Høgskulen på Vestlandet

BACHELOROPPGAVE

Brukertilpasningssystem ved bruk av maskinlæring

User Personalization System using Machine Learning

Bonn Benyapha Aarsheim, Heine Fjeldberg og Kristoffer Helland Perminow

Dataingeniør/Informasjonsteknologi

Fakultet for ingeniør – og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Veileder: Sven-Olai Høyland

Innleveringsdato: 02.06.2020

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel:	*Dato:*
Brukertilpasningssystem ved bruk av maskinlæring User Personalization System using Machine Learning	02.06.2020
Forfatter(e):	*Antall sider u/vedlegg:* 41
Bonn Aarsheim, Heine Fjeldberg og Kristoffer Helland Perminow	*Antall sider vedlegg:* 3
Studieretning: Dataingeniør/Informasjonsteknologi	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Sven-Olai Høyland	*Gradering:* Ingen
Merknader:	

Oppdragsgiver:	*Oppdragsgivers referanse:*
Wide Assessment AS	Andreas Hammerbeck
Oppdragsgivers kontaktperson:	*Telefon:*
Andreas Hammerbeck	+47 97661466

Sammendrag:

Rapporten omhandler utvikling av et anbefalingssystem for Wide Assessment AS (WA). Bedriften har behov for et system som kan forbedre prosessen for å opprette en brukerprofil på deres plattform. Systemet skal kunne anbefale skills til en bruker under registrering. Målet er derfor å utvikle en maskinlæringsmodell som støtter dette.

Maskinlæringsmodellene som ble fremstilt for å gjennomføre anbefalingene er utviklet ved hjelp av Pythonbibliotekene Spotlight og Keras. Disse bibliotekene bruker nevrale nettverk til å trene frem den ferdige modellen.

Resultatene av maskinlæringsmodellene er gitt ved evalueringsmetoden *precision at k,* hvor *k* er antall prediksjoner totalt. Denne metoden returnerer en verdi mellom 0 og 1 som beskriver nøyaktighet av prediksjonen. Desto nærmere verdien er 1, jo mer nøyaktig er resultatet. Resultatene for modellen utviklet med Spotlight var 0.36 ved k=5, og 0,61 ved k=5 for Keras-modellen. Ved prediksjoner på fem skills, var 36% korrekt i gjennomsnitt for Spotlight og 61% for Keras.

WA ser verdi i modellen dersom nøyaktigheten er over 0.20. Basert på resultatene kan en si at det er blitt utviklet to fungerende maskinlæringsmodeller som predikerer skills med tilfredsstillende sannsynlighet for bedriften. Videre testing er nødvendig for å finne ut hvilken av de to modellene som egner seg best på WA sin plattform.

Stikkord:

Maskinlæring	Python	Anbefalingssystem
Nevralt nettverk	Jupyter Notebook	

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitskap

Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen

Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Denne rapporten omhandler teori og utførelse av bacheloroppgaven *Brukertilpasningssystem ved bruk av maskinlæring*. Oppgaven er gjennomført for oppdragsgiver Wide Assessment As. Bacheloroppgave er fullført våren 2020 ved Høgskulen på Vestlandet (HVL).

Først vil vi gjerne takke Wide Assessment for muligheten til å jobbe med et spennende prosjekt, samt å bli inkludert i et vennlig miljø. Spesielt takk til vår kontaktperson Andreas Hammerbeck som har stilt sin kompetanse til rådighet gjennom hele prosjektperioden. Vi vil også takke vår veileder fra HVL, Sven-Olai Høyland som har bidratt med viktige tilbakemeldinger på rapport, og fått oss til å føle oss sett og hørt fra start til slutt.

Av Bonn Benyapha Aarsheim, Heine Fjeldberg og Kristoffer Helland Perminow.

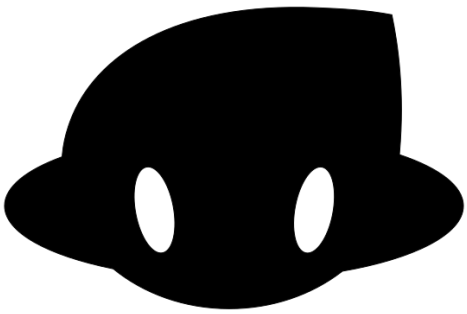
Innholdsliste

1 INNLEDNING

Forbrukere av internettjenester i dag har høye forventninger til brukeropplevelse. Registrering av profil er intet unntak. Det skal være lett, forståelig og kunne gjøres så fort som mulig. Et godt utviklet brukergrensesnitt er dermed viktig. Rapporten beskriver en bacheloroppgave som ble gjennomført for å utvikle en maskinlæringsmodell som oppnår dette for bedriften Wide Assessment (WA). Gjennom denne rapporten har gruppen valgt å bruke engelske ord og begrep der det ikke finnes gode oversettelser. Forkortelser og tekniske begrep blir utdypet i en liste under vedlegg.

1.1 Kontekst

WA ble stiftet i 2016 og er et It-selskap som jobber med å utvikle en plattform for bedrifter og arbeidssøkende. Denne plattformen skal i større grad kunne tilby bedrifter muligheten til å finne de rette ansatte, og tilby arbeidssøkende en oversikt over hvilke jobbmuligheter de har. Dette skal være en enkel prosess for både bedrift og arbeidssøker. Det som gjør at WA skiller seg ut fra andre rekrutteringsplattformer er at de rangerer kandidater etter ferdigheter, heretter omtalt som skills. Dette kaller de en skill-basert CV. Denne fokuserer ikke kun på arbeidserfaring og utdanning, men også på ferdigheter en bruker har innen sitt felt. Per dags dato har bedriften størst fokus på IT-bransjen. Plattformen finnes på nettsiden app.wa.works.



Figur 1.1 Wide Assesment logo

WA ønsker at brukerregistering på plattformen deres skal gå raskt og enkelt. Bedriften har derfor laget en bacheloroppgave som omhandler å forenkle brukerregistreringen med en maskinlæringsmodell.

1.2 Motivasjon og mål

Under registrering som kandidat på WA sin plattform må en gjennom flere steg. Et av disse stegene er å registrere erfaringer med programmeringsspråk, rammeverk og prinsipper. Dette kalles skills og velges fra en liste. Noen eksempler på skills er Java, C#, Scrum og HTML, som er kjent innen IT-bransjen.

Registreringsprosessen kan ta tid ettersom listen er lang og omfattende. WA ønsker at denne prosessen skal være enklere å gjennomføre. For å øke plattformens troverdighet, oppfordrer de brukere til å fylle ut profiler så mye som mulig. Dette vil føre til at plattformen oppfattes som et mer seriøst produkt. Bacheloroppgaven har som mål å bidra til denne troverdigheten ved å gjøre registreringsprosessen litt mer tilrettelagt for brukere.

Målet for dette prosjektet er å utvikle en ferdighetsanbefaler for WA sin plattform. For å oppnå dette vil det først og fremst bli utviklet en maskinlæringsmodell. Modellen skal kunne anbefale hvilke ferdigheter en ny bruker kan velge. Anbefalingen vil basere seg på ferdigheter som bruker selv har valgt, og ferdigheter valgt av eksisterende brukere. Prosjektet vil også utarbeide andre alternative løsninger som kan håndtere ferdighetsanbefalingen. På bakgrunn av dette er det blitt utviklet ett hovedmål, samt to delmål som favner beskrivelsen av prosjektet.

Hovedmål:

- å utvikle en ferdighetsanbefaler for arbeidssøkende, hovedsakelig innen IT

Delmål:

- Utvikle en maskinlæringsmodell som kan håndtere ferdighetsanbefalingen
- Utvikle en mer statistisk algoritme som kan håndtere ferdighetsanbefalingen

1.3 Avgrensninger

For å oppnå målene i denne oppgaven er det nødvendig å avgrense hvilken metode og teknologi som skal benyttes. WA ønsker at oppgaven skal gjennomføres ved bruk av maskinlæring. Oppgaven vil derfor avgrenses til å hovedsakelig benytte maskinlæring som metode, men andre relevante metoder vil bli vurdert.

For å utvikle maskinlæringsmodellen vil oppgaven avgrenses til å bruke profiler laget av brukere på WA sin plattform. Profilene vil bli brukt fordi det er den best egnede data som er tilgjengelig. Hvor godt disse profilene er blitt fylt ut av brukerne, vil i seg selv kunne avgrense oppgaven. Oppgaven vil se på skills som brukerne har registrert. På WA sin plattform kan brukerne velge

mellom ulike skills og ulike ferdighetsnivå på skills. Denne oppgaven vil avgrenses til å kun se på om en skill er valgt eller ikke. For å ytterlige forbedre resultatene til modellen vil det kun bli sett på ferdigheter som flere enn ti brukere har registrert.

1.4 Ressurser

Oppgaven vil i stor grad utføres i samarbeid med oppdragsgiver, WA. Kontaktperson fra oppdragsgiver, CTO Andreas Hammerbeck, vil være en viktig ressurs for gruppen. Hovedsakelig for at gruppen skal få tilgang til bedriftens data, men også fordi han har god kjennskap til WA sin plattform. Hammerbeck opprettet og administrerer en felles kodebase på Github og har gitt gruppen tilgang til bedriftens database. Bedriften tilbyr også kontorplass.

Offisiell kodedokumentasjon vil være relevant å se på og regnes derfor også som en ressurs. Det vil bidra til bedre forståelse underveis i utviklingsprosessen. Det vil bli tatt i bruk kommunikajsonsprogramvare som Slack og Skype. Særlig viktig blir Skype siden flere møter vil skje over internett.

1.5 Oppbygging av rapporten

Denne rapporten inneholder åtte kapitler som omtaler prosessen ved å utvikle maskinlæringsmodellen. Kapittel 1 har gitt oversikt over prosjektet og en beskrivelse av prosjektets mål. Videre går kapittelet inn på avgrensninger for prosjektet og ressurser som har vært tilgjengelig i prosjektperioden.

Kapittel 2 gir en dypere innføring i prosjektet og hva løsningen skal inneholde. Kapittelet inneholder også informasjon om produkteier, tidligere arbeid og hvordan løsningen var tenkt, samt kravspesifikasjoner før prosjektstart.

Design av prosjektet beskrives i kapittel 3. Kapittelet omhandler hvordan prosjektet vil bli utført med tilhørende informasjon om valgt verktøy, utviklingsmetodikk, risikovurdering, evalueringsplan, og diskusjon rundt valgt løsning. Videre blir designet for prosjektet detaljert beskrevet i kapittel 4. Dette innebærer en forklaring rundt fremgangsmåte og løsning.

Kapittel 5 tar for seg evalueringsmetoder som er valgt for prosjektet og resultatene av disse. Deretter beskrives prosjektets resultat i kapittel 6. Avslutningsvis diskuteres gjennomføring av prosjektet og konsekvenser i kapittel 7, og konklusjon, samt videre arbeid i kapittel 8.

Til slutt finnes referanseliste og vedlegg.

2 PROSJEKTBESKRIVELSE

I dette kapittelet beskrives bakgrunnen for prosjektet og relevant teoribakgrunn. I avsnittene under blir prosjekteier, tidligere arbeid med prosjektet, samt tenkt løsning og kravspesifikasjoner forklart.

2.1 Praktisk bakgrunn

2.1.1 Prosjekteier

Bedriften WA har alle rettigheter knyttet til produktløsningen. Dette gjelder GitHub *repository* og alt av kode. Gruppen har ingen eierskap til produktet, men har lov til å omtale og vise frem grafer og kode i bacheloroppgaven.

2.1.2 Tidligere arbeid

Oppdragsgiver har tidligere vurdert mulighetene rundt bruk av maskinlæring for en ferdighetsbehandler. CTO Hammerbeck har utført et mindre forsøksprosjekt, som har veiledet dette prosjektet, med eksempelkode og bruk av biblioteket Spotlight. Forsøksprosjektet består for det meste av metoder Hammerbeck har utviklet for oppretting og trening av en maskinlæringsmodell.

Dette prosjektet inneholder lite av preprossesering og utforsking av data, men gruppen vil ta inspirasjon fra det som er gjort. Gruppen har planlagt en litt annen fremgangsmåte, men vil bruke det tidligere arbeidet som veiledende hjelpemiddel til å fremstille en ferdig modell.

2.1.3 Innledende krav

Kravene for denne oppgaven er å produsere en fungerende metode som skal foreslå skills til en bruker. Metoden vil omhandle registrering av bruker, ved bruk av maskinlæring. Det er forventet at denne løsningen skal fungere slik at den oppleves som nyttig og utgjør en forskjell for brukerne av plattformen.

2.1.4 Innledende løsningsidé

Forslaget om å løse ferdighetsanbefalingen ved hjelp av maskinlæring er utarbeidet av WA. Som nevnt i kapittel 1 er idéen bygget på at bedriften ønsker å gjøre registreringsprosessen på plattformen så enkel som mulig. WA sitt forslag til fremgangsmåte er å bruke Python til å prosessere data, samt utvikling av modell. Dersom tiden strekker til kan modellen videre importeres til TensorFlowJS, og kjøres direkte i *frontend* prosjektet til bedriften.

Løsningen er derfor en maskinlæringsmodell som skal ta inn en bruker-ID eller en skill-ID, som er unike nummer for å identifisere bruker og skill. Modellen skal ut ifra dette klare å predikere nye og relevante skills for en spesifikk bruker.

Prediksjon baserer seg på trening på eksisterende data. Det vil si at maskinlæringsalgoritmen vil se på forekomster av skills som ofte er valgt sammen basert på data fra eksisterende brukere. Algoritmen vil dermed gjøre en prediksjon under registrering. Eksempelvis om en velger Java som en skill under registrering, kan metoden foreslå JavaScript basert på hva den har lært fra de eksisterende brukerne.

2.2 Teoretisk bakgrunn

2.2.1 Maskinlæring

Maskinlæring er vitenskapen og kunsten å programmere datamaskiner til å lære fra data. En mer teknisk beskrivelse er at en datamaskin sies å lære fra erfaring E i forhold til en oppgave T og et utførelsesmål P, hvis utførelsen på T, som målt av P, forbedres med erfaring E (Géron, 2019).

Et vanlig eksempel på et maskinlæringsprogram er et søppelpostfilter. Dersom brukeren av en e-post konto sletter søppelpost og beholder ønsket e-post, kan programmet lære å skille mellom disse. Det systemet bruker til å lære kalles for et treningssett. Hvert treningseksempel blir kalt en treningsinstans. I dette tilfellet er oppgaven T å flagge søppelpost fra nye e-poster, erfaringen E er treningsdata, og utførelsesmålet P må defineres. For eksempel kan en bruke andel av e-poster som er klassifisert riktig. Dette bestemte utførelsesmålet blir kalt *accuracy*, og blir ofte brukt i klassifiseringsoppgaver (Géron, 2019).

Maskinlæring kan deles inn i tre kategorier: veiledet læring, ikke-veiledet læring og forsterket læring (Brownlee, J., 2019).

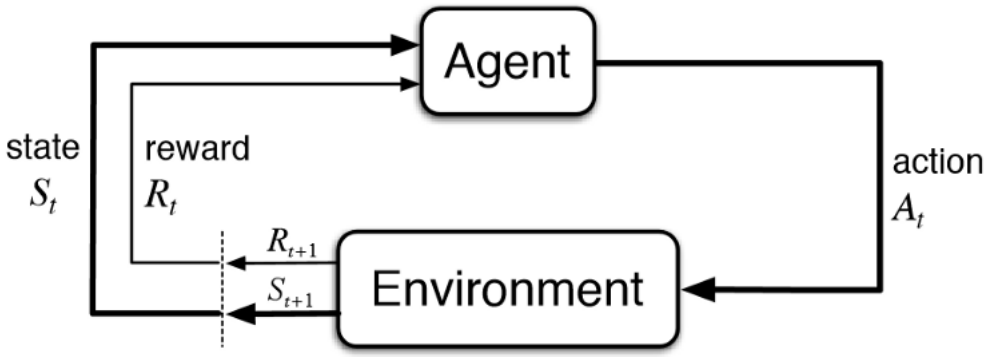
Veiledet læring er når en har en *input*-variabel (x) og en *output*-variabel (y) hvor en algoritme blir brukt for å lære en kartleggingsfunksjon fra *input* til *output*. Målet er at algoritmen skal kunne tilnærme kartleggingsfunksjonen best mulig. Dette vil da gjøre at algoritmen skal kunne predikere *output*-variabel (y) når den får inn en ny *input*-variabel (x) (Brownlee, J., 2019).

Det kalles veiledet læring fordi prosessen hvor en algoritme lærer av å trene på et datasett kan anses som en lærer som veileder læringsprosessen. Ønsket resultat er kjent og algoritmen gjøre prediksjoner iterativt på treningsdata og blir sådan irettesatt av læreren. Læringen stopper når algoritmen oppnår et akseptabelt utførelsesnivå (Brownlee, J., 2019).

Ikke-veiledet læring er når en kun har *input*-variabel (x) og ikke tilhørende *output*-variabler. Målet her er å modulere den underliggende strukturen eller distribusjonen i data for å lære mer. Det kalles ikke-veiledet læring fordi i motsetning til veiledet, så finnes det ikke et kjent resultat og det finnes ingen lærer. Algoritmens funksjon er å analysere data og presentere interessante strukturer som kan være nyttig (Brownlee, J., 2019).

Oppsummert så kan det sies at veiledet læring er når data er merket og algoritmen lærer å predikere *output*- fra *inputdata*. Ikke-veiledet læring er når data ikke er merket og algoritmen lærer å tolke struktur fra *inputdata*.

Forsterket læring er en læringsmetode hvor en agent samhandler med et nytt miljø gjennom å ta avgjørelser, og oppnår med dette det som kan kalles feil eller belønning. Forsterket læring handler om å utføre passende handling for å maksimere belønning i en bestemt situasjon (Perera, S., 2019).



Figur 2.1 Illusttrasjon av forsterket læring (Perera, 2019)

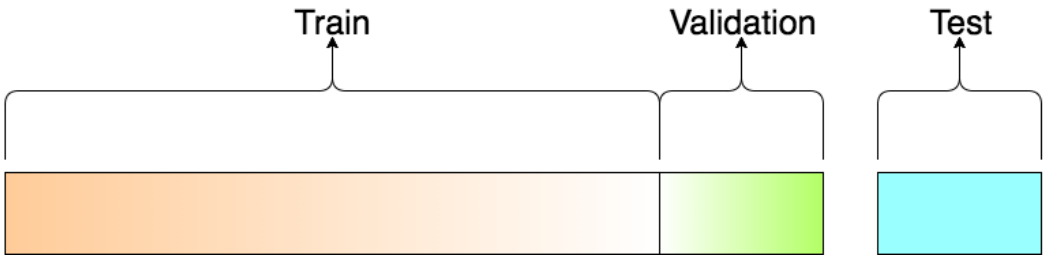
Kort oppsummert så kan forsterket læring sies å fungere på følgende måte: Observasjon av miljøet. Avgjør hvilke handlinger som skal utføres basert på en strategi, utfør handling for å motta feil eller belønning. Gjennom resultatet fra disse handlingene vil strategien forbedres. Dette vil gjøres iterativt helt til strategien anses som optimal. I figur 2.1 kan en se illustrasjon av forsterket læring.

Trening

For å oppnå en maskinlæringsmodell er en nødt til å lære den opp, dette kalles å trene. For å trene en modell er en nødt til å ha tilgang på data (Tideman og Ester, 2019). Vanligvis deles datasettet inn et treningssett og et testsett. Modellen blir trent opp på treningssettet, og testet på testsettet. Data fra testsettet er data som modellen ikke har sett før, og vil avsløre hvorvidt den har lært det den skal (Tideman og Ester, 2019). Dette er viktig for å unngå *overfitting,* som er at modellen tilpasser seg data den trener på for godt (Brownlee, 2016). I tillegg vil dette være avgjørende for evaluering av modellen i sluttfasen.

Noen ganger kan det være nyttig å evaluere modellen mens den blir laget. Dette er for å kunne finne de beste parameterne for modellen. Derfor opprettes det tredje datasettet, som er for validering. Denne splittes fra treningssettet for å holde testdata ukjent for modellen (Jordan, J. 2017).

Figur 2.2 viser en typisk deling, som er 60% for trening, 20% for validering og resterende 20% for testing (Jordan, J. 2017).



Figur 2.2 Deling av dataset(Shah, 2017)

2.2.2 Samarbeidsfiltrering for datasett med implisitt tilbakemelding

En vanlig oppgave til et anbefalingssystem er å forbedre brukeropplevelse gjennom personlige anbefalinger. De mest brukte metodene innen anbefalingssystemer er innholdsbasert og samarbeidsfiltrering (Hu, Koren og Volinsky, 2008).

Innholdsbasert tilnærming går ut på å lage en profil for hvert produkt i samlingen. Eksempelvis et system som anbefaler film for en bruker. Produktet kan bestå av popularitet, hvilke skuespillere som er medvirkende, sjanger og lignende. Resulterende profiler vil da hjelpe programmet å knytte sammen brukere til produkt. Dette er selvsagt avhengig av at data er godt utfylt og mulig å samle inn (Hu, Koren og Volinsky, 2008).

Et annet alternativ er samarbeidsfiltrering og baserer seg på brukerinteraksjoner uten at det kreves eksplisitte profiler. Metoden analyserer forhold mellom bruker og produkt. Anbefalinger baseres dermed på eksisterende historie fra likesinnede brukere. Eksempelvis en metode som anbefaler bøker. En bruker som kjøper en bok kan få anbefalt andre bøker basert på andre brukere som har kjøpt samme bok sin kjøpshistorie. Dette regnes som implisitt tilbakemelding (Hu, Koren og Volinsky, 2008).

3 DESIGN AV PROSJEKTET

I dette kapittelet beskrives prosjektets design. Dette innebærer valgt løsning, valg av verktøy, utviklingsmetodikk, prosjektplan og risikovurdering. For å kunne velge en løsning som er optimal for prosjekt, er det foretatt en vurdering av de ulike løsningsalternativene. Disse blir beskrevet og diskutert i punkt 3.1.

3.1 Forslag til løsning

Bruken av maskinlæring har fått mye oppmerksomhet i nyere tid, men måten det blir gjort på er ikke så forskjellig fra hva det var for en tid tilbake. Forskjellen ligger i at datakraften har vokst kraftig de siste 40 år som følger *Moor's law. Moor's law* sier at hvert andre år øker antallet transistorer per areal, som vil igjen øke kraften i en CPU (Roser og Richie, 2013).

Tilgangen på data har også økt kraftig bare de siste fem årene, og vil øke enda kraftigere videre (Reinsel, Gantz og Rydning, 2018). Dette er til fordel for maskinlæring, siden jo større menger data som er tilgjengelig, desto bedre mulighet er det for å lage gode maskinlæringsmodeller. I dag er det vanlig å bruke maskinlæring, noe som kan være nyttig i mange situasjoner. Det er derfor viktig at en også ser på hvilke alternative metoder en kan benytte for å oppnå det beste og mest effektive resultatet.

Data hentet fra WA inneholder ingen eksplisitte verdier. Dermed blir det naturlig å vurdere modeller som forholder seg til implisitt data. Det vil si data som ikke baserer seg på forskjellige parameter som kan ha en innvirkning på hvordan algoritmen velger å vekte sine anbefalinger.

3.1.1 Alternativ 1 – Keras og Tensorflow

Python-bibliotekene Keras og TensorFlow har metoder som er ment for implisitte datasett. Derfor er det naturlig å undersøke disse bibliotekene videre.

3.1.2 Alternativ 2 - Spotlight

Spotlight er et Pythonbibliotek som er spesifikt rettet mot anbefalingsmodeller (Spotlight, 2017). WA har allerede, som nevnt i punkt 2.1.2 i rapporten gjort et forsøksprosjekt ved bruk av Spotlight. Basert på dette vil det også være naturlig å se på dette biblioteket som et alternativ.

3.1.3 Alternativ 3- Algoritme

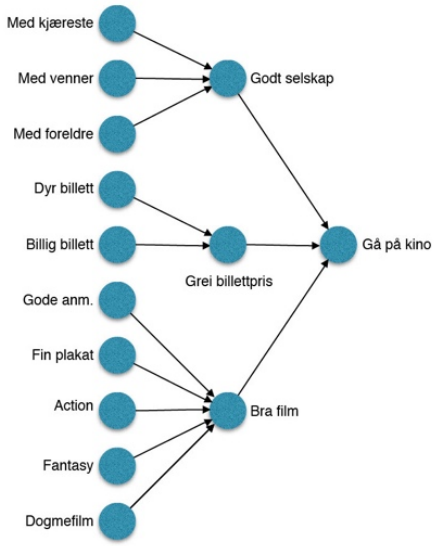
Det finnes også alternativer til maskinlæring som kan sørge for anbefaling av ferdigheter på WA sin plattform. En algoritme som ser på antall ferdigheter valgt av de forskjellige brukerne av plattformen, og anbefaler de som er valgt flest ganger kan være et alternativ. En algoritme er en instruks på hvordan en oppgave skal løses (Hovde og Grønmo, 2020). Dette vil sannsynligvis resultere i høy gjennomsnittlig treffsikkerhet, siden en vil gjette på skills som passer best for de fleste brukere. Det som kan være negativt med en slik algoritme er at den vil ha store avvik når det gjelder brukere med mindre normale sammensetninger av ferdigheter.

Det er også mulig å bruke en algoritme som ser på kombinasjoner av ferdigheter. For eksempel, hvilke ferdigheter som har blitt valgt oftest som par eller fler. En kan også se på hvor stor korrelasjon det er mellom de ulike ferdighetene, og kun anbefale de ferdighetene med høyest statistisk korrelasjon basert på andre brukere.

3.1.4 Alternativ 4 - Nevralt nettverk

En tidlig idé var bruken av nevralt nettverk. Kort fortalt er dette metoder og algoritmer som bruker inspirasjon fra biologisk nervevev. Dette er et sentralt begrep innen maskinlæring (Store norske leksikon, 2019).

Det fungerer ved bruk av flere *input* og *output*-lag. *Input*-lagene bruker vekting og bias for å gi et positivt eller negativt svar til *output*-laget. Mellom disse punktene kan en også implementere skjulte lag som gjør filtreringen mer nyansert. Et vanlig eksempel på bruk av nevralt nettverk er klassifisering av mennesker ut fra handlingsmønstre (Dvergsdal, 2019). Dette illustreres i figur 3.1.



Figur 3.1 Nevralt nettverk med skjult lag (Store norske leksikon, 2019)

Nevralt nettverk er svært relevant i forhold til denne oppgave, og er dermed en løsning som vil bli undersøkt videre.

3.1.5 Diskusjon av alternativene

Det viktigste å ta høyde for innledningsvis når en skal velge en løsning, er formatet på data. Løsningsalternativene som er nevnt over har alle mulighet til å produsere et ønsket resultat.

Keras og TensorFlow viser lovende tendenser ut ifra eksempler som har blitt undersøkt. Dette eksempelet omhandler et anbefalingssystem for bøker. Biblioteket har funksjonalitet som retter seg mot implisitte datasett, noe som er relevant for denne oppgaven.

Det samme gjelder Spotlight biblioteket. Det teller også positivt at dette biblioteket er spesifikt utviklet for anbefalingsmodeller, som er målet med denne oppgaven. Det tidligere prosjektet utviklet av WA tas også i betraktning da det vil fungere som en verdifull mal.

Idéen om å utvikle egne algoritmer var tenkt som et alternativ til maskinlæring. Dette var i stor grad for komparative årsaker, da sammenligning av løsninger er effektivt for å evaluere resultat. Denne løsning ansees som den mest krevende da løsningen må utvikles fra bunn av. Dette er da i motsetning til de andre alternativene som baserer seg på Pythonbibliotek.

Nevralt nettverk ble tidlig vurdert, noe en kan stille spørsmål til i ettertid, da både Keras, TensorFlow og Spotlight er biblioteker som bruker nevralt nettverk. Alternativet vil da bli å utvikle andre løsninger som ikke implementerer det nevrale nettverket på samme måte.

På bakgrunn av diskusjonen blir den valgte løsningen en kombinasjon av Spotlight og Keras/TensorFlow.

3.2 Valg av verktøy

For å gjennomføre oppgaven blir det tatt i bruk ulike verktøy. Verktøyene blir brukt i de forskjellige fasene av utviklingen, fra planlegging til ferdigstilling av produktet.

3.2.1 Pivotal Tracker

Pivotal Tracker er et planleggingsverktøy tilpasset *agile* utviklingsmetodikk. Den visualiserer prosjektets gjøremål med prioriteringer for de ulike gjøremålene. Alle i utviklingsteamet kan til enhver tid legge til gjøremål og oppdatere status for gjøremålene. Dette gir alle i teamet full oversikt over prosjektet (Pivotal Tracker, u.å.).

3.2.2 Jupyter Notebook

Jupyter Notebook er en gratis web-applikasjon som en kan bruke til å lage og dele dokumenter. Det kjøres på localhost og alt kan redigeres i nettleseren. Det er også enkelt å legge til dokumentasjon om fremgangsmåter på en oversiktlig måte (Jupyter, 2020).

3.2.3 Versjoneringskontroll

Versjoneringskontroll er et system som tar vare på endringer i en fil, så en kan gå tilbake til tidligere versjoner hvis feil skulle oppstå. I dette prosjektet er Github blitt tatt i bruk for å versjonskontrollere alle filene som er blitt brukt. Github er et versjoneringskontrollsystem som tar i bruk Git.

Git gjør det mulig for migrering av filer til å danne en fil som flere har jobbet på. En kan lage grener som hver av gruppemedlemmene kan jobbe på uten at endringene påvirker andre sine filer. Etter en er fornøyd med det en har gjort kan det sendes ut en *pull request* mot hovedgrenen (master), som da enten kan akseptere eller forkaste endringene som vil bli påført hovedgrenen (Github, Inc., 2020). Dette gjør samarbeid på samme prosjekt mulig uten å skape problemer for hverandre.

3.3 Programmeringsspråk, rammeverk og biblioteker

3.3.1 Python

Python er et programmeringsspråk som har et stort bruksområde. Det legger stor vekt på leselighet av kode. Det er dynamisk typet som vil si at en ikke trenger å definere type ved oppretting av variabler. Det støtter flere programmeringsparadigmer som for eksempel objektorientert og funksjonell programmering. Python er et programmeringsspråk som det er lett å komme i gang med uten å avhenge av noe ekstra biblioteker. Dette baseres på dets omfattende standard bibliotek (Wikipedia, 2020e).

Python er det mest brukte programmeringsspråket i maskinlæring (H. Bansal, 2019). Det benyttes blant annet i Jupyter Notebook, som er miljøet gruppen bruker for denne oppgaven.

3.3.2 Keras

Keras er et *open-source* nevralt nettverk bibliotek skrevet i Python. Det er designet for å muliggjøre rask eksperimentering med dype nevrale nettverk, og fokuserer på å være brukervennlig, modulært og omfattende. (Wikipedia, 2020b)

3.3.3 Pandas

Pandas er et programvarebibliotek skrevet for bruk i Python. Det blir brukt til datamanipulering og analysering av data. Det brukes hovedsakelig til maskinlæring i form av *dataframes.* Pandas tillater importering av data i format som for eksempel CSV, Excel og JSON (Wikipedia, 2020d).

3.3.4 Scikit-learn

Scikit-Learn også kjent som Sklearn, er et gratis maskinlæringsbibliotek for Python. Det inneholder forskjellige algoritmer for klassifisering, regresjon og gruppering og er designet for å samarbeide med Python sitt numeriske og vitenskapelige bibliotek NumPy og SciPy (Wikipedia, 2020g).

3.3.5 Tensorflow

TensorFlow er et gratis *open-source* programvarebibliotek for *dataflow* og *differentiable* programmering over et bredt spektrum av oppgaver. Det er et symbolsk matematikkbibliotek, og blir også brukt for maskinlæringsapplikasjoner som for eksempel nevrale nettverk. (Wikipedia, TensorFlow, 2020h)

3.3.6 Spotlight

Spotlight er et Python-bibliotek som er designet spesifikt for å utvikle anbefalingsmodeller. Biblioteket byr på omfattende funksjonalitet, blant annet en klasse som kalles *implicit factorization model*s. Denne klassen benytter seg av matrisefaktorisering med latente vektorer brukt for å representere både bruker og skill. Resultatet av denne klassen blir en prediksjon basert på skalarproduktet gitt av bruker-skill paringen.

Modellen trener gjennom *negativ sampling*. For hvert kjente bruker-skill par blir en eller flere skills tilfeldig utvalgt til å opptre som en negativ. Dermed kan algoritmen bruke det den har lært fra treningssettet ved å predikere på skjulte bruker-skill par i testsettet (Spotlight, 2017.).

3.4 Prosjektmetodikk

For at arbeidsflyten i et prosjekt skal være bra er det viktig å velge utviklingsmetode og sette opp gjøremål. Dette avsnittet omhandler utviklingsmetodikken brukt under prosjektet. Det inneholder også en oversikt over mål og tidsfrister, samt hvordan gruppen vurderer potensiell risiko knyttet til disse gjøremålene.

3.4.1 *Agile* metoder

Agile metoder er et fellesbegrep på arbeidsmetoder som er utviklet for håndtering av endringer i kravspesifikasjoner. Metodologien sentrerer seg rundt idéen om iterativ- og inkrementell utvikling. Det vi si at teamet planlegger en iterasjon om gangen og neste iterasjon tar utgangspunkt i den forrige. Prosessen blir da iterativ og tilpasser seg prosjektet over tid i stedet for at alt planlegges fra

start. Dette gjør det mulig for teamet å levere verdi raskere, med bedre kvalitet og forutsigbarhet, og større evne til å svare på endringer (Cprime, 2020).

3.4.2 Scrum

Scrum er en prosessramme basert på *agile* metoder og brukes vanligvis for utvikling av programvare. Det kan også bli brukt til andre typer prosjekter, og er derfor betraktet som en av de mest populære rammeverkene innenfor *agile* metoder. Scrum er strukturert for å hjelpe utviklingsteamet med å tilpasse seg brukerkrav og endringer underveis i prosjektet. Omprioritering og korte utgivelsessykluser er innebygd i prosessen slik at teamet alltid kan lære og forbedre arbeidet (Drumond, 2020).

Det finnes flere *artifacts* og faser i Scrum og noen av disse vil bli nærmere forklart senere. I tillegg finnes det også noen kritiske roller for hver person som er involvert i Scrum. Disse rollene er produkteier, utviklingsleder og utviklingsteam.

Produkt Backlog

Product backlog er en liste som til enhver tid viser hva som må gjøres. Denne listen holdes oppdatert av produkteier. *Product backlog* blir stadig revurdert og reorganisert av produkteier ut ifra hvordan kravene til produktet eller markedet endres (Drumond, 2020).

Sprint planlegging og *Sprint Backlog*

Arbeidet som skal gjennomføres i hver sprint planlegges i denne fasen. Dette gjennomføres i et møte med utviklingsleder og alle i utviklingsteamet. Her diskuteres oppgaver og mål for sprinten. En sprint er en tidsperiode der utviklingsteamet jobber sammen for å nå mål og fullføre et inkrement. To uker er vanlig sprintperiode. Under planleggingen utvikles det en *Sprint Backlog*, som er en liste over gjøremål for den spesifikke sprinten (Drumond, 2020).

Daglig Scrum

Dette blir også kalt *Stand Up*, som er et daglig møte på rundt 15 minutter der hver og en i teamet informere om hva som har blitt gjort og hva som skal gjøres denne dagen. Her kan en også ta opp utfordringer eller problemer en har møtt på (Drumond, 2020).

Sprint gjennomgang

På slutten av hver sprint samles teamet igjen for å demonstrere og vurdere det endelige produktet, som også kalles for Inkrement eller *Sprint Goal*. Produkteier kan her bestemme om inkrementet er godkjent eller ikke. I denne fasen blir det også vurdert om noe i Product Backlog-en skal endres. Endringene vil da påvirke neste sprintplanlegging (Drumond, 2020).

Sprint tilbakeblikk

Her samles teamet for å diskutere hva som har fungert og ikke i en sprint. Det kan være alt fra verktøy til relasjoner i teamet. Tanken bak dette er å skape en plattform for teamet der de kan fokusere på hva som har gått bra og hva som kan forbedres til neste gang (Drumond, 2020).

3.4.3 Kanban

Som Scrum, er Kanban utviklet for å hjelpe teamene med å samarbeide mer effektivt. Kanban legger stor vekt på visualisering av arbeidsoppgaver (Blueprint, 2020). Visualisering er et av hovedprinsippene i metoden, og gir enkel oversikt og kontroll på fremtidig, pågående og fullført utvikling. Begrensning og flyt er de andre hovedprinsippene i metoden. Det er viktig å begrense arbeid som er under utvikling for å sikre kvalitet og effektivitet. Videre er det viktig at oppgaven som blir startet på er en oppgave med høy prioritet, slik at flyten i prosjektet holdes jevnt. (Blueprint, 2020).

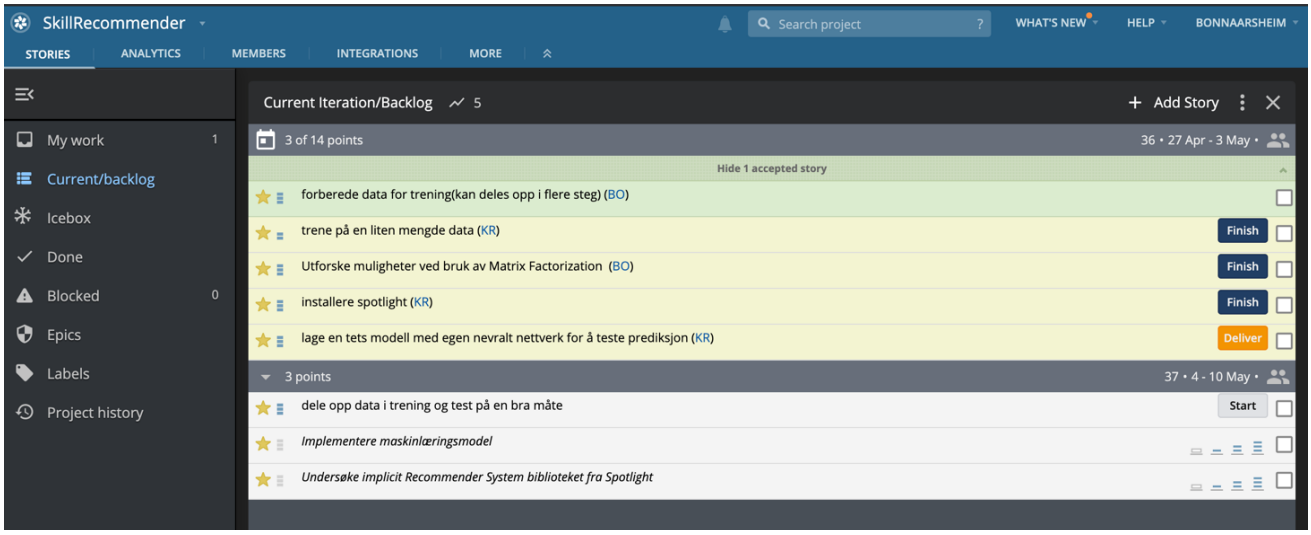
3.4.4 Utviklingsmetodikk

Dette prosjektet bruker en agil utviklingsmetodikk, som beskrevet i avsnitt 3.4.1, for å kunne møte kravene fra oppdragsgiver på best mulig måte. Det har blitt tatt mye inspirasjon fra den kjente metoden Scrum, som beskrevet i 3.4.2, og Kanban, som beskrevet i 3.4.3.

De viktigste faktorene som har blitt tatt i bruk fra Scrum er daglige møter, tett kommunikasjon med oppdragsgiver og faser som kan ligne litt på sprintene i Scrum. De daglige møtene varte fra 15 til 30 minutter. På møtene fortalte alle i gruppen litt om hva som var blitt gjort siden sist og hva som skulle gjøres den dagen. Her ble det også snakket litt om utfordringer eller problemer som kan ha oppstått. Gruppen passet også på å ha god kontakt med oppdragsgiver for å sikre at resultatet tilfredsstiller alle krav.

Ut ifra målene har det ikke vært naturlig å jobbe i sprint, men heller i ulike faser. En fase tar utgangspunkt i resultatet oppnådd i den forrige fasen. Fasene blir forklart nærmere i avsnitt 3.4.5. Oppgavene i de ulike fasene ble lagt inn i PivotalTracker, som beskrevet i 3.2.1. PivotalTracker har

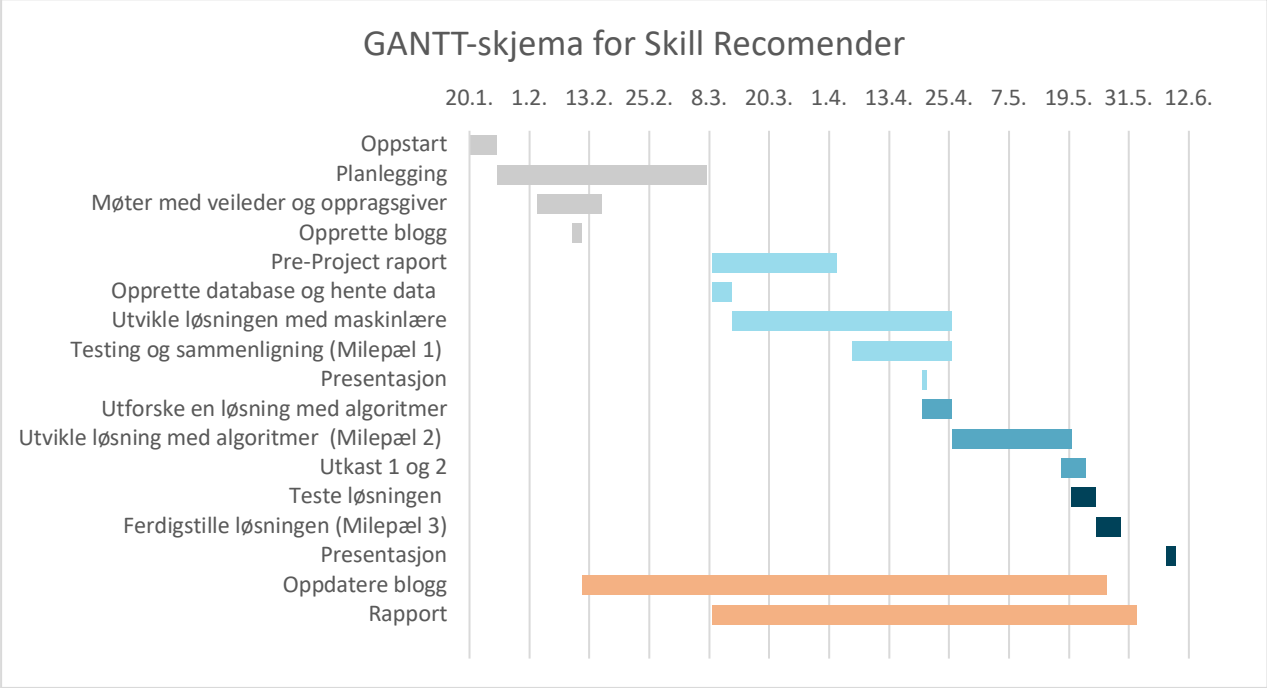
blitt benyttet som visualisering av gjøremål i prosjektet, der alle i gruppen kan legge til gjøremål med tilhørende prioritering. Figur 3.2 viser en skjermdump av noen gjøremål oppført i PivotalTracker som gruppen hadde under prosjektperioden.



Figur 3.2 Skjermdump av noen gjøremål i PivotalTracker under prosjektet

3.4.5 Prosjektplan

Et GANTT-skjema har blitt benyttet for å planlegge og kartlegge oppgavene for prosjektet.



Figur 3.3 GANTT-skjema

Som vist i figur 3.3 er prosjektet er delt opp i fire forskjellige faser, i tillegg til to oppgaver som så og si varer gjennom hele prosjektet.

Fase 1 – grå

I den første fasen er det størst fokus på oppstart og planlegging av prosjektet. Dette innebærer dialog med veileder og oppdragsgiver, samt oppsett av blogg og andre verktøy som skal brukes i prosjektet.

Fase 2 – lys blå

I fase to er målet å utvikle en optimal løsning med maskinlæring, som også er første milepæl. En stor del av denne fasen er også forprosjektrapporten som skal leveres 03.04.2020. I første omgang vil det være viktig å få hente ut data fra plattformen til WA, og å opprette en database som kan inneholde data. Gruppen må også utforske hvilket rammeverk og modeller som skal benyttes under utvikling av løsningen. Testing og sammenligning av resultat vil også utgjøre store deler av denne fasen.

Fase 3 – blå

Overgangen fra fase to til denne fasen til være en presentasjon av prosjektet og utforsking av mulighetene for å løse problemstillingen med algoritmer. Milepæl nr.2 er nådd når en løsning har blitt utviklet med algoritmer og testet. Her skal også første og eventuelt andre utkast av rapporten leveres.

Fase 4 – mørk blå

Siste fase innebærer forberedelser til EXPO, presentasjon og ferdigstilling av den endelige løsningen som også er milepæl nr. 3. I denne fasens skal også rapporten skrives ferdig og leveres 02.06.2020.

3.4.6 Risikovurdering

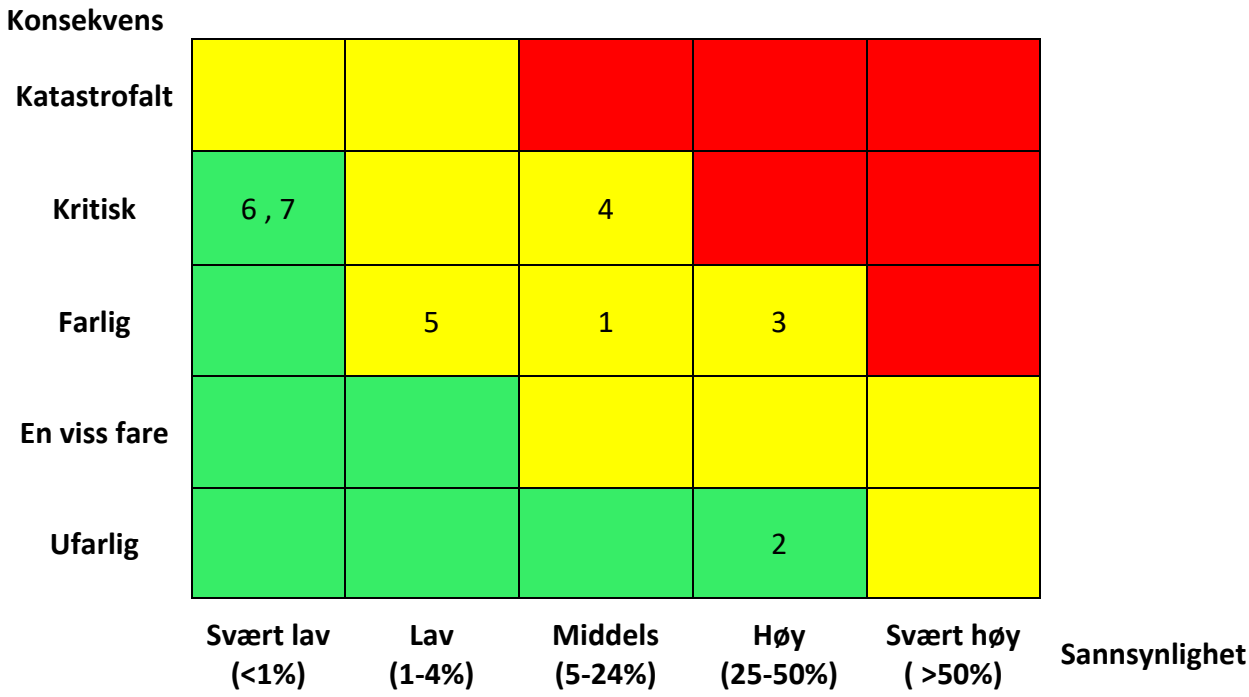
Koronaviruset, Covid-19, har påvirket samfunnet den siste perioden. Mye av utfordringene og risikoen gruppen står overfor vil derfor være knyttet til denne situasjonen. Dette gjenspeiles i både risikomatrisen, som illustrert i figur 3.4 og risikolisten i tabell 3.2. Konsekvensene forklares i tabell 3.1.

Konsekvens	
Ufarlig	Vil ikke påvirke prosjektet
En viss fare	Vil påvirke prosjektet i liten grad.
Farlig	Vil påvirke prosjektet, må gjøre tiltak for å unngå konsekvenser
Kritisk	Kritisk for prosjektet. Viktig å gjøre tiltak for å sikre prosjektet.
Katastrofalt	Prosjektet vil mislykkes dersom det ikke blir gjort noen tiltak.

Tabell 3.1 liste over konsekvenser

ID	Beskrivelse	Ansvar	Tiltak
1	Tom for tid	Alle	Jobbe jevnt og prioritere oppgaver
2	Sikkerhetsbrudd ved spørringer til database med persondata	Alle	Alltid spørre Andreas før spørringer eller endringer på databasen.
3	Endringer rundt dagens situasjon (Covid-19) som kan påvirke prosjektet	Alle	Holde seg oppdatert på dagens situasjon.
4	Sykdom i gruppen (spesielt dersom Covid-19 smittet)	Den det gjelder	Prøve å holde seg friske. Holde karantene og andre regler i henhold til dagens situasjon.
5	Problemer med hardware.	Den det gjelder	Låne eller skaffe seg ny
6	Ikke internettilgang og manglende kommunikasjon	Spørs hvilke abonnement en har.	Jobbe med ting som ikke krever internett. Kontakte hverandre via telefon
7	Strømbrudd over lengre tid	Kommunen	Tålmodighet

Tabell 3.2 Risikoliste



Figur 3.4 Risikomatrise

Realisering av prosjektet innenfor den gitte rammen

Gruppen anser det som høyst sannsynlig å kunne realisere prosjektet innenfor gitte rammer. Dersom prosjektet skulle feile, vil det mest sannsynlig være på grunn av tidsbegrensing eller alvorlig sykdom i store deler av gruppen.

Suksessfaktorer og trusler mot suksess

Dersom PC-er svikter underveis i prosjektet og gruppen ikke klarer å erstatte utstyret, vil dette kunne påvirke suksessfaktoren. Det er også mye usikkerhet rundt dagens situasjon med tanke på Koronaviruset. I verste fall kan dette være kritisk for prosjektet.

Mulige risikoelementer og sikkerhetsaspekter.

Det er viktig at datasikkerhet står i fokus ettersom gruppen henter data fra en database som inneholder personopplysninger. Det er derfor viktig å ha god dialog med Andreas Hammerbeck, når spørringer mot databasen eller lignende er nødvendig for prosjektet.

3.5 Evalueringsplan

For at en maskinlæringsmodell skal være i stand til å skape reelle verdier, må modellen kunne gi nøyaktige prediksjoner. Modellevaluering har som mål å estimere generaliseringsnøyaktigheten til en modell på ukjent data (Mutuvi, 2019). For å oppnå dette må data bli delt opp i trening, test valideringssett, som beskrevet i avsnitt 2.1.1.

I maskinlæring finnes det mange evalueringsberegninger som brukes til å måle kvaliteten på en modell. Valg av beregningstype er avhengig av hvilke oppgaver modellen skal utføre. For denne oppgaven vil for eksempel *precision and recall at K* være relevant. Dette beskrives nærmere i avsnitt 5.1

I tillegg til dette kan kunnskapen gruppen har om ulike ferdigheter i IT-bransjen komme til nytte under evalueringen. Gruppen kan se sammenhenger og vite hva som burde bli anbefalt dersom en bruker for eksempel har Java og C# som en skill. Dette er bare for å ha en pekepinn på om anbefalingene modellen gir er relevante eller ikke.

Et viktig aspekt å poengtere er at systemet ikke kan garantere korrekte prediksjoner. Oppgaven er å predikere et forslag til bruker. Disse forslagene er ikke basert på brukerpreferanse, men brukerferdigheter. Kombinasjoner av ferdigheter er ofte ulike, og det er dermed umulig å si at en skill har noen garanti knyttet opp til en annen skill.

Den beste måten å evaluere et eventuelt resultat på vil være gjennom brukertesting. Gruppen håper å kunne utvikle en løsning i god nok tid til å kunne undersøke muligheten for eventuelle brukertester.

4 DETALJERT DESIGN

I dette kapittelet blir designet for prosjektet beskrevet i detalj. Fremgangsmåte og produktets løsning vil bli forklart nærmere i de underliggende avsnittene.

4.1 Hente inn data

Data benyttet i dette prosjektet hentes fra WA sine servere gjennom API-kall. Den eksisterende koden i kodebasen til WA var ikke ferdigutviklet. Det ble da videreutviklet en kontroller som skulle sørge for å sende ut data som trengtes til dette prosjektet.

Plattformen WA er utviklet med et MVC designmønster. Dette designmønsteret deler programmet inn i tre deler, *model*: database som lagrer data, *view*: brukergrensesnitt som viser data og *controller* (kontroller) som flytter data mellom disse. En kontroller er en del av systemet som har forbindelse til *backend*. Kontrolleren gjør det mulig å hente ut data fra serveren til WA (Krasner & Pope, 1988 s2-3). Denne kontrolleren ble skrevet i C#. Dette programmeringsspråket er svært likt Java, noe gruppens medlemmer kjenner godt til fra studie.

For å hente ut data ble det laget kode for en GET-spørring til API-forespørselen. GET er en metode som spør databasen om å hente ut data (Ducket, 2011, s151). Denne metoden skal returnere all data om ferdighetene brukeren har valgt, som bruker-ID, skill-ID og navn på skill.

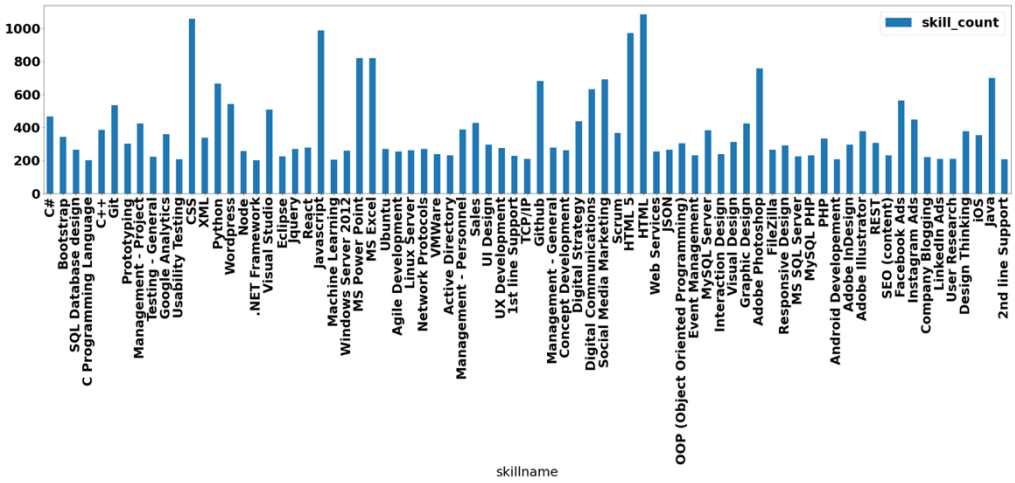
Formatet på data etter API-spørring er JSON. JSON er et format for overføring av data som er mye brukt i web-løsninger. Fordelen med å bruke dette formatet, er at det er både lesbart for mennesker og behandles enkelt i de fleste programmeringsspråk (Kristoffersen, 2016, s400).

4.2 Visualisering og utforskning av data

Visualisering av data er viktig for å få en oversikt over hvilken maskinlæringsmodell som passer. En korrelasjonsmatrise er en form for visualisering som er nyttig for å få en oversikt over hvilke skills som ofte er valgt sammen. Det er også nyttig med en oversikt over hvor mange ganger en enkelt skill er valgt. Dette kan bidra til evalueringen av maskinlæringsmodellens resultat. Pythonbiblioteket matplotlib gir en oversiktlig visualisering av data. Dette ble brukt til å lage tabeller og grafer av data.

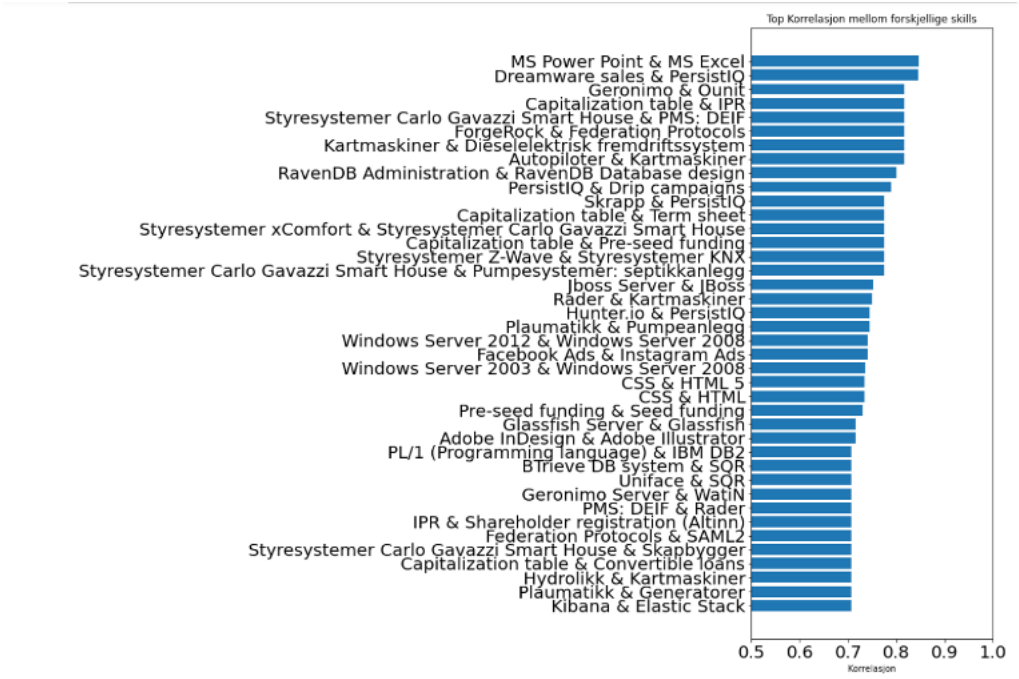
Eksempler på visualisering som ble gjennomført, er blant annet et søylediagram som visualiserer antall ganger en skill har blitt valgt av en bruker. For å forbedre modellen kan en se på denne

oversikten og luke ut skills som aldri, eller svært sjeldent ble valgt. I figur 4.1 vises et utdrag av skills som har blir valgt flest ganger av brukere.



Figur 4.1 Oversikt over mest populære skills

For å forstå sammenhengen mellom forskjellige skills, ble det laget en korrelasjonsmatrise. Antallet skills totalt er høyt, så korrelasjonsmatrisen ble omfattende. Et utdrag av skill-par med høy korrelasjon er vist i figur 4.2.



Figur 4.2 Oversikt, høy korrelasjon

I koden er det også prøvd å finne negativ korrelasjon, men dette var ikke å finne under –0.3. Den viktigste korrelasjonen er den positive for denne oppgaven. Som tidligere nevnt sier korrelasjon noe om hvor stor sannsynlighet det er for at en har en skill basert på valg av en annen. Denne informasjonen blir relevant å se på når det kommer til evaluering av modellen.

Når modellen er ferdigtrent kan en også se på anbefalte skills, og finne ut via korrelasjonsmatrisen hvor stor korrelasjon det er mellom dem. Dette vil oppgaven gå nærmere inn på i evalueringskapittelet.

4.3 Data preprosessering

Etter innhenting av data, er neste steg å dele den opp i relevante lister. Deler av denne inndelingen blir brukt som referansetabeller for oversikt, mens andre blir klargjort for maskinlæring.

Pythonbiblioteket Pandas gir en oversiktlig strukturering av data. Dette ble brukt til å utforske data, som la grunnlaget for hvordan en skulle tilpasse datastrukturen på den måten som kreves for maskinlæringsmodellen. Denne utviklingsdelen kalles preprosessering av data.

Den viktigste delen av preprosesseringen er tabellen som maskinlæringsmodellen skal trene på og basere sine prediksjoner på. Denne tabellen skal bestå av bruker-ID som rader og skills-ID som kolonner, og er en sammensetning av JSON-lister som tidligere ble oppdelt. Tabellen blir laget som Pandas DataFrame og innholdet skal være 0 eller 1 avhengig av om bruker er registrert med en skill eller ikke. Dette oppnås ved å iterere gjennom hver kolonne og sammenligner med listen som inneholder all informasjon om brukere og tilhørende skills. Figur 4.3 viser et utvalg av dette, men datasettet er for stort til å kunne vise hele.

user_ids	C#	Apache Server	MS BizTalk Server	MS IIS Server	Oracle Server	Azure	MS SharePoint Server	MS Intune	Dreamweaver	...	Business Model Canvas	Mockups	Networking	Business plan	Operations Research	Arduino
1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3500	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3501	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3502	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3503	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3504	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0

Figur 4.3 Utvalg Pandas DataFrame

For modellen er det optimalt at all data representeres av tall. Det var flere datapunkter innhentet fra WA som ble representert av en blanding av bokstaver og tall, for eksempel bruker-ID. Det ble da nødvendig å konvertere disse til tall. Skill-ID datapunktene var gitt ved unike tall, men det var stor spredning mellom verdiene for hver ID. Det ble også gjennomført en konvertering av ID på skills, som gjorde om alle skill-ID til en unik ID fra 0 til antall skills som var i datasettet. Siden skill-ID og bruker-ID var konvertert, ble det nødvendig å opprette en tabell som hold styr på hvilken original ID den nye ID tilhørte.

4.4 Maskinlæringsmodell

4.4.1 Implicit Factorization Models ved bruk av Spotlight

Denne metoden bruker matrisefaktorisering med latente vektorer som representerer både bruker-ID og skill-ID. Prediksjonen blir gitt gjennom skalarproduktet av bruker-skill paring.

Metoden tar inn parameter som har forskjellig innvirkning på resultatet. Disse parameterne kalles for hyperparametere og er vanlig å finjustere for å optimalisere modellen, dette kalles hyperparameter *tuning*.

```
def create_model():
     return ImplicitFactorizationModel(
                                    loss='adaptive_hinge',
                                    batch_size=720,
                                    embedding_dim=75,
                                    learning_rate=0.008,
                                    sparse=True,
                                    n_iter=50,
                                    optimizer_func=torch.optim.SparseAdam,
                                    num_negative_samples=6
                                    )
```

Figur 4.4 Oppretting av maskinlæringsmodell

Figur 4.4 viser de hyperparameterne som ble brukt for fremstilling av den ferdige modellen. De øvrige parameterne vil bli beskrevet nedenfor.

***Loss*-funksjon** er funksjonen som prøver å minimere feilen som blir gjort gjennom iterasjonene av trening (Goodfellow, Bengio og Courville, 2016, s82.). I denne modellen er det blitt tatt i bruk *adaptive_hinge* loss-funksjon. For å oppnå minst mulig *loss,* tar denne funksjonen ut et sett prediksjoner for implisitt negative elementer, og velger ut de som er høyest. Dette gjør så en får et utvalg av negative skills som er nærmest å overskride den implisitte rangeringen i mønsteret av brukerinteraksjonene (Kula, 2007).

Batch-size sier hvor stor del av datasettet som skal undersøkes av gangen (Kula, 2007).

Embedding-dim definerer antall *embedding*-dimensjoner for brukere og skills (Kula, 2007).

Learning-rate er en fininnstillingsparameter som blir brukt for å optimalisere algoritmen. Det vil si når det gjelder å avgjøre størrelsen på stegene i hver iterasjon av algoritmen som jobber mot en minimal *loss*-funksjon (Kula, 2007).

Sparse er en boolsk parameter som definerer om datasettet skal tolkes som sparsomt. Det vil si at algoritmen ikke mottar sterke nok signaler for å fininnstille sammenhengen. (Kula, 2007).

Optimizer_func er en funksjon som tar inn modulære parameter som første argument og returnerer en instans av en PyTorch *optmizer*. I dette tilfelle brukes Adam som *optmizer*. Adam er en tilpasningsdyktig optimaliseringsalgoritme for læringsrate (Kula, 2007).

4.4.2 *Low Rank Matrix Factorization* ved bruk av Keras

Denne modellen er basert på prinsippene rundt samarbeidsfiltrering. Data er formatert spesielt for bruk i denne modellen. Som figur 4.5 viser, er det tre kolonner i datasettet. En for bruker-ID, en for skill-ID og en for om skill er valgt eller ikke.

	userID	skillID	chosen
0	0	7	0
1	0	18	0
2	0	3	0
3	0	4	0
4	0	1	0
5	0	2	0
6	0	17	0
7	0	23	0
8	0	34	0
9	0	10	0
10	0	11	0

Figur 4.5 Datasett i Matrix Factorization

Hovedoppgaven er altså matrisefaktorisering, som vil si at en bryter datasettet opp i to mindre matriser med mindre dimensjoner. Disse matrisene kalles for *embeddings*. Som vist på figur 4.6 lages det *embeddings* for både brukere og skills, og definerer *input*-verdier modellen skal ta inn. Antall dimensjoner, kjent som latente faktorer i *embeddings*, er hyperparametre som brukes i denne implementeringen av samarbeidsfiltreringen.

```
user_input=Input(shape=(1,),name='user_input',dtype='int64')
user_embedding=Embedding(n_users,n_latent_factors,name='user_embedding')(user_input)
user_vec =Flatten(name='FlattenUsers')(user_embedding)

skill_input=Input(shape=(1,),name='skill_input',dtype='int64')
skill_embedding=Embedding(n_skills,n_latent_factors,name='skill_embedding')(skill_input)
skill_vec=Flatten(name='FlattenSkills')(skill_embedding)
```

```
sim = dot([user_vec,skill_vec],name='Simalarity-Dot-Product',axes=1)
model = keras.models.Model([user_input, skill_input], sim)
model.summary()
```

Figur 4.6 Lager maskinlæringsmodell med Matriz Factorization

Videre blir det beregnet skalarprodukt av begge matrisene ved bruk av et *merge layer,* som legger sammen modellens input-verdier. Skalarprodukt blir her brukt til å måle likheten på input-verdiene modellen får.

4.5 Trening

4.5.1 *Interactions* ved bruk av Spotlight

Treningen i Spotlight-modellen baserer på en innebygget metode som kalles *interactions*. *Interactions* er definert som en forbindelse mellom en bruker og en skill. En *interaction* oppstår kun når en bruker har en skill. Det er laget en tabell over alle *interactions* som forekommer i datasettet sett i figur 4.7.

user_ids	skill_idsML
1	1
1	230
1	231
1	259
1	314
...	...
3444	395
3444	403
3444	445
3444	449
3444	451

Figur 4.7 Tabell ovr alle interactions (komprimert)

User_ids er alle de forskjellige bruker-ID-ene og skill_idsML er de forskjellige skill-ID-ene. Disse bruker-ID-ene er laget for maskinlæringen og er ikke de faktiske bruker-ID-ene som brukes på plattformen til WA. Det er opprettet en tabell for å holde oversikt over alle bruker-ID-ene og skill-ID-ene slik at en kan konvertere tilbake for å si hvilke skill og bruker anbefalingen tilhører.

Interactions er en klasse som tilhører Spotlight biblioteket. Den tar inn bruker-ID og skill-ID og antall skills. Data blir behandlet i bruker-skill par som en observasjon. Dette blir da laget som et datasett som skal tas inn som *input* i funksjonen *fit* som utfører treningen på data, vist i figur 4.8.

```
def train_model(df):
    train_interactions = Interactions(user_ids,item_ids, num_items=(number_of_items))
    model = create_model()
    model.fit(train_interactions)
    torch.save(model, "./model")
    return model
```

Figur 4.8 Trener modell

4.5.2 Model fit ved bruk av Keras

Treningen i Keras skjer ved bruk av *model.fit*, som vist i figur 4.9. Denne tar inn antall epoker og *input*-verdier. *Input*-verdiene som brukes i denne modellen er to matriser. En matrise med bruker-ID og skill-ID, og en med verdier som viser om en skill er valgt eller ikke. Disse matrisene er fra treningsettet og måles opp mot data fra valideringsdata for å se hvor bra modellen trener.

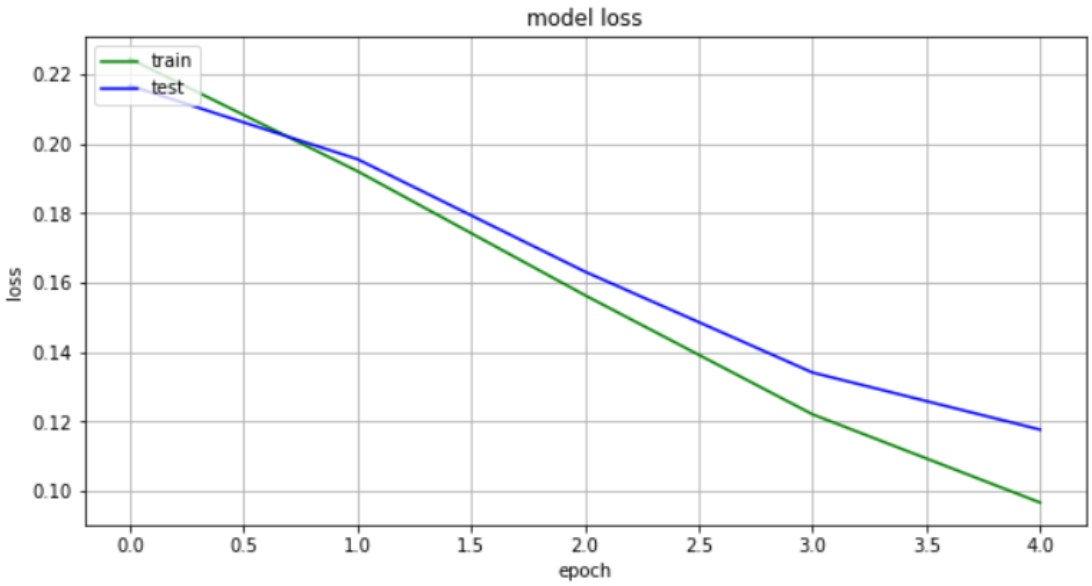
```
History = model.fit([x_train.userID, x_train.newSkillID],y_train,
                    batch_size=128,
                    epochs = 40,
                    validation_data=([x_val.userID, x_val.newSkillID], y_val))
```

Figur 4.9 Kode for trening av modell i Keras

Keras gir muligheten til å registrere og visualisere tilbakeregninger når en maskinlæringsmodell skal trenes. Tilbakeregningen som har blitt tatt i bruk i denne modellen registrere treningsmålinger for hver epoke. Dette inkluderer *loss*-verdien for treningssettet, samt valideringssettet. *Loss*-verdien som vist på figur 4.10, beskriver den predikerte *output*-verdien og den virkelige *output*-verdien. Grafen i figur 4.11 illustrer *loss*-verdien for hver epoke.

```
Train on 1533087 samples, validate on 511029 samples
Epoch 1/5
1533087/1533087 [==============================] - 50s 32us/step - loss: 0.2245 - accuracy: 0.9788 - val_loss: 0.2167 - val_acc
uracy: 0.9790
Epoch 2/5
1533087/1533087 [==============================] - 48s 31us/step - loss: 0.1921 - accuracy: 0.9788 - val_loss: 0.1956 - val_acc
uracy: 0.9790
Epoch 3/5
1533087/1533087 [==============================] - 48s 32us/step - loss: 0.1564 - accuracy: 0.9788 - val_loss: 0.1632 - val_acc
uracy: 0.9790
```

Figur 4.10 Tilbakeregningene under trening av modell



Figur 4.11 Visualisering av loss-verdiene

4.6 Prediksjon

4.6.1 Prediksjonsmetode med Spotlight

Spotlight har en innebygget prediksjonsmetode som ble brukt til å gjennomføre anbefalinger basert på *inputdata*. For å få den informasjonen som WA ønsket var det behov for å prosessere det som ble returnert fra prediksjonsmetoden. Spotlight sin prediksjonsmetode returnerer en liste med desimaltall for alle skills modellen har tilgjengelig. Dette tallet indikerer hvor relevant modellen mener at skillen er for brukeren, jo høyere tall desto mer relevant.

Først var det nødvendig å kartlegge hvilke skill-ID som tilhørte hvilke prediksjoner. Det ble gjort ved å lage en Python Dictionary (nøkkel verdipar) mellom skill og prediksjon. Deretter ble prediksjonene i Python Dictionary sortert etter høyest verdi. Dictionary-en inneholder etter dette en sortert liste med skills og prediksjoner for alle skills. Det interessante med denne listen er de øverste skillsene, siden de har fått høyest prediksjonsverdi. Det ble derfor laget en ny Python Dictionary med kun de øverste fem skills som ble kalt *top5Predictions*.

Etter *top5Predicitons* var fremstilt måtte skill-ID-ene omgjøres til skillnavn for å manuelt kunne se om prediksjonene var fornuftig. Dette ble gjort ved å iterere gjennom tabellen med original-ID og finne den som matchet skillnavn i den nye listen. Til slutt ble en DataFrame opprettet. Denne inneholder all informasjon om prediksjonen, skill-ID, skill-navn og original skill-ID fra databasen til WA.

4.6.2 Prediksjonsmetode med Keras og Tensorflow

Prediksjonsmetoden tar inn bruker-ID som argument. Metoden vil i utgangspunktet predikere alle skills som en mulig anbefaling. Det som skiller anbefalingene er at modellen gir hver skill en verdi som sier noe om relevant den er. Jo høyere verdien er, desto mer relevant er den for brukeren. Modellen sorterer dermed skills etter denne verdien og returnerer fem skills med høyest verdi som en anbefaling.

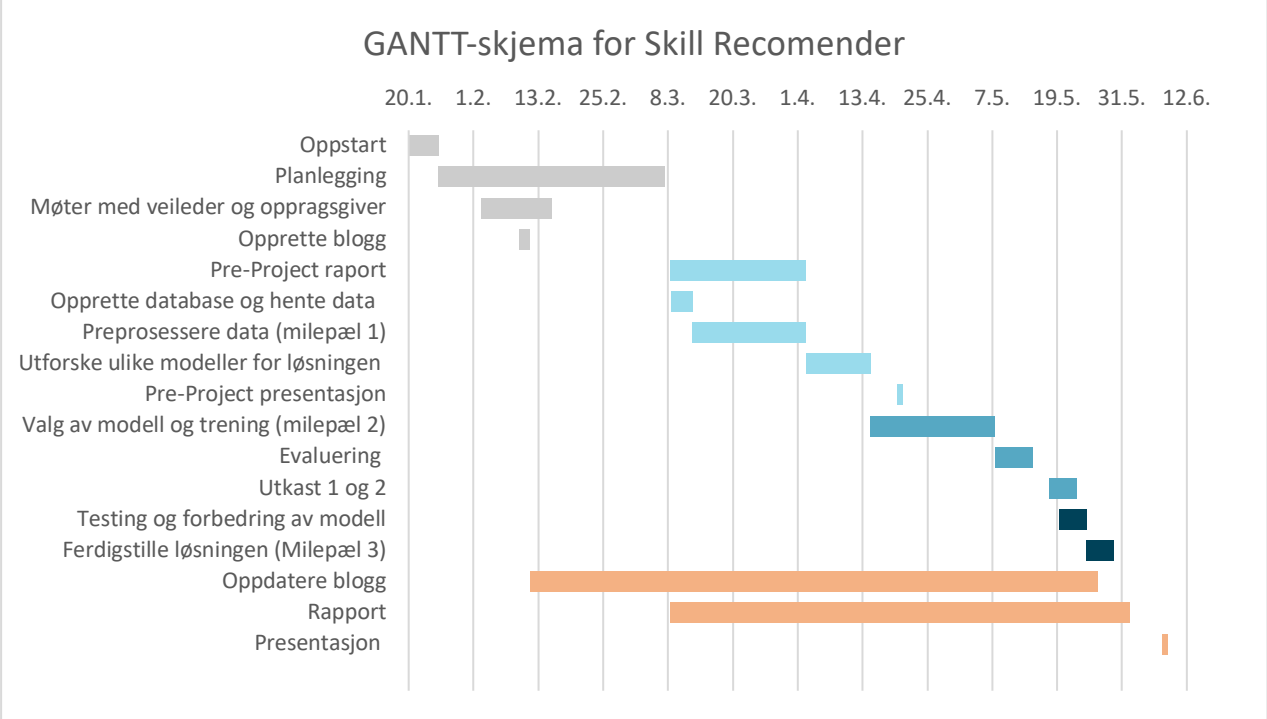
For evaluering av modellen blir det også regnet ut en gjennomsnittsverdi av hvor mange skills i prediksjonen som stemmer overens med allerede valgt skills. Denne verdien brukes i evaluering av modell, som forklart i avsnitt 5.2.2.

Anbefalingssystemet skal helst anbefale skills som er relevant, men også skills som ikke er valgt fra før. På bakgrunn av dette fjernes skills som prediksjonen har til felles med allerede valgte skills.

Metoden skriver så ut de fem mest relevante skills for brukeren, verdien som viser hvor relevante de er og en oversikt over skill-ID og navn.

4.7 Revidert prosjektplan

Under utviklingen ble det klart at milepæl og arbeidsoppgaver som ble satt ved prosjektstart måtte endres. Viser til punkt 3.4.5 som beskriver hvordan prosjektplanen var tenkt og satt opp fra start. Det er tatt høyde for at planen kunne endres dersom arbeidet ikke skulle gå som planlagt.



Figur 4.12 Revidert GANTT-skjema

Figur 4.12 viser revidert GANTT-skjema. De største endringene ligger i fase 2 og fase 3. I utgangspunktet skulle gruppen være ferdig med en løsning i maskinlæring og nå første milepæl i fase 2. Da dette ikke ble nådd, ble planen endret og fokuset satt over til å ferdigstille en løsning i maskinlæring.

Fokuset i fase 2 ble endret til å preprosessere data og utforske ulike maskinlæringsmodeller for løsningen. Fase 3 ble endret til å lage modell samt trene og evaluere modellen. Milepælene ble også justert etter endringene i de to fasene.

5 EVALUERING

Dette kapittelet tar for seg valgt evalueringsmetoder for prosjektet samt resultatene av disse.

5.1 Evalueringsmetode

5.1.1 Precision and Recall at K

Precision og *recall* er vanlige mål i binærklassifiseringsalgoritmer. Disse målene har blitt oversatt for å hjelpe til å evaluere anbefalingssystemer (Malaeb, 2017).

Precision og *recall* er altså binære måleenheter som bruker for å evaluere modeller med binær *output*. Dermed er det nødvendig å oversette de numeriske problemene. Vanligvis gjøres dette ved å sette en terskel. For eksempel om en har data som er basert på en rangering fra en til fem, så kan en sette terskelen til tre. Da vil data som er mindre enn tre bli regnet som irrelevant og over tre som relevant. I dette prosjektet er *output* allerede binært i form av valgt og ikke valgt skill (Malaeb, 2017).

I konteksten av anbefalingssystemer er det som regel mest interessant å anbefale topp N-antall punkt til bruker. Dermed gir det mer mening å regne ut *precision* og *recall* i de første N-antall punkt i stedet for alle punktene. Da blir forestillingen av *precision and recall at k,* hvor *k* er en definerbar tallverdi satt av bruker for å tilpasse topp N-antall anbefalingsforsøk (Malaeb, 2017).

Antall relevante forslag vil ofte være mer enn hva som er ønsket å presentere for bruker. Derfor er neste steg å skille mellom relevante punkt og anbefalte punkt. Relevante punkt er alle punkt i datasettet som allerede er kjent og merket som relevant. Relevante punkt er altså all data som tilfredsstiller betingelsene satt av terskelen, som nevnt over. Anbefalte punkt er alle punkt som er generert av anbefalingsalgoritmen. Denne algoritmen tar utgangspunkt i alle punkt som er relevant (Malaeb. 2017).

Definisjoner av både *precision* og *recall* kan formuleres på denne måten: *precision at k* er andelen av anbefalte punkt i topp-k datasettet som er relevant. Matematisk kan det forklares som *precision* at k = (antall anbefalte punkt i k som er relevant) / (antall anbefalte punkt i k) (Malaeb, 2017).

Recall at k er andelen av relevante punkt funnet i topp-k anbefalinger. Matematisk kan det forklares som *recall at k* = (antall anbefalte punkt i k som er relevant) / (totalt antall relevante punkt) (Malaeb, 2017).

5.1.2 Mean Reciprocal Rank

Reciprocal Rank (Gjensidig påvirkning eller RR) er et mål for innhenting av informasjon. RR regner ut det gjensidige av rangeringen for hvor det første relevante resultatet blir funnet i en liste (Craswell, 2009). I denne oppgaven vil en slik liste være en rangering av skills, og et relevant resultat vil være en relevant skill for brukeren. RR er 1 hvis et relevant resultat blir funnet ved første posisjon i listen, 0.5 ved andre posisjon i listen osv. Om en benytter seg av gjennomsnittet av RR over flere *cases,* blir dette kalt *Mean reciprocal rank (*MRR*)* (Craswell, 2009).

Tallet som produseres av evalueringsmetoden sier noe om hvor mange elementer som ble sett på før et relevant resultat ble funnet. For eksempel om evalueringen må gå igjennom tre skills før en relevant skill blir funnet, vil evalueringsresultatet være 1/3 = 0.33. For å regne tilbake fra MRR resultat, til antall skills gjennomgått, vil en i dette tilfelle kunne bruke 1/0.33 = 3. MRR brukes når det kun er ønskelig å se på ett relevant resultat (Craswell, 2009).

Fordelen ved bruk av MRR er at det er enkle utregninger og enkelt å tolke resultat. Det er en metode som passer bra for anbefalingssystemer. Metoden kan allikevel kritiserer for å fokusere for mye på enkelte og forholdsvis tilfeldige punkter, da det fortsatt kan være mange relevante punkter som oversees.

5.2 Evalueringsresultat

5.2.1 Recall and Precision at K – Spotlight

Spotlight modellen tar inn både *test* og treningssett for evaluering ved bruk av *precision and recall at k* metoden. Metoden tar altså inn selve modellen i tillegg til listene som er trent på gjennom *interactions* som forklart i avsnitt 4.4.1. Deretter vil metoden gå gjennom en løkke som vil evaluere flere ganger i henhold til k som vil inkrementere med én for hver iterasjon. Dette gjøres for å gi tilbake en detaljert oversikt over hvor effektiv prediksjonen er i forhold til k.

```
precision @ 3 train: 0.7552229497973184 recall @ 3 train: 0.410428220418999
precision @ 3 test: 0.3899802515331046 recall @ 3 test: 0.3749075242110555
precision @ 4 train: 0.6926255067040848 recall @ 4 train: 0.49168060463600044
precision @ 4 test: 0.37449329591518554 recall @ 4 test: 0.4554931888129079
precision @ 5 train: 0.6388525101340817 recall @ 5 train: 0.5586640442531292
precision @ 5 test: 0.3616463985032741 recall @ 5 test: 0.5240366508599477
```

Figur 5.1 Resultater av precision at k

Resultatet for evalueringen på spotlight modellen ved bruk av *precision at k*, var en score på 0.36, som vist i figur 5.1. Ved dette resultatet var k =5. Dette er *precision* på anbefalingen dersom det er

fem skills som anbefales. Dette betyr at når vi gir ut en prediksjon til brukeren, er det sannsynlig at nært to av fem skills er relevant for denne brukeren.

5.2.2 *Precision at K* – Tensorflow/Keras

Denne løsningen vurderer kun *precision at k.* I stedet for å se på flere resultat i henhold til en varierende verdi k, så er k definert som fem. Grunnlaget for dette er prediksjonsmetoden som gir ut fem anbefalinger. Denne metoden forklares nærmere i avsnitt 6.2.

Som nevnt i 5.1.1 er det ønskelig å finne ut hvor stor andel av prediksjonene som er relevant. Dette gjøres ved å gi evalueringsmetoden en bruker-ID liste som er ukjent for modellen, altså en liste av testdata. Evalueringsmetoden bruker så prediksjonsmetoden for å vurdere hvor mange anbefalinger som stemmer overens med det brukeren allerede har valgt.

```
results_precision_at_k = evaluate(testliste)
results_precision_at_k
```

```
100%|██████████████████████████████████████████████████████████████████████████| 3505/3505 [19:08<00:00,  3.05it/s]
0.6079315263908668
```

Figur 5.2 Precision at k resultat

Prediksjonsmetoden returnerer en gjennomsnittsverdi som gir en indikasjon på hvor bra anbefalingsalgoritmen fungerer på den enkelte brukeren. Evalueringsmetoden summerer disse verdiene for alle brukere i listen. Denne verdien utgjør da *precision* for modellen, som illustrert i figur 5.2

5.2.3 MRR - Spotlight

```
print(mrr_score(model, test_interactions).mean())
```

```
0.3170419560680051
```

Figur 5.3 MRR-verdi

Resultatet for evalueringen på spotlight modellen ved bruk av *mean reciprocal rank*, var en score på 0.317 (se figur 5.3). Som nevnt i beskrivelsen av MRR vil dette tallet vise til hvor mange skills evalueringsmetoden har gått igjennom, før en relevant skill ble funnet (Craswell, 2009). Basert på MRR resultatet vil en i gjennomsnitt måtte gå igjennom 3.154 skills før en relevant skill blir foreslått (1/0.317 = 3.154). Dette sier at modellen er mest verdifull dersom den gir ut mer enn tre skills ved anbefaling, for å sikre at en relevant skill kommer opp for brukeren.

6 RESULTATER

6.1 Resultat fra Spotlightmodellen

Precision at K scoren på test settet var relativt lav, og ble lavere jo større K var. Datasettet som benyttes er lite, og WA får daglig inn nye brukere som legger til skills. Derfor vil en kunne tenke at modellen blir mer og mer treffsikker, ettersom mer data blir tilgjengelig. Det er derfor viktig at WA trener og evaluerer modeller kontinuerlig fremover.

6.2 Resultat fra Keras/Tensorflow

Figur 6.1 viser et resultat fra prediksjonsmetoden. Maskinlæringsmodellen ser ut til å kjenne igjen sammenhenger på en god måte, da prediksjonsmetoden anbefaler skills som er svært relevante. For eksempel er det høy sannsynlighet for at brukeren har kunnskap innen GitHub når den har valgt C# og Java som skills.

```
see_user_skill_with_skillname(121)
predict_on_user(121)
```

```
C#
C Programming Language
MATLAB
CSS
Python
Wordpress
Javascript
Swift
HTML 5
HTML
Adobe Photoshop
Adobe Illustrator
Adobe Lightroom
Java
Xcode
user ID:  121
Top five recomended skills for this user:  [344, 446, 324, 460, 445]
[0.15972468 0.14643848 0.11737588 0.11604887 0.11549182]
         skillnames  skill ID  newSkillID
324       UI Design       250         324
344          Github       425         344
445   Visual Design        16         445
446  Graphic Design        17         446
460       MySQL PHP       196         460
```

Figur 6.1 Et resultat av prediksjonsmetoden

Evalueringsresultatet som vist på figur 5.2 er på 0.60. Dette resultatet er svært godt, som også gjenspeiler prediksjonsresultatene.

6.3 Drøfting av resultat

Målene som ble satt for denne oppgaven var å utvikle en ferdighetsanbefaler som har som mål å anbefale relevante skills for nye brukere. Oppnådd resultat er to maskinlæringsmodeller som innfrir dette målet. Begge tar inn brukere og modellen returnerer forslag som tolkes som relevante. Med anbefalingssystemer av denne typen, vil være umulig å oppnå definitive korrekte anbefalinger. Oppdragsgiver har derfor satt et mål der en av fem anbefalinger ønskes å være relevant (20 %). Dette målet er oppnådd, som underbygges i evalueringsmetodene brukt i dette prosjektet. Metodene er nærmere forklart i avsnitt 5.1.

7 DISKUSJON

7.1 Gjennomføring av prosjektet

Helt i starten av dette prosjektet ble det diskutert hvilke prinsipper og metoder som skulle benyttes i arbeidet. Det ble klart at en *agile* arbeidsmetode var det som passet gruppen best. Gjennom hele perioden har gruppen holdt daglig kontakt. Tanken var å sette realistiske delmål hele veien for å oppnå et godt resultat. Et annet viktig punkt i oppstarten var å sette opp et GANTT-skjema. Alle gruppens medlemmer var enig i viktigheten av å sette opp mål og tidsfrister som dette, for at arbeidet skulle flyte bra.

Python er det mest populære programmeringsspråket innen maskinlæring (Chand, 2019). Alle gruppens medlemmer har erfaring med Python fra studiet, og det var også språket som var foretrukket av WA. Valget av Python som programmeringsspråk ble også påvirket av dets omfattende standardbibliotek. På bakgrunn av dette ble Python et naturlig valg som programmeringsspråk.

Gjennom mye utforsking av flere metoder og biblioteker, valgte gruppen å benytte bibliotekene som virket best tilrettelagt for WA sine data. Det å i stor grad la data diktere valgene av metoder og biblioteker, har vist seg å være en effektiv tilnærming, og er en erfaring som gruppen vil ta med videre. På bakgrunn av dette ble bibliotekene Spotlight og Keras valgt, samt prinsippene som omhandler samarbeidsfiltrering for implisitte datasett. Dette var avgjørende for å oppnå resultatene som ble presentert i forrige kapittel.

En av de store utfordringene under utviklingsprosessen var å manipulere data til å passe de valgte metodene. Data tilgjengelig fra WA var kun implisitte verdier, som vil si at de var representert med kategoriske variabler som sier om en skill er valgt eller ikke. Det som var kjent, var hvor populær data var, altså hvor mange ganger hver enkelt skill ble valgt av eksisterende brukere. Dette medførte muligheten til å begrense hvilke skills som inngikk i anbefalingsalgoritmen, og dermed øke sannsynligheten for at anbefalingen var relevant.

Det har vært utfordrende å implementere evalueringsmetoder på en god måte. Grunnen til dette er at det finnes mange forskjellige måter å evaluere en maskinlæringsmodell på. På grunn av dette ble mye av arbeidskapasiteten brukt på å finne evalueringsmetoder som passet best til den modellen som ble framstilt. Evalueringsresultatene fra de valgte metodene, innfrir kravene gitt av WA.

Det er nødvendig å nevne Koronapandemien under reflektering av gjennomføringen. Bare tre dager etter oppstart av prosjektet ble karantenetiltakene innført (Statsministerens kontor og Helse- og

omsorgsdepartementet, 2020). Dette medførte at gruppen ble tvunget til å jobbe separat. Daglig kontakt med WA ble ansett som en viktig del av utviklingsprosessen. Det ble avholdt møter over Skype, men mangel på kontinuerlig tilbakemelding har vært utfordrende.

7.2 Konsekvenser

Den største utfordringen for dette prosjektet har vært fraværet av fysiske møter. Konsekvensene av dette ble at utviklingen av produktet tok lenger tid enn det som ble estimert under oppstart. I utgangspunktet var målet å ferdigstille et forslag til en maskinlæringsløsning innen slutten av april. Overskridelse av denne fristen var noe som ble ansett som lav risiko i risikoanalysen. Det viste seg tidlig at det ble vanskelig å nå denne tidsfristen. Resultatet av dette er at det ikke har blitt gjennomført noen alternative løsninger til maskinlæringsmodellen. Dette har også resultert i en omorganisering av rapporten og noe omformulering av mål og GANTT-skjema. Selv om det tok lengere tid å utvikle maskinlæringsmodellen, har rapportskrivingen i stor grad gått som planlagt. Pandemien har også gjort at alle ansatte i WA har blitt permittert. Muligheten for å implementere modellen på plattformen har med dette falt bort.

7.3 Begrensninger

Mange av ressursene som skulle bli brukt til denne oppgaven har vist seg å være begrenset på grunn av Koronapandemien. I utgangspunktet var kontoret til WA en ressurs som skulle bli tatt mye i bruk. Kontoret ville gitt gruppen god plass til jobbing og muligheten til tett kommunikasjon med WA, og deres ressursperson Hammerbeck. Alle ansatte hos WA har i tillegg vært i permisjon, som har medført dårlige muligheter til å få til implementering av modellen på plattformen.

Kommunikasjon med gruppens medlemmer måtte for det meste skje over Skype, eller Zoom. Dette har ikke vært optimalt i forhold til å holde kommunikasjonen igjennom utviklingsperioden. Dette har begrenset effektiviteten i arbeidet.

7.4 Refleksjon

Det kan konstateres at ingen var forberedt på arbeidsforholdene under denne bacheloroppgaven. Å reflektere over hvordan ting har blitt gjort sammenlignet med hvordan ting burde blitt gjort, blir derfor noe annerledes. Basert på håndteringen av de ekstra utfordringene som har oppstått på grunn av pandemien er gruppen fornøyd med sin evne til tilpasning. Det har blitt avholdt daglige møter og målene har blitt evaluert og oppdatert underveis. Den *agile* tilnærmingen til arbeidet har blitt overholdt.

Under oppgaven har gruppen dratt mye nytte av innspill fra både veileder fra HVL og kontaktperson fra WA. I retrospekt vurderes det om gruppen brukte disse ressursene til dets fulle potensial. Spesielt da i forhold til kontaktperson i WA, Hammerbeck. Det var til tider utfordrende å sette ord på problemer gruppen hadde med utviklingen, samt å forstå tilbakemeldinger gitt over nettet. Denne delen kan vurderes som et område med forbedringspotensial. Gruppen kunne brukt de ressursene som var til rådighet på en mer effektiv måte.

Gruppen selv kunne vurdert ytterligere møter per dag. Utenom spesielle tilfeller ble møtene stort sett avgrenset til et statusmøte om morgenen. På disse møtene ble det diskutert arbeid fra dagen før, samt planer for videre arbeid samme dag. Et potensielt møte midtveis i arbeidsdagen eller ved avslutning kunne vært verdifullt.

8 KONKLUSJON OG VIDERE ARBEID

8.1 Videre arbeid

Oppgaven har gode muligheter for videre utvikling. Ved endt prosjekt er det utviklet en løsning som tilfredsstiller krav i forhold til nøyaktighet av prediksjoner. Et naturlig neste steg vil være å *tune* (finstille) modellen mer, og gjøre den mer generalisert.

Som nevnt i begrensninger har det ikke vært mulig å implementere modellen på WA sin plattform. Videre arbeid er derfor nødvendig for å gjennomføre denne implementeringen.

Når en registrerer seg og velger skills på WA sine sider så skal en også velge erfaringsnivå. Denne skalaen rangerer fra *Level 1: Basic* til *Level 4: Expert*. Muligheten om å bruke den skalaen som innflytelse i maskinlæringsmodellen kan vurderes. Den kan fungere som en rangeringsmåleenhet. Noe som har blitt observert i eksempler under utforsking i dette prosjektet, er at en kan bruke tilleggsinformasjon som dette til å forbedre en modell.

Modellen er utviklet for å lære. Når WA får inn flere nye brukere vil det medføre mer data for modellen å trene på. Som nevnt i 5.2.1 i rapporten vil dere være viktig med kontinuerlig trening av modellen. Dette er et viktig aspekt i maskinlæring da en håper at ytelsen vil øke etter hvert som den lærer.

8.2 Konklusjon

Anbefalingssystemer er noe som blir mer og mer brukt i moderne applikasjoner. Flere forskjellige tjenester som for eksempel filmstrømming og netthandel drar nytte av anbefalingsalgoritmer. WA hadde allerede undersøkt muligheten og potensiell verdi av en slik løsning på sitt produkt. Dette ble grunnlaget for denne bacheloroppgaven.

Gruppen har hatt tett samarbeid med oppdragsgiver, spesielt under oppstart av prosjektet. Sammen med WA ble gruppen enig om utviklingsmiljø, samt anbefaling av bibliotek basert på eksisterende prosjekt utført av WA.

Resultat av bacheloroppgaven er to fungerende maskinlæringsmodeller som predikerer skills med tilfredsstillende sannsynlighet. Disse modellene er utviklet ved bruk av Python bibliotekene Keras/Tensorflow og Spotlight. Begge bibliotekene er tilrettelagt for å utvikle modeller som forholder seg til implisitte datasett.

Gruppen tar med seg mye verdifull læring fra denne bacheloroppgaven. Det har medført erfaring med analysering og manipulering av data, samt en dypere forståelse for maskinlæring. Dette både med tanke på utnyttelse av Python og tilhørende standardbibliotek, og evaluering av maskinlæringsprinsipper.

En viktig erfaring er også tilpasningsdyktighet til uventede eksterne påvirkninger. Koronapandemien har unektelig påvirket dette prosjektet. Et positivt aspekt kan sies å være erfaring med individuelt arbeid og hjemmekontor. Mest av alt har gruppen erfart krisen som en bekreftelse på verdien av kontinuerlig kontakt gjennom et prosjekt.

Sluttresultatet av dette prosjektet er noe gruppen kan stå inne for. Det har blitt utviklet to forslag til anbefalingssystem, som begge produserer resultat innenfor WA sine forventninger. Resultatene av *precision at k* for modellen utviklet med Spotlight var 0.36 ved k=5. *Precision at k* resultatene for modellen utviklet med Keras var 0.61 ved k=5. En implementering av modellen på WA sin plattform samt videre trening av modellen, må gjennomføres i videre arbeid.

REFERANSER

Aurélien Géron (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd Edition

Blueprint (2020) Agile Methodologies. Tilgjengelig fra: https://www.blueprintsys.com/agile-development-101/agile-methodologies (Hentet 24. mars 2020).

Brownlee, J (2016) Overfitting and Underfitting With Machine Learning Algorithms. Tilgjengelig fra: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ (Hentet 30. mai 2020).

Chand, M (2019) Best Programming Language for Machine Learning. Tilgjengelig fra:https://www.c-sharpcorner.com/article/best-programming-language-for-machine-learning/ (Hentet: 28. mai 2020)

Clair Drumond (2020) What is Scrum? Tilgjengelig fra: https://www.atlassian.com/agile/scrum (Hentet 23. mars 2020).

Cprime (u.å.) What Is AGILE? - What Is SCRUM? - Agile FAQ's. Tilgjengelig fra: https://www.cprime.com/resources/what-is-agile-what-is-scrum/ (Hentet: 19. mars 2020)

Craswell, N (2009) Mean Reciprocal Rank. Tilgjengelig fra: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_488 (Hentet: 01. Juni 2020)

Goodfellow, I., Bengio,Y. & Courville, A. (2016) *Deep Learning*. Cambridge, MA, USA The MIT Press.

Duckett, J. (2011) *HTML & CSS. Design and Build Websites.* Indianapolis, Indiana: John Wiley & Sons, Inc.

Dvergsdal, H (2019) Nevralt nettverk – Tilgjengelig fra: https://snl.no/nevralt_nettverk (Hentet: 14. mai 2020)

Github, Inc., (2020) About pull requests, Tilgjengelig fra: https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests (Hentet: 28. mai 2020)

Himani Bansal (2019) Best Languages for Machine Learning in 2020, Tilgjengelig fra: https://becominghuman.ai/best-languages-for-machine-learning-in-2020-6034732dd242 (Hentet: 20. mai 2020)

Hovde, K og Grønmo, S (2020) Algoritme, Tilgjengelig fra: https://snl.no/algoritme (Hentet: 01. juni 2020)

Jason Brownlee (2019) Supervised and unsupervised Machine Learning Algorithms. Tilgjengelig fra: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ (Hentet: 21. mai 2020)

Jordan, J. (2017) Evaluatin a machine learning model. Tilgjengelig fra: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/ (Hentet: 22. mai 2020)

Jupyter (2020) Jupyter. Tilgjengelig fra: https://jupyter.org/index.html (Hentet: 26. mars 2020)

Koehrsen, W (2018) Neural network Embeddings Explained. Tilgjengelig fra: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526 (Hentet: 24. mai 2020)

Krasner, G. E., & Pope, S. T. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, *1* (3), s. 26-49.

Kristoffersen, B. (2016) *Databasesystemer.* 4 utg. Oslo: Universitetsforlaget AS.

Kula, M 2007 *Implicit factorization models.* Tilgjengelig fra: https://maciejkula.github.io/spotlight/losses.html (Hentet: 28. mai 2020)

Malaeb, M (2017) Recall and Precision at k for Recommender Systems. Tilgjengelig fra: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54 (Hentet: 26. mai 2020)

Mutuvi, S. (2019) Introduction to Machine Learning Model Evaluation, i Heartbeat. Tilgjengelig fra: https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f (Hentet: 19. mai 2020)

Perera, S (2019) An introduction to Reinforcement Learning. Tilgjengelig fra: https://towardsdatascience.com/an-introduction-to-reinforcement-learning-1e7825c60bbe (Hentet: 20. mai 2020)

Perera, S (2019) Reinforcement learning illustration [Illustrasjon]. Tilgjengelig fra: https://miro.medium.com/max/770/1*c3pEt4pFk0Mx684DDVsW-w.png (Hentet: 24. mai 2020)

Pivotal Tracker (u.å.) Quick start & demos. Tilgjengelig fra: https://www.pivotaltracker.com/help/articles/quick_start/ (Hentet: 26. mars 2020)

Reinsel, D., Gantz, J. & Rydning, J. (2018). The Digitization of the World. Tilgjengelig fra: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (Hentet: 28. mai 2020)

Roser, M & Ritchie, H (2013). Technological Progress. Tilgjengelig fra: https://ourworldindata.org/technological-progress (Hentet: 28. mai 2020)

Shah, T. (2017) A Visualisation of the splits [Illustrasjon]. Tilgjengelig fra: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 (Hentet: 20. mai 2020)

Spotlight (2017.) Implicit factorization models. Tilgjengelig fra: https://maciejkula.github.io/spotlight/factorization/implicit.html (Hentet: 27. mai 2020)

Store Norske Leksikon (2019) Nevralt nettverk med et skjult lag [Graf]. Tilgjengelig fra: https://storage.googleapis.com/snl-no-media/media/28536/standard_NervraleNett_Fig2.jpg (Hentet: 27. mai 2020)

Tidemann, A. Og Elster, A.C. (2019) Maskinlæring, i Store norske leksikon. Tilgjengelig fra: https://snl.no/maskinl%C3%A6ring (Hentet: 31. mars 2020)

Wikipedia (2020a) Front end and back end. Tilgjengelig fra: https://en.wikipedia.org/wiki/Front_end_and_back_end (Hentet: 01. Juni 2020)

Wikipedia (2020b) Keras. Tilgjengelig fra: https://en.wikipedia.org/wiki/Keras (Hentet: 19. mars 2020)

Wikipedia (2020c) Open-source software. Tilgjengelig fra: https://en.wikipedia.org/wiki/Open-source_software (Hentet: 01. Juni 2020)

Wikipedia (2020d) Pandas. Tilgjengelig fra: https://en.wikipedia.org/wiki/Pandas_(software) (Hentet: 13. mai 2020)

Wikipedia (2020e) Python (Programming language). Tilgjengelig fra: https://en.wikipedia.org/wiki/Python_(programming_language) (Hentet: 14. mai 2020)

Wikipedia (2020f) PyTorch. Tilgjengelig fra: https://en.wikipedia.org/wiki/PyTorch

(Hentet 31.mai 2020)

Wikipedia (2020g) Scikit-learn. Tilgjengelig fra: https://en.wikipedia.org/wiki/Scikit-learn (Hentet: 19. mars 2020)

Wikipedia (2020h) TensorFlow. Tilgjengelig fra: https://en.wikipedia.org/wiki/TensorFlow (Hentet: 19. mars 2020)

Yifan Hu, Yehuda Koren og Chris Volinsky (2008) Collaborative filtering for implicit feedback datasets. Tilgjengelig fra: http://yifanhu.net/PUB/cf.pdf (Hentet: 12. mai 2020)

Statsministerens kontor og Helse- og omsorgsdepartementet (2020) Omfattende tiltak for å bekjempe koronaviruset. Tilgjengelig fra: https://www.regjeringen.no/no/aktuelt/nye-tiltak/id2693327/ (Hentet: 28. mai 2020)

Vedlegg

Vedlegg A: Akronym

API	Application Programming Interface
CPU	Central Processing Unit
CSV	Comma-separated values
CTO	Chief Technical Officer
CV	Curriculum Vitae
HVL	Høgskulen på Vestlandet
HTML	HyperText Markup Language
IT	Informasjonsteknologi/Informatikk
JSON	JavaScript Object Notation
MVC	Model View Controller
WA	Wide Assessment

Vedlegg B: Ordliste

Backend	Delen av utviklet produkt som er nærmest database. Funksjonalitet som ligger i bakgrunn og omhandler håndtering av data. Dataaksesseringslag (Wikipedia, 2020a).
Frontend	Delen av utviklet produkt som er nærmest bruker. Det som kan sees og samhandles med. Presentasjonslag (Wikipedia, 2020a).
C#	Programmeringsspråk.
Java	Programmeringsspråk.
Embeddings	En embedding er en kartleggelse av diskret, kategoriske variabler til en vektor av kontinuerlige verdier. (Koehrsen, W 2018).
Localhost	Tilgang til lokalnettverk via datamaskin som brukes.
Open-source	Open-source (Wikipedia, 2020h).
Skill	En ferdighet som kan legges til en profil på WA sin plattform.
Skills	Flere ferdigheter.
DataFrame	To-dimensjonell datastruktur brukt i Python
PyTorch	Open source-maskinlæringsbibliotek som er basert på Torch-biblioteket (Wikipedia, 2020h). Open source-maskinlæringsbibliotek som er basert på Torch-biblioteket (Wikipedia, 2020h).

Vedlegg C: Tabell-og figurliste