LAB University of Applied Sciences

Koneoppiminen osana tarjouslaskentaa

LAB-ammattikorkeakoulu
Insinööri (ylempi AMK), Digitaaliset ratkaisut
2020
Jarmo Ala-Rakkola

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Ala-Rakkola, Jarmo	Opinnäytetyö, ylempi AMK	2020
	Sivumäärä 53	

Työn nimi

Koneoppiminen osana tarjouslaskentaa

Tutkinto

Insinööri (ylempi AMK)

Ohjaavan opettajan nimi, titteli ja organisaatio

Minna Asplund, lehtori, Tieto- ja viestintätekniikka

Tiivistelmä

Tässä opinnäytetyössä selvitettiin koneoppimisen soveltuvuutta rakennusteknisiin tarjouslaskennan mittakuviin. Mittakuvat olivat muodoltaan käsinpiirrettyjä hahmotelmia ja niiden tulkinta perustui aiemmin ihmisen suorittamaan kuvanlukuun. Opinnäytetyö rajattiin mittakuvien kahden eri rakenteen muotoihin ja käytettävän koneoppimisen osalta ohjattuun oppimiseen.

Tutkimuksen tekeminen aloitettiin syventymällä koneoppimisen kuvantunnistukseen ja siihen liittyviin avoimen lähdekoodin ohjelmointikirjastoihin. Kuvantunnistuksen ohjelmointikirjastoista valittiin käytettäväksi kaksi suosittua ja hyvin dokumentoitua kirjastoa. Teorian ohjaamana valittiin kuvantunnistuksessa käytettävät menetelmät ja algoritmit.

Opinnäytetyön aikana luotiin prosessi ja ohjelma, jolla koneoppimisen kuvantunnistusta voitiin soveltaa analysoitaviin mittakuviin. Lopputuloksena saatu ohjelma oli toimiva, mutta rajattu toimimaan vain opinnäytetyön rajauksien puitteissa. Kuvantunnistuksen toimivuutta arvioitiin teknologian toimivuuden ja hyötyjen suhteen.

Asiasanat

koneoppiminen, kuvantunnistus, tarjouslaskenta

Abstract

Author(s)	Type of Publication	Published
Ala-Rakkola, Jarmo	Master's thesis	2020
	Number of Pages	
	53	

Title of Publication

Machine learning as part of offer processing

Name of Degree

Master of Engineering, Digital Solutions

Name, title and organization of the supervising teacher

Minna Asplund, Senior Lecturer, Information and Communications Technology

Abstract

The suitability of machine learning to the dimensional drawings of offer processing in civil engineering was studied in this thesis. The dimensional drawings were hand-drawn sketches and they were earlier interpreted by a person. The thesis was limited to two different forms of dimensional structures and to supervised learning of the machine learning that was used in this process.

The research began by focusing on image recognition of machine learning and open-source programming libraries related to it. Two popular and well-documented libraries were selected for use from the image recognition programming libraries. The methods and algorithms used in image recognition were chosen according to the theory.

A process and a program in which image recognition of the machine learning could be applied to the analyzed dimensions that were created during the thesis. The resulting program was functional but limited to work within the limits of the thesis. The functionality of image recognition was evaluated by the functionality and benefits of the technology.

Keywords

machine learning, image recognition, offer calculation

Sisällys

1 Johdanto

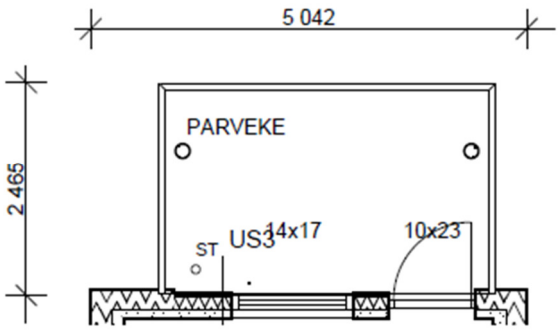
Digitaalisten ratkaisujen kehitys antaa mahdollisuuksia päivittää useilla aloilla pitkään vallinneita toimintamalleja. Näiden toimintamallien pohjalla on usein ihminen, joka toimii linkkinä digitaalisen ja fyysisen maailman välissä. Ihmisen kyky hahmottaa kuvissa esiintyviä objekteja ja asiakokonaisuuksia on ilmiömäinen ja aiemmin ihmisen kaltaisen hahmotuskyvyn omaavaa konetta pidettiin lähes mahdottomana valmistaa. Tietokoneiden tehonkasvun ja sen mahdollistaman tekoälyn kehityksen myötä tähän on kuitenkin tullut muutos. Tekoälyyn pohjautuvien ratkaisujen avulla on alettu siirtämään aiemmin ihmisten tekemää työtä tietokoneille sekä kehittämään aivan uuden tyyppisiä tekoälyyn pohjautuvia sovelluksia. Tekoäly ja sen alalajina oleva koneoppiminen ovat osoittautuneet tehokkaiksi rutiinien korvaajiksi useilla aloilla, joilla ne voivat toimia itsenäisesti ja jopa ihmistä paremmin.

Nykyään useat yritykset käyttävät lähes poikkeuksetta tietoisesti tai tiedostamattaan joitakin koneoppimiseen liittyviä digitaalisia ratkaisuja. Tulevaisuudessa yritysten on kuitenkin todennäköisesti alettava käyttämään yhä tietoisemmin tekoälyyn pohjautuvia ratkaisuja pysyäkseen mukana kehityksessä. Onkin siis luonnollista yrittää soveltaa alan uusimpia teknologioita uusiin käyttökohteisiin, joihin niiden oletetaan sopivan. Tämän opinnäytetyön tavoitteena on hakea suuntaviivat yhdelle rutiininomaiselle toiminnalle koneoppimista hyödyntäen.

1.1 Tarjouslaskennan nykytila

Nykyaikainen tarjouslaskenta perustuu useilla toimialoilla vielä hyvin manuaalisiin prosesseihin, joissa ihminen tulkitsee saamaansa tietoa ja laatii sen pohjalta tarjouksen. Hyvin usein tarjouskyselyt noudattavat samantyyppistä kaavaa ja ne voitaisiin näin ollen siirtää koneoppimisen kautta tietokoneen automaattisesti käsiteltäviksi. Usein suurin syy automatisoinnin esteeksi on tietämättömyys olemassa olevista teknologioista tai prosessin automatisoinnin kalleus.

Rakennusalalla ovat käytössä, kuvan 1 kaltaiset, viralliset piirustustavat, joita noudatetaan pääpiirteissään myös pienten tilausten ja projektien käsinpiirretyissä mittakuvissa. Tällaiset piirustukset tulevat usein sähköpostin mukana liitteinä ja niissä on hahmotettu tuotteiden päämuodot ja mitat. Kuvien heikko laatu, asettelu sekä käsin piirretyt muodot ja numerot tekevät kuvista vaikeita tulkita, verrattuna vakioaseteltuihin ja tietokoneella luotuihin mittakuviin.



Kuva 1. Ote virallisen piirustustavan mukaisesta rakennuspiirustuksesta (Peltonen 2015)

1.2 Tutkimuskysymykset ja kuvaus

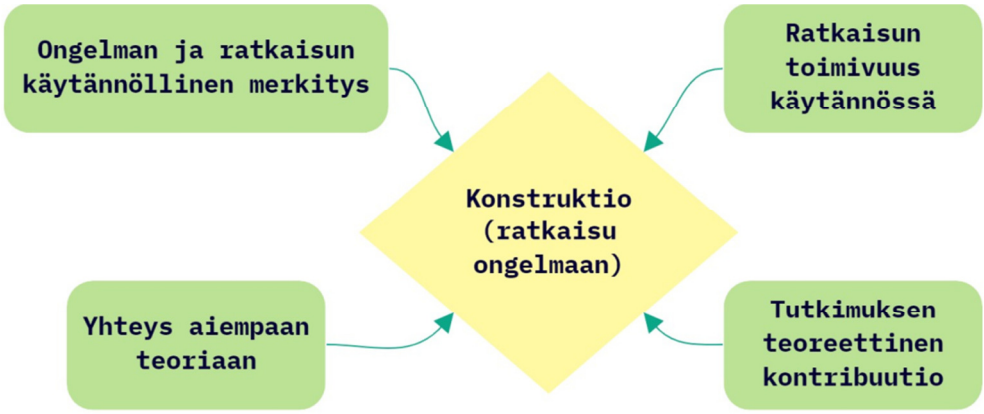
Työn tavoitteena oli testata tarjouskyselydokumenttien kuvien analysointia koneoppimista hyödyntäen. Tarjouskyselydokumentteina tässä työssä tarkoitetaan yleisesti lasijärjestelmien yhdentyyppisiä leikkauskuvia, jotka ovat kuvattu ylhäältäpäin. Tämän kaltaisia kuvia lähettävät pääsääntöisesti tuotteisiin ja rakennusteknisiin piirustuksiin perehtyneet henkilöt yrityksiltä toisille. Näin ollen muodot ja merkintätavat noudattelevat pääsääntöisesti tiettyjä vakioituneita kuvien rakenteita. Opinnäytetyön päätutkimuskysymyksenä oli siis tavoite vastata kysymykseen: kuinka analysoidaan lasijärjestelmien ylhäältäpäin kuvattuja leikkauskuvia koneoppimisella?

Perehtymällä koneoppimismenetelmiin tässä työssä oli myös tarkoitus saada valittua ja testattua soveltuva menetelmä edellä mainittujen kuvien tulkintaan. Alatutkimuskysymyksenä oli tavoite vastata kysymykseen: millaisella prosessilla voidaan tuottaa käsinpiirretyistä lasijärjestelmien kuvista sopivia koneoppimisen avulla tehtävälle analysoinnille? Käytännön sovelluksena, tavoitteena oli luoda koneoppimismalli, jolla voidaan havainnoida ja testata koneoppimisen soveltuvuutta tarjouskyselydokumenttien kuvien analysointiin. Tarkoituksena oli arvioida myös menetelmän käytäntöön soveltuvuutta kannattavuuden näkökulmasta eli hyötyjä suhteena kuluihin eli vastata toiseen alatutkimuskysymykseen: onko järjestelmän kehitys ja käyttö kaupallisen käytön kannalta taloudellisesti kannattavaa?

Työn suunnitteluvaiheessa yhtenä haasteena oletettiin olevan tallennetun tarjouskyselykuvien vähäinen määrä. Tästä syystä työssä jouduttiin luomaan myös omaa viitteellistä tarjouskyselykuvadataa koneoppimisen tarpeisiin. Oman aineiston luomisen ei kuitenkaan katsottu olevan ongelma tai tuloksia vääristävä tekijä, sillä aineistoa oli muutenkin rajattu kattamaan vain tyypillisimpiä esimerkkitapauksia.

Työssä käytettiin kuvassa 2 esitettyä konstruktiivista tutkimusotetta ja suunnittelutiedettä (eng. design science). Pohjatietona näille toimi tutkimustieto, alan kirjallisuus, olemassa

olevat kuvatiedostot ja rekonstruoidut kuvatiedostot. Koneoppimisen nopean kehityksen takia koneoppimiseen liittyvän pohjatiedon osalta pyrittiin käyttämään vain mahdollisimman uusia julkaisuja. Julkaisujen tiedon ajantasaisuutta myös varmistettiin välillä muista lähteistä.



Kuva 2. Konstruktiivisen tutkimusotteen kuvakartta (mukailtu Lukka 2020)

Koneoppiminen sisältää paljon alaan liittyviä omia sanoja ja käsitteitä. Kaikki alan sanat ja käsitteet eivät kuitenkaan ole vielä vakiintuneet, varsinkaan suomen kielessä, tai niillä voi olla useita rinnakkaisia nimiä. Osa suomenkielisistä käännöksistä ei myöskään kuvaa sanan sisältöä sen alkuperäisessä laajuudessaan. Tässä opinnäytetyössä on jätetty osa sanoista niiden englanninkielisiin muotoihinsa, koska vastaavan suomenkielisen sanan ei katsottu vielä saaneen oikeaa muotoaan. Alan tärkeimmän suomenkielisen erikoissanaston rinnalle on ensimmäisissä asiayhteyksissään sisällytetty myös sanan englanninkielinen vastine väärinymmärrysten välttämiseksi ja koneoppimisen englanninkielisen pääsanaston kartuttamiseksi.

2 Koneoppiminen

Koneoppiminen on tekoälyn yksi osa-alueista, josta on tullut viime vuosina yhä tärkeämpi osa ihmisten ja yritysten jokapäiväistä toimintaa. Aiemmin usein automatisoimattomina pidettyjä tehtäviä hallitaan nyt enenevissä määrin koneoppimisen avulla. Yksi yleisimmistä esimerkeistä on kasvojentunnistusohjelmisto, joka kykenee tunnistamaan digitaaliseen valokuvaan merkityt henkilöt. Tällaista sovellusta käytetään esimerkiksi Facebookissa, jossa tunnistetaan automaattisesti kuvassa olevia henkilöitä ja ehdotetaan käyttäjän ystävien merkitsemistä palveluun ladattuihin valokuviin. (Gollapudi 2016, 2; Bonaccorso 2018, 1.)

Koneoppimisen peruslähtökohta on nimensä mukaisesti kyky oppia monimutkaisten tietorakenteiden avulla. Oppimisen perustana on älykäs järjestelmä, joka hyödyntää tietolähteenään aiempia havaintoja ja tietoja, eli dataa (eng. data). Viimeisen vuosikymmenen aikana on kehitetty korkean tason avointa lähdekoodia, jonka avulla koneoppiminen onnistuu jopa kotitietokoneilla. Näin ollen monimutkaistenkin mallien suunnittelu ja toteutus on tullut kaikkien saataville ja koneoppimisen käytöstä osana sovelluksia on tullut yleistä. (Gollapudi 2016, 2; Bonaccorso 2018, 1.)

Koneoppimisessa suosituimpana ohjelmointikielenä on Python, joka tarjoaa hyvän sekoituksen toiminnallisuutta ja useita koneoppimiseen erikoistuneita algoritmipaketteja. Python sisältää myös useita laskentaan soveltuvia kirjastoja, jotka toimivat koneoppimisen parissa. Pythonin käyttöalueet ovat laajat ja se on varsin suosittu kieli tieteellisissä ja teollisissa sovelluksissa laajan funktioiden kehittäjäjoukon ansiosta. (Bowles 2015, xxiv.)

2.1 Koneoppimismenetelmät

Koneoppimisen algoritmit ovat rakennettu ratkaisemaan ongelmia, joita koneoppimisessa kutsutaan tehtäviksi (eng. task). Suoritettaessa ongelman ratkaisua on tärkeä mitata ratkaisun suorituskykyä. On kuitenkin huomioitava, että eri algoritmit tuottavat erilaisen mallin suoritettaessa niitä eri tietolähteillä eli dataseteillä (eng. dataset). Näin muodostettuja malleja ei saisikaan verrata keskenään, vaan vertailussa tulisi käyttää tulosten johdonmukaisuutta eri datasettien ja mallien kanssa. (Gollapudi 2016, 9-10.)

Mallit (eng. models) ovat keskeisessä asemassa koneoppimisessa, sillä niillä kuvataan dataa, joita järjestelmässä havaitaan. Mallit muodostuvat algoritmien luomana, datakokonaisuuden avulla. Monissa tapauksissa luotuja malleja käytetään uusiin datasetteihin, jotka auttavat malleja oppimaan uusia käyttäytymismalleja tai ennustamaan mallien avulla ratkaisuja. (Gollapudi 2016, 9-10.)

Datasetistä riippumatta on suositeltavaa käyttää soveltuvinta algoritmia ratkaisemaan ongelma. Eri algoritmeilla on eri vahvuuksia ja käyttöympäristöjä, joissa ne toimivat parhaiten. Oppimistavan kannalta algoritmit voidaan jakaa useisiin pääluokkiin, kuten ohjattuun, ohjaamattomaan ja vahvistettuun oppimiseen. (Gollapudi 2016, 9-10.)

2.1.1 Ohjattu oppiminen

Koneoppimisessa algoritmeja voidaan opettaa datalla, jonka lopputulos on tiedossa. Tätä opettamisen muotoa kutsutaan ohjatuksi oppimiseksi (eng. supervised learning). Opetusdatan avulla algoritmin kehitystä voidaan syötteen pohjalta ohjata halutun lopputuloksen saavuttamiseksi. Ohjattu koneoppiminen on siis koneoppimisen muoto, jossa käytettävä data ohjaa tiedossa olevilla tuloksillaan algoritmin opetusta oikeaan suuntaan. (Garreta & Moncecchi 2013, 25-26; Gollapudi 2016, 8.)

Ohjatun oppimisen suurin haaste piilee opetusdatan hankinnassa ja luokittelussa. Luokittelulla (eng. classification) tarkoitetaan datalle annetun arvon tai muun tiedossa olevan määritteen antamista. Kaiken datan pitää olla valmiiksi luokiteltua, jotta ohjatun oppimisen algoritmit voivat sitä luotettavasti hyödyntää. Huonosti luokiteltu data voi pahimmillaan johtaa algoritmin vääristyneeseen oppimiseen ja sen aiheuttamiin vääriin ennustuksen lopputuloksiin. Vääristyneestä oppimisesta voidaan puhua myös asioiden alisovittamisena (eng. underfitting) tai ylisovittamisena (eng. overfitting). Alisovittaminen oppimisessa johtaa yleistettyihin lopputuloksiin, jolloin ennusteen tarkkuus kärsii. Ylisovittaminen johtaa taas liian tarkkaan opetusdatan seuraamiseen, jolloin algoritmi toimii opetusdatalla hyvin, mutta muulla datalla huonosti. Alisovittamisessa ja ylisovittamisessa algoritmit eivät siis onnistu erottamaan opeteltavia luokkia parhaalla mahdollisella tavalla. (Garreta & Moncecchi 2013, 25-26; Gollapudi 2016, 8.)

Koneoppimisessa ohjatun oppimisen etuna pidetään pienempää datamäärän tarvetta. Datan sisältäessä tiedetyn lopputuloksen arvon tai muun määritteen voidaan koneoppimismalli kouluttaa ja testata tehokkaammin, kuin tilanteissa joissa lopputulos ei ole tiedossa. Ennalta määriteltyjen tulosten avulla opetetuista koneoppimismalleista saadaan todennäköisesti myös luotettavampia. Ongelmana ohjatussa oppimisessa on kuitenkin siinä käytettävän datan saatavuus, sillä luokiteltua dataa ei ole aina saatavilla tai datan luokittelu voi vaatia todella paljon työtä. (Garreta & Moncecchi 2013, 25-26; Gollapudi 2016, 8.)

2.1.2 Ohjaamaton oppiminen

Koneoppimisessa voidaan käyttää pohjana myös luokittelematonta dataa, jolloin puhutaan ohjaamattomasta oppimisesta (eng. unsupervised learning). Ohjaamattomassa oppimisessa koneoppimisratkaisu oppii itsenäisesti luokittelemattoman datan pohjalta jakamalla dataa luokkiin niiden samanlaisuuden, tiettyjen yhtenäisten mallien tai rakenteiden mukaan. Ohjaamatonta oppiminen on parhaimmillaan käytettäessä tunnistamaan sillä poikkeavaisuuksia suuressa datavirrassa ja näiden poikkeavaisuuksien pohjalta luoduissa päätöksissä. (Gollapudi 2016, 22.)

Ohjaamattoman oppimisen vaativuus ilmenee erityisesti opetuksen ja datan suhteen. Opetustilanteissa ohjaamattoman oppimisen riskinä on huonon koneoppimisalgoritmin käyttö, joka johtaa vääristyneisiin tulkintoihin. Ohjaamattomassa oppimisessa kokonaisuus on huomattavasti monimutkaisempi verrattuna ohjattuun oppimiseen, jolloin oppimisen virheellinen suunta ja siitä seuranneet mahdollisuudet virheellisiin lopputuloksiin kasvavat selkeästi. Datan suhteen ohjaamaton oppiminen vaatii suuria datamääriä koneoppimismallien luomiseen, joka voi rajoittaa sen käyttöä joissain sovelluskohteissa. Suurten datamassojen kanssa ohjaamaton oppiminen voi kuitenkin olla lähes ainoa varteenotettava vaihtoehto. Ohjaamaton oppiminen oikeilla algoritmeilla käytettynä on myös erittäin tehokas löytämään suurista datamassoista piilomerkityksiä, joiden havaitseminen muuten olisi todella vaikeaa. (Gollapudi 2016, 22.)

2.1.3 Vahvistusoppiminen

Koneoppimisessa algoritmin oppimista luokittelemattoman datan kanssa voidaan ohjata vahvistamalla oikeita ja vääriä valintoja, jolloin puhutaan vahvistusoppimisesta (eng. semi supervised learning). Vahvistusoppimisessa on yritetty yhdistää ohjatun ja ohjaamattoman oppimisen hyviä puolia. Luokittelemattoman datan helppo saatavuus ja ohjatun oppimisen tulosten tarkkuus on yritetty sovittaa vahvistusoppimisessa yhteen. Ympäristöstä tulevat positiiviset ja negatiiviset palautteet optimoivat algoritmin toimintaa kohti positiivisempia palautteita. Käytännössä vahvistusoppimisessa tietoa luokitellaan algoritmin oppimisen aikana. (Gollapudi 2016, 22; Bonaccorso 2018, 46.)

Vahvistusoppiminen sisältää haasteita ohjatun ja ohjaamattoman oppimisen osa-alueista. Mallien monimutkaisuus ja ympäristön hahmotuskyky ovat monimutkaisina kokonaisuuksina todella haastava yhteen sovitettava yhtälö. Vahvistusoppiminen saattaa kuitenkin olla yksi osa kohti vahvempaa tekoälyn muodostumista. (Gollapudi 2016, 22.)

2.2 Koneoppimiskirjastot

Python sisältää useita suosittuja kirjastoja, joilla voidaan laajentaa ja tehostaa Pythonin toimintaa, laskennan ja visuaalisuuden osalta, paremmin koneoppimiselle soveltuvaksi. Useat kirjastot ovat kirjoitettu Pythonia nopeammilla ohjelmointikielillä, jolloin niiden käyttö myös tehostaa ohjelmointikoodin ajamista. Jotkin kirjastot vaativat toimiakseen myös jonkin toisen kirjaston. (Hackling 2014, 16.)

Suosittuja kirjastoja ovat esimerkiksi NumPy, joka laajentaa Pythonia tukemaan tehokasta suurten taulukoiden ja moniulotteisten matriisien käyttöä. Matplotlib puolestaan tarjoaa visualisointifunktioita, joilla voidaan esimerkiksi visualisoida erilaisia taulukoita niiden havainnollistamiseksi. Pandas on kirjasto datan muokkaukseen ja analysointiin tukien useita eri tiedostomuotoja. SciPy on tieteelliseen ja tekniseen laskentaan erikoistunut kirjasto. Scikit-image eli Skimage on kuvien prosessointiin luotu kirjasto. (SciPy community 2020b; Hunter ym. 2020; Pandas developement team 2020; SciPy community 2020a; Scikit-image development team 2020.)

2.3 Konenäkö

Konenäkö on digitaalisessa muodossa olevan kuvan analysointia, jossa tietokoneohjelma tekee kuvan analysoinnin ja siihen liittyvät muut toimenpiteet. Konenäkö ei siis ole pelkästään näkemistä, vaan se on kokonainen prosessi sisältäen kaiken näkemisestä kohteen hahmottamiseksi. Konenäöllä voidaan tunnistaa esimerkiksi esineiden etäisyyksiä, rakenteiden muotoja ja pinnan tekstuureita. Konenäön toimintaperiaatteena on jakaa kuva pienempiin analysoitaviin osiin esimerkiksi tunnistamalla kuvasta ensin hahmoja ja sen jälkeen tunnistaa hahmot ihmisiksi. Ennen analysointia kuvaa joudutaan usein muokkaamaan analysoinnin käyttökohteen mukaan. Analysoinnin lähtökohtana on myös se, että sille on aiemmin opetettu analysoitava kohde ja luotu prosessi, jolla analysointi etenee. (Dadhich 2018, 7-13.)

Konenäön kehityksen myötä kuvien luokittelu, esineiden havaitseminen ja objektien seuranta ovat nousseet käytetyimmiksi konenäkösovellusten käyttökohteiksi. Erilaisia sovelluskohteita on jo laaja joukko erilaisista kuvantunnistuksista itse ajaviin autoihin. Jatkuvasti kehitetään myös uusia tapoja kuvien analysointiin ja kohteisiin, jossa konenäköä voidaan hyödyntää. (Dadhich 2018, 7-13.)

2.4 Koneoppimisen data

Data on koneoppimisen tärkein oppimisen lähde. Datalla tarkoitetaan kaiken tyyppisiä ja kaiken kokoisia dataryhmiä, joita luokittelemalla pystytään kertomaan käytettävän datan jäsennyksen tilasta. Jäsentämätön data on yleensä datan raakamuoto, jota harvoin voidaan käyttää suoraan koneoppimisessa. Raakamuotoinen data koostuu näytteistä luonnollisista tai ihmisen luomista näytteistä, kuten esimerkiksi videotiedostoista, äänimateriaalista, valokuvista tai Twitter twiiteistä. Tämän luokan dataa on runsaasti saatavilla sekä sitä on verrattain helppo kerätä. Raakamuotoisessa datassa ei kuitenkaan yleensä ole selitystä liitetystä merkityksestä, joten se joudutaan jäsentämään ennen todellista hyödyntämistä. Jäsentämättömästä datasta tulee jäsennettyä dataa, kun sille lisätään merkitys, jolla tarkoitetaan tunnisteen tai nimiön liittämistä osaksi dataa. Datan jäsentäminen on yleensä pakollista tulkittaessa ja määriteltäessä tiedonhaussa osuvuutta. Esimerkiksi valokuvan nimiöt voivat olla yksityiskohtia siitä, mitä se sisältää, kuten eläin, puu, yliopisto ja niin edelleen, tai äänitiedoston yhteydessä poliittisessa kokouksessa, jäähyväisjuhlissa ja niin edelleen. Useammin ihmiset luokittelevat dataa, ja luokitellun datan hankkiminen onkin huomattavasti kalliimpaa kuin luokittelemattoman raakadatan. Oppimismalleja voidaan soveltaa sekä jäsennettyihin että jäsentämättömään dataan. Tarkimpia malleja saadaan käyttämällä jäsennettyjen ja jäsentämättömien datajoukkojen yhdistelmää. Voidaankin sanoa, että koneoppimismalli voi olla korkeintaan yhtä hyvä, kuin sen käyttämä data on. (Hackling 2014, 12; Gollapudi 2016, 6-8.)

Datan määrälle koneoppimisessa ei ole annettu mitään vakiomäärää ja näin ollen datan määrä vaihtelee huomattavasti eri sovelluskohteista riippuen. Vähäisimmillään datamäärät voivat olla vain satoja havaintoja, mutta laajimmillaan miljoonia tai miljardeja havaintoja. Datan määrän vaikuttaa kuitenkin yleensä koneoppimismallin toimivuuteen. Suurilla datamäärillä päästään usein parempaan ennustettavuuteen verrattuna pienempiin datamääriin. (Hackling 2014, 30.)

Käytetyn datan tulee olla kuitenkin riittävän laadukasta ja puhdasta. Suurikaan datamäärä ei ole sen parempi, kuin sen laadukasta dataa sisältävä osa. Pahimmillaan datan seassa olevat virheelliset tiedot saattavat vaikuttaa koneoppimismallin luontiin negatiivisesti aiheuttamalla ennustukseen vääristymää. Monet opetusdatoina käytetyt datakokoelmat valmistetaan manuaalisesti tai puoliautomaattisilla prosesseilla, joilla varmistetaan niiden hyvä laatu. (Hackling 2014, 31.)

2.4.1 Datan jakaminen

Ohjatussa koneoppimisongelmissa jokainen havainto koostuu havaitusta vastemuuttujasta ja yhdestä tai useammasta havainnollistavasta muuttujasta. Havainnot vaativat pohjalle datajoukon, jota koneoppimisessa kutsutaan datasetiksi. Datasetin avulla malli saadaan opetettua ja testattua. Tätä toimenpidettä varten datasetti pitää jakaa osiin, joista pääjoukkoa kutsutaan opetusdataksi (eng. training data) ja loppua testausdataksi (eng. testing data). Alkuperäistä datasettiä jaettaessa on tärkeä varmistua, etteivät opetusdatan havainnot sisälly testausdataan. Testausdatan sisältäessä esimerkkejä opetusdatasta on vaikea arvioida onko algoritmi todella oppinut päättelemään opetusdatan tietojen pohjalta vai onko kyseessä vain alkuperäisen arvon suora muistaminen. (Hackling 2014, 11-13.)

Opetus- ja testausdatan lisäksi tarvitaan joskus kolmas havaintojoukko, jota kutsutaan vahvistusdataksi (eng. validate data). Vahvistusdatalla viritetään mallin oppimista ohjaavia muuttujia, joita kutsutaan hyperparametreiksi. Vahvistusdata on pidettävä yhtälailla erossa opetus- ja testausdatasta, sillä samalla datalla opetettu, vahvistettu ja testattu koneoppimismalli antaa todellisessa ympäristössä todennäköisesti vääriä tuloksia. (Hackling 2014, 11-13.)

Jaettaessa datasetiä opetus- ja testausdataksi on tavallista, että datasta noin 80 % on opetusdataa ja loppu 20 % testausdataa. Jaettaessa datasettiä opetus-, vahvistus ja testausdataksi on tavallista, että noin 50 % datasetistä on opetusdataa, 25 % testausdataa ja loput vahvistusdataa. Mitään osioiden välistä kokovaatimusta ei kuitenkaan ole olemassa ja loppujen lopuksi osioiden kokoon vaikuttaa eniten käytettävissä olevan datan määrä. Suuri datasetti mahdollistaa yleensä suhteessa suuremman opetusdatan käytön, sillä silloin testausdataksi jäävä osuus on vielä riittävän suuri antamaan hyvän testaustuloksen. (Hackling 2014, 11-13.)

Tapauksissa, joissa dataa on vähän saatavilla, voidaan opetuksessa käyttää myös ristiinvahvistamista (eng. cross-validation). Ristiinvahvistamisessa datasetti jaetaan useisiin osiin ja siitä erotetaan testausta varten yksi osio. Algoritmin opetus ja testaus suoritetaan tämän jälkeen vaihtamalla osioiden paikkaa useita kertoja siten, että kaikki datasetit ovat ajettu toistensa kanssa ristiin. (Hackling 2014, 11-13.)

Kuvassa 3 käytetty datasetti on jaettu neljään yhtä suureen osaan, joita kuvaavat kirjaimet A, B, C ja D. Jokainen osio käy opetuksen ja testauksen ristiin, jolloin saadaan neljä koulutus- ja testausjaksoa, numerot 1, 2, 3 ja 4. Ristiinvahvistaminen antaa varsinkin pienillä tietojoukoilla paremmin toimivan algoritmin, kuin yksinkertainen opetusmalli. (Hackling 2014, 11-13.)

	A	B	C	D
1	Testaus	Opetus	Opetus	Opetus
2	Opetus	Testaus	Opetus	Opetus
3	Opetus	Opetus	Testaus	Opetus
4	Opetus	Opetus	Opetus	Testaus

Kuva 3. Neljään osaan jaettu ristiinopetuksen datasetti (mukailtu Hackling 2014)

Datasetti voidaan jakaa opetusta ja testausta varten osajoukoiksi esimerkiksi Pythonin Pandas -kirjaston Series.str.split -funktioilla tai Scikit-learn -kirjaston train_test_split -funktiolla. Datasetin jakamisen yhteydessä määritellään jakamisen suhteet eli jaettujen joukkojen osien koot. Jakamisen yhteydessä voidaan myös alkuperäistä datajoukkoa sekoittaa, jolla varmistetaan datasetin arvojen tasalaatuisuutta jaetuissa osajoukoissa. (Scikit-learn developers 2020f; Pandas developement team 2020b.)

3 OpenCV -kirjasto

OpenCV on avoimen lähdekoodin kirjasto konenäkösovelluksia varten, jonka kehityksen on alun perin aloittanut teknologiayhtiö Intel Corporation. Se tarjoaa korkean tason rajapintoja kuvadatan käsittelemiseen ja esittämiseen. Monialustaisena kirjastona siitä on tullut suosittu yliopistojen ja teollisuuden sovellusten parissa. OpenCV -kirjasto on yhteensopiva muun muassa NumPy ja SciPy -kirjastojen kanssa. (Howse ym. 2016, s.i.; OpenCV team 2020e.)

Kuvantunnistusta tehtäessä kuvaa joudutaan yleensä käsittelemään ennen varsinaista tunnistusta. OpenCV -kirjasto sisältää monia sisäänrakennettuja toimintoja kuvankäsittelyyn imgproc -moduulissaan. Kuvaa voidaan esimerkiksi suodattaa, tehdä morfologisia operaatioita, geometrisiä muunnoksia, värimuunnoksia, piirtää kuviin, tehdä muotoanalyysejä, lohkoa kuvaa ja suorittaa useita muita toimintoja. (Joshi ym. 2016, 5.)

3.1 Binäärikuva

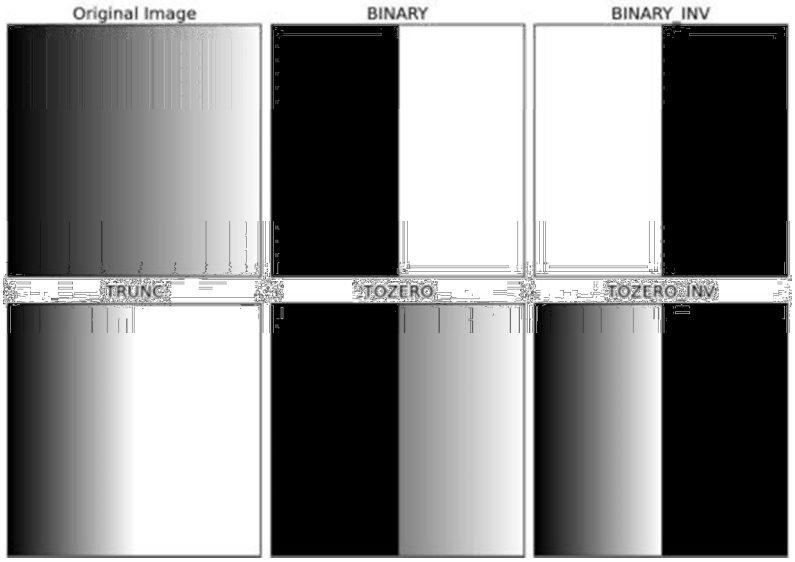
Kuvan binäärisointi on kuvan kontrastin tehostamista, kunnes täydellinen kylläisyys on saavutettu. Kylläisyydellä tarkoitetaan jäljelle jääviä kahta tasoa, jotka ovat yleensä musta ja valkoinen tai numeerisesti esitettynä 0 mustalle ja 1 tai 255 valkoiselle. Kuvan binäärisointia käytetään yleisesti koneoppimisen kuvantunnistuksen yhteydessä kuvan irrottamiseksi taustastaan ja yksinkertaisemman rakenteen takia. Binäärisoinnilla voidaan myös saada tunnistettavista kuvista tasalaatuisempia. (Blanchet & Charbit 2014, 222-223; Vlerick 2020.)

3.1.1 Kuvan kynnysarvot

Kuvaa voidaan siivota monilla eri tavoilla, joita yhdistävänä tekijänä on epätoivottujen ominaisuuksien tai osien poisto. Useimmiten kuvansiivous liittyy erilaisten kuvassa esiintyvien kohinapisteiden tai kokonaisten alueiden muokkaamiseen. OpenCV -kirjastossa kuvapisteiden kynnystä määrittelevällä soveltamisfunktiolla cv.THRESH_ ja sen jälkeen tulevalla kynnysarvolla voidaan kuvasta erotella ja eristää alueita, jotka halutaan tutkia tarkemmin. Kuvapisteitä muokataan tällöin määrittelemällä niille kynnysarvo, jonka yläpuolella olevat pisteet määritetään valkoisiksi ja alapuolella olevat pisteet määritetään mustiksi. (Nixon & Aquado 2012, 93-94; OpenCV team 2020a.)

OpenCV -kirjasto tukee useita erityyppisiä vakiokynnyksiä, joilla voidaan suorittaa hieman toisistaan poikkeavia kuvamuunnoksia. OpenCV -kirjaston funktiot näille vakiokynnyksille ovat BINARY, BINARY_INV, TRUNC, TOZERO, TOZERO_INV. Kuvassa 4 nähdään va-

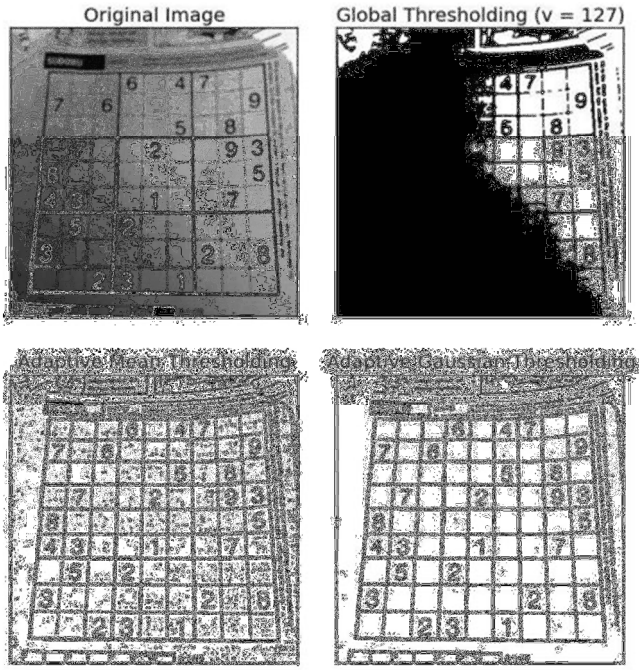
kiokynnysten käyttäytymistä lähtien vasemman yläkulman alkuperäisestä kuvasta. (OpenCV team 2020a.)



Kuva 4. OpenCV -kirjasto vakiokynnyksien toiminta (OpenCV team 2020a)

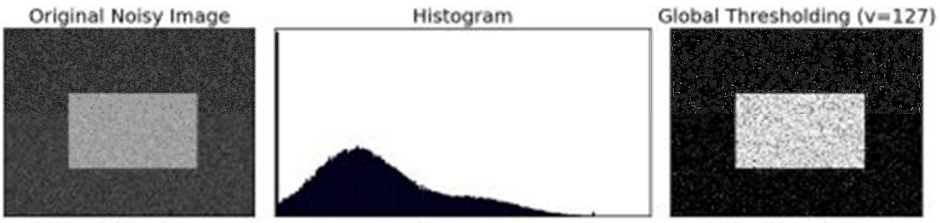
Yksi yleinen kynnys ei ole kaikissa tapauksissa kuitenkaan toimiva ratkaisu, sillä se ei osaa huomioida esimerkiksi kuvassa olevia varjoalueita. Mukautuvaa kynnysaluetta käyttämällä kuvapisteen arvo lasketaan kuvapisteen ympärillä olevan pienen alueen mukaan. Tällä menetelmällä saadaan eri kynnykset saman kuvan eri alueille, jolloin eri valoisuusalueet tasoittuvat kuvissa. (OpenCV team 2020a.)

Kuvassa 5 on esitetty kuvamuunnoksia erityyppisiä kynnysarvolaskentoja käyttämällä. Vasemman yläkulman kuvassa on esitetty alkuperäinen kuva. Oikeassa yläkulmassa on vakioarvoisella kynnyksellä esitetty kuvamuunnos. Vasemmassa alakulmassa kuva on muunnettu OpenCV:n MEAN_C -muunnosta käyttäen, joka laskee kynnysarvon naapurialueen keskiarvon perusteella ja vähentäen siitä vakion C. Oikeassa alakulmassa kuva on muunnettu OpenCV -kirjaston GAUSSIAN_C -muunnosta käyttäen, joka laskee kynnysarvon naapuruusarvojen Gaussin painotetulla summalla ja vähentäen siitä vakion C. (OpenCV team 2020a.)



Kuva 5. Vakio ja mukautuva kynnys (OpenCV team 2020a)

Otsun binäärisoinnilla saadaan määritettyä automaattisesti kuvan kynnysarvo. Kynnysarvon määrittäminen tapahtuu kuvan histogrammin kautta ja lopputuloksena saadaan yksi vakio kynnysarvo. Kuvassa 6 on esitetty Otsun binäärisoinnin toimintaa, vasemmalla on alkuperäinen kuva ja keskellä siitä saatu histogrammi. Oikealla kuvassa on Otsun binäärisoinnin automaattisesti histogrammista laskemalla vakioarvolla 127 käsitelty kuva. OpenCV -kirjastossa Otsun binäärisointi saadaan kutsuttua OTSU -funktiolla. (OpenCV team 2020a.)



Kuva 6. Otsun binäärisointi (OpenCV team 2020a)

3.2 Ääriviivojen tunnistus

Kuvien muotoja voidaan esittää yksinkertaisesti piirtämällä muodon reunojen ympärille viiva, joka rajaa kuvan. Ääriviivojen paikannus on hyödyllinen funktio muotojen analysointiin sekä esineiden havaitsemiseen ja tunnistamiseen. OpenCV -kirjaston sisältämillä findContours ja drawContours -funktioilla voidaan hakea ja rajata kuvissa esiintyviä muotoja. Parhaimman tarkkuuden saavuttamiseksi kuvan pitää olla binäärimuotoinen. Muotojen tulee olla myös selkeästi taustastaan erottuvia ja yhtenäisiä, jotta funktio toimii halutul-

la tavalla. Muotoja voidaan rajata kuvan 7 mukaisesti hakemalla viivojen kaikki pisteet tai tallentaa vain muodon ääriarvojen (x,y) koordinaatit. (OpenCV team 2020c.)

Kuva 7. OpenCV:n findCountour ja drawContour funktioiden toimintaperiaate (OpenCV team 2020c)

FindContours -funktioilla voidaan hakea samalla kertaa kaikki kuvassa esiintyvät muodot ja luoda niille hierarkia. Useiden muotojen tunnistuksessa hierarkia etenee ennalta määrätysti oikealta vasemmalle ja ylhäältä alas. Sisäkkäisistä muodoista muodostuu omat useampi tasoiset hierarkiat. Kaikista muodoista saadaan taulukkomuotoinen luettelo niiden suhteista toisiin muotoihin ja muodon koordinaatit. (OpenCV team 2020d.)

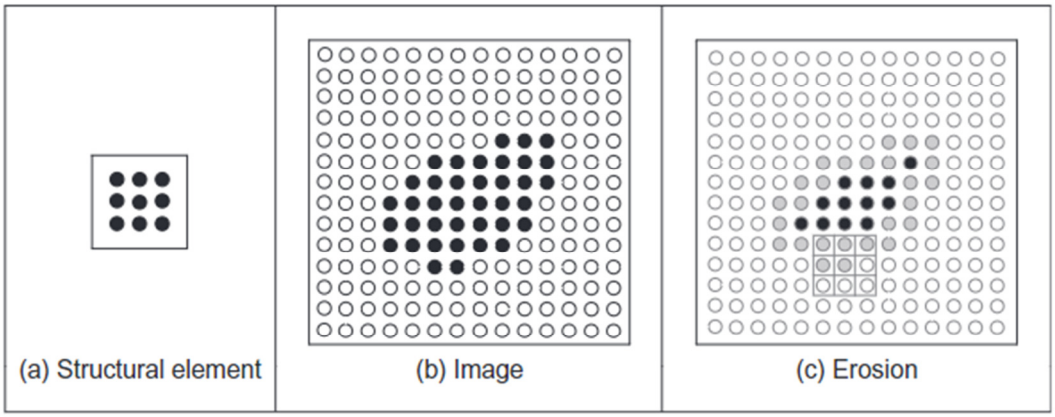
3.3 Morfologinen muokkaus

Morfologisessa kuvankäsittelyssä ideana on saada esille kuvan muoto muokkaamalla kuvan geometrisia rakenteita. Yleisimpiä morfologisia käsittelytoimenpiteitä ovat kuvan laajennus (eng. dilate) ja kuvan pienennys (eng. eroding). Harvemmin käytettyjä, mutta hyödyllisiä muunnosmuotoja, ovat kuvan avaaminen (eng. opening) ja sulkeminen (eng. closing). Yleisimmin morfologiset muokkaukset suoritetaan binäärimuotoisille kuville, mutta tarvittaessa niitä voidaan käyttää myös harmaasävykuvissa. (Howse ym. 2016, 115; OpenCV team 2020b.)

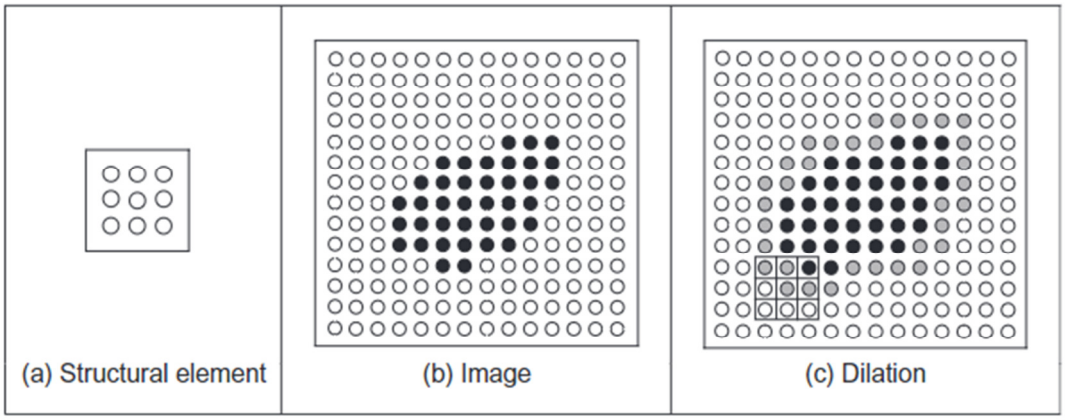
Kuvan laajennus ja pienennys eivät määrittele itsessään kuvan rakenne-elementin muotoa. Yleensä morfologinen muokkaus tuottaa vain pieniä muutoksia kuvan elementtiin, itse elementin koon määrätessä muutoksen voimakkuuden. Morfologinen muokkaus voidaan kuitenkin suorittaa useita kertoja peräkkäin halutun lopputuloksen aikaansaamiseksi. (Nixon & Aquado 2012, 126.)

Morfologisessa muokkauksessa kuvassa ajetaan halutun muotoista ydintä, jonka ankkuripiste on yleensä ytimen keskipiste. Ytimen alueelta lasketaan aina suurin kuvapistearvo, jolla ytimen keskipisteen arvo korvataan. Tämä toimenpide saa luonnollisesti kuvaa pienennettäessä kirkkaat alueet kasvamaan ja tummat katoamaan. Kuvaa suurennettaessa vaikutus on päinvastainen. (OpenCV team 2019.)

Kuvissa 8 ja 9 on kuvattuina morfologisen pienennyksen ja laajennuksen toimintaa. Jokainen piste kuvaa yhtä kuvapistettä ja mustat kuvapisteet ovat kuvan elementti. Kuvassa ajetaan 3x3 kuvapisteen kokoista neliön muotoista ydintä ja voidaan havaita, kuinka operaatio vaikuttaa kuvaan. (Nixon & Aquado 2012, 125; Howse ym. 2016, 115.)

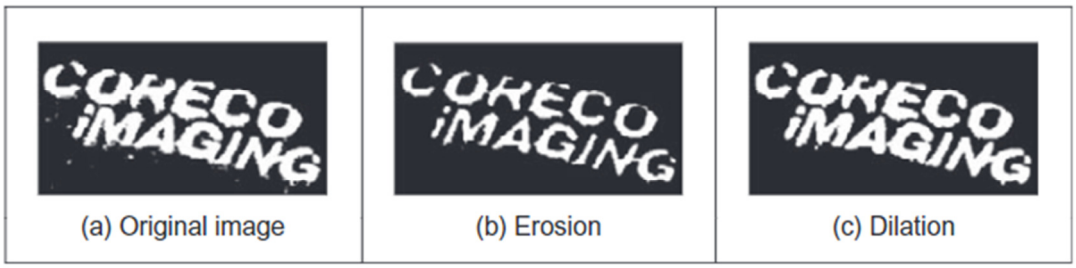


Kuva 8. Kuvan pienennyksen toimintaperiaate (Nixon & Aquado 2012)



Kuva 9. Kuvan laajennuksen toimintaperiaate (Nixon & Aquado 2012)

Kuvassa 10 on esitetty morfologisen kuvanmuokkauksen lopputulokset kuvissa, jossa kuvan pienennys (b) ja kuvan laajennus (c) ovat verrattuna alkuperäiseen kuvaan (a). Pienennys-operaatiolla kuvasta saadaan selkeämpi ja sillä voidaan erottaa toisissaan kiinni olevia kuvan elementtejä ja hävittää kokonaan kuvan pieniä elementtejä. Laajennus-operaatio taas vahvistaa kuvan elementin rakennetta ja korjaa siinä mahdollisesti olevia aukkoja. (Nixon & Aquado 2012, 125.)



Kuva 10. Morfologinen muokkaus (Nixon & Aquado 2012)

OpenCV -kirjastosta löytyy seitsemän funktiota kuvan morfologiseen muokkaukseen. Perusfunktioina sisältyvät kuvan laajentaminen ja pienentäminen. Muunnosmuotoisista funktioista löytyvät muun muassa funktioiden muunnosmuodot avaamiseen ja sulkemiseen. Perusfunktiot saadaan auki funktioilla cv.erode() ja cv.dilate(). (OpenCV team 2020b.)

4 Scikit-learn -kirjasto

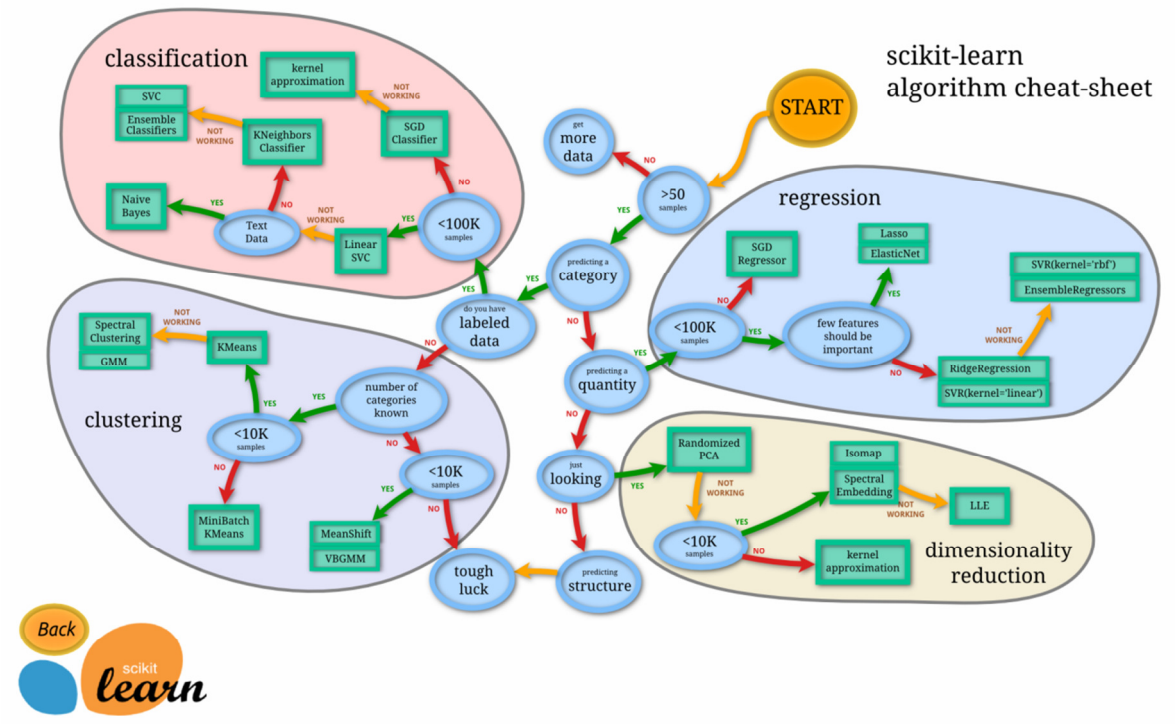
Scikit-learn on vuonna 2007 alkunsa saanut avoimen lähdekoodin koneoppimiskirjasto, jota rahoittavat useat suuret tietotekniikka-alan yritykset. Scikit-learn -kirjasto sisältää koneoppimiseen soveltuvien algoritmien lisäksi moduuleja ominaisuuksien purkamiseen, datan käsittelyyn ja mallien arviointiin. SciPy-kirjaston laajennuksena Scikit-learn tukee myös suosittuja NumPy ja Matplotlib Python -kirjastoja. (Hackling 2014, 16; Scikit-learn developers 2020g.)

Scikit-learn -kirjasto on suosittu osana akateemista tutkimusta hyvin dokumentoidun, helppokäyttöisen ja monipuolisen ohjelmistorajapintansa ansiosta. Eri algoritmien käyttö onnistuu helposti muuttamalla vain muutamaa koodiriviä, joka tekee käytöstä tehokasta ja helposti muunneltavaa. Monet Scikit-learn -kirjaston algoritmeista ovat myös nopeita ja skaalautuvat suurillekin tietojoukoille. (Hackling 2014, 16.)

Scikit-learn -kirjaston algoritmeja käytetään usein jonkin Jupyter Notebook -tyyppisen sovelluksen kautta. Tietokoneille asennettuihin Notebooks -sovelluksiin asennetaan erikseen käytettävät kirjastot, joita halutaan käyttää. Jotkin pilvipalvelut, kuten Microsoft Azure Notebooks ja Google Colaboratory, tarjoavat kuitenkin tehokkaan alustan Python-pohjaiseen koneoppimiseen. Tämän kaltaiset pilvipalvelut sisältävät usein esiasennettuna yleisimmät koneoppimiskirjastot, mukaan lukien myös Scikit-learn -kirjaston. Yleisimpiä kirjastoja ja niiden osia voidaan asentaa usein myös itse, elleivät ne ole esiasennettujen kirjastojen ja niiden osien joukossa. (Ingargiola 2015; Microsoft 2019; Google 2020.)

4.1 Algoritmit

Useimmiten suurin työ koneoppimismallin rakentamisessa saattaa olla oikean koneoppimismenetelmän löytäminen lukuisten eri menetelmien joukosta. Eri menetelmillä on pääsääntöisesti vahvuutensa erityyppisten tietojen ja ongelmanratkaisujen käytössä. Scikit-learn -kirjaston menetelmille on kuitenkin luotu yksinkertaistettu kuvan 11 mukainen vuokaavio-tyyppinen ohje, jolla voi aloittaa kokeilut löytääkseen kulloiseenkin käyttötarkoitukseen parhaiten sopivan ratkaisun kaikkien vaihtoehtojen joukosta. Scikit-learn -kirjasto algoritmit ovat jaettu päätasolla ohjatun ja ohjaamattoman oppimisen luokkiin, joista luokittelu (eng. classification) ja regressio (eng. regression) sopivat parhaiten ohjattuun oppimiseen ja klusterointi (eng. clustering) ohjaamattomaan oppimiseen. (Hackling 2014; Scikit-learn developers 2019a; Scikit-learn developers 2019b.)



Kuva 11. Yksinkertaistettu ohje algoritmien valinnan vertailuun (Scikit-learn developers 2019b)

Ohjatun oppimisen algoritmeissa käytetty regressio ennustaa määrää. Regression perustana on regressioanalyysi, joka numeerisen datan ennustamiseen käytettävä tilastollinen malli. Regression vahvuutena on ennustaa tuloksen jatkuva arvo. Regressiota voidaan käyttää esimerkiksi henkilön tulojen ennustamiseen iän ja koulutustason perusteella. (Bonaccorso 2018, 36-49; Brownlee 2019.)

Ohjatun oppimisen algoritmeissa käytetty luokittelu ennustaa tuloksien luokkaa ja tuloksena saadaan aina luokan ennustettu arvo. Luokittelu vaatii toimiakseen aina vähintään kaksi luokkaa, johon data on jaettu. Data voi sisältää kuitenkin myös useampia luokkia yhtä näytettä kohti, jolloin sitä kutsutaan usean luokan luokitteluongelmaksi (eng. multi-class classification problem). Luokittelun ennustavan mallin kykyä voidaan arvioida useilla tavoilla, mutta yleisin tapa on laskea luokittelun osumisprosenttia. Luokittelu sopii käytettäväksi hyvin esimerkiksi kuvantunnistuksessa. (Bonaccorso 2018, 36-49; Brownlee 2019.)

Ohjaamattoman oppimisen algoritmeissa käytetty klusterointi ryhmittelee datan sisältöä ryhmiksi eli klustereiksi. Ryhmittely tapahtuu ilman tiedossa olevaa luokkaa jakamalla samanlaisia arvoja omiksi ryhmikseen. Klusterointia voidaan käyttää esimerkiksi tietojen analysoinnissa, etsittäessä olemassa olevan ja uuden datan yhteneväisyyksiä. (Bonaccorso 2018, 36-49; Brownlee 2019.)

4.1.1 Algoritmin mallin sovittaminen

Koneoppimisalgoritmilla luodessa koneoppimismallia se tulee sovittaa (eng. fitting) opetusaineiston dataan. Data sisältää yleensä tuloksien suhteen hajontaa, joten sovittamisella pyritään hakemaan algoritmille datasta malli, jolla on taipumus yleistää datan pohjalta ennustettuja tuloksia. Ennustettaessa opetuksen ulkopuolisella datalla, antaa yleistetty malli riittävän tarkkoja tuloksia, jotta sitä voidaan hyödyntää ennustamiseen. (Bonaccorso 2018, 23-28.)

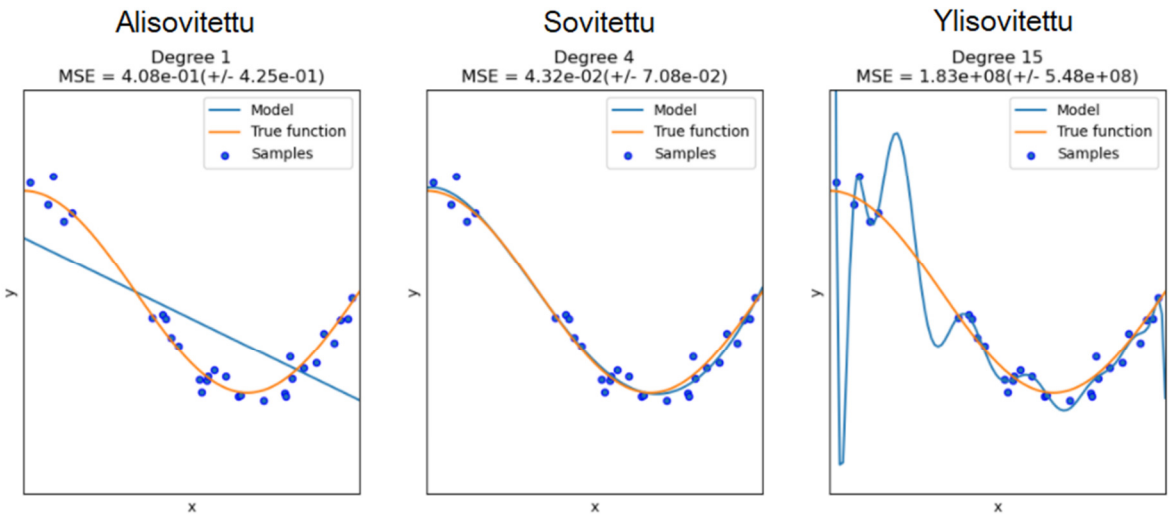
Algoritmin sovittamisella voidaan arvioida käytettävän algoritmin ja sen arvojen soveltuvuutta ennustuksen toimivuuteen ja tuloksien osuvuuteen. Väärin sovitettu malli johtaa aina vääristyneisiin tuloksiin, joten mallin sovittaminen käytettävään dataan on erittäin tärkeä vaihe koneoppimismallin luomisessa. Yleisin virhe mallin sovituksessa on niin sanottu ylisovittaminen, jolloin luotu koneoppimismalli seuraa liian tarkasti opetuksessa käytetyn datan arvoja. Ylisovitetulla mallilla ennustettaessa opetusaineiston datassa olleet satunnaiset vaihtelut aiheuttavat vääristymiä opetusdatan ulkopuolisella datalla ennustettaessa. Toinen ääripää on alisovitettu malli, joka ennustaa liian yleistäviä tuloksia verrattuna opetusdatana käytetyn aineiston tarkkuuteen. (Bonaccorso 2018, 23-28.)

Algoritmin sovittamista voidaan kuvata ja arvioida helpoiten graafisesti, yleisimmin kaksiulotteisena kuvaajana. Kuvaajissa esitetään opetusvaiheessa käytetyn datan arvoja esimerkiksi pisteinä ja algoritmin mallin sovitusta kuvataan viivoilla tai alueilla. Kuvaajista voidaan tehokkaasti havainnoida sovituksen tila ja arvioida mallin toimintaa. (Bonaccorso 2018, 23-28.)

Kuvassa 12 on esitetty koneoppimismallin yli- ja alisovittamisen ongelmat. Siniset pisteet kuvaavat näytteiden tuloksia ja oranssi viiva kuvaa tuloksia mukailevaa todellista funktiota. Sininen viiva kuvaa erilaisten funktioiden kykyä sovittaa malli vastaamaan oranssilla kuvattua viivaa. Kuvan 12 alisovitettu -osiossa, lineaarinen funktio ei riitä sovittamaan näytteitä, vaan jakaa näytealueen epätarkasti puoliksi. Koneoppimismallin tarkkuus ei siis riitä vastaamaan tällaisessa tilanteessa näytteiden tuloksia, jolloin mallin tulokset poikkeavat todellisuudesta. (Scikit-learn developers 2020a.)

Kuvan 12 sovitettu -osiossa, polynomi onnistuu sovittamaan näytteet lähes täydellisesti mukailemalla näytteiden arvoja yleistävää, oranssilla merkittyä, funktiota. Tällä tarkkuudella sovitettu koneoppimismalli onnistuu ennustamaan tarkasti opetusdatan ulkopuolisia tuloksia. Sovitus voi kuitenkin helposti muuttua ylisovitukseksi, jota havainnollistaa kuvan 12 ylisovitettu -osio. Polynomi kulkee useiden näytteen pisteiden kautta, jolloin mallin sovitus seuraa liian tarkasti opetusdataa. Kuvan mukaan sovitettu algoritmi ennustaa tarkasti

opetusaineiston dataa, mutta opetusdatan ulkopuolisten arvojen ennustettavuus ei ole tarkka. (Scikit-learn developers 2020a.)

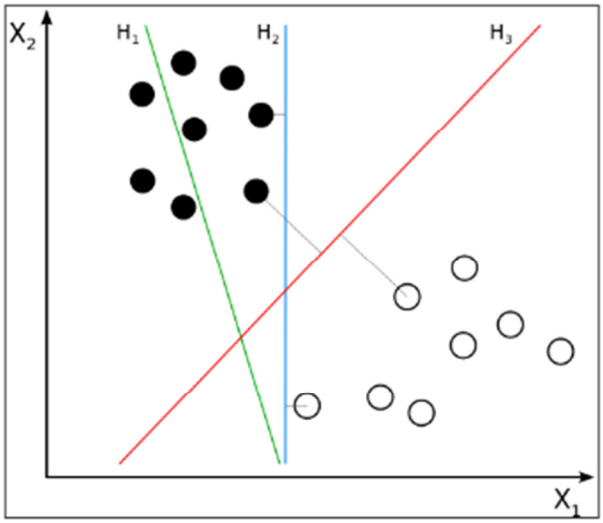


Kuva 12. Mallin sovitus kuvattuna graafisesti (mukailtu Scikit-learn developers 2020a)

4.1.2 Tukivektorikone

Tukivektorikoneet (eng. support vector machines), eli lyhenteenä SVM, ovat ohjattuja oppimismenetelmiä. Tukivektorikoneissa kukin tietoyksikkö havainnollistetaan pisteenä n-ulotteisessa tilassa, joissa arvo n kuvaa ominaisuuksien lukumäärää. Tavoitteena tukivektorikoneilla on saada hypertaso kulkemaan datajoukkojen välissä mahdollisimman laajan aukon läpi. (Garreta & Moncecchi 2013 25-26; Hamid & Sjarif 2019, 2.)

Kuvassa 13 on esitetty kaksi ominaisuutta kaksiulotteisessa tilassa (X1 ja X2). Mustat ja valkoiset pisteet kuvaavat luokkia, jotka kuvassa on erotettu lineaarisilla hyper-tasoilla (H1, H2 ja H3). Kuvassa 13 H3-hypertaso on ainoa, joka erottaa luokat parhaalla mahdollisella tavalla, ollen kauimpana molemmista luokista. (Garreta & Moncecchi 2013, 26.)

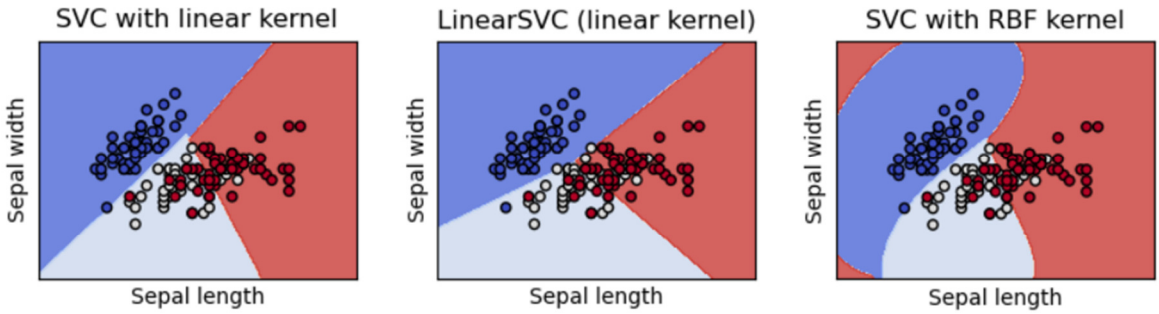


Kuva 13. Tukivektorikoneella suoritettu luokkien jako (Garreta & Moncecchi 2013)

Tukivektorikoneet soveltuvat tehtäviin, joita käytetään luokitteluun, regressioon tai poikkeamien havaitsemiseen. Tukivektorikoneet toimivat tehokkaasti moniulotteisissa tiloissa, joissa ulottuvuuksien määrä on suurempi, kuin näytteiden lukumäärä. Moniulotteisia tiloja löytyy esimerkiksi kuvantunnistuksessa, sillä kuvan jokaista pistettä pidetään erillisenä ulottuvuutenaan. Jokaisen ulottuvuuden ollessa oma tilansa on lopputuloksena erittäin moniulotteinen tila. (Garreta & Moncecchi 2013, 26-27; Scikit-learn developers 2020b.)

Tukivektorikoneiden haittana ovat ylisovittamisen riski tilanteissa, joissa luokkien määrä on paljon suurempi kuin näytteiden määrä. Myös pienillä datamäärillä algoritmin arvojen oikein valitseminen on tärkeää, jotta ennusteista saadaan luotettavia. Tukivektorikoneet eivät mahdollista myöskään suoraan todennäköisyysennusteita, vaan ne lasketaan käyttäen viisinkertaista ristiinvalidointia. (Hamid & Sjarif 2019, 2; Scikit-learn developers 2020d.)

Scikit-learn -kirjasto sisältää tukivektorikoneista erilaiset algoritmit käytettäväksi luokittelussa ja regressiossa. Luokitteluun Scikit-learn -kirjasto sisältää useita eri tukivektorikoneiden variaatiota, jotka kuuluvat Scikit-learn -kirjaston ryhmään SVC (Support Vector Classification). Algoritmeista SVC, SVC RBF, LinearSVC ja NuSVC sopivat parhaiten pienten datajoukkojen ennustamiseen. Ne sisältävät kuvan 14 mukaisesti luokittelualueiden suhteen pieniä eroja ja ovat parametreiltaan laajalti säädettävissä. (Scikit-learn developers 2020b.)



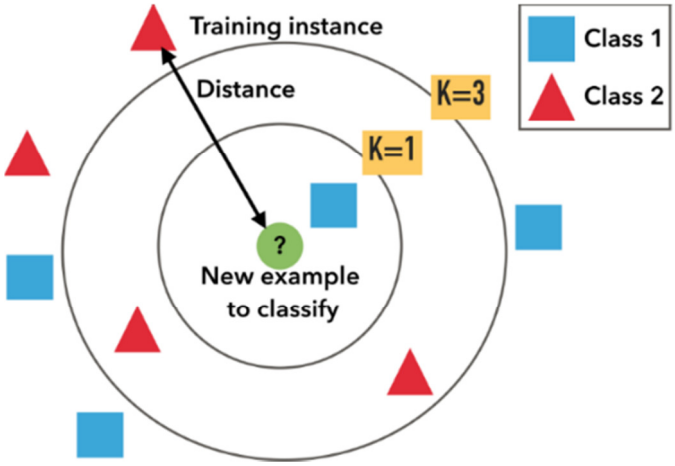
Kuva 14. Scikit-learn SVC-algoritmien eroja (Scikit-learn developers 2020b)

4.1.3 K:n lähimmän naapurin menetelmä

Yksi koneoppimisen käytetyimmistä algoritmeista on K:n lähimmän naapurin menetelmä (eng. K-Nearest Neighbor Classification), josta lyhenteenä käytetään usein myös nimitystä KNN. Scikit-learn -kirjasto sisältää algortimeja K:n lähimmän naapurin menetelmään, joista luokittelun osalta KNeighborsClassifier on käytetyin ja soveltuu parhaiten moniulotteiseen kuvantunnistukseen. (Navlani 2018; Scikit-learn developers 2020c.)

K:n lähimmän naapurin menetelmän toiminta perustuu arvopisteiden sijaintiin. Koneoppimismallin arvojen etäisyyttä verrataan ennustettavaan dataan ja näin ollen saadaan pisteiden erotus, jonka pohjalta luokitellaan arvo. K:n lähimmän naapurin menetelmää muistaa kaikki opetustilanteessa saamansa arvot ja epäparametrisenä järjestelmänä se onnistuu usein luokitustilanteissa, joissa päätösraja on hyvin epäsäännöllinen. K:n lähimmän naapurin menetelmä on hyvin monipuolinen ja se sopii sekä luokittelu-, että regressiotyyppisiin ongelmiin. Monipuolisuutensa ja yksinkertaisuutensa takia sitä käytetäänkin hyvin moninaisissa sovelluksissa rahoituslaskennasta kuvantunnistukseen. (Navlani 2018; Scikit-learn developers 2020c.)

Kuvassa 15 on esitetty K:n lähimmän naapurin menetelmän toimintaa. Algoritmille annettujen parametrien mukaan kuvassa vihreälle pallolle haetaan luokiteltua arvoa ympärillä olevien punaisten kolmioiden ja sinisten neliöiden etäisyyksien mukaan. K:n arvon suurentuessa määräytyy luokiteltava arvo suuremman ympärillä olevan opetusjoukon mukaan. (Bronshtein 2017.)

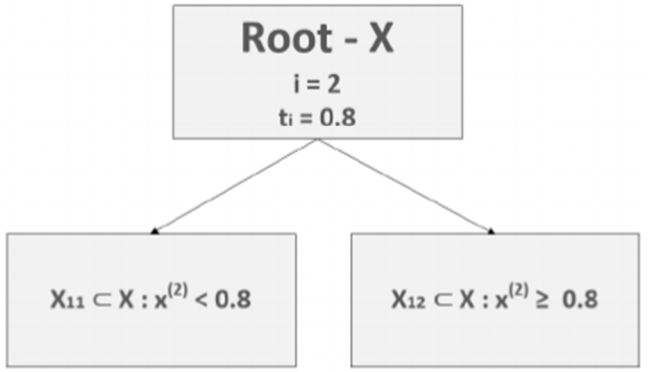


Kuva 15. K:n lähimmän naapurin menetelmän toiminta (Bronshtein 2017)

4.1.4 Satunnainen metsä

Satunnainen metsä (eng. random forest) on ohjatun oppimisen koneoppimismalli, joka perustuu päätöksentekopuihin. Sitä käytetään pääasiassa luokitteluun, mutta se toimii myös regressiossa. Se muistuttaa tavanomaista hierarkkista päätöksentekoprosessia, jossa valitaan paras lopputulos äänestämällä. Yksittäisillä päätöksentekopuilla on tyypillisesti suuri vaihteluväli ja taipumus ylisovittamiseen. Tämän takia satunnainen metsä luo datan arvojen pohjalta useita päätöspuita, joiden lopputuloksista lasketaan keskiarvo. (Bonaccorso 2018, 278-284.)

Kuvassa 16 on esitetty satunnaisen metsän puun hierarkia, jossa opetusdata X on jaettu kahteen osajoukkoon. Osajoukoilla X11 ja X12 on ominaisuus i=2, joka on suurempi tai pienempi, kuin kynnys ti=0.8. Luokittelupäätökset jatkavat opetusdatan halkaisua, kunnes lehdet sisältävät yhteen luokkaan kuuluvia näytteitä. Tällä tavoin data kulkee puun läpi ja saavuttaa lopullisen luokitellun arvonsa. (Bonaccorso 2018, 278.)



Kuva 16. Satunnaisen metsän luokittelijan toiminta (Bonaccorso 2018)

Scikit-learn -kirjasto sisältää useita algoritmeja satunnaiselle metsälle, joista RandomForestClassifier -algoritmi on käytetyin. Alkuperäisestä, vuonna 2001 kehitetystä, satunnaismetsän rakenteesta poiketen Scikit-learn -kirjaston RandomForestClassifier -algoritmi yhdistää luokittelijat keskittämällä niiden todennäköisyysennusteen sen sijaan, että jokainen luokittelija äänestäisi yhtä luokiteltua arvoa. (Scikit-learn developers 2020d; Scikit-learn developers 2020e.)

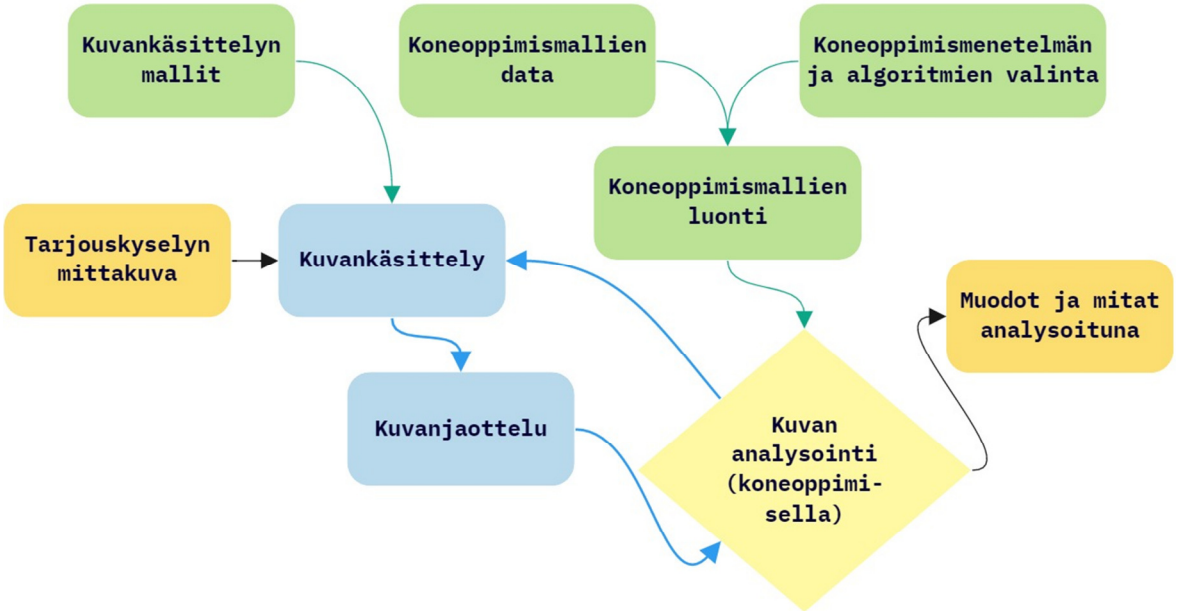
5 Tarjouskyselyn kuvien analysointi

Kuvien analysointi tarjouskyselyn käsinpiirretyistä dokumentista oli monivaiheinen prosessi, joka vaihteli huomattavasti pelkästään jo yhden analysoitavan tuotetyypin sisällä. Käsinpiirrettyjen kuvien muodot ja tyylit olivat vaikeimpia hahmottaa, sillä tietokoneella piirretyistä ja vakiomuotoisista kuvista poiketen käsinpiirrettyjen kuvien variaatiot perustuivat niitä piirtävän henkilön käden motoriikkaan ja piirtotekniikkaan. Tässä opinnäytetyössä keskityttiin analysoimaan yhdentyyppisen lasijärjestelmän tarjouskyselykuvia. Kuvat olivat käsinpiirrettyjä U- ja L-mallisia mittakuvia tärkeimpine mittoineen ja tietoineen.

Keskittyminen vain yhden tyyppisen tuotteen, yleiseen piirtotapaan, ei kuitenkaan rajannut kuvien analysointimahdollisuuksia jatkossa muiden tuotteiden ja piirtotapojen osalta. Opinnäytetyössä oli tarkoitus luoda kuvien analysointitapa, jonka käyttö mahdollistaisi samantyyppisen prosessin hyödyntämisen myös lukuisille muille tuotteille ja piirtotavoille.

5.1 Analysointiprosessin kuvaus

Opinnäytetyön aiheeseen liittyvän teoriaan perehtymisen aikana syntyi melko selkeä käsitys analysointiprosessin toteutustavasta. Analysointiprosessista luotiin kaavio, jonka pohjalta työtä lähdettiin toteuttamaan ja toimintaa testaamaan. Toteutus jakautui päätasolla kolmeen osaan, jotka olivat datan hankinta, kuvankäsittelyn luominen ja koneoppimismallien luominen. Tarkempi prosessin kuvaus noudatteli kuvan 17 mukaista kaaviota.



Kuva 17. Opinnäytetyössä käytetyn analysointiprosessin kuvaus

5.2 Tiedon koonti

Opinnäytetyössä käytetty data oli tyypiltään sellaista, jonka välityskanava oli pääasiallisesti liitetiedostoina sähköpostin välityksellä. Yritysten päivittäisten rutiinien lomassa syntyvän datan järjestelmällisen ja päämäärätietoisen keräämisen tiedettiin olevan opinnäytetyön datalähteenä riittämätön. Tällaista tarvittavaa dataa olivat esimerkiksi sähköpostit liitetiedostoineen, joiden säilyttämisajan todettiin olevan lyhimmillään vain tiedoston käsittelyyn tarvittava aika. Sähköpostien jatkohyödynnettävyys on ollut aiemmin olematon, joten oli melko ymmärrettävää, ettei niiden säilyttämiseen oltu panostettu. Koneoppimisen aikakaudella olisi kuitenkin ollut järkevää kerätä ja tallentaa dataa jo jonkin aikaa, mahdollista jatkohyödynnettävyyttä silmällä pitäen.

Tarjouskyselyjen mittakuvia saatiin kuitenkin niin paljon, että niiden yhtenäinen perusrakenne saatiin selville. Kerätyn datan määrä ei kuitenkaan vastannut koneoppimisen kuvantunnistukseen tarvittavia määriä, joten dataa jouduttiin luomaan alkuperäisten kuvien pohjalta lisää. Luotujen kuvarakenteiden osalta pyrittiin varioimaan myös erilaisia käden motorisia liikkeitä, jotta kuviin saatiin vaihtelua.

Kuvien analysointiohjelmaa varten luotiin myös kuvakirjastoja esimerkiksi muotojen ja numeroiden tunnistusta varten. Näiden kuvakirjastojen kuvat piirrettiin käsin varioimalla opetettavia muotoja, sillä valmiita koneoppimista varten luotuja datasettejä ei muodoista ollut olemassa. Kuvien leikkaus ja luokittelu suorittiin sitä varten luodulla omalla apuohjelmallaan.

5.3 Koneoppimismenetelmän valinta

Tämän työn pohjana olisi voinut käyttää kaikkia yleisimpiä koneoppimismenetelmiä. Työ päätettiin kuitenkin tehdä ohjattua oppimista käyttäen. Syy ohjatun oppimisen käyttämiseen oli selkeä ja perustui kahteen asiaan. Ohjattu oppiminen soveltui hyvin tämän työn kaltaisien yksinkertaisien piirrosten tulkitsemiseen, sillä koneoppimisessa käytettävien opetus- ja testausdatojen luokat tunnettiin ennakolta ja niiden luokittelu oli mahdollista. Koulukursseilla saatujen ohjatun oppimisen perusteiden pohjalta tämä koneoppimismenetelmä oli myös ennestään hieman tuttu ja siitä oli hyvä jatkaa asiaan syventymistä.

5.3.1 Koneoppimiskielen ja kirjastojen valinta

Koneoppimiseen käytettävistä useista ohjelmointikielistä valittiin käytettäväksi Python, joka on tulkattava ohjelmointikieli. Pythonin valintaan ohjasivat sen suosio tieteellisessä laskennassa, yksinkertainen syntaksi ja lukuisat kirjastot. Python soveltui myös hyvin ko-

neoppimiseen, sillä siihen oli saatavana useita suosittuja koneoppimiskirjastoja. Koulun kursseilla koneoppimiseen oli perehdytty myös pääosin Pythonin kautta, joten se oli luonteva valinta käytettäväksi ohjelmointikieleksi.

Koneoppimisen tehokkaimmaksi toteutustavaksi varmistui olemassa olevien koneoppimiseen tehtyjen ohjelmointikirjastojen hyödyntäminen. Yksi saatavilla olleista, suosituimmista ja laajimmin dokumentoiduista kirjastoista oli Scikit-learn, joka vaikutti hyvältä työkalulta syvennyttäessä koneoppimiseen. Kirjaston testikäytön positiivisen tuntuman, kattavan dokumentaation ja laajan kirjallisen opasaineiston perusteella se valikoitui käytettäväksi tässä työssä.

Konenäköön perehdyttäessä löydettiin myös muita siihen kehitettyjä kirjastoja, joilla kuvista saatiin muokattua koneoppimiseen sopivia. Yhdeksi suosituimmista kirjastoista osoittautui OpenCV, jonka käyttö oli mahdollista myös Python-ohjelmointikielen kanssa. Kirjaston testikäytön positiivisen tuntuma, kattavan dokumentaation ja laajan kirjallisen opasaineiston perusteella myös se valikoitui käytettäväksi tässä työssä.

Opinnäytetyössä olivat käytössä myös Pythonin standardikirjaston ulkopuoliset kirjastot NumPy, Matplotlib, SciPy, Pandas ja Scikit-image. Näiden kirjastojen käytön todettiin olevan tärkeä osa hyvin toteutettua koneoppimista ja toisaalta niitä käytettiin myös kirjastojen ominaisuuksiin tutustumiseksi. Numpy -kirjaston avulla saatiin luotua ja muokattua taulukoita. Matplotlib -kirjasto puolestaan mahdollisti koneoppimisessa käytettävien kuvaajien luonnin. SciPy -kirjaston avulla saatiin muun muassa tallennettua lopputuloksia. Pandas -kirjaston avulla muokattiin taulukoita. Scikit-image -kirjaston avulla prosessoitiin kuvia.

5.4 Kuvankäsittely

Opinnäytetyössä käytetyt piirrokset olivat käsinpiirrettyjä ja muutettu digitaaliseen muotoon monitoimitulostimen skannerilla. Kuvat olivat rakenteeltaan usein vaihtelevan heikkotasoisia ja ennen kuvien hyödyntämistä koneoppimisessa oli niitä käsiteltävä laadun parantamiseksi. Tärkeää oli myös kuvien laatuerojen tasoittaminen, sillä koneoppimiseen perustuvan kuvantunnistuksen tasalaatuisten ja oikeiden tulosten vaatimuksena oli analysoitavien kuvien tasalaatuisuus.

Analysoitavat mittapöytäkirjat sisälsivät myös useita mittaustuloksia yhtä paperia kohden. Mittaustulokset kuvineen olivat ryhmiteltynä paperille melko hajanaisesti, mutta kuitenkin hieman irrallaan toisistaan. Ihmiselle kuvien tulkitseminen olisi ollut osakokonaisuuksien osalta helppo ja selkeä tehtävä, sillä ihminen olisi hahmottanut viereisten osien kuulumisen aina samaan osakokonaisuuteen. Koneelle haastetta toi paperilla olevien kuvien ha-

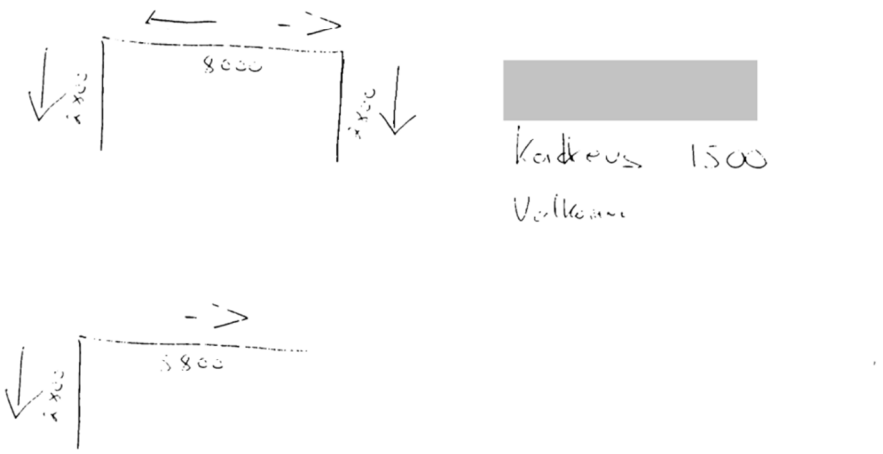
janainen sijoittelu ja eri osakokonaisuuksien hahmottaminen. Kuvien osien erotteluun löytyi kuitenkin toimenpide, jossa ensin osakokonaisuudet yhdistettiin ja sen jälkeen rajattiin.

5.4.1 Binäärikuva

Kuvien laadun parantaminen ja laatuerojen tasoittaminen aloitettiin kuvan binäärisoinnilla. Kokeilemalla erilaisia arvoja ja tapoja kuvan binäärimuotoon löytyi lopulta tapa, jolla kaikki testatut kuvat saatiin tasalaatuisiksi. Pohjana binäärisoinnille käytettiin kuvan 18 kaltaisesti OpenCV -kirjastoa, joka sisälsi binäärisointiin ja kuvanmuokkaukseen hyviä funktioita. Binäärisointi aloitettiin muokkaamalla kuvasta harmaasävykuva, sillä suoraan värikuvaa ei OpenCV -kirjaston avulla voitu binäärisoida. Tällä tavalla kuvaa binäärisoimalla muodostui kuitenkin ongelmaksi kuvassa esiintyvien, ei toivottujen, pisteiden korostuminen. Näiden pisteiden poistoon löytyi kuitenkin yksinkertainen ja tehokas tapa, eli niiden himmennys. Pehmentämällä kuvaa OpenCV -kirjaston Blur-funktiolla siinä esiintyvät pienet, ei toivotut, pisteet pehmenivät riittävästi, jotta kuvaa binäärisoimalla ne katosivat. Binäärisoinnin suhteen testattiin useita eri variaatiota, joista parhaimmaksi kuvan rakenteen siivoajaksi osoittautui OpenCV -kirjaston BINARY-funktio sopivilla arvoillaan. Binäärisoinnin lopputuloksena saadut kuvat olivat kuvan 19 kaltaisia, valkopohjaisia ja niistä erottuivat piirretyt muodot.

```
#OpenCV kuvan lukeminen
img = cv2.imread('kuva.jpg')

#OpenCV Blur
img = cv2.medianBlur(img,5)

#OpenCV harmaasävy
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#OpenCV Binäärikuva
ret,thresh1 = cv2.threshold(img,180,255,cv2.THRESH_BINARY)
```

Kuva 18. Binäärisoinnin OpenCV -kirjaston komentoja



Kuva 19. Muokattu ja binäärisoitu kuva. Kuvasta on piilotettu henkilökohtaiset tiedot

5.4.2 Muotojen yhdistäminen

Kuvien osakokonaisuuksia lähdettiin hahmottamaan yhdistämällä osakokonaisuuksien muodot selkeiksi omiksi alueikseen. OpenCV -kirjasto sisälsi tähän tarkoitukseen hyvän funktion, jolla kuvia voitiin muokata morfologisesti eli muokkaamalla kuvan geometrisia rakenteita. Rakenteiden muokkausta varten kuvan piti olla muunnettuna binääri- tai harmaasävykuvaksi. Tämä toimenpide oli kuitenkin jo tehty muokattaessa kuvaa kohdassa 5.4.1, joten sitä ei tarvinnut toistaa.

Morfologisen muokkauksen kuvan laajennus sopi hyvin kuvien osien yhdistämiseksi yhdeksi kokonaisuudeksi. Kokeilemalla erilaisia arvoja löytyi lopulta sopiva yhdistelmä, jolla osakokonaisuuksista saatiin yhtenäinen, mutta jolla viereiset osakokonaisuudet eivät vielä kuroutuneet yhteen kuvan 20 mukaisesti. Irrallisista tai kuroutuneista osakokonaisuuksista olisi aiheutunut kuvan jatkokäsittelyssä ongelmia, sillä ohjelman tarkoitus oli hahmottaa osakokonaisuudet yhtenevinä osioina. Irrallisista muotojen osakokonaisuuksista olisi aiheutunut kaksi erikseen analysoitavaa kuvaa, joten osa tiedoista olisi jäänyt puuttumaan. Yhteen kuroutuneiden muotojen osakokonaisuuksista olisi taas aiheutunut yhtenäinen muoto, jota ohjelma ei olisi osannut käsitellä oikein.

Morfologisen muokkauksen läpikäynyt kuva kulki omana kuvanaan, sillä sen lopputarkoitus oli toimia vain seuraavan vaiheen aputasona. Jotkin käsitellyt kuvat saattoivat sisältää myös alkuperäisessä paperissa olleiden ja kuvansiivouksen läpäisseiden roskapisteiden jäänteitä. Kuvassa 20 roskapisteiden jäänteet näkyvät yksittäisinä neliön muotoisina pisteinä.

Kuva 20. OpenCV -kirjastolla yhdistettyjä kuvan osakokonaisuuksia

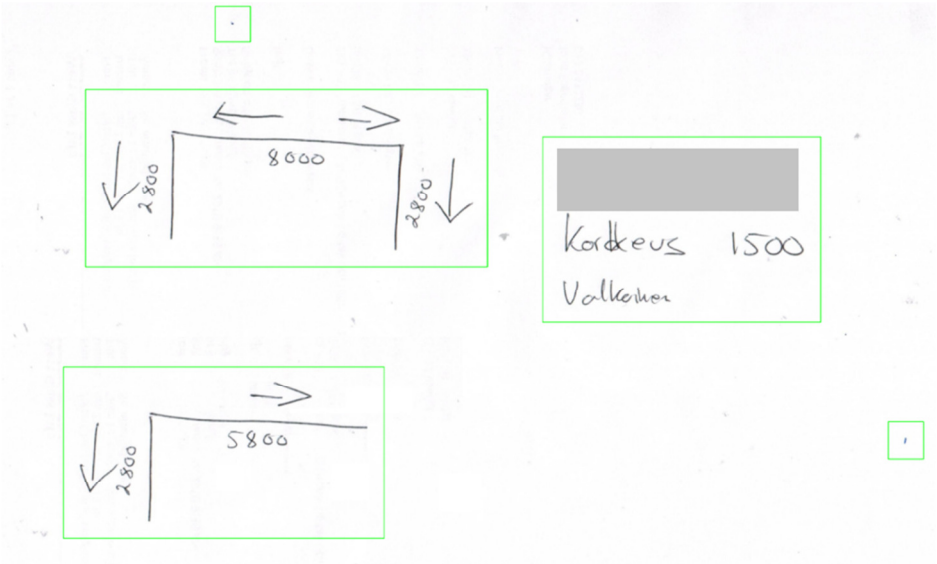
5.4.3 Muotojen rajaaminen

Muotojen yhdistämisen jälkeen kaikki erillisiksi tarkoitetut muodot olivat edelleen samassa kuvapohjassa. Tarkoituksena oli irrottaa muodot, joten muodoille oli suoritettava rajaus, jonka mukaan irrotus voitiin toteuttaa. OpenCV -kirjasto sisälsi findContours -funktion, jolla voitiin tunnistaa muotoja ja hakea niille äärirajat kuvan 21 kaltaisesti. Äärirajojen etsinnässä käytettiin muotojen yhdistämisessä, kohdassa 5.4.2, saatua aputasoksi luotua kuvaa. Aputasokuvasta haettiin kuvan muotojen reunojen ääriarvot, jotka yhdistettiin kuvankäsittelyn läpikäyneeseen kuvaan. Muotojen rajaaminen suoritettiin findContours -funktion vaakasuoraan rajaavaa suorakulmiota käyttäen, koska rajattavien kuvien muoto vastasi tätä parhaiten. FindContours -funktio vaati kuvan olevan binääri- tai harmaasävymuodossa, mutta kuvan ollessa jo käsitelty aiemmin samoilla vaatimuksilla, ei sitä tarvinnut tehdä uudelleen.

```
im = cv.imread('test.jpg')
imgray = cv.cvtColor(im, cv.COLOR_BGR2GRAY)
ret, thresh = cv.threshold(imgray, 127, 255, 0)
im2, contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
```

Kuva 21. OpenCV:n findContours -funktion esimerkkikoodi

Muotojen rajaamisen toiminta tarkastettiin findContours -funktion drawContours -funktiolla, jolla voitiin piirtää ääriviivat löydettyihin arvoihin. Viivojen piirtäminen auttoi hahmottamaan ja varmistamaan luotujen rajojen oikeaa sijaintia. Viivat piirrettiin tarkastuksena myös alkuperäisiin muokkaamattomiin kuviin, jolloin saatiin varmuus aiempien vaiheiden toiminasta, kuten kuvasta 22 selviää.



Kuva 22. Funktiot FindContours ja drawContours yhdistettynä muokkaamattomaan kuvaan

5.4.4 Muotojen irrotus

Muotojen irrotus toisistaan suoritettiin muotojen rajaamisessa saatujen rajojen mukaan. OpenCV -kirjaston findContours -funktion rajaamasta alueesta saatiin bounding.Rect -funktioon arvot muotojen irrotusta varten. Jokainen muoto irrotettiin kuvan 22 -tyyppisestä kuvasta ja lopputuloksena saatiin kuvan jokaisesta osakokonaisuudesta yksittäinen kuva, joka sisälsi osakokonaisuuteen sisältyneen muodon ja mittojen kuvauksen. Myös kuvassa olleet muut osat saatiin näin irrotettua erilleen.

5.5 Koneoppimismallin luonti

Muotojen tunnistus oli kuvien analysoinnissa yksi keskeisimmistä tehtävistä, joka vaati perehtymistä laajemmin koneoppimisen kuvantunnistukseen. Työkaluna kuvantunnistus osoitti käytettävyytensä heti ensimmäisten kuvantunnistuskokeilujen aikana. Kuvantunnistus melko yksinkertaisten viivamaisten piirrosten osalta ei tuntunut tuottavan ongelmia, mutta opetusdatan vähäisyys ja luokittelemattomuus nousivat ajallisesti haasteeksi. Dataa oli teoriassa helppo luoda ja luokitella lisää, mutta käytännössä yksittäisten kuvien editointi ja luokittelu veivät suhteettoman paljon työaikaa.

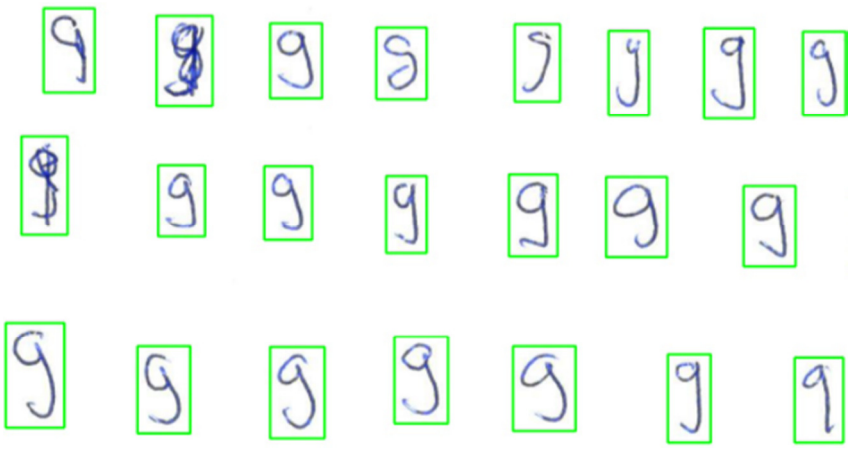
5.5.1 Datanluontityökalu

Koneoppimisen vaatiman datan määrä tuloksien tarkkuuden suhteen oli alusta lähtien tiedossa. Numeroiden ja kirjaimien suhteen Internetistä löytyi valmiita luokiteltuja kirjastoja, mutta opinnäytetyössä käytettyjen erikoisempien muotojen osalta ei valmiita kirjastoja

löytynyt. Mittakuvien analysoinnissa tarvittavat kuvat olivat muodoiltaan pääsääntöisesti niin erikoisia, ettei muodoista ole saatavilla valmiita kirjastoja. Datan luonnin helpottamiseksi muodostui idea luoda oma datanluontityökalu, joka perustui osittain aiemmin käytettyyn kuvanmuokkaukseen ja rajaukseen.

Käsinpiirretyn datan luonti aloitettiin luomalla paperille yhden luokan kuvioita. Muotojen tuli olla vaihtelevasti piirrettyjä ja hieman irrallaan toisistaan. Yhden A4-kokoisen paperin täyttäminen muodoilla vei vain muutaman minuutin ja kuvioita siihen saatiin mahtumaan, käsialasta riippuen, toista sataa. Muotoja piirrettiin kaikista opetettavista muodoista siten, että yhteen paperiin tuli vain yhden luokan muotoja. Muotoja sisältävät paperit skannattiin ja tallennettiin kuvina tietokoneelle.

Kuvien siivoaminen ja niissä olevien muotojen tunnistaminen sekä rajaaminen suoritettiin samoilla OpenCV -kirjaston työkaluilla, kuin kohdissa 5.4 esiteltiin. Kuvassa 23 on esitetty tällä tavalla siivottuja, tunnistettuja ja rajattuja muotoja. Muotojen irrotuksen jälkeen ne tallennettiin automaattisesti omiin kansioihinsa ja näin saatiin valmiiksi luokiteltu kuvia sisältävä datakirjasto. Kuvien tallennus kuvamuodossa omiin kuvakirjastoihinsa nähtiin tässä vaiheessa perustelluksi tulevaa käyttöä silmällä pitäen. Kuvamuotoisina luokiteltuja kuvia oli tarvittaessa mahdollista muokata vielä myöhemminkin ja niiden hyödyntäminen opinnäytetyön ulkopuolisiin projekteihin oli näin ollen myös varmistettu.



Kuva 23. Datanluontityökalun rajaamia muotoja

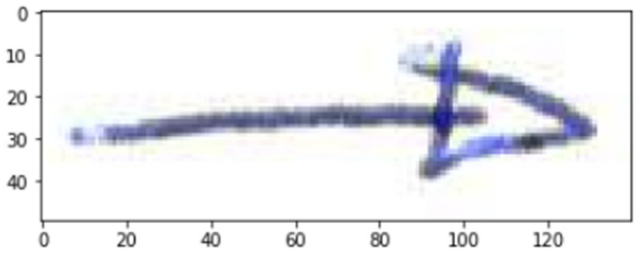
Kuvamuotoiset kirjastot piti muuntaa koneoppimisen käyttämiksi taulukkomuotoisiksi, numeerisia arvoja sisältäviksi, dataseteiksi. Datasettejä varten tallennetut kuvat luettiin ensin automaattisesti yksitellen ohjelmaan ja suoritettiin kuvan muunnos binäärikuvaksi käyttäen OpenCV -kirjaston funktiota. Tämän jälkeen kuvan kokoa pienennettiin käyttäen Scikit-image -kirjaston transform.resize -funktiota. Koon pienentämisen jälkeen kuvan pikseleiden arvot pyöristyvät harmaasävykuvan asteikolle, johtuen Skimagen transform.resize -

funktion toiminnasta. Muuttuneet binääriarvot korjattiin takaisin arvoihin 0 ja 1 käyttäen Numpy -kirjastoa.

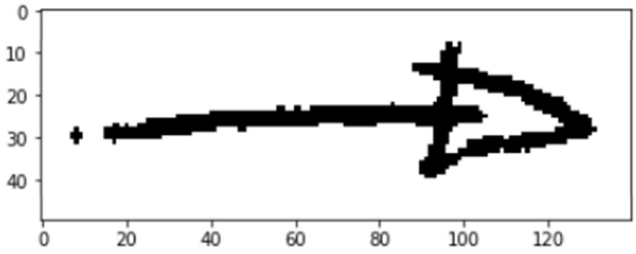
Datasetissä olevilla kuvilla oli oltava samat mittasuhteet, kuin kuvantunnistuksessa käytettävillä kuvilla ja tässä vaiheessa piti valita käytettävä kuvakoko. Kuvakoon valinta oli tasapainoilua kuvan tulevan ennustustarkkuuden ja taulukon koon välillä. Esimerkiksi kuvakoko 10x10 pikseliä antoi taulukon sarakkeiden määräksi 100 saraketta. Vielä suhteellisen pieni kuvakoko 100x100 pikseliä antoi kuvakooksi jo 10000 saraketta. Kokeilemalla huomattiin, että ennustettavien luokkien pieni määrä mahdollistaa suhteellisen pienien kuvakokojen käytön. Esimerkiksi numeroiden ennustamisen osalta todettiin kuvakoon 8x8 pikseliä riittävän hyvän ennustettavuuden pohjaksi. Kuvien koko vaihteli ennustettavan muodon mukaan ja kuvasuhteet pyrittiin säilyttämään lähellä alkuperäistä mittasuhdetta.

Kuvantunnistuksessa käytettyjen kuvien muunnokset olivat kuvan 24 kaltaisia, jossa on suoritettu nuolta kuvaavan muodon muunnos. Kuvassa 24 muunnos on havainnollistettu graafisesti alkuperäisestä (Original) kuvasta binäärikuvaan (Binary original) ja siitä 8x16 pikselin kokoiseen binäärikuvaan (Binary 8x16). Kuvassa 24 on havainnollistettu myös 8x16 kokoisen binäärikuvan matriisia (Binary array 8x16), arvojen 1 ollessa valkoisia ja arvojen 0 ollessa mustia kuvapisteitä.

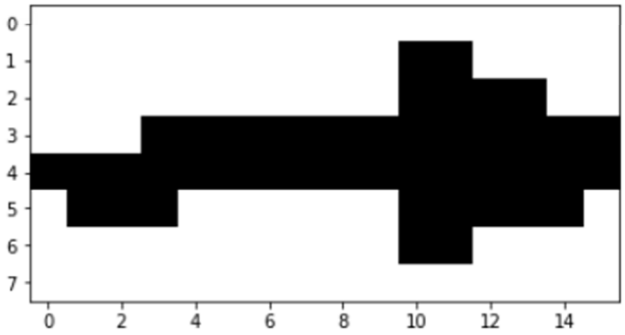
```
pl_mallit/nuolet/leikatut/nuox_0.png

Original
```



```
Binary original
```



```
Binary 8x16
```



```
Binary array 8x16
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1],
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

Kuva 24. Kuvan havainnollistettu pienentäminen ja muuttaminen binäärimuotoon

Numeroarvoiksi muutetut kuvat muunnettiin lopuksi matriisista taulukkomuotoon Numpy -kirjaston flatten ja reshape -funktioilla. Taulukkomuotoisten arvojen tallennus suoritettiin csv-formaatin tekstitiedostoon. Yhteen tiedostoon, eli datasettiin, tallennettiin kaikki samaa luokkaa edustavat kuvat, jokaisen rivin ollessa yksi edellä kuvatulla tavalla muunnettu muoto.

5.5.2 Luodut datasetit

Datanluontityökalulla luotiin omat luokitellut datasetit koneoppimisalgoritmien opetusta varten. Yhden A4-kokoisen ja toista sataa muotoa sisältävän paperin muuntaminen osoittautui melko nopeaksi ja vaivattomaksi prosessiksi, joten ohjelman testauksen ja käytön aikana valmistuivat datasetit kaikista opinnäytetyössä käytetyistä muodoista. Erilaisia datasettejä valmistui kolme kokonaisuutta liittyen mittakuvan analysoinnin kolmeen erilliseen kuvantunnistusosaan.

Kaikkiin datasetteihin pyrittiin saamaan melko tasainen jakauma eri muotoja, jottei mikään muoto olisi aiheuttanut koneoppimismallin luomisessa vääristymää ollen yliedustettuna opetustilanteessa. Käsinpiirretyt numerot sisältänyt datasetti sisälsi numerot nollasta yhdeksään. Mittakuvan osakokonaisuuksien muotojen analysointiin valmistunut datasetti sisälsi muodot L- ja U-mallisille muodoille. Mittakuvien merkkien analysoinnin datasetti sisälsi muodot numerosarjasta, nuolista ja viivarakenteista.

5.5.3 Datasetin jakaminen

Datasettien käyttöönotto tapahtui lukemalla luokittain tallennetut csv-tiedostojen arvot yhteen taulukkoon. Csv-tiedostojen tallentamisen yhteydessä niissä oleville arvoille ei ollut määritetty vielä omaa luokan arvoa, vaan ne annettiin tietojen csv-tiedostosta lukemisen yhteydessä. Datasettien sisältämille riveille annettiin jokaiselle oma luokkaa kuvaava arvonsa, yksi arvo luokkaa kohti. Kaikki samalla algoritmilla opetettavat datasetit liitettiin yhteen taulukkoon ja niille suoritettiin sekoitus ja jako opetus- ja testausdataseteiksi. Jakosuhteena käytettiin tämän kokoisille dataseteille melko vakiintunutta suhdetta, jossa 80 % datasta oli opetusaineistoa ja 20 % oli testausaineistoa. Validointiaineistoon jakamista ei suoritettu datan melko vähäisen määrän ja yksinkertaisen luokittelun takia.

5.5.4 Koneoppimisalgoritmin valinta

Ensimmäisten datasettien valmistuttua aloitettiin sopivien koneoppimisalgoritmien valinta ja vertailu. Valinnan lähtökohtana käytettiin alkuun kuvan 11 mukaista yksinkertaistettua Scikit-learn -kirjastoon liittyvää valintakaaviota ja teoriaan pohjautuvaa tietoa eri algoritmien ominaisuuksista. Lähtökohtana valinnalle oli löytää riittävän hyvä yksi algoritmi, joka arvoja muuttamalla, saataisiin kattamaan kaikkien opinnäytetyössä käytettyjen koneoppimismallien luominen.

Opinnäytetyössä käytetyistä käsinpiirretyistä muodoista koostuvat datasetit olivat pohjimmiltaan hyvin pitkälle toistensa kaltaisia viivamuotoja. Opetuksessa ja testauksessa läh-

dettiin liikkeelle numeroista luoduista dataseteistä, sillä niiden sisältämien luokkien määrä, kymmenen, oli datasettien suurin. Numeroiden muotojen opetus koettiin myös vaikeimmaksi ennustettavaksi, sillä käsinpiirrettyjen numeroiden muodot voivat olla lähellä toisiaan.

Koneoppimisalgoritmien valinnan suhteen päädyttiin tarkastelemaan neljää algoritmia, jotka kuuluivat kolmeen eri toimintaperiaatteeseen, mutta kuitenkin samaan luokittelu ryhmään. Luokittelun tiedettiin sopivan kuvien muotojen tunnistukseen, joten algoritmien ryhmä päätettiin pitää samana vertailun helpottamiseksi. Valinta toteutettiin kolmessa vaiheessa, jotka toistettiin jokaisen algoritmin kohdalla. Ensimmäisessä vaiheessa haettiin pistekuvaajaa apuna käyttäen koneoppimismalleille arvot, joilla saavutettaisiin todennäköisesti parhaat opetustulokset. Luoduille koneoppimismalleille annettiin tämän jälkeen ennustettavuustarkkuus, joita vertailtiin keskenään. Näistä parasta koneoppimismallia vertailtiin lopuksi muilla dataseteillä ja etsittiin riittävän hyvät arvot kattamaan kaikkien luotujen datasettien ennustukset.

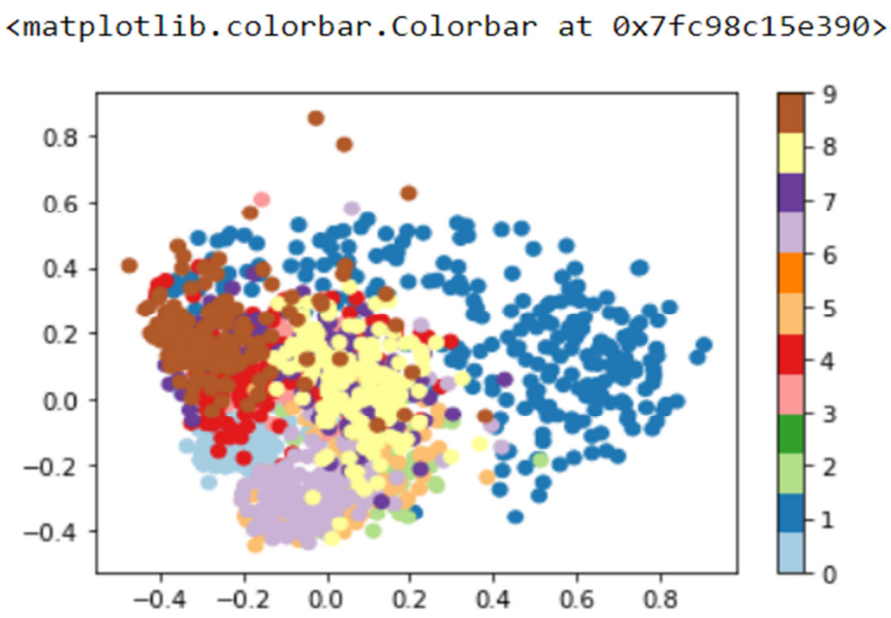
Lineaarisesta tukivektorikoneesta päätettiin valita kaksi eri versiota, joista toinen kulki Scikit-learn -kirjaston LinearSVC (Linear Support Vector Classification) nimellä ja toinen SVC RBF (Support Vector Classification Radial Basis Function) nimellä. Lineaarisen tukivektorikoneen tiedettiin soveltuvan kuvien muotojen tunnistukseen ja toimivan myös pienillä dataseteillä. Scikit-learn -kirjaston kuvan 11 mukainen yksinkertaistettu valintakaavio ohjasi myös kokeilemaan tukivektorikonetta opinnäytetyön tyyppisillä dataseteillä.

K:n lähimmän naapurin menetelmän Scikit-learn -kirjastossa nimellä KNeighborsClassifier kulkeva algoritmi valittiin toiseksi testattavaksi algoritmiksi. K:n lähimmän naapurin menetelmän vahvuuksiksi tiedettiin käsinpiirrettyjen kuvamuotojen tunnistuskyky. Tämä algoritmi oli yleisesti myös yksi suosituimmista, joten sen ottaminen vertailuun tuntui tämän kaltaisessa opinnäytetyössä hyvin luontevalta. Myös Scikit-learn -kirjaston kuvan 11 mukainen yksinkertaistettu valintakaavio ohjasi kokeilemaan K:n lähimmän naapurin menetelmää opinnäytetyön tyyppisillä dataseteillä.

Satunnainen metsä luokittelija (RandomForestClassifier) ei kuulu Scikit-learn -kirjaston yleisimpien algoritmien joukkoon ja se otettiin juuri tästä syystä kolmanneksi vertailtavaksi algoritmiksi. Algoritmin ominaisuuksiin ei dokumentaation ja kirjallisuuden mukaan kuulunut hyvä kuvan muotojen tunnistus. Tässä suhteessa satunnainen metsä luokittelija oli hyvä vertailupohja muille valituille algoritmeille.

5.5.5 Koneoppimismallien opetus ja testaus

Koneoppimismallin valintaa varten luokitellut numerodatasetit oli yhdistetty yhdeksi isoksi datasetiksi, jossa olivat kaikki numerot nollasta yhdeksään. Datasetistä luotiin Matplotlib -kirjaston Colorbar -funktiolla kuvan 25 mukainen visualisoitu pistekaavio. Pistekaaviosta oli helppo havainnoida eri luokissa olevien arvojen hajontaa ja suhteutumista toisiinsa kaksiulotteisessa tilassa. Kaavion tarkoitus oli toimia myös tukena koneoppimismallin opetuksessa. Datasetistä luotiin pistekaaviot koko datasetistä sekä erikseen opetus ja testaus osioista.

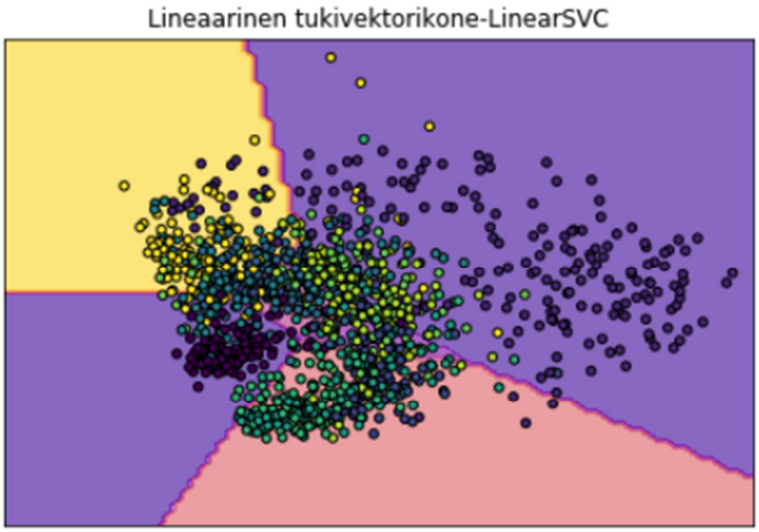


Kuva 25. Numerodatasetti luokittelun mukaan pistekaaviossa

Algoritmien sopivien arvojen etsiminen aloitettiin luomalla Matplotlib -kirjaston PyLab -moduulilla jokaisen algoritmin toiminnasta visuaalinen kuvaaja. Kuvaajassa yhdistettiin datasetin opetusaineistosta luotu pistekaavio ja algoritmilla luotu malli. Kuvaajasta käytettiin kahta versiota, joista toinen antoi yleisemmän kuvan kokonaisuudesta ja toinen versio keskittyi kuvaamaan tarkemmalla resoluutiolla keskellä olevaa tiivistä pisterykelmää. Kuvaajien perusteella pääteltiin algoritmilla luodun koneoppimismallin toimivuutta varsinkin ylisovituksen suhteen. Kuvaaja luotiin aina uudelleen algoritmiin tehtyjen muutosten jälkeen, jotta voitiin varmistua muutosten vaikutuksista algoritmin toimintaan. Algoritmien arvojen määrityksen jälkeen jokaisesta algoritmista luotiin koneoppimismalli, jolle annettiin ennustettavuustarkkuus.

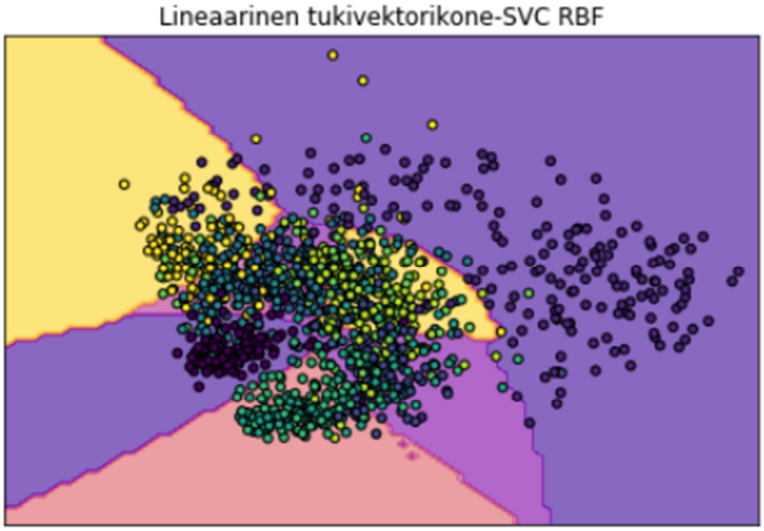
Lineaarisen tukivektoriluokittelijan algoritmin LinearSVC:n arvoja lähdettiin määrittämään etsimällä arvoja, joilla tukivektorit jakavat eri luokittelun omaavia arvoalueita mahdollisimman hyvin. Kuvan 26 mukaisesti todettiin lineaarisen tukivektoriluokittelijan tukivektorien

olevan nimensä mukaisesti hyvin suoria, joten niiden asemointi koneoppimismallia sovitettaessa ei mahdollistanut optimaalisimpia tuloksia. Mallin opetusdataan sovittamisen jälkeen, sen toimintaa vertailtiin vielä sovittamalla malli testausdataan. Mallin toiminta testausdatalla arvioitiin hyväksi ja se testattiin ajamalla koneoppimismallin ennustettavuustarkkuus. Luotu koneoppimismalli antoi testausdatalla ennustettavuustarkkuudeksi 92 %, joka oli hyvä tulos datasetin kokoon ja käsialan vaihteluun nähden.



Kuva 26. LinearSVC algoritmi tuotti lineaarisia vektoreita

Lineaarisen tukivektoriluokittelijan toinen algoritmi, SVC RBF, mahdollisti kuvan 27 mukaisen, hieman paremman, mallin sovittamisen. Mallin ääriarvoilla oli havaittavissa myös hieman selkeämpää ylisovittamista, kuin aiemmin testatulla LinearSVC algoritmilla. Mallin sovittamisen jälkeen suoritettu testausdatasetti antoi ennustettavuustarkkuudeksi 94 %, joka oli kaksi prosenttiyksikköä parempi, kuin LinearSVC:n vastaava arvo. Kuvassa 28 on esitetty tulosten numeerista vertailua, testauksessa käytetyn datasetin luokitellun arvon ja ennustetun arvon, yhteneväisyyksistä.



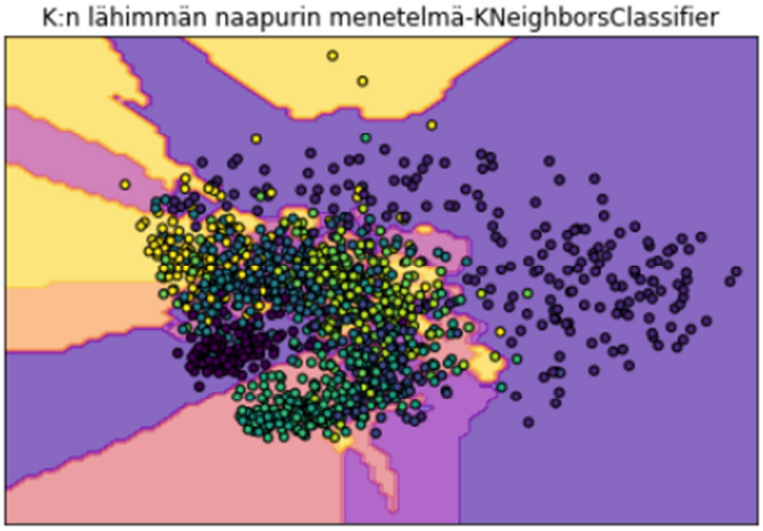
Kuva 27. SVC RBF algoritmin mallin sovitus

Out[61]:

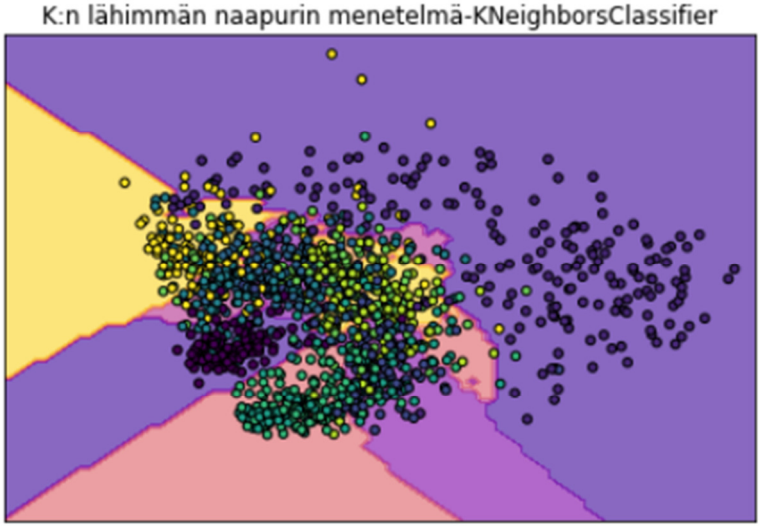
	Arvo	Ennustettu
0	7	7
1	8	8
2	1	1
3	9	9
4	9	1
5	6	6
6	8	8

Kuva 28. SVC RBF algoritmin mallin toimintaa testidatalla

K:n lähimmän naapurin menetelmän luokittelijan, KNeighborsClassifier -algoritmin, arvojen määritykseen jouduttiin hakemaan oikeita arvoja lukuisia kertoja. Algoritmilla huomattiin olevan kuvan 29 mukainen taipumus ylisovittaa koneoppimismalli hyvin herkästi. Ylisovittamiselle oli kuitenkin luontainen syy, sillä K:n lähimmän naapurin menetelmä muisti käytännössä kaikki oppimistilanteen arvot. Lopulta saatiin aikaan kuvan 30 mukainen malli, jossa ylisovittaminen saatiin kohtuulliseksi. Vertailemalla kuvan 27 algoritmin SVC RBF ja kuvan 30 algoritmin KNeighborsClassifier mallin sovitusta voidaan huomata mallien samankaltaisuus. Sovitetun mallin testaus datasetillä antoi ennustettavuuden tarkkuudeksi 89 %. Testauksen aikana huomattiin myös mallin ajamisen testidatalla kestävän huomattavasti pitempään, kuin lineaarisen tukivektoriluokittelijan algoritmeilla. Vertailun vuoksi kokeiltiin myös selkeästi ylisovitettua koneoppimismallin ennustettavuuden tarkkuutta, josta tulokseksi saatiin 93 %. Selkeästi ylisovitettu malli jäi näin ollen lineaarisen tukivektoriluokittelijan, algoritmin SVC RBF, tuloksen alle.

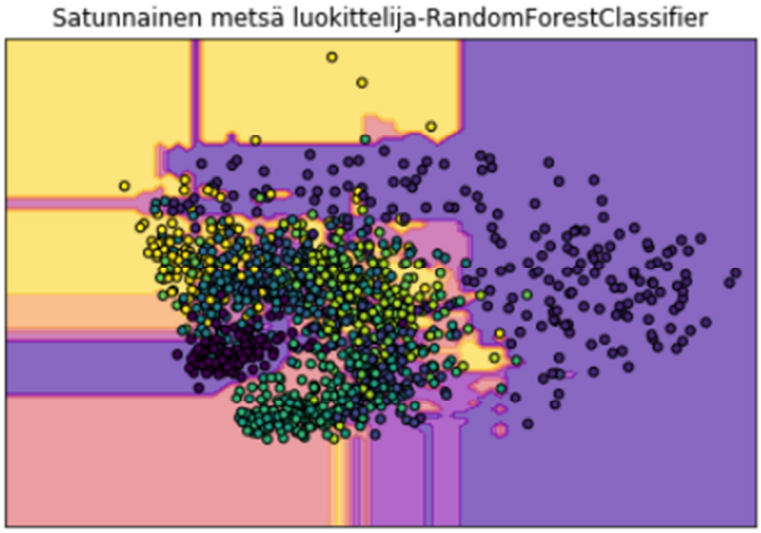


Kuva 29. K:n lähimmän naapurin luokittelija ylisovitti herkästi



Kuva 30. K:n lähimmän naapurin luokittelijan sovitus

Satunnainen metsä luokittelijan algoritmin RandomForestClassifier arvoja lähdettiin testaamaan kasvattamalla algoritmin arvoja. Hyvin nopeasti kävi selväksi, että algoritmilla oli kuvan 31 mukaisesti voimakas taipumus ylisovittaa mallia. Ylisovituksen poistaminen vaati lukuisia sovitusyrityksiä, ennen halutun mallin aikaansaamista. Testidatasetillä ajettu malli antoi lopuksi ennustettavuuden tarkkuudeksi 89 %, joka oli samaa tasoa K:n lähimmän naapurin luokittelijan kanssa.



Kuva 31. Satunnainen metsä luokittelijan ylisovitus

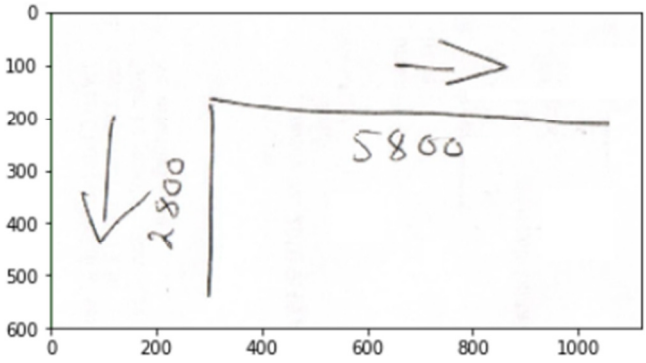
Numerodatasetillä testattujen neljän koneoppimisalgoritmin ennustuksien osumatarkkuudet saatiin sovitettua viiden prosenttiyksikön sisään. Algoritmeista yksi osoittautui kuitenkin osumatarkkuuden, nopeuden ja sovittamisen varmuuden suhteen muita paremmaksi. Lineaarisen tukivektoriluokittelijan toinen algoritmi, SVC RBF, valittiin käytettäväksi myös muiden luotujen datasettien koneoppimismallien luomiseen.

Loput luodut datasetit nuolista, muodoista ja numerosarjoista sisälsivät vain kahdesta kolmeen luokkaa, joihin muodot olivat luokiteltu. Koneoppimismallin sovitus toteutettiin numerodatasetin sovitusta vastaavalla tavalla ja valittu algoritmi saatiin sovitettua muiden datasettien opetusta varten jaettuihin osiin. Koneoppimismallien osumatarkkuuksiksi testidatalla saatiin hieman parempia, kuin numerodatasetillä saadusta ennustettavuuden tarkkuudesta.

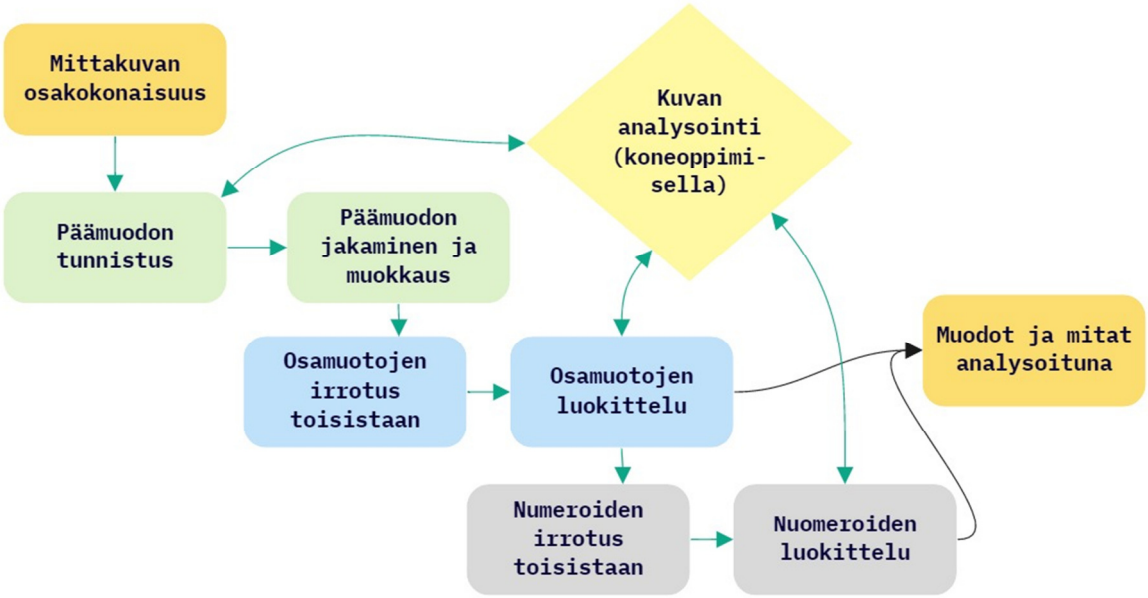
Koneoppimismallien opetuksen jälkeen luodut mallit nimettiin ja tallennettiin tulevaa käyttöä varten. Tiedostojen tallentamisessa käytettiin SciPy -kirjaston joblib -funktiota. Mallien sovitus-, opetus- ja testaustyökalua ei enää tämän jälkeen käytetty.

5.6 Osakokonaisuuksien analysointi

Mittakuvat muodostuivat kuvan 32 mukaisista osakokonaisuuksista, joista ilmeni rakenteen muoto, avautumissuunnat ja mitat. Osakokonaisuuksien analysointi suoritettiin kuvan 33 mukaisesti kolmiportaisena, jakamalla mittakuvaa aina pienempiin osiin ja tunnistamalla osien luokkia koneoppimisen avulla. Lopputuloksena saatiin alkuperäisen kuvan osakokonaisuudesta analysoitu lopputulos. Tuloksesta selvisivät kuvassa olleet pituusmitat ja avautumissuunnat mittamuodon jokaiselle sivulle.



Kuva 32. Alkuperäisestä kuvasta irrotettu osakokonaisuus



Kuva 33. Osakokonaisuuksien analysoinnin periaate

L- ja U-mallisissa mittakuvissa osa mittatiedoista oli sijoitettu pystyasentoon ja osa vaaka-asentoon. Eri asennoissa olevien mittatietojen tunnistus ei ollut mahdollista suoraan samoilla koneoppimismalleilla, sillä algoritmi oli opetettu tunnistamaan vain vaaka-asennossa olevia kuvia. Pysty- ja vaaka-asennossa olevien mittatietojen tunnistus olisi voitu tehdä useilla eri tavoilla, mutta tämän opinnäytetyön puitteissa päätettiin tehdä kuvien osien kääntäminen vaaka-asentoon hyödyntämällä opinnäytetyössä käytettyä kuvantunnistusta. Vaaka-asentoon kääntämistä varten oli tunnistettava ensin kuvissa esiintyvä mittakuvan L- tai U-muoto ja tämän jälkeen eriteltävä mittakuvan eri asennoissa olevat alueet.

Osakokonaisuuden muotojen tunnistusta varten oli aiemmin luotu käsinpiirretty datasetti, jolla oli mallinnettu OpenCV -kirjaston dilate -funktiolla muokattuja osakokonaisuuksien muotoja. Muodoissa yhdistyivät kuvan 34 kaltaisesti L- ja U-malliset mittakuvat ja niihin liittyvät mitat ja suuntanuolet. Muotojen tunnistukselle tuli näin ollen vain kaksi luokkaa,

joiden tunnistamisen ei oletettu tuottavan suurempia ongelmia. Muodoista oli luotu koneoppimismalli kohdan 5.6.3 mukaisesti.

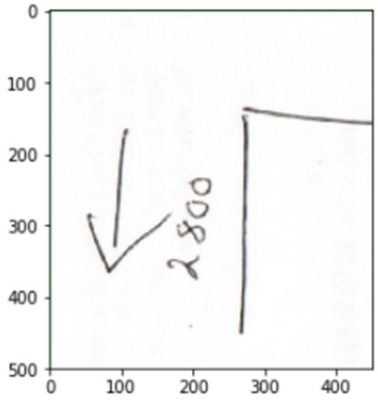
Osakokonaisuuden muotojen tunnistus toteutettiin muokkaamalla ensin kuvan 32 kaltaisia osakokonaisuuden sisältäviä kuvia OpenCV -kirjaston dilate -funktiolla. Muokatun kuvan L- tai U-muoto tunnistettiin tämän jälkeen aiemmin luodulla koneoppimismallilla. Vaikka koneoppimismalli oli opetettu käyttäen kuvan 34 kaltaisia käsinpiirrettyjä muotoja, ei muotojen tunnistaminen tuottanut kahden luokitteluluokan suhteen ongelmia.

Kuva 34. Muotojen tunnistusta varten luotiin oma datasetti

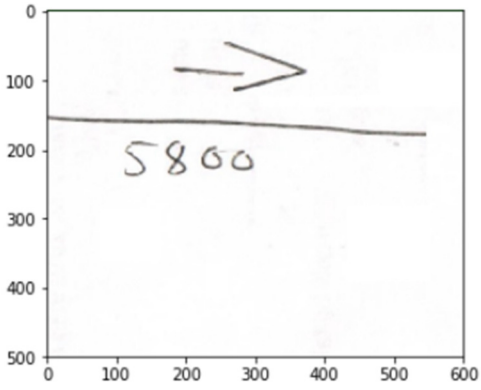
Alkuperäiset mittakuvat sisälsivät myös muita mittakuviin liittyviä merkintöjä ja kuviin kuulumattomia roskapisteitä. Näille muodoille olisi voitu luoda myös oma luokittelunsa, jolla ne olisi voitu suodattaa pois. Opinnäytetyössä päätettiin kuitenkin keskittyä vain tärkeimpien mittatietojen analysoinnin kehittämiseen ja kuviin liittyvien muiden merkintöjen ja roskapisteiden luokittelu rajattiin pois.

5.6.1 Mittatietojen tunnistus

Osakokonaisuuksien L- ja U-mittakuvien muotojen tunnistamisen jälkeen kuvista oli tunnistettava mittatiedot. Mittatietojen tunnistusta varten kuvat jaettiin osiin käyttäen Skimage -kirjaston kuvien jakamiseen tarkoitettua resize -funktiota. L-muotoiset kuvat jaettiin kahteen osaan ja U-muotoiset kuvat jaettiin kolmeen osaan. L-mallisista kuvista erottuivat kuvien 35 ja 36 mukaiset erisuuntaisia tietoja sisältävät osat. Mittatietojen tunnistusta varten pystysuuntaisia tietoja sisältävä kuva käännettiin vaaka-asentoon Skimage -kirjaston kuvan kääntämiseen tarkoitetulla rotate -funktiolla.



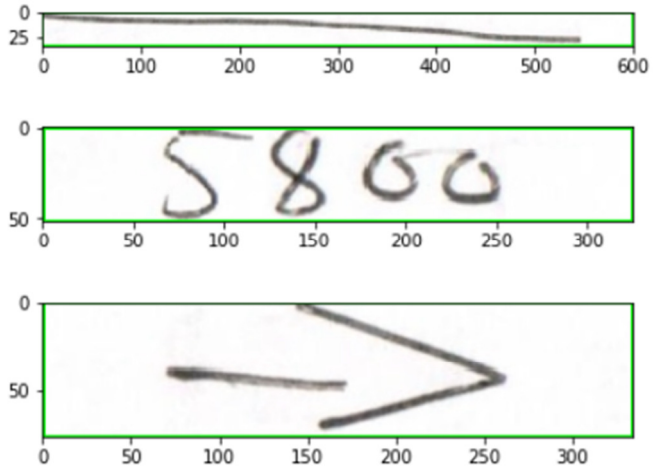
Kuva 35. Kuvasta jaettu pystysuuntainen mittatieto



Kuva 36. Kuvasta jaettu vaakasuuntainen mittatieto

Vaakasuuntaan käännetyistä kuvista rajattiin ja erotettiin yksittäiset muodot kohdan 5.4.4 tapaan, käyttäen OpenCV -kirjaston dilate ja findContours -funktiota. Kuvista saatiin kuvien 37 kaltainen laajennettu alue, josta jälleen irrotettiin bounding.Rect -funktiota käyttäen kuvien osat erilleen kuvan 38 mukaisesti. Kuvissa olevien mittatietojen paikka ei kuitenkaan ollut kaikissa käsitellyissä kuvissa samalla kohtaa, vaan tietojen paikka vaihteli mittapiirroksen tekijästä riippuen. Tällä asialla ei kuitenkaan enää ollut väliä, koska viimeinen vaihe oli tulkita koneoppimista käyttäen irrotettujen muotojen sisältö.

Kuva 37. L-mittakuva, joka on jaettu ja käsitelty



Kuva 38. Kuvan 37 muodot irrotettuina

Löydetyille muodoille oli vaiheessa 5.5.4 luotu omat koneoppimismallit, joilla ennustettiin muodon luokka. Muotoja ennustettiin sivusuuntaisista nuolista, muodon osista ja numerosarjoista. Nuolista saatiin lopputuloksen kannalta tärkeä tieto nuolen suunnasta. Muodon osista, kuten viivoista ja mahdollisesti viereisten numerosarjojen jäänteistä saatiin luokka, jonka avulla pääteltiin niiden olevan lopputuloksen kannalta merkityksettömiä muotoja.

Numerosarjojen suhteen jouduttiin vielä kerran toistamaan kohdan 5.4.4 tapa käyttää OpenCV -kirjaston dilate ja findContours -funktioita erottelemaan numeroiden muodot toisistaan. Numeroista oli luotu kohdassa 5.5.4 koneoppimismalli, jossa olivat numerot 0-9. Koneoppimismallille ei ollut opetettu neljää numeroa sisältävän sarjan tulkintaa, vaan jokainen numero jouduttiin tunnistamaan erikseen.

Lopputulokseksi saatiin kuusi tai yhdeksän muodon kuvausta, riippuen mittakuvan L- tai U-muodoista. Kuvassa 39 esiintyvät tulokset saatiin kuvan 32 mukaisesta L-mallisesta mittakuvasta. Arvot olivat numeerisessa muodossa ja vaikka niiden järjestys ei ollut yhtenevä, niin tulosten arvot voitiin tulkita ja järjestää yksinkertaisella Pythonin koodilla. Tulosten ennustus ei kuitenkaan läheskään aina mennyt oikein, sillä ennustuksen osumatarkkuus oli kaikilla ennustettavilla osilla alle sata prosenttia.

```
Sivu_A Muoto1: 1
Sivu_A Muoto2: 2800
Sivu_A Muoto3: 0

Sivu_B Muoto1: 0
Sivu_B Muoto2: 5800
Sivu_B Muoto3: 2
```

Kuva 39. Yhden analyysin arvot, josta ilmenevät mitat, nuolien ja viivojen luokat

6 Tulosten arviointi

Opinnäytetyön tuloksena saavutettiin ennalta määritellyt tavoitteet, mutta myös asioita joita ei tarkemmin ollut kirjattu suoriin tavoitteisiin. Tuloksena saatiin toimivan ohjelman runko, jolla voitiin analysoida kahdella eri muodolla toteutettuja mittakuvia. Lopputuloksena saatiin myös teknologian osia, joita voidaan jatkossa käyttää muissa vastaavissa käyttökohteissa. Suurimpana hyötynä oli kuitenkin perehtyminen syvällisemmin koneoppimisen kuvantunnistukseen ja sen tuomiin mahdollisuuksiin.

Tutkimuksessa käytettyjä menetelmiä voitiin pitää luotettavina, sillä ne perustuivat kirjallisuudesta ja dokumentaatioista saatujen menetelmien soveltamiseen. Luodun ohjelman koneoppimisen toiminnan suhteen saatiin myös ennustettavuudelle tarkkuudet, jotka perustuivat visuaalisista kuvaajista ja testauksessa käytetyistä datoista saatuihin tuloksiin. Näin ollen myös luodun analysointiohjelman toimintaa voitiin pitää luotettavana, niissä rajoissa joissa menetelminä käytetyt koneoppiminen ja konenäkö sen määrittelivät.

6.1 Toimivuus ja hyödyt

Opinnäytetyössä luotu ohjelma näytti koneoppimisen tehon ja hyödyt käsinpiirrettyjen mittakuvien analysoinnissa. Analysoinnin osoitettiin olevan teknisesti mahdollista ja nopeuttavan analysointiprosessia. Analysoitavana olleiden mittakuvien analysointi saatiin onnistumaan riittävällä tarkkuudella, jotta sitä voidaan kehittää ja käyttää kuvien analysoinnin tukena myös jatkossa.

Luotu ohjelma perustui koneoppimiseen eli ennustamiseen, jossa saadut tulokset eivät kuitenkaan aina ole täysin oikein. Ohjelman tuottamien tulosten lopullinen arviointi tuli suorittaa ihmisen toimesta, sillä ohjelma ei yltänyt ihmisen kykyyn kyseisten kuvien analysoinnissa. Tämä ei kuitenkaan ollut ongelma sillä suurin esityö saatiin suoritettua jo ohjelman toimesta.

6.2 Kannattavuuden arviointi

Opinnäytetyön yksi tavoite oli arvioida luodun mittakuvien tunnistusohjelman taloudellista kannattavuutta, koska kyseessä oli täysin kaupalliseen käyttöön kohdistettu ratkaisu. Ohjelman kehitys ja toteutus suoritettiin täysin opinnäytetyöhön varattujen 30 opintopisteen, eli tunneiksi muutettuna 810 tunnin, aikana. Pääpainon prosessissa voitiin kuitenkin katsoa olleen nimenomaan kouluttautumisessa, ei niinkään suoraan rahaksi laskettavana aikana. Lisäkoulutuksesta saatavan hyödyn katsottiin realisoituvan takaisin niin pitkällä

aikavälillä, ettei ohjelman kehittämisen kulunutta aikaa katsottu järkeväksi arvioida suoraan rahallisesti.

Ohjelman ja sen osien toiminnan tuoma lisähyöty oli kuitenkin arvioitavissa helpommin. Yhden mittakuvan analysointiin kuluneen ajan arvioitiin olevan noin kymmenesosa ihmisen suorittamasta analysoinnin ajasta. Ihmisen tehtäväksi katsottiin kuitenkin aina analysoidun mittakuvan arvojen oikeellisuuden tarkastus, joten karkeasti arvioituna voitiin laskea analysointiin kuluneen kokonaisajan olevan noin viidesosa ihmisen suorittamasta ajasta. Mittakuvien analysoitavien määrien ei kuitenkaan edes vuositasolla oletettu olevan niin suuria, että ajansäästöä voitaisiin pitää merkittävänä.

Koneoppimisessa datan keräämisen ja luokittelun tiedettiin olevan yksi eniten aikaa vievistä toimenpiteistä. Opinnäytetyössä luodun analysointiohjelman lisäosan, datanluontityökalun, arvioitiin kuitenkin olevan todella hyödyllinen tulevien projektien parissa. Koneoppimisen kuvantunnistukseen tarvittavien, käsinpiirrettyjen, opetusmuotojen leikkaamisen ja luokittelun arvioitiin olevan datanluontityökalulla todella tehokasta. Datanluontityökalun tuoman lisähyödyn katsottiin realisoituvan tulevien projektien parissa.

6.3 Jatkokehitys

Opinnäytetyössä luotu ohjelma ei sisältänyt erillistä käyttöliittymää, vaan ohjelmaa ajettiin suoraan koodista. Ensisijaisena jatkokehitystoimenpiteenä ohjelmalle katsottiin tarvittavan käyttöliittymä, joka mahdollistaa ohjelman helpon käytön. Käyttöliittymällä ja sen mahdollisesti tuottamalla visuaalisella ulosannilla oletettiin voitavan tehostaa myös ihmisen suorittamaa tulosten tarkastusta.

Luotu mittakuvien analysointiohjelma sisälsi tunnistuksen vain rajattuihin muotoihin ja merkkeihin. Tuntemattomien muotojen ja merkkien suhteen aiheutuvia ongelmia ei testattu, mutta luonnollisesti ohjelma ei niitä olisi osannut tulkita. Ohjelman tehokkaan hyödynnettävyyden takia se on opetettava tunnistamaan useampia muotoja ja rakenteita, jolloin suurin osa tuntemattomien muotojen ja merkkien mahdollisesti aiheuttamista ongelmista saadaan ratkaistua. Ohjelmaa ei myöskään ohjelmoitu käsittelemään kuin yksi mittakokonaisuus kuvasta, jonka jälkeen ohjelma oli käynnistettävä uudelleen. Useiden mittakokonaisuuksien käsittelyn ei arvioitu kuitenkaan olevan ison työn takana, sillä loput mittakokonaisuudet käsitellään samalla koodilla.

Tulevaisuuden suhteen arvioitiin, että on pohtimisen arvoista, millä aikataululla opinnäytetyössä käsitellyt käsinpiirretyt kuvat jäävät historiaan ja siirrytään kaikkien henkilöiden osalta syöttämään tiedot suoraan digitaalisiin järjestelmiin. Mahdollista voi olla myös opinnäytetyössä kehitetyn ohjelman hyödyntäminen integroituna muihin käytössä olevien oh-

jelmiin. Ohjelman jatkokehitystä arvioitaessa on otettava huomioon muiden digitaalisten ratkaisujen kehitys ja suhteutettava jatkokehitys niihin.

Opinnäytetyössä sivutuotteena luotu erillisen datanluontityökalun arvioitiin olevan myös jatkokehityksen arvoinen. Datanluontityökalun arvioitiin soveltuvan kaikkien vastaavien datasettien koostamiseen, joten sen käyttömahdollisuuksia pidettiin erittäin potentiaalisina. Ohjelman helpompaa käyttöä varten katsottiin sen tarvitsevan käyttöliittymän ohjelman arvojen muokkaamiseen. Ohjelman kehitysideaksi mietittiin myös automaattista opetuskuvien kuvakokojen pienentäjää datasettejä varten. Koneoppimista käyttäen ohjelma voisi mahdollisesti arvioida kuvalle pienimmän toimivan kuvakoon.

7 Yhteenveto

Opinnäytetyön tekeminen oli teknologisena ja luovana haasteena mielenkiintoinen yhtälö, sillä koneoppiminen ja ohjelmointikielenä Python olivat tulleet tutuiksi vasta koulun kurssien kautta. Oman haasteensa työhön toi koneoppimisen nopean kehitystahdin takia saatavissa olleiden kirjojen ja oppaiden tietojen ajantasaisuuden varmistaminen. Lähtökohtana koneoppimisen kuvantunnistus oli kuitenkin todella mielenkiintoinen ja inspiroiva, joten aiheeseen syventyäkseen oli hyvä ymmärtää alan kehitystahdin nopeus.

Opinnäytetyössä käytetylle teknologialle olisi löytynyt myös muita vaihtoehtoisia toteutustapoja, mutta valittu teknologia palveli käyttötarkoituksessaan ja antoi monipuolisen näkökulman koneoppimisen eri osa-alueisiin. Koneoppiminen prosessina noudattelee pääpiirteissään samaa kaavaa, vaikka muu asiakokonaisuuteen liittyvä rakenne olisikin toteutettu hieman toisella tavalla. Suurimman eron toteutustapojen välillä luo lähinnä datan käsittely ja siihen liittyvät lisäsovellukset.

Kokonaisuutena opinnäytetyö onnistui prosessinsa ja oppimistavoitteensa suhteen hyvin. Opinnäytetyölle laaditun aikataulun suhteen ei myöskään ollut ongelmia, vaan opinnäytetyö eteni ennakkosuunnitelmien mukaan. Opinnäytetyön lopputuloksena valmistunut ohjelma ja siihen liittyneet apuohjelmat tulevat palvelemaan oman aikansa, mutta pitkäikäisin ja suurin hyöty saavutettiin opinnäytetyöprosessissa saatuna syvennettynä ymmärryksenä koneoppimisen kuvantunnistuksesta.

Lähteet

Bonaccorso, G. 2018. Mastering machine learning algorithms: Expert techniques to implement popular machine learning algorithms and fine-tune your models. UK, Birmingham: Packt Publishing Ltd.

Bowles, M. 2015. Machine Learning in Python: Essential techniques for predictive analysis. USA, Indianapolis: John Wiley & Sons Inc.

Blanchet, G., Charbit, M. 2014. Digital Signal and Image Processing using MATLAB®: 2nd Edition Revised and Updated Volume 1 - Fundamentals. UK, London: ISTE Ltd and USA: John Wiley & Sons, Inc.

Bronshtein, A. 2017. A Quick Introduction to K-Nearest Neighbors Algorithm. Viitattu 27.2.2020. Saatavissa https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

Brownlee, J. 2019. Difference Between Classification and Regression in Machine Learning. Viitattu 14.2.2020. Saatavissa https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/

Dadhich, A. 2018. Practical Computer Vision: Extract insightful information from images using TensorFlow, Keras and OpenCV. UK, Birmingham: Packt Publishing Ltd.

Garreta, R. & Moncecchi, G. 2013. Learning Scikit-learn : Machine Learning in Python. UK, Birmingham: Packt Publishing Ltd.

Gollapudi, S. 2016. Practical Machine Learning. UK, Birmingham: Packt Publishing Ltd.

Google. 2020. What is Colaboratory. Viitattu 5.10.2020. Saatavissa https://colab.research.google.com/notebooks/intro.ipynb

Hackling, G. 2014. Mastering Machine Learning with Scikit-learn. UK, Birmingham: Packt Publising Ltd.

Hamid, N. & Sjarif, N. 2019. Handwritten Recognition Using SVM, KNN and Neural Network. Viitattu 28.12.2019. Malysia. Universiti Teknologi Malysia. Saatavissa https://arxiv.org/ftp/arxiv/papers/1702/1702.00723.pdf

Howse, J., Joshi, P. & Beyeler, M. 2016. OpenCV: Computer Vision Projects with Python. UK, Birmingham: Packt Publishing Ltd.

Hunter, J., Dale, D., Firing, E., Droettboom, M. & Matplotlib development team. 2020. Matplotlib: Introductory. Viitattu 1.1.2020. Saatavissa https://matplotlib.org/tutorials/index.html#introductory

Ingargiola, A. 2015. Viitattu 12.1.2019. Saatavissa https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#:~:text=1.1.-,Notebook%20document,%2C%20links%2C%20etc%E2%80%A6)

Joshi, P., Escrivá, D. & Godoy, V. 2016. OpenCV By Example. UK, Birmingham: Packt Publishing Ltd.

Lukka, K. 2020. Konstruktiivinen tutkimusote. Viitattu 14.11.2020. Saatavissa https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote

Microsoft. 2019. Overview of Azure Notebooks Preview. Viitattu 12.1.2020. Saatavissa https://docs.microsoft.com/en-us/azure/notebooks/azure-notebooks-overview

Navlani, A. 2018. KNN Classification using Scikit-learn. Viitattu 5.1.2020. Saatavissa https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Nixon, M. & Aquado, A. 2012. Feature Extraction and Image Processing for Computer Vision: Third Edition. UK, Oxford: Elsevier Ltd.

OpenCV team. 2019. Eroding and Dilating. Viitattu 29.12.2020. Saatavissa https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html

OpenCV team. 2020a. Image Thresholding. Viitattu 2.1.2020. Saatavissa https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

OpenCV team. 2020b. Morphological Transformations. Viitattu 2.3.2020. Saatavissa https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html

OpenCV team. 2020c. Contours: Getting Started. Viitattu 14.1.2020. Saatavissa https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

OpenCV team. 2020d. Contours Hierarchy. Viitattu 14.1.2020. Saatavissa https://docs.opencv.org/master/d9/d8b/tutorial_py_contours_hierarchy.html

OpenCV team. 2020e. OpenCV: Intel. Viitattu 1.10. 2020. Saatavissa https://opencv.org/intel

Pandas developement team. 2020a. Pandas: pandas.Series.str.split. Viitattu 1.1.2020. Saatavissa https://pandas.pydata.org/pandasdocs/stable/reference/api/pandas.Series.str.split.html

Pandas developement team. 2020b. About pandas. Viitattu. 1.1.2020. Saatavissa https://pandas.pydata.org/about/index.html

Peltonen, J. 2015. Pohjapiirustus As Oy Jyväskylän Åströmin Saarni. Jyväskylä: Lemminkäinen Talo Oy

Scikit-image development team. 2020a. Image processing in Python. Viitattu 2.10.2020. Saatavissa https://scikit-image.org

Scikit-learn developers. 2019a. User Guide. Viitattu 26.12.2019. Saatavissa https://scikit-learn.org/stable/user_guide.html

Scikit-learn developers. 2019b. Choosing the right estimator. Viitattu 26.12.2019. Saatavissa https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-learn developers. 2020a. Underfitting vs. Overfitting. Viitattu 5.1.2020. Saatavissa https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Scikit-learn developers. 2020b. sklearn.svm.SVC. Viitattu 15.2.2020. Saatavissa https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Scikit-learn developers. 2020c. Nearest Neighbors. Viitattu 17.2.2020. Saatavissa https://scikit-learn.org/stable/modules/neighbors.html

Scikit-learn developers. 2020d. Ensemble methods. Viitattu 17.2.2020. Saatavissa https://scikit-learn.org/stable/modules/ensemble.html

Scikit-learn developers. 2020e. sklearn.ensemble.RandomForestClassifier. Viitattu 22.2.2020. Saatavissa https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Scikit-learn developers. 2020f. sklearn.model_selection.train_test_split. Viitattu 23.3.2020. Saatavissa https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Scikit-learn developers. 2020g. About us. Viitattu 13.10.2020. Saatavissa https://scikit-learn.org/stable/about.html

SciPy community. 2020a. Introduction. Viitattu 2.10.2020. Saatavissa https://docs.scipy.org/doc/scipy/reference/tutorial/general.html

SciPy community. 2020b. Numpy: Quickstart tutorial. Viitattu 1.1.2020. Saatavissa https://numpy.org/devdocs/user/quickstart.html

Vlerick, A. 2020. A visual introduction to Binary Image Processing. Viitattu 24.2.2020. Saatavissa https://towardsdatascience.com/a-visual-introduction-to-binary-image-processing-part-1-d2fba9f102a4