feature selection). Kuten aikaisemmin on mainittu, raaka datajoukko usein sisältää paljon tarpeettomia ja haitallisia attribuutteja. Ominaisuuksien valinnan tarkoitus on poistaa tällainen merkityksetön data, jolla ei mallin kouluttamisen kannalta ole hyödyllistä vaikutusta. Toisin sanoen, Ominaisuuksien valinnalla pyritään poistamaan datajoukosta kaikki paitsi parhaimmat ominaisuudet.

Toinen ominaisuuksien suunnittelun kategoria on *ominaisuuksien muutos* (feature transformation). Ominaisuuksia muuttamalla pyritään löytämään uusia ja parempia ominaisuuksia. Ominaisuuksien muutos menetelmät yhdistelevät valmiita ominaisuuksia luodakseen uusia, rakenteellisesti erilaisia ominaisuuksia. Näin luodut uudet ominaisuudet ovat paljon tehokkaampia kuvailemaan datajoukkoa, kuin alkuperäiset ominaisuudet (Ozdemir & Susarla, 2018). Yleisimmät algoritmit ominaisuuksien muutokselle ovat *PCA* (Principal Componenet Analysis), sekä *LDA* (Linear Discriminant Analysis). PCA:lla pyritään sisäistämään datajoukon kaikki oleellinen data pienempään määrään attribuutteja, kun taas LDA:lla pyritään optimoimaan datajoukon luokkaerottelu *pienemmässä ulottuvuudessa* (lower dimension) (Ozdemir & Susarla, 2018).

Edellä mainitut ominaisuuksien suunnittelun menetelmät ovat käsin suoritettavia, ja näin työläitä ja aikaa vieviä. Artikkelissaan Janiesch ym. (2021) käsittelevät *syvänoppimisen algoritmien* (deep learning algorithms) tapaa automatisoida käsin tehtävä ominaisuuksien suunnittelu. Tällasta automatisoitua ominaisuuksien erottelemista kutsutaan *ominaisuuksien oppimiseksi* (feature learning).

Ominaisuuksien oppiminen, kuten käsin suoritettavan ominaisuuksien suunnittelun, tarkoitus on löytää datajoukosta hyödyllisiä ominaisuuksia. Ominaisuuksien

oppimismenetelmät etsivät automaattisesti datassa piileviä ominaisuuksia ja erottelee ne muusta datasta (Mirtaheri, & Shahbazian, 2022). Etuna manuaaliseen työhön, automatisoitu ominaisuuksien oppiminen mahdollista minimaalisen inhimillisen vaivan, sekä mahdollistaa sellaisten ominaisuuksien löytymisen, mitä ei manuaalisesti pysty erottelemaan (Janiesch ym., 2021; Ozdemir & Susarla, 2018).

4.2 Koneoppimismallin valinta ja kouluttaminen

Kun raaka data valmisteltu, on seuraavaksi valittava ja koulutettava koneoppimismalli. Mallin kouluttamista varten valmisteltu data tulee ensin erotella kahteen eri osajoukkoon: *opetusdataan* (training data) ja *testausdataan* (test data). Opetusdata pitää sisällään suurimman osan käytettävänä olevasta datasta. Opetusdataa, kuten nimestä jo huomaa, käytetään koneoppimismallin kouluttamiseen. Testausdata puolestaan sisältää opetusdatan ulkopuolelle jätetyn datan, jota käytetään koulutetun mallin testaamiseen. Koulutusdatasta voidaan edelleen eritellä *validointi datajoukko* (validation dataset), jonka avulla voidaan saada aikaisempi arvio mallin taidokkuudesta.

4.2.1 Koneoppimismallin tyypin valinta

Oikean tyyppisen koneoppimismallin ja algoritmien valinta on erittäin tärkeää. Kuten aikaisemmin on todettu, valintaan vaikuttaa niin ratkaistava ongelma, kuin käytössä oleva data. Jos valitut algoritmit eivät sovellu käsiteltävän ongelman ratkaisemiseen, tai ne eivät kykene käsittelemään käytössä olevaa dataa, ei minkään lainen optimointi taikka kouluttaminen auta koneoppimismallia antamaan tarkkoja päätelmiä.

Kuten Russell ja Norvig (2021) toteavat, koneoppimismallin tyyppin ja algoritmien valitsemiselle ei ole parasta tapaa. Koneoppimismalleja on monia erilasia ja ne tulee räätälöidä jokaiselle ongelmalle erikseen. Russell ja Norvig (2021) kuitenkin antavat monia eri suosituksia valintojen tekemiseen. Esimerkiksi päätöspuihin kuuluvaa *satunnaista metsää* (random forest) on hyvä käyttää luokittelumalleissa, jos käytettävän datajoukon uskotaan sisältävän paljon epäolennaisia luokkia. Ja epäparametrisia menetelmiä suositellaan käytettävän, kun käytössä on suuri datajoukko, eikä datan ominaisuuksia ole täsmällisesti eroteltu.

4.2.2 Koneoppimismallin kouluttaminen

Kun koneoppimismallin tyyppi on valittu, on seuraavana edessä mallin kouluttaminen. Koneoppimismallin kouluttaminen on iteratiivinen prosessi.

Ensimmäisenä tulee säätää koulutettavan mallin sisäiset *hyperparametrit* (hyperparameters). Hyperparametreiksi luokitellaan kaikki datan ulkopuolelle jäävä hienosäätö,

millä on vaikutusta koulutus prosessiin. Tällaisia ovat esimerkiksi *koulutus vaiheiden* (training step) määrä, *kerrosten* (layers) määrä neuroverkoissa, sekä erilaiset aloitusarvot valitulle algoritmille. Koulutus vaiheiden määrän tarkka valitseminen on tärkeää, sillä liian suuri määrä koulutus iteraatiota voi johtaa *ylisovittamiseen* (overfitting) ja liian pieni *alisovittamiseen* (underfitting) (Russell & Norvig, 2021)

Hyperparametrien säätämisen jälkeen alkaa koneoppimismallin koulutus. Koneoppimismallin koulutus on yksinkertaistettuna opetusdatan syöttämistä koneoppimisalgoritmille. Syöttämällä opetusdataa koneoppimisalgoritmi säätää sen sisäisiä parametreja ja näin oppii tekemään ennusteita ja päätelmiä samanlaisesta datasta. Jos opetusdatasta on eroteltuna validointi data, voidaan opetusdatan syöttämisen aikana syöttää mallille validointi dataa, ja tarkastella mallin sen hetkistä osaamista.

Kouluttamisvaiheen jälkeen, mallin tarkkuus testataan syöttämällä sille testausdataa. Toisin kuin koulutusdatan kanssa, testausdataa syöttäessä mallin parametreja ei enää muuteta, vaan testausdataa käytetään tarkastamaan mallin tarkkuus. Jos malli on ylisovitettu, kykenee malli tekemään oikeita päätelmiä vain opetusdatasta, mutta ei testausdatasta. Alisovitettu malli puolestaan ei kykene tekemään minkäänlaisia päätelmiä (Russell & Norvig, 2021). Testauksen jälkeen malli voidaan kouluttaa uudelleen ja hyperparametreja voidaan uudelleen säätää paremman tuloksen saavuttamiseksi. Testausta ei myöskään ole välttämätöntä suorittaa välittömästi koulutusvaiheen jälkeen, vaan jos koulutusvaiheen aikana huomataan tulosteiden olevan erittäin huonoja, voidaan hyperparametreja säätää ja koulutusvaihe toistaa ilman testausta.

4.3 Koneoppimismallin optimointi ja arviointi

Koneoppimismallin arviointi on tärkeässä roolissa lopulliseen mallin käyttöönottopäätöksen tekemiseen. Toisinkuin testaaminen, koneoppimismallin arviointiin ei käytetä testausdataa, vaan sitä verrataan muihin iteraatioihin mallista, tai jopa muihin koneoppimismalleihin. Esimerkiksi, jos nykyisen mallin kouluttamisessa on käytetty KNN algoritmia, voidaan sitä verrata malliin, joka on koulutettu käyttämällä SVM algoritmia, tai vaikka aiemmin mainittua satunnaisen metsän menetelmää. Vertailemalla nykyistä mallia itsensä ja muiden mallien kanssa, voidaan tehdä parempia päätelmiä mallin osaamisesta.

Koneoppimismallia arvioidessa, mallin osaamisen ja tehokkuuden lisäksi, huomioidaan resurssien käyttö, sekä mallin *tulkittavuus* (interpretability). Jos kaksi eri koneoppimismallia pystyy tekemään päätelmiä yhtä nopeasti ja tarkasti, on laskennallisesti kevyempi ja tulkittavampi malli parempi. Laskennalliset kustannukset ilmenevät koneoppimisessa muistin kulutuksen, sekä päätelmiin menevän ajan kautta (Janiesch ym., 2021). Mallin tulkittavuudella puolestaan tarkoitetaan sitä, kuinka helposti voidaan ymmärtää, miten malli päätyi kyseiseen päätelmään (Russell & Norvig, 2021).

Koneoppimismallia kouluttaessa sitä usein myös optimoidaan vertailemalla sitä johonkin *optimointi algoritmiin* (optimization algorithm). Kuten koneoppimisalgoritmeja, optimointialgoritmeja on monia erilaisia, joista jokainen sopii eri optimointiongelmien ratkaisemiseen (Mirtaheri & Shahbazian, 2022). Optimoinnin tarkoituksena on vähentää yli- ja alisovittamisen riskiä, sekä minimoida virheellisiä päätelmiä.

Optimointialgoritmin valinta riippuu käsiteltävästä koneoppimisongelmasta, sekä koneoppimismallin tyypistä. Ohjatun oppimisen mallien ylisovittamisen riskiä, voidaan pienentää *k-kertaista ristiinvalidoinnin* (k-fold cross-validation) avulla. Regressiomalleja voidaan optimoida mittaamalla estimointivirheitä, kuten *keskineliövirheen neliöjuuri* (root mean square error) ja *keskimääräinen absoluuttinen prosenttivirhe* (mean absolute percentage error). Luokittelumalleja puolestaan optimoidaan laskemalla oikeiden ja väärien päätelmien suhteita, kuten *tarkkuus* ja *F1-pisteet* (F1-score) (Janiesch ym., 2021)

5 Yhteenveto

Tässä tutkielmassa tarkasteltiin koneoppimismallin kehitysprosessia, sekä yleisimpiä koneoppimismallien käyttämiä algoritmeja. Tavoitteena oli selvittää, kuinka tekoälyllä varustetut järjestelmät koulutetaan koneoppimismallin avulla.

Koneoppimismallin kehitysprosessin alkaa jo datankeruu vaiheessa. Mallin ja sen sisältämien koneoppimisalgoritmien tehokkuus ja tarkkuus riippuu siitä, kuinka hyvin käytettävä datajoukko vastaa ratkaistavaa ongelmaa. Tutkielmassa todettiin, kuinka raaka ja käsittelemätön data sisältää paljon virheellistä ja mallin koulutuksen kannalta tarpeetonta dataa. Samalla esiteltiin erilaisia tapoja muuttaa tällainen raaka datajoukko koneoppimismallille suotuisampaan muotoon datan valmistelun avulla. Tutkielmassa todettiin myös, että datan valmistelun vievän suurimman osan mallin kehitysprosessiin käytetystä ajasta.

Koneoppimismalli tulee räätälöidä jokaiseen tilanteeseen erikseen parhaimman tarkkuuden ja tehokkuuden saavuttamiseksi. Mallin ytimessä on sen koulutus ja arviointi. Mallin kouluttaminen on iteratiivista datan syöttämistä ja hyperparametrien säätämistä siihen asti, kunnes malli pystyy vastaamaan esitettyyn kysymykseen halutulla tarkkuudella. Mallia arvioidessa sitä vertaillaan muihin kehitettyihin malleihin ja tarkastellaan sen osaamisen ja tehokkuuden lisäksi laskennallisia kustannuksia ja mallin tulkittavuutta.

Lähdeluettelo

Baloglu, O., Latifi, S. Q., & Nazha, A. (2022). What is machine learning? *Archives of Disease in Childhood*. *Education and Practice Edition, 107*(5), 386–388. https://doi.org/10.1136/archdischild-2020-319415

Chen K. -C. (2020). *Artificial Intelligence in Wireless Robotics.* River Publishers. https://doi.org/10.1201/9781003337256

El Mrabet, M. A., El Makkaoui, K., & Faize, A. (2021). Supervised Machine Learning: A Survey. *2021 4th International Conference on Advanced Communication Technologies and Networking (CommNet)*, 1–10. https://doi.org/10.1109/CommNet52204.2021.9641998

Fan, X., Iacob, M., Nicolae, M., & Dong, E. (2020). Machine learning basics with IBM data science experience. *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, CASCON 2017,* 340–342.

Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019). A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification. *2019 International Conference on Intelligent Sustainable Systems (ICISS),* 24–28. https://doi.org/10.1109/ISS1.2019.8908018

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets, 31*(3), 685–695. https://doi.org/10.1007/s12525-021-00475-2

Ozdemir, S., & Susarla, D. (2018). *Feature engineering made easy: identify unique features from your dataset in order to build powerful machine learning systems (1st ed.).* PACKT Publishing.

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence : a modern approach (4th edition.).* Pearson.

Mirtaheri, S.L. & Shahbazian, R. (2022). *Machine Learning: Theory to Applications.* CRC Press. https://doi.org/10.1201/9781003119258

Sinaga, K. P., & Yang, M.-S. (2020). Unsupervised K-Means Clustering Algorithm. *IEEE Access, 8,* 80716–80727. https://doi.org/10.1109/ACCESS.2020.2988796