Harri Huikuri

Koneoppimismallien käyttö luokittelutehtävässä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

12.10.2022

Tiivistelmä

Tekijä:	Harri Huikuri
Otsikko:	Koneoppimismallien käyttö luokittelutehtävässä
Sivumäärä:	54 sivua + 1 liitettä
Aika:	12.10.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ohjaaja:	Osaamispäällikkö Janne Salonen

Tekoäly on osa ihmisten jokapäiväistä arkea ilman, että siihen kiinnitetään mitään huomiota. Se toimii taustalla erilaisissa sovelluksissa ja laitteissa tarjoten parempia käyttökokemuksia kuluttajille, mutta mahdollistaa tuottavuuden kasvun liiketoiminnassa sekä synnyttää kokonaan uudenlaista liiketoimintaa.

Digitalisaation myötä lähes kaikesta mitä teemme, on tullut tallennettavaa ja mitattavaa dataa. Valtavien datamassojen louhinta ja tiedon tuottaminen niistä sopii mitä parhaiten tekoälyalgoritmien tehtäväksi lukemattomissa käyttötarkoituksissa, joissa vain mielikuvitus on rajana.

Tässä opinnäytetyössä perehdytään koneoppimiseen, joka on yksi tekoälyn osa-alueista. Siinä käytetään strukturoitua tai puolistrukturoitua dataa etukäteen määritellyn tehtävän toteuttamiseen. Työssä esimerkkinä käytetään luokittelutehtävää, jossa avoimesti verkosta saatavilla olevasta sensoridatasta pyritään päättelemään tarve laukaista palohälytys. Tehtävä toteutetaan Python ohjelmointikielellä.

Työn teoreettisessa osuudessa perehdytään koneoppimisen yleisiin periaatteisiin ja osa-alueisiin. Ohjatun-, ohjaamattoman- ja vahvistusoppimisen lisäksi käydään läpi neuroverkkoja ja syväoppimista. Työssä käsitellään koneoppimismallien suorituskykyyn vaikuttavia tekijöitä ja sen arvioinnissa käytettäviä mittareita.

Useimmat algoritmit sopivan nimenomaisesti tiettyä tehtävää varten, esimerkiksi luokitteluun, regressio-ongelmien ratkaisemiseen tai ryhmittelyyn. Tehtävään parhaiten soveltuvan algoritmin löytäminen perustuu usein kokemukseen ja kokeiluun. Tässä työssä esitellään lyhyesti joitakin yleisemmistä luokittelutehtävään soveltuvista algoritmeista, kokeillaan niiden käyttöä käytännössä ja arvioidaan niiden soveltuvuutta esimerkkinä käytettyyn luokittelutehtävään.

Käytännön esimerkki etenee tavanomaista koneoppimisprosessin kulkua noudattaen. Ensimmäiseksi määritellään ratkaistava ongelma ja tavoitteet. Käytettävä data analysoidaan ja valmistellaan algoritmeille sopivaan muotoon, jonka jälkeen rakennetaan koneoppimismallit ja lopuksi arvioidaan niiden suoriutumista.

Avainsanat: tekoäly, koneoppiminen, Python

Abstract

Author:	Harri Huikuri
Title:	Machine Learning in classification
Number of Pages:	54 pages + 1 appendices
Date:	12 October 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Supervisors:	Janne Salonen, Principal Lecturer

Artificial Intelligence is nowadays an unnoticeable part of everyday life. It runs in the background in various applications and devices offering a better user experience, but also enables productivity growth in business and creates a completely new kind of business.

Almost everything we do has become recordable and measurable data by digitalization. Mining huge masses of data and generating information from them is best suited to the task of artificial intelligence algorithms in countless uses where only imagination is the limit.

This thesis introduces machine learning, which is one of the areas of artificial intelligence. It uses structured or semi-structured data to implement a specific pre-defined task. A practical example introduces a simple classification task, where the aim is to deduce the need to trigger a fire alarm using a machine learning model implemented using Python programming language. The data used for the task is freely available on the web.

The principles and areas of machine learning are introduced in the theoretical part of the thesis. In addition to supervised, unsupervised, and reinforcement learning, neural networks and deep learning are covered. The factors influencing the performance of machine learning models and the metrics used in their evaluation are discussed as well.

Most algorithms are specifically suited for a specific task, for example, classification, or solving regression or clustering problems. Finding the best algorithm for the task is often based on experience and experimentation. This thesis introduces six of the common algorithms suitable for the classification task. Those algorithms are used in the practical part of the study.

The practical example goes ahead following the usual machine learning process flow. First, the problem is to be solved and the goals are defined. The data to be used in the task is analyzed using explorative data analysis and then preprocessed in a format suitable for the algorithms. Finally, the machine learning models are built, and their performance is evaluated.

Keywords: Artificial Intelligence, Machine learning, Python

Sisällys

1 Johdanto

Digitalisaation myötä datan määrä on räjähdysmäisesti kasvanut. Lähes kaikesta mitä teemme, on tullut mitattavaa ja digitaalisessa muodossa tallennettavaa dataa. Luodun, kulutetun ja tallennetun datan määrän vuonna 2022 ennustetaan olevan 97 zettatavua (1 zettatavu = 10^{21} tavua eli biljoona gigatavua) ja kaksinkertaistuvan vuonna 2025. Viisi vuotta sitten vuonna 2017 määrän arvioidaan olleen 26 zettatavua. (1). Tekoäly ja entistä pienempään tilaan mahtuvat ja tehokkaammat prosessorit ovat mahdollistaneet näiden valtavien datamassojen käytön kokonaan uudenlaisten teknologisten ratkaisujen ja liiketoimintamallien kehittämisessä.

Tekoäly on tullut osaksi ihmisten jokapäiväistä arkea ilman, että siihen kiinnitetään erityistä huomiota. Esimerkiksi älypuhelimet, joissa kasvontunnistus, puheohjaus tai kameralla otettujen kuvien parantaminen hyödyntävät tekoälyä. Verkkokauppojen ja -palvelujen tuotesuositukset tai sosiaalisissa medioissa käyttäjän syötevirtaa ohjaavat algoritmit sekä hakukoneet perustuvat tekoälyratkaisuihin. Monissa uudemmissa autoissa tekoälyä käytetään esimerkiksi vaaratilanteiden, liikennemerkkien tai kuljettajan vireystilan havainnoinnissa, puhumattakaan itseohjautuvissa ajoneuvoissa. (2) Espoolaiset ovat voineet kuluvana vuonna havaita liikenteessä Alepan robotteja, jotka toimittavat ostokset tekoälyn ohjaamana perille asiakkaalle (3).

Tässä opinnäytetyössä perehdytään yleisellä tasolla koneoppimisen perusteisiin, sen keskeisiin käsitteisiin ja menetelmiin. Tavoitteena on laajentaa teoreettista tietopohjaa kokeilemalla joidenkin teoriaosuudessa käsiteltävien luokittelutehtävään soveltuvien koneoppimisalgoritmien käyttöä käytännössä. Koska työllä ei ole toimeksiantajaa, mallien opetuksessa käytetään verkosta avoimesti saatavilla olevaa data-aineistoa. Tarkoituksena ei ole asettaa eri koneoppimismalleja paremmuusjärjestykseen, mutta niiden suoriutumista luokitteluongelman ratkaisemisessa arvioidaan erikseen teoriaosuudessa esitettyjä soveltuvia mittareita hyödyntäen. Käytännön osuus toteutetaan Python ohjelmointikielellä.

2 Koneoppimisen perusteet

2.1 Tekoäly ja koneoppiminen

Koneoppiminen on tällä hetkellä yksi suosituimmista datatieteen ja älykkäiden järjestelmien teknologioista, jota käytetään laajalti monilla eri aloilla. Aihe ei ole kuitenkaan ole uusi. Terminä koneoppiminen on peräisin jo vuodelta 1959, jolloin Arthur Samuel määritteli sen antavan tietokoneille kyvyn oppia asioita ilman nimenomaista ohjelmointia. (4). Koneoppimisessa yhdistyvät laskenta ja algoritmit sekä tilastollinen ajattelu, jotka yhdessä johtavat datan analysointiin, matemaattisten mallien rakentamiseen ja päättelyyn automaattisesti annetusta datasta tehtyjen havaintojen perusteella (5 ss. 330-332).

Termejä tekoäly ja koneoppiminen käytetään välillä toistensa synonyymeinä, vaikka ne eivät ole täsmällisesti sama asia. Tekoäly on laajempi sateenvarjomainen käsite, jolla tarkoitetaan yleisesti tietojärjestelmän kykyä jäljitellä ihmisaivojen toimintaa (6). Koneoppiminen on tekoälyn osa-alue, joka käyttää strukturoitua tai puolistrukturoitua dataa etukäteen tarkasti määritellyn tehtävän toteuttamiseen (7). Syväoppiminen on puolestaan koneoppimisen osa-alue, jossa neuroverkot jäljittelevät ihmisen aivojen rakennetta ja toimintaa kyeten käsittelemään myös strukturoimatonta dataa, kuten puhetta, kuvia ja videotallenteita (6). Termien välistä yhteyttä havainnollistetaan kuvassa 1.

Kuva 1: Koneoppiminen on tekoälyn osa-alue (8)

Tekoäly eroaa koneoppimisesta siinä, että sen algoritmit kykenevät käsittelemään myös etukäteen määrittelemättömiä tilanteita (7). Tällaisesta hyvänä esimerkkinä toimii aiemmin mainittu Alepa-robotti, joka kuljettaessaan ostoksia perille joutuu selviytymään kohtaamistaan liikennetilanteista ja esteistä valitsemallaan reitillä. Muita esimerkkejä ovat kuvan- tai puheentunnistus.

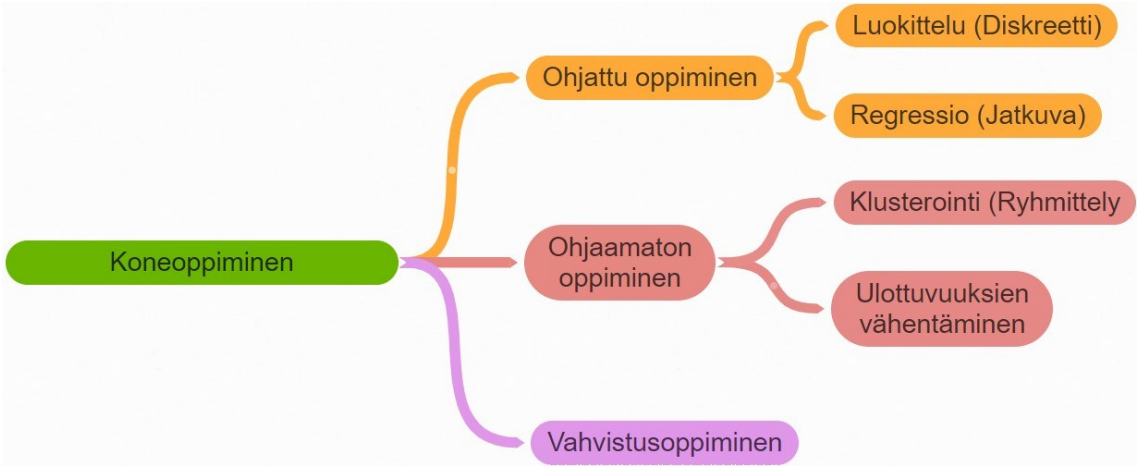
Tekoäly voidaan luokitella heikoksi tai vahvaksi. Tietokone käsittelee tietoa ja suorittaa laskentaa ilman, että sillä olisi kykyä ymmärtää käsittelemänsä datan sisältöä tai inhimillisiä merkityksiä. Koneella ei ole ihmisen kaltaista tietoisuutta. (9 s. 45). Kone pystyy toimimaan älykkäästi vain nimenomaista tehtävää suorittaessaan tai tietyllä aihealueella. Tämän päivän tekoäly on kuvatun kaltaista heikkoa tekoälyä. (10). Vahvalla tekoälyllä tarkoitetaan laaja-alaisia keskenään kommunikoivia järjestelmiä, joilla olisi yhteinen ihmisen kaltainen sähköinen tietoisuus olemassaolosta, ja jotka ylittävät ihmisen älylliset ja toiminnalliset kyvyt (9 s. 45). Vahva tekoäly kykenee itsenäiseen ihmisen kaltaiseen ajatteluun ja käsittelemään minkä tahansa tehtävän tai ongelman miltä tahansa aihealueelta (10). Toistaiseksi vahvaa tekoälyä ei ole kyetty luomaan.

Koneoppimisessa aiemmin kerätyn datan ja mahdollisen käyttäjän toiminnan perusteella pyritään siihen, että kone oppii toistuvista tilanteista entistä paremmin ilman, että sitä opetetaan erikseen (9 s. 316). Tavoitteena on löytää kontekstiin liittyvästä datasta ne piirteet, joiden perusteella erilaisia matemaattisia malleja ja tilastollisia menetelmiä hyödyntäen voidaan määrittää todennäköisin tulos myös uuden aiemmin näkemättömän datan perusteella (11). Esimerkiksi roskapostisuodatin pyrkii tunnistamaan ja oppimaan sähköposteista ne piirteet, joiden perusteella kaikki myöhemmin saapuvat viestit pystytään luokittelemaan joko roskapostiksi tai ei-roskapostiksi.

Koneoppimismalli on koneoppimisalgoritmin tuottama tulos, joka koostuu mallissa käytetystä datasta ja ennustamisessa käytetystä algoritmista. Koneoppimisalgoritmia voidaan ajatella samanlaisena algoritmina kuin muita ohjelmoinnissa käytettäviä algoritmeja, jotka voidaan kuvata matemaattisesti ja pseudokoodina. Algoritmi sovitetaan opetusdataan, se käsittelee datan ja oppii siitä.

Malli syntyy, kun algoritmia käytetään opetusdataan. Se sisältää algoritmin datasta tuottamat säännöt, vakiot ja rakenteet, joiden avulla uudesta datasta pystytään tuottamaan ennusteita. (12).

Koneoppiminen voidaan jakaa kolmeen pääluokkaan: Ohjattuun (supervised) ja ohjaamattomaan (unsupervised) oppimiseen sekä vahvistusoppimiseen (reinforcement learning) (13 s. 39). Joissakin lähteissä mainitaan myös puoliohjattu oppiminen, joka on osin ohjattua, osin ohjaamatonta oppimista. Se on käyttökelpoinen, kun ennustettavan muuttujan arvot ovat vain osin tiedossa (5 s. 333). Seuraavissa luvuissa perehdytään näihin tarkemmin.



Kuva 2: Koneoppimisen luokittelua

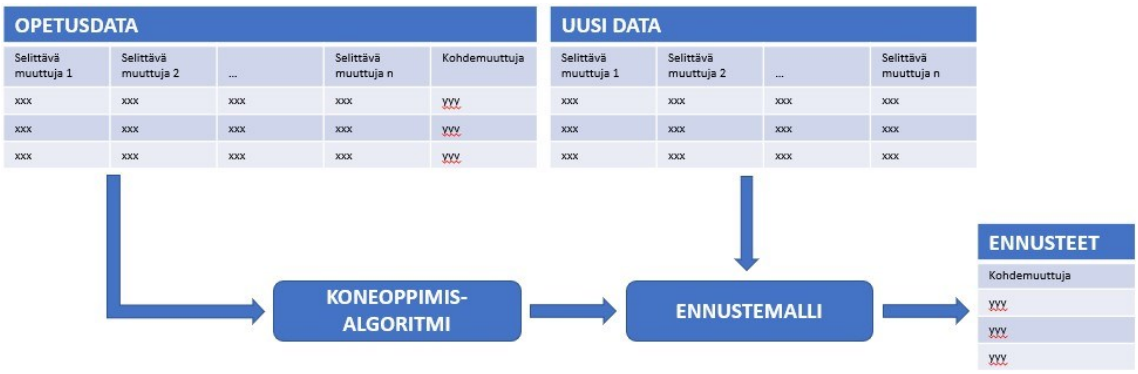
2.2 Ohjattu oppiminen

Ohjattu oppiminen on koneoppimisen perustyyppi, jossa oppimisalgoritmi opetetaan sekä selittävät muuttujat (features) että kohdemuuttujan (label) sisältävällä datalla (14 s. 39). Algoritmin tavoitteena on tuottaa funktio, joka selittävien muuttujien arvojen perusteella tuottaa mahdollisimman tarkasti kohdemuuttujan arvon (15 s. 100). Kohdemuuttujalla tarkoitetaan sitä lopputulosta, joka koneoppimismallin halutaan ennustavan, esimerkiksi kategorinen luokittelu onko sähköpostiviesti roskapostia vai ei. Selittävät muuttujat ovat kohdemuuttujan arvoon vaikuttavia ominaisuuksia, esimerkiksi sähköpostiviestin lähettäjään, lähetykseen ja sisältöön liittyviä piirteitä, joiden perusteella viestit voidaan luokitella.

Ohjatussa oppimisessa käytetään opetusdatana havaintoaineistoa, jossa kohdemuuttujan arvot eli lopputulos on valmiiksi annettu. Oppimisalgoritmi etsii havaintoaineistosta muuttujien välisiä yhteyksiä ja syy-seuraussuhteita, joiden perusteella se pystyy rakentamaan yleistävän funktion kohdemuuttujan arvon ennustamiseksi. (14 ss. 39-40). Eri algoritmit on suunniteltu etsimään yhteyksiä ja tuottamaan tai suosimaan tietynlaisia funktioita. Yksi koneoppimisen haasteista onkin löytää tietylle aineistojoukolle parhaiten soveltuva algoritmi. Käytännössä se yleensä edellyttää eri algoritmien kokeilua ja tulosten vertailua. (15 s. 102).

Algoritmin opetuksen tuloksena syntyvää koneoppimismallia käytetään uuteen, ennen näkemättömään dataan. Malli tuottaa datassa olevan havaintoaineiston perusteella ennusteet uuden aineiston kohdemuuttujien arvoista käyttäen opetuksessa rakennettua funktiota. Samalla algoritmi voi oppia lisää uudesta aineistosta ja kykenee näin jatkuvasti säätämään mallia paremmaksi itse itseään korjaten. (14 ss. 39-40). Koneoppimismallit voivat tuottaa tarkkoja tuloksia, mutta tarkkuuden riittävyyttä on tarkasteltava aina tapauskohtaisesti. (16 ss. 110-111). Mallin suorituskykyyn vaikuttavat useat eri tekijät, joita käsitellään myöhemmin suorituskykyä käsittelevässä luvussa 2.6.

Ohjatun oppimisen prosessin etenemistä on havainnollistettu kuvassa 3.



Kuva 3: Ohjatun oppimisen prosessin eteneminen (17)

Ohjattua oppimista käytetään sekä luokittelussa että regressio-ongelmien ratkaisemissa. Luokittelussa koneoppimismalli ennustaa havaintoaineiston

perusteella, onko kohdemuuttuja etukäteen määritellyssä ryhmässä vai ei. Kohdemuuttuja saa aina diskreetin eli rajalliseen joukkoon kuuluvan arvon. Yksinkertaisimmillaan luokittelun tulos voi olla binäärinen, esimerkiksi kyllä/ei. Jos kohdemuuttuja voi saada useampia arvoja, kyse on moniluokka-luokittelusta. (14 s. 38). Joskus tulos voi kuulua samanaikaisesti useampaan luokkaan. Luokittelua voidaan käyttää myös tunnistamaan datasta normaalista poikkeavat tapaukset (16 s. 115). Joitakin yleisesti luokittelussa käytettäviä algoritmeja käydään lyhyesti läpi luvussa 3.

Regressiomalleja käytetään, kun ennustettava kohdemuuttujan arvo on jatkuva lukuarvo (14 s. 38). Mallit perustuvat tilastollisiin menetelmiin, joilla etsitään riippuvan muuttujan (kohde) ja yhden tai useamman itsenäisen numeerisen muuttujan välisiä korrelaatioita. Regressiomalleja käytetään pääasiassa ennustamiseen, aikasarjojen mallintamiseen sekä muuttujien välisten syy-seuraussuhteiden määrittämiseen. (18). Koordinaatistossa selittävät muuttujat sijoitetaan akseleille, joita on yhtä monta kuin selittäviä muuttujia. Tähän pistejoukkoon sovitettua kuvaajaa hyödyntäen voidaan ennustaa arvo kohdemuuttujalle. (16 ss. 113-114). Kuvaaja ja regressioalgoritmit ovat joko lineaarisia tai ei-lineaarisia (18).

Luokittelumallien tyypillisiä käyttötapauksia ovat esimerkiksi asiakassuhteen jatkuvuuden arviointi, lääketieteelliset diagnoosit, huijauksien tai virheiden tunnistaminen tai suosittelu. Regressiomalleja käytetään esimerkiksi hinnanmäärittelyyn, kuluttajakäyttäytymisen tai talouden ennustamiseen. (16 ss. 110-111).

2.3 Ohjaamaton oppiminen

Ohjaamattomassa oppimisessa algoritmien tyypillisenä tehtävänä on löytää datasta samankaltaisia ominaisuuksia ja muodostaa oleellisten piirteiden perusteella samankaltaisia ryhmiä ilman, että niitä olisi määritelty etukäteen (14 ss. 43-44). Tätä kutsutaan klusteroinniksi. Erona ohjattuun oppimiseen, yksittäisen piirteen ennustamisen sijasta ohjaamattoman oppimisen fokus on aineiston rakenteen tunnistamisessa ja kuvaamisessa (17). Koneoppimismalli ei tiedä mitä

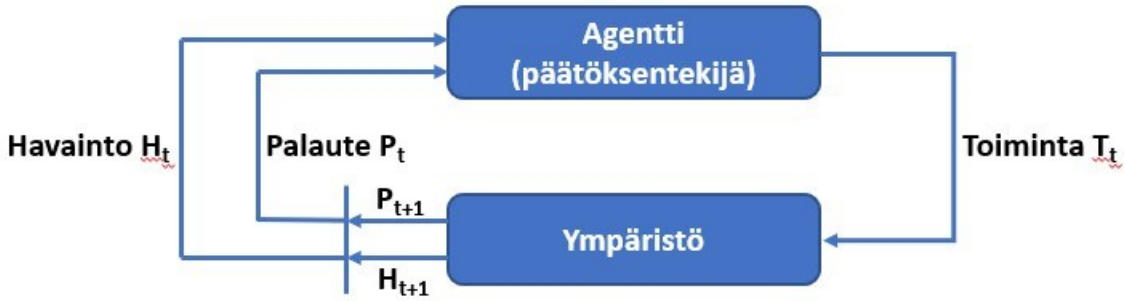
ryhmät edustavat, mutta se kykenee oppimansa perusteella ryhmittelemään uuden datan näihin ryhmiin. (14 ss. 43-44).

Ulottuvuuksien kirous (The curse of dimensionality) viittaa Richard E. Bellmannin kirjassaan Dynamic Programming (1957) käyttämään termiin, joka viittaa monimuuttuja-analyysiin liittyviin ongelmiin ulottuvuuksien eli ominaisuuksien määrän kasvaessa (19). Nykyisin termillä voidaan viitata dataan, jossa on valtava määrä ulottuvuuksia. Ulottuvuuksien kirous aiheuttaa potentiaalisia ongelmia koneoppimismalleille, esimerkiksi heikentämällä luokittelumallin tarkkuutta mallin ylioppimiseen seurauksena tai aiheuttaen klusterointialgoritmeille vaikeuksia löytää mielekkäitä ryhmiä. (20). Muita ongelmia ovat suuren ominaisuusmäärän algoritmien toimintaa hidastava vaikutus, josta voi seurata myös ylimääräisiä kustannuksia. Näitä ongelmia voidaan ehkäistä vähentämällä havaintoaineistosta ulottuvuuksia siten, että jäljelle jäävät olennaiset tiedot edistävät ennustemallin tarkkuutta ja seuraavien algoritmien toimintaa. (16 s. 102).

Yksi dimensioiden vähentämisen sovelluskohde on datan esikäsittely ennen varsinaisen koneoppimisalgoritmin opettamista. Tehtävä voi olla myös dataa yksinkertaistamalla havainnollistaa ja tunnistaa taustalla piilevät selitettävän ilmiön kannalta olennaiset muuttujat. Tästä on hyötyä erityisesti datan visualisoinnissa, ymmärtämisessä ja tulkinnassa. Muita käyttökohteita ovat esimerkiksi tekstien luokittelu tai lääketieteellisten kuvien segmentointi (21).

2.4 Vahvistusoppiminen

Kolmas koneoppimisen muoto on vahvistusoppiminen, jossa algoritmi oppii ihmismäisesti yrityksen ja erehdyksen kautta. Vahvistusoppimisen keskeisiä elementtejä ovat agentti ja ympäristö. Koneoppimisessa agenttina toimii algoritmi, joka havainnoi ja tulkitsee ympäristöään ja päättää itsenäisesti, kuinka toimii tehtävän suorittamiseksi. Se oppii saamalla palkintoja toivotusta suorituksesta ja rangaistuksia väärästä ja mukauttaa näin toimintaansa. (16 s. 158). Tätä havainnollistetaan kuvassa 4.



Kuva 4: Vahvistusoppimisen kulku (22)

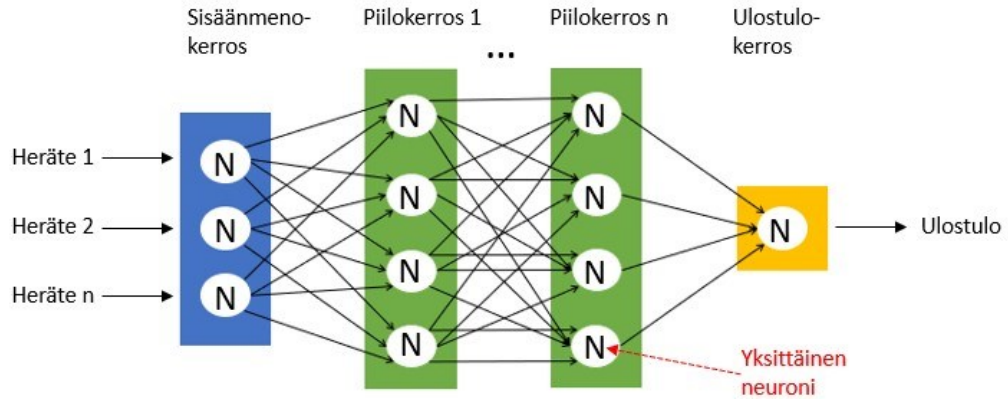
Vahvistusoppimisalgoritmit pyrkivät löytämään toimintamallin, joka maksimoi palkintojen kumulatiivisen kokonaismäärän pitkällä aikavälillä, vaikka se lyhyellä aikavälillä aiheuttaisi uhrauksia (22). Koska vahvistusoppiminen oppii lennosta, sen käyttämiseen ei välttämättä tarvita muiden oppimismuotojen tapaan historiadataa. Haasteena on, että useimmat algoritmeista toimivat parhaiten sääntöjen, olosuhteiden ja tavoitteiden pysyessä muuttumattomina, mutta myös oppimiskierros voi olla ajallisesti pitkä erityisesti liiketoimintasovelluksissa. (16 s. 159). Vahvistusoppimisen käyttökohteita ovat esimerkiksi itseohjautuvat ajoneuvot, teollisuusautomaatio ja robotiikka sekä pelit, joista yksi tunnetuimmista sovelluksista on Googlen AlphaGo Zero. Se oppi 40 päivän itsenäisen harjoittelun jälkeen Go-pelin tyhjästä pelaamalla itseään vastaan ja voitti lopulta pelin maailmanmestarit Lee Sedolin ja Ke Jien (19).

2.5 Neuroverkot ja syväoppiminen

Neuroverkot ovat syväoppimisen perusta. Ne muistuttavat rakenteeltaan aivoja muodostuen joukosta yksittäisiä toisiinsa kytkettyjä neuroneja, jotka saavat syötteenä yhden tai useamman numeerisen arvon ja tuottavat niistä yhden ulostuloarvon (15 s. 119). Yksittäisen neuronin toiminta jakautuu kahteen vaiheeseen: Ensin suoritetaan usean syötteen lineaarinen regressiofunktio (15 s. 120), joka käytännössä laskee sisääntulojen painokertoimella painotetun summan. Algoritmi muodostaa painokertoimet datan ominaisuuksien perusteella. Mitä suurempi painokerroin on, sitä enemmän ominaisuus vaikuttaa tutkittavaan ilmiöön. Jos painokerroin on nolla, ominaisuudella ei ole merkitystä ja yhteys

seuraavaan neuroniin häviää. Toisessa vaiheessa summa siirretään epälineaarista kuvausta soveltavalle aktivointifunktiolle, joka tuottaa neutronin ulostuloarvon, joka on 0 tai 1. (16 ss. 129, 131-133).

Vaikka yksittäinen neuroni kykenee vain hyvin yksinkertaiseen toimintaan, kytkemällä suuri joukko neuroneita yhteen neuroverkoksi voidaan toteuttaa erittäin monimutkaisia toimintoja. Kuvassa 5 havainnollistetaan syvää neuroverkkoa, joka muodostuu useista kerroksista. Termi syvä viittaa peräkkäisten piilokerrosten määrään, joissa voi olla eri määrä neutroneita. (15 s. 128).



Kuva 5: Neuroverkon rakenne (16 s. 133)

Neuroverkon opetus tapahtuu kuten ohjatussa oppimisessa vertaamalla opetusaineistossa annettua tunnettua kohdemuuttujan arvoa neuroverkon laskemaan arvoon. Näiden erotuksesta lasketaan kustannusfunktio, jonka arvo pienenee lasketun ja todellisen arvon lähetessä toisiaan. Tämä tapahtuu toistamalla laskentakierroksia eli epookkeja, joiden aikana painokertoimien arvoja muutetaan, kunnes riittävän tarkka lopputulos on saavutettu. Laskentakierroksia voi olla jopa tuhansia. (16 ss. 136-137).

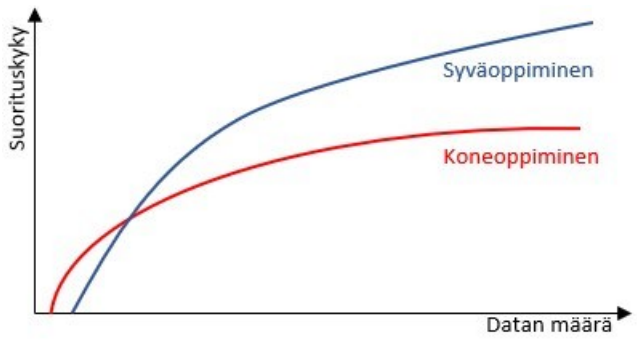
Neuroverkkoja hyödyntävän syväoppimisen käytännön sovelluskohteet eivät poikkea muista koneoppimisen kohteista. Sitä voidaan hyödyntää esimerkiksi asiakassuhteen jatkuvuuden ennustamisessa, eri ilmiöihin vaikuttavien tekijöiden etsimiseen tai luonnollisen kielen käsittelyyn liittyvät sovellukset kuten chatbotit, tekstien luokittelut. Neuroverkot toimivat hyvin myös kuvankäsittelyssä,

jossa niiden tapa toimia ihmisaivojen tapaan mahdollistaa kuvatunnistuksen myös osittaisten piirteiden perusteella sekä riippumatta hahmon sijainnista kuvassa. (16 ss. 138-149).

2.6 Mallien suorituskykyyn vaikuttavia tekijöitä

2.6.1 Datan määrä ja laatu

Koneoppimismallien suorituskykyä heikentävät tekijät löytyvät usein algoritmien opetuksessa käytetystä datasta ja sen laadusta. Huono data johtaa huonoon lopputulokseen. (16 s. 172). Myös datan ja siihen sisältyvien lähdemuuttujien määrät vaikuttavat suorituskykyyn. Lähtökohtaisesti mallin opetuksessa käytettävän datan määrän kasvaessa suorituskyky kasvaa. (23). Suorituskyvyn kasvu ei kuitenkaan ole lineaarista. Koneoppimismalleilla tietyn rajan jälkeen opetusdatan määrän muutos ei enää olennaisesti paranna suorituskykyä, kun syväoppimismenetelmät edelleen hyötyvät datan määrän kasvusta. Tätä havainnollistetaan kuvassa 6. (16 s. 62). Karkea nyrkkisääntö on, että koneoppimismallien opetuksessa esimerkkitapausten määrän tulisi olla vähintään kertaluokkaa suurempia kuin lähdemuuttujien määrät (23).



Kuva 6: Datan määrän vaikutus suorituskykyyn (16 s. 62)

Datan laatu ei ole yksiselitteinen käsite. Siihen sisältyvät kysymykset datan luotettavuudesta, sen ominaisuuksista, määrästä ja esitystavasta sekä aineiston jakauman vinoudesta. Datan luotettavuuteen liittyviä tekijöitä ovat esimerkiksi opetusdatan kohdemuuttujien arvojen oikeellisuus, selittävien muuttujien

kohina, puuttuvat arvot tai arvojen monikerrat ja arvojen tilastolliset poikkeamat. (23).

Ominaisuuksiin liittyviä tekijöitä ovat esimerkiksi niiden määrä ja hyvyys. Datan selittävien muuttujien joukossa voi olla lopputuloksen kannalta merkityksettömiä muuttujia. Mallin suorituskyvyn kannalta on olennaista valita mukaan ne muuttujat, joilla on suurin merkitys tulokseen. Selittävien muuttujien hyvyyttä voidaan mitata erilaisilla tunnusluvuilla, kuten gini-indeksillä, informaatiolisällä (info-gain), uskottavuusosamäärällä (likelihood-ratio) ja odds-suhteella. (16 s. 96). Myös lähdemuuttujien esitysmuoto voi vaikuttaa mallin suorituskykyyn. Algoritmit käsittelevät vain numeerista aineistoa, lisäksi muuttujien erilaiset skaalat vaikuttavat niiden saamiin painokertoimiin. (23). Näitä käsitellään tarkemmin datan esikäsittelyä koskevassa kappaleessa 4.4.

Yksi merkittävästi koneoppimismallin tuottamaan lopputulokseen vaikuttava tekijä on, kuinka opetuksessa käytettävä data on jakautunut. Mikäli aineisto on tilastollisesti vinoutunut, se heijastuu väistämättä mallin tuottamiin ennusteisiin. (23). Datan vinoutumalla voidaan tarkoittaa esimerkiksi sitä, että yksi tai useampi selittävistä muuttujista yli- tai alikorostuu. Jos aineisto ei ole edustava populaatiossa, algoritmin tuottama malli ei täsmää normaalisti jakautuneesta datasta tuotettuihin ennusteisiin. (15 s. 138). Tämä voi johtaa epäreiluun tai syrjivään lopputulokseen koneoppimista hyödyntävässä päätöksenteossa, esimerkiksi rekrytoinneissa tai luottopäätöksissä. (24).

2.6.2 Hyperparametrit

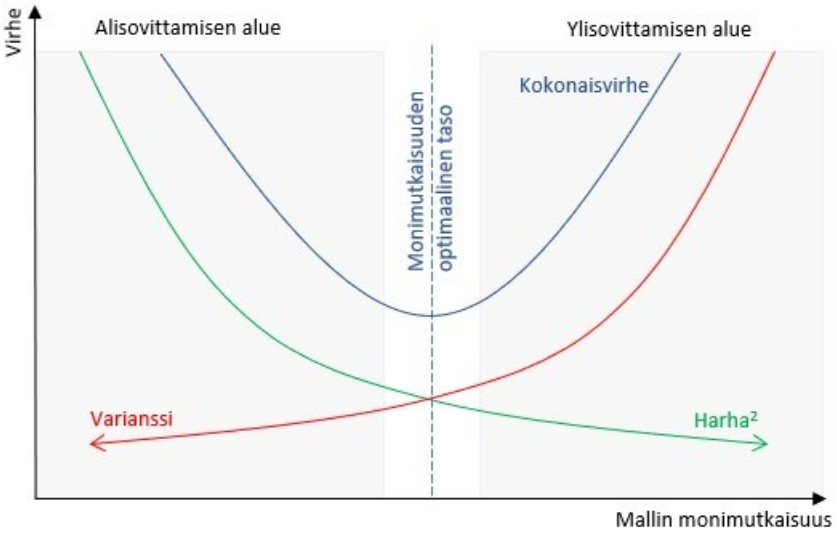
Koneoppimisalgoritmien hyperparametreilla voidaan vaikuttaa mallin kompleksisuuteen ja suorituskykyyn. Ne eivät ole sama asia kuin mallin parametrit kuten painoarvot, joita algoritmi säätää yrittäessään rakentaa datasta muuttujien välisiin yhteyksiin perustuvaa funktiota kohdemuuttujan arvon ennustamiseksi. Etsimällä algoritmikohtaisille hyperparametreille optimaaliset arvot voidaan minimoida mallin ennustevirheet. Hyperparametrit valitaan ja asetetaan ennen algoritmin suoritusta ja ne pysyvät vakiona koko mallin opettamisen ajan. (25).

Esimerkkeinä hyperparametreista ovat esimerkiksi käytettävän optimointialgoritmin valinta, klustereiden lukumäärä, neuroverkoissa käytettävän aktivointifunktion valinta, piilokerrosten määrä, epookkien määrä sekä virhe- tai tappiofunktioiden valinta. Tappiofunktio on olennainen suorituskyvyn arvioinnin kannalta. Se tuottaa tiedon siitä, kuinka tarkasti mallin tuottama ennuste vastaa todellista arvoa. Myös aineiston jaon suhde koulutus- ja testiaineistoon on yksi hyperparametri. (25)

2.6.3 Yli- ja alisovittaminen

Koneoppimismallin yleistämiskyky (generalization) on sen suorituskyvyn kannalta äärimmäisen tärkeä. Se kertoo, kuinka hyvin malli toimii aiemmin näkemättömällä datalla. (19). Koneoppimismalli on käytännössä tietoaineistoon sovitettu funktio, eli aineistoa parhaiten kuvaava matemaattinen malli. Yleistämiskyvyn kannalta on ongelmallista, mikäli malli on liian yksinkertainen tai liian monimutkainen. Monimutkaisuuden vaihteluun liittyvät käsitteet harha (bias) ja varianssi (variance). (16 s. 100).

Harhalla tarkoitetaan systemaattista mittausvirhettä, siis eroa mallin tuottaman keskimääräisen ennustevirheen ja todellisten arvojen välillä. Kun harha on suuri, malli yksinkertaistaa liikaa. Tätä kutsutaan alisovittamiseksi. Varianssi kuvaa mallin tuottamien ennustearvojen vaihtelua suhteessa todellisiin arvoihin. Mitä lähempänä ennustearvot ovat todellisia, sitä pienemmän arvon varianssi saa. Jos ennustearvot ovat kaukana todellisista, varianssi kasvaa. Tällöin malli on liian monimutkainen ja sen sanotaan olevan ylisovittava. (26). Optimaalisesti toimivan mallin löytyy siitä, missä harha ja varianssi ovat samanaikaisesti mahdollisimman pienet kuvassa 7 havainnollistetulla tavalla.



Kuva 7: Mallin optimaalinen monimutkaisuus (27)

Alisovittava mallin tuottamat tulokset ovat epätarkkoja sekä opetus- että testidatalla. Malli ei kykene löytämään tietojoukosta korrelaatioita selittävien muuttujien ja kohdemuuttujan välillä. Syynä tähän voi olla liian vähäinen opetusdatan määrä tai liian yksinkertaisen mallin käyttö monimutkaisempaan dataan, esimerkiksi yritys käyttää lineaarista mallia epälineaarisella datalla. (26).

Ylisovittava malli tuottaa opetusdatalla tarkkoja tuloksia, mutta suoriutuu huonosti, kun sitä käytetään aiemmin näkemättömällä datalla. Syynä tähän on, ettei opetusdata edusta koko populaatiota joko datassa olevasta kohinasta, liian pienestä opetusdatan määrästä tai liian suuresta määrästä selittäviä muuttujia. (26). Malli voi olla myös liian monimutkainen dataan nähden, esimerkiksi syvät päätöspuut (19).

Mallin yleistämiskyvyn arvioimiseksi tietoaineisto jaetaan vähintään kahteen erilliseen osaan, opetusdataan ja testidataan. Opetetun mallin yleistämiskykyä voidaan arvioida käyttämällä sitä mallille tuntemattomalla testidatalla. Tätä yksinkertaisinta menetelmää kutsutaan Holdout-tekniikaksi. (28). Tyypillisenä jakosuhteena käytetään opetuksessa 70 - 80 % osuutta ja 20 - 30 % osuutta testissä (19). Menetelmä on kuitenkin huono, jos aineiston määrä on vähäinen.

Vaihtoehtoisena tapana voidaan käyttää ristiinvalidointia. Siinä aineisto jaetaan useaan osajoukkoon, joista yhtä kerrallaan käytetään mallin validointiin ja muita sen opetukseen. Esimerkiksi usein käytetty K-kertainen ristiinvalidointi toimii tällä periaatteella: opetus ja validointi suoritetaan K-kertaa ja lopuksi tulokset keskiarvoistetaan. (28).

2.6.4 Mallien suorituskyvyn mittarit

Koneoppimismallien suorituskykyä arvioidaan eri tavoilla riippuen siitä, onko kyseessä luokittelutehtävä tai regressioon perustuva ennustaminen (29). Luokittelijan arvioinnissa yleisimmin käytetty mittari on *tarkkuus* (accuracy). Mallin kykyä yleistää arvioidaan laskemalla sen tarkkuus erikseen opetus- ja testidatalla. Luokittelijan tarkkuus lasketaan alla olevalla kaavalla. (30).

$$Tarkkuus\ (accuracy) = \frac{Oikein\ luokitellut\ tapaukset}{Kaikki\ tapaukset}$$

Tämä mittari toimii vain, jos eri luokkiin kuuluvien tapausten määrät ovat aineistossa suunnilleen samansuuruisia. Luokkien voimakas epätasapaino voi johtaa virheellisiin päätelmiin lopputuloksesta. (30). Esimerkiksi, jos luokittelutehtävän tavoitteena on diagnosoida vakavasti sairastuneet ja aineiston tapaukset jakautuvat suhteessa terveet 99 % / sairastuneet 1 %, voidaan 99 % tarkkuus saavuttaa pelkästään ennustamalla kaikki tapaukset suurempaan terveiden ryhmään kuuluvaksi. Tällöin joukosta ei tunnisteta ainoatakaan sairastunutta. Tästä syystä luokittelijan suorituskykyä on arvioitava myös muilla mittareilla.

Luokittelijan toimintaa binääriluokittelussa voidaan arvioida hyödyntämällä sekaannusmatriisia (confusion matrix), jossa tapaukset on jaettu 2 x 2 matriisiin, jossa oikein tunnistetut (TP) ja oikein hylätyt (TN) ovat luokiteltu oikein. Virheellisesti tunnistetut (FP) ja virheellisesti hylätyt (FN) ovat väärin luokiteltuja. (31).

	Kyllä (ennustettu)	**Ei (ennustettu)**
Kyllä (todellinen)	Oikein tunnistettu (True positive, TP)	Väärin hylätty (False negative, FN)
Ei (todellinen)	Väärin tunnistettu (False positive, FP)	Oikein hylätty (True negative, TN)

Näitä arvoja käytetään arvioitaessa luokittelijan hyvyyttä kahdella mittarilla. Sisäinen tarkkuus (precision) kertoo, kuinka suuri osa oikein tunnistetuista tapauksista on todellisuudessa oikein. Herkkyys (recall) kertoo, kuinka suuri osa oikein tunnistetuista tapauksista on tunnistettu oikein. (16 s. 176). Mitä suuremman arvot mittarit saavat, sen parempi tulos on. Sisäistä tarkkuutta käytetään, kun tavoitteena on minimoida väärin tunnistettujen määrä. Herkkyyttä käytetään, kun halutaan välttää väärin hylätyt tapaukset. (19).

$$Sisäinen\ tarkkuus\ (precision) = \frac{Oikein\ tunnistetut}{Oikeat\ positiiviset + väärät\ positiiviset}$$

$$Herkkyys\ (recall) = \frac{Oikein\ tunnistetut}{Oikeat\ positiiviset + väärät\ negatiiviset}$$

Kun mallin suorituskykyä optimoidaan, näiden kahden mittarin välillä joudutaan tekemään kompromisseja riippuen luokittelijan tavoitteista. Hyväksyttävien kynnysarvojen määrittely on aina tapauskohtainen liiketoimintapäätös. Binäärisessä luokittelutehtävässä kynnysarvon muutoksen vaikutuksia voidaan havainnollistaa ROC-kuvaajaa (Receiver Operator Characteristics) käyttämällä. Kuvaaja muodostetaan oikein tunnistettujen väärien positiivisten määrien funktiona. (16 s. 178). Kuvaajan rajaamasta pinta-alasta käytetään nimitystä AUC (Area Under Curve), joka tunnuslukuna kertoo luokittelijan hyvyydestä: Mitä lähempänä tunnusluku on arvoa 1, sitä paremmin luokittelija toimii. (16 s. 181).

Sisäisen tarkkuuden ja herkkyyden välisen tasapainon löytämiseksi voidaan apuna käyttää F1-arvoa (F1-score), joka on edellisten mittarien harmoninen keskiarvo. Mittari on käyttökelpoinen, kun aineiston luokat ovat jakautuneet epätasaisesti. Mitä suurempi F1 arvo on, sitä parempi mallin suorituskyky on. (30).

$$F1 = 2\ x\ \frac{Sisäinen\ tarkkuus\ x\ herkkyys}{Sisäinen\ tarkkuus + herkkyys}$$

Moniluokkaluokittelussa mallin suorituskyvyn arviointiin voidaan käyttää logaritmista tappiota (Logarithmic Loss, Log Loss). Sitä käytettäessä luokittelijan on määriteltävä jokaiselle luokalle todennäköisyys kaikista tapauksista. Logaritminen tappio saadaan laskemalla keskiarvo kaikista yksittäisten tapausten logaritmisista tappioista. Alla olevassa kaavassa y_{ij} kertoo, kuuluuko tapaus i luokkaan j vai ei. P_{ij} on todennäköisyys sille, että tapaus kuuluu joukkoon j. Mitä pienemmän arvon logaritminen tappio saa, sitä parempi tarkkuus mallilla on. Täydellisessä mallissa arvo on siis 0. (30).

$$Logaritminen\ tappio\ (Log\ Loss) = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log\left(p_{ij}\right)$$

Regressiomallien suorituskyvyn arviointi on luokittelumalleihin verrattuna suoraviivaisempaa. Hyvä malli tuottaa ennustearvoja, jotka ovat hyvin lähellä todellisia havaintoarvoja. Suorituskykyä mitataan vertaamalla näitä arvoja sekä koulutus- että testidatalla. Mittareina käytetään keskineliövirhettä (Mean Squared Error, MSE) ja keskimääräistä absoluuttista poikkeamaan (Mean Absolute Error, MAE). (19).

Keskineliövirhe lasketaan keskiarvona todellisten havaintoarvojen ja mallin ennustamien arvojen erotuksen neliöstä. Neliöön korottaminen korostaa suurempien erojen vaikutusta. (30):

$$Keskineliövirhe\ (MSE) = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2$$

Keskimääräinen absoluuttinen poikkeama on keskiarvo todellisten havaintoarvojen ja mallin ennustamien arvojen erotuksen itseisarvosta. Se kertoo, kuinka kaukana ennustetut arvot ovat todellisesta, mutta eivät sitä ovatko ennustetut

arvot todellisen ylä- vai alapuolella. Mitä pienempi arvo on, sen paremmin ennustetut arvot pitävät paikkansa. (30):

$$Keskimääinen\ absoluuttinen\ poikkeama\ (MAE) = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_j|$$

Jos mittarit saavat mallin testauksessa käytetyllä datalla olennaisesti suurempia arvoja kuin koulutusdatalla, viittaa tulos mallin ylioppimiseen.

2.6.5 Ongelmaan sopivan mallin valinta

Koneoppimismallin suoriutumisen kannalta on oleellista, että se sopii sekä käsillä olevaan ongelmaan (esimerkiksi luokittelu, regressio tai ryhmittely) että dataan (esimerkiksi lineaarinen/ei-lineaarinen) (32). Valinnassa huomioitavia tekijöitä ovat kokonaissuorituskyvyn lisäksi esimerkiksi käytössä oleva laskentateho ja muistin määrä, tietoaineisto, sen ominaisuudet ja datan määrä sekä mallin nopeus (19). Muita huomioitavia tekijöitä ovat esimerkiksi ylläpidettävyys ja käytössä oleva laskentakapasiteetti ja muut resurssit (32).

Mallin valinta on prosessi, jossa etsitään erilaisten mallien lisäksi myös kokeilemalla tiettyä mallia erilaisilla hyperparametrien arvoilla käsillä olevaan tehtävään parhaiten sopivaa mallia. Olennaista on huomata, että toimivin malli on aina tapauskohtainen, vaikka samankaltaisissa ongelmissa ja samankaltaisilla datoilla voidaan tietyntyyppisten mallien olevan yleensä toimivia. Datassa on aina tilastollista kohinaa ja puutteita. Osa algoritmeista vaatii lisäksi erityistä datan valmistelua. Eri malleissa on niiden käyttöön liittyviä rajoitteita, jotka vaikuttavat mallin suoriutumiseen tapauskohtaisesti. (32).

3 Luokittelussa yleisesti käytettyjä algoritmeja

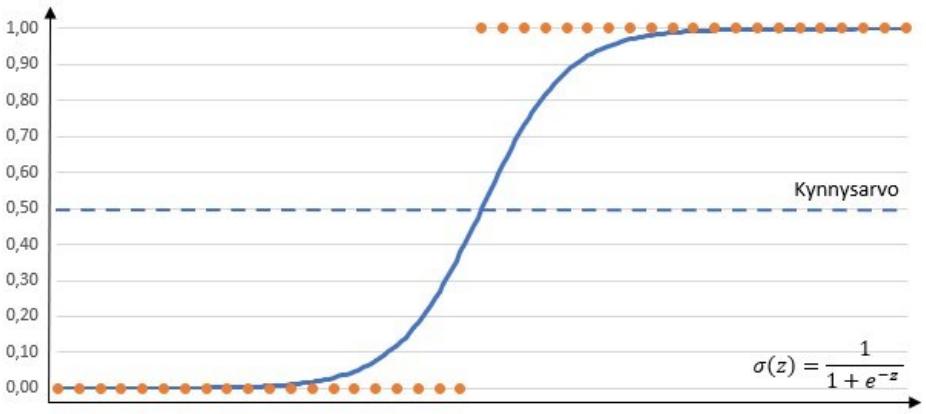
Erilaisia luokittelutehtävään soveltuvia algoritmeja on runsaslukuinen määrä, jonka vuoksi niitä kaikkia ei ole mahdollista tässä yhteydessä käydä läpi. Tässä luvussa esitellään lyhyesti kuusi erilaista algoritmia, jotka useimmiten ovat

nousseet esille opinnäytetyön lähteissä luokittelussa käytettyinä esimerkkeinä. Eri algoritmit soveltuvat eri käyttötarkoituksiin, jolloin käsillä olevan ongelman-ratkaisemiksi on osattava löytää juuri siihen tilanteeseen sopiva(t) algoritmi(t).

3.1 Logistinen regressio

Logistinen regressio on usein käytetty lineaarista regressiota laajentava luokittelualgoritmi, joka sopii käytettäväksi erityisesti binääriluokitteluun. Muuttujien välille muodostetaan lineaarinen riippuvuussuhde, mutta lineaarisen regression tuottamat jatkuvat arvot muunnetaan logistisella funktiolla (sigmoid) todennäköisyyksiin perustuen binäärisiksi arvoiksi 0 tai 1. (33 s. 169).

Kuvassa 8 on esitetty sigmoid-funktio ja sen tuottama havaintojen todennäköisyyksiä edustava S-muotoinen käyrä. Funktio palauttaa todennäköisyyttä vastaavan arvon väliltä 0 < y < 1. Kun funktion palauttama arvo on määriteltyä kynnysarvoa pienempi, kohdemuuttuja (kuvaajan oranssit pisteet) saa arvon 0, muutoin arvon 1. Kynnysarvoksi määritellään yleensä 50 % todennäköisyyttä vastaava arvo 0,5 (33 s. 169).



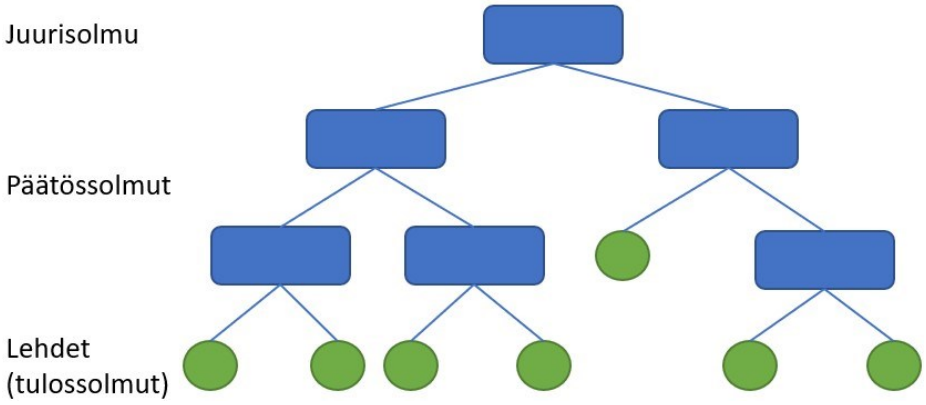
Kuva 8: Logistinen regressio ja sigmoid aktivointifunktio (34 s. 67)

Logistista regressiota voidaan käyttää myös multiluokkaluokittelussa. Se toteutetaan yleensä *yksi-vastaan-loput* tai *yksi-vastaan-yksi* menetelmillä. Yksi-vastaan-loput menetelmässä jokaiselle luokalle opetetaan binäärinen luokittelija,

joka yrittää erottaa luokan kaikista muista luokista. Testinäyte annetaan syötteenä erikseen jokaiselle luokittelijalle, jotka palauttavat sekä ennustetun kohdemuuttujan arvon ja todennäköisyyden ennusteelle. Suurimman ennustearvon saaneen luokittelijan luokkanimike palautetaan lopputuloksena. Yksi-vastaan-yksi menetelmässä koulutetaan yksi binäärinen luokittelija jokaiselle binääriselle luokkaparille. Kukin luokittelija ennustaa testinäytteelle luokan ja luokka, jota on ennustettu eniten, palautetaan lopputuloksena. (19).

3.2 Päätöspuu

Päätöspuut (decision tree) ovat yksi eniten käytetyimmistä ja tehokkaimmista koneoppimisen menetelmistä. Niiden algoritmit voivat tuottaa sekä jatkuvia arvoja että kategorisia arvoja, joten ne sopivat sekä regressio- että luokittelutehtäviin. (29). Päätöspuualgoritmi saa syötteenä näytteen selittävistä muuttujista muodostetun vektorin ja palauttaa kohdemuuttujan ennustetun arvon suorittamalla sarjan testejä tai kysymyksiä (33 s. 173). Testit voidaan visualisoida kuvan 9 mukaisesti ylösalaisin käännettynä puuna, joka muodostuu juuri- ja päätössolmuista sekä lehdistä. Jokainen päätössolmu esittää testiä, josta päätössääntöjen perusteella siirrytään seuraavaan solmuun, kunnes päädytään päättelyn tulosta edustavaan lehteen. (16 s. 125).



Kuva 9: Päätöspuu (14 s. 80)

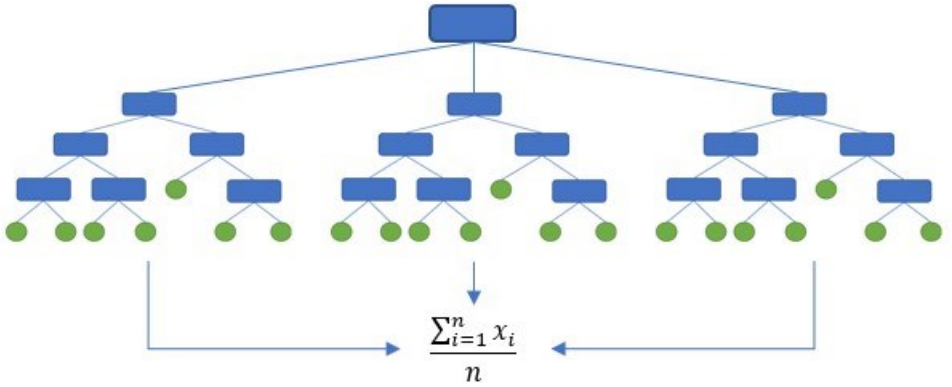
Päätöspuiden toimintaperiaate on yksinkertainen: Algoritmi jakaa aineistoa toistuvasti päätössolmuissa osiin selittävien muuttujien ja jakokriteerien avulla (16 s. 126). Yleisimmin käytettyjä jakokriteereitä ovat epäpuhtautta mittaavat giniindeksi ja entropia. Puhtaudella tarkoitetaan tilannetta, jossa solmun kaikki havainnot ovat samaan luokkaan kuuluvia. Jos puhtautta ei voi lisätä, jaon informaatiohyöty (information gain) on suurin mahdollinen ja kyseistä solmusta tulee lehti eli tulossolmu. (35).

Tavoitteena on löytää joukko päätöspuun juurisolmusta yksittäiseen lehteen johtavien polkujen määrittämiä päätössääntöjä, jotka jakavat opetusaineiston saman kohdemuuttujan arvon sisältämiksi joukoiksi. Uudelle näytteelle ennustettu kohdemuuttujan arvo määräytyy päätössääntöjen perusteella. (15 s. 134).

Yksi päätöspuiden hyödyllisistä ominaisuuksista on, että opetuksen jälkeen puusta saa selville jokaisen selittävän muuttujan merkityksen päätöksissä. Päätöspuut ovat myös helppo visualisoida ja ymmärtää, koska ne noudattavat pitkälti ihmismäistä päättelylogiikkaa. Huonoja puolia on, että pienetkin muutokset opetusaineistossa voivat johtaa merkittäviin puun rakenteen muutoksiin. Tämä osaltaan tekee päätöspuut alttiiksi suurelle varianssille ja ylisovittamiselle. (19).

3.3 Satunnaismetsä

Satunnaismetsä (Random Forest) muodostuu suuresta joukosta edellä kuvattuja päätöspuita, jotka on luotu Bagging-menelmää (Boostrap Aggregating) käyttäen. Menetelmässä opetusaineistosta valitaan satunnaisia näytteitä, joista jokaisella on yhtä suuri todennäköisyys tulla valituksi satunnaisesti valittuun aineistoon. Jokainen metsään kuuluva päätöspuu saa tällaisen satunnaisesti valitun opetusaineiston osajoukon, mikä vähentää päätöspuiden välisiä korrelaatioita. Satunnaismetsän tuottama lopputulos on siihen kuuluvien päätöspuiden tulosten keskiarvo. (33 ss. 177-178).

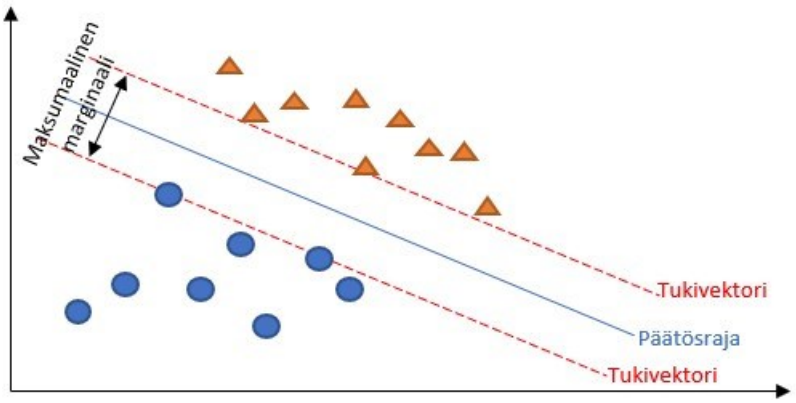


Kuva 10: Satunnaismetsä

Satunnaismetsät ovat tehokkaita, koska ne vähentävät päätöspuille tyypillistä suurta varianssia lisäämättä harhaa. Mallit eivät ole yleensä ylisovittavia, paitsi jos opetusaineistossa on paljon kohinaa. Satunnaismetsien hyviin puoliin kuuluu, että ne pystyvät käsittelemään suuria moniulotteisia data-aineistoja ja aineistosta puuttuvia arvoja. (33 ss. 178-179).

3.4 Tukivektorikone

Erityisesti binäärisissä luokittelutehtävissä käytetty tukivektorikone (Support Vector Machine, SVM) pyrkii luomaan aineiston havainnot toisistaan erottavan hypertason eli päätösrajan. Lineaarisesti eroteltavalla datalla on ääretön määrä tasoja, jotka luokittelevat havainnot oikein. Tukivektorikoneen algoritmi pyrkii löytämään tason, jossa kaikki havainnot ovat oikein luokiteltuja ja samalla mahdollisimman kaukana päätösrajana toimivasta hypertasosta. Lähimpänä päätösrajaa olevien havaintojen kautta kulkevia hypertason suuntaisia tasoja kutsutaan tukivektoreiksi. Algoritmi maksimoi näiden tukivektorien välisen etäisyyden. Luokittelija jakaa uudet havainnot luokkiin sen perusteella, kummalla puolella päätösrajaa ne ovat. (33 ss. 169-170). Tukivektorikoneen periaatetta on havainnollistettu kuvassa 11.



Kuva 11: Tukivektorikoneen periaate (14 s. 85)

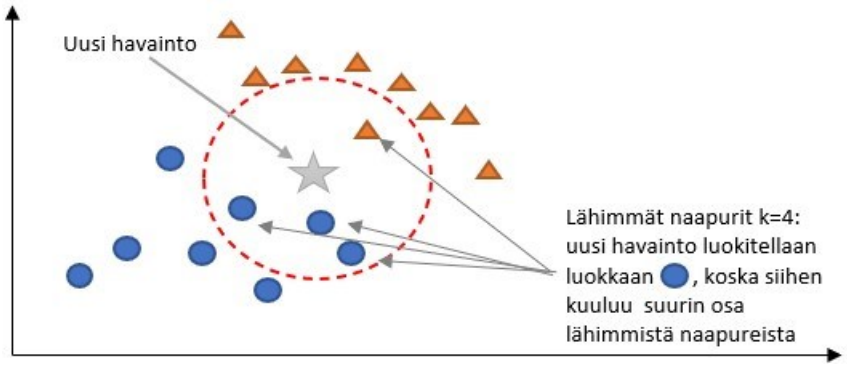
Reaalimaailmassa havaintoaineisto on harvoin täysin lineaarisesti eroteltavissa. Ei-lineaarisesti separoituvan datan käsittelyssä voidaan hyödyntää kernel-funktioita, jotka laajentavat alkuperäisten selittävien muuttujien joukkoa lisäämällä siihen uusia dimensioita. Tällöin kaksiulotteisessa koordinaatistossa viivana esitetty hypertaso sovitetaan tasoksi moniulotteiseen koordinaatistoon tukivektorikoneen periaatteen mukaisesti maksimoiden tason etäisyyden lähimmistä havainnoista. (33 ss. 170-171).

Tukivektorikonetta voidaan käyttää moniluokkaluokittelussa yksi-vastaan-loput tai yksi-vastaan-yksi menetelmillä vastaavalla tavalla, kuin aiemmin logistisen regression kohdalla kuvattiin (19). Tukivektorikonetta voi käyttää myös regressio-ongelmiin. Algoritmin vahvuuksia on sen hyvä tarkkuus ja suorituskyky pienillä havaintoaineistoilla. Aineistomäärien kasvaessa algoritmi kuitenkin vaatii paljon laskentatehoa ja aikaa. Aineiston kohina ja päällekkäiset luokat heikentävät algoritmin suorituskykyä. (16 s. 123).

3.5 K-lähintä naapuria

K-lähimmän naapurin (K-Nearest Neighbour, KNN) algoritmin käyttö luokittelussa perustuu siihen, mihin luokkaan uutta havaintoa lähimmät aiemmat havainnot kuuluvat. Menetelmässä lasketaan uuden havainnon ja kaikkien opetusaineiston havaintojen väliset etäisyydet. Mittana voidaan käyttää esimerkiksi

euklidista etäisyyttä tai Manhattan etäisyyttä. Tämän jälkeen valitaan hyperparametrille K valittu määrä uutta havaintoa lähimpänä olevia havaintoja ja lasketaan, mitä luokkaa edustavia havaintoja on eniten uuden havainnon lähellä. Tämä kertoo todennäköisyyden sille, mihin luokkaa uusi havainto kuuluu. Havainto sijoitetaan siihen luokkaan, johon suurin osa lähimmistä naapureita sijoittuu. (33 ss. 171-172). Menetelmää on havainnollistettu kuvassa 12.



Kuva 12: K-lähimmän naapurin luokittelija (16 s. 119)

KNN-algoritmi eroaa muista algoritmeista siinä, ettei se rakenna opetusvaiheessa mallia, vaan toistaa edellä kuvattua laskentaa jokaiselle uudelle havainnolle. Algoritmin vahvuutena on sen yksinkertaisuus ja ymmärrettävyys, käyttökelpoisuus monenlaisille aineistoille ja tehokkuus, kun aineistoa on paljon. Toisaalta, koska jokaiselle havainnolle on laskettava erikseen etäisyys muihin, algoritmi vaatii paljon laskentatehoa. Jos hyperparametrin K arvo valitaan huonosti, myös tulokset voivat olla huonoja. (16 ss. 119-120). Liian pienet K:n arvot johtavat ylisovittamiseen, liian suuret alisovittamiseen (29).

3.6 Naïve Bayes

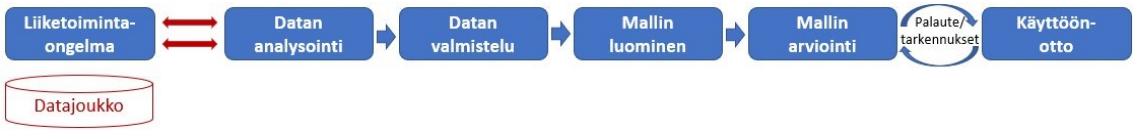
Bayesin teoreemaa P(A | B) = (P(B | A) P(A)) / P(B) soveltava Naïve Bayes perustuu tapahtuman A (kohdemuuttujan luokka) todennäköisyyteen ehdolla, että B (selittävä muuttuja) havaitaan. Malli oppii tiettyyn luokkaan kuuluvan havainnon todennäköisyyden jokaista muuttujaa erikseen arvioimalla. (33 s. 182). Luokittelussa oletetaan ”naiivisti”, että kaikki havaintoaineiston selittävät muuttujat

ovat ehdollisesti toisistaan riippumattomia. (14 s. 77). Todellisuudessa muuttujien välillä on yleensä korrelaatioita, mutta tästä huolimatta Naïve Bayes tuottaa usein monia muita algoritmeja parempia tuloksia. Se on nopea ja toimiva myös pienillä havaintoaineistoilla, eikä se ole herkkä kohinalle. (16 s. 124).

4 Mallien käyttö luokittelutehtävässä

4.1 Koneoppimisprosessin kulku

Tässä opinnäytetyössä luokittelutehtävän toteutus noudattaa kuvassa 13 havainnollistettua koneoppimisen prosessin tyypillistä etenemistä.



Kuva 13: Koneoppimisprosessin kulku

Ensimmäiseksi määritellään ratkaistava ongelma, tavoitteet, prosessi ja oletukset. Määrittelyssä on tunnistettava ongelman luonne: Onko kyse luokittelu-, regressio- vai optimointiongelma. Tämän perusteella valitaan mittarit, joilla mallin suoriutumista voidaan arvioida. (29)

Seuraavassa vaiheessa luodaan määritellyn ongelman ratkaisuun sopiva koneoppimismalli. Sen suorituskyvyn varmistamiseksi on huolehdittava siitä, että käytettävä data-aineisto on käyttökelpoista. (29). Datan analysointi ja valmistelu varmistavat, että data on koherenttia ja yhteismitallista sen lisäksi, että se on koneen ymmärtämässä numeraalisessa muodossa (16 s. 79). Tähän vaiheeseen kuuluu myös data-aineiston jakaminen erikseen algoritmin opetuksessa käytettävään opetusdataan ja mallin validoinnissa käytettävään testidataan (5 s. 360).

Koneoppimismallin luomisen jälkeen sen suorituskykyä arvioidaan kokeilemalla mallia aiemmin näkemättömään dataan (testidata) ja arvioimalla mallin

suoriutumista määritellyillä mittareilla. Parhaan suorituskyvyn löytämiseksi mallin luomiseen ja arviointiin sisältyy ennen käyttöönottoa usein monia iteraatioita mahdollisten algoritmien vaihdosten ja niiden parametrien säätämisen vuoksi. (29).

4.2 Tehtävän kuvaus

Vuonna 2010 perustettu ja 2017 Googlen omistukseen siirtynyt Kaggle on kaikille avoin verkkoyhteisö datatieteilijöille sekä koneoppimisen ja tekoälyn parissa työskenteleville ja alan opiskelijoille. Sivuston käyttäjät voivat etsiä ja julkaista tietojoukkoja, tutkia ja rakentaa niistä malleja sekä osallistua sivustolla julkaistuihin kilpailuihin erilaisten datatieteen haasteiden ratkaisemiseksi. Yhteisöllä oli vuonna 2021 yli 8 miljoonaa rekisteröityä käyttäjää. (36).

Tässä opinnäytetyössä verrataan joitakin koneoppimismalleja luokittelutehtävästä suoriutumisessa. Koska työlle ei ole toimeksiantajaa, on siinä käytetty Kagglesta löytyvää tehtävänantoa ja tietoaineistoa tekoälypohjaisen savuilmaisimen kehittämiseksi. Tehtävä ja siihen liittyvä tietoaineisto löytyvät osoitteesta https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset

Tietoaineisto sisältää 62 630 riviä sensoreilla kerättyä dataa, jonka perusteella mallin tulee tehdä binäärinen päätös kyllä/ei palohälytyksen tarpeesta. Alkuperäisen tietoaineiston on kerännyt Stefan Blattmann projektissaan ”Real-time Smoke Detection with AI-based Sensor Fusion.”. Elokuussa 2022 julkaistu projekti ja kaikki siihen liittyvät tekniset yksityiskohdat löytyvät sivustolta https://www.hackster.io/stefanblattmann/real-time-smoke-detection-with-ai-based-sensor-fusion-1086e6.

Binäärisen luokittelutehtävän toteutukseen on valittu joitakin yleisimpiä luokittelutehtävissä käytettyjä algoritmeja, jotka on esitelty lyhyesti aiemmassa luvussa. Datan käsittelyssä ja koneoppimismallin toteutuksessa käytetty ohjelmointikieli on Python 3.9.7 ja toteutusympäristö Jupyter Notebook 6.4.5.

Toteutettuja malleja verrataan luvussa 2.6.4 esitetyillä suorituskykymittareilla:

- Tarkkuus (accuracy)
- Sisäinen tarkkuus (precision)
- Herkkyys (recall)
- F1
- Sekaannusmatriisi (confusion matrix)

Suorituskyvyn optimoinnissa ja arvioinnissa on kiinnitettävä huomiota tehtävän luonteeseen: Koska mallin tulee kyetä laukaisemaan palohälytys, mahdollisissa optimointitilanteissa hyväksytään ennemmin väärin tunnistetut (FP) kuin väärin hylätyt (FN) tapaukset.

4.3 Datan analysointi

Kuten aiemmissa luvuissa on kerrottu, algoritmit osaavat käsitellä vain numeerista tietoa. Todellisessa elämässä data sisältää usein myös muita kuin numeerisia tietotyyppejä, lisäksi myös monikertaisia, virheellisiä, poikkeavia tai kokonaan puuttuvia arvoja. Turhat muuttujat lisäävät aineiston kohinaa, mutta aineistosta voi myös puuttua olennaisia lopputulokseen vaikuttavia muuttujia. Toisaalta muuttujien liiallinen määrä hidastaa algoritmeja, mutta voi joissain tilanteissa johtaa ylioppimiseen. Datan tilastolliset piirteet, esimerkiksi jakauman vinoudet tai kohdemuuttujan luokkien epätasapaino vaikuttavat koneoppimismallin suorituskykyyn. Tästä syystä ensimmäiseksi tutkitaan käytössä olevaa dataa ja tarvittaessa esikäsittelyvaiheessa siivotaan, korjataan ja muunnetaan se koneoppimisalgoritmien käyttöön soveltuvaan muotoon.

Aloitetaan lukemalla datatiedosto Pythonin dataframe-tietorakenteeseen:

```
# Luetaan sensoridata dataframeen
import pandas as pd
df = pd.read_csv("../data/smoke_detection_iot.csv")
```

Sarakkeet, niiden tyypit ja tyhjien arvojen määrät:

```
print(df.info())

RangeIndex: 62630 entries, 0 to 62629
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      62630 non-null  int64
 1   UTC             62630 non-null  int64
 2   Temperature[C]  62630 non-null  float64
 3   Humidity[%]     62630 non-null  float64
 4   TVOC[ppb]       62630 non-null  int64
 5   eCO2[ppm]       62630 non-null  int64
 6   Raw H2          62630 non-null  int64
 7   Raw Ethanol     62630 non-null  int64
 8   Pressure[hPa]   62630 non-null  float64
 9   PM1.0           62630 non-null  float64
 10  PM2.5           62630 non-null  float64
 11  NC0.5           62630 non-null  float64
 12  NC1.0           62630 non-null  float64
 13  NC2.5           62630 non-null  float64
 14  CNT             62630 non-null  int64
 15  Fire Alarm      62630 non-null  int64
dtypes: float64(8), int64(8)
memory usage: 7.6 MB
None
```

Kaikkien sarakkeiden tyyppinä on kokonais- tai liukuluku, joten muuttujat eivät sisällä kirjaimia tai muita ylimääräisiä merkkejä. Aineistoa ei siis tämän osalta tarvitse korjata tai siivota. Kaikissa sarakkeissa on sama määrä arvoja.

Tarkastellaan esimerkkejä aineiston sisällöstä tulostamalla siitä muutamia ensimmäisistä ja viimeisistä riveistä:

```
print(df)

       Unnamed: 0         UTC  Temperature[C]  Humidity[%]  TVOC[ppb]  \
0               0  1654733331          20.000        57.36          0
1               1  1654733332          20.015        56.67          0
2               2  1654733333          20.029        55.96          0
3               3  1654733334          20.044        55.28          0
4               4  1654733335          20.059        54.69          0
...           ...         ...             ...          ...        ...
62625       62625  1655130047          18.438        15.79        625
62626       62626  1655130048          18.653        15.87        612
62627       62627  1655130049          18.867        15.84        627
62628       62628  1655130050          19.083        16.04        638
62629       62629  1655130051          19.299        16.52        643
```

```
       eCO2[ppm]  Raw H2  Raw Ethanol  Pressure[hPa]  PM1.0  PM2.5  NC0.5  \
0            400   12306        18520        939.735   0.00   0.00   0.00
1            400   12345        18651        939.744   0.00   0.00   0.00
2            400   12374        18764        939.738   0.00   0.00   0.00
3            400   12390        18849        939.736   0.00   0.00   0.00
4            400   12403        18921        939.744   0.00   0.00   0.00
...          ...     ...          ...            ...    ...    ...    ...
62625        400   13723        20569        936.670   0.63   0.65   4.32
62626        400   13731        20588        936.678   0.61   0.63   4.18
62627        400   13725        20582        936.687   0.57   0.60   3.95
62628        400   13712        20566        936.680   0.57   0.59   3.92
62629        400   13696        20543        936.676   0.57   0.59   3.90

       NC1.0  NC2.5   CNT  Fire Alarm
0      0.000  0.000     0           0
1      0.000  0.000     1           0
2      0.000  0.000     2           0
3      0.000  0.000     3           0
4      0.000  0.000     4           0
...      ...    ...   ...         ...
62625  0.673  0.015  5739           0
62626  0.652  0.015  5740           0
62627  0.617  0.014  5741           0
62628  0.611  0.014  5742           0
62629  0.607  0.014  5743           0

[62630 rows x 16 columns]
```

Tulostetaan tilastollista tietoa aineistosta. Tässä vaiheessa kannattaa tarkastella erityisesti keskiarvoa, minimi- ja maksimiarvoja sekä keskihajontaa. Tilastolliset tarkastelut on kuitenkin usein helpompi tehdä visuaalisesti, joten tarkempi tarkastelu tehdään myöhemmin aineiston visualisoinnin yhteydessä.

```
print(df.describe())
```

```
         Unnamed: 0           UTC  Temperature[C]   Humidity[%]     TVOC[ppb]
\
count  62630.000000  6.263000e+04    62630.000000  62630.000000  62630.000000
mean   31314.500000  1.654792e+09       15.970424     48.539499   1942.057528
std    18079.868017  1.100025e+05       14.359576      8.865367   7811.589055
min        0.000000  1.654712e+09      -22.010000     10.740000      0.000000
25%    15657.250000  1.654743e+09       10.994250     47.530000    130.000000
50%    31314.500000  1.654762e+09       20.130000     50.150000    981.000000
75%    46971.750000  1.654778e+09       25.409500     53.240000   1189.000000
max    62629.000000  1.655130e+09       59.930000     75.200000  60000.000000

          eCO2[ppm]        Raw H2   Raw Ethanol  Pressure[hPa]         PM1.0
\
count  62630.000000  62630.000000  62630.000000   62630.000000  62630.000000
mean     670.021044  12942.453936  19754.257912     938.627649    100.594309
std     1905.885439    272.464305    609.513156       1.331344    922.524245
min      400.000000  10668.000000  15317.000000     930.852000      0.000000
25%      400.000000  12830.000000  19435.000000     938.700000      1.280000
50%      400.000000  12924.000000  19501.000000     938.816000      1.810000
75%      438.000000  13109.000000  20078.000000     939.418000      2.090000
max    60000.000000  13803.000000  21410.000000     939.861000  14333.690000
```

```
              PM2.5         NC0.5         NC1.0         NC2.5           CNT  \
count  62630.000000  62630.000000  62630.000000  62630.000000  62630.000000
mean     184.467770    491.463608    203.586487     80.049042  10511.386157
std     1976.305615   4265.661251   2214.738556   1083.383189   7597.870997
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        1.340000      8.820000      1.384000      0.033000   3625.250000
50%        1.880000     12.450000      1.943000      0.044000   9336.000000
75%        2.180000     14.420000      2.249000      0.051000  17164.750000
max    45432.260000  61482.030000  51914.680000  30026.438000  24993.000000

         Fire Alarm
count  62630.000000
mean       0.714626
std        0.451596
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
```

Edellä olevan mukaisesti aineisto koostuu 62630 rivistä ja 16 sarakkeesta, jotka tehtävänannon perusteella sisältävät alla olevat tiedot:

#	Sarake	Kuvaus
1	Unnamed	Rivinumero
2	UTC	Sensorilukemien aikaleima UTC-sekunteina
3	Temperature[C]	Lämpötila celcius-asteina
4	Humidity[%]	Kosteusprosentti
5	TVOC[ppb]	Haihtuvien orgaanisten yhdisteiden kokonais-määrä miljardisosina
6	eCO2[ppm]	eCO2-ekvivalenttipitoisuus miljoonasosina (las-kettu eri arvoista, kuten TVCO)
7	Raw H2	Raaka molekyylivetypitoisuus
8	Raw Ethanol	Raakaetanolipitoisuus
9	Pressure[hPa]	Paine hehtopascalleina
10	PM1.0	Hiukkaset, halkaisija alle 1,0 µm
11	PM2.5	Hiukkaset, halkaisija 1,0 - 2,5 µm
12	NC0.5	Hiukkaspitoisuus, halkaisija alle 0,5 µm
13	NC1.0	Hiukkaspitoisuus, halkaisija 0,5 - 1,5 µm
14	NC2.5	Hiukkaspitoisuus, halkaisija alle 1,5 - 2,5 µm
15	CNT	Näytelaskuri
16	Fire Alarm	Kohdemuuttuja, saa arvon 1, kun tehdään palo-hälytys

Koska sarakkeet Unnamed, UTC ja CNT ovat selvästi tarpeettomia, poistetaan ne jo tässä vaiheessa aineistosta:

```
df.drop(['Unnamed: 0', 'UTC', 'CNT'], axis=1, inplace=True)
```

Tarkistetaan, sisältääkö aineisto tyhjiä arvoja:

```
print(df.isnull().sum())
```

```
Temperature[C]    0
Humidity[%]       0
TVOC[ppb]         0
eCO2[ppm]         0
Raw H2            0
Raw Ethanol       0
Pressure[hPa]     0
PM1.0             0
PM2.5             0
NC0.5             0
NC1.0             0
NC2.5             0
Fire Alarm        0
dtype: int64
```

Aineistossa ei ole tyhjiä arvoja, joten niiden käsittelystä ei tarvitse huolehtia.

Tarkastellaan luokkien tasapainoisuus:

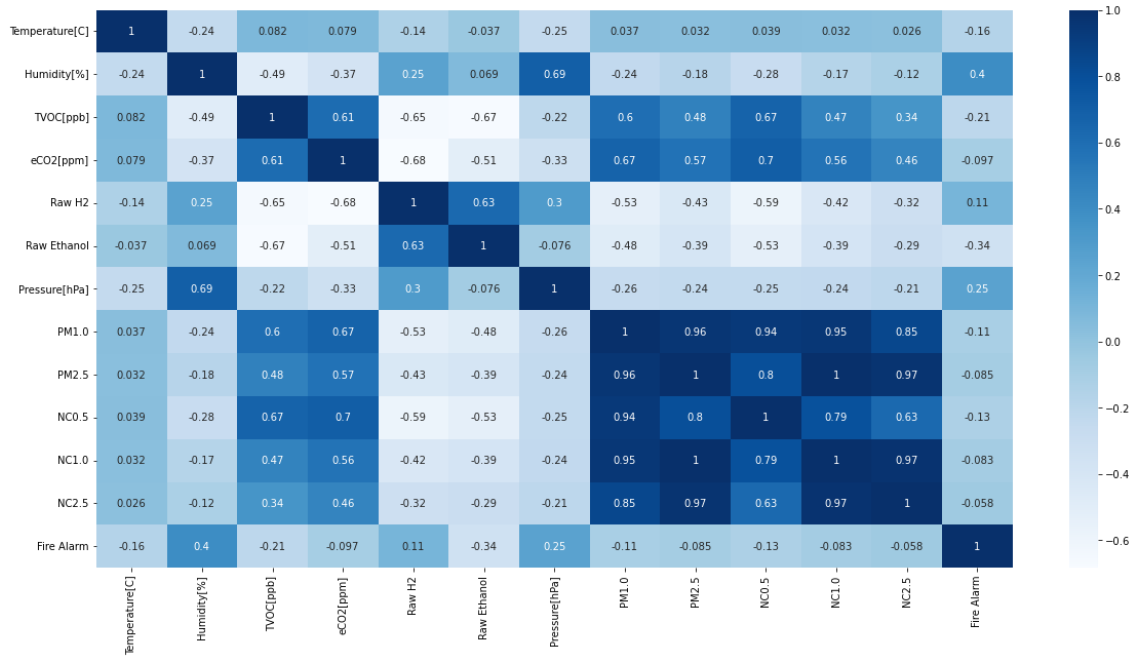
```
X, y = df.iloc[:, 0:13], df['Fire Alarm']
print(df['Fire Alarm'].value_counts())
```

```
1    44757
0    17873
Name: Fire Alarm, dtype: int64
```

Luokat ovat epätasapainoiset: Luokkaan 0 (ei hälytystä) kuuluu 17873 havaintoarvoa (28,5 %) ja luokkaan 1 (hälytys) 44757 havaintoarvoa (71,5 %). Tämä tulee huomioida suorituskykyä arvioinnissa, sillä epätasapaino vaikuttaa yhtenä mittareista käytettävään tarkkuuteen (accuracy).

Tarkastellaan muuttujien välisiä korrelaatioita käyttäen korrelaatiomatriisia:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True,cmap = 'Blues')
```



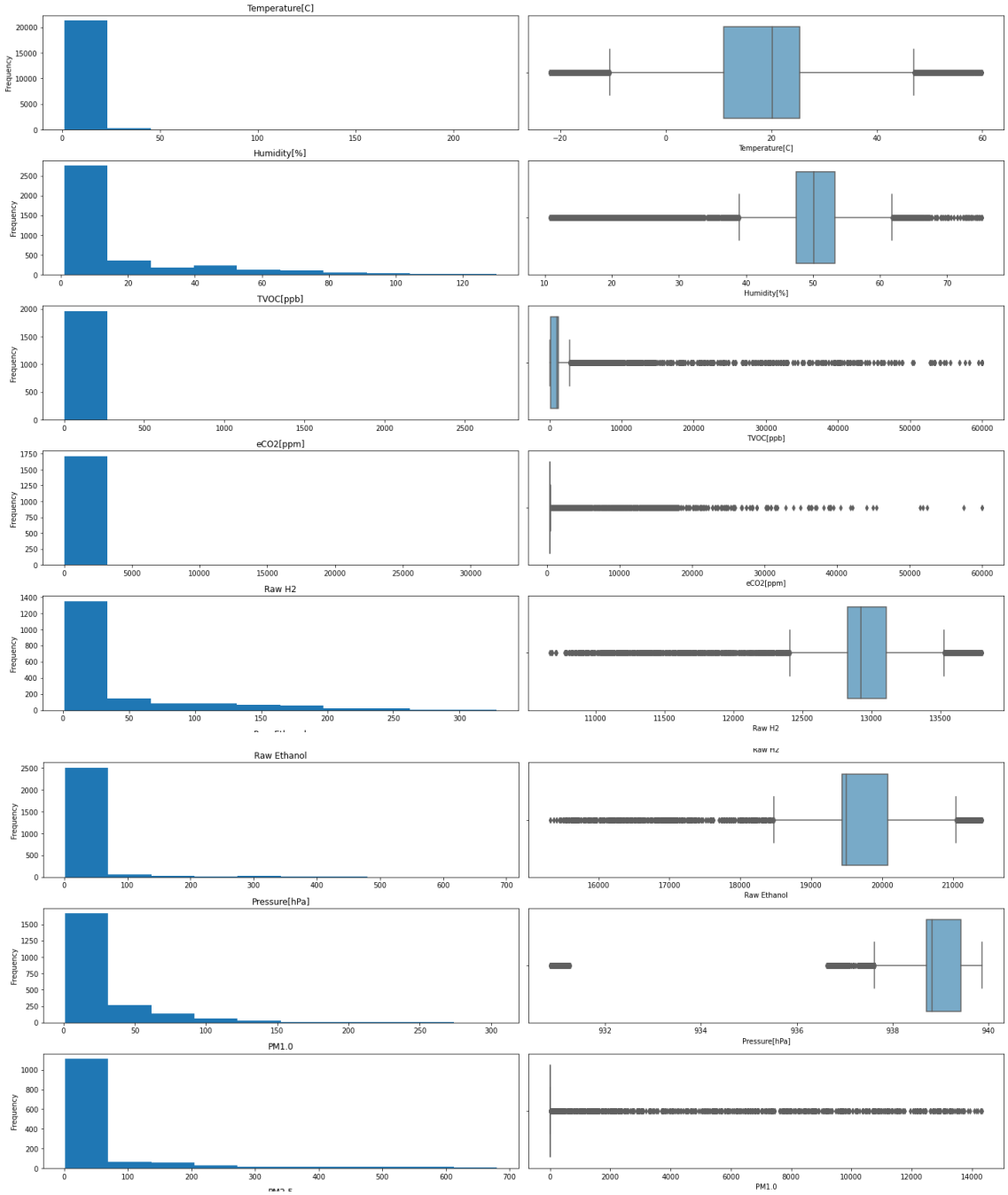
Korrelaatiomatriisissa esitetään kaikkien muuttujien keskinäiset korrelaatiota. Kahden muuttujan välistä korrelaatiota arvioidaan korrelaatiokertoimella: Mitä lähempänä kertoimen itseisarvo on arvoa 1, sitä suurempi korrelaatio muuttujilla on. Kertoimen arvolla 0 muuttujien välillä ei ole korrelaatiota ollenkaan. (37) Korrelaatiokertoimen ollessa alle 0,40 korrelaatio on heikkoa. Välillä 0,40 - 0,60 korrelaatio on kohtalainen, välillä 0,60 - 0,80 voimakas ja korrelaatiokertoimen ollessa yli 0,80 voidaan korrelaation sanoa olevan erittäin voimakas.

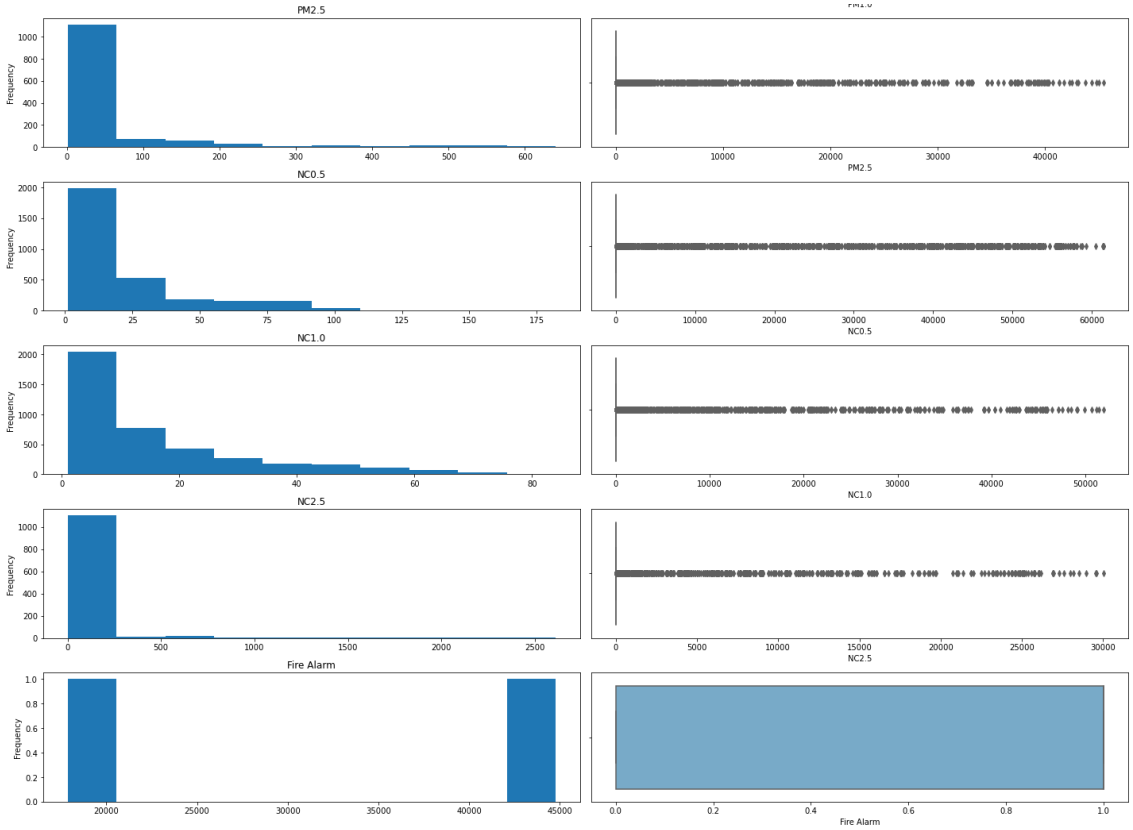
Korrelaatiomatriisia tarkasteltaessa voidaan havaita, ettei mikään selittävistä muuttujista korreloi kohdemuuttujan (Fire Alarm) kanssa. Hiukkaset (PM) ja hiukkaspitoisuudet (NC) korreloivat erittäin voimakkaasti keskenään. Haihtuvien orgaanisten yhdisteiden kokonaismäärä (TVOC) ja eCO2-ekvivalenttipitoisuudet korreloivat voimakkaasti, molemmat korreloivat myös hiukkasiin (PM) ja hiukkaspitoisuuksiin (NC). Paine ja kosteus korreloivat voimakkaasti keskenään, mutta lämpötilalla ei ole juurikaan korrelaatiota minkään muun muuttujan kanssa.

Korrelaatiomatriisin lisäksi muuttujien välisiä suhteita voidaan tarkastella sirontakaavioiden avulla. Ne auttavat tunnistamaan lineaarisen tai epälineaarisen riippuvuuden, jotka voivat vaikuttaa käytettävän algoritmin valintaan.

Tarkastellaan vielä muuttujia histogrammien ja boxplot-kaavioiden kautta:

```
i=1
plt.figure(figsize = (20,40))
for col in df.columns:
    plt.subplot(14,2,i)
    plt.title(col)
    df[col].value_counts().plot(kind ='hist')
    plt.subplot(14,2,i+1)
    sns.boxplot(x=pd.to_numeric(df[col]), palette='Blues')
    i=i+2
plt.tight_layout()
```





Histogrammien perusteella havaintoaineiston muuttujien jakaumat ovat vinoja. Boxplotkaaviot osoittavat, että poikkeavia havaintoarvoja (outliers) on hyvin paljon. Kuten luvussa 2.6.1 kerrottiin, tilastolliset vinoutumat voivat heikentää mallin kykyä tuottaa ennusteita normaalisti jakautuneesta aineistosta. Poikkeavien havaintoarvojen käsittely tulee huomioida datan esikäsittelyvaiheessa ja aineiston vinous suorituskykyä arvioitaessa.

4.4 Datan esikäsittely

Raakadata soveltuu vain harvoin sellaisenaan koneoppimisalgoritmien käsiteltäväksi. Edellisessä vaiheessa tehty datan analysointi auttaa havaitsemaan asiat, jotka datan esikäsittelyvaiheessa on korjattava. Hyviin käytäntöihin kuuluvat esimerkiksi puuttuvien, poikkeavien ja virheellisten arvojen käsittely, muuttujien koodaus (feature encoding) ja muuttujien skaalaus (feature scaling). (19). Tässä vaiheessa voidaan poistaa tarpeettomat muuttujat, mutta myös lisätä uusia muuttujia uusista tietolähteistä tai muodostaa niitä olemassa olevista muuttujista (feature engineering) (16 s. 102).

Datan analysointivaiheessa havaittiin, ettei aineistossa ollut puuttuvia arvoja. Jos niitä olisi ollut, tässä vaiheessa olisi päätettävä, kuinka ne käsitellään. Mikäli puuttuvia arvoja on suhteellisen vähän (esimerkiksi alle 5 % kaikista aineiston riveistä), puuttuvat arvot sisältävät rivit voi vain poistaa. Tässä kuitenkin on riski, että samalla muissa muuttujissa olevaa mallin kannalta olennaista tietoa poistuu. Vaihtoehtoinen tapa on täydentää puuttuvien arvojen tilalle arvot käyttämällä niiden tilalla kyseisen muuttujan keskiarvoa tai moodia. Joissakin tapauksissa voidaan puuttuvan arvon paikalle sijoittaa muuttujan arvovälin ulkopuolinen vakioarvo ja antaa algoritmin päättää, kuinka se toimii niiden kanssa. Suosittu tapa on myös käyttää K-NN -algoritmia ennustamaan puuttuvat arvot. (19).

Havaintoaineistossa oli ainoastaan numeerisia arvoja, joten muuttujien koodausta ei tarvitse tehdä. Usein data sisältää kategorisia muuttujia, esimerkiksi väri. Tällaisissa tapauksissa data on muunnettava algoritmien ymmärtämään numeeriseen muotoon koodaamalla muuttujat. Yksinkertaisin tapa on koodata muuttujat korvaamalla merkkijono numerolla (label encoding), esimerkiksi *punainen = 1, keltainen = 2, vihreä = 3* jne. Toinen tapa on lisätä aineistoon jokaiselle kategoriselle muuttujalle oma sarake (one-hot encoding), jossa arvoilla 0 ja 1 esitetään, esiintyykö arvo rivillä. (19).

Datan analysoinnissa selvisi myös, että aineistossa useimmilla muuttujilla on poikkeavia havaintoarvoja. Tilastolliset poikkeamat voivat vaikuttaa epäsuotuisasti algoritmin koulutukseen, joten poistetaan aineistosta rivit, joissa muuttujien arvot ovat yli 3 standardipoikkeaman päässä keskiarvosta:

```
from scipy import stats

z_scores = abs(stats.zscore(df))
filter = (z_scores < 3).all(axis=1)
df_filtered = df[filter]
print(df_filtered.shape)
```

Jäljelle jäi 59000 riviä:

```
(59000, 13)
```

Datan analysoinnissa selvisi, että tri muuttujien arvojen minimi- ja maksimiarvot vaihtelivat merkittävästi. Esimerkiksi paineen arvot ovat vaihtelevat välillä 930,852–939,861, mutta TVOC saa arvoja väliltä 0–60000. Tämä voi vaikuttaa joidenkin algoritmien toimintaan siten, että suuret lukuarvot saavat voimakkaasti ylikorostuvan painotuksen. Algoritmi ei kykene tunnistamaan asiaa, vaan ainoastaan numeraalisen arvon. Tästä syystä muuttujat skaalataan samalle asteikolle (feature scaling). Suosittu tapa on tehdä tämä normalisoimalla muuttujien arvot välille [-1, 1] tai [0, 1] siten, että minimi ja maksimiarvot saavat ääriarvot ja muut arvot skaalataan suhteessa niiden välille. Toinen tapa on standardointi, jossa kunkin muuttujan keskiarvoksi tulee nolla ja arvot jakautuvat yksikkökeskihajonnan mukaisesti sen ympärille. (19).

Normalisoidaan selittävien muuttujien arvot välille [0, 1]:

```
from sklearn.preprocessing import MinMaxScaler
min_max=MinMaxScaler()
df_scaled=pd.DataFrame(min_max.fit_transform(df_filtered),co-
lumns=df_filtered.columns)
print("\nNormalisoidut muuttujat\n")
print(df_scaled.describe())
```

Nyt kaikkien muuttujien minimiarvona on 0, maksimiarvona 1 ja muut arvot skaalattu suhteellisesti näiden välille:

```
       Temperature[C]   Humidity[%]     TVOC[ppb]     eCO2[ppm]         Raw H2
\
count    59000.000000  59000.000000  59000.000000  59000.000000  59000.000000
mean         0.503062      0.534018      0.032867      0.070828      0.522147
std          0.176357      0.089949      0.053311      0.144171      0.130051
min          0.000000      0.000000      0.000000      0.000000      0.000000
25%          0.446426      0.486258      0.005149      0.000000      0.422311
50%          0.566496      0.538215      0.038615      0.000000      0.493227
75%          0.633936      0.590550      0.047383      0.048860      0.636653
max          1.000000      1.000000      1.000000      1.000000      1.000000

        Raw Ethanol  Pressure[hPa]         PM1.0         PM2.5         NC0.5
\
count  59000.000000   59000.000000  59000.000000  59000.000000  59000.000000
mean       0.441565       0.676567      0.059601      0.059685      0.059577
std        0.157596       0.236611      0.027238      0.027155      0.027299
min        0.000000       0.000000      0.000000      0.000000      0.000000
25%        0.318292       0.630344      0.051058      0.051246      0.051086
50%        0.339465       0.678239      0.066010      0.065988      0.066084
75%        0.536619       0.883317      0.075857      0.075816      0.075835
max        1.000000       1.000000      1.000000      1.000000      1.000000
```

```
              NC1.0         NC2.5    Fire Alarm
count  59000.000000  59000.000000  59000.000000
mean       0.059708      0.015115      0.739593
std        0.027157      0.018541      0.438860
min        0.000000      0.000000      0.000000
25%        0.051239      0.012590      0.000000
50%        0.066098      0.015827      1.000000
75%        0.075851      0.018345      1.000000
max        1.000000      1.000000      1.000000
```

Aiemmassa datan analysointivaiheessa käsiteltävästä aineistosta poistettiin turhat sarakkeet: unnamed, UTC ja CNT. Esikäsittelyvaiheessa on vielä tarkasteltava tarvetta vähentää muuttujien määrää lisää. Aineisto sisältää useita hiukkasia (PM) ja hiukkaspitoisuuksia (NC) kuvaavia muuttujia, joiden välillä on erittäin voimakkaat korrelaatiot. Regressioanalyysissä tämä multikollineaarisuusongelmaksi kutsuttu tilanne voi olennaisesti vääristää regressiokertoimia kasvattaen niiden keskivirheitä, kun korrelaatiokertoimet ovat yli 0,9. (38). Tästä syystä on mahdollista, että toinen muuttujista PM1.0 tai PM2.5 sekä kaksi muuttujista NC0.5, NC1.0 tai NC2.5 on poistettava. Tässä vaiheessa voidaan kuitenkin jättää ne vielä talteen ja poistaa ne myöhemmin, mikäli siihen on aihetta.

Seuraavaksi jaetaan havaintoaineisto kahteen osaan. Käytetään 80 % datasta opetukseen ja 20 % mallien testaamiseen:

```
from sklearn.model_selection import train_test_split
# eriytetään kohdemuuttja aineistosta
X = df_scaled.drop(columns='Fire Alarm')
y = df_scaled['Fire Alarm']
# jaetaan aineisto 80 % opetukseen, 20 % testiin
x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=
0.2, random_state= 18, stratify= y)
```

Ennen kuin algoritmien opetus voidaan aloittaa, on vielä käsiteltävä datan analysoinnissa havaittu luokkien epätasapainoisuus: Havainnoista 28,5 % kuului luokkaan 0 (ei hälytystä) ja 71,5 % luokkaan 1 (hälytys). Epätasapaino voi johtaa mallin suorituskyvyn heikkenemiseen erityisesti niissä tapauksissa, joissa kohdemuuttuja kuuluu määrällisesti pienempään luokkaan. Jakauman epätasapaino voidaan korjata opetusaineistossa useammalla eri tavalla. Tässä on käytetty Imbalanced-Learn -kirjastoa ja RandomOverSampler -luokkaa. Alkuperäinen aineisto jää talteen, jolloin opetus voidaan tehdä molemmilla aineistolla ja verrata molemmilla tavoin koulutettujen mallien suoriutumista testidatalla.

```
# käsitellään luokkien epätasapainoisuus
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
print("\nLuokkien epätasapaino opetusaineistossa:")
print(Counter(y))
oversample = RandomOverSampler(sampling_strategy='minority')
X_over, y_over = oversample.fit_resample(X, y)
print("\nLuokat tasapainotettuina opetusaineistossa:")
print(Counter(y_over))
```

4.5 Algoritmien opetus

Kun data on edellisten vaiheiden jälkeen valmisteltu algoritmeille sopivaan muotoon, luodaan mallit aiemmin luvussa 3 esiteltyjä algoritmeja käyttäen ja käytetään mallia edellisessä vaiheessa havaintoaineistosta eroteltuun testiaineistoon.

Logistinen regressio:

```
titles, models, predictions = [], [], []
# Mallin luonti
print("Logistinen regressio, mallin luonti...")
from sklearn.linear_model import LogisticRegression
model_lr = LogisticRegression(max_iter=200)
model_lr.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_lr = model_lr.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Logistinen regressio")
models.append(model_lr)
predictions.append(prediction_lr)
print("...valmis!")
```

Päätöspuu:

```
# Mallin luonti
print("Päätöspuu, mallin luonti...")
from sklearn.tree import DecisionTreeClassifier
model_tree = DecisionTreeClassifier()
model_tree.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_tree = model_tree.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Päätöspuu")
models.append(model_tree)
predictions.append(prediction_tree)
print("...valmis!")
```

Satunnaismetsä:

```
# Mallin luonti
print("Satunnaismetsä, mallin luonti...")
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier()
model_rf.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_rf = model_rf.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Satunnaismetsä")
models.append(model_rf)
predictions.append(prediction_rf)
print("...valmis!")
```

Tukivektorikone:

```
# Mallin luonti
print("Tukivektorikone, mallin luonti...")
from sklearn import svm
model_svm = svm.SVC(kernel='linear')
model_svm.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_svm = model_svm.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Tukivektorikone")
models.append(model_svm)
predictions.append(prediction_svm)
print("...valmis!")
```

K-lähintä naapuria:

```
# Mallin luonti
print("K-lähintä naapuria, mallin luonti...")
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=5)
model_knn.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_knn = model_knn.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("K-lähintä naapuria")
models.append(model_knn)
predictions.append(prediction_knn)
print("...valmis!")
```

Algoritmille annetaan hyperparametrina lähimpien naapurien lukumäärä. Sopivan arvon määrittäminen tapahtuu käytännössä kokeilemalla. Tässä arvona on käytetty k=5, joka kokeilujen perusteella tuotti parhaat tulokset.

Naïve Bayes:

```
# Mallin luonti
print("Naïve Bayes, mallin luonti...")
from sklearn.naive_bayes import GaussianNB
model_nb = GaussianNB()
model_nb.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_nb = model_nb.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Naïve Bayes")
models.append(model_nb)
predictions.append(prediction_nb)
print("...valmis!")
```

4.6 Mallien suorituskyvyn arviointi

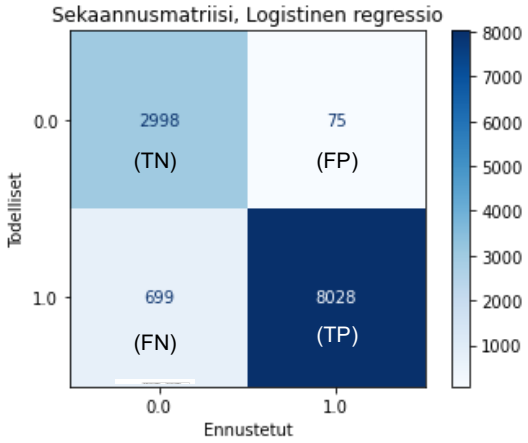
Mallien suorituskyvyn arvioinnissa on käytetty luvussa 2.6.4 esiteltyjä mittareita: Tarkkuus, sisäinen tarkkuus, herkkyys, F1 sekä sekaannusmatriisi.

```
from sklearn.metrics import precision_recall_fscore_support as prf
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Mallien suorituskyvyn mittarit
accuracy, precision, recall, fscore = [0]*6, [0]*6, [0]*6, [0]*6

for i in range(6):
    print("\n" + titles[i] + ", tulokset:")
    accuracy[i] = models[i].score(x_test, y_test)
    precision[i], recall[i], fscore[i], _ = prf(y_test, /
    predictions[i], average='binary')
    print("Tarkkuus (accuracy) = {:.4f}".format(accuracy[i]))
    print("Sisäinen tarkkuus (precision) {:.4f}".format(precision[i]))
    print("Herkkyys (recall) {:.4f}".format(recall[i]))
    print("F1 {:.4f}".format(fscore[i]))
    # Sekaannusmatriisi (confusion matrix)
    cm = confusion_matrix(y_test, predictions[i], /
    labels=models[i].classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, /
    display_labels=models[i].classes_)
    disp.plot(cmap=plt.cm.Blues)
    disp.ax_.set_title("Sekaannusmatriisi, " + titles[i])
    disp.ax_.set_ylabel("Todelliset")
    disp.ax_.set_xlabel("Ennustetut")
    plt.show()
```

Kun kyseessä on palohälytin, hyväksytään mieluummin aiheettomat hälytykset (väärä positiivinen, FP) kuin ettei hälytin reagoi (väärä negatiivinen, FN). Tällöin herkkyys (recall) kuvaa parhaiten mallin onnistumista.

Logistinen regressio, tulokset:

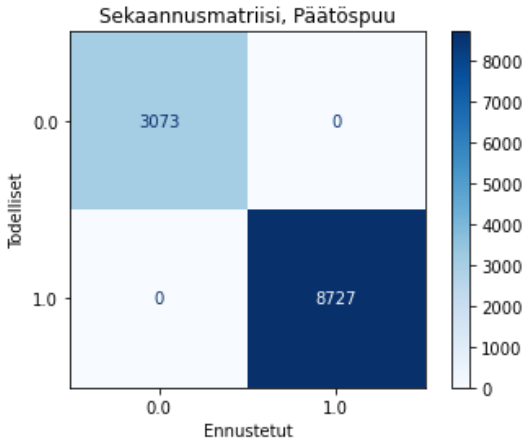
```
Tarkkuus (accuracy) = 0.9344
Sisäinen tarkkuus (precision) = 0.9907
Herkkyys (recall) = 0.9199
F1 = 0.9540
```



Mittarien perusteella logistisen regression tarkkuus on 93,44 %. Herkkyys on 91,99 % (sekaannusmatriisista TP / (TP+FN) = 8028 / (8028+699) = 0,9199). Tulosten perusteella hälytin ei reagoisi kaikissa tapauksissa, joissa olisi siihen olisi aihetta. Tämän vuoksi malli ei vaikuta olevan paras mahdollinen.

Päätöspuu, tulokset:

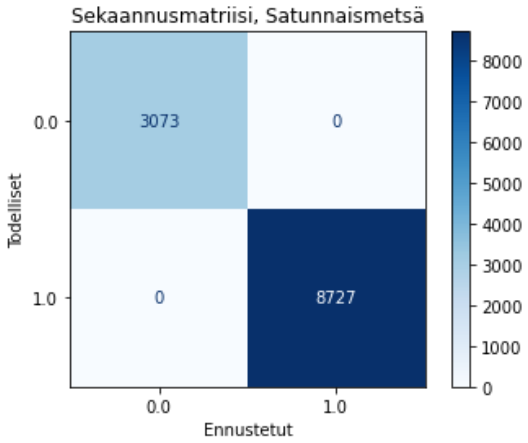
```
Tarkkuus (accuracy) = 1.0000
Sisäinen tarkkuus (precision) 1.0000
Herkkyys (recall) 1.0000
F1 1.0000
```



Päätöspuun kaikki mittarit ovat parhaat mahdolliset: Malli näyttää ennustavan tapaukset 100 % oikein. Tämän perusteella malli sopii hyvin käyttötarkoitukseensa palohälyttimessä.

Satunnaismetsä, tulokset:

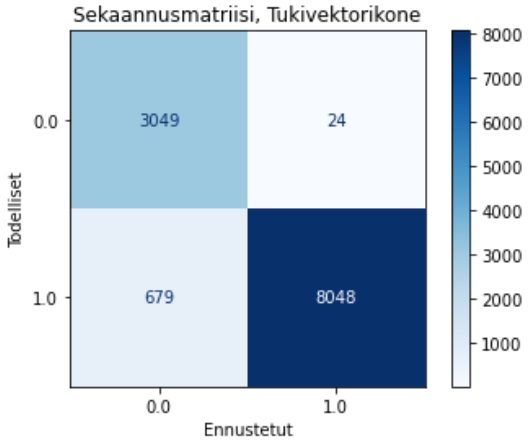
```
Tarkkuus (accuracy) = 1.0000
Sisäinen tarkkuus (precision) 1.0000
Herkkyys (recall) 1.0000
F1 1.0000
```



Satunnaismetsä tuottaa päätöspuun tavoin täysin virheettömät tulokset.

Tukivektorikone, tulokset:

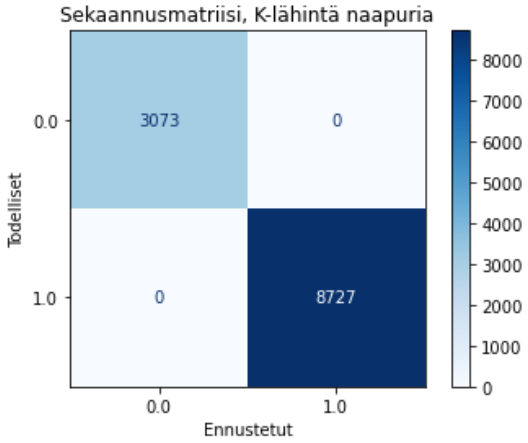
```
Tarkkuus (accuracy) = 0.9404
Sisäinen tarkkuus (precision) 0.9970
Herkkyys (recall) 0.9222
F1 0.9582
```



Tukivektorikoneen tarkkuus on 94,04 % ja herkkyys 92,2 %, eli suunnilleen logistisen regressiomallin tasolla. Hälytin ei reagoisi kaikissa tapauksissa, joissa sen tulisi reagoida, jonka vuoksi malli ei vaikuta olevan käyttötarkoitukseen paras mahdollinen.

K-lähintä naapuria, tulokset:

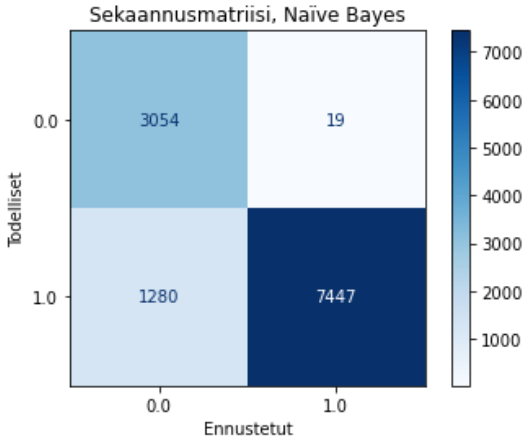
```
Tarkkuus (accuracy) = 1.0000
Sisäinen tarkkuus (precision) 1.0000
Herkkyys (recall) 1.0000
F1 1.0000
```



Kuten päätöspuu ja satunnaismetsä, K-lähintä naapuria tuottaa virheettömät tulokset, kun hyperparametrina käytettävä naapurien lukumäärä n=5. Parametrin arvoa muuttamalla voidaan vaikuttaa mallin tarkkuuteen. Suuremmilla arvoilla, esimerkiksi n=10 mallin tulos on edelleen hyvä, mutta ei enää täydellinen.

Naïve Bayes, tulokset:

```
Tarkkuus (accuracy) = 0.8899
Sisäinen tarkkuus (precision) 0.9975
Herkkyys (recall) 0.8533
F1 0.9198
```



Naïve Bayes tuotti kokeilluista algoritmeista heikoimmat tulokset. Mallin tarkkuus oli 88,99 % ja herkkyys vain 85,33 %. Malli ei reagoi tapauksissa, joissa hälytykseen olisi aihetta ja näin suurella erolla se ei vaikuta sopivan käyttötarkoitukseen ollenkaan.

Luvussa 4.3 korrelaatiomatriisin yhteydessä havaittiin mahdollinen multikollineriaalisuusongelma, joka saattaa vaikuttaa haitallisesti tuloksiin. Ongelman poistamiseksi voidaan havaintoaineistosta poistaa joitakin muuttujia:

```
df.drop(['NC0.5','NC1.0','PM1.0'], axis=1, inplace=True)
```

Tämä ei käytännössä vaikuttanut mallien suorituskykyyn lukuun ottamatta Naïve Bayesia, jonka tulokset paranivat hieman. Muuttujien poistamisen jälkeen mallin tarkkuus parani 91,70 %:iin ja herkkyys 89,99 %:iin. Malli ei kuitenkaan tämän jälkeenkään vaikuta käyttötarkoitukseen sopivalta.

5 Yhteenveto

Viimeisen vuosikymmenen aikana digitalisaatio on räjäyttänyt datan määrän valtavaan kasvuun koneiden laskentatehon samalla kasvaessa. Data on ravintoa tekoälyalgoritmeille, jotka kykenevät hyödyntämään valtavia datamassoja lukemattomissa eri käyttötarkoituksissa. Käytämme sovelluksia päivittäisessä arjessa huomaamattamme. Vaikka tekoäly tarjoaa ennen näkemättömät

mahdollisuudet parantaa ihmisten elämää, sitä voidaan käyttää myös pahassa. On tärkeää pitää keskustelussa mukana myös tekoälyn käytön eettiset näkökulmat.

Terminologiassa tekoäly on sateenvarjomainen käsite, joka tarkoittaa yleisellä tasolla koneen kykyä jäljitellä ihmisaivojen toimintaa. Koneoppiminen on yksi tekoälyn osa-alue, jota käytetään etukäteen tarkasti määritellyn tehtävän toteuttamisessa. Datasta eristetään ne piirteet, joiden perusteella matemaattisia malleja ja tilastollisia menetelmiä hyödyntäen voidaan ennustaa todennäköisin tulos myös uudesta, aiemmin näkemättömästä datasta. Algoritmit kykenevät löytämään korrelaatioita piilevien tekijöiden väliltä, mutta on huomattava, etteivät ne kykene ymmärtämään kausaliteettia. Algoritmit käsittelevät vain numeraalista dataa ymmärtämättä sen sisältöä tai inhimillisiä merkityksiä

On erittäin tärkeää ymmärtää koneoppimismallien opetuksessa käytettävän datan vaikutus lopputulokseen. Vinoutunut data voi johtaa vääriin tulkintoihin ja eettisestä näkökulmasta tuottaa esimerkiksi syrjiviä päätöksiä. Aineiston tasapainoon ja opetusaineiston edustavuuteen populaatiossa on syytä kiinnittää erityistä huomiota.

Datan laadulla ja määrällä on iso merkitys koneoppimismallien suoriutumiseen. Tämän opinnäytetyön käytännön osuuden esimerkki antaa osviittaa siitä, että datan analysointi ja esikäsittely algoritmeille sopivaan muotoon ovat koneoppimisprojekteissa todennäköisesti kaikkein työläin ja eniten aikaa vievä vaihe. Vaikka tässä työssä käytetty valmiiksi luokiteltu aineisto oli laadullisesti hyvää, oman kokemukseni mukaan todellisuudessa datassa on usein virheellisiä ja puuttuvia arvoja, mutta myös ylimääräisiä tai puuttuvia muuttujia. Datan hyödyntämisen kannalta olisi tärkeää huolehtia sen laadusta jo datan syntyhetkellä.

Työn käytännön osuus osoittaa, kuinka koneoppimismalli voidaan luoda vain muutamalla koodirivillä. Samaan käyttötarkoitukseen voidaan käyttää useita eri algoritmeja. Niiden suoriutumista tehtävästä voidaan arvioida käyttämällä malliin opetusaineistosta eristettyä testiaineistoa ja erilaisia mallien suorituskykyä

kuvaavia mittareita. Soveltuvat mittarit valitaan tapauskohtaisesti. Käytännössä paras lopputulos saavutetaan kokeilemalla eri algoritmeja ja säätämällä niiden hyperparametreja sekä käyttämällä opetuksessa esimerkiksi ristiinvalidointia. Työ osoitti, että samaan tehtävään soveltuvien algoritmien välillä on merkittäviä eroja. Esimerkkitapauksessa päätöspuun, satunnaismetsän ja K-lähimmän naapurin käyttö tuottivat täysin virheettömät lopputulokset, kun Logistinen regressio, tukivektorikone ja Naïve Bayes eivät osoittautuneet tässä tapauksessa käyttötarkoitukseen soveltuviksi.

Opinnäytetyö täydensi aiemmin opintojen aikana koneoppimisesta kertynyttä osaamistani. Teoreettista tietopohjaa kirjoittaessa suurin haaste ilmeni suomenkielisen termistön käytössä. Alan kirjallisuus ja muut lähteet ovat lähes täysin englanninkielisiä, eikä termeille vaikuta olevan vakiintuneita suomenkielisiä käännöksiä. Käytetyt suomennukset perustuvat Kimmo Pietiläisen suomentaman Ethem Alpaydinin kirjan Koneoppiminen sanastoon sekä työssä yhtenä lähteenä käytetyn Heidi Kanasen ja Harri Puolitaipaleen kirjassa Tekoäly, bisneksen uudet työkalut käytettyyn termistöön. Toivon työn toimivan muille aiheesta kiinnostuneille johdatuksena koneoppimisen maailmaan.

Lähteet

1 Amount of data created, consumed, and stored 2010-2020, with forecasts to 2025. 2022. Verkkoaineisto. Statista Gmbh.<https://www.statista.com/statistics/871513/worldwide-data-created/>. 8.9.2022. Luettu 26.9.2022.

2 Mitä tekoäly on ja mihin sitä käytetään? 2020. Verkkoaineisto. Euroopan parlamentti. <https://www.europarl.europa.eu/news/fi/headlines/society/20200827STO85804/mita-tekoaly-on-ja-mihin-sita-kaytetaan>. 4.9.2020. Luettu 21.8.2022.

3 Alepa ottaa ruoan robottikuljetukset käyttöön ensimmäisenä Suomessa. 2022. Verkkoaineisto. HOK-Elanto. < https://hok-elanto.fi/news/alepa-ottaa-ruoan-robottikuljetukset-kayttoon-ensimmaisena-suomessa/> 12.4.2022. Luettu 21.8.2022.

4 Brown, Sara. 2021. Machine learning, explained. Verkkoaineisto. <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>. 21.4.2021. Luettu 21.8.2022.

5 VanderPlas, Jake. 2016. Python Data Science Handbook. Sebastopol: O'Reilly Media.

6 Babu, Alan Davis. Artificial Intelligence vs Machine Learning vs Deep Learning (AI vs ML vs DL). 2019. Verkkoaineisto. <https://medium.com/@alanb_73111/artificial-intelligence-vs-machine-learning-vs-deep-learning-ai-vs-ml-vs-dl-e6afb7177436 >. 5.11.2019. Luettu 21.8.2022.

7 Maasalo, Paulus. 2021. Mitkä ovat tekoälyn, koneoppimisen ja algoritmien väliset erot? Verkkoaineisto. <https://www.quinyx.com/fi/blogi/mitka-ovat-tekoalyn-koneoppimisen-ja-algoritmien-valiset-erot>. 6.10.2022. Luettu 21.8.2022.

8 Behl, Anmol. 2019. An Introduction to Machine Learning. Verkkoaineisto. <https://becominghuman.ai/an-introduction-to-machine-learning-33a1b5d3a560>. 27.7.2019. Luettu 21.8.2022.

9 Siukonen, Timo. 2019- Mitä tulisi tietää tekoälystä. Jyväskylä: Docendo.

10 Kathleen, Walch. 2019. Rethinking Weak Vs. Strong AI. Verkkoaineisto. <https://www.forbes.com/sites/cognitiveworld/2019/10/04/rethinking-weak-vs-strong-ai>. 4.10.2019. Luettu 27.8.2022.

11 Bonaccorso, Giuseppe. 2018. Machine learning algorithms: Popular algorithms for data science and machine learning. Birmingham: Packt Publishing.

12 Brownlee, Jason. 2020. Difference Between Algorithm and Model in Machine Learning. Verkkoaineisto. <https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/>. 29.4.2020. Luettu 27.8.2022.

13 Satavisa, Pati. 2021. The difference between Artifical Intelligence and Machine Learning. Verkkoaineisto. <https://www.analyticsinsight.net/the-difference-between-artificial-intelligence-and-machine-learning/>. 3.8.2021. Luettu 21.8.2022.

14 Sakarkar, Gopal. Patil, Gaurav. Dutta, Prateek. 2021. Machine Learning Algorithms Using Python Programming. New York: Nova Science publishers.

15 Kelleher, John D. Tierney, Brendan. 2021. Datatiede. Helsinki: Terra Cognita.

16 Kananen, Heidi. Puolitaival, Harri. 2019. Tekoäly. Bisneksen uudet työkalut. Helsinki: Alma Talent.

17 Taanila, Aki. 2022. Akin menetelmäblogi - Koneoppimisen käsitteitä. Verkkoaineisto. <https://tilastoapu.wordpress.com/2019/08/04/koneoppimisen-kasitteita/>. Päivitetty 25.8.2022. Luettu 28.8.2022.

18 JavaTpoint. Regression Analysis in Machine learning. Verkkoaineisto. <https://www.javatpoint.com/regression-analysis-in-machine-learning>. Luettu 1.9.2022.

19 TechClass Ltd. 2022. Fundamentals of Machine Learning. Kurssimateriaali.

20 Yiu, Tony. 2019. The Curse of Dimensionality. Why High Dimensional Data Can Be So Troublesome. Verkkoaineisto. <https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e>. 20.7.2019. Luettu 2.9.2022.

21 Barla, Nilesh. 2022. Dimensionality Reduction for Machine Learning. Verkkoaineisto. <https://neptune.ai/blog/dimensionality-reduction>. Päivitetty 22.7.2022. Luettu 2.9.2022.

22 Bhatt, Shweta. 2018. Reinforcement Learning 101. Learn the essentials of Reinforcement Learning!. Verkkoaineisto. <https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292>. 19.3.2018. Luettu 3.9.2022.

23 Google. 2022. Data Preparation and Feature Engineering in ML. Verkkoaineisto. <https://developers.google.com/machine-learning/data-prep/construct/collect/data-size-quality>. Päivitetty 18.7.2022. Luettu 11.9.2022.

24 Himberg, Johan. 2020. Harhoja. Verkkoaineisto. <https://medium.com/@ingaskrap/harhoja-ecec2bce41ea>.15.4.2020. Luettu 11.9.2022.

25 Nyuytiymbiy, Kizito. 2020. Parameters and Hyperparameters in Machine Learning and Deep Learning. Verkkoaineisto. <https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac>. 30.11.2020. Luettu 11.9.2022.

26 Singh, Seema. 2018. Understanding the Bias-Variance Tradeoff. Verkkoaineisto. <https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229>. 21.5.2018. Luettu 12.9.2022.

27 Singh, Amarpreet. 2020. Difference Between ML Algorithm and Model. Verkkoaineisto. <https://medium.com/brandlitic/difference-between-ml-algorithm-and-model-801a798a6dc0>. 17.6.2020. Luettu 27.8.2022.

28 Gupta, Prashant. 2017. Cross-Validation in Machine Learning. Verkkoaineisto. <https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f>. 6.5.2017. Luettu 13.9.2022.

29 Gollapudi, Sunila. 2016. Practical Machine Learning. E-kirja. Packt Publishing.

30 Mishra, Aditya. 2018. Metrics to Evaluate your Machine Learning Algorithm. Verkkoaineisto. <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>. 24.2.2018. Luettu 10.9.2022.

31 Data School. 2014. Simple guide to confusion matrix terminology. Verkkoaineisto. <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>. Maaliskuu 2014. Luettu 10.9.2022.

32 Brownlee, Jason. 2019. A Gentle Introduction to Model Selection for Machine Learning. Verkkoaineisto. < https://machinelearningmastery.com/a-gentle-introduction-to-model-selection-for-machine-learning/>. 2.11.2019. Luettu 13.9.2022.

33 Håkansson, Anne & Hartung, Ronald Lee. 2020. Artifical Intelligence. Concepts, Areas, Techniques and Applications. Lund: Studentlitteratur Ab.

34 Perrotta, Paolo. 2020. Programming Machine Learning. From Coding to Deep Learning. Raleigh: The Pragmatic Programmers.

35 Tyagi, Neelam. 2020. Understanding the Gini Index and Information Gain in Decision Trees. Verkkoaineisto. <https://medium.com/analytics-steps/understanding-the-gini-index-and-information-gain-in-decision-trees-ab4720518ba8>. 24.3.2020. Luettu 18.9.2022.

36 Wikipedia. Kaggle. Verkkoaineisto. <https://en.wikipedia.org/wiki/Kaggle>. Luettu 24.9.2022.

37 Kestilä-Kekkonen, Elina. Kovarianssi ja korrelaatio. Teoksessa Kvantitatiivisen tutkimuksen verkkokäsikirja. Verkkoaineisto. Yhteiskuntatieteellinen tietoarkisto. <https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvanti/korrelaatio/korrelaatio/>. Luettu 24.9.2022.

38 Kaakinen, Markus & Ellonen Noora. Regressiomallin arviointi. Teoksessa Kvantitatiivisen tutkimuksen verkkokäsikirja. Verkkoaineisto. Yhteiskuntatieteellinen tietoarkisto. <https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvanti/regressio/arviointi/>. Luettu 24.9.2022.

Liitteet

Työssä käytetty Python-koodi kokonaisuudessaan:

```
# 1 - DATAN LUKEMINEN JA ANALYSOINTI
# Luetaan sensoridata dataframeen

import pandas as pd
df = pd.read_csv("../data/smoke_detection_iot.csv")
print("Sarakkeet, niiden tyypit ja tyhjien arvojen määrät:\n")
print(df.info())
print("\nEsimerkkejä aineiston sisällöstä:\n")
print(df)
print("\nTilastollisia esimerkkejä aineistosta:\n")
print(df.describe())

# Poistetaan turhat muuttujat: Unnamed, UTC ja CNT
df.drop(['Unnamed: 0', 'UTC', 'CNT'], axis=1, inplace=True)

print("\nTyhjien arvojen määrät sarakkeittain:\n")
print(df.isnull().sum())
print("\nLuokkien tasapainoisuuden tarkastelu:\n")
X, y = df.iloc[:, 0:13], df['Fire Alarm']
print(df['Fire Alarm'].value_counts())

# Tarkastellaan muuttujien välisiä korrelaatioita
# korrelaatiomatriisin avulla
import matplotlib.pyplot as plt
import seaborn as sbn
plt.figure(figsize=(20,10))
sbn.heatmap(df.corr(),annot=True,cmap = 'Blues')
plt.show()

# Tarkastellaan muuttujia histogrammien ja boxplot-kaavioiden avulla
i=1
plt.figure(figsize = (20,40))
for col in df.columns:
    plt.subplot(14,2,i)
    plt.title(col)
    df[col].value_counts().plot(kind ='hist')
    plt.subplot(14,2,i+1)
    sns.boxplot(x=pd.to_numeric(df[col]), palette='Blues')
    i=i+2
plt.tight_layout()

# 2 - DATAN ESIKÄSITTELY

# Poikkeavien havaintojen poistaminen
# Poistetaan havainnot, jotka ovat yli 3 standardipoikkeaman
# päässä keskiarvosta
from scipy import stats

z_scores = abs(stats.zscore(df))
```

```
filter = (z_scores < 3).all(axis=1)
df_filtered = df[filter]

print("\nAineisto, josta poikkeavat havainnot on poistettu\n")
print(df_filtered)
print(df_filtered.shape)

# Tarkastellaan muuttujia histogrammien ja boxplot-kaavioiden avulla
i=1
plt.figure(figsize = (20,40))
for col in df_filtered.columns:
    plt.subplot(14,2,i)
    plt.title(col)
    df_filtered[col].value_counts().plot(kind ='hist')
    plt.subplot(14,2,i+1)
    sns.boxplot(x=pd.to_numeric(df_filtered[col]), palette='Blues')
    i=i+2
plt.tight_layout()

# Muuttujien arvojen normalisointi välillä [0, 1]
from sklearn.preprocessing import MinMaxScaler
min_max=MinMaxScaler()
df_scaled=pd.DataFrame(min_max.fit_transform(df_filtered), /
columns=df_filtered.columns)
print("\nNormalisoidut muuttujat\n")
print(df_scaled.describe())

# Jako opetus- ja testiaineistoon
from sklearn.model_selection import train_test_split
# eriytetään kohdemuuttja aineistosta
X = df_scaled.drop(columns='Fire Alarm')
y = df_scaled['Fire Alarm']
# jaetaan aineisto 80 % opetukseen, 20 % testiin
x_train, x_test, y_train, y_test = train_test_split(X,y, /
test_size= 0.2, random_state= 18, stratify= y)

# käsitellään luokkien epätasapainoisuus
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
# summarize class distribution
print("\nLuokkien epätasapaino opetusaineistossa:")
print(Counter(y))
# define oversampling strategy
oversample = RandomOverSampler(sampling_strategy='minority')
# fit and apply the transform
X_over, y_over = oversample.fit_resample(X, y)
# summarize class distribution
print("\nLuokat tasapainotettuina opetusaineistossa:")
print(Counter(y_over))

# 3 - MALLIEN RAKENTAMINEN

titles, models, predictions = [], [], []

# LOGISTINEN REGRESSIO
# Mallin luonti
print("Logistinen regressio, mallin luonti...")
from sklearn.linear_model import LogisticRegression
model_lr = LogisticRegression(max_iter=200)
```

```
model_lr.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_lr = model_lr.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Logistinen regressio")
models.append(model_lr)
predictions.append(prediction_lr)
print("...valmis!")

# PÄÄTÖSPUU
# Mallin luonti
print("Päätöspuu, mallin luonti...")
from sklearn.tree import DecisionTreeClassifier
model_tree = DecisionTreeClassifier()
model_tree.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_tree = model_tree.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Päätöspuu")
models.append(model_tree)
predictions.append(prediction_tree)
print("...valmis!")

# SATUNNAISMETSÄ
# Mallin luonti
print("Satunnaismetsä, mallin luonti...")
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier()
model_rf.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_rf = model_rf.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Satunnaismetsä")
models.append(model_rf)
predictions.append(prediction_rf)
print("...valmis!")

# TUKIVEKTORIKONE
# Mallin luonti
print("Tukivektorikone, mallin luonti...")
from sklearn import svm
model_svm = svm.SVC(kernel='linear')
model_svm.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_svm = model_svm.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Tukivektorikone")
models.append(model_svm)
predictions.append(prediction_svm)
print("...valmis!")

# K-LÄHINTÄ NAAPURIA
# Mallin luonti
print("K-lähintä naapuria, mallin luonti...")
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=5)
model_knn.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_knn = model_knn.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
```

```
titles.append("K-lähintä naapuria")
models.append(model_knn)
predictions.append(prediction_knn)
print("...valmis!")

# NAÏVE BAYES
# Mallin luonti
print("Naïve Bayes, mallin luonti...")
from sklearn.naive_bayes import GaussianNB
model_nb = GaussianNB()
model_nb.fit(X_over, y_over)
# Mallin käyttö ennen näkemättömään testiaineistoon
prediction_nb = model_nb.predict(x_test)
# Tulokset talteen suorituskyvyn arviointia varten
titles.append("Naïve Bayes")
models.append(model_nb)
predictions.append(prediction_nb)
print("...valmis!")

# 4 – SUORITUSKYVYN ARVIOINTI

from sklearn.metrics import precision_recall_fscore_support as prf
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Mallien suorituskyvyn mittarit
accuracy, precision, recall, fscore = [0]*6, [0]*6, [0]*6, [0]*6

for i in range(6):
    print("\n" + titles[i] + ", tulokset:")
    accuracy[i] = models[i].score(x_test, y_test)
    precision[i], recall[i], fscore[i], _ = prf(y_test, /
    predictions[i], aerage='binary')
    print("Tarkkuus (accuracy) = {:.4f}".format(accuracy[i]))
    print("Sisäinen tarkkuus (precision) = {:.4f}"/
    .format(precision[i]))
    print("Herkkyys (recall) = {:.4f}".format(recall[i]))
    print("F1 = {:.4f}".format(fscore[i]))
    # Sekaannusmatriisi (confusion matrix)
    cm = confusion_matrix(y_test, predictions[i], /
    labels=models[i].classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, /
    display_labels=models[i].classes_)
    disp.plot(cmap=plt.cm.Blues)
    disp.ax_.set_title("Sekaannusmatriisi, " + titles[i])
    disp.ax_.set_ylabel("Todelliset")
    disp.ax_.set_xlabel("Ennustetut")
    plt.show()
```