Erika Sternad

MLflow’n käyttö supertietokoneella

Insinööri (AMK)

Tieto- ja viestintätekniikka

Syksy 2024

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Sternad Erika

Työn nimi: MLflow'n käyttö supertietokoneella

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: machine learning, machine learning operations, artificial intelligence

Opinnäytetyön tarkoituksena oli kehittää MLflow-työkalun käyttöä CSC:n Puhti-supertietokoneella koneoppimismallien hallinnan ja versionhallinnan tueksi. MLflow on avoimen lähdekoodin työkalu, joka tarjoaa kattavat välineet mallien elinkaaren hallintaan, seurannan automatisointiin ja tulosten analysointiin. Työn keskeinen tavoite oli luoda selkeä ja helposti lähestyttävä tutoriaali, joka auttaa asiakkaita hyödyntämään MLflow'n ominaisuuksia tehokkaasti CSC:n laskentaympäristöissä ilman syvällistä teknistä taustaa.

Työssä käsiteltiin MLflow'n lisäksi laajemmin koneoppimismallien hallinnan haasteita ja esiteltiin MLOps (Machine Learning Operations) käsitteenä.

Työn tuloksena syntyi tutoriaali, jonka avulla käyttäjä tutustuu MLflow'n käyttöön CSC:n Puhdissa. Tutoriaaliin haluttiin sisällyttää kaikki MLflow'n perusominaisuudet, mutta kuitenkin pitää se mahdollisimman tiiviinä ja lyhyenä. Käyttäjän tulisi päästä tutoriaalin avulla hyvin alkuun MLflow'n käytössä, mutta sen tulisi myös madaltaa käyttäjän kynnystä hankkia lisätietoa aiheesta.

Tutoriaali on Jupyter Notebook -muodossa, ja se tullaan lisäämään CSC:n viralliseen dokumentaatioon, jossa se on maksutta kaikkien saatavilla. Projekti täyttää sille asetetut vaatimukset ja testaajien palaute on ollut positiivista, joten se voidaan arvioida onnistuneeksi. Lisäksi sekä tekijä että yritys saivat arvokasta tietoa MLflow'sta ja MLOpsista tutoriaalin kehitysprosessin myötä.

Abstract

Author(s): Sternad Erika

Title of the Publication: Using MLflow on Puhti Supercomputer

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: machine learning, machine learning operations, artificial intelligence

The purpose of this thesis was to develop the use of the MLflow tool on CSC's Puhti supercomputer to support the management and versioning of machine learning models. MLflow is an open-source tool that provides comprehensive tools for lifecycle management, automation of monitoring, and analysis of results for models. The main objective of the work was to create a clear and accessible tutorial that helps clients effectively utilize MLflow's features in CSC's computing environments without requiring an in-depth technical background.

The thesis also addressed the challenges of managing machine learning models and introduced MLOps (Machine Learning Operations) as a framework for handling these complexities.

The result of the work was a tutorial that helps the user get acquainted with the use of MLflow on CSC's Puhti supercomputer. The tutorial was designed to cover all the basic features of MLflow while keeping it as concise and short as possible. The goal was for the user to be able to get started with MLflow effectively through the tutorial, while also lowering the barrier to seeking further information on the topic.

The tutorial is presented in the form of a Jupyter Notebook, and it will be added to CSC's official documentation and made freely accessible to everyone. The project meets all requirements and has been deemed successful, with positive feedback from co-workers who tested it. Additionally, both the author and the company gained valuable insights into MLflow and MLOps through the development of this tutorial.

1 Johdanto 1

2 Koneoppiminen 2

2.1 Koneoppimismallien hallinta 3
2.2 Machine Learning Operations 5

3 MLflow 8

3.1 Komponentit 9
3.2 MLflow’n käyttö tilaajayrityksessä 10

4 Tutoriaali MLflow’n käyttöön Puhdissa 13

4.1 Tracking Server 15
4.2 Models 16
4.3 Model Registry 17
4.4 Projects 18

5 Tulokset 20

5.1 Tutoriaalin testaussuunnitelma 20
5.2 Palaute 22
5.3 Vaihtoehtoja MLflow’lle 23
5.4 Tutoriaalin jatkokehitys ja ylläpito 25
5.5 Tutoriaalin saatavuus 26

Symboliluettelo

Avoin lähdekoodi (open source)	Ohjelmisto, joka on vapaasti saatavilla ja muokattavissa.
Jupyter Notebook	Interaktiivinen ympäristö, jossa voi kirjoittaa ja suorittaa koodia sekä visualisoida ja dokumentoida työtä useilla eri ohjelmointikielillä.
Git- repositorio	Versionhallintaan käytettävä palvelu, johon tallennetaan projektin koodia ja muita tiedostoja.
Pipeline	Ohjelmistokehityksessä sarja vaiheita tai askelia, jotka suoritetaan tietyssä järjestyksessä.
REST API endpoint	Lyhenne sanoista Application Programming Interface. Se on yksittäinen verkko-osoite, jonka kautta voi lähettää pyyntöjä saadakseen tietoa tai suorittaakseen toimintoja. Jokaisella yksittäisellä endpointilla on oma toimintonsa järjestelmässä.
CI/CD	Lyhenne sanoista Continuous Integration / Continuous Delivery. Sillä tarkoitetaan ohjelmistokehityksen prosessia, jossa automatisoidaan koodin julkaisu ja käyttöönotto.
HTTP	Protokolla, joka mahdollistaa tiedonsiirron verkkoselaimen ja palvelimen välillä.

1 Johdanto

Tämän opinnäytetyön tilaajana toimii CSC – Tieteen tietotekniikan keskus Oy. Se on suomalainen valtion ja korkeakoulujen omistama tietotekniikan osaamiskeskus, jonka tehtävänä on kehittää ja tarjota palveluita muun muassa tutkimukselle, opetukselle, julkishallinnolle sekä myös yrityksille [1]. CSC hallinnoi Kajaanin datakeskuksessa kahta kansallista supertietokonetta, Mahtia ja Puhtia sekä kansainvälistä LUMIa [2]. Näistä Puhti on tämän opinnäytetyön keskiössä. Supertietokoneet eroavat tavallisista tietokoneista niin, että ne käyttävät samanaikaisesti useita keskusyksikköjä eli CPU:ita (Central Processing Unit), kun tavallisissa koneissa on yleensä vain yksi CPU. Nämä CPU:t on ryhmitelty nodeiksi, joita voi supertietokoneessa olla kymmeniätuhansia. Näiden nodejen välinen nopea yhteys on avainasemassa supertietokoneen suorituskyvyssä, sillä se mahdollistaa useiden CPU:iden tehokkaan käytön saman tehtävän suorittamiseksi.

CSC:n palveluita hyödyntävät tutkijat käyttävät supertietokoneita muun muassa kehittääkseen ja vertaillakseen koneoppimismalleja. Tämä prosessi sisältää mallien koulutusta, testien ajamista ja tulosten analysointia, mikä vaatii huomattavasti laskentatehoa ja on aikaa vievää. Usein nämä vaiheet tehdään manuaalisesti tutkijan itse kirjoittaman koodin avulla, mikä voi olla työlästä ja altista inhimillisille virheille. Tämä haaste koskee monia koneoppimisen parissa työskenteleviä, sillä mallien tehokas hallinta, seuranta ja tulosten vertailu vaativat monia vaiheita, jotka olisivat automatisoitavissa.

Suurin haaste on luoda toistettavia, optimoituja prosesseja mallien kehitykseen ja analysointiin, ja samalla varmistaa, että prosessit pysyvät hallittavina ja dokumentoituina. Tätä varten monilla koneoppimiseen keskittyvillä organisaatioilla ja tutkijoilla on tarve työkaluille, jotka tukevat kokeilujen kirjaamista, mallien versionhallintaa ja suorituksen seurantaa koko kehitysprosessin ajan. Tällaiset työkalut auttavat automatisoimaan monimutkaisia työnkulkuja ja lisäävät samalla prosessin tehokkuutta ja toistettavuutta.

Eräs tällaisista työkaluista on MLflow, avoimen lähdekoodin työkalu, joka on suunniteltu erityisesti koneoppimismallien hallintaan ja seurantaan. Työkalun ohjeistuksen laajentaminen ja käytön helpottaminen asiakasyrityksessä on tärkeä askel, jotta asiakkaat voivat hyödyntää sitä tulevaisuudessa tehokkaasti.

2 Koneoppiminen

Koneoppiminen ei ole uusi käsite, vaan sen kehitys alkoi jo 1950-luvulla. Silloin Alan Turing kehitti tunnetun “Turingin testin”, joka mittaa koneen älykkyyttä testaamalla, tunnistaako sen kanssa keskusteleva ihminen sen olevan kone. Muutaman vuoden myöhemmin psykologi Frank Rosenblatt ja hänen tiiminsä kehittivät ensimmäisen prototyypin neuroverkosta, joka sai innoituksensa ihmisen hermoston toiminnasta [3].

Koneoppimisella tarkoitetaan prosessia, jossa kone oppii itsenäisesti sille annetun datan perusteella. Oppiminen voi olla joko ohjattua tai ohjaamatonta. Ohjatussa oppimisessa koneelle määritellään tarkasti asia, joka pitää oppia koulutusdatan avulla [3]. Esimerkiksi kone opetetaan koirakuvien avulla tunnistamaan, miltä koira näyttää, jotta se voi jatkossa tunnistaa koiria uusista kuvista. Ohjaamattomassa oppimisessa kone etsii yhtäläisyyksiä, rakenteita tai esiintymiä ilman tarkkaa ennalta määriteltyä tavoitetta, kuten asiakassegmenttejä markkinointia varten.

Koneoppimisella pyritään saavuttamaan tilanne, jossa kone voi tehdä päätöksiä ja toimia ilman ihmisen puuttumista. Parhaimmillaan kone oppii ongelmista ja sopeutuu uusiin tilanteisiin oppimansa perusteella. Menetelmää voidaan hyödyntää moninaisissa ongelmissa, kuten suuren datamäärän analysoinnissa, luokittelussa ja ennustamisessa historiallisen datan perusteella.

Koneoppimismalli on prosessin tulos, jossa sopivaa koneoppimisalgoritmia ja dataa käyttäen koulutetaan kone suorittamaan tiettyä tehtävää [4]. Koneoppimisalgoritmit ovat monimutkaisia matemaattisia ohjeistuksia, joilla määritellään koneoppimisprosessin suoritus. Koulutetut mallit kykenevät tekemään itsenäisesti ennusteita tai päätelmiä uudesta, ennalta näkemättömästä datasta. Ne voidaan automatisoida suorittamaan tehtäviä, jotka olisivat hankalia tai jopa mahdottomia ohjelmoida perinteisin tavoin. Esimerkiksi kuvantunnistamiseen tarkoitettu malli voi oppia tunnistamaan kuvasta kukan ja sen lajikkeen, vaikka ohjelmoija ei antaisi tarkkoja ohjeita piirteistä, joita etsiä.

Koneoppimismallit ovat keskeisessä asemassa nykyaikaisessa tietojenkäsittelyssä ja muodostavat olennaisen osan tekoälyjärjestelmiä. Tekoälyä hyödynnetään laajasti eri sovelluksissa, kuten:

- Lääketieteessä, missä koneoppiminen toimii päätöksenteon tukena esimerkiksi syöpädiagnoosien tekemisessä [5].

- Autonomisissa autoissa, mahdollistaen päätöksenteon ilman ihmisen suoraa ohjausta ja edistäen siten itsenäistä ajamista [6].

- Kuvan-, tekstin- ja puheentunnistuksessa ja -käsittelyssä, kuten käännöspalveluissa ja generatiivisessa tekoälyssä, kuten kuvanluontiohjelmissa.

- Sähköpostin roskapostisuodattimissa, missä koneoppiminen auttaa tunnistamaan ja erottamaan ei-toivotun viestiliikenteen.

2.1 Koneoppimismallien hallinta

Koneoppimismallien käytön yleistyessä niiden kehitystä ja käyttöönottoa pyritään nopeuttamaan ja tehostamaan. Ohjelmistokehityksestä voidaan ottaa mallia: nykyaikaisilla käytänteillä päivitykset ja ominaisuudet saadaan tuotantoon lähes reaaliaikaisesti. Koneoppimisprosessissa on kuitenkin vaiheita, joita perinteisessä ohjelmistokehityksessä ei ole ja jotka tuovat omat erityiset haasteensa. Esimerkiksi koulutuksessa käytettävän datan esikäsittely, mallin kouluttaminen, hyperparametrien optimointi ja mallin jatkuva seuranta vaativat paljon aikaa ja laskentatehoa. Nämä vaiheet vaativat huolellista suunnittelua ja hallintaa, jotta mallin suorituskyky ja oikeellisuus säilyy tuotannon versiota päivitettäessä.

Kuten aiemmin kerrottiin, koneoppimismallit koulutetaan tiettyä tehtävää varten antamalla niille asiaankuuluvaa koulutusdataa. Data jaetaan koulutus- ja testaussetteihin, joiden avulla mallia säädetään koulutuksen aikana vertaamalla sen tuloksia ”oikeisiin” arvoihin. Lopuksi mallia vielä testataan, ja jos tulokset ovat tyydyttäviä, uusi malli voidaan viedä tuotantoon [7].

Tuotannossa oleva koneoppimismalli käsittelee jatkuvasti uutta ja ennennäkemätöntä dataa. Malli toimii luotettavasti niin kauan kuin tuotantodatan ominaisuudet pysyvät lähellä koulutusdatan ominaisuuksia. Todellisuudessa data kuitenkin muuttuu ajan myötä, ja tällaiset muutokset voivat vaikuttaa merkittävästi mallin suorituskykyyn.

Otetaan esimerkiksi talojen hintakehityksen ennustaminen: malli koulutetaan aluksi historiallisella datalla, joka sisältää hintoihin vaikuttavia tekijöitä, kuten sijainnin, talon koon ja alueen hintatason. Niin kauan kuin nämä tekijät säilyvät suhteellisen ennallaan, malli pystyy tuottamaan tarkkoja ennusteita.

Olosuhteet kuitenkin usein muuttuvat, esimerkiksi asuntomarkkinoilla tapahtuu hintavaihteluita, asuinalue muuttuu merkittävästi tai taloustilanne muuttuu nopeasti, ja mallin kyky tehdä luotettavia ennusteita heikkenee. Tämä johtuu siitä, ettei malli sopeudu uusiin muutoksiin datassa eikä voi hyödyntää uutta tietoa ilman uudelleenkoulutusta. Tätä tilannetta kutsutaan termillä ”data drift” [8].

Tällaisen tilanteen havaitsemiseksi ajoissa mallin suoriutumista tulisi seurata aktiivisesti, sillä useissa käyttötapauksissa pienikin muutos mallin ennusteiden tarkkuudessa voi olla merkittävä. Tällöin malli tulee kouluttaa uudelleen ajanmukaisella datalla, jotta sen ennusteet säilyvät luotettavina, ja uudelleenkoulutuksen tulisi tapahtua nopeasti ja sujuvasti, jotta uusi toimiva malli saataisiin käyttöön mahdollisimman pian.

Tarvitaan siis keinoja hallita ja seurata koneoppimismallia ja sen toimintaa tuotantoympäristössä läpi mallin elinkaaren. Seurannan lisäksi uusia malliversioita koulutettaessa on tärkeää, että koulutusprosessi, data ja muut taustatekijät dokumentoidaan, jotta prosessi on helposti jäljitettävissä ja toistettavissa myös tiimien välillä erilaisissa kehitysympäristöissä. Siksi mallien ja datan versiointi tulisi tehdä systemaattisesti, jotta tarvittaessa voidaan helposti palata aiempiin vaiheisiin ja ymmärtää tapahtumien kulku.

Koneoppimismallien kehittäminen ja käyttö vaatii monialaista osaamista, johon kuuluu usein koneoppimisspesialistien ja -insinöörien lisäksi tiimi ammattilaisia myös ohjelmoinnin ja data-analytiikan puolelta. Tämä monimuotoisuus tuo mukanaan haasteita mallin kehitys- ja hallintaprosessin koordinoinnille, sillä tiimin jäsenten käyttämät työkalut ja menetelmät voivat vaihdella huomattavasti.

Kehitysprosessin sujuvuuden takaamiseksi onkin olennaista, että käytetyt kehitysympäristöt ja työkalut ovat saumattomasti integroituja ja yhteensopivia keskenään. Tämä mahdollistaa tiedon jakamisen ja tiimityöskentelyn saman projektin parissa, vaikka eri tiimin jäsenet käyttäisivätkin erilaisia työkaluja ja ohjelmointikieliä. Näin edesautetaan tehokasta kommunikointia ja yhteistyötä, mikä puolestaan varmistaa sujuvan kehitysprosessin ja korkealaatuisen lopputuloksen.

2.2 Machine Learning Operations

Koneoppimismallien kehityksessä ja hallinnassa kohdataan siis monia haasteita. Näihin kuuluvat muun muassa aiemmin kuvatut datan saatavuuden ja laadun varmistaminen, mallin suorituskyvyn seuranta ja ylläpito, kehitys- ja tuotantoympäristöjen väliset eroavaisuudet sekä turvallisuuskysymykset.

Näihin haasteisiin vastaa MLOps (Machine Learning Operations), muunnelma DevOpsista (Development Operations), joka on tuttu termi sovelluskehityksestä. DevOpsin tavoitteena on sujuvoittaa ja automatisoida prosesseja ohjelmistojen kehityksen ja tuotannon välillä, jolloin kehityssyklit lyhenevät ja tuotteiden laatu paranee. Tärkeä osa DevOpsia on CI/CD- käytöntö (Continuous Integration / Continuous Deployment), joka varmistaa, että ohjelma pysyy jatkuvasti ajantasaisena yhdistämällä, testaamalla ja julkaisemalla uutta koodia sitä mukaa kun sitä tuotetaan [9].

MLOps laajentaa tätä toimintamallia koneoppimismallien tarpeiden mukaan lisäämällä prosessiin mallin uudelleenkoulutuksen ja siihen tarvittavan datan käsittelyn, tavoitteena varmistaa mallien jatkuva toimivuus ja luotettavuus [9].

MLOps-pipeline, eli koneoppimismallin elinkaari (Kuva 1), sisältää karkeasti seuraavat kohdat [10]:

1. Datan keräys (Data Collection)

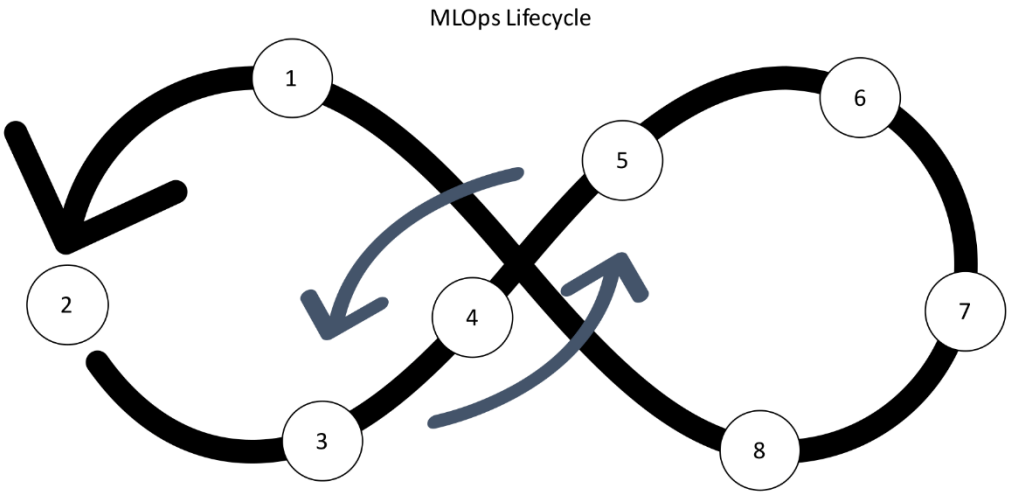
   Mallin tehokkuus riippuu siitä, että käytössä on tarpeeksi laadukasta ja monipuolista dataa. Tässä vaiheessa varmistetaan, että data kerätään oikeista lähteistä ja että se on relevanttia mallin tarkoituksen kannalta.

2. Datan käsittely (Data Preprocessing)

   Aiemmassa vaiheessa kerätyn raakadatan muokkaaminen käyttökelpoiseen muotoon on keskeinen askel. Datan esikäsittely voi sisältää esimerkiksi puuttuvien arvojen korjaamista tai standardointia.

3. Mallin kehitys (Model Development)

Tässä vaiheessa luodaan ja optimoidaan mallin arkkitehtuuri valiten algoritmit, jotka soveltuvat parhaiten kyseiseen ongelmaan.



Kuva 1. Koneoppimismallin elinkaari. Kuva: Erika Sternad 2024

4. Mallin (uudelleen)koulutus ja säätäminen (Training and Validation)

   Mallia koulutetaan aiemmin kerätyllä ja käsitellyllä datalla, joka jaetaan erillisiksi kokoelmiksi koulutusta ja testausta varten. Mallia säädetään parhaan mahdollisen ennustetarkkuuden saavuttamiseksi. MLOpsin avulla prosessia voidaan optimoida automatisoimalla koulutuskierroksia, ja se helpottaa etenkin tilanteessa, jossa kokeillaan useita eri malleja erilaisilla asetuksilla. Usein tällaisia koulutuskierroksia voi olla satoja.

5. Testaus (Model Evaluation)

   Ennen mallin tuotantoon viemistä sen suorituskykyä arvioidaan vielä erillisen datasetin avulla. Tämä varmistaa, että malli toimii odotetusti ja tuottaa luotettavia tuloksia.

6. Tuotantoon vieminen (Deployment)

   Kun malli on testattu ja todettu toimivaksi, se siirretään tuotantoympäristöön. MLOpsin avulla tämä vaihe voidaan automatisoida, jolloin malli voidaan ottaa käyttöön nopeasti ilman manuaalista työtä.

7. Mallin käyttö (Inference)

Tuotannossa mallille syötetään ennennäkemätöntä dataa ja se tekee ennusteita tai luokitteluja sen perusteella. Tässä vaiheessa mallin todellinen arvo näkyy sen kyvyssä tuottaa tarkkoja, luotettavia tuloksia tosielämän datalla.

8. Suorituskyvyn seuranta (Monitoring)

   Jatkuva seuranta on välttämätöntä, jotta voidaan havaita mahdollinen muutos datassa tai mallin suorituskyvyssä. Jos malli alkaa menettää tarkkuuttaan uudenlaisen ja poikkeavan datan vuoksi, MLOpsin avulla voidaan automatisoida toimenpiteitä johtaen mallin uudelleenkoulutukseen.

Ihannetilanteessa MLOps-pipeline toimii automaattisesti ohjelmoitujen sääntöjen mukaan, esimerkiksi tarkkaillen tuotannossa toimivan mallin antamien ennusteiden tarkkuutta ja tarvittaessa käynnistää mallin uudelleen koulutuksen. Uudelleen koulutuksella tarkoitetaan, että mallille tarjotaan uudenlaista dataa, jolla se päivittää mallin sisäisiä tekijöitä, jotka vaikuttavat sen antamaan lopputulokseen. Keskeistä on täysin uuden datan käyttö ja erityisesti sen laatu, sillä laadukas ajanmukainen data on edellytys mallin onnistuneelle päivitykselle. Esimerkiksi, jos koneoppimismalli koulutettaisiin virheellisellä tai epäluotettavalla datalla, se voisi heikentää mallin suorituskykyä sen sijaan, että parantaisi sitä. Uusi koulutettu ja testattu malli siirretään tuotantoon vanhan mallin tilalle, ja pipelinen suoritus jatkuu edelleen kohti uutta koulutuskierrosta.

Todellisuudessa toimivan MLOps-pipelinen toteuttaminen on monimutkainen ja vaativa prosessi. Koneoppimismallien kouluttaminen vie aikaa ja vaatii huomattavaa laskentatehoa, erityisesti kun työskennellään suurilla datamäärillä ja monimutkaisilla malleilla. Pipelineen liittyvien vaiheiden automatisointi ja optimointi edellyttävät laajaa asiantuntemusta eri osa-alueilta, joka harvoin on yhden henkilön hallittavissa.

MLOps on verrattain uusi toimintamalli, ja vaikka kehityksen tueksi on olemassa useita työkaluja, täydellistä ”yhtä ratkaisua” ei ole. MLOpsilla tarkoitetaankin enemmän joukkoa hyviä käytänteitä, kuin tarkkarajaista tapaa toimia ja rakentaa. Koska jokaisen koneoppimisprojektin ympäristö, tavoitteet ja resurssit ovat erilaisia, MLOps-prosessi on aina suunniteltava ja toteutettava juuri näiden muuttujien mukaisesti.

3 MLflow

Täysin automatisoitua pipelinea voidaan harvoin ylläpitää ilman ihmisen työpanosta, erityisesti uutta mallia kehitettäessä. Eri mallivaihtoehtoja ja parametreja voi olla tarpeen testata ja vertailla satoja, mikä vaatii ajan lisäksi asiantuntemusta, jota on vaikea ohjelmoida. Tämä vertailuprosessi myös tuottaa paljon dataa, jonka käsittely ja erottelu manuaalisesti on työlästä ja aikaa vievää. Vaikka pipelinen vaiheiden automatisointi ja tehtävien ketjuttaminen on sinänsä mahdollista, mallien tehokas hallinta, monitorointi ja näiden integrointi elinkaareen ovat haasteellisia. Kehittäjä voi kirjoittaa koodia, joka tallentaa tietoja mallin toiminnasta ja hyödyntää niitä vertailussa, mutta tämä lähestymistapa monimutkaistuu nopeasti ja muuttuu vaikeasti hallittavaksi.

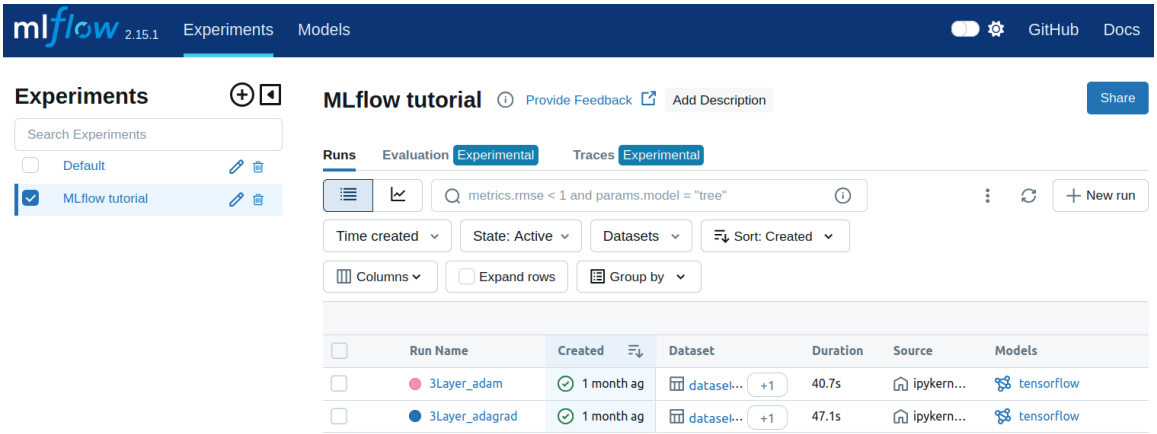
Ratkaisuna edellä mainittuihin haasteisiin on kehitetty useita työkaluja, jotka mahdollistavat mallien hallinnan ja toimenpiteiden tekemisen ilman kehittäjän jatkuvaa väliintuloa. Yksi näistä työkaluista on MLflow, joka on tämän opinnäytetyön keskiössä. Se on avoimen lähdekoodin työkalu, jota voidaan hyödyntää koneoppimisprosessin tukena MLOps-pipelinen eri vaiheissa [11].

MLflow'n kehitys sai alkunsa vuonna 2018 Databricksin asiakkaiden kanssa käydyistä keskusteluista, joiden perusteella voitiin tunnistaa koneoppimismallien elinkaaren pullonkaulat. Päätelmien perusteella he loivat ratkaisun, jossa kootaan datatieteilijöiden ja -insinöörien työ järjestelmällisesti ja kaikkien saataville. Näin kehitystyöstä tulee avoimempaa ja helpommin jaettavaa. Heidän luomallaan työkalulla voidaan vastata kolmeen keskeiseen haasteeseen mallien ja niiden elinkaaren kehityksessä: kokeilujen hallinta, toistettavuus sekä tuotantoon siirto [12].

3.1 Komponentit

MLflow tarjoaa merkittäviä etuja mallin kehityksessä, tuotantoon viemisessä ja monitoroinnissa. Se koostuu neljästä eri komponentista:

1. Tracking Server [13] mahdollistaa koneoppimismallin metriikoiden seurannan ja niiden tallennuksen tulosten vertailua varten. Se tallentaa koulutuksessa käytetyt parametrit ja koulutuksen metriikat, kuten mallin tarkkuuden ja muiden tulosten kehityksen koulutuksen aikana, yhteneväisesti samaan paikkaan. Komponentti sisältää myös kätevän käyttöliittymän (Kuva 2), josta tallennettuja tietoja voi tarkastella keskitetysti ja jopa reaaliaikaisesti. Sen avulla voidaan tallentaa myös keskeisiä tietoja koulutukseen käytetystä datasta, kuten sen sijainti ja ominaisuudet.

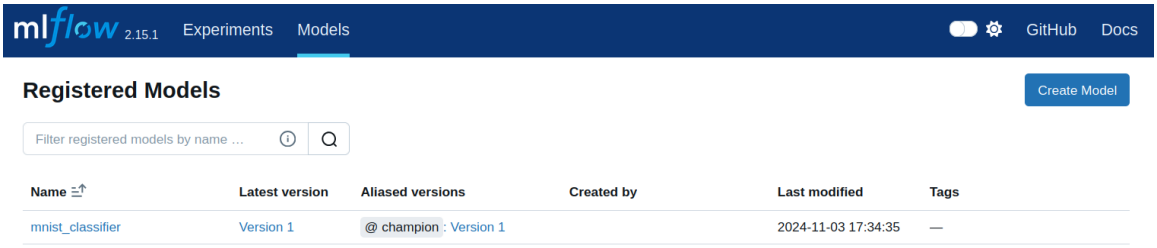


Kuva 2. MLflow käyttöliittymän aloitussivu

2. Models [14] tarjoaa työkaluja mallin paketointiin, julkaisuun ja jakeluun. Se mahdollistaa mallin pakkaamisen eri muotoihin, mikä helpottaa mallin jakamista ja käyttöönottoa eri ympäristöissä ja sovelluksissa. Tämä sujuvoittaa mallin kehitystä, testausta ja tuotantoon viemistä sekä tekee siitä skaalautuvaa. Komponentin avulla malleja voidaan tarjota myös käytettäväksi paikallisella palvelimella, mikä mahdollistaa mallin käytön nopeasti ja turvallisesti käyttäjän omalla alustalla.

3. Model Registry [15] on mallien versionhallintaan tarkoitettu komponentti, joka nimensä mukaisesti tarjoaa keskitetyn tavan rekisteröidä ja hallita koneoppimismallien versioita. Se helpottaa mallien elinkaaren seurantaa sekä malliversioiden jakamista tiimin ja organisaation sisällä. Tracking Serverin käyttöliittymässä on oma välilehti Model

Registrylle (Kuva 3), josta voi kätevästi nähdä kaikki versioidut mallit ja vertailla niitä. Niille voi asettaa aliaksia ja esimerkkisyötteitä, joilla voidaan helpottaa mallien käyttöönottoa ja malliversioiden siirtymiä järjestelmässä.



Kuva 3. MLflow'n mallirekisterin näkymä käyttöliittymässä

4. Projects [16] sisältää koneoppimisprojektin hallintaan ja toistettavuuteen liittyviä työkaluja. Sen avulla voidaan järjestää mallin hallintaan liittyviä osia, kuten ympäristön konfiguroinnin ja riippuvuuksien hallinnan, yhdeksi kokonaisuudeksi, joka muistuttaa tiedostokansiota tai ohjelmistoversionhallinnasta tuttua Git-repositoriota. Se on hyödyllinen työkalu, kun halutaan varmistaa, että koulutusprosessi on helposti jaettavissa ja toistettavissa esimerkiksi kehitystiimin kesken.

MLflow voi olla olennainen osa mallin elinkaaren hallintaa, erityisesti mallin kehitys- ja koulutusvaiheessa, ja sitä voidaan hyödyntää myös mallin käyttöönotossa ja tuotantovaiheen monitoroinnissa. On kuitenkin tärkeää huomata, että se ei tarjoa suoritusympäristöä tehtävien suorittamiseen.

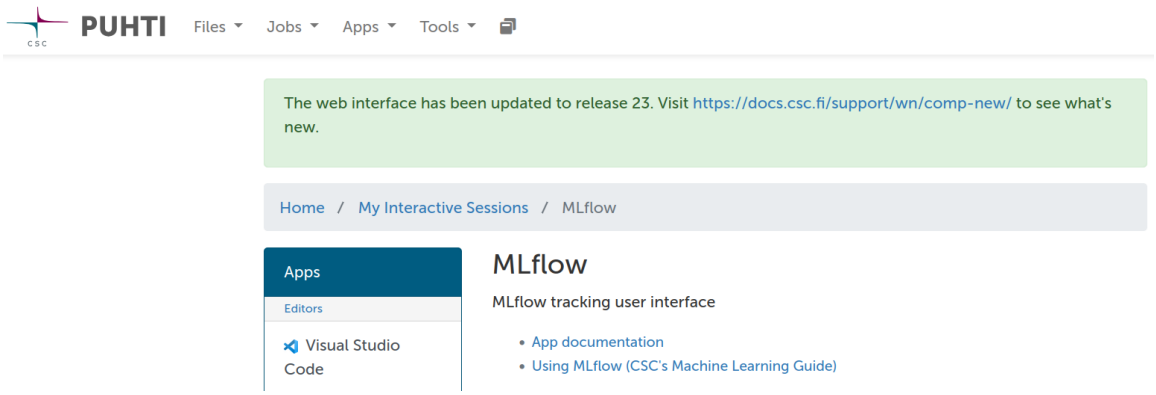
3.2 MLflow'n käyttö tilaajayrityksessä

Tässä työssä tarkastellaan MLflow'n laajamittaista ja tehokasta käyttöä tilaajayrityksen supertietokoneella. Vaikka työkalu on jo käytössä yrityksessä ja tarjolla myös asiakkaille, sen ohjeistukset ovat vähäiset ja keskittyvät pääasiassa käyttöönottoon [17]. Tavoitteena on mahdollistaa MLflow'n laaja-alainen hyödyntäminen, mikä voi johtaa säästöihin sekä asiakkaiden että yrityksen resurssien osalta, kun kokeilujen hallintaa tehostetaan merkittävästi.

Työssä hyödynnetään Puhti-supertietokonetta, joka tarjoaa monipuolisen ja tehokkaan laskentaympäristön. Lisäksi käytettävissä on CSC:n Allas-varastopalvelin datan hallintaan ja

varastointiin. Allas on saavutettavissa mistä tahansa verkosta, ja myös Puhdista voidaan ladata ja tallentaa dataa suoraan Altaaseen [18].

MLflow on tällä hetkellä saatavilla Puhdin sovellusvalikoimassa, ja se on myös integroitu ylläpidettyihin moduuleihin, jotka sisältävät erilaisia koneoppimiseen soveltuvia työkaluja ja kirjastoja. Käyttöönoton tueksi on tarjolla tekstimuotoinen ohjeistus [17], joka auttaa pääsemään alkuun, mutta ei kata työkalun kaikkia komponentteja ja ominaisuuksia syvällisesti. Ohjeistukseen on linkki sivulla, josta MLflow'n käyttöliittymän voi käynnistää (Kuva 4).



Kuva 4. MLflow'n käynnistyssivu Puhdin käyttöliittymässä

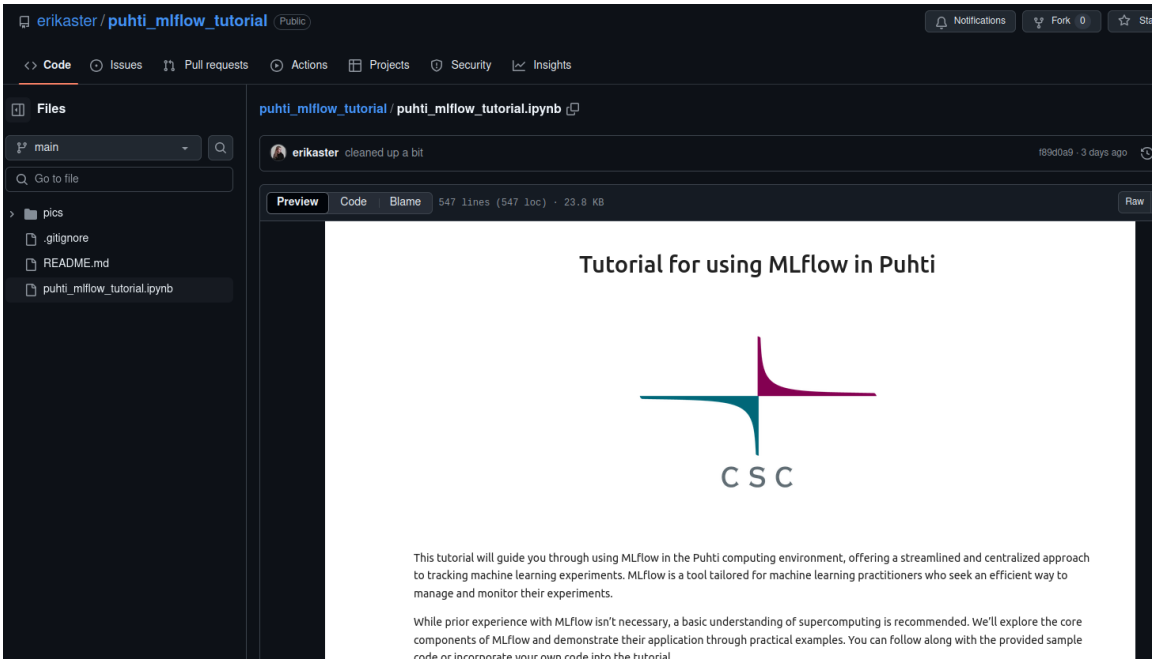
Ohjeistukset ja tämän opinnäytetyön myötä kehitetty tutoriaali ovat tarkoitettu kaikille, jotka haluavat tehostaa koneoppimismalliensa kehitystyötä. Käyttäjiksi soveltuvat esimerkiksi tutkijat ja opiskelijat. Hyödyntämällä MLflow'ta koneoppimisprosessien seurantaan, he välttyvät mallien manuaaliselta seurannalta, kuten tulosten kirjaamiselta ja visualisoimiselta. MLflow kirjaa ja tallentaa nämä tiedot automaattisesti, ja käyttäjät voivat vertailla malleja helposti valmiiksi luotujen kaavioiden avulla. Käyttöliittymä, joka on nopeasti käynnistettävissä Puhdin sovellusvalikoimasta, tarjoaa suoran pääsyn tähän toiminnallisuuteen. Lisäksi käyttäjät voivat hakea yksityiskohtaisempaa analyysitietoa MLflow'n avulla.

Tällä hetkellä MLflow ei ole laajasti käytössä Puhdin käyttäjien keskuudessa, vaikka kiinnostusta sen hyödyntämiseen on. Eräässä esimerkkitapauksessa tutkija otti yhteyttä CSC:hen tavoitteenaan tehostaa ja automatisoida koneoppimisprosessia, jossa ennustettiin viljasatoja satelliittikuvien avulla. Osana kokonaisratkaisua MLflow integroitiin tutkijan koodiin, joka kattaa prosessin mallin koulutuksesta ennusteiden tekemiseen. Tutkijalla ei ollut aiempaa kokemusta MLflow'sta, ja vaikka hän oli nähnyt työkalun logon Puhdin sovellusvalikoimassa, hän ei ollut osannut aloittaa sen käyttöä itsenäisesti. Käytön myötä hän kuitenkin koki sen tarjoamat hyödyt erittäin merkittäviksi.

4 Tutoriaali MLflow’n käyttöön Puhdissa

Tutoriaalin kehittämisen lähtökohtana oli varmistaa, että käyttäjä voi aloittaa MLflow’n käytön ilman aikaisempaa kokemusta työkalusta tai syvällistä tietoa koneoppimismalleista. Tutoriaali pohjautuu valmiiseen esimerkkikoodiin, jota seuraamalla käyttäjä voi helposti edetä vaiheittain. Esimerkissä käytetään kahta neuroverkkopohjaista mallia: ReLU_2Layer_Adagrad ja ReLU_2Layer_Adam. Molemmat mallit ovat rakenteeltaan samanlaisia, mutta ne eroavat käyttämissään optimointialgoritmeissa. Esimerkkikoodissa nämä kaksi mallia koulutetaan tunnistamaan käsin kirjoitettuja numeroita MNIST-datasetin avulla. Kokeneemmille käyttäjille tarjotaan myös mahdollisuus integroida omia koneoppimismalleja tutoriaaliin. Tavoitteena oli luoda selkeä, nopeasti omaksuttava ja yksinkertainen opas, joka kuitenkin on tarpeeksi kattava antamaan käyttäjälle varmuutta jatkaa MLflow’n käyttöä itsenäisesti sekä tutkia työkalun muita ominaisuuksia.

Tutoriaali on Jupyter Notebook -muodossa, mikä mahdollistaa interaktiivisen oppimiskokemuksen. Käyttäjä ei vain lue ohjeita, vaan pääsee myös kokeilemaan koodin toimivuutta ja näkemään tulokset heti. Tämä tekee oppimisesta konkreettisempaa ja auttaa ymmärtämään työkalun toimintaa käytännössä.

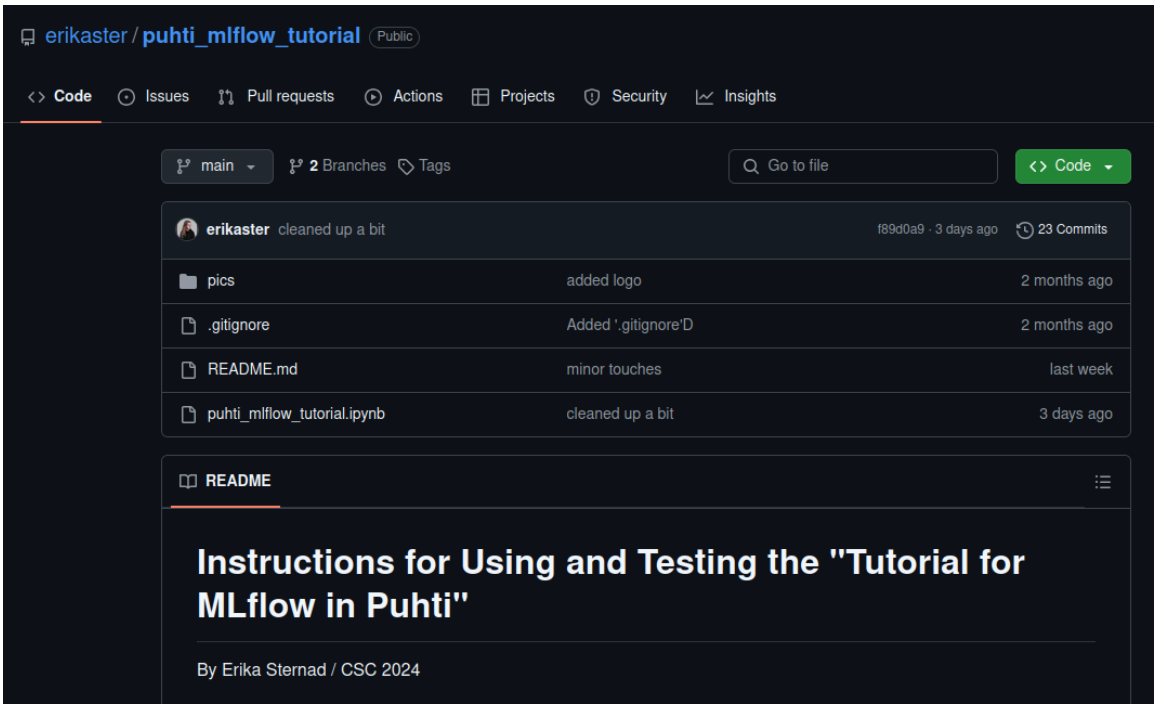


Kuva 5. Tutoriaali Git-repositoriossa

Tutoriaalissa edetään johdonmukaisesti komponentti kerrallaan ja esitellään kunkin osa-alueen keskeiset funktiot sekä niiden käyttö esimerkkien avulla. Jokaisen funktion kohdalla tarjotaan lisäksi linkki MLflow'n viralliseen dokumentaatioon, jotta käyttäjä voi halutessaan perehtyä toimintoihin tarkemmin. Näin tutoriaali pysyy selkeänä ja yksinkertaisena, mutta samalla tarjoaa mahdollisuuden lisäoppimiseen ja työkalun syvällisempään hyödyntämiseen.

Ennen tutoriaalin suorittamista käyttäjän on tehtävä muutamia esivalmisteluja, jotta se toimii sujuvasti ja ilman ongelmia. Yksi keskeisimmistä valmisteluista on varata riittävästi muistia Jupyter- sovellusta varten, sillä koneoppimismallit ja datankäsittely kuluttavat paljon laskentatehoa. Tämä takaa, että tutoriaalin suoritus sujuu ilman teknisiä haasteita ja turhaa odottelua, ja käyttäjä voi keskittyä oppimisprosessiin keskeytyksettä.

Ohjeet tarvittavista esivalmisteluista annetaan ennen tutoriaalin aloittamista. Tutoriaali on saatavilla julkisena Git-repositoriona [19] (Kuva 6), jonka käyttäjä voi helposti kloonata eli siirtää suoraan Puhdin ympäristöön käyttöönsä. Kaikki tarvittavat ohjeet ja tiedot löytyvät repositoriosta, mikä tekee käyttöönotosta vaivatonta.



Kuva 6. Tutoriaalin Git-repositorion aloitussivu

Seuraavaksi tarkastellaan MLflow'n toimintaa ja sen keskeisiä komponentteja tutoriaalissa. Jokaista komponenttia on käsitelty omana "lukunaan", mikä auttaa käyttäjää ymmärtämään, mitä

ominaisuuksia kukin osa sisältää ja miten ne tukevat koneoppimismallien hallintaa missäkin vaiheessa.

4.1 Tracking Server

MLflow Tracking Server on palvelin, joka välittää tietoa REST API -rajapintojen kautta. Se voidaan konfiguroida käyttämään erilaisia tallennustiloja, kuten paikallista levytilaa, tietokantaa tai ulkoista objektitallennusta, käyttäjän tai projektin tarpeiden mukaan. Serverin käynnistäminen ohjelmallisesti onnistuu oletusasetuksilla yksinkertaisesti yhdellä komennolla, mutta tarvittaessa käyttäjä voi tehdä lisäasetuksia, kuten määrittää käyttäjänimiä ja salasanoja turvallisuuden varmistamiseksi [13].

MLflow Tracking Serveriä voivat käyttää samanaikaisesti useat käyttäjät, mikä tekee siitä ihanteellisen työkalun tiimeille, jotka työskentelevät saman projektin parissa. Se mahdollistaa kaikkien käyttäjien kokeiden tallentamisen keskitetysti ja yhtenäisesti samaan paikkaan, mikä helpottaa tulosten hallintaa ja vertailua. Suuremmissa projekteissa se voidaan esimerkiksi ajaa omassa erillisessä ympäristössään, osana laajempaa mikropalveluarkkitehtuuria, mahdollistaen sen laajan käytön osana suurempaa projektikokonaisuutta.

Tutoriaalissa käyttäjä opastetaan vaiheittain käyttämään MLflow’n Tracking Serveriä seuraamaan mallien koulutusta. Aluksi käydään läpi Tracking Serverin perustaminen ja kokeen määrittäminen, mikä luo pohjan koneoppimisprosessin seurannalle (Kuva 7).

Esimerkkikoodissa käyttäjä kouluttaa molemmat mallit, ja MLflow tallentaa sekä koulutusprosessin parametrit että tulokset automaattisesti käyttäen mlflow.autolog()- funktiota. Tämän jälkeen tutoriaalissa ohjeistetaan kuvakaappausten avulla, miten käyttäjä voi käynnistää MLflow’n käyttöliittymän Puhdissa ja sen jälkeen tarkastella koulutusprosessin tuloksia sen kautta. Tämä tarjoaa selkeän ja konkreettisen tavan ymmärtää, kuinka Tracking Server-komponentti tehostaa mallien optimointia ja tulosten hallintaa.

In [ ]:

```
# Set tracking URI
project_id = "project_2001234" # Insert your CSC project id here
mlruns_uri = f"/scratch/{project_id}/path/to/mlruns" # NOTE! Check and change the path!
mlflow.set_tracking_uri(mlruns_uri) # https://mlflow.org/docs/latest/python_api/mlflow.html#mlflow-tracing-apis

# Set experiment
experiment_name = "MLflow tutorial"
experiment = mlflow.set_experiment(experiment_name) # https://mlflow.org/docs/latest/python_api/mlflow.html#mlfl
print(f"Artifacts and metadata are stored here: {experiment.artifact_location}")
```

Experimenting with MLflow

Next, we will run training sessions using MLflow. We'll utilize the autolog function, which automatically logs all the data generated during the run. By default, the model will be logged as an artifact by the **Models** component, making it easy to access later on and enabling automatic versioning. This and other features can be modified in arguments.

In the example code, we perform two training rounds with slightly different models, allowing us to compare the results in the UI.

In [ ]:

```
mlflow.autolog() # https://mlflow.org/docs/latest/tracking/autolog.html#automatic-logging
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# Let's compile two slightly different models to compare. You can either use the example code or insert your own.

(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train / 255.
X_test = X_test / 255.
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

model_1 = Sequential(
    [
        layers.Flatten(input_shape=(28, 28)),
        layers.Dense(128, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(10, activation='softmax')
    ]
)
model_1.compile(optimizer='adagrad',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

model_2 = Sequential(
    [
```

Kuva 7. Tracking Serverin alustus ja esimerkkikoodia tutoriaalissa

4.2 Models

MLflow Models on komponentti, josta on erityisesti apua mallien pakkaamisessa, siirtämisessä ja käyttöönotossa. Tutoriaalissa käyttäjä johdatetaan käytännönläheisesti käyttämään Models-komponenttia mallien tallentamiseen ja uudelleenkäyttöön. Koulutettu malli pakataan ja tallennetaan automaattisesti MLflow'n avulla, mikä tarjoaa selkeän esimerkin siitä, kuinka tämä prosessi toimii. Käyttäjälle näytetään, miten ladattuja tallennettu malli voidaan myöhemmin ottaa käyttöön ja hyödyntää ennustamistehtävissä, kuten tutoriaalissa esiteltävässä demossa.

Models on monipuolinen työkalu, jonka avulla voi paitsi tallentaa ja jakaa koneoppimismalleja, myös asettaa mallin käytettäväksi paikallisesti. Tällöin malli voidaan tuoda saataville REST API -rajapintojen kautta, jolloin ulkoiset järjestelmät tai sovellukset voivat käyttää sitä HTTP-kutsujen avulla. Tämä helpottaa mallin integrointia muihin järjestelmiin ja palveluihin, mahdollistaen

esimerkiksi sen käytön tuotantoympäristössä tai reaaliaikaisten ennusteiden tekemisen ulkoisten sovellusten kautta.

4.3 Model Registry

Tutoriaalissa näytetään, kuinka MLflow'n Model Registry auttaa hallitsemaan koneoppimismallien eri versioita. Esimerkkikoodin ja kuvallisten ohjeiden avulla käyttäjä oppii rekisteröimään mallin sekä manuaalisesti käyttöliittymän kautta että ohjelmallisesti (Kuva 8). Mallien versiointia selkeytetään aliasten avulla, joiden avulla voidaan merkitä esimerkiksi, mikä malli on käytössä tuotannossa ja mikä on seuraava kandidaatti. Niitä voidaan käyttää myös tunnisteina, kun mallia haetaan tuotantoon, mikä helpottaa mallien hallintaa ja mahdollistaa tuotannon automatisoinnin.

Model Registry toimii käytännössä välikätenä Tracking Serverin ja mallia käyttävien sovellusten välillä. Rekisteröidyn mallin tiedoista löytyy polku siihen koulutusajoon, jossa malli on tallennettu, ja tämän tiedon avulla malli voidaan ladata ja ottaa käyttöön tuotannossa.

Registering a model via API

To register another trained model using the API, you need the run ID of the model you want to register.

Since a previously registered model via the UI had a lower accuracy, we will now programmatically find and register the model with the highest `validation_accuracy`. Additionally, we will retrieve the model's name using API calls.

Finally, we assign the alias "challenger" to the model for easier identification. To accomplish this, we use the MLflow Client, which allows us to programmatically manage model aliases and streamline the process.

In [ ]:

```
from mlflow import MlflowClient

client = MlflowClient()

try:
    # https://mlflow.org/docs/latest/search-runs.html
    runs = mlflow.search_runs([experiment.experiment_id])
    run_id = runs.loc[runs['metrics.validation_accuracy'].idxmax(), 'run_id']

    # https://mlflow.org/docs/latest/python_api/mlflow.html#mlflow.register_model
    filter_string = "name LIKE 'mnist%'"
    model_uri = f"runs:/{run_id}"

    registered_models = mlflow.search_registered_models(filter_string=filter_string)
    if len(registered_models) == 0:
        raise IndexError("No registered model found")

    model_name = (mlflow.search_registered_models(filter_string=filter_string))[0].name
    mv = mlflow.register_model(model_uri, model_name)

    # https://mlflow.org/docs/latest/python_api/mlflow.client.html#mlflow.client.MlflowClient.set_model_version_
    client.set_registered_model_alias(mv.name, "challenger", mv.version)

except IndexError:
    print("No registered model found. Please go back and register the model in the MLflow UI first.")
except Exception as e:
    print(f"An exception occurred: {e}")
```

Kuva 8. Mallirekisterin käyttöä esimerkkikoodin avulla tutoriaalissa.

Mallirekisteristä on merkittävää hyötyä koneoppimismallien tuotantoon viemisessä ja niiden hallinnassa. Versioinnin avulla voidaan helposti palata aiempiin malleihin tarvittaessa, mikä takaa joustavuuden ja riskinhallinnan tuotantoympäristössä. Rekisteri mahdollistaa mallien hallinnan ja testaamisen automatisoinnin, mikä vähentää kehittäjien manuaalista työtä. Lisäksi rekisteri voidaan integroida osaksi CI/CD-putkea, mikä nopeuttaa ja tehostaa mallien siirtymistä kehitysvaiheesta tuotantoon saumattomasti ja hallitusti.

4.4 Projects

Projects-komponentti helpottaa kokonaisten koneoppimisprojektien pakkaamista ja siirtämistä, parantaen projektien toistettavuutta ja hallintaa eri tiimien ja suoritusympäristöjen välillä. Tämä on erityisen hyödyllistä, kun projekteja jatkokehitetään, kun kaikki projektin suorittamiseen tarvittavat osat ovat selkeästi määriteltyinä ja helposti jaettavissa.

Tutoriaalissa Projects-komponentti esitellään lyhyesti verrattuna muihin ominaisuuksiin, ja käyttäjää kannustetaan tutustumaan siihen tarkemmin itsenäisesti (Kuva 9). Käytännössä projekti pakataan Git-repositoriota muistuttavaan formaattiin, joka sisältää kaiken tarvittavan tiedon ja koodin projektin suorittamiseksi. Projektin ydin on MLproject-tiedosto, joka määrittää, miten koneoppimisprojekti suoritetaan. Tiedostossa voidaan määritellä esimerkiksi ohjelman suorittamiseen tarvittavat komennot ja niiden argumentit sekä mahdolliset ympäristövaatimukset, kuten riippuvuudet ja ympäristömuuttujat. Koko tiedosto ja sen sisältö suoritetaan ajamalla se yhdellä komennolla, mikä tekee projektin toistamisesta vaivatonta.

MLflow Projects (advanced)

MLflow Projects organizes code into **self-contained, reusable components** that can be run on different environments. A project in MLflow is essentially a **directory** or **Git repository** containing:

1. **Code (scripts)** for performing machine learning tasks.
2. **`MLproject` file**: A YAML configuration file that defines:
   - Project name
   - Dependencies (e.g., environment, for Puhti see https://docs.csc.fi/apps/python/#pre-installed-python-environments)
   - Entry points (main scripts/functions with parameters for tasks like training or evaluation)

The `MLproject` file might look like this:

```
name: my_ml_project
entry_points:
        main:
                parameters: learning_rate: {type: float, default: 0.01}
                epochs: {type: int, default: 10}
                command: |
                        module load ptensorflow
                        python train.py --learning_rate {learning_rate} --epochs {epochs}
```

It would be utilized in a job script in the following manner:

```
#!/bin/bash
#SBATCH --job-name=mlflow_project
#SBATCH --output=mlflow_output_%j.log
#SBATCH --error=mlflow_error_%j.log
#SBATCH --time=00:01:00
#SBATCH --partition=test
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=1G
#SBATCH --account=project_20001234

module load python-data

mlflow run . -P learning_rate=0.01 -P epochs=10
```

Kuva 9. Projects-moduuli esiteltynä tutoriaalissa

5 Tulokset

Asiakkaiden tueksi halutaan kehittää ratkaisuja, jotka mahdollistavat heidän keskittymisensä olennaiseen eli koneoppimismalliensa kehitykseen ja analysointiin. MLflow'n tarjoaminen on yksi keinoista, joka tukee asiakkaiden työtä sekä tutoriaalin että yrityksen tarjoaman asiakastuen muodossa. Projektin tavoitteena oli myös kehittää yrityksen omaa osaamista tällä alueella.

Tavoitteena oli luoda selkeä ja helposti lähestyttävä tapa tutustua MLflow'hun ja ottaa se käyttöön asiakasyrityksen alustalla. Ratkaisun tuli olla helposti ylläpidettävä sekä sellainen, että sitä voitaisiin soveltaa myös muilla supertietokoneilla ja alustoilla pienin muutoksin.

Kokonaisuudessaan tutoriaali on onnistunut ja tarjoaa selkeän lähestymistavan MLflow'n perusominaisuuksiin. Se helpottaa työkalun käyttöönottoa asiakkaiden projekteissa ja vastaa kasvavaan tarpeeseen hallita koneoppimismalleja tekoälyratkaisujen yleistyessä. Sen käyttöönoton laajentaminen voi säästää paitsi käyttäjien aikaa ja rahaa, myös muita resursseja, kuten laskentakapasiteettia ja levytilaa, joita voidaan kohdentaa muihin projekteihin.

Lisäksi yrityksen sisäisen osaamisen ja palvelutarjonnan vahvistaminen tällä alueella on tärkeää, sillä se mahdollistaa asiakkaille tehokkaan tuen tavoitteidensa saavuttamisessa ilman, että heidän tarvitsee perehtyä kaikkiin teknisiin yksityiskohtiin tai kehittää ratkaisuja täysin alusta alkaen.

5.1 Tutoriaalin testaussuunnitelma

Tutoriaalin toimivuutta ja hyödyllisyyttä arvioi kolme henkilöä, jotka kuuluvat samaan tiimiin tutoriaalin tekijän kanssa. Kaikilla oli taustaa koneoppimismallien käytöstä, mutta MLflow oli heille korkeintaan nimeltä tuttu työkalu. Heille annettiin ohjeet ja kysymykset tutoriaalin tarkasteluun eri näkökulmista. Testaaminen kohdistui seuraaviin osa-alueisiin:

1. Selkeys ja käytettävyys

   Testaajilta kysyttiin, olivatko Jupyterin ja MLflow'n asennusohjeet Puhdin ympäristössä selkeitä ja helppoja ymmärtää. Tavoitteena oli varmistaa, että tutoriaalin ohjeita on helppo seurata ja että käyttöönottoprosessi sujuu sujuvasti ilman merkittäviä esteitä.

2. Tekninen sisältö

   Teknisen sisällön osalta haluttiin selvittää, tarjosiko tutoriaali riittävästi taustatietoa MLflow'n keskeisistä käsitteistä, kuten mallien seurannasta, versioinnista ja rekisteröinnistä. Kysymysten avulla pyrittiin myös arvioimaan, selittikö tutoriaali aiempaa osaamista olettamattomia käsitteitä riittävästi ja oliko esimerkkikoodi dokumentoitu niin, että käyttäjä ymmärsi, mitä tapahtuu jokaisessa vaiheessa.

3. Virheiden käsittely

   Testaajien tuli raportoida mahdolliset virheet, joihin he törmäsivät koodin suorittamisen aikana. Tavoitteena oli arvioida, kuinka helposti virheet pystyttiin ratkaisemaan ja löytyikö koodista tai ohjeista riittävää tukea virhetilanteisiin.

4. Tulokset

   Tutoriaalin tulososiossa selvitettiin, olivatko MLflow'n keräämät parametrit ja metriikat selkeästi ymmärrettäviä. Testaajia pyydettiin arvioimaan, kuinka helppoa oli analysoida ja vertailla eri mallien suorituskykyä ja hyödyntää Tracking Serveristä saatavia tietoja.

5. Parannusehdotukset

   Testaajat saivat esittää omia parannusehdotuksia tutoriaaliin. Heiltä pyydettiin pohtimaan, oliko mukana turhia osioita tai jäikö jotain oleellista uupumaan, mikä voisi auttaa käyttäjää paremmin.

6. Ajankäyttö ja vaivannäkö

   Tutoriaalin kestoa mitattiin ja arvioitiin, oliko käytetty aika sopiva sisällön ja tutoriaalin monimutkaisuuden suhteen.

7. Yleinen palaute

   Lopuksi testaajilta pyydettiin yleistä palautetta tutoriaalin laadusta ja toimivuudesta. Heidän tuli arvioida tutoriaalin kokonaislaatu ja antaa suosituksia siitä, kannattaisiko tätä tutoriaalia tarjota muille käyttäjille ja miksi.

5.2 Palaute

Kaksi testaajaa sai jo ensimmäisen raakaversion testattavakseen. Heiltä tulleen vapaamuotoisen palautteen perusteella tutoriaalia hienosäädettiin, ja lopullinen versio testattiin suunnitelman mukaan. Testauksen tuloksena saatiin mm. seuraavanlaisia palautteita:

- "*Overall, the tutorial was good and a nice and quick introduction to the basics of MLflow. Would recommend to others.*" Tämän palautteen antaja käytti tutoriaalin läpikäymiseen noin kaksi tuntia testaten sekä valmiilla esimerkkikoodilla että integroimalla siihen oman mallinsa.

- *"Getting started ohjeet oli kattavat niin tutoriaalin saamiksesi Puhtiin kuin Jupyter ja MLFlow ympäristön pystystä koskien. Tutoriaalissa kaikki oli selitetty auki ja otsikoitu selkeästi vaiheet omiin osioihinsa, tutoriaali eteni loogisesti ja siinä oli helppo pysyä. Kuvissa merkitty erikseen kohdat, mistä tulee klikata."*

- *"For one with a solid background in ML and related topics, the tutorial included sufficient explanations and the code was well-documented. However for a complete novice, maybe the objective of classifying the MNIST-numbers could be explained by e.g. a figure to captivate the user right away? Of course in the end there is an interactive demo, but perhaps something explanatory could be also in the start.*

  The tutorial was high-quality and should be adopted to e.g. CSC courses utilising MLFlow! I would recommend this at least to fellow CSC workers, as I believe one of the main benefits to the standard guides is that the steps are Puhti-specific, and one doesn't have to figure out this stuff by themselves."

Tutoriaalin suorittamiseen meni testaajilta noin 30–120 minuuttia, riippuen siitä, kuinka tarkasti ja syvällisesti he siihen suhtautuivat. Parilta testaajalta tuli toive, että esimerkkinä käytettävä malli ja datasetti esiteltäisiin tarkemmin, erityisesti ajatellen käyttäjiä joilla ei ole vielä paljoa kokemusta koneoppimisesta. Lisäksi he antoivat arvokasta palautetta pienistä virheistä, jotka aiheuttivat ongelmia koodin toiminnassa. Näiden huomioiden pohjalta tutoriaalia pystyttiin parantamaan, minkä myötä se tuli selkeämmäksi, virheettömämmäksi ja luotettavammaksi. Erityisen hyvä kehitysidea oli lisätä pieni tiivistelmä, joka toimisi lopussa muistilappuna käyttäjälle, miksi MLflow'n käyttö koneoppimisprosessissa on hyödyllistä ja millaisia etuja se voi

tarjota. Tämä tiivistelmä voisi auttaa käyttäjiä ymmärtämään työkalun arvon ja motivoimaan heitä ottamaan sen osaksi omaa työskentelyään.

Tutoriaalin julkistamisen jälkeen sitä markkinoidaan kohderyhmälle eri kanavissa, mutta varsinaista palautetta ei enää kerätä aktiivisesti. Palautteenantoon kuitenkin kannustetaan ja sitä varten tarjotaan suora kanava. On toivottavaa, että tutoriaalin julkistaminen näkyy MLflow'n käyttäjätilastoissa, mutta sen vaikutukset selviävät vasta myöhemmin.

Varsinaisen testauksen lisäksi tutoriaali otettiin käyttöön myös Kajaanin ammattikorkeakoulussa, jossa tieto- ja viestintätekniikan insinööriopiskelijat saivat mahdollisuuden suorittaa sen vapaaehtoisena osana koneoppimismenetelmiä käsittelevää kurssia.

5.3 Vaihtoehtoja MLflow'lle

Product	Open source	Experiment tracking	Model versioning / registry	Data versioning	UI	Support (other than community)	Other
MLflow	✓	✓	✓	✓ Not that comprehensive	✓	✗	- Easy setup - Can be used widely in ML lifecycle but doesn't offer any workflow tools - Supports most languages
Guild AI	✓	✓	✗	✗	✓	✓ Email provided	- External tool - Offers some pipeline features - No dbs or containers required - Jupyter Notebook integration
DVC	✓	✓	✓	✓	✓ Git based web app + VS Code extension	✓ Discord	- Works on top of Git **- Would work well with alonside MLflow etc.** - Completely language agnostic
Metaflow	✓	✓	✗	✓	✗	✓ Email provided	- More for building and managing data science workflows - Automatically versions all data and code **- Tightly related to AWS, not entirely sure if can be used with other backend?** - Python library
Kubeflow	✓	✓	✓ Achieved through pipelines	✗	✓	✓ Slack	-To be used with Kubernetes - Heavy if the only need is to track training - Since Kubernetes is used either way, would be good to have as an option since can be used for building workflows too

Kuva 10. Vertailutaulukko vaihtoehdoista MLflow'lle. Kuva: Erika Sternad [2024]

MLflow'lle on olemassa useita vaihtoehtoja, joista muutamia on esitelty ja vertailtu edeltävässä taulukossa (Kuva 10). Kuten MLflow, kaikki vaihtoehdot ovat avoimen lähdekoodin työkaluja, mikä mahdollistaa kustannustehokkaan ja joustavan käytön. Avoin lähdekoodi tarjoaa mahdollisuuden muokata työkaluja omiin tarpeisiin, mahdollistaen riippumattomuuden kaupallisista toimittajista sekä paremman integraation eri ympäristöihin. Lisäksi avoin lähdekoodi

tukee kehittäjäyhteisön panosta ja innovaatioita, mikä voi nopeuttaa työkalujen kehitystä ja parantaa ominaisuuksia.

MLflow on monipuolinen työkalu koneoppimismallien elinkaaren hallintaan, mutta sen ominaisuudet datan versioinnissa ovat rajalliset. Tähän tarpeeseen vastaa DVC (Data Version Control), joka keskittyy erityisesti datan ja koodin versiointiin [20]. DVC mahdollistaa myös mallin koulutuksen ja tulosten versionhallinnan Git-pohjaisella lähestymistavalla, mutta sen mallinhallinnan ominaisuudet eivät ole yhtä kattavat kuin MLflow'n. Nämä kaksi työkalua täydentäisivät hyvin toisiaan yhdistämällä MLflow'n seurantaominaisuudet ja DVC:n versiointikyvyt.

Guild AI on kevyempi vaihtoehto MLflow'lle, ja se on suunniteltu erityisesti kokeiluihin, joissa ei vaadita kattavaa mallien versiointia tai rekisteröintiä. Guild AI tarjoaa kuitenkin perustoiminnot kokeilujen seurantaan ja tulosten vertailuun, joten se sopii yksinkertaisempiin projekteihin tai vaikkapa opetuskäyttöön [21]. Se on myös helposti integroitavissa Jupyter Notebook -ympäristöön, joka on monille tutkijoille ja opiskelijoille tuttu ja joustava työkalu kokeiluihin.

Metaflow on monipuolinen työkalu, joka keskittyy ensisijaisesti koneoppimisprojektien työnkulkujen hallintaan ja toistettavuuteen. Vaikka se tarjoaa joitain samoja ominaisuuksia kuin MLflow, kuten kokeilujen seurantaa, se on suunniteltu enemmän prosessien ja datavirtojen hallintaan kuin mallien versiointiin. Tämän vuoksi se soveltuu erityisesti ympäristöihin, joissa datan ja kokeilujen hallinta ja skaalautuvuus ovat keskiössä, mutta varsinainen mallien versiointi ja rekisteröinti eivät ole tärkeimpiä vaatimuksia [22].

Kubeflow on tehokas ja monipuolinen työkalu, joka tukee koneoppimismallien elinkaaren hallintaa ja mahdollistaa monimutkaisten työnkulkujen luomisen Kubernetes-ympäristössä. Sen skaalautuvuus ja joustavuus tekevät siitä sopivan erityisesti laajoihin projekteihin, joissa tarvitaan pysyviä ja luotettavia työnkulkuja. Kubeflow'n käyttöönotto on kuitenkin työläs prosessi, joka vaatii osaamista sekä Kubernetesin että koneoppimistyönkulkujen saralla. Tämä tekee siitä paremman vaihtoehdon suurille tiimeille ja yrityksille, jotka haluavat automatisoida ja hallita koko ML-prosessin kattavasti [23].

Vertailun perusteella voidaan todeta, että MLflow erottuu yhä omassa luokassaan helppokäyttöisyytensä ja kokonaisvaltaisten ominaisuuksiensa ansiosta. Se kattaa laajasti koneoppimismallien elinkaaren hallinnan keskeiset toiminnot ja tarjoaa selkeän käyttöliittymän, joka helpottaa mallien seuranta- ja hallintaprosessia. Vaikka markkinoilla on vaihtoehtoja, jotka

voivat täydentää tai tukea erikoistarpeita, kuten DVC datan versiointiin tai Kubeflow suurille Kubernetes-pohjaisille projekteille, MLflow pysyy ykkösvalintana tilaajayritykselle sen yksinkertaisuuden ja monipuolisuuden ansiosta.

5.4 Tutoriaalin jatkokehitys ja ylläpito

MLflow on aktiivisesti päivittyvä työkalu, joka kehityksen myötä saa jatkuvasti uusia ominaisuuksia ja parannuksia aiempiin toimintoihin. Tämän vuoksi on tärkeää, että tutoriaalin ylläpitäjä seuraa sen päivityksiä ja muutoksia, vaikka tutoriaalin esittelemät perusominaisuudet pysyisivätkin pitkälti samoina. Tutoriaalin ylläpitovastuu säilyy kirjoittajalla ja hänen tiimillään, eikä erillistä ylläpitosuunnitelmaa ole toistaiseksi tarpeen laatia, sillä MLflow'n kehityksen seuranta kuuluu myös muihin työtehtäviin. Tutoriaalin ajantasaisuudesta huolehtimisen lisäksi myös Puhti-supertietokoneen MLflow-sovelluksen versio on pidettävä ajan tasalla, kuten myös Python-moduulit joihin työkalu sisältyy. Myös näiden ylläpito kuuluu samaiselle tiimille.

MLflow on äskettäin julkaissut uusia ominaisuuksia, jotka tukevat erityisesti suurten kielimallien hallintaa ja käyttöä. Esimerkiksi MLflow AI Gateway tarjoaa organisaatioille keskitetyn keinon käyttää kaupallisten toimijoiden, kuten OpenAI:n, kielimalleja [24]. Lisäksi MLflow Tracing laajentaa MLflow'n seurantamahdollisuuksia kokeiluista ja koulutuksesta tuotantoon, jolloin koneoppimismallien suoritusta voidaan monitoroida reaaliajassa [25]. Nämä molemmat ominaisuudet ovat vielä kehityksen alla. Tutoriaalin tarjotessa perusteet MLflow'n käyttöön sen rinnalle voitaisiin tulevaisuudessa kehittää toinen versio, jossa esitellään uusia ominaisuuksia edistyneempään käyttöön esimerkiksi kielimallien hallinnan näkökulmasta.

Käytettävyyden varmistamiseksi tutoriaalin voisi mukauttaa kattamaan myös muita CSC:n alustoja, jottei se olisi sidottu vain Puhtiin. Tutoriaali voi olla käytettävissä suoraan supertietokone LUMIlla, mutta sitä ei ole vielä testattu. MLflow'n voi ottaa käyttöön myös konttipilvi Rahdissa, jossa se toimii omana ulkoisena palvelunaan. Tämä on hyvä vaihtoehto projekteissa, joissa työskentelee useampi ihminen ja MLflow on käytössä pitkäkestoisesti. Tutoriaali voisi MLflow'n käytön osalta toimia olla käytettävissä myös Rahdissa, mutta palvelun konfiguroinnin osalta se vaatii mukautusta.

5.5 Tutoriaalin saatavuus

Tutoriaali on saatavilla toistaiseksi tekijän henkilökohtaisten tunnusten alla, mutta se siirretään osaksi CSC:n ohjekokonaisuutta. Siitä tulee myös maininta CSC:n ohjekokoelmaan sekä Puhdin applikaation käynnistyssivulle. Tutoriaalia voitaisiin esitellä esimerkiksi CSC:n käyttäjätukeen liittyvissä tapahtumissa. Yksi tällainen on viikoittain toistuva "CSC Research Support" -tilaisuus, joka on suunnattu CSC:n palveluja hyödyntäville tutkijoille [26].

6 Päätäntö

Työ oli kokonaisuudessaan onnistunut, ja se täytti sille asetetut tavoitteet erinomaisesti. Tutoriaalin testaajilta saatiin kiitettävää ja rakentavaa palautetta, mikä osoitti, että työkalun keskeiset osat oli esitetty selkeästi ja tehokkaasti. Esimerkiksi ohjeistuksen rakenne ja sisältö olivat helposti seurattavissa, mikä mahdollisti käyttäjien sujuvan etenemisen. Tutoriaali integroituu hyvin CSC:n ohjeistuskokoelmaan, ja koska MLOps on ajankohtainen ja tärkeä aihe, sen julkaisu osuu erityisen hyvään hetkeen.

Tekijä sai arvokasta oppia paitsi itse työkalusta myös asioiden esittämisestä ymmärrettävästi ja tiedon jakamisen taidoista. Tämän myötä oppi oli hyödyksi myös muille asiantuntijoille ja organisaatiolle, jolle työ tehtiin. Työkalun ja tutoriaalin käyttöönottoprosessi ja testaus toivat esiin käytännön haasteita, jotka auttoivat tekijää kehittämään omia taitojaan entistä selkeämmäksi ja systemaattisemmaksi.

Työnantajan edustaja, joka oli mukana opinnäytetyön ohjauksessa, oli myös tyytyväinen lopputulokseen. Työn edistyminen ja sen laatu vastasivat odotuksia, ja palautteen perusteella työ tuo lisäarvoa sekä organisaatiolle että käyttäjille. Opinnäytetyö osoitti, kuinka teoreettinen osaaminen voidaan viedä käytäntöön konkreettisessa työssä, ja se täyttää sekä työnantajan että projektin sidosryhmien tarpeet.

Lähteet

[1] Mikä CSC? – CSC Company Site. [Internet]. [Viitattu marraskuu 2024]. Saatavilla: https://csc.fi/tietoa-meista/mika-csc/

[2] LUMI supertietokone - CSC Company Site. [Internet]. [Viitattu elokuu 2024]. Saatavilla: https://www.csc.fi/lumi

[3] Alzubi, J., Nayyar, A., & Kumar, A. Machine Learning from Theory to Algorithms: An Overview. J. Phys, 12012. 2018 [Viitattu kesäkuu 2024]. Saatavilla: https://doi.org/10.1088/1742-6596/1142/1/012012

[4] Bishop, C. M. Model-based machine learning. 2013 [Viitattu kesäkuu 2024]. Saatavilla: https://doi.org/10.1098/rsta.2012.0222

[5] Sidey-Gibbons, J. A. M., & Sidey-Gibbons, C. J. Machine learning in medicine: a practical introduction. 2019 [Viitattu kesäkuu 2024]. Saatavilla: https://doi.org/10.1186/s12874-019-0681-4

[6] Mohseni, S. & Pitale M. & Singh V. & Wang Z. Practical Solutions for Machine Learning Safety in Autonomous Vehicles. 2019 [Viitattu kesäkuu 2024]. Saatavilla: https://arxiv.org/abs/1912.09630

[7] Mäkinen, S. & Skogström, H. & Laaksonen, E. & Mikkonen T. Who Need MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help? 2021 [Viitattu marraskuu 2024]. Saatavilla: https://helda.helsinki.fi/server/api/core/bitstreams/46a243f5-80d5-4064-ba78-9971e8801629/content

[8] The Importance of Data Drift Detection that Data Scientists Do Not Know [Viitattu marraskuu 2024] https://www.analyticsvidhya.com/blog/2021/10/mlops-and-the-importance-of-data-drift-detection/

[9] Symeonidis G. & Nerantzis E. & Kazakis A. & Papakostas G. MLOps – Definitions, Tools and Challenges. 2022 [Viitattu marraskuu 2024] Saatavilla: https://arxiv.org/abs/2201.00162

[10] Moreschini, S. & Hästbacka D. & Taibi D. MLOps Pipeline Development: The OSSARA Use Case. 2023 [Viitattu heinäkuu 2024]. Saatavilla: https://dl.acm.org/doi/10.1145/3599957.3606211

[11] What is MLflow? - MLflow 2.11.3 documentation. [Internet]. [Viitattu heinäkuu 2024]. Saatavilla: https://mlflow.org/docs/latest/introduction/index.html

[12] Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. Accelerating the Machine Learning Lifecycle with MLflow. 2018 [Viitattu heinäkuu 2024]. Saatavilla: http://sites.computer.org/debull/A18dec/p39.pdf

[13] MLflow Tracking Server - MLflow 2.17.0 documentation. [Internet]. [Viitattu lokakuu 2024]. Saatavilla: https://mlflow.org/docs/latest/tracking.html

[14] MLflow Models – MLflow 2.17.0 documentation. [Internet]. [Viitattu lokakuu 2024]. Saatavilla: https://mlflow.org/docs/latest/models.html

[15] MLflow Model Registry – MLflow 2.17.0 documentation. [Internet]. [Viitattu lokakuu 2024]. Saatavilla: https://mlflow.org/docs/latest/model-registry.html

[16] MLflow Projects – MLflow 2.17.0 documentation. [Internet]. [Viitattu lokakuu 2024]. Saatavilla: https://mlflow.org/docs/latest/projects.html

[17] Managing machine learning workflows on CSC's supercomputers - Docs CSC. [Internet]. [Viitattu elokuu 2024]. Saatavilla: https://docs.csc.fi/support/tutorials/ml-workflows/#mlflow

[18] Introduction to Allas Storage service - Docs CSC. [Internet]. [Viitattu elokuu 2024]. Saatavilla: https://docs.csc.fi/data/Allas/introduction/

[19] Tutorial for Using MLflow in Puhti – GitHub Repository. [Internet]. [Viitattu marraskuu 2024]. Saatavilla: https://github.com/erikaster/puhti_mlflow_tutorial/

[20] DVC - Homepage. [Internet]. [Viitattu lokakuu 2024]. Saatavilla: https://dvc.org/

[21] What is Guild AI? – Guild AI FAQ. [Internet]. [Viitattu lokakuu 2024]. Saatavilla: https://guild.ai/faq/#what-is-guild-ai

[22] What is Metaflow? – Metaflow documentation. [Internet]. [Viitattu lokakuu 2024]- Saatavilla: https://docs.metaflow.org/introduction/what-is-metaflow

[23] Kubeflow Introduction – Kubeflow documentation. [Internet]. [Viitattu lokakuu 2024]. Saatavilla: https://www.kubeflow.org/docs/started/introduction/

[24] MLflow AI Gateway – MLflow documentation 2.17.2 documentation. [Internet]. [Viitattu marraskuu 2024]. Saatavilla: https://mlflow.org/docs/latest/llms/deployments/index.html

[25] MLflow Tracing – MLflow documention 2.17.2 documentation. [Internet]. [Viitattu marraskuu 2024]. Saatavilla: https://mlflow.org/docs/latest/llms/index.html#id1

[26] CSC Research Support Coffee - CSC Training Calendar. [Internet]. [Viitattu marraskuu 2024]. Saatavilla: https://csc.fi/en/training-calendar/csc-research-support-coffee-every-wednesday-at-1400-finnish-time-2-2/?pk_vid=61a47fba2e2a404b17305