Norges miljø- og biovitenskapelige universitet

**Masteroppgave 2021 30 stp**
Fakultetet for realfag og teknologi

# Prediksjon av feil i vindturbiner på Smøla vindpark ved bruk av dyp læring

Wind turbine fault prediction using deep learning at Smøla wind farm

Sindre Tallaksrud
Miljøfysikk og fornybar energi

## Forord

Denne masteroppgaven fra våren 2021 er avslutningen på mastergraden min i *Miljøfysikk og fornybar energi* med fordypning i energifysikk ved fakultetet for realfag og teknologi ved Norges Miljø- og Biovitenskapelige Universitet (NMBU). Oppgaven omhandler predikering av feil ved vindturbiner på Smøla vindpark. Målet med oppgaven er å predikere om feil oppstår i fremtiden ved hjelp av maskinlæring. Om dette er mulig vil feilen kunne fikses før den oppstår, og dermed opprettholdes produksjonen i vindparken. Den personlige grunnen for den valgte oppgaven handler om interesse innen temaet samtidig som et av mine personlige mål er å forbedre min kunnskap og forståelse innen maskinlæring. Oppgaven omfatter 30 studiepoeng, og er utskriftvennlig (inneholder blanke sider som kan virke unødvendige ved digital lesning).

Det er mange å takke, min veileder Heidi Samuelsen Nygård har vært en fantastisk veileder som svarer på alle type henvendelser, samtidig som hun forklarer at dette er et arbeid som vi må gjøre på egen hånd. Dette har hjulpet mye på arbeidsmoral og fremdrift. Jeg må også takke Jørgen Olsen fra Statkraft som har gitt meg tilgang til datasettene som brukes i oppagaven, samtidig som han har vært behjelpelig med gode svar på spørsmål angående disse datasettene. Data er konfidensiell, og dermed er det ikke stor uttredelse av innholdet i oppgaven. Konkrete navn på parametere er blant annet ikke nevnt.

Tiden i Ås har vært helt fantastisk og jeg vil rette en stor takk til alle de jeg har blitt kjent med som jeg har knyttet livslange bånd til. Uten dem ville ikke denne masteroppgaven blitt levert. Familien min som jeg alltid kan snakke med fortjener også en stor takk. Sist, men ikke minst vil jeg vise min takknemlighet ovenfor den fantastiske snille og alltid støttende samboeren min som jeg har vært så heldig å dele hjemmekontor med det siste halve året. Tusen takk, alle sammen.

Ås, mai 2021
Sindre Tallaksrud

## Sammendrag

Klimaendringer er en av de største utfordringene vår generasjon står ovenfor. Å effektivisere produksjonen til fornybare energikilder er ett av mange små bidrag som kan gjøre samfunnet mindre avhengig av fossile energikilder. Denne oppgaven handler om effektivisering av vindturbiner på Smøla vindpark, hvor målet er å effektivisere vindturbinene som allerede er utplassert ved å predikere feil som oppstår.

Fokusområdet for å løse denne oppgaven er å se på tidspunktene hvor turbinene opplever at det er en feil i systemet, ved hjelp av data som er levert av Statkraft. Ved disse tidspunktene kan det oppstå tilfeller hvor turbinene må repareres og er, i en periode, ute av produksjon. Denne perioden er ønskelig å unngå for å opprettholde produksjonen til turbinene. For å unngå disse periodene vil det være ønskelig å predikere feilen før den oppstår. Dette gjøres ved hjelp av en maskinlæringsmodell som predikerer sannsynligheten for at en alarm inntreffer i fremtiden.

Maskinlæringsmodellen blir brukt flere ganger på forskjellige varianter av datasettene. Dette for å se om modellen blant annet predikerer bedre for alarmer som omhandler yaw-systemet på turbinene i forhold til alle typer alarmer. Forskjellen på hvor mange timer i forkant en alarm inntreffer er det også mulig å endre og sammenlikne. Det er ønskelig å vite så tidlig som mulig om en feil oppstår. Ved bedre tid vil det være enklere å reparere feilen.

De forskjellige resultatene for prediksjonen til modellen er relativt stabile. Resultatene blir presentert som en utregning som viser avviket (feilen i predikeringen) fra det modellen predikerer til det som er den korrekte verdien. Disse verdiene omhandler intervallet som er en valgt tid *før* alarmen oppstår, helt til alarmen oppstår. Her er resultatet for de forskjellige datasettene relativt like. Det gjennomsnittlige avviket er 0,329 for alle resultatene. Den maksimale verdien til avviket som er mulig er 1. Det betyr at det er en svært lav andel som predikerer korrekt som igjen gjør at det blir vanskelig å stole på om det modellen predikerer er riktig eller ikke. Dette er viktig hvis det skulle være ønskelig å implementere modellen i ulike systemer.

## Abstract

The climate change is one of our generations biggest challenges. By optimizing the use of renewable energy sources, the need of fossil fuels is less necessary. This is one of many small steps to a better future. In this master thesis the goal is to make the wind turbines at Smøla windfarm more efficient.

The main focus of this thesis, is analyzing alarms in wind turbines. This alarms happens when a fault occurs in the system. The source of the data used is Statkraft. At the timestamps where an alarm occur, the turbines need to be fixed. If it is possible to predict when an alarm occurs it will be easier to fix the problem in the future. Another advantage is that it will be easier to stabilize the production of power. This could optimize the production.

Different datasets will be implemented in the same machine learning model for comparison. This is to get an overview of which dataset has the best fit for the model. One of the differences in the datasets is the range of hours in probability before an alarm is activated. The earlier an alarm can be predicted, the more time is available to solve the problem. It will be an advantage if it is possible to predict an alarm happening very early. Then it will be more time to fix the upcoming problem. Another change in the datasets, is when the alarms are connected to the yaw-system, and not all alarms.

The results for the model in total are very much the same. In other situations, the results are presented as a calculation of the deviation between the correct values and the predicted values. There are different results for the model, but for some important datasets the mean deviation between the correct and predicted values is about 0.33. The maximum possible deviation is 1 so it is a model that has a low reliability. The low reliability makes it hard to implement this model in another tasks, where the data and the theory is about the same.

# Innhold

# Figurer

# Tabeller

# Forkortelser

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial neural network |
| LSTM | Long Short Term Memory |
| MAE | Mean Absolute Error |
| RNN | Recurrent neural network |
| SCADA | Supervisory Control And Data Acquisition |

# 1. Introduksjon

## 1.1 Motivasjon

Mengden energi som brukes i verden øker for hvert eneste år. I tillegg skal den energien som blir brukt være ren. Fornybare energikilder gjør det mulig å ha en økende bruk av energi samtidig som den brukte energien skal være bærekraftig og ha lave klimaavtrykk. Norge er en del av Parisavtalen som sier at temperaturen på kloden ikke må stige mer enn 2 grader før dette århundret er over [11].

Vindkraft er en omstridt, men viktig energikilde for Norge. Vannkraftverk derimot, er den største kilden til elektrisk energi, og står for 90% av kraftproduksjonen i dette landet. Vindkraftanleggene som er blitt installert i begynnelsen av 2021 vil ha en normalårsproduksjon på 13,1 TWh som er 8,5 % av årlig norsk kraftproduksjon [9]. Norge har et godt tilrettelagt klima og en geografisk form som er godt tilpasset vindkraft. Med lang kyst og lav befolkningstetthet er mulighetene mange. Det å utnytte de gode mulighetene vil ikke bare være til fordel for de nasjonale utslippene. Mulighetene til å bidra med å bremse de globale utslippene ved å eksportere fornybar energi er også til stede. Eksporten av denne energitypen er nå under 1 % av norsk krafteksport i følge Otto Søberg som er administrerende direktør i Eksportkreditt. Han mener at hvis eksporten av energi skal fortsette å være bærekraftig vil det være helt nødvendig å satse mer på fornybare energikilder [27].

Det grønne skiftet består av at mer elektrisk energi skal brukes fordi det er en energitype som har muligheter til å bli omdannet på miljøvennlige måter. Der hvor det tidligere har vært ønskelig å bruke en fossil energikilde har det i den siste tiden blitt mulig å erstatte den teknologien som krever fossilt brensel med teknologi som kun har behov for elektrisk energi. Dette betyr samtidig at det blir en større påkjenning for det elektriske kraftsystemet. Muligheten til å transportere og distribuere i større skala blir viktigere og viktigere. I tillegg er kostnadene rundt vindkraft en utfordring for å kunne konkurrere med fossile energikilder.

For å utkonkurrere de fossile energikildene i fremtiden vil vindkraft være en viktig del av dette grønne skiftet. Dermed vil overgangen til bruken av flere fornybare energikilder gjøre at utbygging og energiutvinning fra vinden vil øke i fremtiden. Norge har et godt utgangspunkt for å bygge ut store vindparker med jevn vind på grunn av den lange kystlinja. Dette er områder hvor det bor få mennesker som også gjør det til en fordel. Samtidig er det utfordringer med at det bor få mennesker i området ettersom det blir omdannet mer elektrisk energi enn nødvendig for lokalsamfunnet. Dette byr på utfordringer ved distribusjon av energien og mulige dimensjonsendringer eller utbyggelse av kraftsystemet er nødvendig. I tillegg, for å opprettholde vindkraft som en bærekraftig ressurs og en bærekraftig økonomi vil det være ønskelig å unngå unødvendig utbygging, og heller effektivisere de turbinene som allerede er utplassert.

Motivasjonen ved å effektivisere vindturbinene ved Smøla vindpark vil dermed være å unngå unødvendig bruk av ressurser samt unngå unødvendig utbygging. Det å effektivisere turbinene samtidig som alle skal være trygge på en god leveringssikkerhet av elektrisk energi, vil være viktig for alle i fremtiden. Ved effektivisering spares samfunnet for flere utbygginger av vindturbiner samtidig som økonomien rundt hvert prosjekt vil forbedres. Effektivisering kan også bidra til en lavere strømpris ved at ressursene blir brukt på en bedre måte samtidig som det blir omdannet mer elektrisk energi.

Ved at drift og vedlikehold endres til mer databaserte systemer vil det være ønskelig å drive dette på en annen måte enn den vanlige periodiske modellen. Det betyr at ved å bytte ut deler på en turbin etter en viss tid, er det knyttet sterkere sikkerhet til at det ikke oppstår en skade på systemet. Ved å se på tidligere observasjoner gjennom loggført data er det muligheter for at det oppstår sammenheng mellom én eller flere parametere. Ved en bredere forståelse av korrelasjoner og feilforløp vil det være mulig å unngå systemfeil og tap i produksjonstid.

Muligheten for at systemansvarlig kan få varsel når en del av systemet er utenfor normal standard ville hjulpet for å unngå feil og ødeleggelser på systemet. På denne måten blir det mindre vedlikehold og delene byttes bare når det trengs og ikke når det fortsatt er mulig å bruke dem. Dette gir også en økonomsik gevinst. En måte å gjennomføre dette på er å predikere feil før feilen oppstår. Ved å lære systemet til å forstå sammenhenger som kan gi en sannsynlighetsberegning om en feil vil oppstå i nær fremtid ville vært svært hjelpsomt.

## 1.2 Mål med oppgaven

Datasettet som er brukt i oppgaven inneholder analysedata som omhandler ett års produksjon, levert av Statkraft. I tillegg er en del av datasettet alarmer over forskjellige feil

som skjer for hver turbin. I følge Statkraft er yaw-systemet det området de har størst problemer med å predikere feil. Samtidig gir det store økonomsike utfordringer ved at det oppstår en feil på turbinen. Oppgaven vil dermed ha et eget fokus på å predikere feil med alarmer tilknyttet yaw-systemet [35].

På grunn av stor datamengde vil det være ønskelig å se på sammenhenger mellom parameterne for turbiner som får feil og parameterne for turbiner som det ikke skjer en feil med i det samme tidsrommet. Hvis en feil inntreffer og informasjonen om vindturbinene har en endring i parametere før en feil inntreffer, vil det være gode muligheter til å forstå feilforløpet bedre. Da er muligheten for effektivisering til stedet.

Ved å bruke produksjonsdataene på flere forskjellige turbiner vil målet være å bygge en maskinlæringsmodell som predikerer feil i systemet. Ved å sammenligne alarmer som oppstår når feilen inntreffer med vinddata, er det mulig å gi en forventet verdi av hva en turbin bør kunne omdanne av energi ved en gitt vindhastighet. Det er ønskelig å unngå de områdene i datasettet som omhandler høy vindhastighet og samtidig har en lav verdi for effekt. Ved å unngå disse verdiene vil vindturbinene muligens bli mer effektive.

Formålet med oppgaven er å bruke informasjonen i datasettet til å forstå om det er en korrelasjon rundt alarmer og tidsserier. I tillegg er formålet å se etter muligheter for å predikere fremtidige feil ved å se på endringer i parametere før en alarm inntreffer.

# 2. Teori

Kapittelet som omhandler teori vil fokusere på energi i vind, vindkraft og vindturbiner. Samtidig vil kapittelet også forklare enkle og kompliserte teoretiske begreper innen maskinlæring. Bakgrunnskunnskap, lignende boken "Python Machine Learning" av Sebastian Raschka og Vahid Mirjalili [26], innenfor maskinlæring er en fordel, men ikke nødvendig for forståelsen av denne delen av teorikapittelet. Store deler av teorien i denne oppgaven er bygd på flere prinsipper og teknikker innenfor maskinlæring som er beskrevet i den nevnte boken.

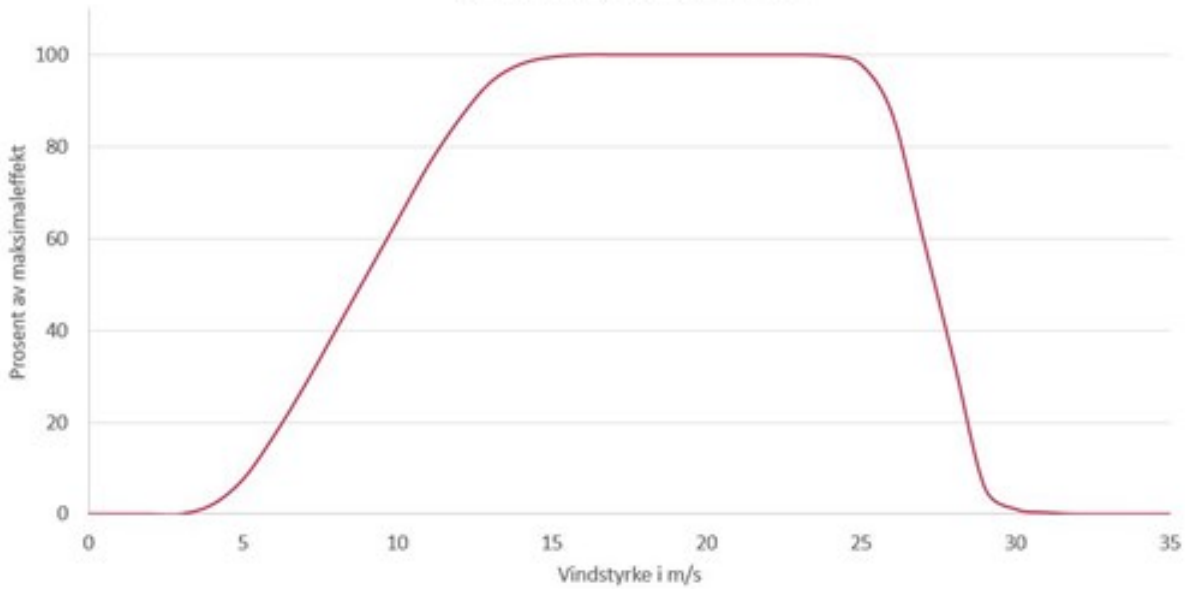
## 2.1 Energi i vind

Vind er luft i bevegelse. Luft er en masse, slik at det blir masse i beveglese. Det er energien fra sola, temperaturforskjeller og trykkforskjeller som påvirker luften slik at det oppstår en bevegelse. Dette gjelder både lokalt og ved store komplekse systemer. Ved økende temperaturforskjeller vil også vinden øke og være mer ujevn. Dette oppstår oftest i vintermånedene hvor temperaturforskjellene er store. I sommermånedene er det en mer konstant vind, samtidig som det er en generelt lavere vindhastighet. Forskjellene i vind og temperatur er også i endring mellom natt og dag på grunn av solpåvirkning over kortere tidsperioder. Ved klarvær har også sola en større påvirkning og dermed øker vindhastigheten og ujevnhetene [30].

Ujevnheter og vindhastighet avhenger også av posisjonering i terrenget og påvirkning fra lokale ujevnheter i form av fjell, bygninger, daler og lignende. Vindenergi i Norge er en energikilde som er mulig å nyte godt av grunnet vår lange kystlinje. Dette gir mange steder store muligheter fordi det ved kysten er mer jevn vind grunnet mindre støy i terrenget. Dermed oppstår jevne og relativt høye vindhastigheter. Samtidig er det områder hvor det bor få mennesker som gjør at det er mulig å bygge vindparker av en størrelse som har god klimaeffekt [19]. Et vindkraftverk har egenskaper til å omgjøre den kinetiske energien i vinden til rotasjonenergi i turbinen og deretter omgjøre rotasjonsenergien til elektrisk energi [15].

## 2.2 Vindkraft

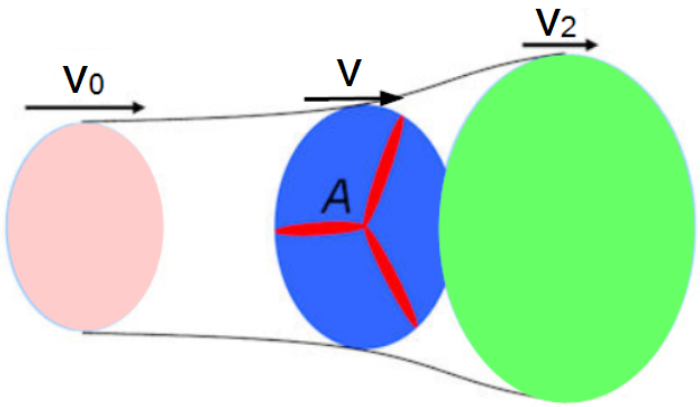
Vindkraft er en type kontaktkraft hvor luften som er i bevegelse presses mot turbinbladene. Bladene begynner deretter å rotere og omdanner energien i vinden til rotasjonsenergi. Hovedprinsippet innen systemer som omhandler vindkraft er å overføre rotasjonsenergi til elektrisk energi ved hjelp av en generator. På denne måten blir strøm tilgjengelig. Dette betyr at ved rotasjon på bladene vil generatoren generere strøm. Sammenhengen mellom effekt og vindstyrke er vist i figur 2.1. Bladene begynner å rotere ved en vindhastighet på cirka 4 m/s. Da begynner også vindturbinen å levere elektrisk energi. Mengden energi som omdannes øker i takt med vindhastigheten opp til en vindhastighet på 13-14 m/s, hvor effektkurven flates ut. Den flates ut fordi dette er maksimalt levert effekt for turbinen. Ved en økende vindhastighet opp til 25 m/s vil vindturbinen stoppe opp og dermed slutte å levere energi. Grunnen til det er at faren for for ødeleggelser i systemet øker ved høye vindhastigheter. Turbinen bremser opp og stopper, men vil i en kort periode fortsette å levere energi på grunn av treghet i systemet [22].




**Figur 2.1:** Grafisk fremstilling av en generell effektkurve for en vindturbin. Figuren viser en prosentvis utvikling av levert effekt i forhold til vindstyrke. Vindturbinen begynner å rotere og dermed levere effekt ved cirka 4 m/s og oppnår sin maksimaleffekt rundt 13-14 m/s før den skrus av og avtar idet den passerer 25 m/s av sikkerhetsmessige årsaker. Gjengitt med tillatelse fra [22].

## 2.3 Vindteori

Når vinden treffer vindturbinen endrer vindhastigheten seg som vist i figur 2.2. Vindhastigheten før vinden treffer turbinen er større enn vindhastigheten etter vindturbinen.




**Figur 2.2:** Beskriver ulike vindhastigheter som oppstår i området før og etter turbinen. Idet vinden treffer vindturbinen vil vinden avta ($v_0 > v_2$) og arealet på vinden vil øke. $v$ er vindhastighet ved turbinen og $A$ er arealet som bladene til turbinen beveger seg i. Inspirert av [10].

Kraften fra vinden som blir tatt opp av en vindturbin regnes ut ifra hvor mye krefter det er i vinden før og etter turbinen, og er gitt ved

$$F = \dot{m}(v_0 - v_2) \tag{2.1}$$

hvor $\dot{m}$ er massestrøm og $v_0$ og $v_2$ er vindhastigheten henholdsvis før og etter vindturbinen [41]. Den effekten som er teoretisk mulig å utnytte fra en vindturbin er gitt ved

$$\begin{aligned} P_T &= 1/2\dot{m}v^2 \\ \dot{m} &= \rho A v \\ P_T &= \frac{1}{2}\rho A v^3 \end{aligned} \tag{2.2}$$

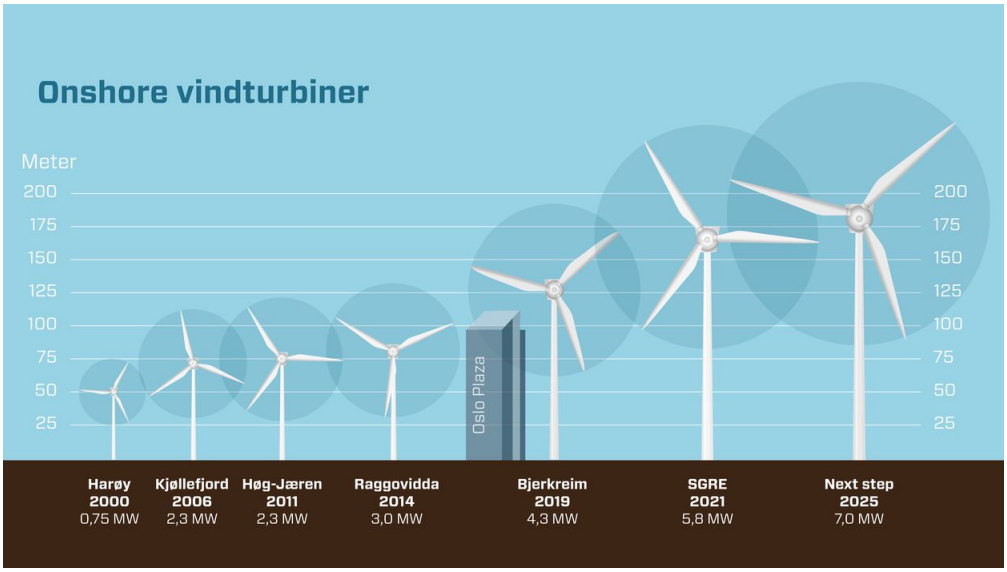
Hvor $\rho$ er massetettheten til luft ($1.2\ \frac{kg}{m^3}$), $A$ er arealet over den flaten vinden treffer turbinen ($m^2$), og $v$ er vindhastigheten målt på vindturbinen ($m/s$) [41]. Dette brukes til å finne virkningsgraden til turbinen ved

$$\eta = \frac{P_M}{P_T}, \tag{2.3}$$

hvor $P_M$ er den målte effekten på vindturbinen [41].

## 2.4 Vindturbin

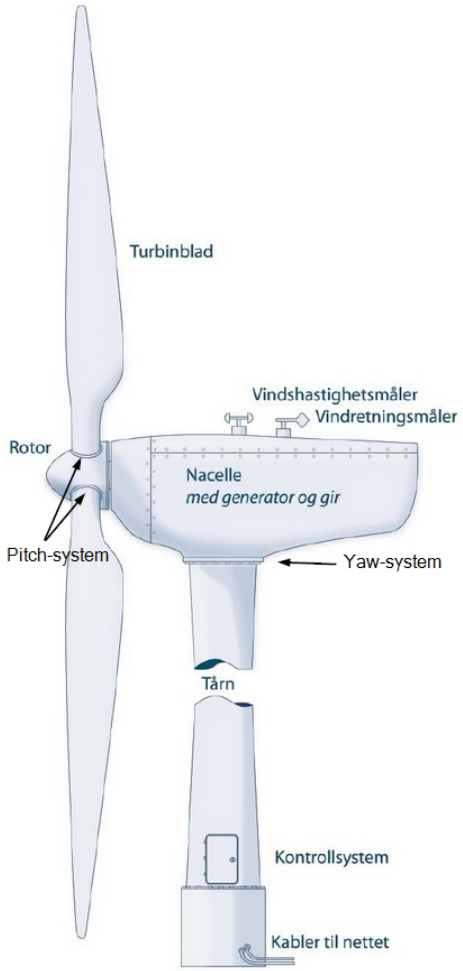
En vindturbin kan være på land og på vann og den består av turbinblader, tårn, maskinhus (med generator og girsystem), fundament og kontrollsystem. Størrelsen på turbinene varierer i forhold til type, men også i forhold til når turbinene er bygget. Figur 2.3 viser turbinene som ble bygget på land på begynnelsen av 2000-tallet hvor standarden på høyde til maskinhuset var 50 meter fra bakken. Utviklingen har vært rask og i de siste årene har høyden på nye turbinene økt, og utviklingen tilsier at høyden vil fortsette å øke i fremtiden [43].




**Figur 2.3:** Viser høydeutviklingen i meter over bakken til utvalgte landbaserte vindturbiner fra år 2000 til i dag og estimerte høyder for fremtiden. Navnet, året og levert effekt på turbinene er også presentert. Gjengitt ved tillatelse fra Kjersti Magnussen, TU Media [43].

### 2.4.1 Oppbygning

Bladene har en spesiell form som er optimalisert for å oppta høyest mulig effekt fra vinden. Innerst på turbinbladet, nærme maskinhuset, er det et pitch-system for hvert blad. Dette systemet endrer hvor stor vinkel turbinbladet har mot vinden. Hver turbin er også utstyrt med vindmålere som måler både retningen og hastigheten til vinden. Turbinen bruker disse målingene til å forstå hvilken retning turbinen skal peke i forhold til vindretning (yaw-system). Vinkelen på bladene tilpasser seg vindhastigheten (pitch-system). Plasseringen av pitch-systemet og yaw-systemet vises i figur 2.4. Turbinen vurderer kontinuerlig om den skal stoppe helt fordi sannsynligheten for skader på systemet øker ved økende vindhastigheter. Til dette brukes også vindhastighetsmåleren [21].

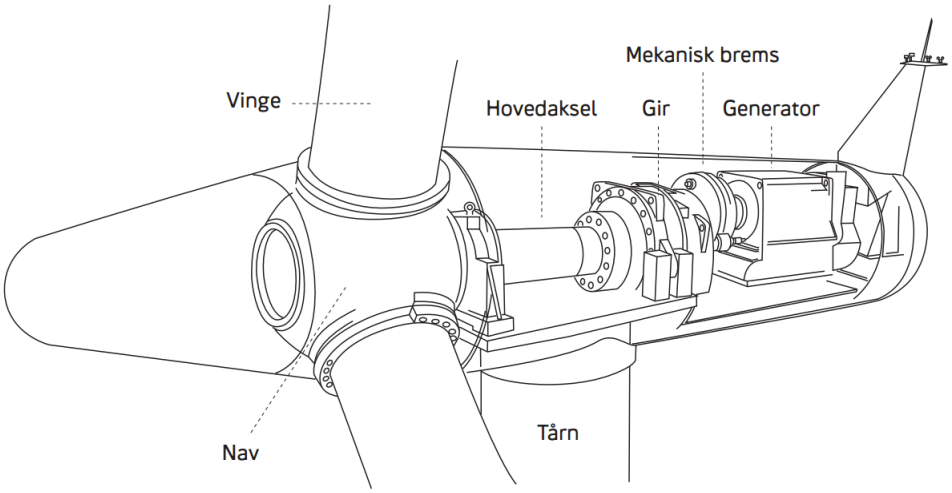



**Figur 2.4:** Viser en illustrasjon av hvordan en vindturbin ser ut og gir en oversikt over hvor de ulike hoveddelene er plassert. Plasseringen av pitch-systemet og yaw-systemet er også presentert. Inspirert av [12].

I tillegg til pitch- og yaw-systemet viser figur 2.4 hele oppbygningen av vindturbinen. Nacellen (maskinhuset) er plassert mellom tårn og turbinbladene. Den inneholder generator, gir, brems og akslinger. Plasseringen av nacellen er på toppen av tårnet. Tårnet inneholder et kontrollsystem og kabler som går til en sentral som flere turbiner er koblet til om turbinen er en del av en vindpark. Det er i nacellen rotasjonsenergi omdannes til elektrisk energi ved hjelp av en generator [29].

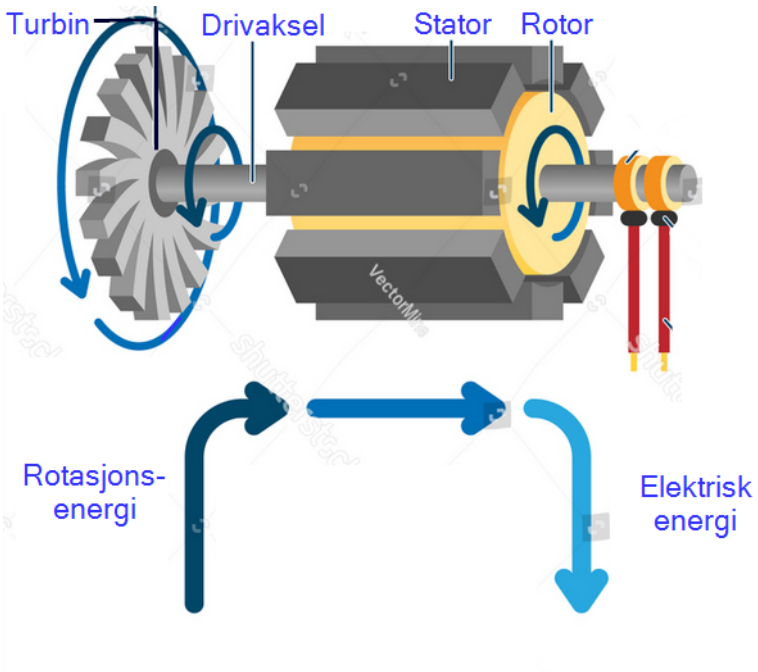
### 2.4.2 Fra rotasjonsenergi til elektrisk energi

På innsiden av nacellen, vist ved figur 2.5, vil drivakselen/hovedakselen rotere når turbinen er i bevegelse. Denne akselen har en påkoblet girkasse. Her endres turtallet fra en relativt lav verdi til en betydelig høyere slik at det er bedre tilpasset generatoren. Generatoren er den delen av systemet som gjør om rotasjonsenergi eller bevegelsesenergi til elektrisk energi [44].




**Figur 2.5:** Viser innsiden av nacellen som er plassert på toppen av tårnet. Idet vingene roterer roterer også hovedakslingen. Giret gjør det mulig å øke turtallet slik at det blir bedre tilpasset generatoren [44].

Ved hjelp av et magnetfelt rundt en magnet med to poler og en spole, vil det ved bevegelse av magneten i spolen bli produsert elektrisk energi. Om magneten (rotor) er i bevegelse inne i spolen (stator) vil spolen oppleve et varierende magnetfelt. Idet rotoren roterer raskt på grunn av økt vindhastighet, vil magnetfeltet i rotoren øke farten og få høyere rotasjonshastighet enn magnetfeltet i statoren. Dette er vist i figur 2.6. De fleste vindturbiner bruker asynkrone generatorer og produserer vekselstrøm [46].




**Figur 2.6:** Ved rotasjon i turbinen vil denne energien omdannes til elektrisk energi i generatoren via en drivaksel. Statoren står stille, mens rotoren roteres og det blir produsert elektrisk energi. Inspirert av [32].

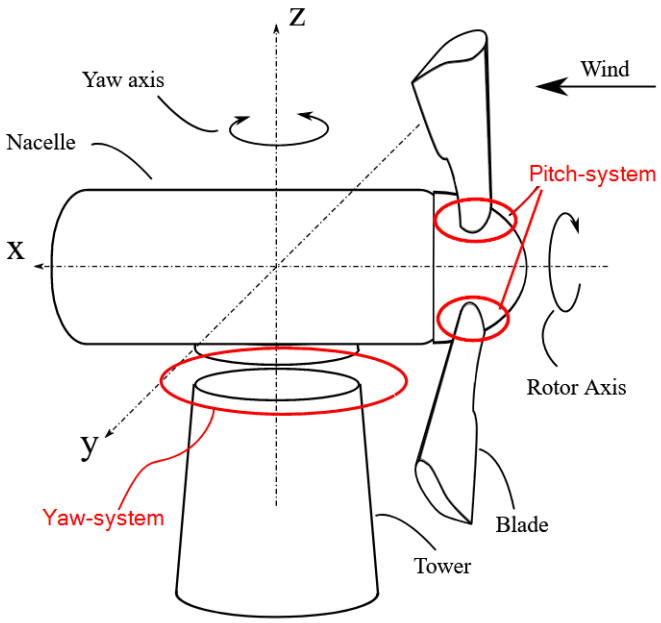
En asynkrongenerator drives av et roterende magnetisk felt. Dette vises i figur 2.6. Rotoren får ved belastning en langsommere hastighet enn det roterende feltet, dermed asynkront. I statoren er det et konstant dreiefelt som bestemmes av frekvens og viklingskonstruksjon. Dette dreiefeltet induserer en vekselspenning i rotorviklingene. Dette gir et dreiemoment i rotoren. Det er hastighetsforskjell mellom statorfeltet og rotoren hele tiden og dermed ordet asynkron. Dette er en optimal løsning for en generator i en vindturbin. Ved denne metoden vil det være mulig å ha forskjellig hastighet på omdreiningene i turbinen og dermed i rotoren [39].

Fra generatoren går strømmen ned, gjennom tårnet, til bunnen av vindturbinen. Her blir strømmen transformert til en høyere spenning via en transformator. Alle turbinene i en vindpark er koblet med jordkabler og ender opp i en felles transformatorstasjon. Deretter sendes strømmen fra transformatorstasjonen og ut på strømnettet [34].

Rotasjon i bladene fører til rotasjon i drivakselen. Drivakselen er en kobling mellom turbinen og generatoren. Dette vises også i figur 2.6. Oppgaven til drivakselen er å overføre bevegelsesenergien i turbinen til generatoren. I generatoren blir bevegelsesenergi omgjort til elektrisk energi [40].

### 2.4.3 Yaw-systemet

Under nacellen, øverst i tårnet, er yaw-systemet plassert. Dette vises i figur 2.7. Yaw-systemet brukes til å rotere en vindturbin mot vinden i horisontal retning. Dette systemet har kun én jobb; å bevege maskinhuset til vindturbinen i riktig retning i forhold til vinden. Akslingen til vindturbinen skal stå vinkelrett mot vinden for å oppnå optimal virkningsgrad. Informasjonen om hvilken retning vinden blåser blir kommunisert via vindmåleren som måler vindretningen. Den står på toppen av nacellen, som vist i figur 2.4. Samtidig er det viktig at turbinene er posisjonert i samme retning, slik at de ikke blir påvirket av hverandre på en måte som ikke er optimal [33].




**Figur 2.7:** Viser en illustrasjon av en vindturbin hvor pitch- og yaw-systemet er markert med rød farge. Inspirert av [45]

.

I delkapittel 2.4.1 er pitch-system nevnt. Dette er systemer som omhandler hvert enkelt blad og er vist ved rød markering, innerst på bladene i figur 2.7. Pitch-systemet blir dimensjonert ut ifra vindhastigheten målt på turbinen. For å opprettholde energiproduksjon over et større spekter av vindhastighet er det nødvendig å ha mulighet til å endre vinkelen på bladene. Derfor er pitch-systemet viktig [42].

### 2.4.4 Treghet

Vindturbiner opplever treghet (inertia). Treghet i en vindturbin er at vindturbinen ikke tilpasser seg vindhastigheten momentant. Den trenger tid på å nå en rotasjonshastighet og dermed en effektkurve som samsvarer med vindhastigheten. Hvor lang tid det tar før turbinen har tilpasset seg vindhastigheten varierer fra 2 til 9 sekunder. Det avhenger av hvilken størrelse turbinen har. Treghet oppstår også når vindturbinen bruker lenger tid til å bremse i forhold til hvor raskt vindhastigheten kan avta. På denne måten oppstår forflytninger i målinger som skyldes treghet [20].

Figur 3.6, i kapittel 3.5, omhandler vindhastighet og levert effekt. Den viser også at vindhastigheten øker og synker før den leverte effekten gjør det samme. Dette betyr at det vil være en treghet i systemet som påvirker når effekt blir levert i forhold til vindhastighet [20].

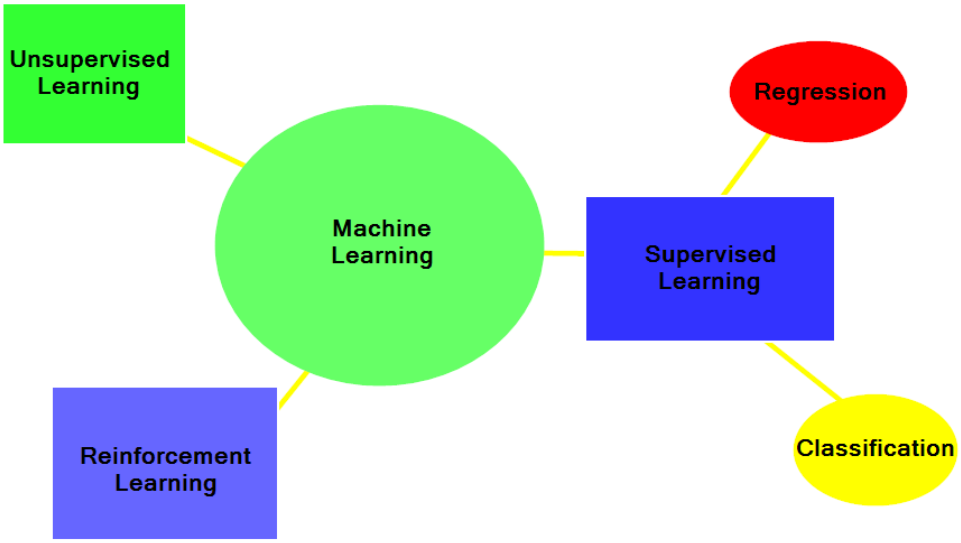
### 2.4.5 Effektivitetsgrense

Effektivitetsgrensen til en vindturbin betyr hvor stor andel av energien til vinden det er mulig for en vindturbin å omdanne til elektrisk energi. Innenfor vindkraft kalles denne grensen for "Betzgrensen". Betzgrensen er en effektivitetsgrense det er teoretisk mulig å oppnå. Den er på 59,3%, men dette er praktisk sett umulig å oppnå. Det er ikke mulig å overskride denne grensen fordi energien som blir tatt opp er regnet ut ifra vindhastighet før og etter turbinen. Dette er beskrevet i likning 2.1 [7]. En kommersiell vindturbin oppnår rundt 40% som praktisk maksimal effektivitetsgrense [41].

## 2.5 Maskinlæring

### 2.5.1 Hva er maskinlæring?

Maskinlæring er den delen av kunstig intelligens (Artificial Intelligence, AI) som inneholder selvlærende algoritmer som forstår seg på mønstre i datasett. Datamaskiner er kjent for å regne raskere enn den menneskelige hjernen. En kalkulator et et godt eksempel på det. Maskinlæring er også en form for oppgave som mennesker kan løse, men en maskin gjør det raskere. Mennesker er også flinke til å forstå seg på sammenhenger mellom forskjellige parametere. En datamaskin kan ved hjelp av algoritmer forstå seg på korrelasjoner og sammenhenger enda raskere enn mennesker. Dermed kan datamaskinen lage et mønster i store datamengder. Dette er grunnen til at maskinlæring er en viktig del av utviklingen innenfor datakunnskap. Maskinlæring brukes for eksempel i fordelingen av e-post i spam-filter, bildegjenkjenning og oversettelsesprogrammer. Figur 2.8 viser at det er forskjellige grener under maskinlæring. Veiledet læring er en av disse [26].

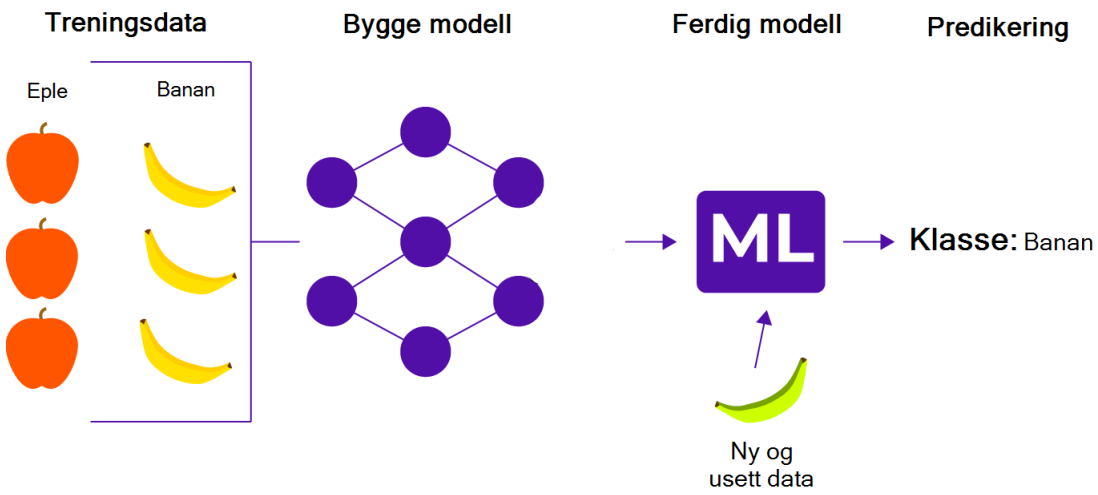



**Figur 2.8:** Overordnet oversikt over undergrupper innen maskinlæring. Regresjon og klassifisering er de mest brukte teknikkene innen veiledet læring, som er en av undergruppene innen maskinlæring.

### 2.5.2 Veiledet læring

Veiledet læring (supervised learning) er en av underkategoriene innen maskinlæring. I veiledet læring er målet å lære opp en modell basert på erfaringer fra utvalgt data. Basert på de lærte erfaringene er det ønskelig at modellen skal kunne predikere verdier fra ny/usett data [26].

Figur 2.9 viser hvordan veiledet læring fungerer. Først skilles treningsdata ut fra datasettet. Deretter blir det bygd en modell som er tilpasset veiledet læring og det valgte problemet. Deretter trener datasettet på å finne sammenhenger mellom parametere ved hjelp av den valgte maskinlæringsmodellen. Hvis modellen er god klarer modellen å predikere riktig når ny data blir implementert. Ut i fra erfaringer fra treningsdatasettet vil modellen med varierende sannsynlighet klare å predikere riktig.




**Figur 2.9:** Viser en oversikt over et eksempel på hvordan en veiledet maskinlæringsmodell er bygd opp. Treningsdata blir brukt til å forstå sammenhenger mellom parametere i den valgte algoritmen. Etter trening blir ny data testet på modellen, som ved hjelp fra erfaringer klarer å predikere hva ny data er. Denne figuren omhandler et *klassifiseringsproblem*. Inspirert av [17].

Figur 2.9 viser et eksempel på at modellen klarer å predikere at banan er ny/usett data i modellen. Modellen bruker ulike parametere til å forstå sammenhenger og dermed oppnår modellen erfaringer fra treningsdatasettet. Datasettet kan også endres slik at mønsteret passer enda bedre til å løse det nødvendige problemet. Ved å gjennomføre endringer i datasettet er kunnskap om hva datasettet inneholder nødvendig. Dette kalles *funksjonsteknikk* [26].

### 2.5.3 Funksjonsteknikk og dataprosessering

Funksjonsteknikk (feature engineering) er en viktig teknikk som brukes for å hjelpe maskinlæringsmodellen til å lære best mulig. Ved kunnskap om de forskjellige parameterene, vil det være mulig å fjerne unødvendig informasjon eller gi mer gjennomslagskraft på de verdiene som er mest relevante. Det er en stor og viktig del av god maskinlæring å ha bakgrunnsinformasjon om datasettene som brukes. På denne måten vil modellen lære mer korrekt [47].

Dataprosessering er å samle og tilpasse data til en maskinlæringsmodell. Flere maskinlæringsalgoritmer krever at data er prosessert på en spesiell form. Det er også viktig at det er korrekte dimensjoner på datasettet. For eksempel, i en regresjonsmodell er det et krav om at verdiene som det er ønskelig å predikere er kun én verdi og ikke flere som gjelder for en klassifiseringsmodell [8].

### 2.5.4 Regresjon og klassifisering

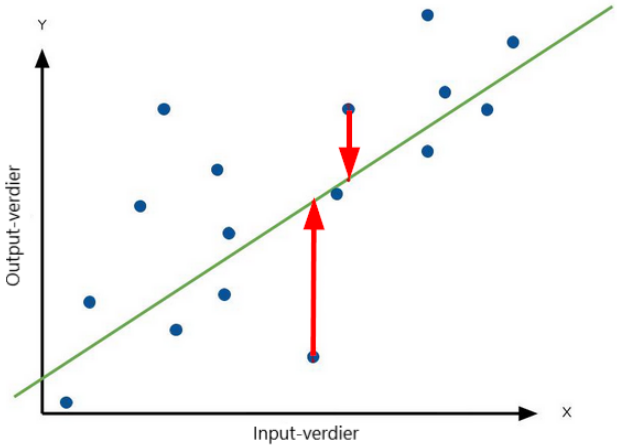
Innenfor veilet læring er det hovedsaklig to varianter. Det er *regresjonsproblem* og *klassifiseringsproblem*. Figur 2.9 viser et eksempel på et klassifiseringsproblem hvor målet er å forstå hvilken kategori en ny input hører til [8].

Ved et *regresjonsproblem* vil målet være å bestemme en verdi. Verdien blir bestemt ut ifra erfaring og sammenheng mellom parameterne i datasettet. Eksempelvis, i et datasett med innhold av ulike parametere hos en stor gruppe mennesker (høyde, vekt, blodtrykk, sykdomsforløp, alder osv.) vil det være mulig å predikere informasjon angående et nytt menneske. Det vil være mulig å predikere alderen til et menneske som oppgir ny, men samme informasjon, bortsett fra alder. Resultatet for predikeringen er basert på erfaring på hvordan korrelasjonen mellom parameterne fungerer. Derfor er det også en fordel å ha mye data, da dette gir økt erfaringsgrunnlag [26].

Dette skiller seg fra et *klassifiseringsproblem* hvor målet er å bestemme hvilken klasse/kategori resultatet passer til [8]. Figur 2.10 derimot viser regresjon. Punktene i figuren er predikerte målinger, hvor målet er å treffe linja i figuren. Avviket mellom punktet og linja kan regnes ut som en feil og kan brukes som et resultat som forklarer hvor god den predikerte verdien er. Gjennomsnittet av disse målingene kalles "Mean Absoulte Error" [16].

### 2.5.5 Mean Absolute Error (MAE)

MAE er en metode for å finne den gjennomsnittlige feilen på predikeringen. Dette er vist i figur 2.10, hvor feilen er vist ved de røde pilene. Ved å måle det absolutte avviket mellom den predikerte verdien til den optimale verdien, oppnås et resultat for feilprediksjon. Punktene i figur 2.10 kan dermed beskrives som de predikerte verdiene, mens linja i figuren inneholder verdiene hvor punktene optimalt sett skulle vært plassert [16].




**Figur 2.10:** Viser en regresjonsgraf hvor punktene i figuren er de predikerte verdiene, mens linja er det optimale resultatet. Den gjennomsnittlige verdien for alle avvikene mellom punktene og linja (visualisert med rød pil) beskriver "Mean Absolute Error". Inspirert av [25].

Likningen for MAE er

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{2.4}$$

hvor $n$ er antall målinger og $y_1 - \hat{y}_1$ er avviket fra punkt til linje for hver måling [16].

### 2.5.6 Begreper innen maskinlæring

Tabell 2.1 gir en kort forklaring av teoretiske begreper som brukes innenfor maskinlæring.

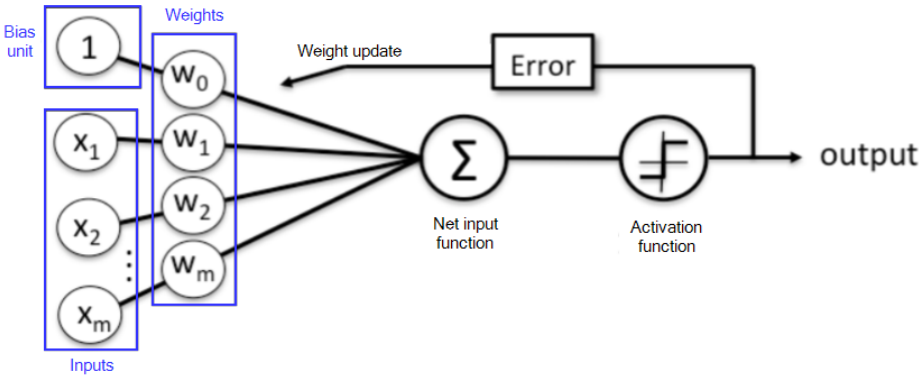
**Tabell 2.1:** Oversikt over viktige begreper innen maskinlæring.

| **Begrep** | **Forklaring** |
|---|---|
| Overfitting | Overfitting eller overtilpasning betyr at modellen samsvarer *for* godt til inputdata, og dermed blir det vanskeligere å tilpasse modellen til ny data. Ved å unngå overfitting blir modellen mer generell og mulig å bruke på lignende datasett [2] |
| Underfitting | Underfitting er det motsatte av overfitting og betyr at modellen ikke er kompleks nok for å forstå sammenhengen i parameterene godt nok idet modellen trenes [26]. |
| Standardisering | Dimensjonene på dataene blir endret fra sine orginalverdier til verdier som er nedskalert og tilpasset alle verdiene i datasettet. Grunnen til dette er å unngå at verdiene får stor variasjon i størrelse. Verdiene i datasettet blir ofte transformert til verdier mellom -1 og 1 [26]. |

| | |
|---|---|
| Aktiverings-funksjon | Aktiveringsfunksjon eller overføringsfunksjon bestemmer hvordan outputen beskrives. En aktiveringsfunksjon kan transformere verdier til en bestemt type outputverdier. For eksempel kan en aktiveringsfunksjon bestemme at negative verdier gir en output lik 0, og øvrige outputverdier lik den verdien de har før aktiveringsfunksjonen. Denne varianten av aktiveringsfunksjon kalles Rectified Linear Unit (ReLU) [26]. |
| Tapsfunksjon | Tapsfunksjonen kartlegger hva som oppstår av feil i en gjennomgang av treningen. Deretter er målet at modellen skal lære av sine feil og prøve å unngå disse til neste gjennomkjøring [26]. |
| Bias | Bias er ekstra input for inputverdiene og inputen i gjemte lag i en maskinlæringsmodell. Den hjelper på dimensjoneringen av datasettet, og er en uunngåelig del av maskinlæringsprosessen [14]. Dette vises i figur 2.11 og 2.13. |
| Vekter | Vekter (weights) betyr kort sagt hvor mye de ulike inputene skal vektlegges. For hver gjennomkjøring av en maskinlæringsmodell tilføres det nye vekter som læres ut ifra den forrige gjennomkjøringen av modellen. Disse vektene endrer seg for hver gjennomkjøring (hver epoch) slik at de forskjellige inputene vektes mer og mer riktig etter som modellen forbedrer seg for hver gjennomkjøring (hvis den gjør det) [18]. |
| Batchsize | Batchsize er hvor stor andel av data som trenes om gangen. Hvis batchsize er 10 og antall samples/rader er 100 betyr det at først trenes de 10 første verdiene, deretter de neste 10 osv. Dette gjentar seg for hver gjennomkjøring av treningen (for hver epoch) [26]. |
| Epoch | Epoch er en gjennomkjøring av hele treningsdatasettet gjennom modellen [26]. |
| Nevron | Et nevron er en del av et lag, og kort sagt flere nevroner og flere lag er unikt innen dyp læring (kapittel 2.5.8). Et nevron fungerer som et ettlagsnettverk (figur 2.11) og tar inn én eller flere inputs og er multiplisert med forskjellige vekter og dermed summert til en verdi, som er innholdet i nevronet. Denne summeringen sammen med en aktiveringsfunksjon bestemmer nevronets output. Et nevron er vist i figur 2.11 i kapittel 2.5.7 [28]. |

### 2.5.7 Ettlagsnettverk

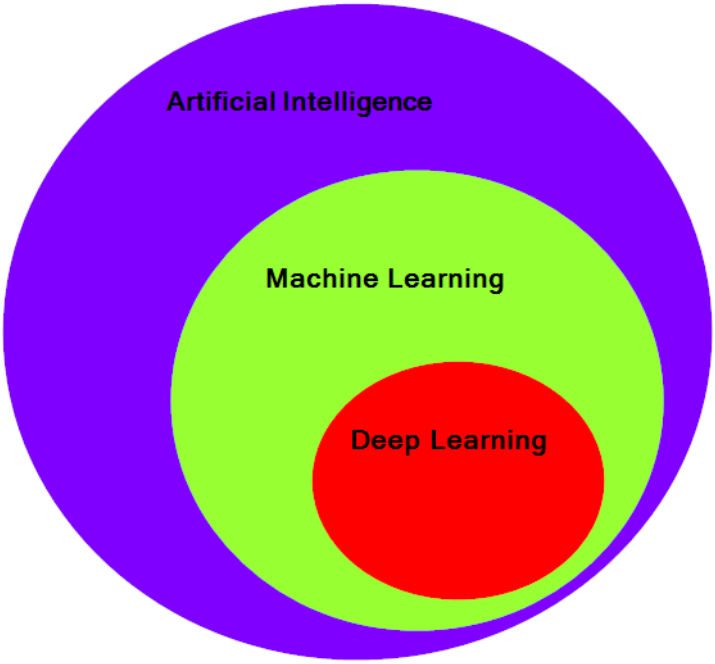
Ettlagsnettverk (single layer neural network) er den enkleste versjonen av et nevralt nettverk. Den innholder kun et input-lag og et output-lag. Figur 2.11 viser den enkleste oppbygningen, med kun én output. Først inneholder figuren et input-lag (med bias unit, som beskrevet i tabell 2.1) med vekter som vekter de forskjellige inputene med ulike verdier. Dette samles ved en inputfunksjon, og deretter blir verdiene ført inn i en aktiveringsfunksjon. Tapsfunksjon (Error) blir også brukt for å oppdatere vektene til neste gjennomkjøring av modellen slik at den kan optimalisere resultatet (se tabell 2.1). Til slutt blir outputverdiene presentert. Hvis det er gjemte lag i tillegg til output- og input-lag, omtales dette som et dypt nevralt nettverk [26].




**Figur 2.11:** Viser oppbygningen av ettlagsnettverk med inputs og bias, hvor disse er vektet forskjellig ved hjelp av vektene og så samles i en inputfunksjon. Fra inputfunksjonen føres verdiene gjennom en aktiveringsfunksjon og en tapsfunksjon (som oppdaterer vektene). Til slutt gir nettverket en output. Inspirert av [26].

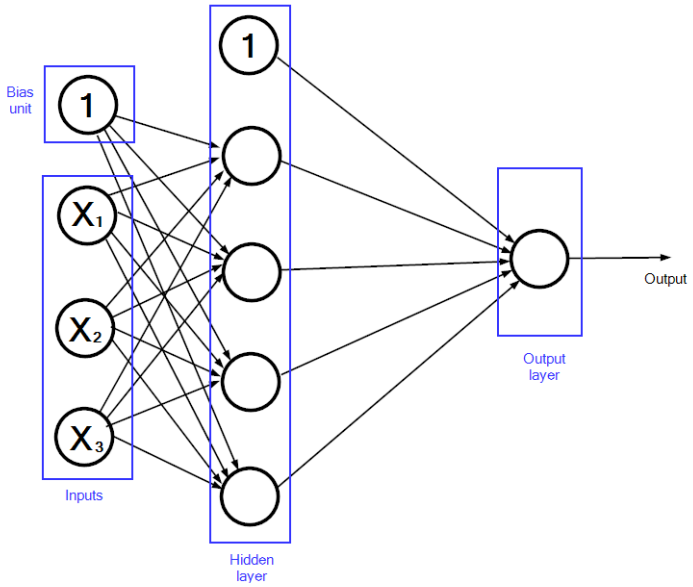
### 2.5.8 Dyp læring

Dypt nevralt nettverk (Deep Neural Network), eller dyp læring, er en utvidet teknikk innen maskinlæring. En av forskjellene som skiller dyp læring fra maskinlæring er at dyp læring omhandler flere lag som gjør at modellen kan lære ved flere kombinasjoner og prøver å finne sammenhenger ved ulike og mer kompliserte teknikker. Figur 2.12 viser at dyp læring er mindre generell enn maskinlæring, men det er også en underkategori som gir sine fordeler som dette delkapittelet skal handle mer om [26].




**Figur 2.12:** Viser dyp læring som en underkategori av maskinlæring og hvor maskinlæring er en underkategori av kunstig intelligens.

Dyp læring er steget videre fra enkeltlag nevralt nettverk til multilag nevralt nettverk, vist i figur 2.13. Det vil si at en modell går fra å ha ett input-lag og ett output-lag til å ha ett input-lag, ett eller flere gjemte lag (hidden layers) og ett output-lag. Figuren predikerer kun én outputverdi, som er tilfelle ved et regresjonsproblem [26].




**Figur 2.13:** Viser et eksempel på en modell innen dyp læring med tre lag, et flerlagsnettverk. Et input-lag, med 3 inputs og en bias-unit, et gjemt lag med totalt 5 units og et output-lag som gir én output. Vektene, aktiviseringsfunksjon og tapsfunksjon er gjemt mellom lagene for tydeligere visualisering.

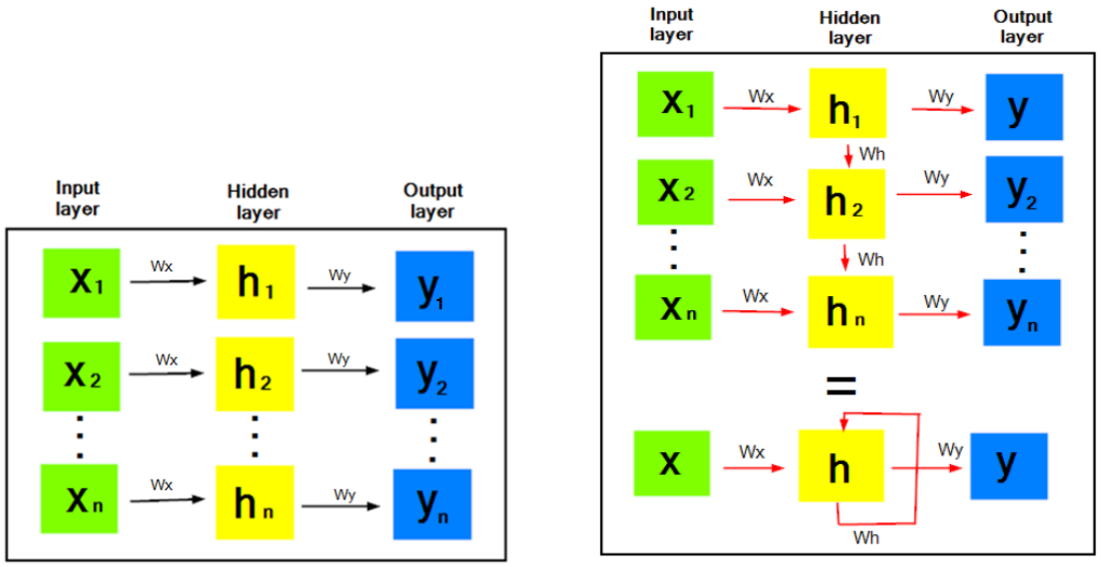
Dyp læring har i tillegg til begrepene i maskinlæring flere begreper som er viktig å forstå for å få en bedre oversikt over hva dyp læring er og hvordan det skiller seg fra vanlig maskinlæring. Noen av disse begrepene er beskrevet i tabell 2.2.

**Tabell 2.2:** Tabellen viser oversikt over begreper som er viktige innen dyp læring og bygger på begrepene innen maskinlæring (tabell 2.1).

| **Begrep** | **Forklaring** |
|---|---|
| Lag | Hvert lag prøver på forskjellige måter å lære seg sammenhengen mellom parametere i et datasett for å minimalisere tapsfunksjonen. Figur 2.11 inneholder ett input-lag og ett output-lag. Alle lag består av inputverdier, vekter og en aktiveringsfunksjon. Figur 2.13 inneholder flere lag. Ved flere lag omtales maskinlæringsprosessen som dyp læring. Outputen til de forskjellige lagene blir inputen i det neste laget. Antall lag er det som skiller dyp læring fra vanlig maskinlæring [26]. |
| Dense | En type lag er *dense-lag*. Dette er den vanligste typen lag i dyp læring. Et eksempel på et dense-lag er det gjemte laget i midten av figur 2.13. Alle units/nevroner får en input og en output som har en sammenheng med henholdsvis det forrige og det neste laget [26]. |
| Dropout | *Dropout-lag* fortynner eller regulerer verdiene slik at det ikke blir mye unødvendig data. Det betyr kort fortalt fjerning av nevroner, og dermed blir modellen mer generell og er mindre utsatt for overfitting [26]. |

Artificial neural network (ANN) eller kunstig nevralt nettverk er informasjonsflyt fra et input-lag via ett eller flere gjemte-lag til et output-lag. Målet er å etterligne en menneskelig måte å se sammenhenger på. Altså det biologiske nevrale nettverket. Ved å se sammenhenger gjør hjernen seg opp en mening ved hjelp av et system. Det samme prinsippet blir brukt ved kunstig nevrale nettverk, som eksemplene i figur 2.11 og i figur 2.13 viser. Målet er å se sammenhenger og vektlegge sammenhengene [26].

Vanlig ANN omtales som direkte informasjonsflyt. Det betyr at nettverket er konstruert til å forstå sammenhengene i prosessen. Den finner sammenhenger og bestemmer seg for hva den vil vektlegge og dermed gir en output. Dette er vist i figur 2.14a. En annen variant av et nevralt nettverk er *recurrent neural network* (RNN). RNN har en lignende metode som ANN, men med hukommelse. Den tar med seg en del fra det forrige gjemte laget (hidden layer) i modellen, og gir det som en ekstra input til neste lag i tillegg til den vanlige inputen. Dette vises ved figur 2.14b [26].




**(a)** Figuren viser vanlig nevralt nettverk (ANN) med tre lag. Et input-lag, et gjemt lag og et output-lag. Vektene som oppdateres mellom lagene blir også vist i figuren. Det gjemte laget får informasjon fra input-laget og output-laget får informasjon fra det gjemte laget. Rekefølgen på inputene har ikke betydning.

**(b)** RNN bruker informasjon fra forrige gjemte lag i tillegg til informasjon fra input-laget. Vektene fra det gjemte laget blir dermed en kombinasjon av informasjon fra det forrige gjemte laget og fra input-laget. Rekkefølgen på inputene har dermed en betydning.

**Figur 2.14:** Figuren viser forskjellen på vanlig nevralt nettverk (ANN), og tilbakevendene nevralt nettverk (RNN). Inspirert av [48].

### 2.5.9 RNN (Recurrent Neural Network)

RNN brukes i de fleste tilfeller i sekvensiell data hvor rekkefølgen har betydning. Tekstdata er et eksempel på hvor rekkefølgen på ordene har stor betydning. Dette gjelder også for tidsserier hvor rekkefølgen på når dataene oppstår er viktig [26].

I RNN er målet å bruke tidligere informasjon til å forstå hva som skjer videre i prosessen. Den vanlige varianten kalles "Long Range Dependicies". Dette er et system som bruker input-informasjon til å predikere hva som skjer i neste steg. Dette systemet fungerer bra for små systemer som for eksempel at "Fargen på himmelen er blå". Hvor ordet "blå" er predikert fra det som tidligere har skjedd i setningen og viten om at sammenhengen mellom "himmelen" og "farge" er blå [26].

Innenfor RNN er det igjen mange mulige valg rundt hvilken type RNN det er nødvendig å velge for den oppgaven som er ønskelig å løse. De to neste delkapitlene omhandler mye tung maskinlæringsteori som beskriver en av disse retningene og er helt nødvendig for å forstå prinsippene bak valgene som er gjort i kapittel 3. For forståelsen av helheten i oppgaven er det ikke nødvendig å forstå dette i detalj, derimot er det viktig for å forstå de valgene som er tatt.

### 2.5.10 Problemet med forminsket eller eksplodert gradient

Forminsket eller eksplodert gradient oppstår når det er stor sekvensiell avstand i informasjon. Om en type informasjon oppstår langt frem i sekvensen vil det være et problem for algoritmen og finne denne sammenhengen (Long-term dependensies) fordi det oppstår et forminsket- eller et eksplodert gradient problem [26].

Denne feilen oppstår på grunn av måten vektene blir oppdatert på. De blir oppdatert ved å gå baklengs gjennom modellen for å bestemme hvor mye de forskjellige inputene i lagene skal vektlegges. Dette kalles "Back Propagation" som skjer i alle maskinlæringsmodeller. Problemet oppstår ved mange lag og viktigheten av at tidlige observasjoner kan bety mye for senere observasjoner som er tilfelle i RNN [26].

Dette et et viktig problem å løse for RNN-modeller, og i tillegg er problemet med dette systemet at det fungerer dårligere på større systemer. Eksempel: "Fargen på himmelen er blå. Dette er grunnen til at det er fint vær". Dette systemet er større og læringen oppfatter at det er fint vær av forrige setning. Denne teknikken fungerer på små problemer, men i store mer praktiske problemer blir dette vanskelig. Om systemet blir større og større har ikke vanlig RNN mulighet til å se sammenhengen så langt tilbake i tid. En løsning på dette er Long Short Term Memory (LSTM) [3].

### 2.5.11 LSTM (Long Short Term Memory)

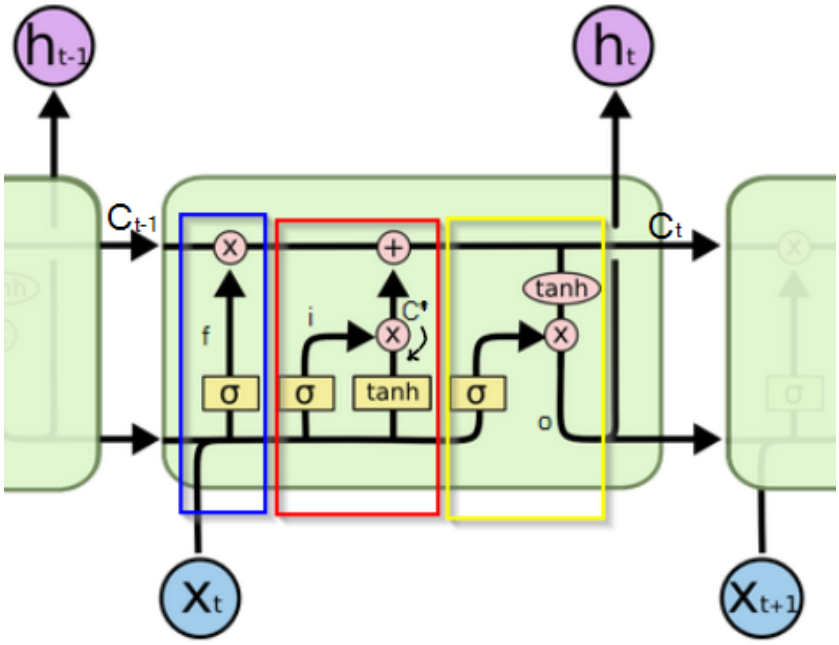
LSTM (Long Short Term Memory) er et godt verktøy for å unngå forminsket- eller eksplodert gradient. LSTM er kort fortalt en type RNN som løser dette problemet. RNN har input fra forrige gjemte-lag og vanlig input slik at outputen blir en blanding av informasjonen fra ny input og fra forrige lag. LSTM har også input fra forrige gjemte lag og ny input, men i LSTM skjer det mer enn at det bare blandes sammen til en ny output [23].

I en LSTM-celle vil det være input som i vanlig RNN. I tillegg vil det være tre porter (gates) som brukes ved oppdatering av cellen. I første port, gult området i figur 2.15, er målet å finne ut hva som *ikke* er ønskelig å ta med videre av informasjon. Det vil si å oppdatere celletilstanden slik at den forstår hva som *ikke* er nødvendig informasjon.

Til det brukes en port som ønsker å glemme informasjonen (forget-gate). Inputen går gjennom en aktiveringsfunksjon (sigmoidfunksjon) som sier at output er mellom 0 og 1 som tilsier hvor stor andel som skal glemmes. Dette vises matematisk ved

$$f = \sigma(W_f[h_{t-1}, x_t] + b_f), \tag{2.5}$$

hvor $\sigma$ er symbolet for sigmoidfunksjonen, $h_{t-1}$ er informasjon fra det gjemte laget som kommer fra forrige LSTM-celle, $b$ er bias knyttet til den konkrete porten og $W$ er matriseparameterene som hører til de respektive verdiene for den konkrete porten [23].




**Figur 2.15:** Viser LSTM-celle hvor blå, rød og gul rute omhandler henholdsvis forget-gate, input-gate og output-gate. $X_t$ er inputen i cella sammen med $h_{t-1}$ som er informasjon fra forrige LSTM-celle. Den øverste horisontale sorte pila beskriver hvilken tilstand cellen er i og fungerer som et transportbånd. Inspirert av [23].

Neste gate (input-gate) sier hvor mye som skal være med i celletilstanden av informasjon fra forrige gjemte lag og input-lag. Dette vises i figur 2.15 ved den røde firkanten. Først bestemmes andelen som skal være med i en sigmoidfunksjon, deretter bestemmes hva som skal være med i en enkel tanhfunksjon, som gir verdier mellom -1 og 1. Dette vises ved

$$\begin{aligned} i &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\ C' &= \tanh\left(W_C[h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{2.6}$$

hvor $C'$ beskriver hva som kan legges til ved tanhfunksjonen, og $i$ bestemmer hvilke verdier som faktisk skal legges til (input-gate) [23]. Til slutt kombineres dette med forget-gate og dermed oppnås celletilstanden til denne LSTM-cella, $C_t$, som er vist ved

$$C_t = f * C_{t-1} + i * C' \tag{2.7}$$

hvor $C_{t-1}$ er celletilstanden til forrige celle, og $C_t$ er den totale celletilstanden etter forget-gate og input-gate.

Til slutt beregnes output-gaten, vist ved den gule firkanten i figur 2.15. Output-gate omhandler beregningen av outputen til cella. Først beregnes mengden av informasjon som skal være en del av outputen. Det beregnes igjen ved hjelp av en sigmoidfunksjon. Gjennom tanhfunksjonen bestemmes celletilstanden til å være verdier mellom -1 og 1 [23]. Deretter multipliseres dette med sigmoidfunksjonen og finner en output. Denne outputen blir outputen til cella og inputen til neste celle. Outputfunksjonen beskrives ved

$$\begin{aligned} o &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o * \tanh(C) \end{aligned} \tag{2.8}$$

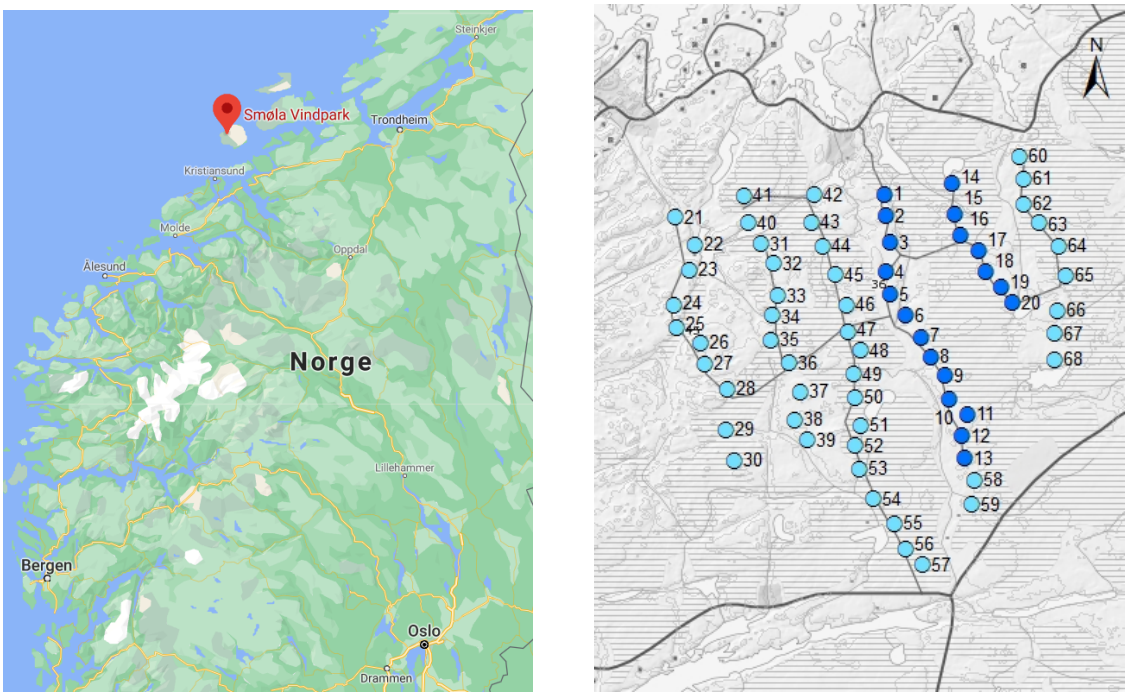
Hvor $o$ er output-gaten som beskriver hvor stor andel av informasjonen som skal være en del av outputen, $\tanh(C)$ er informasjonen i cella konvertert til verdier mellom -1 og 1 og $h_t$ er verdien for den totale output-matrisa. Denne verdien er outputen til LSTM-cella og inputen til neste LSTM-celle [23].

# 3. Metode

I dette kapittelet er målet å gjøre rede for hvordan arbeidet er utført. Bakteppet for hele arbeidet er å få oversikt og forstå produksjonsdata for vindturbiner på Smøla vindpark. Målet er å bygge en modell som passer best mulig til datasettene, som omhandler disse turbinene. Datasettene er tilsendt fra Statkraft. Kapittelet vil også inneholde sammenligninger av forskjellige typer datasett som er brukt på den samme modellen. Visualisering, beskrivelse og forståelse av datasettene er også en viktig del av oppgaven. Det vil derfor bli presentert hvilke datapunkter som er interessante å få et innblikk i. I tillegg vil det bli presentert hva slags type data som blir brukt, og hvordan denne type data blir prosessert. Det vil bli visualisert sammenhenger mellom når det oppstår feil i systemet og hva slags produksjonsmengde turbinene har når feilen oppstår. Dette er for å øke tiltro til datasettene og målingene. Til slutt blir rekkefølgen på maskinlæringsprosessen presentert.

## 3.1 Vindturbinene på Smøla vindpark

Som vist i figur 3.1a er Smøla vindpark plassert i Smøla kommune, i Møre og Romsdal. Figur 3.1b viser mer konkret hvor på Smølas vestkyst de 68 turbinene er plassert. Fargeforskjellen på punktene i figur 3.1b viser de forskjellige byggetrinnene. De mørkeblå punktene er det første byggetrinnet, og de lyseblå er det andre byggetrinnet. Disse byggetrinnene omtales som *Smøla 1* og *Smøla 2* [34].



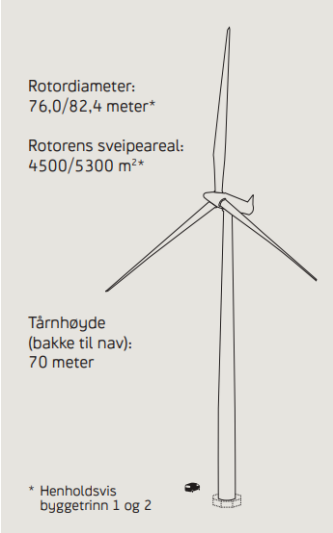

**(a)** Viser den geografiske plasseringen av Smøla vindpark [13].

**(b)** Plassering av de 68 turbinene på Smøla vindpark. Mørkeblå prikker gjelder for Smøla 1, og lyseblå prikker gjelder for Smøla 2. Gjengitt med tillatelse fra [36].

**Figur 3.1:** Figuren viser hvor i Norge Smøla vindpark er lokalisert, og plasseringen av de ulike turbinene.

Smøla 1 ble bygget i 2001 og 2002, hvor 20 turbiner ble åpnet av Kong Harald 05.09.2002. Utbyggingen av Smøla 2 foregikk i 2004 og 2005 og sto klar med 48 nye turbiner den 27.09.2005 [34]. Turbinene er av typen "Bonus B76/2000" og "Siemens SWT-2.3-82" for henholdsvis Smøla 1 [37] og Smøla 2 [38].

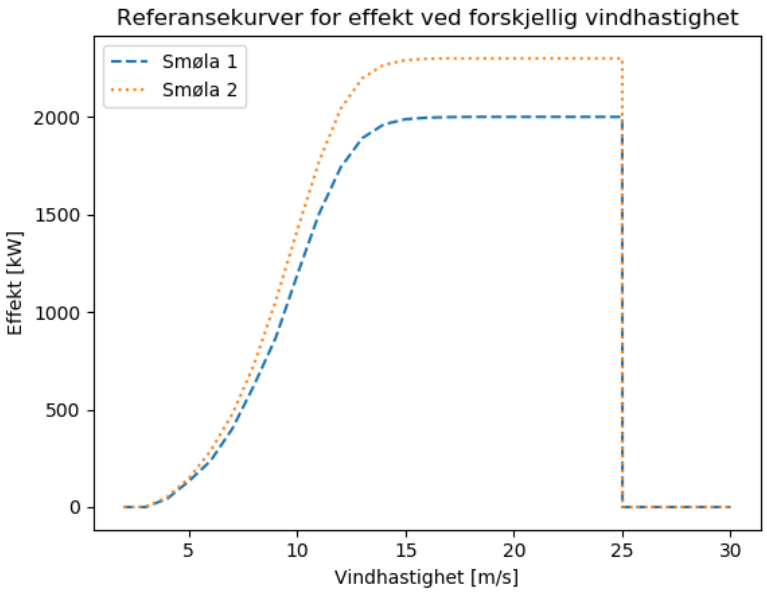
Figur 3.2 viser at konstruksjonen på Smøla 1 og Smøla 2 er relativt like. Det har vært en utvikling ved at turbinene i byggetrinn 2 har større dimensjoner på sveipeareal og rotordiameter. Det er en naturlig teknologisk utvikling, som tidligere nevnt i teorien, at vindturbinene øker i størrelse over tid. Dette gjelder også turbinene på Smøla [34].




**Figur 3.2:** Figuren viser størrelser knyttet til de ulike typene av vindturbiner på Smøla vindpark. Rotordiameter og sveipeareal er større på Smøla 2 enn på Smøla 1. Tårnhøyden er den samme. Gjengitt med tillatelse fra [34].

## 3.2 Referansekurver på Smøla

Verdiene på effektkurven varierer fra type vindturbin, og grafen i figur 3.3 viser en tilnærming av effektkurvene for turbinene Smøla 1 og turbinene Smøla 2. Grafene er brukt som referansekurver på hvordan effektkurvene utvikler seg med vindhastighet i de forskjellige byggetrinnene.




**Figur 3.3:** Datoene målingene for referansekurvene fant sted var 16.02.2018 og 11.06.2018 på henholdsvis Smøla 1 og Smøla 2. Turbinene på Smøla 2 har en høyere effekttopp enn Smøla 1. 2000 $kW$ på Smøla 1 og 2300 $kW$ på Smøla 2 [37][38].

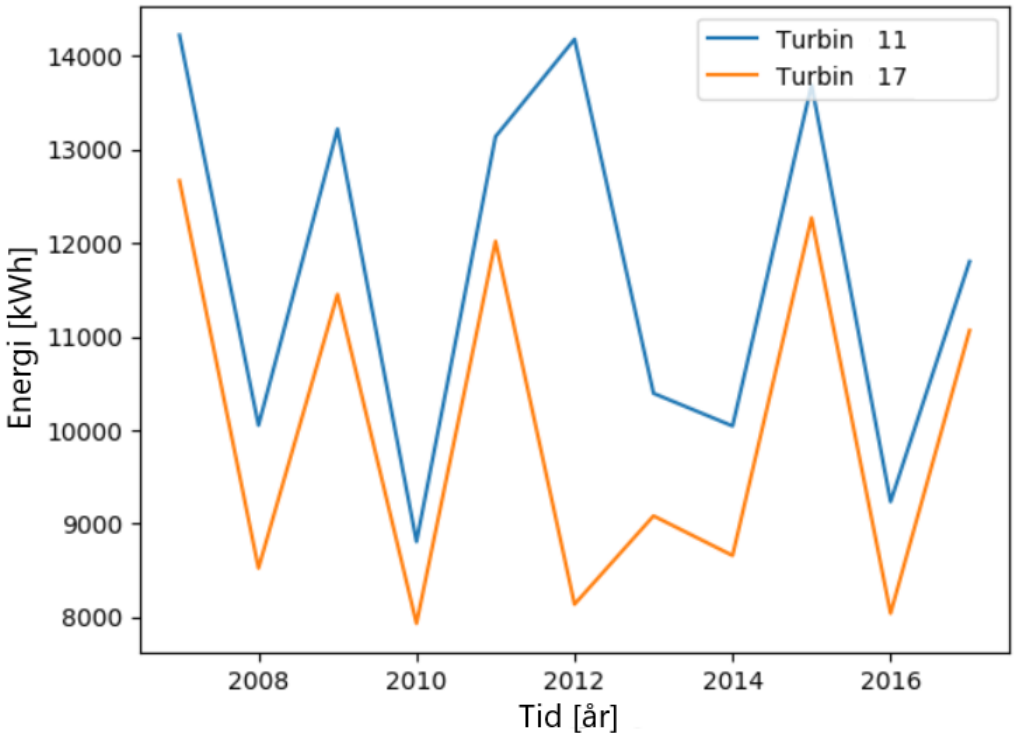
## 3.3 Datasett Smøla vindpark

Datasettene er delt opp i to hoveddeler og er levert sammen med informasjon om de forskjellige parameterene. De to nevnte hoveddelene er SCADA-data (Supervisory Control And Data Acquisition) som omhandler de første 20 installerte turbinene fra 2002 og de neste 48 turbinene som ble installert i 2005. Informasjonen om de forskjellige parameterne av vindparken blir omtalt som *Smøla 1* [37] og *Smøla 2* [38] i datasettene.

SCADA er et tilsynskontroll- og datainnsamlingssystem [1]. Dette systemet logger forskjellige parametere hvert 10. minutt for hver turbin. Dette er til bruk av statistiske analyser. I tillegg til SCADA-data innholder datasettet data om vind fra flere metrologimaster som er plassert i nærheten av turbinene. Det er også loggførte alarmer som viser når en systemfeil har oppstått. Alarmene har forskjellige alarmkoder som beskriver type feil. Datasettet for Smøla 2 er mer avansert og har flere parametere enn Smøla 1. Dermed er presentasjonen av data kun de parameterene som overlapper i Smøla 1 [37] og Smøla 2 [38]. En del av den interessante informasjonen i datasettene er vindhastighet og levert effekt for hver turbin.

## 3.4 Kritisk analyse av datasettet

Ved slike oppgaver er det viktig å forstå datasettene og dermed også verifisere om datasettet passer til den utvalgte modellen. For eksempel et enkelt år som ikke er egnet for en god modell kan oppstå i et datasett på Smøla. Ved å se på to turbiner på Smøla 1, er det mulig å vise dette. Ved å se på figur 3.4 er det relativt like og konsekvente verdier for de to turbinene, men ikke for året 2012. Derfor er det viktig å ha en underbyggende forståelse av datasettet for å implementere det i modellen. Det må være et representativt år eller en representativ turbin som ikke har unntakstilstand for at modellen kan implementeres og maskinlæringen kan være brukbar.



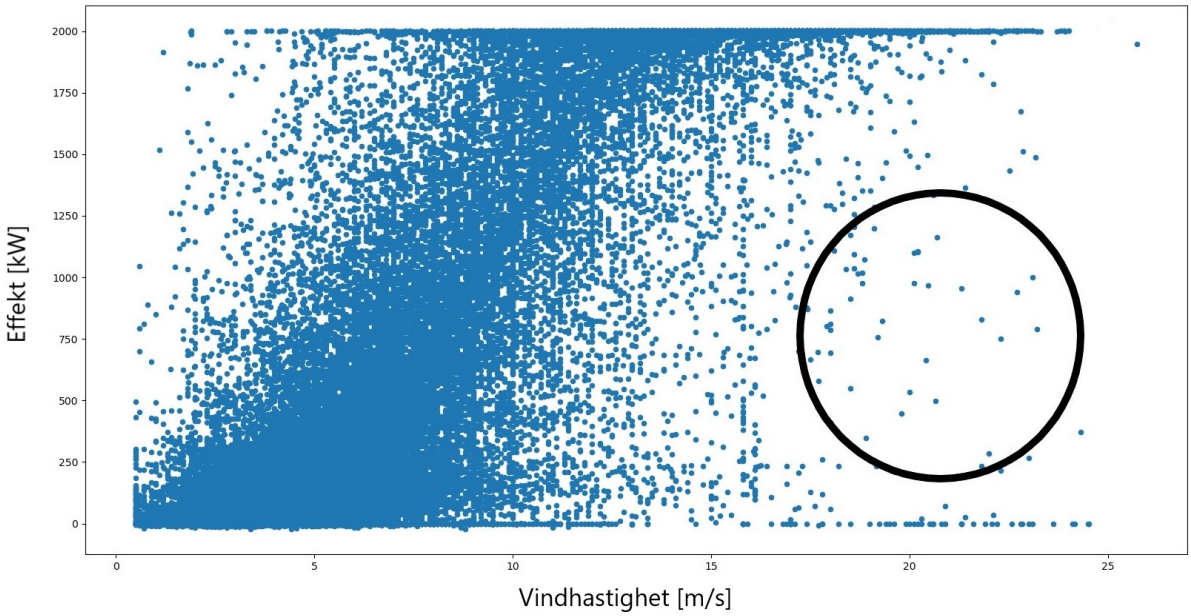

**Figur 3.4:** Viser total mengde energi levert per år over en tidsperiode på 10 år for to ulike turbiner.

## 3.5 Visualisering av data

Ved å sammenligne vinddata over tid og levert effekt over samme tidsperiode blir det mulig å visualisere hvordan vindhastigheten korrelerer med levert effekt. Som beskrevet i delkapittel 3.2 ved figur 3.3, som viser referansekurven til Smøla 1 og 2, er det slik en vindturbin optimalt vil oppføre seg i forhold til vindhastighet og levert effekt.

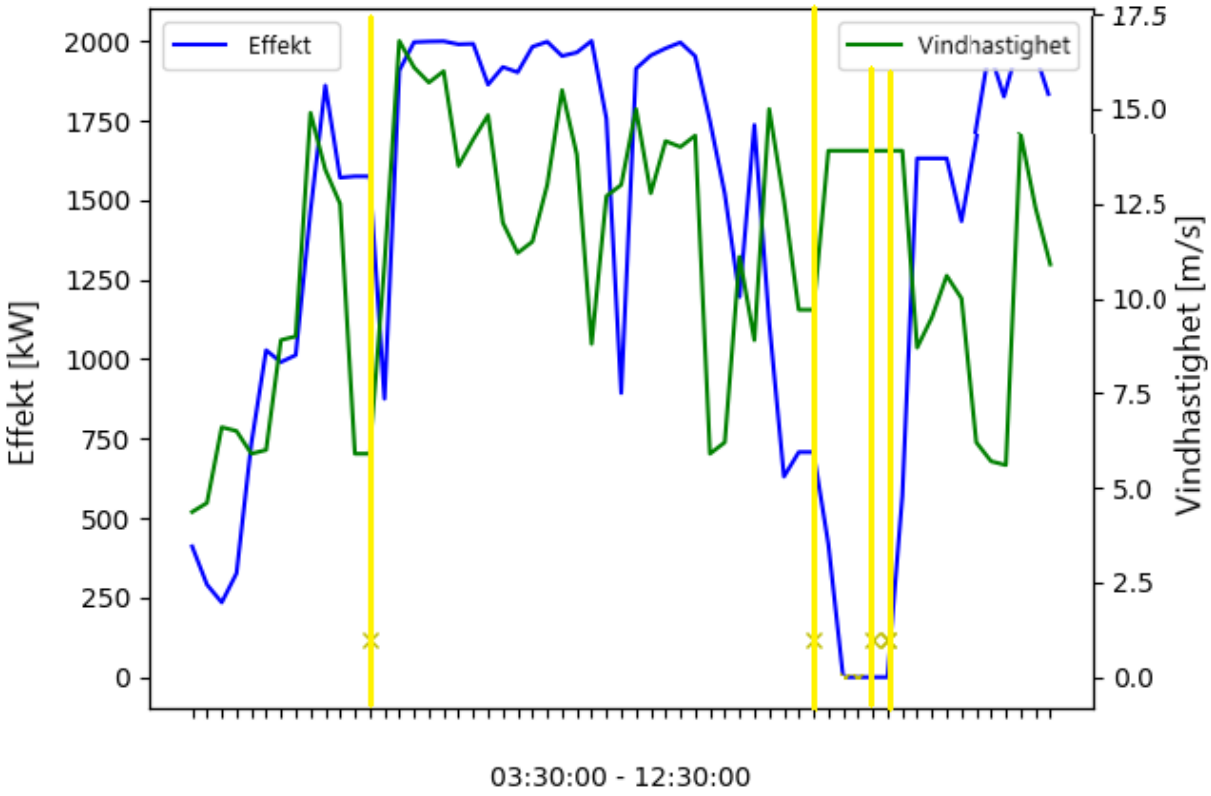
For å få oversikten over hvilke datapunkter som viker fra normalen er det ønskelig å finne de punktene som yter lav effekt ved høy vindhastighet, men under 25m/s, som beskrevet i teorien (2.2). Dette er målinger som utgjør en svakhet hos vindturbinen. Ved å sammenligne datapunktene for levert effekt og vindhastighet kan sammenhengen mellom målingene vises. Disse punktene er det verdt å sette spørsmålstegn ved.

Dette er visualisert i figur 3.5, som omhandler den tilfeldige valgte turbin 14 på Smøla 1 (plassering vist i kart ved figur 3.1b). Figurens sirkel beskriver målingene som er ønskelig å se nærmere på. Her er det fordelaktig å finne en grunn til hvorfor det er lav effekt ved høy vindhastighet.




**Figur 3.5:** Viser sammenhengen mellom vindhastighet og levert effekt for turbin 14. Punktene er målte verdier med 10 minutters mellomrom over ett år (2017). Sirkelen i plottet representerer punktene som vil være av interesse å unngå. Lav effekt og høy vindhastighet er utenfor normalen, som er et tegn på at det er mulig å effektivisere vindturbinen.

Ved å se nærmere på oppførselen til turbinen oppnås oversikt og innsyn på om datasettet gir sannsynlige og nøyaktige punkter. Vindhastighet og levert effekt burde være informasjon som korrelerer for hver enkelt turbin. Når det er høy vindhastighet (men under 25 m/s) bør det også være høy levert effekt. Figur 3.6 viser sammenhengen mellom vindhastighet og levert effekt av turbinen. Dette er et relativt tilfeldig valg av tidsperiode som omhandler 9 timer den 3. januar 2017. På figuren er det også mulig å se at det er en liten treghet i systemet, ved at effektkurven har en tidsforsinkelse relativt til vindhastighet.




**Figur 3.6:** Viser en utvalgt periode på 9 timer den 3. januar 2017. Den blå kurven representerer levert effekt, den rød viser vindhastighet ved de samme tidspunktene. De gule, vertikale linjene representerer tidspunktet en alarm oppstår.

I tillegg til å vise effekt og vindhastighet viser figur 3.6 når det oppstår en alarm, som er ved de gule, loddrette linjene. Her er det tydelig at selv om vinden er relativt stabil/høy så synker effektkurven kraftig. Når dette skjer inntreffer alarmen som indikerer at en feil har oppstått. Ved dette tidspunktet klarer ikke turbinen å levere like mye effekt som vindhastigheten tilsier at den burde. Dermed vil en effektivisering av vindturbinen være mulig dersom feilen unngås. For å finne ut når feilene oppstår er predikering ved hjelp av maskinlæring en mulig metode.

## 3.6 Maskinlæringsprosessen

Ved å bruke veiledet læring innen maskinlæring, som beskrevet i teorien, vil det være mulig å lære av erfaringene fra et datasett for å predikere at feil oppstår. I tillegg er det i denne oppgaven ønskelig å vite når feilen oppstår en valgt tid før den gjør det. I dette delkapittelet vises en mulig metode å gjennomføre denne prediksjonen ved å bruke maskinlæring.

### 3.6.1 Valg av datasett

I de fleste maskinlæringsmodeller har datakvaliteten stor betydning for å lykkes. Om det er mye tilgjengelig data er det svært komplisert å finne ut hvilke data som er den beste. Prøving og feiling er en lite komplisert metode å finne ut dette på. En annen mulighet er å gå inn i datasettet og finne parametere som har sammenheng med målet i oppgaven. Denne teknikken blir beskrevet i kapittel 2.5.3 og kalles *feature engineering*. Valget i denne oppgaven er å bruke TUR-filene for året 2017 for turbin nummer 47.

TUR-filene, som er forklart kort i tabell 3.1, er den delen av dataene som omhandler "signaler som ikke relateres til de andre grupperingene" [37]. I tillegg omhandler dette datasettet flere parametere som knyttes til yaw-systemet. Predikering av feil tilknytttet yaw-systemet blir forklart senere. Beskrivelsen av de andre datasettene er i tabell 3.1 og er tilsendt fra Statkraft [37].

**Tabell 3.1:** Kort beskrivelse av de ulike datasettene som er tilgjengelige.

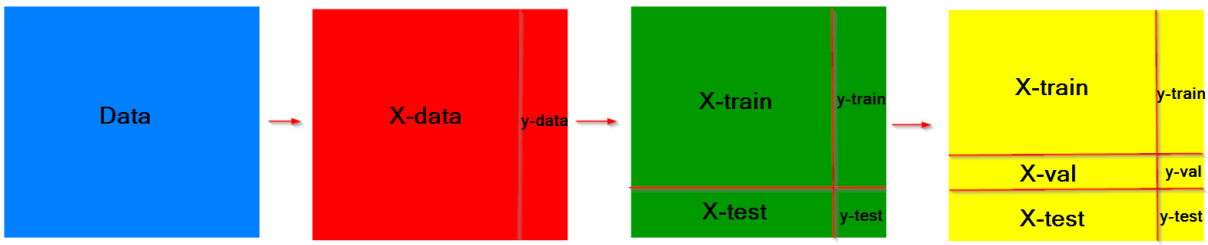
| **Datasett** | Forklaring |
|---|---|
| **TUR** | Signaler som ikke relaterer til andre grupper. |
| **FLG** | Statusinformasjon om turbinen. |
| **CNT** | Inneholder motsignaler. |
| **TMP** | Temperaturmålinger på turbinen. |
| **GRD** | Nettrelaterte målinger til turbinen. |

### 3.6.2 Databehandling før modellbygging

For å bruke en maskinlæringsmodell er det viktig å samle inn data. Dette steget er det mest åpenbare, men også et av de viktigste. Store mengder relevant data er ønskelig. Type data bestemmer også mange av valgene senere i maskinlæringsprosessen. Databehandlingen som gjøres er en tilpassing av alarmdata til datasettet samt fjerning av "not a number"-verdier. Databehandlingen fortsetter ved å splitte opp datasettet i nødvendige deler.

Fra hele datasettet (illustrert ved den blå boksen i figur 3.7) splittes en parameter (en søyle) fra datasettet, som vist ved den røde boksen. Den kalles *ydata*, som er verditypen det er ønskelig å predikere for modellen. I dette tilfellet er det alarmene i datasettet som er *ydata*. Etter den første splitten, splittes datasettet mellom treningsdata og testdata, som vist i den grønne boksen. Treningsdatamengde i denne modellen er valgt til 70% av datasettet, og dermed er de resterende 30% satt av til å teste hvor god modellen er til slutt. Dette for å se hvor godt modellen predikerer.

Til slutt blir treningsdata splittet en gang til. Dette deles opp i ny treningsdata. I dette tilfelle er det 80% og 20% som splittes opp til henholdsvis ny treningsdata og valideringsdata. Dette er vist ved den gule boksen i figur 3.7. Bakgrunnen til mengden som blir splittet fra de forskjellige datasettene er at det er ønskelig å ha mer data til treningen enn til testingen. Størrelsen på de forskjellige splittene kan variere, men det vil alltid være ønskelig å bruke den største delen av datasettet på trening. Grunnen til dette er at det er ønskelig å bruke mest mulig data til trening [26].




**Figur 3.7:** Viser hvordan datasettet splittes opp i nødvendige deler før modellbygging. Den blå boksen visualiserer hele datasettet. Den røde boksen visualiserer en kolonne som blir splittet fra datasettet. Den grønne boksen splitter vekk en del som brukes til å teste modellen senere. Den gule boksen splitter treningsdatasettet en gang til for å få et valideringsdatasett.

Treningsdata er data som brukes til å trene modellen. Modellen trenes med disse verdiene for hver gjennomkjøring (epoch). Med god trening vil modellen bli bedre rustet til å fungere godt på ny data. Valideringssettet er en del av treningssettet. Valideringdataen evaluerer modellen under trening. På denne måten er det mulig å se etter hver gjennomkjøring hvor mye modellen nærmer seg den optimale løsningen. Modellen bør forstå hva den gjør feil og prøve en gang til. Valideringen av treningen skjer etter hver epoch og den kan da endre på vektene slik at modellen optimaliseres mer og mer [26].

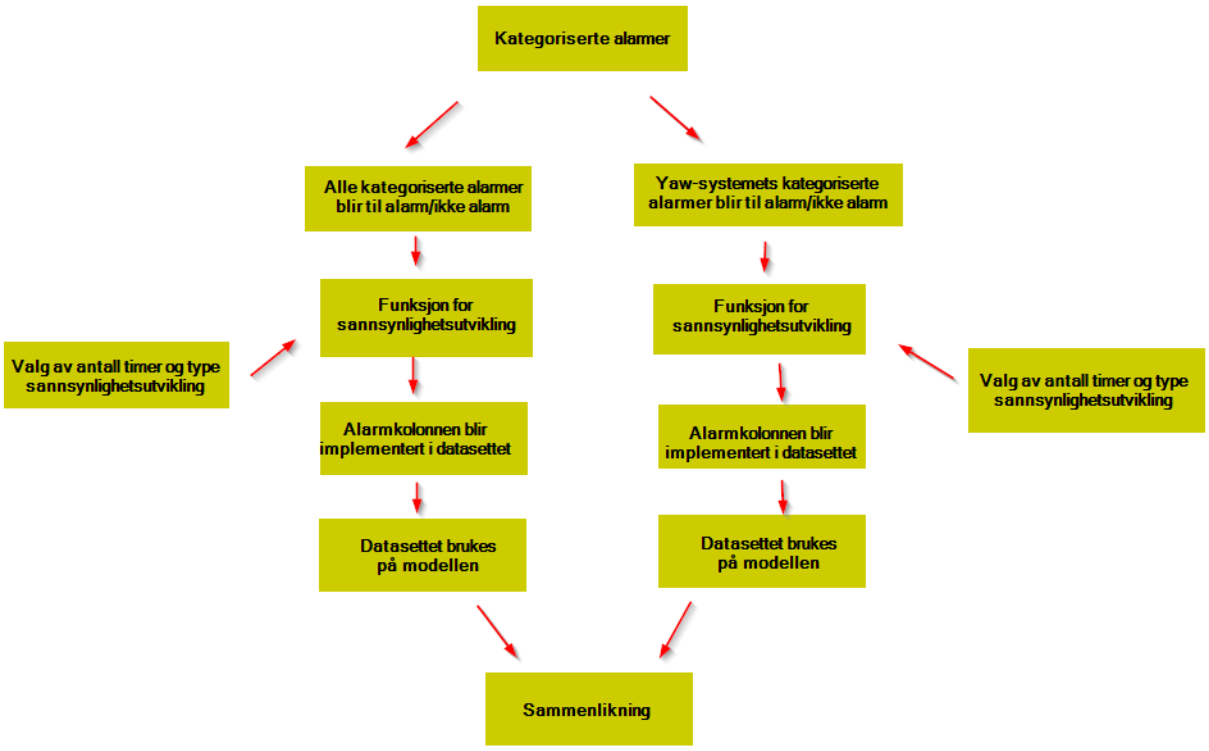
Vektene vil etter hver epoch bli kalkulert og oppdatert i treningssettet. Dermed blir modellen bedre tilpasset datasettet for hver epoch. Samtidig er det viktig at modellen blir generell nok, slik at den unngår overfitting (beskrevet i kapittel 2.5.6). Datasettet er det samme, men vektene er forskjellig for hver epoch. Før treningen gjennomføres også en standardisering ved å transformere data slik at de tilpasser seg hverandre i størrelse.

### 3.6.3 Alarmdata tilpasset en regresjonsmodell

De observerte verdiene omhandler alarmene. Som vist i figur 3.8 blir alarmene delt opp i to kategorier: alle alarmer og yaw-alarmer. Disse blir igjen samlet i to helt forskjellige datasett hvor i det ene datasettet som omhandler alle alarmene er alarmene gjort om fra kategoriserte alarmer til boolske verdier. Boolske verdier betyr verdier som bare har to muligheter, i dette tilfellet “alarm” eller “ikke alarm” [24]. I datasettet med bare

yaw-alarmer er det samme gjort. Altså enten yaw-alarm eller ikke alarm. Dette er vist ved andre linje i figur 3.8.

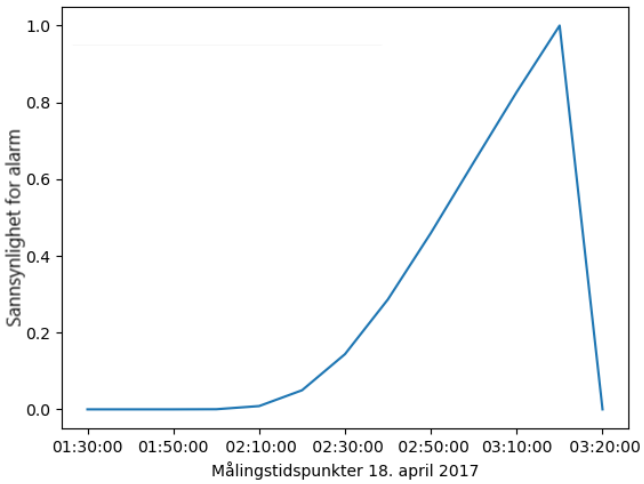
Dette er igjen gjort om til en sannsynlighetsutvikling som viser hvor sannsynlig det er at en alarm inntreffer. Figur 3.8 viser dette som betyr at det blir lagt inn "falske" verdier før den observerte alarmen, i en valgt periode før alarmen inntreffer. Dette for å prøve å hinte til modellen at det utløses en alarm med høyere og høyere sannsynlighet før den faktisk inntreffer. Dette er neste steg i figur 3.8, hvor koden for funksjonen er lagt i vedlegg A.1.




**Figur 3.8:** Viser fremgangsmåten på hvordan alarmdata blir behandlet. Alarmene blir først endret fra kategoriserte alarmer til å ikke være kategoriserte. Deretter blir alarmene brukt til å lage en sannsynlighetsutvikling for hver alarm før alarmene blir implementert i datasettet og brukt på modellen.

Dette er en viktig del av oppgaven fordi det er ønskelig å vite tidlig når en feil inntreffer. I denne oppgaven er det testet med 2 og 10 timer før alarm blir utløst. For å se på utviklingen fra tiden før alarmen oppstår til alarmen faktisk oppstår, er det testet med en lineær og en eksponentiell utvikling. Dette er inputs i funksjonen og beskrevet som valg i figur 3.8.

Grunnen til at modellen er testet på en eksponentiell og en lineær graf, er at på denne måten er det mulig å få forskjellige data i datasettet og dermed få muligheten til å se hvordan maskinlæringsmodellen yter best i forhold til hva slags type vekst det er. Deretter blir den nylagde alarmkolonnen implementert i datasettet (linje 4 i figur 3.8). Et utdrag av denne alarmkolonnen er visualisert i figur 3.9. Denne figuren viser utviklingen i sannsynlighet før en alarm oppstår.




**(a)** Grafisk fremstilling for sannsynlighetsutvikling før en alarm oppstår.

| | |
|---|---|
| 2017-04-18 01:10:00 | 0 |
| 2017-04-18 01:20:00 | 0 |
| 2017-04-18 01:30:00 | 7.86845e-63 |
| 2017-04-18 01:40:00 | 6.30512e-16 |
| 2017-04-18 01:50:00 | 3.05902e-07 |
| 2017-04-18 02:00:00 | 0.000335463 |
| 2017-04-18 02:10:00 | 0.00856561 |
| 2017-04-18 02:20:00 | 0.0497871 |
| 2017-04-18 02:30:00 | 0.14388 |
| 2017-04-18 02:40:00 | 0.286505 |
| 2017-04-18 02:50:00 | 0.459426 |
| 2017-04-18 03:00:00 | 0.644036 |
| 2017-04-18 03:10:00 | 0.826891 |
| 2017-04-18 03:18:05 | 1 |
| 2017-04-18 03:20:00 | 0 |

**(b)** Viser de konkrete verdiene til den grafiske fremstillingen.

**Figur 3.9:** Begge figurene viser hvordan sannsynlighetsutviklingen til alarmdata ser ut der en alarm inntreffer. Eksempelet viser en eksponentiell utvikling to timer før en alarm oppstår.

I eksempelet i figur 3.9 har sannsynlighetsutviklingen av alarmkolonnen en utvikling over 2 timer med en eksponentiell vekst. Det er også disse verdiene som er brukt i koden i vedlegg A.1. Til slutt, som beskrevet i siste del av figur 3.8, vil de forskjellige typene datasett sammenlignes slik at det er mulig å forstå hva som er den beste løsningen av de testede metodene.

### 3.6.4 Typer datasett

Uavhengig av hvordan kolonnen som omhandler alarmene, er det i denne oppgaven mulig å velge mellom datasett som omhandler ulik informasjon fra turbinene. Varianter av datasett er nevnt i tabell 3.1 i kapittel 3.6.1. Denne tabellen inneholder korte beskrivelser av de ulike datasettene. Datasettene inneholder ulike parametere, men turbinene er de samme. Dette betyr at datasettene kunne vært satt sammen til ett stort datasett med flere hundre parametere. Dette er ikke hensiktsmessig, og derfor delt opp.

Antall parametere, altså informasjonsmengden, har ulik størrelse i de forskjellige datasettene. Datasettene for Smøla 1 (turbin 1-20) og Smøla 2 (turbin 21-68) er splittet. Grunnen til dette er at innholdet i Smøla 1 er mye mindre omfattende enn i Smøla 2. Det er mange målinger i Smøla 2 som ikke er til stede i Smøla 1 [37] [38].

Typene av datasett gir ulike resultater av maskinlæringen. Det er mulig å lese og forstå seg på datasettene og forstå hvilket datasett som trolig vil fungere for det valgte problemet, men dette må også testes. Datasettet som denne oppgaven i all hovedsak er lagt vekt på er TUR-datasettet. Det er fordi innholdet har flere parametere som omhandler yaw-systemet, og dermed er aktuelt for oppgaven.

### 3.6.5 Valg av modell

Valget av modell faller på en versjon av et nevralt nettverk som kalles tilbakevendene nevralt nettverk (RNN), beskrevet i kapittel 2.5.9. RNN er valgt fordi rekkefølgen på dataene i denne oppgaven er viktig. Arbeidet inneholder tidsserier og dermed har rekkefølgen på tidspunktene stor betydning. Varianten av RNN som er valgt til modellen er LSTM (Long Short Term Memory). Grunnen er at denne varianten unngår eksplodert/forminsket gradient som er beskrevet i teorien (2.5.10).

### 3.6.6 Bygge modellen

Det å velge verdier til blant annet antall lag og antall units/nevroner til en modell innenfor maskinlæring, og i dette tilfelle dyp læring, handler mye om prøving og feiling. Det er ønskelig at modellen skal forstå seg på sammenhenger mellom parametere som er vanskelig å se. Modellen for denne oppgaven har valgte verdier som har en viktig betydning, samtidig inneholder den flere relativt tilfeldige valg.

Figur 3.10 viser en visualisering over oppbygningen av modellen hvor enkel informasjon om hvert lag er presentert. Den første inputverdien, beskrevet som “None”, er *batchsize* (beskrevet i kapittel 2.5.6). Den er “None” fordi den blir bestemt senere idet modellen begynner treningen, som beskrives i kapittel 3.6.8.

Som figur 3.10 viser er inputen i modellen, som er kalt “InputLayer”, en input som har tre dimensjoner (samples/rader, timesteps og features/kolonner) og dette er dermed inputen i første gjemte lag, som er et LSTM-lag (“lstm”). Tallet 96 i første LSTM-lag er antall kolonner/features som treningsdatasettet inneholder. Denne verdien varierer hvis modellen blir brukt på et annet datasett som har et annet antall parametere. Outputen til dette laget er 32, som er antall valgte units i dette LSTM-laget. Én unit inneholder en LSTM-celle, beskrevet i kapittel 2.5.11. Antall units er relativt tilfeldig valgt, men dette antallet kan økes og senkes ettersom hvor generell det er ønskelig at modellen er. En

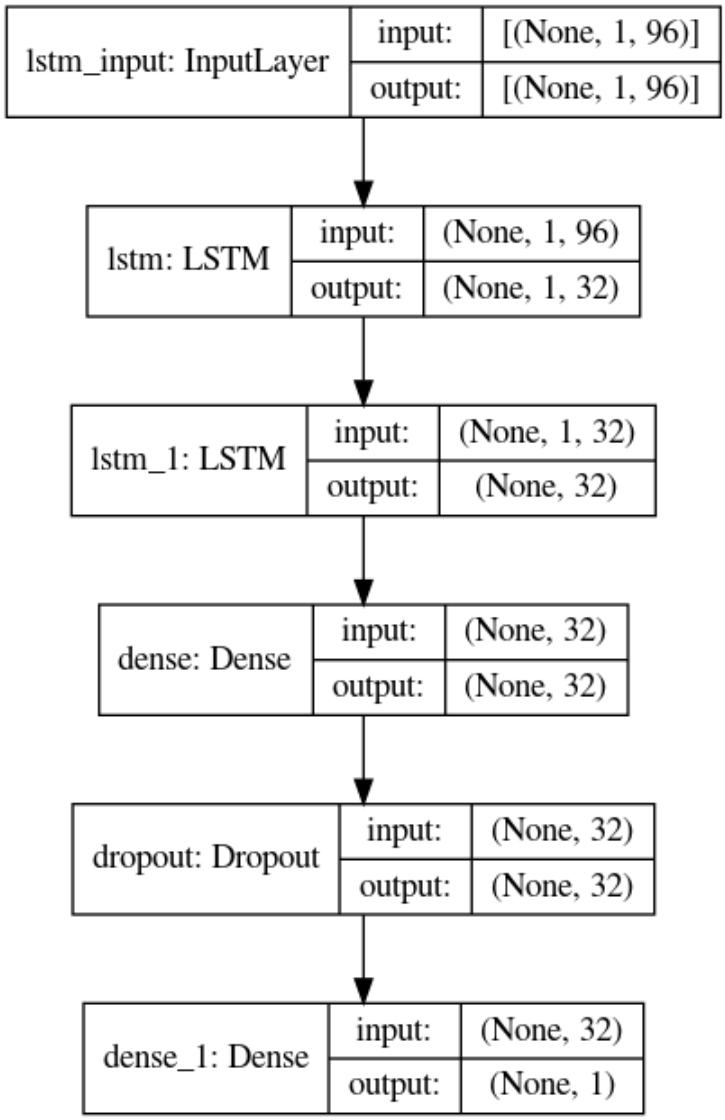
økning i antall units vil gi mer overfitting, og en minking vil gi en mer generell modell (underfitting). Overfitting og underfitting er beskrevet i kapittel 2.5.6.

Den andre av inputverdiene i LSTM-lagene, tallet 1, omhandler antall *timesteps*. Dette betyr hvor mange målinger tilbake i datasettet det er ønskelig å huske fra. Denne modellen husker dermed forrige LSTM-unit, som igjen husker sin forrige unit [4].

Neste lag, "$lstm1$", er også et LSTM-lag med likt antall units, 32. Begrunnelsen for antall units i dette laget er den samme som for forrige lag. At valget på antall LSTM-lag er 2 er også mye av samme grunnen som grunnen til antall units. Det er et relativt tilfeldig valg, men ved en eventuell økning i antall units er det muligheter for overfitting. Generelt betyr dette at ved et større nettverk, blir utregningene mer detaljerte og faren for overfitting øker [5]. I og med at dette laget er et LSTM-lag er det viktig at modellen opprettholder riktig antall dimensjoner, som er 3. Det betyr at det forrige LSTM-laget må returnere modellen slik at den fortsetter å være sekvensiell. Dette vises i koden i vedlegg B.1 ved "return sequences=True".

Neste lag er et *dense-lag* hvor også antall units er 32. Dette dense-laget er todimensjonalt og brukt for å få en mer komplisert og dypere modell. Det er usikkert om dette er et godt valg. Andre valg er fullt mulig, men dense-lag er ofte brukt og er en enkel måte å gå dypere i modellen på. Resultatet kan forbedres, men det kan også føre til overfitting. Neste lag i modellen er et dropout-lag.

I denne modellen er det muligheter for overfitting, som dybden av modellen tilsier. Et *dropout-lag*, beskrevet i kapittel 2.5.8, kan brukes for å unngå overfitting. I dette laget er input og output det samme. Dropoutmengden er valgt til 0,3, vist i vedlegg B.1. Dette betyr at 30 % av nevronene blir ignorert, og modellen blir da mer generell [26]. Til slutt blir et nytt *dense-lag* presentert som er output-laget til modellen. Dette laget gir bare én output fordi det bare er én type verdi det er ønskelig å predikere. Dette gjelder for alle regresjonsproblemer. Hadde det vært et klassifiseringsproblem hadde det vært fler enn én verdi i dette output-laget.




**Figur 3.10:** Viser alle lagene modellen består av i rekkefølge, hvilken verdi input og output er for alle lagene, hva slags dimensjon det er på inputen og hvilken type lag det er.

### 3.6.7 Samle modellen

Etter at modellen har blitt definert må det legges til en tapsfunksjon, en optimaliserer og en verdi som forteller hvor god/dårlig modellen er (metrics). Tapsfunksjonen som er brukt kalles "Mean Square Error". Optimalisereren som brukes er "RMSprop" for å endre strukturen vektene for å unngå dårlig predikering. Svakheten til modellen bedømmes med metricsen, "Mean Absolute Error" [6].

### 3.6.8 Trene modellen

Idet modellen skal trenes må treningsdata og valideringsdata være tilgjengelig. Valideringsdata brukes til å teste hvor god modellen er. Treningsdata valideres mot vali-

deringsdataene, som er plukket ut ved splitting av modellen som forklart i figur 3.7. Valideringsdatasettet brukes for å forstå treffsikkerheten. Dette gjør at bruken av test-data unngås, som senere brukes til å vise hvor god modellen er på ny data etter at modellen er ferdig trent [26].

Rett før treningen bestemmes også *batchsize* og antall *epochs*, beskrevet i kapittel 2.5.6. For hver batchsize trenes modellen. Dette gjentar seg helt til datasettet har blitt gjennomkjørt, altså for hver epoch. Batchsize, i denne modellen, er satt til 32 og antall epochs er 10. 32 er en mye brukt mengde innenfor batchsize, som sier hvor stor mengde av datasettet som trenes. En lav batchsize er mer presis, men ofte unødvendig liten, fordi det er mulig å få den samme treffsikkerheten med en større batchsize. For stor batchsize vil gi en svakhet i generalisering og dermed er muligheten for overfitting til stedet. Antall gjennomkjøringer, beskriver antall treningsrunder modellen har på treningsdatasettet, og er valgt slik at modellen trener helt til den slutter å forbedre seg [31].

### 3.6.9 Evaluerer modellen og predikerer ny data

Modellen blir evaluert på test-data. Det er data som er splittet tidlig, og vist i figur 3.7, som omhandler splitting av data før modellen blir laget. Test-data har ikke blitt brukt tidligere. De ubrukte dataene brukes til å teste hvor godt modellen predikerer ved bruk av ny data [26].

Predikeringen skjer på *X-test* for å prøve å predikere verdiene i *y-test*. Verdiene i y-test er "fasiten" til X-test. Disse testverdiene var en del av det opprinnelige datasettet før det ble splittet, beskrevet ved figur 3.7. Dermed er det ønskelig at de predikerte verdiene som modellen gir fra X-test, skal være så like y-test som mulig. Det er svært interessant å forstå hvor stor forskjell det er på de predikerte verdiene til X-test og verdiene til y-test.

### 3.6.10 Sammenlikning

I vedlegg C.1 vises koden som brukes til å finne verdiene som beskriver hvor god modellen er. Utregningen av verdiene er basert på *Mean Absolute Error*, forklart i kapittel 2.5.5. Verdiene er utregnet ved å se på avviket mellom den predikerte verdien og testverdien. Det regnes ut to forskjellige verdier av *MAE* for hvert datasett. Den ene verdien gjelder gjennomsnittet av avviket for alle predikerte verdier. Den andre verdien er der testverdien er $> 0$. Det betyr det intervallet hvor sannsynlighetutviklingen starter til alarmen oppstår, og gjennomsnittet av disse verdiene. Dette blir returnert av funksjonen i koden vist i vedlegg C.1.

# 4. Resultater og diskusjon

Dette kapittelet omhandler presentasjon av resultater samt diskusjon rundt funn og observasjoner knyttet til modellens prestasjon. Det blir også fremhevet fordeler og ulemper ved de ulike resultatene. Hvorfor modellen oppnår resultatet den gjør blir presentert og diskutert. Diskusjonen er også knyttet opp mot uregelmessigheter og observasjoner samt målinger/resultater som er ureglementerte. Styrker og svakheter som er valgt innenfor databehandling og modellbygging presenteres også. Kapittelet diskuterer muligheter for implementering av modellen i systemer og hvilke løsninger modellen kan bidra med. I tillegg vil en del av dette kapittelet foreslå muligheter til videre arbeid av oppgaven samt videre bruk av funn. Avslutningsvis vil mulige forbedringer og utfordringer til modellen bli presentert.

## 4.1 Oversikt over resultatene

Tabell 4.1 presenterer de ulike datasettene som er testet på modellen, og resultater knyttet til disse. Tabellen viser 8 forskjellige varianter av datasettet som er knyttet til turbin nummer 47, året 2017 og TUR-verdier (som er beskrevet i tabell 3.1 i kapittel 3.6.1). De endringer som er utført i datasettet er kun gjort i alarmkolonnen, resten av verdiene er uendret. De to siste datasettene i tabellen er knyttet til turbin 68 og 14. Verdiene i tabellen som benevnes "Hele datasettet" beskriver avviket mellom den predikere verdien og testverdien for alle predikerte verdier. Verdiene i "Intervall" beskriver avviket, i gjennomsnitt, mellom den predikerte verdien og testverdien for alle punktene hvor testverdien er $> 0$. Det vil si alle intervallene hvor det er en sannsynlighetsutvikling. I tillegg er det lagt inn en grenseverdi ved utregning som betyr at alle predikerte verdier som er under 0,15 er satt til null. Denne utregningen er beskrevet i metoden og koden til dette ligger i vedlegg C.1.

**Tabell 4.1:** Resultater for de ulike datasettene brukt i maskinlæringsmodellen. Datasettene 1-8 omhandler turbin 47, mens 9 og 10 omhandler henholdsvis turbin 68 og 14. Alarmtypene i datasettene inneholder enten yaw-alarmene eller alle alarmene. Intervallets start før alarm beskriver hvor lang tid i forkant av en alarm sannsynlighetsutviklingen begynner. Om det er eksponentiell eller lineær sannsynlighetsutvikling er beskrevet. MAE-verdiene for alarmkolonnen i hele datasettet (testverdier $\geq 0$) og for intervallene (testverdier $> 0$) er også presentert.

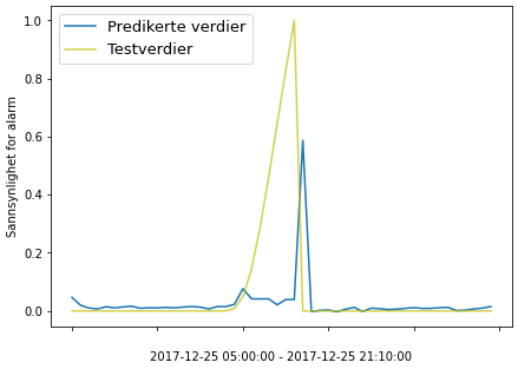
| # | Alarmtype | Intervallets start før alarm | Sannsynlighets-utvikling | Hele datasettet | Intervall |
|---|---|---|---|---|---|
| 1 | Yaw | 2 timer | Eksponentiell | 0,181 | 0,252 |
| 2 | Alle | 2 timer | Eksponentiell | 0,332 | 0,374 |
| 3 | Yaw | 10 timer | Eksponentiell | 0,244 | 0,259 |
| 4 | Alle | 10 timer | Eksponentiell | 0,287 | 0,273 |
| 5 | Yaw | 2 timer | Lineær | 0,534 | 0,542 |
| 6 | Alle | 2 timer | Lineær | 0,289 | 0,285 |
| 7 | Yaw | 10 timer | Lineær | 0,375 | 0,449 |
| 8 | Alle | 10 timer | Lineær | 0,258 | 0,196 |
| 9 | 68 Yaw | 10 timer | Eksponentiell | 0,161 | 0,267 |
| 10 | 14 Yaw | 2 timer | Eksponentiell | 0,0610 | 0,381 |

## 4.2 Forskjellen på yaw-alarmer og alle alarmer

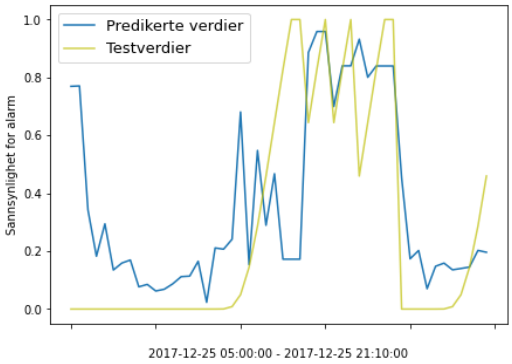
Målet med oppgaven er å finne ut om det er mulig å predikere om en alarm vil oppstå etter en bestemt tid. Den bestemte tiden før en alarm oppstår er i oppgaven satt til 2 timer og 10 timer. Figur 4.1 viser sannsynlighet for at en alarm oppstår. Den gule grafen representerer testverdiene som er plukket ut av datasettet før treningen av modellen og dermed oppfører seg som en fasit til de predikerte verdiene. De predikerte verdiene er den blå grafen. Det optimale resultatet er når grafene har like verdier. Den gule grafen har en eksponentiell sannsynlighetsutvikling over 2 timer som beskrevet i kapittel 3.6.3.

I figur 4.1 er det klippet ut en utvalgt del av datasettet hvor en yaw-alarm oppstår. I figur 4.1a oppstår kun én yaw-alarm, og i figur 4.1b er det flere alarmtyper som oppstår i tillegg til yaw-alarmen. Datasettene er identiske, med unntak av alarmkolonnen. Alarmkolonnen innholder yaw-alarmer i figur 4.1a og alle alarmtyper i figur 4.1b. Ved å fokusere på grafen som omhandler yaw-alarmene er det tydelig at den predikerte verdien har gjort utslag idet alarmen inntreffer. Problemet er at den har svært små, ubetydelige utslag *før* alarmen inntreffer. Figur 4.1a viser er at det er 60% sannsynlighet for at alarmen oppstår i det den faktisk gjør det. Men den har ikke en økende sannsynlighetutvikling, og da er det i prinsippet umulig å forutsi om alarmen oppstår i fremtiden.

Figur 4.1b viser flere alarmer. Her har de predikerte verdiene en mer uforutsigbar struktur. Også her er det vanskelig å se sammenheng mellom når alarmen oppstår og når den predikerte verdien mener at en alarm oppstår. Det er tydelige sammenhenger mellom den blå og gule grafen i figur 4.1b, men den er svært upresis og skaper en usikkerhet. Denne varianten er dermed vanskelig å stole på grunnet denne usikkerheten.




**(a)** Viser hvor godt modellen predikerer for en periode hvor det oppstår én yaw-alarm.




**(b)** Viser hvor godt modellen predikerer for den samme tidsperioden, men med alle typer alarmer.

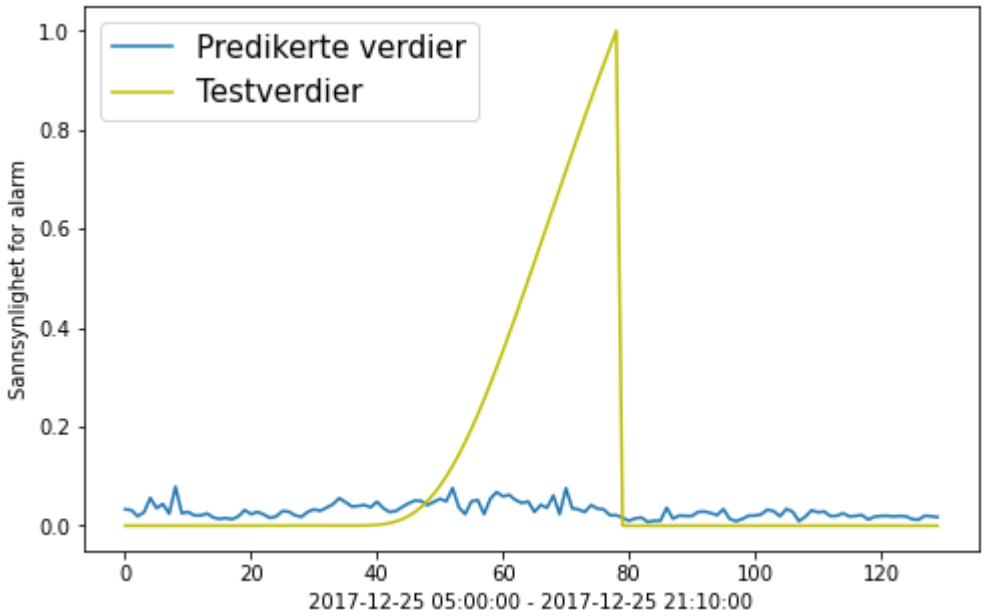
**Figur 4.1:** Figuren viser en utvalgt periode hvor det oppstår alarmer. De ulike fargene representerer testverdiene og de predikerte verdiene for det utvalgte datasettet. Datasettene er helt like utenom alarmene. Det er valgt eksponentiell sannsynlighetsutvikling og intervallets start før alarm er 2 timer for turbin 47.

Disse datasettene omhandler en regresjonsgraf på alarmverdiene som starter 2 timer før alarmen oppstår og har en eksponentiell utvikling samt en total varighet på ett år (2017). Tabell 4.1 viser at avviket mellom den gule og den blå grafen i gjennomsnitt for "Hele datasettet" er 0,181 for yaw-alarmene. For "Intervall", hvor testverdiene ikke er 0, er avviket på 0,252. Dette er måling nummer 1 i tabell 4.1, som tilsier de beste resultatene for turbin 47. Det generelle problemet med resultatene er at alle er relativt svake. Verdiene optimalt sett burde hatt en større tendens til å være mye nærmere 0. En god verdi for "Hele datasettet" tilsier at den treffer godt når den er nær 0. Svakheten til den første målingen er hovedsakelig i "Intervall". Dette er problematisk fordi her er det viktig å predikere en lav verdi siden det er her alarmen oppstår.

Verdiene for datasettet som inneholder alle alarmene er svakere enn for datasettet som inneholder yaw-alarmene. Årsaken til dette er i hovedsak flere alarmer og dermed mer støy. Det er også usikkerhet og støy når det oppstår alarmer som ikke nødvendigvis har en konkret sammenheng. Det gjelder alarmer for andre områder av turbinen. Mulig svakhet ved datasettet er at det inneholder alarmer som ikke nødvendigvis har med hverandre å gjøre. Ulike alarmer har sammenheng med ulike parametere. Dette blir et problem fordi datasettet beskriver alle alarmene som like.

## 4.3 TMP-resultater

Figur 4.2 viser prestasjonen til modellen for en turbin med den samme databehandlingen som er gjort på figur 4.1a, nå med et annet datasett. Datasettet omhandler TMP-verdier i stedet for TUR-verdier hvor forskjellen er beskrevet i kapittel 3.6.1 ved tabell 3.1. Denne tabellen beskriver de ulike datasettene. Figur 4.2 viser at modellen ikke er egnet til å predikere tidspunktene alarmene inntreffer på. Det er tydelig ved at den blå grafen ikke viser den ønskede endringen i sannsynlighet for alarm i den perioden testverdiene forteller at en alarm oppstår.



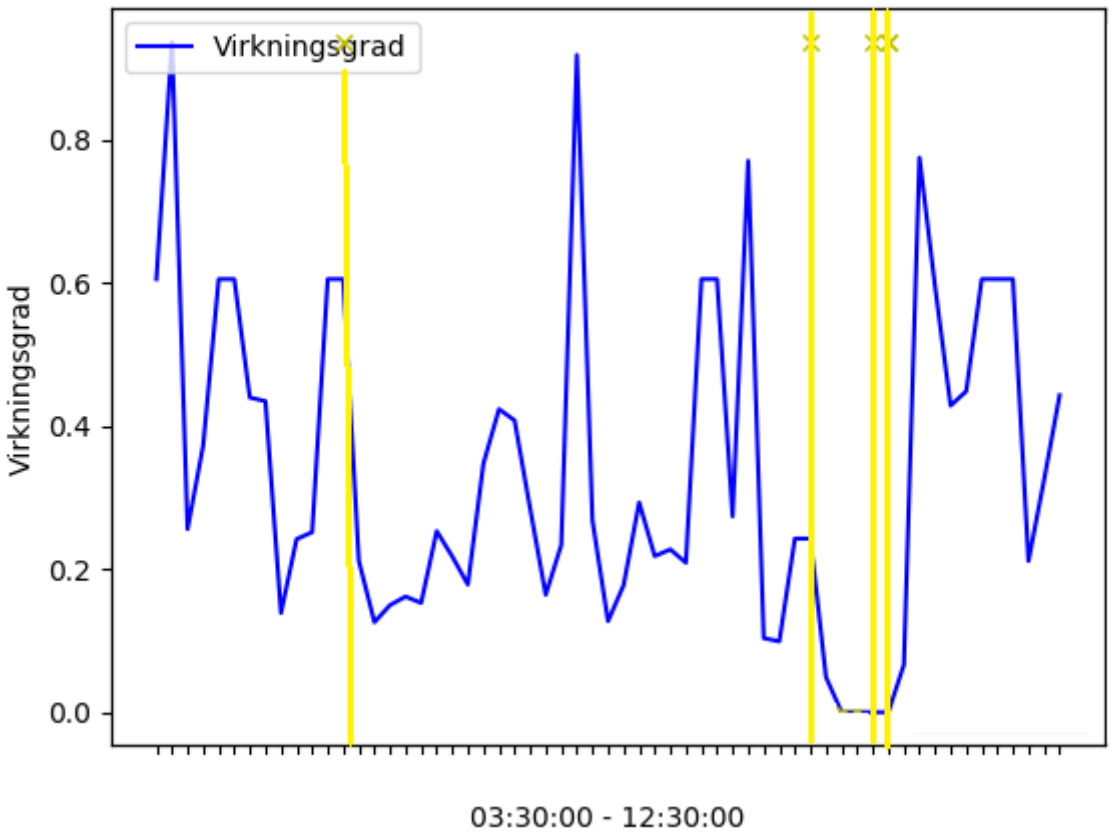

**Figur 4.2:** Viser hvor godt modellen predikerer yaw-alarmer for TMP-datasett for samme tidsperiode og sannsynlighetsutvikling som figur 4.1.

## 4.4 Treghet

Tregheten i systemet gjør at forståelsen og prosesseringen av data blir mer komplisert. Det er ønskelig å unngå verdiene som vist i figur 3.5. Figur 3.5 er plassert i kapittel 3.5 og forklarer hvilke punkter som er ønskelig å unngå slik at det er mulig å effektivisere turbinen. På grunn av treghet knyttet til vindhastighet og levert effekt vil det være en forskyvning i disse målingene. Virkningsgraden vil dermed påvirkes og vise usannsynlig lav og høy virkningsgrad som vist i figur 4.3. Denne figuren viser virkningsgraden til turbinen (turbin 14, som er turbinen som er blitt brukt som eksempel i kapittel 3) ved ulike tidspunkt over en kort periode. Alarmene som oppstår er også vist i figuren. Dette er den samme turbinen som er beskrevet i kapittel 3.5 ved figur 3.5.

I figur 4.3 er det vist at idet det oppstår en feil, vil virkningsgraden på turbinen bli meget

lav. Virkningsgraden til turbinen er regnet ut ved likning 2.2 og 2.3 som er beskrevet i kapittel 2.3. Figur 4.3 viser denne utregningen for en utvalgt periode på 9 timer. At virkningsgraden av og til er meget høy er trolig fordi turbinene opplever treghet. Dette er et problem for forståelsen av hvor effektiv turbinene er på et bestemt tidspunkt.

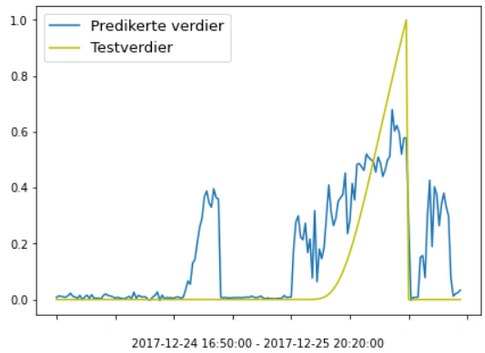



**Figur 4.3:** Viser virkningsgraden til turbin 14 over en periode på 9 timer den 3. januar 2017. De loddrette, gule linjene viser når det oppstår en alarm.

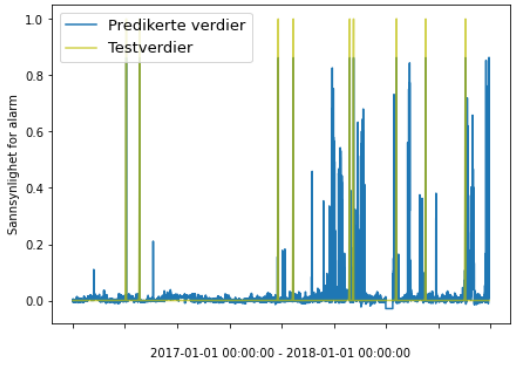
En løsning for å unngå problemet med treghet er å tilpasse kurven for vindhastighet til effektkurven, og ikke motsatt. De fleste parametere omhandler det maskinelle på turbinen, og det gjør også levert effekt. Derfor er det bedre å flytte verdiene på vindhastigheten. Samtidig er ikke dette et stort problem i en maskinlæringsmodell. Den vil etter trening forstå mønsteret som omhandler treghet, men det kan samtidig ha andre påvirkninger. Maskinlæringsmodellen må dermed forstå at det er en treghet mellom vind og effekt, men dette gjelder trolig flere parametere uten at det er like tydelig. Dette er også en metode for en mulig utvikling av oppgaven, som omhandler forståelsen og bruken av de forskjellige parameterne.

## 4.5 Modellen i korte intervaller

Figur 4.4 viser et resultat av modellen ved yaw-alarmer og utviklingen av sannsynligheten for at en alarm oppstår. Figur 4.4a viser en utvalgt periode på litt over ett døgn, mens figur 4.4b viser det samme resultatet, men visualisert for et helt år.




**(a)** Figuren viser resultatet for datasettet som inneholder yaw-alarmer med 10 timer eksponentiell sannsynlighetsutvikling. Visualisert for litt over et døgn.




**(b)** Figuren viser resultatet for datasettet som inneholder yaw-alarmer med 10 timer eksponentiell sannsynlighetsutvikling. Visualisert for et år.

**Figur 4.4:** Datasettene og resultatet på de forskjellige figurene er like, men de er plottet for en ulik tidsperiode. Figur a er en utvalgt del av figur b.

Figur 4.4a gir et inntrykk av at det er en modell som finner sammenhenger mellom parametere og alarmkoder i datasettet. Det er tendenser som tyder på at modellen kan brukes til å predikere alarmer før de inntreffer. Samtidig viser figur 4.4b at modellen finner sammenhenger flere steder i datasettet der en alarm oppstår. Dermed klarer modellen å predikere når en alarm oppstår. Utfordringen er at i tillegg til å vise at en alarm oppstår, viser den predikerte grafen at det oppstår alarmer også når det ikke er grunnlag for det. Dette er en stor svakhet da det blir vanskelig å stole på modellen. Resultatene til 4.4 blir også vist i tabell 4.1, hvor det er beskrevet to verdier som omhandler datasettet som er brukt i figuren over. Tabellverdi nummer 3 i tabell 4.1 viser resultatene for figur 4.4.

Resultatet i tabellen viser at avviket har en MAE-verdi på 0,244 over “Hele datasettet”. At det er få feil i datasettet er en utfordring for å få et godt resultat ved hjelp av en maskinlæringsmodell. Predikeringen i dette tilfelle er i gjennomsnitt nesten 25% feil, som er svært høyt. Dette gjør at modellen igjen har store problemer med å vise troverdighet. I kolonnen “Intervall” i tabell 4.1 vises den samme verdien, men kun for verdiene som omhandler intervallene. Her blir verdien målt til 0,259, som ikke er en veldig god verdi, men det betyr også at modellen ikke går glipp av at det oppstår en feil. Modellen sliter med å formidle nøyaktig når feilen oppstår. Ved mer data og mer trening av modellen, vil resultatet trolig bli forbedret.

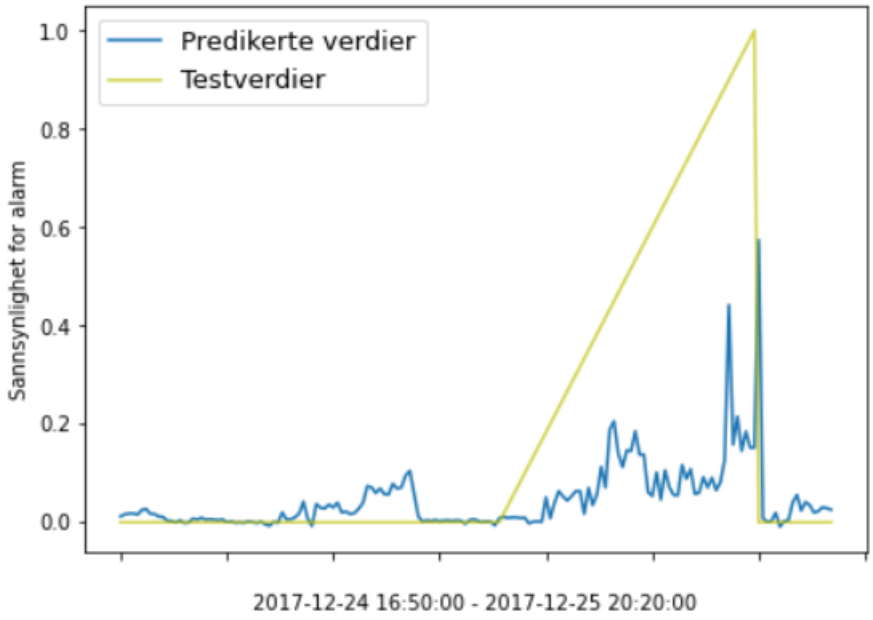
## 4.6 2 eller 10 timer før alarm

I figur 4.1a og figur 4.4a blir den samme yaw-alarmen presentert. Forskjellen på figurene er at den ene starter regresjonsmodellen, som omhandler sannsynlighetutvikling, 2 timer før og den andre starter 10 timer før. Det vil alltid være ønskelig å vite så tidlig som mulig når alarmen inntreffer. På de to nevnte figurene ser det ut til at det er et bedre resultat på grafen som viser 10 timer. Det ville også vært det beste med tanke på hvor god tid som er nødvendig for å rette opp systemet slik at feilen ikke oppstår. Samtidig, som tidligere nevnt, er dette bare et utdrag av resultatet. Tabell 4.1 viser det motsatte. Her vises resultatet for at datasettet som begynner predikeringen 2 timer før er bedre enn datasettet som predikerer 10 timer før. Dette er også mer sannsynlig for det vil være enklere å observere en alarm i en kortere tidsperiode før alarmen inntreffer enn lang tid i forveien. Parameterne gir trolig mer utslag mot at en alarm oppstår nærmere alarmtidspunktet enn lenge før.

## 4.7 Lineær eller eksponentiell sannsynlighetsutvikling før alarm

Ved å teste en lineær sannsynlighetsutvikling, som vist i figur 4.5, vil datasettet ha andre verdier å predikere mot i forhold til eksponentiell sannsynlighetsutvikling. Ved å se på figuren vil modellen tildels predikere verdier som har en økende sannsynlighet for at en alarm skal inntreffe. Denne utviklingen har en lignende utvikling som figur 4.4, men ikke like god. Dette fremheves også ved resultatet nummer 3 og 7 i tabell 4.1 hvor den eneste forskjellen på datasettene er at sannsynlighetsutviklingen for alarmene er lineær eller eksponentiell.

Resultatene viser at modellen er bedre for den eksponentielle utviklingen. En av flere grunner til dette kan være at modellen finner sammenhenger mellom parametere og alarmer på ulike måter. Ved eksponentiell utvikling betyr det at sannsynligheten for en alarm øker mer og mer helt til alarmen oppstår. Ved å *ikke* bruke sannsynlighetsutvikling ligner grafen mer på en eksponentiell graf enn en lineær graf ved at verdien går fra 0 til 1 momentant. Dette kan være grunnen til at den eksponentielle grafen passer best til modellen. Samtidig er det trolig flere parametersammenhenger i en lineær graf som muligens ikke hadde gitt utslag i en eksponentiell graf. Den eksponentielle utviklingen gir best resultat på dette datasettet, men på et annet datasett er det fullt mulig at den lineære sannsynlighetsutviklingen kunne fått et bedre resultat.




**Figur 4.5:** Viser et utdrag av de predikerte verdiene og testverdiene for datasettet som inneholder lineær sannsynlighetutvikling over 10 timer for yaw-alarmer.

## 4.8 Yaw-alarmer eller alle type alarmer

Den viktigste delen av oppgaven er å fokusere på yaw-systemet. Ved å sammenligne yaw-alarmene opp mot alle alarmene, vil det være mulig å forstå ulikhetene. Resultatene i tabell 4.1 tilsier at modellen predikerer et bedre resultat ved lineær sannsynlighetsutvikling i datasettet for alle alarmene, enn ved å kun se på alarmene som omhandler yaw-systemet. Dette gjelder alle utregningene i datasettene som innholder alle alarmtypene. Det vil si resultatnumrene 5-8 i tabell 4.1. Modellen har predikert bedre for datasettene som inneholder alle alarmtyper med en lineær sannsynlighetsutvikling enn for en eksponentiell utvikling.

For datasettene som inneholder en eksponentiell sannsynlighetsutvikling er det motsatt tendens. Her blir modellen bedre på datasettene som inneholder yaw-alarmene enn på alarmene som inneholder alle alarmtypene. Dette gjelder resultatnumrene 1-4 i tabell 4.1.

Hovedårsaken til at dette er tilfelle er vanskelig å fastslå, men det tilsier viktigheten av antall alarmer. Hvor stor andel data som er til stede, og som det er mulig å predikere mot, har stor betydning for resultatet. Samtidig gir det en sterk indikasjon på hva slags type datasett det er ønskelig å bruke de to variantene av sannsynlighetsutvikling på. I tillegg er det tydelig at resultatene i tabell 4.1 beskriver eksponentiell graf som best uavhengig av type datasett. Med den begrunnelsen er det enklest å anbefale datasett med eksponentiell sannsynlighetutvikling på et generelt grunnlag.

## 4.9 Få alarmer

Ved gjennomkjøring av et datasett i en maskinlæringsmodell er antall alarmer viktig for resultatene. En stor svakhet i denne oppgaven er at datasettene inneholder for få yaw-alarmer. Tabell 4.2 viser hvor mange alarmer de forskjellige datasettene inneholder. Tabellen viser også at i forhold til antall alarmer totalt er det veldig få alarmer som omhandler yaw-systemet. Det er 46 yaw-alarmer på turbin 47 og dette utgjør kun 0,25 % av alle alarmene til denne turbinen. Få alarmer betyr også at det er få verdier det er mulig for modellen å trene mot. Modellen klarer å se sammenhenger, men sammenhengene mellom alarmene og parameterene oppstår skjeldent. Dermed blir det for lite informasjon til at modellen klarer å forbedre seg ved trening. Modellen fokuserer like mye på hver måling som betyr at den fokuserer flere ganger på målinger som ikke tilsier at det oppstår en alarm enn tilfeller hvor det oppstår en alarm. Dette betyr igjen at resultatet for modellen med fokus på når en alarm oppstår blir svak.

**Tabell 4.2:** Viser antall yaw-alarmer og antall alarmer totalt for 3 ulike turbiner

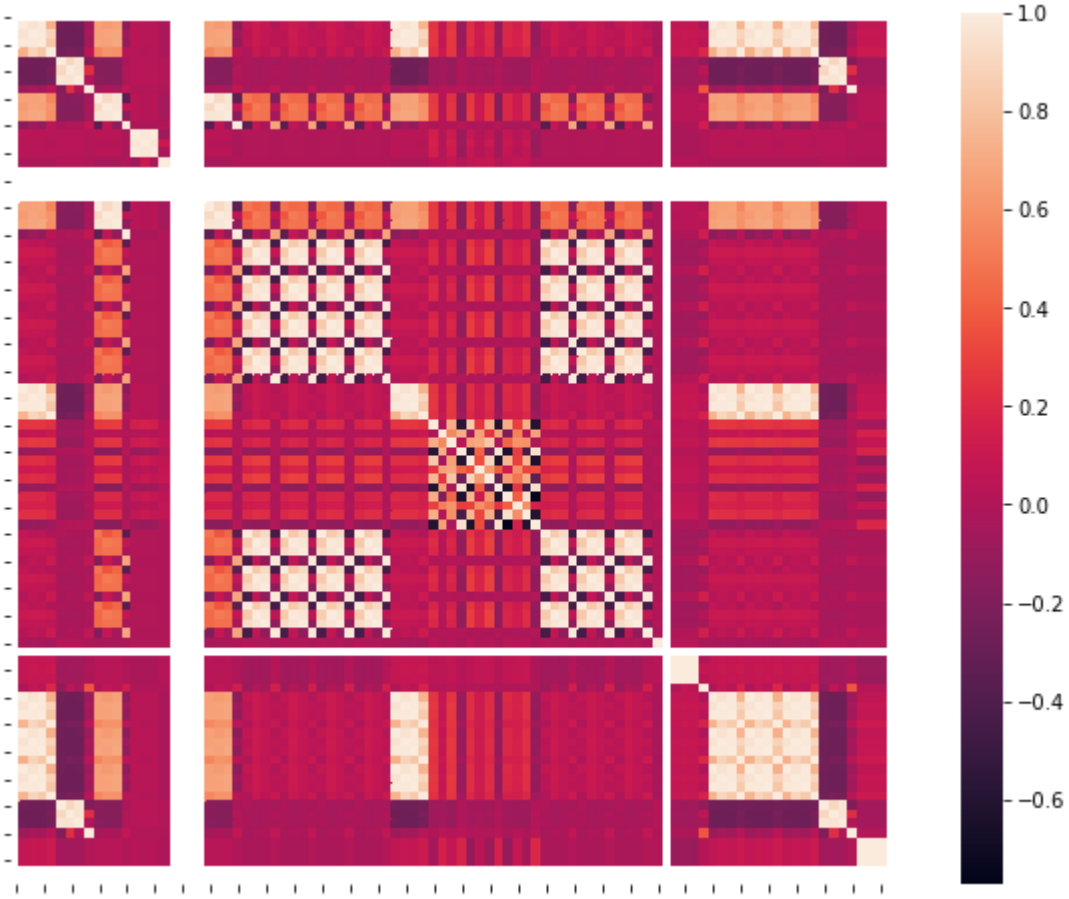
| Turbinnummer | Antall yaw alarmer | Antall alarmer totalt |
|---|---|---|
| **Nummer 14** | 69 | 820 |
| **Nummer 47** | 46 | 18640 |
| **Nummer 68** | 36 | 16641 |

En mulig løsning på denne utfordringen er å fjerne lange dataserier som ikke inneholder spesifiserte alarmer. Dette gjelder hovedsaklig yaw-systemet hvor det få alarmer. Modellen vil da fokusere på tidspunktene hvor en alarm oppstår og har et mindre fokus på tidspunktene hvor det ikke oppstår en alarm. Det betyr at den fokuserer mer på testverdienes intervaller som omhandler sannsynlighetsutvkling enn der testverdiene er 0. En annen fordel med denne fjerningen av data er at treningen på datasettet bruker mindre datakraft og tar kortere tid per gjennomkjøring.

Denne metoden motsier teknikken om å beholde mest mulig data som er ønskelig i en maskinlæringsmodell. Det hadde vært bedre for resultatet om datasettet inneholdt flere feil. Dette er selvfølgelig ikke ønskelig for Smøla vindpark, men ved flere feil er det enklere å predikere nye feil ved bruk av maskinlæring. Derfor er det viktig å tilpasse data etter behov. Dette gjelder også ved fjerning av parametere som også er en mulighet for å forbedre modellen.

Figur 4.6 viser en korrelasjonsmatrise over parameterne i TUR-datasettet. I dette datasettet, som er brukt mest i oppgaven, inneholder 96 forskjellige parametere. Dette er også antallet som er vist i korrelasjonsmatrisa. Matrisa består av mange små kvadrater som representerer sammenheng mellom de forskjellige parameterne. Parameterne har en

perfekt positiv korrelasjon når fargen tilsvarer verdien 1, helt hvit. Helt svart, altså -1, betyr at parameterne har en perfekt negativ korrelasjon. Hvis verdien tilsvarer 0, har parameterne ingen korrelasjon [26].




**Figur 4.6:** En korrelasjonsmatrise for parametere i TUR-datasettet. Lysere farge betyr mer positiv korrelasjon. Mørkere farge betyr mer negativ korrelasjon. Midt mellom, betyr ingen korrelasjon.

Det å analysere hver enkelt parameter og forstå om den bidrar positivt eller negativt, kan forbedre resultatet til den valgte maskinlæringsmodellen. Dette er en lang og analytisk prosess, men modellen vil trolig oppnå en forbedring ved å fokusere mer på informasjonen som er tilpasset problemet, og unngå parametere som kan villede.

*Feature engineering*, som er beskrevet i kapittel 2.5.3, vil også være en god teknikk for å bruke de riktige parameterne i datasettet. Denne teknikken kan brukes til å utføre valg på hvilke parametere som skal brukes, som muligens vil forbedre modellen. Denne teknikken sammen med fokusering på hvor godt parameterne korrelerer vil være en interessant utvikling av oppgaven.

En annen mulig videreføring av oppgaven er å fremskaffe data over en lengre tidsperiode enn ett år. Ved tidsserier over lang tid oppstår det flere feil, og modellen vil kunne forbedres. Parametere som finner sammenhenger med alarmer over lange tidsserier vil trolig gi modellen et bedre resultat. Optimaliserte parametere som brukes for å finne bestemte alarmer vil gi den samme gevinsten.

Tabell 4.2 viser antall alarmer totalt for turbin 47. Her er det mange alarmer, men det er alarmer som omhandler ulike problemer. Hva slags type alarm det er blir ikke presisert i datasettene. På denne måten kan modellen oppleve uenigheter mellom hvilke parametere som tilsier at en alarm inntreffer. Uenigheter gjør at det oppstår usikkerhet og predikeringen kan bli svak. Å se på problemet som et kategoriseringsproblem og ikke et regresjonsproblem, vil være en god mulighet til å optimalisere resultatet til modellen. Klassifisering er beskrevet i kapittel 2.5.4. Dette gjør det enklere å si hvilken type alarm som oppstår når det oppstår en alarm. Flere ulike alarmtyper gjør arbeidet med datasettene mer omfattene. En mulighet er å bruke regresjonsmodellen på hver alarmtype. Dette er også en mulig forbedring av resultatet og en interessant utvikling av oppgaven.

## 4.10 Smøla 1 eller Smøla 2

Tabell 4.2 viser antall alarmer per turbin. Denne tabellen viser også forskjellene på Smøla 1 og Smøla 2. Turbin 14 er en del av Smøla 1. Turbin 47 og 68 er en del av Smøla 2. Som tabellen indikerer er det rundt 20 ganger så mange alarmer totalt på Smøla 2 enn på Smøla 1. Målingene på Smøla 2 er mer omfattende. Det er flere deler av turbinsystemet som har alarmsystemer på Smøla 2 [38], enn på Smøla 1 [37]. Ved at det er flere alarmer er det mulig å finne flere ulike feil i systemet. Samtidig kan det ha en negativ påvirkning ved at viktigheten og nødvendigheten av alarmmengden reduseres. På den måten er det vanskeligere å forstå sammenhengen mellom parametere og alarmene for en maskinlæringsmodell. Dette er også en god grunn til stille spørsmålet om oppgaven burde vært rettet mot kategoriserte data og dermed en kategorisert modell, i stedet for regresjon.

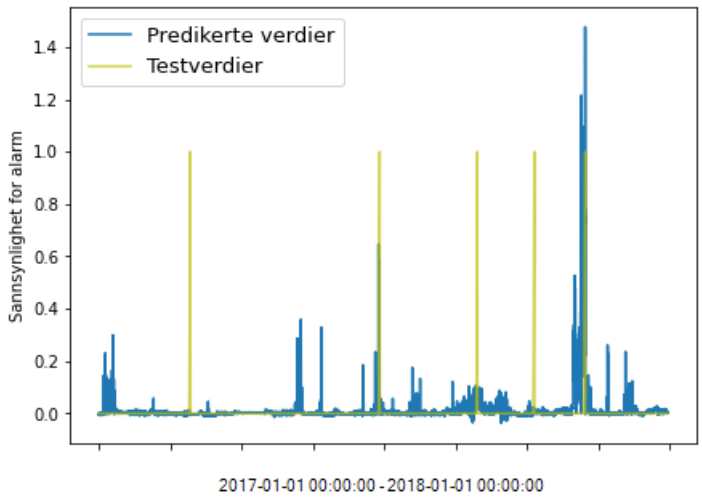
I tillegg viser tabell 4.2 at det er nesten dobbelt så mange yaw-alarmer på turbin nummer 14 i forhold til turbinene på Smøla 2. Med den observasjonen er det naturlig å tro at turbin 14 ville gjort det bedre i maskinlæringsmodellen. Det er dessverre ikke tilfelle. På grunn antall parametere er det nærliggende å anta at alle datasettene som omhandler turbinene på Smøla 1 gir et dårligere resultat. Dette er vist i begynnelsen av dette kapittelet, ved tabell 4.1 som omhandler de forskjellige datasettenes prestasjon. Resultatnummer 10 i denne tabellen viser turbin 14 på Smøla 1. I kolonnen “Hele datasettet” gjør modellen en veldig god predikering. Verdien for feilen i dette tilfelle er på kun 0,0610

som er et veldig godt resultat. Problemet oppstår etter analyse rundt dette resultatet og ved å se på verdien for hvor godt resultat det er for "Intervall". Dette resultatet er mye svakere og derfor er det med stor sannsynlighet en modell som predikerer 0 hele tiden. Det er riktig i store deler, men ingen verdi for løsningen av oppgaven.

## 4.11 Turbin 68

Tabell 4.2 viser at turbin 47 og turbin 68 har relativt like verdier. Turbin 68 er plassert litt i utkanten av vindparken, som vist i figur 3.1b, som beskriver plasseringen av turbinene på Smøla. Samtidig har turbin 68 færre alarmmålinger for yaw-systemet enn turbin nummer 47, som vist i tabell 4.2. I figur 4.7 og figur 4.4b, som er beskrevet i delkapittel 4.5, er samme type datasett med to ulike turbiner presentert. I tabell 4.1 vises resultater for datasettene til turbin 47 og turbin 68 som er henholdsvis resultatnummer 3 og 9. I kolonnen "Hele datasettet" for resultatnummer 9 vises en lav verdi som betyr at de predikerte verdiene treffer svært godt ved sammenligning med testverdiene.

For kolonnen "Intervall" derimot er det bedre resultat for turbin nummer 47 enn for turbin nummer 68. En mulig forklaring kan være ved å se på tabell 4.2 som tilsier at det er flere alarmer for turbin 47 enn for turbin 68. Ved at det er færre alarmer vil de predikerte verdiene oftere være nærmere null enn ved flere alarmer. Verdiene som skal være null er enklere for modellen å predikere enn verdiene som ikke skal bli null. Dette betyr også at for turbin 47, som inneholder flere alarmer, vil det være enklere å predikere riktig i intervallene fordi da har modellen flere alarmer å trene mot. På en annen side er det trolig flere underliggende grunner til at prediksjonen blir som den blir i de forskjellige datasettene. Spesielt når verdiene for intervallene hos de to forskjellige turbinene er meget like.




**Figur 4.7:** Viser testverdiene og de predikerte verdiene for turbin 68 med en 10 timer, eksponentiell sannsynlighetsutvikling for yaw-alarmer.

## 4.12 Brukbare resultater

Bruksområdene til funn og resultater i denne oppgaven er todelt. I deler av oppgaven hviler en stor usikkerhet rundt implementering av nye datasett i modellen. Samtidig er deler av resultatene svært opplysende: Som forrige delkapittel beskriver er det små ulikheter i turbinenes resultater. Ved å utvikle en modell som predikerer alarmene i de ulike turbinene gir det relativt like resultater. Resultatene er relativt svake, men det er likheter blant turbinene og derfor er det mulig at modellen kan passe til alle datasettene for alle turbinene. Det viktige er at verdiene i datasettene er relativt like. Dette er sannsynlig siden turbinene er plassert i samme vindpark, er av samme type og har stått like lenge. Unntaket er at turbinene på Smøla 1 og turbinene på Smøla 2 er forskjellige. Dermed er det nødvendig at en modell blir tilpasset Smøla 1 og en annen blir tilpasset Smøla 2, men innad i de forskjellige Smøla-variantene er det trolig kun nødvendig med én modell.

## 4.13 Modellens muligheter

Modellen som blir bygd i kapittel 3.6.6 er ikke nødvendigvis den optimale modellen. Det er fullt mulig å endre på forskjellige modellparametere. Valgene omhandler antall gjemte lag, hvilke type lag, antall units i de forskjellige lagene, type tapsfunksjon og så videre. Det er mange muligheter, og det gjør at denne oppgaven er det mulig å bygge videre på. Testing av nye varianter og andre typer teknikker er en viktig del av et eventuelt videre arbeid. Størrelsen og kompleksiteten av modellen kan også vurderes. Maskinlæring blir aldri perfekt, men det kan ofte bli godt nok i mange sammenhenger. Det varierer hvor viktig et nøyaktig resultat er mellom ulike problemstillinger.

## 4.14 Fordeler ved et bedre resultat

Om det hadde vært mulig å oppnå et bedre resultat ved bruk av modellen, ville det åpnet seg muligheter for effektivisering av vindparken. Ved å implementere en fungerende modell i systemene som omhandler vedlikehold og optimalisering, vil det være mulig å opprettholde en produksjon nesten uten stans. I tillegg vil det være mer økonomisk lønnsomt ved å produsere mer elektrisk energi. Det er også muligheter for å spare kostnader ved å unngå feil og dermed unngå mulige ødeleggelser på systemet. Kostnadene knyttet til feil ved yaw-systemet på Smøla var på 172.319 kroner i 2017 [35]. Dette er utfordrende fordi det er en stor sum per år for bare én liten del av systemet.

Det vil ha en positiv klimaeffekt ved å opprettholde produksjonen/effektivisere vindparken fordi mer fornybar energi blir produsert. I tillegg til å øke grønn kraftproduksjon er nye naturinngrep mindre nødvendig siden effektivisering vil øke totalproduksjonen i vindparken. Ved å effektivisere og unngå utbygging er det enklere å opprettholde en mer stabil leveringssikkerhet, og dermed blir det enklere å dimensjonere for kraftnettet. I fremtiden vil det være helt nødvendig å bygge ut kraftnettet, fordi befolkningen bruker mer og mer elektrisk energi. Et steg på veien for å unngå *unødvendig* utbygging av kraftsystemet og kraftverk er å effektivisere de systemene som allerede er bygd, og optimalisere de systemene som er planlagt for fremtiden.

# 5. Konklusjon

For å oppnå klimamålene raskere og forbedre lønnsomheten med fornybar energi i forhold til fossile energikilder, er en løsning å effektivisere de energikildene vi allerede har installert. Ved å unngå feil i systemet til en vindturbin, er det mulig å opprettholde produksjonen over lengre tid. I tillegg er det økonomiske fordeler ved å unngå skader på systemet. Yaw-systemet har vært en økonomisk utfording ved at dette systemet utgjør store deler av feil med høy reparasjonskostnad.

Resultatene viser at predikering av feil en tid før den oppstår er vanskelig. Resultatene som omhandler yaw-systemet har rundt 30 % feilprediksjon i gjennomsnitt for turbinen. Dette tilsier at det er vanskelig å stole på modellens prediksjoner. Dette igjen betyr at det vil være et stort pålitelighetsproblem ved å implementere denne modellen i andre systemer. Det samme problemet er relativt likt idet modellen testes for alle typer alarmer og ikke bare alarmer tilknyttet yaw-systemet.

En økning i antall yaw-alarmer hadde trolig gjort resultatene bedre. Lengre tidsserier og tilpasset dataprosessering slik at modellen får mer og bedre data å jobbe med, hadde trolig forbedret modellen. Samtidig er det mange andre faktorer som er viktige ved å optimalisere en maskinlæringsmodell. For resultatene som omhandler alle alarmer er ikke antall alarmer nødvendigvis problemet. Trolig er grunnen til at dette resultatet også blir svakt, er at det er stor forskjell på alarmtypene slik at modellen ikke forstår sammenhengen. Det er vanskelig å si dette med sikkerhet siden det er, i en maskinlæringsmodell, vanskelig å tyde hvilke sammenhenger i datasettet modellen fokuserer på.

Om valget av en LSTM-modell er det riktige, er ikke helt sikkert. Etter en teoretisk begrunnelse og nødvendigheten av en modell som predikerer verdier hvor rekkefølge er viktig, vil LSTM-modell være et godt valg. Resultatene viser at modellen er relativt god, men at svakheten i resultatene skyldes at datasettet inneholder for få yaw-alarmer og for korte dataserier. Dette er to av utfordringene som gir svake resultater i oppgaven.

Det er forskjell på turbinene på Smøla 1 og Smøla 2. Siden det ikke er den samme turbintypen gir det utfordringer fordi det er en ulikhet i antall parametere i datasettene. Et av funnene ved oppgaven er at det ikke er en betydelig forskjell på resultatene for

turbin 47 og turbin 68 (som begge er en del av Smøla 2). Ved å utvikle oppgaven eller ved et forsøk på å forbedre modellen, er dette et funn som gjør denne prosessen enklere. Ved at det ikke har en stor påvirkning hvilken turbin som modellen bruker, vil det bety at det er mulig å lage en generell modell som kan brukes på turbinene.

## 5.1 Videre arbeid

Utviklingen av oppgaven har flere muligheter. De fleste mulighetene omhandler utvikling av maskinlæringsmodellen samt dypere behandling av datasettene. Kombinasjonen av dette er også viktig. Ved å endre datasettet slik at hovedfokuset er på hva slags type alarm det er, vil det være naturlig å bruke en maskinlæringsmodell som er tilpasset et klassifiseringsproblem.

En annen metode er å prosessere antall parametere som brukes i datasettet. Ved å lære om bakgrunnsinformasjon og forstå hvilke parametere som kan være gode til å løse det valgte problemet, kan dette forbedre resultatene. Dette gjelder spesielt ved yaw-systemtet. I tillegg er det mulig å fjerne rader hvor det er lang tid mellom hver alarm. Dette passer best på yaw-systemet hvor det oppstår få alarmer. På denne måten vil modellen fokusere mer på når alarmene oppstår og ikke like mye på tiden hvor det ikke oppstår alarm. Disse teknikkene er tidkrevende og krever god analyse og bakgrunnskunnskap. Dette er en god utvikling av oppgaven og forhåpentligvis kan dette ende med et bedre resultat og muligheten til å implementere modellen i andre systemer.

# Referanser


[1] I. Automation. *What is SCADA? Supervisory Control and Data Acquisition. (Hentet: 2021-05-22)*. 2018. URL: https://www.inductiveautomation.com/resources/article/what-is-scada.

[2] P. L. Bartlett, P. M. Long, G. Lugosi og A. Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences of the United States of America* 117 (48) (2020). DOI: 10.1073/pnas.1907378117.

[3] J. Browlee. *A Gentle Introduction to Exploding Gradients in Neural Networks. (Hentet: 2021-05-09)*. 2017. URL: https://machinelearningmastery.com/exploding-gradients-in-neural-networks/.

[4] J. Browlee. *How to Reshape Input Data for Long Short-Term Memory Networks in Keras. (Hentet:2021-05-26)*. 2017. URL: https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/.

[5] J. Browlee. *How to Avoid Overfitting in Deep Learning Neural Networks. (Hentet: 2021-05-26)*. 2018. URL: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/.

[6] J. Browlee. *Your First Deep Learning Project in Python with Keras Step-By-Step. (Hentet: 2021-05-26)*. 2019. URL: https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/.

[7] M. De Lellis, R. Reginatto, R. Saraiva og A. Trofino. The Betz limit applied to Airborne Wind Energy. *Renewable Energy* 127 (nov. 2018): 32–40. DOI: 10.1016/j.renene.2018.04.034.

[8] E. Dufourq og B. A. Bassett. Automated problem identification. I: *Proceedings of the South African Institute of Computer Scientists and Information Technologists on - SAICSIT '17*. New York, New York, USA: ACM Press, 2017: 1–9. DOI: 10.1145/3129416.3129429.

[9] EnergiFaktaNorge. *Kraftproduksjon. (Hentet: 2021-05-14)*. 2021. URL: https://energifaktanorge.no/norsk-energiforsyning/kraftforsyningen/.

[10] Energy fundamentals. *Physics of Wind Turbines. (Hentet: 2021-05-14)*. 2020. URL: https://home.uni-leipzig.de/energy/energy-fundamentals/15.htm.

[11] FN. *Parisavtalen. (Hentet: 2021-05-14)*. 2020. URL: https://www.fn.no/om-fn/avtaler/miljoe-og-klima/parisavtalen.

[12] Fornybar.no. *Vindturbin illustrasjon. (Hentet:2021-05-07)*. 2021. URL: https://www.fornybar.no/upload_images/88877C728C60480CA68F721EBBEB51AF.jpg.

[13] Google Maps. *Smøla vindpark kart*. 2021. URL: https://www.google.no/maps/place/Sm%C3%B8la+Vindpark/@61.7827117,8.1223382,6.71z/data=!4m5!3m4!1s0x46123311bbd61679:0x7122a880d608f27d!8m2!3d63.405134!4d7.913827.

[14] S. Gunaratne. *Why We Need Bias in Machine Learning Algorithms. (Hentet: 2021-05-09)*. 2020. URL: https://towardsdatascience.com/why-we-need-bias-in-machine-learning-algorithms-eff0343174c0.

[15] K. Hofstad. *Vindenergi. (Hentet: 2021-04-06)*. 2021. URL: https://snl.no/vindenergi.

[16] J.J. *MAE and RMSE — Which Metric is Better? (Hentet: 2021-05-09)*. 2016. URL: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.

[17] M. Jensen. *What is Supervised Learning? (Hentet: 2021-05-07)*. 2020. URL: https://neurospace.io/blog/2020/08/what-is-supervised-learning/.

[18] F. Malik. *Neural Networks Bias And Weights. (Hentet: 2021-05-09)*. 2019. URL: https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da.

[19] J. Mamen. *Klima i Norge. (Hentet:2021-05-07)*. 2021. URL: https://snl.no/klima_i_Norge.

[20] J. Morren, J. Pierik og S. W. de Haan. Inertial response of variable speed wind turbines. *Electric Power Systems Research* 76 (11) (jul. 2006): 980–987. DOI: 10.1016/j.epsr.2005.12.002.

[21] Naturvernforbundet. *Det er energi i vinden*. 1990: 17.

[22] NVE. *Kraftproduksjon fra vindturbiner. (Hentet: 2021-02-25)*. 2019. URL: https://www.nve.no/energiforsyning/kraftproduksjon/vindkraft/kraftproduksjon-fra-vindturbiner/.

[23] C. Olah. Understanding LSTM Networks [Blog]. (Hentet: 2021-02-25). *Web Page* (2015). URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[24] M. Olsson. *C Quick Syntax Reference*. 2015. DOI: 10.1007/978-1-4302-6500-9.

[25] C. Pascual. *Tutorial: Understanding Linear Regression and Regression Error Metrics. (Hentet: 2021-05-09)*. 2018. URL: https://www.dataquest.io/blog/understanding-regression-error-metrics/.

[26] S. Raschka og V. Mirjalili. *Python Machine Learning: Machine Learning & Deep Learning with Python, Scikit-Learn and TensorFlow 2, Third Edition*. 2019.

[27] Regjeringen. *Sterk vekst i fornybarnæringen og leverandørindustrien i 2019. (Hentet: 2021-05-14)*. 2020. URL: https://www.regjeringen.no/no/aktuelt/sterk-vekst-i-fornybarnaringen-og-leverandorindustrien-i-2019/id2781161/.

[28] S. Ronaghan. *Deep Learning: Overview of Neurons and Activation Functions. (Hentet: 2021-05-27)*. 2018. URL: https://srnghn.medium.com/deep-learning-overview-of-neurons-and-activation-functions-1d98286cf1e4.

[29] K. Rosvold. *Vindturbin. (Hentet: 2021-05-07)*. 2019. URL: https://snl.no/vindturbin.

[30] K. Seter. *Vind. (Hentet: 2021-05-07)*. 2020. URL: https://snl.no/vind.

[31] K. Shen. *Effect of batch size on training dynamics*. 2018. URL: https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e.

[32] Shutterstock. *Electric Generator Drawing. (Hentet: 2021-05-07)*. 2021. URL: https://www.shutterstock.com/nb/image-vector/electric-generator-drawing-flat-vector-illustration-1642810255.

[33] J. Smalley. *How are motors and drives used in a wind-turbine nacelle? (Hentet: 2021-05-07)*. 2015. URL: https://www.windpowerengineering.com/where-do-motors-and-drives-find-applications-in-wind-turbine-nacelles/.

[34] Statkraft. Smøla vindpark. *(PDF-fil)* (2011): 2–3.

[35] Statkraft. Materialforbruk YAW Smøla (4157). Internt dokument. *(Excel-fil)* (2020).

[36] Statkraft. Smola_map. Internt dokument. *(JPG-fil)* (2020).

[37] Statkraft. Smola1 info. Internt dokument. *(Excel-fil)* (2020).

[38] Statkraft. Smola2 info. Internt dokument. *(Excel-fil)* (2020).

[39] Store Norske Leksikon. *Asynkronmotor. (Hentet: 2021-02-25)*. 2020. URL: https://snl.no/asynkronmotor.

[40] Store Norske Leksikon. *Drivaksel. (Hentet: 2021-02-25)*. 2020. URL: https://snl.no/drivaksel.

[41] J. Twidell og T. Weir. *Renewable Energy Resources*. 2015. DOI: 10.4324/9781315766416.

[42] T. Ulery og M. Riley. *Increasing wind turbine reliability through blade pitch control upgrades. (Hentet: 2021-05-07)*. 2020. URL: https://www.renewableenergyworld.com/wind-power/increasing-wind-turbine-reliability-through-blade-pitch-control-upgrades/#gref.

[43] E. S. Viseth. *Vindturbinene blir stadig større: – Vi må sette en grense. (Hentet: 2021-04-06)*. 2019. URL: https://www.tu.no/artikler/vindturbinene-blir-stadig-storre-vi-ma-sette-en-grense/476738.

[44] R. Vågstøl, S. Ranaweera og Ø. Edvardsdal. Dobbelmatet Asynkronmaskin. *Høgskulen på Vestlandet* (2019): 42.

[45] Wikipedia. *Yaw system. (Hentet: 2021-05-07)*. 2021. URL: https://en.wikipedia.org/wiki/Yaw_system.

[46] T. Wildi. *Electrical Machines, Drives, and Power Systems*. Bd. 1. 2002.

[47] C. Zhang, L. Cao og A. Romagnoli. On the feature engineering of building energy data mining. *Sustainable Cities and Society* 39 (2018). DOI: 10.1016/j.scs.2018.02.016.

[48] X. Zhang, Q. Zhang, G. Zhang, Z. Nie, Z. Gui og H. Que. A novel hybrid data-driven model for daily land surface temperature forecasting using long short-term memory neural network based on ensemble empirical mode decomposition. *International Journal of Environmental Research and Public Health* 15 (5) (mai 2018). DOI: 10.3390/ijerph15051032.

# A. Kode for sannsynlighetsutvikling

```
from math import exp

def linear(x):
    return x

def exponential(x):
    return exp(1-(1/x**2))

def dist_prob(y, hours, function):
    n_steps = hours*6
    step_list=[]
    y_copy = y.copy()
    y_copy = pd.Series(y_copy)
    for step in range(1, n_steps):
        step_list.append(function(step/n_steps))

    for index, element in enumerate(y_copy):
        if element == 1:
            counter = 1
            while counter <= len(step_list):
                if y_copy.iloc[index-counter] == 0:
                    y_copy.iloc[index-counter] = step_list[-counter]
                counter += 1

    return y_copy



result = dist_prob(y_list,2,exponential)
result = list(result)
result = pd.Series(result)
```

**Figur A.1:** Koden for utregningen av sannsynlighetutviklingen før hver alarm i alarmkolonnen

# B. Kode for modellbyggingen

```
# Building model

import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense, LSTM, Dropout

def build_model(input_shape):
    """Generates RNN-LSTM model
    :param input_shape (tuple): Shape of input set
    :return model: RNN-LSTM model
    """

    # Build network topology
    model = Sequential()

    # 2 LSTM layers
    model.add(LSTM(32, input_shape=input_shape, return_sequences=True)) #␣
↪sequence to sequence layer
    model.add(LSTM(32)) # sequence to vector layer

    # Dense layer
    model.add(Dense(32, activation='relu'))
    # Dropout layer
    model.add(Dropout(0.3))

    # output layer (Dense)
    model.add(Dense(1)) # number of different output values we want to predict

    return model
```

**Figur B.1:** Koden for funksjonen som viser oppbygning av modellen

# C. Kode for utregningen av feilprediksjon (MAE)

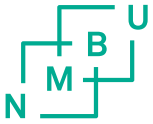
```python
def deviation(pred,test,lower=0.15):
    """Generates the average difference error between the two inputs
    :param input: pred, test: Predicted values and test values for the same␣
 ↪period
    :returns: The average error for the chosen interval and all values (total)
    """
    total_error = []
    interval_error = []
    for pred_value, test_value in zip(pred,test):
        if pred_value >= lower and test_value == 0:
            total_error.append(abs(pred_value-lower))
        if test_value != 0:
            interval_error.append(abs(test_value-pred_value))
            total_error.append(abs(test_value-pred_value))

    average_interval_error = sum(interval_error)/len(interval_error)
    average_total_error = sum(total_error)/len(total_error)

    return average_interval_error, average_total_error

result_interval, result_total = deviation(pred, y_test)
print(result_interval, result_total)
```

**Figur C.1:** Koden for utregningen av feilprediksjonen, det gjennomsnittlige avviket mellom predikerte verdier og testverdier. Funksjonen returnerer utregning for feilprediksjon i intervallet der testverdien er >0, og feilpredikjsonen for alle målingene.




**Norges miljø- og biovitenskapelige universitet**
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway