MLOps på Google Cloud Platform
Designrapport

Ingvild Andersen, Jon Akselberg Langholm, Erik Flæsen Dalen

19. mai 2021

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
11.03.2021	1.0	Førsteutkast	Erik F.D., Ingvild A. og Jon L.
14.03.2021	1.1	Mindre revidering etter tilbakemeldinger fra veileder	Erik F.D., Ingvild A. og Jon A. L.
19.05.2021	1.2	Språkvask	Erik F.D., Ingvild A. og Jon A. L.

Innhold

Figurer

1 Innledning

1.1 Dokumentets hensikt

Følgende dokument er en designrapport utarbeidet i sammenheng med bacheloroppgaven 'MLOps på Google Cloud Platform' på oppdrag fra Statens vegvesen. Dokumentets hensikt er å tilby leseren en konseptuell og tekstlig beskrivelse av løsningen oppgaven skal resultere i. Dokumentet er en del av prosjektprosessens kvalitetssikring, og omfatter derfor en innsikt i bedriftens krav og behov, samt en oversikt over foreslått løsningsdesign for å møte disse. Denne beskrivelsen vil sikre gjensidig kommunikasjon mellom oppdragsgiver og oppdragstaker, slik at alle parter sitter på samme mengde informasjon når dokumentet er ferdigstilt. Målet er at prosjektgruppen og Statens vegvesen skal bli samstemte i valg av løsningsdesign, tekniske detaljer og forventninger. Designrapporten bygger videre på beslutningene som ble tatt i forstudierapporten. Dokumentet begrenses til å beskrive forslag til løsning av prosjektoppgaven, men ikke hvordan man kan reprodusere denne løsningen. En slik innføring vil finne sted i driftsrapporten.[34] Forslag til løsning vil også innebære en innføring i valgt teknologi og hvordan den samsvarer med oppgavekrav fra Statens vegvesen.

1.2 Innhold

Første kapittel i dokumentet beskriver dokumentets hensikt (1.1) og en oversikt over dets innhold (1.2). Videre presenteres en kort oversikt over dokumentet og oppgavens avgrensning (1.3). Definisjoner og forkortelser befinner seg i ekstern ordbok.[32] Kapittel 2 går overordnet inn på kunden og behov, altså hvem kunden er og hva de ønsker ut av prosjektprosessen. Kapittel 3 og 4 går inn på bakgrunnen for valg av teknologi, samt en oversikt over disse. Helt konkret innebærer det overordnet innføring i MLOps (4.1.1), Google Cloud Platform (4.1.2), TensorFlow Extended (4.1.3) og Kubeflow Pipelines (4.1.4). Videre i kapittel 4 får leseren en innføring i prosjektets verktøy og ressurser, da spesielt en beskrivelse av prosjekter i Google Cloud Platform (4.2.1), samt en gjennomgang av utviklingsmiljø (4.2.2), akseptanse- og testmiljø (4.2.3) og produksjonsmiljø (4.2.4). I avsnitt 4.3 gjennomgås

detaljerte løsningsbeskrivelser, TFX (4.3.1), GCP (4.3.2), KFP (4.3.3), flyt i pipelinen (4.3.4) forutsetninger og avhengigheter (4.3.5), programvare og systemkrav (4.3.6) før oppdatert Gantt-diagram (4.3.7). Til slutt i rapporten finnes en overordnet oversikt over alternative flyter i GCP (5) før en liste over referanser og vedlegg.

1.3 Avgrensning

Som beskrevet i forstudierapporten[25] går prosjektets resultatmål ut på å ferdigstille et 'Proof of Concept' på en produksjonspipeline for en maskinlæringsmodell i Google Cloud Platform. Dette omfatter å kunne kjøre ut maskinlæringsmodeller i Google Cloud Platform, samt nye versjoner av disse. Det handler også om å kunne bruke maskinlæringsmodellene til å hente ut prediksjoner, overvåke deres prediksjonskvalitet, trening og skalering. Løsningen skal være kompatibel med Google Cloud Platform, men prosjektgruppen står fritt til å vurdere andelen selvskreven kode opp i mot eksisterende grensesnitt i GCP. Det tas forbehold om at noe funksjonalitet kan sløyfes ettersom oppgavens omfang spesifiseres videre.

Prosjektgruppen skal ikke utvikle maskinlæringsmodellene, drifte eller implementere produksjonspipeline, drive opplæring, sette opp miljø i Google Cloud Platform, eller legge til funksjonalitet som ikke er etterspurt. Dette dokumentet skal presentere og begrunne et forslag til løsningsdesign, men omfatter ikke en gjennomgang av denne løsningen. Sistnevnte kommer i driftsrapport.[34] Det må også nevnes at løsningsforslaget ikke testes i sin helhet før arbeidet med driftsdokumentasjon, noe som kan føre til endringer i valgt løsning. Om dette skulle skje vil det kommuniseres tett med oppdragsgiver, og det vil beskrives i sluttrapporten.[35]

1.4 Definisjoner og forkortelser

Det er utarbeidet en egen ordbok for prosjektet, som kan finnes i vedlagt ordbok.[32]

2 Kort om kunden og behov

Oppdragsgiver er dataplattformen Saga, på vegne av Statens vegvesen. Organisasjonen har ansvaret for Norges riksveier, noe som innebærer forvaltning, utredning, planlegging, bygging, drift og vedlikehold av disse. Statens vegvesen har gjennom Saga en visjon om å gjøre organisasjonen mer datadrevet. Hensikten er å bruke data til å oppdage nye sammenhenger og muligheter, ta mer faktabaserte beslutninger, samt utnytte organisasjonens store datamengder på en mer hensiktsfull måte. Dagens løsning preges av uheldige formater og lite tilfredsstillende API-er. Deling av data og tversanalyser vektlegges også i liten grad.[1]

Saga ønsker at prosjektgruppen skal utforske og utvikle et 'Proof of Concept', i form av en produksjonspipeline som kjører i Google Cloud Platform. Denne løsningen skal kunne ta inn og bytte ut maskinlæringsmodeller, mate disse med ny inndata, gjennomføre trening, være skalerbar, hente ut prediksjoner, samt overvåke kvaliteten på disse. Produksjonspipelinen skal være mest mulig automatisert, og skal kreve minst mulig interaksjon fra datavitere. En ønskesituasjon er at man kan sette maskinlæringsmodeller ut i produksjon, og at trening, testing, overvåking og serving av prediksjoner er tilnærmet selvgående. Dette vil dekke de behov SVV ønsker for å kunne ta i bruk dataen de sitter på så effektivt som mulig. Oppgaven skal løses ved bruk av Google Cloud Platform, og underliggende verktøy er til disposisjon. Prosjektgruppen står likevel fritt til å utforske andre alternativer, så lenge løsningen fungerer godt med plattformen. Oppdragsgiver er fortsatt i utforskningsfasen når det kommer til maskinlæring, og ønsker derfor at løsningen skal inneholde en sammenligning av løsninger, og en forklaring på hvorfor gjeldende produkt-teknologi-løsning ble valgt. Bakgrunnen for dette er at de vil finne teknologien som passer best for deres drift og bruk.

3 Hvorfor valgt teknologi

Valg av plattform ble som tidligere nevnt Google Cloud Platform. Dette valget ble innlysende da Statens vegvesen allerede bruker denne skyløsningen, og å samle alle tjenester på ett sted forenkler utvikling, drift og vedlikehold betraktelig. GCP er også en skyplattform som tilbyr flere verktøy for maskinlæring, noe som gir en stor mengde muligheter når man skal angripe maskinlæringsoppgaver. Plattformen er også langt framme når det kommer til maskinlæring, sammenlignet med lignende plattformer på markedet. Videre ble det besluttet å ta i bruk skalerbar teknologi som kunne integreres sømløst med GCP, og som spesielt kunne prosessere maskinlæringsmodeller basert på TensorFlow. Bakgrunnen for at prosjektgruppen har valgt å fokusere på kun maskinlæringsmodeller basert på TensorFlow, er at TensorFlow er det mest utbredte rammeverket for utvikling av maskinlæringsmodeller.[31] Det ble besluttet å ta i bruk TensorFlow Extended (TFX) som rammeverk og Kubeflow Pipelines som orkestrator, da disse teknologiene er godt tilpasset slike modeller. TFX er de individuelle komponentene i pipelinen, og Kubeflow Pipelines passer på at disse kjøres i riktig rekkefølge. Disse tjenestene er også skalerbare ende-til-ende tjenester som kan utvides og egendefineres etter bedriftens ønske i senere tid. Blant annet kan Kubeflow Pipelines nyttes som orkestrator for maskinlæringsmodeller basert på andre teknologier enn TensorFlow.

Rammeverk og orkestrator er begge tjenester som er åpen for fri bruk og vil ikke påløpe kostnader ved bruk av disse. Pipelinen vil kjøre integrert med GCP, og det vil derav påløpe kostnader tilknyttet bruk av GCPs tjenester. Da prosjektet fortsatt er i en eksperimentell fase er det vanskelig å gi eksakte tall. For øvrig kan bedriften senere tilpasse selv hvor mye de vil integrere pipelinen med GCP. En tett integrasjon vil føre til høyere kostnader, men skyplattformen har også avanserte maskinlæringsverktøy som kan sikre kvalitet i trening, overvåking og levering av prediksjoner. Disse verktøyene er også svært avanserte, noe som gjør dem vanskelig å gjenskape på egen hånd utenfor plattformen.

4 Tekniske løsninger

Valg av løsning ble gjort på bakgrunn av krav fra Statens vegvesen og omfang av prosjektet. Det ble tidlig i planleggingen av løsningsdesign besluttet at teknologivalg skulle implementere prinsippene bak MLOps, som er maskinlæringens svar på DevOps.[32] Bakgrunnen var at løsningen skulle være en automatisk trigget pipeline, der de som utvikler maskinlæringsmodeller, og de som har dem i produksjon, skal kunne samarbeide uten problemer. Dette gjelder helt fra modellene påbegynnes til de kjøres ut i produksjon og leverer prediksjoner. MLOps reflekterer dette ved å legge opp til automatisering hele veien fra man setter modeller ut i produksjon, til man leverer prediksjoner. Videre ble det besluttet å ta i bruk Google Cloud Platform for å lagre og kjøre løsningen og dens underliggende data, da dette er en skyplattform Statens vegvesen bruker i det daglige og er godt kjent med.

Når man skal opprette en automatisert maskinlæringspipeline behøver man ikke bare komponenter, men også et rammeverk og en orkestrator. Rammeverket konfigurerer pipelinen, og orkestratoren kontrollerer kjøringen av denne. Det vil si at orkestratoren styrer og dirigerer rekkefølgen på komponentene og arbeidet pipelinen består av. Det finnes flere rammeverk å velge i når man skal sette sammen maskinlæringspipeliner, og det ble i dette prosjektet besluttet å gå videre med TensorFlow Extended (TFX). TensorFlow er svært utbredt innen utvikling av modeller i dag, og det er derfor ønskelig med en pipeline som best mulig kan tilpasses disse. TFX er et ende-til-ende verktøy, og rammeverket støtter opp om alt fra førprosessering av data til trening og levering av produksjonsklare modeller. Rammeverket implementerer også prinsippene bak MLOps, og derav ønsket funksjonalitet og krav fra oppdragsgiver. Som orkestrator skal prosjektet ta i bruk Kubeflow Pipelines, som er en plattform for å bygge og kjøre ut portable og skalerbare maskinlæringspipeliner basert på Docker-konteinere. Løsningen tilbyr en ende-til-ende orkestrering som legger til rette for gjenbruk av kode, pipeliner og komponenter, noe som lønner seg for Statens vegvesen, som ønsker en mulighet for rask skalering.

4.1 Introduksjon til teknologi

Dette avsnittet skal overordnet beskrive de mest aktuelle verktøyene for å løse oppgavebeskrivelsen. Det er ikke en oversikt over valgt løsning, men heller en mulighet for leser å sette seg inn i de ulike verktøyenes funksjonalitet. Det er fordelaktig at leser kjenner til den viktigste funksjonaliteten som er aktuell, da det senere i rapporten skal argumenteres for valg av endelig løsning. Statens vegvesen er relativt ferske innen maskinlæring, og ønsker derfor en oversikt over mulige løsninger, i tillegg til en beskrivelse av og begrunnelse for valgt løsning. Sistnevnte gjennomgås i avsnittet detaljert løsningsbeskrivelse".

4.1.1 MLOps

Følgende informasjon som beskriver MLOps er i store deler basert på dokumentasjon fra Google Cloud Platform. Dette omfatter i hovedsak løsningsartiklene *Architecture for MLOps using TFX, Kubeflow Pipelines, and Cloud Build*[5] og *MLOps: Continuous delivery and automation pipelines in machine learning*.[2]

MLOps og DevOps baserer seg begge på konseptene rundt kontinuerlig integrasjon (CI) og kontinuerlig levering (CD), men CI innen MLOps handler ikke kun om testing og validering av kode, men også av data, dataskjema og modeller. CD omfatter ikke kun levering av en enkelt programvarepakke, men hele systemer og pipeliner. MLOps presenterer også et nytt konsept kalt kontinuerlig trening (CT), som er unikt innen maskinlæringssystemer. Dette beskriver en automatisk og iterativ om-trening og levering av modeller. Metoden er mye mer omfattende enn DevOps innen testing, som omfatter datavalidering, samt en kontinuerlig evaluering av modellenes kvalitet. Utkjøring kan også være utfordrende, da modeller behøver en produksjonspipeline som trener og kjører ut nye versjoner. Dårlige ytelse i maskinlæringssystemer skyldes som regel endringer i dataprofiler, i motsetning til lite optimal kode hos programvaresystemer. Maskinlæringsmodeller må konstant overvåkes, og ha klare tiltak ved endringer i data eller en reduksjon i prediksjonskvalitet.[2]

MLOps handler altså om å forenkle samarbeidet mellom de som utvikler maskinlæringssystemer, og de som vedlikeholder dem. Praksisen vektlegger

en størst mulig grad av automatisering når modeller skal ut i produksjon. Hos bedrifter som planlegger få maskinlæringsmodeller med et lavt vedlikeholdsbehov, så kan en produksjonspipeline med manuelle triggere for trening, validering og utkjøring holde mål. Dette gjelder ikke Statens vegvesen, da det er en stor organisasjon som satser tungt på digitalisering, maskinlæring og automatisering av arbeidsoppgaver. I deres tilfelle vil det lønne seg å innføre MLOps først som sist, noe som vil omfatte at hele pipelinen består av automatiske triggere. Her skal alt fra innførsel av data til levering av prediksjoner skje så sømløst som mulig.

Levere en modell til produksjon

Stegene fra utvikling til overvåkning av en maskinlæringsmodell kan overordnet beskrives som vist under. Når modellen utarbeides for første gang handler de første punktene også om å skape selve modellen, når modellen settes i produksjon kan disse tolkes som å hente inn ny data, analysere datatype og forberede disse for modellen.

1. **Hente ut data**
   Velge relevant data fra diverse datakilder.
2. **Analysere data**
   Undersøkende dataanalyse utført av dataviter. Handler om å forstå tilgjengelig data som behøves for å bygge maskinlæringsmodellen.
3. **Forberede data**
   Her formatteres og renses data, samt splittes inn i sett for trening, validering og testing.
4. **Trening av modellen**
   Her implementeres i første omgang forskjellige algoritmer som trenes på forberedt data, samt en innstilling av hyperparametere. Dette er parametere som kontrollerer maskinlæringsmodellens læringsprosess. De påvirker ikke læringen i seg selv, men kvaliteten på prosessen rundt. Disse varierer for eksempel ut ifra hvilken algoritme dataviteren velger å basere modellen på. Resultatet er en ferdigtrent modell.
5. **Evaluering av modellen**
   Modellen evalueres på et sett av data den ikke har blitt eksponert for

tidligere, også kalt test-sett. Utdata fra denne fasen er metrikker som måler modellens prestasjon.

6. **Validering av modellen**
   Om modellen bekreftes god nok for utkjøring, og dens prediksjoner er bedre enn grunnlinja (baseline), eller tidligere godkjente modell, blir den godkjent for levering.

7. **Levering av modellen**
   Den validerte modellen kjøres ut til et miljø for å levere prediksjoner. Denne utkjøringen kan enten være som en mikrotjeneste med et REST API for å levere nettbaserte prediksjoner, en innebygget modell til en mobil enhet, eller som en del av et system som henter ut prediksjoner via terminal.

8. **Overvåkning av modellen**
   Når modellen er kjørt ut i produksjonsmiljø, blir den kontinuerlig overvåket. Om evnen til korrekte prediksjoner reduseres, kan det igangsettes en ny iterasjon av hele prosessen.

Videre gjennomgang av eksisterende teknologiløsninger vil så nært som mulig gjenspeile denne prosessen.

4.1.2 Plattform: Google Cloud Platform (GCP)

Google Cloud Platform er en skytjeneste som leveres av Google. Plattformen tilbyr et stort antall datatjenester som blant annet datalagring, dataanalyse og maskinlæring. Den tilbyr både IaaS, PaaS og 'serverless computing'. Statens vegvesen er allerede kjent med GCP, noe som gjorde det naturlig at bedriften ønsket en løsning som er integrerbar med denne. Utviklings-, akseptansetest- og produksjonsmiljø er av den grunn opprettet her. Sett bort i fra bedriftens krav, så er Google Cloud Platform fortsatt et solid valg av maskinlæringsplattform. GCP tilbyr et stort antall maskinlæringsverktøy, og man skal teoretisk sett kunne trene, validere og levere modeller uten å bruke programmeringskode. Plattformen tilbyr lagring av data under alle steg av pipelinen, samt trening direkte på plattformen gjennom tjenesten AI Platform. Om utviklere ønsker å lage pipelinen selv, men med egen kode, så kan dette

også kjøres i GCP, og man har valget om hvor mange av plattformens egne tjenester som skal blandes inn.[16] Dette legger opp til at pipeliner kan egendefineres, og tilpasses bedriften og maskinlæringsalgoitmene best mulig.

Google Cloud Platform tilbyr et stort antall tjenester for maskinlæring. I dette prosjektet skal man ikke nødvendigvis ta i bruk alle nevnte tjenester, men siden GCP er så tett integrerbart med denne løsningen, er det fordelaktig at bedriften har en overordnet oversikt over mulighetene.[20]

- **Cloud AutoML**
  En tjeneste som lar brukerne automatisk trene, evaluere, forbedre og kjøre ut egendefinerte maskinlæringsmodeller direkte i Google Cloud. Her benyttes kun et grafisk grensesnitt. Om brukeren ønsker det kan man også koble egendefinert kode opp mot AutoML.[17]

- **AI Platform**
  Her kan utviklere av maskinlæringsmodeller ta deres modeller fra utvikling til produksjon og levering. AI Platform er integrert med blant annet BigQuery og Data Labeling Service (DLS). BigQuery hjelper med lagring av store mengder data, og DLS bistår med å merke og forberede data for trening og testing av modellen. AI Platform kan også integreres med pipeliner man lager selv, da spesielt gjennom Kubeflow Pipelines og TFX-komponenter.[18] Dette gjennomgås nærmere i neste avsnitt.

- **AI Hub**
  Google Cloud tilbyr hver enkelt bruker en egen katalog for å laste opp og eksperimentere med maskinlæringsmodeller, AI-komponenter og ende-til-ende pipeliner. Her kan man dele ML-pipeliner, notisbøker, modeller og annet ML-innhold.[19]

- **BigQuery ML**
  Lar deg opprette og kjøre maskinlæringsmodeller i BigQuery gjennom bruk av standard SQL-spørringer. Lar altså SQL-brukere bygge modeller ved hjelp av eksisterene SQL-kunnskap. Nyttig verktøy spesielt for datavitere som vil evaluere modeller i BigQuery, uten at de har mye kunnskap om maskinlæringrammeverk i forkant.[21]

- **Cloud Storage Buckets**
  Ikke et maskinlæringsverktøy i seg selv, men svært sentralt for å kunne sette modeller i produksjon. Når man skal behandle en maskinlæringsmodell må modellen, data og tilhørende artefakter ha en lagringsplass. Dette skjer via Cloud Storage Buckets. Her oppretter enten utvikleren en bucket manuelt, eller kjører kode som oppretter dem automatisk. Disse kan nås innad hvert enkelt prosjekt, om ikke man tildeler spesielle tillatelser. Cloud Storage behøves blant annet for å lagre modelldata som treningsapplikasjon, modell, artefakter, inndata (treningsdata) og utdata.[22]

Dette er bare et lite utvalg av hva som finnes av maskinlæringsverktøy i GCP. Det finnes også andre tjenester som kan være nyttig, som Dataflow[23] og Dataproc,[24] men disse gjennomgås ikke i dybden per nå.

4.1.3 Rammeverk: TensorFlow Extended (TFX)

TensorFlow Extended (TFX) tilbyr et rammeverk for å opprette pipeliner bestående av TFX-komponenter. Dette inkluderer delte biblioteker for integrering, som behøves for å kjøre og overvåke TensorFlow maskinlæringsmodeller.[3] Dette rammeverket er et ende-til-ende-verktøy som støtter opp under alt fra transformasjon av data til trening og levering av produksjonsklare modeller.[4] TFX implementerer prinsippene bak MLOps, og de forskjellige komponent tilbyr lagring, konfigurering og orkestrering av maskinlæringsmodeller. TFX-pipeliner er skalerbare og takler tunge maskinlæringsoppgaver, som modellering, trening, og validering. TFX-pipeliner takler også levering og administrering av utkjøringer. TFX-pipeliner kan orkestreres av Apache Airflow og Kubeflow Pipelines.

TensorFlow Extended har flere underliggende biblioteker som støtter opp under ønsket pipeline-funksjonalitet, slik som trening og utkjøring, samt integrasjon med GCP. Videre vil fasene i MLOps listes opp, sammen med TFX-komponenter som løser de forskjellige fasene, og tjenester disse integreres med i Google Cloud Platform.

Eksempelpipeline basert på TFX:
Informasjon er hentet fra Google Cloud Platforms dokumentasjon.[5]

1. **Hente ut data**
   I første steg høstes data, og det kommer datafiler ut som brukes til trening og evaluering av modellen.

2. **Validere data**
   Her kan man bruke TensorFlow-biblioteket *TF Data Validation (TFDV)*[6] for å detektere dataanormaliteter. Her valideres ny data mot dataskjema, og om det oppdages avvik må enten skjema eller data oppdateres. TFDV fungerer godt på GCP gjennom Dataflow.
   Kan integreres med Google Cloud: Dataflow.

3. **Transformasjon av data (førprosessering)**
   Her kan biblioteket *TF Transform (TFT)*[7] tas i bruk. Når inndata er ferdig validert, splittes det som kjent opp i test- og treningssett for maskinlæringsmodellen, noe TFT kan bidra med. Systemet sitter igjen med filer for å trene og evaluere modellen, vanligvis i *TFRecords* format. Det opprettes også transformasjonsartifakter som bidrar med å eksportere den lagrede modellen etter trening.
   Kan integreres med Google Cloud: Dataflow.

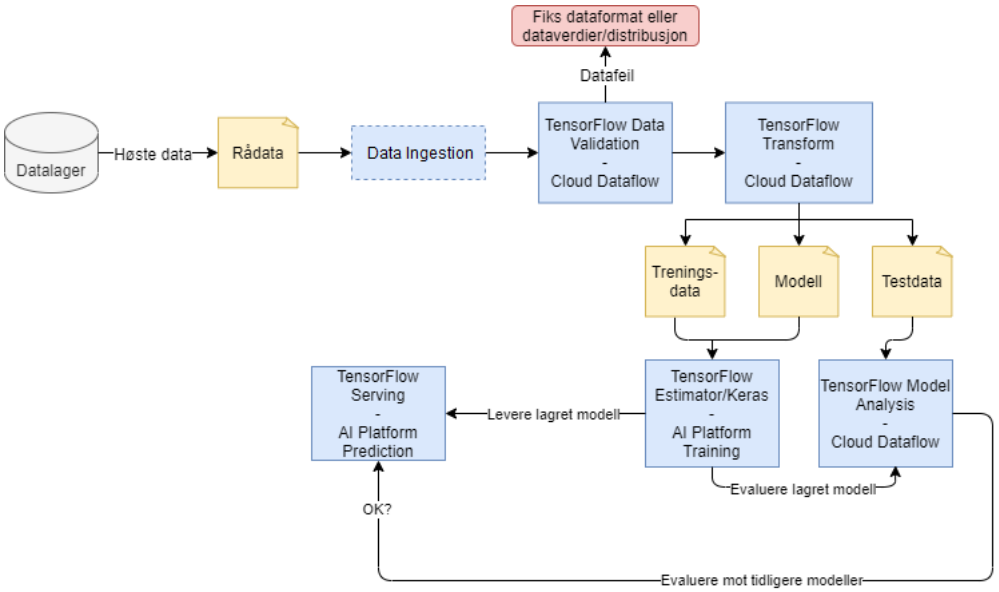
4. **Trening og tilpassing av modell**
   For å implementere og trene maskinlæringsmodellen kan *TF Estimators og Keras*[8] tas i bruk. Disse tilbyr API-er som kan ta i bruk utdata fra punkt 3 (transformasjon). *Keras tuner* kan tilpasse hyperparametere, eller man kan bruke andre tjenester som *Katib*.[33] Dette steget ender med en lagret modell som brukes til evaluering, og en annen som brukes til levering av modellen og prediksjoner til et grensesnitt.
   Kan integreres med Google Cloud: AI Platform: Jobs.

5. **Evaluering og validering av modell**
   Modellen evalueres på et test-datasett for å måle dens prediksjonskvalitet. Her kan man bruke *TF Model Analysis (TFMA)*,[9] som evaluerer modellkvaliteten i sin helhet, og identifiserer hvilke deler av modellen som yter dårlig. Denne delen bestemmer om modellen skal settes ut i produksjon eller ikke.
   Kan integreres med Google Cloud: Dataflow.

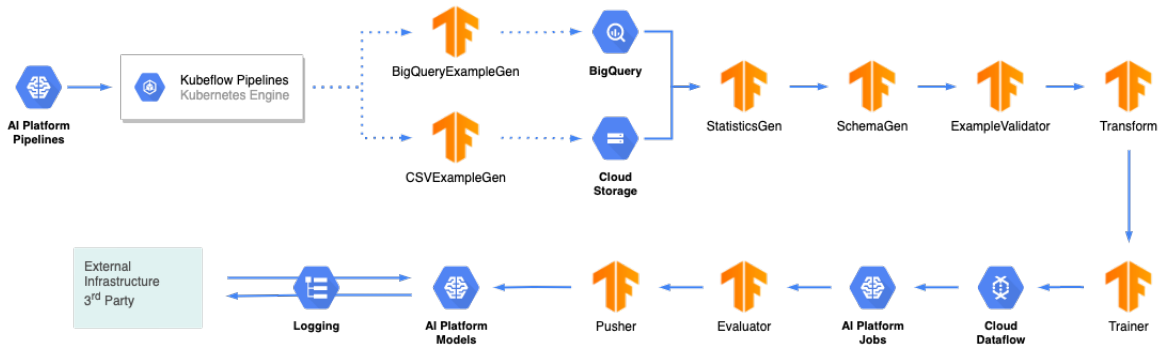
6. **Levering av modell for prediksjoner**

Når modellen er validert, kan den leveres som en mikrotjeneste for å gi nettbaserte prediksjoner via biblioteket *TF Serving (TFS)*.[10] Utdata blir en prediksjon fra den trente maskinlæringsmodellen. Eventuelt kan man lagre den trente modellen i et modellregister.

Kan integreres med Google Cloud: AI Platform: Models.



Figur 2: Overordnet flyskjema for TFX-pipeline. Inspirert av TensorFlow-dokumentasjon.[5]



Figur 3: Komponentspesifikk eksempelflyt i TFX-pipeline på GCP

4.1.4 Orkestrering: Kubeflow Pipelines (KFP)

En orkestrator behøves for å koble sammen de ulike komponentene i produksjonspipelinen. Den sørger for at pipelinen går automatisk i ønsket sekvens, etter planlagte triggere[11] slik at neste komponent i pipelinen får

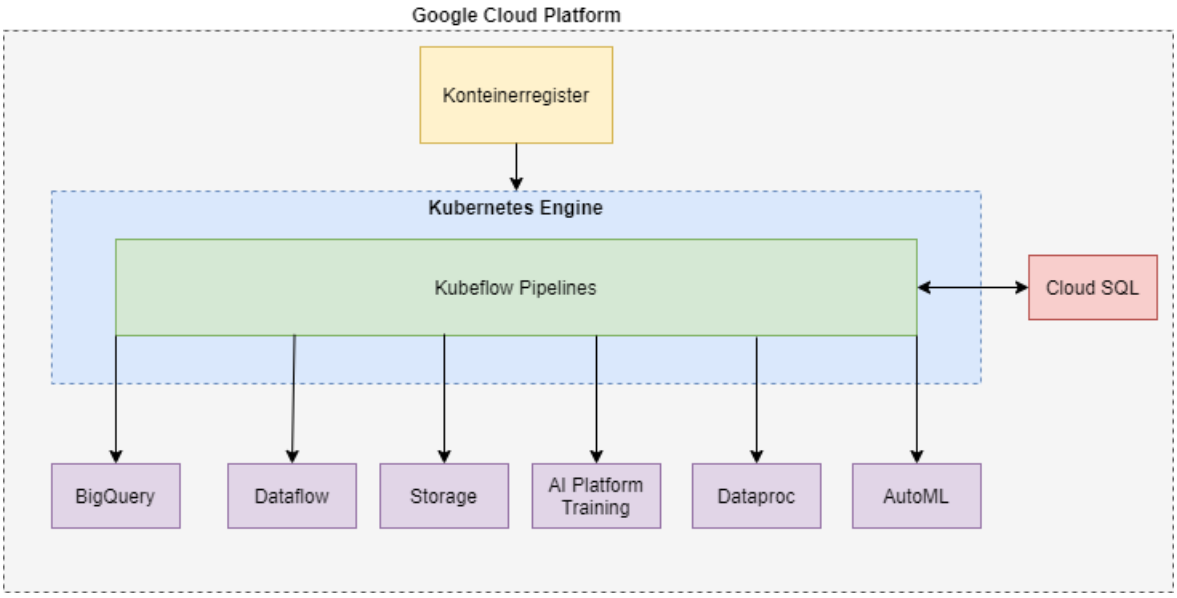
nødvendig inn-data fra foregående komponent. Kubeflow er et rammeverk som stammer fra konteiner-orkestratoren Kubernetes, og det består av flere tjenester for maskinlæring. En av disse tjenestene kalles for Kubeflow Pipelines (KFP). KFP lar deg sette sammen, orkestrere og automatisere maskinlæringssystemer. Hver komponent kan kjøre på Kubeflow eller Google Cloud Platform. Målet er å skape en ende-til-ende orkestrering, samt legge til rette for å enkelt kunne gjenbruke kode og pipeliner. KFP består av en 'engine' for planlegging av hvordan pipelinene skal kjøres. Her brukes Argo Workflows[12] for å orkestrere Kubernetes-ressursene. Videre innehar KFP en Python SDK[13] for å definere og manipulere komponentene. Orkestrereren kan også samhandle med notebooks som Jupyter, og den har en metadata-lagring for informasjom om kjøring, modeller, datasett og andre artefakter. Kubeflow Pipelines tilbyr også flere definerte komponenter som kan kjøre tjenester hos Google Cloud Platform. Disse komponentene bidrar til at man kan utføre oppgaver gjennom tjenester som BigQuery, Dataflow, Dataproc og AI Platform.

Oppbygging

- **Komponenter**
  Kode pakket som Docker-avbildninger.[14] En enkelt komponent tar inndata-argumenter, produserer utdata-filer og utfører en fase av pipelinen.
- **Bestemme sekvens**
  Her spesifiseres sekvesen pipelinen skal kjøre i. Dette defineres gjennom et Python *domain-specific language (DSL)*.[15] Et steg i pipelinens definisjon vekker en komponent i pipelinen. Om den er svært kompleks kan komponenter også kjøre iterativt, eller kun kjøres etter planlagte hendelser.
- **Inndata-parametere**
  En pipeline behøver et sett av inndata-parametere. Disse verdiene sendes til alle pipelinens komponenter, og innehar blant annet kriterier for hvordan data skal filtreres, og hvor man skal lagre artefaktene som pipelinen produserer.

Om man velger å kjøre Kubeflow Pipelines integrert med Google Cloud

Platform kan det se ut som i figur 4. Dette prosjektets løsning er ikke like tett integrert med tjenestene i GCP, men det er en fordel å ha oversikt over mulig integrasjon om bedriften ønsker en større sammenkobling i fremtiden. Flytskjema er inspirert av figur i Google Cloud Platforms dokumentasjon.[5]



Figur 4: Kubeflow Pipelines integrert med Google Cloud Platforms ML-verktøy

4.2 Verktøy og ressurser

Som det tidligere har blitt nevnt så utvikles løsningen i Google Cloud Platform, og oppdragsgiver har valgt å dele dette inn i tre forskjellige GCP-prosjekt, videre omtalt som *miljø*. Disse består av utvikling-, akseptansetest- og produksjonsmiljø. Dette avsnittet vil først beskrive hva et prosjekt i Google Cloud Platform er, før det går inn på de tre miljøene denne utviklingsoppgaven er delt inn i. Implementasjon og drift av løsningen er som beskrevet i forstudierapport[25] ikke inkludert i oppgavens omfang, og vil derfor ikke tas hensyn til.

4.2.1 Prosjekter i Google Cloud Platform

Et prosjekt i Google Cloud Platform er en måte å organisere bedriftens ressurser på. Det består av et gitt antall brukere, API-er med innstillinger for autentisering og monitoring, samt en oversikt over hvor mye penger

prosjektet har brukt. Brukere har altså valget mellom å sette alle sine ressurser i et enkelt prosjekt, eller å fordele og sortere dem utover flere. Hos en stor organisasjon som Statens vegvesen har sistnevnte vært den beste metoden. Dette forenkler prosessen med å sikre at brukere kun har tilgang til de prosjektene de er involverte i, slik at man ikke ødelegger viktige eller urelaterte bedriftsressurser om man gjør en feil innad eget prosjekt. GCP har noe som kalles Identity and Access Management (IAM). Dette er prosjektes og organisasjonens «tilgangskontroll». Tilgang kan gis i forskjellige grader til ansattes Google-kontoer, eller til såkalte Service Accounts. Disse kontoene tillater applikasjoner å autentisere seg på tvers over Google Cloud-ressurser og tjenester.[28] Eksempelvis om man ønsker å flytte data fra en applikasjon til en 'bucket', så kan en Service Account være nyttig. I dette prosjektet er det som nevnt innledningsvis tildelt tre prosjekter, disse følger fremgangsmåten bak 'Utvikling, testing, akseptanse og produksjon (DTAP)[29]', som består av følgende steg:

- **Utvikling**
  Løsningen utvikles i et uviklingssystem, dette behøver ikke å være innad prosjektet. Dette systemet behøver ikke å ha noen testmuligheter.
- **Testing**
  Når utvikleren mener at løsningen er klar, leveres den til testmiljøet for å verifisere at den fungerer som forventet. I denne løsningen har det vært nødvendig å teste såpass iterativt, at utviklings- og testmiljø har gått noe om hverandre.
- **Akseptanse**
  Når testingen er suksessfull leveres løsningen til et akseptansemiljø. Her vil bedriften sikre at løsningen møter deres krav.
- **Produksjon**
  Når bedriften er fornøyd med løsningen, kan den implementeres eller tas i bruk. Dette skjer i produksjonsmiljøet.

4.2.2 Utviklingsmiljø

Prosjektgruppen har fått tildelt tre miljøer i Google Cloud Platform fra Statens vegvesen. Det første fungerer som et utviklingsmiljø, der forslag til løsning blir skapt. Selv om et utviklingsmiljø i teorien ikke behøver testmuligheter, så har dette vært et behov i denne sammenhengen. Dette har skjedd på bakgrunn av at flere verktøy for utvikling av pipeliner ligger inne i GCP, og det blir svært vanskelig å komme frem til en løsning kun basert på lokale utviklingsplattformer.

Jupyter Notebook

Det mest brukte lokale verktøyet for utvikling av løsningen har vært Jupyter Notebooks. Dette verktøyet kan kan kjøres gjennom AI Hub og virtuelle maskiner (instanser) på GCP, eller på lokal datamaskin. Jupyter Notebooks er en nettbasert applikasjon med åpen kildekode, som lar brukeren opprette og dele dokumenter som inneholder blant annet kode, likninger og virtualiseringer. Dette egner seg godt til å utvikle og trene maskinlæringsmodeller lokalt, og brukes derfor mye av datavitere.[27] Teknologien er godt integrert med Google Cloud Platform, noe som gjør det lett for datavitere å enten bruke notisbøker rett i GCP, eller å overføre ferdige notisbøker til egen AI Hub når de skulle ønske. I dette prosjektet har Jupyter Notebooks blitt brukt til å blant annet skrive og kompilere pipelinene.

4.2.3 Akseptanse- og testmiljø

Når løsningen er ferdig utviklet skal den inn i akseptanse- og testmiljø. Disse miljøene er vanligvis separerte, men i denne sammenhengen er de slått sammen. Her skal man resprodusere løsningen, altså pipelinen, med minst mulig komponenter. Målet er å se om man kan reprodusere løsningen enkelt, og bruke flere modeller og/eller pipeliner i samme prosjekt. Løsningen er altså klar, men man må oppdage hvordan den best mulig kan implementeres.

4.2.4 Produksjonsmiljø

Da oppgaven er avgrenset til utvikling og testing, men ikke implementasjon, utgår dette miljøet i denne sammenhengen. Ved videre arbeid med løsningen vil det involveres, men dette gjennomgås ikke i denne rapporten.

4.3 Detaljerte løsningsbeskrivelser

For å tilby en kontekst for løsningsbeskrivelser vil en overordnet liste over deloppgaver som må løses presenteres, som videre peker på tekniske løsninger på deloppgavene. Videre presenteres de tekniske løsningene som omfattes i detalj.

1. **Nye versjoner av modell**
   Det er ønskelig å kunne deploye nye versjoner av maskinlæringsmodeller
   Løsning: TFX Evaluator, TFX Pusher
2. **Eksponere maskinlæringsmodell**
   Maskinlæringsmodellen må kunne eksponeres for input og brukes fra andre applikasjoner
   Løsning: TFX Pusher, AI Platform: Models
3. **Overvåke maskinlæringsmodell**
   Det er ønskelig å kunne overvåke maskinlæringsmodellen og dens evne til å predikere riktig
   Løsning: GCP logging og monitoring
4. **Skalere pipeline**
   Pipeline må kunne skaleres med tanke på kapasitet og ytelse
   Løsning: Kubernetes Engine i GCP
5. **Trene maskinlæringsmodeller effektivt**
   Maskinlæringsmodeller må kunne trenes i GCP
   Løsning: TFX Trainer, AI Platform: Jobs
6. **Lagre maskinlæringsmodeller**
   Maskinlæringsmodeller må kunne lagres på en hensiktsmessig måte
   Løsning: TFX Pusher, AI Platform: Models

4.3.1 TensorFlow Extended (TFX)

Her kommer detaljert beskrivelse av de TFX-komponenter som er direkte relevant for deloppgavene i kapittel 4.3. En gjennomgang av hvordan TFX-komponentene henger sammen, samt en forklaring av de øvrige komponentene er å finne i kapittel 4.3.4.

4.3.1.1 TFX Evaluator Evaluator analyserer modellen som har blitt trent i pipelinen og ser hvordan den yter på evalueringssettet. Denne kan man konfigurere for å se nærmere på ytelse i spesifikke tilfeller. I forstudiet så vi på muligheten for å velge den modellen som yter best i det spesifikke tilfellet den skal brukes til, og beslutningsgrunnlaget for dette gis fra TFX Evaluator. Evaluator kan også brukes til å sammenligne den nye modellen med den gamle modellen, og automatisk velge å deploye den nye modellen eller skrote den.

4.3.1.2 TFX Pusher TFX Pusher deployer modellen fra pipelinen til et endepunkt. I dette tilfellet deployes modellen til AI Platform: Models. Modellen som deployes er den som blir godkjent av TFX Evaluator.

4.3.1.3 TFX Trainer Trainer utfører selve treningen av modellen. Komponenten tar inn data og konfigurasjon, og trener en modell ut i fra dette. Denne kan konfigureres til å kjøre treningen i AI Platform: Jobs, og er ønskelig å gjennomføre i dette prosjektet. Konfigurasjon av treningen må gjøres av en dataviter, så bachelorgruppen må lage en løsning for hvordan dataviteren skal få lagt inn konfigurasjonen.

4.3.2 Google Cloud Platform (GCP)

4.3.2.1 AI Platform AI Platform i GCP tilbyr en rekke funksjonalitet for maskinlæring på skyen. På AI Platform finner man alt man trenger fra å skape modeller, trene dem og eksponere dem. Funksjonalitet herfra skal integreres i pipelinen for å integrere funksjonalitet med resten av GCP.

Models Her lagres trente maskinlæringsmodeller. Disse kan man eksponere for tredjepartsapplikasjoner med API-kall som er ferdig laget i GCP. Dette gjør at man slipper å lage API-kallene selv. Pipelinen skal deploye modeller hit på slutten av pipelinen. Versjoner grupperes etter modell, og man kan spesifisere en standardversjon for hver modell.

Pipelines Under pipelines ligger instanser av Kubeflow Pipelines. Hver instans kan ha flere pipeliner og modeller, men det kan være hensiktsmessig

å skille disse i forskjellige instanser. Mer om dette i kapittel 4.3.3.

Jobs Her skjer selve treningen av modeller i AI Platform. Pipelinen skal kjøre treningen her fremfor å gjøre det lokalt på orkestratoren. All konfigurasjon av treningen vil komme fra TFX Trainer.

Notebooks Her ligger Jupyter Notebooks. Dette skal brukes som utviklingsmiljø for å utvikle pipeliner og deploye dem til orkestratoren. Når man oppretter en notebook, har den allerede alt man trenger av verktøy for å drive med maskinlæring installert, og gir på den måten et miljø som er klart til bruk og enkelt å reprodusere.

4.3.2.2 Logging og monitorering

En fordel med å integrere GCP i så mange ledd som mulig i pipelinen, er at logger samles på ett sted. Logging i GCP samles på ett sted og har et eget spørrespråk for å finne frem til logger. Ved hjelp av dette kan logger fra de forskjellige stegene i pipelinen leses av.

4.3.2.3 Kubernetes Engine

Her tilbys et kontainerbasert miljø for deploying av applikasjoner. I dette prosjektet er selve pipelinen som kjører i Kubeflow Pipelines å anse som en applikasjon som kjører i Kubernetes Engine. Applikasjoner nytter clustere for porsjonering av kapasitet.

4.3.3 Kubeflow Pipelines (KFP)

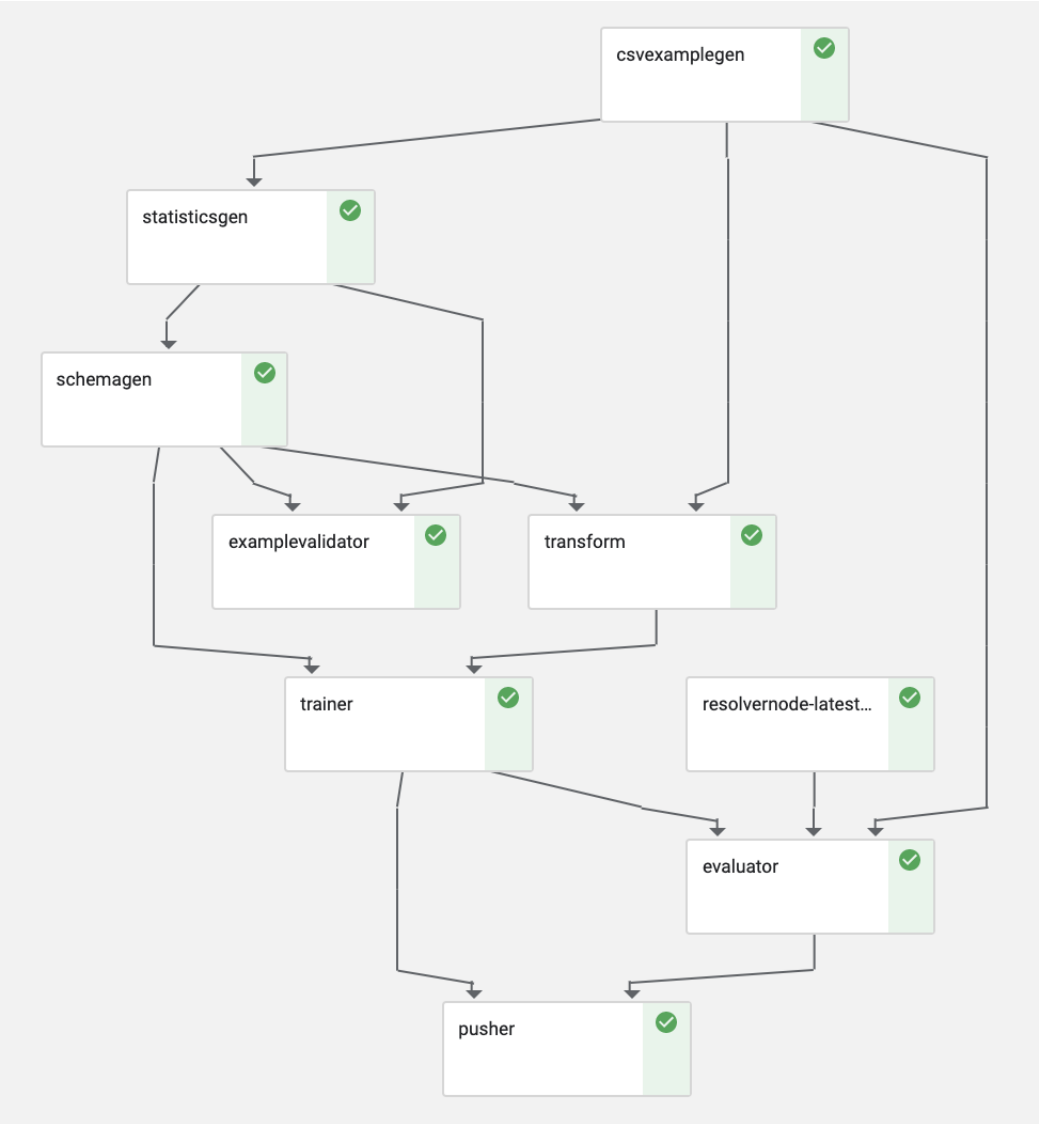
Rent teknisk konfigureres pipelinen i TFX, hvor den blir satt opp til å bruke komponenter og funksjoner i GCP. Videre kompileres TFX-koden til en konfigurasjonsfil som er lesbar for KFP. Deretter opprettes pipelinen i KFP, som kjører på et cluster i Kubernetes Engine på GCP. Dette betyr at GCP står for alt av nødvendig datakraft og at alt kan logges og monitoreres i GCP. Dette er svært nyttig da SVV allerede kjenner skyplattformen og benytter den i stor grad.

Valget om KFP som orkestrator er i hovedsak begrunnet i at det er god integrasjon mot GCP, men også at KFP er kompatibelt med andre ML-rammeverk enn TensorFlow. Dette legger til rette for å utvide støtte for

andre maskinlæringsmodeller i fremtiden. Øvrige argumenter omhandler skalerbarhet, noe Kubernetes Engine har god støtte for.

4.3.4 Flyt i pipelinen

Dette kapittelet skal beskrive flyten i pipelinen slik den blir konfigurert med TFX-komponentene beskrevet i kapittel 4.3.1.



Figur 5: TFX-komponentene satt sammen i Kubeflow

Examplegen er første stopp i pipelinen og er ansvarlig for inndataen. Dataen blir delt opp i trenings- og evalueringssett som blir brukt av Transform

og StatisticsGen. ExampleGen kan konfigureres til å hente data fra csv-filer eller direkte fra BigQuery.

Resolvernode er den andre inngangsnoden i pipelinen. Denne henter inn gjeldende versjon av modellen for å sammenligne den nye i Evaluator.

StatisticsGen genererer statistikk fra dataen fra examplegen.

Schemagen lager skjemaer av dataen fra examplegen. Skjemaet beskriver inndataen med typer, kategorier og rammer. TFX-komponenter for datavalidering, transformering og modellanalyse bruker skjemaet som kommer herfra. Skjemaet er automatisk generert og stemmer ikke nødvendigvis 100%, så dataviter bør se over og eventuelt modifiesere. Skjemaet kan skrives helt manuelt, men det er gjerne greiest å modifisere et som kommer fra denne komponenten. Skjemaet skrives i klartekst, noe som gjør det enkelt å redigere av datavitere.

Examplevalidator validerer inndataen fra examplegen mot skjemaet i schemagen og ser etter avvik. Resultatet fra denne komponenten brukes ikke videre i pipelinen.

Transform tar imot skjemaet fra Schemagen og data fra Examplegen og førprosesserer dataen ut ifra en funksjon som må forsynes av dataviter.[30] Denne funksjonen er spesifikk for hver modell og krever teknisk kompetanse innen maskinlæring, så prosjektgruppen skal ikke utvikle disse funksjonene, men legge til rette for at dataviteren kan legge inn sin.

Trainer utfører treningen, og tar inn utdataen fra Transform og SchemaGen, såvel som en konfigurasjonsfil for treningen som forsynes av dataviter. Etter Trainer har kjørt, kommer det ut en ferdigtrent modell.

Evaluator sammenligner den nylig trente modellen med en baseline eller en tidligere versjon for å vurdere ytelsen mot hverandre. Denne kan konfigureres

til å se på mindre segmenter av dataen for å sammenligne ytelsen i spesielle tilfeller. Evaluator returnerer en vurdering av beste modell.

Pusher tar modellen fra Trainer og vurderingen til Evaluator og deployer modellen til endepunktet. Dette er siste stopp i pipelinen.

4.3.5 Forutsetninger og avhengigheter

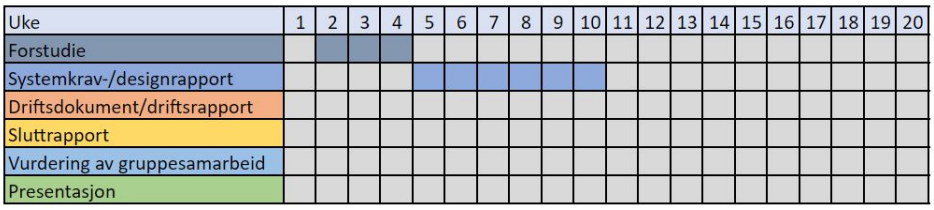
- Google Cloud Platform
- Brukerkonto eller Service Account tilknyttet prosjekt i GCP
- Maskinlæringsmodell skrevet i TensorFlow
- Konfigurerte moduler for TFX (trainer og transform)

4.3.6 Programvare og systemkrav

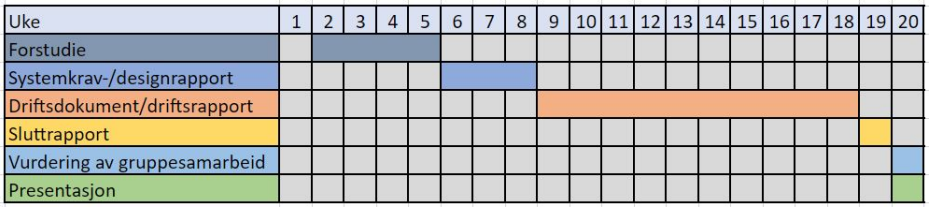
- TensorFlow >= 2.3.2
- TFX >= 0.26.1
- KFP >= 1.4.0
- Google Cloud SDK >= 327.0.0
- Python >= 3.7 < 3.9
- Kommandolinjegrensensnitt; CMD (Win), Bash (Lin), Bash/Terminal (Mac)

4.3.7 Oppdatert GANTT-diagram

Oppdatert GANTT-diagram ligger på prosjektsiden i Teams. Om leser ikke har tilgang til denne kan kopi forespørres. Figur 6 representerer den faktiske arbeidsprosessen, og figur 7 viser det som var planlagt fra start. Som man ser har designrapporten vært mer omfattende enn forventet, noe som er naturlig i et slikt innovativt prosjekt.



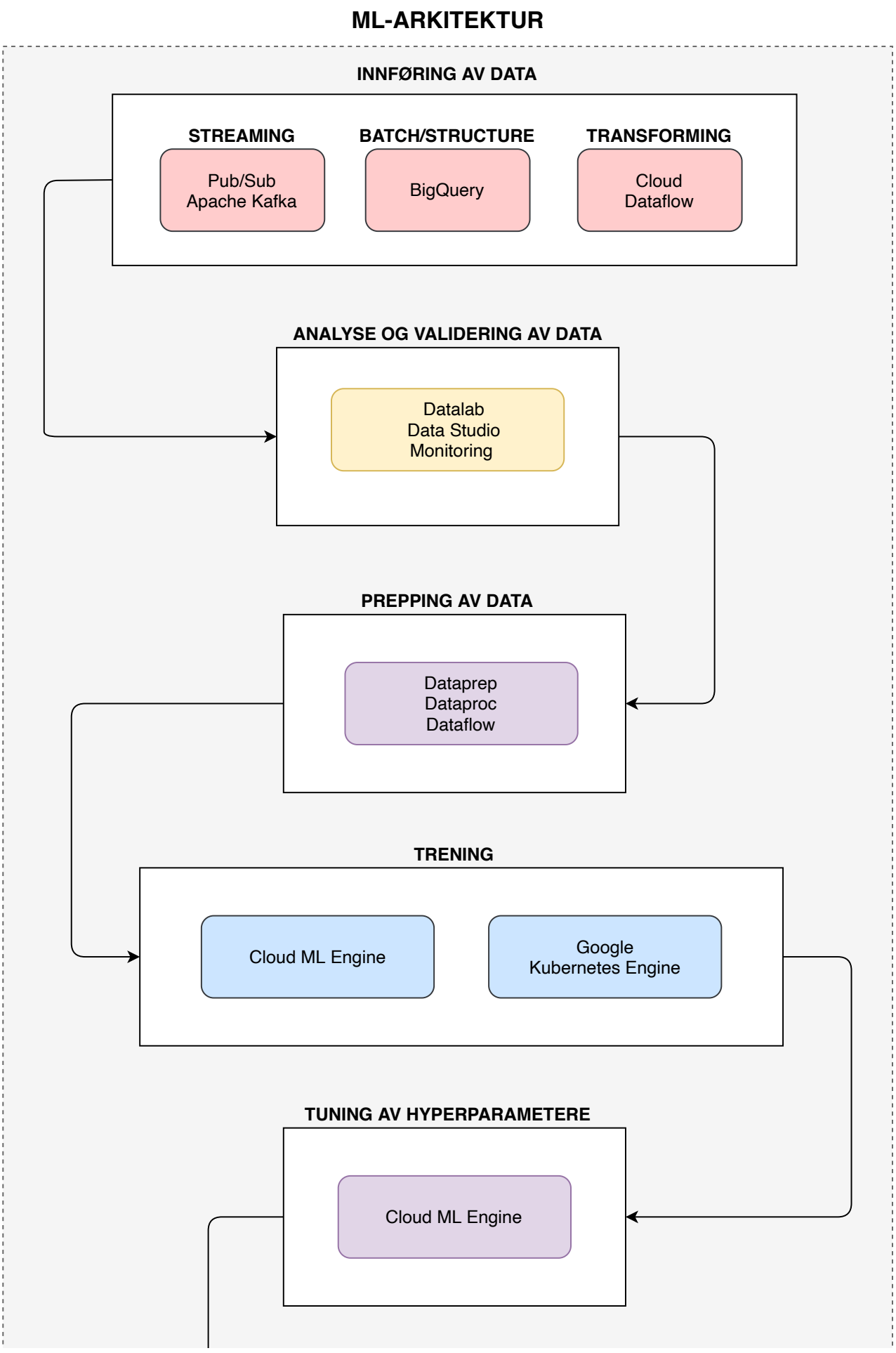
Figur 6: Faktisk arbeidsprosess.

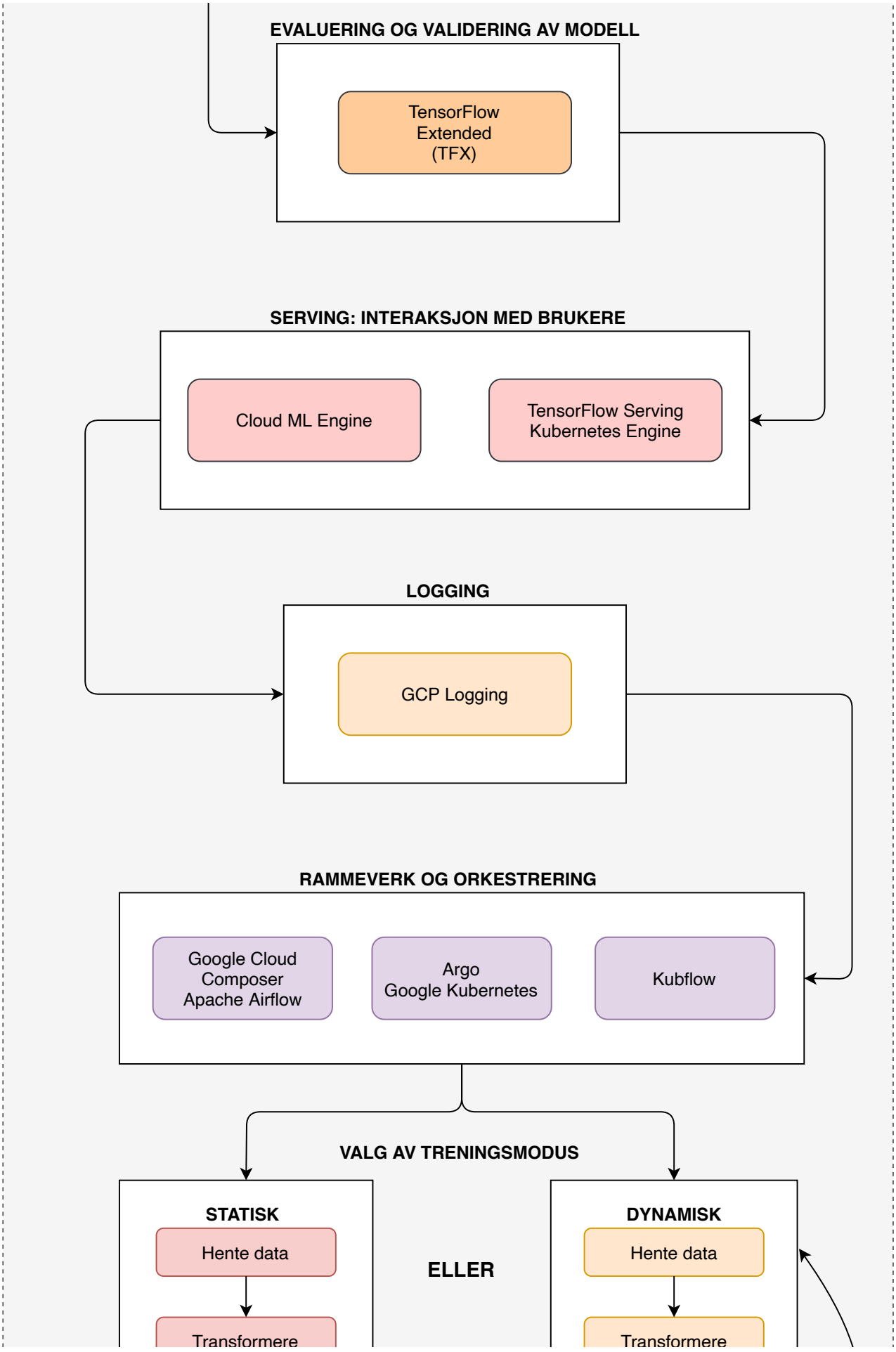


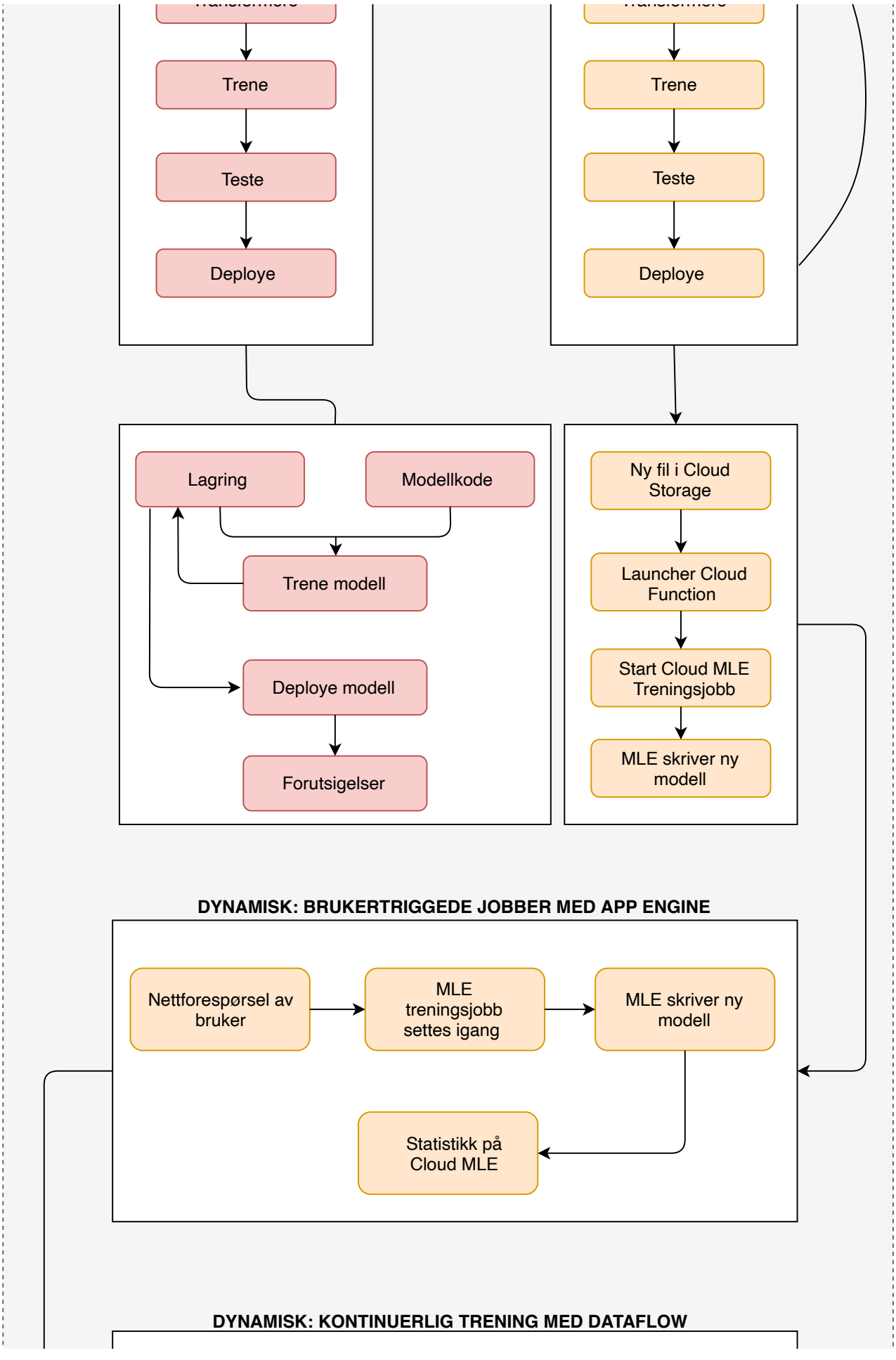
Figur 7: Planlagt arbeidsprosess.

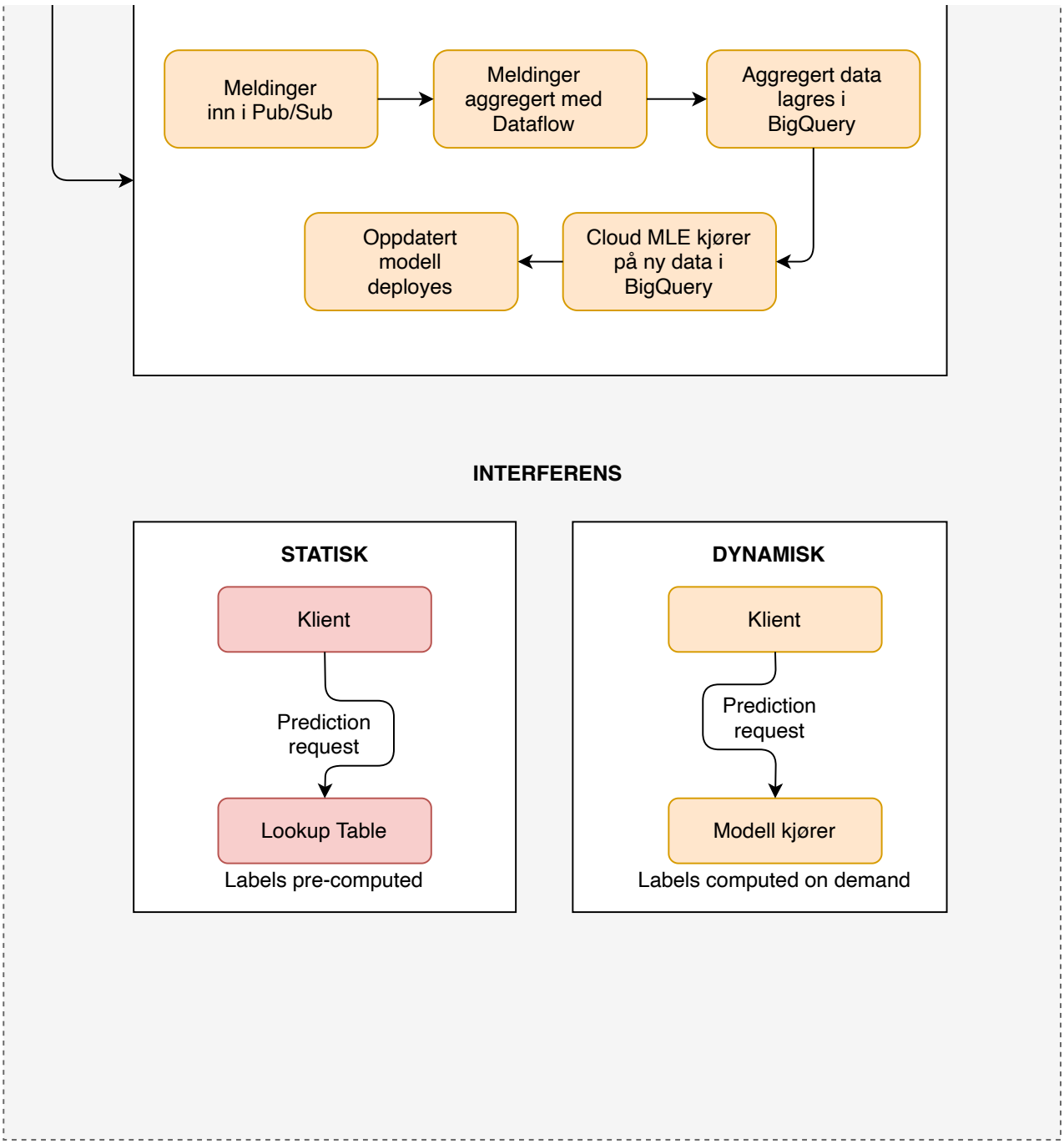
5 Alternative flyter i GCP

Ut over valgt teknologi og løsning har prosjektgruppen utarbeidet en oversikt over alternative flyter for pipeline i Google Cloud Platform som kan være til nytte for vurdering av andre løsninger. Dette er kun en overordnet oversikt og vil ikke beskrives ytterligere i dybden.


ML-ARKITEKTUR
INNFØRING AV DATA
STREAMING
Pub/Sub
Apache Kafka
BATCH/STRUCTURE
BigQuery
TRANSFORMING
Cloud
Dataflow
ANALYSE OG VALIDERING AV DATA
Datalab
Data Studio
Monitoring
PREPPING AV DATA
Dataprep
Dataproc
Dataflow
TRENING
Cloud ML Engine
Google
Kubernetes Engine
TUNING AV HYPERPARAMETERE
Cloud ML Engine


EVALUERING OG VALIDERING AV MODELL
TensorFlow Extended (TFX)
SERVING: INTERAKSJON MED BRUKERE
Cloud ML Engine
TensorFlow Serving Kubernetes Engine
LOGGING
GCP Logging
RAMMEVERK OG ORKESTRERING
Google Cloud Composer Apache Airflow
Argo Google Kubernetes
Kubflow
VALG AV TRENINGSMODUS
STATISK
Hente data
Transformere
ELLER
DYNAMISK
Hente data
Transformere


Trene
Teste
Deploye
Trene
Teste
Deploye
Lagring
Modellkode
Trene modell
Deploye modell
Forutsigelser
Ny fil i Cloud Storage
Launcher Cloud Function
Start Cloud MLE Treningsjobb
MLE skriver ny modell
DYNAMISK: BRUKERTRIGGEDE JOBBER MED APP ENGINE
Nettforespørsel av bruker
MLE treningsjobb settes igang
MLE skriver ny modell
Statistikk på Cloud MLE
DYNAMISK: KONTINUERLIG TRENING MED DATAFLOW


Meldinger inn i Pub/Sub
Meldinger aggregert med Dataflow
Aggregert data lagres i BigQuery
Cloud MLE kjører på ny data i BigQuery
Oppdatert modell deployes
INTERFERENS
STATISK
Klient
Prediction request
Lookup Table
Labels pre-computed
DYNAMISK
Klient
Prediction request
Modell kjører
Labels computed on demand

Referanser

[1] L. Meisingseth *Presentasjon om Saga*, 2019.
Hentet fra:
https://www.vegvesen.no/_attachment/2849432/binary/1349698?fast_title=Saga-+Stordataplatform+for+transportdata.pdf.
Lastet ned: 15.02.2021

[2] Google Cloud, *MLOps: Continuous delivery and automation pipelines in machine learning*, 2020.
Hentet fra:
https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.
Lastet ned: 22.02.2021

[3] TensorFlow, *The TFX User Guide*, 2021.
Hentet fra:
https://www.tensorflow.org/tfx/guide.
Lastet ned: 23.02.2021

[4] V. Tatan, "Intro to ML Ops: Tensorflow Exntended (TFX)", *towards data science*, Apr. 2020. [Online].
Hentet fra:
https://towardsdatascience.com/intro-to-ml-ops-tensorflow-extended-tfx-39b6ab1c7
Lastet ned: 23.02.2021

[5] Google Cloud, *Architecture for MLOps using TFX, Kubeflow Pipelines, and Cloud Build*, 2021.
Hentet fra:
https://cloud.google.com/solutions/machine-learning/architecture-for-mlops-using-tfx-kubeflow-pipelines-and-cloud-build.
Lastet ned: 23.02.2021

[6] GitHub: TensorFlow, *TensorFlow Data Validation*, 2021.
Hentet fra:
https://github.com/tensorflow/data-validation.
Lastet ned: 23.02.2021

[7] TensorFlow, *Get Started with Tensorflow Transform*, 2021.
Hentet fra:
https://www.tensorflow.org/tfx/transform/get_started.
Lastet ned: 23.02.2021

[8] TensorFlow, *Estimators*, 2021.
Hentet fra:
https://www.tensorflow.org/guide/estimator.
Lastet ned: 23.02.2021

[9] TensorFlow, *Getting Started with TensorFlow Model Analysis*, 2021.
Hentet fra:
https://www.tensorflow.org/tfx/model_analysis/get_started.
Lastet ned: 23.02.2021

[10] TensorFlow, *Serving Models*, 2021.
Hentet fra:
https://www.tensorflow.org/tfx/guide/serving.
Lastet ned: 23.02.2021

[11] Kubeflow, *Overview of Kubeflow Pipelines*, 2020.
Hentet fra:
https://www.kubeflow.org/docs/pipelines/overview/pipelines-overview/.
Lastet ned: 27.02.2021

[12] Argo Project, GitHub, *Argo Documentation*, 2021.
Hentet fra:
https://argoproj.github.io/argo-workflows/.
Lastet ned: 01.03.2021

[13] Kubeflow, *Building Pipelines with the SDK*, 2020.
Hentet fra:
https://www.kubeflow.org/docs/pipelines/sdk/.
Lastet ned: 01.03.2021

[14] Docker Docs, *docker image*, 2021.
Hentet fra:

https://docs.docker.com/engine/reference/commandline/image/.
Lastet ned: 01.03.2021

[15] Wikipedia, *Domain-specific language*, 2021.
Hentet fra:
https://en.wikipedia.org/wiki/Domain-specific_language.
Lastet ned: 01.03.2021

[16] Google Cloud, *Google Cloud Overview*, 2020.
Hentet fra:
https://cloud.google.com/docs/overview.
Lastet ned: 05.03.2021

[17] Google Cloud, *AutoML*, 2020.
Hentet fra:
https://cloud.google.com/automl/docs.
Lastet ned: 05.03.2021

[18] Google Cloud, *AI Platform*, 2020.
Hentet fra:
https://cloud.google.com/ai-platform.
Lastet ned: 05.03.2021

[19] Google Cloud, *AI Hub documentation*, 2020.
Hentet fra:
https://cloud.google.com/ai-hub/docs.
Lastet ned: 05.03.2021

[20] crossML engineering, "Google Cloud Platform (GCP) for Machine Learning AI", *Medium*, Jul. 2020. [Online].
Hentet fra:
https://medium.com/crossml/google-cloud-platform-gcp-for-machine-learning-ai-361
Lastet ned: 05.03.2021

[21] Google Cloud, *What is BigQuery ML?*, 2020.
Hentet fra:
https://cloud.google.com/bigquery-ml/docs/introductions.
Lastet ned: 05.03.2021

[22] Google Cloud, *Working with Cloud Storage*, 2020.
Hentet fra:
`https://cloud.google.com/ai-platform/training/docs/working-with-cloud-storage`.
Lastet ned: 05.03.2021

[23] Google Cloud, *Dataflow*, 2020.
Hentet fra:
`https://cloud.google.com/dataflow`.
Lastet ned: 05.03.2021

[24] Google Cloud, *Dataproc*, 2020.
Hentet fra:
`https://cloud.google.com/dataproc`.
Lastet ned: 05.03.2021

[25] J. A. Langholm, I. Andersen og E. F. Dalen, "Forstudierapport", *NTNU*, Feb. 2021. [Vedlegg].
Hentet fra:
Vedlegg i innlevering av oppgave.

[26] J. A. Langholm, I. Andersen og E. F. Dalen, "Ordbok", *NTNU*, Feb. 2021. [Vedlegg].
Hentet fra:
Vedlegg i innlevering av oppgave.

[27] Jupyter, *The Jupyter Notebook*, 2020.
Hentet fra:
`https://jupyter.org/`.
Lastet ned: 05.03.2021

[28] Google Cloud, *Projects*, 2021.
Hentet fra:
`https://cloud.google.com/storage/docs/projects`.
Lastet ned: 05.03.2021

[29] Wikipedia, *Development, testing, acceptance and production*, 2009.
Hentet fra:
`https://en.wikipedia.org/wiki/Development,_testing,_acceptance_`

and_production.
Lastet ned: 05.03.2021

[30] Microsoft, *Feature engineering in machine learning*, 2020.
Hentet fra:
https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/create-features

[31] towards data science, *Which deep learning framework is the best?*, 2020.
Hentet fra:
https://towardsdatascience.com/which-deep-learning-framework-is-the-best-eb51431

[32] J. A. Langholm, I. Andersen og E. F. Dalen, "Ordbok", *NTNU*, Mai 2021. [Vedlegg].
Hentet fra:
Vedlegg i innlevering av oppgave.

[33] Kubeflow, *Kubeflow Katib: Scalable, Portable and Cloud Native System for AutoML*, 2021
Hentet fra:
https://blog.kubeflow.org/katib/
Lastet ned: 07.04.2021

[34] J. A. Langholm, I. Andersen og E. F. Dalen, "Driftsrapport", *NTNU*, Mai 2021. [Vedlegg].
Hentet fra:
Vedlegg i innlevering av oppgave.

[35] J. A. Langholm, I. Andersen og E. F. Dalen, "Sluttrapport", *NTNU*, Mai 2021. [Vedlegg].
Hentet fra:
Vedlegg i innlevering av oppgave.