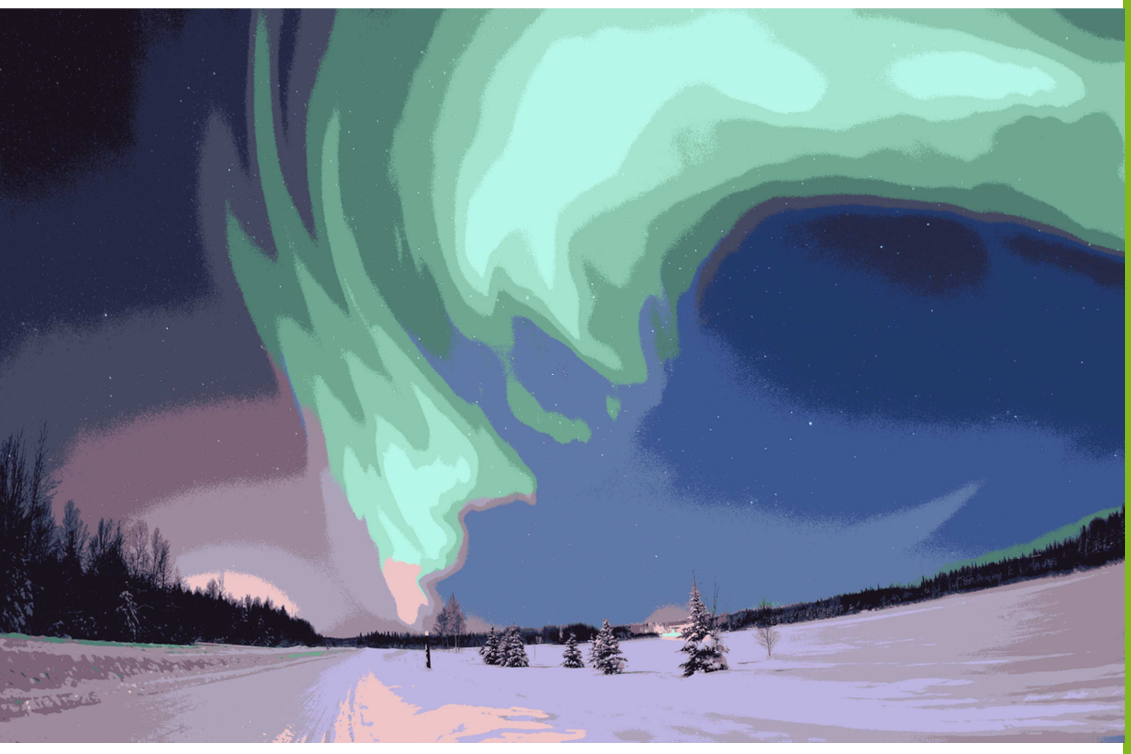
Antti Siipola

Koneoppimisen kehitysympäristöt



Insinööri (ylempi AMK)

Teknologiaosaamisen johtamisen koulutusohjelma

Syksy 2017

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tiivistelmä

Tekijä: Siipola Antti

Työn nimi: Koneoppimisen kehitysympäristöt

Tutkintonimike: Insinööri (ylempi AMK), teknologiaosaamisen johtaminen

Asiasanat: koneoppiminen, tekoäly, pilvipalvelut, lisenssit, avoin lähdekoodi

Tässä opinnäytetyössä lähdettiin selvittämään eri tekoälyalustojen soveltuvuutta Bittiumin tarpeisiin, taustalla Bittiumin 2016 aloittamat panostukset tekoälyosaamisen kasvattamiseen. Tavoitteena oli ymmärtää markkinoiden tarjontaa, minkä tyyppisiä, kuinka kypsiä ja millä ehdoilla hyödynnettäviä eri teknologiat ovat. Miten eri teknologioita voisi vertailla keskenään?

Tutkimuksen tekeminen aloitettiin miettimällä käyttötapauksia jotka olisivat Bittiumin kannalta oleellisimpia. Käyttötapauksia tunnistettiin runsaasti ja ne organisoitiin toimintaympäristön ja tiedonkäsittelyn vaatimusten perusteella. Tämän pohjalta lähdettiin valitsemaan analysoitavia teknologioita.

Valituista teknologioista kerättiin tietoja ja analysoitiin niiden soveltuvuutta käyttötapauksiin. Analyysin tuloksista tehtiin väliraportti Bittiumin tekoäly-koulutusohjelman jäsenille. Jo tässä vaiheessa oli nähtävissä, että markkinoilla on hyvin runsaasti tarjontaa ja myös selvästi kaupalliseen käyttöön kelpaavia järjestelmiä. Alustat myös jakautuivat eritasoisiin pilvipalveluihin ja kirjastoihin.

Lopuksi tehtiin käytännönkokeiluita kolmella eri alustalla: Microsoft Azure Machine Learning, Scikit-learn ja Google TensorFlow sekä yhdellä käyttötapauksella. Alustat valikoitiin analysoidusta joukosta edustamaan eri abstraktiotasoja. Tarkoitus oli saada kokemusperäistä tietoa alustan käyttöönotosta, käytettävyydestä ja rajoitteista kehittäjän kannalta. Käyttötapauksen toteutuksena käytettiin valmista datasarjaa, ”A Public Domain Dataset for Human Activity Recognition Using Smartphones” (Anguita;Ghio;Oneto;Parra;& Reyes-Ortiz, 2013), joka tarjosi sensoridataa käyttäjän liikkeistä erilaisissa aktiviteeteissa.

Tutkimustuloksien perusteella voitiin havaita, että Microsoftin Azure ML tarjosi helpoimmin lähestyttävän alustan, jolla sai tuloksia aikaan hyvin nopeasti. Azuren ja muiden pilvipalveluiden suurin ongelma liittyy datan kontrollin menettämiseen, läheskään kaikkea dataa ei haluta tai voida ladata pilveen. Googlen TensorFlow osoittautui hankalimmaksi käyttää ja ymmärtää, mutta toisaalta tarjoaa eniten vapauksia kehittäjälle sekä tuen hajautetulle laskennalle sekä kovokiihdyttimille. TensorFlowta pystyy ajamaan myös pilvipalvelussa ja toisaalta halutessaan myös sulautetussa järjestelmässä. Scikit-learn asettui näiden väliin tarjoten miellyttävän ympäristön opiskeluun ja nopeiden kokeilujen tekoon valmiiden mallien myötä, ilman datan siirtämistä pilveen.

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Abstract

Author: Siipola Antti

Title of the Publication: Machine Learning Development Environments

Degree Title: Master of Engineering, Technology Competence Management

Keywords: machine learning, artificial intelligence, cloud services, licensing, open source code

This thesis started from a need to evaluate different machine learning platforms and their suitability for Bittium's business. Bittium started to invest in artificial intelligence (AI) related competencies in 2016. Objective was to understand the market offering, what kind of platforms are available, how mature they are technically and on what terms they are available for commercial use. How to compare different machine learning platforms?

Research was started with listing most relevant use cases from Bittium's business point of view. Several use cases were identified and they were organized by their environment and by their requirements for data processing. With this understanding of the business needs, platforms for analysis were chosen.

Information was collected on the chosen platforms and their suitability for previously discovered use cases was evaluated. An interim report was created on the analysis for the participants of Bittium's AI training. Already in this phase it was seen that there is a lot of players on the market and also commercially usable solutions exist. Also, it was apparent that the studied platforms would form emergent categories of AI services in cloud and AI libraries.

Finally three platforms were chosen for empirical trials: Microsoft Azure Machine Learning, Scikit-learn and Google TensorFlow. The three platforms were chosen to represent different abstraction levels. The purpose of the trials was to gain experience on deployment, usability and possible limitation of these platforms from developer point of view. One sensory-data oriented use case was selected and to implement the use case, an existing dataset was used, "A Public Domain Dataset for Human Activity Recognition Using Smartphones" (Anguita;Ghio;Oneto;Parra;& Reyes-Ortiz, 2013).

Trials found that Microsoft Azure ML was easy to approach that allowed to get results quickly. Azure and other cloud services are put in disadvantage by the loss of control over the data a in cloud, not all data can be or is even allowed to a cloud service. Google TensorFlow was found to have the steepest learning curve, but it was also most flexible and has support for distributed computing and hardware accelerators. One can deploy TensorFlow in a cloud service or run it in an embedded system. Scikit-learn was in between, making it a good environment for studying and prototyping with ready-made AI models while all data can be stored locally.

Sisällys

Käsitteiden määrittely

Käsite	Määrittely
AGI	Artificial General Intelligence, keinotekoinen yleinen älykkyys, komplementtina kapealle älykkyydelle (esim. kasvojen tunnistaminen).
AI	Artificial Intelligence, keinoäly
AMI	Amazon Machine Image, valmiita virtuaalikoneaihioita, joista voi muodostaa virtuaalikoneita Amazon EC2:een.
Android	Googlen älypuhelinkäyttöjärjestelmä
API	Application Programming Interface, ohjelmointirajapinta. Ohjelmointirajapinta tarjoaa tarkkaan määritellyn tavan käyttää ohjelmiston / kirjaston / palvelun tarjoamia toimintoja ilman, että käyttävän ohjelmiston tarvitsee tietää toteutuksesta tai olla riippuvainen sen toteutustavasta ohjelmointirajapintaa lukuun ottamatta.
Asiantuntijajärjestelmä	Expert system, yksi keinoälyjärjestelmien alatyypeistä, kulta-aika sijoittui 80- ja 90-luvuille.
Corpus	Lähdemateriaali, jota IBM Watson käyttää vastauksiensa pohjana.
CPU	Central Processing Unit, yleisprosessori (tässä dokumentissa).
CUDA	Nvidian kehittämä rinnakkaislaskenta-alusta ja ohjelmointirajapinta joka mahdollistaa GPU:den käytön yleiseen laskentaan, erotuksena pelkkään grafiikka-kiihdytykseen.

DDoS	Distributed Denial of Service, hajautettu palvelunestohyökkäys.
DNN, Deep learning	Deep Neural Network, syväoppiminen.
EC2	Amazon Elastic Compute Cloud, Amazonin IaaS palvelu.
GPU	Graphics Processing Unit, erityisesti grafiikka-algoritmien kiihdyttämiseen suunniteltu prosessorityyppi.
IaaS	Infrastucture as a Service, infrastruktuuri palveluna.
Ingestion	”Sulattelu”, Watsonin käyttämä nimitys Corpuksen indeksoinnille ja metadatan luomiselle.
IoT	Internet of Things, asioiden internet.
IPA	Intelligent Personal Assistant, älykäs henkilökohtainen avustaja.
Klusterointi	Clustering, ryvästys, osajoukkojen tunnistaminen suuremman (näennäisen) satunnaiselta näyttävän joukon sisältä.
LTE	Long Term Evolution, matkapuhelinverkon neljännen sukupolven teknologioista käytetty yleisnimitys.
LTU	Linear threshold unit, Perceptronin (osa MLP:tä) käyttämä neuronityyppi.
Luokiteltu joukko	Labeling, datan merkitseminen niin, että tiedetään mihin joukkoon se kuuluu. Tällaista luokiteltua datasarjaa voidaan käyttää algoritmin opettamiseen ja testaamiseen.
Luokittelu	Classification, algoritmikategoria.

MLP	Multi-layer Perceptron, neuroverkkotyyppi
ML	Machine Learning, koneoppiminen
Noise	Häiriötä tai kohinaa ilmenee datassa ja se laskee osaltaan datan luotettavuutta ja käyttökelpoisuutta esim. koneoppimisalgoritmin opettamiseen.
Ohjaamaton oppiminen	Unsupervised learning
Ohjattu oppiminen	Supervised learning
PaaS	Platform as a Service, toimintaympäristö -palveluna
Pattern	Malli, kuvio.
SaaS	Software as a Service, ohjelmisto palveluna
Skaalautuvuus	Järjestelmän kyky käsitellä kasvava määrä työtä, mahdollisuus laajentaa järjestelmää suuremmalle määrälle työtä.
SVM	Support Vector Machine, tukivektorikone. Mm. luokitteluun käytetty koneoppimisalgoritmien kategoria.
SVR	Support Vector Regression, tukivektoriregressio, SVM:ään perustuva regressioalgoritmi.
TPU	Tensor Processing Unit, erityisesti Google TensorFlow'n tensoreiden laskennan nopeuttamiseen suunniteltu yksikkö.
Vahvistettu oppiminen	Reinforcement learning
Web-service	SaaS palvelu, jota tarjotaan HTTP (Hyper Text Transfer Protocol) -protokollan yli.
Yleistäminen	Generalization, opetettu malli toimii hyvin myös

	sellaisella datalla, jota se ei ole nähnyt. Liittyy ylisovittamiseen, jossa malli toimii vain opetusmateriaalilla, mutta heikosti muulla datalla (Buduma, 2017).

1 Johdanto

Bittium Oyj (HEL: BITTI) tunnettiin 1.7.2015 asti nimellä Elektrobit Oyj. Nimi muuttui samalla kun Elektrobit myi Automotive-liiketoiminnan saksalaiselle Continental AG:lle. Yhtiö perustettiin vuonna 1985 ja sen pääkonttori on Oulussa, toimistoja löytyy Suomessa myös Kajaanista, Tampereelta, Espoosta ja Kuopiosta. Yhtiöllä on toimistot tämän lisäksi Singaporessa sekä USA:ssa Restonissa ja Bothellissa.

Bittiumin perusliiketoimintaa on tarjota asiakkailleen korkean teknologian tuotteita ja tuotealustoihin perustuvia ratkaisuja sekä tuotekehityspalveluita. Viime vuosina tuotetarjontaan on lisätty myös tietoturvaratkaisut (Safemove VPN) ja 10.11.2016 lähtien myös terveydenhuollon teknologian tuotteita ja palveluita (Mega Elektroniikka Oy).

Bittiumin tuotteet ja tuotealustat pyrkivät vastaamaan erityisesti puolustus- turvallisuus ja muiden viranomaismarkkinoiden tarpeisiin. Teollisuussovelluksia ja IoT (Internet of Things) kehityspalveluita tarjotaan laajalle joukolle asiakkaita. Bittiumilla on pitkä historia langattoman viestinnän parissa ja tuotekehityspalveluita onkin saatavissa lähes mihin tahansa missä on jonkinlainen antenni.

Bittiumin tuotteita: Bittium Tough Mobile –älypuhelin, Bittium Tactical Wireless IP Network –järjestelmä, Bittium Tough VoIP –tuotteet, Bittium Tactical LTE access Point –liityntäratkaisu. Alustoja: Bittium Special Device Platform Android pohjaisiin päätelaitteisiin ja Bittium IoT Device Platform langattomilla yhteyksillä ja erilaisilla sensoreilla varustettujen tuotteiden kehittämiseen. Bittium Safemove VPN mahdollistaa turvalliset ja saumattomat yhteydet yrityksen tietoverkkoon, erityisesti liikkuville käyttäjille.

1.1 Keinoälyn hyödyntäminen

Vuonna 2016 Bittiumilla käynnistettiin ’AI taskforce’ jonka tarkoituksena on kehittää AI (Artificial Intelligence) -osaamista ja liiketoimintaa. Ensimmäisinä toimina on ollut palkata yritykseen AI-osaajia ja kehittää henkilöstön osaamista räätälöidyllä koulutuksella, joka käynnistyi 2016 loppuvuodesta. Myös ensimmäisiä bisnestarpeita kartoitettiin ja valmisteltiin demonstraatiota keinoälystä. Tarkoituksena on jatkossa tunnistaa lähitulevaisuuden AI-projekteja ja kehittää asiakashankkeita, jossa ko. teknologioita hyödynnetään. Näiden pohjalta voidaan jatkaa yrityksen strategista suunnittelua AI-sovellusalueella.

Teollisuuden näkökulmasta AI ei ole niin suuren ’hypen’ kohteena mitä sen arvo voisi liiketoimintana olla. Odotetaankin, että siitä voisi kehittyä merkittävää liiketoimintaa kymmenen vuoden sisällä.

Tämän työn tarkoituksena on olla osaltaan selkiyttämässä Bittiumin käsitystä AI:sta hyödynnettävänä teknologiana, mitä mahdollisuuksia se tarjoaa ja mitä vaihtoehtoja markkinoilla on tarjolla. Mitä teknologioita Bittiumilla on syytä tuntea, jotta voidaan olla vakavasti otettava toimija? Seuraavassa kappaleessa käydään tarkemmin läpi yrityksen vaatimuksia kehitystehtävälle. Kehitystehtävää voi pitää onnistuneena, jos se pystyy

- tuomaan uutta ymmärrystä AI-tehtäväryhmän käyttöön
- luomaan keinoja AI-teknologian valintaan johonkin tiettyyn sovellukseen.

1.2 Ongelmakenttä

Osana AI –osaamisen kehittämistä ja ensimmäisiä hankkeita Bittiumilla täytyy selvittää eri AI-alustojen ominaisuuksia ja soveltuvuutta Bittiumilla tunnistettuihin ja todennäköisiin liiketoimintamahdollisuuksiin. Mahdollisia selvitettäviä asioita listattiin loppuvuodesta 2016:

- mitä alustoja on tarjolla
- kenen tekeminä (yritykset, yhteisöt)
- alustojen tekninen kypsyys
- lisensointiehdot ja hinnoitteluperusteet
- sovelluskohteiden mahdolliset rajaukset kuten:
    - akkukäyttöiset laitteet vs. verkkovirtaan kytketyt
    - jatkuvasti tietoverkossa olevat vs. Internetistä erotetut laitteet
    - tietomassan hallinta – tarvitaanko klusterilaskentaa vai tehdäänkö esim. kannettavassa laitteessa.

Tekoäly ja koneoppiminen ovat Big Datan ohella hyvin suosittuja aiheita tietojenkäsittelyssä. Erilaisia tekniikoita ja ratkaisuja on paljon. Yritykset ovat viime vuosina kaupallis-

taneet eri tekoälytekniikoita ja tarjoavat niitä erilaisina palveluina ja ratkaisuina – ei ole itsestään selvää mitä nämä oikeasti pystyvät tekemään ja mitä rajoituksia niihin liittyy. Lisäksi on erilaisia OpenSource -kirjastoja jotka pintapuolisesti näyttäisivät tekevän samoja asioita – mitkä näiden tosiasialliset erot ovat kun ollaan tekemässä sovellusta johonkin teollisuuden tarpeeseen? Onko yrityksellä tarvetta tai hyötyä käyttää syväoppimista?

Nopeasti kehittyvällä alueella ratkaisuja on paljon ja oikeat kysymykset (vaatimukset järjestelmälle) ovat paljon arvokkaampia kuin vastaukset. Teknologioita on myös runsaasti ja aikaa rajallisesti, joten työtä tehdessä työn laajuutta rajattiin jatkuvasti ja toisaalta sitä ohjattiin yrityksen tarpeita kohti.

2 Teoria

Keinoäly- ja koneoppimisympäristöjen vertailun kannalta oleellisimpia teorioita ovat itse keinoälyn ja koneoppimisen tekniikat, mitä eri teknologioilla voidaan tehdä ja mitä harkittavien alustojen olisi syytä tukea. On myös joukko teknologioita, joita tarvitaan tukemaan alustoja ja ympäristöjä, kuten erilaiset tavat tuottaa palveluita verkossa ja skaalata laskentaa tarpeen mukaan. Näiden teoriaa käydään myös läpi vaikka ne eivät olekaan spesifisiä keinoälyn rakentamiseen.

Hieman eri alueen teoriaa sivutaan vielä kappaleessa 2.4, jossa käydään läpi ohjelmistojen hankintaa. Tämän tarkoituksena on luoda valmiuksia arvioida ratkaisun mielekkyyttä kaupallisessa käytössä.

Käsitteenä älykkyys ymmärretään tässä rationaalisten päätösten tekemiseksi, älykäs agentti valitsee aina parhaan mahdollisen toimen joka sopii tilanteeseen tai ongelman ratkaisuun (Russell & Norvig, 2016, s. 30).

2.1 Keinoälyn lyhyt historia

Koneoppimista ja keinoälyä on tutkittu ainakin 1950-luvulta, ensimmäiset julkaisut löytyvät näiltä ajoilta. Ensimmäiset maininnat keinoälystä menevät kuitenkin aina antiikin tarinoihin tietoisista, keinotekoisista olioista.

Hallitukset rahoittivat keinoälyn tutkimusta vuosikymmeniä kunnes 1973 USA:n ja Britannian hallitukset lopettivat rahoituksen perustuen James Lighthillin (Lighthill, 1973) raporttiin. Raportin mukaan keinoälytutkimus oli epäonnistunut tuottamaan mitään merkittävää vaikutusta – keinoälytutkimuksen monimutkaisuus oli aliarvioitu. Keinoälytutkimuksen rahoitus onkin ollut vaihtelevaa ja tuloksiin on petytty moneen kertaan myös tämän jälkeen.

1980-luvulla tehtiin ensimmäinen kaupallisesti menestynyt asiantuntijajärjestelmä, Digital Equipment Corporationin R1 (McDermott, 1982). Tämä oli käytössä DEC:illä itsellään sekä muutamilla muilla yrityksillä tuottaen rahallisia säästöjä.

Nykyään monia keinoälytutkimuksen tuloksia hyödynnetään rutiininomaisesti ja ne ovat siirtyneet tavallaan pois keinoäly-otsikon alta. Tiedonlouhinta (big data analyysi), robo-

tiikka, logistiikka, hakukoneet – teknologioita käytetään jatkuvasti mutta niitä ei pidetä keinoälynä. Tämä on leimallista keinoälytutkimukselle, maali siirtyy aina kun teknologiat kehittyvät ja niitä otetaan käyttöön, ”keinoäly” on siten aina 10 vuoden päässä tulevaisuudessa.

2000-luvulla tietokoneiden laskentateho sekä suurien datamäärien kerääminen ovat mahdollistaneet keinoälyteknologioiden käyttöönoton myös tutkimusprojektien ulkopuolella. Syväoppiminen yhtenä tällaisista teknologioista hyödyntää molempia, mutta hyötyy aivan erityisesti suurista tietovarannoista, joilla oppimista voidaan tehdä. Tämä pätee yleisimminkin oppiviin algoritmeihin, kuten vaikkapa puuttuvan datan korvaamisessa kuvista, jossa parilla tuhannella kuvalla opetettu algoritmi tekee huonon korvauksen mutta kahdella miljoonalla kuvalla opetettu tekee jo erinomaista jälkeä (James & Alexei, 2007). Mahdollisuus hyödyntää suuria datamääriä antaa kilpailuedun suurille yrityksille ja vielä erityisesti sellaisille, jotka toimivat maissa joissa yksityisyyden suoja ei ole vahva, kuten Yhdysvallat ja Kiina.

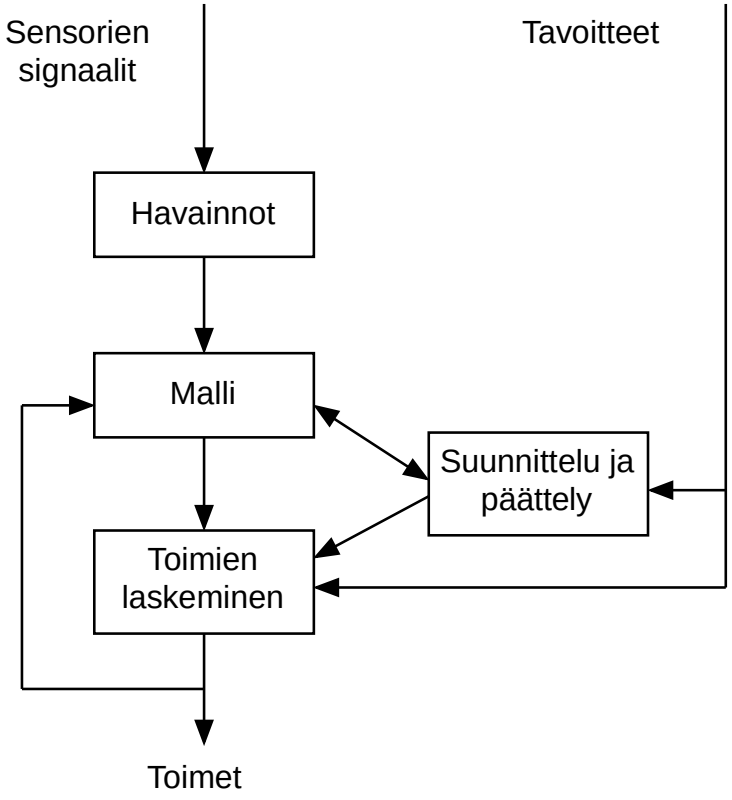
Viime vuosina keskustelua on käyty ”keinotekoisesta yleisestä älykkyydestä” ja tarpeesta varautua siihen. Sen sijaan, että kehitetään nykyisillä teknologioilla itsestään ajavia autoja ja parempaa puheentunnistusta eli hyvin tarkkaan rajattuja keinoälyalgoritmeja, osa tutkijoista haluaisi enemmän panostusta yleispätevän algoritmin tutkimiseen, joka oppisi ja toimisi missä hyvänsä ympäristössä. Jos ja kun tällainen onnistutaan tekemään, tulee sen hallitsemisesta keskeinen ongelma, eli miten voidaan huolehtia, että tällaisen keinotekoisen älykkyyden tavoitteet ovat ihmisten tavoitteiden kanssa yhtenevät (Russell & Norvig, 2016, s. 27).

2.2 Koneoppiminen

Keinoäly on terminä haastava – milloin kyse on näppärästä algoritmista ja milloin keinotekoisesta ”älykkyydestä”? Yksi määritelmä keinoälylle on kyky yli-inhimillisiin suorituksiin. Tällä määritelmällä jopa yksinkertainen taskulaskin voisi olla keinoäly – se laskee luvusta neliöjuuren nopeampaa kuin kukaan ihminen. Nilsson määrittelee keinoälyn toimijaksi, joka havainnoi ja mallintaa ympäristöään, laskee sopivat toimenpiteet ja mahdollisesti ennakoi niiden vaikutukset (Nilsson, 1998, s. 2). Kaavio tällaisen toimijan tai ”älykkään agentin” arkkitehtuurista esitetään kuvassa 1. Kaikki edellä mainitut toimet voivat olla myös koneoppimista:

> ”kone oppii kun se muokkaa rakennettaan, ohjelmistoaan tai dataansa ulkoisen syötteen perusteella niin, että sen suorituskyky paranee ajan funktiona” (Nilsson, 1998, s. 1).

tai ”agentti oppii jos se parantaa suorituskykyään seuraavilla kerroilla tehtyään havaintoja maailmasta” (Russell & Norvig, 2016, s. 693).



Kuva 1 Keinoälyjärjestelmä (mukaillen Nilsson, 1998)

Mihin tarvitaan koneoppimista? Koneoppiminen, verrattuna ennalta määriteltyyn logiikkaan, antaa seuraavia etuja:

- Jotain tehtäviä ei ole mielekästä määritellä muuten kuin esimerkkien kautta – voidaan määrittää syöte ja toivottu lopputulos, mutta ei mitään selkeää sääntöä näiden määräytymiselle toisistaan. Koneoppimisen, ja riittävän määrän esimerkkejä, avulla kone voi muokata toimintaansa niin, että se tuottaa enimmäkseen oikeita vastauksia.

- Tärkeitä yhteyksiä piilotettuna isoihin datamassoihin – tiedon louhinta.

- ML parantamaan teknisten ratkaisujen siirrettävyyttä sovelluksesta toiseen.

- Isojen tietomäärien koodaus ohjelmalogiikaksi on työlästä, koneoppiminen voi auttaa tässä siirtämällä työn koneen tehtäväksi.

- Ympäristöt muuttuvat, koneoppiminen auttaa adaptoitumaan ilman koneen uudelleensuunnittelua ihmisen toimesta.

- Uutta tietoa löytyy jatkuvasti ihmisten toimesta, kieli muuttuu. Jatkuva uudelleensuunnittelu on epäkäytännöllistä, joten koneoppiminen auttaa pysymään ajan tasalla.

(Nilsson, 1998)

Russell ja Norvig tunnistavat kolme kategoriaa:

- Järjestelmän suunnittelijat eivät voi ennakoida kaikkia mahdollisia tilanteita joihin agentti voi joutua. Robotin, joka suunnistaa sokkelossa täytyy selviytyä kaikista sokkeloista joita se kohtaa.

- Järjestelmän suunnittelijat eivät voi ennakoida miten asiat kehittyvät tulevaisuudessa, osakemarkkinoita ennustavan ohjelman täytyy mukautua eri markkinatilanteisiin, kun noususuhdanne vaihtuu laskusuhdanteeseen.

- Järjestelmän suunnittelijat eivät välttämättä osaa ohjelmoida ratkaisua itse. Ihmiset tunnistavat vaivatta perheenjäsentensä kasvot, mutta parhaatkaan ohjelmoijat eivät osaa ohjelmoida tietokonetta tekemään tätä ilman oppivia algoritmeja.

(Russell & Norvig, 2016, s. 693)

2.3 Koneoppimisen algoritmit

Algoritmi käsitteenä tarkoittaa kuvausta miten jokin asia tehdään tai miten jokin ongelma ratkaistaan. Esimerkiksi keittokirja voidaan ajatella olevan kokoelma algoritmeja, joiden avulla voi valmistaa erilaisia aterioita. Seuraavaksi esitellään, ikään kuin keittokirjassa, erilaisia koneoppimisen algoritmeja joiden avulla, ja joita yhdistelemällä, voidaan suorittaa monenlaisia tietojenkäsittelyn tehtäviä.

Koneoppimisen voi jakaa karkeasti seuraaviin kategorioihin sen perusteella miten algoritmit saavat palautetta oppimisestaan (Russell & Norvig, 2016, s. 694):

- Ohjaamaton oppiminen, opitaan malleja ilman suoraa palautetta.
- Vahvistusoppiminen, saadaan positiivista ja negatiivista palautetta oppimisesta tulosten perusteella.
- Ohjattu oppiminen, tarjotaan algoritmille valmiita kysymys-vastaus pareja ja oppimisen tuloksena algoritmi osaa yhdistää nämä keskenään.
- Puoliohjattu oppiminen, käytännössä eri kategoriat eivät ole näin selkeitä, käytössä voi olla muutamia luokiteltuja esimerkkejä ja tämän lisäksi suuri määrä luokittelematonta tietoa. Esimerkiksi iän tunnistamisesta valokuvista: aineistossa voi olla suuri määrä kuvia ja muutama, jossa kuva on yhdistetty ikään. Henkilö on voinut myös valehdella ikänsä, joten luokiteltuunkaan tietoon ei voi täysin luottaa. Tämä häiriö ja luokittelun puute muodostavat jatkumon ohjatun ja ohjaamattoman oppimisen välille.

Edellä mainittu jaottelu on vain yksi näkökulma koneoppimiseen eikä se aina muodosta selkeää jakolinjaa algoritmien välille, esimerkiksi neuroverkoilla voidaan toteuttaa kaikkia oppimismalleja. Seuraavissa kappaleissa käydään läpi kolme oppimisparadigmaa (ohjattu, ohjaamaton ja vahvistusoppiminen) ja muutamia niille ominaista koneoppimisalgoritmia.

2.3.1 Ohjattu oppiminen

Ohjatun oppimisen tehtävä voidaan kuvata seuraavasti (Russell & Norvig, 2016, s. 695):

> Opetusjoukko N koostuu syöte-vaste pareista
>
> $$(x_1, y_1), (x_2, y_2), \dots (x_N, y_N),$$
>
> jossa jokainen y_j on tuntemattoman funktion $y = f(x)$ tuottama ja on löydettävä funktio h joka tuottaa likimäärin saman tuloksen kuin todellinen funktio f.

Funktio h on hypoteesi ja ollakseen hyvä sen täytyy opetusjoukon lisäksi toimia myös datalla, joka ei kuulunut alkuperäiseen opetusjoukkoon. Hypoteesin hyvyyttä voidaan siis mitata testijoukolla ja sen sanotaan yleistävän hyvin, jos se ennustaa oikein y:n arvot myös opetusjoukkoon kuulumattomalla datalla.

Jos funktion vaste on jostain rajatusta joukosta arvoja, kyseessä on **luokittelutehtävä**. Jos arvoja on vain kaksi, puhutaan binäärisestä tai Boolean luokittelusta. Jos vaste on numero, oppimistehtävä on tyypiltään **regressio**. Luokittelua ja regressiota voidaan myös yhdistellä.

Regressiossa on pohjimmiltaan kyse käyrän sovittamisesta joukkoon numeerisia arvoja x ja y akseleilla, mitä parempi malli sitä parempia ennustuksia tuntemattomalla x:n arvolla saadaan. Yksinkertaisimmillaan puhutaan lineaarisesta regressiosta, jossa sovitetaan suora datan joukkoon. Edistyneempiä algoritmeja ovat mm. nearest neighbors, joka muodostaa käyrän perustuen seuraavien arvojen keskiarvoon. K-nearest neighbor regressio perustuu käyttäjän määrittämän kokonaisluvun k etäisyydellä olevien naapureiden arvoihin. Nearest neighbors regressiolla saadaan tarkempia ennusteita kuin lineaarisella regressiolla, se seuraa paikallisia piirteitä paremmin (Pedregosa, ym., 2011).

Tukivektoriregressio (SVR) perustuu tukivektorikoneeseen (SVM), jota käytetään myös luokitteluun. SVM:n idea on löytää taso, joka jakaa osajoukot optimaalisesti, eli sen etäisyys osajoukoista on mahdollisimman suuri. Tämä tapahtuu etsimällä tukivektoreita. Tukivektorit ovat pisteitä, jotka ovat osajoukkojen reunoilla. SVM on siten kiinnostunut vain osasta opetusjoukkoa, tukivektoreista. SVM algoritmeja on useita erilaisia: lineaarinen, polynominen, Gaussian RFB, jne. ja nämä ovat hyvin käyttökelpoisia pienille ja keskisuurille määrille dataa jossa ei ole paljon muuttujia (Géron, 2017).

Luokittelussa lajitellaan syöte kuulumaan johonkin rajattuun joukkoon arvoja, esimerkiksi tunnistetaan automerkkejä (luokittelu) sen sijaan, että ennustettaisiin niiden polttoaineen kulutusta painon, moottorin tilavuuden ja iän perusteella (regressio). Luokitteluun voidaan käyttää mm. tukivektorikoneita (SVM), Multi-layer Perceptron (MLP) neuroverkkoja sekä k-nearest neighbors algoritmia.

MLP koostuu syötekerroksesta, yhdestä tai useammasta linear threshold unitista (LTU), joita kutsutaan myös piilokerroksiksi ja tuloskerroksesta. Jos neuroverkossa on kaksi tai useampia piilokerroksia, sitä nimitetään syväksi neuroverkoksi (DNN) (Géron, 2017). Etuna MLP:llä luokittelussa on se, että se kykenee oppimaan ei-lineaarisia malleja ja sitä voidaan opettaa reaaliaikaisesti. Huonona puolena on opettamisen herkkyys alkuarvoille ja suuri määrä säädettäviä parametreja (neuronien ja kerrosten määrä, opetusiteraatioiden määrä) (Pedregosa, ym., 2011).

2.3.2 Vahvistusoppiminen

Vahvistusoppiminen on behaviorismista inspiraationsa saanut koneoppimisen ongelmanratkaisutekniikka. Tässä tekniikassa agentti tutkii ympäristöä ja toimii havainnon perusteella, pyrkien maksimoimaan positiivisen palautteen. Ongelmat, joissa tehdään sarja päätöksiä joiden lopputulosta ei voida tietää varmuudella voidaan formalisoida Markovin päätösongelmiksi. Markovin päätösprosessissa (Markov decision process, MDP) määritellään **tila**, joka kertoo agentin sen hetkisen tilanteen, **toimenpide**, joka vaikuttaa prosessiin ja **palkinto**, joka havainnoidaan jokaisen tilamuutoksen yhteydessä (Sigaud & Buffet, 2013). MDP tarjoaa kehyksen, jolla kuvata vahvistusoppimisen ongelmia.

Vahvistusoppiminen eroaa ohjatusta oppimisesta siinä, ettei oikeita kysymys-vastaus pareja käytetä eikä suboptimaalisia toimia erikseen korjata. Sen sijaan pyritään jatkuvaan parantamiseen, jossa haetaan kompromissia tutkimisen (mitä ei tunneta vielä) ja hyödyntämisen (olemassa oleva tieto) välillä (Sigaud & Buffet, 2013). Vahvistusoppiminen sopiikin erityisen hyvin ongelmiin, joissa täytyy tehdä kompromisseja lyhyen tähtäimen ja pitkän tähtäimen palkintojen tai saavutusten välillä. Hyviä sovelluskohteita ovat olleet robotiikka, hissinohjaus, telekommunikaatio, backgammon, tammi (Sutton & Barto, 1998). Skenaarioissa, joissa voidaan opetella yritys – erehdys menetelmällä vahvistusoppiminen on erityisen vahvoilla. Esimerkiksi edellä mainituissa peleissä voidaan tehdä simulointeja, jotka vastaavat varsinaista peliä, kun taas itseajavassa autossa vastaava toiminta ei onnistu.

Tämä koneoppimistekniikka on vahvasti riippuvainen hyvästä tutkimusmekanismista, jolla valitaan mitä agentti seuraavaksi tekee tavoitteeseen/palkintoon päästäkseen. Algoritmityyppejä hyöty-funktion opettamiseen (Russell & Norvig, 2016, ss. 830-853):

- Direct utility estimation
- Adaptive dynamic programming (ADP)
- Temporal-difference (TD)

2.3.3 Ohjaamaton oppiminen

Ohjaamaton oppiminen on koneoppimisen menetelmä, jossa muodostetaan malli, joka sopii havaintoihin. Se eroaa ohjatusta oppimisesta siten, että luokkia ei tunneta ennalta.

Erona ohjattuun oppimiseen aineistoa ei ole luokiteltu, eikä ole testiaineistoa jolla mallin voisi validoida, joten algoritmin tarkkuutta ei pystytä testaamaan suoraan (Pacheco, 2013). Alla muutamia tärkeimpiä ohjaamattoman oppimisen kategorioita ja algoritmeja:

Klusterointi, ryvästys, etsitään datasta jollain kriteerillä muodostuvia osajoukkoja. Ryvästysalgoritmeja on useita, esimerkiksi k-means, sekoitemallit ja hierarkinen klusterointi. K-means algoritmille annetaan parametrina klusterien lukumäärä ja se alkaa etsimään näitä kolmessa vaiheessa:

1. Valitaan alkupisteet, yksinkertaisimmillaan k näytettä tietojoukosta.
2. Jokainen näyte yhdistetään lähimpään pisteeseen.
3. Valitaan uudet pisteet laskemalla keskiarvot edellisiin pisteisiin yhdistetyistä näytteistä.

K-means iteroi vaiheita 2 ja 3 kunnes pisteet eivät enää liiku asetettua kynnysarvoa enempää (Pedregosa, ym., 2011). Tämän dokumentin kansikuvassa (Strang, 2012) alkuperäinen kuva revontulista on käsitelty k-means algoritmilla niin, että siinä on 16 väriä (klusterien lukumäärä yllä).

Anomalioiden tunnistaminen tarkoittaa poikkeamien tunnistamista suuresta joukosta normaalia toimintaa. Tämä voidaan tehdä esimerkiksi autoenkooderin avulla, mallintamalla normaali data/signaali, rakentamalla signaali uudestaan mallin pohjalta ja vertaamalla syntetisoitua ja oikeaa signaalia keskenään – jos ero ylittää raja-arvon on signaalissa jotain poikkeavaa (Dunning & Friedman, 2014). Muita algoritmeja joita tähän voidaan käyttää on Isolation Forest, EllipticEnvelope (etsii poikkeamia olettaen datan olevan Gaussin jakaumalla) ja LocalOutlierFactor (perustuu k-nearest neighbors algoritmiin, kuinka kaukana näyte on naapurien tihentymästä) (Pedregosa, ym., 2011).

Neuroverkot mahdollistavat myös ohjaamattoman oppimisen. Replicator Neural Network (RNN) onnistuu tunnistamaan anomalioita datasta ilman ennakko-oletuksia datan luonteesta (Hawkings;He;Williams;& Baxter, 2002). Restricted Boltzmann konetta (RBM) voidaan käyttää piirteiden poimintaan. Kun datasta tunnistetaan piirteet RBM:llä ja syötetään sen jälkeen SVM:lle, saadaan merkittävästi parempi tarkkuus kuin ilman RBM:ää (Pedregosa, ym., 2011).

2.4 Ohjelmistojen hankkiminen

Suuri osa hankinta-alan kirjallisuudesta keskittyy tilanteeseen, jossa hankkijana on jokin muu kuin teknologiaa kehittävä asiantuntijaorganisaatio (TIEKE, 2016). Luonnollisesti teknologiayritysten ulkopuolella tämä onkin normaali tilanne eikä voida olettaa, että yrityksellä olisi oma teknologian kehitysorganisaatio, korkeintaan jonkinasteinen kyvykkyys ostaa kehityspalveluita. Julkisella sektorilla tämä tarkoittaa monesti julkisen hankinnan tekemistä eli kun hankinnan rahallinen arvo ylittää laissa säädetyn rajan (taulukko 1), täytyy siitä järjestää julkinen kilpailutus, johon kaikki kriteerit täyttävät yritykset voivat osallistua yhdenvertaisesti.

Taulukko 1 Kansalliset kynnysarvot (Finlex, 2016)

Hankintalaji	Kynnysarvo (EUR)
Tavarahankinnoissa, palveluhankinnoissa ja suunnittelukilpailuissa	60000
Rakennusurakoissa	150000
Sosiaali- ja terveyspalveluja koskevissa hankinnoissa	400000
Erityisiä palveluja koskevissa hankinnoissa	300000
Käyttöoikeussopimuksissa	500000

Yksityisiä yrityksiä eivät koske samanlaiset lakisääteiset rajoitukset hankinnoissa, mutta sen sijaan kilpailu markkinoilla pakottaa yritykset tekemään järkeviä hankintoja. Riippuen tilanteesta teknologian ja markkinoiden suhteen, eri asiat painottuvat ja usein päätöksiä halutaan tehdä nopeasti – jos oma tuote viivästyy markkinoilta, kilpailijat voittavat markkinaosuutta. Voimakkaasti kilpailluilla globaaleilla markkinoilla, erityisesti Internetin kautta saavutettavissa palveluissa, ne saattavat viedä sen kokonaan.

Korkeaa teknologiaa kehittävällä organisaatiolla voi perustellusti olettaa olevan kokeneita henkilöitä, joilla on kertynyt syvällistä osaamista eri teknologioista ja siten kyky arvioida teknisiä ratkaisuja hyvinkin perusteellisesti. Tarvittaessa voidaan tehdä jotain osia itse, jos jokin teknologia nähdään yrityksen kannalta ydinosaamiseksi (Prahalad, 1999). Ratkaisevaa lopputuloksen kannalta on, kuinka hyvä tuote tai palvelu tästä hankinnasta yhdistettynä yrityksen itse tekemään osuuteen saadaan loppuasiakkaan kannalta. Hankinnan ja oman kehitystyön osuudet vaikuttavat myös siihen, minkälainen ansaintamalli tästä muodostuu yritykselle.

Hankinnan luonne muuttuu myös sen mukaan, minkä tyyppistä koneoppimistuotetta tai kirjastoa ollaan ottamassa käyttöön. Jos jo tuotekonsepti rajaa räätälöinnin mahdollisuu-

det vähäisiksi, lähestytään valmisohjelmiston hankkimista. Jos taas valintaa tehdään kirjastojen välillä, ollaan jo lähtökohtaisesti tekemässä ohjelmistoprojektia, jossa on uuden koodin tuottamista ja kirjastojen integrointia johonkin alustaan.

Seuraavissa kappaleissa käydään läpi tavallisimmat tavat hankkia ohjelmistoja käyttöön ja kerrotaan miten ne soveltuvat eri tarpeisiin.

2.4.1 Lisensointi

Tavallisesti kirjastoja ja valmisohjelmistoja ei myydä, vaan ne lisensoidaan, mikä tarkoittaa käyttöoikeuden luovuttamista korvausta vastaan. Immateriaalioikeudet säilyvät alkuperäisellä tekijällä tai yrityksellä. Lisensointimalleja on lukuisia ja tämä onkin teknisen arvioinnin lisäksi kaikista tärkein asia ymmärtää kun ollaan tekemässä kaupallista toimintaa – tuotteeseen lisensoitavat teknologiat eivät saa heikentää ainakaan liikaa tuotteen myytävyyttä tai tuotteesta saatavaa katetta. Muutama esimerkki lisensointimalleista:

Kertamaksu, pysyvä – maksetaan ohjelmiston käytöstä kertamaksu. Tämä voi sisältää oikeuden päivityksiin ja oikeuden käyttää ohjelmistoa osana kaupallista tuotetta rajattomasti tai sitten asennusten lukumäärän tai vaikkapa kapasiteetin perusteella johonkin rajaan asti. Kustannus on helppo ymmärtää, toisaalta tämä voi tehdä ohjelmiston hankinnasta kallista, koska valmistajan täytyy kattaa kaikki kulunsa ja liikevoittonsa yhdestä maksusta. Tämän lisäksi valmistaja hinnoittelee tavallisesti erikseen päivitykset ja teknisen tuen.

Rojalti – ohjelmiston valmistajalle maksetaan tietty summa per myyty tuote tai prosenttiosuus myydyn tuotteen arvosta. Edullinen tapa tuotekehittäjälle, rahaa ei sidota lisensseihin, mutta jos tuote käy kaupaksi voi rojaltien kautta merkittävä osa katteesta mennä muille kuin tuotekehittäjälle, varsinkin jos rojalti on kiinteä mutta tuotteen hinta laskee kilpailun myötä. Yksi mielenkiintoinen aspekti on teknologiatoimittajan sitominen tuotekehittäjän liiketoiminnan onnistumiseen – on molempien kannalta hyödyllistä, että tuotteesta tulee hyvä ja sitä saadaan kaupaksi.

Aikaperusteinen, tilauspohjainen – ohjelmiston valmistajalle maksetaan esimerkiksi kerran vuodessa lisenssimaksu. Tässä teknologiaa ostetaan ikään kuin osamaksulla. Yleensä tähän sisällytetään korvaus ylläpidosta, eli valmistaja tarjoaa päivityksiä, korjauksia ja tukipalveluita samaan hintaan.

Avoin lähdekoodi – poikkeuksena edellisiin, avoimen lähdekoodin lisensoinnissa luovutetaan tietyillä ehdoilla myös lähdekoodit ja lisenssin tavoitteena on antaa käyttäjille

sellaisia oikeuksia mitä heillä ei normaalien tekijänoikeuksien perusteella olisi, kuten levittää tekijänoikeuksien alaista materiaalia edelleen ja muokata sitä. Avoimen lähdekoodin käyttö ei tavallisesti maksa suoraan mitään vaikka lisenssiehdoissa mikään ei estä jakelua myös korvausta vastaan.

Avoimen lähdekoodin lisensoinnissa tekijä tai tekijät eivät siis luovu oikeuksistaan vaan antavat muille oikeuden käyttää lähdekoodeja, vaatien joskus muutosten luovuttamista muille tai vaikkapa tekijöiden nimien sisällyttämistä tuotteeseen (The Open Source Initiative (OSI), 2007). Open source lisenssit voivat myös sisältää kieltoja ohjelmistopatenttien hyödyntämiseen tai ne voivat olla viraalisia, tarttuvia. Seuraavassa listassa tärkeimpiä avoimen ja vapaan lähdekoodin lisenssejä:

General Public License (GPL) (Free Software Foundation, 2007) vaatii luovuttamaan sekä alkuperäisen ja muokatun mutta myös kaikki siihen linkitetyt ohjelmistot samojen ehtojen alla – lisenssi näin ikään kuin tarttuu ja leviää. Tällä tavalla lisensoitua lähdekoodia ei normaalisti haluta mukaan kaupallisiin ohjelmistoprojekteihin.

Lesser General Puclib License (LGPL) (Free Software Foundation, 2007) poikkeaa GPL:stä sallimalla dynaamisen linkityksen. Tämä mahdollistaa LGPL:n alaisten kirjastojen hyödyntämisen myös sellaisten ohjelmien yhteydessä, joiden lähdekoodia ei haluta julkistaa. Erityisen tärkeää tämä on mm. käyttöjärjestelmän peruspalveluiden osalta (C-kirjasto) ja ylipäätään sellaisten kirjastojen kohdalla joille toivotaan laajaa käyttäjäkuntaa.

Berkeley Software Distribution (BSD) -lisenssit (Regents of the University of California, 1999), **MIT**- ja **X11**-lisenssi, **zlib/libpng** -lisenssit, eivät aseta vaatimuksia lähdekoodin julkaisemisesta tai ohjelmistopatenteista, estä lisensoidun koodin myymistä edelleen, eikä sen käytöstä tarvitse maksaa. Yrityksen kannalta tämä lisenssi on ehkä kaikista kätevin eikä sido mihinkään, tavallisesti ei edes antamaan tunnustusta, että jotain teknologiaa on otettu avoimen lähdekoodin projektista. MIT-lisenssi (Open Source Initiative) tekee edellä mainitussa poikkeuksen ja vaatii maininnan tekijänoikeuksista ja lisenssiehdoista ohjelmassa.

Apache 2.0 lisenssi (Apache Software Foundation, 2004) on hyvin lähellä BSD-lisenssiä, eli muutoksia ei tarvitse julkaista, mutta sen sijaan vaaditaan muutosten dokumentointia ja näiden dokumenttien säilyttämistä lähdekoodin yhteydessä. Jos lisensoidun teoksen mukana on ”NOTICE”-tiedosto, se täytyy toimittaa luettavassa muodossa käyttäjälle, joko osana dokumentaatiota tai ohjelmassa itsessään.

Ohjelmiston valmistaja, taho joka omistaa tekijänoikeudet, voi myös halutessaan yhdistellä eri lisensointimalleja ja tarjota kirjastojaan esimerkiksi avoimen lähdekoodin lisenssin alla, mutta myös jollain kaupallisella lisenssillä jos asiakas ei halua jakaa oman tuotteensa lähdekoodeja koko maailmalle.

2.4.2 Kirjastot ja valmisohjelmistot

Ohjelmistojen kehittäjät pyrkivät hyödyntämään valmiita ohjelmistoja mahdollisimman pitkälle, yleensä ohjelmistojen ostaminen valmiina on aina halvempaa kuin saman toiminnallisuuden toteuttaminen itse. Perinteisin tapa uudelleenkäyttöön on paketoida aliohjelmia ja luokkia kirjastoiksi, jotka voi myöhemmin liittää osaksi ohjelmistoa. Kirjastojen tuomia palveluita käytetään varsinaisten ohjelmien apuna, laajentamaan toiminnallisuutta (Wheeler, 2003).

Samaan tapaan kuin ohjelmistojen kehittämisessä kirjastoja käyttämällä, voidaan valmisohjelmiston hankinnalla säästää kustannuksia. Tämä pätee kunhan voidaan välttää ohjelmiston räätälöinti ja valmisohjelmisto tarjoaa tarvittavat toiminnot eikä hinta ole liian korkea.

2.4.3 Pilvipalvelut ja infrastruktuurin ulkoistaminen

Pilvipalveluille, oli sitten kyse jonkin tietyn sovelluksen ostamisesta palveluna tai koko infrastruktuurin tilaamisesta, on tyypillistä jokin tilaus-tyyppinen liiketoimintamalli. Ostetaan tietyllä aikavälillä rajattu määrä API-kutsuja, virtuaalikoneinstansseja, massamuistia, tiedonsiirtoa jne. Ostettavan virtuaalihyödykkeen laatu riippuu luonnollisesti pilvipalvelun tyypistä ja tarkoituksesta, mutta leimallista on matala aloituskynnys ja helppo skaalaus – hinnat laskevat per yksikkö volyymien kasvaessa.

IaaS – infrastruktuuri palveluna

Infrastructure as a Service, palvelimien, tallennustilan, verkkoyhteyksien ja ylläpidon ulkoistaminen. Järjestelmän infrastruktuurin ulkoistaminen pilveen mahdollistaa palveluiden tuottamisen hajautetusti ja skaalautuvasti. Samankaltaisen järjestelmän rakentaminen yrityksen omiin tiloihin on edes teoriassa mahdollista vain isoille yrityksille kun halu-

taan globaali saatavuus palvelulle ja kallista tuottaa edes ilman maantieteellistä hajautusta. Globaalit IaaS-toimittajat tarjoavat lisäksi palveluita hyvin matalilla aloitushinnoilla, lähes ilmaiseksi. IaaS toimittajia: Amazon Web Services, Microsoft Azure, Google Cloud Platform – Compute.

IaaS tarjoaa lähtökohtaisesti vain laskentatehoa ja tallennustilaa. Varsinainen ohjelmisto joka hyödyntää näitä on asiakkaan vastuulla. Asiakkaan täytyy myös huolehtia ohjelmistonsa ylläpidosta.

Eri käyttäjät on loogisesti erotettu toisistaan käyttämällä virtuaalikoneita eli eri tilaajien tiedot eivät pääse sekoittumaan. Looginen erottelu ei ole kuitenkaan sama kuin fyysinen erottelu. Jos hyökkääjä pystyy ohittamaan virtuaalisoinnin, voi hyökkääjä päästä käsiksi samassa fyysisessä palvelimessa sijaitseviin virtuaalikoneisiin.

PaaS – palvelualusta palveluna

Platform as a Service, palvelualusta pilvipalveluna. Mahdollistaa ohjelmistokehityksen pilvimallin mukaan eli kehittäjä voi hyödyntää PaaS:issa olevia palveluita sekä yleensä tuoda sinne omia palveluitaan tarjolle muiden käyttöön. Kehittäjän kannalta palveluiden käyttö on vaivatonta ja nopeaa, palvelun tarjoaja on vastuussa skaalautumisesta tehontarpeen mukaan esimerkiksi käyttäjämäärien noustessa. PaaS:in huonona puolena on rajoittuminen palveluntarjoajan tarjoamiin teknologioihin. Jos jotain puuttuu, on joko suostuteltava palveluntarjoaja lisäämään tämä tai tehtävä tarvittavat asiat muualla, esimerkiksi toisessa pilvessä tai omassa palvelinsalissa, jolloin osa PaaS:in hyödyistä menetetään.

Tietoturvan kannalta PaaS ei oleellisesti eroa SaaS:ista, on tietenkin mahdollista tehdä loogista erottelua eri asiakkaita palveluille (esim. jokaiselle asiakkaalle luodaan oma virtuaalinen palvelininstanssi, jonka sisällä prosessointi tapahtuu, samoin levytila voidaan erottaa muista tilaajista). Tämän toteuttaminen riippuu palveluntarjoajasta.

PaaS -toimittajia: Google App Engine, Amazon AWS, IBM Bluemix, Microsoft Azure

SaaS – ohjelmisto palveluna

SaaS (Software as a Service), ohjelmiston hankkiminen palveluna. Perinteisesti ohjelmistot on lisensoitu ja kopio asennettu paikalliselle, käyttäjäorganisaation hallitsemalle työasemalle tai palvelimelle. SaaS-mallissa ohjelmisto on asennettu vain palveluntarjo-

ajan palvelimille. Käytöstä maksetaan tavallisesti sen mukaan kuinka paljon (ajallisesti) ja kuinka laajasti (eri ominaisuuksia) sitä käytetään. Tavallisesti palvelu tuotetaan kaikille asiakkaille samassa tuotantoympäristössä eli eri asiakkaiden tietoja säilytetään samassa palvelimessa ja samoilla massamuisteilla, vain ohjelmistoon rakennetut käyttöoikeuden hallintamekanismit erottavat ne toisistaan. Palvelun tuottaja huolehtii tuotantoympäristön ylläpidosta ja skaalaamisesta, loppukäyttäjälle nämä eivät näy mitenkään niin kauan kun palvelu toimii.

Kuluttaja-asiakkaiden kannalta SaaS-ohjelmistoja käytetään yleensä selaimella, jolloin ohjelmisto on asiakkaan kannalta alustariippumaton, kunhan moderni selain on käytettävissä.

Tunnettuja SaaS sovelluksia: Google Docs, Sheets, Slides, Microsoft Office 360.

2.4.4 Pilvipalveluiden haasteita

Kaikille pilvipalveluille ominainen piirre on laskennan ja tallennustilan siirtyminen pois yrityksen omista tiloista suuriin konesaleihin, yleensä yhdysvaltalaisten yritysten omistamiin ja hallitsemiin. Konesalit voivat olla fyysisesti lähes missä tahansa (tavallisesti samalla mantereella kuin kuluttaja) ja tästä seuraa hankalasti hallittavia tilanteita tietojen luottamuksellisuuden ja noudatettavien lakien suhteen. Minkä maan lakien perusteella esimerkiksi poliisi- tai tiedusteluviranomaiset saavat pääsyn tietoihin? Tästäkin johtuen monet luottamuksellisia tai kansallisesti merkittäviä tietoja käsittelevät yritykset ja virastot ovat kieltäneet tietojensa käsittelyn muualla kuin esimerkiksi Suomessa tai EU-maissa.

3 Tutkimuksen tekeminen

Tutkimus aloitettiin miettimällä käyttötapauksia jotka olisivat oleellisia Bittiumin liiketoiminnalle. Käyttötapaukset on kuvattu tarkemmin kappaleessa 3.1. Mukaan otettiin useampi erilainen käyttötapaus jotta yrityksen eri liiketoiminta-alueiden tarpeet tulisivat mukaan.

Tämän jälkeen kerättiin tietoja eri keinoälyalustoista ja analysoitiin niiden soveltuvuutta käyttötapauksiin. Alustoja on tarjolla paljon erilaisia ja valinta analyysin kohteiksi tehtiin ensisijaisesti työnohjaajilta (Bittiumilta) ja toissijaisesti valittiin mukaan analyysin aikana lupaaviksi havaittuja kilpailijoita tai teknologiansa puolesta mielenkiintoisia, jollain tapaa joukkoa täydentäviä alustoja.

Viimeisenä vaiheena tehtiin käytännön kokeiluja kolmella eri alustalla, jotka valikoituivat edellisen vaiheen analyysin perusteella. Nyt haettiin kokemusperäistä tietoa alustojen käyttöönotosta ja käytöstä, mitä ne vaativat käyttäjältään, analysoitavalta tiedolta, miten niitä voisi vertailla keskenään. Datana käytettiin valmiita datasarjoja Internetistä, https://www.kaggle.com/datasets (Kaggle.com, 2016), näin tulosten varmistaminen ja alustojen suorituskyvyn vertailu oli mahdollista sekä tutkittujen alustojen kesken että myös laajemmin Internetistä löytyvän data perusteella.

3.1 Käyttötapaukset

Ennen varsinaisen analyysin tekoa kerättiin yrityksen asiantuntijoilta eri toimintojen kannalta oleellisia käyttötapauksia, joissa voidaan kuvitella jonkinlaisesta koneoppimisesta ja tekoälyalgoritmeista olevan hyötyä. Tässä törmättiin nopeasti ongelmaan: kun ei tiedetä mitä näillä algoritmeilla oikeastaan voidaan saada aikaan, ei voida määritellä välttämättä parhaita käyttötapauksia. On hyvin todennäköistä, että jokin myöhemmin täysin ilmeinen asia jää pois.

Yksi tärkeä asia on myös mitä prosessointia kannattaa tehdä pilvessä ja mitä laitteessa. Laitteet ovat yleensä akkukäyttöisiä ja myös niiden prosessointikapasiteetti voi olla rajallinen.

3.1.1 Sensori-orientoitunut AI (KT1)

Ympäristö: Käyttäjällä on matkapuhelin (Bittium Tough Mobile (Bittium, 2017)) ja rannetietokone. Rannetietokone on yhteydessä matkapuhelimeen esim. Bluetooth Low Energy (BLE) yhteydellä, tai vastaavalla jatkuvan yhteyden mahdollistavalla teknologialla. Sekä matkapuhelimessa että rannetietokoneessa on kiihtyvyysanturit. Rannetietokoneessa on myös sykkeenmittaus sekä ihonsähkönjohtavuutta mittaava anturi. Molemmat laitteet ovat akkukäyttöisiä, joten virrankulutus on niiden kannalta tärkeä tekijä.

Alikäyttötapaukset:

KT1.1. Käyttäjän tunnistaminen anturidatan perusteella

KT1.2. käyttäjän tilanteen tunnistaminen – onko käyttäjä esimerkiksi

- stressaantunut
- kaatunut maahan
- ylipäätään elossa.

Huomioitavaa: Android käyttöjärjestelmän rajoitukset anturidatan hyödyntämiselle, alustan laskentakyky (mitä voidaan tehdä rannetietokoneessa, puhelimessa, mitä välttämätöntä viedä pilveen). Virran ja verkkoresurssien kulutus. Mahdollisuus toimintaan ilman yhteyttä pilveen (KT1.1).

3.1.2 Semi-structured data -orientoitunut AI (KT2)

Ympäristö: Hajautettu järjestelmä yrityksen IT-infrastruktuurissa tai pilvessä, osin tai kokonaan. Ei oleellisia rajoituksia virrankulutuksen suhteen, jos tarjotaankin käyttöliittymä mobiililaitteille, prosessointi tapahtuu silti muualla käyttötapauksesta johtuen.

Alikäyttötapaukset:

KT2.1. Suurien tietomassojen analysointi esimerkiksi sosiaalisesta verkosta:

- lainapäätöksen tekeminen pankissa
- vakuutussopimus tai vakuutuskorvaushakemuksen käsittely automaattisesti.

KT2.2. Yrityksen verkon IP-liikenteen analysointi

- Tilannekuvan muodostamiseksi (esimerkiksi: ollaanko DDoS hyökkäyksen kohteena, vuotaako joku tietoja kilpailijalle).

- Erilaisten yhteysteknologioiden vertailu kustannussäästöjen saamiseksi tai suorituskyvyn parantamiseksi.

KT2.3. IPA (intelligent Personal Assistant) toiminteet, case: treeniohjelman räätälöinti

- Yhdistellään käyttäjän tiedot ja tavoitteet vs. valmiit ohjelmat tai tutkimustieto siitä, minkälaiset harjoitteet tukisivat parhaiten ko. käyttäjää.

- Lisäksi voidaan ottaa huomioon sää, vuodenaika, salien aukioloajat/saatavuus käyttäjälle jotta hienosäädetään lopputulosta olemaan käytännössä järkevä.

KT2.4. Tuotantotestausdatan hyödyntäminen tuotannon ohjauksessa ja vikojen ennakoimiseksi

- Ennustavaa analytiikkaa, ennakoidaan ongelmia tuotannossa ennen kuin ne tapahtuvat.

- Voidaan parantaa tuotannon läpimenoaikaa ja saantoa sekä varmistaa, että asiakkaalle lähtee laadukkaampia tuotteita.

3.1.3 Puheen ja kuvan tunnistus AI (KT3)

Ympäristö: Koulutus ja isommat tietomassat täytyy käsitellä tietokoneella tai pilvessä, tunnistaminen olisi kätevää tehdä sulautetussa järjestelmässä (olisi mahdollista ennalta koulutetulla mallilla). Tämän hetken toteutukset kuluttajille tekevät (kaiken) prosessoinnin pilvessä, muutamaa poikkeusta lukuun ottamatta. Yksi tällainen on Applen FaceID, joka käyttää valmiiksi opetettua neuroverkkoa mutta tekee käytönaikaisen prosessoinnin täysin paikallisesti (Apple Inc., 2017). IoT laitteet voivat syöttää dataa järjestelmälle (ei vain desktop-tietokoneet tai älypuhelimet), vrt. IP-kamerat, puheentunnistuskodinkoneet kuten Amazon Echo (Amazon, 2017) tai Google Home (Google, 2017).

Alikäyttötapaukset:

KT3.1. Puheentunnistus (kuten Amazon Echo, Google Home, Apple Siri)

- tiedon hakeminen

- komentojen suorittaminen
- käyttäjän tunnistus
- käyttäjän mielentilantunnistus
- puheentunnistus Bittium Tough Mobileen (suomeksi Code-Q Oulussa, Nuance laajemmin maailmalla).

KT3.2. Kuvantunnistus (IP-kamerassa tai taustajärjestelmässä)

- liikkeentunnistus
- kasvojentunnistus
- asiakasmäärien laskeminen.

3.2 Arvioidut teknologiat

Ennen kuin valittiin teknologiat käytännön kokeiluihin, esiselvityksessä selvitettiin:

- mitä on tarjolla
- miten alustat näyttävät eroavan toisistaan
- mihin ala on menossa?

Kappaleen 3.1 käyttötapaukset olivat ohjaamassa myös tätä, mietittiin samalla voisiko näitä käyttötapauksia tehdä selvityksen alla olevalla alustalla. Tähän selvitykseen valikoitiin teknologiat yrityksen ohjaajien toiveista ja Internetin hakupalveluista ja blogeista löydetyistä oletetuista ”alan parhaista” alustoista. Selvityksen tuloksista laadittiin PowerPoint-esitys ja pidettiin esitelmä Bittiumin AI-koulutusohjelmaan osallistuneille henkilöille. Esiselvityksen teknologiat ja yhteenveto niiden tarjoamista mahdollisuuksista esitetään kappaleessa 5.1.

Esiselvityksen perusteella markkinoilla oli huomattavan paljon tarjontaa ja vaikka yleisimmin näytti, että käyttökelpoisimmat teknologiat ovat kaikki pilvessä, oli perusteltua painottaa kokeiluvaihetta kirjasto-tyyppisillä alustoilla. Tämä siksi, että Bittiumin asiakkaiden ja liiketoiminnan kannalta julkisten pilvipalveluiden käyttö on usein hankalaa. Käyttötapaus voi estää tai asiakasvaatimukset jopa kieltää tavanomaisten pilvipalvelui-

den käytön kokonaan johtuen tietoturvavaatimuksista (käsitellään luokiteltua tietoa) tai siitä, ettei esimerkiksi viranomaisverkoissa voida olla riippuvaisia pääsystä Internetiin tai välttämättä mihinkään keskitettyyn palvelin- tai pilviresurssiin.

Tässä vaiheessa tutkimusta rajattiin käyttötapauksen ja teknologioiden osalta. Käyttötapaukseksi otettiin KT1.2 (kappaleessa 3.1.1), tähän valintaan päädyttiin koska käyttötapaukselle löytyi hyvä tietosarja, voitiin keskittyä teknologioiden vertailuun datan käsittelyn sijaan. Tietosarjana käytettiin valmista Kagglen tarjoamaa sarjaa ihmisen toiminnan tunnistamisesta älypuhelimella (Kaggle.com, 2016). Tämä datasarja on alun perin UCI Machine Learning Repository:stä (Lichman, 2013) ja perustuu julkaisuun ”A Public Domain Dataset for Human Activity Recognition Using Smartphones” (Anguita;Ghio;Oneto;Parra;& Reyes-Ortiz, 2013). Tämä materiaali on tarjolla esikäsiteltynä. Koetilanteessa 30 vapaaehtoista oli tehnyt erilaisia aktiviteetteja (kävelyä, portaiden nousua ja laskua, istumista, seisomista, makaamista) joiden tuottama data oli kerätty puhelimen kiihtyvyysanturilta ja gyroskoopilta. Vapaaehtoisten toiminta oli myös videoitu ja myöhemmin luokiteltu manuaalisesti (mitä koehenkilö milläkin hetkellä teki). Data oli jaettu opetus- ja testijoukkoihin jotka muodostettiin satunnaisesti. Myös sensoridataa oli esikäsitelty poistamalla häiriötä ja keskiarvoistamalla sitä.

Datan esikäsittelyn tekeminen kuten yllä vaatii osin täysin manuaalista työtä (koehenkilöiden tekemisen luokittelu videolta, jos sitä ei jollain muulla keinolla voida yhdistää anturidataan) ja osin sellaista, mitä voitaisiin tehdä myös tutkittujen kirjastojen tarjoamilla menetelmillä, tai millä hyvänsä ohjelmointikielellä.

Rajauksessa käytännön kokeiluihin valittiin kolme eri teknologiaa:

- Microsoft Azure ML – state-of-the-art ML pilvipalvelu
- Scikit-learn – monipuolinen kirjasto joka tarjoaa valmiita ML-algoritmeja
- TensorFlow – kehyskirjasto, joka mahdollistaa ML-algoritmien joustavan kehityksen.

Näillä alustoilla tutkittiin kokemusperäisesti käyttötapauksen toteuttamista ja suorituskykyä, vastaamaan kysymyksiin mitä ko. alustalla voi tehdä, kuinka helppoa se on (vaatiiko paljon perehtymistä tai asiantuntemusta), onko lopputulos käyttökelpoinen käyttötapaukseen suorituskykynsä ja mahdollisten rajoitteiden takia.

Lisäksi jatkettiin esiselvitystä puheentunnistamisen, luonnollisten kielten prosessoinnin ja kuvantunnistuksen osalta, tarkoituksena ymmärtää mitä yrityksen kannalta relevantte-

ja teknologioita voitaisiin hyödyntää. Näiden esiselvitysten tuloksia ei käydä tässä tarkemmin läpi.

3.3 Tuloksien käsittely

Kehitystehtävän tuloksena saatiin

1. Tietoa eri alustoista Bittiumille työn edistymisen aikana, suoraan käyttöön AI taskforcelle.
2. Perusta alustojen ominaisuuksien vertailulle, käytettäväksi myös myöhemmin uusien bisnestapausten yhteydessä.

Kohta 1 toteutettiin PowerPoint materiaalina ja esitelmänä AI-koulutusohjelman henkilöille. Kohdan 2 toteutus on tämän raportin kappaleet 5 - 7.

4 Toiminnallinen viitekehys

Tutkimusympäristönä oli Bittiumin tuote- ja palveluliiketoiminta, jonka tarpeisiin lähdettiin hakemaan tietoa ja menetelmiä alustojen vertailuun. Bittiumin toiminta erityisesti sulautettujen ja langatonta tiedonsiirtoa käyttävien laitteiden parissa sekä painotus tietoturvallisten ratkaisuiden tekemiseen pyrittiin huomioimaan. Tutkimusta aloitettaessa Bittium oli hiljattain ostanut terveysteknologiayrityksen, Mega Electronics Ltd. (Bittium Oyj, 2016), ja sen myötä mietittiin myös terveysteknologiaan liittyviä käyttötapauksia.

Tutkimusta lähdettiin tekemään seuraavia välietappeja kohti:

1. Tunnistetaan Bittiumin liiketoiminnan kannalta relevantit AI-teknologiat -> käyttötapaukset listattuna.

2. Analysoidaan saatavilla olevat, yrityksen kannalta oleelliset alustat -> AI-esitelmä, tiedonkeruu-excel.

3. Saadaan ymmärrys siitä, miten alustojen tarjoamat ominaisuudet täyttävät vaatimukset -> opinnäytetyn raportti, viimeistelty ”kehikko-excel”.

Työn tuloksia haluttiin käyttää AI-koulutusohjelman tarpeisiin, joten tarvittiin läpileikkaus tarjolla olevista vaihtoehdoista esitelmän muodossa, mutta taustalla oli koko ajan jonkinlaisen raamin tai yleisemmän työkalun kehittämisestä alustojen valintaan. Pyrittiin hakemaan vastausta kysymykseen ”mitkä kriteerit ovat oleellisia alustaa valittaessa”.

Ongelman ymmärtämiseksi käyttötapauksia määriteltiin työnohjaajien kanssa ja näiden pohjalta lähdettiin tekemään analyysiä. Samaan aikaan kun tietoa kerättiin ja analysoitiin, tehtiin esitelmän lisäksi myös koostetta Excel-muodossa vertailua helpottamaan. Ehkä hieman naiivi, mutta idean tasolla oikeana pidetty tavoite oli, että tällaiseen taulukkoon voisi myöhemmin myös joku muu kuin sen alkuperäinen laatija lisätä jonkin uuden arvioitavan teknologian tiedot. Näin päästäisiin tekemään vertailuja myös vanhaa tietoa vasten.

Käytännön kokeiluilla lähdettiin hakemaan kokemusperäistä tietoa ja syvempää ymmärrystä miten käyttökelpoisia alustat ovat kehittäjän kannalta. Käytetyt teknologia-alustat, oikein käytettynä ja tuotteiksi jalostettuina, eivät näy loppukäyttäjälle mitenkään vaan tässä arvioidut käytettävyys yms. asiat koskevat vain kehitystyötä tekeviä asiantuntijoita.

5 Tutkimuksen tulokset

Tulokset jakautuvat kahteen osaan, analyysiin tutkituista ratkaisuista (esiselvitys) ja toiminnallisiin kokeiluihin. Ensimmäisestä vaiheesta esitetään yhteenveto ja viitataan samalla alkuperäisiin lähteisiin. Toiminnallisista kokeiluista käydään läpi tekijän havainnot prosessin aikana sekä yhteenveto jokaisen arvioidun teknologian suoriutumisesta.

5.1 Analyysin tulokset

Analyysin tarkoitus oli kerätä tietoa valituista teknologioista ja tämän jälkeen muodostaa yhteenvetoja jokaisesta teknologiasta erikseen ja kaikista teknologioista yhdessä.

Analyysin jälkeen ratkaisut voitiin jakaa neljään eri kategoriaan alla esitetyillä kriteereillä:

1. **Sovellukset**: valmis tuote/palvelu jota voi alkaa käyttää ilman järjestelmän opettamista.

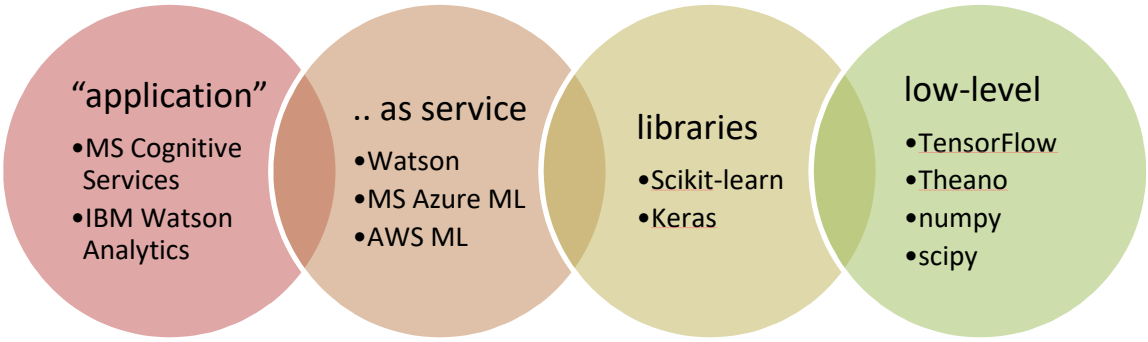
2. **Palvelut**: teknisesti kypsät, tuotteistetut palvelut, joista voi rakentaa haluamansa järjestelmän suhteellisen nopeasti ja suhteellisen helposti palvelun komponentteja yhdistelemällä. Opettaminen on käyttäjän vastuulla.

3. **Kirjastot**: vaihtelevalla teknisellä kypsyydellä olevia algoritmikirjastoja. Vaatii käyttäjältä vahvaa osaamista oikean algoritmin valintaan ja sen integroimiseen haluttuun sovellukseen.

4. **Matalan tason kirjastot**: matematiikkakirjastoja jotka helpottavat koneoppimisessa tarvittavan laskennan tekemisessä ja erityisesti rinnakkaislaskennassa. Eivät välttämättä tarjoa valmiita algoritmeja, vaativat enemmän asiantuntemusta koneoppimisesta ja ohjelmoinnista kuin korkean tason kirjastot.

Siinä missä kuvassa 2 kategorioiden vasen reuna tarjoaa nopeita ratkaisuja joita rakennetaan valmiista kaupallisista sovelluksista, on oikealla reunalla vain kasa rakennuspalikoita joista osaava tekijä voi tehdä mitä vain. Leimallista valmiille ratkaisuille on myös se, että niitä tarjotaan käytännössä vain pilvipalveluina. Vain matalan tason kirjastoja voisi kuvitella osaksi sulautettua järjestelmää, jossa kaikista resursseista on pulaa ja yhteys verkkoon on rajattu tai ajoittainen. Edelleen, vasemman reunan pilvipalveluilla voidaan käyttöön valjastaa paljon laskentaresursseja ja tietomassoja. Oikean reunan tek-

nologiat mahdollistavat kehittäjän valitsemat kompromissit sovelluksen ja käyttöympäristön mukaan.



Kuva 2 Koneoppimispalveluiden ja kirjastojen kategoriat

Analysoidut teknologiat on listattu taulukossa 2, teknologiat on tässä kategorisoitu aikaisemmin esitellyllä tavalla. Seuraavissa kappaleissa jokainen käydään läpi tarkemmin ja kolmelle teknologialle tehtiin myös käytännön testejä.

Taulukko 2 Yhteenveto analysoiduista koneoppimisjärjestelmistä

Palvelu / kirjasto	**Tyyppi**	**Analyysi kappaleessa**	**Toiminnallinen kokeilu**
Microsoft Cognitive Services	Sovellus	5.1.1	
IBM Watson	Palvelu	5.1.2	
Microsoft Azure Machine Learning	Palvelu	5.1.3	5.2.1
Amazon AWS Machine Learning	Palvelu	5.1.4	
Scikit-learn	Kirjasto	5.1.5	5.2.2
TensorFlow	Matalan tason kirjasto	5.1.6	5.2.3

5.1.1 Microsoft Cognitive Services

Microsoft Azure tarjoaa monipuolisia palveluita pilvipalveluina, SaaS, PaaS ja osin jopa IaaS –tyylisesti. Cortana Intelligence Suiten osana Cognitive Services tarjoaa valmiita koneoppimispalveluita, joita voi alkaa käyttää välittömästi. Käyttäjän tai kehittäjän ei tarvitse tietää mitään tekoälystä tai koneoppimisesta. Cognitive Services tarjoaa seuraavat valmiit toiminteet (Microsoft, 2017):

- kuva-analyysi, tunteiden tunnistaminen, kasvontunnistus
- tekstin moderointi
- videon stabilointi, kasvontunnustus ja seuranta, videoyhteenvedot (thumbnailing), videoanalyysi (mitä videossa näkyy)
- puhe tekstiksi, puhujan tunnistaminen
- kielenkäsittely, ymmärtäminen, lingvistinen analyysi, tekstin analysointi
- web-sisältö: haut, ehdotukset, kuvahaku, uutishaku, videohaku.

Siinä missä MS Cognitive Services tekee käyttöönotosta helppoa ja nopeaa, se myös rajaa käyttäjän käyttötapaukset eikä anna mahdollisuutta opettaa palvelua omalla datalla. Todennäköisesti MS pystyy kuitenkin tarjoamaan paremmin opetettuja algoritmeja kuin moni pystyisi itse tekemään ilman merkittävää investointia sekä rahallisesti että erityisesti ajan suhteen.

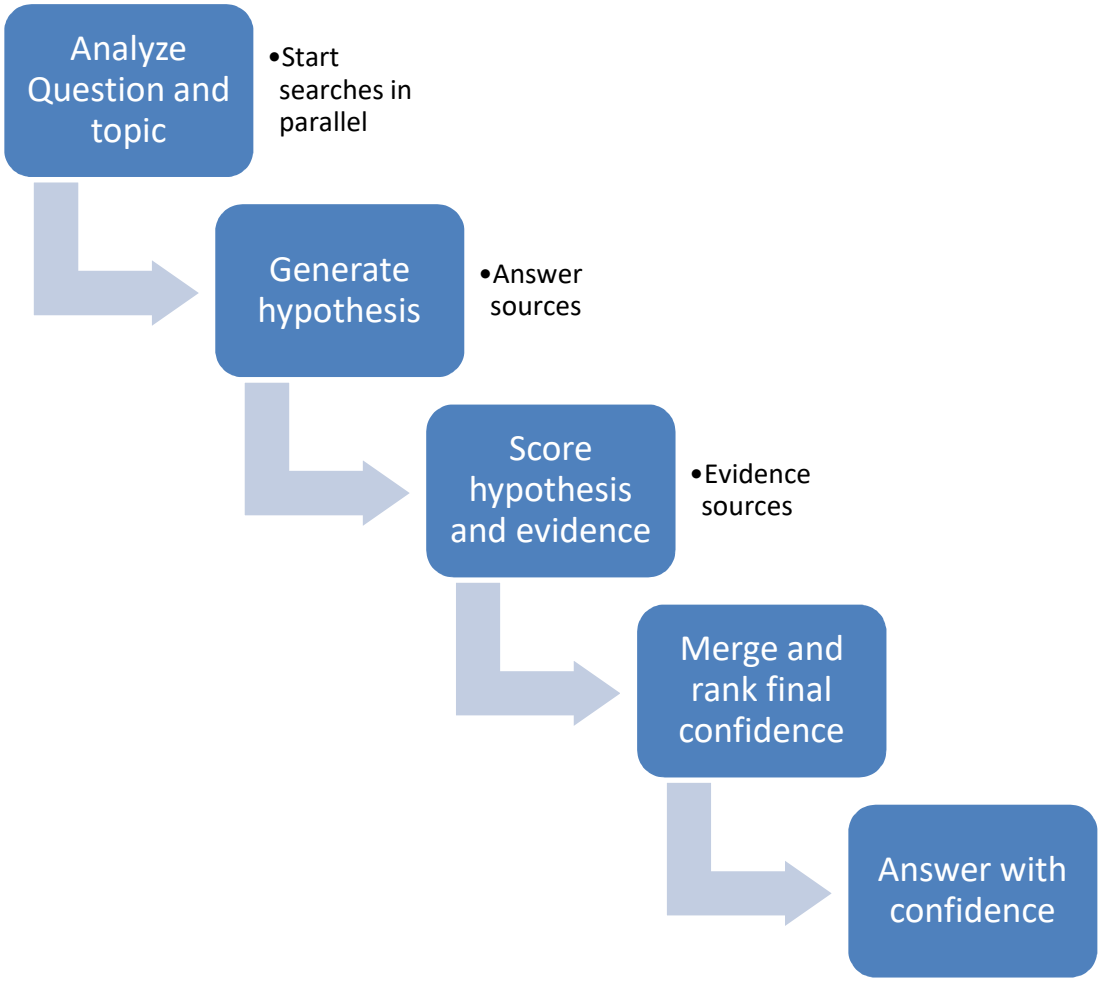
Cognitive Services:in ilmeisimpiä käyttökohteita ovat

- Älykkäät asiakaspalvelu-botit jotka voivat neuvoa asiakasta ja toimia ilman jäykkiä valikkorakenteita, tunnistaen erilaisia käyttäjän fraaseja. Botit voivat toimia teksti- tai puhekäyttöliittymällä.
- Käyttöliittymiä voidaan parantaa ennustamalla käyttäjän toimia ja tarjoamalla hakutoimintoja.
- Sisällön tunnistaminen ja moderointi keinoälyn toimesta tulee halvemmaksi kuin ihmisen palkkaaminen samaan tehtävään ja skaalautuu nopeammin. Cognitive services sallii moderoinnin tekstin lisäksi kuville ja videoille.

Jos MS Cognitive Services tarjoaa sopivat toiminnot, on sovellusten rakentaminen nopeaa ja helppoa. Konkreettisia ja käyttökelpoisia tuloksia saadaan heti kokeiltavaksi ja julkaiseminen Microsoftin Azure Marketplacessa tai REST-rajapinnan kautta jollekin omalle sovellukselle tai alustalle on vaivatonta.

5.1.2 IBM Watson

IBM Watson on astetta matalamman tason palvelu verrattuna MS Cognitive Serivices:iin. Sen idea ei ole tarjota valmiiksi opetettuja keinoälyjä, vaan joukko palveluita joista pystytään rakentamaan tehokkaita keinoälyjärjestelmiä. Watsonin erityinen vahvuus on luonnollisten kielten käsittely; käyttäjä kysyy Watsonilta kysymyksen, joka analysoidaan asiayhteydessään, luodaan hypoteesi ja kerrotaan käyttäjälle lopuksi kuinka luottavainen Watson on eri vastaustensa oikeellisuudesta (Ferrucci, 2012). Kuva 3 kertoo tarkemmin miten Watson käsittelee kysymystä ja päätyy vastaukseensa.



Kuva 3 IBM Watsonin vastausprosessi (IBM, 2016)

Watson toimii täysin IBM:n pilvialustalla eikä vaadi käyttäjältä mitään omia asennuksia. Myös ilmaisia käyttöoikeuksia on tarjolla. Kuten kappaleessa 2.4.3 selvitettiin, palvelun tuottaminen pilvipalveluna on monella tapaa hyödyllistä sekä toimittajalle että tilaajalle. Watsonin yhteydessä yksi merkittävimmistä sovelluskohteista ainakin julkisuudessa on ollut lääkärien apuna toimiminen diagnoosia tehtäessä. Watsonin kyky yhdistellä suuria määriä aineistoja ja prosessoida ihmisen (luonnollisilla kielillä) tekemää tekstiä on tässä avainasemassa. Jos ja kun Watsonin aineistoon halutaan yhdistää myös ihmisten terveystietoja ja historiaa – mistä olisi suuressa mittakaavassa paljon hyötyä – tullaan hyvin nopeasti yksityisyyden ja tietosuojan rajoituksiin. Miten voidaan varmistaa, että ihmisten tiedot eivät päädy palvelimelle, joka on väärän valtion alueella tai kuinka tiukasti tiedot täytyy erottaa muiden organisaatioiden tiedoista? Pilvipalveluthan ovat tavallisesti “multi-tenant” tyyppisiä, eli samalla palvelimella/pilvessä palvellaan useita asiakkaita.

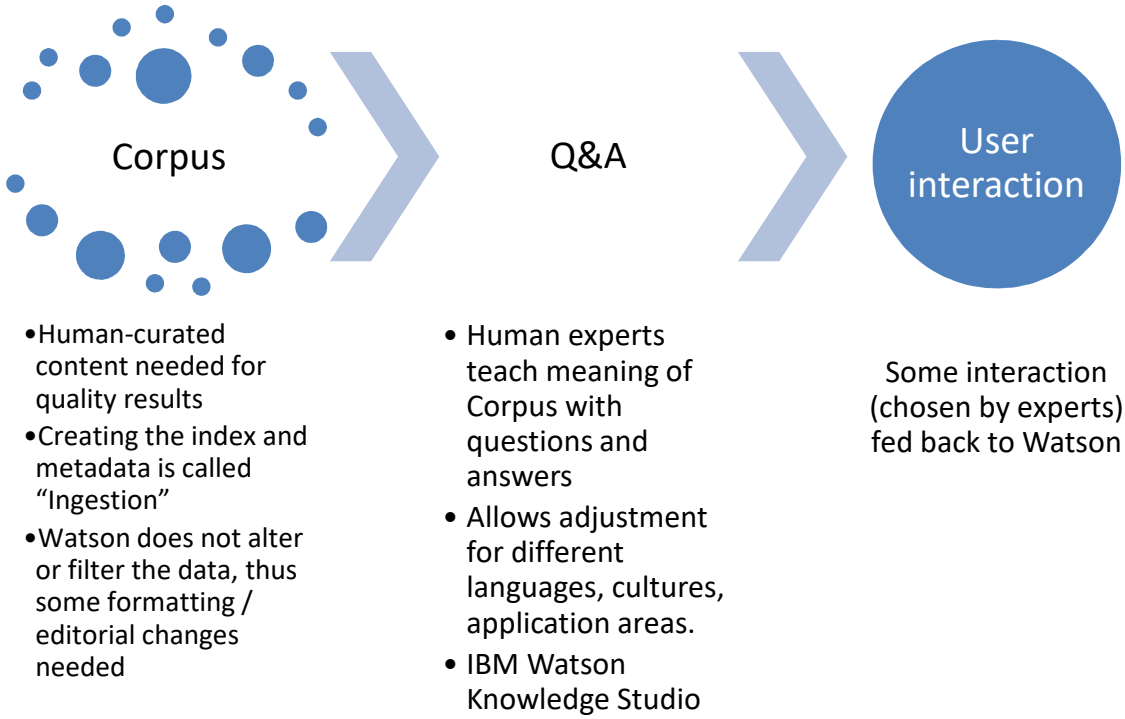
Miten IBM Watson käsittelee dataa ja miten se opettelee

IBM Watson ei ole mikään yksittäinen algoritmi tai ohjelma vaan suuri joukko algoritmeja ja palvelukomponentteja jotka yhdistettynä ihmisasiantuntijoihin tekevät Watsonista toimivan kokonaisuuden. Jokainen tapaus täytyy räätälöidä asiakkaan tarpeiden ja datan mukaan. Kuvassa 4 esitetään Watsonin opetusprosessi:

1. Watson käy ensin läpi suuren määrän materiaalia, esim. julkaisuja, kirjoja, tutkimustuloksia, web-sivuja jne. jotka asiantuntijat ovat valinneet. Lähdemateriaalin laatu on tärkeää. Watson ei osaa erottaa vaikkapa sivun kehyksiä leipätekstistä, joten todennäköisesti ainakin toimituksellisia muutoksia tarvitsee tehdä datan normalisoinniksi. Watson itsessään ei muokkaa lähdemateriaalia mitenkään indeksoidessaan ja luodessaan omaa metadataansa. Tämän vaiheen nimi on ”Ingestion” ja sen tuloksena Watson muodostaa ”Corpuksen”.

2. Tämän jälkeen Watsonille syötetään kysymys – vastaus pareja, joiden avulla Watson hahmottaa Corpuksen merkityksen ja sen käyttämän lingvistiikan.

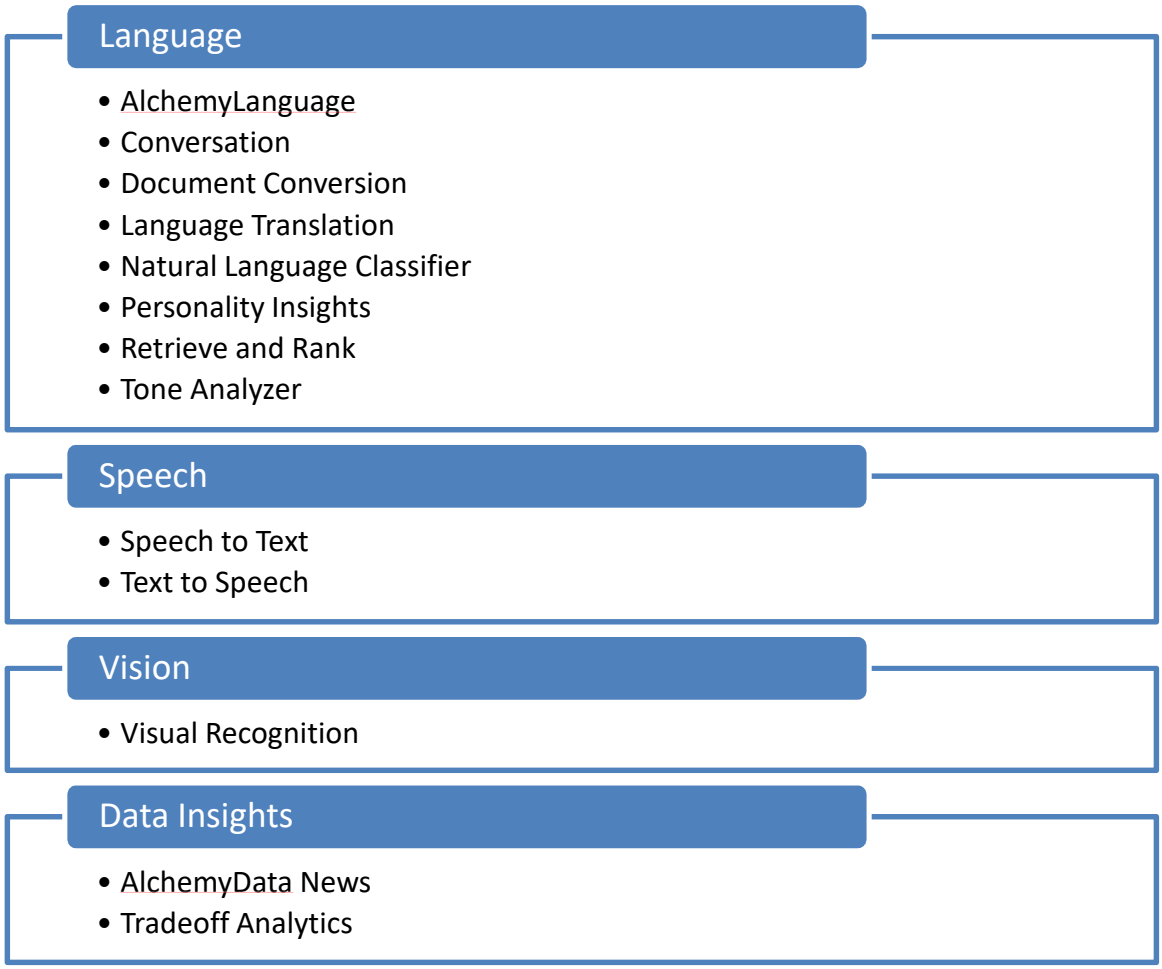
3. Nyt Watson on valmis käytettäväksi. Edelleen käytön aikana osa käyttäjien kysymyksistä syötetään Watsonille parantamaan tuloksia.



Kuva 4 IBM Watsonin opetusprosessi (IBM, 2016)

IBM Watsonin vahvuudet ja erot muihin järjestelmiin

Watson käyttää strukturoimatonta dataa. Suurin osa maailman tiedosta on tietokoneen kannalta strukturoimatonta, se on tarkoitettu ensisijaisesti ihmisen käytettäväksi. Watson myös ymmärtää luonnollisia kieliä, joille on tyypillistä keskenään erilaiset kieliopit, riippuvuus asiayhteydestä ja kulttuurista. Watson on uranuurtaja keinoälybisneksessä ja IBM:llä on paljon yhteistyökumppaneista ja onnistuneita asiakaskertomuksia. Watson-markkinointinimen alta löytyy hyvin monipuolinen tarjonta ohjelmointirajapintoja, nämä on listattu kuvassa 5 (IBM, 2017).



Kuva 5 IBM Watsonin ohjelmointirajapinnat

Huomioita

Watsonin käyttöönotto vaatii paljon asiantuntijatyötä:

- Sisältö täytyy valita ja suodattaa Corpusta varten.
- Kysymys-vastaus parit Corpuksen ymmärtämiseen.
- Palautteen syöttäminen järjestelmän ollessa käytössä.

Asiantuntijoiden täytyy tässä tapauksessa tuntea juuri sovellusalueen asiat, esim. lääketiede, ei niinkään AI-teknologioita.

Watsonin vastausten laadukkuus riippuu asiantuntijatyön laadusta ja lähdemateriaalista. Luonnollisesti Watson ei koskaan unohda mitään ja tuloksia voidaan koko ajan parantaa. Näin periaatteessa sen antamat vastaukset ovat ajan myötä aina vain tarkempia ja hyödyllisempiä. Tämä vaatii jatkuvaa panostusta tietojen pitämiseksi ajan tasalla eli asi-

antuntijoita tarvitaan jatkuvasti Watsonin käyttöön. Jos taas käyttäjäkuntaa laajennetaan ja jos käyttäjien kyky käyttää Watsonin vastauksia (sanallinen vastaus, referenssitiedot ja Watsonin luottamus omaan vastaukseensa) on huonompi, todennäköisesti tullaan pettymään suorituskykyyn. Maallikosta ei tule lääkäriä Watsonilta kysymällä.

Watsonin kaupallinen hyödyntäminen

IBM käyttää Watson-nimeä lukuisissa tuotteissaan, jotka liittyvät jotenkin Big dataan, data-analyysiin tai Internet of Things:iin (IoT) ja onkin välillä hyvin epäselvää mistä tuotteesta tai palvelusta lopulta on kyse. Yksityiskohtaista tietoa näiden palveluiden sisällöstä on myös hankala saada, materiaali on hyvin korkealla tai yleisellä tasolla.

IBM Watson (AI-palvelu) on hinnoiteltu API-kutsujen määrän mukaan. Hinnat riippuvat käytetystä API:sta ja siitä tarvitaanko asiakkaalle räätälöityä opetusta. Erilaisia hinnoittelumalleja on useita, osa myös ilmaisia. Ilman tiettyä käyttötapausta ja edes arvausta käytön volyymistä palvelun lopullista hintaa ei pysty määrittelemään.

Riskit

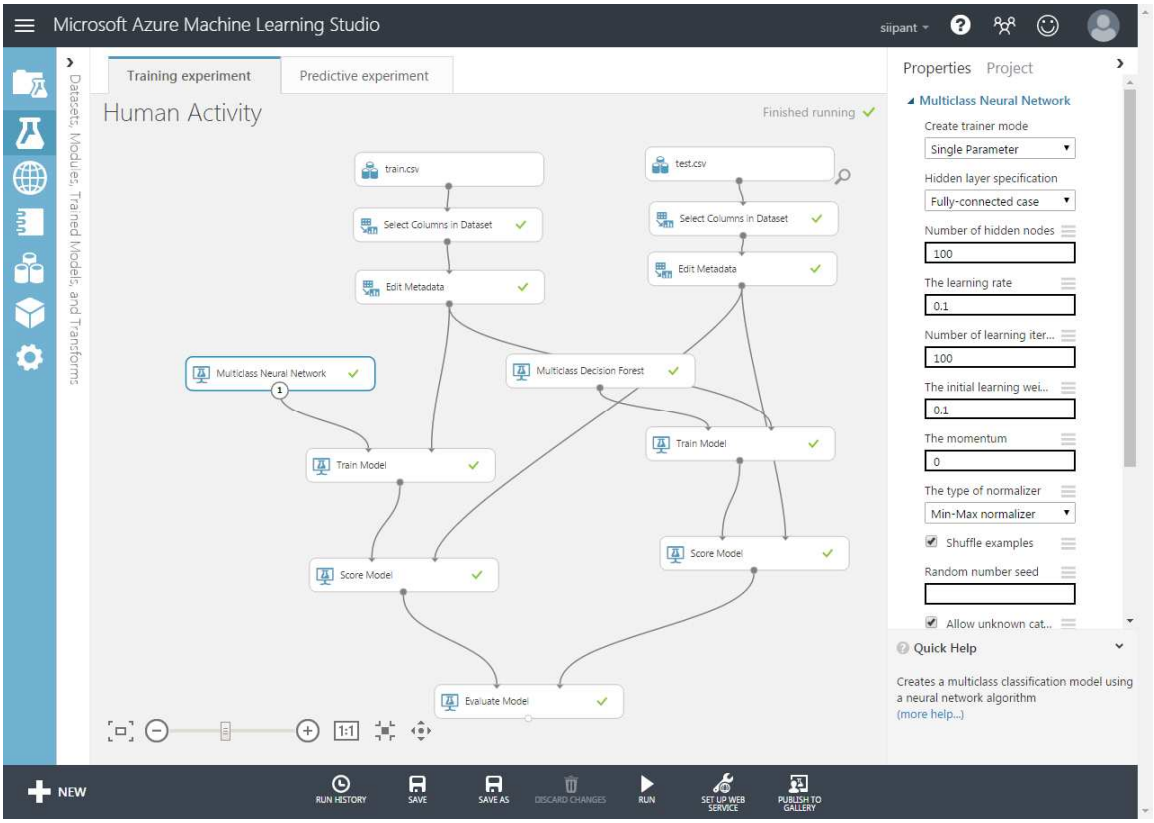
Watson näyttäisi olevan tarjolla vain pilvipalveluna ja vaikka yksityisyyttä (datan parempaa erottelua) voi saada lisähinnasta enemmän on kaikki siltikin pilvessä. IBM:n mukaan data kryptataan kun sitä siirretään verkossa ja se tallennetaan kryptattuna, IBM:n oman standardin ITCS300 mukaisesti. IBM pyrkii saamaan ISO 27001:en (ISO, 2017) joka määrittelee tietoturvallisuuden hallintajärjestelmien vaatimuksia.

5.1.3 Microsoft Azure Machine Learning

Microsoftin Azure (pilvipalvelu, jonka Cognitive Services –palvelurajapintoja kuvattiin kappaleessa 5.1.1) tarjoaa myös täysimittaista koneoppimispalvelua. Palvelu on toteutettu täysin pilveen, käyttäjä tarvitsee vain selaimen ja dataa analysoitavaksi.

Kuvassa 6 nähdään kuvakaappaus toiminnallisesta kokeilusta (tarkemmin kappaleessa 5.2.1) tällä alustalla. Palvelun avulla käyttäjä ei tarvitse kovin syvällistä tietoa koneoppimisesta ja pystyy hyödyntämään valmiita moduleita (algoritmeja) jotka Microsoft ja muut Azure-ekosysteemissä toimivat ovat tehneet. Keinoälyn opettaminen ja käyttö onnistuu hiirellä valmiita moduleita yhdistelemällä. Lopputuloksen voi avata ulkomaailmalle web-

servicenä, eli opetetun keinoälyn hyödyntäminen onnistuu sitten mistä sovelluksesta hyvänsä.



Kuva 6 Microsoft Azure Machine Learning Studio –kuvakaappaus

MS Azure ML tarjoaa hyvän valikoiman valmiita algoritmeja (Microsoft, 2017) joita voi konfiguroida hyvin laajasti. Sen lisäksi sitä voi laajentaa Pythonilla ja R:llä sekä on mahdollista käyttää OpenCV luokittelumalleja.

Jos tarjolla olevat dataformaatit, datamuunnokset ja algoritmit ovat riittäviä käyttötarkoitukseen, on MS Azure ML erittäin hyvä palvelu. Käyttöönotto on vaivatonta eikä vaadi syvää asiantuntemusta koneoppimisesta. Toisaalta palvelu on tarvittaessa laajennettavissa helposti joko Azure Market placen kautta tai yrityksen omalla asiantuntemuksella.

Koska kaikki on pilvessä, suurten tietomassojen hallinta ja prosessointitehon saatavuus voivat muodostua haasteiksi, erityisesti keinoälyä opetettaessa jos iteraatioita tulee paljon. MS Azure ML sisältää nykyisellään rajan tietosarjan koolle. Se voi olla korkeintaan 10GB kokoinen tai pienempi riippuen hyödynnettävistä moduleista.

5.1.4 Amazon AWS Machine Learning

Amazonin näkemys koneoppimispalvelusta on hieman erilainen kuin IBM:llä tai Microsoftilla. Palvelu mahdollistaa hyvin pinnallisella asiantuntemuksella koneoppimisen käytön ennusteisiin. Järjestelmän käyttö tapahtuu ohjatusti, käyttäjä opastetaan mallien luontiin vaihe vaiheelta. Vain rajattu määrä parametreja on käyttäjän säädettävissä, enimmäkseen palvelu valitsee ne taustalla käyttäjän puolesta. Kuvaus käyttöönotto-prosessista esitetään kuvassa 7, suurimmat haasteet liittyvät datan normalisointiin ja sen analysointiin.

Amazon AWS on teollisuusstandardi pilvipalveluissa ja luonnollisesti sen kautta tarjottuna myös koneoppimismallit ovat saavutettavissa web-servicenä. Hinnoittelu on myös hyvin selkeää, data-analyysi ja mallien rakentaminen on tuntihinnalla, ennusteet (mallien käyttö) taas käytön mukaan. Hinta vaihtelee vain sen mukaan halutaanko prosessointi erä-ajona vaiko reaaliaikaisena. Minkään muun koneoppimispalvelun hinnoittelu ei ole näin suoraviivaista.

Kuva 7 Amazon Web Services Machine Learning -mallien opetus käyttö

AWS ML on myös monella tapaa hyvin rajoittunut palvelu. Vain dataa joka on AWS:ssä ja malleja, jotka on määritetty AWS ML:ään voidaan hyödyntää. Syväoppimismoduleita ei ollut tarjolla vielä marraskuussa 2016 ja algoritmeja oli vain rajattumäärä muutenkin, lisäksi näiden tarkempi olemus ja parametrointi on aika pitkälle piilotettu käyttäjältä. Vuoden aikana Amazon on lisännyt valikoimaansa joukon syväoppimis Amazon Machine Imageja (AMI), eli virtuaalikonekuvia joita käynnistämällä Amazon Elastic Compute Cloudissa (EC2) käyttäjä saa laajennettua algoritmeja mitä käyttää Amazonin pilvessä. AMI:t tarjoavat yleisimpiä syväoppimisteknologioita kuten TensorFlow, Caffe, Theano ja Microsoft Cognitive Toolkit. Lisäksi Amazon EC2 yhdessä näiden AMI:ien kanssa tarjoaa neuroverkkojen laskentaan GPU kiihdytystä NVIDIA CUDA ajureilla ja tuen Intel Math Kernel Librarylle (Amazon, 2017). Vaikka koneoppimisen osalta hinnoittelu on selkeää,

myös muiden AWS-palveluiden käyttö täytyy ottaa huomioon. Vähintään S3 tallennustilasta tulee kustannuksia.

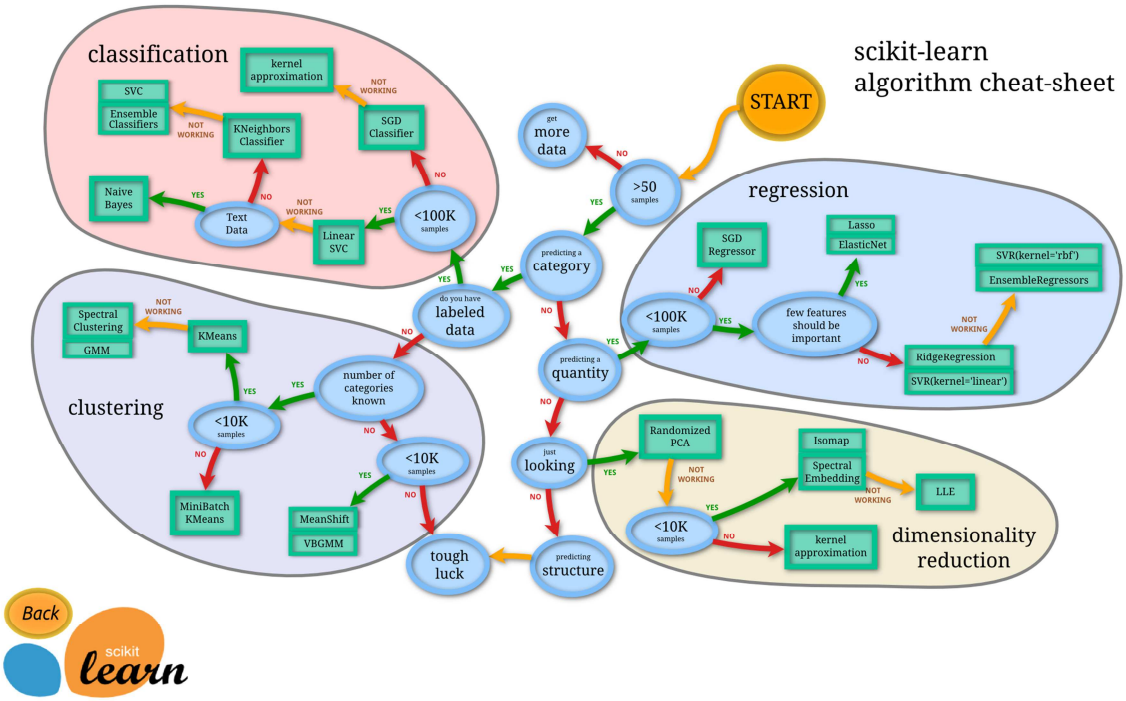
5.1.5 Scikit-learn

Scikit-learn on Python kirjasto, joka perustuu NumPy, SciPy ja matplotlib-kirjastoihin (Buitinck, ym., 2013). Scikit-learn sisältää lukuisia koneoppimisalgoritmeja jotka on paikoin optimoitu C-kielellä, joten suorituskyky on järkevällä tasolla, vaikka ohjelmointiin voikin käyttää korkean tason kieltä. Tukea ei kuitenkaan ole GPU:den käytölle, joten joidenkin mallien opettaminen voi viedä aikaa. Matplotlibin avulla myös visualisoinnit onnistuvat (Buitinck, ym., 2013), osaavalle käyttäjälle lopputulos on parhaassa tapauksessa jopa parempi kuin aikaisemmissa kappaleissa kuvatuissa pilvipalveluissa.

Asiantuntemuksen taso mitä vaaditaan tämän kirjaston käyttöön on hieman suurempi kuin aikaisemmin käsitellyissä pilvipalveluissa. Koneoppimisen ja data-analytiikan perusteiden lisäksi on hyvä ymmärtää edes perusteet Pythonista. Hieman pidempää kokemusta näistä sekä hyvää matplotlibin hallintaa vaaditaan, että päästäisiin samanlaisiin tuloksiin mitä aloittelija voi tehdä esimerkiksi MS Azure ML:llä (5.1.3). Algoritmeja ei kuitenkaan tarvitse koodata itse, joten aloituskynnys on vielä matala ja ideoiden kokeilu Pythonilla on erittäin nopeaa.

Scikit-learn on BSD-lisensoitu (Scikit-learn developers, 2017) eli Scikit-learn:ia pystyy hyödyntämään helposti myös osana kaupallista tuotetta. Myös se, ettei scikit-learn ole pilvipalvelu eikä mitenkään riippuvainen verkosta, vaan voidaan ajaa yrityksen omilla tietokoneilla on monissa tapauksissa vahvuus, koska käytettyä dataa ei tarvitse ladata ulkopuolelle. Kuvassa 8 esitellään scikit-learning algoritmeja eri käyttötarkoituksiin sekä työnkulku sopivan algoritmin (estimaattorin) valintaan sen perusteella,

- mitä halutaan tehdä (luokittelu, regressio, klusterointi, ulottuvuuksien vähentäminen)
- minkälaista dataa (luokiteltua, luokittelematonta)
- ja kuinka paljon dataa on käytettävissä.



Kuva 8 Scikit-learn algoritmin valinta (Scikit-learn developers, 2017)

5.1.6 TensorFlow

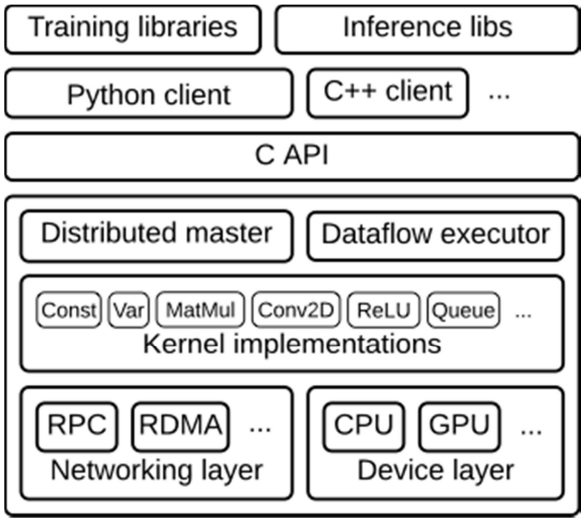
TensorFlown lyhyt kuvaus Googlen (2017) mukaan:

> TensorFlow™ on avoimen lähdekoodin kirjasto data flow graafien laskentaan. Solmut graafissa kuvaavat matemaattisia operaatioita ja graafin reunat kuvaavat moniulotteisia tietorakenteita (tensoreita) joilla viestitään solmujen välillä.

TensorFlow ei rajaa sovelluskohteita tai tekniikoita, mutta sitä on yleisimmin käytetty neuroverkkojen rakentamiseen ja opettamiseen. Kirjasto pystyy hyödyntämään useita prosessoreita (CPU) ja grafiikkaprosessoreita (GPU) rinnakkain sekä Tensor Processing unit:eja (TPU), mahdollistaen korkean suorituskyvyn erityisesti neuroverkkojen opettamisessa. TPU on Googlen kehittämä AI-kiihdytin, joka on rakennettu tätä tarkoitusta varten. Se mahdollistaa paremman hyötysuhteen (operaatioita / kulutettu teho) kuin yleiskäyttöiset prosessorit tai grafiikkaprosessorit. Samaa koodi toimii kuitenkin myös vaikka sulautetussa järjestelmässä, mahdollistaen hyvin erilaisten alustojen käytön tarpeen mukaan. Mallin voi vaikkapa suunnitella ja opettaa tietokoneella (tai isolla klusterilla tietokoneita joissa on erityisiä AI-kiihdyttimiä) mutta ajaa lopputulosta akkukäyttöisessä IoT-laitteessa.

TensorFlow on ollut kaupallisessa käytössä jo pitkään ja Google kehittää sitä edelleen aktiivisesti. Kehitystyöhön on mahdollista osallistua myös muiden ja lähdekoodi on saatavissa Apache 2.0 (TensorFlow Authors, 2017) lisenssin alla, mahdollistaen sen sisällyttämisen muihin kaupallisiin tuotteisiin ilman oleellisia rajoituksia.

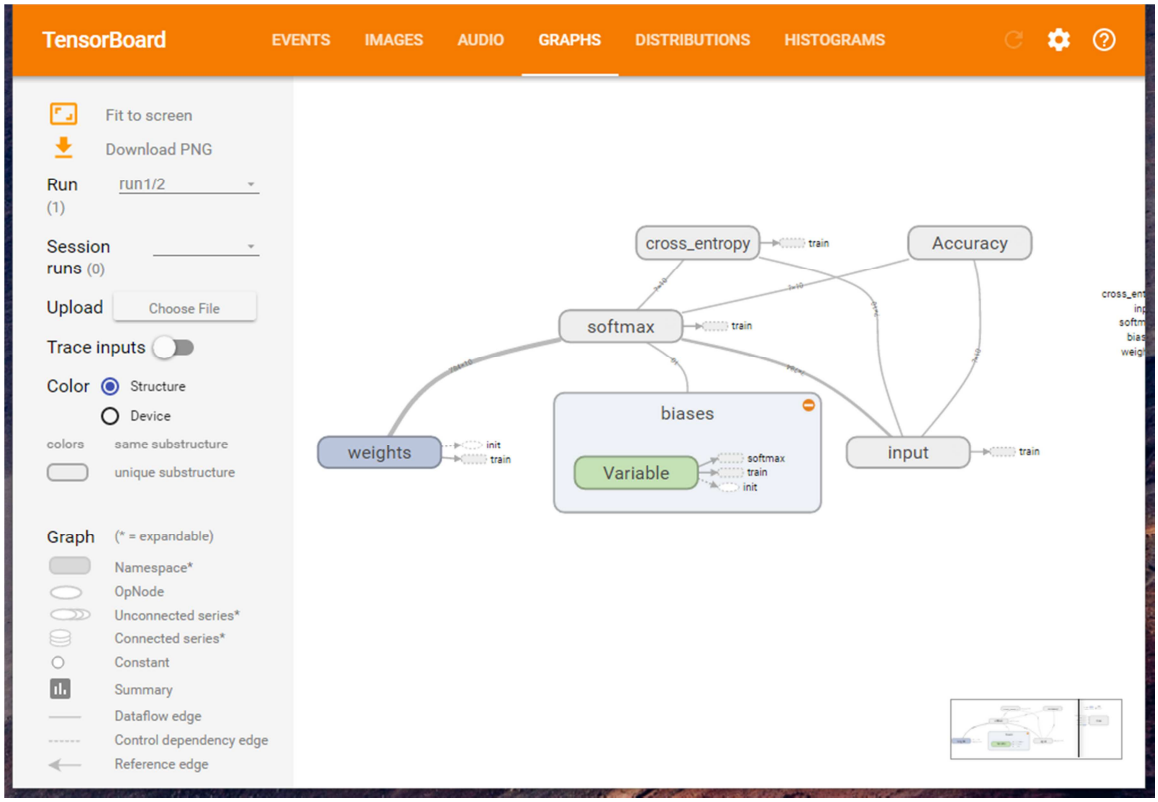
Kuvassa 9 esitellään TensorFlow'n arkkitehtuurin kerrokset. Kirjaston käyttö on mahdollista Pythonilla, C++:lla, Javalla ja Go:lla jotka kaikki voivat hyödyntää samoja toimintoja laskennan suorittamiseen ja hajauttamiseen niin verkon yli kuin saman laitteen suoritusyksiköiden kesken (alimmalla tasolla Networking layer ja Device layer). Tavallisesti kirjaston käyttäjät hyödyntävät sitä tarjottujen ohjelmointirajapintojen kautta, mutta koska koko toteutuksen lähdekoodi on saatavissa, on järjestelmän laajentaminen ja syvällisempi ymmärtäminen mahdollista.



Kuva 9 TensorFlow'n arkkitehtuuri (Google, 2017)

TensorFlow käyttää CUDA:aa NVidian GPU:den hyödyntämiseen. CUDA mahdollistaa usean GPU:n käyttämisen rinnakkaislaskentaa joustavasti, mutta ei tue kuin NVidian tuotteita. Tuki Googlen TPU:lle on hyödyllinen lähinnä Googlelle itselleen, koska TPU-kiihdyttimiä ei ole yleisesti saatavilla.

TensorFlow projekti sisältää myös TensorBoardin (kuva 10), jolla voi visualisoida graafeja ja näyttää kuvaajia neuroverkkojen avaintiedoista. Tämä on hyödyllistä koska tensorikerrokset ja niiden parametrit eivät ole aina helppoja hahmottaa. TensorBoardilla osan kaaviosta voi myös piilottaa, jolloin näkymässä voi keskittyä tiettyyn alueeseen ilman, että lukuisat parametrit sotkevat sitä.



Kuva 10 TensorBoard kuvakaappaus (Paperspace, 2017)

TensorFlown käyttöön vaaditaan huomattavasti asiantuntemusta. Käyttäjän tai kehittäjän täytyy osata

- muotoilla ongelma / algoritmi data flow graafeiksi eli käyttää tiettyä matemaattista tapaa algoritmin esittämiseen.
- tuntea halutut keinoälyalgoritmit tarpeeksi hyvin osatakseen koodata algoritmi oikein ja valita sopivat parametrit sen opettamiseen.
- tuntea TensorFlow'n teknologia riittävän hyvin osatakseen valita sille riittävän tehokkaan suoritusalustan. Tämä riippuu vahvasti algoritmista ja sovelluksesta.

Todennäköisesti tarvitaan useampi asiantuntija jos mikään valmis referenssitoteutus ei toimi auttavasti sellaisenaan. Työmäärä valmiin toteutuksen aikaansaamiseksi ja ylläpitämiseksi on väistämättä suuri verrattuna korkeamman tason palveluihin ja kirjastoihin. Toisaalta, TensorFlow on kypsä teknologia joka mahdollistaa erittäin suorituskykyisen keinoälyjärjestelmän rakentamisen ilman mitään ilmeisiä rajoitteita.

TensorFlow'n ja monen muun syväoppimista tukevan kehyksen suoritusalustana voi olla myös jokin IaaS-alusta. Myös AI-kiihdyttimiä (GPU-pohjaisia) on tarjolla usealta toimijal-

ta, mm. AWS Deep Learning AMI, MS Azure, Google Cloud Platformin Cloud Machine Learning Engine. IaaS-alusta mahdollistaa tarvittaessa lisäresurssien käyttöönoton ilman, että kehittäjät joutuvat hankkimaan lisää fyysisiä laitteita – ison klusterin ylläpito ei ole triviaalia jo sen tarvitseman tilan, verkkokaapeloinnin ja jäähdytyksen takia.

5.2 Toiminnallisten kokeilujen tulokset

Analysoiduista teknologioista valituilla kolmella alustalla tehtiin koemielessä mallin opettaminen ja testattiin miten ko. opetettu malli toimi testiaineistoilla. Tarkoitus oli arvioida tässä tarkemmin miten järjestelmät toimivat käytännössä, hankkia kokemusperäistä tietoa järjestelmien hankkimisesta, asentamisesta käytöstä, minkälaista osaamistasoa ne mahdollisesti edellyttävät käyttäjältään. Löytyisikö jotain ristiriitaa käytännön, dokumentaation ja markkinointimateriaalien välillä? Myös lisensointi ja maksujen kertyminen, mikäli relevanttia, pantiin merkille.

5.2.1 Microsoft Azure Machine Learning

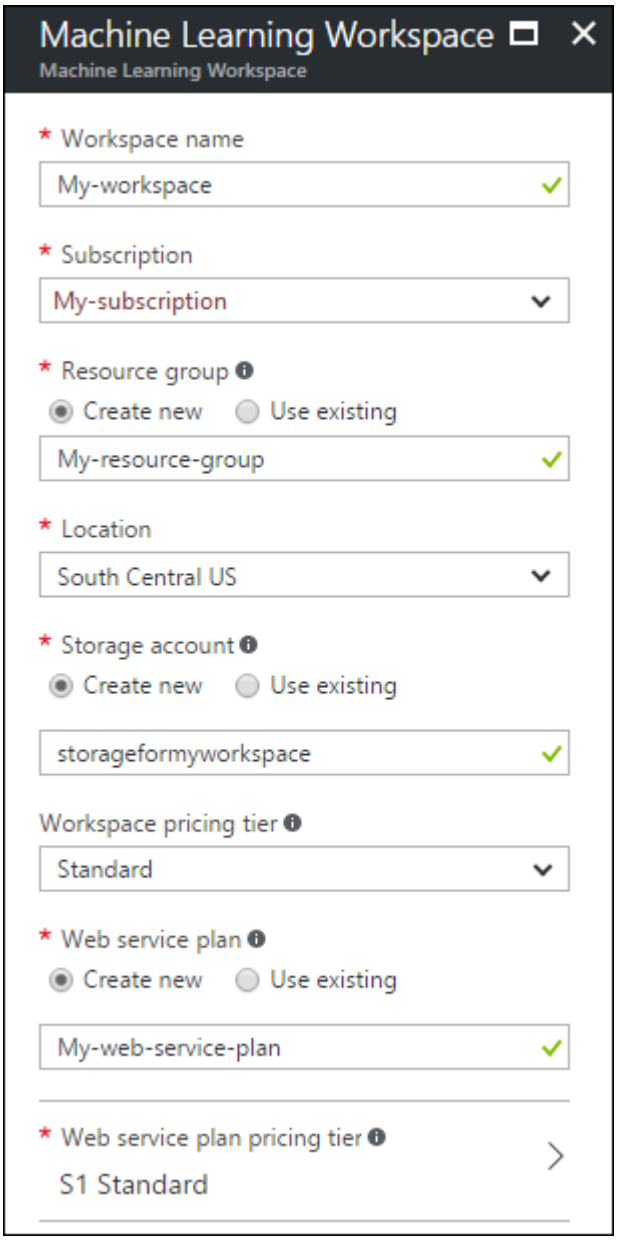
Microsoft Azure:n käyttöönotto helppoa, mutta jokseenkin monivaiheista. Mitään ei tarvitse asentaa tietokoneelle, riittää kun käytössä on jokin moderni selain. Luonnollisesti tietokoneen teholla ei ole myöskään väliä. Laadukas ja riittävän iso näyttö tekee käyttämisestä miellyttävämpää.

Ennen kuin päästiin suunnittelemamaan mallia, täytyi tehdä seuraavat valmistelut:

- perustaa käyttäjätili Microsoftin palveluihin (normaali tili jota voi käyttää esim. OneDrive-palvelun kanssa)
- ottaa käyttöön tai ostaa Azure tälle tilille (Microsoft tarjoaa määräaikaisen kokeilujakson ilmaiseksi)
- määritellä Azuresta tarvittavat palvelut, vähintäänkin täytyy luoda ”Machine Learning Workspace” ja varata tallennustilaa Azureen.

Ehkä kaikista hankalin vaihe liittyy Azuren palveluiden määrittelyyn, josta on kuvakaappaus kuvassa 11. Machine Learning Workspacen luomisen yhteydessä täytyy luoda ja määritellä koko joukko teknis-kaupallisia asioita, jotka on pakko selvittää ja osata määri-

tellä oikein ennen kuin pääsee itse asiaan kiinni. Esimerkiksi täytyy määritellä ”Subscription”, ”Resource group”, ”Location”, ”Storage account” jne. Microsoftille ominaiseen tapaan kaikki on kyllä dokumentoitu, jopa erittäin hyvin ja ymmärrettävästi, kun vain osaa löytää oikean dokumentaation. Machine Learning studion oma dokumentaatio kehottaa pyytämään ylläpidolta näiden määritysten tekemistä mikä onkin hyvä idea, jos tällainen ylläpito on yrityksessä erikseen olemassa.



Kuva 11 MS Azure Machine Learning Workspacen määrittely (Microsoft, 2017)

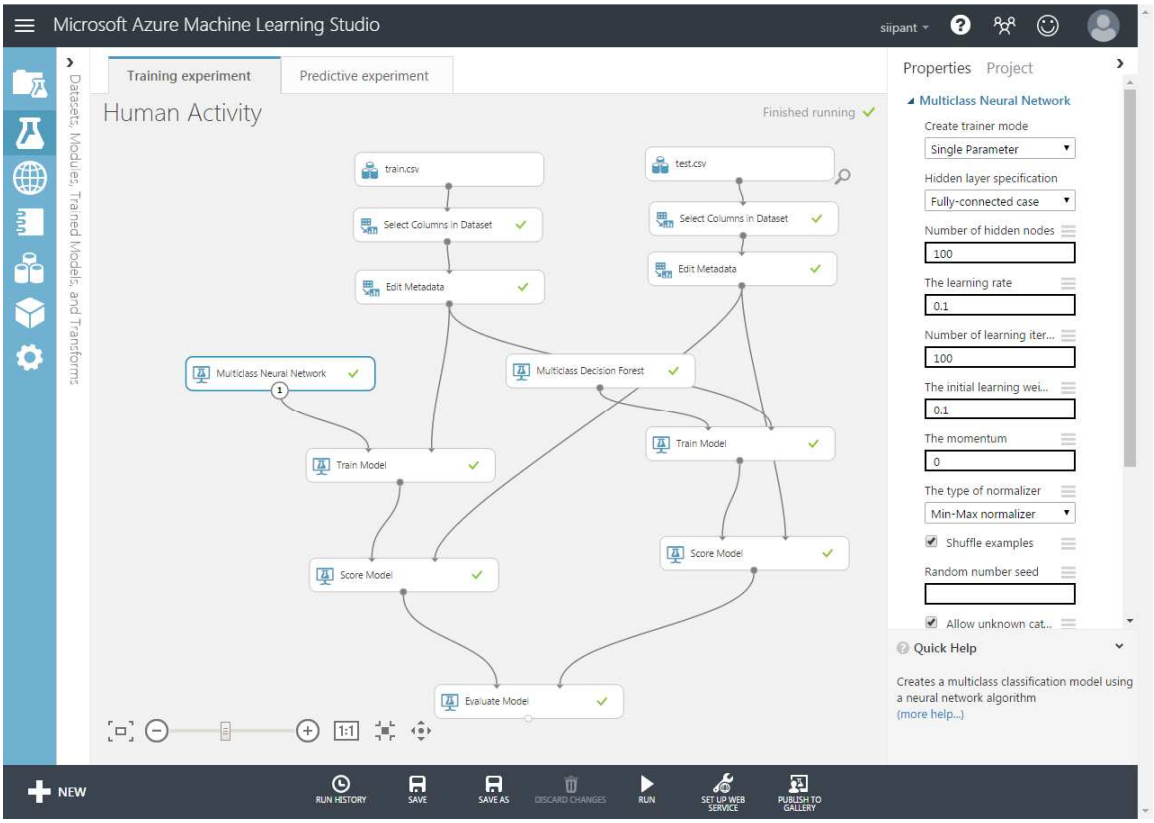
Kun määrittelyt on tehty ja Azuren toimintaan on tutustunut, se tarjoaa hyvin joustavan ja visuaalisen tavan määrittää ja seurata oman Azure-palvelun toimintaa ja kustannuksia. Eri toimintoja voi ryhmitellä ja erilaisia tilauksia hallita helposti.

Machine Learning Studion käyttö on helppoa kun seuraa opetusvideoiden neuvoja. Kaikki opetusmateriaali on saatavilla Microsoftin sivuilla ilman kirjautumista. Muutaman tunnin perehtymisen jälkeen oman mallin määrittely tyhjältä pöydältä oli helppoa ja ohjeisiin oli helppo palata.

Training experiment

Machine Learning Studion työtilassa määritellään ensin Training experiment, ”opetuskokeilu” (kuva 12), jonka luominen tapahtuu vetämällä ja pudottamalla tietolähteitä, tiedon muokkaus-moduleita, keinoäly-moduleita ja opetus- sekä arviointimoduleita, muun muassa (kuva 12, vasemman reunan kuvakkeet). Opetuskokeilun luominen tuskin voisi olla helpompaa, dataa voi myös visualisoida haluamastansa kohdasta kaaviota hiiren napautuksella. Järjestelmän käyttö erilaisten datamassojen analysointiin vaikuttaisikin hyvin luontevalta ja suurimmat haasteet myös liittyvät tähän – data ja sen esitysmuoto täytyy ymmärtää ja muokata sopivaksi, jotta sen voi syöttää AI-algoritmille opetustarkoituksessa.

Kun halutut tietolähteet, muokkaukset ja modulit ovat paikoillaan, suoritetaan mallin opetus ja arviointi (kuva 12, ”Run” painike alareunassa). Tämän jälkeen uusi välilehti, Predictive experiment, on käytettävissä



Kuva 12 Microsoft Azure Machine Learning Studio kuvakaappaus

Predictive experiment

Mallin opettamisen jälkeen toiselle välilehdelle ilmestyy Predictive experiment (kuva 12, ei-aktiivinen välilehti), ”ennustuskokeilu” joka on valmis työkulku data-analyysin tekemiseen ja sen voi halutessaan julkaista myös web-servicenä. Tällä välilehdellä malleja voidaan vertailla keskenään ja testiaineistoa vasten (jos koulutusdatasta on erotettu osa testiaineistoksi, kuten on tavallista ohjatussa oppimisessa). Azure ML Studio antaa hyvät valmiudet vertailun tekemiseen piirtämällä taulukko-datasta kuvaajia ilman käyttäjän omaa vaivannäköä. Tosin vakiovisualisoinneilla suuri määrä malleja on hankala esittää samalla näytöllä, samoin käyttöliittymä menee hieman sekavaksi jos useita malleja on jakamassa samoja tietolähteitä.

5.2.2 Scikit-learn

Scikit-learn asennettiin käytössä olleelle tietokoneelle Anacondan (Continuum analytics, 2017) avulla. Anaconda on Python jakelu, mahdollistaa Python ohjelmointikielen ja eri-

laisten data-analytiikkaan liittyvien kirjastojen asentamisen ja hallinnoinnin. Usean eri Python version pitäminen samalla koneella ja vaihtaminen niiden välillä on näppärää.

Anacondan asentaminen ja konfigurointi oli helppoa käyttäjälle, joka on ennenkin asentanut ohjelmointikieliä tietokoneelle. Jotain perustietoja on hyvä olla Pythonista ja data-analytiikkaan käytettävistä kirjastoista, jotta osaa valita sopivat versiot ja kirjastot. Dokumentaatiota on, mutta se ei ole läheskään samalla tasolla aloittelijan kannalta mitä vaikkapa Microsoftin vastaava. Anaconda tarjoaa myös maksullisia lisäosia datan analysointiin, mutta näitä ei nyt tarvittu eikä niiden hankkiminen ole pakollista. Anaconda on lisensoitu BSD-tyyppisellä (3-clause BSD) lisenssillä eikä siten rajoita käyttöä myöskään kaupallisessa mielessä. Anaconda on kuitenkin tässä vain jakelutapa, varsinaiset toiminnalliset osat ovat Python ja lukuisa joukko matematiikkakirjastoja joilla kaikilla on omat lisensointiehtonsa. Tärkeimmät näistä on esitetty taulukossa 3, jossa listataan kirjastot, niiden käyttämä lisenssi. Anacondan tarjoamien ja tässä käsitellyiden kirjastojen lisenssit ovat hyvin sallivia (tyypillisesti 3-lause BSD) eivätkä vaadi kaupallisen tuotteen tekijältä lähdekoodin jakamista.

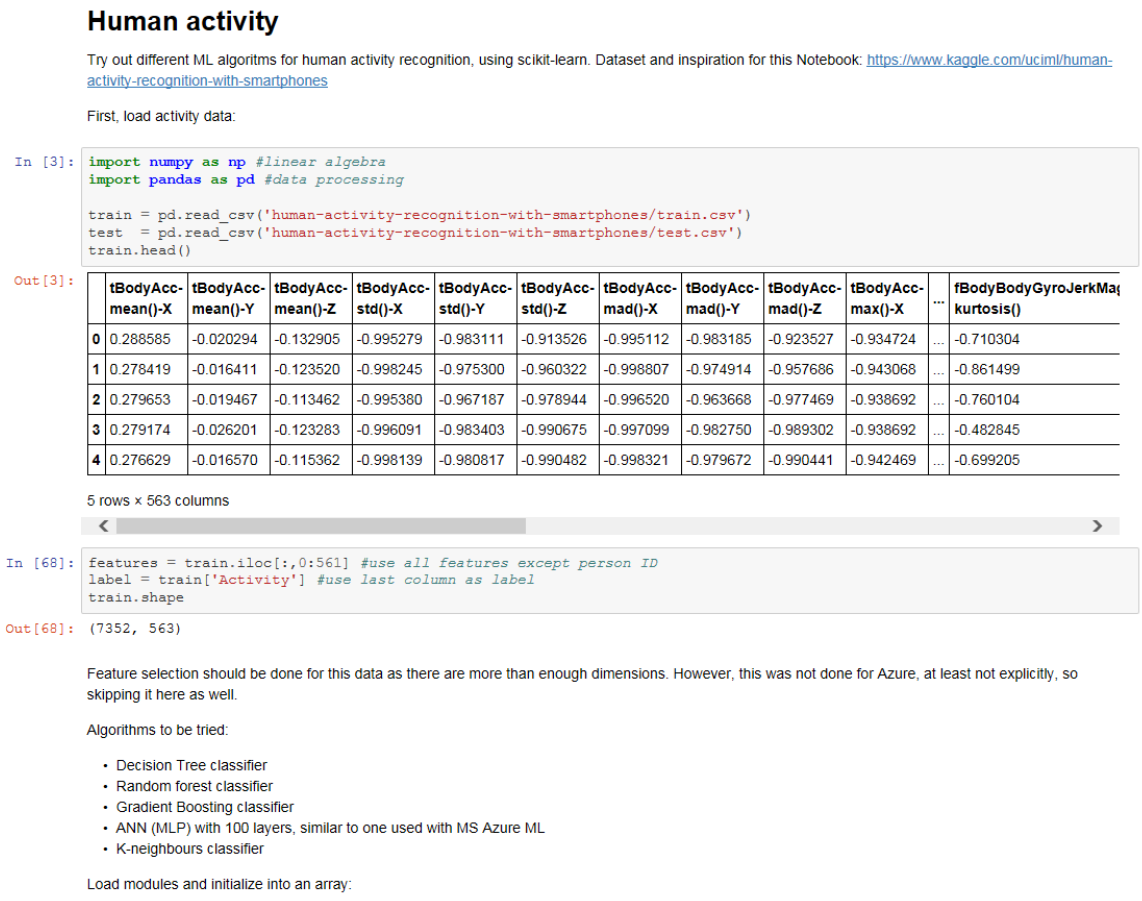
Taulukko 3 Anaconda-jakelun tärkeimpien ohjelmistojen lisenssit

Ohjelmisto	**Lisenssi**
Anaconda	3-clause BSD License
Python 2.7	Python Software Foundation License (PSFL)
Jupyter	3-clause BSD License
Scikit-learn	3-clause BSD License
Matplotlib	matplotlib license (PSFL-tyylinen)
TensorFlow	Apache 2.0

Kokonaisuutena edellä mainitut ohjelmistot luovat hyvin tehokkaan ja modernin työskentely-ympäristön joka ei ole samalla tavalla hiirellä operoitava ja valmiita työkulkuja sisältävä, mutta ohjelmointitaitoiselle käyttäjälle tämä on jopa luonnollisemman tuntuinen. Koko ohjelmointikielen ilmaisuvoima on käytettävissä koko ajan.

Jupyterin Notebook:ien avulla koodin, dokumentaation ja visualisoinnit voi yhdistää samaan kokonaisuuteen. Kuvassa 13 esitetään Human activity-ongelman ratkaisun alku, jossa ensin kerrotaan mitä ollaan tekemässä, esitetään koodi ja heti sen jälkeen koodin tuottama tuloste. Koodin muokkaaminen ja uudelleen ajaminen onnistuu milloin vain Notebookissa joten muokkaaminen on vaivatonta. Notebookin voi jakaa muille käyttäjille tai esittää webissä ja sen interaktiivinen luonne säilyy koko ajan, kunhan toisella käyttäjällä tai web-palvelimella on yhteensopivat Python-kirjastot asennettuna.

Notbookin käyttö ei ole kuitenkaan itsestään selvää ja vaatii dokumentaatioon ja esimerkkeihin perehtymistä. Erityisesti visualisointien tekeminen vaatii kohtalaista ymmärrystä matplotlibin käytöstä ja hieman luovuutta – ei ole mitään valmiita pohjia mistä valita vaan täytyy osata itse määrittää miten haluaa datan esitettävän.



Human activity

Try out different ML algoritms for human activity recognition, using scikit-learn. Dataset and inspiration for this Notebook: https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

First, load activity data:

In [3]:
```
import numpy as np #linear algebra
import pandas as pd #data processing

train = pd.read_csv('human-activity-recognition-with-smartphones/train.csv')
test  = pd.read_csv('human-activity-recognition-with-smartphones/test.csv')
train.head()
```

Out[3]:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMa(kurtosis()
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205

5 rows × 563 columns

In [68]:
```
features = train.iloc[:,0:561] #use all features except person ID
label = train['Activity'] #use last column as label
train.shape
```

Out[68]: (7352, 563)

Feature selection should be done for this data as there are more than enough dimensions. However, this was not done for Azure, at least not explicitly, so skipping it here as well.

Algorithms to be tried:

- Decision Tree classifier
- Random forest classifier
- Gradient Boosting classifier
- ANN (MLP) with 100 layers, similar to one used with MS Azure ML
- K-neighbours classifier

Load modules and initialize into an array:

Kuva 13 Jupyter notebook -kuvakaappaus

Kun data on muokattu sopivaan formaattiin ja halutut tekijät valittu, on se helppo syöttää useammalle algoritmille ja vertailla tuloksia keskenään. Eri algoritmien parametrien valinta ei ole triviaalia ja onkin täysin käyttäjän asiantuntemuksen varassa. Kuva 14 näyttää miten kokeilussa määritettiin käytettävät mallit ja niiden parametrit. Koska tarkoituk-

sena oli vertailla useita malleja, niitä määritetään kerralla isompi joukko. Oikeiden parametrien hakeminen vaatii ongelman tuntemista ja algoritmien toteutukseen perehtymistä sen verran, että tietää mitä parametreja on ja miten ne vaikuttavat algoritmin toimintaan.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier

Classifiers = [DecisionTreeClassifier(),
               RandomForestClassifier(n_estimators=200,
                                      n_jobs=-1),
               GradientBoostingClassifier(n_estimators=200),
               MLPClassifier(solver='sgd',
                             alpha=1e-4,
                             tol=1e-19,
                             hidden_layer_sizes=(100,),
                             random_state=1,
                             learning_rate_init=0.001,
                             max_iter=1000),
               KNeighborsClassifier(n_neighbors=20,
                                    n_jobs=2,
                                    weights='distance')]
```

Kuva 14 valmiiden mallien määrittäminen Scikit-learnissa

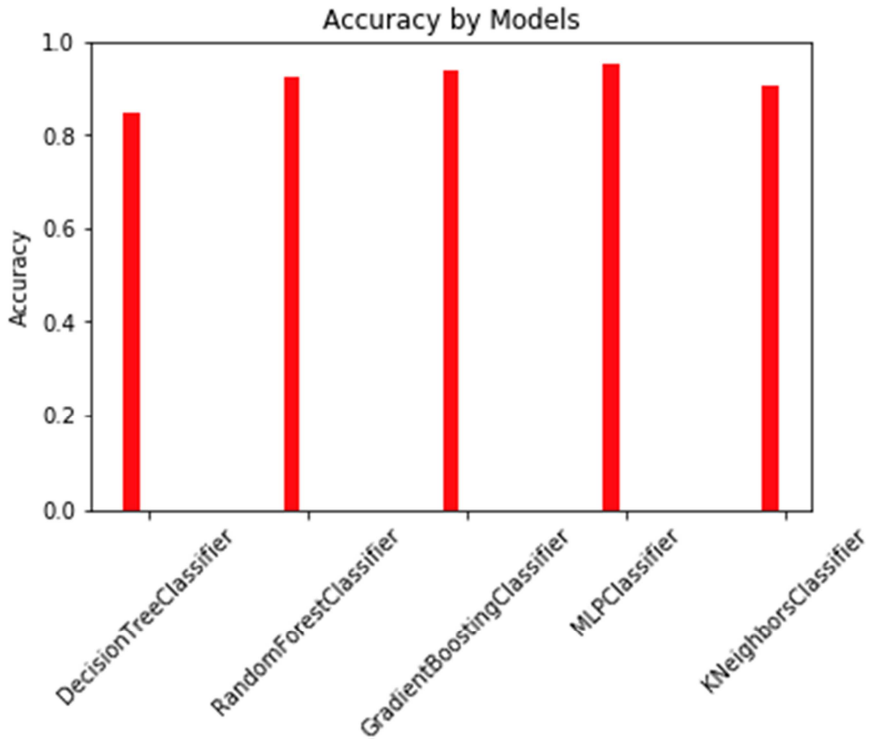
Vaihtoehtoja voi kokeilla helposti ja tuloksia vertailla useamman algoritmin kesken helposti yksinkertaisesti iteroimalla kaikki määritetyt mallit samalla datalla. Tämä on ehdottomasti Scikit-learnin vahvuus, samanlaisen iterointien teko graafisesti Azuressa ei onnistu (tietysti algoritmeja voi vaihdella ja näin saada samat tulokset ennen pitkää) ja jos jokaisen algoritmin joutuisi rakentamaan itse, olisi vertailu todella työlästä ja veisi paljon aikaa.

Kuva 15 näyttää edellä määriteltyjen mallien opettamisen. Suurin osa koodista liittyy metriikoiden laskemiseen ja tulostamiseen, nämä eivät ole pakollisia mutta toisaalta täysin välttämättömiä, jotta eri malleja voidaan vertailla keskenään, tai saman mallin eri parametrien vaikutusta.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import timeit
test_features= test.iloc[:,0:561]
Time_1=[]
Model_1=[]
Out_Accuracy_1=[]
for clf in Classifiers:
    start_time = timeit.default_timer()
    fit=clf.fit(features,label)
    pred=fit.predict(test_features)
    elapsed = timeit.default_timer() - start_time
    Time_1.append(elapsed)
    Model_1.append(clf.__class__.__name__)
    Out_Accuracy_1.append(accuracy_score(test['Activity'],pred))
    print(clf.__class__.__name__)
    print(classification_report(test['Activity'],pred))
```

Kuva 15 Scikit-learn mallien opettaminen

Kuvassa 16 nähdään matplotlibillä tehty visualisointi mallien ennustetarkkuuksista ja kuvassa 17 Python koodi, jolla kuvaajat tuotetaan. Matplotlibin käyttö ei ole vaikeaa, mutta ei myöskään täysin itsestään selvää. Data täytyy luonnollisesti järjestää (tässä tapauksessa Out_Accuracy_1) taulukkoon jotta se voidaan näyttää kätevästi. Kirjaston referenssiopas on varmasti tarpeen jos jotain monimutkaisempaa halutaan esittää.



Kuva 16 Scikit-learn opetettujen mallien ennustetarkkuus

```
In [*]: # plot performance metrics
        import matplotlib.pyplot as plt
        import numpy as np
        %matplotlib inline
        ind =  np.arange(5)   # the x locations for the groups
        width = 0.1       # the width of the bars
        fig, ax = plt.subplots()
        rects1 = ax.bar(ind, Out_Accuracy_1, width, color='r')
        ax.set_ylabel('Accuracy')
        ax.set_title('Accuracy by Models')
        ax.set_xticks(ind + width)
        ax.set_xticklabels(Model_1,rotation=45)
        plt.show()
```

Kuva 17 Scikit-learn koodi kuvaajien tuottamiseen

Scikit-learn tarjoaa modulin opetettujen mallien arviointiin (sklearn.metrics) ja kokeilussa tehty luokittelu oli helppo arvioida testijoukon avulla, sklearn.metrics.accurecy_score antoi tämän suoraan (arvojen kuvaajat kuvassa 16). Muitakin arviointifunktioita on tarjolla erityyppisille algoritmeille (luokittelu, regressio, ryvästys jne. tarvitsevat omansa) sekä vaikkapa metriikkaa joka kertoo tarkkuuden jokaista labelia kohden. Esimerkki tällaisesta on kuvassa 18 yhdelle luokittelijalle (DecisionTreeClassifier).

DecisionTreeClassifier

	precision	recall	f1-score	support
LAYING	1.00	1.00	1.00	537
SITTING	0.83	0.76	0.79	491
STANDING	0.79	0.85	0.82	532
WALKING	0.83	0.90	0.86	496
WALKING_DOWNSTAIRS	0.84	0.84	0.84	420
WALKING_UPSTAIRS	0.83	0.77	0.80	471
avg / total	0.86	0.86	0.86	2947

Kuva 18 Scikit-learn DecisionTreeClassifierin luokkakohtaisia metriikoita

5.2.3 TensorFlow

Scikit-learnia asentaessa tarjoutui mahdollisuus asentaa samaan Anaconda-ympäristöön myös TensorFlow, joten se myös tehtiin. Tämä mahdollisti myös näiden kahden vertailun mahdollisimman pitkälle samalla tavalla, samassa ympäristössä. Ana-

condan asennus ja käyttöönotto on kuvattu kappaleessa 5.2.2 ja tähän ympäristöön liittyvät lisenssit taulukossa 3. TensorFlow on Apache 2.0 lisensoitu (TensorFlow Authors, 2017). TensorFlow vaati tässä yhteydessä vain TensorFlow-paketin lisäämisen Anacondan paketteihin ja sen valitsemisen osaksi ympäristöä, joka on parin minuutin operaatio ympäristöjä asentaessa. Myös TensorFlown kokeilu tehtiin Jupyter Notebookissa ja TensorFlown Python API:a hyödyntäen.

TensorFlow'n määrittelyt tehtiin versiolla 0.12 ja sen jälkeen projekti on julkaissut 1.0:n sekä useita uudempia versioita. Muun muassa projekti on lisännyt korkean tason Estimator-API:n (Google, 2017) jonka avulla voi nopeasti määrittää käyttöönsä yleisimpiä luokittelu- ja regressiomalleja. Tämän API:n käyttö olisi analogista sille, mitä tehtiin kappaleessa 5.2.2 Scikit-learnin valmiilla malleilla.

Neuroverkon määrittely TensorFlowssa nähdään kuvassa 19. Verrattuna kuvassa 14 esitettyyn, täytyy nyt neuroverkko rakentaa osista ja samalla määritellä parametrit. Tämä antaa merkittäviä vapauksia käyttäjälle joka osaa ja tarvitsee tällaista – jos esimerkiksi haluaa käyttää sellaista verkon rakennetta, jota valmiiden mallien tekijät eivät tarjoa.

```
import tensorflow as tf

# set parameters
def weight_variable(shape):
    return tf.Variable(tf.truncated_normal(shape, stddev=0.1))

def bias_variable(shape):
    return tf.Variable(tf.truncated_normal(shape, stddev=0.1))

def add_layer(inputs, input_size, output_size, activation=None):
    W = weight_variable([input_size, output_size])
    b = bias_variable([output_size])
    wxb = tf.matmul(inputs, W) + b
    if activation:
        return activation(wxb)
    else:
        return wxb

# define layers
X = tf.placeholder(tf.float32, [None, 562])
layer1 = add_layer(X, 562, 1000, tf.nn.relu)
layer2 = add_layer(layer1, 1000, 300, tf.nn.relu)
layer3 = add_layer(layer2, 300, 50, tf.nn.relu)
output = add_layer(layer3, 50, 6)

y_ = tf.placeholder(tf.float32, [None, 6])

loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits = output,
    labels = y_))
optimizer = tf.train.GradientDescentOptimizer(0.001)
train_step = optimizer.minimize(loss)

correct = tf.equal(tf.argmax(output,1), tf.argmax(y_,1))
score = tf.reduce_mean(tf.cast(correct, "float"))

# init tensors
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)
```

Kuva 19 Neuroverkon määrittely TensorFlowssa (Asuru, 2016)

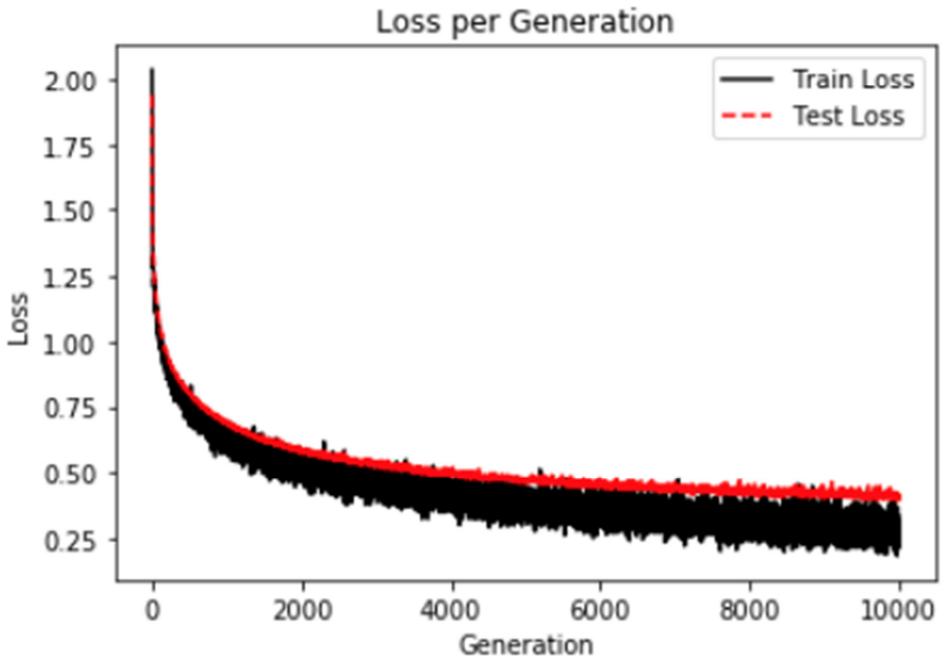
Määrittelyn jälkeen neuroverkko täytyy opettaa, samalla (tässä esimerkissä) kerätään statistiikkaa mallin oppimisesta iteraatioiden aikana. Koodi tämän toteuttamiseen on esitetty kuvassa 20.

```
import timeit
loss_vec = []
test_loss = []
train_score = []
tst_score = []

start_time = timeit.default_timer()
for i in range(10000):
    batch   = np.random.choice(train_features.shape[0], 100)
    # train
    sess.run(train_step,
             feed_dict = {X:train_features[batch],
                          y_:train_labels[batch]})
    # save training loss & score
    temp_loss = sess.run(loss,
                         feed_dict = {X:train_features[batch],
                                      y_:train_labels[batch]})
    loss_vec.append(np.sqrt(temp_loss))
    temp_score_train = sess.run(score,
                                feed_dict = {X:train_features[batch],
                                             y_:train_labels[batch]})
    train_score.append(temp_score_train)
    # run and save test-set loss & score
    tst_temp_loss = sess.run(loss,
                             feed_dict={X: test_features,
                                        y_: test_labels})
    test_loss.append(np.sqrt(tst_temp_loss))
    temp_score_tst = sess.run(score,
                              feed_dict={X: test_features,
                                         y_: test_labels})
    tst_score.append(temp_score_tst)

elapsed = timeit.default_timer() - start_time
print(elapsed)
```

Kuva 20 TensorFlow neuroverkon opetus

Esimerkki neuroverkon kustannusfunktion kehityksestä opetus-iteraatioiden aikana on kuvassa 21, tämä kuvaaja generoitiin matplotlibillä kuten kuvaajat Scikit-learning yhteydessä. Metriikoiden tuottaminen vaatii ainakin merkittävästi enemmän vaivannäköä kuin Scikit-learnissa. TensorBoard, joka mainittiin kappaleessa 5.1.6 auttaa visualisointien tekemisessä. Sen käyttö täytyy kuitenkin ottaa huomioon koodissa jotta TensorFlown tarvitsemat tiedot generoidaan ajonaikana. Tässä kokeilussa TensorBoardin käyttöönottoa ei tehty.

```
# plot performance metrics
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
plt.plot(loss_vec, 'k-', label='Train Loss')
plt.plot(test_loss, 'r--', label='Test Loss')
plt.title('Loss per Generation')
plt.xlabel('Generation')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.show()
```



Kuva 21 TensorFlow kustannusfunktion kehittyminen opetusiteraatioiden aikana

6 Tulosten käsittely

Tutkimuksen tulokset, analyysi ja toiminnalliset kokeilut, yhdistetään seuraavissa kappaleissa. Tässä yhteydessä otetaan huomioon myös alustan soveltuvuus Bittiumin liiketoimintaan ja sulautettuihin järjestelmiin yleensä.

6.1 Microsoft Azure Machine Learning

Kokeiluun käytettiin määräaikaista kokeilujaksoa, jossa sai käyttää MS Azuren palveluita 200 dollarin edestä ilman varsinaisia kustannuksia. Kokeilun aikana (muutamalla ajolla) tästä kului muutama dollari, enimmäkseen alustan perusmaksuihin. Tätä nykyä Microsoft tarjoaa myös aikarajattoman tilauksen ilmaiseksi, hieman rajoitetuilla ominaisuuksilla. Kustannusten seuranta on sinänsä helppoa ja tehty läpinäkyväksi käyttäjälle, mutta tämän kokeilun perusteella ei voida arvioida kuinka kallista tai halpaa MS Azuren käyttö olisi jollain toisella tietosarjalla ja ratkaistavalla ongelmalla.

MS Azure ML Studio oli miellyttävä käyttää, dokumentaatio oli erittäin hyvää ja havainnollista. Ihan ensimmäisten asetusten teko palvelua käyttöönottaessa vaati enemmän kärsivällisyyttä. Varsinkin jos tämä vaihe on mahdollista ulkoistaa, voi ”kuka hyvänsä” tehdä data-analyysiä Azurella. Alustan käyttö liiketoimintaan onnistuu tekniikan puolesta varmuudella, perusta toimii ja Azure ML Studio sallii myös laajentamisen R:llä ja Pythonilla. Lisäksi Azuressa on tarjolla myös kaikki muut Microsoftin pilven palvelut tukemaan sovelluksia.

Suurin haaste MS Azuren hyödyntämisessä on juuri tuo edellä mainittu, kaikki tapahtuu pilvessä ja kontrolli datasta menetetään. Tästä oli pohdintaa aikaisemmin IBM Watsonin yhteydessä kappaleessa 5.1.2. Microsoftilla on palvelinkeskuksia ympäri maailman, myös Euroopassa ja Aasiassa, mutta esimerkiksi USA:n viranomaisten pääsy tietoihin saattaa olla mahdollista konesalin sijainnista huolimatta. Microsoft luonnollisesti noudattaa myös kohdemaan lakeja.

Pilven keskeisestä roolista johtuen käyttö sulautetuissa järjestelmissä hankaloituu, lukuun ottamatta sellaisia käyttötapauksia missä yhteys Internetiin on mahdollista järjestää ja ratkaistava ongelma sallii prosessoinnin pilvessä. Esimerkiksi IoT-laitteiden, vaikkapa älykkäiden sähkömittareiden datan analysointiin, tällainen järjestely käy hyvin.

6.2 Scikit-learn

Scikit-learn on sallivasti lisensoitu koneoppimiskirjasto, joka tarjoaa valmiita malleja helpottamaan koneoppimisen opiskelua ja tutkimista. Mitään esteitä sen käytölle kaupallisessa tuotteessa IP-oikeuksien puolesta ei havaittu.

Alustan käyttöönotto Anacondan avulla oli vaivatonta ja ympäristön pystyy rakentamaan tarvittaessa uudestaan helposti.

Scikit-learn ei tue GPU-kiihdyttimiä joten sen suorituskyky on erityisesti neuroverkkojen opettamisessa rajallinen. Scikit-learn ei tue syväoppimista ja vahvistusoppimista. Tämän kokeilun osalta ja sen ”tietomassojen” kannalta normaalin kannettavan tietokoneen suorituskyky oli täysin riittävä – kertoo tietysti enemmän aineiston koosta (pieni) ja sovelluksen yksinkertaisuudesta (yksittäisten algoritmien testailua vs. oikean, reaalimaailman ongelmia ratkovan sovelluksen käyttöä).

Scikit-learnin käyttäminen sulautetussa järjestelmässä vaikuttaisi kuitenkin haastavalta, vakiona mallien hyödyntäminen vaatii Python-ympäristön joka on suhteellisen raskas vaikka mallien opettaminen ja kehittäminen voidaankin tehdä muualle. Scikit-learnin suurin vahvuus on helppo omaksuttavuus ja valmiit algoritmit, data-analyysiä pääsee tekemään hyvin nopeasti ilmaisuvoimaisen kielen ja siistin API:n kautta.

6.3 TensorFlow

TensorFlown käyttäminen testeihin osoittautui monimutkaisemmaksi ja vaati dokumentaation lukemista ja esimerkkeihin perehtymistä huomattavasti enemmän kuin vaikkapa Scikit-learn. Tämä oli täysin odotettavissa analyysin perusteella, mutta subjektiivinen kokemus vaikeusasteen erosta yllätti, TensorFlown vaatima työmäärä ei ole 2-3 kertainen vaan vähintään 5-10 kertainen (vaadittu ajankäyttö ja dokumenttien sekä kirjojen määrä).

Alustan lisensointi ei aseta rajoituksia kaupalliselle käytölle. Käyttöönotto oli yhtä vaivatonta kuin Scikit-learnin ja samaa jakelua (Anaconda) käytettiinkin molempiin. Muita ym-

päristöjä tai konfiguraatioita, kuten mobiilia, hajautettua laskentaa, C++ API:ia ei testattu, todennäköisesti niiden käyttöönotto on hankalampaa.

TensorFlown avulla voi rakentaa millaisen koneoppimisalgoritmin osaa ja haluaa, GPU-kiihdytys sekä hajautettu laskenta ovat natiivisti tuettuna. TensorFlow:n käyttö ja mallien kehittäminen onnistuu ilmaisuvoimaisella Pythonilla, mutta myös sulautettuihin järjestelmiin paremmin soveltuvat C++ ja Java API:t löytyvät. TensorFlow on tunnettu syväoppimisjärjestelmien alustana.

Jos ollaan tekemässä sulautettuun järjestelmään integroitavaa koneoppimista, TensorFlow tai vastaava matalantasonkirjasto on todennäköisesti paras vaihtoehto. Myös jos halutaan todenteolla hyödyntää syväoppimista tarjoaa TensorFlow siihen työkaluja. TensorFlowlla on toki kilpailijoita ja oli tässä tutkimuksessa kategorian edustajana, vertailua saman tason kirjastoihin ei vakavasti tehty.

7 Johtopäätökset

Alun perin lähdettiin hakemaan menetelmää sopivan keinoälyalustan valintaan Bittiumille, tietyille käyttötapauksille. Jo analyysin perusteella voitiin päätellä, että on monta teknologiaa ja toimittajaa jotka mahdollistavat käyttötapauksien toteutuksen ja jokaisella näistä on omat reunaehtonsa. Ehkä yksi yllättävä aspekti analyysivaiheessa oli, että moniin ongelmiin on jo nyt tarjolla valmiita tai lähes valmiita ratkaisuja pilvipalveluna (SaaS tai PaaS –tyyppisesti). Jos organisaatio on valmis lataamaan tietonsa pilveen ja sallii sovellukselle riippuvuuden Internet-yhteyteen, voidaan sovelluksia kehittää todella nopeasti ja pienellä alkuinvestoinnilla.

Riippuen sovelluskohteesta ovat jotkin palvelut todennäköisesti vahvempia kuin toiset. Esimerkiksi luonnollisten kielten käsittelyssä IBM Watson on vahvoilla. Jos taas halutaan valmiiksi opetettuja malleja ja näppärästi mukautuvia asiakaspalvelu-botteja, voi Microsoft Cognitive Services olla hyvä valinta. Tarjontaa on kuitenkin todella paljon ja haastajia näille isoille toimijoilla löytyy. Valinnan tekemiseksi täytyy ottaa huomioon myös hinnoittelu, lisensointi ja tietoturvaan/suojaan liittyvät asiat.

Jos julkiset pilvipalvelut joudutaan hylkäämään, ollaan hyvin nopeasti tekemisissä koneoppimiskirjastojen kanssa ja sovellusta kehittävä organisaatio joutuu investoimaan merkittävästi enemmän AI-asiantuntijoiden hankkimiseen, joko osaksi omaa henkilöstöä tai ulkoistettuna palveluna. Tämä sen lisäksi, että yrityksen täytyy myös ymmärtää itse ongelmasta ja koneoppimisen tarjoamista mahdollisuuksista. Risto Siilasmaan sanoin, ”Tarvitset ihmisiä joilta kysyä kun kohtaa merkittävän vastuksen ja mietit: Voisinko ratkaista tämän koneoppimisella?” (Siilasmaa, 2017)

Haluttuja käyttötapauksia voidaan toteuttaa monella eri teknologialla ja alustalla. Toisaalta ei kannata valita vain jotain tiettyä alustaa ja yrittää ratkaista sillä kaikkia ongelmia/käyttötapauksia. Keinoäly tietotekniikassa, kuten ihmisten taidot ja tiedot arkielämässä, ei ole mikään yksittäinen asia vaan kokonainen spektri erilaisia kapeita menetelmiä, hyvänä esimerkkinä Scikit-learnin kartta eri algoritmeista ja niiden valintaperusteista aikaisemmin (kuva 8). Näitä valitsemalla ja sopivasti yhdistelemällä voidaan luoda todellista lisäarvoa.

Tarkemmin tutkituista kolmesta alustasta (Microsoft Azure ML, Scikit-learn ja TensorFlow) kaikki kolme osoittautuivat käyttökelpoisiksi ja hyvin dokumentoiduiksi, ammattimaiseen käyttöön soveltuviksi alustoiksi. MS Azure oli näistä helpoin käyttää eikä vaadi

mitään ohjelmoinnilta näyttävää, ainakaan yksinkertaisemmissa tehtävissä. Scikit-learn ja TensorFlow vaativat molemmat vähintään perusteita ohjelmoinnista ja TensorFlown kohdalla, jos ei käytä valmiita Estimator-luokan malleja, kohtuullista osaamista myös koneoppimisen teoriasta ja sen soveltamisesta. Kaikille yhteistä on se, että data täytyy saada sellaiseen muotoon, että sen voi syöttää algoritmille. Lähes saman asian voi tehdä Pythonilla koodaamalla tai vetämällä ja pudottamalla MS Azuren käyttöliittymässä.

Lisensoinnin puolesta kaikki tutkitut alustat soveltuvat hyvin kaupalliseen käyttöön eivätkä aseta siinä mielessä mitään rajoitteita IP-oikeuksien osalta. Sen sijaan pilvipalveluihin liittyy potentiaalisia ongelmia datan hallinnan suhteen – voidaanko yrityksen tai asiakkaan arvokkainta omaisuutta (dataa) luovuttaa kolmannen osapuolen pilveen, joka toimii vielä pahimmassa tapauksessa toisen valtion lakien alla?

Tutkimatta jäi, täysin tietoisestikin, merkittäviä keinoälyalustoja ja kirjastoja sekä mm. Googlen pilvialustan palvelut. Näiden kaikkien läpikäynti vaatisi paljon aikaa ja toisaalta juuri nyt alan kehitys on niin nopeaa, että analyysien tulokset vanhenevat muutamassa kuukaudessa. Tätä raporttia kirjoittaessa moni analyysin kohteista on tuonut uusia ominaisuuksia markkinoille ja OpenSource-projektit ovat tehneet julkaisuja tuoden lisää toiminnallisuutta ja vikakorjauksia, TensorFlow-projektin tarjoamat rajapinnat olivat myös muuttuneet ja testikoodiin piti tehdä muutoksia jotta uudelleen ajo onnistui.

Jos tätä samaa aihetta tutkisi lisää ja haluaisi yksiselitteisiä, tai edes hieman paremmin sovellettavia vastauksia, täytyisi panostaa käyttötapausten tiukempaan määrittelyyn. Tämä mahdollistaisi teknologioiden valintaa pois sulkevalla tavalla. Tässä tutkimuksessa alussa määritellyt käyttötapaukset ehkä päinvastoin laajenivat kattamaan kaikki mahdolliset teknologiat mahdollisina ratkaisuina. Kun käyttötapaus on tiedossa voidaan järkevästi (kohtuullisella työmäärällä ja riittävän lyhyessä ajassa) myös vertailla horisontaalisesti samantyyppistä ratkaisua tarjoavia teknologioita.

Lähteet

Amazon. (20. lokakuu 2017). *Amazon Echo.* Haettu 20. lokakuu 2017 osoitteesta Amazon.co.uk: https://www.amazon.co.uk/gp/product/B06Y5ZW72J

Amazon. (2. joulukuu 2017). *AWS Deep Learning AMIs*. Haettu 2. joulukuu 2017 osoitteesta AWS: https://aws.amazon.com/machine-learning/amis/

Anguita, D.;Ghio, A.;Oneto, L.;Parra, X.;& Reyes-Ortiz, J. L. (24-26. April 2013). A Public Domain Dataset for Human Activity Recognition Using Smartphones. *21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* .

Apache Software Foundation. (tammikuu 2004). *Apache License.* Haettu 7. marraskuu 2017 osoitteesta https://www.apache.org/licenses/LICENSE-2.0

Apple Inc. (1. syyyskuu 2017). *FaceID Security.* Haettu 20. lokakuu 2017 osoitteesta https://images.apple.com/business/docs/FaceID_Security_Guide.pdf

Asuru, C. (27. joulukuu 2016). *Neural Network Classifier*. Haettu 01. tammikuu 2016 osoitteesta Kaggle.com: https://www.kaggle.com/casuru/neural-network-classifier

Bittium. (2017). *Bittium Tough Mobile*. Haettu 24. marraskuu 2017 osoitteesta Bittium: https://www.bittium.com/BittiumToughMobile

Bittium Oyj. (10. marraskuu 2016). *Stock Exchange & Press Releases 2016.* Haettu 2. syyskuu 2017 osoitteesta Bittium.com: https://www.bittium.com/index.php?id=1045&locate=PRM%2F2016%2F2359681

Buduma, N. (2017). *Fundamentals of Deep Learning.* O'Reilly Media.

Buitinck, L.;Louppe, G.;Blonder, M.;Pedregosa, F.;Mueller, A.;Grisel, O.;ym. (2013). API design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (ss. 108-122). European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases.

Continuum analytics. (30. elokuu 2017). *What is Anaconda?* Haettu 30. elokuu 2017 osoitteesta Anaconda: https://www.anaconda.com/what-is-anaconda/

Dunning, T.;& Friedman, E. (2014). *Practical Machine Learning: A New Look at Anomaly Detection.* O'Reilly Media.

Ferrucci, D. A. (4. joulukuu 2012). *This is Watson.* Haettu 4. joulukuu 2017 osoitteesta IBM developerWorks: https://www.ibm.com/developerworks/community/files/form/anonymous/api/library/1cf6a41d-fac9-478f-b3af-1c2bc3d59449/document/39ade3ad-8991-465a-826e-bfd39cd1e541/media/This%20is%20Watson%20white%20paper.pdf

Finlex. (29. joulukuu 2016). *Laki julkisista hankinnoista ja käyttöoikeussopimuksista.* Haettu 27. lokakuu 2017 osoitteesta Finlex: https://www.finlex.fi/fi/laki/ajantasa/2016/20161397#a1397-2016

Free Software Foundation. (29. kesäkuu 2007). *GNU General Public License.* Haettu 7. marraskuu 2017 osoitteesta https://www.gnu.org/licenses/gpl.html

Free Software Foundation. (29. kesäkuu 2007). *GNU Lesser General Public License.* Haettu 7. marraskuu 2017 osoitteesta https://www.gnu.org/licenses/lgpl.html

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* O'Reilly Media.

Google. (24. elokuu 2017). *About TensorFlow*. Haettu 24. elokuu 2017 osoitteesta TensorFlow: https://www.tensorflow.org

Google. (16. lokakuu 2017). *Google Home.* Haettu 20. lokakuu 2017 osoitteesta Google Store: https://store.google.com/uk/product/google_home

Google. (17. elokuu 2017). *TensorFlow API documentation.* Haettu 13. syyskuu 2017 osoitteesta TensorFlow: https://www.tensorflow.org/api_docs/python/tf/estimator

Google. (26. elokuu 2017). *TensorFlow Architecture.* Haettu 26. elokuu 2017 osoitteesta TensorFlow: https://www.tensorflow.org/versions/r1.0/images/layers.png

Hawkings, S.;He, H.;Williams, G.;& Baxter, R. (2002). Outlier detection using replicator neural networks. *DaWaK* (ss. 170-180). Springer.

IBM. (23. elokuu 2017). *Products and services*. Haettu 23. elokuu 2017 osoitteesta IBM Watson: https://www.ibm.com/watson/developercloud/services-catalog.html

IBM. (21. lokakuu 2016). *What is Watson*. Haettu 21. lokakuu 2016 osoitteesta Watson: http://www.ibm.com/watson/what-is-watson.html

ISO. (13. lokakuu 2017). *Information technology — Security techniques — Information security management systems — Requirements.* Haettu 13. lokakuu 2017 osoitteesta Online Browsing Platform (OBP): https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en

James, H.;& Alexei, A. E. (2007). Scene Completion Using Millions of Photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)* .

Kaggle.com. (2016). *Human Activity Recognition with Smartphones*. Haettu 01. 01 2017 osoitteesta Kaggle: https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

Lichman, M. (2013). *UCI Machine Learning Repository*. (University of California, Irvine, School of Information and Computer Sciences) Haettu 01. 01 2017 osoitteesta https://archive.ics.uci.edu/ml/index.php

Lighthill, J. (1973). Artificial intelligence: a paper symposium. *Science Research Council, London* .

McDermott, D. (1982). A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science* , 101-155.

Microsoft. (23. elokuu 2017). *Azure Machine Learning.* Haettu 23. elokuu 2017 osoitteesta Microsoft Azure: http://download.microsoft.com/download/0/5/A/05AE6B94-E688-403E-90A5-6035DBE9EEC5/machine-learning-basics-infographic-with-algorithm-examples.pdf

Microsoft. (20. elokuu 2017). *Cognitive Services*. Haettu 20. elokuu 2017 osoitteesta Microsoft Azure: https://azure.microsoft.com/en-us/services/cognitive-services/?v=17.29

Microsoft. (27. elokuu 2017). *Create and share an Azure Machine Learning workspace.* Haettu 27. elokuu 2017 osoitteesta Microsoft Azure: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-create-workspace

Nilsson, N. J. (1998). *Introduction to Machine Learning.* Haettu 20. elokuu 2017 osoitteesta Artificial Intelligence Laboratory Department of Computer Science Stanford University: https://ai.stanford.edu/~nilsson/mlbook.html

Open Source Initiative. (ei pvm). *The MIT License.* Haettu 7. marraskuu 2017 osoitteesta https://opensource.org/licenses/MIT

Pacheco, E. R. (2013). *Unsupervised Learning with R.* Packt Publishing.

Paperspace. (27. elokuu 2017). *Running TensorFlow + TensorBoard on a GPU+.* Haettu 27. elokuu 2017 osoitteesta Paperspace.com: https://paperspace.zendesk.com/hc/en-us/articles/235203648-Running-TensorFlow-TensorBoard-on-a-GPU-

Pedregosa, F.;Varoquaux, G.;Gramfort, A.;Michel, V.;Thirion, B.;Grisel, O.;ym. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* (12), 2825-2830.

Prahalad, C. K. (1999). The core competence of the corporation. *Knowledge and strategy* , 41-59.

Regents of the University of California. (22. heinäkuu 1999). Haettu 7. marraskuu 2017 osoitteesta BSD ohjelmistojakelu: ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change

Russell, S.;& Norvig, P. (2016). *Artificial Intelligence: A Mordern Approach, Global Edition.* Pearson Education Limited.

Scikit-learn developers. (24. elokuu 2017). *Choosing the right estimator.* Haettu 24. elokuu 2017 osoitteesta scikit-learn: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-learn developers. (26. elokuu 2017). *COPYING*. Haettu 26. elokuu 2017 osoitteesta Github: https://github.com/scikit-learn/scikit-learn/blob/master/COPYING

Sigaud, O.;& Buffet, O. (2013). *Markov Decision Processes in Artificial Intelligence.* John Wiley & Sons.

Siilasmaa, R. (9. marraskuu 2017). *Why you should study AI and Machine Learning and how I did it*. Haettu 3. joulukuu 2017 osoitteesta Nokia: https://blog.networks.nokia.com/innovation/2017/11/09/study-ai-machine-learning/

Strang, J. (20. marraskuu 2012). *File:Polarlicht 2 kmeans 16 large.png.* (K. o. Hearts, Toim.) Haettu 6. joulukuu 2017 osoitteesta Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Polarlicht_2_kmeans_16_large.png

Sutton, R. S.;& Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press.

TensorFlow Authors. (24. tammikuu 2017). *tensorflow/LICENSE*. Haettu 3. joulukuu 2017 osoitteesta GitHub: https://github.com/tensorflow/tensorflow/blob/master/LICENSE

The Open Source Initiative (OSI). (22. maaliskuu 2007). *The Open Source Definition*. Haettu 4. joulukuu 2017 osoitteesta Open Source Initiative: https://opensource.org/osd

TIEKE. (20. lokakuu 2016). *Tietotekiikkahankinnat, lähdeluettelo ja muita hyviä linkkejä.* Haettu 20. lokakuu 2016 osoitteesta Tietotekniikkahankinnat: https://www.tieke.fi/pages/viewpage.action?pageId=3441268

Wheeler, D. A. (11. huhtikuu 2003). *Program Library HOWTO*. Haettu 22. marraskuu 2017 osoitteesta TLDP.org: http://tldp.org/HOWTO/Program-Library-HOWTO/index.html