Tampereen yliopisto

Joonas Salo

SÄHKÖISTEN TENTTIEN ESSEEKYSYMYSTEN AUTOMAATTINEN ARVIOIMINEN

Informaatioteknologia ja viestintä
Diplomityö
Marraskuu 2020

TIIVISTELMÄ

Joonas Salo: Sähköisten tenttien esseekysymysten automaattinen arvioiminen
Diplomityö
Tampereen yliopisto
Teknis-luonnontieteellinen
Marraskuu 2020

Tässä tutkimuksessa tavoitteena on etsiä vaihtoehtoisia arviointikeinoja sähköiseseti tehtäville tenteille. Nykypäivään mennessä suurin osa tenteistä tarkistetaan ja arvioidaan edelleen käsin. Sähköisiä tenttejä on otettu enenevässä määrin viime vuosina otettu käyttöön koulutuksen kaikilla asteilla ja niillä on monia hyötyjä paperisiin tentteihin nähden. Tenttien aikataulut voi olla joustavia, jolloin oppilas voi varata itselleen parhaan mahdollisen ajan ennakkoon päätetyltä aikaväliltä. Tentit eivät myöskään käytä paperia lainkaan, ja opetushenkilöstön ei tarvitse valvoa tenttimistilaisuuksia erikseen. Sähköisiä alustoja kuten esimerkiksi STACK-ympäristöä voidaan käyttää numeeristen laskutehtävien ja sanallisten täydennystehtävien automaattiseen arviointiin. Tämä jättää ainoastaan täysin sanalliset esseetehtävät ja matemaattiset todistustehtävät täysin käsin arvioitavaksi. Koska sähköisten tenttien vastaukset ovat joko tekstiä tai kuvia, ne voidaan aina muuttaa binääriseen muotoon, ja koneoppimisen yleistymisen ansiosta on mahdollista tuoda automatisointia arviointiprosessiin. Tarkoituksena ei siis ole automatisoida tenttien arviointia täysin, vaan antaa opettajan arvioida osa vastauksista jonka jälkeen vastaukset ja arvioinnit annetaan algoritmille opeteltavaksi. Tästä saadaan automaattinen malli, joka sitten arvioi loput tentit parhaansa mukaan.

Tutkimusta varten luotiin neljä lyhyttä kysymystä, jotka jotka kysyttiin eräässä ensimmäisen vuoden matematiikan kurssin tentissä. Opettaja arvioi tehtävät käsin, jonka jälkeen vastaukset ja pisteet annettiin neljälle eri algoritmille opeteltavaksi. Vastaukset vektorisoitiin ja niistä poistettiin välimerkit sekä isot alkukirjaimet, jonka jälkeen vastauksien sanoista muodostetaan sanasäkki, jossa yksi rivi vastaa yhtä vastausta ja yksi sarake vastaa yhtä sanaa. Täällä tavalla saatiin vastaukset puhtaasti numeeriseen muotoon. Tutkimuksessa käytettyjä algoritmejä oli neljä, joista ensimmäisessä käytettiin Latent Semantic Analysis -mallia yhdessä k:n lähimmän naapurin algoritmin kanssa. Toinen algoritmi hyödynsi syväoppimista, joka pyrkii mallintamaan ihmisen ajattelutapaa. Kolmas algoritmi oli satunnaismetsäalgoritmi, joka optimoitiin vertailemalla eri oppimisnopeuksia sekä päätöspuiden pituuksia. Neljännessä algortimissa käytettiin myös satunnaismetsäalgoritmia, mutta tällä kertaa se optimoitiin automaattisesti.

Tuloksia tarkastellessa nousi esille ongelma, joka vaikutti suuresti varsinkin sanavaraston suuruuteen. Suomen kielessä muodostetaan eri sanamuotoja liittämällä eri päätteitä sanojen loppuun, mikä paisuttaa sanasäkin kokoa turhaan. Konjunktiot, prepositiot ja muut apusanat voivat pasiuttaa sanasäkkiä myös. Koska otoksien koot olivat pienet ja pistemäärät eivät jakautuneet tasaisesti tehtävien sisäisesti. Tämä vaikutti varsinkin probabilististen mallien luokitteluvarmuuteen, sillä annettujen luokkien todennäköisyydet eivät olleet selkeästi suurimpia.

Avainsanat: sähköinen tentti, automaattinen arviointi, sähköinen arvointi, essee, koneoppiminen

ABSTRACT

Joonas Salo: Automatic Evaluation of Essay-based EXAM-questions
Master of Science Thesis
Tampere University
Sciences and Engineering
November 2020

The aim of this thesis is to search for alternatives to grading electronic exams by hand. Electronic exams are becoming more and more common in the school environment and it has many positive sides. They can be done on the student's own time, they save paper, and they free teachers from supervising the examinations. Still, the grading of these exams is largely done by hand by the teaching staff. STACK-environment offers fully automated numerical evaluation and grading for questions with multiple choice, algebraic and "fill in the blanks" -questions. This has left question types such as proofs and essays as the only ones requiring grading done by hand. With the emergence of machine learning, and because the answers of the exams are just bits of data, there is a possibility to introduce automation to the grading process. The end goal is not to replace human grading entirely but for the teacher to grade a portion of the exams and the machine then grades the rest based on a model it creates based on the teachers grading.

To test this, a set of short essay-type questions was devised for this purpose as part of an exam in a first-year mathematics course. The answers were gathered, graded by hand, and afterwards put through four different machine learning algorithms, which then created models to assign points to each answer. The first model used a combination of a natural language processing method called Latent Semantic Analysis and k-Nearest Neighbor. The second model used Deep Learning, which tries to mimic the human thought process. The third and fourth methods were different implementations of the random forest algorithm, one of which optimizes itself automatically and the other uses 5-fold cross-validation. The different models were then compared to each other by accuracy and partial accuracy, where the impact of assigning points incorrectly by one was decreased. The completion time was also an important comparison, since some programs are designed to be fast but are prone to overfitting, whereas some algorithms specialize in eliminating overfitting but are slower in comparison. The answers are vectorized in all models and then converted to a bag of words, where each word is assigned a row and each answer a column in a vocabulary matrix. The elements are the occurrences of each word in each different answer. This is done to transform string data into purely numerical data.

When analyzing the answers, a problem arises in the most common words in every question. Since the Finnish language is one that forms most of its grammar by using inflections on words, a lot of the most common words are different cases of the same word, which artificially bloats the model's vocabulary. Another problem regarding the vocabulary is stop words, which are common conjunctions, prepositions etc that just add noise to the data. The small sample sizes and the fact that the points were not evenly distributed also means that while the probabilistic models, i.e. the second, third and fourth models, have decent accuracies, the probabilities for the choices were not clearly above the other choices. This means that models are not exactly confident of their classifications.

Keywords: electronic exam, automatic evaluation, electronic evaluation, essay, machine learning

ALKUSANAT

Tämä diplomityö kirjoitettiin Tampereen Yliopiston Matematiikan laitokselle vuosien 2019-2020 aikana. Tutkimuksessa käytettyjen esseekysymysten suunnittelu alkoi heinäkuussa 2019 ja kysymyksistä koostunut tentti oli suoritettavana lokakuussa 2019 osana Insinöörimatematiikka 123 -kurssia. Kirjoitustyö alkoi joulukuussa 2019 ja loppui marraskuussa 2020. Diplomityön tekemisen ohella tein assistentin töitä Laskutuvassa ja työskentelin tutkimusapulaisena ensimmäisen vuoden matematiikan kurssien parissa.

Kiitos kaikille.

Tampereella, 9. marraskuuta 2020

Joonas Salo

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

D_p	yleistetty esitysmuoto etäisyydelle normeeratussa vektoriavaruudessa
L_{jl}	tappion määrä tappiomatriisin alkiossa jl
$P(A)$	tapahtumien A ja B pistetodennäköisyydet
$\mathbf{u}_i$	eri attribuutteihin jaettava datapiste
$\mathcal{C}_l$	luokkaan l luokiteltujen pisteiden joukko
θ_n	päätöspuun yksittäiseen solmuun sidottu parametri
$a(\mathbf{u}_i)$	pisteen $\mathbf{u}_i$ keskietäisyys sijaitsemansa klusterin muihin pisteisiin
$b(\mathbf{u}_i)$	pisteen $\mathbf{u}_i$ keskietäisyys lähimmän kluisterin pisteisiin
$b(\mathbf{u}_i, C)$	pisteen $\mathbf{u}_i$ keskietäisyys klusterin C pisteisiin
$d(x, y)$	käytetään matematiikassa merkitsemään metriikkaa
i,j	kokonaislukuindeksi
k	ristiinvalidoinnissa käytetty kertaluku
k	indeksi, joka määrittää tarkasteltavien naapuripisteiden määrän
k_l	luokkaan $\mathcal{C}_l$ kuuluvien pisteiden määrä k:n lähimmän naapurin joukossa
p	kokonaislukukerroin etäisyydelle
$p(\mathbf{u})$	pisteen $\mathbf{u}$, ja luokan $\mathcal{C}_l$ todennäköisyysjakaumat
$p(\mathbf{u}\|\mathcal{C}_l)$	pisteen u posterioritodennäköisyysjakauma luokan $\mathcal{C}_l$ ehdolla
$s(\mathbf{u}_i)$	pisteen $\mathbf{u}_i$ siluettiarvo
u_{ij}	datapisteen yksittäinen attribuutti
v_i	luokka, johon datapiste kuuluu
x_i	yleistetty merkintä x-koordinaatille
y_i	yleistetty merkintä y-koordinaatille
CAS	Computer Algebra System
HTML	hypertekstin merkintäkieli (engl. *Hypertext Markup Language*=
LSA	piilevän semantiikan analyysi (engl. *Latent Semantic Analysis*)
SEFI	Euroopan insinööritieteiden opetusyhteisö (ransk. *Société Européenne pour la Formation des Ingénieurs*)
TAU	Tampereen yliopisto (engl. *Tampere University*)

TUNI	Tampereen korkeakouluyhteisö (engl. *Tampere Universities*)
URL	verkkosivun osoite (engl. *Uniform Resource Locator*)

1 JOHDANTO

Sähköiset tentit kouluissa ja muissa oppilaitoksissa ovat olleet ajankohtaisia aiheita viime vuosina, ja esimerkiksi ylioppilaskirjoitukset siirtyivät ensimmäistä kertaa sähköiseen toteutukseen vuoden 2016 syksynä. Tuolloin kirjoitettiin ainoastaan reaaliaineista maantieto, filosofia ja saksa, jonka jälkeen aineita on lisätty myöhemmissä kirjoituksissa aina vuoden 2019 kevääseen asti, jolloin matematiikka muutettiin sähköiseksi [40]. Sähköisissä tenteissä käytettävät tekstintuotto-ohjelmat mahdollistavat tekstin rakenteen standardoimisen ja myös kuvien avulla on mahdollista vastata. Tällöin ei tule käsialaeroja oppilaiden välillä, mitä tulkittaessa arvostelijoilla voi olla vaikeuksia.

Vaikka tenttien digitalisoinnin myötä tenttijöiden tekstintuotto on helpottunut, joudutaan tehtävät suureksi osaksi edelleen tarkastamaan käsin. Monivalintakysymyksiä on pitkään voitu arvioida automaattisesti ja matematiikassa on ryhdytty käyttämään STACK-järjestelmää arvioimaan tehtäviä automaattisesti. Muiden tehtävätyyppien eli esimerkiksi essee- ja todistustehtävien arvioimiseen ei ole luotu automaattista järjestelmää. Tämän tutkimuksen tarkoituksena on tutkia, voiko ohjatun oppimisen algoritmeja kouluttaa luokittelemaan esseevastauksia oikeisiin luokkiin eli antamaan oikeat pistemäärät opettajan tekemän opetusarviointien avulla. Työssä käytetään neljää eri algoritmia vastausten luokitteluun, sekä klusterianalyysia selvittämään vastausten keskinäisiä rakenteellisia yhtäläisyyksiä.

Tutkimuksen suunnittelu aloitettiin heinäkuussa 2019 ja työn kirjoitus jatkui kesäkuuhun 2020 asti. Ensiksi suunniteltiin neljä arvioitavaa lyhyttä esseemuotoista tehtävää Insinöörimatematiikka 123 -kurssille, joka suoraan maisterivaiheessa aloittaneille opiskelijoille suunnattu ensimmäisen vuoden matematiikan kurssi. Tehtävät sisällytettiin kurssin 5. välikokeeseen, joka käsitteli matriiseja. Kaikki kysymykset arvosteltiin käsin kokonaislukuasteikolla pistein 0-3, jonka jälkeen vastaukset koottiin yhteen taulukkoon. Osalla vastauksista opetettiin algoritmit luomaan luokittelumallit, jotka luokittelivat jäljelle jääneen osan vastauksista. Jokaista tehtävää kohden tutkittiin algoritmien osumatarkkuutta tai tappion yhteismäärää, sekä algoritmien luokittelumatriisieja keskenään eri tehtävissä. Samalla myös pohdittiin mahdollisia syitä luokitteluvirheille ja millä keinoilla tuloksia voitaisiin parantaa.

Tuloksissa tutkittiin luokittelutarkkuuksia algoritmien välillä ja myös tehtävien sisäisiä luokittelujakaumia tutkittiin matriisien avulla. Luokittelutarkkuudet vaihtelivat suuresti algoritmien valillä eri tehtävissä pienimpien tarkkuuksien ollessa noin 40% ja suurimpien tarkkuuksien ollessa yli 70%. Luokittelumatriiseissa näkyi erityisesti niiden tehtävien, joiden

pistejakauma ei ole tasainen, kohdalla vääristynyt kuva luokittelun tarkkuudesta. Koska joissakin tehtävissä oli runsaasti nollan pisteen vastauksia tai kolmen pisteen vastauksia, niitä luokitellut malli ei välttämättä arvioi hyvin muiden pistemäärien vastauksia. Vastauksista yritettiin myös etsiä rakenteellisia yhtäläisyyksiä k-means-klusteroinnin avulla, ja vaikka paras klusterijako oli neljä klusteria, kaksi niistä koostui yksittäisistä pisteistä ja muiden klustereiden siluettiarvot eivät olleet kovin suuria, joten klusterit eivät olleet hyvin muodostuneita.

2 AIHEESEEN LIITTYVÄT KÄSITTEET JA MENETELMÄT

2.1 Sähköiset tentit

Teknologian vaikutus ihmiskunnan arkielämään kasvanut vuosi vuodelta enemmän, joten on ryhdytty tutkimaan, miten sitä voitaisiin hyödyntää opetuksen yhteydessä. Varsinkin tenteissä (ja kokeissa) on ryhdytty antamaan vaihtoehtoisia suoritustapoja perinteiselle paperitentille. Sähköisissä tenteissä on monia etuja paperitentteihin organisoinnin näkökulmasta. Perinteisesti paperitenteissä tenttitilaisuuksia on vain muutama ja niitä kaikkia varten täytyy varata tila, kun taas sähköisissä tenteissä oppilas varaa itse sopivan ajan suoritukselle, joka tehdään varta vasten sille tarkoitetussa huoneessa [10]. Sähköisissä tenteissä myöskään käsialaerot eivät tuota opettajille ongelmia, sillä vastaukset kirjoitetaan suurimmaksi osaksi tekstinkäsittelyohjelmilla. Tentit voivat sisältää myös automaatisesti arvioitavia tehtävätyyppejä kuten monivalintatehtäviä ja täydennystehtäviä, mikä vähentää opettajan työtä arvioimisessa. Lisäksi paperitenteissä on ekologinen haittapuoli, sillä tenttipaperit ja vastauspaperit täytyy laatia jokaiselle tenttiin osallistuvalle opiskelijalle. Paperimäärää lisää myös suttupaperit, joita jotkut opiskelijat käyttävät vastausten luonnosteluun. Sähköisissä tenteissä kaikki voidaan siirtää virtuaaliseen ympäristöön, joka ei kuluta kirjoitusmateriaaleja.

Yksi sähköisten tenttien suurimmista haasteistä nykyään on esseekysymysten arvioiminen. Ne joudutaan edelleen arvioimaan käsin siinä missä STACK- ja vastaavat järjestelmät ovat automatisoineet usean muun tehtävätyypin arvioinnin.

2.2 EXAM-järjestelmä

Moni Suomen yliopisto käyttää sähköisissä tenteissä apuna EXAM-järjestelmää, jossa ensin opettaja laatii tentin opettamalleen kurssille, jonka jälkeen opiskelija voi ilmoittautua kyseiseen tenttiin. Opiskelija voi järjestelmään kirjautuessaan katsoa ja muuttaa tenttiilmoittautumisiaan, hakea tenttejä ja ilmoittautua niihin, katsoa suoritettuja tenttejä ja niiden arvosteluja. Järjestelmä mahdollistaa myös tenttien tekemisen muissa korkeakouluissa kuin missä itse kurssi järjestetään. [10]

Tässä tutkimuksessa EXAM-järjestelmän käyttötarkoitus on antaa tenttitilaisuudessa tehtävänannon kautta linkki tenttijälle, joka ohjaa Älyoppi-sivustolle, jossa itse tentit sijaitsevat.

2.3 ÄlyOppi

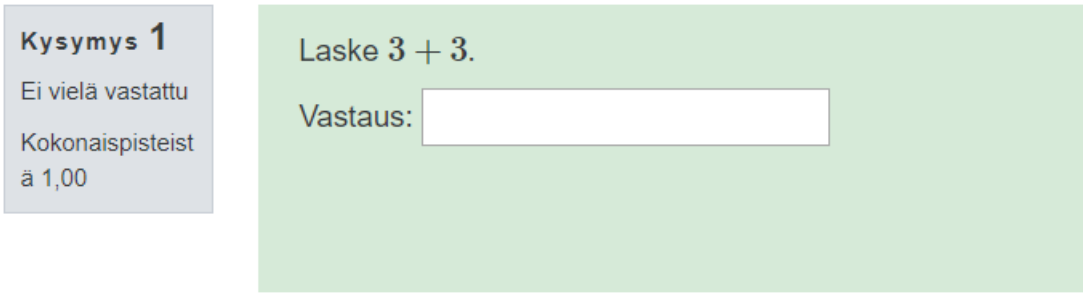
Älykkäät oppimisympäristöt ja niiden sisällöntuotanto -hanke on Suomen opetus- ja kulttuuriministeriön rahoittama sähköisten oppimisympäristöjen kehitys- ja tutkimushanke. Hankkeessa on mukana yksitoista eri korkeakoulua Suomesta ja koordinoijana toimii Aalto-yliopisto. Tarkoituksena on painottaa reaaliaikaisia vuorovaikutus- ja palautemahdollisuuksien osuutta oppimistapahtumassa. [22] [2]

Älyoppi-sivuston tarkoitus on toimia EXAM-järjestelmän kanssa sähköisten tenttien koontipaikkana. Sivustolle tehdään Moodle-sivuston tavoin kurssille oma sivu, jolle voi koota aiheittain halutun määrän tenttejä kuvan 2.1 mukaisesti.

Kuva 2.1. *Esimerkki kurssisivun yleisnäkymästä Älyoppi-sivustossa*

Tenteissä hyödynnetään pääosin STACK-tehtäviä, joissa tarkoituksena on luoda satunnaistettuja matemaattisia tehtäviä, joihin opiskelija tenttitilanteessa syöttää tehtävänantojen vaatimat vastaukset tekstikenttiin [28] [29]. Lopuksi järjestelmä tarkistaa automaattisesti vastauksen oikeellisuuden ja antaa pisteet suoritukselle. Kurssisivulle voi luoda myös muita aktiviteetteja, kuten palautuslaatikon sivuston ulkopuolella tehtävälle tehtävälle, ja aineistoja, kuten ladattavia tiedostoja tai linkkejä muille sivustoille, joita ei ole estetty palomuurilla. Kun jonkin tentin suoritusaika päättyy, voidaan sen opiskelijakohtaiset pistemäärät viedä csv-tiedostoon, josta ne voidaan siirtää sopivaan muotoon muutettuna esimerkiksi EXAM-järjestelmään. [11] [16]

STACK-tehtävien laatiminen tapahtuu suurimmaksi osaksi hyödyntämällä Maxima-kieltä tehtävän muuttujien laatimisessa. Muuttujat pyritään satunnaistamaan edes hieman, jolloin tehtävissä ei pitäisi pärjätä pelkällä ulkoa opettelemisella. [13]



Kuva 2.2. *Esimerkki yksinkertaisesta STACK-tehtävästä*

Kuvassa 2.2 tehtävä koostuu tehtävänannosta, joka on yhdistelmä puhdasta tekstiä ja matemaattista tekstiä, sekä vastauslaatikoista. Tehtävissä voi olla lisäksi ylimääräinen tekstikenttä, johon voi hahmotella omaa ratkaisuaan. Tässä työssä on Insinöörimatematiikka 123 -kurssin viidenteen välikokeeseen tehty lyhyet esseekysymykset, joiden vastauksia käytetään algoritmin opettamiseen.

2.4 MathCheck

Mathcheck on matematiikan tehtävien välivaiheiden tarkastamiseen tarkoitettu CAS (*Computer Algebra System*), joka on Antti Valmarin kehittämä ja tällä hetkellä vielä kehitysvaiheessa [36]. Opiskelijan on tarkoitus syöttää tehtävänannon vaatima vastaus, jolloin MathCheck tarkistaa vastauksen päättelyketjun yksi välivaihe kerrallaan ja ilmoittaa, missä kohtaa vastaus ja lähtökohta eivät enää täsmää. Normaalisti MathCheckin käyttöliittymä on HTML-sivu, johon vastaus syötetään opiskelijan toimesta, ja jos tehtävän laatijalla on enemmän kokemusta HTML:n käytöstä, voi hän muokata tehtävistä hyvinkin omalaatuisia [36].

Käyttöliittymä sisältää kolme osasta eli tehtävänannosta, vastauslaatikosta ja palautuspainikkeesta. Kun oppilas on palauttanut vastauksensa, MathCheck käy vastauksen läpi riveittäin, ja antaa tarvittaessa palautteen välivaiheista. Tarkoituksena on varmistaa, ovatko välivaihe matemaattisesti samanlaisia tehtävänannon kanssa, eli MathCheck joko ilmaoittaa vastauksen olevan oikein tai näyttää tarkalleen missä välivaiheessa on tapahtunut virhe. Palautetta voi saada virheellisten päätelmien lisäksi myös syntaksivirheistä. Jos halutaan korostaa vastausten sieventämistä tarpeeksi yksinkertaiseen muotoon, vastauksien enimmäismerkkimäärää on mahdollista rajoittaa vastauskentissä. Jos välitöntä palautetta ei haluta antaa palautuksen yhteydessä, voidaan palautuspainikkeen toimintaa muuttaa siten, että painike kirjoittaa vastauksen lähdekoodiin. Tämän jälkeen opettaja voi jälkeenpäin syöttää saamansa koodin MathCheckiin tarkastusta varten.

Vaikka MathCheckin käyttö pyritään tekemään mahdollisimman helpoksi sekä tehtävän laatijan että vastaajan puolesta, vaatii käyttö silti hieman käsitystä ohjelmoinnista samalla tavalla kuin STACK-tehtävissä. MathCheckin etuja on vaivattomuus, sillä käyttäjä tarvitsee ainoastaan palvelintilaa ja tekstinmuokkausohjelman tehdäkseen tehtäviä, eli erillisiä

ohjelmia ei tarvitse asentaa. Haittana voidaan pitää tehtävien tekemisen vaikeutta, jos käyttäjältä puuttuu kokemusta HTML:n käytöstä. Tällöin voi ottaa käyttöön MathCheckin tehtävien laadintaohjelman, jolla saa luotua yksinkertaisia mutta toimivia tehtäviä. Myös alustan runsas yhtäaikainen käyttö voi kuormittaa palvelinta, jolloin MathCheck voi toimia tavallista hitaammin, mikä on tavanomaista verkkoselaimen kautta käytettävissä ohjelmissa.

2.5 Sähköinen arviointi

Arvioinnista puhuttaessa yleensä tarkoitetaan tehtävien pisteytystä ja palautteen antamista tenttitilanteessa tai yleisesti harjoitustehtävien yhteydessä. Koska tämän tutkimuksen painotuksena on selvittää koneen kykyä pisteyttää tehtäviä ihmisen tavoin, voidaan sivuuttaa palautteen antaminen ja keskittyä vastauksen oikeellisuuden selvittämiseen ja pisteyttämiseen. Tehtävät voidaan tällöin jakaa kahteen ryhmään: tehtäviin joiden pisteytys on binäärinen eli vastaus on joko oikein tai väärin, ja moniosaisiin tehtäviin, jotka yleensä koostuvat useasta osasta ja siten antavat useamman pisteen oikeasta vastauksesta.

2.5.1 Monivalintatehtävät

Monivalintatehtävissä on annettu tehtävänannon lisäksi rajallinen määrä valmiiksi laadittuja vastauksia, joista tenttijä pyrkii valitsemaan oikean vastausvaihtoehdon. Arvioinnin kannalta tämä on yksinkertaisin tehtävätyyppi, sillä tarkastuksen automatisoimiseen riittää tarkistää, täsmääkö opiskelijan valitsema vastausvaihtoehto oikean vastausvaihtoehdon kanssa. Tällöin ei tarvitse ottaa huomioon ollenkaan vastausvaihtoehtojen sisältöä. Tehtävätyypistä saa myös monipisteisen tehtävän määrittämällä useita oikeita vastausvaihtoehtoja ja antamalla tenttjän valita useamman vaihtoehdon. Tehtävät on nopea tarkistaa ihmistarkastajan toimesta sekä vielä nopeampaa koneen toimesta. Jopa paperille laadittuja monivalintatehtäviä on helppo tarkistaa, sillä tehtävät voidaan skannata optisella lukijalla [5]. Lukemista varten on olemassa monia eri ohjelmia, joista osa on vapaasti ladattavissa [7].

2.5.2 Ohjelmointitehtävät

Ohjelmointitehtävien automaattisessa tarkastuksessa hyödynnetään tarkastusohjelmia, joiden käyttötarkoitus rajoittuu pelkästään ohjelman toimivuuden määrittämiseen. Ohjelman toimivuudella yleensä tarkoitetaan ohjelman antavan kaikilla mahdollisilla oikeilla syötteillä halutut lopputulokset ja väärillä syötteillä oikeat virheilmoitukset. Tällöin arvosteluasteikko rajoittuu binääriseksi "hyväksytty/hylätty-asteikoksi. Koodin tehokkuutta voidaan arvioida seuraamalla resurssien käyttöä, jolloin korkean pistemäärän saa mahdollisimman pienellä käytetyllä resurssimäärällä. Koodin työstettävyyttä on jo paljon vaikeampi arvioida koneellisesti, sillä se on täysin ihmisen määrittämä ominaisuus, joka vaatii

koodilta helppoa luettavuutta ihmiselle. Automaattisen järjestelmän on hyvin vaikeaa tietää, mikä on ihmiselle helppo lukea, ja tämän takia työstettävyyttä ei arvioda automaattisesti. Koneoppimisella on teoriassa mahdollista valvotun oppimisen avulla opettaa konetta arvostelemaan työstettävyyttä, mutta koodi ei koostu sanoista, jolloin tekstin erottelu järkevästi on vaikeampaa. [8]

2.5.3 Matematiikan tehtävät

Matemaattisissa tehtävissä käytetään useita erilaisia automaattisia tarkastusmenetelmiä riippuen tehtävien laadusta. Yleisin tarkastusmenetelmä on yksinkertainen vastauslaatikkoon annetun arvon tarkistaminen oikean vastauksen kanssa. Tämän johdosta tehtävän välivaiheiden olemassaololla ei ole merkitystä, vaan riittää että lopputulos on oikein. Matematiikan luonteesta johtuen saman arvon voi esittää useammalla tavalla, mutta useimmiten haetaan aina vastauksen yksinkertaisinta muotoa. Tällöin oikeasta vastauksesta, joka kuitenkin on väärää muotoa, ei välttämättä saa täysiä pisteitä. Laskutehtävissä pidetään myös ratkaisuun johtaneita välivaiheita tärkeinä pelkän ratkaisun lisäksi [41], joten automaattisessa tarkastuksessa ei tällöin saada kokonaisvaltaista kuvaa tenttijän osaamisesta. STACK-tehtävissä on myös mahdollista kiinnittää vastaus laskukaavaan yksittäisen arvon sijaan. Tämä mahdollistaa tehtävän alkuarvojen satunnaistamisen sekä funktioiden ja muiden matemaattisten merkintöjen käyttämisen tehtävien oikeina vastauksina. Matemaattista tekstiä onkin nykyään mahdollista tuottaa yhä useammassa kirjoitusohjelmassa, LaTeX-järjestelmää on mahdollista käyttää ja nykyään myös Matlab-ohjelmistoon on kehitetty matemaiikan kirjoittamiselle tarkoitettu osio [23]. Välivaiheiden ja sanallisten perustelujen arviointi on taas paljon monimutkaisempaa, ja sillä on tehtävän kannalta myös suurempi merkitys. Tavanomaisesti arvosteltavat välivaiheet ovat olleet ennalta määrättyjä [28], mutta nykyään on myös mahdollista arvioida välivaiheita, joita ei ole erikseen määrätty [36]. Tällöin pitää tarkistaa välivaiheiden oikeellisuus sekä lähtötilanteen suhteen, että lopputuloksen suhteen. Matemaattisten tehtävien tarkastuksen automatisoinnista on aiemmin tehty tutkimusta diplomityön muodossa, ja tätä työtä voidaan pitää jatkona samalle tutkimukselle [18].

2.5.4 Esseet ja sanalliset tehtävät

Esseetehtävien arvioinnin automatisoinnissa esiintyy muita tehtävätyyppejä enemmän kysymyksiä kuin moni muu tehtävätyyppi. Näistä ehkä selkeimpänä on ihmisen luovuuden tunnistaminen ja ymmärtäminen. Varsinkin pohtivissa esseissä, joissa ei ole selkeää yhtä oikeaa vastausta, vastaukset voivat poiketa toisistaan huomattavasti enemmän kuin muissa tehtävissä. Tämän vuoksi esseiden tarkistaminen on monimutkainen prosessi. Opettajat usein haluavat nähdä tekstien käsittelevän tietyt aiheet, jotta niille voidaan antaa täysiä pisteitä. Usein myös arvostelussa painotetaan tekstin luettavuutta, eli tekstin tyyli ja sujuvuus voi olla jopa sisältöä tärkeämpi [35]. Tämän takia saman pistemäärän vastaukset voivat olla täysin erilaisia keskenään.

Jo 1960-luvulla on yritetty tutkia esseetehtävien automaattista arviointia E. G. Pagen julkaiseman PEG-ohjelman myötä [25]. Tietokoneiden yleistyessä etenkin 1990-luvulla ajateltiin automaattisen tarkastuksen olevan tutkimisen ja kehittämisen arvoista, ja Pagen työ on luonut pohjan myöhemmin kehitettyjen arviointiohjelmien toimintaan. Nykyään onkin olemassa monia avoimella lähdekoodilla toimivia ohjelmia kaupallisella puolella, jotka hyödyntävät monia eri tarkastusmalleja [39].

Koska esseiden tarkastus voi painottua eri osa-alueille, tuo sen automatisoiminen omia haasteita. Tietokone ei välttämättä ymmärrä tavallisesta poikkeavaa tyyliä, oli se runollista tai selkokieltä. Eri kielet myös suoriutuvat erilaisesti varsinkin jos kielestä löytyy paljon eri sanamuotoja. Silti on onnistuttu kehittämään useita erilaisia tarkastusohjelmia, jotka antavat paljolti samoja arvosanoja kuin ihmistarkastaja. Ohjelmat perustuvat aina johonkin tiettyyn tarkastusmalliin, jonka tarkastustoiminta yleensä painottuu joko tyyliin tai sisältöön. Usein niiden pohjalla on jokin määrä valmiiksi tarkastettuja esseitä ihmisen toimesta. Määrä on yleensä kymmeniä ellei jopa satoja esseitä, jotta ohjelma saa luotua tarpeeksi hyvät arvosteluasteikot ja -kriteerit. [35]

3 TAUSTATEORIA

3.1 Matemaattinen tausta

Kun mitataan kahden pisteen välistä etäisyyttä, tavanomaisesti siihen käytetään Pythagoraan lausetta. Kaksiulottesisessa karteesisessa koordinaatistossa lause on muotoa

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.1}$$

ja tämä etäisyys tunnetaan euklidisena etäisyytenä. Tämä ei ole ainoa mahdollinen tapa mitata kahden pisteen etäisyyttä, koska sovelluskohteesta riippuen voi olla olemassa parempi metriikka. Kaikkien metriikkojen täytyy täyttää neljä ehtoa

1. $d(x, y) \geq 0$, eli etäisyys on aina ei-negatiivinen.
2. $d(x, x) = 0$, eli objektin etäisyys itseensä on aina nolla.
3. $d(x, y) = d(y, x)$, eli etäisyys on symmetrinen funktio.
4. $d(x, y) \leq d(x, z) + d(z, y)$, eli kahden pisteen etäisyys ei ikinä ole suurempi kuin kahden pisteen etäisyys kiertotien kautta, ts. etäisyys toteuttaa kolmioepäyhtälön.

Nämä ehdot täyttävää funktiota kutsutaan matematiikassa metriikaksi ja euklidinen etäisyys on osa suurempaa, yleistettyä metriikkaa. Sitä kutsutaan Minkowskin etäisyydeksi, ja se on muotoa

$$D_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}, \ p > 0,$$

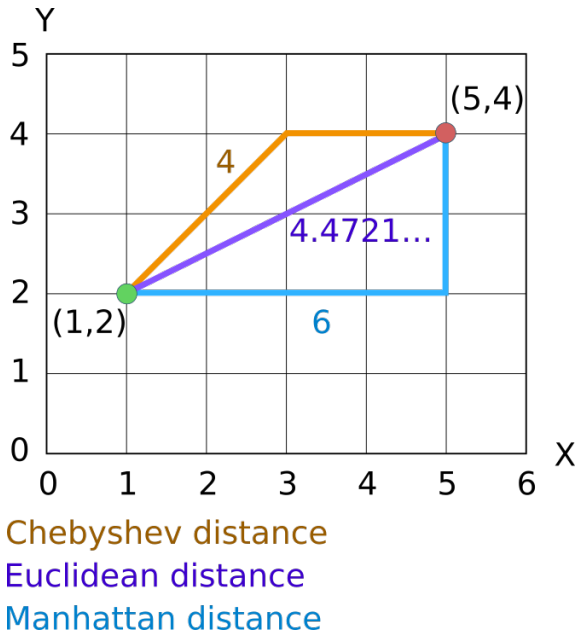
jossa vakion p arvoa muuttamalla määritetään mitä metriikkaa käytetään etäisyyden laskemiseen. Euklidisen etäisyyden p-arvo on 2, kun taas arvolla $p = 1$ saadaan Manhattan-etäisyytenä tunnettu metriikka, ja kun p lähestyy ääretöntä, saadaan Tchebyshevin etäisyytenä tunnettu metriikka.

Taulukko 3.1. *Joidenkin etäisyyksien matemaattisia määritelmiä*

Metriikka	Kaava	p
Manhattan	$D_1 = \sum_{i=1}^{n} \|x_i - y_i\|$	1
Euklidinen	$D_2 = \sqrt{\sum_{i=1}^{n} \|x_i - y_i\|^2}$	2
Tshebyshev	$D_\infty = \max(\|x_i - y_i\|), i = 1, 2, \ldots, n$	∞

Manhattan-etäisyyttä voidaan havainnollistaa käyttämällä ruudukkoa, jossa on neljä ete-

nemissuuntaa. Etäisyys on silloin suoraan koordinaattiakseleiden suuntiin liikuttujen pisteiden määrän summa. Etäisyyttä kutsutaan myös korttelietäisyydeksi, sillä kaupungissa kuten New Yorkin Manhattanissa, jossa on melkein täysin ruudukkomainen asemakaavoitus, voidaan laskea korttelien lukumäärällä etäisyyttä esimerkiski taksilla kuljettaessa. Tchebyshevin etäisyyden voi esittää käytännön maailmassa shakkilaudan ja kuninkaan avulla. Pisteen $\mathbf{x}$ ja pisteen $\mathbf{y}$ välinen etäisyys on sama kuin vähimmäismäärä siirtoja, jota kuningas tarvitsee siirtyäkseen kahden shakkiruudun välillä. Tämän vuoksi Tchebyshevin etäisyyttä kutsutaan myös shakkilautaetäisyydeksi. [30]



Kuva 3.1. *Kahden koordinaatin välinen etäisyys laskettuna eri etäisyyksillä*

Kuvassa 3.1 esitetyt etäisyydet voidaan laskea määrittelemällä alkupisteeksi $(x_1, y_1) = (1, 2)$ ja loppupisteeksi $(x_2, y_2) = (5, 4)$ ja sen jälkeen sijoittamalla taulukon 3.1 kaavoihin.

Taulukko 3.2. *Kuvan 3.1 pisteiden etäisyydet laskettuina*

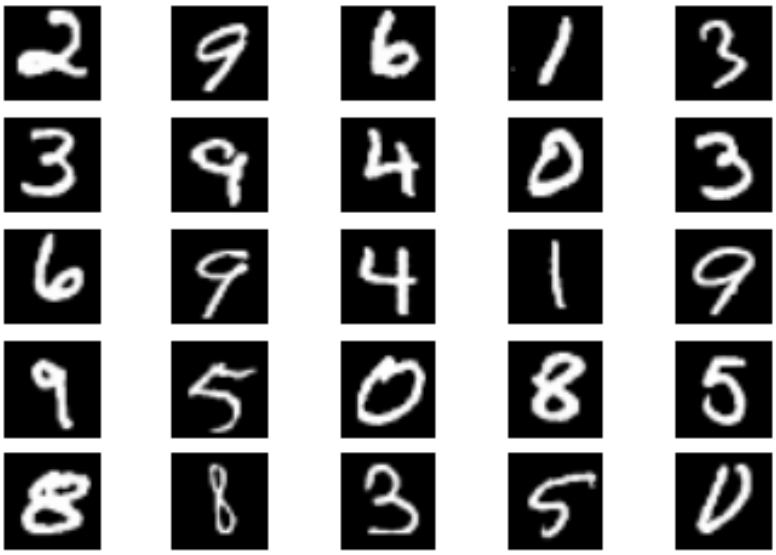
Metriikka	Kaava
Manhattan	$(5-1)+(4-2) = 4+2 = 6$
Euklidinen	$\sqrt{(5-1)^2+(4-2)^2} = \sqrt{16+4} = 2\sqrt{5} \approx 4.4721\ldots$
Tshebyshev	$\max\{(5-1),(4-2)\} = \max\{4,2\} = 4$

3.2 Ohjattu ja ohjaamaton oppiminen

Koneoppiminen on tieteenala, jossa tutkitaan ohjelmistojen kykyä suorittaa komentoja ilman, että ne erityisesti ohjelmoidaan tekemään kyseistä komentoa. Koneen syöte on joukko vektoreita $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$, jotka on jaettu mittausten tai havaintojen mukaan ryhmiin. Jokainen vektori on muotoa $\mathbf{u}_i = \langle u_{i1}, u_{i2}, \ldots, u_{im} \rangle$, jossa vektorin alkiolla u_{ij} on jokin arvo. Lisäksi ohjatussa oppimisessa (engl. *supervised learning*) vektoreilla $\mathbf{u}_i$ on jo-

kin luokka v_i. Tarkoituksena on syöttää N vektoria $\mathbf{u}_i$, missä $i = 1, \ldots, N$ yhdessä niiden luokkien v_i kanssa algoritmille ja luoda valitun algoritmin avulla malli, joka ennustaa uusille annetuille havainnoille $\mathbf{u}$ niiden luokat v. Kun luokat koostuvat yhdestä tai useammasta joukosta diskreettejä arvoja, puhutaan luokitteluongelmasta (engl. *classification problem*, ja jos luokat koostuvat yhdestä tai useammasta jatkuvasta arvosta, puhutaan regressiosta.[6, s.1, 137–138]

Yksi esimerkki hahmontunnistusongelmasta on käsin kirjoitettujen numeroiden tunnistus ja luokittelu, jossa pyritään opettamaan konetta tunnistamaan eri ihmisten kirjoittamia numeroita [21]. Koska ihmisten käsiala on niin monimuotoista, vaatisi tunnistusalgoritmin ohjelmoiminen suuren määrän koodia selvittääkseen kaikki erot eri käsialoissa. [6][15]



Kuva 3.2. *Esimerkkejä käsin kirjoitetuista numeroista [37]*

Kuvassa 3.2 nähdään vertailemalla samoja numeroita keskenään, kuinka erilaisia ne ovat. Koneoppimisessa ongelmaa jakamalla aineisto, tässä tapauksessa kuvat, kahteen osaan eli opetusdataan ja testidataan. Kaikille kuville on myös olemassa luokat, jotka vastaavat arabialaisia numeroita 0-9 [21].

Valvomattomassa oppimisessa (engl. *unsupervised learning*) vektorien $\mathbf{u}_i$ luokkia v_i ei tiedetä valmiiksi. Tällöin tarkoituksena ei ole ennustaa luokkia havainnoille vaan verrata havaintoja keskenään ja etsiä niistä samankaltaisuuksia ja sitä kautta jakaa niitä pienempiin joukkoihin, jolloin on kyse klusteroinnista (engl. *clustering*). Muita ohjaamattoman oppimisen käyttötarkoituksia on tiheysarviointi (engl. *density estimation*), jossa määritetään havaintojen levinneisyys syöteavaruudessa, ja korkean dimensioluvun omaavan datan projisoiminen kaksi- tai kolmiulotteiseen avaruuteen visualisaatiota varten. [6, s.3]

3.3 Tappiofunktio ja ristiinvalidointi

Luokitteluongelmissa ja regressiossa tavoitteena on saada luokiteltua havainnot mahdollisimman tarkasti oikeisiin luokkiin, mutta sen lisäksi oleellinen osa ongelmaa on mää-

rittää väärään luokkaan luokittelun seuraukset. Tällöin määritellään tappiofunktio (engl. *loss function*), joka antaa kaikille erilaisille virheellisille luokituksille arvot, ja niistä lasketaan lopussa kokonaistappio, joka pyritään minimoimaan. Jos jokin arvo $\mathbf{u}$, jonka oikea luokka on $\mathcal{C}_j$, luokitellaan luokkaan $\mathcal{C}_l$ ($j \neq l$), se tuottaa jonkin ennalta määritellyn määrän tappiota L_{jl}. Tämä voidaan tulkita olevan tappiomatriisin (engl. *loss matrix*) alkio j, l. Koska uuden havainnon luokkaa ei tunneta, joudutaan käyttämään oikean luokan sijaan yhteistodennäköisyysjakaumaa $p(\mathbf{u}, \mathcal{C}_l)$. Tämän jakauman suhteen lasketaan keskimääräinen tappio

$$E[L] = \sum_j \sum_l \int_{\mathcal{R}_j} L_{jl} p(\mathbf{u}, \mathcal{C}_l) d\mathbf{u}.$$

Kaikki havainnot $\mathbf{u}$ voidaan toisistaan riippumatta asettaa johonkin päätösalueeseen $\mathcal{R}_j$, jotka pyritään valitsemaan siten, että keskimääräinen tappio minimoituu. Tämä tarkoittaa, että jokaiselle havainnolle $\mathbf{u}$ tulisi minimoida

$$\sum_l L_{jl} p(\mathbf{u}, \mathcal{C}_l),$$

josta voidaan eliminoida tulosäännön $p(\mathbf{u}, \mathcal{C}_l) = p(\mathcal{C}_l|\mathbf{u})p(\mathbf{u})$ nojalla yhteinen tekijä $p(\mathbf{u})$. Minimi on helppo laskea, kun tiedetään posterioritodennäköisyys $p(\mathcal{C}_l|\mathbf{u})$. [6, s.39-42]

Ristiinvalidoinnissa pyritään parantamaan mallia jakamalla data kahteen osaan jo ennen algoritmin opettamista. Toista osaa kutsutaan opetusdataksi, ja se on selkeästi suurempi kuin jäljelle jäävä osa, jota kutsutaan testidataksi. Tällöin voidaan tarkistaa mallin suorituskyky antamalla sen ennustaa luokat testidatalle. K-kertaisessa ristiinvalidoinnissa data jaetaan k joukkoon, ja jokainen joukko otetaan vuorollaan testidataksi ja loput $k - 1$ joukkoa käytetään opetusdatana. Tämän jälkeen luodaan k mallia käyttäen kaikkia eri opetus- ja testidatayhdistelmää, ja tulokseksi voidaan ottaa keskiarvo mallien minimitappioiden keskiarvo. [6, s.32-33]

3.4 K:n lähimmän naapurin menetelmä

Ehdollisissa todenäköisyyksissä kahden tai useamman tapahtuman välillä hyödynnetään usein Bayesin teoreemaa, joka määrittää esimerkiksi kahden toisistaan riippuvan tapahtuman A ja B välisen suhteen

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad P(B) \neq 0,$$

missä $P(A)$ ja $P(B)$ ovat tapahtumien A ja B toteutumisten vastaavia marginaalitodennäköisyyksiä ja $P(A|B)$ on todennäköisyys tapahtuman A todennäköisyydelle ehdolla B. Vastaavasti $P(B|A)$ on tapahtuman B toteutumiselle ehdolla A. Todennäköisyyksiä $P(A)$ ja $P(B)$ kutsutaan myös prioritodennäköisyyksiksi, ja todennäköisyyksiä $P(A|B)$ ja $P(B|A)$ vastaavasti posterioritodennäköisyyksiksi. Luokitteluongelmissa teoreemaa käytetään määrittämään havaintojen $\mathbf{u}_j$ ja luokkien $\mathcal{C}_l$ välisiä suhteista ja k:n lähimmän naapurin menetelmässä (engl. *k-Nearest Neighbor algorithm*) luodaan pisteen $\mathbf{u}$ ympärille

pallo, joka sisältää k muuta pistettä. Olkoon N Pisteiden kokonaislukumäärä ja N_l luokkaan $\mathcal{C}_l$ kuuluvien pisteiden lukumäärä, jolloin $\sum_l N_l = N$. Oletetaan, että pistettä $\mathbf{u}$ ympäröivä pallo sisältää k_l pistettä ja on tilavuudeltaan V. Tällöin pisteen $\mathbf{u}$ ehdollinen todennäköisyysjakauma luokkaa $\mathcal{C}_l$ kohden on muotoa

$$p(\mathbf{u}|\mathcal{C}_l) = \frac{k_l}{N_l V},$$

pisteen $\mathbf{u}$ marginaalitiheys muotoa

$$p(\mathbf{u}) = \frac{k}{NV}$$

ja luokkien priorijakaumat muotoa

$$p(\mathcal{C}_l) = \frac{N_l}{N}.$$

Nyt voidaan esittää kunkin luokan posteriorijakauma Bayesin teoreeman avulla

$$p(\mathcal{C}_l|\mathbf{u}) = \frac{p(\mathbf{u}|\mathcal{C}_l)p(\mathcal{C}_l)}{p(\mathbf{u})} = \frac{k_l}{k}.$$

Pisteen $\mathbf{u}$ luokitteluvirheen minimoimiseksi sille annetaan suurimman posterioritodennäköisyyden omaava luokka eli suurin arvo k_l/k eli selvitetään k:n lähimmän pisteen luokat ja annetaan useimmin esiintyvä luokka pisteelle $\mathbf{u}$. Kun $k = 1$, kutsutaan menetelmää yksinkertaisesti lähimmän naapurin menetelmäksi. [6]

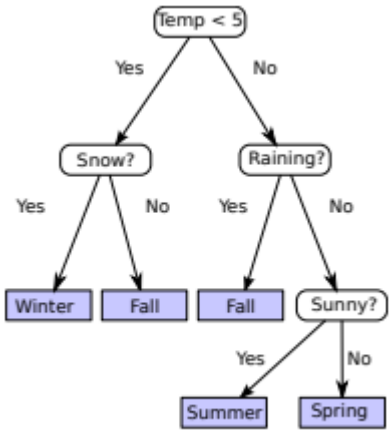
3.5 Syväoppiminen ja neuroverkot

Syväoppimisesta puhuttaessa tarkoitetaan usein neuroverkkoja (engl. *Neural Networks*) hyödyntävää oppimisjärjestelmää. Tavanomainen neuroverkkojärjestelmä koostuu useista toisiinsa yhteydessä olevasta neuronista, joille on määritelty erilaiset painoarvot. Sisääntuloneuronit aktivoituvat ympäristöä havaitsevista sensoreista, ja nämä neuronit aktivoivat seuraavan kerroksen neuroneita. Ketju jatkuu kunnes kerrokset loppuvat verkosta, ja verkko toistaa prosessin useita kertoja muuttaen aina painoarvoja ja optimoimalla ne, kunnes saadaan halutut arvot ulostuloista sisääntulosta riippumatta. Kun useampi verkko asetetaan päällekkäin siten, että verkkojen väliset sisään- ja ulostulot ovat toisiinsa yhteyksissä, puhutaan matalaoppimismallista (engl. *shallow learning*). Päällekkäin laitettujen verkostojen ei tarvitse olla keskenään samanlaisia ja matalaoppimismalleissa niitä on yleensä vain muutama. Kerrosten lisääntyessä siirrytään syväoppimismalleihin (engl. *deep learning*). [32]

Ajatus neuroverkoista on ollut olemassa 1940-luvulta asti [24], mutta ne ovat nousseet suureen tietoisuuteen koneoppimisen suosion myötä. Neuroverkkoja käytetään usein kuvan- ja puheentunnistuksessa [38], ja niillä on myös mahdollista lajitella tekstejä. Syväoppimismallit ovat myös tuottaneet muita algoritmeja parempia tuloksia erilaisissa mallinnus- ja ennustustehtävissä fysiikan, kemian ja lääketieteen saralla [20].

3.6 Päätöspuut ja satunnaismetsät

Jotkin mallit käsittelevät dataa jakamalla sitä kuutiomaisiin alueisiin, joiden rajapinnat ovat koordinaattiakselien suuntaisia. Kun valitaan mallia pisteelle avaruudessa, avaruus voidaan jakaa yksi rajapinta kerrallaan pienempiin alueisiin, ja sen jälkeen määritetään jokaiselle alueelle oma malli. Kaikissa rajapinnoissa alueet jaetaan kahteen osaan ja päätöksistä koottu diagrammi muistuttaa ulkomuodoltaan binääripuuta, ja tällaista luokitteluun luotua puudiagrammia kutsutaankin päätöspuuksi. Päätöspuu koostuu solmukohdista, joita jokaista vastaa yksi parametri θ_n, jossa n on solmujen kokonaislukumäärä. (engl. *decision tree*). [31][12]



Kuva 3.3. *Esimerkki päätöspuusta, joka pyrkii määrittämään vuodenajan [12]*

Kun datapisteet syötetään algoritmiin, päätöspuu tarkistaa jokaisen solmun kohdalla parametriin sidotun muuttujan arvon kaikista pisteistä ja jakaa ne kahteen osajoukkoon. Lopullinen osajoukkojen määrä ei välttämättä täsmää luokkien lukumäärän kanssa, joten toisistaan kaukana olevat puun haarat voivat luokitella osajoukon keskenään samaan luokkaan. Kun päätöspuun koko kasvaa attribuuttien myötä, kasvaa myös datan ylisovittamisen (engl. *overfitting*) mahdollisuus. Ylisovittamisella tarkoitetaan opetusdatan liiallista oppimista siten, että testidatan luokittelun virheprosentti kasvaa [12]. Satunnaismetsäalgoritmi (engl. *random forest algorithm*) pyrkii välttämään ongelman luomalla monta jonin satunnaiskomponentin omaavaa puuta, joiden päätöksentekokyky ei välttämättä ole täydellinen, toisin sanoen jos parametrien kokonaismäärä on n, puiden solmujen määrä on välillä $[1, n-1]$. Koska puut ovat satunnaisesti luotuja, voivat samat attribuutit toistua yksittäisessä puussa. Kun kaikki päätöspuut ovat antaneet oman luokituksensa datalle, tuloksista saadaan selville todennäköisyysjakaumat tiheysfunktiolle $p(\mathbf{u}|\mathcal{C}_l)$ yksittäisille datapisteille. Tämän jälkeen annetaan pisteelle luokka, jonka tiheysfunktion arvo on suurin. Tällöin kun käytetään suurempaa metsää luokitteluun, ylisovittamisen riski suppenee kohti nollaa. [27]

3.7 LSA-malli

Piilevän semantiikan analyysi (engl. *Latent Semantic Analysis*) eli LSA-malli on luonnollisen kielen käsittelymalli, joka pyrkii luomaan sanoille erilaisia konteksteja ja siten luomaan niille asiayhteyksiä hyödyntämällä singulaariarvohajotelmaa (engl. *Singular Value Decomposition*). Singulaariarvohajotelmaa, lyhyemmin SVD:tä, käytetään matriisilaskennassa hajottamaan matriisi $A \in \mathbb{R}^{m \times n}$ kahteen unitaariseen matriisiin $U \in \mathbb{R}^{m \times m}$ ja $V \in \mathbb{R}^{n \times n}$, sekä diagonaalimatriisiin $\Lambda \in \mathbb{R}^{m \times n}$, missä

$$A = U \Lambda V^T.$$

Diagonaalimatriisi $\Lambda \in \mathbb{R}^{m \times n}$ koostuu alkioista $a_{ij}, i = 1, 2, ..., m, j = 1, 2, ..., n$ siten, että

$$a_{ij} = \begin{cases} \sigma_r & , i = j \vee i \leq r \\ 0 & , i \neq j \wedge i > r \end{cases},$$

missä r on matriisin A singulaariarvojen kokonaislukumäärä ja

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 > 0.$$

Singulaariarvohajotelma voidaan suorittaa mille tahansa matriisille, vaikka se olisi kompleksinen reaalisen sijaan. Matriisit U ja V eivät myös ole yksikäsitteisiä, jos matriisille A löytyy moninkertaisia singulaariarvoja.

Vektorihaussa luodaan dokumenteista ja niissä läytyvistä sanoista vektoriavaruus, jossa jokainen sana ilmaisee yhtä ulottuvuutta. Pyrkimyksenä on palauttaa kaikki sanat perusmuotoonsa ja poistaa turhia sanoja, kuten pronomineja [9]. Avaruuden tekstit istutetaan samaan avaruuteen vektoreiksi, joiden pituus kullakin akselilla määritetään esimerkiksi kyseisen sanan esiintymismääränä tekstissä. Nyt voidaan erinäisistä halutuista hakusanoista luoda vektori, joka sisällytetään vektoriavaruuteen muiden tekstien tavoin. Tämän jälkeen verrataan muita tekstejä hakuvektoriin ja annetaan sitä eniten vastaavat tekstit hakijalle.

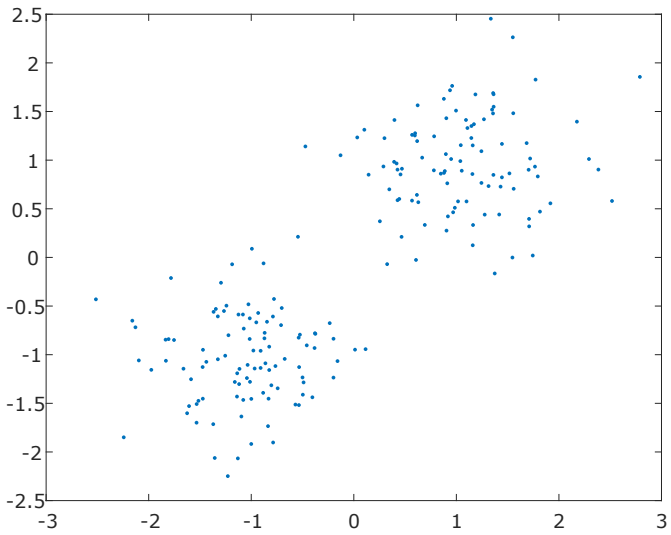
LSA pyrkii muuttamaan avaruuden rakennetta luomalla sanoille yhteyksiä hyödyntämällä singulaariarvohajotelmaa, jonka jälkeen haku vektorissa esiintyneet sanat voivat saada uusia tekstejä hakutuloksiksi riippuen sanojen välille muodostuneista yhteyksistä. Kun tavallinen vektorihaku etsii tietoa, se käsittelee sanoja sellaisenaan, ongelmaksi muodostuvat esimerkiksi synonyymit, joilla kaksi eri tekstiä voi käsitellä samaa asiaa. Esimerkiksi sanat "Rekka"ja "Kuorma-auto"toisivat tässä tapauksessa eri hakutulokset, vaikka ne tarkoittavat käytännössä samaa asiaa. LSA-mallilla voidaan myös ottaa hakuvaiheessa huomioon jonkin tietyn sanan yleiset kirjoitusvirheet [17].

Sanojen merkityksen huomioimiseksi haussa, suoritetaan teksteihin singulaariarvohajotelma. Hajotelmasta saaduista arvoista otetaan jokin määrä suurimpia arvoja, ja muut

arvot muutetaan nolliksi. Näiden muiden arvojen oletetaan olevan täytesanoja ja muuta kohinaa. Valitut singulaariarvot muodostavat nyt uuden tarkasteluavaruuden. Nyt hakusanalla "Rekka"saadaan myös hakutuloksiksi tekstejä, joissa esiintyy pelkästään sana "Kuorma-auto", ja vastaavasti saadaan hakutuloksia myös toiseen suuntaan.

3.8 Klusterianalyysi

Kun tilastoitua dataa on suuri määrä, sen käsittely voi sellaisenaan olla työlästä tai hidasta. Tällöin voidaan ositella data pienempiin ryhmiin datassa esiintyvien yhtäläisyyksien avulla, jolloin puhutaan datan klusteroinnista. Klusteroinnissa pyritään saavuttamaan eri klustereille välille mahdollisimman vähän päällekkäisyyttä ja klustereihin keskenään mahdollisimman samankaltaista dataa. Jokainen osa dataa kuuluu samalla vain yhteen klusteriin kerrallaan ja koska niissä ei oteta huomioon ennalta määriteltyjä luokkia, klusterointi on ohjaamattoman oppimisen yksi muoto. [1]



Kuva 3.4. *200 satunnaisesti kahdella eri odotusarvolla normaalijakauman mukaan luotua pistettä*

Kuvasta 3.4 nähdään, että pisteet jakautuvat kahteen selkeään joukkoon, joten ositellaan ne klusteroimalla kahteen joukkoon ja erotellaan ne kuvaajassa.

Kuva 3.5. *Kahteen klusteriin jaetut pisteet, joiden klusterit eroteltu värin mukaan*

Kuvassa 3.5 on nyt samat pisteet väritetty kahdella eri tavalla riippuen siitä, mihin klusteriin piste kuuluu. Menetelmä, jota pisteiden osittelussa käytettiin, on nimeltään k-means-klusterointi. Algoritmissa määritetään ennalta määritetty määrä keskipisteitä klustereille, jonka jälkeen ohjelma jakaa pisteet klustereihin vertaamalla yksittäisen datapisteen etäisyyttä keskipisteisiin ja valitsee etäisyyksistä pienimmän. Keskipisteiden sijainnit mää-

rittyvät siten, että yhteisetäisyys kaikkiin klusterin sisäisiin pisteisiin on mahdollisimman pieni. Etäisyysmetriikoina käytetään eniten euklidista etäisyyttä mutta myös korttelietäisyyttä käytetään joskus.

Klustereiden laatua voidaan tutkia laskemalla klusterien sisäisten pisteiden siluettiarvot. Määritellään sitä varten ensin suure $a(\mathbf{u}_i)$, joka kertoo klusteriin A kuuluvan pisteen $\mathbf{u}_i$ keskimääräisen etäisyyden klusterin kaikkiin muihin pisteisiin ja joka lasketaan kaavalla

$$a(\mathbf{u}_i) = \frac{1}{|A|-1} \sum_{\mathbf{u}_j \in A, i \neq j} d(\mathbf{u}_i, \mathbf{u}_j), \quad |A| > 1,$$

missä $|A|$ on klusterin A pisteiden lukumäärä. Tämän jälkeen määritellään suure $b(\mathbf{u}_i, C)$, joka kertoo pisteen $\mathbf{u}_i$ keskimääräisen etäisyyden jonkin toisen klusterin C pisteisiin, ja joka lasketaan kaavalla

$$b(\mathbf{u}_i, C) = \frac{1}{|C|} \sum_{\mathbf{u}_j \in C} d(\mathbf{u}_i, \mathbf{u}_j), \quad |C| > 0,$$

jossa $|C|$ on klusterin C pisteiden lukumäärä. Kun $b(\mathbf{u}_i, C)$ on laskettu kaikille klustereille $C \neq A$, niistä otetaan pienin luku ja määritellään

$$b(\mathbf{u}_i) = \min\{b(\mathbf{u}_i, C)\},$$

mikä kertoo pisteen $\mathbf{u}_i$ keskimääräisen etäisyyden lähimmän toisen klusterin pisteisiin. Tämän jälkeen lasketaan siluettiarvo $s(\mathbf{u}_i)$ kaavalla

$$s(\mathbf{u}_i) = \frac{b(\mathbf{u}_i) - a(\mathbf{u}_i)}{\max\{b(\mathbf{u}_i), a(\mathbf{u}_i)\}}.$$

Kun klusteri koostuu yhdestä pisteestä, sen siluettiarvo asetetaan nollaksi. Siluettiarvon laskutapa voidaan myös ilmoittaa muodossa

$$\begin{cases} 1 - a(\mathbf{u}_i)/b(\mathbf{u}_i), & b(\mathbf{u}_i) > a(\mathbf{u}_i) > 0 \\ 0, & b(\mathbf{u}_i) = a(\mathbf{u}_i) \\ b(\mathbf{u}_i)/a(\mathbf{u}_i) - 1, & b(\mathbf{u}_i) < a(\mathbf{u}_i). \end{cases}$$

Siluettiarvot ovat välillä $[-1, 1]$ ja klusteri on hyvin muodostunut eli klusteri on erillään toisista klustereista, jos sen pisteiden siluettiarvot ovat lähellä arvoa 1 ja huonosti muodostunut, jos sen pisteiden siluettiarvot ovat negatiivisia.

4 EMPIIRISEN TUTKIMUKSEN TULOKSET

4.1 Tarkasteltava kurssi

Ensimmäisen vuoden kurssi Insinöörimatematiikka 123 on maisterivaiheessa aloittaneille opiskelijoille suunnattu ensimmäisen vuoden matematiikan kurssi, jossa käydään yhden lukukauden aikana suurin osa The European Society for Engineering Education -yhteisön eli SEFI:n määrittämistä ensimmäisen tason (engl. *Core level 1*) aiheista [4, s. 29-35] [3]. Kurssin arvosana perustuu lopputentin lisäksi viikottaisiin harjoitustehtäviin sekä viikkokokeisiin, joita on yhteensä 14 kappaletta. Harjoitukset on jaettu kahteen osaan: Moodlessa tehtäviin STACK-tehtäviin, joita on kuusi jokaista aihetta kohden, sekä käsin ratkaistaviin kysymyksiin, joiden ratkaisut kirjataan Matlab-ohjelman avulla luotavaan pdf-tiedostoon. Kurssin jokaisesta aiheesta on osasuorituksena sähköisesti tehtävät viikkokokeet, jotka pääasiassa noudattavat kurssin viikkoaikataulua muutamaa poikkeusta lukuunottamatta, ja jotka tehdään suurimmaksi osaksi Älyoppi-järjestelmässä. Kaikista viikkokokeista on mahdollista saada maksimissaan 24 pistettä, paitsi tenttiviikoilla tehtävistä uusintatenteistä, joissa tehdään viikkokokeiden 3-7 tehtävistä yhdistelty kokonaisuus ja viikkokokeiden 9-13 tehtävistä yhdistelty kokonaisuus. Näistä uusintakokeista voi saada enintään 120 pistettä.

Taulukko 4.1. *Insinöörimatematiikka 123 -kurssin viikkokokeet viikkojärjestyksessä*

Periodi	Viikkonro	Aihe
1	1	Perustaitotesti
	2	Matlabin alkeet
	3	Kompleksiluvut ja vektorit
	4	Lineaarinen yhtälöryhmä
	5	Matriisit
	6	Matriisien ominaisarvot ja-vektorit
	7	Todistusmenetelmät
2	8	Matlab-verkkokurssi "Introduction to Linear Algebra with MATLAB" tai "Introduction to Symbolic Math with MATLAB"
	9	Funktiot
	10	Derivaatta
	11	Integraali
	12	Lukujonot ja sarjat
	13	Differentiaaliyhtälöt
	14	Matlab-verkkokurssi "Solving Ordinary Differential Equations with MATLAB"

Taulukon 4.1 viikokokeiden lisäksi 1. periodin tenttiviikolla on mahdollista tehdä viikkokokeet 1 ja 2 uudestaan, jolloin näistä ja alkuperäisistä suorituksista jää voimaan paras pistemäärä. Samalla on myös mahdollista tehdä viikkokoe 3-7, joka käsittelee viikkoja 3-7. 2. tenttiviikolla järjestetään kurssin loppukoe, joka käsittelee koko kurssin aiheet, ja samalla on myös mahdollista tehdä viikkokoe 9-13, joka kokoaa viikkojen 9-13 aiheet yhteen tenttiin.

Viikkokokeissa normaalisti on neljä STACK-tehtävää, jotka kaikki ovat automaattisesti tarkastettavia. Viikkokokeeseen 5 kehitettiin tutkimusta varten lisäksi neljä esseekysymystä, joihin riittää vastata muutamalla lauseella täysien pisteiden saamiseksi. Tehtävänannot ovat:

1. Kerro luonnollisella kielellä, mitä tarkoittaa, että vektorit ovat lineaarisesti riippuvia.
2. Kerro luonnollisella kielellä, miten voidaan selvittää, onko matriisi kääntyvä.
3. Kerro luonnollisella kielellä todistus sille, että matriisin nolla-avaruus on aliavaruus.
4. Kerro luonnollisella kielellä, miten voidaan tutkia, onko annettu vektorijoukko kanta jonkin toisen vektorijoukon virittämälle vektoriavaruudelle.

Tehtävien pyrkimyksenä on tuottaa mahdollisimman paljon kirjakieltä muistuttavaa tekstiä, joka on helposti jaettavissa erillisiin sanoihin algoritmien opettamisprosessissa. Tutkimuksessa hyödynnetään myös Porissa järjestettävän Matematiikka P1 -kurssin viidennen viikkokokeen vastauksia. Kurssi on rakenteeltaan samanlainen kuin Insinöörimatematiikka 123 -kurssin seitsemän ensimmäistä viikkoa, ja tällöin myös viikkokokeet ovat

samat.

Kaikissa algoritmeissa vastaukset pisteiden kanssa luetaan yhteen taulukkoon, jonka jälkeen kaikki isot kirjaimet muutetaan pieniksi, ja kaikki välimerkit poistetaan vastauksista. Tämän jälkeen vastaukset muutetaan vektorimuotoon, jotta saadaan luotua sanasäkki (engl. *bag of words*) mallien opettamista varten. Osa työssä käytetyistä vastausten esiprosessointi- ja testauskoodeista (liitteet A ja B) on tutkimuksen tarkoitukseen muokattuja versioita aiemmassa vastaavassa tutkimuksessa käytetyistä koodeista [18].

4.2 Tulosten tarkastelu

Luokittelumallien vertailussa on helppoa tutkia suoraan mallien tarkkuutta eli oikein luokiteltujen vastauksien osuutta kaikista vastauksista. Luokkina käytetään tässä tutkimuksessa vastauksien pistemääriä

Taulukko 4.2. *Luokittelumallien tarkkuudet jokaista tehtävää kohti*

	Tehtävä 1	Tehtävä 2	Tehtävä 3	Tehtävä 4
LSA 3.4 3.7	63,0%	70,4%	77,8%	70,4%
Syväoppiminen 3.5	44,4%	70,4%	66,7%	66,7%
Ristiinvalidoimalla optimoitu satunnaismetsä 3.6	59,3%	70,4%	63,0%	63,0%
Automatisoitu satunnaismetsä 3.6	48,2%	66,7%	70,4%	59,3%

Taulukosta 4.2 voidaan nähdä, että LSA-malli antoi eniten oikeita pistemääriä vastauksille. Lisäksi tehtävää 1 lukuunottamatta molemmat satunnaismetsäalgoritmit saivat huonommat tarkkuudet kuin LSA- ja syväoppimismallit. Kaikissa algoritmeissa on käytetty yksinkertaista ristiinvalidointia, joten tulokset voivat olla hyvinkin riippuvaisia datan jaosta. Tätä voidaan torjua k-kertaisella ristiinvalidoinnilla. Tutkimuksen data jaetaan opetus- ja testidataan suhteessa 4:1, joten sopiva k olisi 5. Taulukosta myös nähdään, että LSA-malli ei ole painotetun tarkkuuden perusteella yksiselitteisesti tarkin algoritmi, eli vaikka se luokittelee parhaiten täsmälleen oikeita luokkia, sen virheelliset luokitukset ovat todennäköisesti useammalla pisteellä väärin. Tämä käy selkeästi ilmi liitteen F kuvista 4.2, F.1 ja F.3.

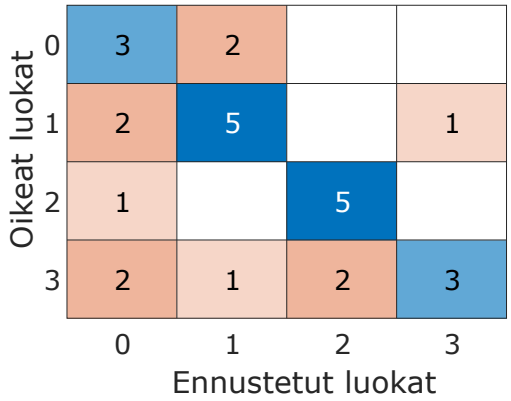
Kysymyksiä erikseen tarkasteltaessa voidaan tutkia esimerkiksi yleisimpiä vastauksissa esiintyviä sanoja tai n sanan mittaisia sanaryhmiä, joita kutsutaan n-grameiksi. Ensimmäisessä kysymyksessä yksi esimerkkivastaus olisi

> "Vektorit ovat lineaarisesti riippuvia, kun jokin joukon vektori voidaan esittää joukon muiden vektorien lineaarikombinaationa."

Kuvassa 4.1 on tehtävän 1 sanoista muodostettu sanapilvi, jossa mitä yleisempi sana on, sitä suurempi on sen kirjasinkoko pilvessä.

Kuva 4.1. *Tehtävän 1 vastauksissa eniten esiintyneet sanat*

Kuvasta ensimmäisenä voidaan huomata yleisimpien sanojen olevan saman sanan kaksi eri taivutusmuotoa. Tämä suomen kielen ominaisuus toistuu kaikissa tehtävissä ja se paisuttaa sanasäkin sanavarastoa vieden sinne kaikki eri yksittäisen sanan eri taivutusmuodot uusina sanoina. Ongelmaa voi korjata lyhentämällä sanat perusmuotoon eli lemmatisoimalla tai katkaisemalla ne sopivasta kohtaa eli stemmaamalla [14]. Matlabissa on sisäinen tuki vain englannin, saksan, korean ja japanin kielille [34]. Python on sen sijaan avoimen lähdekoodin ohjelmointikieli, joten käyttäjät voivat itse jakaa tekemiään kielipakkauksiaan. Myös jotkin erilliset sanat tuovat sanavarastoihin täytettä. Näitä ovat muun muassa konjunktiot, kysymyssanat, apuverbit, jotkin adjektiivit ja jotkin adverbit. Näille sanoille on olemassa yhteinen nimitys *stop words*, ja Matlabissa on valmis funktio näiden poistamiselle sanavarastosta.



Kuva 4.2. *Tehtävän 1 luokittelumatriisi, LSA-malli*

Kuvan 4.2 matriisista voi nähdä mallilla olevan vaikeuksia etenkin kolmen pisteen vastauksia, sillä ainoastaan puolelle niistä oli annettu oikea pistemäärä. Tämä voi johtua vastauksien monimuotoisuudesta, sillä kolmen pisteen vastauksia oli perusteltu esimerkkivastauksen tavan lisäksi muun muassa lineaarikombinaation kertoimilla tai redusoidun riviporrasmuodon vapailla muuttujilla. Muilla pistemäärillä vastaukset oli taas arvioitu enimmäkseen oikein.

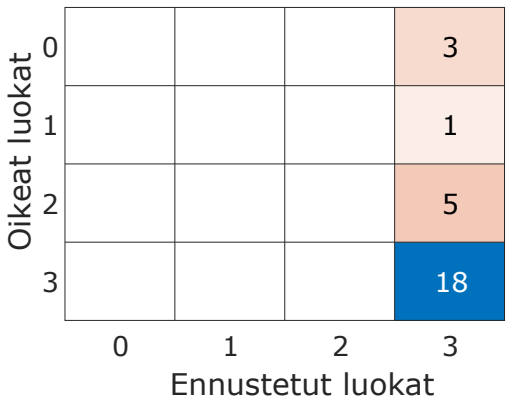
Tehtävästä 2 olisi saanut täydet pisteet vastaamalla esimerkiksi

> "Matriisi on kääntyvä, jos sen determinantti on erisuuri kuin nolla."

Tämä oli vastauksien pistemäärien perusteella selkeästi helpoin, sillä 66% vastanneista oli saanut täydet pisteet. Tämä saattaa aiheuttaa ongelmia jaettaessa dataa opetus- ja testiosiin, sillä opetusdatan ollessa 80% koko otoksesta testiosuuteen voi jäädä helposti vain kolmen pisteen arvoisia vastauksia. Tällöin mahdollinen korkea luokittelutarkkuus ei kerro koko, kuinka hyvin malli luokittelee esimerkiksi yhden pisteen vastauksia. Toinen tapa vastata oikein olisi myös ollut viitata matriisin rivien ja sarakkeiden lineaariseen riippumattomuuteen, mutta kurssilla painotetaan enemmän determinantin tarkastelemista. Tämä voidaan nähdä tutkimalla tehtävän sanapilveä.

Kuva 4.3. *Tehtävän 2 vastauksissa eniten esiintyneet sanat*

Kuvasta 4.3 havaitaan sanan *determinantti* olevan yksi vastauksien yleisimmistä sanoista, joten suuri osa opiskelijoista on perustellut vastaustaan kyseisen ominaisuuden avulla. Pilvestä huomataan taas sama ongelma kuin kuvassa 4.1, eli yksi sana löytyy yleisimpien sanojen joukosta useammassa eri taivutusmuodossa.



Kuva 4.4. *Tehtävän 2 luokittelumatriisi, Deep Learning -malli*

Kuvasta 4.4 käy ilmi, että suurin osa vastauksista oli luokiteltu kolmen pisteen arvoisiksi.

Valtaossa näistä vastasi oikeaa pistemäärää, mutta kaikkien muiden pistemäärien arvoisia vastauksia oli myös luokiteltu täysin oikeiksi. Tämä saattaa johtua pisteiden epätasaisesta painottumisesta, eli luokittelumalli ei välttämättä saa tarpeeksi tietoa alempien pistemäärien vastauksista. Täytyy myös ottaa huomioon käytetyn 0-3 pisteen pisteytyksen kelvollisuus kysymyksen arvioinnissa.

Tehtävälle 3 yksi mallivastaus olisi

> "Nolla-avaruus täyttää aliavaruuden kolme ehtoa:
>
> 1. Se on epätyhjä, sillä nollavektori kuuluu aina nolla-avaruuteen.
> 2. Nolla-avaruuteen kuuluvien vektorien summavektori kuuluu myös nolla-avaruuteen.
> 3. 3. Nolla-avaruuteen kuuluva vektori kerrottuna skalaarilla kuuluu myös nolla-avaruuteen."

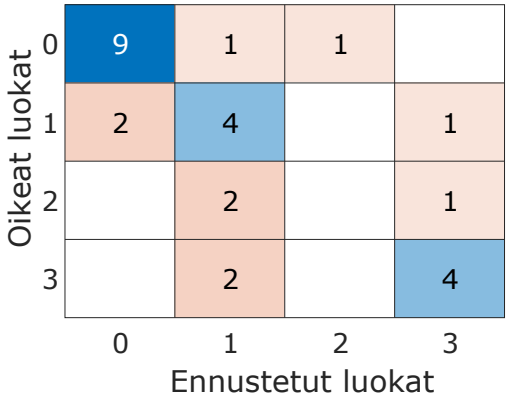
Kuva 4.5. *Tehtävän 3 vastauksissa eniten esiintyneet sanat*

Kuvasta 4.5 nähdään samat ongelmat sanojen taivutusmuotojen suhteen mutta sieltä myös löytyy muun muassa sanat "kolme", joka viittaa aliavaruuden kolmeen ehtoon, "suljettu", joka viittaa vektorien laskutoimitusten eheyteen. Koska nämä sanat eivät esiinny kovin suurena pilvessä, tarkoittaa se että vain osa oli kirjoittanut noita sanoja vastauksiin. Tämä voidaan varmistaa kyseisen tehtävän luokittelumatriiseista.

Kuvasta 4.6 huomataan samoja piirteitä kuin kuvasta 4.2, erona vain pisteiden painottuminen pienempään päähän. Erityisesti kahden pisteen vastauksia oli sen verran vähän, että ne luokiteltiin muihin pistemääriin. Samalla nähdään myös mallin luokitelleen vastauksille keskimäärin alempia pistemääriä, esimerkiksi kolme kahdeksasta yhden pisteen vastauksista on luokiteltu nollan pisteen arvoisiksi.

Tehtävä 4 oli kaikista selkeästi vaikein, sillä nollan ja yhden pisteen vastaukset kattoivat yhteensä noin 80% kaikista vastauksista. Esimerkkivastaus kysymykseen olisi ollut esimerkiksi

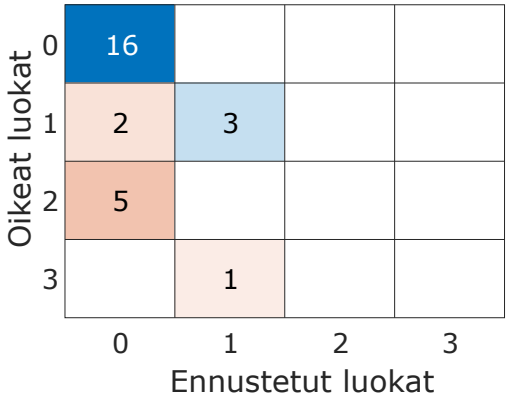
> "Jos vektorijoukon vektorien määrä on sama kuin vektoriavaruuden dimensio, ja vektorit ovat lineaarisesti riippumattomia, joukko muodostaa kannan."



Kuva 4.6. *Tehtävän 3 luokittelumatriisi, ristiinvalidoitu satunnaismetsämalli*

Kuva 4.7. *Tehtävän 4 vastauksissa eniten esiintyneet sanat*

Kuvan 4.7 sanapilvestä nähdään mainintoja lineaarikombinaatiosta ja lineaarisuudesta vektoriavaruuden ja -joukon mainintojen lisäksi, joilla pääsee jo lähelle oikeaa vastausta. Koska tehtävän pisteytys painottui alemmalle puolelle, vastauksdatasta luodut mallit eivät osanneet kovin hyvin arvioida muita kuin nollan pistemäärän vastauksia.



Kuva 4.8. *Tehtävän 4 luokittelumatriisi, automatisoitu satunnaismetsämalli*

Kuvasta 4.8 voidaan varmistaa, että malli ei arvioinut nollaa lukuunottamatta minkään pistemäärän vastauksia oikein. Vaikka muutama vastaus on arvioitu liian suurella piste-

määrällä, jälleen huomataan sama ongelma kuin tehtävässä 3, eli vastaukset arvioidaan liian alas. Ongelman torjumiseksi pitäisi saada tasaisesti pisteytettyjä vastauksia.

Probabilistisissa koneoppimisalgoritmeissa mallit antavat suoran luokituksen sijaan vastaukselle todennäköisyysjakauman kaikille pisteluokille. Lopullinen annettu luokitus määritetään näistä lukemista ottamalla suurin luku ja antamalla sitä vastaava luokka datalle. Työssä käytetyistä algoritmeista syväoppimis- ja satunnaismetsäalgoritmi ovat probabilistisia algoritmeja, joten niitä voidaan arvioida tutkimalla vastauskohtaisia todennäköisyysjakaumia. Käytetään esimerkkinä neljää vastausta tehtävästä 1 ja ristiinvalidoidun satunnaismetsän antamia todennäköisyyksiä.

1. "Vektorit x ja y ovat lineaarisesti riippuvia silloin, kun vektori x voidaan esittää kertomalla vektori y jollakin reaaliluvulla tai vektori y kertomalla vektori x jollakin reaaliluvulla."
2. "Sitä että vektorien esittäminen niiden avulla, edellyttää kahden tai useamman useamman vektorin käyttöä."
3. "Lineaarisesti riippuvilla vektoreilla on jokin nollasta poikkeava kerroin c, joka totetuttaa yhtälön c1v1+c2v2+c3v3....cnvn = 0. Eli yhtälöllä on epätriviaali vastaus."
4. <tyhjä vastaus>

Verrataan näille vastauksille määritettyjä todennäköisyyksiä, niille ennustettuja luokkia ja niiden todellisia luokkia keskenään kokoamalla ne yhteen taulukkoon 4.3.

Taulukko 4.3. *Eräiden tehtävän 1 vastauksien jakaumia ja luokat*

	$P(Y=0)$	$P(Y=1)$	$P(Y=2)$	$P(Y=3)$	Ennustettu luokka	Todellinen luokka
1. Vastaus	12.7%	25.6%	15.7%	46.0%	3	1
2. Vastaus	43.6%	17.1%	0%	39.3%	0	0
3. Vastaus	15.2%	32.0%	13.0%	39.8%	3	3
4. Vastaus	53.1%	30.0%	3.7%	13.7%	0	0

Kun tarkastellaan taulukon 4.3 toista riviä, nähdään kahden suurimman todennäköisyyden eroavan vain noin neljällä prosenttiyksiköllä kyseisten luokkien ollen vielä mahdollisimman kaukana toisistaan. Erilaisella opetusdatalla algoritmi olisi voinut antaa kolme pistettä nollan sijaan. Taulukosta huomataan myös 1. vastauksen todennäköisyydet, jossa luokan 3 todennäköisyys on huomattavasti suurempi kuin luokan 1 eli vastauksen todellisen luokan. Mielenkiintoista on myös 3. vastauksen jakauma, sillä jälleen luokille 1 ja 3 on annettu suurimmat lukemat eikä esimerkiksi vierekkäisille luokille 2 ja 3. Malli on myös antanut tyhjälle vastaukselle jakauman, jossa kaikki todennäköisyydet poikkeavat nollasta ja malli on vain hieman yli 50% varma vastauksen luokasta.

Vastauksiin sovellettiin myös klusterointia, jotta saataisiin selville, onko vastauksien teksteissä yhtäläisyyksiä. K-means-algoritmiin perustuva ohjelma testasi jakaa vastaukset eri määriin klustereita välillä [2,69], jossa välin yläraja on vastauksien kokonaislukumäärä

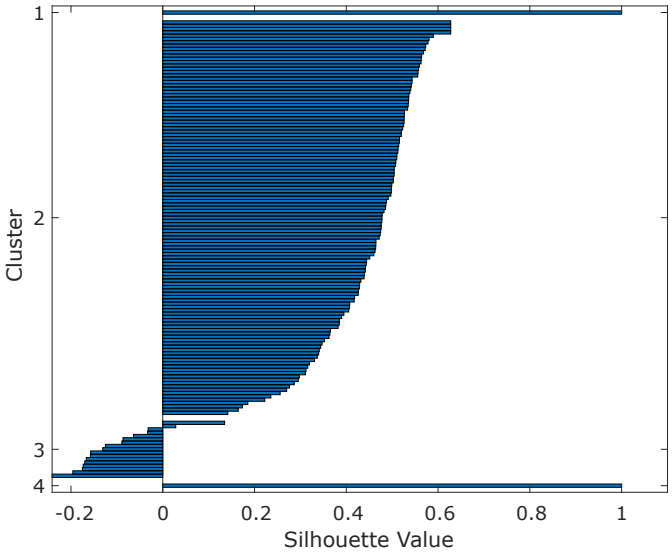
jaettuna kahdella. Ohjelma kävi kaikki klusterimäärät läpi, laski kaikille pisteille klustereiden sisäiset siluettiarvot ja otti niistä keskiarvot. Suurten klusterimäärien estämiseksi yhden datapisteen muodostamien klustereiden siluettiarvot nollataan.

..

Taulukko 4.4. *Viisi parasta siluettikeskiarvoa ja niitä vastaavat klusterimäärät*

Siluettiarvo	Klustereiden määrä
0,3920	4
0,3635	3
0,3179	2
0,1199	8
0,0915	10

Keskimääräiset siluettiarvot eivät millään klusterimäärällä ole kovin suuria, mikä vahvasti viittaisi, että klusterit eivät ole hyvin muodostuneita. Tällöin joko klustereiden sisäiset pisteet eivät ole tarpeeksi samankaltaista tai klustereiden välillä ei ole tarpeeksi erilaisuutta. Tarkastellaan neljään klusteriin jaettujen vastausten siluettiarvoja. Kuvassa 4.9 nähdään,



Kuva 4.9. *Neljään klusteriin jaettujen vastausten siluettiarvot*

että klusterit 1 ja 4 sisältävät yhden pisteen ja niiden arvoiksi on määritelty luvun 0:n sijasta 1. Tämä johtuu MATLABin silhouette-funktion ominaisuudesta, sillä se antaa yhden pisteen klustereiden siluettiarvoksi aina virheellisesti luvun 1. Klusterin 2 siluettiarvot ovat lähes kaikki negatiivisia, minkä perusteella pisteiden tulisi kuulua johonkin toiseen klusteriin.

4.3 Tuloksien parantaminen

Luokittelumallien suorituskyvyn määrittämiseen ei ole yksiselitteistä vastausta. Mallien suorituskyvyn parantaminen voidaan jakaa datan paratamiseksi ja algoritmin parantamiseksi. Yksinkertaisin tapa tehdä datasta parempaa on antaa algoritmille enemmän dataa opittavaksi. Pienessä otoksessa suurilla poikkeamilla on suurempi vaikutus opetusdataan, ja tämän tutkimksen tapauksessa vastauksien määrää nostamalla saadaan enemmän keskenään erilaisia vastauksia kullekin pistemäärälle, jolloin vastauksien sisällä syntyy vahvempia korrelaatioita. Kaupallisten arviointimallien vaatimusten perusteella saatetaan tarvita tutkimuksessa käytettyä vastausmäärää moninkertainen määrä vastauksia kuhunkin tehtävään.

Toinen yleinen parannustapa datalle on puuttuvien ja poikkeavien arvojen käsitteleminen. Numeerisessa datassa tämä usein tarkoittaa keskiarvojen, varianssien tai keskihajontojen määrittämistä. Koska tässä tutkimuksessa muuttujat ovat tekstissä esiintyviä sanoja, voidaan poikkeavana vastauksena pitää sellaista, jossa esiintyy paljon harvinaisia sanoja. Koska tavanomainen vektorihaku ei anna samankaltaisille sanoille yhteyksiä vaikka merkityksen muodossa, poikkeava vastaus saattaa myös sisältää paljon kirjoitusvirheitä. LSA-menetelmää voidaan soveltaa myös syväoppimis- ja satunnaismetsäalgoritmeissa vähentämään vektoriavaruuden dimensionaalisuutta, jonka pitäisi parantaa mallien luokittelutarkkuutta. Suomen kielen tapauksessa ongelmana on myös sanojen lukuisat eri taivutusmuodot, jotka voidaan poistaa lemmatisoimalla, eli muuttamalla kaikkien sanojen taivutusmuodot perusmuotoonsa [14]. Toinen vastaavanlainen toimenpide on stemmaaminen, jossa sanojen päätteet poistetaan kokonaan jättäen vain yhteisen kannan sanasta jäljelle. Sanavarastoa voidaan myös pienentää heti alussa poistamalla nk. hukkasanat (*stop words*) teksteistä. Hukkasanoja ovat yleensä pronominejä, konjuktioita ja muita kielten yleisempiä sanoja, jotka tavanomaisesti poistetaan tekstin prosessoinnin yhteydessä tuomasta turhaa täytettä sanasäkkiin [33]. Matlabin Text Analytics Toolbox sisältää funktiot edellä mainittuja toimintoja varten, mutta sillä on tuki ainoastaan englannin, korean, saksan ja japanin kielille, jolloin suomen kieltä käsiteltäessä täytyy määrittää kaikki asetukset ja toiminnot manuaalisesti [19]. Pythonille on olemassa valmiit lemmatisointifunktiot ja hukkasanalistat suomen kielen prosessointia varten. [26][9]

Malleja voidaan parantaa myös muuttamalla algoritmien hyperparametreja, ja jokaisella algoritmilla on omat hyperparametrit. Esimerkiksi k-lähimmän naapurin algoritmin ainoa hyperparametri on naapureiden määrä k, ja tutkimuksessa käytetty koodi käy läpi useamman k:n arvon, jolloin sitä ei erikseen tarvi erikseen säätää. LSA-algoritmissa voidaan muuttaa avaruuden dimensionaalisuuden määrittävien suurten singulaariarvojen määrää, ja sitä ei myös tarvitse erikseen optimoida, sillä tutkimuksessa käytetty algoritmi käy läpi dimensioluvut $5 - 80$ viiden luvun välein.

Syväoppimismallissa neuroverkkojen hyperparametreja on useampia mitä optimoida, kuten piilotettujen kerroksien lisääminen, neuronien lisääminen yksittäiseen kerrokseen tai oppimisnopeuden muuttaminen. Kerroksien kasvattaminen ja lisääminen yleensä antaa

parempia tuloksia, mutta ne samalla tekevät luokittelusta hitaampaa. Oppimisnopeuden muuttaminen on mutkikkaampi prosessi, sillä sen nopeuttaminen suurentaa virhegradienttifunktion askelia ja nopeuttaa luokitteluprosessia tarkkuuden kustannuksella, sillä gradientti ei välttämättä suppene. Hidastaminen taas luonnollisesti saa prosessin kestämään kauemmin saadakseen paremman tarkkuuden. Nämä ongelmat voidaan välttää käyttämällä hajoavaa oppimisnopeutta (engl. *decaying learning rate*), missä askeleen suuruus lasketaan uudestaan joka välissä.

Satunnaismetsäalgoritmien hyperparametrien optimoinnin yleisiä keinoja on puiden määrän lisääminen, puiden laajuuksien kasvattaminen tai pienentäminen ja oppimisnopeuden muuttaminen. Puiden määrän lisääminen ei välttämättä kasvata tarkkutta mutta antaa kuitenkin luotettavamman tarkkuuslukeman, mutta suurella metsällä luokitteleminen vie paljon suoritusaikaa. Puiden solmujen lisääminen tarkentaa puiden luokittelukykyä mutta saattaa myös aiheuttaa ylioppimista.

5 YHTEENVETO

Jo 1960-luvulla aloitettu tutkimustyö esseiden tarkastamisen automatisointiin on näyttänyt tietä lukuisalle nykyajan julkisesti saatavalle arviointiohjelmalle. Vaikka automaattiselle tarkastamiselle ja arvioinnille löytyykin paljon vastarintaa luovan kirjoituksen arvioinnin osalta, ovat ohjelmat tutkimuksissa arvioineet esseitä lähes yhtä hyvin kuin ihmisarvioijat.

Kuten mitkä tahansa algoritmit, eivät automaattiset esseiden arvioijat ole täydellisiä ja tätä onkin käytetty vasta-argumenttina automatisointia vastaan. Teknologia kuitenkin kehittyy nopeasti varsinkin nykyään ja samalla myös arviointiohjelmat. Niitä voidaan myös käyttää pitkien pohtivien esseetehtävien lisäksi muissa oppiaineissa esiintyviin sanallisiin tehtäviin, joissa haetaan yleensä jonkin käsitteen määritelmää. Tällaisissa tehtävissä on yleensä olemassa muutama erilainen oikea vastaus, joten vastaukset eivät poikkea toisistaan yhtä paljon. Selitystehtävissä taas ongelmaksi saattavat muodostua matemaattiset merkinnät luonnollisen kielen seassa.

Tehtävät eivät saa olla liian helppoja tai vaikeita, koska silloin opetettavasta datasta tulee liian yksipuolista, ja testidatassa saattaa esiintyä poikkeavia vastuksia, joita malli ei osaa luokitella hyvin. Mallin tarkkuus voi olla näennäisesti korkea, koska se osaa luokitella lukuisat täyden pisteen tai nollan pisteen vastaukset oikein. Tämän vuoksi täytyy kiinnittää erityistä huomiota tehtävän vaikeustasoon, jotta saadaan paljon tasaisesti pisteittäin jakautuneita vastauksia.

Tukimuksessa käytetyt mallit antoivat lupaavia tuloksia vaikka tutkimuksen otos oli suppea, sillä alaraja tarkkuudelle oli neljän eri pistemäärän tehtävissä oli 25%. Näistä LSA-malli suoriutui parhaalla tarkkuudella ja käytti myös selkeästi vähiten aikaa arviointiprosessiin. Syväoppimismalli ja satunaismetsämallit eivät antaneet yhtä hyviä tarkkuuksia ja niiden arviointiprosessi myös kesti jopa useita minuutteja. Kaikkien mallien tarkkuudet liikkuivat laajasti $40-80\%$ välillä tehtävästä ja mallista riippuen, ja tasaisemmin pisteytetyt tehtävät antoivat parhaimmillaan noin 65% tarkkuuden. Suuremmalla kurssilla saataisiin enemmän tenttivastauksia ja samalla enemmän opetus- ja testidataa, jolloin saataisiin parempia luokittelumalleja. Lisäksi Matlabin ongelmana oli suomenkielen luontaisen tuen puuttuminen, joka korjaantuisi jatkamalla tutkimusta Pythonilla. Avoimen lähdekoodin ansiosta käyttäjät ovat onnistuneet kehittämään tekstinprosessointifunktiot suomenkielelle.

LÄHTEET

[1] J. Abonyi ja B. Feil. *Cluster Analysis for Data Mining and System Identification*. Tammikuu 2007. ISBN: 376437988X, 9783764379889. DOI: 10.1007/978-3-7643-7988-9.

[2] J. Alatalo, S. Ali-Löytty, A. Knutas, L. Malmi, V. Pinkkilä ja P. Salo. *ÄlyOppi hankkeen STACK-tehtävien esittely*. URL: https://digicampus.fi/course/view.php?id=1033.

[3] S. Ali-Löytty. *Insinöörimatematiikka 123*. URL: https://www.tut.fi/opinto-opas/wwwoppaat/opas2019-2020/perus/aineryhmat/Matematiikka/MAT-04601.html.

[4] B. Alpers. *A Framework for Mathematics Curricula in Engineering Education*. URL: http://sefi.htw-aalen.de/Curriculum/Competency%5C%20based%5C%20curriculum%5C%20incl%5C%20ads.pdf.

[5] B. P. Bergeron. *Optical mark recognition*. URL: https://web.archive.org/web/20060613072246/http://postgradmed.com/issues/1998/08_98/dd_aug.htm.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] A. Borsetta, G. L. C. Young ja K. Young. *FormScanner*. URL: http://www.formscanner.org/about-us.

[8] B. Cheang, A. Kurnia, A. Lim ja W.-C. Oon. On automated grading of programming assignments in an academic institution. *Computers And Education* 41 (syyskuu 2003), 121–131.

[9] G. Diaz. *Stopwords Finnish*. URL: https://github.com/stopwords-iso/stopwords-fi.

[10] *EXAM ohjeet*. URL: https://wiki.eduuni.fi/display/CSCEXAM/EXAM+ohjeet.

[11] *Grade export*. URL: https://docs.moodle.org/32/en/Grade_export.

[12] H. Huttunen. *Pattern Recognition and Machine Learning, Slide Set 5: Ensemble Methods*. URL: https://www.cs.tut.fi/courses/SGN-41006/slides/Lecture5.pdf.

[13] *Introduction to Maxima for STACK users*. URL: https://stack2.maths.ed.ac.uk/demo2018/question/type/stack/doc/doc.php/CAS/Maxima.md.

[14] H. Jabeen. *Stemming and Lemmatization in Python*. URL: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python.

[15] G. Jain ja J. Ko. *Handwritten Digits Recognition*. URL: http://individual.utoronto.ca/gauravjain/ECE462-HandwritingRecognition.pdf.

[16] M. Koivumaa. *Kurssin etusivu*. URL: https://docs.moodle.org/3x/fi/Kurssin_etusivu.

[17] K. Kukich. Spelling Correction for the Telecommunications Network for the Deaf. *Communications of the ACM* 35.5 (1992), 80–90.

[18] V. Laaksonen. Sähköisten tenttien automaattinen tarkastaminen. Tutkielma. http://urn.fi/URN:NB 201811012521: Tampereen Teknillinen Yliopisto, lokakuu 2018.

[19] *Language Considerations*. URL: https://www.mathworks.com/help/textanalytics/ug/language-considerations.html.

[20] Y. LeCun, Y. Bengío ja G. Hinton. Deep Learning. *Nature* 521 (2015), 436–444. DOI: 10.1038/nature14539.

[21] Y. LeCun, C. Cortes ja C. J. C. Burges. *The MNIST Database of handwritten digits*. URL: http://yann.lecun.com/exdb/mnist/.

[22] L. Malmi ja N. Pirttinen. *ÄlyOppi - sähköisten oppimisympäristöjen käyttö ja kehitys*. URL: https://www.aalto.fi/fi/aalto-yliopisto/alyoppi-sahkoisten-oppimisymparistojen-kaytto-ja-kehitys.

[23] T. Mathworks. *MATLAB Live Editor*. URL: https://se.mathworks.com/products/matlab/live-editor.html.

[24] W. McCulloch ja W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5 (1943), 115–133. DOI: https://doi.org/10.1007/BF02478259. URL: http://www.inf.ufrgs.br/~engel/data/media/file/cmp121/mcp.pdf.

[25] E. B. Page. Computer grading of student prose, using modern concepts and software. *Journal of Experimental Education* 62.2 (1994), 127–142. URL: http://search.ebscohost.com/login.aspx?direct=true&AuthType=cookie,ip,uid&db=ehh&AN=9612041568&site=ehost-live&scope=site.

[26] H. Pitkänen, T. Likonen ja F. Pirinen. *Libvoikko and essential linguistic resources*. URL: https://github.com/voikko/corevoikko/tree/master/libvoikko.

[27] S. Polamuri. *How The Random Forest Algorithm Works in Machine Learning*. URL: https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/.

[28] C. Sangwin. *About the STACK Project*. URL: https://stack2.maths.ed.ac.uk/demo2018/question/type/stack/doc/doc.php/About/index.md.

[29] C. Sangwin. *STACK | Online Assessment*. URL: https://stack-assessment.org/.

[30] K. Santosh ja C. Nattee. A Comprehensive Survey on On-line Handwriting Recognition Technology and its Real Application to the Nepalese Natural Handwriting. *Kathmandu University Journal of Science, Engineering, and Technology* 5 (maaliskuu 2010), 41–43. DOI: 10.3126/kuset.v5i1.2845.

[31] R. Saxena. *How Decision Tree Algorithm Works*. URL: https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/.

[32] J. Schmidhuber. Deep Learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117. DOI: 10.1016/j.neunet.2014.09.003. URL: http://www.ncbi.nlm.nih.gov/pubmed/25462637.

[33] C. Silva ja B. Ribeiro. The importance of stop word removal on recall values in text categorization. *Proceedings of the International Joint Conference on Neural Networks, 2003.* Vol. 3. 2003, 1661–1666 vol.3. DOI: 10.1109/IJCNN.2003.1223656.

[34] *Stem or lemmatize words*. URL: https://se.mathworks.com/help/textanalytics/ref/normalizewords.html.

[35] S. Valenti, F. Neri ja A. Cucchiarelli. An Overview of Current Research on Automated Essay Grading. *Journal of Information Technology Education: Research* 2 (2003), 319–330. URL: https://doi.org/10.28945/331.

[36] A. Valmari. *MathCheck*. URL: http://math.tut.fi/mathcheck/.

[37] B. Wei, X. Sun, X. Ren ja J. Xu. Minimal Effort Back Propagation for Convolutional Neural Networks (syyskuu 2017), 3.

[38] *What Is Deep Learning?* URL: https://www.mathworks.com/discovery/deep-learning.html.

[39] R. Williams. The extension of factor analysis to three-dimensional matrices. *New horizons in university teaching and learning: Responding to change*. Toim. M. Kulski ja A. Herrman. Perth Australia: Centre for Educational Advancement, Curtin University, 2001, 173–184. URL: http://hdl.handle.net/20.500.11937/11929.

[40] *Ylioppilastutkinto digitalisoituu asteittain*. URL: https://www.ylioppilastutkinto.fi/images/sivuston_tiedostot/Sahkoinen_tutkinto/aikataulu.pdf.

[41] Ylioppilastutkintolautakunta. *Hyvän vastauksen piirteet*. URL: https://drive.google.com/file/d/1aNoEGLaBxP_vfjIhJnEHTjejCvm-LHYh/view.

A LSA-MALLIN OPETUS- JA TESTAUSKOODI

```
% Tekee xlsx-tiedostosta taulukon sisältäen tehtävien vastaukset ja
% niille ennalta annetut pisteet omissa sarakkeissaan jokaista
% tehtävää kohden.
filename = "Arvostelut.xlsx"; %
data = readtable(filename,'TextType','string');

% Lukitaan satunnaislukugeneraattori, jotta tulokset voidaan toistaa.
rng('default');
rng(1);


% Jaetaan vastaukset kahteen osaan, jossa on koulutus- ja testausosat
% suhteessa 4:1, ja erotetaan tekstit ja pisteet toisistaan.
cvp = cvpartition(data.Pisteet1,'HoldOut',0.2); %
dataTrain = data(training(cvp),:);
dataTest = data(test(cvp),:);

textDataTrain = dataTrain.Vastaus1; %
textDataTest = dataTest.Vastaus1; %
YTrain = dataTrain.Pisteet1; %
YTest = dataTest.Pisteet1; %

% Siistitään teksteistä välimerkit, isot alkukirjaimet ja
% hukkasanat sekä luodaan sanasäkki
textDataTrain = erasePunctuation(textDataTrain);
textDataTrain = lower(textDataTrain);

stopWords = ["on" "jos" "kun" "siis" "ett " "niin" "vain" "eli"...
    "koska"];

documentsTrain = tokenizedDocument(textDataTrain);

bag = bagOfWords(documentsTrain);
% bag = removeShortWords(bag,2);
bag = removeWords(bag,stopWords);

% Vektoroidaan myos testausosan tekstit
documentsTest = tokenizedDocument(textDataTest);

tic
% Alustetaan muuttujia:
% LSA-mallin antamat arvosanat
```

```
43 result = zeros(size(YTest,1),1);
44 % painotettu tarkkuus, jossa yhden pisteen eron arvo on 1/2
45 partialaccuracy = 0;
46 % tarkkuus identtisille arvosanoille
47 accuracy = 0;
48 % parhaimman tuloksen saavuttanut LSA-avaruuden ulottuvuuksien
49 % määrä
50 bestdim = 0;
51 % parhaimman tuloksen saavuttanut k-lähimman naapurin k
52 bestk = 0;
53 % paras tarkkuus
54 bestaccuracy = 0;
55 % paras painotettu tarkkuus
56 bestapartialaccuracy = 0;
57 % parhaan tuloksen erot
58 bestcompare = [];
59
60 % kokeillaan 5-80 ulottuvuuksilla askelvälillä 5
61 for dim = 5:5:min(80,size(textDataTrain,1))
62
63     % luodaan LSA-malli koulutussanasäkista
64     mdl = fitlsa(bag,dim);
65     % poimitaan mallista vektorien arvot
66     dscores = mdl.DocumentScores;
67     % sovitetaan testattavat vektorit LSA-malliin
68     dscoresTest = transform(mdl,documentsTest);
69
70     % haetaan k-lähintä naapuria kun k on 1-9, askelvälilla 2 jotta
71     % keskiarvon laskeminen on yksiselitteinen
72     for k = 1:2:9
73
74         % haetaan jokaisen testattavan vektorin k-lähin naapuri, ja
75         % niiden valmiiden arvosanojen mediaani
76         for i = 1:size(dscoresTest,1)
77             kIdx = knnsearch(dscores,dscoresTest(i,:),'K',k);
78             result(i) = median(YTrain(kIdx,1));
79         end
80
81         % lasketaan tarkkuudet
82         accuracy = sum(YTest==result)/(numel(YTest));
83         partialaccuracy = sum(2-abs(YTest-result))/(numel(YTest)*2);
84
85         % tarkastetaan onko saavutettu parempi tarkkuus
86         if accuracy > bestaccuracy
87             bestaccuracy = accuracy;
88             bestapartialaccuracy = partialaccuracy;
89             bestk = k;
90             bestdim = dim;
91             bestcompare = [YTest,result];
92         end
93     end
94 end
```

```
95  aika = toc
96  figure(1); clf;
97  cm = confusionchart(bestcompare(:,1),bestcompare(:,2),'XLabel',...
98      'Ennustetut luokat','YLabel','Oikeat luokat','FontSize',20)
99
100
101 figure(2)
102 pilvi = wordcloud(bag)
```

B SYVÄOPPIMISMALLIN OPETUS- JA TESTAUSKOODI

```
% Tekee xlsx-tiedostosta taulukon sisältäen tehtävien vastaukset ja
% niille ennalta annetut pisteet omissa sarakkeissaan jokaista
% tehtävää kohden.
filename = "Arvostelut.xlsx"; %
data = readtable(filename,'TextType','string');

% Lukitaan satunnaislukugeneraattori, jotta tulokset voidaan toistaa.
rng('default');
rng(1);


% Jaetaan vastaukset kahteen osaan, jossa on koulutus- ja testausosat
% suhteessa 4:1, ja erotetaan tekstit ja pisteet toisistaan.
cvp = cvpartition(data.Pisteet1,'HoldOut',0.2); %
dataTrain = data(training(cvp),:);
dataTest = data(test(cvp),:);

textDataTrain = dataTrain.Vastaus1; %
textDataTest = dataTest.Vastaus1; %
YTrain = dataTrain.Pisteet1; %
YTest = dataTest.Pisteet1; %

% Siistitään teksteistä välimerkit, isot alkukirjaimet ja
% hukkasanat
textDataTrain = erasePunctuation(textDataTrain);
textDataTrain = lower(textDataTrain);
YTrain = categorical(dataTrain.Pisteet4);
YTest = categorical(dataTest.Pisteet4);

% Siistitään teksteistä välimerkit, isot alkukirjaimet ja
% hukkasanat
textDataTrain = erasePunctuation(textDataTrain);
textDataTrain = lower(textDataTrain);
documentsTrain = tokenizedDocument(textDataTrain);

stopWords = ["on" "jos" "kun" "siis" "ett " "niin" "vain" "eli"...
    "koska"];
documentsTrain = removeWords(documentsTrain,stopWords);

tic;
```

```
% Luodaan vektori-istutus
embeddingDimension = 100;
embeddingEpochs = 50;

emb = trainWordEmbedding(documentsTrain, ...
    'Dimension',embeddingDimension, ...
    'NumEpochs',embeddingEpochs, ...
    'Verbose',0)

% Vapaaehtoinen vektoriavaruuden kuvaus
words = emb.Vocabulary;
V = word2vec(emb,words);
XY = tsne(V);
figure(3)
textscatter(XY,words)

% Luodaan teksteistä vielä verkostolle sopivat vektorit matriisiin,
% hyodyntäen luotua istutusta
sequenceLength = 100;
documentsTruncatedTrain = docfun(@(words)...
    words(1:min(sequenceLength,end)),documentsTrain);
XTrain = doc2sequence(emb,documentsTruncatedTrain);
for i = 1:numel(XTrain)
    XTrain{i} = leftPad(XTrain{i},sequenceLength);
end

% Hermoverkoston alustus
inputSize = embeddingDimension;
outputSize = 180;
numClasses = numel(categories(YTrain));

layers = [ ...
    sequenceInputLayer(inputSize)
    lstmLayer(outputSize,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer]

options = trainingOptions('adam', ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.01, ...
    'Plots','training-progress', ...
    'Verbose',0);

% Hermoverkoston koulutus
net = trainNetwork(XTrain,YTrain,layers,options);

% Käsitellaan testausosa samoin kuin koulutusosa käsiteltiin, uutta
% istutusta ei tehdä, vaan käytetään koulutusosan istutusta
textDataTest = erasePunctuation(textDataTest);
textDataTest = lower(textDataTest);
documentsTest = tokenizedDocument(textDataTest);
```

```
93  documentsTruncatedTest = docfun(@(words)...
94      words(1:min(sequenceLength,end)),documentsTest);
95  XTest = doc2sequence(emb,documentsTruncatedTest);
96  for i=1:numel(XTest)
97      XTest{i} = leftPad(XTest{i},sequenceLength);
98  end
99
100 % Lasketaan verkon tarkkuus identtisten arviointien antamisessa
101 [YPred, scores] = classify(net,XTest);
102 accuracy = sum(YPred == YTest)/numel(YPred)
103 strYPred = string(YPred);
104 strYTest = string(YTest);
105 numYPred = double(YPred);
106 numYTest = double(YTest);
107 partialaccuracy = sum(2-abs(numYTest-numYPred))/(numel(YTest)*2);
108 aika = toc
109 figure(1); clf;
110 cm = confusionchart(YTest,YPred,'XLabel',...
111     'Ennustetut luokat','YLabel','Oikeat luokat','FontSize',20)
112
113 % Edellä käytetyt funktiot sanojen vektorointiin istutuksen mukaan,
114 % ja lyhyiden vektorien pidentäminen matriisiin sovitettavaksi
115 function C = doc2sequence(emb,documents)
116     parfor i = 1:numel(documents)
117         words = string(documents(i));
118         idx = ~ismember(emb,words);
119         words(idx) = [];
120         C{i} = word2vec(emb,words)';
121     end
122 end
123
124 function MPadded = leftPad(M,N)
125     [dimension,sequenceLength] = size(M);
126     paddingLength = N-sequenceLength;
127     MPadded = [zeros(dimension,paddingLength) M];
128 end
```

C RISTIINVALIDOIMISEN AVULLA OPTIMOIDUN SATUNNAISMETSÄMALLIN OPETUS- JA TESTAUSKOODI

```
% Avaa taulukon xlsx-tiedostosta, jossa on annetut vastaukset ja
% ennalta annetut pisteet eroteltuna omissa sarakkeissa "Vastaus"
% ja "Piste".
filename = "Arvostelut.xlsx"; %
data = readtable(filename,'TextType','string');

% Lukitaan satunnaislukugeneraattori, jotta tulokset voidaan toistaa
rng(1);

% Poistetaan tyhjat vastaukset
% idxEmpty = strlength(data.Vastaus3) == 0;
% Jaetaan vastaukset koulutus- ja testausosiin, suhteessa 4:1,
% ja jaetaan jaetut osat teksteihin ja pisteisiin
cvp = cvpartition(data.Pisteet1,'Holdout',0.2); %
dataTrain = data(training(cvp),:);
dataTest = data(test(cvp),:);

% Siivotaan, vektoroidaan koulutusosan tekstit ja luodaan sanasäkki
textDataTrain = dataTrain.Vastaus1; %
textDataTest = dataTest.Vastaus1; %
textDataTrain = erasePunctuation(textDataTrain);
textDataTest2 = erasePunctuation(textDataTest);
textDataTrain = lower(textDataTrain);
textDataTest2 = lower(textDataTest2);
YTrain = dataTrain.Pisteet1; %
YTest = dataTest.Pisteet1;

stopWords = ["on" "jos" "kun" "siis" "ett " "niin" "vain" "eli"...
    "koska"];

documentsTrain = tokenizedDocument(textDataTrain);
bag = bagOfWords(documentsTrain);
bag = removeWords(bag,stopWords);
XTrain = full(bag.Counts);

% Vektoroidaan myos testausosan tekstit
documentsTest = tokenizedDocument(textDataTest2);

```

```
tic;
% Ristiinvalidoidaan 50 puuta 5-kertaisella ristiinvalidoinnilla,
% vaihdellen oppimisnopeutta ja jakojen enimmäismäärää puissa, siten että
% jakojen enimmäismäärä otetaan sarjasta [3^0, 3^1, ..., 3^(m-2)], kuitenkin
% 3^m ollessa pienempi kuin n-1. Jaot 3^(m-1) ja 3^m tuottivat tuloksiksi NaN,
% joten niitä ei ole käytetty.
n = size(XTrain,1);
m = floor(log(n - 1)/log(3));
learnRate = [0.1 0.25 0.5 1];
numLR = numel(learnRate);
maxNumSplits = 3.^(0:m-2);
numMNS = numel(maxNumSplits);
numTrees = 50;
Mdl = cell(numMNS,numLR);

% Luodaan malli kaikille oppimisnopeuksille ja jakojen enimmäismäärälle
for k = 1:numLR
    for j = 1:numMNS
        t = templateTree('MaxNumSplits',maxNumSplits(j));
        Mdl{j,k} = fitcensemble(XTrain,YTrain,'NumLearningCycles',...
            numTrees,'Learners',t,'KFold',5,'LearnRate',learnRate(k));
    end
end

% Arvioidaan kumulatiiviset luokitteluvirheiden suhteet kaikille malleille
kflAll = @(x)kfoldLoss(x,'Mode','cumulative');
errorCell = cellfun(kflAll,Mdl,'Uniform',false);
error = reshape(cell2mat(errorCell),[numTrees numel(maxNumSplits)...
numel(learnRate)]);


% Määritetään jakojen enimmäismäärä, puiden lukumäärä ja oppimisnopeus,
% jotka antavat pienimmän virheen
[minErr,minErrIdxLin] = min(error(:));
[idxNumTrees,idxMNS,idxLR] = ind2sub(size(error),minErrIdxLin);


% Luodaan uusi malli optimaalisten parametrien avulla
tFinal = templateTree('MaxNumSplits',maxNumSplits(idxMNS));
MdlFinal = fitcensemble(XTrain,YTrain,'NumLearningCycles',idxNumTrees,...
    'Learners',tFinal,'LearnRate',learnRate(idxLR))

XTest = transform(MdlFinal,documentsTest);


[YPred, score] = predict(MdlFinal,XTest);
accuracy = sum(YTest==YPred)/(numel(YTest))
partialaccuracy = sum(2-abs(YTest-YPred))/(numel(YTest)*2)
```

D AUTOMAATTISESTI OPTIMOIDUN SATUNNAISMETSÄMALLIN OPETUS-JA TESTAUSKOODI

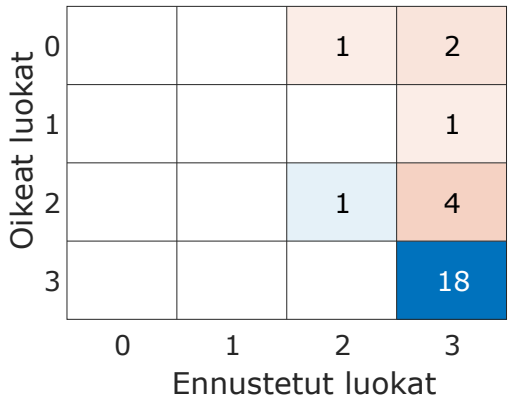
```
% Avaa taulukon csv-tiedostosta, jossa on annetut vastaukset ja
% ennalta annetut pisteet eroteltuna omissa sarakkeissa "Vastaus"
% ja "Piste".
filename = "Arvostelut.xlsx"; %
data = readtable(filename,'TextType','string');

% Lukitaan satunnaislukugeneraattori, jotta tulokset voidaan toistaa
rng(1);

% Poistetaan tyhjat vastaukset
% idxEmpty = strlength(data.Vastaus3) == 0;
% Jaetaan vastaukset koulutus- ja testausosiin, suhteessa 4:1,
% ja jaetaan jaetut osat teksteihin ja pisteisiin
cvp = cvpartition(data.Pisteet1,'Holdout',0.2); %
dataTrain = data(training(cvp),:);
dataTest = data(test(cvp),:);

% Siivotaan, vektoroidaan koulutusosan tekstit ja luodaan sanasakki
textDataTrain = dataTrain.Vastaus1; %
textDataTest = dataTest.Vastaus1; %
textDataTrain = erasePunctuation(textDataTrain);
textDataTrain = lower(textDataTrain);
YTrain = dataTrain.Pisteet1; %
YTest = dataTest.Pisteet1; %

% stopWords = ["onjoskunsiisett?niinvaineli"...
% "koska"];


documentsTrain = tokenizedDocument(textDataTrain);
bag = bagOfWords(documentsTrain);
% bag = removeWords(bag,stopWords);
XTrain = full(bag.Counts);

% Vektoroidaan myos testausosan tekstit
documentsTest = tokenizedDocument(textDataTest);
XTest = full(encode(bag,documentsTest));

```

```
cost.ClassNames=[0 1 2 3];
cost.ClassificationCosts=[0 1 2 3;1 0 1 2;2 1 0 1;3 2 1 0];

tic;
t = templateTree('Reproducible',true);
Mdl = fitcensemble(XTrain,YTrain,'OptimizeHyperparameters','auto',...
    'Learners',t,'HyperparameterOptimizationOptions', ...
    struct('AcquisitionFunctionName','expected-improvement-plus'));


[YPred, score] = predict(Mdl,XTest);
accuracy = sum(YTest==YPred)/(numel(YTest))
aika = toc
figure(1); clf;
cm = confusionchart(YTest,YPred,'XLabel',...
    'Ennustetut luokat','YLabel','Oikeat luokat','FontSize',20)

cost.ClassNames=[0 1 2 3];
cost.ClassificationCosts=[0 1 2 3;1 0 1 2;2 1 0 1;3 2 1 0];

MdlCost = fitcensemble(XTrain,YTrain,'OptimizeHyperparameters','auto',...
    'Learners',t,'HyperparameterOptimizationOptions', ...
    struct('AcquisitionFunctionName','expected-improvement-plus'),'Cost',
        cost);

[YPredCost, scoreCost] = predict(MdlCost,XTest);
accuracyCost = sum(YTest==YPredCost)/(numel(YTest))
figure(2); clf;
cmCost = confusionchart(YTest,YPredCost,'XLabel',...
    'Ennustetut luokat','YLabel','Oikeat luokat','FontSize',20)
```

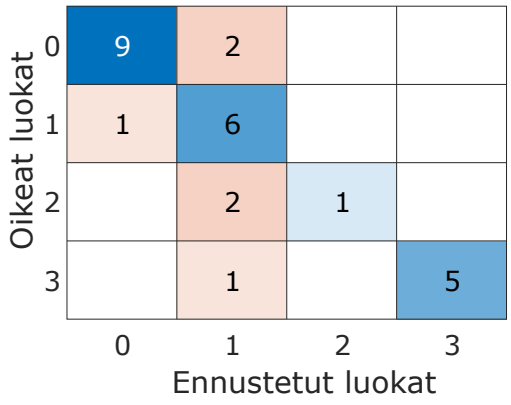
E K-MEANS-KLUSTEROINNIN JA SILUETTIARVOJEN MÄÄRITTÄMISEN KOODI

```
%Alustetaan muuttujia
sils=zeros(138,67);
clus=zeros(1,67);
bestclus=0;
avg=0;

% Ositellaan data eri määrään klustereita k-means-klusteroinnilla
for cl=2:69
    idx = kmeans(XTrain,cl);
    sil=silhouette(XTrain,idx); % Lasketaan klusterien sisäiset siluettiarvot
    sils(:,cl-1) = sil; % Talletetaan siluettiarvot yhteiseen matriisiin
    avg = mean(sil); % Siluettiarvojen keskiarvo
    clus(cl-1) = avg; % Talletetaan keskiarvo matriisiin
    if mean(sil)>bestclus % Talletetaan parhaat tulokset erikseen
        bestclus=mean(sil); % Paras keskiarvo
        bestsil=sil; % Parhaat siluettiarvot
        bestidx=idx; % Parhaat klusteri-indeksit
        bestcl=cl; % Klustereiden määrä, mikä tuotti parhaat tulokset
    end
end

% Talletetaan parhaat klusterit indekseineen
% yhteiseen solumatriisiin
total = cell(1,bestcl);
for l=1:bestcl
    kidx = bestidx==l;
    clust_l = textData(kidx);
    Yclust_l = Y(kidx);
    comb = [clust_l, Yclust_l];
    total{l}=comb;
end

% Piirretään siluettikuvaaja parhaista klustereista
[s,h]=silhouette(XTrain,bestidx);
```

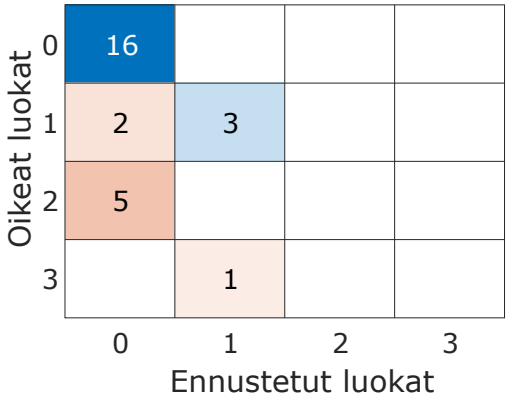
F LUOKITTELUMATRIISIT



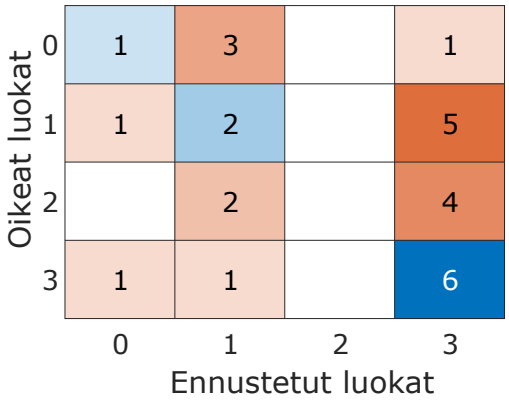
Kuva F.1. *Tehtävän 2 luokittelumatriisi, LSA-malli*



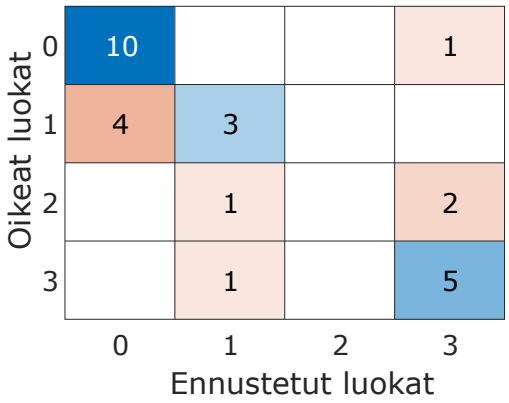
Kuva F.2. *Tehtävän 3 luokittelumatriisi, LSA-malli*



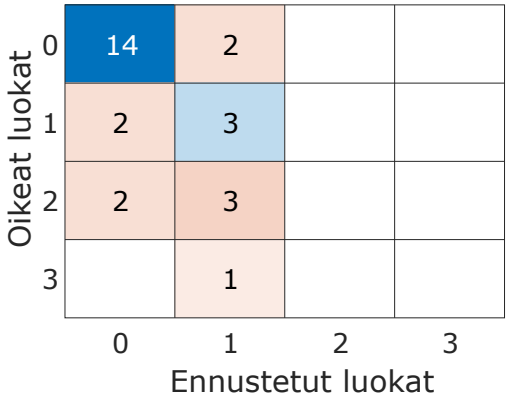
Kuva F.3. *Tehtävän 4 luokittelumatriisi, LSA-malli*



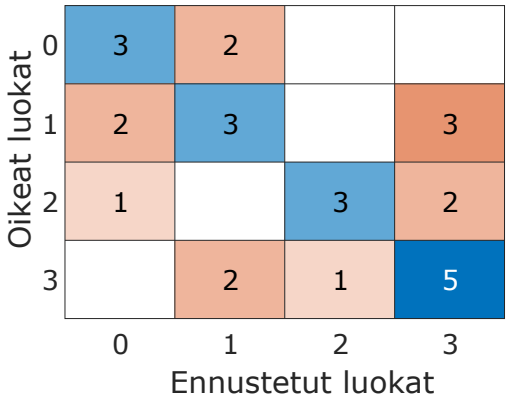
Kuva F.4. *Tehtävän 1 luokittelumatriisi, Deep Learning -malli*



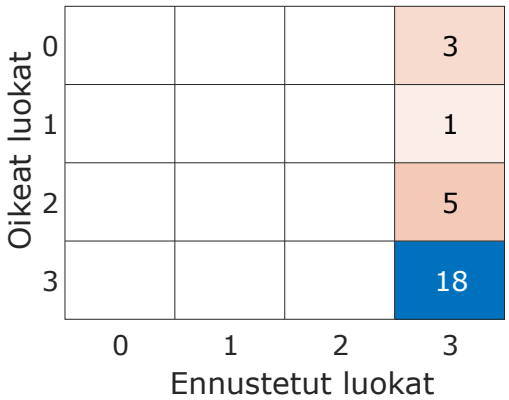
Kuva F.5. *Tehtävän 3 luokittelumatriisi, Deep Learning -malli*



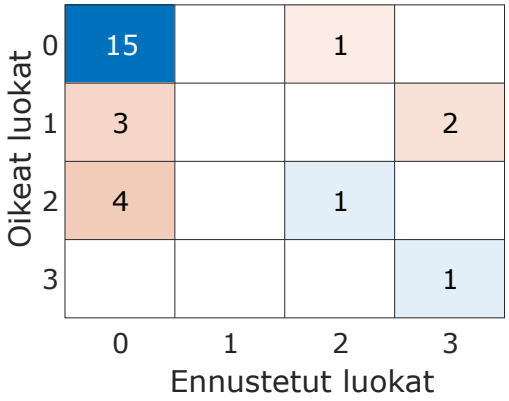
Kuva F.6. *Tehtävän 4 luokittelumatriisi, Deep Learning -malli*



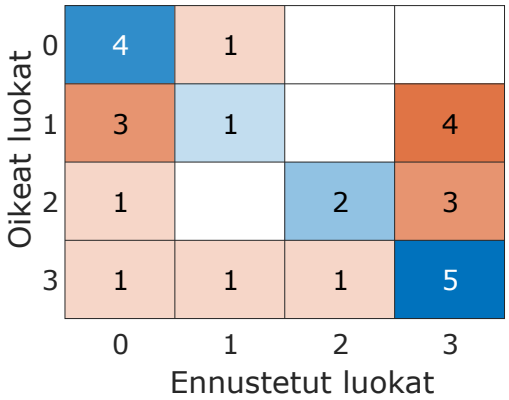
Kuva F.7. *Tehtävän 1 luokittelumatriisi, ristiinvalidoitu satunnaismetsämalli*



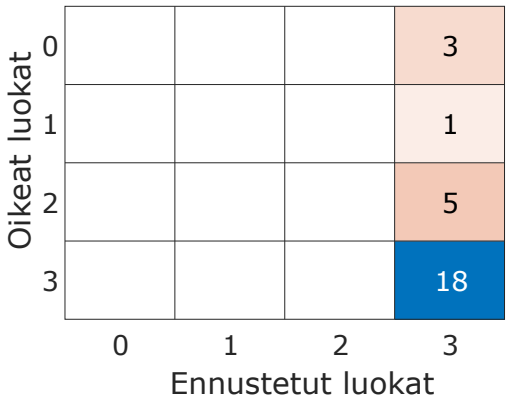
Kuva F.8. *Tehtävän 2 luokittelumatriisi, ristiinvalidoitu satunnaismetsämalli*



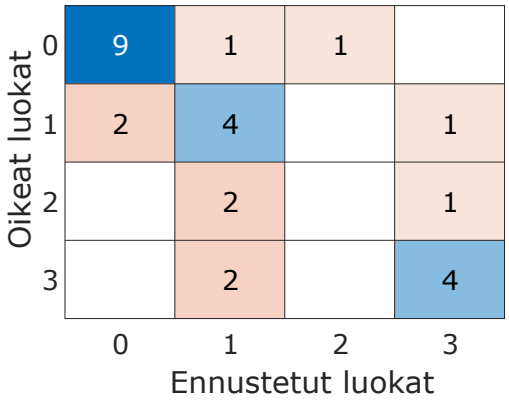
Kuva F.9. *Tehtävän 4 luokittelumatriisi, ristiinvalidoitu satunnaismetsämalli*



Kuva F.10. *Tehtävän 1 luokittelumatriisi, automatisoitu satunnaismetsämalli*



Kuva F.11. *Tehtävän 2 luokittelumatriisi, automatisoitu satunnaismetsämalli*



Kuva F.12. *Tehtävän 3 luokittelumatriisi, automatisoitu satunnaismetsämalli*