

BACHELORPROSJEKT vår 2022

# AI-BASERT ANALYSE

## av tidsseriedata fra atleter

### SKREVET AV:

Bernadette Fanni Finheim  
Hanna Bækken Nilsen  
Tonje Martine Lorgen Kirkholt  
Helene Birkeflet Prescott



### Institutt for Informasjonsteknologi

Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo  
 Besøksadresse: Holbergs plass, Oslo

PROSJEKT NR.
9

TILGJENGELIGHET
Åpen

Telefon: 22 45 32 00

# BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
AI-basert analyse av tidsseriedata fra atleter	25.05.2022
	ANTALL SIDER / BILAG
	107 / 1
PROSJEKTDELTAKERE	INTERN VEILEDER
Bernadette Fanni Finheim (S341857) Hanna Bækken Nilsen (S341879) Tonje Martine Lorgen Kirkholt (S341844) Helene Birkeflet Prescott (S341873)	Pål Halvorsen
OPPDRAKGIVER	KONTAKTPERSON
Forzasys	Pål Halvorsen Steven Hicks

SAMMENDRAG
<p>Denne rapporten er et endelig resultat av bachelorprosjektet til fire Dataingeniørstudenter ved fakultet for Teknologi, Kunst og Design (TKD) på OsloMET. Arbeid og ferdigstillelse av oppgaven har blitt gjennomført i løpet av våren 2022, med Forzasys som oppdragsgiver. Deres overordnede mål er å utvikle en funksjon for prediksjon av skade som kan implementeres i applikasjonen pmSys, som til nå hovedsakelig er brukt til loggføring av velværeparametere og treningsbelastning fra idrettsutøvere.</p> <p>Rapporten presenterer prosessen med å utforske løsninger innen maskinlæring, og videre predikere en utøvers opplevde dagsform basert på historisk data. Det har blitt fokusert på tre valgte maskinlæringsmodeller; Random Forest, Autoregressive Integrated Moving Average (ARIMA) og Long Short-Term Memory(LSTM). Samtidig har gruppen sett på om resultatet potensielt kan brukes som et skadeforebyggende verktøy for både trener og atlet. På denne måten vil man kunne fremme videre prosesjon og helse for idrettsutøveren.</p> <p>Med en gjennomgående drøfting av valgene som har blitt tatt i løpet av prosjektperioden, forklares arbeidet som er gjort tydelig for leseren. Vi gjøre rede for beslutninger angående teknologi, fremgangsmåter og modeller presentert i rapporten. Den endelige konklusjonen vil baseres på det helhetlige forskningsarbeidet, hvor vi tilbyr alternativer og forslag for fremtidige mål. Her vil vi diskutere det gjennomførte arbeidet, samt gi et innblikk i refleksjoner som ble gjort i sluttprosessen.</p>

3 STIKKORD
Maskinlæring
Sport
Prediksjon

# Sammendrag

Denne rapporten er et endelig resultat av bachelorprosjektet til fire Dataingeniørstudenter ved fakultet for Teknologi, Kunst og Design (TKD) på OsloMET. Arbeid og ferdigstillelse av oppgaven har blitt gjennomført i løpet av våren 2022, med Forzasys som oppdragsgiver. Deres overordnede mål er å utvikle en funksjon for prediksjon av skade som kan implementeres i applikasjonen pmSys, som til nå hovedsakelig er brukt til loggføring av velværeparametere og treningsbelastning fra idrettsutøvere.

Denne rapporten presenterer prosessen med å utforske løsninger innen maskinlæring, og videre predikere en utøvers opplevde dagsform basert på historisk data. Det har blitt fokusert på tre valgte maskinlæringsmodeller; Random Forest, Autoregressive Integrated Moving Average (ARIMA) og Long Short-Term Memory (LSTM). Samtidig har gruppen sett på om resultatet potensielt kan brukes som et skadeforebyggende verktøy for både trener og atlet. På denne måten vil man kunne fremme videre progresjon og helse for idrettsutøveren.

Med en gjennomgående drøfting av valgene som har blitt tatt i løpet av prosjektperioden, forklares arbeidet som er gjort tydelig for leseren. Vi gjør rede for beslutninger angående teknologi, fremgangsmåter og modeller presentert i rapporten. Den endelige konklusjonen vil baseres på det helhetlige forskningsarbeidet, hvor vi tilbyr alternativer og forslag for fremtidige mål. Her vil vi diskutere det gjennomførte arbeidet, samt gi et innblikk i refleksjoner som ble gjort i sluttprosessen.

# Innholdsfortegnelse

<b>Sammendrag</b>	<b>2</b>
<b>Innholdsfortegnelse</b>	<b>3</b>
<b>1 Introduksjon</b>	<b>6</b>
1.1 Oppdragsgiver	6
1.2 Prosjektgruppe	7
1.3 Veiledere	8
1.4 Bakgrunn og motivasjon for oppgaven	8
1.5 Dagens situasjon	10
1.6 Problemstilling	11
1.7 Mål for prosjektet	12
1.8 Begrepsliste og forkortelser	13
1.9 Disposisjon for rapporten	16
<b>2 Arbeidsmetoder</b>	<b>17</b>
2.1 Teamkontrakt	17
2.2 Arbeidsmetodikk	17
2.3 Faglige forutsetninger	19
2.4 Rammebetingelser	20
2.5 Rammeverk	20
2.5.1 Verktøy	21
Jupyter Notebook	21
GitHub	21
Miro	22
Slack	22
Zoom	22
2.5.2 Programmeringsspråk	23
Python	23
2.5.3 Bibliotek	23
Pandas	23
Numpy	23
Scikit-learn	23
Skforecast	24
Statsmodels	24
Pmdarima	24
Keras/Tensorflow	24
<b>3 Bakgrunn og tidligere arbeid</b>	<b>25</b>
3.1 Tidligere arbeid	25
3.2 Introduksjon til maskinlæring	26

3.2.1 Tradisjonell maskinlæring	27
3.2.2 Deep Learning	29
3.3 Vårt utgangspunkt	30
<b>4 Innsamling og bearbeiding av data</b>	<b>31</b>
4.1 Datasett	31
4.1.1 Tidsserier	31
4.1.2 Informasjon om dataen	32
4.1.3 Manglende verdier	35
4.1.4 Datamengder	37
4.2 Ferdigstilt datasett	38
4.3 Videre forberedelser	39
4.3.1 Undersøkelse om stasjonaritet	39
4.3.2 Test- og treningssett	40
4.3.3 Lags og steps	41
<b>5 Utvikling av maskinlæringspipeline</b>	<b>42</b>
5.1 Prosess og livssyklus	42
5.2 Teknikker og problemstillinger	43
5.3 Valg av maskinlæringsmodell	43
5.3.1 Random Forest	44
5.3.2 Autoregressive Integrated Moving Average	47
5.3.3 Long Short-Term Memory	50
<b>6 Trening og testing av maskinlæringsmodeller</b>	<b>53</b>
6.1 Forhåndsbestemte variabler og innstillinger	53
6.2 Måling av feilestimat	54
6.3 Baseline for test score	56
6.4 Random Forest Regression	58
6.4.1 Hyperparamter tuning	59
6.4.2 Kalkulere prediktorene fra tidsserie	64
6.4.3 Konklusjon	67
6.5 ARIMA	67
6.5.1 Auto-Arima	70
6.5.2 Manuell ARIMA-modell	73
6.5.3 Konklusjon	74
6.6 LSTM	74
6.6.1 Oppbygging av modell	75
6.6.2 Gjennomføring av eksperimentene	75
6.6.3 Konklusjon	80
6.7 Sammenligning av prediksjonsevne	81
6.8 Eksperimentering med Convolutional Neural Network	81
6.9 Eksperimentering med prediksjon av skade	84
6.10 Konklusjon	91

<b>7 Diskusjon og refleksjoner</b>	<b>92</b>
7.1 Fremgangsmåte	92
7.2 Alternative utgangspunkt	93
7.3 Fremtidige mål	94
7.4 Arbeidsmetodikk	94
7.5 Læringsutbytte	95
<b>8 Konklusjon</b>	<b>97</b>
<b>9. Referanser</b>	<b>99</b>
<b>10. Vedlegg</b>	<b>102</b>
10.1 Teamkontrakt	102

# 1 Introduksjon

Denne rapporten er en del av det avsluttende prosjektet for emnet DATA3900, bacheloroppgaven for vårt studium. Prosjektet er et gruppearbeid mellom 4 studenter, hvor kurset og arbeidet startet høsten 2021. Høstsemesteret var planleggingsfasen for prosjekt og oppgave, hvor hovedfokuset var å finne en oppdragsgiver, fastsette prosjektoppgave og starte med forarbeidet. Dette bestod av å skrive statusrapport, prosjektskisse og kontrakt mellom gruppe og oppdragsgiver. Deretter gjenstod forprosjektrapporten på nyåret, før arbeidet med sluttrapporten startet 24. januar 2022.

I dette kapittelet presenteres oppdragsgiver, prosjektgruppe og veilederne som har bistått gjennomgående i arbeidsperioden. Videre følger en beskrivelse av motivasjon og bakgrunn for prosjektet, deretter en presentasjon av problemstilling og redegjørelse av målene vi har satt. Til slutt er det lagt ved en begrepsliste med ord og uttrykk som er ment å fungere som et oppslagsverk under lesingen av rapporten, etterfulgt av prosjektets disposisjon.

## 1.1 Oppdragsgiver

Forzasys er et teknologisk selskap som ble dannet i juni 2014, med utspring fra Simula Research Laboratory. Hovedmålet til bedriften er å engasjere interesser innenfor sport ved hjelp av teknologi. Bedriften står for innovative, effektive og fleksible videosystemer samt digital sportsanalyse av utøvere og lag.<sup>1</sup>

Ved hjelp av maskinlæring, teknologi i videostrømming, distribuerte systemer og nettverk tilbyr Forzasys løsninger for fremtidens videostrømming og sportsanalyse. Bedriften har tre vektlagte produkt; Forzify, et direkte og on-demand video system som støtter OTT (over-the-top) strømming; Live Camera, et analytisk verktøy og en applikasjon som tilbyr direkte detaljerte tilbakemeldinger for video; pmSys (player monitoring system), en applikasjon som monitorerer ytelsen hos idrettsutøvere ved hjelp av registrert data. Sistnevnte, pmSys, er en sentral del i vårt prosjekt.

---

<sup>1</sup> <https://forzasys.com/>

## 1.2 Prosjektgruppe

Vi er fire studenter som alle går siste og fullførende år på dataingeniør ved OsloMet. Da vi startet skolegangen høsten 2019 fikk vi raskt kontakt gjennom oppstart og fadderuke, og har siden jobbet jevnlig sammen i grupper i andre fag under studieløpet. Som gruppe har vi opparbeidet oss en forståelse for hvordan vi på best mulig måte kan samarbeide mot et felles mål. Dette gir oss et godt grunnlag og gode forutsetninger for arbeidet med bachelorprosjektet.



Navn: Bernadette Fanni Finheim  
Studentnummer: S341857  
Epost: [fanni.szeplaki@gmail.com](mailto:fanni.szeplaki@gmail.com)



Navn: Hanna Bækken Nilsen  
Studentnummer: S341879  
Epost: [hannabn00@gmail.com](mailto:hannabn00@gmail.com)



Navn: Tonje Martine Lorgen Kirkholt  
Studentnummer: S341844  
Epost: [tonjelorgen@gmail.com](mailto:tonjelorgen@gmail.com)



Navn: Helene Birkeflet Prescott  
Studentnummer: S341873  
Epost: [heleneprescott@gmail.com](mailto:heleneprescott@gmail.com)

## 1.3 Veiledere

Pål Halvorsen

*Intern veileder fra OsloMET*

[palh@oslomet.no](mailto:palh@oslomet.no)

+47 970 80 007

*Professor*

Steven Hicks

*Ekstern veileder fra Simula*

[steven@simula.no](mailto:steven@simula.no)

+47 950 88 557

*PhD Student ved Simula*

## 1.4 Bakgrunn og motivasjon for oppgaven

pmSys er en applikasjon utviklet av Forzasys i samarbeid med studenter og forskere ved Simula Research Laboratory, sammen med Universitetet i Tromsø (UiT).<sup>2</sup> Dette er et monitoreringsverktøy som brukes til å logge og overvåke data fra toppidrettsutøvere. Appen brukes av utøvere og trenerne for å overvåke treningsbelastning og helse. Gjennom applikasjonen kan spillerne etter kamp og trening rapportere informasjon om deres generelle velvære og helse, skader og opplevd anstrengelse. For å måle anstrengelse brukes Borg skala (RPE, Ratings of perceived exertion), som er en metode der man setter et subjektivt tall på hvordan man selv opplever treningsbelastningen.<sup>3</sup> I Figur 1 og Figur 2 ser man spørreskjemaet som spilleren trykker seg gjennom i appen.

---

<sup>2</sup> <https://www.forzasys.com/pmSys.html>

<sup>3</sup> <https://www.revmatiker.no/aktiweb/borg-skala/>

Figure 1 displays the PMSys survey interface for daily well-being logging. The interface consists of five screens followed by a summary screen:

- Screen 1: Report for**  
When is this report for?  
Buttons: Today, Yesterday, 2 Days ago, 1 Week ago  
Time: 12:27  
Date: Friday 18 February 2022  
Done
- Screen 2: Readiness**  
How ready are you to train?  
Scale: 1 - Not ready at all to 10 - Can't wait!
- Screen 3: Fatigue**  
How fatigued do you feel?  
Scale: 1 - Very tired to 5 - Very fresh
- Screen 4: Sleep Duration**  
How much did you sleep last night?  
Duration: Hours  
Quick Select grid:  
7h, 1h, 2h  
3h, 4h, 4.5h  
5.5h, 6h, 6.5h  
7h, 7.5h, 8h  
8.5h, 9h, 9.5h  
10h, 10h, 12h
- Screen 5: Sleep Quality**  
How well did you sleep?  
Scale: 1 - Insomnia to 5 - Excellent sleep

A large blue arrow points from the fifth screen to a summary screen on the right.

**Summary Screen:**

- Date: 18.02.2022, 12:27
- Readiness: 5
- Fatigue: 3
- Sleep Duration: 4 h
- Sleep Quality: 3
- Soreness: 4
- Stress: 4
- Mood: 4
- Menstruation: Off

Figur 1: Oversikt over spørreskjema i PMSys for loggføring av daglig velvære.

Figure 2 displays the PMSys survey interface for RPE logging. The interface consists of five screens followed by a summary screen:

- Screen 1: Session end time**  
When did the session end?  
Buttons: Just now, 5m ago, 15m ago, 30m ago, 60m ago, 90m ago  
Time: Pick Date  
Date: Pick Date  
Done
- Screen 2: Session duration**  
How many long was your session?  
Duration: Minutes  
Quick Select grid:  
10 min, 20 min, 30 min  
40 min, 50 min, 60 min  
70 min, 80 min, 90 min  
100 min, 110 min, 120 min  
130 min, 140 min, 150 min
- Screen 3: Session type**  
What type of session was this?  
Options: Competition, Individual Session, Team Session (selected)  
Please select keywords:  
strength, endurance, soccer, other
- Screen 4: Session RPE**  
How hard was this session?  
Scale: 1 - Very light activity to 10 - Max effort activity
- Screen 5: Summary**  
Review and submit  
Date: 18.02.2022, 13:32  
Duration: 50 min  
Categories: team | soccer  
Perceived Exertion: 6

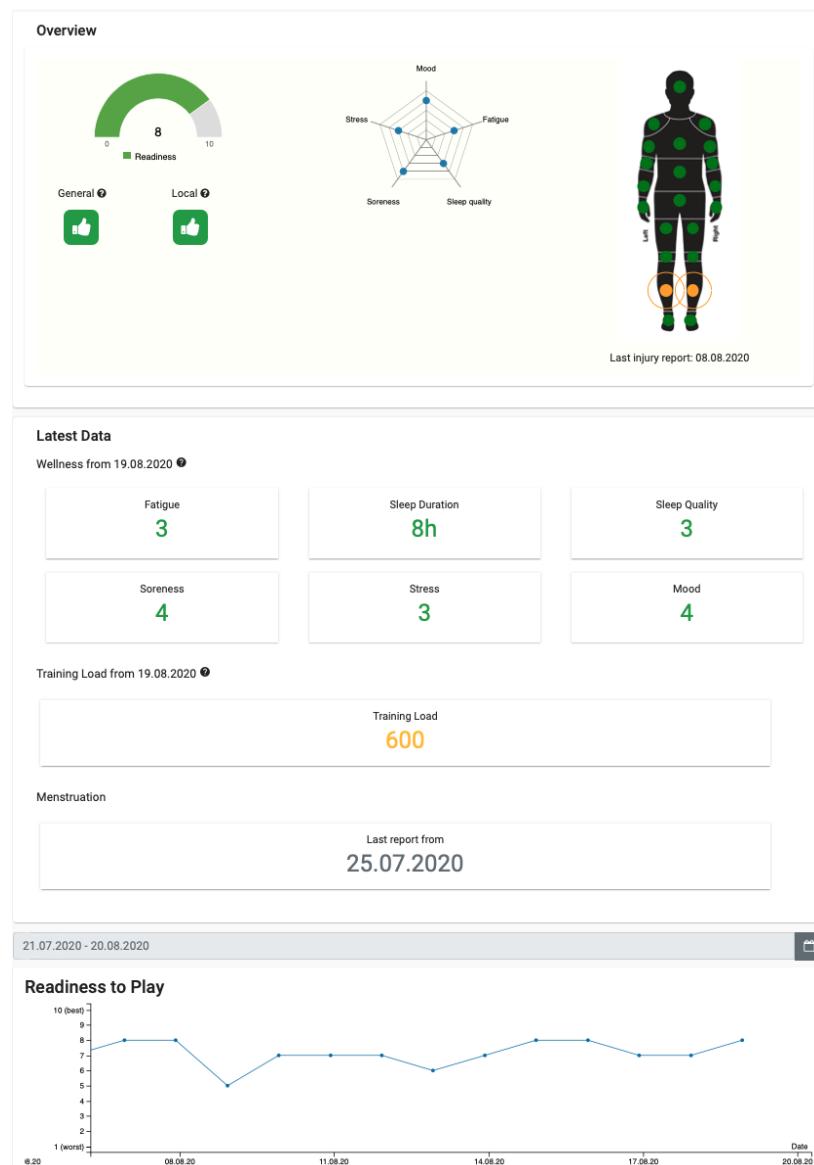
Figur 2: Oversikt over spørreskjema i PMSys for loggføring av RPE.

Applikasjonen er et verktøy som tilbyr mye informasjon til spillerne selv, men også trenerne deres. Registrerte data fra spillere tilhørende et helt fotballag gir en oversikt over hvordan den totale situasjonen og formen er, hvilket gir trenere muligheten til å blant annet planlegge en treningsmengde som er tilpasset etter loggført helsetilstand.

Forzasys ønsker å utnytte potensialet ved å tilby enda mer informasjon på bakgrunn av den innsamlede dataen. Hvor man i dag selv gjør vurderinger ut i fra oversikten over helse- og velværet tilstand, er det ønskelig å tilby dette direkte i applikasjonen for både spiller og trener. Dette krever mer analyse og behandling av dataene, samt utvikling av optimale modeller som kan gjøre utregninger og presentere samtlige resultat.

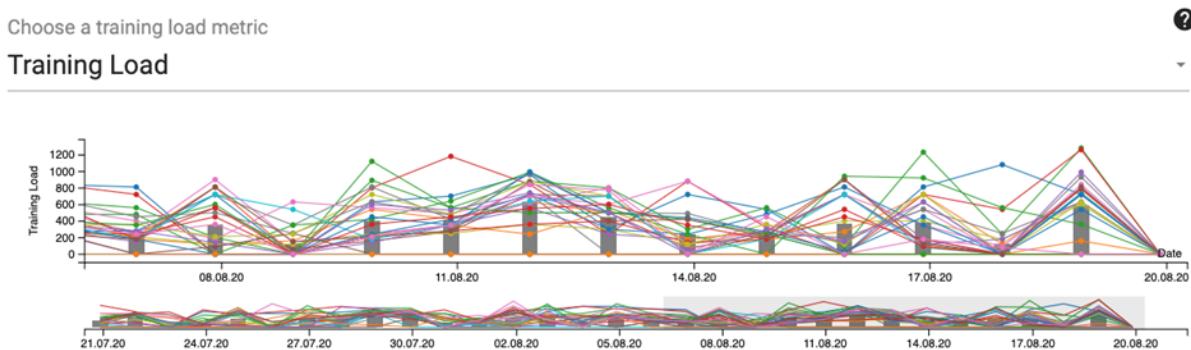
## 1.5 Dagens situasjon

pmSys har allerede vist seg å være et nyttig verktøy for å overvåke atletenes fysiske helse, og er i dag tatt i bruk av flere toppfotballag i Norge. Per dags dato gir ikke applikasjonen mer enn en oversikt over informasjonen som registreres fra spillerne, sammen med enkle trender basert på denne dataen. I Figur 3 ser man hvordan oversikten for en spiller fremstilles i trenerportalen. Her presenteres nylig registrerte verdier for generell helsetilstand, eventuelle skader og hvor på kroppen de er, kombinert med grafisk fremstilling av loggførte Readiness to Play parameterne.



Figur 3: Fremstilling av hvordan trenerdashboardet ser ut for hver enkelt spiller i pmSys (Johansen et al., 2019, s.3, Figure 2).

Trenerportalen viser også en grafisk fremstilling av den totale treningsbelastningen for hele laget over tid, vist i Figur 4.



Figur 4: Trenerdashboard med hele lagets treningsbelastning visualisert (Johansen et al. 2019, s.3, Figure 3).

Ut ifra figurene ser vi at det ikke blir gitt noen form for prediksjoner av generell form eller Readiness to Play, et aspekt ved applikasjonen som Forzasys ønsker å utvikle. I tillegg til at dette kan gjøre det enklere å finjustere treningsbelastningen hos enkeltspillere og lag, har det potensiale til å bidra med forebygging og forhindring av skade. En slik funksjon kunne gjort pmSys til en uvurderlig del av treningshverdagen.

## 1.6 Problemstilling

Inntekten til de største fotballlagene i verden ligger på flere hundre millioner kroner. Det er flere faktorer som kan påvirke den økonomiske situasjonen, og noe man vil unngå er å måtte sette en spiller på benken på grunn av en skade. Samtidig som det kan få alvorlige konsekvenser for spilleren selv, kan det resultere i store brister i økonomien. En slik hendelse vil koste et fotballag svimlende summer, og har samtidig potensiale til å påvirke sluttlassering i en liga eller cup. Eliakim et al. (2020) presenterer i sin artikkel en oversikt over engelske toppfotballlag, og hvordan antall skadede spillere har påvirket lagenes endelige posisjon i English Premier League (EPL) i sesongen 2016-2017. Her observerer de at det er en sammenheng i differansen mellom forventet og faktiske resultat i ligaen, og antall registrerte skader, og konkluderer med at skadede spillere har en stor påvirkning på den totale prestasjonen til laget.(Eliakim et al., 2020). Fra og med spilleren har blitt skadet vil det skape ringvirkninger som påvirker laget som en helhet.

Hva om en slik hendelse kunne ha blitt unngått om man hadde hatt en oversikt over en spillers forventede form i forkant av kamp og trening, basert på parameteren Readiness to Play? En prediksjon som kunne antyde i hvor stor grad det eksisterte en sjanse for at en

spiller, med nåværende treningsbelastning, var utsatt for skade? En slik løsning vil kunne bidra til både skadeforebyggelse og forsterking av rehabiliteringsprosesser, hvor man kan få automatisert styringen av planlagt treningsbelastning. For fotballag vil det være potensiale for store økonomiske gevinster å investere i en slik løsning.

I vårt prosjekt sentrerer problemstillingen seg rundt muligheten for å predikere Readiness to Play, og fungerer som et første steg mot prediksjon av resterende velværeparametere. Dette kan potensielt bidra til optimal treningsplanlegging, og i fremtiden prediksjon av skader. Gjennom testing av ulike maskinlæringsmodeller forsøker vi å finne frem til de alternativene som viser mest potensiale for videre utvikling. Dette gjøres gjennom analysering av registrerte tidsseriedata fra utvalgte spillere, som videre tas i bruk i utvalgte modeller. Vårt arbeid vil ikke resultere i et ferdig produkt eller en modell som kan implementeres i pmSys, fordi vi utfører et forskningsarbeid som fungerer som et steg på veien til nettopp et potensielt ferdig produkt. Målet med prosjektet er å bidra til løsninger som kan danne et grunnlag for arbeid inn i det overordnede målet; prediksjon av skade.

## 1.7 Mål for prosjektet

Det har allerede blitt utført testing rundt bruk av maskinlæring for å forutsi fremtidig dagsform og skader på toppfotballspillere, og dette går vi nærmere inn på i seksjon [3.1](#). Resultatene er så langt lovende, men det må utføres mer arbeid og testing før man kan implementere en slik funksjon i pmSys. I dag tilbyr ikke pmSys-applikasjonen mer enn en oversikt over informasjonen som registreres av hver enkelt utøver, oppsettet av dette vist i Figur 3 og Figur 4 i seksjon [1.5](#). En utvidelse av applikasjonen innebærer at man kan få prediksjoner av spillernes form et visst antall dager frem i tid, og i trenerportalen vil man kunne få all denne informasjonen presentert for hele laget. Denne informasjonen vil potensielt bidra til at trenere har et bedre grunnlag for å ta valg for unngå skader på spillerne, og samtidig forenkle rehabilitering.

Det overordnede målet for oppdragsgiver er å utvikle en maskinlæringsmodell som utfører prediksjon av skade hos en utøver. Formålet med vårt prosjekt er å være steg i prosessen dit, og er et delmål i det store bildet. Vårt arbeid vil undersøke ulike algoritmer innenfor maskinlæring, for å se hvilke som gir nøyaktige resultat gjennom prediksjon av parameteren Readiness to Play. Vi vil drøfte testresultatene, sammenligne modellene med hverandre, og til slutt vil komme frem til en endelig konklusjon om prestasjonsevnene til hver enkelt algoritme. Arbeidet vi har gjort under prosjektperioden resulterer i foreslåtte fremtidige

mål for oppdragsgiver, hvor denne rapporten er ment å være et bidrag i Forzasys sitt videre arbeid for innføring av prediksjon i trenerportalen til pmSys.

## 1.8 Begrepsliste og forkortelser

Det har blitt valgt å inkludere en oversikt over ulike begrep og beskrivelser som blir brukt gjennomgående i oppgaven. Dette for å gjøre det enklere for leseren å forstå, samt lete opp ord og uttrykk underveis. Gruppen opplever at de fleste ord og uttrykk som brukes innenfor kunstig intelligens og maskinlæring er mest utbredt på engelsk, og derfor har vi valgt å bruke originaluttrykkene der det er mest naturlig og forståelig med tanke på konteksten. Det vil bli gitt en alternativ oversettelse når nye begrep presenteres i rapporten, samtidig som de finnes oppført i Tabell 1 under.

<b>Artificial Intelligence (AI)</b>	Kunstig intelligens. Simulering av menneskelig intelligens i teknologi og maskiner.
<b>Artificial Neural Network (ANN)</b>	Modell som forsøker gjennom ulike algoritmer å etterligne hvordan de biologiske nervecellene i en hjerne er organisert på.
<b>Autoregression (AR)</b>	Modell som bruker tidligere observasjoner fra tidligere tidssteg for å predikere neste.
<b>Autoregressive Integrated Moving Average (ARIMA)</b>	Modell som brukes til å analysere tidsseriedata.
<b>Backtesting</b>	En metode hvor man tester ulike varianter av en modell for å finne den mest optimale.
<b>Baseline</b>	Terskelverdi for en forventet test score.
<b>Boolsk variabel</b>	Definisjon på en variabel som kun defineres av verdien sant/usant.
<b>Deep Learning (DL)</b>	En underkategori innenfor maskinlæring. Tar for seg algoritmer som forsøker å etterligne funksjon og struktur i det

	nevrale nettverket i hjernen.
<b>Decision Trees (DT)</b>	Modell som bruker en trestruktur der data blir delt inn i noder, «sub-groups» etter en gitt betingelse.
<b>Dummy Model</b>	Et verktøy som setter en terskelverdi som man kan sammenligne faktiske resultat fra maskinlæringsmodeller med.
<b>Dummy Regressor</b>	Dummy modell som lager prediksjoner med enkle strategier uten å ta hensyn til input data.
<b>Lag</b>	Fiksert tidsperiode som har passert, tidsperiode av tidligere observasjoner.
<b>Long Short-Term Memory (LSTM)</b>	Modell som brukes til å analysere tidsseriedata.
<b>Machine Learning (ML)</b>	Maskinlæring. Innfører prinsippet om kunstig intelligens inn i teknologiske komponent.
<b>Moving Average (MA)</b>	En variabel som kalkuleres for å identifisere eventuelle trender og sesongmessige forhold i et datasett.
<b>Multivariate</b>	Flere variabler som inngår i en analyse.
<b>Neural Network (NN)</b>	Nevrale nettverk. En betegnelse innenfor maskinlæring som omfatter datastrukturer og algoritmer som er inspirert av organiseringen av nerveceller i hjernen.
<b>Overtilpasning (Overfitting)</b>	Når en modell tar inn støy fra datasettet og passer for tett til treningssettet vil da modellen ikke klare å generalisere til ny data, og vi ender opp med overtilpasning.
<b>Plotte</b>	Grafisk fremstille innsamlet data ved å sette inn punkter for hver observasjon og produsere en graf.
<b>Random Forest (RF)</b>	Ensemble-algoritme som bruker flere decision trees algoritmer med tilfeldig prøvetaking.

<b>Regression (Regresjon)</b>	Regresjon. Prosessen med å finne en funksjon som passer til et datasett.
<b>Scrum</b>	En smidig prosess-rammeverk utviklet for å støtte kompleks produktutvikling basert på erfaring og kunnskap.
<b>Sesongmessige forhold</b>	Sesongmessige forhold i et datasett.
<b>Smidige metoder</b>	En utbredt arbeidsmetodikk. Går ut på å dele opp arbeidet i mindre deler med korte tidsfrister (kalt sprint), hvor man overveier arbeidet med en gang en sprint er gjennomført.
<b>Stasjonaritet</b>	Stasjonaritet. Om dataen er stasjonær vil det si at man ikke kan identifisere trender eller sesongmessige forhold.
<b>Step</b>	Antall steg av prediksjoner, antall prediksjonshopp.
<b>Supervised Learning</b>	Trening og testing av maskinlæringsalgoritmer skjer gjennom manuelle prosesser og tilbakemeldinger (nor. overvåket læring).
<b>Time Series</b>	Tidsserie. En samling av observasjoner hentet inn gjennom kontinuerlig loggføring over tid.
<b>Trend</b>	Observasjoner av langvarig økninger eller reduksjon i dataen.
<b>Univariate</b>	Enkel variabel som inngår i en analyse.
<b>Unsupervised Learning</b>	Trening og testing av maskinlæringsalgoritmer med mindre påvirkning manuelt gjennom prosessen (nor. uovervåket læring).

Tabell 1: Begrepsliste

## 1.9 Disposisjon for rapporten

Denne sluttrapporten består av totalt 10 kapitler. Det første kapittelet har bestått av en introduksjon til oppgaven og problemstillingen vi arbeider med, hvor det har blitt presentert utfyllende informasjon som er nødvendig for å forstå helheten av prosjektet. I kapittel [2](#) går vi gjennom besluttede arbeidsmetoder og fremgangsmåter som har blitt satt i forkant av arbeidet og tatt i bruk gjennomgående. Kapittel [3](#) gir en introduksjon til den grunnleggende teorien bak maskinlæring og dyp læring. Kapittel [4](#), [5](#) og [6](#) består av selve maskinlæringsarbeidet som er utført under prosjektet, hvor vi presenterer informasjon om innsamling og bearbeiding av datasett, utvikling av maskinlæringspipeline og til slutt trening og testing av maskinlæringsmodellene. I kapittel [7](#) drøfter vi rundt resultatene vi har fått, samt ulike aspekt ved arbeidsmetodikk, fremgangsmåte og alternative utgangspunkt som kunne ha blitt gjort annerledes. Helt til slutt, i kapittel [8](#), trekker vi en endelig konklusjon for rapporten og arbeidet. I kapittel [9](#) finner man vedlagt referanseliste, og kapittel [10](#) inneholder vedlegg.

## 2 Arbeidsmetoder

Det er viktig å legge et solid grunnlag gjennom spesifisering av krav og mål til arbeidsprosessen før man går i gang med selve prosjektarbeidet. For gruppen var dette et spesielt viktig punkt å drøfte forhånd, siden ingen av oss har arbeidet med en utredningsoppgave tidligere. En arbeidsplan alle kunne stå inne for ble viktig, da dette gir en større garanti for at vi får gjort det planlagte arbeidet innenfor fristen.

I denne seksjonen vil vi utdype krav som har blitt satt til arbeidsprosessen og planlegging av arbeidet. Her vil det bli forklart hvordan gruppen har gått frem for å få mest mulig ut av prosjektperioden. Videre vil det komme underseksjoner som gir informasjon om de ulike verktøyene og bibliotekene som har blitt brukt innenfor programmering, kommunikasjon og dokumentasjon.

### 2.1 Teamkontrakt

For å sikre et godt samarbeid utarbeidet gruppen en teamkontrakt ved prosjektstart. En teamkontrakt er en avtale mellom alle i gruppen som definerer det ønskelige arbeidsforholdet under prosjektarbeidet. Kontrakten tar for seg dimensjonene for prosjektet, slik som viktige mål og tidsfrister. I tillegg er det lagt inn egne seksjoner som beskriver arbeidsmetoder innad i gruppen, samt konflikthåndtering og rollefordeling. Gruppen har arbeidet med teamkontrakt sammen tidligere, og alle mener det legger til rette for et godt samarbeid og en strukturert arbeidsprosess. Teamkontrakten finnes som vedlegg under seksjon [10.1](#).

### 2.2 Arbeidsmetodikk

Gruppen kom til en enighet om at arbeidsmetodikken som passet best for oss og prosjektet ville være smidige metoder, nærmere bestemt scrum. Dette innebærer blant annet at arbeidsprosessen blir delt opp i kortere sprinter med delmål, hvor man etter hver periode gjennomfører en sprint review. Her vil man overveie sprinten som har blitt fullført, for å kunne gjøre eventuelle endringer som forbedrer den kommende. Rammeverket for scrum er fleksibelt og enkelt å tilpasse, både som gruppe og for den enkeltes arbeidsoppgaver, og fremstår som det beste valget for oss i dette prosjektet.

Etter å ha vurdert den totale tiden for prosjektarbeidet, ble det fastslått å ha en total på seks sprinter som hadde en lengde på to uker hver. Denne varigheten gjorde at man hadde godt med tid til de satte delmålene, men også korte nok til at man raskt og effektivt kunne gå videre til nye arbeidsoppgaver. For hver sprint ble det forsøkt å legge en plan for perioden, med en passende mengde oppgaver og delmål. Et eksempel på hvordan oppsettet for en av sprintene kunne se ut er presentert i Figur 5 under, hvor vi har utformet et eget scrum board. Dette er en oversikt over hva slags oppgaver man har planlagt, hvilke av disse som er under arbeid og hva som har blitt utført. Det bidrar til å holde en god oversikt over innholdet i sprinten.

Sprint 2		
Uke 6-7		
Oppgaver	Under arbeid	Utført
Oppnå forståelse for ulike rammeverk	Ser på Scikit learn-biblioteket Undersøke på ulike kodeeksempler	Forstår Pandas og numpy
Bearbeide datasett	Behandling av nullverdier Anonymisere spillere	Lagde egne datasett for utvalgte spillere
Se etter korrelasjon i parametrene	Finne sammenhenger mellom registrerte Readiness-verdier og resten av parametrene	Fremstil korrelasjons-matrise

Figur 5: Et eksempel på hvordan en av sprintene kunne settes opp. Her er Sprint2 fremstilt i et scrum board.

For å få en god oversikt over frister og sprinter ble det utformet en fremdriftsplan, visualisert i et Gantt-diagram, vist i Figur 6. Dette er et stolpediagram som gir en visuell oversikt over hvilke prosjektoppgaver som er planlagt over tid.



Figur 6: Oversikt over Gantt-diagrammet laget for arbeidsprosessen til prosjektet.

Dette prosjektet er en utredningsoppgave, hvor store deler av rapporten baseres på drøfting og konklusjoner som har blitt gjort underveis. Gruppen var inneforstått med at arbeidsprosessen ville inneholde mye testing og detaljarbeid som måtte loggføres og holdes oversiktig. Derfor har vi ført individuelle prosjektdagbøker, med oversikt over hva som har blitt gjort til hvilken tid. Dette var et verdifullt verktøy som gjorde det enkelt å organisere og holde oversikt, samt reflektere over arbeidet underveis.

## 2.3 Faglige forutsetninger

Valget om å arbeide med en oppgave innenfor kunstig intelligens og maskinlæring var noe hele gruppen var bevisst på i forkant. Det var derimot tydelig at en slik type oppgave ville være mer utfordrende, fordi vi ikke hadde et stort grunnlag innenfor kunstig intelligens og maskinlæring fra før. Det vi kunne fra før kom fra et fag alle hadde felles under høstsemesteret 2021, som var introduksjon til kunstig intelligens. Her fikk vi erfare å bruke maskinlæring til å jobbe med ulike datasett, data utvinning av disse og videre trenere enkle modeller til å utføre ulike oppgaver. Dette bidro til en grunnleggende forståelse i bunn, som gjorde at gruppen har gode forutsetninger for å utføre prosjektoppgaven og samtidig øke kunnskapene innenfor temaet.

I seksjon [3.1](#) gjør vi rede for eksisterende forskning og arbeid nærliggende problemstillingen i prosjektet. Denne informasjonen har vært viktig for beslutninger som har

blitt tatt i forbindelse med fremgangsmåter for dataanalyse, hvilke maskinlærings-modeller som allerede har blitt testet på området, hvilke resultater som har blitt oppnådd og konklusjonene som har blitt gjort i ettertid. Det var ønskelig å bygge videre på arbeidet som presenteres i disse rapportene ved å utforske andre alternative fremgangsmåter som viser potensiale. Samtidig ville vi se nærmere på alternativer som hadde blitt valgt vekk på grunn av dårligere prestasjon, slik som regression som prediksionsmetode for tidsseriedata. Gruppen hadde en oppfatning av at det ville være verdifullt å forstå grunnen til dette, sidestille dette med modeller som viste bedre prestasjon og tydeliggjøre forskjellene.

## 2.4 Rammebetingelser

Det ble ikke gitt noen bestemte rammebetingelser fra oppdragsgiver når det gjaldt bruk av teknologi og rammeverk, annet enn at oppgaven løses ved hjelp av maskinlæring. På grunn av personvernrettigheter og etiske hensyn har dataen som ble tildelt for bruk blitt anonymisert før resultat har blitt presentert i rapporten. Datasettene blir ikke publisert på GitHub under vedlagt kildekode, men små utsnitt vil fortsatt være mulig å observere.

Som forklart i seksjon [1.7](#) går vårt prosjekt ut på å forsøke å predikere en spillers form i forkant av kamp og trening, basert på en parameter kalt Readiness to Play. Derfor var det nødvendig å ta en beslutning angående hva slags maskinlæringsmodell vi ville gjøre prediksjonene med. Valget er basert på en helhetsvurdering av den totale prosjekttiden, størrelsen på gruppen, og nødvendig bakgrunnsinformasjon om de ulike modellene som var relevante alternativer. Det ble besluttet å velge følgende 3 modeller: Random Forest (RF), Autoregressive Integrated Moving Average (ARIMA) og Long Short-Term Memory (LSTM). De to førstnevnte er modeller innenfor regression, og sistnevnte er den del av det som kalles Deep Learning (DL) innenfor maskinlæring. Hvordan disse fungerer og hva som skiller de fra hverandre gjennomgås i [kapittel 5](#), hvor vi også gjør rede for valget av maskinlæringsmodellene i seksjon [5.3](#).

## 2.5 Rammeverk

I denne seksjonen gis en kort innføring av programvare, språk og bibliotek som har blitt tatt i bruk under arbeidsprosessen. Det er delt inn i en tydelig oversikt over verktøy som omfatter skrive- og kjøreprosess av maskinkode, samt felles lagring for selve koden. Deretter følger en introduksjon av det brukte programmeringsspråket, og til slutt en oversikt

over de nødvendige bibliotekene som ble tatt i bruk for implementasjon av maskinlæringsalgoritmene. Bibliotekene er et resultat av valget av maskinlæringsalgoritmer, presentert i forrige seksjon [2.4](#).

## 2.5.1 Verktøy

### Jupyter Notebook

Jupyter Notebook er en nettbasert applikasjon som gir deg mulighet til å opprette og dele dokumenter.<sup>4</sup> Dokumentene kan utformes på ulike måter, og ut i fra input dataen som tilføres kan man få returnert blant annet maskinkode, tekst, grafer og resultat av matematiske formler. Denne prosessen utføres i ulike celler, og kan bygges på den måten man selv ønsker. Dette gjøre det mulig å utføre mindre seksjoner med kode om gangen, noe som resulterer i at man slipper å kjøre store mengder kode for mindre endringer som er gjort underveis. I vårt prosjekt har programmet blitt brukt til å skrive kode i Python, og visualisere og fremstille grafer i kombinasjon med beskrivende tekst paragrafer. Programmet støtter flere ulike programmeringsspråk, og tillater lettvinte implementasjoner av biblioteker. Alle i gruppen var kjent med applikasjonen på forhånd, noe som gjorde at vi bestemte for å arbeide videre med programmet under prosjekttiden.

### GitHub

GitHub er en plattform som tilbyr versjonskontroll og samarbeid mellom flere parter.<sup>5</sup> Programmet har blitt brukt av alle i gruppen siden vi startet på studiet, og var et essensielt verktøy under prosjektarbeidet. Ved å ta i bruk GitHub sikrer vi at alle har tilgang til oppdatert kode til enhver tid, og det gjør det enklere for oss å samarbeide, teste og vedlikeholde. Bruk av GitHub gir også en ekstra trygghet med tanke på lagring av kode, og sørger for at arbeid ikke går tapt.

GitHub repositoryet er tilgjengelig som åpen kildekode. På grunn av nødvendige begrensninger er det ikke ønskelig fra oppdragsgiver sin side at datasettene brukt i dette prosjektet er lagt ved. Siste opplasting på GitHub viser derimot hvordan koden så ut ved siste kjøring, og det har blitt godkjent fra oppdragsgiver at utsnitt fra datasett kan vises. Om det blir forsøkt å kjøre koden på nytt, vil man ikke få en brukbar output på grunn av manglende datasett.

---

<sup>4</sup> <https://docs.jupyter.org/en/latest/projects/architecture/content-architecture.html>

<sup>5</sup> <https://github.com/>

Lenke: <https://github.com/bachelor22gruppe9/Bachelor---ReadinessPrediksjon>

## Miro

Miro er en applikasjon som tilbyr utforming av blant annet tabeller, figurer og diagrammer.<sup>6</sup> For å tydeliggjøre enkelte deler av rapporten og informasjonen som blir presenterte, valgte vi Miro som verktøy til dette formålet. I vårt arbeid har det blitt brukt til å beskrive grafer og visualiseringer gjort gjennom Jupyter Notebook, fremstille fakta i tabeller og utvikle enkle venn-diagram og figurer.

## Slack

Sammen med veilederne har det blitt besluttet å ta i bruk Slack som en ekstra kommunikasjonskanal, i tillegg til jevnlige zoom-møter.<sup>7</sup> Her hadde vi mulighet til å stille spørsmål og eventuelt dele dokument/kode, hvor vi raskt kunne få nødvendig respons og hjelp. Det er også nyttig og verdifullt å ha en kanal hvor man har god oversikt over tidligere kommunikasjon og samtale, og lett kan aksessere informasjon som er delt ved tidligere anledninger. Gruppen har også opprettet en egen privat kanal innenfor Slack, hvor vi kan dele privat.

## Zoom

Zoom er en applikasjon som tilbyr lyd- og videokommunikasjon med et gitt antall deltakere, og vil bli brukt for jevnlige oppdateringer med veilederne i arbeidsprosessen.<sup>8</sup> Alle i gruppen har brukt Zoom aktivt under studieløpet, og dette har vært hovedkanalen mesteparten av forelesningene på skolen har blitt gjennomført digitalt. Siden dette verktøyet er kjent for alle involverte i arbeidsprosessen er det naturlig å ta dette i bruk som en fast kommunikasjonskanal.

---

<sup>6</sup> <https://help.miro.com/hc/en-us/articles/360017730533-What-is-Miro->

<sup>7</sup>

[https://slack.com/intl/en-gb/what-is-slack?utm\\_medium=ppc&utm\\_source=google&utm\\_campaign=ppc\\_google\\_nordics\\_brand&utm\\_term=slack%20messenger&campaign=7013a000001ruBMAAY&qclid=Cj0KCQjwhLKUBhDiARIsAMaTLnEnF8KTSc28yMNjbw3Ff4vLb\\_Vap3wH-ah-IICqdreTY7CRaKSwhEaApfYEALw\\_wcB&qclsrc=aw.ds](https://slack.com/intl/en-gb/what-is-slack?utm_medium=ppc&utm_source=google&utm_campaign=ppc_google_nordics_brand&utm_term=slack%20messenger&campaign=7013a000001ruBMAAY&qclid=Cj0KCQjwhLKUBhDiARIsAMaTLnEnF8KTSc28yMNjbw3Ff4vLb_Vap3wH-ah-IICqdreTY7CRaKSwhEaApfYEALw_wcB&qclsrc=aw.ds)

<sup>8</sup> <https://explore.zoom.us/en/about/>

## 2.5.2 Programmeringsspråk

### Python

Python (versjon 3.10.2) er et høynivå objektorientert programmeringsspråk som tillater polymorfisme og har enkle syntakser.<sup>9</sup> Språket har en rekke rammer og biblioteker egnet for maskinlæringsalgoritmer som blant annet Pandas, Numpy, Scikit-learn. Disse er nevnt i seksjon [2.5.3](#). Python tilrettelegger for implementasjon av ulike matematiske formler og notasjoner gjennom biblioteker som hjelper til å organisere og håndtere data, løse avanserte utregninger og lage modeller som kan løse komplekse problemer.

## 2.5.3 Bibliotek

### Pandas

Pandas er et bibliotek basert på Numpy, og brukes i forbindelse med bearbeiding og analyse av data innen maskinlæring.<sup>10</sup> Biblioteket fungerer godt sammen med andre moduler innenfor Python, og gir deg mulighet til å bearbeide dataen gjennom visualisering og analysering. Manipulasjon av numeriske tabeller og tidsserier er også oppgaver som enkelt gjennomføres ved hjelp av dette biblioteket. Dette biblioteket ble tatt i bruk for databehandling gjennomgående i prosjektet.

### Numpy

Numpy, sammensatt av «Numerical Python», er en modul i Python som brukes til å arbeide med og bruke multidimensjonale arrays og matriser.<sup>11</sup> Fordelen med å bruke Numpy at det støtter n-dimensjonale arrays, som i vanlig Python kun kan være homogene. At et array er homogent vil si at alle elementer i arrayet må være av samme type. Bruken av Numpy i Python resulterer i tilnærmet lik funksjonalitet som MATLAB.

### Scikit-learn

Scikit-learn er et gratis maskinlæringsbibliotek for Python, basert på NumPy, SciPy og matplotlib<sup>12</sup>. Scikit-learn er brukt for prediktiv data analyse og er kjent for å tilby lettvinde implementasjoner av flere kjente maskinlæringsalgoritmer, som blant annet classification, regression, clustering og RF.

---

<sup>9</sup> <https://www.python.org/doc/essays/blurb/>

<sup>10</sup> [https://pandas.pydata.org/docs/user\\_guide/index.html](https://pandas.pydata.org/docs/user_guide/index.html)

<sup>11</sup> <https://numpy.org/doc/stable/>

<sup>12</sup> <https://scikit-learn.org/stable/index.html>

## Skforecast

Skforecast er et bibliotek i Python som brukes til prediksjon av tidsserier.<sup>13</sup> Biblioteket tilbyr en samling av klasser og funksjoner som kan kombineres med regression-modeller fra Scikit-learn. Denne kombinasjonen resulterer i modeller som kan brukes til prediksjon av tidsserier.

## Statsmodels

Statsmodels er en Python module, som gir ulike klasser og funksjoner for å estimere ulike statistiske modeller, gjennomføre statistiske tester og utforskning av ulike statistiske data. En omfattende liste over resultatstatistikk er tilgjengelig for hver estimator. Resultatene testes mot eksisterende statistiske pakker for å sikre at de er korrekte.<sup>14</sup>

## Pmdarima

Pmdarima er et statistisk bibliotek som fungerer som et tillegg til Python sin kapabilitet til å analysere tidsserier, og tilbyr mye funksjonalitet i arbeid med ARIMA-modeller.<sup>15</sup> I dette biblioteket finner man metoder som brukes til å teste stasjonaritet og eventuelle sesongmessige forhold i datasettet, i tillegg til funksjoner innenfor differensiering. Biblioteket bygger på statsmodels-modulen, og tilbyr et grensesnitt som ligner på scikit-learn.

## Keras/Tensorflow

Keras er programmeringsgrensesnitt innenfor DL skrevet i Python, og kan kjøres over plattformene TensorFlow og Theano.<sup>16</sup> Selve grensesnittet fokuserer på å tilby raske og brukervennlige fremgangsmåter for eksperimentering av DL algoritmer, og gjør det til en kraftfull plattform.<sup>17</sup>

---

<sup>13</sup> <https://joaquinamatreroigo.github.io/skforecast/0.4.3/index.html>

<sup>14</sup> <https://www.statsmodels.org/stable/index.html>

<sup>15</sup> <https://pypi.org/project/pmdarima/>

<sup>16</sup> <https://keras.io/about/>

<sup>17</sup> <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>

## 3 Bakgrunn og tidligere arbeid

For å forstå arbeidet gjort i denne rapporten er det viktig med en innføring av selve teorien og bakgrunnen for problemstillingen. I dette kapittelet vil vi gå gjennom to publiserte artikler og en masteroppgave som har blitt publisert, hvor alle tar for seg arbeid med prediksjon og maskinlæring. Det vil også bli gitt en generell introduksjon til maskinlæring, samt de mest relevante underkategoriene av læringsmetoder og algoritmer. Her vil vi forklare de viktigste forskjellene mellom disse og legge til rette for en forståelse av bruksområdene de har. Videre gjør vi rede for tidsserier og forklarer hvordan maskinlæring kan brukes til å analysere denne typen data. Målet med denne seksjonen er å gjøre rede for tidligere arbeid som har stått i fokus under vår arbeidsprosess, og å gi leseren et godt grunnlag for forståelse rundt diskusjon og utredning som følger i rapporten.

### 3.1 Tidligere arbeid

Det eksisterer allerede rapporter og artikler som er basert på innsamlet data fra applikasjonen pmSys. Før igangsettingen av vårt prosjekt var det nyttig å ha innsikt i arbeid som allerede er gjort rundt lignende problemstillinger, spesielt med tanke på at vårt prosjekt er forskningsbasert.

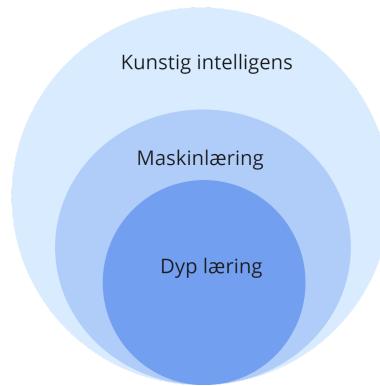
Wiik et al. (2019) har brukt LSTM og Random Forest (RF) for å trenne en modell med innsamlet data fra pmSys. Denne dataen ble loggført av spillerne fra to profesjonelle fotballag i Norge. I rapporten viser de til en modell som har oppnådd en nøyaktighet på 90% når de ser etter topptreninger og bunner i predikerte Readiness-verdier. Dette er et lovende resultat, som tilsier at modellen potensielt kan bidra til å gi trenere en bedre innsikt i oppsett av treningsøkter for lag spesielt. Denne rapporten er et spesielt godt utgangspunkt for gruppens arbeid, siden vår problemstilling i blant annet bygger på det samme hovedfokuset som Wiik et al. Samtidig presenterer artikkelen relevante læringsmetoder og modeller når det kommer til maskinlæring, som tilfører et godt grunnlag for oss å jobbe videre ut i fra.

Johansen et al. (2020) har gått nærmere inn på utviklingen av pmSys som applikasjon og beskriver forsøk på å få bedre innsikt i innsamlet data. Forbedring og videreutvikling av teknologi som gagner utøvere i toppidretten er svært attraktivt. Nødvendigheten for store datamengder fører med seg avansert og tidkrevende arbeid, og understreker behovet for teknologi som gjennomfører dette arbeidet på egenhånd. Rapporten inneholder beskrivelser av aspekt innenfor backend, frontend og skybasert støtte for tjenesten.

Kulakou (2021) har skrevet en masteroppgave om maskinlæringsanalyse og observasjon av utøverdata. Innholdet beskriver forsøk med testing på velværedata fra atleter som er relevant å ta i betraktning under arbeidet med vår oppgave. I masteroppgaven presenterte Kulakou (2021) biblioteket «Tsai», som er av høy kvalitet når det kommer til Deep Learning (DL) og analysering av tidsserier . I utgangspunktet var det planlagt for oss å implementere og ta i bruk dette i en av våre modeller, men vi innså at biblioteket var mer avansert enn det var behov for i vårt arbeid. Det fremstår likevel som et bibliotek med mye potensiale og viser lovende resultat for videre forskning.

## 3.2 Introduksjon til maskinlæring

Kunstig intelligens definerer et hvert forsøk på å implementere menneskelig intelligens i maskiner og teknologi. Maskinlæring er en underkategori av kunstig intelligens, som prøver å oppnå dette ved hjelp av ulike metoder, fremgangsmåter og algoritmer. Om AI er definisjonen, er maskinlæring verktøyet. Dette forholdet er representert i Figur 7. I dagens samfunn kan man se kunstig intelligens tydelig implementert i selvkjørende biler, filtrering av epost og ansikts- og stemmegjenkjenning (Belyadi & Haghagh, 2021, s. 1). Det er også grunnen til at vi ofte ser «vi tror dette hadde passet for deg»-overskrifter når vi handler og surfer på internett.

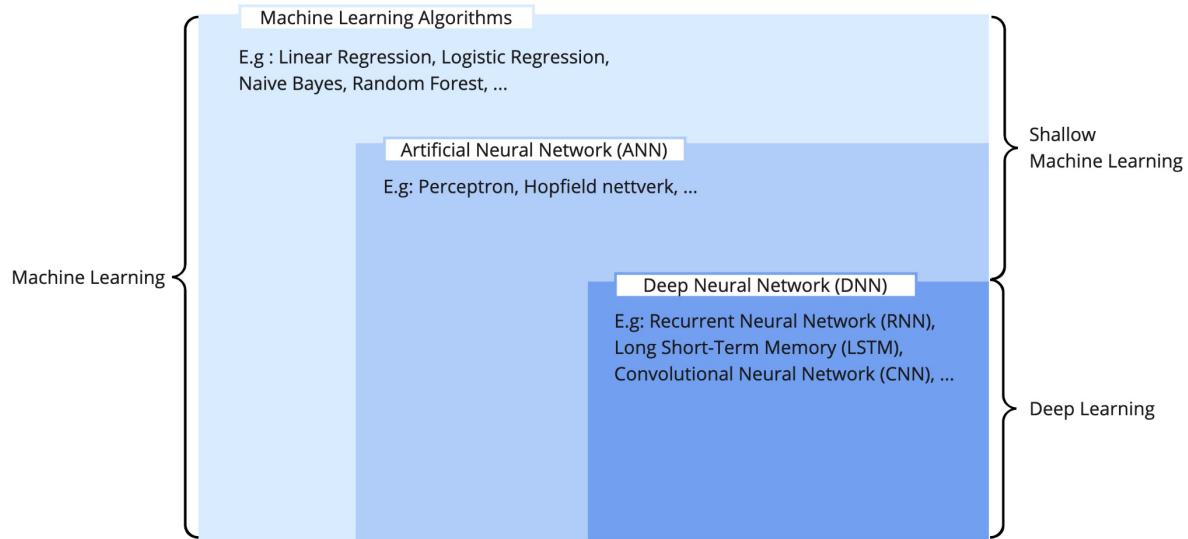


*Figur 7: Venndiagram for kunstig intelligens, maskinlæring og dyp læring.<sup>18</sup>*

Akkurat som maskinlæring er en underkategori av kunstig intelligens, har maskinlæring igjen egne underkategorier. I Figur 8 under er det presentert et venndiagram, som fremstiller de viktigste underkategoriene i maskinlæring med eksempler på ulike

<sup>18</sup><https://lotuslabs.medium.com/clarifying-ai-machine-learning-deep-learning-data-science-with-venn-diagrams-c94198faa063>

algoritmer. I de to følgende underseksjonene vil vi gå nærmere inn på tradisjonell maskinlæring og deretter dyp læring.



*Figur 8: Venndiagram som viser de ulike lagene innenfor maskinlæring (inspirert av Goodfellow et al., 2016, s.9 og Janiesch et al., 2021, s. 687).*

### 3.2.1 Tradisjonell maskinlæring

Edgar og Manz (2017) beskriver at det er to dimensjoner som omfatter hvordan maskinlæring kategoriseres: prosessen hvor modellen lærer og hva slags type resultat eller problem som skal løses. Den første kategorien, som består av treningsmetoder, deles hovedsakelig inn i supervised learning og unsupervised learning. Man har også semi-supervised learning og reinforcement learning, men disse kategoriene vil vi ikke gå nærmere inn på i denne rapporten. I supervised learning får algoritmen tildelt både input og output data som er merket på forhånd. Modellen skal predikere output og får kontinuerlig tilbakemelding på om resultatet stemmer, nettopp fordi output dataen er tildelt på forhånd. Hovedmålet er at algoritmen skal trenes til å kunne gi nøyaktige prediksjoner når den blir tildelt ny data. I unsupervised learning får maskinlæringsmodellen kun tildelt input data som ikke er merket, slik at modellen selv må oppdage mønstrene i dataen, og lære av dette. De viktigste forskjellene mellom disse to metodene er listet opp i Figur 9 under.

Supervised learning	Unsupervised learning
Algoritmene trenes ved bruk av merket data.	Algoritmene trenes ved bruk av umerket data.
Modellen får kontinuerlig tilbakemelding for å verifisere om prediksjon av output data er korrekt.	Modellen får ikke tilbakemelding på analyser den gjør.
Modellen får tildelt data både i input og output.	Modellen får kun data som input.
Overordnet mål er å trenne modellen til å gi nøyaktige prediksjoner når den får tildelt ny inndata.	Overordnet mål er å finne underliggende mønster og opparbeide seg innsikt fra et ukjent datasett.
Treningen må overvåkes.	Treningen trenger ikke å overvåkes.
Bruksområde: Problemstillinger hvor man har tilgang til input-data i tillegg til output-data.	Bruksområde: Problemstillinger hvor man kun har tilgang til input-data.
Modellen kan produsere nøyaktige resultat.	Modellen produserer mindre nøyaktige resultat sammenlignet med supervised learning.

Figur 9: Oversikt over de viktigste forskjellene mellom supervised og unsupervised learning.<sup>19</sup>

Det finnes ulike underkategorier av algoritmer innenfor hver enkelt læringsmetode, som blir valgt på grunnlag av hva slags type problem som skal løses og dataen man har. I supervised learning har man blant annet classification og regression, i unsupervised learning har man clustering. I dette prosjektet har vi tilgang til både input og output data, samtidig som målet er å trenne en modell til å gjøre prediksjoner. Det ble derfor riktig å gå videre med supervised learning som læringsmetode, og videre utforske algoritmer innenfor denne kategorien som viste potensiale for vårt arbeid.

De mest relevante kategoriene innenfor supervised learning i maskinlæring er classification og regression.<sup>20</sup> I algoritmer innenfor classification må modellen trekke en konklusjon ut i fra observerte verdier for å vurdere hvilken kategori de tilhører. De observerte datapunktene blir deretter fordelt i to eller flere klasser, basert på den mest signifikante

<sup>19</sup> <https://www.javatpoint.com/difference-between-supervised-and-unsupervised-learning>

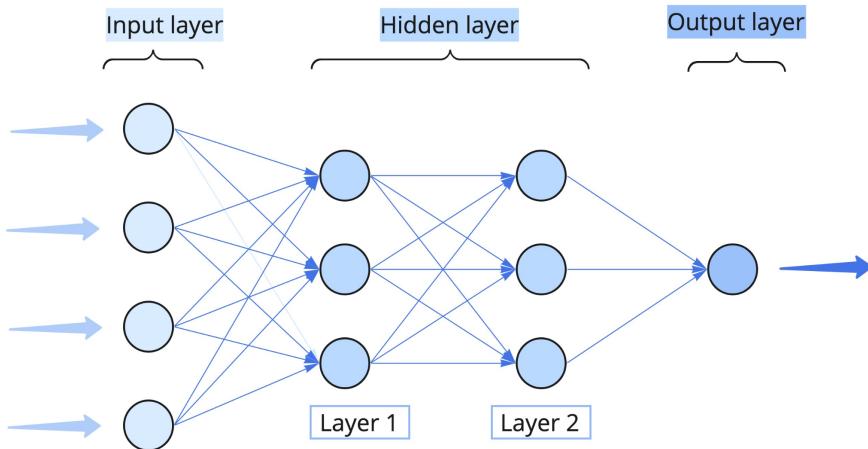
<sup>20</sup> <https://azure.microsoft.com/nb-no/overview/machine-learning-algorithms/#uses>

differensiatoren mellom observasjonene. Et eksempel på classification er filtrering av epost, hvor et program analyserer innholdet i en epost for så plassere den i kategorien «spam» eller «ikke spam». I regression-algoritmer må modellen trenes til å forstå relasjonen mellom ulike variabler, og beregne et estimat. Algoritmen lærer av historiske loggførte verdier fra input data og vil ut i fra dette gi en prediksjon på fremtidig output data. Regression brukes i problemstillinger hvor det er nødvendig å forutsi fremtidige verdier. Dette gjør det til en nyttig metode i prediksionsoppgaver, slik som problemstillingen i vårt prosjekt.

### 3.2.2 Deep Learning

Slik det er fremstilt i Figur 8 i seksjon [3.2](#) har man to viktige underkategorier av maskinlæring; Artificial Neural Network (ANN) og Deep Neural Network (DNN). ANN forsøker gjennom ulike algoritmer å etterligne hvordan de biologiske nervecellene i en hjerne er organisert på.<sup>21</sup> Nettverkene utgjør en mengde lag, bygd opp av sammenkoblede nerveceller. Her eksisterer det prosesserende enheter, også kalt nevroner, som det konstant går signaler mellom. Denne måten å transportere informasjon på etterlignes i konstruksjonen av ANN, hvor man på samme måte sender informasjon mellom et stort antall tilknyttede noder.

En ANN-modell består av tre lag: input layer, hidden layer og output layer. I Figur 10 ser man et eksempel på hvordan strukturen i et slikt nettverk kan konstrueres.<sup>22</sup>



Figur 10: Et eksempel på hvordan strukturen til et Artificial Neural Network (ANN) kan se ut.

<sup>21</sup> [https://snl.no/nevralt\\_nettverk](https://snl.no/nevralt_nettverk)

<sup>22</sup> <https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks>

Hver enkelt sirkel representerer et nevron og strekene mellom disse er koblingen de har til hverandre. Totalen av alle disse bindeleddene blir til et Neural Network (NN) som prosesserer informasjon. Det første laget består av noder som får tilført selve dataen og videresender dette til de skjulte lagene. Hver kobling mellom de ulike lagene har en egen vektlagt verdi. Når nodene i de skjulte lagene mottar informasjon bestemmer vekten av koblingen hvordan verdien kalkuleres og sendes videre. Verdien til påfølgende lag blir derfor påvirket av de tidligere lagene, en prosess som foregår helt til informasjonen ender opp i output layer med det endelige resultatet.

I Figur 10 er det et relativt lite og grunt nettverk som er representert. Et dypere nettverk vil vise en kombinasjon av flere antall noder og lag i hvert ledd. Dette medfører en samling av nettverk med ulik dybde og dermed hensikt. Eksempler på dette er Recurrent Neural Network (RNN), Convolutional Neural Network (CNN) og Deep Neural Network (DNN). Innenfor disse har man ulike arkitekturen i forbindelse med oppsett av modell, hvor vi i denne rapporten blant annet vil se nærmere på Long Short-Term Memory (LSTM) som er basert på RNN.<sup>23</sup>

### 3.3 Vårt utgangspunkt

Basert på informasjonen presentert i dette kapittelet har vi et tydelig springbrett for inngangen til vårt arbeid. Vår problemstilling tar utgangspunkt i konklusjonene presentert i artiklene til Wiik et al. og Johansen et al, som er beskrevet i seksjon [3.1](#). I seksjon [2.4](#) introduserer vi de tre maskinlæringsmodellene som skal brukes i vår rapport, og to av disse ble også testet av Wiik et al.; RF og LSTM. Sammen med modellen Autoregressive Integrated Moving Average vil disse analyseres gjennom eksperimentering med prediksjon av tidsserier. I seksjon [1.6](#) ble det beskrevet at vi vil forsøke å predikere variabelen Readiness to Play, en verdi som forteller oss om en fotballspillers opplevde form i forkant av kamp og trening.

Før vi går nærmere inn på maskinlæringsalgoritmene i kapittel [5](#) og selve eksperimentene med prediksjon som har blitt gjennomført i kapittel [6](#), vil vi gjøre rede for datasettene som skal brukes. I kapittel [4](#) vil vi gjennomgå akkurat dette; hva settene består av, hvordan de har blitt samlet inn, og hvordan dataene har blitt klargjort for predikering.

---

<sup>23</sup> <https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e>

## 4 Innsamling og bearbeiding av data

Ved kontinuerlig bruk av applikasjonen pmSys har fotballspillere bidratt til å samle inn informasjon om deres nåværende helsetilstand, generell velvære og eventuelle skader. Den innsamlede dataen vi har fått tildelt viser disse detaljene om fotballspillere fra to lag, i en tidsperiode i underkant av to år. Datasettene kan videre omtales som tidsserier, et begrep som er forklart i neste seksjon [4.1.1](#). I vår rapport blir denne dataen brukt til å gjøre prediksjoner, med hensikt å bidra til skadeforebygging og forbedre den totale prestasjonsevnen til spillerne. I de kommende seksjonene vil vi gi en introduksjon til datasettene vi har brukt, hva de inneholder og hvordan de har blitt preparert for maskinlæringsarbeid.

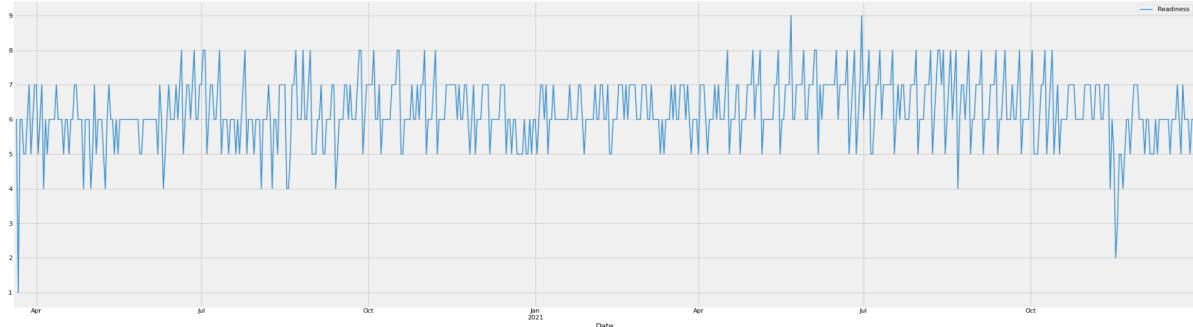
### 4.1 Datasett

Et datasett er en samling av rådata med en spesifikk struktur, hentet inn fra en konkret kilde (Kotu, V. & Deshpande, B., 2019, s. 23). Innholdet består av ulike datapunkter fordelt ut over rader, og innenfor en bestemt attributt i en kolonne. En attributt definerer navnet til verdien datapunktet representerer. Verdiene til datapunktene kan være for eksempel numeriske, boolske, et tidspunkt eller tekst. En oversikt over passasjerer på et fly som inneholder informasjon om valgt sete, hvor mye bagasje de har, hva slags klasse de reiser på i tillegg til personlig informasjon som navn, telefonnummer og epost er et eksempel på et datasett. Ofte registreres slik informasjon med et tidsstempel som en viktig ID, noe som gjør at man kan samle opp utallige tidsstempler med all ovennevnte informasjon i hver loggføring. Dette tidsstemplet er inngangen til datatypen som beskrives som tidsserie, og er datatypen vi arbeider med i vårt prosjekt.

#### 4.1.1 Tidsserier

Data av typen tidsserie er en samling av observasjoner som er loggført ved ulike tidsintervall over en gitt periode, og det er et slikt datasett vi arbeider med i oppgaven. Denne typen data består av gitte parametere hvor informasjonen samles inn suksessivt, noe som bidrar til en oversikt over endringer og mønster som oppstår i den gitte perioden. Et visuelt plott av dataen vil som regel vise tiden som har passert på x-aksen og parameterverdiene på y-aksen. I Figur 11 ser man et eksempel på et plott med innsamlet

data fra pmSys, her fremstilt med parameteren readiness på y-aksen og, og tiden på x-aksen.



*Figur 11: Eksempel på fremstilling av tidsseriedata grafisk, hvor man kan se readiness-parameteren loggført over 651 dager.*

Terence C. Mills (2019) gir en konkret forklaring på tidsserier; sett at man har en variabel  $x$  som får sin verdi etter tiden  $t$  som har passert;  $x_t$ . Setter vi  $t = 1$  vil dette indikere den første observasjonen som er loggført, og videre kan vi sette  $t = 1, 2, 3, 4 \dots n$ . Der  $n$  representerer selve observasjonsperioden (Mills, 2019, s.1). Dermed kan man lese av  $y$ -verdien for å finne den loggførte verdien av variabelen  $x_t$  manuelt.

#### 4.1.2 Informasjon om dataen

Som nevnt i seksjon [4.1.1](#), er våre datasett tidsserier. Dette er på grunn av at informasjonen som er registrert har skjedd med en daglig frekvens i tidsperioden 21.03.2020 til 31.12.2021. De innsamlede parametrerne som gir informasjon om generell være er følgende; Fatigue, Mood, Readiness, Sleep Duration, Sleep Quality, Sorenness og Stress. De fleste parametrerne registreres med tall fra 1 til 5, bortsett fra Readiness som registreres med verdier fra 1-10 og Sleep Duration som registreres med antall timer søvn. For alle parametrerne representerer høye verdier en god opplevd form og lave verdier en negativ opplevd form. I Figur 12 ser vi et eksempel på hvordan datasettet til en av spillerne ser ut, før noe bearbeiding har blitt gjort.

	Date	Fatigue	Mood	Readiness	Sleep Durhation	Sleep Quality	Soreness	Stress
0	2020-03-20	3.0	2.0	6.0	9.0	3.0	2.0	3.0
1	2020-03-21	3.0	3.0	6.0	10.0	4.0	2.0	3.0
2	2020-03-22	2.0	3.0	1.0	10.0	3.0	2.0	3.0
3	2020-03-23	3.0	3.0	6.0	9.0	4.0	2.0	3.0
4	2020-03-24	2.0	2.0	6.0	9.0	4.0	2.0	3.0
...	...	...	...	...	...	...	...	...
647	2021-12-27	3.0	3.0	6.0	8.0	2.0	2.0	3.0
648	2021-12-28	3.0	2.0	5.0	9.0	3.0	2.0	3.0
649	2021-12-29	3.0	3.0	6.0	8.5	2.0	2.0	3.0
650	2021-12-30	3.0	3.0	6.0	8.0	2.0	2.0	3.0
651	2021-12-31	2.0	3.0	7.0	7.5	2.0	2.0	3.0

652 rows x 8 columns

Figur 12: Eksempel på hvordan datasettet til en av spillerne ser ut, før bearbeiding. Her er kun registrerte parametere innenfor generell velvære presentert.

For enkelte spillere er det også loggført informasjon om oppstått sykdom, eventuelle skader og prestasjon på kamp. Med unntaket av datasettet om skade, har ikke disse vært relevant å bruke i vår rapport og blir på grunn av dette ikke fremstilt i oppgaven.

I seksjon [1.6](#) med informasjon om problemstillingen for prosjektet, understreker vi at vi kommer til å arbeide med prediksjon av kun én av parametrene fra Figur 12 over; Readiness. Dette er en variabel som forteller om hvor god formen til en spiller oppleves i forkant av kamp eller trening, og derfor mest relevant å gjøre prediksjoner med. Dette innebærer arbeid basert på såkalt univariate training, en spesifikk metode hvor man kun tar i bruk én variabel under arbeid med analyse og prediksjon. Metoder hvor man tester modeller ved å bruke et flertall parametere kalles for multivariate training, og ble testet ut under eksperimentering med prediksjon av skade. Dette presenteres i seksjon [6.9](#).

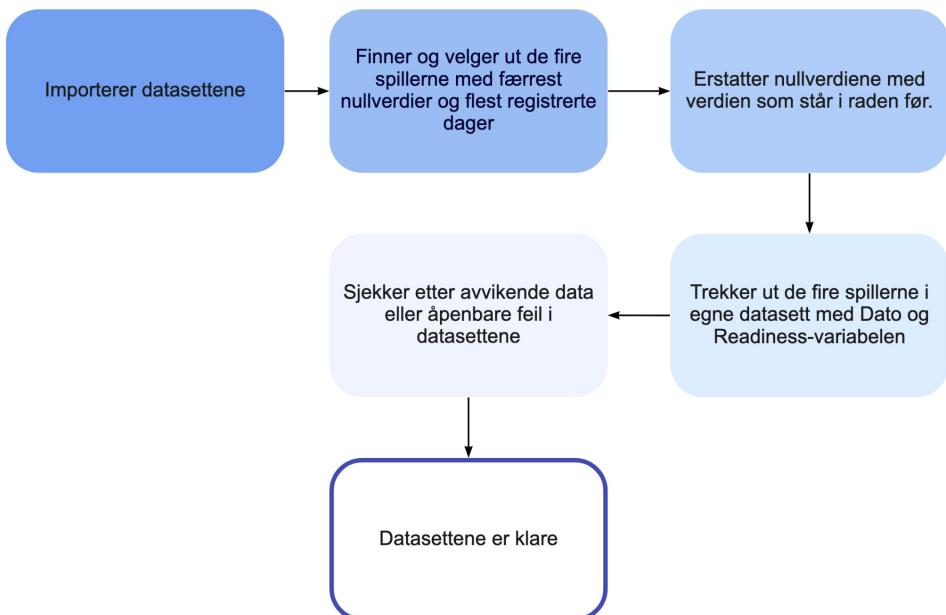
Vi tok en avgjørelse om å trenere modellene på kun én spiller og et datasett av gangen, versus hele laget, for å se om dette var nok til å få en nøyaktig prediksjon. For å sikre at modellene ble testet med varierte datasett, valgte vi ut de fire spillerne med høyest antall registrerte dager og færrest antall nullverdier. For å ta hensyn til personvern har spillerne blitt anonymisert, og vi vil heretter referere til de som Spiller1, Spiller2, Spiller3 og Spiller4. Datasettet til hver enkelt består av registrerte datapunkter fra 21.03.2020 til 31.12.2021, som resulterer i totalt 651 rader med data hver. I Figur 13 er de fire datasettene slått sammen, hvor man kan se de fem første og siste registrerte verdiene.

Date	Spiller1	Spiller2	Spiller3	Spiller4
2020-03-21	6.0	6.0	6.0	5.0
2020-03-22	6.0	1.0	3.0	6.0
2020-03-23	7.0	6.0	7.0	6.0
2020-03-24	6.0	6.0	5.0	7.0
2020-03-25	8.0	5.0	7.0	7.0
...	...	...	...	...
2021-12-27	3.0	6.0	3.0	9.0
2021-12-28	4.0	5.0	3.0	9.0
2021-12-29	5.0	6.0	4.0	8.0
2021-12-30	5.0	6.0	4.0	9.0
2021-12-31	5.0	7.0	3.0	8.0

651 rows × 4 columns

Figur 13: Oversikt som viser sammenslåingen av de fire spillernes datasett med registrerte verdier for Readiness parameteren.

I Figur 14 har vi fremstilt de ulike stegene i prosessen med å klargjøre datasettene, basert på beslutningene gjort rede for i denne seksjonen. Videre i dette kapittelet vil vi beskrive innholdet i de ulike stegene, og presentere de endelige datasettene.



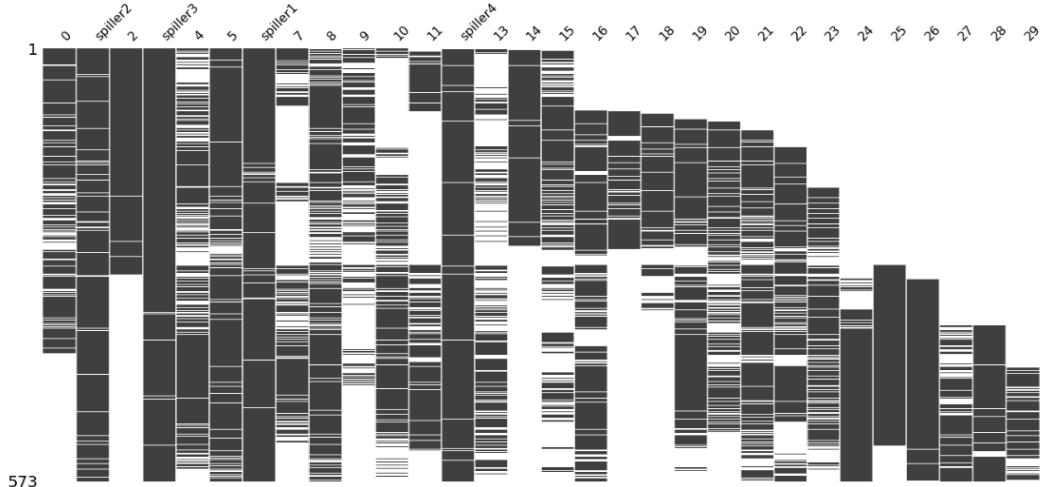
Figur 14: Oversikt over de ulike stegene under prosessen med å klargjøre datasettene.

### 4.1.3 Manglende verdier

Etter å ha importert datasettene, var første steg å finne spillerne med flest registrerte datapunkter, og da også færrest manglende verdier. Det er naturlig at det mangler data i enkelte perioder når man benytter seg av datasett satt sammen av personlig loggført data. I og med at loggføringen skal skje på daglig basis, stiller man også en forventning til at spilleren selv må prioritere å bruke tid på det. Om enkelte dager uteblir fra registreringen kan dette være på grunn av at personen ikke hadde applikasjonen tilgjengelig, vært bortreist, ikke hatt trening eller glemt å gjennomføre registreringen. Slike manglende registreringer kalles nullverdier, og i datasett representerer de udefinerte verdier oppført som NaN (Not a Number).

Håndtering av nullverdier er en essensiell del av klargjøring av data. En av grunnene til dette er fordi enkelte maskinlæringsalgoritmer ikke klarer å kjøre når det er nullverdier til stede i datasettet, så man må manuelt håndtere disse. Metoden man velger til dette formålet må være et bevisst valg, spesielt siden nye tilførte verdier vil være med på å påvirke utfallet av sluttprediksjonen. Derfor var det gunstig å bruke datasett med få nullverdier fra starten av, slik at påvirket den ekte loggføringen i minst mulig grad.

I våre datasett varierte antallet fra spiller til spiller, som man ser i visualiseringen av nullverdiene i Figur 15. Hver spiller har datasettet fremstilt som en søyle, bestående av mange små streker som representerer tidssteg med registrert data. Om det oppstår hvite streker og hull i søylene, er dette fordi det er nullverdier i datasettet. Studerer man figuren ser man at Spiller1, Spiller2, Spiller3 og Spiller4 har høyest antall registrerte dager, og videre lavest forekomst av hvite hull.



*Figur 15: Visualisering av datasettene til et av fotballagene basert på antall registrerte nullverdier. Flere registrerte dager gir større sorte felt, og hvite streker og hull representerer nullverdier.*

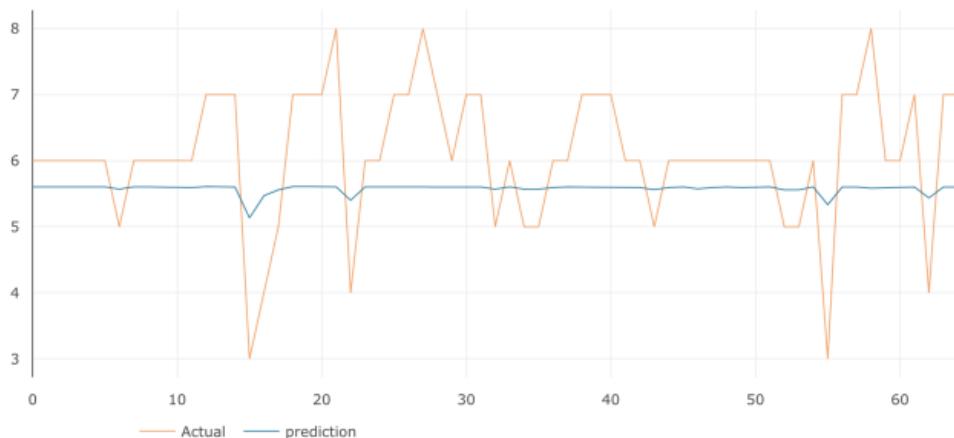
Basert på oversikt over totale nullverdier i datasettene, besluttet vi å arbeide med følgende fire spillere:

- Spiller1 med 20 nullverdier
- Spiller2 med 35 nullverdier
- Spiller3 med 6 nullverdier
- Spiller4 med 19 nullverdier

Etter å ha tatt et valg på hvilke spillere vi skulle bruke datasettene fra kunne vi bestemme hvordan nullverdiene skulle håndteres. Både Wiik et al. og Kulakou valgte erstatter de tomme datapunktene med verdien «0» for å få et realistisk scenario, et valg begrunnet med at det realistisk sett ikke er mulig å unngå nullverdier i slike datasett. Derfor bør nullverdiene inkluderes i datasettene. Andre alternativ er å fjerne radene med nullverdiene, eller fylle inn gjennomsnittsverdien beregnet fra hele datasettet. Når vi analyserte datasettene oppdaget vi tendenser til at samme verdi ble loggført over et strekk med dager. Derfor ble det besluttet å bruke en funksjon som etterfyller de tomme feltene med verdien fra foregående dag. For oss fremstår dette som en effektiv måte å håndtere nullverdier på, og fordi denne metoden ikke var blitt utforsket i tidligere arbeid bestemte vi oss for å implementere denne.

#### 4.1.4 Datamengder

Det finnes ingen fasitsvar når det gjelder hva som er en optimal mengde med data når man trener og tester maskinlæringsmodeller. Dette kan variere fra modell til modell, hvilket problem man ønsker å løse og hvilken datatype som brukes.<sup>24</sup> I Figur 16 ser man resultatene til Wiik et al. fra prediksjon av Readiness parameteren, hvor testingen ble utført med datasettet fra én spiller i en LSTM-modell. Resultatet fra modellen viser en svak antydning til predikerte topp- og bunnpunkter, men gir ikke gode nok verdier til å kunne implementeres i pmSys.



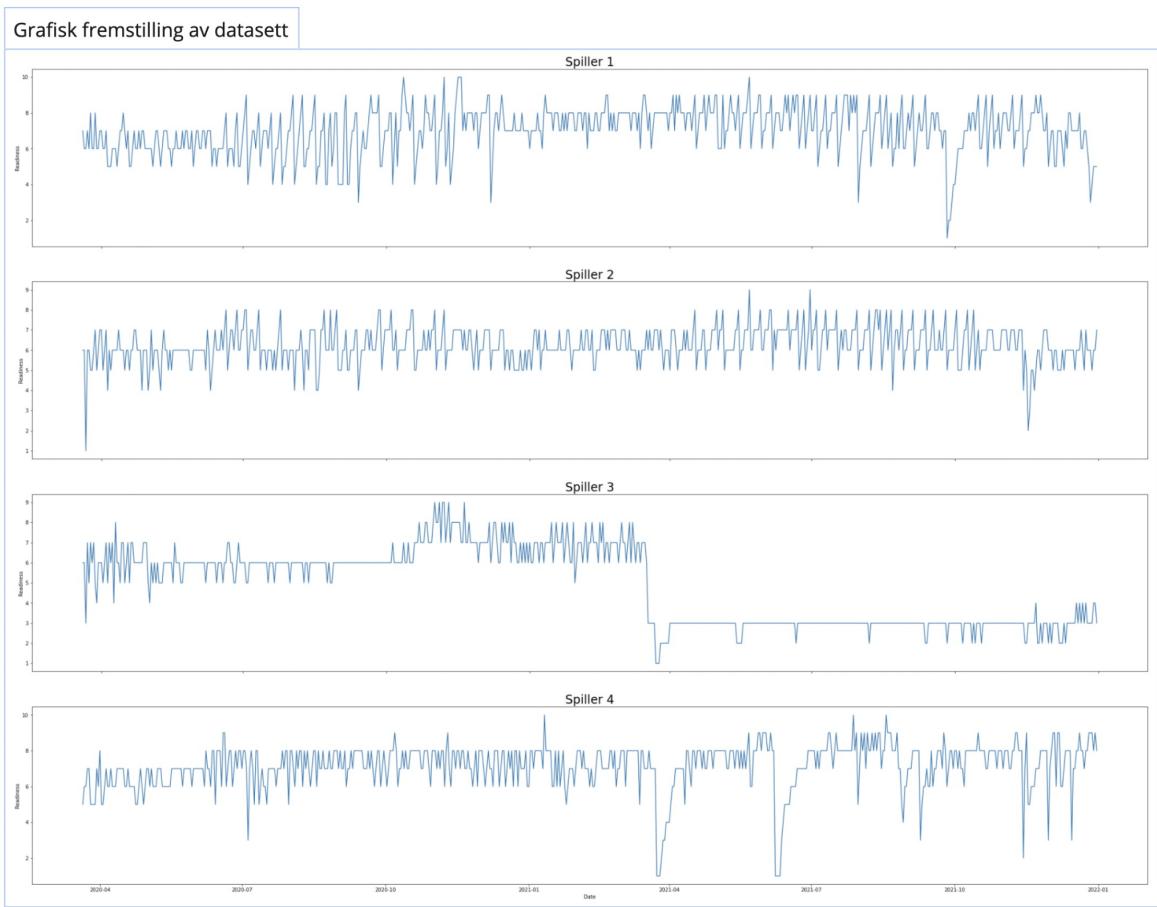
Figur 16: Grafisk fremstilling av prediksjsionsresultat for Readiness parameteren til én spiller, beregnet av en LSTM-modell presentert av Wiik et al (2019).

Når det gjelder tidsserieprediksjon vil man kunne argumentere for at et større datasett kan bidra til å få en mer nøyaktig prediksjon enn et mindre ett. Grafen fremstilt i Figur 16 viser resultatet fra en LSTM-modell med en treningsperiode på 100-200 tidssteg. Datasettene vi bruker har 651 tidssteg hver, som er nesten tre ganger så mye som hos Wiik et al. Siden vi har valgt å gjøre eksperimenter med LSTM i vår rapport, presentert i seksjon [6.6](#), vil det være interessant å se hvordan våre resultat blir med datasett som inneholder et større antall verdier. Det er viktig å bemerke at vi ikke kan sammenligne måten vi utfører eksperimentene på, og at eventuelle forskjeller i resultat kan komme av at innholdet i datasettene ikke er like. Likevel er det en interessant observasjon vi ønsker å belyse.

<sup>24</sup> <https://machinelearningmastery.com/much-training-data-required-machine-learning>

## 4.2 Ferdigstilt datasett

Etter å ha gått gjennom alle stegene fremstilt i Figur 14 i seksjon [4.1.2](#), står vi igjen med fire datasett som er input data til modellene vi skal trenne og teste i kapittel [6](#). I Figur 17 under ser man grafen til de ferdigstilte tidsseriene for de totalt fire spillerne. Med denne visualiseringen kan vi også bekrefte at verdiene ligger innenfor riktig område, som er verdier mellom 1-10 for Spiller1 og Spiller4, og 1-9 for Spiller2 og Spiller3. Grunnen til at de to sistnevnte datasettene kun har verdier opp til 9 er fordi de ikke har loggført noe høyere for Readiness-variabelen. Dersom vi hadde valgt å bruke data fra hele laget ville vi ha måttet skalere verdiene, og vise til samme verdiområde for alle spillerne. Siden vi kun skal trenne og teste på én og samme spiller om gangen, har ikke dette stor innvirkning på resultatene.



Figur 17: Grafisk fremstilling av de ferdigbehandlede datasettene til hver av de fire utvalgte spillerne.

## 4.3 Videre forberedelser

Etter å ha ferdigstilt datasettene vi skal bruke videre i prosjektet, er det en del aspekt vi må ta høyde for før vi bruker de som input i maskinlæringsalgoritmene vi skal teste. I de kommende underseksjonene vil vi gjøre rede for videre analysering av datasettene. Vi vil forklare de siste beslutningene som må tas, før dataen er klar til å brukes som input maskinlæringsmodellene som presenteres i kapittel [5](#) og videre inn i eksperimentene som gjennomføres i kapittel [6](#).

### 4.3.1 Undersøkelse om stasjonaritet

Tidsseriedata kan inneholde underliggende informasjon som det er behov for å hente ut, fordi enkelte maskinlæringsmodeller trenger input av spesifikke variabler som representerer denne informasjonen. Dette gjelder spesielt for ARIMA-modellen, som presenteres i seksjon [5.3.2](#). Analysering av dataen vil likevel, uavhengig av algoritme, bidra til en større forståelse og et bedre grunnlag for nøyaktige prediksjoner. Det første steget i analyseringen vil være å visualisere tidsserie dataen, slik det har blitt gjort i Figur 11 i seksjon [4.1.1](#). Denne fremstillingen bidrar til at man raskt kan avdekke tre konkrete aspekt; stasjonaritet, sesongmessige forhold og trender. Hva hver enkelt av disse betyr er kort forklart i Figur 18 under (Mills, 2019, s. 4-8).



Figur 18: Beskrivelse av ulike aspekt ved tidsseriedata.

Er det tydelige forekomster av enten sesongmessige forhold eller trender i tidsseriedata, er den ikke stasjonær. Om hovedmålet er å gjøre prediksjoner basert på tidsserier er det en fordel å transformere dataen slik at den oppnår stasjonaritet. Dette tilslirer at aspekt som varians og gjennomsnitt i dataen er konstante over tid, noe som bidrar til et bedre prediksionsgrunnlag. Som nevnt tidligere i denne seksjonen, er dette mest relevant når man arbeider med ARIMA-modellen og har ikke blitt fokusert på i arbeidet med Random Forest (RF) i seksjon [6.4](#) og LSTM i seksjon [6.6](#).

### 4.3.2 Test- og treningssett

Et sentralt steg i forberedelse av data er å splitte datasettet inn i såkalte test- og treningssett. Det vil si at man har en del av det komplette datasettet, testsettet, som brukes til å trenne modellen. Det andre, treningssettet, brukes til å teste modellen for å sjekke om den produserer et nøyaktig resultat. Grunnen til at man splitter dataen på denne måten er for å simulere at modellen får tilgang til ny input som den ikke har sett tidligere. Dersom man ikke gjør dette, og istedenfor trener modellen på det totale datasettet, kan algoritmen bli

overtilpasset. Det vil si at modellen ikke klarer å justere seg til ny data, fordi den i stor grad har tilpasset seg til det den har tilgjengelig.<sup>25</sup>

Ved å splitte datasettet, har man en måte å teste om modellen blir overtilpasset fordi dersom treningssettet får lave testfeil, men testsettet får høye, viser det at modellen ikke vil kunne predikere på ny input av data. På den andre siden av overtilpasning har man undertilpasning. Dette betyr at modellen ikke klarer å predikere på hverken test- eller treningssettet.

### 4.3.3 Lags og steps

En lag er et begrep innenfor maskinlæring som definerer det tidsvinduet man ser tilbake på av verdier når man skal predikere den neste. I vårt tilfelle er lags antall dager av Readiness-variabelen som maskinlæringsalgoritmene skal bruke for å forutsi den neste verdien. Under, i Figur 19, kan man se et eksempel hvor det blir brukt en lag-verdi på 5. Her brukes de første 5 verdiene, fra 1-5, til å predikere den den neste, som i dette tilfellet er den 6. verdien. Deretter forskyves antall lags en kolonne til høyre, slik at verdiene fra 2-6 predikerer den 7. Denne forskyvningen defineres som en step.

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

Figur 19: Visuell fremstilling av hvordan forskyvningen av lags ser ut i praksis.

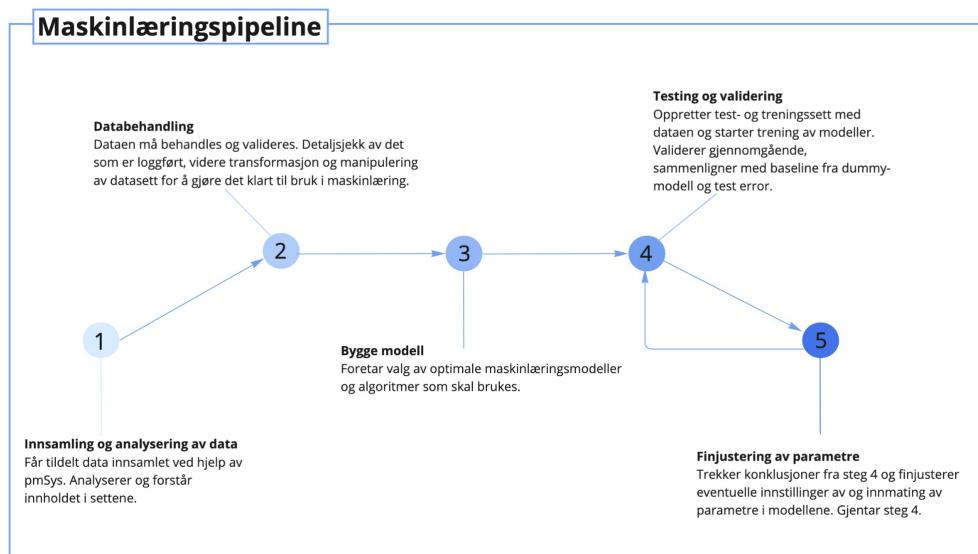
<sup>25</sup> <https://www.ibm.com/cloud/learn/overfitting>

# 5 Utvikling av maskinlæringspipeline

I forrige kapittel gjorde vi rede for dataen som skal implementeres i maskinlæringsmodellene, og hvordan den ble klargjort. Innsamling og bearbeiding av data er de første, om ikke viktigste, stegene i den totale prosessen som kan omtales som en maskinlæringspipeline. Her blir det definert hva de ulike stegene i produksjonen av en maskinlæringsmodell består av, og gjennomgående i dette kapittelet vil vi gå nærmere inn på hva de resterende stegene omfatter i vårt prosjekt. Vi vil, basert på teorien bak maskinlæring som ble presentert i kapittel 3, begrunne valgene av maskinlæringsmodellene i prosjektet og gi en kort introduksjon til disse.

## 5.1 Prosess og livssyklus

I Figur 20 presenteres et oppsett av maskinlæringspipelinens for vårt prosjekt. Et vanlig avsluttende steg i en slik prosess vil være å utplassere modellen for bruk, noe som i dette tilfellet ville ha vært som en implementasjon i pmSys-appen. I denne rapporten sitter vi ikke igjen med noe ferdig produkt som vil kunne tas i bruk, derfor er dette steget utelatt fra figuren. Det fremvises av modellen at vi har fokusert på kontinuerlig testing og finjustering av modellene vi har, ved at steg 4 og 5 har blitt gjentatt for å oppnå så nøyaktige resultat som mulig. På grunn av denne løkken vil man ikke ha en tydelig definert



Figur 20: Maskinlæringspipeline for vårt prosjekt.

ende i livssyklusen til pipelinens, fordi den gjentas til man eventuelt oppnår et ønsket resultat. Steg 3, 4 og 5 i figur 20 vil gjøres rede for gjøres rede for i kapitlene som følger.

## 5.2 Teknikker og problemstillinger

Det eksisterer ulike kategorier av problemstillinger innenfor maskinlæring. Basert på problemet som skal løses kan man velge å enten bruke mer tradisjonell maskinlæring eller Deep Learning (DL). Disse er begge gjort rede for i seksjonene [3.2.1](#) og [3.2.2](#).

Hovedforskjellen mellom disse er at DL i større grad takler mer omfattende mengder data som input, samtidig som slike modeller er mer selvstendige i gjennomføringen av enkelte oppgaver som man i tradisjonell maskinlæring må gjøre manuelt. I Figur 21 har vi presentert noen av de mer tradisjonelle algoritmene innenfor maskinlæring, sammen med en oversikt over de ulike bruksområdene til hver enkelt.

Problemstilling	Beskrivelse	Algoritmer	Eksempel
Classification	Avgjør hvilken klasse et gitt datapunkt tilhører. Valget gjøres basert på læring fra et datasett som er kjent.	Decision tree, neural network, Bayesian modeller, k-nearest neighbours, Random forest	<ul style="list-style-type: none"> <li>• Avgjøre om en epost er spam eller ikke</li> <li>• Predikere om en aksjekurs vil gå opp eller ned</li> </ul>
Regression	Prediksjon av en numerisk verdi til et gitt datapunkt. Basert på læring fra et kjent datasett.	Linear regression, logistic regression, Random forest	<ul style="list-style-type: none"> <li>• Prediksjon av temperatur</li> <li>• Indikasjon på verdien til en aksje</li> </ul>
Clustering	Identifisere naturlige grupperinger innenfor et datasett. Baseres på tydelige aspekt ved gitt datasett.	k-Means, density-based clustering (e.g., DBSCAN)	<ul style="list-style-type: none"> <li>• Segmentering av kundegrupper hos butikker og bedrifter basert på ulike kriterier</li> <li>• Inndeling av ulike bøker hos et bibliotek basert på ulike kriterier</li> </ul>
Prediksjon av tidsserie	Predikere en fremtidig verdi for en spesifik parameter basert på historiske verdier.	Regression, ARIMA	<ul style="list-style-type: none"> <li>• Prediksjon av forventet salg hos en bedrift basert på tidligere historikk, i tillegg til forventet produksjon</li> </ul>

Figur 21: Oversikt over ulike problemstillinger innenfor maskinlæring, sammen med enkel beskrivelse, eksempler på algoritmer og enkle problemer (inspirert av Kotu & Deshpande, 2019, s. 13, Table 1.1)

## 5.3 Valg av maskinlæringsmodell

Forarbeidet som har blitt gjort med tidsseredataen forklart i kapittel [4](#), i kombinasjon med prediksionsarbeid som utgangspunkt, gjorde at vi hadde tydelige kriterier til valget av maskinlæringsmodeller. Etter å ha vurdert den totale arbeidstiden til prosjektet sidestilt med at arbeidet kan deles på fire, endte vi med totalt tre modeller. Vi var enige om at det ville være hensiktsmessig å sammenligne modeller med tydelige forskjeller, blant annet når det

gjaldt kompleksitet og dybde. Det er interessant å se i hvilken grad en tradisjonell grunn maskinlæringsalgoritme presterer i forhold til en modell innenfor DL.

I seksjon [3.1](#) gjorde vi rede for konklusjoner fra tidligere arbeid, hvor enkelte modeller blir trukket frem som bedre alternativ enn andre. Wiik et al. forklarer at de har testet tradisjonelle maskinlæringsmetoder som linear regression og Random Forest (RF) på tidsseriedata, men valgt å presentere kun LSTM på grunn av bedre statistisk resultat. På bakgrunn av denne rapporten, har vi valgt begge disse modellene selv om RF og linear regression sees på som mindre optimale i gitt kontekst. Dette har vi valgt for å videre underbygge resultat fra tidligere arbeid, samt presentere konkrete tall og eksempler for å tydeliggjøre de faktiske forskjellene.

Tredje og siste modell er ARIMA, som dukket opp når vi gjorde spesifikke søk rundt prediksjoner av tidsserier.<sup>26</sup> Dette blir sett på som en enkel og effektiv modell innenfor statistisk analyse, og er mye brukt i arbeid med tidsserier. Modellen tar blant annet i bruk algoritmer basert på linear regression og gjør kalkulasjoner ved hjelp av historiske verdier. Vi mente at dette var et bra tilskudd til de to ovennevnte, da ARIMA ikke ser ut til å ha blitt testet i tidligere arbeid.

I de påfølgende underseksjonene vil det bli gitt en teoretisk innføring i hver av de tre modellene, for å gi leseren en større forståelse av de viktigste komponentene til hver enkelt algoritme. Vi starter med den minst komplekse i vår rapport, RF i seksjon [5.3.1](#), deretter videre inn i ARIMA i seksjon [5.3.2](#) og avslutter med den mest avanserte modellen LSTM i seksjon [5.3.3](#). Dette vil gjøre [kapittel 6](#) om selve treningen og testingen av modellene lettere å forstå for leseren.

### 5.3.1 Random Forest

Random forest (RF) er en ensemble algoritme som kombinerer flere Decision Trees (DT), og bruker Bootstrap Aggregating (Bagging). Et annet navn på DT er Classification and Regression Trees (CART). Denne algoritmen kan brukes til problemstillinger innenfor både regression og classification (Elgaaen et al., 2017). I vårt tilfelle, med datasett av typen tidsserie, vil vi bruke Regression Trees (RT) for å predikere. For å forklare RF-algoritmen, vil vi raskt beskrive hva et DT er.

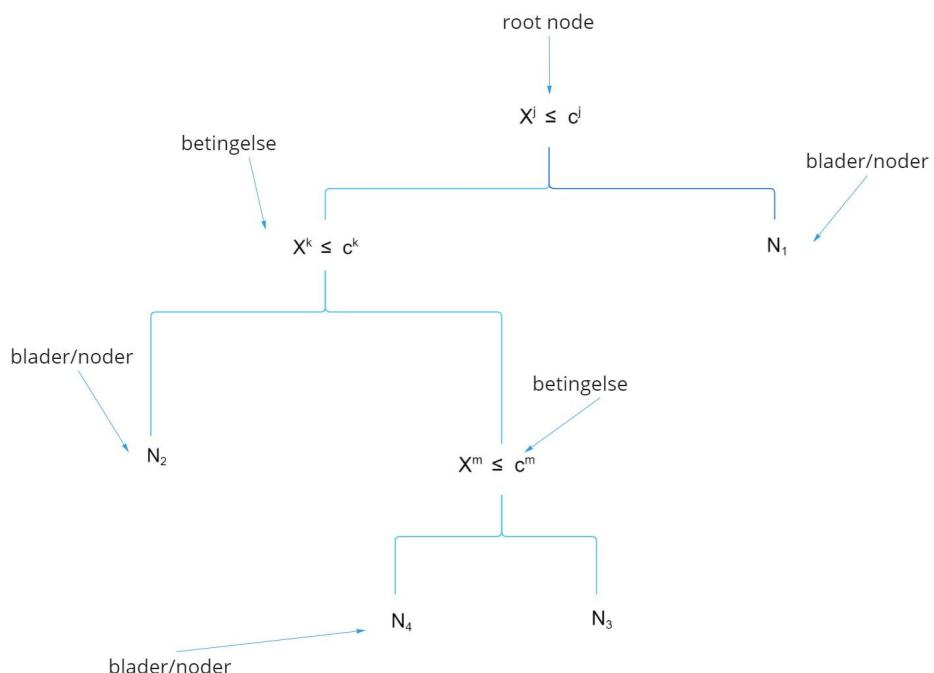
Et DT for regression deler dataen rekursivt inn i en trestruktur, med en rotnode på toppen, deretter beslutningsnoder og til slutt bladnoder. I disse oppsettene blir dataen delt

---

<sup>26</sup>

<https://www.advancinganalytics.co.uk/blog/2021/06/22/10-incredibly-useful-time-series-forecasting-algorithms>

rekursivt inn i såkalte «sub-groups», med et satt vilkår; variansen av en gitt beslutning skal bli så lav som mulig. Først velger algoritmen ut en delevariabel og et delepunkt som splitter dataen på best mulig måte. Algoritmen itererer deretter gjennom alle nodene til de nådde bladene har et oppnådd minimum av observasjoner (Woloszko, 2018). Et eksempel på et regression tree er vist under i Figur 22 (Biau et al., 2010).



Figur 22: Viser hvordan et Regression Decision Tree kan trenes ved hjelp av gitte betingelser.

DT er en mye brukt algoritme innen maskinlæring, men har noen svakheter.

Algoritmene kan ha en tendens til å bli overtilpasset. Dette konseptet blir forklart i seksjonen [4.3.2](#). I tilfeller med overtilpasning, vil DT-et lage et tre som blir for komplekst og resulterer i at det ikke klarer å generalisere trendene i dataen godt nok.<sup>27</sup> En slik modell tilpasser seg godt til datasettet, og vil derfor produsere feil resultat når den testes på ny input av data. DT viser også svakheter i møte med et datasett som inneholder lite variasjon i verdiene.. Grunnen til dette er selv om det kun er liten varians mellom beslutningene, kommer algoritmen til å produsere helt nye og ulike trær.<sup>28</sup> Effekten av disse svakhetene i resultatet

<sup>27</sup> <https://scikit-learn.org/stable/modules/tree.html>

<sup>28</sup> <https://scikit-learn.org/stable/modules/tree.html>

kan reduseres ved å bruke en samling av flere DT-er i en ensemble algoritme, som for eksempel RF.

En ensemble algoritme kombinerer et visst antall modeller for å lage en enkel modell som har bedre nøyaktighet i gjennomførte prediksjoner enn de individuelle komponentene. Det eksisterer ulike typer ensemble-algoritmer, slik som stacking, boosting, og bagging.<sup>29</sup> Bagging er en metode brukt for å samle og kombinere samme typer algoritmer for å lage en aggregert modell. Dette kan løse problemene med DT-er der kompleksiteten er begrenset hvis antallet grenar blir for stort. Konseptet til bagging er å ta et tilfeldig utvalg fra datasettet og utføre regression på hvert individuelle utvalg, slik at resultatene blandes for å lage en kvalifisert prediksjon. Det trekkes ut tilfeldige variabler fra treningssettet, og variablene danner grenene i et DT. For å avgjøre splitten i et DT velges den mest gunstige variabelen. Når analysene blir gjennomført, blir de valgte variablene lagt tilbake i det originale datasettet før det trekkes ut nye tilfeldige variabler. Algoritmen lager mange hundre ulike noder. Basert på prediksjonene fra disse nodene, blir et gjennomsnitt regnet ut med aggregering. Aggregering vil si at man tar gjennomsnittet av resultatene i hver node, aggregeringsgjennomsnittet, og utfører en flertallsavstemming som danner den ferdige prediksjonen av modellen (Bankson et al., 2019).

Bagging gir veldig høy nøyaktighet, men kan gi feil korrelasjon. Dette oppstår fordi det ikke er føringer for hvilke karakteristikker som skal være gjeldende for de ulike valgte variablene. Dermed kan man risikere at viktige variabler danner duplikater i treelet, som videre fører til at nodene blir høyt korrelerte dersom variablene med stor prediktiv kraft korrelerer. Derfor kan en helt ren bagging-metode medføre overtilpasning (Bankson et al., 2019).

RF skiller seg fra ren bagging ved å legge til et randomisert utvalg internt i hver node i valget av variablene som skal bestemme splitten videre i DT-et. Variabler med høy korrelasjon, som potensielt beskriver den samme variasjonen, vil derfor kun forekomme i et fåtall noder (Bankson et al., 2019). Deretter blir aggregeringsgjennomsnittet kalkulert av hver node som utgjør en samlet prediksjon, og bestemmer den endelige modellen til testobjektet. Denne fremgangsmåten reduserer korrelasjonen mellom hver node, slik at den inneholder flere særtrekk gjennom datasettet. RF unngår overtilpasning ved å generere nødvendig mengde trær, da overtilpasningen blir jevnet ut ved å aggregere resultatet til hvert tre. RF senker variansen i modellen og øker presisjonen i de endelige beregningene.

Fordelene med RF er at det kan både brukes til classification og regression, den kan håndtere manglende verdier og er nyttig til modellering av ikke-lineær data, altså data som

---

<sup>29</sup><https://medium.com/analytics-vidhya/ensemble-methods-bagging-boosting-and-stacking-28d006708>

ikke er ordnet lineært eller sekvensielt. RF klarer å håndtere store datasett og lager mer nøyaktige prediksjoner enn DT.<sup>30</sup>

Noen ulemper med RF-algoritmen er at den krever mye beregningskraft og ressurser, noe som kommer av at algoritmen danner mange hundre ulike trær. Perioden med treningsdata tar også mye lengre tid enn perioden til DT-algoritmen. Vi får lite oversikt over hva algoritmen egentlig gjør. Dette kalles for en black box algoritme, nettopp fordi vi ikke kan kontrollere hvordan algoritmen fungerer eller hvordan beslutningene blir tatt.<sup>31</sup>

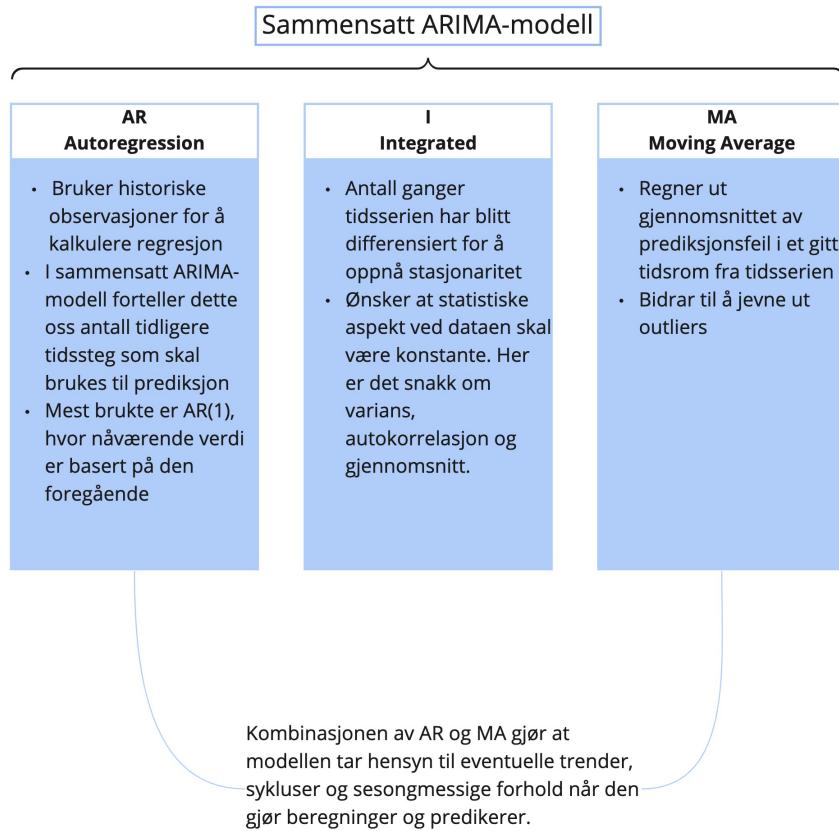
### 5.3.2 Autoregressive Integrated Moving Average

ARIMA, et akronym som står for Autoregressive Integrated Moving Average, er en sammensatt modell som brukes til analysering og prediksjon av tidsseriedata. Modellen er som RF en ensemble-modell, men med en kombinasjon av tre ulike komponenter, istedenfor flere av den samme. Hver av disse komponentene er også fungerende som selvstendige sett med modeller. Dette gjør ARIMA til et enkelt og kraftfullt verktøy, som ved hjelp av regression ser hvordan tidligere historikk i et datasett kan predikere fremtidige verdier. For å forstå oppbygningen av modellen skal vi se nærmere på hver enkelt av de tre komponentene , og videre hvordan sammensetningen av disse gjør ARIMA optimal for tidsserieprediksjon. I Figur 23 kan man se en enkel oversikt over de viktigste detaljene i modellen.

---

<sup>30</sup> <https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04>

<sup>31</sup> <https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04>



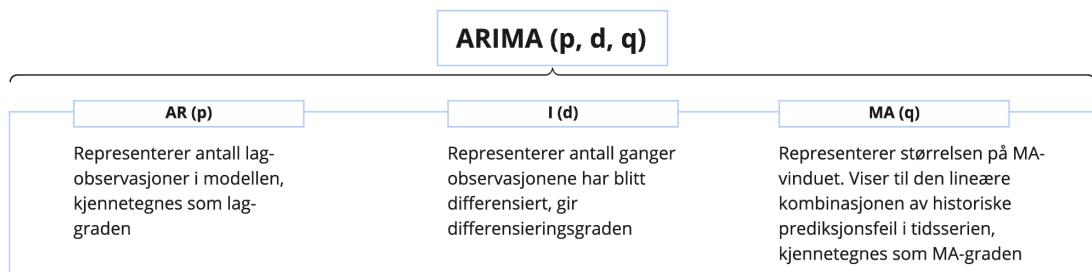
Figur 23: Tabell med beskrivelse av de tre komponentene ARIMA er satt sammen av.

AR er en forkortelse av «Autoregression», og tar i bruk en type linear regression for å gjøre prediksjoner. Til forskjell fra standard linear regression, hvor kalkulasjonen gjøres basert på et utvalg predikerte variabler, bruker autoregression historiske verdier som grunnlag for utregning og prediksjon. Verdiene som brukes er en samling av bestemte mengder inndelt i et gitt antall lag, forklart i seksjon [4.3.3](#), konstruert fra den originale tidsserien. Antall lag som brukes definerer graden til AR-delen av modellen, videre referert til som variabel  $p$ , som gir definisjonen  $AR(p)$  (Kotu & Deshpande, 2019, s. 420–421).

MA, en forkortelse for «Moving Average», står for en utregnet lineær kombinasjon som består av gjennomsnittet av prediksjonsfeil i tidsserien. Graden av MA-delen blir, i likhet med AR-delen, definert ved hjelp av en variabel som representerer antall lags. Videre vil dette være referert til som variabel  $q$ , som gir definisjonen  $MA(q)$  (Kotu & Deshpande, 2019, s. 423). Kombinasjonen av  $AR(p)$  og  $MA(q)$  utgjør ligningen som kalkulerer prediksjonsresultatet.

Mellan AR og MA har man bokstaven I, som står for «Integrated». Denne delen av en helhetlig ARIMA-modell representerer graden av differensiering som har blitt gjennomført på tidsserien (Kotu & Deshpande, 2019, s. 422). En slik prosess gjøres om det originale

datasettet ikke kan defineres som stasjonært, et aspekt som er forklart i seksjon 4.3.1. I de fleste tidsserie holder det å differensiere én gang, men det kan også være nødvendig med to om det er kraftige sesongmessige forhold og trender med behov for å jevnes ut. Antall ganger dette gjennomføres blir graden til  $I$ , videre referert til som variabel  $d$ , og gir definisjonen  $I(d)$ . I Figur 24 kan man se en enkel oversikt over de tre komponentene og hva hensikten til hver enkelt er.



Figur 24: Oversikt over sammensetningen til de ulike komponentene i ARIMA-modellen.

Før man går i gang med å bruke ARIMA-modellen er det viktig å bemerke seg de tre aspektene nevnt i seksjon 4.3.1 om tidsserier; stasjonaritet, sesongmessige forhold og trender. Med denne informasjonen er første steg å avgjøre om tidsserien kan beskrives som stasjonær. Om ikke dette er tilfellet er det hensiktsmessig å differensiere datasettet, hvor det i følge Granger og Newbold (1974) bidrar til å gi mer nøyaktige og presise resultat når stasjonære variabler brukes i regression modeller. En grafisk fremstillelse av dataen vil raskt avdekke om trender og sesongmessige forhold er tilstede, men det finnes også ulike tester man kan ta i bruk. En nøyaktig evaluering gis ved bruk av en Augmented Dickey-Fuller Test (ADF), hvor man tester nullhypotesen og ser om det eksisterer en såkalt unit root.<sup>32</sup>

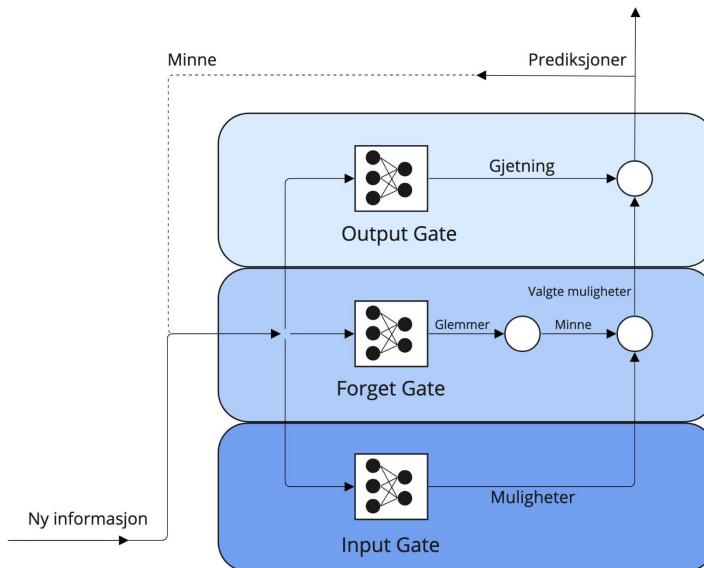
Hvert enkelt sett med data har sin egen optimale sammensetning av variablene  $p$ ,  $d$  og  $q$ . Gjennom hypertuning vil man forsøke å finne frem til denne, men man kan få ekstra hjelp av et innebygd verktøy i modellen kalt «The Akaike Information Criterion» (AIC).<sup>33</sup> Dette verktøyet estimerer en verdi som definerer kvaliteten til en gitt modell av ARIMA. Dermed kan man for hver ulike kombinasjon få ut AIC-informasjonen og sammenligne med aktuelle alternativer.

<sup>32</sup> <https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/>

<sup>33</sup> <https://otexts.com/fpp2/arima-estimation.html>

### 5.3.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) er et Artificial Neural Network (ANN) som brukes innenfor de komplekse områdene av DL, og blir ansett som et modifisert RNN. Modellen ble først foreslått av Hochreiter og Schmidhuber i 1997 (Li et al., 2018), og tar for seg algoritmer som forsøker å etterligne strukturen av den menneskelige hjernen for å avdekke underliggende relasjoner i data, forklart i seksjon [3.2.2](#). I motsetning til tradisjonelle DNN, hvor man antar at input og output er uavhengige av hverandre, er RNN sin output avhengig av de historiske komponentene i en sekvens. Motivasjonen for innføringen av LSTM var basert på at RNN-er kan lide av feilaktige observasjoner, fordi de ikke klarer å huske og bevare informasjon fra de første tidsstegene til de som kommer senere (Li et al., 2018). Dette kalles «Long-Term»-avhengighetsproblemet. LSTM-modellen løser dette problemet ved å introdusere tre ekstra porter; input gate, forget gate og output gate. Disse portene inneholder hvert sitt Neural Network (NN), som vist i Figur 25.



*Figur 25: En visuell framstilling av en typisk cellestruktur i LSTM med de ekstra portene som ble introdusert i LSTM-modellen (Inspirert av Al-jabery et al., 2020, s. 110, Figure 4.6).*

De tre portene i LSTM-cellens samarbeider med hverandre for å kontrollere informasjonen som skal bli lagret, husket eller lest. Forget gate bestemmer hvilken informasjon som skal bli sendt videre eller glemmes, og gjør dette ved å multiplisere verdien i minne med enten 0 eller 1. Mindre viktig informasjon blir glemt fra minnet, og det som anses som viktig blir sendt videre, som man kan observere i Figur 25. Input gate forsøker å

kvantifisere viktigheten av informasjonen hentet fra input, og deretter sende denne videre.<sup>34</sup> Informasjonen som er sendt videre fra forget og input gate brukes til å oppdatere «cell state» med ny informasjon som cellen ser på som viktig. Dette er en kombinasjon av både tidligere cell state og informasjonen fra forget og input gate. I output gate blir verdien av den neste hidden state bestemt, ved å bruke både nåværende cell state og tidligere hidden state. Til slutt sender porten den oppdaterte informasjonen av hidden state og cell state fra det nåværende tidsstempellet til den neste cellen, som er fremstilt i Figur 25 av den stiplete linjen.<sup>35</sup>

Etter å ha sett på de forskjellige portene, kan de matematiske funksjonene i LSTM uttrykkes. Hvis man ønsker å få en matematisk forståelse på hvordan disse portene funker, kan man lese artikkelen «*Single-point wind forecasting methods based on deep learning*» (Liu, H. 2021, s. 142-143), som tar for seg dette.

LSTM utpeker seg fra andre NN, og kan forutsi verdier for punktdata og predikere sekvensiell data. I motsetning til alle såkalte «feedforward»-NN, har LSTM «feedback connections», som man bemerker på Figur 25 ved at det er to retninger på pilene - både fram og tilbake.<sup>36</sup> Dette lar modellen beholde informasjon i «minnet» over tid. LSTM-nettverk har noen interne kontekstuelle state cells som fungerer som langtids- eller korttidsminneceller.<sup>37</sup> Utgangen fra LSTM-nettverket moduleres av tilstanden til disse cellene. Dette er en veldig viktig egenskap når vi trenger at prediksjonen til modellen er avhengig av den historiske konteksten til input, ikke bare den nyeste inputen. I datasettet som vi jobber med, hvor vi skal predikere Readiness to Play, er denne egenskapen viktig. Når vi vil predikere Readiness-variabelen for neste dag, ønsker vi at prediksjonen er avhengig av hele datasettet, altså flere dager - ikke bare dagen før.

På den andre siden har også modellen sine feil og svakheter. Selv om et LSTM kan huske langsiktige avhengigheter, gjør dette også modellen svært ressurskrevende å trenе.<sup>38</sup> Dette gjør at tiden det tar med testing og forbedringer av modellen blir betydelig lengre. I våre eksperimenter observererer vi at LSTM presterer bedre når det har en økende datamengde, som vi har fremvist i seksjon [6.6](#) på Figur 42. I rapporten til Wiik et al. undersøker de også at modellen får bedre prediksionsresultat ved bruk av et større datasett. Vi har datasett med innhold fra i underkant av 2 år, og kan ikke forvente at resultatet blir like nøyaktig sammenlignet med et datasett med innhold loggført over enda flere år.

---

<sup>34</sup> <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/>

<sup>35</sup> <https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn>

<sup>36</sup> <https://analyticsindiamag.com/tutorial-on-univariate-single-step-style-lstm-in-time-series-forecasting/>

<sup>37</sup> <https://medium.com/datathings/the-magic-of-lstm-neural-networks-6775e8b540cd>

<sup>38</sup> <https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0>

Noe litteratur viser også at CNN gir bedre resultater enn LSTM, samt mye enklere og raskere å trenne (Weytjens & de Weerdt, 2020). Ifølge rapporten viser det seg at CNN-modeller leverer de samme resultatene som de nyeste LSTM-modellene på en brøkdel av tiden, og kan derfor anbefales som førstevalg av de to modellene. Likevel valgte vi i dette prosjektet å fokusere på LSTM og egenskapene denne modellen tilbyr.

# 6 Trening og testing av maskinlæringsmodeller

I kapittel 5 ble prosjektets sammensatte maskinlæringspipeline presentert, en fremstillelse av de ulike stegene som inngår i maskinlæringsarbeidet for denne rapporten. I seksjonene 5.3.1, 5.3.2 og 5.3.3 introduserte vi også de tre algoritmene vi skal eksperimentere med: Random Forest (RF), ARIMA og LSTM. I disse seksjonene ble det gitt en innføring i grunnleggende teori og bruksområder, og i dette kapittelet vil vi presentere selve testingen og treningen av modellene med datasettene som ble presentert i kapittel 4. Hver enkelt modell har ulike utgangspunkt når det gjelder å kunne predikere en spillers prestasjonsevne i forkant av kamp og trening. I de neste seksjonene vil vi utføre flere eksperimenter, for å videre analysere og drøfte resultatene vi får fra dem. Til slutt vil vi sammenligne resultatene til de tre modellene, og trekke en endelig konklusjon for dette kapittelet.

## 6.1 Forhåndsbestemte variabler og innstillinger

Implementasjonen av de tre modellene har blitt utført med samme forhåndsbestemte utgangspunkt. Datasettene til de fire spillerne inneholder 651 loggførte verdier fra perioden 21.03.2020 til 31.12.2021. Videre måtte vi dele datasettene inn i trening- og testsett, forklart i seksjon 4.3.2. Fordelingen av trening- og testsett ble satt til 80% og 20%, resulterende i en treningsperiode som inneholder 520 verdier (fra 21.03.2020 til 23.08.2021) og en testperiode som inneholder resterende 131 verdier (fra 23.08.2021 til 31.12.2021). Grunnen til at man vil splitte datasettet på en slik måte er for å unngå overtilpasning, definert i seksjon 4.3.2. Størrelsen på test- og treningssettet er valgt på bakgrunn av at det er en vanlig fordeling brukt i maskinlæring.<sup>39</sup>

Som beskrevet i seksjonene 1.5 og 1.6, skal vi forsøke å beregne og analysere om modellene er i stand til å predikere en fremtidig Readiness-variabel til en spiller. Det eksisterer flere ulike fremgangsmåter for å utvikle en prediksjonsmodell. I de fleste tilfeller velger man en gitt tidsperiode som skal brukes til å predikere den neste, og i seksjon 4.3.3 ble det forklart at denne perioden kan refereres til som en lag. I våre datasett er én lag én dag, og vi måtte ta en avgjørelse på hvor mange lags som skulle brukes til å predikere neste dag og verdi. Oppdragsgiver hadde ingen konkrete krav på dette området, men vi diskuterte rundt hvilke verdier det ville være mest realistisk å ta i bruk. Ofte innebærer forberedelse til kamp og trening rundt tre dager med forberedelse, hvor kampdager ofte er midt i uken eller i

---

<sup>39</sup> <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/>

helger. Dette ga oss et naturlig utgangspunkt for prediksjon, og lag-verdien ble derfor satt til 3. I Figur 26 under er det en presisering av de satte verdiene og innstillingene for et felles utgangspunkt brukt i modellene.

Utgangspunkt
<ul style="list-style-type: none"> <li>• <b>Fire datasett:</b> Spiller1, Spiller2, Spiller3 og Spiller4</li> <li>• <b>651 verdier</b> fra tidsperioden 21.03.2020 til 31.12.2021</li> <li>• Fordeling av trenings- og testsett satt til 80% og 20%</li> <li>• <b>Treningssett: 520 verdier (21.03.2020 til 23.08.2021)</b></li> <li>• <b>Testsett: 131 verdier (23.08.2021 til 31.12.2021)</b></li> <li>• <b>Lagverdi satt til 3;</b> tre dager brukes til å predikere den neste</li> </ul>

*Figur 26: En oversikt over det satte utgangspunktet for verdier og innstillinger til trening og testing av modellene RF, ARIMA og LSTM.*

I seksjon [4.2](#) forklarer vi at det er totalt 4 spillere som har blitt valgt for videre arbeid under forsøkene som presenteres i seksjonene [6.4](#), [6.5](#) og [6.6](#) i dette kapittelet. Det vil bli gitt numeriske og grafiske eksempler fra alle fire datasett, hvilke det gjelder vil spesifiseres der det er nødvendig. Underveis i testingen ble det også forsøkt å eksperimentere med verdier utenom det satte utgangspunktet, slik at det var mulig å se om andre innstillinger kunne gi bedre resultat. Dette gjøres rede for i seksjonene om hver enkelt modell.

## 6.2 Måling av feilestimat

Når vi starter prosessen med å trenere og teste modellene, finnes det ulike metoder man kan ta i bruk for å validere resultatene man får. Vi kan måle feilestimatet av modellens evne til å predikere nye data på. Prediksjon er en iterasjonsoppgave; først bygger man og trener modellen, deretter tester man, og til slutt måler man ytelsen og prediksjonsevnen. Denne prosessen gjentas til man opplever forbedringer i resultatene. Fremgang måles blant annet med beregninger, og er en måte å forstå om modellen gjør en god jobb eller ikke.

Test error er det beregnede feilestimatet man får når man gir en trenert modell input data den ikke har sett før. Denne feilmarginen blir brukt for å måle nøyaktigheten til modellen før den blir satt ut i produksjon. Det er spesielt tre metoder som ofte benyttes i denne

sammenhengen; Mean Squared Error (MSE)<sup>40</sup>, Mean Absolute Error (MAE)<sup>41</sup> og Coefficient of Determination ( $R^2$ )<sup>42</sup>. Alle disse brukes hovedsakelig til problemstillinger som har med regression å gjøre, som innebærer at de er alle naturlige målinger av de tre modellene vi har valgt (se seksjonene [5.3.1](#), [5.3.2](#) og [5.3.3](#) for mer informasjon).

MSE kan beregnes ved å ta gjennomsnittet til kvadratet av differansen mellom modellens prediksjoner og de faktiske loggførte verdiene i datasettet (Vallantin, 2020). MAE gjør en nesten helt lik beregning, men tar kun gjennomsnittet av de absolutte forskjellene mellom prediksjonene. Begge metoder har en skala med verdier som går fra 0 til uendelig, hvor 0 definerer en perfekt modell med helt nøyaktige prediksjoner. MSE bør minimeres for å få en mer nøyaktig modell. En MSE på 0 vil bety at modellen har overtilpasset dataene, og forteller oss at modellen er for kompleks og presterer dårligere. En MSE på 1 vil bety at modellen er undertilpasset, som vil si at modellen er for enkel og ikke vil gi gode beregninger.<sup>43</sup>

$R^2$  forteller oss hvor stor andel av variasjonen mellom faktiske verdier og predikerte datapunkt som henger sammen. Estimatet måles av verdier mellom 0 og 1, men noen definisjoner av  $R^2$  tillater verdier fra  $-\infty$  til 1. Generelt er verdier nærmere 1 bedre enn verdier nær 0 eller med negativt fortegn (Vallantin, 2020). Hvis  $R^2$  gir oss en negativ verdi som resultat, betyr det at den valgte modellen ikke følger trendene i datasettet. Det betyr at modellen gir en bedre prestasjon på grunn av overtilpasning, forklart i seksjonene [4.3.2](#).

Figur 27 illustrerer de tre presenterte metodene for måling av feilestimat, sammen med en kort beskrivelse.

---

<sup>40</sup> [https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8\\_528](https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_528)

<sup>41</sup> [https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8\\_525](https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_525)

<sup>42</sup> <https://www.scribbr.com/statistics/coefficient-of-determination/>

<sup>43</sup> <https://medium.com/@lightwrx818/regression-model-evaluation-metrics-7eb82fcfec94>

Metode	Beskrivelse	Formel
Mean squared error (MSE)	Kalkulerer gjennomsnittet til kvadratet av differansen mellom faktiske datapunkt og predikerte verdier.	$\frac{1}{N} \sum_{i=1}^N (y_i - \mu)^2$ $y_i$ er en forekomst, $N$ er en rekke med forekomster, $\mu$ er gjennomsnittet gitt av $\frac{1}{N} \sum_{i=1}^N y_i$
Mean absolute error (MAE)	Kalkulerer gjennomsnittet av den totale differansen mellom faktiske datapunkt og predikerte verdier.	$\frac{1}{N} \sum_{i=1}^N  y_i - \mu $ $y_i$ er en forekomst, $N$ er en rekke med forekomster, $\mu$ er gjennomsnittet gitt av $\frac{1}{N} \sum_{i=1}^N y_i$
Coefficient of determination ( $R^2$ )	Forteller oss hvor stor andel av variasjonen i faktiske verdier og predikerte datapunkt som henger sammen.	$R^2 = 1 - \frac{\sum_{i=1}^N (\hat{y}_i - y_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}$ $N$ er en rekke med forekomster, $y_i$ er verdier, $\bar{y}$ er gjennomsnittet av alle verdier, $\hat{y}_i$ representerer prediksjoner/punkter punkter på regresjonslinjen

Figur 27: Oversikt over de ulike metodene for måling av feilestimat, sammen med en kort beskrivelse og formlene for beregning.

## 6.3 Baseline for test score

I forrige seksjon, [6.2](#), introduserte vi tre ulike former for test score, noe som gir en numerisk verdi representativt for et beregnet resultat fra en maskinlæringsmodell. Disse verdiene forteller oss ikke nødvendigvis så mye dersom vi ikke har noe å sammenligne med. Akkurat hva man kan forvente som resultat i form av test score kan være vanskelig å vite, uansett hva slags problem man forsøker å løse ved hjelp av maskinlæring. Vi ønsker derfor å sette en tallverdi på denne forventningen, og dette får vi gjennom en «baseline». En baseline setter en terskelverdi for forventet test score, og gir oss nettopp det vi ville ha; en verdi å sammenligne med. Dette bidrar til en større indikasjon på hvor mye man kan endre på innstillinger og hypertune parametrene til modellen.

Et verktøy som kan bidra med dette er en dummy modell. Det finnes to forskjellige typer, begge tilgjengelige gjennom biblioteket scikit-learn (se seksjon [2.5.3](#)) i Python: classifier<sup>44</sup> og regressor<sup>45</sup>. Navnene tilsier at man velger den algoritmen som passer i henhold til maskinlæringsmodellen man arbeider med. I begge kategoriene vil dummyen utlate forsøk på å forstå eller lete etter mønster i datasettet den trenes og testes på, men gjør prediksjoner basert på enkle retningslinjer. Deretter beregner man test score på prediksjonene gjort av dummy modellen, noe som gir en baseline man kan sammenligne med i arbeidet med en komplett maskinlæringsmodell. Terskelen er ment som en grense man kan forvente at den faktiske modellen skal kunne overgå resultatløftet.

<sup>44</sup> <https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html>

<sup>45</sup>

<https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyRegressor.html?highlight=dummymodel#sklearn.dummy.DummyRegressor>

I vårt prosjekt har vi tatt i bruk en dummy regressor, hvor oppsettet fremstår som en forenklet form for en maskinlæringsmodell. Videre deles datasettet inn i en trenings- og testperiode, hvor størrelsesforholdet settes til 80% og 20%. Dette utgangspunktet er likt det som implementeres i modellene i eksperimentene videre, beskrevet i seksjon [6.1](#). Deretter stilles dummyen til å predikere en gjennomsnittsverdi for datasettet, etterfulgt av beregnet test score på resultatet. Vi beregner verdier for MSE, MAE og  $R^2$  for hver spiller sin dummy, hvor resultatet er presentert i Figur 28.

```
Baseline for testscore - Spiller1:
MAE: 1.0992660011743982
MSE: 2.071485952391707
R^2: -0.0015995274708127116
```

```
Baseline for testscore - Spiller2:
MAE: 0.7145038167938931
MSE: 1.031584082388545
R^2: -0.0022086977960720233
```

```
Baseline for testscore - Spiller3:
MAE: 1.731664709336465
MSE: 3.505341789376214
R^2: -0.006781095355401057
```

```
Baseline for testscore - Spiller4:
MAE: 0.841441573693482
MSE: 1.1025241372690726
R^2: -0.012220025662024403
```

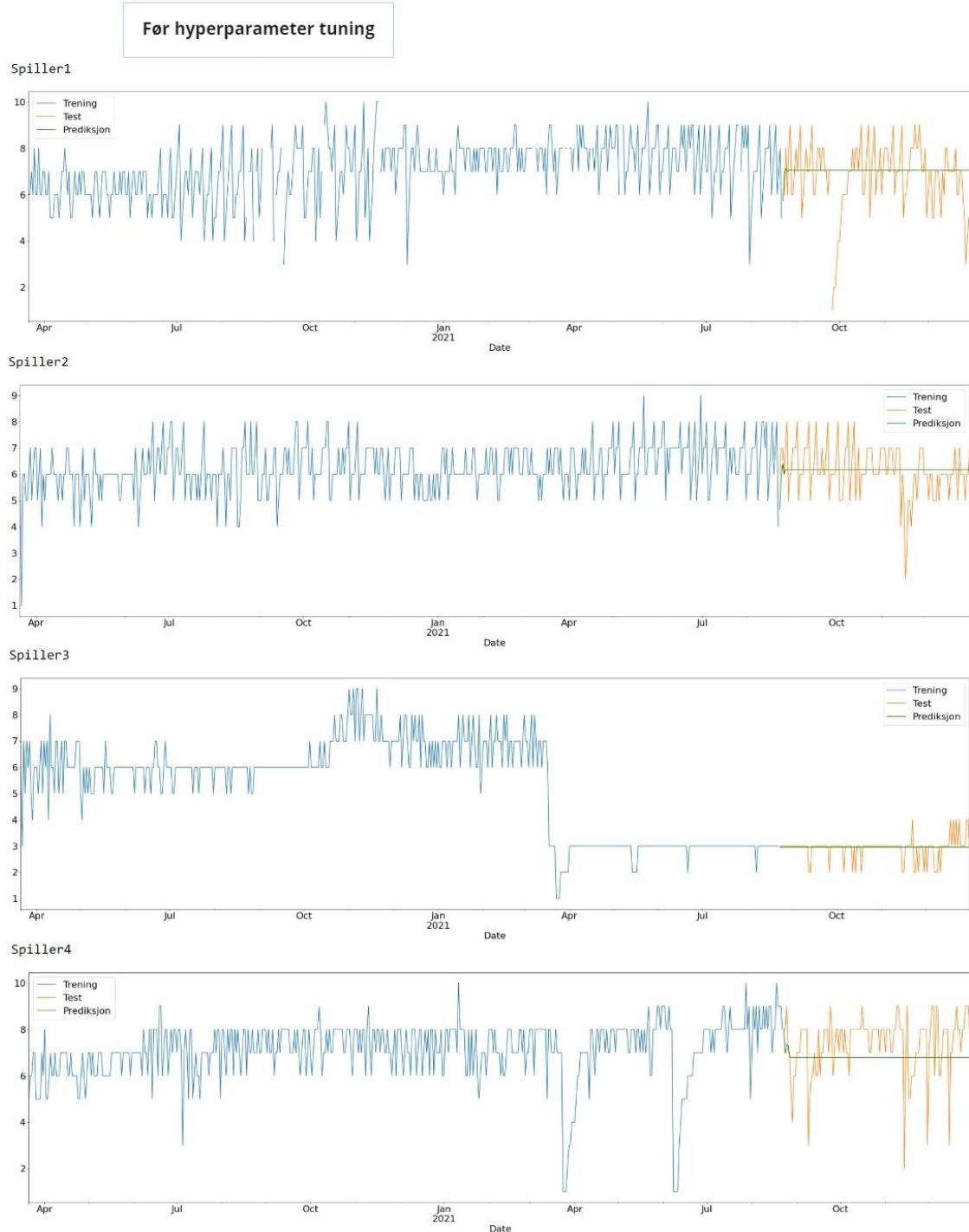
Figur 28: Oversikt over resultat fra dummy regressor for hver av de fire spillerne.

I seksjon [6.2](#) ble det gjort rede for hvilke verdier det er ønskelig å nærme seg resultatmessig når man beregner feilestimatet til en modell. Tester man MSA eller MAE vil en verdi tett opp mot 0 være å foretrekke. I terskelverdiene fra dummy modellen ser vi at det finnes et potensiale for å nærme seg nullgrensen, men det er tydelige forskjeller mellom de ulike spillerne. Ser vi på verdien for  $R^2$  kan vi observere at denne slår ut negativt hos alle spillerne. Dette kan være et resultat av overtilpasning, forklart i seksjonene [6.1](#) og [6.2](#), noe som betyr at dummy-modellen tar inn støy fra datasettet som kan påvirke resultatet negativt. Her spiller de originale datasettene en avgjørende rolle, hvor plutselige store endringer kan gjøre utslag på denne måten. Dette vil vi gå nærmere inn på i testingen av hver enkelt modell i seksjon [6.4](#), [6.5](#) og [6.6](#).

## 6.4 Random Forest Regression

Random Forest (RF) var en av de første algoritmene vi valgte, basert på informasjon presentert i seksjonene [2.3](#) og [5.3.1](#). RF er en fleksibel algoritme brukt i problemstillinger innenfor både classification og regression, samtidig som den kan brukes på store og små datamengder. Ettersom gruppen har erfaring med enkle maskinlæringsalgoritmer fra før, var dette en optimal kandidat å starte prediksjonsarbeidet med. Bibliotek og program brukt under testingen er presentert i seksjon [2.5](#) om rammeverk.

Første steg er å bygge og trenre forecaste funksjonen til skforecast med RF algoritmen. Innstillingene for verdiene som implementeres er presentert i Figur 26 under seksjon [6.1](#). Figur 29 viser de første prediksjonsgrafene til hver enkelt spiller før vi har gjennomført en metode kalt hyperparameter tuning, som vi går nærmere inn på i seksjon [6.4.1](#). For å kunne måle forskjellen mellom predikerte og faktiske loggførte verdier, vil vi se nærmere på metoder for måling av feilestimat presentert i seksjon [6.2](#).



Figur 29: Fremstilling av RF prediksjoner for hver enkelt av de fire spillerne, før hyperparameter tuning.

#### 6.4.1 Hyperparamter tuning

I prosessen med å forsøke å forbedre prestasjonsevnen til RF tar vi i bruk en funksjon gjennom Skforecaster-biblioteket. Denne metoden heter hyperparameter tuning og funksjonen bruker en såkalt «grid search»-strategi med backtesting for å oppdage kombinasjoner av lags og hyperparametere som fremstår som mer optimale og skal forbedre

modellens prediksjonsevne.<sup>46</sup> <sup>47</sup> Vi har et utgangspunkt for lag-verdien satt til 3, men denne verdien kan endres ved å bruke en funksjon kalt `grid_search_forecaster` hvor oppnådde resultater blir sammenlignet med hver modellkonfigurasjon. Resultatet gir oss forslag til optimale lag-verdier, antall Decision Trees (DT) som skal brukes og et maks antall nivåer i hvert DT som er anbefalt å ta i bruk. I stedet for å avgrense lags til 3 vil vi sette et avgrenset område med mulige verdier, som er lengden på den lengste veien fra en rot til et blad.

Etter å ha gått gjennom hyperparameter tuning vil metoden skrive ut et forslag med de mest optimale parametrene RF-modellen kan ha. I dette tilfellet ble antall DT avgrenset fra 100 til 500, lengden på forgrening ble satt til 3, 5,10 og avgrensingen med lags ble satt til [3, 10]. I Figur 30 under ser vi hva resultatet av denne implementasjonen gir oss for hver enkelt av de fire spillerne.

<b>Spiller1</b>
Lags: [1 2 3]
Parameters: {'max_depth': 5, 'n_estimators': 500}
Backtesting metric: 1.3015751155256465
<b>Spiller2</b>
Lags: [ 1 2 3 4 5 6 7 8 9 10]
Parameters: {'max_depth': 3, 'n_estimators': 500}
Backtesting metric: 0.865937816959681
<b>Spiller3</b>
Lags: [1 2 3]
Parameters: {'max_depth': 3, 'n_estimators': 500}
Backtesting metric: 2.350595671468066
<b>Spiller4</b>
Lags: [ 1 2 3 4 5 6 7 8 9 10]
Parameters: {'max_depth': 3, 'n_estimators': 100}
Backtesting metric: 2.51713781514796

Figur 30: Fremstilling av resultatet fra metoden Hyperparameter tuning for hver av de fire spillerne.

Figur 30 viser de beste resultatene som kan oppnås ved å bruke en lag-verdi opp til 3 eller 10, forgreningsnivåer på 3, 5 eller 10 og 100 eller 500 DT innad skogen. Til slutt trenes modellen ved hjelp av funksjonen `ForecasterAutoreg` med de mest optimale konfigurasjonene. For å se om det har oppstått forbedringer fra prediksjonen presentert i Figur 29, ser vi på resultatet fra de nye innstillingene og verdiene fra test error (se seksjon [6.2](#)). I Figur 31 ser vi en fremstilling av resultatene fra MSE før og etter hyperparameter

<sup>46</sup>

<https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/>

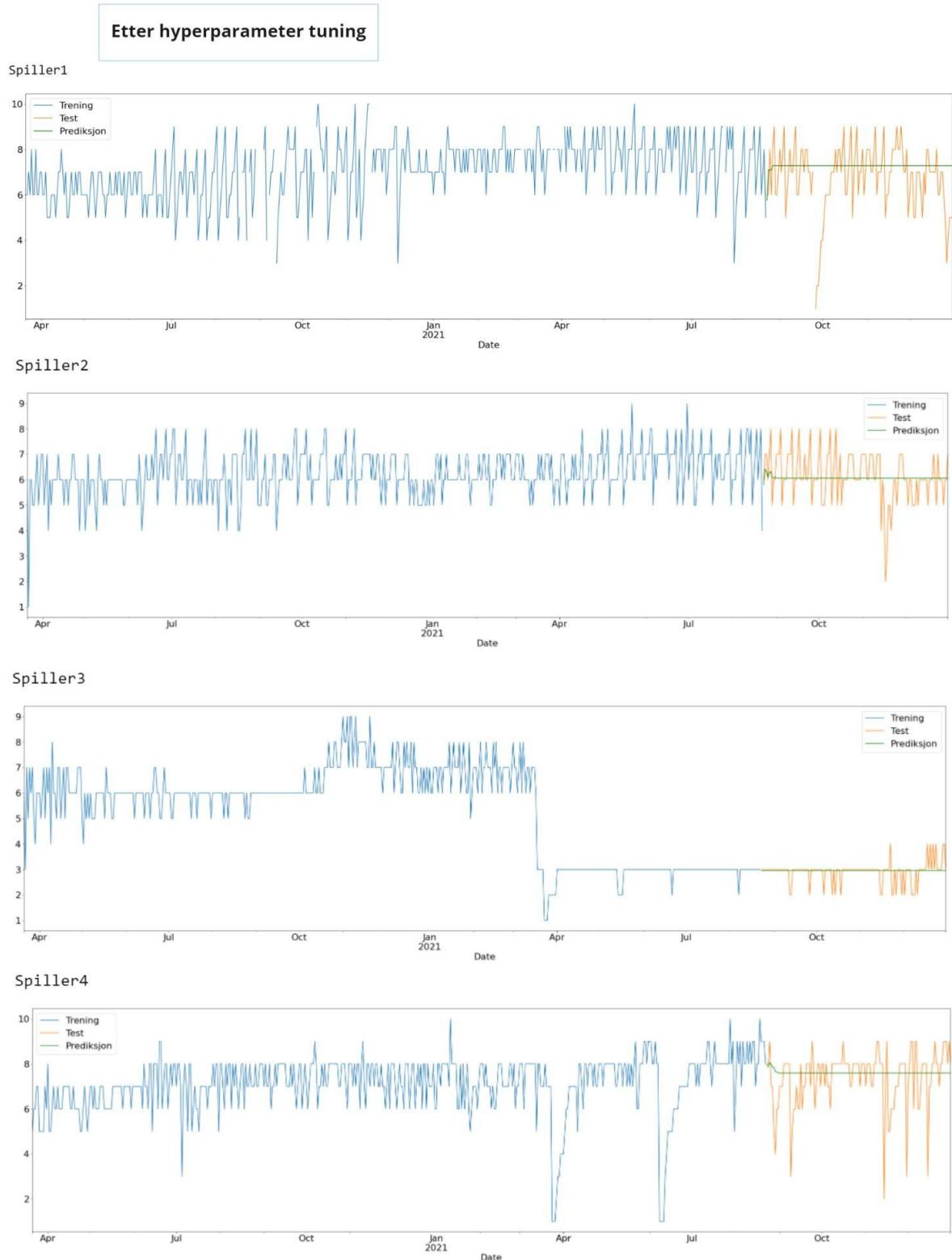
<sup>47</sup> <https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e>

tuning, og  $R^2$ -score. I Figur 32 under er det en grafisk fremstilling av test- og treningssett kombinert med prediksjon, for hver enkelt av de fire spillerne, etter hyperparameter tuning.

<b>Spiller1:</b>
MSE test error for prediksjon: 2.2419845907397637
MSE test error for prediksjon med hyper tuning: 2.418658440986441
R2 score test error for prediksjon: -0.035379374641686834
<b>Spiller2:</b>
MSE test error for prediksjon: 0.9491093044203369
MSE test error for prediksjon med hyper tuning: 0.92411795670246
R2 score test error for prediksjon: -0.017089095363894025
<b>Spiller3:</b>
MSE test error for prediksjon: 0.18593324298842032
MSE test error for prediksjon med hyper tuning: 0.18589706896368038
R2 score test error for prediksjon: -0.011667844934775351
<b>Spiller4:</b>
MSE test error for prediksjon: 1.976505769843072
MSE test error for prediksjon med hyper tuning: 1.7116073562672722
R2 score test error for prediksjon: -0.19365201000411614

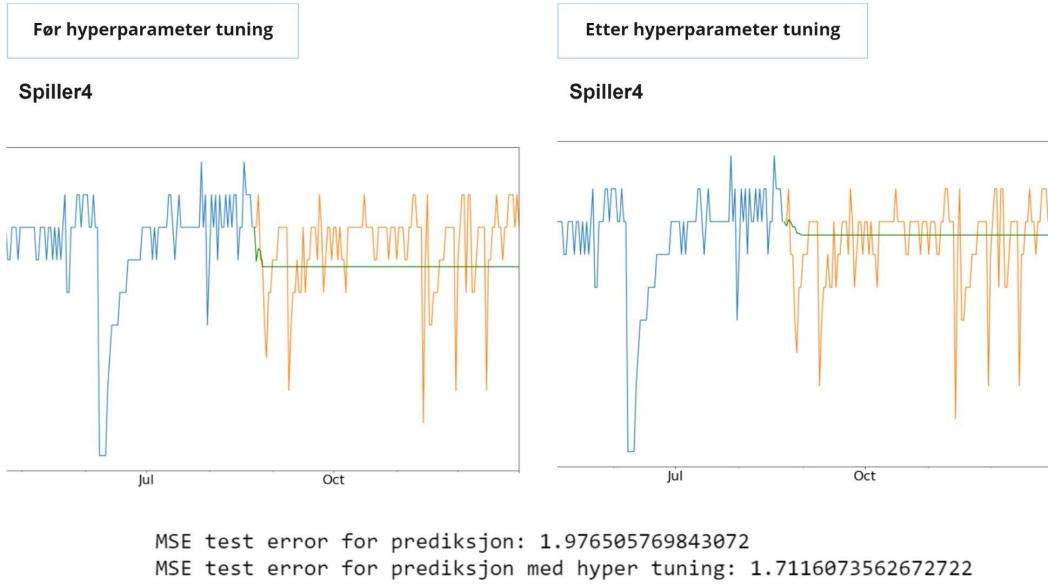
Figur 31: Oversikt over resultatene fra test-score for MSE før og etter hyperparameter tuning, og  $R^2$ -score, for hver enkelt spiller.

For alle spillerne ser vi at test erroren beregnet fra MSE har sunket ved hjelp av hyperparameter tuning, fremstilt i Figur 31. Sammenligner vi disse resultatene med baseline satt av dummy modellen i Figur 29 under seksjon [6.3](#), ser vi at Spiller2 og Spiller3 presterer bedre enn terskelverdien for prestasjon. Det er en vesentlig stor forskjell mellom resultatene fra dummy modellen sammenlignet med Spiller3. En vurdering av denne grafen spesielt, i forhold til de resterende grafene i Figur 32, forteller oss at det er et mye mindre spenn i faktiske registrerte verdiene under prediksionsperioden hos Spiller3. I alle datasettene får vi en graf hvor slutten av prediksionsperioden gir oss en gjennomsnittsverdi som prediksjon, og grafen gir oss kun en rett linje. I prediksionsresultatet til Spiller3 gir dette et ekstra godt resultat på grunn av mindre variasjon i den originale tidsserien.



Figur 32: Fremstilling av prediksjon for hver enkelt av de fire spillerne etter hypertuning.

I fremstillingen av prediksjonsgrafen med de nye innstillingene i Figur 32, ser vi at det ikke er store forskjeller sammenlignet med grafen i Figur 29. Vi ser nærmere på hvor stor forbedringen er, og sammenligner prediksjonen til Spiller4 før og etter hyperparameter tuning i Figur 33.



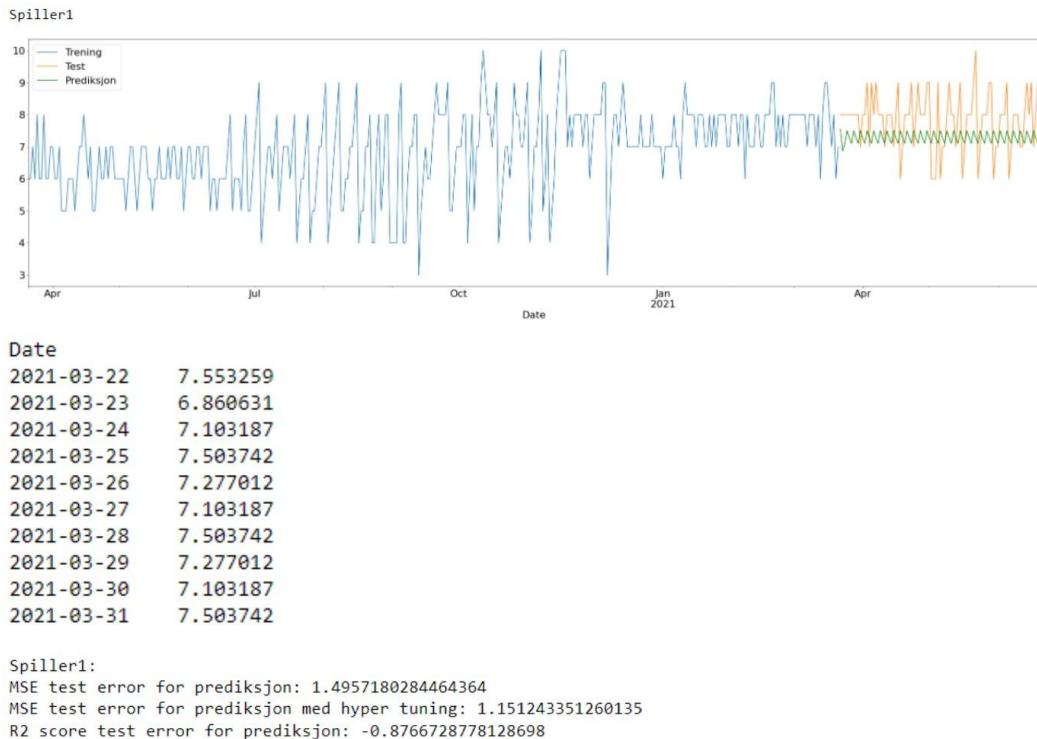
Figur 33: Viser hvor stor forskjell det er i prediksjonen av datasettet til Spiller4, før og etter hyperparameter tuning.

Fra Figur 33 ser vi at test erroren beregnet fra MSE har sunket etter gjennomføringen av hyperparameter tuning, men det er ikke en stor forskjell på den grafiske fremstillingen og dermed de faktiske predikerte verdiene. Utslaget i starten av prediksjonen ser ut til å være forbedret, deretter får vi en tilnærmet gjennomsnittsprediksjon i ettermat. Dette ser ut til å være tilfellet i alle prediksjonsgrafene fremstilt, se Figur 32.

Basert på resultatene etter hyperparametertuning ser vi hos alle en nedgang i test error kalkulert fra MSE. De største nedgangene ser ut til å oppstå hos spillerne med datasett hvor de loggførte verdiene er mer regelmessige. Det er naturlig at det vil oppstå uregelmessigheter og store variasjoner i informasjonen fotballspillere registrerer. RF-modellen ser ut til å ha vanskelig for å prestere godt når den får et mer variert datasett, og med tanke på kompleksiteten til problemstillingen for oppgaven er dette en tydelig svakhet.

Etter å ha gjennomført flere tester med de fire spillerne, ble det også gjort eksperimenter med en annen tidsperiode enn det som ble satt som utgangspunkt i seksjon 6.1. Vi ville forsøke å se om vi klarte å forbedre prestasjonsevnen til RF-modellen, og satte

en ny tidsperiode fra 21.03.2020 til 20.06.2021 der treningssett og testsett er fortsatt på 80% og 20%. Dette resulterer i et treningssett som består av 91 dager med registrerte verdier.



Figur 34: Fremstilling av en ny avgrenset tidsperiode fra 20.03.2020 til 21-06.2021 for Spiller1.

I Figur 34 ser vi resultatet av den nye tidsperioden hos Spiller1. Sammenligner vi test score fra MSE fra Spiller 1 i Figur 34 med resultatet i Figur 31 ser vi en betydelig nedgang og forbedring av resultatet, men det er fortsatt tendenser til undertilpasning. Ser vi på den grafiske fremstillingen får vi nå en graf som ender opp med å gjenta enkelte verdier kontinuerlig gjennom testperioden etter 27.03.2021.

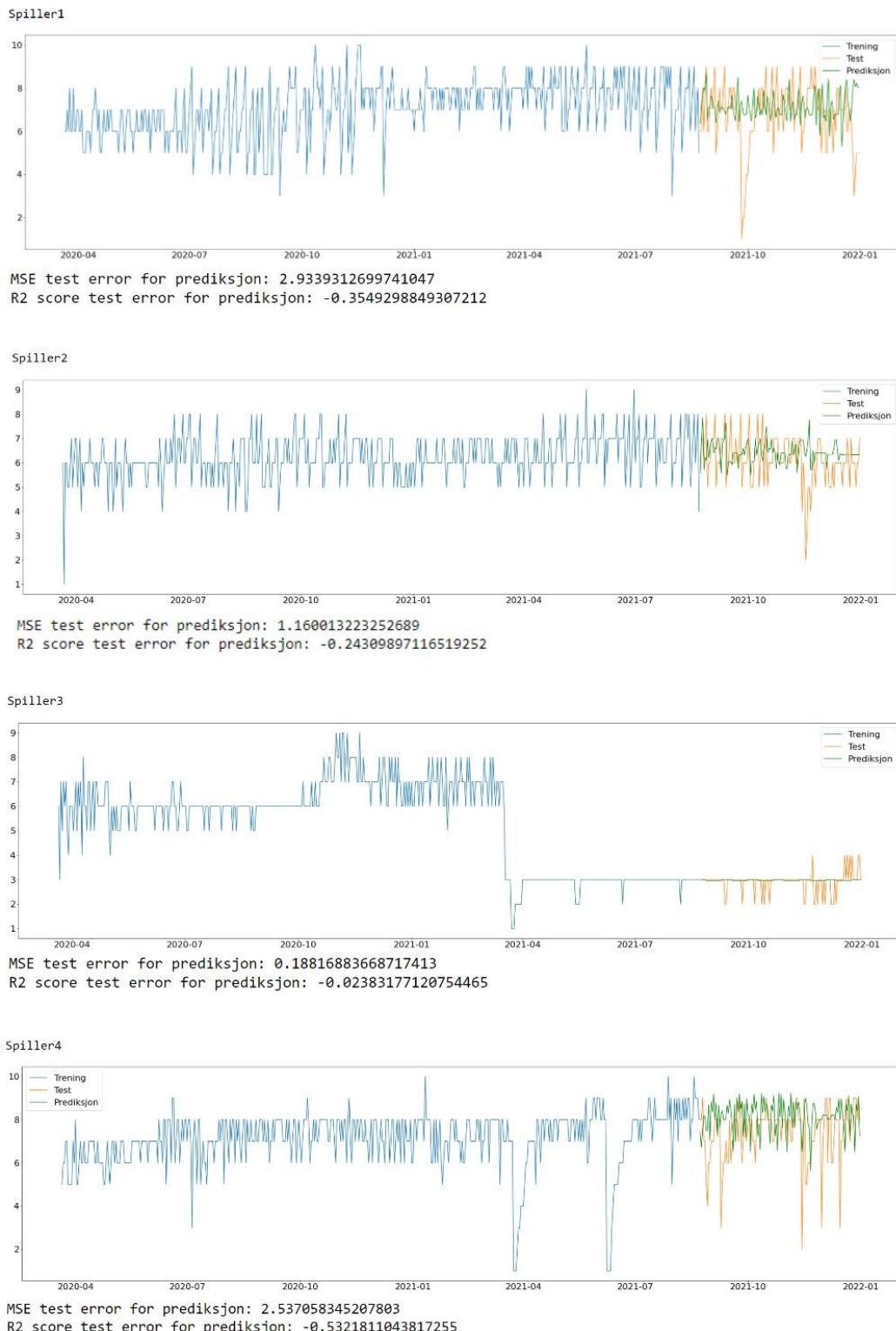
Testing på resten av spillerne i den gitte tidsperioden førte ikke til noen betydelige forbedringer som hos Spiller1, noe som kan forklares ved at denne testperioden inneholdt mer regelmessige og konstante verdier. Med en nedgang fra 2.24 til 1.15 i MSE hos Spiller1 kan vi si at resultatet ble bedre med den nye tidsperioden. Likevel ser vi indikasjoner på at forbedringen bare er grunnet et letttere datasett å predikere på.

#### 6.4.2 Kalkulere prediktorene fra tidsserie

Etter å ha studert resultatene til nå, og prøvd å finne ut om det er en grunn til at det blir presentert mer variasjon i prediksjonene i starten av testeperioden, viser det seg at RF-modellen kun predikrer et fåtall dager inn i fremtiden. Dette vil si at den ikke forskyver

lags-perioden utover resterende steps i tidsperioden, som forklart i seksjon [4.3.3](#). Vi forsøkte derfor å innføre en ny funksjon som lar oss gjøre nettopp dette; forskyve lags utover prediksjonsperioden, slik at den tar hensyn til de tre forrige predikerte verdiene. Det blir laget en funksjon som beregner de 3 første lagsene og setter et moving average til de siste 20 verdiene, brukt til å lage prediktorene. Resten av prosessen med trening og testing er lik som i seksjonene [6.4](#) og [6.4.1](#), hvor størrelsen til trening- og testsett og total tidsperiode er slik som utgangspunktet presentert i seksjon [6.1](#). Ved hjelp av den nyimplementerte funksjonen får vi resultatet fremstilt i Figur 35 under. Her ser vi grafene til hver enkelt til de fire spillerne, sammen med beregnet test score fra MSE og  $R^2$ .

Vi ser med en gang på grafene til hver spiller i Figur 35 at det er en stor forskjell fra de tidligere fremstilte testene i Figur 29 og Figur 32. Prediksjonene følger i mye større grad de faktiske loggførte verdiene i testsettet, og vi får en mer realistisk fremstilling av prediksjonsverdiene. Det er tydelig at funksjonen som ble implementert har forbedret RF-modellens evne til å følge mønstrene i testsettet. Om vi videre ser på resultatet kalkulert av MSE og  $R^2$  er det ikke store forskjeller fra estimatene fra tidligere tester. Spiller3 presterer fortsatt bedre enn satt baseline fra Dummy modellen, og er den eneste som gjør det i dette eksperimentet. Det er likevel viktig å understreke den store forbedringen i faktiske predikerte verdier vist gjennom hvordan grafen ser ut visuelt, selv om de i dette tilfellet ikke kan beskrives som nøyaktige nok.



*Figur 35: Viser prediksjonen til alle 4 spillere etter innføring av en ny funksjon som forskyver lags-periodene.*

### 6.4.3 Konklusjon

Prediksjonsevnen til RF tyder på at algoritmen har store vanskeligheter med å predikere når det kommer til opplevd variasjon i datasettet. Best resultat får vi når datasettet inneholder mer konstante variabler, noe vi ser når man sammenligner de grafiske fremstillingene og kalkulert test score. Dette observeres også gjennom de negative verdiene beregnet fra  $R^2$ , og betyr at modellen tar inn for mye støy fra datasettet og overtilpasser for å kunne håndtere ny inputdata (se seksjon [6.2](#)).

Vi ser at prediksjonene for Spiller3 overgår Dummy modellen i testene gjort i seksjon [6.4.1](#), men basert på en visuell analysering kan vi se at dette er et resultat av en nesten konstant gjennomsnittsverdi som prediksjon. På grunn av at testsettet i denne perioden varierer mellom kun et par verdier, blir dette i følge MSE gode prediksjoner. Sammenligner vi disse resultatene med de resterende spillerne kan man, basert på testingen i seksjon [6.4.1](#), konkludere med at RF-modellen ikke håndterer datasett med mye variasjon i loggførte verdier.

Resultatet vi fikk i seksjon [6.4.2](#) er en stor forbedring om man ser på den grafiske fremstillingen og sammenligner testene gjort i seksjon [6.4.2](#) med [6.4.1](#). Man kan argumentere for at beregnet test score ikke viser antydning til like god forbedring, og derfor konkludere med at implementasjonen av den nye funksjonen ikke hadde noen hensikt. Her er det viktig å understreke at måling av feilestimat hos en modell ikke er nok til å komme til en konklusjon, da dette er ment som et hjelpemiddel for å forstå hvordan man kan forbedre modellen. Med tanke på konteksten problemstillingen vår befinner seg i, beskrevet i seksjon [1.6](#), er det viktig å finne en modell som klarer å følge mønsteret i datasettet. I siste del av testingen klarte vi å finne en funksjon som sørget for at RF-modellen klarte dette, som vi mener er en stor forbedring som legger et godt grunnlag for videre eksperimentering.

## 6.5 ARIMA

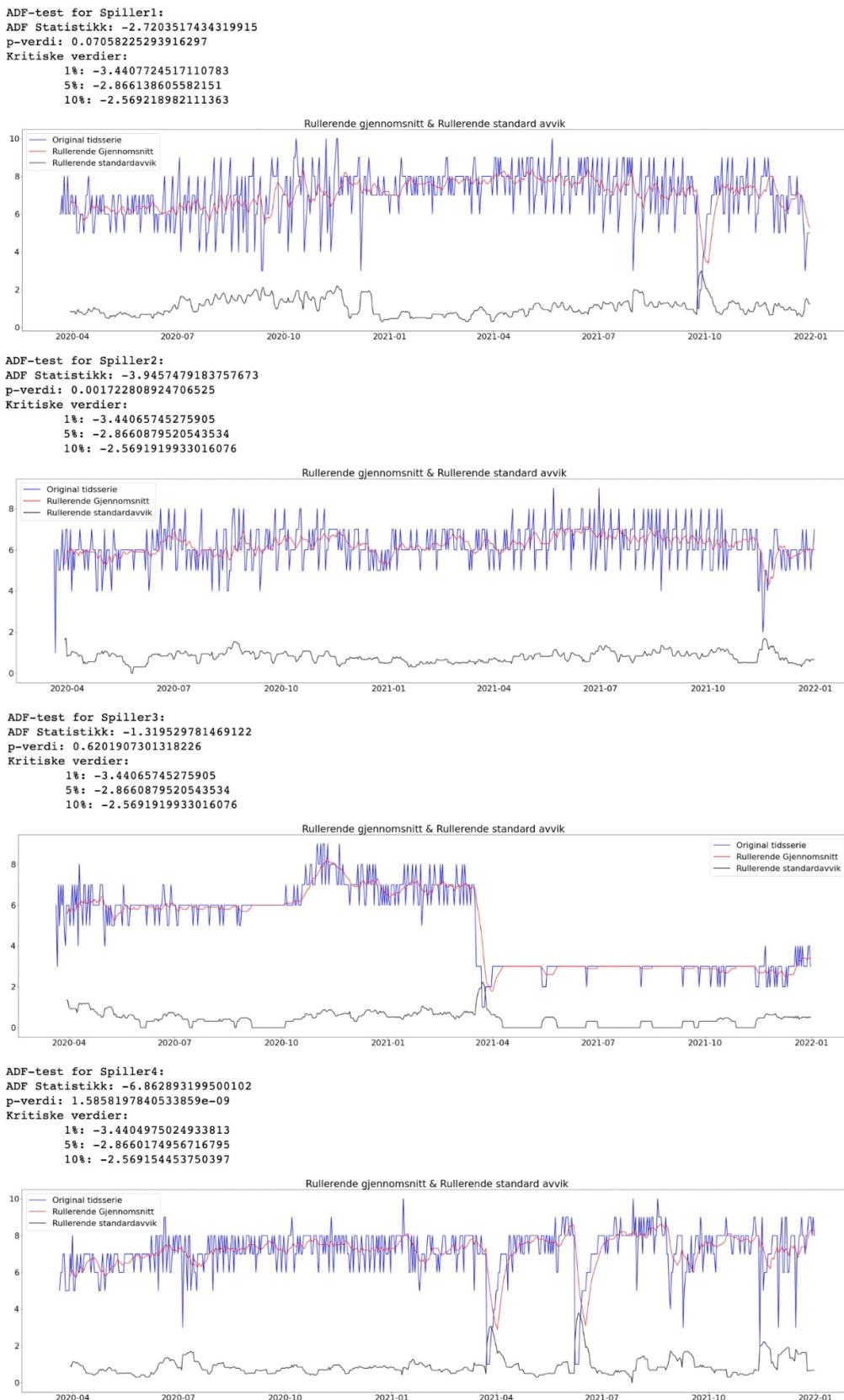
Når man arbeider med en ARIMA-modell, har man fra starten to retninger man kan velge; enten bruke en «auto»-versjon av modellen, eller gjennomgå en prosess for å finne parametrerne p, d og q manuelt. Disse parametrerne bestemmer hvordan ARIMA-modellen er satt opp og skal behandle input av data for prediksjon. Dette er forklart dypere i seksjon [5.3.2](#). I det første alternativet, auto-ARIMA, forsøker modellen selv å finne de mest optimale parameterverdiene. Her skjer hypertuning ved å kjøre ulike tester på modellen. For å kunne sammenligne, i tillegg til å utforske de presenterte mulighetene, har vi testet begge

versjonene av modellen. På denne måten kunne vi også få hjelp til hypertuning av den fulle ARIMA-modellen gjennom autoversjonen, som potensielt kan gi en pekepinn til et optimalt sett med parameterverdier.

For å kunne legge et godt grunnlag for ARIMA-modellen er det hensiktsmessig å teste tidsserien for stasjonaritet, beskrevet i seksjon [5.3.2](#). Dette betyr at vi bruker en ADF-test for å teste nullhypotesen som sier om det eksisterer en enhetsrot. Resultatet fra denne testen på alle spillerne kan man se under, i Figur 43. Den viktigste informasjonen vi får er en beregnet p-verdi, som har et signifikansnivå på 0.05. Verdier under dette antyder at tidsserien er stasjonær og at det ikke er behov for å differensiere datasettet. Det betyr at den ikke er påvirket av trender eller sesongmessige forhold. I samme oversikt fra ADF-testen vil vi få ut en statistisk verdi og tre kritiske verdier som brukes til sammenligning (Fuller, 1976, s. 642). Her vil vi se et resultat av ADF-statistikken som er i nærheten av de kritiske verdiene eller lavere.

I Figur 36 er resultatet fra de fire spillerne fremstilt, med oversikt over numeriske verdier og fremstilling av rullerende gjennomsnitt og standardavvik grafisk. Ut i fra oversikten er det kun Spiller2 som har en p-verdi under signifikansnivået, men Spiller1 er også relativt nært. Ser man på fremstillingen i Figur 43, og sammenligner med Figur 18 i seksjon [4.3.1](#) om tidsserier, er det ingen av grafene som bærer tydelige preg av å inneholde verken trender eller sesongmessige forhold. Selv om Spiller3 og Spiller4 ikke nødvendigvis tilfredsstiller de numeriske kravene for stasjonaritet, vil vi ut i fra den grafiske fremstillingen beskrive alle som stasjonære. Resultatet til Spiller3 og Spiller4 vil likevel bli tatt hensyn til, og testing av ulike differensieringsgrader vil bli gjennomført under hypertuning av parametrene som gjøres i seksjonene [6.5.1](#) og [6.5.2](#).

Testingen med hver av de to ARIMA-modellene vil forklares i hver sin seksjon for å holde det oversiktlig angående hvilken modell det er snakk om.



Figur 36: Fremstilling av ADF-test gjort på de fire utvalgte spillerne.

### 6.5.1 Auto-Arima

Implementasjon av auto-ARIMA er, i motsetning til den manuelle prosessen ved å bygge ARIMA-modellen, en relativt enkel prosess. Hver parameter i auto-ARIMA har en standardverdi som blir brukt i modellens implementering. Man har likevel muligheten til å legge inn verdier til en god del parametere i selve funksjonen, dersom dette er noe man har behov for.<sup>48</sup> Vår implementasjon av modellen innebar hovedsakelig en spesifisering av tidsfrekvens og lag-verdi, som er definert i [4.3.3](#). Dette blir brukt til å finne den mest optimale sammensetningen av p, d og q ved hjelp av å kontinuerlig sammenligne AIC-tester. Etter disse testene blir modellen trent og testet med resulterende verdier som innstillinger.

Siden datasettet til hver enkelt spiller er unikt, vil også beregnet sammensetning av p, d og q bli forskjellig. Det som er identisk for alle modellene er forhåndsinnstillingene, samt utgangspunktet for konkrete aspekt ved testingen. Dette ble tydeliggjort i seksjon [6.1](#). Etter å ha gjennomført flere tester får vi varierende resultat i prediksjonen. For å gi et tydelig bilde på variasjonen fremstilles prediksjonene til alle fire spillere, kombinert med test score, i Figur 37.

Sammenligner vi de ulike estimatene for test score fra Figur 37 med verdiene fra dummy modellen i Figur 28 under seksjon [6.3](#), viser både Spiller1, Spiller2 og Spiller4 litt høyere verdier enn baseline. Sammenligner vi Spiller3 med dummy modellen kan vi se at vi allerede har klart å få et bedre resultat enn forventet. Ser vi derimot litt nærmere på grafen og verdiene på y-aksen, har det kun blitt registrert verdier fra og med 2 til og med 4 i testperioden. Dette kan dermed ha bidratt til å forenkle prediksionsarbeidet til modellen, da spennet i registrerte verdier er såpass lite. Hos alle de andre spillerne i dette eksempelet går registrerte verdier fra og med 2 til og med 9 i testperioden. Det er tydelig at dette påvirker modellens evne til å predikere nøyaktige resultat.

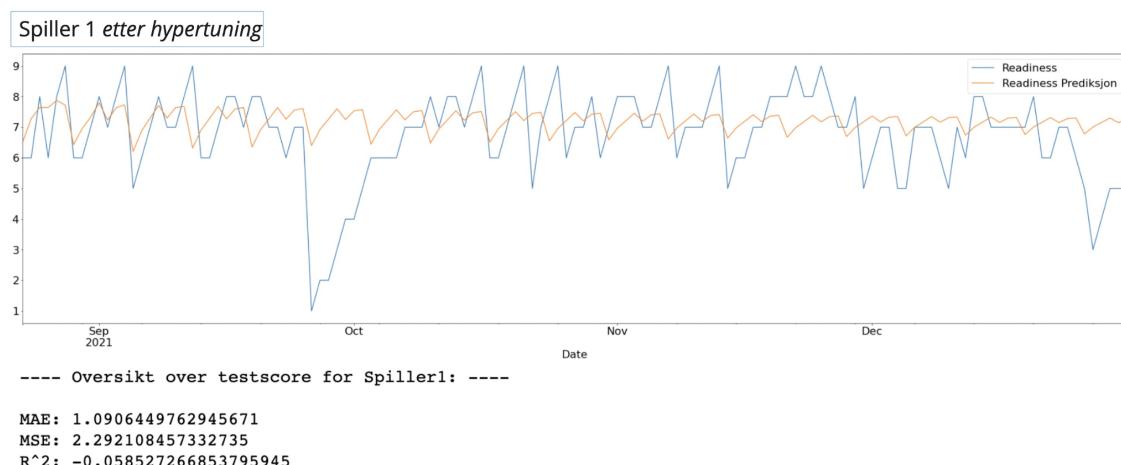
---

<sup>48</sup> [https://alkaline-ml.com/pmdarima/0.9.0/modules/generated/pyramid.arima.auto\\_arima.html](https://alkaline-ml.com/pmdarima/0.9.0/modules/generated/pyramid.arima.auto_arima.html)



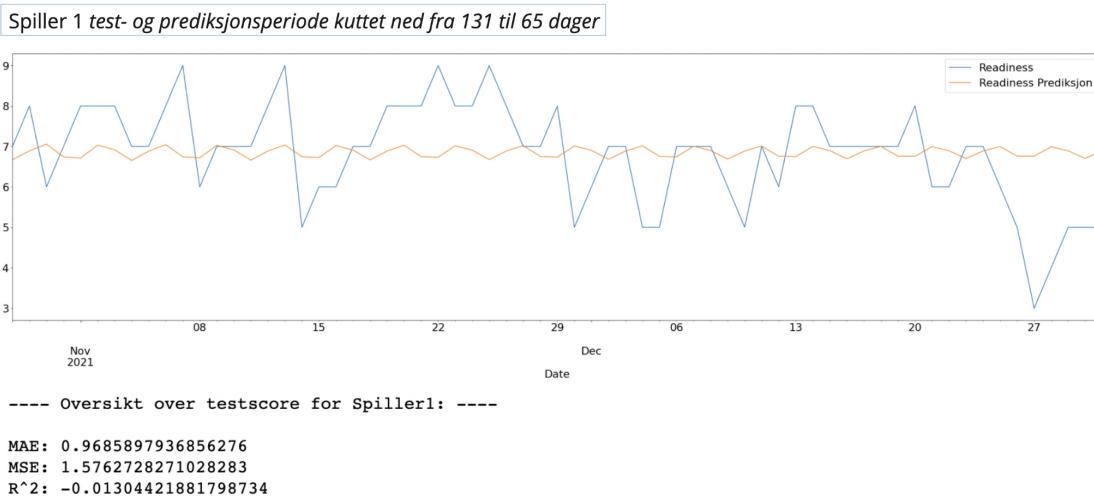
Figur 37: Prediksjon av alle spillerne i tidsperioden 2021-08-23 til 2021-12-31.

Vi kan også observere fra Figur 37 at utslaget til prediksjonen ikke er like stort som vi skulle ha ønsket. Det kan se ut som at den legger seg rundt en kontinuerlig gjennomsnittsverdi, og størrelsen på utslaget varierer ut i fra hvor stort spenn det er i de faktiske registrerte verdiene. Forsøk på å hypertune auto-ARIMA med verdien til input parametrene gir ikke statistiske bedre resultat enn hva vi allerede har oppnådd hos de fleste av spillerne, bortsett fra hos Spiller1. Her har det blitt gitt et større spillerom i testing av minimum- og maksimumsverdi for antall lag som skal testes. Dette resulterer i lavere test score og en tydelig forskjell i grafisk fremstilling, fremstilt i Figur 38.



*Figur 38: Resultatet til Spiller1 etter hypertuning av parametere.*

Vi tester deretter å korte ned prediksionsperioden, og velger en fordeling på 90% og 10% av trenings- og testsett. Dermed blir treningsperioden på 586 verdier og test- og prediksionsperioden blir totalt 65 verdier. Ved å ta i bruk satte utgangspunkt for innstillinger (se seksjon 6.1), utgjorde ikke dette noen stor forskjell for testresultatet hos noen av spillerne. Resultatet fra AIC-testene viste de samme ARIMA-modellene som ble ansett som mest optimale i tidligere forsøk, og det var marginale forskjeller på beregnet test score. Om en spiller fikk et bedre testresultat kunne man også se at det originale testsettet hadde mindre variasjoner i de registrerte verdiene. Store og plutselige opp- eller nedganger i verdiene viser seg å påvirke modellens evne til å predikere i stor grad. Dette ser man tydelig hos Spiller1. Sammenligner vi resultatet i Figur 39 under med Figur 38 over, ser man at den store og plutselige nedgangen ikke lenger er en del av prediksjonperioden. Derfor kan det se ut som at dette har hjulpet testresultatet, og prediksjonene passer bedre til de mer gjennomsnittlige loggførte verdiene.



Figur 39: Resultatet til Spiller1 etter forkortet test- og prediksjonsperiode.

Det ble forsøkt å kutte ned test- og prediksjonsperiode enda litt mer, uten at dette ga vesentlig bedre resultat. Vi ser de samme tilfellene som fra tidligere resultat, men det er ingenting som skiller seg ut eller fremstår som mer optimalt.

### 6.5.2 Manuell ARIMA-modell

Fra testingen gjort av auto-ARIMA-modellen ble vi presentert for de parameterverdiene som modellen anså som mest optimale. Disse sammensetningene av p, d og q ble derfor tatt i betraktnsing under forsøkene gjort med den manuelle ARIMA-modellen. Vi forsøkte å manuelt komme frem til disse verdiene ved å gjennomføre egne tester, men det viste seg at vi måtte hypertune disse resultatene en del. Resultatene vi hadde fra før ble en god retningslinje vi kunne jobbe etter, samtidig som de manuelle testene bidro til en større forståelse av hva de enkelte verdiene betyr for modellen.

Vi observerte raskt at auto-ARIMA hadde beregnet noen av de beste sammensetningene som resulterte i best mulig prediksjon og test score, etter å ha sammenlignet disse resultatene med våre egne beregninger. Dermed ble det utført tester med samme verdier som i seksjon [6.5.1](#), og dette resulterte i de samme grafene og beregningene av test score. Hos Spiller4 fant vi en mer optimal modell, som resulterte i en bedre test score og resultat nærmere baseline satt av dummy modellen. Ser man på selve prediksjonsverdiene, i tillegg til den grafiske fremstillelsen, er det tydelig at vi får en prediksjon som baserer seg på gjennomsnittet av verdiene i datasettet.

### 6.5.3 Konklusjon

Arbeidet med å hypertune en ARIMA-modell i vår kontekst ga oss varierte resultater. Verdiene og innstillingene vi hadde satt som utgangspunkt viste seg å gi gode verdier å starte med når det gjaldt prediksjon og test score. Forsøk på å forbedre prediksjonsevnen var ikke like enkelt, da auto-modellen til ARIMA baserer seg på å nettopp gjøre dette arbeidet fra starten av. På bakgrunn av disse innstillingene og grenseverdiene vi hadde avgjort i seksjon [6.1](#), viste dette seg å være det mest optimale når det gjaldt prediksionsarbeidet med ARIMA-modellene og resultatene vi fikk..

I ettertid har vi innsett at frekvensparameteren i auto-ARIMA mest sannsynlig hadde vært lovende å gjøre mer testing og observasjoner rundt. Frekvensen til den registrerte dataen er daglig, og er informasjon som tilføres modellen man bruker. I sluttprosessen av rapporten dukket det opp informasjon som antydet at man kunne ha testet å endre på denne basert på hva slags informasjon som var ønskelig å predikere.<sup>49</sup> Om tiden hadde strukket til hadde vi valgt å fokusere på prøve ut ulike varianter av frekvensparameteren for å se om dette ga bedre utslag i prediksjonen.

Det overordnede inntrykket av ARIMA er at det kan være en svært god modell for prediksjon i den rette konteksten. Under våre forsøk opplevde vi de fleste tilfellene at den beste løsningen fra modellen sin side var å basere prediksjonene på et gjennomsnitt, med små utslag i form av topper og bunner. På bakgrunn av vår problemstilling er ikke dette et resultat vi kan anse som godt nok, og det vil være nødvendig med noe som er mer komplekst i forståelse av selve dataen.

## 6.6 LSTM

Etter å ha eksperimentert med modellene RF i seksjon [6.4](#) og ARIMA i seksjon [6.5](#), skal vi nå se nærmere på Long Short Term Memory (LSTM). Dette er en Deep Learning (DL) modell, som betyr at den er mer kompleks og avansert sammenlignet med RF og ARMA. Som det ble nevnt i seksjon [5.3.3](#) har modellen et såkalt «long short» minne hvor den husker informasjon over tid, samtidig som den glemmer det som er irrelevant. Dette betyr at den er godt egnet til arbeid med tidsseriedata, og det er derfor vi har valgt å eksperimentere med den i vårt prosjekt.

---

<sup>49</sup> [https://alkaline-ml.com/pmdarima/tips\\_and\\_tricks.html](https://alkaline-ml.com/pmdarima/tips_and_tricks.html)

## 6.6.1 Oppbygging av modell

For å bygge LSTM-modellen tok vi inspirasjon fra en artikkel som predikerer aksjepris ved hjelp av LSTM.<sup>50</sup> Artikkelen bygger modellen med 50 neurons og 4 hidden layers, kombinert med et neuron i output layer for å forutsi den normaliserte verdien for prediksjonen man ønsker (se Figur 10 under seksjon [3.2.2](#) for eksempel på et konstruert Neural Network [NN]). Modellen bruker også 100 epoker og en batchstørrelse på 32. Datasettene vi bruker kan fremstå som små for bruk i en typisk DL-modell, men vi har nok datapunkter i settene til å kunne trenne LSTM-modellen og gjøre prediksjoner. I vårt prosjekt blir det brukt en enkel versjon av modellen, noe som gjorde det lettere å gjenta og tilpasse eksperimentene.

Til forskjell fra RF og ARIMA klarer LSTM å håndtere manglende verdier i et datasett. Likevel har vi valgt å håndtere verdiene manuelt med metoden forklart i seksjon [4.1.3](#). Utgangspunktet for verdiene som blir brukt under testingen av LSTM er gjort rede for i seksjon [6.1](#), sammen med oppdelingen av trening- og testsett.

## 6.6.2 Gjennomføring av eksperimentene

Eksperimentene som har blitt gjennomført i denne rapporten baseres på et utvalg på fire enkeltspillere og deres datasett, som kan ses ferdigstilt i seksjon [4.2](#). Grafene til hver spiller, som viser den selvrapporterte parameteren Readiness to Play, viser fire ganske ulike datasett. Dette gjør at vi kan teste om modellen klarer å få en like bra prediksjon på hver enkelt, til tross for at individer kan ha ulik oppførsel i sine loggførte mønstre.

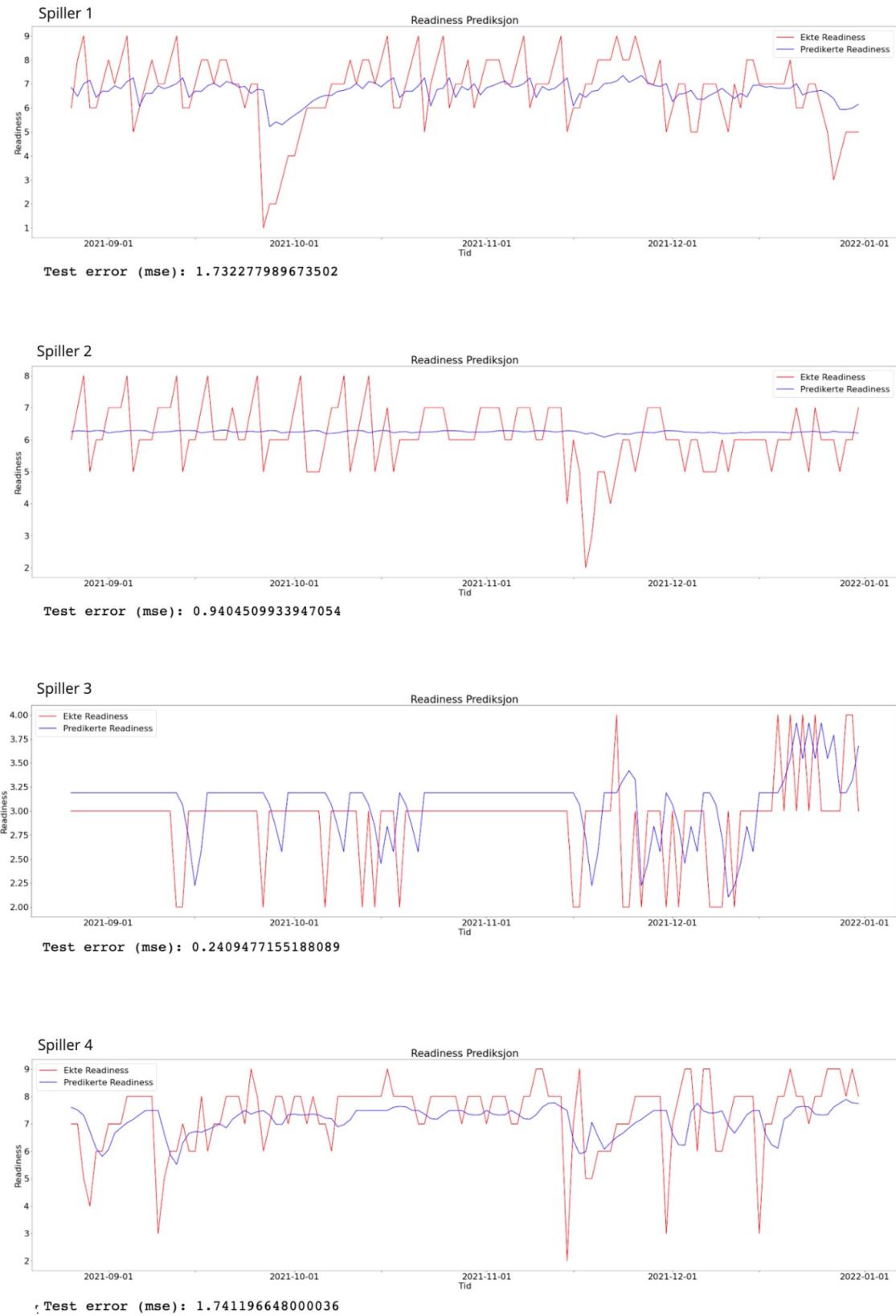
I Figur 40 ser vi resultatet av prediksjonen vi får, fremstilt med hver enkelt spiller i hver sin graf. Modellene resulterte i en relativt god prediksjon av fremtidige verdier på enkelte av spillerne, men man kan observere at modellen har mye å gå på. Vi kan anta at prediksjonene hadde blitt forbedret om modellen hadde blitt trent på et større datasett, spesielt siden modeller innenfor DL gir mer nøyaktige resultat om de har lært av et stort antall datapunkter.<sup>51</sup>

---

<sup>50</sup><https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f>

<sup>51</sup>

<https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/>



Figur 40: Prediksjon av Readiness-parameter kombinert med faktiske loggførte verdier hos de fire spillerne. X-aksen er tid vist ved måneder og y-aksen er readiness to play, hvor skalaen går fra 0 til 10. Inkluderer også

*test error til hver spiller.*

Vi observerer på Figur 40 at prediksjonene både kan karakteriseres som gode og mindre gode for de fire spillerne. Hos Spiller1, Spiller3 og Spiller4 ser man at prediksjonen klarer å følge de vanlige trendene til de faktiske loggførte datapunktene. I motsetning til dette observerer vi at Spiller2 sin prediksionsverdi kun klarer å holde seg rundt engjennomsnittsverdi til den faktiske Readiness-parameteren. Årsaken til dette kan være for lite data, slik at modellen ikke klarer å lære seg spillerens mønster. Observerer vi denne grafen nøyne, kan vi se at modellen prøver å følge spillerens mønster, men får ikke like stort utslag som hos de tre andre. Vi ser at modellen til en viss grad klarer å fange opp eventuelle «topper» hvor spillerne er ekstra klare til kamp, eller «bunner» hvor spilleren ikke føler seg like klar.

	Predikerte Readiness	Ekte Readiness
1	7.245555	6.0
2	7.011645	8.0
3	7.464703	9.0
4	7.649073	6.0
5	7.141888	6.0
6	7.179410	7.0
7	7.083778	8.0
8	7.363984	7.0
9	7.340334	8.0
10	7.597483	9.0

*Figur 41: Viser hva den predikerte verdien er forhold til den ekte verdien for Spiller1. Verdiene er hentet ut fra starten av test-settet - de første 10 dagene.*

For å kunne se på enkeltverdier av prediksjonene sammenlignet med faktiske loggførte Readiness-verdier, fremstiller vi dette i Figur 41. Som vi observerer her, viser det seg at modellen ikke treffer særlig godt på den faktiske Readiness-verdien. Likevel kan vi bemerke at verdien prøver å justere seg etter hva de foregående verdiene er. For eksempel, når man i Figur 41 ser på den loggførte verdien fra rad 1 til 2 går verdien fra 6 til 8. Vi observerer at Spiller1 føler seg mer klar basert på en betydelig økning av denne verdien, og modellen antyder også at spilleren vil føle seg mer klar neste dag med en økning i

prediksjonsverdi. Som vi ser på rad 3, hvor verdien til den predikerte Readiness-parameteren er 7.46, predikerer da modellen at spilleren vil oppleve en høyere verdi av Readiness to Play. Her ser vi antydninger til at modellen klarer å forutsi stigningen av loggført verdi. Dette kan fremstå som en riktig tolkning av modellen, men kan også være en tilfeldighet. Når det er sagt, observerer vi også visuelt med grafene i Figur 40 at modellen, spesielt for Spiller3 og Spiller4, klarer å predikere verdier tett opp mot de faktiske registrerte Readiness-verdiene i datasettene.

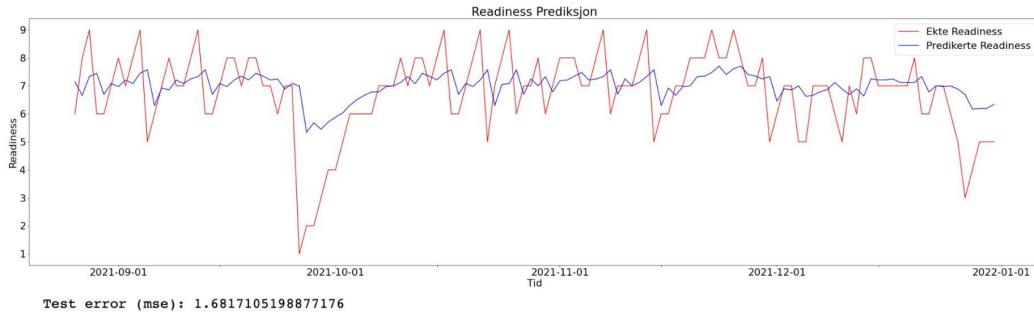
Ved å bruke en test error, forklart i seksjon [6.2](#), kan vi få en indikator på hvor god prediksjonsevne modellene har. Test erroren som blir brukt på denne modellen er MSE. Som vi ser på Figur 40 er MSE-verdiene for hver spiller følgende; Spiller1: 1.73, Spiller2: 0.94, Spiller3: 0.24 og Spiller4: 1.74. Den observerte MSE-verdien har et spenn fra 0.2 til 1.7. Det er ulike grunner til hvorfor vi får så store forskjeller i de beregnede verdiene for prediksionsresultatene til hvert datasett. En av årsakene kan være forskjell på spennetav de loggførte verdiene av Readiness to Play blant spillerne. Spiller1, som har test error på 1.73, har loggført verdier fra 0 til og med 9, i motsetning til Spiller 3, med en test error på 0.24, som i prediksionsperioden kun har loggførte verdier fra 2 til 4. Dette indikerer at det er mye enklere å predikere en spiller som har konstante og regelmessige verdier, fremfor en spiller som loggfører uregelmessige og ulike verdier for Readiness to Play. Observasjonen gjort hos Spiller3, hvor man ser en plutselig nedgang i verdien som blir loggført for Readiness-parameteren, viste seg å være på grunn av en pådratt skade og forklares nærmere i seksjon [6.9](#).

Resultatet av test error fra MSE er både bedre og dårligere enn terskelverdiene til Dummy modellen, gjort rede for i seksjon [6.3](#). Hos Spiller1, Spiller2 og Spiller3 får vi bedre resultat, men Spiller4 får en dårligere test score. Fra dette resultatet ser vi at LSTM-modellen gir lovende prediksionsresultat i 3 av 4 datasett vi har testet på. Basert på resultatet av MSE-verdiene, og studering av prediksionsgrafene, ser vi tydelig at selve prediksjonsevnen likevel varierer fra datasett til datasett.

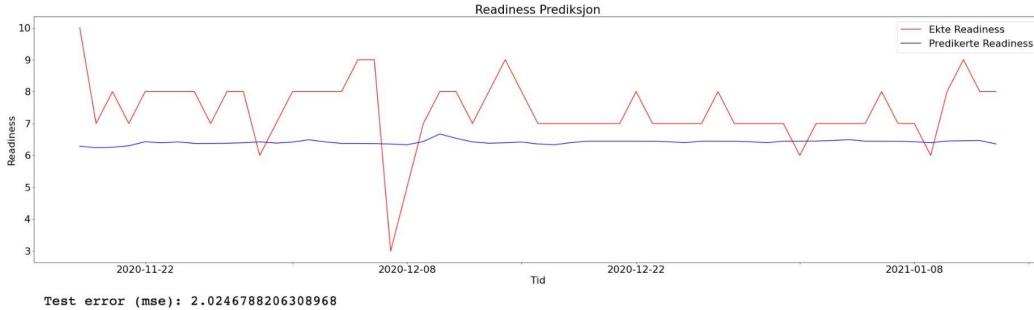
Etter å ha gjennomført eksperimentene i denne seksjonen observerer vi at det er ulike faktorer som påvirker hvor gode prediksjoner LSTM-modellen gir oss. Blant annet er det mye lettere for modellen å predikere relativt riktig for en spiller som har konstante og regelmessige verdier, fremfor en spiller som loggfører uregelmessige og ulike verdier for Readiness to Play. Et eksempel på dette ser man om man sammenligner grafene til Spiller3 og Spiller4 i Figur 40. En annen grunn kan være at en spiller har et lettere og et mer «lesbart» mønster i sine loggførte verdier, og kan på grunn av dette være enklere å predikere i forhold til en spiller som har en Readiness-verdi som kan oppfattes som mer

tilfeldig. Hvis spilleren også har loggført verdier med mindre spenn, slik som Spiller3, vil dette gjøre det lettere for modellen å predikere en tilnærmet lik verdi.

### Spiller1 - periode på 652 dager:



### Spiller1 - periode på 300 dager:



*Figur 42: To eksempler fra Spiller1 som viser forskjellig resultat og test error fra MSE kun basert på ulik datamengde.*

Som nevnt i seksjon [5.3.3](#), utfører LSTM bedre prediksjoner med en større datamengde sammenlignet med en mindre datamengde. Dette kan vi også observere i et eksperiment hvor vi testet å gi modellen mindre tilførsel av data, som ga oss et dårligere resultat. I Figur 42 ser vi i den øverste grafen at LSTM-modellen har bearbeidet og bygget opp modellen på 652 dager - hele datasettet som er tilgjengelig. Modellen har klart å få en relativt god graf. Vi kan observere at de fleste predikerte datapunktene følger den faktiske verdien til Readiness-parameteren. Til motsetning ser vi på den nedre grafen at modellen kun har fått 300 dager til å bygge opp en prediksjon, noe som resulterer i en dårligere prediksjonsevne i forhold til øvre graf. Grafen viser at modellen ikke klarer å lære seg spillerens mønster, og holder seg på en beregnet gjennomsnittsverdi under prediksjonsperioden. Sammenligner vi MSE-resultatene fra eksperimentet, ser vi at resultatet er dårligere med et mindre datasett.

Tester man ulike datamengder og observerer den beregnede test erroren, kan man oppleve at den gir et varierende resultat. Dette kan blant annet være et resultat av perioden det skal gjøres prediksjoner for. Om spilleren kun har loggført verdier kun mellom 5 og 7, vil

ikke prediksjonen nødvendigvis oppfattes som like unøyaktig om prediksjonsverdien legger seg på 6 som et gjennomsnitt. Dette ser vi spesielt hos Spiller2 i Figur 40. Dette gir visuelt en nesten helt rett og jevn linje, som ikke følger datasettet i like stor grad som hos de andre spillerne.

### 6.6.3 Konklusjon

Prediksjonsevnen til LSTM kan oppleves som varierende basert på resultatene vi har fått for de fire spillerne og deres datasett. Vi studerer ulikheter både visuelt i grafene i Figur 40 og bemerker en rekke forskjeller ved beregnet MSE-verdi. I de testingen opplevde vi at Spiller1, Spiller2 og Spiller3 hadde en bedre test score enn terskelverdien satt av dummy modellen (se seksjon [6.3](#)), hvor Spiller4 får et dårligere resultat. Sammenligner vi grafene til alle spillerne kan det være mulig at LSTM-modellen ikke klarte å lære seg mønsteret til Spiller4 i like stor grad. Vi kan se i Figur 40 og i grafen til Spiller 4 at det oppstår noen store og plutselige endringer i løpet av prediksjonsperioden. Dette skaper større forskjeller i registrerte verdier, og gir modellen større utfordringer med å predikere nøyaktige verdier som er i nærheten av de faktiske datapunktene. Dermed kan vi konkludere med at modellen i enkelte datasett kan være «heldigere» med prediksjonsevnen.

Ser vi på den grafiske fremstillingen av prediksjonen hos Spiller1, Spiller3 og Spiller4, er vi av den oppfatning at modellen klarer å gi tilfredsstillende prediksjoner som følger mønsteret i datasettene. Likevel er det tydelig at vi ikke får helt nøyaktige prediksjoner. Ser vi på Figur 41, hvor vi sammenligner prediksjonsverdier med loggførte datapunkter for Spiller1, ser vi at prediksjonene ikke alltid stemmer. Men, som det ble påpekt i seksjon [6.6.2](#), ser vi at den predikerte Readiness-parameteren til en viss grad øker og minker i takt med faktiske Readiness-verdier. LSTM klarer å fange opp eventuelle «topper» og «bunner» til spilleren, som man kan se fra grafene til hver enkelt spiller i Figur 40. Dette er uten tvil en viktig observasjon, fordi det er hensiktsmessig å få oversikt over spillerens antagelige toppler eller bunner før viktige kamper.

Basert på eksperimentet fremstilt i Figur 42 har vi tydeliggjort at modellen i vårt tilfelle har en betydelig bedre prediksjonsevne når den får et større datasett som input. Vi kan derfor konkludere med at denne modellen har et lovende potensiale for videre eksperimentering og testing, etter hvert som man får tilgang til større datasett enn det vi har brukt.

## 6.7 Sammenligning av prediksjonsevne

En sammenligning av de beste resultatene innenfor test score og prediksjonsevne fra hver enkelt modell viser at det ikke er mye som skiller de fra hverandre. Dette er derimot kun tilfelle hvis man skal gi en vurdering basert på en prestasjonsdefinisjon presentert gjennom numeriske verdier. ARIMA og RF deler på flere av de samme observerte problemene, og det har blitt tydeliggjort under arbeidet at regression ikke er en optimal modell for vår problemstilling, beskrevet i seksjon [1.6](#). Forsøk på å hypertune disse modellene har ikke gitt så gode resultat som vi hadde håpet. Den eneste spilleren som får et bedre resultat enn dummy modellen er Spiller3, hos både ARIMA og RF. Dette er, etter vår mening, tilfelle av et heldig sammentreff med kombinasjon av tidspunkt og registrerte verdier med lite spenn. Vi får derfor et resultat som er et konstant gjennomsnitt av kun to registrerte verdier, og dette gir oss en god test score.

LSTM får en MSE-score som overgår baseline hos alle spillerne, bortsett fra Spiller4. Den grafiske fremstillelsen av prediksjonen viser at verdiene i større grad følger den originale tidsserien, noe som understreker en mer nøyaktig kalkulasjon av prediksjonsverdiene. I denne modellen ser man også at en tidsserie bestående av mer uregelmessige verdier gjør det mer utfordrende å gi gode prediksjoner, i motsetning til om en spiller har mindre spenn i topp- og bunnverdiene. Ser vi kun på test scoren til alle modellene, kan man trekke en konklusjon om at ingen av modellene skiller seg sterkt ut som aktuelle.

En viktig detalj som skiller LSTM fra RF og ARIMA er at den lærer på en helt annen måte, og lagring og prosessering av data er mye mer komplekst. Man kan argumentere for at beregnet test scor for resultatene til LSTM ikke var bedre, men her må vi ta hensyn til konteksten for å kunne trekke videre konklusjoner. Selv om det er numeriske verdier som skal predikeres, sier disse utregningene noe om en menneskelig tilstand. Prediksjonene vil ha innvirkning på beslutninger som kan påvirke en helsetilstand. Dette er en av grunnene til at vi ikke kan basere valget av modell kun på test score, men må gjøre en vurdering av helheten.

## 6.8 Eksperimentering med Convolutional Neural Network

I seksjon [5.3.3](#) om LSTM ble det henvist til en rapport som viser at at modellen Convolutional Neural Network (CNN) i enkelte tilfeller kan prestere bedre enn Long Short-Term Memory (LSTM), både når det gjelder prediksjon og kjøretid. På grunn av de lovende resultatene fra eksperimenteringen med LSTM, ville vi bruke litt tid på kjøre noen

tester med CNN. Modellen er hovedsakelig brukt innenfor feltet «Computer Vision», der man forsøker å lære opp datamaskiner til å se verden rundt, altså å forstå og trekke mening ut av den visuelle verdenen rundt oss. Dette betyr at CNN-modellen er mest brukt på todimensjonal data, slik som bilder og video, men med litt justering av algoritmen og datasettet kan man også bruke den på endimensjonal data, slik som tidsserier. Dette betyr at CNN kan brukes til å predikere Readiness-parameteren. Vi vil ikke gå veldig dypt inn i teorien av denne modellen, hvor det grunnleggende er presentert i seksjon [3.2.2](#), men gjøre noen få tester for å se om den kan produsere gunstige prediksjoner. Slik som andre Neural Network (NN), består et CNN av flere layers (se Figur 10 i seksjon [3.2.2](#)). For å lage prediksjonsmodellenmodellen har vi hentet inspirasjon fra en artikkel om hvordan man setter opp CNN-modellen for tidsserier.<sup>52</sup> Det som omtales som hidden layers i modellen i dette eksempelet er ett convolutional, ett pooling, et flatten og et dense layer. Man kan bruke flere Convolutional layers, men dette burde ikke være nødvendig for oss. Grunnen til dette er fordi dataen vi bruker ikke er spesielt komplisert. I tillegg til hidden layers, har modellen også et input og output layer. Vi tok i bruk de samme utgangspunktene for startverdier og innstillinger som ble presentert i seksjon [6.1](#) når vi gjennomførte testene med CNN. Slik som i testene gjennomført i kapittel [6](#), har vi i dette tilfellet også eksperimentert med en spiller. Det er også kun utviklet en modell til hver spiller, basert på den spillerens registrerte Readiness-parametre. Resultatene fra modellene er fremstilt i grafene under, i Figur 43.

---

<sup>52</sup><https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/>



Figur 43: Fremstilling av resultatet fra eksperimentene med CNN-modell. Her ser vi de grafiske resultatene til hver enkelt spiller, sammen med en utregnet MSE-verdi.

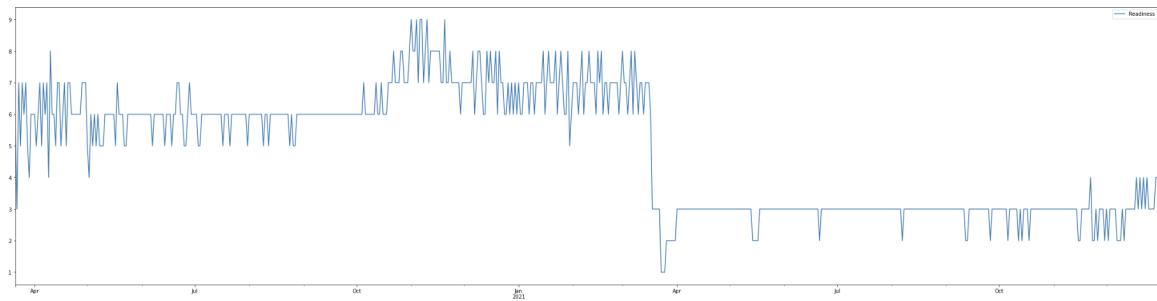
Man kan se at prediksjonen visuelt følger de faktiske loggførte verdiene i test settet. Den estimerte test erroren MSE, forklart i seksjon [6.2](#), er hos alle fire spillere lavere enn den estimerte baselinen fra dummy modellen i seksjon [6.3](#). Ser vi på resultatet av estimert MSE-verdi i Figur 40 fra eksperimentet med LSTM-modellen i seksjon [6.6.2](#) og sammenligner med Figur 43 over, kan vi observere at CNN-modellen har lavere MSE-verdi for Spiller1 og Spiller 2, og LSTM har lavere verdi for Spiller3 og Spiller4. Vi konkluderte i seksjon [6.6.3](#) med at LSTM-modellen ga mindre nøyaktige prediksjoner når spennet i de registrerte verdiene i datasettet var større. Det kan dermed tyde på at CNN er noe bedre til å predikere Readiness-verdier for spillere med mer variasjon i de registrerte tidsstegene.

Basert på de få testene vi fikk gjort, ser vi at det er mye potensiale i de oppnådde resultatene. Eksperimentet viser at det er enkelte likheter med resultatene fra LSTM-modellen, både når det gjelder grafisk fremstilling av prediksionsresultat og test error. Vi kan dermed konkludere med at NN-modeller fremstår som de mest optimalene under arbeidet med prediksjon av Readiness-parameter.

## 6.9 Eksperimentering med prediksjon av skade

Gjennom prediksionsarbeidet med modellene i seksjonene [6.4](#), [6.5](#) og [6.6](#) fikk vi et nytt syn på de loggførte Readiness-verdiene til spillerne. Dette gjaldt spesielt for Spiller3, som vi ser grafen til visuelt presentert i Figur 44. I begynnelsen av datasettet er det registrert relativt høye verdier, ofte fra og med 6 til og med 9. Etter en gitt tidsperiode, omtrent ved inngangen til april, ser vi at grafen får et plutselig drastisk fall. Ut ifra grafen tyder det på at en hendelse kan ha inntruffet, som videre har hatt en påvirkning på de resterende loggførte verdiene i datasettet.

For de predikerte Readiness-verdiene hadde Spiller3 i testingen av alle modellene veldig lav estimert MSE-verdi, sammenlignet med dummy modellen med en MSE verdi som var 3 ganger så høy. For eksempel på dette, se estimert MSE-verdi fra dummy modell for Spiller3 presentert i Figur 28 under seksjon [6.3](#), sammenlignet med predikert MSE-verdi i Figur 40 under seksjon [6.6.2](#). Resultatet fra de andre spillerne viste mindre forskjell mellom dummy modellen og prediksjonene som ble gjort. Det ble spekulert i om det var noe annerledes med resten av Spiller3 sin loggførte informasjon, noe som førte til at vi oppdaget en registrert skade i datasettet.



*Figur 44: Fremstilling av datasettet til Spiller3 som viser registrerte Readiness-variable fra 20.03.2020 - 31.12.2021.*

Figur 44 viser klart at fra slutten av mars 2021 og utover 2021 har verdiene endret seg drastisk fra et gjennomsnitt i Readiness-verdiene på 5-7 ned til 2-3. Dette førte til at vi analyserte det gitte datasettet med andre parametere for å forstå endringen. Det er tre parametere som har noe med denne hendelse å gjøre, og de kommer fra et annet datasett med annen registrert informasjon (se seksjon [1.4](#)). Disse parametrene er Game Performance, Illness og Injury. Ved å sjekke alle datoene i mars 20201, oppdaget vi at Injury-parametren skilte seg ut. Det viste seg at Spiller3 hadde fått en alvorlig kneskade på høyre bein 25.03.2021, og dette forklarte hvorfor de loggførte verdiene plutselig hadde blitt veldig lave.

	Date	Mood	Fatigue	Readiness	SleepDurH	SleepQuality	Sorenness	Stress
355	2021-03-10	3.0	2.0	6.0	6.5	2.0	2.0	3.0
356	2021-03-11	3.0	3.0	7.0	9.0	3.0	2.0	2.0
357	2021-03-12	4.0	3.0	7.0	8.0	4.0	2.0	3.0
358	2021-03-13	3.0	3.0	6.0	9.0	4.0	2.0	3.0
359	2021-03-14	4.0	2.0	7.0	8.0	3.0	2.0	3.0
360	2021-03-15	3.0	2.0	7.0	6.5	3.0	3.0	2.0
361	2021-03-16	3.0	3.0	7.0	9.0	3.0	2.0	3.0
362	2021-03-17	2.0	2.0	6.0	5.0	2.0	2.0	2.0
363	2021-03-18	3.0	3.0	3.0	8.0	4.0	3.0	2.0
364	2021-03-19	4.0	3.0	3.0	10.0	4.0	3.0	3.0
365	2021-03-20	4.0	2.0	3.0	7.0	3.0	3.0	3.0
366	2021-03-21	4.0	3.0	3.0	10.0	4.0	3.0	3.0
367	2021-03-22	3.0	3.0	3.0	6.0	3.0	3.0	3.0
368	2021-03-23	4.0	3.0	1.0	6.0	3.0	3.0	2.0
369	2021-03-24	4.0	3.0	1.0	9.0	4.0	3.0	3.0
370	2021-03-25	3.0	3.0	1.0	10.0	4.0	3.0	3.0
371	2021-03-26	3.0	3.0	2.0	9.0	3.0	3.0	3.0
372	2021-03-27	3.0	3.0	2.0	9.0	3.0	3.0	3.0
373	2021-03-28	3.0	3.0	2.0	7.5	3.0	3.0	2.0
374	2021-03-29	3.0	2.0	2.0	5.5	3.0	3.0	2.0
375	2021-03-30	3.0	2.0	2.0	7.0	3.0	3.0	2.0
376	2021-03-31	3.0	3.0	2.0	8.0	3.0	3.0	3.0
377	2021-04-01	3.0	3.0	3.0	NaN	4.0	3.0	3.0
378	2021-04-02	4.0	3.0	3.0	8.0	4.0	3.0	3.0
379	2021-04-03	4.0	2.0	3.0	8.5	2.0	3.0	2.0

Figur 45: Viser alle parametere til Spiller 3 i tidsperioden 10.03.2021 - 03.04.2021

Figur 45 viser alle parametrene for generell velvære noen dager før skade 25.03.2021, og 8 dager etterpå. Her kan man se at av alle verdiene er det kun de som er registrert under Readiness-kolonnen som har endret seg drastisk. Fra og med 18.03.2021 kan vi observere at Readiness-parameteren har gått fra verdier mellom 6 og 7 rett ned til 3 i totalt 5 dager, før det synker helt ned til 1 noen få dager før skaden inntraff.

Denne observasjonen gjorde at vi ønsket å eksperimentere med prediksjon av skade, som er et av de overordnede målene til Forzasys presentert i seksjonene [1.6](#) og [1.7](#). For å kunne predikere en skade må vi benytte oss av en såkalt multivariate metode for tidsserieprediksjon, som vi ga en kort introduksjon til i seksjon [4.1.2](#). Til forskjell fra de tidligere testene hvor vi benyttet univariate testing, måtte vi nå gjøre prediksjonene ved å gjøre kalkulasjoner basert på andre parametere enn kun Readiness-verdien. Vi kan anta at

skader lettere kan oppstå om en spiller har mangel på søvn, er støl i kroppen og opplever mye stress, i tillegg til at andre faktorer kan påvirke en spiller opplevde «Readiness to Play». Vi har forsøkt å implementere en veldig enkel metode for å kunne eksperimentere med prediksjon av skade. Det har ikke vært mye overskudd av tid til dette eksperimentet, da mesteparten av arbeidet har blitt fokusert på prediksjon av Readiness-parameteren i seksjonene [6.4](#), [6.5](#) og [6.6](#).

Vi startet med å lage et datasett som inneholdt de spillerne vi fant som hadde registrert en skade. I dette tilfellet fokuserer vi mest på Spiller3, men Spiller2 hadde også en registrert skade 20.03.2020. Dette var en mindre skade som viste seg å ikke påvirke de andre parametrene i datasettet, men vi valgte likevel å ta det med i denne testingen. Skaden påført hos Spiller3 var derimot av en høyere alvorlighetsgrad og, som beskrevet tidligere, påvirket Readiness-variabelen i stor grad. Etter databehandlingen, bestående av de fleste stegene presentert i kapittel [4](#), laget vi en funksjon som henter ut datoan skaden har skjedd hos den aktuelle spilleren. Når denne ble hentet ut ble det opprettet en ny parameter som vi kalte Injury, og registrert med verdien 0 eller 1. Dette fungerte som en boolsk representasjon, hvor 0 betyr at spilleren ikke er skadet og 1 betyr at den er skadet. I Figur 46 er vi hvordan en output av dette datasettet ser ut.

	Date	Mood	Fatigue	Readiness	SleepDurH	SleepQuality	Sorenness	Stress	Injury
0	2020-03-20	3.0	3.0	6.0	8.5	3.0	2.0	2.0	0
1	2020-03-21	4.0	3.0	6.0	10.0	3.0	2.0	3.0	0
2	2020-03-22	4.0	2.0	3.0	9.0	4.0	2.0	3.0	1
3	2020-03-23	3.0	3.0	7.0	8.5	3.0	3.0	3.0	1
4	2020-03-24	3.0	3.0	5.0	10.0	3.0	3.0	3.0	0
...	...	...	...	...	...	...	...	...	...
647	2021-12-27	3.0	3.0	3.0	9.0	4.0	2.0	4.0	0
648	2021-12-28	3.0	2.0	3.0	8.0	3.0	2.0	3.0	0

Figur 46: Viser det ene datasettet til Spiller3, hvor det har blitt lagt til en ny parameter kalt Injury som forteller oss om det eksisterer en registrert skade.

Etter at vi har opprettet Injury-parameteren var neste steg å hente ut totalt 10 dager med informasjon, et minimumskrav for å gjennomføre testingen.<sup>53</sup> Likevel er det slik at et større sett med rådata å trenne på vil legge et bedre grunnlag for prestasjonsevnen til

<sup>53</sup>

[statisticssolutions.com/dissertation-resources/sample-size-calculation-and-sample-size-justification/sample-size-formula/](http://statisticssolutions.com/dissertation-resources/sample-size-calculation-and-sample-size-justification/sample-size-formula/)

modellen. Vi valgte bevisst en tidsperiode hvor en skade hadde inntruffet, og på grunn av dette lave Readiness-verdier. Tidsperioden ble satt til 5 dager før skaden oppstod hos Spiller3, og 4 dager i etterkant. Ettersom Spiller2 sin skade skjedde på den første dagen i datasettet, 20.03.2020, valgte vi her å ta med de 9 påfølgende dagene.

### Spiller3

	Mood	Fatigue	Readiness	SleepDurH	SleepQuality	Soreness	Stress	Injury
365	4.0	2.0	3.0	7.0	3.0	3.0	3.0	0
366	4.0	3.0	3.0	10.0	4.0	3.0	3.0	0
367	3.0	3.0	3.0	6.0	3.0	3.0	3.0	0
368	4.0	3.0	1.0	6.0	3.0	3.0	2.0	0
369	4.0	3.0	1.0	9.0	4.0	3.0	3.0	0
370	3.0	3.0	1.0	10.0	4.0	3.0	3.0	1
371	3.0	3.0	2.0	9.0	3.0	3.0	3.0	0
372	3.0	3.0	2.0	9.0	3.0	3.0	3.0	0
373	3.0	3.0	2.0	7.5	3.0	3.0	2.0	0
374	3.0	2.0	2.0	5.5	3.0	3.0	2.0	0
375	3.0	2.0	2.0	7.0	3.0	3.0	2.0	0

### Spiller2

	Fatigue	Mood	Readiness	Sleep Durh	Sleep Quality	Soreness	Stress	Injury
0	3.0	2.0	6.0	9.0	3.0	2.0	3.0	1
1	3.0	3.0	6.0	10.0	4.0	2.0	3.0	0
2	2.0	3.0	1.0	10.0	3.0	2.0	3.0	0
3	3.0	3.0	6.0	9.0	4.0	2.0	3.0	0
4	2.0	2.0	6.0	9.0	4.0	2.0	3.0	0
5	2.0	3.0	5.0	9.0	2.0	2.0	3.0	0
6	3.0	4.0	5.0	9.0	4.0	2.0	4.0	0
7	4.0	3.0	6.0	8.5	3.0	2.0	4.0	0
8	3.0	3.0	7.0	8.5	4.0	2.0	3.0	0
9	2.0	3.0	5.0	10.0	4.0	2.0	4.0	0
10	3.0	3.0	6.0	8.5	4.0	2.0	4.0	0

Figur 47: Fremstilling av nye datasett for Spiller3 og Spiller2, med opprettet kolonne for Injury-parameter.

Hos Spiller3 ser vi 5 dager før skade og 4 dager etter, hos Spiller2 ser vi dagen skaden oppstod og de 9 påfølgende.

Det neste steget var å gjøre tester med radene presentert i datasettet til Spiller2 og Spiller3 presentert i Figur 47. Disse skulle transformeres til et numpy array for å gjøre det litt enklere for oss. Deretter deles arrayet opp i X og y, hvor vi setter at X representerer alle parametrene for generell velvære, og y representerer Injury. Resultatet av dette er fremstilt i Figur 48 under.

X	y
array([[ 4. ,  2. ,  3. ,  7. ,  3. ,  3. ,  3. , [ 4. ,  3. ,  3. , 10. ,  4. ,  3. ,  3. , [ 3. ,  3. ,  3. ,  6. ,  3. ,  3. ,  3. , [ 4. ,  3. ,  1. ,  6. ,  3. ,  3. ,  2. , [ 4. ,  3. ,  1. ,  9. ,  4. ,  3. ,  3. , [ 3. ,  3. ,  1. , 10. ,  4. ,  3. ,  3. , [ 3. ,  3. ,  2. ,  9. ,  3. ,  3. ,  3. , [ 3. ,  3. ,  2. ,  9. ,  3. ,  3. ,  3. , [ 3. ,  3. ,  2. ,  7.5,  3. ,  3. ,  2. , [ 3. ,  2. ,  2. ,  5.5,  3. ,  3. ,  2. ]])	[ 0. ], [ 0. ], [ 0. ], [ 0. ], [ 0. ], [ 1. ], [ 0. ], [ 0. ], [ 0. ], [ 0. ]])

Figur 48: Viser hvordan det ferdigstilte numpy arrayet for datasettet til Spiller3 ser ut. X representerer parametrene for generell velvære, og y representerer den nyopprettede Injury-parameteren.

Vi bruker deretter en funksjon kalt LogisticRegression() fra Scikit-learn, presentert i seksjon [2.5.3](#), til å bygge en prediksjonsmodell. Her forsøker vi å se hvor nøyaktig denne funksjonen er med tanke på dataene i arrayet, og input verdiene vi tester på er X og y.

Denne testen returnerer en verdi for  $R^2$ , se seksjon [6.2](#) for forklaring, som hos Spiller3 er 1 og 0.8 hos Spiller2. I denne testen, som er fra et såpass lite sett med datapunkter, har ikke denne testen stor betydning. Neste steg er å forutsi sannsynligheten for skade med ny input av data. På grunn av at vi jobber med en multivariate metode, falt valget på Logistic Regression (LR). Dette er en algoritme som er enkel å implementere, og kan brukes som en baseline for binære problemstillinger innenfor classification.

Vi lager et eksempelarray med verdier: [2., 3., 1., 9., 3., 2., 3.] til Spiller3. Her representerer hver verdi de ulike velvære-parametrene, og kommer i følgende rekkefølge: Mood, Fatigue, Readiness, Sleep Duration, Sleep Quality, Soreness og Stress. Som et eksempel array for Spiller2 har vi følgende array: [1., 3., 9., 5., 2., 3., 2.]. Disse arrayene blir brukt som input i LR-modellen, og deretter testet for å forsøke å predikere om denne sammensetningen av verdier kan føre til en skade. Vi kaller funksjonen logisticRegression.predict(), og kjører arrayet til Spiller3 gjennom først. Resultatet vi får ut er «1», som tilsier skade. Deretter kjører vi arrayet til Spiller2 gjennom, og får «0» som resultat; ikke skade.

Siste steg er å forsøke å gjøre en prediksjon med funksjonen kalt predict\_proba(X). Denne returnerer en sannsynlighetsestimat for alle klasser. Vi ber også modellen om å vise hvor stor risiko spilleren har for skadegjennom en returnert prosentverdi. Resultatet av dette ser vi presentert i Figur 49 under.

```

[[0.33553656 0.66446344]]
PPlayer3 risk of injury is:66%
-----
[[0.94886319 0.05113681]]
PPlayer2 risk of injury is:5%

```

*Figur 49: Viser resultatet av forsøk på prediksjon av skade hos Spiller2 og Spiller3. Risikovurderingen gis gjennom en estimert prosentverdi, hvor høy prosent indikerer større risiko for skade.*

Basert på resultatet presentert i Figur 49 ser vi at Spiller3 har risiko på 66% for at en skade vil inntreffe. Dette er basert på om de loggførte verdiene i det konstruerte arrayet vi viste til tidligere holder seg på samme nivå. Hos Spiller2 derimot, ser vi en vesentlig lavere risiko basert på eksemplarrayet, med kun 5% risiko.

Det ble ikke gjort mye eksperimentering med LR og prediksjon av skade, men ut fra testene gjort med eksemplarrayene kan man se at det er et potensiale for det. Vi fikk ikke tid til å gå i dybden av å bruke disse modellene på PMSys datasettet, og undersøke om det hadde gått å predikere de skadene som inntraff de ulike spillerne. Vi fikk heller ikke gjort gode vurderinger av test score fra MSE eller  $R^2$  for å se om resultatene vi fikk kunne oppfattes som gode prediksjoner.

For å kunne teste med modellen kan vi lage eksperimenter med å mate modellen med mye data som mulig slik modellen kan trenere på de verdiene. Logistic Regression trenger minst 10 prøver per variabel for å kunne gjøre prediksjoner. Likevel kan regnes ut hva som er den optimale antall input ved å beregne en effektstørrelse til prøvestørrelsen. Effektstørrelse er kjent som forskjellen mellom prøvestatistikken delt på standard error.<sup>54</sup> Ved å gjøre de stegene kan vi teste og forske mer på denne modellen.

Dette eksperimentet ble gjort for å se om vi kunne oppnå lovende resultat når det gjaldt forsøk på prediksjon av skade, og hvilke alternativer som har potensiale for videre testing. Den lille mengden med testing som ble gjennomført gir ikke et godt nok grunnlag til å gi noen konkrete konklusjoner angående prestasjonsevnen til modellen. Vi kan derimot se at det ligger et potensiale for videre eksperimentering med fremgangsmåten som ble presentert i denne seksjonen.

---

<sup>54</sup>

[statisticssolutions.com/dissertation-resources/sample-size-calculation-and-sample-size-justification/sample-size-formula/](http://statisticssolutions.com/dissertation-resources/sample-size-calculation-and-sample-size-justification/sample-size-formula/)

## 6.10 Konklusjon

Basert på testingen vi har gjort, og en helhetlig vurdering av visuelle og numeriske resultater, er gruppen enige om at LSTM skiller seg ut fra RF og ARIMA som modellen med best potensiale for nøyaktig prediksjon av Readiness. Ser man på videre potensiale er det tydelig at LSTM vil kunne forbedre prediksjonsevnen parallelt med en økning av verdier i datasettene til spillerne. Denne modellen legger derfor et godt grunnlag for videre arbeid og forsøk.

## 7 Diskusjon og refleksjoner

Utgangspunktet for dette bachelorprosjektet var en åpen problemstilling som kunne gå i svært mange retninger og gruppen hadde mye frihet når det gjaldt å komme frem til og konkretisere denne. I rapporten har vi gjort rede for hvordan vi har planlagt og gjennomført arbeidsprosessen, og presentert vår endelige konklusjon som et resultat. Etter gjennomført arbeidsperiode sitter vi igjen med resultater vi mener kan begrunnes ut i fra testingen som har blitt gjort, i tillegg til å ha lagt et grunnlag for videre arbeid som kan baseres på våre resultater.

I dette kapittelet vil vi oppsummere prosjektet ved å diskutere fremgangsmåten, funnene, og kunnskapen vi har tilegnet oss. Videre vil vi drøfte hvordan valgene vi har tatt i starten av prosjektet har påvirket beslutningene gjort i etterkant, og se nærmere på konsekvensene av disse.

### 7.1 Fremgangsmåte

Basert på utgangspunktet for vårt prosjekt, var det flere beslutninger vi måtte ta angående planleggingen av arbeidet. Det første steget var å definere en problemstilling, som vi visste ville gå inn på bruk av maskinlæring og prediksjon. Vi kom til slutt frem en problemstilling som baserte seg på prediksjon av Readiness to Play. På grunn av størrelsen på gruppen, og en vurdering av prosjektets totale tid, var alle enige om at vi ville velge å teste flere mer enn kun én maskinlæringsmodell for arbeide med. To av modellene valgte vi ut i fra grunnlaget av tidligere arbeid, altså LSTM og Random Forest (RF). Disse ble kombinert med ARIMA, en modell som skulle være spesielt god på prediksjon av tidsserier. Gjennom arbeidet med disse modellene ville vi se på forskjellene og likhetene mellom dem, samt undersøke hvilken av dem som viste best potensiale for videre arbeid. Selv om noen modeller kunne gi et mindre lovende resultat for prediksjon av tidsseriedata, ble det vurdert som hensiktsmessig å gjenta testingen for å kunne bidra til å verifisere tidligere resultat.

Vi valgte datasettet til de fire spillerne med færrest antall nullverdier for å sørge for at håndteringen av de manglende verdiene ikke ville påvirke resultatene til modellene i stor grad. Det er gunstig å ha data som i inneholder mest mulig faktiske loggførte verdier, slik at man får mer virkelighetsnære prediksionsverdier. De tydelige ulikhettene i tidsserien til hver enkelt spiller ville også være med på å teste modellenes håndteringsevne av nettopp disse forskjellene.

## 7.2 Alternative utgangspunkt

Ved ferdigstillelse av prosjektet satt gruppen igjen med tanker om konsekvensene av beslutninger som ble tatt i startfasen av arbeidet. Dette dreide seg spesielt angående valget av maskinlæringsmodeller. Resultatet av testingen gir nok grunnlag til å trekke gode konklusjoner og skape et godt utgangspunkt for videre arbeid, men vi er enige om at det finnes rom for forbedring. Vi har drøftet rundt enkelte alternative utgangspunkt, og om dette kunne ha forbedret arbeidsprosessen og resultatene vi kom frem til.

Etter at de tre modellene var valgt, brukte vi en del tid på å opparbeide oss en forståelse av teorien og koden som var nødvendig for implementasjon av disse. Det er ikke snakk om store mengder kode, men å forstå de ulike kombinasjonene av input parametere, og hvordan man bygger opp en god modell krevde en god del utprøving. Enkelte kombinasjoner av inputparametere resulterte i lange kjøre- og kalkulasjonstider for modellene. Med tanke på tiden vi hadde til rådighet for fullføringen av prosjektet, er det i ettertid tydelig at tre modeller var litt i overkant. Det gjorde det utfordrende å holde en gjennomgående god oversikt. Det hadde vært mer hensiktsmessig å fordype seg godt i enten én, maks to, av de valgte maskinlærings-modellene, og heller bruke oppstartstiden til å undersøke teorien bak hver modell.

Som nevnt i seksjon [1.7](#) ville vi at vårt arbeid skulle fungere som et tillegg til tidligere resultater gjort med de samme modellene, slik at resultatene kan verifiseres og sammenlignes med andre nye alternativer. I seksjon [6.10](#) konkluderte vi med at LSTM er det beste alternativet, sammenlignet med resultatene fra RF i seksjon [6.4](#) og ARIMA i seksjon [6.5](#). Basert på dette resultatet, og den nåværende kunnskapen vi har om funksjonalitet for hver av de tre modellene, kan det ha vært fordelaktig å fokusere mer på modeller innenfor Deep Learning (DL). I seksjon [3.2.2](#) forklarte vi litt av teorien og algoritmer som finnes i DL, hvor LSTM presenteres som en form for RNN. Derfor kunne et mulig alternativ ha vært testing av ulike typer algoritmer basert på Neural Network (NN), slik som CNN og RNN beskrevet i seksjon [3.2.2](#). I seksjon [5.3.3](#) refererer vi også til litteratur som viser at CNN i enkelte tilfeller har gitt bedre resultater, og er enklere å kjøre enn LSTM, og i seksjon [6.8](#) prøvde vi ut en enkel CNN-modell. Denne viste tegn til å kunne gi en god prediksjon.

Utgangspunktet for problemstillingen var å predikere Readiness to Play variabelen, forklart i seksjon [1.6](#). Vi valgte å legge fokuset på prediksjoner som var tilnærmet like de faktiske loggførte verdiene, altså verdier i spennet 1-10. I stedet for dette kunne vi ha predikert «topper» og «bunner», definert av spesifikke verdier på forhånd, fremfor å

predikere nøyaktige verdier. Dette var måten Wiik et al. også gjennomførte eksperimentene, ved å definere toppene med verdier fra 8 og over, og bunnene med verdier fra 3 og nedover. På denne måten kan man se for seg at man heller stiller modellen et spørsmål; er denne spilleren klar for kamp? Hvor den predikerte toppen eller bunnen resulterer i et sant/usant eller ja/nei svar. En slik fremgangsmåte for prediksjon kunne uten vanskeligheter ha blitt implementert i våre modeller og vært en del av testingen.

## 7.3 Fremtidige mål

For vår oppdragsgiver, Forzasys, var det i første omgang ønskelig å teste alternativer innenfor maskinlæring som gjorde prediksjoner for Readiness to Play, som et første steg i prediksjon av andre velværeparametere, og potensielt av skade. I seksjon [1.5](#) forklarer vi hvordan pmSys fungerer i dagens bruk, og potensialet appen har for videre utvikling ved hjelp av dataen som blir samlet inn og registrert av spillerne. I det store bildet kan denne utviklingen bidra til å løse en overordnet problemstilling som omhandler mulig prediksjon av om en spiller nærmer seg en skade, presentert i seksjon [1.6](#). Å kunne modellere en slik type prediksjon vil kreve at man tar i bruk alle parametrene som blir loggført, både når det gjelder generell helse og Ratings of Perceived Exertion (RPE). Dette betyr at man må bruke en såkalt multivariate modell, hvor det er et flertall av parametere som tas med i kalkulasjonene. Vi gjorde forsøk med dette, presentert i seksjon [6.9](#), hvor vi så på en eventuell fremgangsmåte for å forsøke å predikere en skadet spiller.

Testingen av univariate modeller vil bidra til å finne alternativer man videre kan bygge til å bli multivariate. Basert på vurderinger gjort av våre egne tester er videreutviklingen til en slik modell et viktig steg i målet om å predikere skader. Likevel vil loggføringen av Readiness to Play parameteren være en verdi som i stor grad er påvirket av de resterende parametrene, noe som kan observeres i korrelasjonsmatrisen Johansen et al. presenterer i sin rapport. Her ser vi at det blant annet er en sammenheng mellom Readiness-verdien og informasjon fra Sleep Quality, Fatigue, Soreness og Perceived Exertion. Dette gjør at man kan fortsette med univariate testing av Readiness-variabelen, fordi denne verdien er et resultat av informasjon loggført innenfor de andre parametrene.

## 7.4 Arbeidsmetodikk

Vi strukturerte arbeidet vårt med Scrum som arbeidsmetodikk, og dette ble presentert i seksjon [2.2](#). Vi hadde totalt 6 sprinter, med en lengde på 2 uker hver. I forkant av hver

sprint ble det lagt en plan for hva som skulle gjennomføres under perioden, og på slutten av hver sprint ble det gjennomført en sprint review. Her gikk vi gjennom perioden som hadde gått, gjorde en overordnet vurdering over hva som fungerte og ikke, for så å legge en plan for neste sprint. For oss fungerte det godt å ha individuelle fokusområder i tillegg til å sette konkrete oppgaver som skulle gjennomføres under hver sprint. Dette gjorde det mulig å utføre samme eksperiment i opptil flere sprinter, men med utprøving av ulike detaljer og aspekt enn i tidligere testing.

I starten av prosessen var det utfordrende å vite hvor mye arbeid vi skulle velge å fokusere på under hvert delmål og sprint, og dette ble noe vi måtte prøve oss litt frem med. Siden vi i tillegg er fire gruppemedlemmer som kan arbeidet kan fordeles på, måtte vi gjøre en vurdering på hvordan dette skulle utnyttes på en fordelaktig måte. Vi opplevde at det i begynnelsen ble planlagt for mye enn det vi rakk å komme gjennom til den planlagt fristen. Dette ble raskt vurdert underveis og tilpasset til den neste sprinten, noe som understreket fordelen ved å ha valgt en smidig arbeidsmetodikk til prosjektarbeidet. Gruppen har i ettertid reflektert rundt at fokuset burde ha blitt fordelt på et mindre antall modeller, som nevnt i seksjon [7.2](#), noe som kunne ha bidratt til å fordele enkelte delmål ut over flere sprinter.

Etter en totalvurdering av planlagt arbeidsmetodikk i forhold til prosjektets utgangspunkt, er gruppen fornøyde med måten vi håndterte prosessen på, spesielt siden ingen i gruppen har jobbet med en utredningsrapport på denne måten.

## 7.5 Læringsutbytte

Etter å ha fullført arbeidsprosessen med prosjektet som er gjort rede for i denne sluttrapporten, er alle i gruppen enige om at vi sitter igjen med et faglig og teknisk læringsutbytte innenfor flere aspekt ved det gjennomførte arbeidet. Det var ingen som hadde spesielt mye erfaring innenfor maskinlæring på forhånd, men som beskrevet i seksjon [2.3](#) hadde alle i gruppen hadde hatt et introduksjonsfag innenfor kunstig intelligens under høstsemesteret i 2021. Det har vært krevende, både når det gjelder tid og faglig kunnskap, å sette seg inn teorien som ble presentert i kapitlene [3](#), [4](#) og [5](#), og videre forstå hvordan vi skulle gå frem for å teste de ulike modellene beskrevet i kapittel [6](#). Gjennom kontinuerlige forsøk, prøving og feiling, har vi oppdaget at det er nå i sluttfasen vi virkelig forstår helheten av maskinlæringsarbeid, spesielt prediksjon av tidsserier.

Vi var inneforstått med at en god del av arbeidet ville gå med på bearbeiding av tildelte datasett, og vi fikk erfare at hver beslutning som ble tatt måtte grundig begrunnes før vi kunne gå videre. I seksjonene [4.1.2](#) og [4.1.3](#) går vi gjennom enkelte av disse valgene,

som blant annet angikk om vi skulle trenere valgte maskinlæringsalgoritmer på kun én spiller eller hele laget, og hvordan vi ville velge å fylle inn eksisterende nullverdier. Alle disse faktorene spiller inn på de endelige ferdigstilte datasettene, hvor små justeringer i disse beslutningene tatt på forhånd kan påvirke den endelige prediksjonen fra en maskinlæringsmodell.

Det har vært spesielt interessant å se hvor forskjellige, men samtidig like, de tre modellene vi har testet i seksjonene [6.4](#), [6.5](#) og [6.6](#) er. Selv om modellene innenfor regression har som hensikt å predikere numeriske verdier, er det tydelig at de ikke «lærer» på samme måte som LSTM, og andre Deep Learning-modeller. Balansen mellom analyse av resultat fra numeriske estimat, slik som måling av feilestimat fra seksjon [6.2](#), og det faktum at vi forsøker å predikere menneskelig atferd har vært viktig å basere konklusjoner på. Denne måten å vektlegge de ulike aspektene på gjør det tydeligere hvilke modeller som presterer bedre med tanke på problemstillingen

Vi har nå en større forståelse for hvor mye tid som burde avses til hvert steg i arbeidsprosessen. Hadde vi gått i gang med et nytt prosjekt nå hadde det vært enklere å konkretisere innholdet i hver enkelt sprint, som videre kunne ha påvirket mengden arbeid som ble gjort.

## 8 Konklusjon

I beskrivelsen av vår problemstilling i seksjon [1.6](#) understrekte vi hva vi ønsket å utforske i denne rapporten, og hva vi ønsket å ende opp med som resultat. Vi skulle gjøre forsøk med prediksjon av parameteren Readiness to Play, et arbeid som baserte seg på innsamlet informasjon fra 4 utvalgte spillere. I denne rapporten har vi ved hjelp av AI-analyse av tidsseriedata fra atleter eksperimentert med mulige fremgangsmåter for prediksjon av én enkelt loggført parameter; Readiness to Play. Dette er en overordnet parameter, som forteller oss om en fotballspillers opplevde dagsform i forkant av kamp og trening. Ved å ta utgangspunkt i maskinlæringsalgoritmene Random Forest (RF), Autoregressive Integrated Moving Average (ARIMA) og Long Short-Term Memory (LSTM) har vi forsøkt å fremstille optimale prediksjoner av Readiness to Play parameteren. Vi har sammenlignet resultatene, og kan konkludere med at hver modell har sine styrker og svakheter.

For oss var det hensiktsmessig å ta utgangspunkt i flere datasett for å sikre oss om at vi fikk testet ulike sammensetninger av registrerte verdier, men holde oss innenfor en realistisk mengde vi hadde tid til å teste. Dette valget gjorde at vi tydeligere kunne se hvordan de tre ulike modellene håndterte datasett med ulikt innhold. En viktig observasjon vi gjorde var at både RF og ARIMA ikke klarte å håndtere store spenn i datasettene som ble testet. I tillegg var resultatene ofte basert på en predikert gjennomsnittsverdi, og var generelt svake i prestasjonsevnen når det gjaldt å forstå mønstre i input dataen. Mot slutten av arbeidsprosessen oppdaget vi derimot en ny måte å gjennomføre prediksionsarbeid på i RF-modellen, hvor vi så en forbedring når det gjaldt en tydeligere sammenheng mellom predikerte toppler og bunner mot det som faktisk var loggført.

I eksperimentene av alle tre modellene ble det gjort sammenligninger med terskelverdien fra en dummy modell, og ble brukt til å måle prestasjonsevnen til modellene. Vi kan tydeliggjøre at en godt estimert test error ikke er nok for å konkludere med at en modell gjør et bra prediksionsarbeid. Problemstillingen, og det overordnede målet for oppdragsgiver, er svært komplekst og krever en løsning som ikke kun viser en lav test error. En optimal løsning skal predikere menneskelig atferd, noe som stiller høyere krav til den helhetlige prestasjonen. Dette gjorde at det visuelle resultatet av prediksionsgrafen ble vektlagt i stor grad når vi skulle gjøre de endelige konklusjonene for resultatet til hver enkelt modell.

Basert på dette kunne vi konkludere med at LSTM viser det mest lovende resultatet gjennom alle eksperimentene vi har gjort. Denne modellen opererer med et eget minne, hvor den kontinuerlig prosesserer ny informasjon og samtidig tar hensyn til tidligere gjennomførte prediksjoner. Med et grunnlag i Artificial Neural Network (ANN), som er bygd opp på en måte

som skal etterligne strukturen i en hjerne, får vi en modell som forsøker å tilnærme seg menneskelig oppførsel i større grad enn RF og ARIMA. Etter å ha gjort tester med datasett av ulike størrelser kunne vi også se at LSTM gir inntrykk av å gi bedre prediksjoner med større datasett. Dette gjør at det er mye potensiale i videre testing av LSTM, spesielt med tanke på at den innsamlede dataen av pmSys kommer til å øke.

På grunn av manuelle observasjoner i datasettet til spillerne oppdaget vi at den tydelige endringen i de registrerte verdiene til Spiller3 var på grunn av en skade. Dette gjorde at vi forsøkte å finne en fremgangsmåte som kunne fungere for prediksjon av skade, som er det overordnede målet til oppdragsgiver. Her ble det ikke gjort nok testing til å kunne trekke noen endelige konklusjoner til å kunne gi noen endelige anbefalinger, men det viser likevel et potensiale for videre eksperimentering.

Vi mener at vi gjennomgående i rapporten har fremstilt et arbeid som kan brukes som et grunnlag inn i fremtidig arbeid. Ettersom gruppen hadde lite erfaring med maskinlæring og prediksjonsarbeid i forkant, har vi likevel klart å belyse viktige aspekt som kan tas med videre. I drøfting vi har gjort i etterkant er vi alle enige om at det er andre utgangspunkt som potensielt kunne ha gitt oss bedre resultat. Her tenker vi hovedsakelig på valg av modeller, og om fokuset heller burde ha blitt lagt på algoritmer innenfor kun Deep Learning og Artificial Neural Networks. Dette kommer også frem når vi ser på det begrensede eksperimentet som ble gjort med Convolutional Neural Network mot slutten av arbeidsprosessen.

Utfordringene vi har støtt på underveis, innenfor både planlegging av arbeidsprosess og selve testingen som har blitt gjennomført, har bidratt til diskusjoner innad i gruppen som har vært givende og interessante. Gjennom prosjektarbeidet har vi lært veldig mye, både med tanke på faglig utbytte og selve arbeidsprosessen som en gruppe. Med tanke på utgangspunktet vi hadde i starten har vi med andre ord utvidet den faglige horisonten vår betraktelig, og vi sitter igjen med kunnskap som vi vil få god nytte av i videre studier og i arbeidslivet. Selv om alternative utgangspunkt og fremgangsmåter kunne ha vært mer optimale måter å gjennomføre prosjektet på, sitter vi igjen med et resultat vi kan stå inne for og er fornøyde med.

## 9. Referanser

Eliakim, E., Morgulev, E., Lidor, R., & Meckel, Y. (2020). Estimation of injury costs: financial damage of English Premier League teams' underachievement due to injuries. *BMJ Open Sport & Exercise Medicine*, 6 (1), e000675. <https://doi.org/10.1136/bmjsem-2019-000675>

Wiik, T., Johansen, H. D., Pettersen, S. A., Baptista, I., Kupka, T., Johansen, D., Riegler, M., & Halvorsen, P. (2019). Predicting Peak Readiness-to-Train of Soccer Players Using Long Short-Term Memory Recurrent Neural Networks. *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*. Published. <https://doi.org/10.1109/cbmi.2019.8877406>

Johansen, H. D., Johansen, D., Kupka, T., Riegler, M. A., & Halvorsen, P. (2020, October). *Scalable Infrastructure for Efficient Real-Time Sports Analytics*. <Https://Home.Simula.No/~paalh/Publications/>. <http://home.simula.no/~paalh/publications/files/MAIStrOPE-2020-pmsys.pdf>

Kulakou, S. (2021). *Exploration of time-series models on time series data: A case study on athlete monitoring* [Master's Thesis, University of Oslo]. Simula. <https://home.simula.no/~paalh/students/SiarheiKulakou.pdf>

Belyadi, H., & Haghigat, A. (2021). Introduction to machine learning and Python. *Machine Learning Guide for Oil and Gas Using Python*, 1–55. <https://doi.org/10.1016/b978-0-12-821929-4.00006-8>

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3), 685–695. <https://doi.org/10.1007/s12525-021-00475-2>

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* [E-book]. MIT Press. <http://www.deeplearningbook.org>

Edgar, T. W., & Manz, D. O. (2017). Machine Learning. *Research Methods for Cyber Security*, 153–173. <https://doi.org/10.1016/b978-0-12-805349-2.00006-6>

Kotu, V., & Deshpande, B. (2019). Data Science Process. *Data Science*, 13–37.

<https://doi.org/10.1016/b978-0-12-814761-0.00002-2>

Mills, T. C. (2019). *Applied Time Series Analysis*. Elsevier Gezondheidszorg.

<https://doi.org/10.1016/C2016-0-03956-6>

Kotu, V., & Deshpande, B. (2019). Time Series Forecasting. *Data Science*, 395–445.

<https://doi.org/10.1016/b978-0-12-814761-0.00012-5>

Elgaaen, F. B. & Larssen, N. M. (19.05.2017). Data mining i banksektoren - Prediksjonsmodellering og analyse av kunder som sier opp boliglån. [ Masteroppgave, Universitetet i Oslo og Høgskulen på Vestlandet]

[https://www.duo.uio.no/bitstream/handle/10852/57507/1/master\\_nml.pdf](https://www.duo.uio.no/bitstream/handle/10852/57507/1/master_nml.pdf)

Woloszko, N. (2018). Adaptive Trees: A novel approach to macroeconomic forecasting. Organisation for Economic Co-operation and Development (OECD), 12-14.

[https://www.unirc.it/documentazione/materiale\\_didattico/1465\\_2017\\_437\\_29329.pdf](https://www.unirc.it/documentazione/materiale_didattico/1465_2017_437_29329.pdf)

Biau O., & D'Elia, A. (2010) Euro area GDP forecasting using large survey datasets. A random forest approach.

[https://ec.europa.eu/economy\\_finance/db\\_indicators/surveys/documents/workshops/2010/ec\\_meeting/biau\\_delia\\_gdp\\_forecasting.pdf](https://ec.europa.eu/economy_finance/db_indicators/surveys/documents/workshops/2010/ec_meeting/biau_delia_gdp_forecasting.pdf)

Bankson, C. A., & Holm, A. M (2019). Kunstig intelligens i makroøkonomisk prognosearbeid. En empirisk studie av hvor godt maskinlæring evner å predikere norsk økonomisk vekst [Bacheloroppgave, Norges Handelshøyskole].

<https://openaccess.nhh.no/nhh-xmlui/bitstream/handle/11250/2644325/masterthesis.pdf?sequence=1&isAllowed=y>

Granger, C., & Newbold, P. (1974). Spurious regressions in econometrics. *Journal of Econometrics*, 2(2), 111–120. [https://doi.org/10.1016/0304-4076\(74\)90034-7](https://doi.org/10.1016/0304-4076(74)90034-7)

Li, Y., Wu, H., & Liu, H. (2018). Multi-step wind speed forecasting using EWT decomposition, LSTM principal computing, RELM subordinate computing and IEWT reconstruction. *Energy Conversion and Management*, 167, 203–219.

<https://doi.org/10.1016/j.enconman.2018.04.082>

Culurciello, E. (2019, January 10). *The fall of RNN / LSTM - Towards Data Science*. Medium. Retrieved May 25, 2022, from  
<https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0>

Al-jabery, K. K., Obafemi-Ajayi, T., Olbricht, G. R., & Wunsch II, D. C. (2020). Selected approaches to supervised learning. *Computational Learning Approaches to Data Analytics in Biomedical Applications*, 101–123. <https://doi.org/10.1016/b978-0-12-814482-4.00004-8>

Liu, H. (2021). Single-point wind forecasting methods based on deep learning. *Wind Forecasting in Railway Engineering*, 137–175. Retrieved 4. April 2022,  
<https://doi.org/10.1016/b978-0-12-823706-9.00004-1>

Weytjens, H., & de Weerdt, J. (2020). Process Outcome Prediction: CNN vs. LSTM (with Attention). *Business Process Management Workshops*, 321–333.  
[https://doi.org/10.1007/978-3-030-66498-5\\_24](https://doi.org/10.1007/978-3-030-66498-5_24)

Vallantin, L. (2020, June 15). Metrics to measure machine learning model performance. Medium. Retrieved May 5, 2022, from  
<https://medium.com/@limavallantin/metrics-to-measure-machine-learning-model-performanc>

Fuller, W. A. (1976). *INTRODUCTION TO STATISTICAL TIME SERIES. A Wiley Publication in Applied Statistics*. ISBN:0471287156. John Wiley.

## 10. Vedlegg

### 10.1 Teamkontrakt

## Bachelorgruppe 9

Helene Birkeflet Prescott (S341873)

Tonje Martine Lorgen Kirkholt (S341844)

Bernadette Fanni Finheim ( S341857)

Hanna Bækken Nilsen (S341879)

### Teamkontrakt

Følgende kontrakt er et resultat av samarbeid og enighet blant alle teammedlemmer, og gjelder for alle arbeidsprosesser i bacheloroppgaven.

#### Formål

Teamets hovedmål er å arbeide med og ferdigstille bachelorprosjekt for 2022. Vi har alle høye standarder og ønsker å jobbe mot en besvarelse av høy kvalitet. Det er ønskelig at gruppen sitter igjen med en følelse av å ha kommet til en gjennomgående forståelse av oppgaven, samt en besvarelse som gjenspeiler dette.

Det vil bli utarbeidet en egen fremdrifts- og arbeidsplan som følger med Forprosjektrapporten. Denne vil bli vedlagt kontrakten og skal følges som sterk ramme gjennom arbeidet med prosjektet. Vi ønsker å bruke hovedmål, delmål og små milepæler gjennom arbeidsprosessen - dette for å styrke effektiviteten og mestringsfølelsen underveis. Videre vil dette også redusere tidspress og stress generelt gjennom arbeidet.

Innlevering	Ferdigstilles	Deadline	Inndeling av innleveringer
<b>Forprosjekt</b>	20.01.2022	24.01.2022 kl.23:59	<ul style="list-style-type: none"> <li>- Avtale med oppdragsgiveren</li> <li>- Talskvinne/mann for prosjektgruppen</li> <li>- <b>Forprosjektrapport</b></li> <li>- Arbeidsplan og fremdriftsplan</li> </ul>
<b>Forprosjektrapport</b>	20.01.2022	24.01.2022 kl.23:59	<ul style="list-style-type: none"> <li>- Presentasjon</li> <li>- Sammendrag</li> <li>- Dagens situasjon</li> <li>- Mål og rammebetingelser</li> <li>- Løsninger /alternativer</li> <li>- Analyse av virkninger</li> </ul>
<b>Prosjekt- og sluttrapport</b>	24.05.2022	25.05.2022 kl. 12:00 Inspera	
<b>Prosjekt-presentasjon</b>		7.-14.06.2022 (tentativt)	

## **Roller og beslutninger innad i teamet.**

Vi har besluttet at teamet ikke skal ha en utvalgt prosjektleder, men at medlemmene har hver sine ansvarsområder og oppgaver. Vi kom også frem til en enighet at vi sammen skulle gjøre en felles innsats for å dele inn i fremtidige oppgaver når den tiden kommer.

## **Konflikthåndtering**

For å unngå konflikter har vi definert ulike problemer, og hvordan vi vil håndtere dem dersom de oppstår. Forøvrig er alle i teamet inneforstått med at *kommunikasjon* er det viktigste verktøyet til å løse opp utfordringer og konflikter. Vi har også alle et ansvar for å si fra om det er noe som oppleves som vanskelig.

Problem	Håndtering
Vi får problemer med å møte tidsfrister eller deadline.	For å forhindre at dette problemet oppstår vil vi ha fokus på planlegging. Vi avtaler hvem som gjør hva av arbeid fortløpende, og har alltid en tanke over hva vi mangler i oppgaven. Leveringsansvarlig passer også på at oppgaven blir ferdigstilt til fristen.
Bruke for mye tid på enkle eller "små" beslutninger som vi ikke blir enige om.	Sette av 15 minutter til å diskutere, og når den tiden er over skal beslutningen være tatt. Hvis ikke teamet er enig kan temaet enten legges vekk til et senere møte, eller så må beslutningstageren ta valget for gruppen.
Vi får vanskeligheter med å planlegge og holde møter. (Sykdom, jobb, reise, karantene eller nedstenging.)	Vi forsøker så godt vi kan å møtes fysisk, men om det ikke er mulig skal teamet være fleksible til å ta møtene digitalt. Hvis man ikke får møtt opp på møter pga jobb eller fritidsreiser er det hvert medlems ansvar for at de får gjort sin del av arbeidet utenom møtene. Man har da ansvar om å sende en oppdatering på teamets kanaler om hvordan arbeidet ligger an.
Vi får for knapp tid til å arbeide.	Vi sjekker jevnlig inn med gruppens medlemmer om hvor man ligger an med oppgavene sine. Vi passer på at vi er effektive med tiden når vi arbeider, og ikke tillater for mye overtid på møtene. Hvis vi bruker for mye tid på en problemstilling eller ikke blir enige, skal beslutningstager gripe inn på gruppens vegne.
Sosial loffing	Jevnlig oppdatering av status på individuelle oppgaver. Vi starter hvert møte med å fortelle hva vi har og ikke har fått gjort, og å fordele oppgaver. Denne teamkontrakten er også et verktøy vi tar i bruk for å unngå sosial loffing.
Vi mister fokus under møter eller arbeid.	Teamet tar 15 minutters pause eller tur for å koble av dersom vi mister fokus og ikke klarer å jobbe.

## Arbeidsmetoder

Under arbeidet med prosjektet er teamet enige om at vi i skal arbeide sammen fysisk i størst mulig grad (så lenge dette er gjennomførbart). Vi vil i hovedsak møtes på skolen og arbeide her, hvor vi vil booke passende grupperom. Booking av grupperom vil gjøres kontinuerlig så snart man har tilgjengelige bookingtidspunkt i TP (tp.uio.no).

## **Arbeid- og fremdriftsplan**

Videre planlegging av arbeidstidspunkt vil utdypes i eget dokument som lages i forbindelse med Forprosjektrapporten. Dette vil være et vedlegg som er tilgjengelig sammen med teamkontrakten.

## **Kommunikasjonskanaler**

Kommunikasjonskanalen vil hovedsakelig foregå via Facebook Messenger. Vi vil også ta i bruk Google Disks for oppbevaring av dokumenter, som gjør det lettere for alle å ha tilgang til alt til enhver tid.

Det er opprettet en egen chatgruppe på Messenger som skal tas i bruk når vi må kommunisere om bachelorprosjektet. Kanalen vil bli tatt i bruk når vi trenger å avtale møter, endre møter, og generelt spørsmål og kommunikasjon rundt prosjektet. Alle teammedlemmene er ansvarlige for å holde seg oppdatert på meldinger og informasjon som legges ut.

Utarbeidelsen av teamkontrakten er et resultat av enighet og samarbeid, og den skal følges i alle arbeidsprosesser gjennom våren i Bacheloroppgaven. Alle punkter er lest gjennom og forstått ved underskrivelse.

## **Sted og dato**

...OSLO...12.01.2022

## **Underskrifter:**

Bernadette Fanni Finheim



Tonje Martine Lorgen Kirkholt



Helene Birkeflet Prescott



A handwritten signature consisting of stylized initials "HB" above a dotted line.

Hanna Bækken Nilsen



A handwritten signature consisting of the name "Hanna BN" above a dotted line.