Juha Leivo

Koneoppimismallin kouluttaminen ja tuotantoon vieminen MLOps-käytänteiden mukaisesti Microsoft Azure -pilvipalvelussa

Insinööri (AMK)
Tieto- ja viestintätekniikka
Kevät 2023

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Juha Leivo

Työn nimi: Koneoppimismallin kouluttaminen ja tuotantoon vieminen MLOps-käytänteiden mukaisesti Microsoft Azure -pilvipalvelussa.

Tutkintonimike: Insinööri (AMK), Tieto- ja viestintätekniikka

Asiasanat: MLOps, Koneoppiminen, Azure Machine Learning, Azure DevOps

Tässä opinnäytetyössä tutustuttiin, kuinka koneoppimismalli saadaan vietyä tuotantoympäristöön toimintavarmasti ja nopeasti. Koneoppimisprojektin elinkaareen kuuluu useita eri työvaiheita, jotta koneoppimalli saadaan luotua. Työvaiheet eivät kuitenkaan etene suoraviivaisesti, sillä useat eri muuttujat vaikuttavat siihen, kuinka hyvin koneoppimismalli pystyy ratkaisemaan sille määritetyn ongelman. Lisäksi koneoppimismallin suorituskyky saattaa laskea ajan saatossa, koska se on koulutettu jo olemassa olevalla datalla, joka ei välttämättä enää täysin kuvasta nykyhetkeä. Useat eri työvaiheet, iteratiivisuus sekä koneoppimismallin suorituskyvyn lasku tekevät koneoppimismallin tuotantoon viemisestä ja sen toimintakunnon ylläpitämisestä haastavan tehtävän.

MLOps on kokoelma hyviä käytänteitä, joita seuraamalla koneoppimismalli saadaan vietyä tuotantoympäristöön entistä tehokkaammin. Sen peruspilareihin kuuluvat automatisointi ja monitorointi, joiden avulla koneoppimismalli saadaan myös koulutettua uudelleen, kun se ei enää kykene suoriutumaan muuttuvassa maailmassa tarpeeksi hyvin. Työssä myös esitettiin, minkälaisia työkaluja ja palveluita Microsoft Azure -pilvipalvelu tarjoaa koneoppimisprojekteille, kun ne toteutetaan MLOps-periaatteita noudattaen. Opinnäyteyön lopuksi demonstrointiin, kuinka Azuren palveluita hyödyntäen voidaan luoda yksinkertainen MLOps-putki.

Abstract

Author(s): Juha Leivo

Title of the Publication: Training and Deploying a Machine Learning Model Following MLOps Practices on the Microsoft Azure Cloud Platform.

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: MLOps, Machine Learning, Azure Machine Learning, Azure DevOps

The purpose of this thesis is to introduce how a machine learning model can be brought into the production environment reliably and quickly. The life cycle of a machine learning project includes several different work phases to create a machine learning model. However, the work steps do not proceed in a straight line, as several different variables affect how well the machine learning model can solve the problem assigned to it. In addition, the performance of the machine learning model may decrease over time, because it has been trained with already existing data, which may no longer fully reflect the present moment. Several different work steps, iterativeness and the decrease in performance of the machine learning model make bringing the machine learning model into production and maintaining its operational condition a challenging task.

MLOps is a collection of good practices, by following which the machine learning model can be brought to the production environment even more efficiently. Its basic pillars include automation and monitoring, with the help of which the machine learning model can also be trained again when it is no longer able to perform well enough in the changing environment. This thesis also shows what kind of tools and services the Microsoft Azure cloud service offers for machine learning projects when they are implemented following MLOps principles. At the end of the thesis, it is demonstrated how a simple MLOps-pipeline can be created using Azure Machine Learning and DevOps services.

Sisällys

1 Johdanto 1

2 Koneoppimisprojekti 2

2.1 Koneoppimisprojektin elinkaari 2

2.1.1 Projektin määrittäminen 3

2.1.2 Datan käsittely 4

2.1.3 Koneoppimismallin kouluttaminen 5

2.1.4 Koneoppimismallin tuotantoon vieminen 6

2.2 Työntekijät osana koneoppimisprojektia 7

3 MLOps 9

3.1 Keskeiset käytänteet 9

3.1.1 Automatisointi 10

3.1.2 Toistettavuus ja versiointi 13

3.1.3 Testaaminen 14

3.1.4 Monitorointi 15

3.2 Yleisesti käytettyjä teknologioita 17

3.2.1 Konttiteknologiat 17

3.2.2 Mikropalveluarkkitehtuuri 18

3.2.3 Kubernetes 19

3.2.4 CI/CD 20

4 Microsoft Azure 22

4.1 Azure Machine Learning 22

4.1.1 Resurssit 23

4.1.2 Assetit 24

4.2 Azure DevOps 24

5 MLOps-putken luominen Microsoft Azure -pilvipalvelussa 26

5.1 Palveluyhteydet 26

5.2 Putkien suoritusympäristö 27

5.3 CI-putki 29

5.3.1 Koodien testaaminen 29

5.3.2 Koneoppimismallin kouluttaminen 30

5.3.3 Koneoppimismallin rekisteröinti 32

5.3.4 Artefaktin luominen 34

5.4 CD-putki 35
5.4.1 Koneoppimismallin julkaiseminen ja testaaminen staging-ympäristössä 35
5.4.2 Koneoppimismallin julkaiseminen tuotantoympäristöön 37
5.4.3 Monitorointi 38

6 Yhteenveto 40

Lähteet 42

Liitteet

Termit ja lyhenteet

AML	Lyhenne sanoista Azure Machie Learning. Microsoftin luoma pilvipalvelu, joka tarjoaa työkaluja koneoppimisprojektien toteuttamiseen.
CLI	Lyhenne englanninkielisistä sanoista Command-Line Interface. Mahdollistaa tekstipohjaisten komentojen syöttämisen tietokoneelle.
MLOps	Lyhenne englanninkielisistä sanoista Machine Learning and Operations. Sateenvarjotermi hyville käytänteille, joita suositellaan seurattavan koneoppimisprojekteissa.
SDK	Tulee sanoista Software Development Kit. Ohjelmistokehityspaketti, joka sisältää kokoelman eri työkaluja.
Datasetti	Kokoelma tietoa, jota voidaan käyttää esimerkiksi koneoppimismallin kouluttamiseen.
Hyperparametri	Parametri, jolla voidaan vaikuttaa koneoppimismallin oppimiseen. Eri koneoppimismalleilla on erilaisia hyperparametrejä.
Koneoppimisalgoritmi	Matemaattinen menetelmä, joka kykenee itsenäisesti oppimaan säännöt tietyn ongelman ratkaisemiseksi datan avulla.
Koneoppimismalli	Datan avulla koulutettu koneoppimisalgoritmi.
Putki	Kokoelma työvaiheita, jotka suoritetaan ennalta määrätyssä järjestyksessä.
Raakadata	Alkuperäislähteestä haettua dataa, jota ei ole esikäsitelty millään tavalla.
Strukturoimaton data	Dataa, jolla ei ole ennalta määriteltyä rakennetta. Esimerkiksi kuva- ja äänitiedostot ovat strukturoimatonta dataa.

Strukturoitu data	Dataa, jolla on ennalta määritelty rakenne. Tyypillisesti taulukkomuotoista dataa, joka voidaan tallentaa relaatiotietokantoihin.
Tietoallas	Englanniksi Data Lake. Tyypillisesti pilvipalvelussa sijaitseva arkisto, johon tallennetaan strukturoitua tai strukturoimatonta raakadataa.
Tietovarasto	Englanniksi Data Warehouse. Keskitetty varastointipaikka, johon tyypillisesti useasta eri lähteestä haettu ja esikäsitelty strukturoitu data tallennetaan.

1 Johdanto

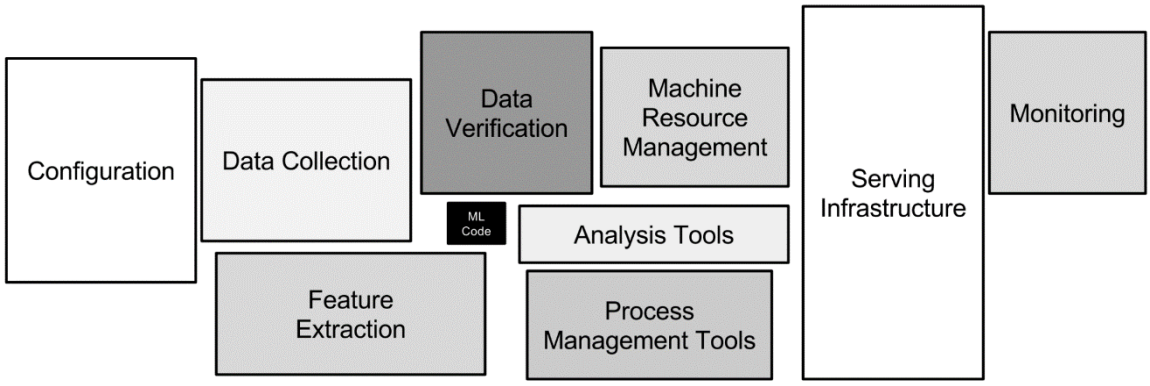
Koneoppimista hyödynnetään jatkuvasti yhä enemmän eri toimialoilla. Koneoppimismallien hyödyntäminen on kuitenkin osoittautunut haastavaksi, sillä ne tarvitsevat monimutkaisen ympäristön toimiakseen luotettavasti myös pitkällä aikavälillä. Tässä opinnäytetyössä tutustutaan, mitä eri työvaiheita koneoppimismallin luomiseen kuuluu sekä minkälaisia käytänteitä ja työkaluja hyödyntämällä koneoppimismalli saadaan päivitettyä tehokkaasti, kun sen suorituskyky ei vastaa enää tarvittavaa tasoa.

Koneoppimisprojekti koostuu useista eri työvaiheista, joiden toteuttamiseen tarvitaan niihin erikoistuneita työntekijöitä. Tarpeeksi hyvin toimivan koneoppimismallin luominen vaatii useiden eri koneoppimisalgoritmien, hyperparametrien sekä datasettien käyttämistä. MLOps on kokoelma hyviä käytänteitä, joita seuraamalla voidaan edesauttaa eri työvaiheista vastaavien työntekijöiden välistä yhteistyötä sekä tehdä koneoppimismallin kouluttamisesta sekä tuotantoon viemisestä helposti hallittava, luotettava sekä tehokas prosessi.

Tämä opinnäytetyö on toteutettu omasta mielenkiinnosta aihetta kohtaan. MLOps on suhteellisen nuori toimintatapa, johon liittyvät työkalut ovat vielä jatkuvan kehitystyön alaisena. Tässä työssä tutkitaan, miten Microsoft Azure -pilvipalvelun tarjoamien työkalujen ja palveluiden avulla koneoppimismalli saadaan koulutettua ja vietyä tuotantoympäristöön MLOps-periaatteita noudattaen.

2 Koneoppimisprojekti

Koneoppimisprojektin tarkoituksena on luoda koneoppimismalli ongelman ratkaisemiseksi ja saada se toimimaan tuotantoympäristössä [1]. Koneoppimismallin luominen ja kouluttaminen on kuitenkin vain hyvin pieni osa onnistunutta koneoppimisprojektia (Kuva 1) [2]. Suuri osa koneoppimisprojekteista epäonnistuu, koska usein koneoppimisprojektien alkuvaiheilla keskitytään ainoastaan toimivan koneoppimismallin luomiseen, eikä oteta huomioon, millaisen ympäristön koneoppimismalli tarvitsee toimiakseen luotettavasti myös pitkällä aikavälillä [1].



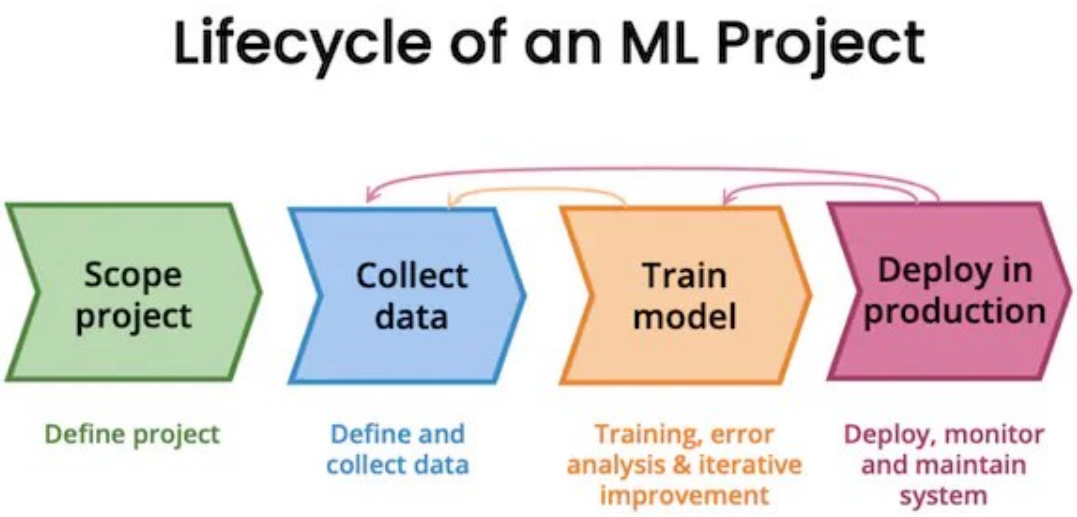
Kuva 1. Koneoppimismalliin liittyvät koodit (musta laatikko keskellä) ovat vain pieni osa koneoppimisprojektia [2].

2.1 Koneoppimisprojektin elinkaari

Koneoppimisprojektit voivat keskenään olla hyvin erilaisia ja tämän takia ei ole olemassa yhtä toimintamallia, joka täsmällisesti kuvaisi niiden kaikkien elinkaarta. Koneoppimisprojekteja kuitenkin yhdistää pääpiirteittäin samat työvaiheet.

Koneoppimisprojekteja yhdistää myös niiden iteratiivinen luonne, eli samoja työvaiheita toisestaan moneen kertaan [3]. Iteratiivisuus johtuu useimmiten siitä, että ymmärrys ratkaistavasta ongelmasta, datan sisällöstä ja koneoppimismallin toimivuudesta lisääntyvät projektin edetessä. Esimerkiksi koneoppimismallin koulutusvaiheessa voidaan huomata, että dataa tuleekin muokata jollakin toisella tavalla tai sitä pitää hankkia lisää, jolloin syntyy tarve palata takaisin datan käsittelyvaiheeseen. Käytettävää koneoppimisalgoritmia ja sen hyperparametrejä voidaan myös joutua vaihtamaan useaan otteeseen, ennen kuin tarpeeksi hyvin toimiva koneoppimismalli saadaan

koulutettua. Lisäksi koneoppimismallit koulutetaan menneisyydessä kerätyllä datalla, mikä johtaa siihen, että koneoppimismalli tulee kouluttaa uudelleen, mikäli tuotantoympäristössä oleva data alkaa poikkeamaan liikaa koulutuksessa käytetystä datasta. Iteratiivisuutta tapahtuu siis usealla eri tasolla (Kuva 2), mikä tekee koneoppimismallien tuotantoon viemisestä hidasta sekä hankaloittaa niiden pitämistä toimintakuntoisena.



Kuva 2. Yleiskuvaus koneoppimisprojektien elinkaaresta [3].

2.1.1 Projektin määrittäminen

Projektin määrittämien on projektin ensimmäinen ja onnistumisen kannalta olennainen vaihe. Tässä vaiheessa on tärkeää, että siihen osallistuvat koneoppimisprojektista vastaavat tekniset osaajat sekä projektin tilaavan puolen asiantuntijat. Tarkoituksena on, että kummallakin osapuolella on yhteisymmärrys, miksi projekti halutaan toteuttaa ja mitä kaikkea sen toteuttamiseen vaaditaan. [4.]

Esimerkiksi seuraavia kysymyksiä voidaan esittää projektin määrittämiseksi [5]:

- Miksi projekti halutaan toteuttaa? Projektilla on oltava jokin konkreettinen tarve. Koneoppimisen avulla voidaan esimerkiksi vähentää kustannuksia optimoimalla erilaisia prosesseja tai kasvattaa tuottoa luomalla suosittelusysteemejä asiakkaille.

- Minkälainen on projektin lopputuote? Projektin alkuvaiheilla on hyvä tietää, miten koneoppimismallia on tarkoitus käyttää. Tämän tiedon pohjalta voidaan suunnitella, millainen käyttöympäristö ja käyttöliittymä koneoppimismallille on tarpeen luoda.

- Kuinka projektin onnistumista ja lopullisen tuotteen tuomaa lisäarvoa mitataan? Koneoppimisella voidaan ratkoa monia erilaisia ongelmia. Projektin alkuvaiheilla on hyvä päättää, mitkä ovat niitä mittareita, jotka ohjaavat projektin etenemistä oikeaan suuntaan.

- Onko tarvittavaa dataa saatavilla? Yrityksille on usein kertynyt paljon dataa useista eri lähteistä. Tyypillisesti dataa tarvitsee yhdistää ja muokata, jotta sitä voidaan käyttää koneoppimismallien kouluttamiseen. Ennen projektin aloittamista tulee tietää, missä data sijaitsee, missä muodossa se on ja onko sitä varmasti lupa käyttää. Joskus vastaan voi esimerkiksi tulla tilanteita, joissa dataa ei ole tarpeeksi tai ollenkaan, jolloin pitää selvittää, onko sitä mahdollista hankkia esimerkiksi ulkopuolisilta tahoilta.

- Mitä osaamista projektin toteuttamiseen tarvitaan ja ketkä siihen osallistuvat? Koneoppimisprojektien toteuttamiseen tarvitaan osaamista useilla eri osa-alueilla. Ennen projektin aloittamista kannattaa selvittää, onko tarvittavaa osaamista jo olemassa vai pitääkö työvoimaa hankkia lisää.

2.1.2 Datan käsittely

Kaikkia koneoppimisprojekteja yhdistää data. Datan käsittelyvaiheen tarkoituksena on varmistaa, että data on oikeassa muodossa, tarpeeksi hyvälaatuista ja helposti saatavilla koneoppimismallin opettamista varten. Monesti dataa on tarpeen hakea useista lähteistä ja se voi olla monessa eri muodossa. Datan rakenne ja sijainti määrittävät hyvin pitkälti, miten dataa on tarpeen käsitellä. Esittelen seuraavaksi yleisimmin käytettyjä menetelmiä ja työvaiheita, joita käytetään datan käsittelyvaiheessa.

ETL (tulee englanninkielisistä sanoista Extract, Transform ja Load) on menetelmä, jossa useasta eri lähteestä haettu data yhdistetään, muokataan tarpeen mukaan ja tallennetaan esimerkiksi tietovarastoon, josta se voidaan helposti hakea analysoimista tai koneoppimismallin kouluttamista varten. Se on alun perin kehitetty käytettäväksi strukturoidulle datalle, mutta nykyisin sitä käytetään myös strukturoimattomalle datalle. Menetelmää käytetään silloin, kun tiedetään, mitä kaikkea dataa tarvitaan ja missä muodossa sen pitää olla. [6.]

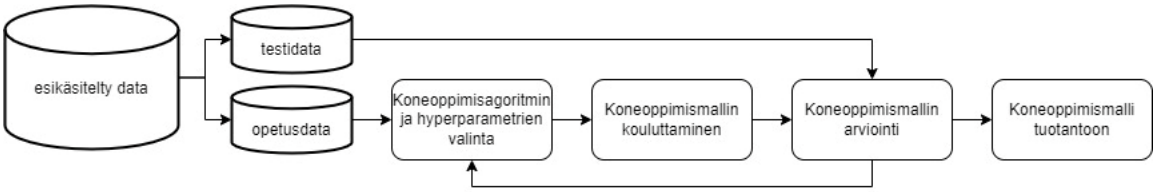
ELT on ETL:n kaltainen menetelmä, mutta siinä kaksi viimeistä vaihetta toteutetaan eri järjestyksessä. Sitä käytetään pääsääntöisesti strukturoimattoman datan siirtämiseen useasta eri lähteestä yhteen kohteeseen. Usein kerätty data varastoidaan tietoaltaaseen, josta se on nopeasti saatavilla jatkotoimenpiteitä varten. ELT-menetelmässä dataa ei muokata ennen varastointia, vaan muokkaus tehdään varastoinnin jälkeen tarpeen mukaan. ETL-menetelmää käytetään silloin, kun ei ole varmaa, mikä kaikki data on tarpeellista, eikä sille ole määritelty tarkkaa muotoa. [7.]

EDA (tulee englanninkielisistä sanoista Exploratory Data Analysis) eli tutkiva data-analyysi on työvaihe, jonka aikana datan sisältöä pyritään ymmärtämään esimerkiksi kuvaajien ja tilastollisten menetelmien avulla. Datasta voidaan myös etsiä säännönmukaisuuksia, poikkeavuuksia tai piirteiden välisiä suhteita. [8.] Tässä vaiheessa on kannattavaa tehdä yhteistyötä dataa syvemmin ymmärtävien asiantuntijoiden kanssa, jotka osaavat usein selventää, mitä datasta löydetyt havainnot käytännössä tarkoittavat. Datan syvempää ymmärtämistä voidaan hyödyntää esimerkiksi piirresuunnittelussa tai datan esikäsittelyvaiheessa tehtävien toimenpiteiden määrittämiseksi.

Piirresuunnittelun lopputuotoksena on koneoppimismallin kouluttamisessa käytettävä datasetti. Tässä työvaiheessa esikäsitellystä datasta poistetaan kaikki epäolennaiset piirteet ja se muokataan koneoppimismallille sopivaan muotoon. Olemassa olevista piirteistä voidaan myös luoda entistä paremmin ongelmaa selittäviä piirteitä [9]. Piirresuunnittelun tarkoituksena on siis muokata dataa siten, että se sisältää mahdollisimman paljon informaatiota, jonka avulla koneoppimismallin on helppo oppia ratkaisemaan ongelma.

2.1.3 Koneoppimismallin kouluttaminen

Esimerkiksi Python-ohjelmointikielelle on olemassa useita koneoppimisalgoritmeja tarjoavia kirjastoja, joiden avulla koneoppimismallien luominen ja kouluttaminen on verrattain helppoa. Mahdollisimman hyvin toimivan koneoppimismallin luominen ei kuitenkaan ole yksinkertainen tehtävä. Koneoppimismallin kouluttamiseen kuuluu useita eri vaiheita (Kuva 3), joiden avulla pyritään varmistamaan, että koneoppimismalli toimii koulutuksessa käytetyn datan lisäksi myös sille täysin ennennäkemättömällä datalla.



Kuva 3. Koneoppimismallin kouluttamisen valmiiksi esikäsitellylle datalle.

Tyypillisesti koulutuksessa käytettävä datasetti jaetaan koulutuksen aikana käytettäväksi opetusdataksi ja koulutuksen jälkeen koneoppimismallin toimivuuden arvioimiseen käytettäväksi testidataksi (Kuva 3). Esimerkiksi neuroverkkojen kanssa käytetään yleensä myös validointidataa, joka mahdollistaa koneoppimismallin arvioimisen jo koulutusvaiheessa. Esikäsitelty data on hyvä jo ennalta jakaa käytettäviin datasetteihin, jotta opetettujen koneoppimismallien arvioinnit ovat vertailukelpoisia.

Ennen koneoppimismallin kouluttamista on valittava ongelmaan sopiva koneoppimisalgoritmi ja löydettävä sille parhaiten toimivat hyperparametrit. Koneoppimismallin valintaan vaikuttavat esimerkiksi, minkä tyyppistä ongelmaa ollaan ratkaisemassa, datan ja piirteiden määrä sekä käytettävissä olevat laskentaresurssit. [10.]

Koneoppimismallin koulutuksen jälkeen on tarpeen arvioida, kuinka hyvin se toimii. Jos koneoppimismalli ei pysty ratkaisemaan ongelmaa tarpeeksi hyvin, voidaan sen hyperparametreja hienosäätää tai valita kokonaan toinen koneoppimisalgoritmi koulutettavaksi [11]. Koneoppimisprojekteissa on useita muuttujia, jotka vaikuttavat siihen, kuinka hyvin koulutettu koneoppimismalli toimii. On helppoa jäädä testaamaan, kuinka yksittäiset pienet muutokset vaikuttavat lopputulokseen. Koulutusvaiheeseen ei kuitenkaan kannata jäädä jumittamaan, vaikka malli ei vielä toimisikaan täydellisesti. Projektin alkuvaiheilla voidaan esimerkiksi kouluttaa jokin yksinkertainen koneoppimisalgoritmi, jonka tiedetään toimivan hyvin kyseisen ongelmatyypin ratkaisemiseen. Projektin edetessä tätä mallia voidaan käyttää vertailukohtana, kun testataan, kuinka monimutkaisemmat ja tarkemmin hienosäädetyt koneoppimismallit toimivat.

2.1.4 Koneoppimismallin tuotantoon vieminen

Koneoppimismalli on tärkeää saada julkaistua tuotantoympäristössä mahdollisimman nopeasti, jotta sillä saadaan tuotettua sitä lisäarvoa, jonka takia se on alun perinkin päätetty luoda [12].

Vasta tämän jälkeen koko prosessia kannattaa lähteä parantamaan askel kerrallaan, kun tiedetään, mitä kaikkea toimivan koneoppimismallin tuotantoon viemiseen vaaditaan, ja nähdään, kuinka koneoppimismalli toimii reaalimaailmassa. [3.]

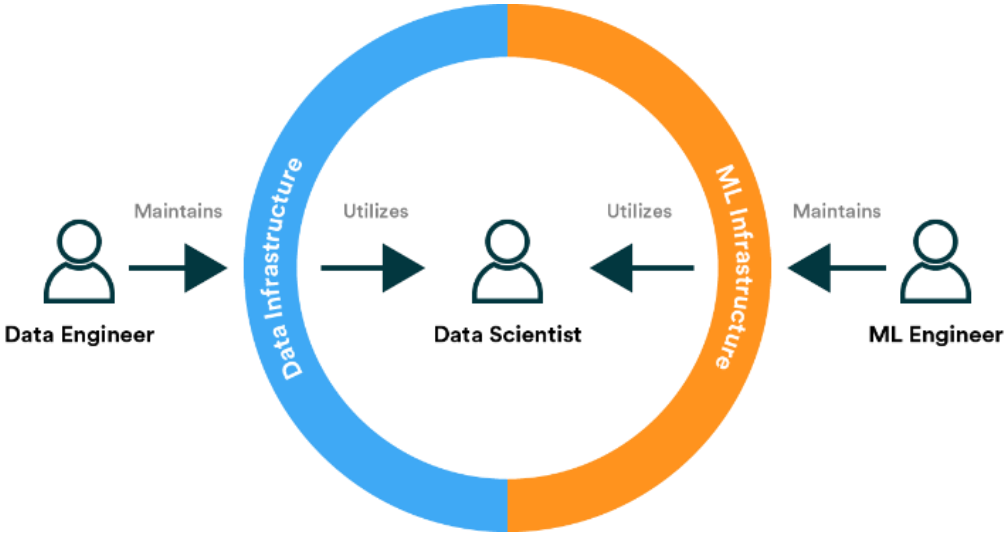
Koneoppimismallin kouluttaminen tapahtuu yleensä suljetussa ympäristössä, kuten esimerkiksi pilvialustan laskentaklusterissa tai datatieteilijän omalla koneella. Koneoppimismallin tuotantoon vieminen on prosessi, jossa luotu koneoppimismalli julkaistaan sellaiseen ympäristöön, jonka kautta sitä voidaan käyttää. [13.]

Tuotantoympäristön valintaan vaikuttaa esimerkiksi, kuinka usein ja millä tavoin ennustuksia koneoppimismallilla on tarkoitus tehdä. Reaaliaikainen ennustaminen ja eräennustaminen (engl. batch inference) ovat kaksi yleisintä koneoppimismallien ennustuksien tekemiseen käytettyä tapaa. Reaaliaikaista ennustamista käytetään silloin, kun ennustuksia tarvitsee tehdä usein ja ne pitää saada käytettäväksi mahdollisimman nopeasti. Eräennustamista käytetään puolestaan silloin, kun ennustaminen tehdään kerralla suurelle määrälle dataa ja tulokset tallennetaan myöhempää käyttöä varten esimerkiksi tietokantaan. [14.]

Koneoppimismallit on myös mahdollista sulauttaa esimerkiksi osaksi mobiili-, web- tai työpöytäsovellusta. Yleensä tuotantoympäristöltä kuitenkin vaaditaan, että koneoppimismallin toimintaa tulee pystyä monitoroimaan tarkasti, jotta saadaan tieto, milloin koneoppimismalli on tarpeen kouluttaa uudelleen. Lisäksi koneoppimismallien käyttämiseen saatetaan tarvita paljon laskentatehoa, jota ei esimerkiksi älypuhelimesta saata löytyä. Näistä syistä koneoppimismallista luodaan usein Docker-kontissa (kts. 3.2.1 Konttiteknologia) toimiva REST API ja tuotantoympäristöksi valitaan Kubernetes (kts. 3.2.3 Kubernetes). [15.]

2.2 Työntekijät osana koneoppimisprojektia

Koneoppimisprojektin elinkaareen kuuluu useita eri työvaiheita, joiden toteuttamiseksi tarvitaan niihin erikoistuneita työntekijöitä. Työntekijöiden vastuualueet voivat vaihdella paljon yrityksen koon ja projektin laajuuden mukaan. Esimerkiksi pienissä startup-yrityksissä yksi työntekijä voi tehdä töitä usealla eri osa-alueella, kun taas suuremmissa yrityksissä työntekijät saattavat erikoistua tarkoin rajattuihin rooleihin. Nykyinen trendi on, että koneoppimisprojektien tekninen projektitiimi koostuu datainsinööreistä, datatieteilijöistä ja koneoppimisinsinööreistä (Kuva 4), jotka tekevät tiivistä yhteistyötä keskenään, mutta ovat kuitenkin erikoistuneita omiin työtehtäviinsä [16].



Kuva 4. Havainnekuva tekniseen projektitiimiin kuuluvien työntekijöiden vastuualueista [16].

Datainsinöörit vastaavat datan keräämisestä, muokkaamisesta ja tallentamisesta. He tekevät yhteistyötä datatieteilijöiden kanssa ja varmistavat, että data on datatieteilijän tarvitsemassa muodossa ja helposti saatavilla.

Datatieteilijät ovat keskeisiä koneoppimisprojektin työntekijöitä. Heidän tehtävänään on kehittää ratkaisuja liiketoiminnan ongelmiin datan avulla. Datatieteilijät tutkivat, millaista dataa vaaditaan ja millaisessa muodossa sen tulee olla, jotta ongelma on mahdollista ratkaista. He myös vastaavat koneoppimismallien kouluttamisesta.

Koneoppimisinsinööri on verrattain uusi rooli. Heidän vastuullaan on viedä koneoppimismalli tuotantoympäristöön ja varmistaa, että se toimii siellä oletetulla tavalla. He myös pystyttävät ja ylläpitävät MLOps-alustoja, jotka edesauttavat koneoppimisprojektissa työskentelevien henkilöiden yhteistyötä ja tarjoavat työkaluja koneoppimismallien kouluttamiseen, versioimiseen, tuotantoon viemiseen sekä prosessien monitoroimiseen.

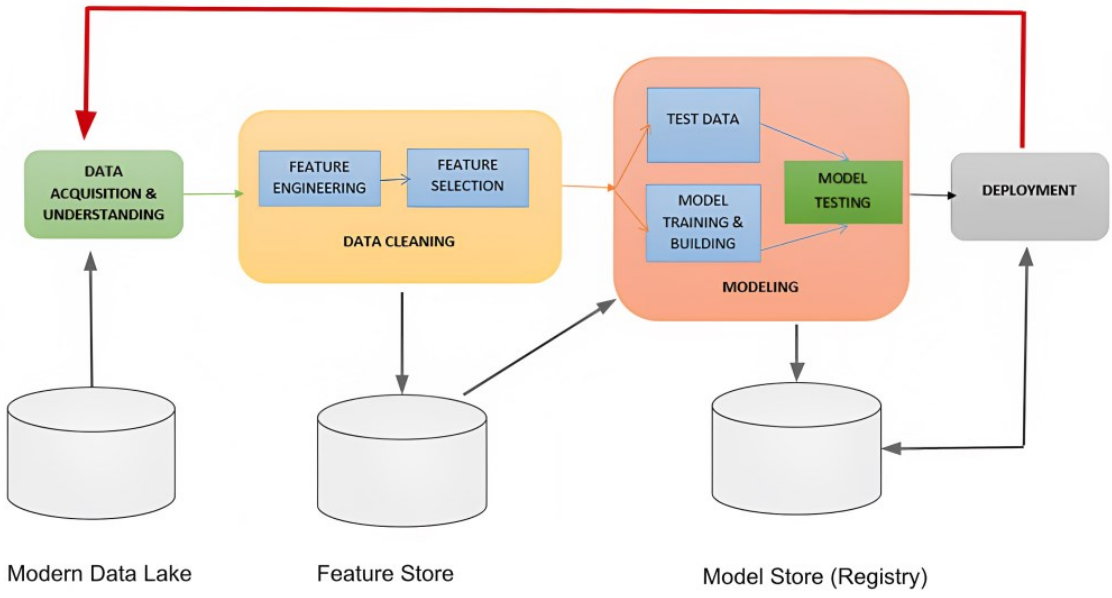
Teknisten osaajien lisäksi projektiin tyypillisesti osallistuu myös asiantuntija. Asiantuntijoilla on merkittävä rooli koneoppimisprojekteissa, joissa tarvitaan syvää ymmärrystä ratkaistavasta ongelmasta. He ovat oman alansa osaajia, jotka auttavat datatieteilijöitä ymmärtämään dataa sekä ratkaistavaa ongelmaa. [17.]

3 MLOps

MLOps toimii sateenvarjoterminä hyviksi todetuille käytänteille ja toimintatavoille, joita seuraamalla voidaan edesauttaa koneoppimisprojektin onnistumisessa [18]. Sen päätarkoituksena on nopeuttaa koneoppimismallien tuotantoon viemistä sekä varmistaa kaikkien siihen vaadittavien välivaiheiden toimivuus hyödyntämällä automatisointia ja monitorointia [19].

3.1 Keskeiset käytänteet

Kun koneoppimisprojekti toteutetaan MLOps-käytänteiden mukaisesti, luodaan jokaisesta työvaiheesta koostuva ”putki” (engl. pipeline), jossa ennalta määritetyt työvaiheet suoritetaan peräkkäin automatisoidusti. Datan käsittelystä, koneoppimismallin kouluttamisesta ja koneoppimismallin tuotantoympäristöön viemisestä koostuvaa kokonaisuutta kutsutaan MLOps-putkeksi (Kuva 5). [12.]

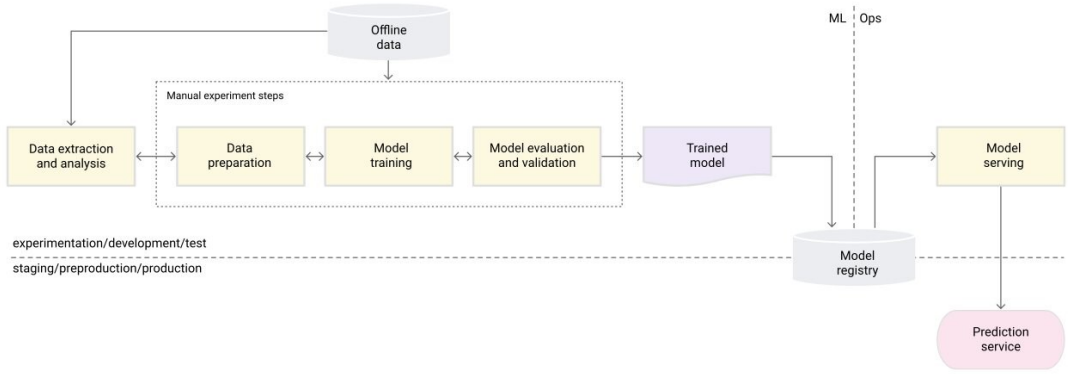


Kuva 5. Havainnekuva MLOps-putken rakenteesta [11].

3.1.1 Automatisointi

Automatisoinnin avulla koneoppimismallin kouluttaminen ja tuotantoon vieminen saadaan toteutettua tehokkaasti ja luotettavasti. Tavoitteena on pyrkiä automatisoimaan koko MLOps-putki siten, ettei ihmisen tarvitse olla osallisena, kun koneoppimismalli pitää kouluttaa uudelleen. Googlen luoman automatisoinnin kypsyysmallin [20] mukaan, automatisointi voidaan jaotella kolmelle eri tasolle:

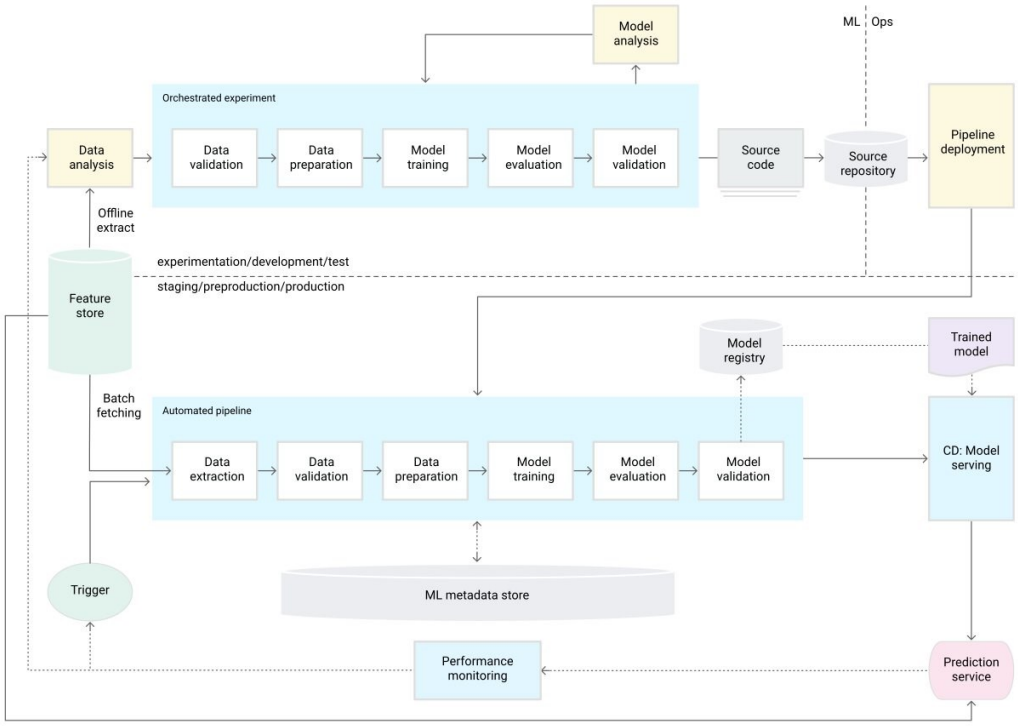
Taso 0: Manuaalinen prosessi (Kuva 6). Tällä tasolla ei ole automatisointia ollenkaan, eli kaikki työvaiheet datan käsittelystä, koneoppimismallin kouluttamisesta sekä validoinnista aina tuotantoympäristöön viemiseen saakka toteutetaan manuaalisesti. Tälle tasolle on tyypillistä, että datatieteilijä luo, kouluttaa ja testaa koneoppimismallin, minkä jälkeen toinen tiimi vie tallennetun mallin tuotantoympäristöön. Tämä on helposti toteutettava toimintamalli ja toimii hyvin, jos datassa ei juuri tapahdu muutoksia ja koneoppimismallia tarvitsee päivittää todella harvoin. Ongelmana kuitenkin on, että ilman automatisointia ja koneoppimismallin monitorointia ei voida olla varmoja, alkaako koneoppimismallin tarkkuus jossakin välissä laskemaan.



Kuva 6. Havainnekuva tasosta 0 [20].

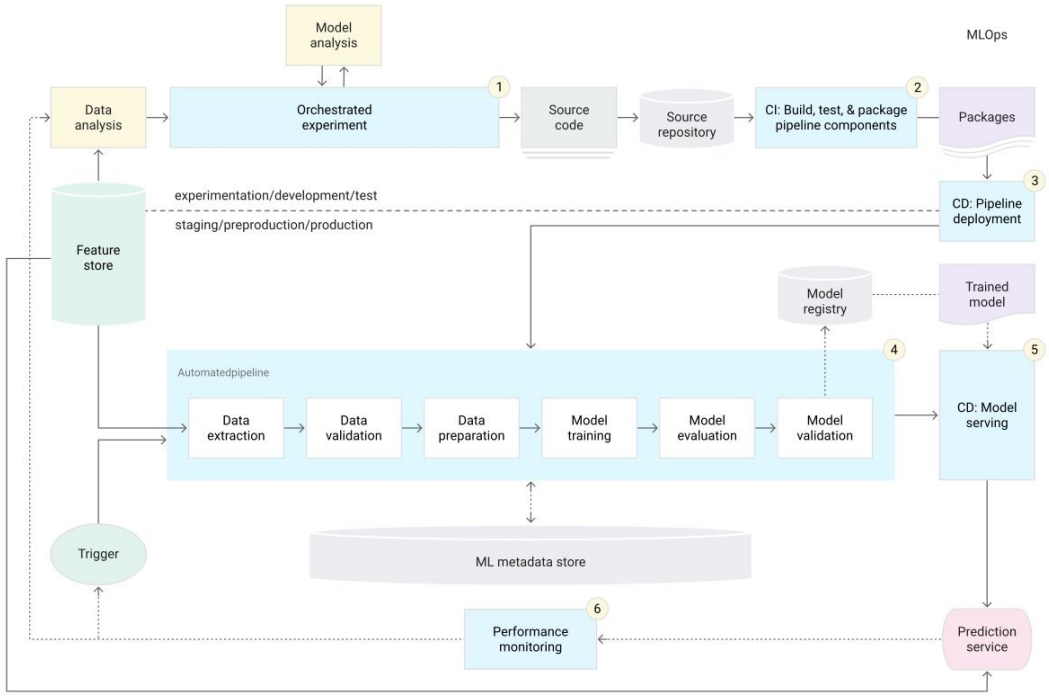
Taso 1: Automatisoitu MLOps-putki (Kuva 7). Tällä tasolla aikaisemmin manuaalisesti suoritettavista työvaiheista luodaan uudelleenkäytettäviä komponentteja, joista koostetaan automatisoidusti suoritettava MLOps-putki. Datan laatua varmennetaan testien avulla ja uuden koneoppimismallin toimivuutta verrataan tuotantoympäristössä olevan koneoppimismallin toimivuuteen, jotta voidaan varmistaa, että tuotantoympäristössä on käytössä aina parhaita tuloksia antava koneoppimismalli. Tuotantoympäristössä toimivan koneoppimismallin suoritusta monitoroidaan ja järjestelmään luodaan hälytyksiä, joiden avulla koneoppimismallin uudelleen kouluttaminen

käynnistetään automaattisesti tarpeen mukaan. Tällä tasolla otetaan myös käyttöön koodien versionhallinta sekä datan, koneoppimismallien ja metadatan versiontiin käytettävät tallennuspaikat (kts. 3.1.2 Toistettavuus ja versiointi). Lisäksi kokeilut paremman koneoppimismallin luomiseksi siirretään sellaiseen ympäristöön, jossa on reilusti laskentatehoa ja jossa eri koneoppimisalgoritmien ja datasettien vaikutusta lopullisen mallin luomiseksi voidaan suorittaa rinnakkain. Tämän tason toimintamalli on hyvä ratkaisu, kun MLOps-putki on melko yksinkertainen eikä siihen ei tarvitse tehdä muutoksia usein.



Kuva 7. Havainnekuva tasosta 1 [20].

Taso 2: Automatisoidut CI/CD-putket (Kuva 8). CI/CD-putkia käyttämällä MLOps-putkea voidaan muokata ja kehittää nopeasti sekä luotettavasti automatisoidun testaamisen ansioista. Viimeisen tason toimintamalli on tarkoitettu suurille ja monimutkaisille MLOps-putkille, joita kehitetään jatkuvasti.



Kuva 8. Havainnekuva tasosta 2 [20].

Tason 2 työnkulku koostuu kuudesta eri päätyövaiheesta (Kuva 8).

1. Kehitystyö: Eri koneoppimisalgoritmeja, hyperparametrejä ja datasettejä kokeillaan, jotta saadaan luotua entistä paremmin toimiva koneoppimismalli. Myös MLOps-putken toimintaan liittyvää kehitystyötä tehdään tässä vaiheessa. Tehdyt muutokset viedään versionhallintaan.

2. Jatkuva integraatio: Versionhallintaan tuodut muutokset testataan kattavasti. Muutokset voivat koskea esimerkiksi koneoppimismallin arkkitehtuuria, datan sisältöä tai MLOps-putken toiminnallisuuksia.

3. Jatkuva toimittaminen: Testatut muutokset viedään osaksi MLOps-putkea.

4. Automatisoitu käynnistäminen: MLOps-putki voidaan määrittää käynnistymään automaattisesti eri tilanteissa. Esimerkiksi muutosten tekeminen putkeen, koneoppimismallin tarkkuuden laskeminen tai putken ajastettu käynnistyminen voivat käynnistää MLOps-putken. MLOps-putken tuotoksena on uusi koneoppimismalli, joka tallennetaan ja versioidaan.

5. Koneoppimismallin jatkuva toimittaminen: Uusi koneoppimismalli viedään tuotantoympäristöön.

6. Monitorointi: Koneoppimismallin suoritusta ja sille syötettyä dataa monitoroidaan. Jos data alkaa poikkeamaan koulutuksessa käytetystä datasta tai koneoppimismalli alkaa antamaan aikaisemmasta poikkeavia arvoja, käynnistetään mallin uudelleenkouluttaminen.

3.1.2 Toistettavuus ja versiointi

MLOps-putkissa toistettavuudella tarkoitetaan, että jokaisen eri työvaiheen on tuotettava sama lopputulos, kun sille annetaan sama syöte [21]. Käytännössä tämä tarkoittaa sitä, että MLOps-putkessa koulutetun mallin tulee olla aina (lähes) samanlainen, kun käytetään samaa dataa, koneoppimisalgoritmia, hyperparametreja sekä koulutusympäristöä. Automatisoiduissa MLOps-putkessa toistettavuus on tärkeä piirre, sillä se tekee sen toiminnasta luotettavaa ja ennakoitavaa, jolloin mahdollisia virhetilanteita ei pääse helposti syntymään [22].

Versioinnin tarkoituksena on pitää kirjaa kaikista muutoksista, joita koodeihin, dataan ja koneoppimismalleihin on tehty. Versiointi mahdollistaa palaamisen viimeiseen toimivaan versioon, jos MLOps-putkeen tai tuotantoympäristöön tehdään muutoksia, jotka vahingossa rikkovat niiden toiminnan. [23.]

Perinteisissä ohjelmistokehitysprojekteissa ohjelmiston lähdekoodien versiointiin käytetään hajautettua versionhallintajärjestelmää, kuten esimerkiksi Git. Koneoppimisprojekteissa koodien lisäksi tulee versioida myös dataa ja koneoppimismalleja. Datan versionhallintaan käytetään feature storea ja koneoppimismallin versionhallintaan käytetään model reqistryä (Kuva 5). Myös koulutuksesta kerätylle metadatalle voi olla oma tallennuspaikka (Kuva 8). Sinne tallennetaan koulutuksen aikana kerättyjä metriikoita (esimerkiksi tarkkuus tai virheen määrä), kuvaajia sekä tietoa koneoppimismallin arkkitehtuurista, käytetystä datasta ja hyperparametreistä. Usein koulutuksesta kerätty metadata tallennetaan kuitenkin suoraan model reqistryyn koneoppimismallin kanssa. [24.]

Koneoppimisprojekteissa versiointi mahdollistaa koneoppimismallien toistettavuuden. Jos esimerkiksi yksi projektitiimin jäsen onnistuu luomaan hyvin toimivan koneoppimismallin, voidaan sen tekemiin päätöksiin luottaa varmemmin, kun joku toinen projektitiimin jäsen pystyy toistami-

seen luomaan saman mallin. Koneoppimisalgoritmeihin ja koulutusprosessiin liittyy paljon satunnaisuutta, mikä tekee täysin vastaavan koneoppimismallin luomisesta melkein mahdotonta. Jos kuitenkin lähes vastaava koneoppimismalli onnistutaan luomaan, voidaan olla aika varmoja, että se on oppinut datasta oikeita asioita. [25.] Koneoppimismallin toistettavuus on hyvin hankalaa, jos ei ole tiedossa, mitä koneoppimismallin arkkitehtuuria, hyperparametreja ja dataa koulutuksessa on käytetty. Versioinnin käytetään, jotta kaikki edellä mainitut tiedot voidaan tallentaa jokaisen koulutetun koneoppimismallin kohdalla. Pitkällä aikavälillä versiointi auttaa myös näkemään, miten esimerkiksi eri hyperparametrit tai piirteet datassa vaikuttavat koneoppimismallin oppimiseen. [26.]

3.1.3 Testaaminen

Testien avulla pyritään varmistamaan, että koodeihin tehdyt muutokset toimivat oletetulla tavalla, ennen kuin ne liitetään osaksi MLOps-putkea. [27.] Testien avulla voidaan myös varmistaa, että kaikki MLOps-putken työvaiheet toimivat säännönmukaisesti, mikä edesauttaa toistettavuuden saavuttamista. Yleisiä koneoppimisprojekteissa käytettyjä testityyppejä ovat savu-, yksikkö-, integraatio- ja regressiotestit.

Savutestit ovat yksinkertaisia testejä, joilla testataan, että kaikki MLOps-putken ja tuotannossa olevan koneoppimismallin kriittiset toiminnot toimivat oikein. Savutesteillä voidaan esimerkiksi testata, saadaanko koneoppimismallilta vastaus onnistuneesti. [28.]

Yksikkötestejä kirjoittaessa pyritään ottamaan huomioon kaikki mahdolliset tilanteet, jotka voivat aiheuttaa virhetilanteen ohjelmistossa. Niiden avulla varmistetaan, että funktiot sekä metodit toimivat suunnitellulla tavalla, kun niille annetaan erilaisia syötteitä. Esimerkiksi dataa käsittelevää funktiota voidaan testata datasetillä, joka sisältää NaN-, None- ja Null-arvoja, jolloin varmistetaan, että se pystyy käsittelemään nämä onnistuneesti. [28.]

Kun yksikkötesteillä on varmistettu, että yksittäiset komponentit toimivat oikein, testataan integraatiotesteillä, toimivatko useasta komponentista koostuvat laajemmat kokonaisuudet yhdessä oletetulla tavalla. MLOps-putken toistettavuutta voidaan myös testata integraatiotesteillä. [28.]

On hankala luoda niin kattavia testejä, että virhetilanteita ei ikinä pääsisi tapahtumaan. Regressiotestien tarkoituksena on varmistaa, että vastaan tulleet virhetilanteet eivät pääse uusiutumaan tulevaisuudessa. Käytännössä tämä tarkoittaa sitä, että joko olemassa olevista testeistä

tehdään entistä kattavampia tai sitten kirjoitetaan uusia testejä, joilla aikaisemmin tapahtunut virhetilanne voidaan estää. Kun virhetilanne on otettu testeillä huomioon, ei koodiin voida tulevaisuudessa tehdä sellaisia muutoksia, että sama virhetilanne pääsisi uusiutumaan. [28.]

Automatisoidun MLOps-putken kompleksisuus kasvaa nopeasti projektin edetessä. Sen lisäksi, että testit estävät virheellisen koodin viemisen osaksi MLOps-putkea, auttavat ne myös projektitiimiä ymmärtämään, miten eri asioiden on tarkoitus toimia ja miten ei. Testien kirjoittaminen kannattaa aloittaa mahdollisimman aikaisessa vaiheessa, kun koko projekti on kokonaisuudessaan vielä hahmotettavissa ja kaikki toimii oikein. Testien kirjoittaminen jälkikäteen on paljon hitaampaa ja kalliimpaa verrattuna siihen, että ne olisi kirjoitettu kehitystyön ohessa.

3.1.4 Monitorointi

Monitoroinnin avulla pidetään huolta, että tuotantoympäristössä oleva koneoppimismalli toimii suunnitellulla tavalla [29]. Myös MLOps-putken työvaiheita ja dataa monitoroimalla voidaan varmistaa, että koneoppimismallin uudelleen kouluttaminen ja tuotantoympäristöön vieminen tapahtuu varmatoimisesti ja tehokkaasti.

Breck E, Cai S, Nielsin E, Salib M ja Scully D luoman listauksen [30 s.5–6] mukaan, seuraavia asioita monitoroimalla ja seuraamalla voidaan varmistaa, ettei automatisoidun MLOps-putken toiminnallisuudessa pääse tapahtumaan yllättäviä muutoksia:

1. Muutokset datalähteissä: Yleensä dataa haetaan monista eri lähteistä. Muutokset alkuperäisissä datalähteissä voivat aiheuttaa ongelmia koneoppimismallin kouluttamisessa ja päätösten oikeellisuudessa. Jotta tätä ongelmaa ei pääse tapahtumaan, tulee datan rakennetta ja sen sisältämiä tietotyyppejä sekä tilastollisia tunnuslukuja monitoroida. Lisäksi, jos mahdollista, on alkuperäisen datalähteen ylläpitäjien oltava tietoisia, mitä osaa datasta käytetään koneoppimisprojektissa. Jos muutoksia alkuperäiseen datalähteeseen on tehtävä, voidaan siitä ilmoittaa hyvissä ajoin koneoppimisprojektin tiimille, jolloin tarvittavat muutokset MLOps-putkeen voidaan tehdä hallitusti.

2. Datan muuttuminen: Koneoppimismallin tekemien ennustusten tai päätösten oikeellisuuden arvioiminen on hankalaa. Yleensä tuotantoympäristössä olevalle koneoppimismallille syötetty data kuitenkin kerätään talteen, jolloin tätä dataa on mahdollista verrata

koulutuksessa käytettyyn dataan. Jos tuotantoympäristöstä kerätty data alkaa merkittävästi poikkeamaan koulutuksessa käytetystä datasta, voidaan olla aika varmoja, ettei koneoppimismalli pysty enää tekemään tarpeeksi tarkkoja päätelmiä.

3. Koneoppimismallin vanheneminen: Mitä kauemmin aikaa sitten koneoppimismalli on koulutettu, sitä todennäköisemmin sen suorituskyky on heikentynyt. Uudelleen kouluttaminen voidaan ajastaa käynnistymään automaattisesti, jos tiedetään, miten usein koneoppimismalli on keskimäärin pitänyt aikaisemmin kouluttaa uudelleen. Koneoppimismalli voi myös vanheta MLOps-putken näkökulmasta. Vastaan voi esimerkiksi tulla tilanne, jossa tuotantoympäristössä oleva koneoppimismalli jatkokoulutetaan uudella datalla. Jos dataan on tuotu uusia piirteitä tai siihen on tehty muita muutoksia, jotka muuttavat sen muotoa siten, ettei se ei ole enää samassa muodossa aikaisemmin käytetyn datan kanssa, ei koneoppimismallia voida jatkokouluttaa uudella datalla. Nämä tilanteet voidaan estää vertaamalla koneoppimismallin ikää siihen, milloin MLOps-putkeen on tehty jatkokoulutuksen estäviä muutoksia.

4. Koneoppimismallilta on saatava stabiileja arvoja: Varsinkin neuroverkkoihin pohjautuvilla koneoppimisalgoritmeilla on riskinä, että ne menevät koulutuksen aikana ”sekaisin” ja alkavat antamaan NaN- tai infinite-arvoja. Koulutusta on syytä monitoroida, jotta näissä tilanteissa ei turhaan jatketa koneoppimismallin kouluttamista. Näihin tilanteisiin voidaan myös luoda sellaisia varmistuksia, että koneoppimisalgoritmin painoarvot alustetaan uudelleen, jolloin koulutus voidaan käynnistää alusta. Tuotantoympäristössä olevan koneoppimismallin antamia vastauksia tulee myös seurata, jotta koneoppimismallin kouluttaminen voidaan käynnistää uudestaan, jos sen antamat vastaukset alkavat poikkeamaan merkittävästi aikaisempaan verrattuna.

5. Koneoppimismallin suorituskyvyn sekä resurssienkäytön seuranta: Useat eri muuttujat vaikuttavat siihen, kuinka nopeasti koneoppimismallin koulutus tapahtuu, miten paljon ne käyttävät laskentaresursseja ja kuinka nopeasti koneoppimismallilta saadaan vastaus tuotantoympäristössä. Jotta MLOps-putki saadaan pidettyä tehokkaana ja tuotannossa oleva koneoppimismalli toiminta nopeana, tulee eri resurssien käyttöä seurata. Jos resurssien käytössä alkaa tapahtumaan merkittävää muutosta, voidaan nopeasti tarkistaa, mikä viimeisimmässä versionhallintaan tehdystä muutoksesta sen aiheuttaa ja onko muutos tarkoituksenmukainen.

6. Koneoppimismallin tekemien päätelmien monitorointi: Jos koneoppimismallia käytetään sellaisessa ympäristössä, jossa on mahdollista nopeasti tarkistaa, tekikö koneoppimismalli oikean päätelmän, tulee koneoppimismallin tarkkuutta tai virheen määrää monitoroida. Koneoppimismallin hyvyydelle voidaan asettaa alaraja, ja jos mallin tarkkuus laskee sen alapuolelle, käynnistetään koneoppimismallin kouluttaminen uudelleen. Jos koneoppimismallin tarkkuutta ei ole mahdollista seurata, tulee pitää huolta, että koneoppimismallin hyvyyden arvioimiseen käytetyn testidatan tulee olla päivitettynä aina, kun huomataan, että tuotantoympäristössä oleva data ei vastaa enää aikaisemmin koulutuksessa käytettyä dataa.

3.2 Yleisesti käytettyjä teknologioita

Automatisoidun MLOps-putken luomiseen tarvitaan useita eri teknologioita. Konttiteknologiat, mikropalveluarkkitehtuuri, Kubernetes ja CI/CD-putket ovat yleisesti käytettyjä teknologioita koneoppimisprojekteissa. Ne auttavat luomaan helposti hallittavan, skaalattavan sekä luotettavan MLOps-putken.

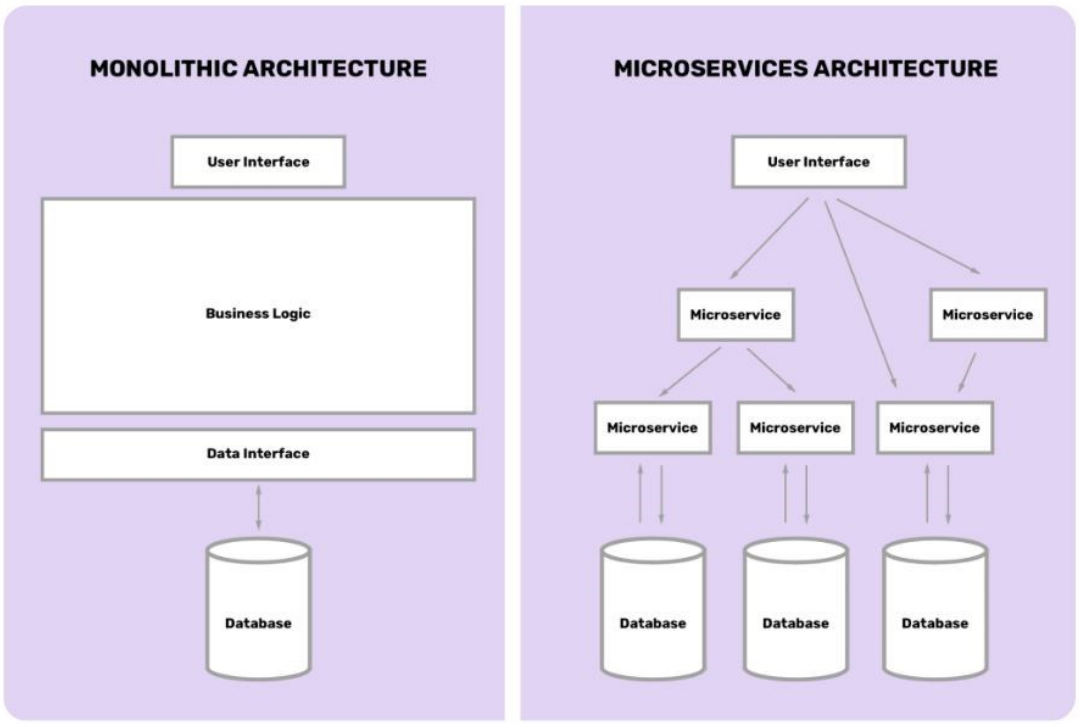
3.2.1 Konttiteknologiat

Konttiteknologioilla tarkoitetaan tapaa suorittaa sovellusta omassa eristetyssä suoritusympäristössä (engl. container). Kontissa suoritettavan sovelluksen etuna on se, että se toimii kaikissa ympäristöissä samalla tavalla, sillä se pitää sisällään kaikki riippuvuudet, asetukset ja työkalut, joita sovelluksen suorittamiseen tarvitaan [31]. Konttiteknologiasta on olemassa useita eri versioita, mutta niistä tunnetuin ja yleisimmin käytetty on Docker [32].

Jokainen kontti muodostetaan omasta kuvasta (engl. image). Kuva toimii sapluunana, joka pitää sisällään tiedon, mitä kaikkea kontin sisälle tulee luoda ja kuinka se tulee rakentaa. [33.] Kuva on pakattu tiedosto, joten se on kooltaan verrattain pieni. Kuvan pieni koko tekee sovelluksen siirtämisen ympäristöstä toiseen nopeaa ja helppoa.

3.2.2 Mikropalveluarkkitehtuuri

Mikropalveluarkkitehtuurilla tarkoitetaan nykyaikaista tapaa luoda sovelluksia, jossa sovellus koostuu useasta mikropalvelusta. Jokainen mikropalvelu vastaa tietystä sovelluksen osa-alueesta, ja niitä suoritetaan tyypillisesti Docker-konteissa. Monoliittinen arkkitehtuuri on vastakohta mikropalveluarkkitehtuurille (Kuva 9). Monoliittisessa arkkitehtuurissa kaikki sovelluksen toiminnallisuudet on kehitetty yhteen isoon sovellukseen. [34.]



Kuva 9. Havainnekuva monoliittisesta- ja mikropalveluarkkitehtuurista [34].

Mikropalveluarkkitehtuuri tarjoaa paljon hyötyjä sovelluksen kehittämiseen. Jokaista mikropalvelua voi esimerkiksi kehittää oma tiimi. Tämä nopeuttaa sovelluksen kehittämistä, sillä ohjelmistokehittäjät työskentelevät pienemmän koodimäärän parissa, eikä heidän tarvitse ottaa huomioon kaikkia sovelluksen toiminnallisuuksia. Jokaista mikropalvelua voidaan päivittää, skaalata ja hallinnoida itsenäisesti, mikä on kustannustehokkaampaa verrattuna monoliittiseen sovellukseen. Mikropalveluarkkitehtuurin mukaisesti luotu sovellus ei myöskään ole yhtä virhealtis kuin monoliittinen sovellus. Jos monoliittisessa sovelluksessa tapahtuu odottamaton virhe, saattaa koko sovellus kaatua, kun taas mikropalveluarkkitehtuurisessa sovelluksessa kaatuu ainoastaan yksittäinen mikropalvelu. [34.]

Mikropalveluarkkitehtuuri on todettu sopivan hyvin koneoppimisprojekteihin, kun ne toteutetaan MLOps-periaatteiden mukaisesti. Kaikki MLOps-putkessa toteutettavat työvaiheet voidaan toteuttaa mikropalveluina. Tällä tavoin jokaiselle työvaiheelle voidaan varata tarvittavat määrät laskentaresursseja ja niihin voidaan sisällyttää vain tarvittavat riippuvuudet. Koska jokainen mikropalvelu pitää sisällään vain rajatun määrän toiminnallisuuksia, on mahdollisten virheiden löytäminen myös helpompaa verrattuna siihen, että kaikki työvaiheet suoritettaisiin peräkkäin samassa ympäristössä [35.]

3.2.3 Kubernetes

Kubernetes on konttien orkestrointiin käytettävä avoimen lähdekoodin alusta. Sen avulla voidaan määrittää ja automatisoida konteissa suoritettavien sovellusten ja mikropalveluiden välisiä yhteyksiä, skaalautumista sekä käytettävissä olevia resursseja. [36.]

Kubernetesin suosio on kasvanut nopeasti. Vuonna 2020 jopa 59 % suurista yrityksistä käytti Kubernetesia tuotantoympäristönä, kun vuonna 2018 sitä käytti vain 18 %. [37]. Yhtenä syynä tähän on, että se tarjoaa useita toiminnallisuuksia, joiden avulla voidaan minimoida tilanteita, jotka voivat aiheuttaa palvelussa käyttökatkoja [36]:

1. Kun uusi versio sovelluksesta tuodaan tuotantoon, käynnistää se uuden version käytettäväksi, ennen kuin vanha versio otetaan pois käytöstä. Tällä tavalla käyttökatkoksia ei pääse syntymään.

2. Esimerkiksi ruuhkatilanteissa, se skaalaa konttien määrää automaattisesti käyttötarpeen mukaan ja jakaa käyttökuorman tasaisesti näiden välille, jolloin palvelu ei pääse ylikuormittumaan.

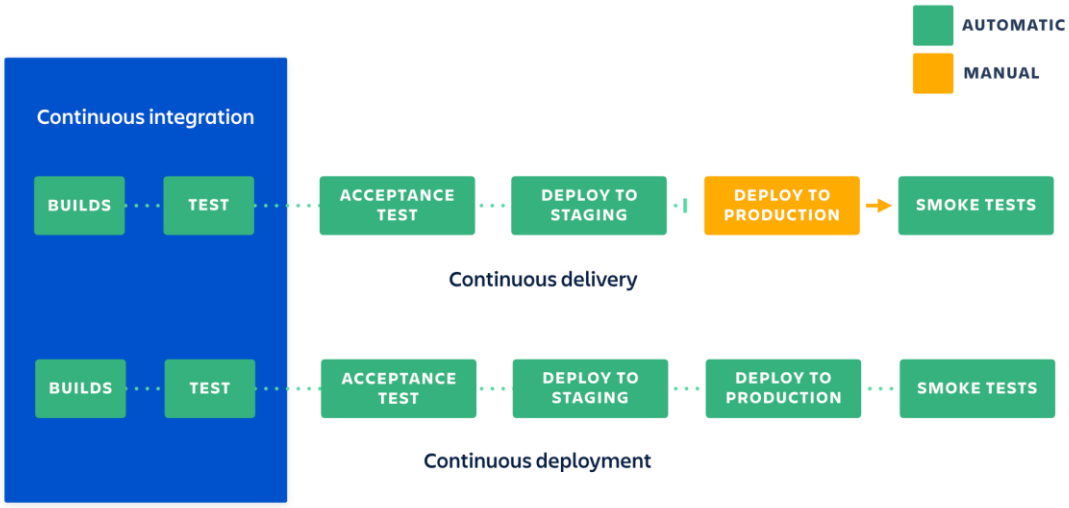
3. Se seuraa jokaisen kontin ”terveydentilaa”. Virhetilanteen mukaan se voi esimerkiksi käynnistää kontin uudestaan tai korvata sen toisella vastaavalla kontilla, jolloin palvelu selviää virhetiloista nopeasti.

Kubernetes tarjoaa myös useita työkaluja ja menetelmiä, joiden avulla voidaan hallinnoida, automatisoida ja monitoroida hyvin kompleksisia kokonaisuuksia, ja tämä tekee siitä hyvän alustan MLOps-putkille. MLOps-putket koostuvat useista eri työvaiheista, joita kehittävät niihin erikois-

tuneet työntekijät. Kukin tiimi voi luoda eri osa-alueisiin tarvittavat työvaiheet käyttäen valitsemiaan ohjelmointikieliä ja työkaluja, minkä jälkeen ne on mahdollista yhdistää toimivaksi kokonaisuudeksi käyttäen Kubernetesiä. [38 s.3.]

3.2.4 CI/CD

CI/CD on ohjelmistokehitysmenetelmä, jossa ohjelmistoon tehtyjen muutosten viemistä tuotantoympäristöön nopeutetaan automatisoinnin avulla. CI, eli jatkuva integraatio (engl. continuous integration), on vaihe, jossa sovellukseen tehdyt muutokset testataan automatisoidusti ohjelmistokehittäjien kirjoittamilla testeillä. CD-vaiheella voidaan viitata joko jatkuvaan toimittamiseen (engl. continuous delivery) tai jatkuvaan julkaisemiseen (engl. continuous deployment). Erona näiden kahden välillä on, että jatkuvassa julkaisemisessa sovellukseen tehdyt muutokset viedään tuotantoon täysin automatisoidusti, kun jatkuvassa toimittamisessa ihmisen pitää käydä manuaalisesti hyväksymässä muutokset, ennen kuin ne julkaistaan tuotantoympäristöön (Kuva 10). [39.]



Kuva 10. Jatkuvassa toimittamisessa ihminen käy manuaalisesti hyväksymässä muutosten viemisen tuotantoon [39].

CI/CD-menetelmässä toteutetaan ennalta määritetyt askeleet, minkä takia niitä yleensä kutsutaan CI/CD-putkiksi. CI/CD-putkia voidaan hyödyntää myös koneoppimisprojekteissa uuden koneoppimismallin tuotantoon viemisen nopeuttamiseksi.

Ohjelmistokehitysprojekteissa ja koneoppimisprojekteissa on muutamia eroavaisuuksia, jotka vaikuttavat siihen, mitä CI/CD-putkilta vaaditaan, kun niitä hyödynnetään osana koneoppimisprojekteja [40]:

- Perinteisessä ohjelmistokehityksessä CI/CD-putkia käytetään ainoastaan koodiin tehtyjen muutosten testaamiseen ja tuotantoon viemiseen. Koneoppimisprojekteissa koodien lisäksi käytetään myös dataa, jolloin näitä kumpaakin pitää pystyä testaamaan ja näistä kumpaan tahansa tehdyt muutokset tulee käynnistää CI/CD-putket.

- Koneoppimismallien kouluttamiseen ja parhaiden hyperparametrien etsimiseen tarvitaan usein paljon laskentatehoa. Tästä syystä CI-putken suoritusympäristössä tulisi olla käytettävissä esimerkiksi tehokkaista grafiikkaprosessoreita, tai CI-putkesta pitäisi päästä käsiksi sellaiseen ympäristöön, jossa koulutus voidaan suorittaa.

- Koneoppimismallien validointi on monimutkaisempaa verrattuna ohjelmistokoodin testaamiseen. Ohjelmisto joko toimii oletetulla tavalla tai ei, mutta koneoppimismallin toimivuuden testaamiseen tarvitaan dataa, ja toimivuutta pitää usein verrata tuotannossa olevan koneoppimismallin toimivuuteen. Staging-ympäristö on perinteisessä ohjelmistokehityksessä käytetty testausympäristö, joka vastaa tuotantoympäristöä. Staging-ympäristössä voidaan esimerkiksi seurata valitun aikajakson ajan, kuinka uusi koneoppimismalli toimii ja verrata sitä tuotantoympäristössä olevaan malliin.

- Perinteisen ohjelmiston suorituskyky ei muutu ajan saatossa, mutta koneoppimismallin suorituskyky saattaa muuttua. Koneoppimismallin kouluttamiseen käytetään menneisyydessä kerättyä dataa, mutta maailma saattaa muuttua äkillisestikin, jolloin koneoppimismalli ei enää toimi oletetulla tavalla. Tämä lisää automatisoidun monitoroinnin tarvetta, jotta CI/CD-putket käynnistyvät automaattisesti, kun koneoppimismallin suorituskyky laskee liikaa.

4 Microsoft Azure

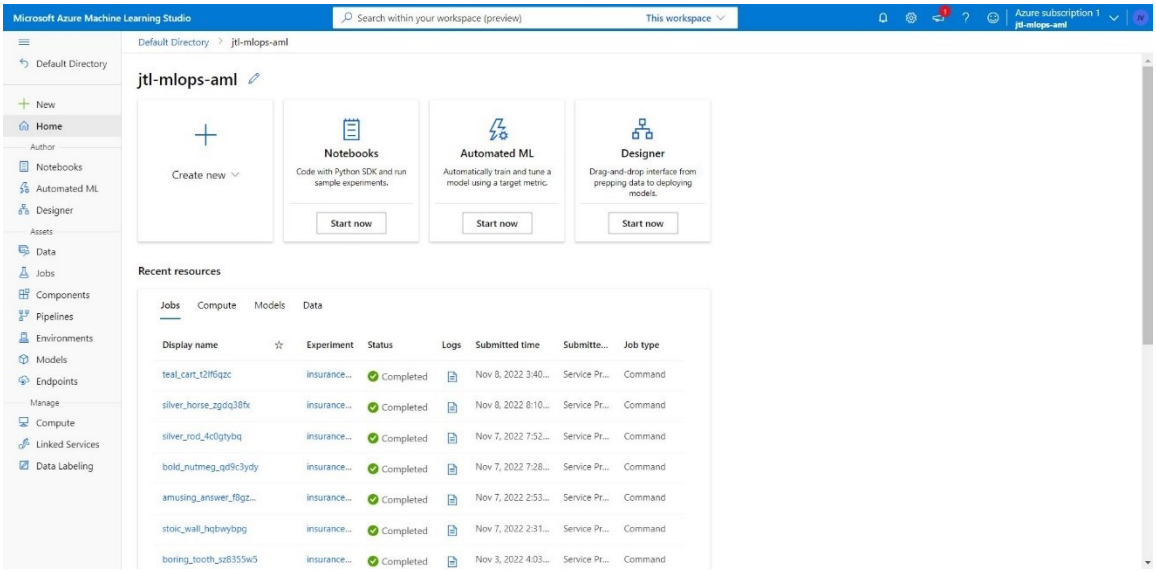
Azure on Microsoftin omistama pilvipalvelualusta. Se tarjontaan kuuluu yli 200 erilaista tuotetta ja palvelua, jotka on suunniteltu tekemään sovellusten kehittämisestä, suorittamisesta ja ylläpidosta mahdollisimman yksinkertaista. Yhdysvaltojen viidestäsadasta suurimmasta yrityksestä jopa 95 % käyttää Azuren palveluita. [41.] Azure on toiseksi suosituin pilvialusta Amazonin omistaman AWS:n jälkeen [42].

Microsoft Azure ja muut pilvipalvelualustat ovat nousseet suureen suosioon viimeisen reilun kymmenen vuoden aikana. Suurimpina syinä tähän on se, ettei yritysten tarvitse enää itse ostaa ja ylläpitää omia palvelimia, vaan he voivat maksaa pilvipalvelutarjoajille sen mukaan, miten paljon ne pilvipalveluiden resursseja ja palveluita hyödyntävät. Tämä tekee pilvipalveluista yrityksille hyvin kustannustehokasta ja vaivatonta. [43].

4.1 Azure Machine Learning

Azure Machine Learning, tai lyhyemmin AML, on Microsoftin luoma palvelu, joka on kehitetty koneoppimisprojekteja varten. Se on suunniteltu nopeuttamaan ja helpottamaan koneoppimisprojektitiimin yhteistyötä ja se tarjoaa työkaluja kaikille koneoppimisprojektin elinkaaren vaiheille. [44.]

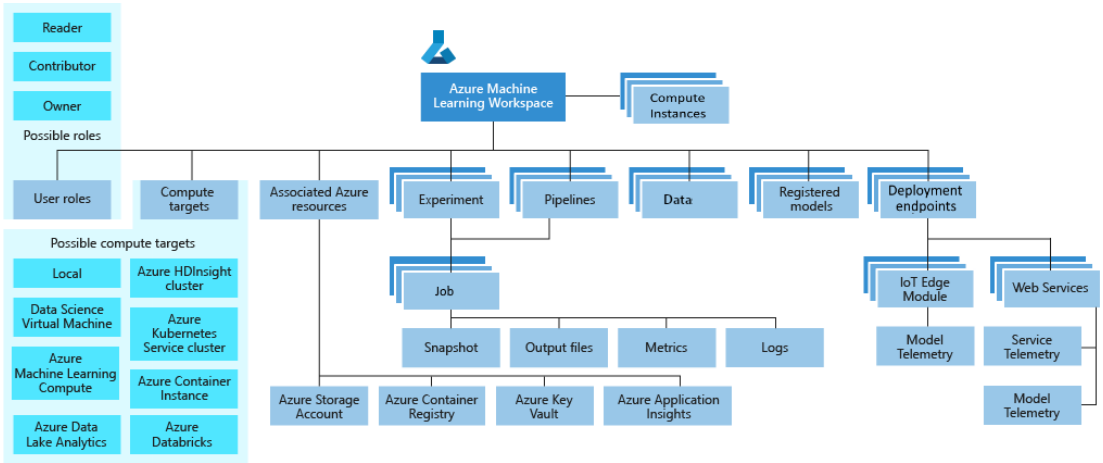
AML-palvelun graafista käyttöliittymää kutsutaan Azure Machine Learning Studioksi (Kuva 11). Käyttäjä pystyy hallitsemaan ja tarkastelemaan työtilan resursseja ja hyödykkeitä (engl. asset) Azure Python SDK -kirjaston, Azure CLI -komentorivityökalun ja graafisen käyttöliittymän kautta.



Kuva 11. Azure Machine Learning Studion etusivunäkymä.

4.1.1 Resurssit

AML-palvelu koostuu resursseista ja hyödykkeistä (Kuva 12). Työtila (engl. workspace) on ylemmän tason resurssi, johon voidaan lisätä laskenta (engl. compute) ja tietovarasto (engl. datastore) -resursseja. [45.]



Kuva 12. Azure Machine Learning -työtilan resurssit [46].

Käyttäjä voi ostaa työtilan käyttöön Azuren tarjoamia CPU- tai GPU-laskentaresursseja datan esikäsittelemistä sekä koneoppimismallien kouluttamista ja tuotantoympäristössä suorittamista

varten. Palveluun on mahdollista myös kytkeä omia laskentaresursseja, jos Azuren resursseja ei halua käyttää. [47.]

Usein käyttäjällä on dataa valmiiksi tallennettuna Azuren tietovarastoihin. Olemassa olevat tietovarastot voidaan yhdistää työtilaan, jolloin niihin pääsee tietoturvallisesti käsiksi esimerkiksi Azure CLI -komentorivityökalun ja Azure Python SDK -kirjaston avulla. Työtilaan on myös mahdollista luoda uusia tietovarastoja.

Työtilassa suoritettavat työtehtävät (engl. jobs) suoritetaan työtilaan rekisteröidyissä laskentaresursseissa. Työtehtävien suoriutumista voi seurata graafisen käyttöliittymän kautta ja ne rekisteröidään automaattisesti myöhempää tarkastelua ja toistettavuutta varten. Työtehtävät voivat olla joko yksittäisiä kokeiluja (engl. experiment) tai useammasta työvaiheesta koostua putkia.

4.1.2 Assetit

Koneoppimismalli (engl. model), ympäristö (engl. environment), data ja komponentti (engl. component) ovat työtilaan rekisteröitäviä hyödykkeitä. Komponentit ovat rekisteröityjä koodeja, joita voidaan käyttää uudelleen, kun samaa komponenttia käytetään osana useampaa putkea [48]. Koneoppimismallien kouluttamiseen käytetty data ja koulutettu koneoppimismalli on mahdollista rekisteröidä ja versioida, jolloin käyttäjä pystyy myöhemmin tarkistamaan, millä datalla mikäkin koneoppimismalli on koulutettu.

Ympäristöt ovat valmiiksi määriteltyjä Docker-kuvia, joita voidaan käyttää koneoppimismallin kouluttamiseen tai julkaistujen koneoppimismallien suoritusympäristöinä. Valmiita ympäristöjä löytyy esimerkiksi TensorFlow-, PyTorch- ja Scikit-learn-kirjastoille. Käyttäjä voi myös luoda ja rekisteröidä palveluun omia Docker-kuvia. [45.]

4.2 Azure DevOps

Azure DevOps on Microsoftin luoma alusta, joka tarjoaa useita eri työkaluja, joiden avulla tiimit voivat kehittää ja suunnitella sovelluksia ketterien ohjelmistokehityksen periaatteiden mukaisesti. DevOps-alustasta on tarjolla kahta eri versiota, jotka ovat Azure DevOps Services ja Azure DevOps Server. Azure DevOps Services -versio sijaitsee Azuren palvelimilla ja sitä suositellaan käytettäväksi niille tiimeille, jotka haluavat päästä nopeasti liikkeelle sekä hyödyntää muita Azuren

tarjoamia palveluita osana projektia. Azure DevOps Server on puolestaan tarkoitettu niille tiimeille, jotka eivät esimerkiksi tietoturvasyistä voi käyttää pilvipalveluita, mutta haluavat käyttää Azure DevOpsia omalla palvelimella. [49.]

Azure DevOps -alustaan kuuluu useita eri työkaluja ohjelmistokehitys- ja koneoppimisprojektien toteuttamiseen. Koneoppimisprojektien näkökulmasta alustan tarjoamista työkaluista keskeisimpiä työkaluja ovat Azure Repos sekä Azure Pipelines.

Azure Repos on koodien versionhallintatyökalu, joka tukee Git- ja TFCV-versionhallintajärjestelmiä. Sen avulla kehitystiimi pystyy tallentamaan ja versioimaan kaikki koodeihin tehdyt muutokset yhteen paikkaan, josta ne ovat koko kehitystiimin saatavilla. [50.]

Azure Pipelines mahdollistaa automatisoitujen CI/CD-putkien luomisen ja suorittamisen. CI/CD-putkien avulla on myös mahdollista kontrolloida muita Azuren tarjoamia palveluita, joka on erinomainen ominaisuus koneoppimisprojekteille. [51.] CI/CD-putkesta käsin voidaan esimerkiksi rekisteröidä ja versioida dataa sekä koneoppimismalleja Azure Machine Learning -palveluun.

5 MLOps-putken luominen Microsoft Azure -pilvipalvelussa

Tässä luvussa demonstroidaan, kuinka Azure DevOps - ja Azure Machine Learning -palveluita käyttämällä luodaan automatisoitu MLOps-putki. Automatisointiin käytetään Azure DevOps -palvelun CI/CD-putkia, joilla kontrolloidaan Azure Machine Learning -palvelun resursseja ja assetteja käyttäen Azure CLI -komentorivityökalua.

Projektissa käytetään Kaggle-sivustolta saatavilla olevaa Porto Segure's Safe Drive Prediction -datasettiä [52], jolla LightGBM-kirjastosta [53] saatavilla oleva päätöspuihin [54] ja Gradient Boosting -menetelmään [55] perustuva koneoppimisalgoritmi opetetaan ennustamaan, kuinka todennäköisesti auton kuljettaja hakee vakuutuskorvausta seuraavana vuonna.

Koneoppimisprojektit ovat keskenään hyvin erilaisia, joten tässä työssä käytetyllä datalla tai koneoppimisalgoritmilla ei ole suurta merkitystä. Työssä pyritään ennemminkin antamaan yksinkertaistettu yleiskuvaus siitä, miten Azuren tarjoamat palvelut saadaan toimimaan yhdessä. Työ pohjautuu Mohammad Ghodratigoharin tekemään videosarjaan [56] sekä hänen GitHub-profiilistaan [57] löytyviin lähdekoodeihin, ja se on jaoteltu neljään eri lukuun. Ensimmäisessä luvussa ohjeistetaan, kuinka Azure DevOps -palvelusta saadaan luotua palveluyhteys Azure Machine Learning -palveluun. Toisessa luvussa kerrotaan yleistietoa Azure DevOps -palvelun CI/CD-putkien suoritusympäristöstä. Kolmannessa ja neljännessä luvussa kuvataan, miten Azure DevOps -palvelun CI/CD-putkia käyttämällä koneoppimismalli voidaan kouluttaa, rekisteröidä ja viedä tuotantoympäristöön MLOps-käytänteiden mukaisesti hyödyntäen Azure Machine Learning -palvelua.

5.1 Palveluyhteydet

Jotta Azure DevOps -palvelusta pystytään kontrolloimaan Azure Machine Learning -työtilan resursseja, tulee käyttäjän luoda palveluyhteydet Azure DevOps -palvelusta Azure Machine Learning -työtilaan sekä käyttäjän Azure-tilaukseen.

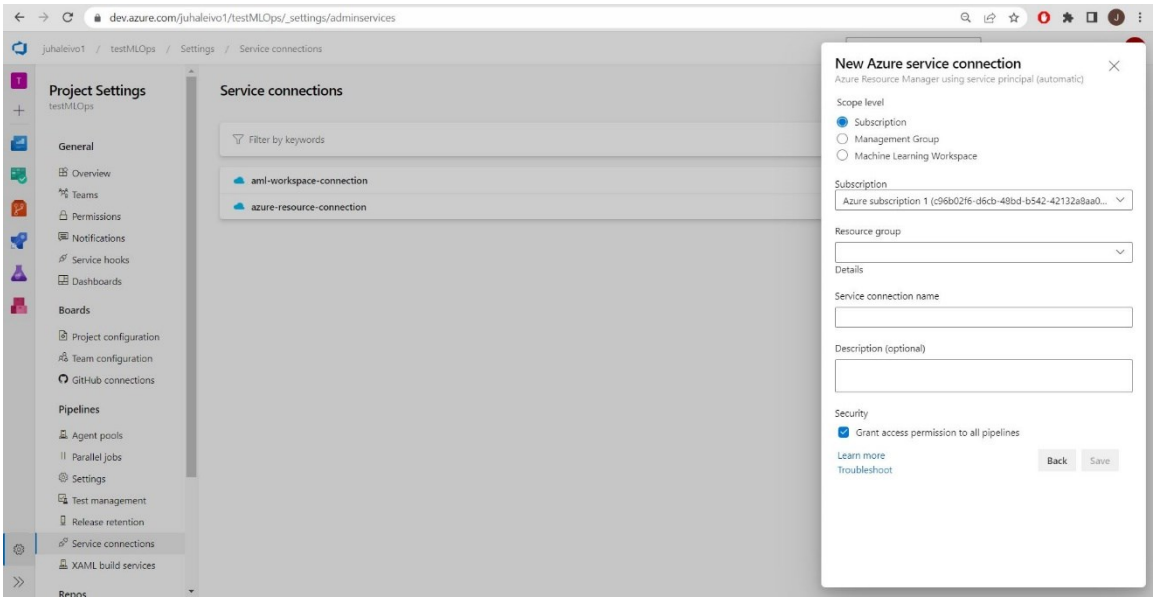
Palveluyhteyksiä varten käyttäjällä tulee olla:

1. Aktiivinen Azure-tilaus
2. Olemassa oleva Resource Group

3. Azure Machine Learning -työtila, joka kuuluu luotuun Resource Groupiin.

Palveluyhteys luodaan Azure DevOps -palveluun luodun projektin asetuksista valitsemalla: Project Settings → Service connections → New Service connection → Azure Resource Manager → Service principal (automatic).

Scope level -kohdasta (Kuva 13) valitaan Subscription, kun palveluyhteys luodaan käyttäjän tilaukseen ja Machine Learning Workspace, kun palveluyhteys luodaan Azure Machine Learning -työtilaan. Palveluyhteyksille kannattaa antaa helposti tunnistettavat nimet ja niille molemmille tulee antaa kaikki oikeudet käyttää CI/CD-putkia rastittamalla Security-kohdasta ”Grant access permission to all pipelines”.

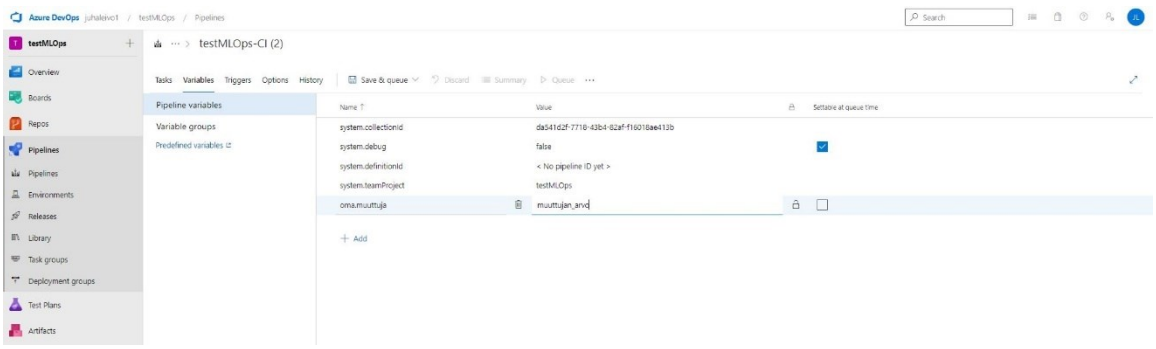


Kuva 13. Uuden palveluyhteyden luominen.

5.2 Putkien suoritusympäristö

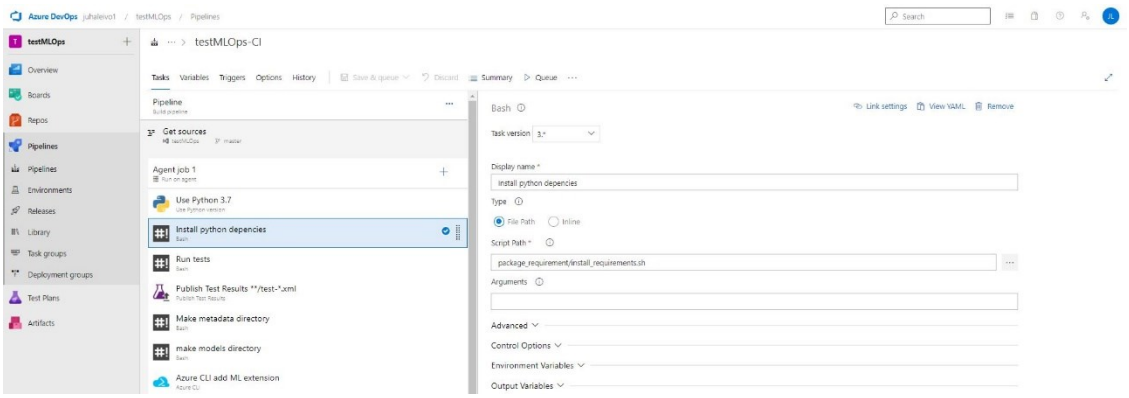
Suoritusympäristönä toimii virtuaalikone, joka luodaan automaattisesti aina uudestaan, kun putki käynnistetään. Suoritusympäristössä ei siis ole tallessa mitään aikaisemmista putkien suoituskerroista. Suoritusympäristöksi voidaan valita jokin Ubuntu-, MacOS- ja Windows-käyttöjärjestelmien tuetuista versioita. Suoritusympäristöstä huolehtii niin kutsuttu agentti, jota voidaan anoa Microsoftilta täyttämällä kaavake [58]. Microsoft voi myöntää yhden agentin (myönnetään vain privaateille projekteilla), jolla CI/CD-putkia voidaan käyttää 30 tunnin ajan kuukaudessa. Maksullisia agentteja [59] on myös saatavilla.

CI/CD-putkille määritetään, mistä versionhallinnan haarasta kaikki tiedostot viedään suoritusympäristön kansiorakenteeseen, josta niitä voidaan suorittaa valituilla työtehtävillä (engl. task) [60]. Putkille voidaan määritellä muuttujia, joita on mahdollista käyttää suoritusympäristössä. Muuttujat määritetään putken Variables-välilehdellä (Kuva 14), tai putkeen voidaan liittää Azure Key Vault [61] -palvelu, josta muuttujat haetaan tietoturvallisesti käyttöön salatusti.



Kuva 14. Muuttujan lisääminen.

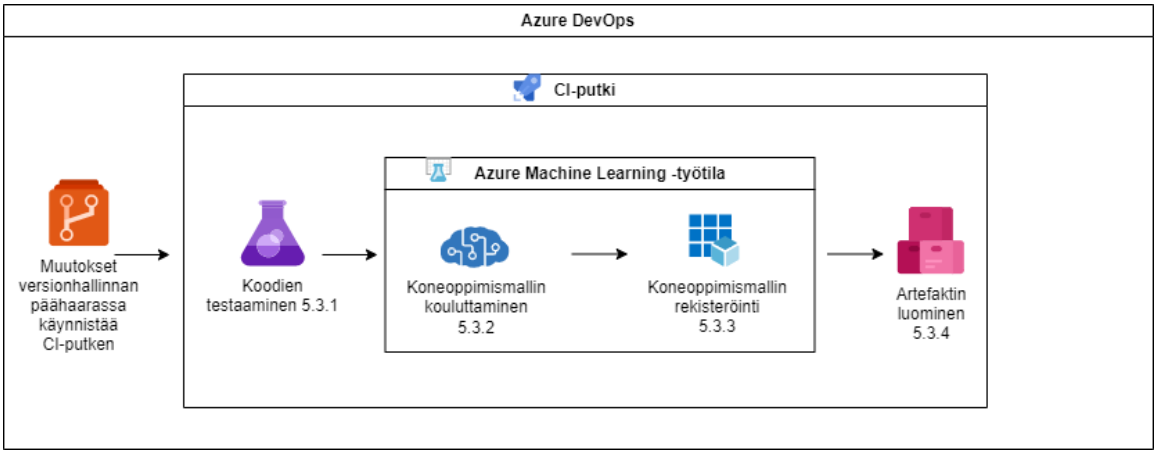
Jotta Python-ohjelmointikielellä kirjoitettua koodia on mahdollista suorittaa, tarvitsee CI/CD-putkien suoritusympäristöihin asentaa Python-tulkki sekä koodin suorittamiseen tarvittavat kirjastot. Azure DevOps -palvelun putkien suunnittelutyökalu tarjoaa tähän ”Use Python version” -työtehtävän, joka automaattisesti asentaa valitun Python-version suoritusympäristöön. Tarvittavat kirjastot saadaan asennettua esimerkiksi käyttämällä Shell-skriptiä (Kuva 15), joka asentaa kaikki tarvittavat kirjastot versionhallinnasta löytyvästä tekstitiedostosta. Tämä on yleinen tapa toimia, sillä tällä tavoin projektin työntekijät voivat helposti pitää sekä toisensa että putken suoritusympäristön ajan tasalla koodien suorittamiseen tarvittavista kirjastoista.



Kuva 15. Riippuvuuksien asentamisen määrittäminen putkien suunnittelutyökalulla.

5.3 CI-putki

CI-putkea käytetään koodin testaamiseen sekä koneoppimismallin kouluttamiseen ja rekisteröintiin. Putken päätteeksi luodusta koneoppimismallista ja sen testaamiseen sekä suorittamiseen tarvittavista koodeista luodaan artefakti, joka otetaan käyttöön CD-putkessa (Kuva 16). Tässä toteutuksessa CI-putki käynnistyy automaattisesti, kun versionhallinnan päähaaraan tehdään muutoksia. CI-putki on myös mahdollista ajastaa käynnistymään määritellyn ajanjakson välein tai heti toisen CI-putken suorituksen päätteeksi.



Kuva 16. CI-putken työnkulku.

5.3.1 Koodien testaaminen

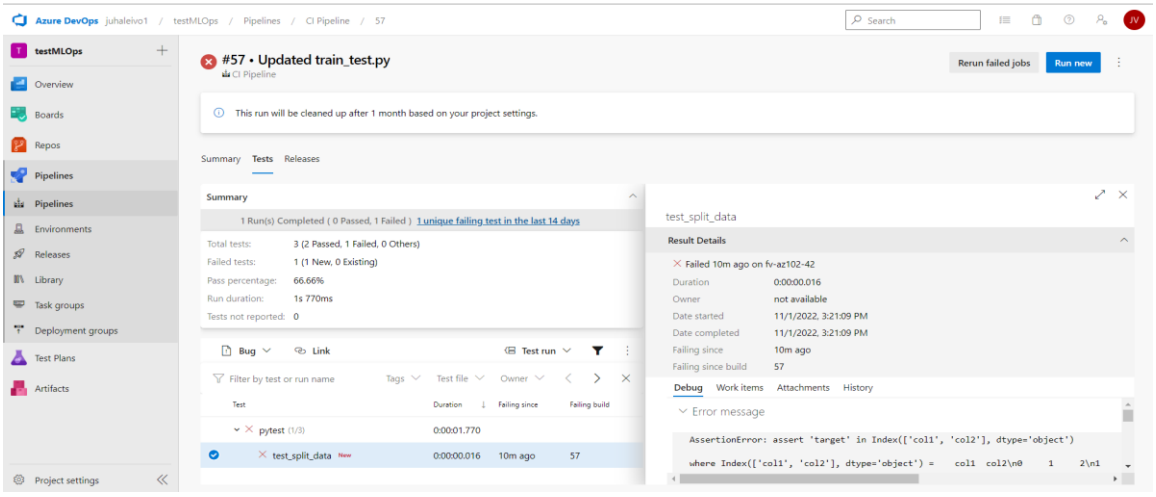
Ennen koneoppimismallin opettamista, on hyvien ohjelmistokehityskäytänteiden mukaista testata koodit. Näin voidaan varmistaa, ettei uudet koodimuutokset ole rikkoneet mitään putken toiminnallisuuksia. Jos jokin testi ei mene läpi, katkeaa putkien suorittaminen. Aina, kun jokin uusi toiminnallisuus lisätään putkeen, tulee sille kirjoittaa kattavat testit.

Tässä toteutuksessa testit ajetaan osana koulutusputkea. Suositeltavaa kuitenkin olisi luoda esimerkiksi erillinen CI-putki, jossa testit suoritetaan, kun mihin tahansa versionhallinnan haaraan tehdään muutoksia. Näin kaikki muutokset saadaan testattua, ennen kuin ne liitetään päähaaraan ja koneoppimismallin koulutus käynnistetään.

Pythonilla kirjoitetut testit suoritetaan Pytest-kirjastolla ja tuloksista muodostetaan raportti Pytest-cov -kirjastolla (kts. Komento 1). Raportti saadaan julkaistua CI-putken suorituskerran Tests-välilehdelle (Kuva 17) käyttämällä ”Publish Test Results” -työtehtävää.

```
pytest training/train_test.py --doctest-modules --junitxml=junit/test-results.xml --cov=data_test --cov-report=xml --cov-report=html
```

Komento 1. Testien suorittaminen CI-putkessa.

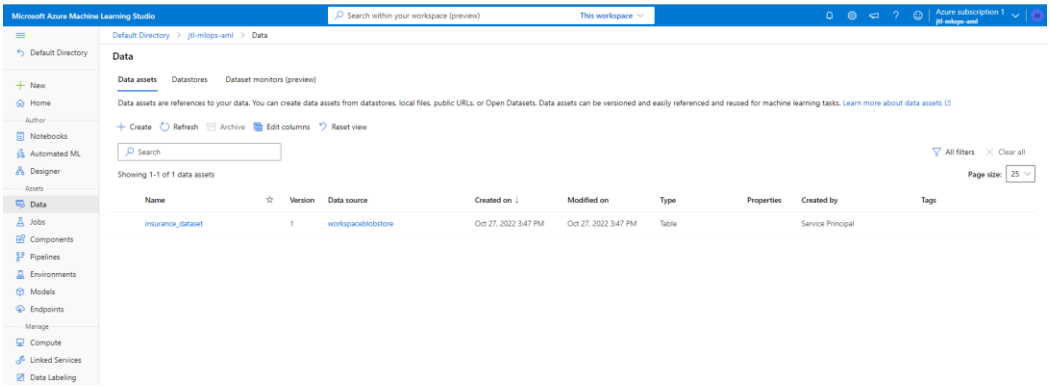


Kuva 17. Putken suorituksen aikana tehtyjen testien tulokset löytyvät kyseisen suorituskerran Tests-välilehden alta. Jokaisesta epäonnistuneesta testistä löytyy tieto, miksi testi epäonnistui.

5.3.2 Koneoppimismallin kouluttaminen

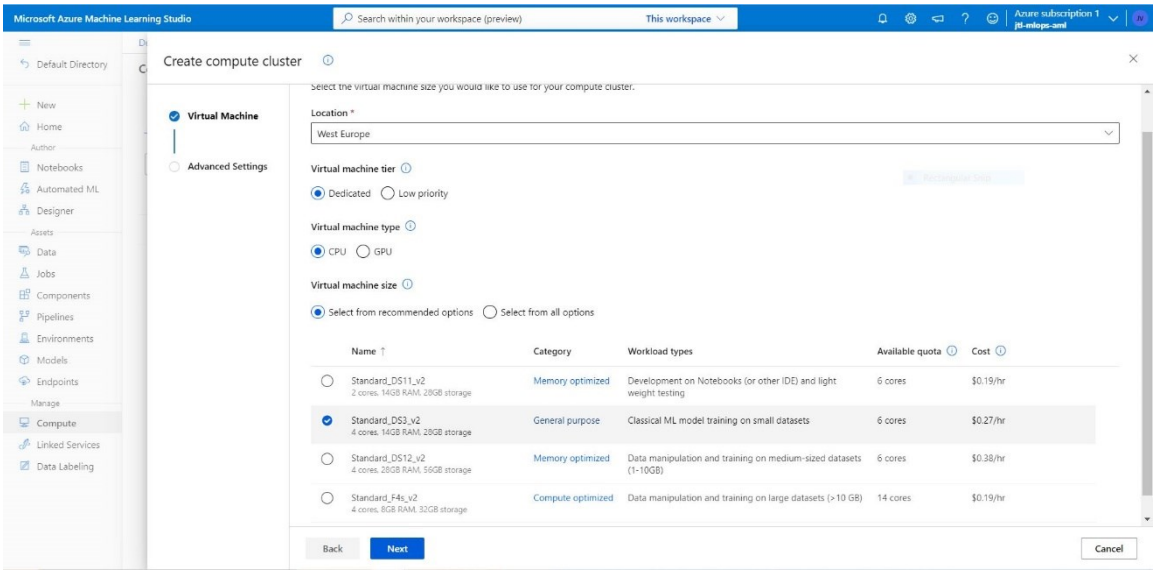
Tässä työssä käytetään valmiiksi esikäsiteltyä dataa, joka on rekisteröity Azure Machine Learning -työtilaan (Kuva 18). Datan käsittely on mahdollista toteuttaa omassa putkessaan, minkä suorittamisen jälkeen koneoppimismallin kouluttamisputki käynnistyy automaattisesti. Azure DevOps -palveluun on mahdollista kytkeä esimerkiksi Azure Data Factory -palvelu [62], joka tarjoaa monipuolisia työkaluja datan käsittelemiseen.

Koulutuksessa käytetty data haetaan Azure Machine Learning -työtilasta Azure Python SDK -kirjaston avulla. Tyypillisesti koneoppimisprojekteissa koneoppimismalli koulutetaan uudestaan, kun uutta dataa on saatavilla. Tällöin versioinnin avulla on helppo seurata, mitä dataa on käytetty minkäkin koneoppimismallin kouluttamiseen. Nyt käytössä on aina sama data, joten versio pysyy samana.



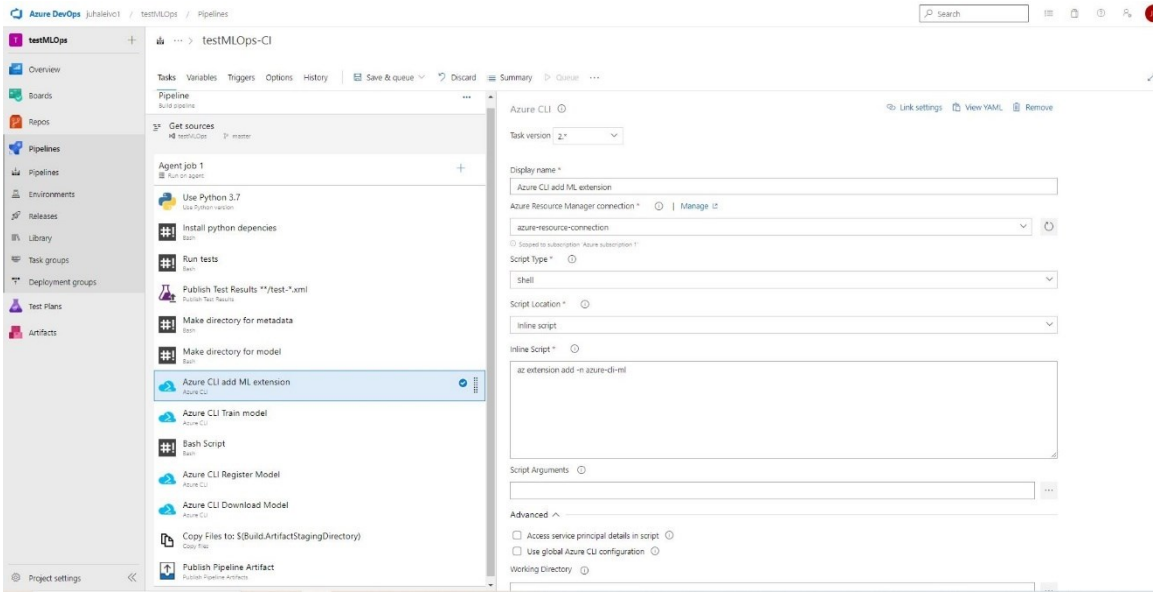
Kuva 18. Azure Machine Learning -työtilaan rekisteröity data.

Koneoppimismallin kouluttaminen tapahtuu Azure Machine Learning -työtilaan luodussa laskentaklusterissa. Laskentaklusterin luomisen yhteydessä määritetään, käytetäänkö CPU- vai GPU-laskentaa ja miten paljon resursseja sille varataan (Kuva 19). Laskentaklusterin taso (engl. tier) tulee myös määritellä. Tasoksi voidaan valita joko ”Low priority” tai ”Dedicated”, joista ensin mainittu on 80 % halvempi, mutta se ei ole aina välttämättä heti saatavilla ja sillä toteutettava koulutus saatetaan katkaista kesken kaiken, jos samalla alueella sijaitsevissa virtuaalikoneissa on ruuhkaa.



Kuva 19. Laskentaklusterin luominen Azure Machine Learning -työtilassa.

Azure CLI -komentorivityökalu suorittaa sille määritetyt komennot CI-putken suoritusympäristössä. Palveluyhteyksiä käyttämällä komentorivityökalu kirjautuu automaattisesti käyttäjällä sisään ja saa oikeudet hallita Azure Machine Learning -työtilan resursseja. Komentorivityökaluun tulee asentaa Azure-cli-ml -laajennus, jotta sillä voidaan kontrolloida Azure Machine Learning -työtilan resursseja (Kuva 20).



Kuva 20. Azure-cli-ml -laajennuksen asentaminen Azure CLI -komentorivityökalulle.

az ml run submit-script -g $(azureml.resourceGroup) -w $(azureml.workspaceName) -e $(experiment.name) --ct $(amlcompute.clusterName) -d conda_dependencies.yaml -c train_insurance -t ../metadata/run.json train_aml2.py

Komento 2. Koulutuksen käynnistäminen.

Komennossa (kts. Komento 2) määritetään, missä resurssiryhmässä ja työtilassa laskentaklusteri sijaitsee, millä nimellä suoritus tallennetaan työtilaan, mitä laskentaklusteria käytetään, millainen Conda-ympäristö laskentaklusteriin luodaan, mitä konfiguraatiotiedostoa käytetään, mihin suorituksen tiedot tallennetaan ja mikä tiedosto siellä suoritetaan.

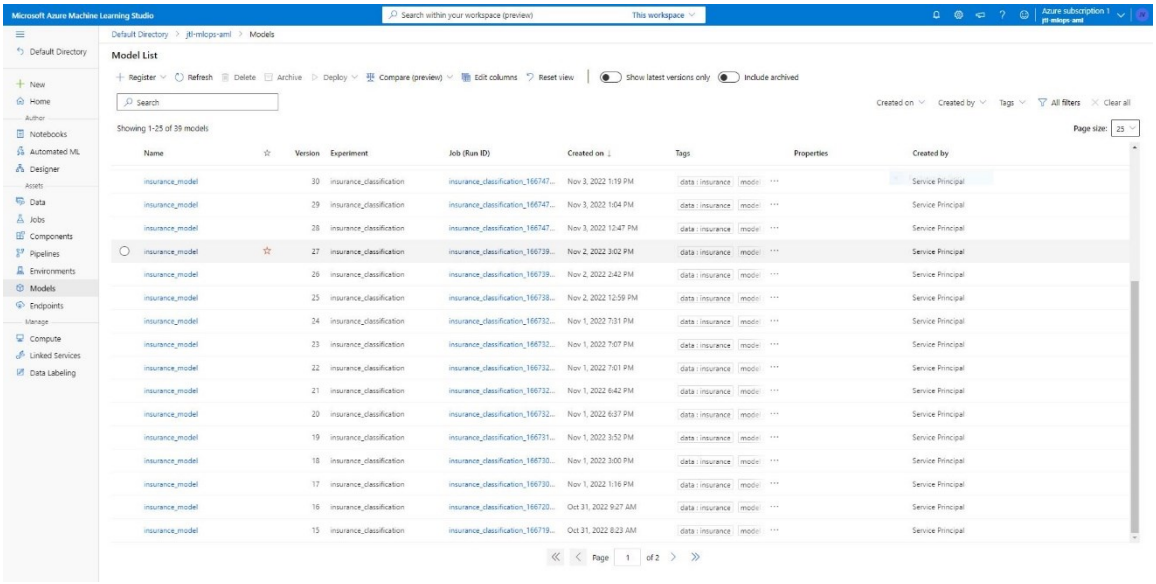
5.3.3 Koneoppimismallin rekisteröinti

Koulutettu koneoppimismalli ja siihen liittyvä metadata tallennetaan koulutuksen päätteeksi CI-putken suoritusympäristön kansiorakenteeseen, mistä ne voidaan rekisteröidä Azure Machine Learning -työtilaan (Kuva 21). Rekisteröidyn koneoppimismallin koulutukseen liittyviä tietoja voi tarkastella Jobs-välilehdeltä (Kuva 22).

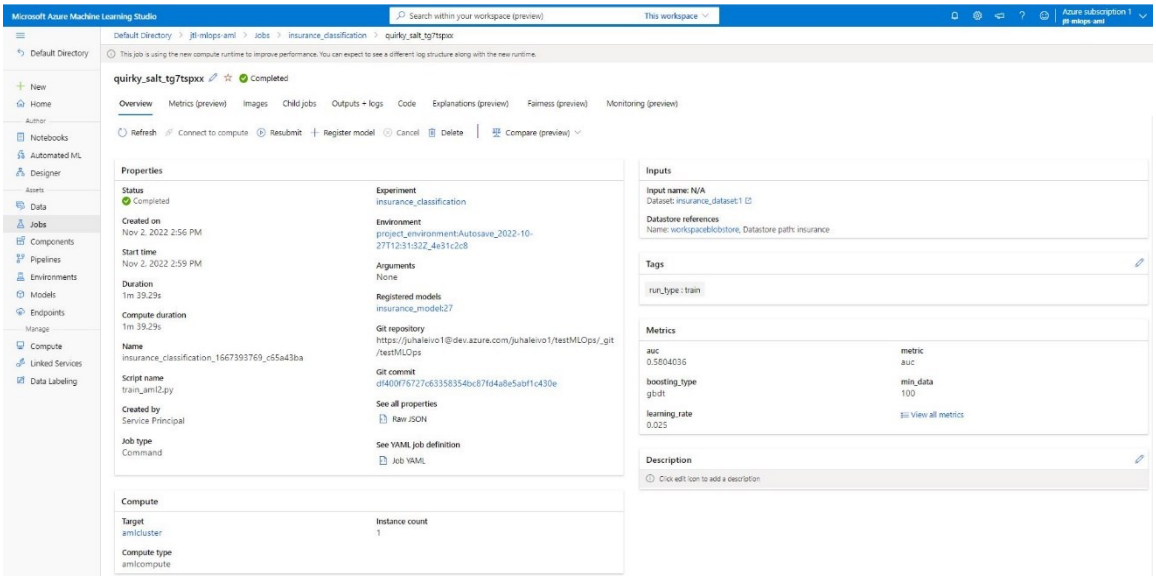
```
az ml model register -g $(azureml.resourceGroup) -w $(azureml.workspaceName) -n $(model.name) -f
metadata/run.json --asset-path outputs/models/insurance_model.pkl -d "Classification model for filling
a claim prediction" --tag "data"="insurance" --tag "model"="classification" --model-framework
ScikitLearn -t metadata/model.json
```

Komento 3. Koulutetun koneoppimismallin ja siihen liittyvän metadatan rekisteröimisestä Azure Machine Learning -työtilaan.

Komennossa (kts. Komento 3) määritetään, mihin työtilaan tiedot halutaan tallentaa, millä nimellä malli rekisteröidään, mistä kaikki tiedot löytyvät ja mitä kirjastoa koneoppimismallin kouluttamiseen on käytetty. Koneoppimismallille voidaan lisätä myös sanallinen kuvaus sekä tarkentavia tageja. Lisätietoa käytettävistä parametreista ja tuetuista koneoppimismallin tallennusmuodoista löytyy Azuren dokumentaatiosta [63].



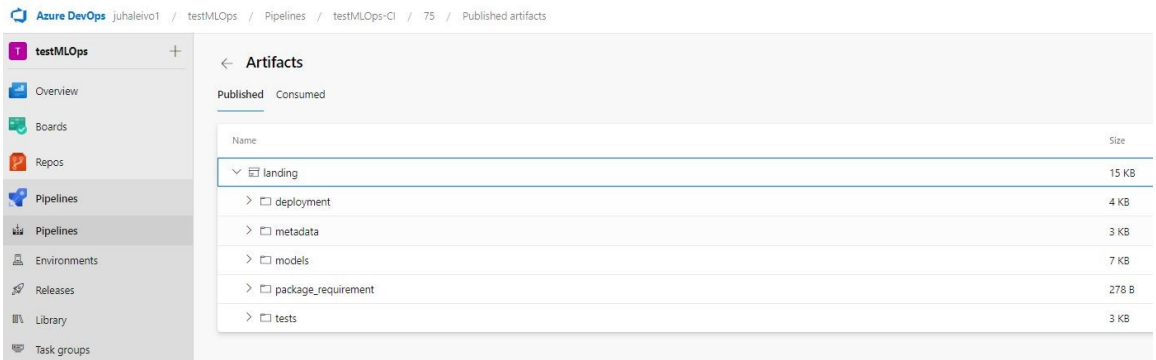
Kuva 21. Azure Machine Learning -työtilaan rekisteröidyt koneoppimismallit.



Kuva 22. Kaikki koulutukseen liittyvä tieto tallennetaan ja ne ovat nähtävillä Azure Machine Learning -työtilassa Jobs-välilehdellä.

5.3.4 Artefaktin luominen

Artefakteja käytetään tiedostojen siirtämiseen CI-putkesta CD-putkeen. Artefaktit luodaan kopioimalla valitut tiedostot CI-putken Build.SourcesDirectory -kansiosta Build.ArtifactStagingDirectory -kansioon. Artefakti julkaistaan putken päätteeksi ”Publish Pipeline Artifact” -työtehtävän avulla. Artefaktiin kannattaa tallentaa vain ne tiedostot, joita tarvitaan koneoppimismallin testaamiseen ja julkaisemiseen (Kuva 23).

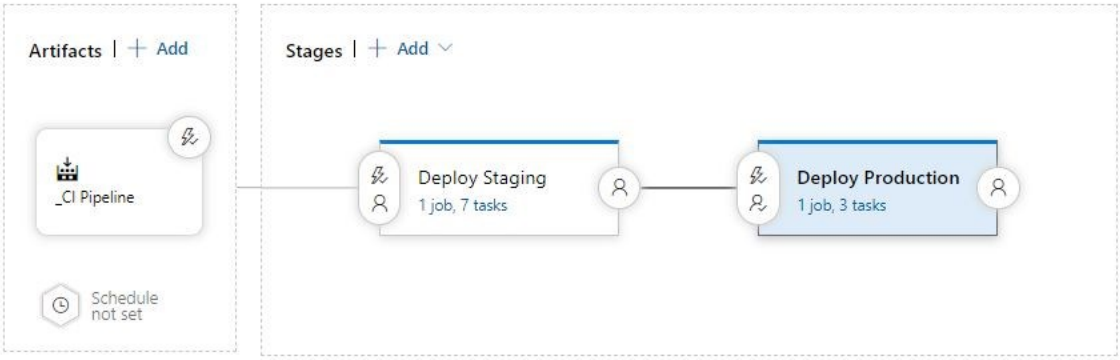


Kuva 23. CI-putken päätteeksi luotu artefakti.

5.4 CD-putki

CD-putkea käytetään koneoppimismallin tuotantoon viemiseen (Kuva 24). Koneoppimismalli julkaistaan ensin staging-ympäristössä, jossa sen toiminta testataan. Jos kaikki toimii oikein staging-ympäristössä, voidaan koneoppimismalli julkaista tuotantoympäristöön. CD-putki käynnistyy automaattisesti, kun CI-putken päätteeksi luodaan uusi artefakti, jolloin tuotannossa pyörii aina tuorein koneoppimismallin versio.

Kuten CI-putkessa, tulee myös CD-putken suoritusympäristöön asentaa Python-tulkki, koodin suorittamiseen tarvittavat riippuvuudet sekä Azure CLI-komentorivityökaluun azure-cli-ml -laajennus. Myös tarvittavat muuttujat määritellään samalla tavalla kuin CI-putkessa.



Kuva 24. CD-putken työvaiheet.

5.4.1 Koneoppimismallin julkaiseminen ja testaaminen staging-ympäristössä

Staging-ympäristö on kopio tuotantoympäristöstä. Koneoppimismalli julkaistaan ja sen toiminnallisuus testataan ensin staging-ympäristössä, ennen kuin se julkaistaan tuotantoympäristöön. Näin voidaan varmistaa, ettei tuotantoympäristössä pääse tapahtumaan mitään käyttökatkoksia aiheuttavia virheitä, kun uusi koneoppimismalli julkaistaan sinne. Staging- ja tuotantoympäristölle luodaan omat klusterit Azure Kubernetes Service -palveluun.

az ml computetarget create aks -g $(azureml.resourceGroup) -w $(azureml.workspaceName) -n $(aks.clusterNameStaging) -s $(aks.vmSizeStaging) -a $(aks.agentCountStaging)

Komento 4. Azure Kubernetes Service -palvelun luominen Azure CLI -komentorivityökalulla.

Koneoppimismallin julkaisemisen yhteydessä määritetään, millainen ympäristö luotuun klusteriin tarvitsee luoda, jotta koneoppimismallia voidaan käyttää siellä. Python-tiedostossa, jossa määritetään, kuinka koneoppimismallilla tehdään ennustuksia, täytyy olla Init() ja Run() -funktiot (kts. Koodi 1).

```
import json
import numpy
import joblib
import time
from azureml.core.model import Model

def init():
    global MODEL

    model_path = Model.get_model_path("insurance_model")
    MODEL = joblib.load(model_path)


def run(raw_data, request_headers):
    data = json.loads(raw_data)["data"]
    data = numpy.array(data)
    result = MODEL.predict(data)

    info = {
        "input": raw_data,
        "output": result.tolist()
        }
    print(json.dumps(info))

    print(('{{"RequestId":"{0}", '
           '"TraceParent":"{1}", '
           '"NumberOfPredictions":{2}}}'
           ).format(
               request_headers.get("X-Ms-Request-Id", ""),
               request_headers.get("Traceparent", ""),
               len(result)
    ))

    return {"result": result.tolist()}
```

Koodi 1. Ennustuksien tekeminen koneoppimismallilla.

az ml model deploy -g $(azureml.resourceGroup) -w $(azureml.workspaceName) -n $(service.name.staging) -f ../metadata/model.json --dc AKSDeploymentConfigStaging.yaml --ic inferenceConfig.yaml --ct $(aks.clusterNameStaging) --overwrite

Komento 5. Koneoppimismallin julkaiseminen luotuun staging-ympäristöön.

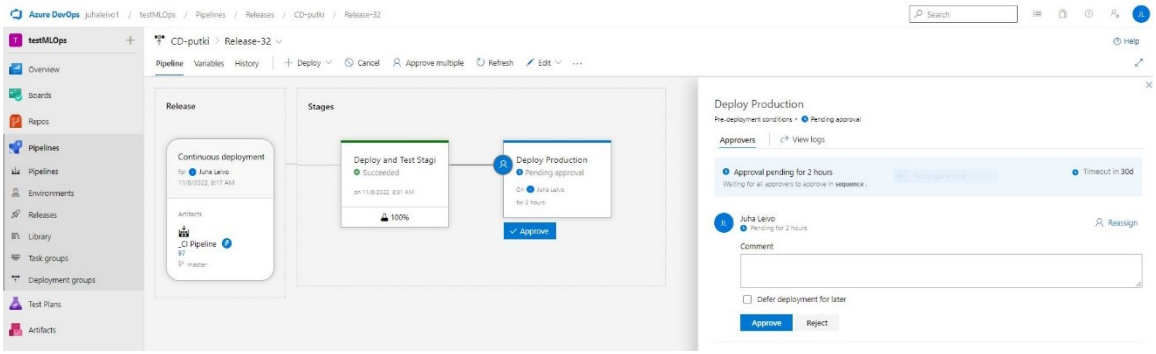
Komento luo koneoppimismallista ja sen suorittamiseen tarvittavista tiedostoista Docker-kuvan, jota suoritetaan laskentaklusterissa yaml-tiedostoissa määriteltyjen asetusten mukaisesti. Koneoppimismalli toimii REST API:na ja sen toiminta voidaan varmistaa esimerkiksi lähettämällä sille

testidataa http-pyyntönä ja tarkistaa, tuleeko vastaus takaisin OK-statuskoodilla ja sisältääkö se saman verran ennustuksia kuin ennustettavia rivejä lähetettiin.

5.4.2 Koneoppimismallin julkaiseminen tuotantoympäristöön

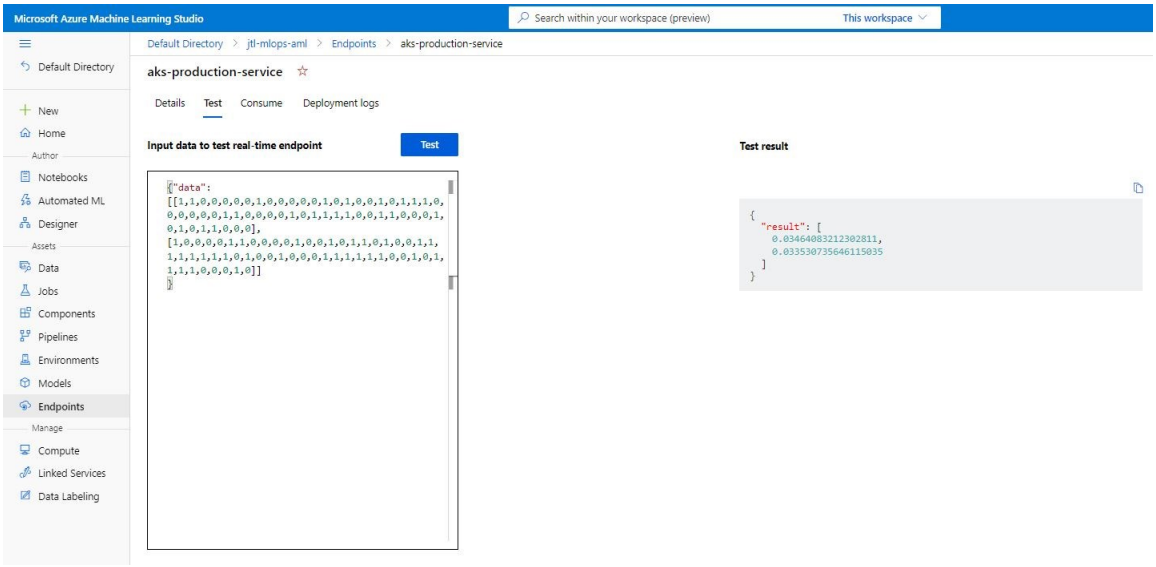
Koneoppimismalli julkaistaan tuotantoympäristöön samalla tavalla kuten staging-ympäristöön. Ainoana erona on, ettei mallia tarvitse välttämättä enää testata uudelleen tuotantoympäristössä. Koneoppimismallit voidaan julkaista Azure Kubernetes Service -palvelun lisäksi esimerkiksi Azure Container Instance -palveluun. Azure Kubernetes Service -palvelua suositellaan kuitenkin käytettäväksi, koska siellä Kubernetes hoitaa uuden mallin julkaisemisen automaattisesti siten, ettei palvelu ole alhaalla hetkeäkään. Lisäksi samaan klusteriin voidaan esimerkiksi julkaista sovellus, josta tehdään kyselyjä koneoppimismallille. Tällöin Kubernetesin avulla voidaan pitää huoli, että sovelluksen ja koneoppimismallin välinen tietoliikenne pysyy aina kunnossa.

Koneoppimismalli voidaan asettaa julkaistavaksi tuotantoympäristöön automaattisesti, jos testit menevät staging-ympäristössä läpi. Azure DevOps -palvelussa on myös mahdollista valita projektiin kuuluva henkilö, joka käy manuaalisesti hyväksymässä, viedäänkö uusi koneoppimismalli tuotantoon (Kuva 25). Azure lähettää valitulle henkilölle sähköpostin, kun uusi julkaisu odottaa hyväksyntää.



Kuva 25. Koneoppimismallin julkaiseminen tuotantoympäristöön odottaa käyttäjän hyväksyntää.

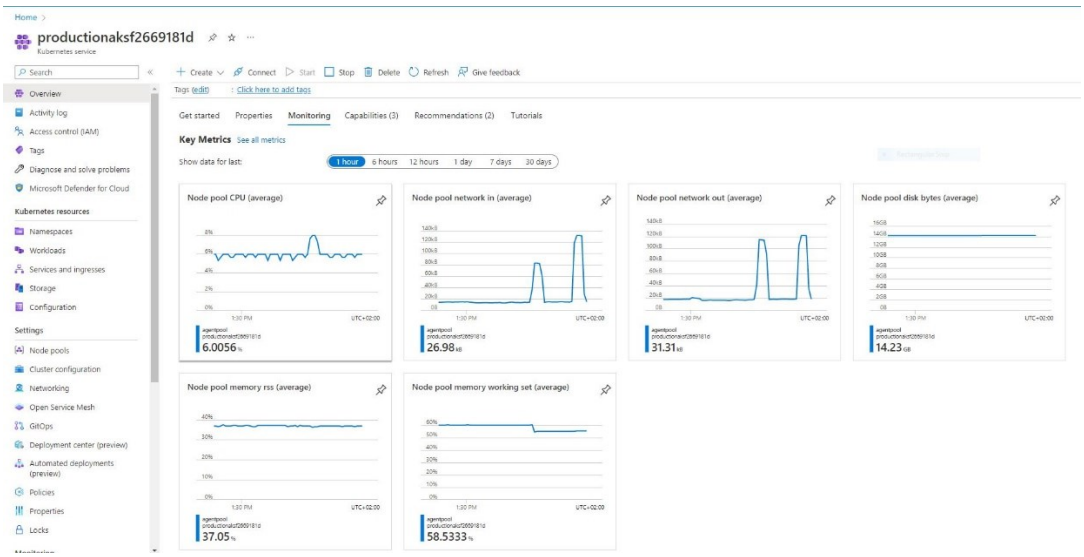
Azure Machine Learning -työtilasta löytyy kaikki julkaistuun koneoppimismalliin liittyvä tieto ”Endpoints”-osiosta. Mallia on mahdollista testata Azuren kautta ”Test”-välilehden kautta (Kuva 26). Azure myös generoi käyttäjälle koodit koneoppimismallin käyttöä varten C#-, R- ja Python-ohjelmointikielille.



Kuva 26. Tuotantoympäristössä olevaa koneoppimismallia voidaan kokeilla ”Test”-välilehdessä.

5.4.3 Monitorointi

Azure Kubernetes Service -palveluun julkaistuja sovelluksia monitoroidaan automaattisesti. Monitoroinnilla seurataan esimerkiksi tuotantoympäristöklusterin tietoliikenteen määrää, resurssien käyttöä sekä terveydentilaa (Kuva 27). Monitoroinnin avulla voidaan helposti huomata, jos esimerkiksi resursseja tarvitsee lisätä tai vähentää.



Kuva 27. Tuotantoympäristöklusterin resurssien monitorointi.

Datan ajelehtimisen (engl. Data drift) monitorointi on vasta julkaistu uutena toimintona Azure Python SDK -kirjastolle, mutta se on vielä ”preview”-tilassa, eli sitä ei suositella vielä käytettäväksi tuotantoympäristössä. Kirjaston avulla voidaan esimerkiksi asettaa hälytys, jos tuotantoympäristöstä kerätty data alkaa poikkeamaan koulutuksessa käytetystä datasta. [64.] Tällöin tiedetään hyvissä ajoin, että koneoppimismalli on aika kouluttaa uudella datalla, ennen kuin koneoppimismalli alkaa tekemään virheellisiä päätöksiä. Datan monitorointia ei tässä työssä toteutettu, mutta se on hyvin oleellinen osa automatisoitua MLOps-putkea.

6 Yhteenveto

Tässä opinnäytetyössä tutustuttiin, mitä kaikkia eri työvaiheita koneoppimismallin luomiseen sekä tuotantoympäristöön viemiseen vaaditaan. MLOps on kokoelma hyväksi havaittuja käytänteitä ja toimintatapoja, joiden avulla koneoppimismallien kehitystyöstä ja tuotantoympäristöön viemisestä saadaan tehtyä entistä nopeampaa, toimintavarmempaa ja tehokkaampaa.

MLOps-käytänteiden mukaisesti toteutetut koneoppimisprojektit ovat kuitenkin suhteellisen hankalasti toteutettavia ja niiden toteuttamiseen luodut työkalut vielä melko nuoria ja jatkuvat kehityksen alla. Usein koneoppimisprojektit sisältävät useita eri työvaiheita, jotka voivat olla hyvinkin kompleksisia ja joiden kehittämiseen kuluu paljon aikaa. Tekoälypioneeri Andrew Ng kehottaakin kirjoituksessaan [3] lähtemään liikkeelle siitä, että luodaan aluksi mahdollisimman yksinkertainen kokonaisuus, jolla koneoppimismalli saadaan automatisoidusti koulutettua ja vietyä tuotantoympäristöön. Vasta tämän jälkeen kannattaa lähteä parantamaan koko prosessia yksi askel kerrallaan.

Tässä opinnäytetyössä luotiin yksinkertainen Azure DevOps -palvelun CI/CD-putkia käyttävä MLOps-putki. Koko kokonaisuus voidaan luoda helposti seuraamalla Mohammad Ghodratigoharin luomaa videosarjaa [56] sekä käyttämällä hänen tekemiään lähdekoodeja [57]. Putkesta puuttuu paljon tärkeitä ominaisuuksia, kuten koneoppimismallin tekemien ennusteiden tallentaminen ja koulutusdataan vertaaminen, koneoppimismallin automatisoitu uudelleenkouluttaminen sekä koulutetun koneoppimismallin hyvyyden vertaaminen tuotannossa olevaan koneoppimismalliin. Se on kuitenkin toimiva kokonaisuus, ja sitä voi lähteä helposti muokkaamaan omien tarpeiden mukaan osio kerrallaan. Tämä tekee siitä toimivan pohjaratkaisun esimerkiksi uusille koneoppimisprojekteille.

On kuitenkin hyvä huomata, että tämän opinnäyteyön tekemisen aikana julkaistiin Azure Python SDK -kirjaston uusi versio Azure Python SDK v2. Tässä työssä on käytettä vanhempaa Azure Python SDK v1 -versiota. Microsoft suosittelee, että kaikissa uusissa projekteissa otetaan käyttöön uudempi versio, sillä kaikki uudet toiminnallisuudet ja päivitykset tullaan julkaisemaan vain sille. Vanhempi versio on kuitenkin vielä täysin käyttökelpoinen, eikä sille ole vielä julkaistu vanhenemispäivää. [65.] Microsoft Azuren dokumentaatiosta löytyy ohjeistus, miten vanhempi Azure Python SDK v1 voidaan päivittää uudempaan v2-versioon [66].

Tämän opinnäyteyön piti alun perin käsitellä, miten KubeFlowlla (avoimen lähdekoodin MLOps-alusta, joka on rakennettu toimimaan Kubernetesin päällä) saadaan luotua automatisoitu MLOps-putki. KubeFlow osoittautui kuitenkin hyvin haastavaksi saada toimimaan oikein, jos ei ole vahvaa kokemusta Kubernetesistä. KubeFlowiin verrattuna Azuren palveluita hyödyntämällä oli verrattain helppoa saada luotua toimiva MLOps-putki, sillä Azuren dokumentaatio on huomattavasti paremmin ajan tasalla verrattuna KubeFlowin dokumentaatioon. Lisäksi Azurelle löytyy paljon enemmän esimerkkejä (kuten tässä työssä käytetty), joiden avulla varsinkin ensikertalaisen on huomattavasti helpompi päästä alkuun MLOps-putkien kanssa. Sen lisäksi, että Azuren palveluiden avulla sai paljon nopeammin ja helpommin toimivan MLOps-putken luotua, on se myös, ainakin tämänkaltaisen pienemmän mittakaavan projektissa, huomattavasti halvempi verrattuna KubeFlowiin. Pelkästään KubeFlowin käynnissä pitäminen Azure Kubernetes Service -palvelussa maksoi noin 10 euroa päivässä. Azure Machine Learning ja Azure DevOps -palveluiden käyttämisen hinnaksi tuli noin 4 euroa päivässä, johon sisältyi koneoppimismallin suorittaminen tuotantoympäristössä sekä 7 yksinkertaisen koneoppimismallin kouluttaminen pienellä datamäärällä.

Lähteet

[1] Kreuzberger D, Kühl N, Hirsch D. Machine Learning Operations (MLOps): Overview, Definitio, and Architecture [Internet]. Saatavilla: 2022; doi: 10.48550/arXiv.2205.02302

[2] Sculley D, Holt G, Golovin D, Davydov E, Phillips T, Ebner D, Chaudhary V, Yong M, Crespo J, Dennison D. Hidden Technical Debt in Machine Learning Systems. [Internet] [Viitattu 8.10.2022]. Saatavilla: https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

[3] Ng A. Iteration in AI Development [Internet] [Viitattu 8.10.2022]. Saatavilla: https://www.deeplearning.ai/the-batch/iteration-in-ai-development/

[4] Brain J. How to Make a Machine Learning Project More Likely to Succeed? [Internet] [Viitattu 12.10.2022]. Saatavilla: https://neptune.ai/blog/how-to-make-machine-learning-project-more-likely-to-succeed

[5] Descoins A. 11 questions to ask before starting a successful Machine Learning project [Internet] [Viitattu 18.10.2022]. Saatavilla: https://tryolabs.com/blog/2019/02/13/11-questions-to-ask-before-starting-a-successful-machine-learning-project

[6] What is ETL? [Internet] [Viitattu 9.9.2022]. Saatavilla: https://www.ibm.com/topics/etl

[7] What is ELT? [Internet] [Viitattu 9.9.2022]. Saatavilla: https://www.ibm.com/topics/elt

[8] What is exploratory data analysis? [Internet] [Viitattu 10.9.2022]. Saatavilla: https://www.ibm.com/topics/exploratory-data-analysis

[9] Feature Engineering [Internet] [Viitattu 9.9.2022]. Saatavilla: https://www.dominodatalab.com/data-science-dictionary/feature-engineering

[10] Weedmark D. Machine Learning Model Training: What it Is and Why it's Important [Internet] [Viitattu 10.9.2022]. Saatavilla: https://www.dominodatalab.com/blog/what-is-machine-learning-model-training

[11] Damji J, Galarnyk M. Considerations for Deploying Machine Learning Models in Production [Internet] [Viitattu 6.12.2022]. Saatavilla: https://www.anyscale.com/blog/considerations-for-deploying-machine-learning-models-in-production

[12] MLOps Basics [Internet] [Viitattu 8.12.2023]. Saatavilla: https://valohai.com/mlops/

[13] Klushin P. What does it tako to deploy ML models in production? [Internet] [Viitattu 8.11.2022]. Saatavilla: https://www.qwak.com/post/what-does-it-take-to-deploy-ml-models-in-production

[14] Machine learning inference during deployment [Internet] [Viitattu 11.11.2022]. Saatavilla https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/innovate/best-practices/ml-deployment-inference

[15] Why do you hear Kubernetes and Machine Learning together so often? [Internet] [Viitattu 23.11.2022]. Saatavilla: https://ubiops.com/why-do-you-hear-kubernetes-and-machine-learning-together-so-often/

[16] Skogström H. The Three Roles in a Machine Learning Team (and Two Technologies to Connect Them) [Internet] [Viitattu 14.9.2022]. Saatavilla: https://valohai.com/blog/the-three-roles-in-an-ml-team/

[17] Weedmark D. 7 Key Roles and Responsibilities in Enterprise MLOps [Internet] [Viitattu 14.9.2022]. Saatavilla: https://www.dominodatalab.com/blog/7-roles-in-mlops

[18] Raitanen N. MLOps – what is it and what can you do with it? [Internet] [Viitattu 31.8.2022]. Saatavilla: https://futurice.com/blog/mlops-what-is-it-and-what-can-you-do-with-it

[19] What is MLOps? [Internet] [Viitattu 1.11.2022]. Saatavilla: https://www.databricks.com/glossary/mlops

[20] MLOps: Continuous delivery and automation pipelines in machine learning [Internet] [Viitattu 3.12.2022]. Saatavilla: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning#top_of_page

[21] MLOps Principles – Reproducibility [Internet] [Viitattu: 24.1.2023]. Saatavilla: https://ml-ops.org/content/mlops-principles#reproducibility

[22] Lev A. Why reproducibility is important for ML [Internet] [Viitattu 24.1.2023]. Saatavilla: https://www.qwak.com/post/why-reproducibility-is-important-for-ml

[23] Patel H. Version Control Guide For Machine Learning Researchers [Internet] [Viitattu 24.1.2023]. Saatavilla: https://neptune.ai/blog/version-control-guide-for-machine-learning-researchers

[24] Hashes A. Version Control for ML Models: Why You Need It, What It Is, How To Implement It [Internet] [Viitattu 24.11.2023]. Saatavilla: https://neptune.ai/blog/version-control-for-ml-models

[25] Onose E. How to Solve Reproducibility in ML [Internet] [Viitattu 24.1.2023]. Saatavilla: https://neptune.ai/blog/how-to-solve-reproducibility-in-ml

[26] Komolafe A. Top Model Versioning Tools for Your ML Workflow [Internet] [Viitattu 24.1.2023]. Saatavilla: https://neptune.ai/blog/top-model-versioning-tools

[27] Higuchi T. MLOps Blog Series Part 1: The art of the machine learning systems using MLOps [Internet] [Viitattu 17.1.2023]. Saatavilla: https://azure.microsoft.com/en-us/blog/mlops-blog-series-part-1-the-art-of-testing-machine-learning-systems-using-mlops/

[28] Zvorničanin E. Automated Testing in Machine Learning Projects [Best Practices for MLOps] [Internet] [Viitattu 17.1.2023]. Saatavilla: https://neptune.ai/blog/automated-testing-machine-learning

[29] MLOps Principles – Monitoring [Internet] [Viitattu 15.1.2023]. Saatavilla: https://ml-ops.org/content/mlops-principles#monitoring

[30] Breck E, Cai S, Nielsen E, Salib M, Sculley D. The ML Test Score: A Rubric for ML Production Readiness and Techical Dept Reduction [Internet]. Saatavilla: 2017; doi: 10.1109/Big-Data.2017.8258038

[31] Use containers to Build, Share and Run your applications [Internet] [Viitattu 10.10.2022]. Saatavilla: https://www.docker.com/resources/what-container/

[32] Rubens B. What are containers and why do you need them? [Internet] [Viitattu 14.2.2022]. Saatavilla: https://www.cio.com/article/247005/what-are-containers-and-why-do-you-need-them.html

[33] Gillis S. Definiton Docker image [Internet] [Viitattu 10.10.2022]. Saatavilla: https://www.techtarget.com/searchitoperations/definition/Docker-image

[34] Introduction to Microservices: What are Microservices? Use Cases and Examples [Internet] [Viitattu 11.10.2022]. Saatavilla: https://www.datarobot.com/blog/introduction-to-microservices/

[35] Microservices-based Architecture: Key to Scaling Enterprise ML Models [Internet] [Viitattu 11.10.2022]. Saatavilla: https://www.sigmoid.com/blogs/microservices-based-architecture-key-to-scaling-enterprise-ml-models/

[36] Kubernetes Documentation [Internet] [Viitattu 12.10.2022]. Saatavilla: https://kubernetes.io/docs/concepts/overview/

[37] Koté M. Why Large Organizations Trust Kubernetes [Internet] [Viitattu 12.10.2022]. Saatavilla: https://tanzu.vmware.com/content/blog/why-large-organizations-trust-kubernetes

[38] Kubernetes for MLOps Engineers [Internet] [Viitattu 12.10.2022]. Saatavilla: https://pages.run.ai/hubfs/PDFs/Kubernetes-for-MLOps-Engineers.pdf

[39] Pittet S. Continuous Integrations vs. delivery vs deployment [Internet] [Viitattu 12.10.2022]. Saatavilla: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

[40] What is CI/CD For Machine Learning? [Internet] [Viitattu 13.10.2022]. Saatavilla: https://valohai.com/cicd-for-machine-learning/

[41] What is Azure? [Internet] [Viitattu 3.12.2022]. Saatavilla: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/

[42] Klint L. What is Azure? Microsoft´s cloud platform explained [Internet] [Viitattu 3.12.2022]. Saatavilla: https://acloudguru.com/blog/engineering/what-is-microsoft-azure

[43] 10 reasons why to choose Azure for your Enterprise [Internet] [Viitattu 2.26.2022]. Saatavilla: https://www.saviantconsulting.com/blog/10-reasons-why-choose-microsoft-azure.aspx

[44] What is Azure Machine Learning? [Internet] [Viitattu 6.12.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-machine-learning

[45] How Azure Machine Learning works: resources and assets [Internet] [Viitattu: 7.12.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/machine-learning/concept-azure-machine-learning-v2?tabs=cli

[46] Kuva: Azure Machine Learning taxonomy [Internet] [Viitattu 7.12.2022]. Saatavilla: https://docs.microsoft.com/en-us/azure/machine-learning/media/concept-workspace/azure-machine-learning-taxonomy.png

[47] Azure Machine Learning glossary [Internet] [Viitattu 7.12.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/machine-learning/azure-machine-learning-glossary#compute

[48] What is Azure Machine Learning component? [Internet] [Viitattu 7.12.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/machine-learning/concept-component

[49] What is Azure DevOps? [Internet] [Viitattu 9.12.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops#choose-azure-devops-services

[50] What is Azure Repos? [Internet] [Viitattu 9.12.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops

[51] What is Azure Pipelines? [Internet] [Viitattu 9.12.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops

[52] Kaggle dataset: Porto Seguro´s Safe Driver Prediction [Internet] [Viitattu 15.11.2022]. Saatavilla: https://www.kaggle.com/c/porto-seguro-safe-driver-prediction

[53] LightGBM-dokumentaatio [Internet] [Viitattu 15.11.2022]. Saatavilla: https://lightgbm.readthedocs.io/en/latest/index.html

[54] skicit-learn: Decision Trees -dokumentaatio [Internet] [Viitattu 15.11.2022]. Saatavilla: https://scikit-learn.org/stable/modules/tree.html

[55] Brownlee J. A Gente Introduction to the Gradient Boosting Algorithm for Machine Learning [Internet] [Viitattu 15.11.2022]. Saatavilla: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

[56] Azure MLOps – DevOps for Machine Learning -soittolista [Internet] [Viitattu 16.11.2022]. Saatavilla: https://www.youtube.com/playlist?list=PLiQS6N-W1p3m9squzZ2cPgGdH5SBhjY6f

[57] MLOps-Workshop Github-repositorio [Internet] [Viitattu 16.11.2022]. Saatavilla: https://github.com/MG-Microsoft/MLOps_Workshop

[58] Azure DevOps Parallelism Request -kaavake [Internet] [Viitattu 7.11.2022]. Saatavilla: https://forms.office.com/pages/respon-sepage.aspx?id=v4j5cvGGr0GRqy180BHbR63mUWPlq7NEsFZhkyH8jChUMlM3QzdDMF-ZOMkVBWU5BWFM3SDI2QlRBSC4u&wdLOR=cFF8F4636-3F29-4BB0-AECC-C28F76D45A9B

[59] Pricing for Azure DevOps [Internet] [Viitattu 6.11.2022]. Saatavilla: https://azure.micro-soft.com/en-us/pricing/details/devops/azure-devops-services/

[60] Azure Pipelines task reference [Internet] [Viitattu 6.11.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/refe-rence/?view=azure-pipelines&viewFallbackFrom=azure-devops

[61] Use Azure Key Vault sercrets in Azure Pipelines [Internet] [Viitattu 6.11.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/devops/pipelines/release/azure-key-vault?view=azure-devops&tabs=yaml

[62] What is Azure Data Factory? [Internet] [Viitattu 8.11.2022]. Saatavilla: https://learn.mic-rosoft.com/en-us/azure/data-factory/introduction

[63] Azure CLI -dokumentaatio [Internet] [Viitattu 8.11.2022]. Saatavilla: https://learn.micro-soft.com/en-us/cli/azure/ml(v1)/model?view=azure-cli-latest#az-ml(v1)-model-register

[64] Detect data drift (preview) on datasets [Internet] [Viitattu 8.11.2022]. Saatavilla: https://learn.microsoft.com/en-us/azure/machine-learning/v1/how-to-monitor-data-sets?tabs=python

[65] What is Azure Machine Learning CLI & Python SDK v2? [Internet] [Viitattu 25.2.2023]. Saatavilla: https://learn.microsoft.com/en-us/azure/machine-learning/concept-v2

[66] Azure Machine Learning documentation - Upgrade to v2 [Internet] [Viitattu 25.2.2023]. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-migrate-from-v1