Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

Peder Enes Ward

Prediksjon av fremtidig energiforbruk i bygg ved bruk av maskinlæring

Masteroppgave i Kybernetikk og robotikk
Veileder: Ole Morten Aamo
Juni 2022

NTNU
Norwegian University of
Science and Technology

Peder Enes Ward

Prediksjon av fremtidig energiforbruk i bygg ved bruk av maskinlæring

Masteroppgave i Kybernetikk og robotikk
Veileder: Ole Morten Aamo
Juni 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU
Norwegian University of
Science and Technology

Sammendrag

I de senere årene har nettleverandører av energi vært misfornøyd med det ujevne forbruket på nettet og har innført tariffer som straffer kundene økonomisk ved høyt energiforbruk innen kort tidsvindu. Dette er gjort for å utnytte strømnettet optimalt, få et mer stabilt nett, samt begrense utbygging og inngrep i naturen. GK Norge er tekniske entreprenør og servicepartner som leverer smarte og bærekraftig løsninger innenfor ventilasjon, kulde og byggautomasjon. Etter innføring av effekttariffer har kunder av GK etterspurt smarte løsninger som kan hjelpe byggherrene å unngå effekttopper samt generelt bespare energi i byggene deres. For å lage en slik løsning må det først utvikles en modell som kan finne tidsrommet når det er forventet effekttopp og høyt energiforbruk. På bakgrunn av dette har GK Norge sammen med NTNU utviklet en masteroppgave som omhandler predikering av energibehovet i bygg. GK Norge ønsker å se på muligheten for å løse dette med maskinlæring da det følger deres visjon om å være et fremoverlent selskap som utnytter nyeste teknologi. Oppgaven er avgrenset til å utvikle prediksjonsmodellene og inneholder derfor ikke noe form for implementasjon i byggautomasjon.

Problemstillingen og oppgaven er løst ved å bygge to modeller gjennom en systematisk og ryddig fremgangsmåte bestående av teknikker innenfor teori og praktisk tilnærming for modellering av maskinlæringsmodeller. Det har blitt brukt en god del tid på uthenting og rensing av datasett for modelleringen. Ulike modelltyper har blitt testet ut fra det tidligere arbeidet som har blitt gjort på feltet, med modelltyper som SARIMAX og LSTM. Modelltypene har blitt optimalisert med ulike hyperparameter og sammenlignet opp mot hverandre ved bruk av kjente evalueringsmetoder og ytelsesmål. Den valgte modelltypen har blitt videreutviklet og flere modeller av samme modelltype har blitt testet. Den endelige modellen har blitt brukt til å finne det optimale datasettet før den til slutt har blitt anvendt på forskjellige eksperimenter for å vise sin fleksibilitet og begrensninger.

Den første modellen predikerer energibehovet for bygget 48 timer frem i tid med lavt avvik og brukes for å oppnå en mer generell forståelse av fremtidig energiforbruket for bygget. Modellen kan brukes for å gi beskjed når termisk energi bør lagres for å bli utnyttet til riktig tid, når andre energikilder bør ta over og når energi bør forskyves.

Den andre modellen predikerer energiforbruket i den aktuelle timen. Denne modellen kan brukes for å unngå effekttopper da den kontinuerlig gjennom en time predikerer den forventede energiforbruket for den aktuelle timen. Dersom modellen predikerer et høyt energiforbruk kan det tidlig i timen gjøres handlinger for å unngå en effekttopp.

Arbeidet i denne rapporten viser hvordan utvikling av i disse to modellene er løst, samt hvordan de presterer etter gitte krav fra GK Norge. Kravene er oppfylt og rapportens resultat er løsningen på de gitte problemstillingene.

Abstract

In recent years, grid suppliers have introduced tariffs that penalize customers financially for high energy consumption within a short period of time. This is implemented to avoid uneven consumption on the electrical grid and therefore make optimal use of the grid, get a more stable grid, as well as limit the expansion of the grid and intervention in nature. GK Norge is a technical contractor and service partner who delivers smart and sustainable solutions within ventilation, cooling and building automation. Following the introduction of grid tariffs, GK's customers have requested smarter solutions that can help building owners to reduce power peaks and also save energy. To create such a solution, it is necessary to first locate when power peaks and high energy consumption are expected. Based on this, GK Norge has collaborated with NTNU and developed a master's thesis which will investigate how to predict energy consumption in buildings. GK Norge wants to explore the possibilities to solve this using machine learning as it corresponds to their vision to be a forward-looking company that utilizes the latest technology. The project is limited to doing the modeling of the prediction models and does not contain any form of implementation in building automation.

The problem is solved by modeling two models through a systematic and orderly approach consisting of techniques within theory and practical approach in modeling of machine learning models. A considerable amount of time was spent on retrieving and cleaning datasets before modeling. Different model types were investigated based on previous work in the field, like SARIMAX and LSTM models. The model types have been optimized with different hyperparameters and compared against each other using well known evaluation methods and KPI's. The selected type of model has been further developed and several variations of the same type have been tested. The final model was used to find the most optimal dataset before the model has been used in various experiments to show its flexibility and limitations.

The first model predicts the energy consumption for the building 48 hours ahead with small deviation and is intended to be used for a more general understanding of the future energy consumption of the building. It can be used to indicate when thermal energy should be stored for utilization, when other energy sources should be used and when energy should be shifted.

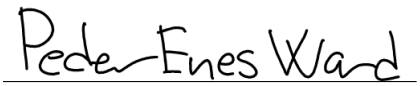
The second model predicts the energy consumption in the current hour. This model can be used to avoid power peaks as it continuously predicts the expected energy consumption for the current hour. If the model predicts a high energy consumption, actions can be taken early in the hour to avoid a power peak.

The work in this report shows how the development of these two models has been conducted, as well as how they perform according to given requirements from GK Norge. The requirements are met, and the results found in this report would serve as the solution to the problems given.

Forord

Denne rapporten er skrevet som en avsluttende del på masterstudiet Kybernetikk og Robotikk ved Institutt for teknisk kybernetikk, Fakultet for informasjonsteknologi og elektroteknikk ved Norges teknisk-naturvitenskapelige universitet - NTNU. Oppgaven er utarbeidet i et samarbeid mellom NTNU og GK Norge, og har blitt utført i tidsperioden mellom januar 2022 og juni 2022. Rapporten gir en grundig beskrivelse på fremgangsmåten fra ide til to ferdig utviklede modeller. Modellenes hensikt er å predikere energibehovet i bygg for forskjellige tidshorisonter frem i tid. Arbeidet gjort i denne rapporten har blitt utført med stor grad av selvstendighet. NTNU har gitt akademisk støtte og GK har bidratt med software, hardware og fagekspertise innenfor bygg-automasjon.

Jeg ønsker å takke Ole Morten Aamo som veileder fra NTNU samt Knut Ivar Grue og Mathias Ekkerhaug som veiledere fra GK Norge. Som en avsluttende del av studiet ønsker jeg også å gi en takk til familie og venner for støtten gjennom studietiden.



Peder Enes Ward

Trondheim, 02.06.2022

Innhold

Figurer

Tabeller

1 Innledning

1.1 Introduksjon

I dette kapittelet skal hensikten med prosjektet komme frem, hva som er bakgrunnen for oppgaven og hvilke problemstillinger oppgaven bygger på. Målene rundt prosjektet er forklart, samt avgrensningene. Til slutt en gjennomgang av hvordan rapporten er bygd opp.

1.2 Oppdragsgiver

Prosjektets oppdragsgiver er GK Norge som er et familieeid selskap opprettet i 1964. GK er Skandinavias ledende tekniske entreprenør og servicepartner og leverer smarte løsninger innen ventilasjon, kulde, byggautomasjon, elektro og rør. GK har rundt 100 kontorsteder, 3000 ansatte og omsetter for 6,1 milliarder i tre land [10].

I tillegg til å tilby en masteroppgave i samarbeid med NTNU har GK bidratt med utstyr i form av PC, lisenser og andre kostnader samt kontorplass om ønskelig. De har også tilbudt domenekunnskap innenfor flere felt.

1.3 Bakgrunn

GK Norge har nylig utarbeidet en strategi der en av komponentene er et tydeligere klima- og fremtidsfokus, i tillegg til at selskapet har opprettet en ny avdeling; Byggteknologi og Utvikling. Slik prismodellen er for enkelte energikilder i dag blir byggherrer straffet hardt økonomisk for høyt energiforbruk innenfor korte tidsperioder. Kunder av GK har derfor etterspurt tjenester for å unngå slike effekttopper og generelt bespare energi. Byggebransjen er under store endringer og det kommer stadig nye krav om energibesparende tiltak. Siden første kvartal i 2021 har også strømprisene økt i Norge [83]. GK ser derfor på mulighetene for å utvikle tjenester som kan forskyve eller kutte energi smart samt bytte mellom forskjellige energikilder. Dette vil følge GK sin visjon da det er fremtidsrettede løsninger som gagner storsamfunnet og gir GK en oppdatert salgsporteføljе med nye og innovative løsninger. GK har i tillegg et stort fokus på å ha et godt samarbeid med studenter og universiteter i Norge og ut fra dette har studentoppgaver som denne masteroppgaven blitt opprettet.

1.4 Problemstilling

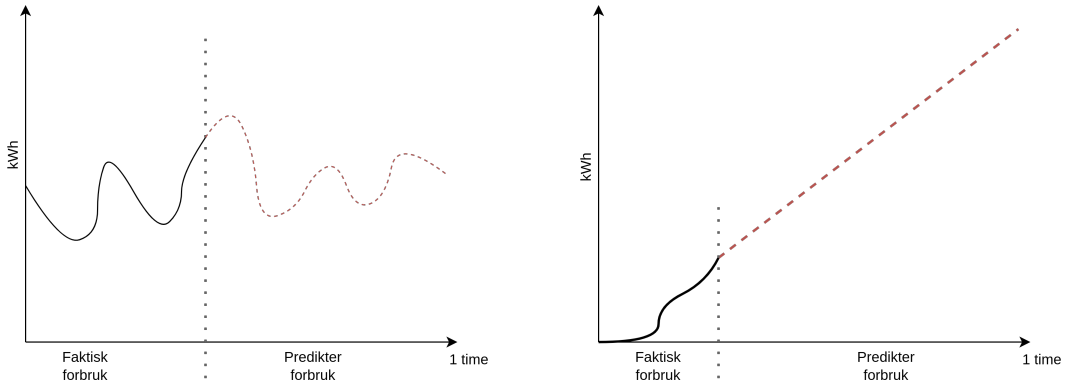
Det finnes i hovedsak to metoder for å unngå effekttopper i dag. En metode er å predikere energibehovet i en lengre tidsperiode fremover for deretter å unngå høyt forbruk, ved å forskyve energiforbruket fra en periode med predikert høyt behov til en periode med lavere behov. Denne metoden

kalles effektforskyver [40].

En annen metode er å overvåke energiforbruket og deretter kutte effekter eller tilføre annen energikilde dersom det nærmer seg en effektgrense som er satt for å unngå en effekttopp. Dette systemet kalles for effektvokter eller maksvokter [88].

For å ha mulighet til å forskyve energi må det finnes en god prediksjonsmodell som kan finne behovet for forskyvningen. Denne prediksjonsmodellen blir videre i rapporten kalt langtidspredikering, og utviklingen av den er del en av problemstillingen. Modellen skal ta i bruk eksogene variabler for å få en bedre prediksjon. Den modellen er illustrert i figur 1a.

I de fleste tilfeller blir effekttopper målt som høyest energiforbruk innenfor en time i den gjeldende måneden. For å unngå høyt forbruk i en time må det utvikles en modell som kan predikere det totale akkumulerte forbruket i den aktuelle timen og dermed finne energiforbruket den aktuelle timen. Dersom forbruket tidlig i timen overstiger en grense må modellen gi beskjed videre slik at energiforbruket kan kuttes eller byttes til en annen energikilde. Denne modellen er del to av problemstillingen og kalles korttidspredikering i denne rapporten. Figur 1b viser at det er ønskelig å predikere det resterende akkumulerte forbruket i den aktuelle timen, som kan brukes til å beregne energiforbruket for timen.



(a) Langtidspredikering.

(b) Korttidspredikering.

Figur 1: Prediksjonsmodellene som inngår i problemstillingen.

1.5 Samfunnsmessig gevinst

Dersom det implementeres et system som kan redusere effektopper og forskyve energi kan dette gi store gevinster for storsamfunnet. De viktigste av disse er presentert er [85]:

- Mindre utbygging av strømnettet.
- Økonomiske besparelser for stat, private selskaper og kommuner.
- Muliggjøre for større andel fornybar energi.

- Redusert CO_2-utslipp.
- Mer pålitelig strømnett.
- Mindre inngrep i naturen.

1.6 Prosjektmål

Hovedmålet og dermed resultatmålet for oppgaven er å levere modeller for prediksjon av energi som kan brukes til energibesparende tiltak. Delmålene for oppgaven er:

- Undersøke om data tilgjengelig fra GK og ulike API er tilstrekkelig for å lage prediksjonsmodeller.
- Analyse av dataen.
- Sette opp en fremgangsmåte som er strukturert og ryddig for valg av modell.
- Utvikle en modell for langtidspredikering.
- Utvikle en modell for korttidspredikering.
- Gjør en analyse av hvilken og hvor mye data som er viktig for modelleringen.

GK ønsker senere å bruke disse modellene i tjenester som kan selges til deres kunder. Det gjør at oppgaven får følgene effektmål:

- Økt oppmerksomhet rundt bedriften for bruk av ny teknologi.
- Økt oppmerksomhet rundt bedriften for energibesparende tiltak.
- Økt salg av tjenester.
- Økt kundeportefølje.

Gjennom prosjektets periode er det også satt opp prosessmål:

- Oppnå kunnskap rundt det å samarbeide med en bedrift.
- Få erfaring rundt prosessen av å utvikle og implementere maskinlæringsmodeller. Fra ide til produkt.
- Få erfaring med databehandling.

1.7 Avgrensninger

Prosjektet er avgrenset slik at oppgaven kun skal ta for seg modelleringen av prediksjonsmodellene og analysene rundt modellene og datasettet. Det vil si at oppgaven ikke tar for seg hvordan systemene på bygget skal styres ut fra informasjonen fra modellene. Det vil derfor ikke implementeres noe styring i form av forskyvning eller kutting av energi. Oppgaven tar for seg uthenting av dataen, modelleringen av prediksjonsmodellene og analyse av dataen og modellene. Modellene skal være generelle som mulig slik at de kan brukes i flest mulig bygg, men likevel er avgrenset til å testes på ett bygg.

1.8 Definisjoner

Array	En datastruktur bestående av en samling av objekter som kan indekseres [113].
ANN	Artificial neural network.
API	Grensesnitt for å samhandle mellom ulike programvarer [90].
Azure Machine Learning	En tjeneste på foretaksnivå for ende-til-ende-livssyklusen for maskinlæring [5].
Bangeløkka	Testbygg for oppgaven. Bangeløkka er også et sted i Drammen.
Dataramme	En todimensjonal merket datastruktur med kolonner av potensielt forskjellige typer [54].
DNN	Deep neural network.
eSight	Energi oppfølging system [2].
Feature	Begrep i maskinlæring og mønstergjenkjenning om individuelle målbare egenskaper [29].
Frost	API levert av MET for tilgang til historisk meteorologiske data [35].
GK Norge	Teknisk entreprenør og servicepartner. Oppdragsgiver av denne oppgaven [10].
GK Cloud	Skybasert teknologiplattform for styring og overvåkning av bygg [39].
Grådig algoritme	En algoritme som iterativt velger det som ser best ut i øyeblikket, men som ikke nødvendigvis er det beste til slutt [42].
HVAC	System som bruker ulike teknologier for å kontrollere temperaturen, fuktigheten og rensing av luft i en bygning eller rom [52].
Hyperparameter	Parametere som settes før modellen trenes og vil dermed påvirke treningsprosessen [7].
Lag-verdi	Verdier ved tidligere tidstrinn [6].
LSTM	Long short-term memory. Type nevralt nettverk [7].
MET	Offentlige meteorologiske tjenesten for blant annet værvarsel [86].
Microsoft Azure	Skyplattform skapt av Microsoft for blant annet infrastruktur, beregninger og databehandling i skyen [38].
NaN-verdi	Brukes til å identifisere udefinerte eller ikke-representerbare verdier [78].

Norsk Klimaservicesenter (KSS)	Tilrettelegger og formidler av klima- og hydrologiske data [57].
Sample	En verdi av signalet for en gitt tid [98].
Samplingstid	Tiden mellom hver sampling av et analogt signal [98].
SARIMAX	Seasonal Auto-Regressive Integrated Moving Average med eXogenous variabler. Type prediksjonsmodell [99].
Skalerbarhet	Evnen til et system, nettverk eller en prosess til å håndtere en voksende mengde med arbeid [11].
SVR	Support vector regression.
SVM	Support vector machine.
PLS	Programmerbar logisk styring.
Visual Studio Code	En kildekoderedigerer laget av Microsoft for Windows, Linux og macOS [122].
U-verdier	Et mål for å angi en bygningdel sin varmeisolerende evne [51].

1.9 Rapportens oppbygging

Rapportens oppbygging baser seg på "Mal for å skrive masteroppgave" som er tilgjengelig på NTNU sine hjemmesider [74]. Rapporten har to målgrupper; studenter som har fullført en mastergrad innenfor teknologistudier og ingeniører ved GK Norge. Siden det kan være noe sprikende kunnskapsnivå på ulike domener mellom de to målgruppene er det forsøkt å legge rapporten på et grunnleggende nivå. Rapporten er skrevet på norsk etter ønske fra oppdragsgiver, GK Norge. Kildekode og datasettet er ikke vedlagt med rapporten da GK ønsket at disse ikke skulle være åpen.

Prosjektet er i hovedsak delt opp i to deler, langtidspredikering og korttidspredikering. Begge delene har eget hovedkapittel som det er mulig å lese hver for seg, men noe referering mellom disse kapitlene er gjort for å unngå mye gjentagelse. Teorikapittelet er felles og tar for seg teori anvendt i oppgaven. Før teorikapittelet er det skrevet noe om bakgrunnen for oppgaven og problemstillingen. I tillegg er ulike mål for oppgaven satt opp, samt avgrensningene tatt med i innledningen. Det er skrevet et kapittel om utviklingsmiljøet som omhandler oppsettet for å utvikle modellen, og et for produksjonsmiljøet som omhandler hvordan modellen settes i produksjon. Datasett og testforberedelser er også felles kapitler for modelleringen. Disse tar for seg uthenting og behandling av datasettet samt forberedelser før modelleringen. Det meste av diskusjonene foregår inne i kapitlene, men hvert hovedkapittel har eget diskusjons- og konklusjonskapittel. Til slutt er det skrevet en hovedkonklusjon for hele rapporten samt skrevet om videre arbeid.

Rapporten er skrevet i Latex der det er utnyttet estetiske hjelpemidler for å gjøre rapporten mer lettleselig, ryddig og oversiktlig. Det er brukt tid på å lage egne figurer for rapporten som skal gjøre rapporten mer forståelig. I tillegg er det brukt tid på å visualisere resultatet og data med figurer, grafer og tabeller fra ulike bibliotek i Python.

Gjennom rapporten er det referert til litteratur ved hjelp av BibTex biblioteket. Referansestilen anvendt er IEEE, som er en vanlig referansestil for tekniske rapporter [87]. Det er brukt mye tid på å finne akademiske artikler og rapporter som kilder og referanser til rapporten. I tillegg er det blitt hentet mye god informasjon om praktisk bruk av maskinlæring som ikke er publisert. Dette kan være kilder som Towards Data Science [117] og Machin Learning Mastery [66]. Dette er informasjon som krever kildekritikk, men kan være informativ og nyttig. En undersøkelse av forfatterne kan avgjøre om de sitter med nok erfaring og kunnskap til at det kan være en god referanse.

I rapporten er det forsøkt å bruke standardiserte benevnelser slik at leseren kan gjenkjenne dem uten å trenge en forklaring. Dersom leseren ikke er kjent med standardiserte benevnelser innen tidsserieprediksjon er de også presentert her. I rapporten blir verdien som skal predikeres kalt y_t der t beskriver tid i forhold til nåtid. $t = t$ er nåtid, $t = t + 1$ er et tidssteg frem i tid og $t = t - 1$ er et tidssteg bakover i tid (historisk verdi). Den predikerte verdien blir kalt $\hat{y}_t$. De ulike verdiene for ulike features blir navnsatt x_t^z der z noterer hvilken feature det er og t hvilket tidssteg det er.

I stedet for å notere alle features hver for seg blir ofte X_t brukt som en fellesbetegnelse for alle features i det aktuelle tidssteget t slik som formel 1 viser. Tabell 1 viser en oversikt over de mest anvendte symbolene i rapporten.

$$X_t = [x_t^1, x_t^2, x_t^3 ... x_t^Z] \tag{1}$$

Symbol	**Betydning**
t	Tid (Nåtid)
$t+1$	Ett tidssteg frem i tid
$t-1$	Ett tidssteg bak i tid
N	Antall samples i datasettet (lengde)
S	Antall tidligere verdier i sekvensen
S_L	Lengde på en sekvens
T	Samplingstid
z	Feature nr.
Z	Antall features
y_t	Verdien som skal predikeres. Målverdien
$\hat{y}_t$	Predikert verdi
x_t^z	En sampel for en feature i gitt tid
X_t	Et array med alle features i gitt tid
L	Lengde på prediksjonsvinduet

Tabell 1: Liste over de mest brukte benevnelsene i rapporten.

2 Tidligere arbeid

Det er utført en god del tidligere arbeid på predikering av energiforbruk. Dette kapittelet tar for seg det tidligere arbeidet som er ansett som mest relevant. Tabell 2 oppsummerer det tidligere arbeidet.

Innenfor timeprediksjon har [55] predikert elektrisk energiforbruk i boligbygg ved hjelp av meteorologiske parametre og SVR. [91] har sett på prediksjon av elektrisk forbruk i kommersielle bygninger ved bruk av LSTM og RNN i dype nevrale nettverk med gode resultater. [20] har sett på timeprediksjon med samplingstid på 15 min og bruk av ANN. Her har det blitt brukt HVAC tidsstyring i tillegg til meteorologiske data som features. Denne artikkelen tar også for seg en analyse av hvordan mengden med tilgjengelig data påvirker modellen. I [123] har flere maskinlærings-algoritmer innenfor Boosted-tree/Random forest, SVM og ANN blitt brukt sammen med data fra 47 kommersielle bygninger til å predikere timelige intervaller. Det er også gjort tidligere arbeid ved å sammenligne dype nevrale nettverk med autoregressive modeller slik som ARIMA for å predikere energiforbruket [97]. Her har meteorologiske data og informasjon fra tidsstemplingen blitt brukt som features. Utvidelsen av ARIMA, SARIMA, har også noe tidligere arbeid innenfor dette feltet, blant annet arbeidet til D.Chikobvu og C. Sigauke [101][22]. I tillegg har en annen utvidelse av ARIMA, SARIMAX, blitt uttestet innenfor denne problemstillingen med gode resultater [114][63]. [119] har sammenlignet SARIMA, SARIMAX, ANN og en modifisert SARIMA i predikering av solcelleenergi og viser viktigheten av eksogene variabler. ARIMAX-modellen har tidligere blitt testet på lignende problemstilling både på kort- og langtidspredikering [25][112]. [56] har brukt ulik oppløsning på datasettet fra minutter til ukentlig for å predikere energiforbruk med bruk av LSTM-modell. Det er tidligere gjort arbeid ved å sammenligne LSTM mot ARMA modell for prediksjon av energiforbruket, der LSTM-modellen har vist et bedre resultat en ARMA [124].

For daglige prediksjoner har [82] sett på prediksjon av kjølebehov i kontorbygg ved hjelp av SVR og ANN. Her har de også sett på muligheten for å kombinere opptil fire modeller for å oppnå bedre resultat. [106] har brukt LSTM i DNN for å predikere oljeproduksjon og sammenlignet det med blant annet ARMA, ARIMA, ANN og RNN, der LSTM har vist et godt resultat. Selv om [82] og [106] ikke direkte omhandler prediksjon av energibehov, er mye av problemstillingen lik. For prediksjon av energibehov med daglige prediksjonsvindu har [100] sett på muligheten for å predikere energiforbruket i hotell ved hjelp av SVR. I tillegg har [12] sammenlignet en rekke maskinlæringsteknikker, blant annet LSTM, RNN og Random Forest til å predikere strømbehovet i bygg.

Beskrivelse	Forfatter	Metode
Forecasting energy consumption of multi-family residential buildings using support vector regression	Jain, Rishee K [55]	SVR
Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks	Rahman, Aowabin [91]	LSTM
Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings	Chae, Young Tae [20]	ANN
Accuracy of different machine learning algorithms and added-value of predicting aggregated-level energy performance of commercial buildings	Walker, Shalika [123]	Random Forest, SVM, ANN
Deep neural network based demand side short term load forecasting	Ryu, Seunghyoung [97]	DNN, ARIMA
Prediction of daily peak electricity demand in South Africa using volatility forecasting models	Sigauke, Caston [101]	SARIMA
Regression-SARIMA modelling of daily peak electricity demand in South Africa	Chikobvu, Delson [22]	SARIMA
Short-term load forecasting using a two-stage sarimax model	Tarsitano, Agostino [114]	SARIMAX
Short-term forecasting of temperature driven electricity load using time series and neural network model	Liu, Nengbao [63]	SARIMAX
Comparison of SARIMAX, SARIMA, modified SARIMA and ANN-based models for short-term PV generation forecasting	Vagropoulos, Stylianos [119]	SARIMAX, SARIMA
Forecasting energy consumption in short-term and long-term period by using arimax model in the construction and materials sector in thailand	Sutthichaimethee, Pruethsan [112]	ARIMAX
Short-term city electric load forecasting with considering temperature effects: an improved ARIMAX model	Cui, Herui [25]	ARIMAX
Predicting residential energy consumption using CNN-LSTM neural networks	Kim, Tae-Young [56]	LSTM
LSTM based long-term energy consumption prediction with periodicity	Wang, Jian Qi [124]	LSTM
Early predicting cooling loads for energy-efficient design in office buildings by machine learning	Ngo, Ngoc-Tri [82]	LSTM
Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model	Song, Xuanyi [106]	LSTM
Prediction of energy consumption in hotel buildings via support vector machines	Shao, Minglei [100]	SVR
Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm	Bouktif, Salah [12]	LSTM, Random forrest

Tabell 2: Oversikt over tidligere arbeid.

3 Teori

3.1 Prising av energi

Prising av strøm som energikilde har i hovedsak tre ledd. Ett fastledd, ett energiledd og ett effektledd. Fastleddet er faste kostnader per år som brukes for å dekke de faste kostnadene til nettselskapene. Energileddet er det leddet som i hovedsak reflekterer mengde energi brukt av kunden. Det er en pris på øre/kWh som bestemmes gjennom kraftbørsen i Nord-Europa, NordPool. Effektleddet er et ledd som ikke er påkrevd av nettselskapene å ha, men de fleste har tatt dette i bruk. Dette leddet skal sørge for at kunder som bruker mye energi på kort tid, og dermed tar opp mye av distribusjonsnettet, blir økonomisk straffet. Hovedgrunnen for å introdusere et slikt ledd i prisingen er å utnytte distribusjonsnettet bedre ved å ha et jevnt energiforbruk over tid i stedet for høyt energiforbruk over kort tid som kan danne effekttopper og som da fører til at nettverket må utvides. Hvordan dette leddet utregnes varierer fra nettselskapene da NVE ikke har noen form for standard. Informasjon om prising av strøm som energikilde er hentet fra [80].

Den vanligste metoden for å beregne effektleddet er å finne den timen det var høyest forbruk gjennom en måned, deretter bruke denne verdien sammen med effekttrinn for å gi effektleddet. Tabell 3 viser effekttrinnene for Tensio under kategorien "NMT Effekt lavspent - alle lavspentprodukter".

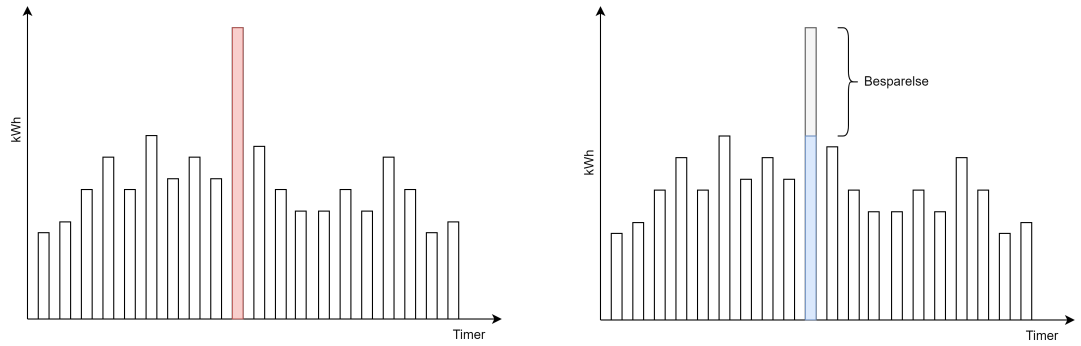
Effekttrinn	**Effektpris vinter (nov - apr)**	**Effektpris sommer (mai - okt)**
0-100 kW	59 kr/kW/mnd	39 kr/kW/mnd
100-400 kW	49 kr/kW/mnd	33 kr/kW/mnd
400+ kW	39 kr/kW/mnd	27 kr/kW/mnd

Tabell 3: Effekttariffer uthentet fra Tensio [81].

Utregningen lar seg forklare enklere ved et eksempel. En kunde har et maksuttak på 550 kW i en time i desember. Da blir beregningene for effektleddet som følger:

$$Effektleddet = 100kW \cdot 59\frac{kr}{kW} + 300kW \cdot 49\frac{kr}{kW} + 150kW \cdot 39\frac{kr}{kW} = 21491kr \tag{2}$$

Dersom det er muligheter for å kutte effekttoppen kan det gi økonomiske besparelser for kunden og et jevnere forbruk på nettet for nettselskapet. Figur 2 viser energiforbruket for en kunde gjennom en måned, der det er den røde søylen utgjør effekttoppen i effektleddet for hele måneden. Ved å kutte effekten for den aktuelle timen vil kunden få en besparelse i form av at effektleddet blir redusert og nettselskapene får et jevnere energiforbruk.



(a) Før effektleddet ble introdusert.

(b) Ønsket virkning av effektleddet.

Figur 2: Ønskelig effekt ved introdusering av effektleddet i tariffen.

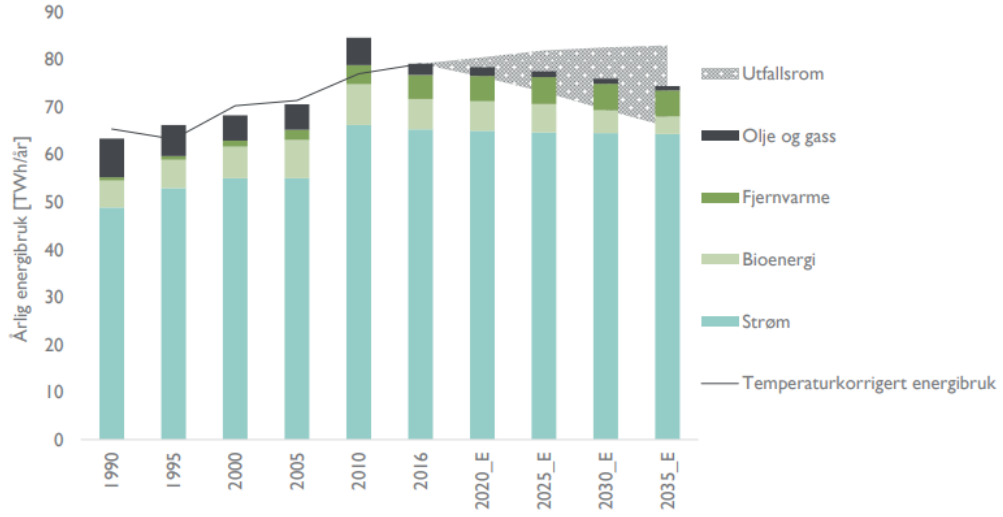
3.2 Energiforbruk i bygg

Energiforbruk i bygg påvirkes av mange faktorer. Faktorene kan deles opp i følgende punkter:

- Uteklima: Lufttemperatur, solstråling, vindhastighet og relativ fuktighet.
- Bygningskarakteristikk: U-verdier, tetthet, areal, type og orientering av bygning.
- Tekniske systemer: Energiforsyningssystemer og tekniske installasjoner.
- Drift og vedlikehold: Energieffektivitet.
- Brukeradferd: Personer som betjener tekniske systemer som oppvarming, ventilasjon og belysning.
- Betingelser for inneklima: Innetemperatur og luftkvalitet.
- Sosiale faktorer: Bruksareal, inntekt, alder og kjønn.

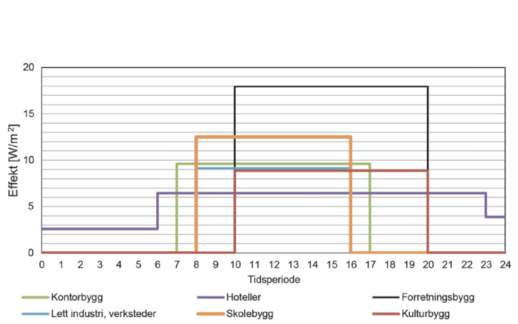
Noen av faktorene er det ikke mulig å påvirke, slik som uteklima. Bygningskarakteristikk er også en nokså fast faktor, da det kreves større arbeid for å påvirke bygningskarakteristikken. Tekniske systemer, drift og vedlikehold er faktorer som kan optimaliseres. De tre siste punktene påvirkes av mennesket og kan være vanskelige å endre uten at det påvirker menneskets komfort. Punktlisten og beskrivelsene er hentet fra [60].

Figur 3 viser at strøm er den energikilden som er mest brukt i bygg i dag. For nye bygg er det ofte strøm, fjernvarme og bioenergi som blir tatt i bruk. Grunnene til at energibruken har flatet ut de siste årene er mer energieffektivt oppvarmingsutstyr, byggene er bedre isolerte, samt elektriske apparater er mer energieffektive [26].

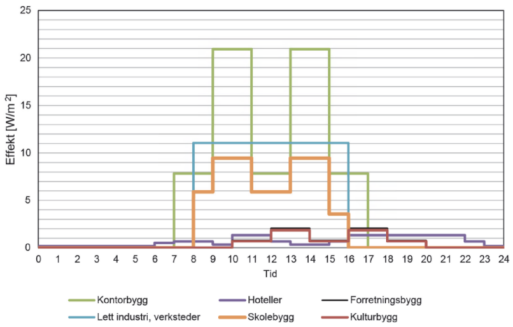


Figur 3: Utvikling i energibruk i bygg mot 2035 [26].

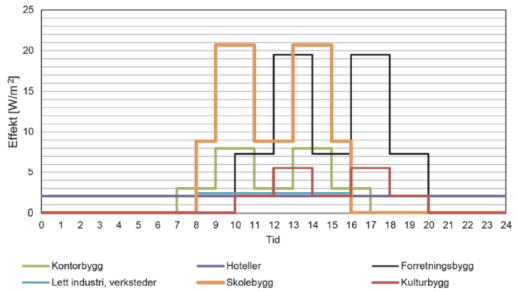
Bruksmønsteret av energi varierer for ulike typer bygg. Dersom energiforbruket analyseres opp mot tid i døgnet viser figur 4 de ulike bruksmønstrene for ulike typer kategorier og bygg. Disse grafene sier også noe om energibehovet.



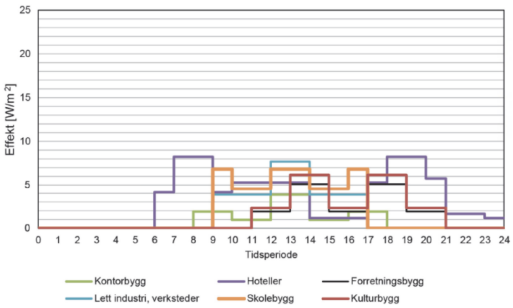
(a) Belysning.



(b) Teknisk utstyr.



(c) Personer.



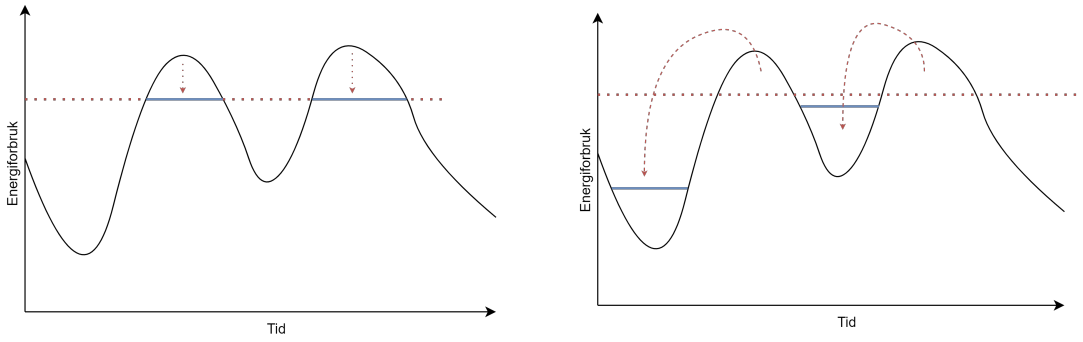
(d) Varmt tappevann.

Figur 4: Bruksmønster for ulike bygg [60][104].

3.3 Effektvokter og effektforsyvning

Ut fra [88] er en effektvokter er et system som kontinuerlig overvåker energiforbruket og følger med på om forbruket nærmer seg en satt grense. Dersom forbruket overstiger grensen gjør effektvokteren tiltak ved å kutte effekter slik at forbruket kommer under grensen. Dette minker energikostnader da effektleddet i prismodellen regnes ut fra høyeste energiforbruk i en time i den aktuelle måned, og en effektvokter kan begrense denne timesverdien. Figur 5a viser hvordan en effektvokter reduserer energiforbruket.

En effektforskyver er et system som jevner ut effektforbruket gjennom en periode. Systemet jevner ut forbruket ved å flytte energiforbruket i en periode med høyt forbrukt til en periode med lavt forbruk. Et eksempel kan være å slå på varmen i et bygg på natten i stedet for morgenen når det allerede er høyt energiforbruk. Slike systemer kan i noen tilfeller bruke mer energi, men spare økonomiske kostnader ved å bruke energi på smartere tidspunkt og unngå effekttopper og forbruk ved høye energipriser. [40][88] Figur 5b viser hvordan en effektforskyver forskyver energiforbruket.



(a) Effektvokter.

(b) Effektforskyver.

Figur 5: Illustrasjon hvordan de to ulike metodene reduserer og flytter på energiforbruket.

3.4 Teknikker og utrykk innenfor maskinlæring

3.4.1 Maskinlæring

Maskinlæring er en algoritme som kan lære fra erfaring E, med noen gitte oppgaver O, og ytelsesmål Y, og kan med utførelse av oppgavene O, målt i Y, øke sin ytelse og presisjon med erfaring E [76].

3.4.2 Veiledet læring

Maskinlæring er i hovedsak delt opp i veiledet læring (supervised learning), ikke veiledet læring (unsupervised learning) og forsterkende læring (reinforcement learning).

Resultatet av veiledet læring i sin enkleste form er en funksjon mellom to variabler y og x slik som

formel 3 viser.

$$y = f(x) \tag{3}$$

Innenfor veiledet læring brukes y som en benevnelse på målverdien eller utgangsverdien, altså den verdien som skal predikeres. x er benevningen på uavhengige features eller input. En feature er en variabel som kan karakterisere eller beskrive egenskaper i datasettet som kan bidra til å predikere y bedre [4]. Veiledet læring kan igjen deles opp i to hovedkategorier; klassifisering og regresjon [79].

3.4.3 Trenings-, valideringss- og testsett

Trenings-, valideringss- og testsett er en en type oppdeling som er vanlig å gjøre på et datasett for å strukturere arbeidet med evaluering og testing av modellen. Definisjonene er gitt som følger [95]:

- Treningssett: Et sett av eksempler brukt for trening av modellen.
- Valideringssett: Et sett med eksempler brukt for å justere hyperparameter i modellen.
- Testsett: Et sett med eksempler bruk til å måle ytelse og generalisering av den endelige modellen.

Det finnes flere anbefalinger på hvordan datasettet bør deles opp. Typiske splittinger er [27]:

- 60% trening, 20% validering og 20% test.
- 70% trening, 15% validering og 15% test.
- 80% trening, 10% validering og 10% test.

Grunnen for en slik type oppdeling er at modellen skal generaliseres godt. Det betyr at modellen skal yte godt på nye og ikke observerte datasett. En typisk framgangsmåte er å først trene modellen, og ut fra gitte ytelsesmål se hvordan den presterer ved testing på valideringssettet. Under testing på valideringssettet justeres hyperparameterene helt til modellen ender opp med et godt resultat. Modellen testes til slutt på testsettet, som den ikke har observert tidligere, for å se om modellen generaliseres godt. Dersom resultatet er dårlig kan det bety at modellen er overtilpasset (overfitted), som igjen tilsier at modellen ikke generaliseres godt.

Innenfor tidsserieprediksjon brukes også utrykkene "hold-out set" for testsettet, siden det er holdt utenfor treningen av modellen. Andre referer til treningssettet med "in-sample data" og testsettet for "out-of-sample data" [53]. I denne rapporten brukes trening-, validering- og testsett.

3.4.4 Ytelsesmål

Ytelsesmål er KPI (Key Performance Indicator) som sier noe om ytelsen på modellen for det gitte datasettet. Det finnes flere type ytelsesmål for prediksjonsmodeller. Makridakis Competitions (M-Competitions) er en serie av åpne konkurranser som har blitt avholdt i flere år for å evaluere de beste prediksjonsmodellene for tidsseriedata [68]. Her har det tidligere blitt brukt ytelsesmål som MSE og RMSE for å finne de beste modellene [67]. MAE er også et vanlig ytelsesmål å anvende for tidsseriedata [127]. Nedenfor presenteres ytelsesmålene som er brukt i denne rapporten.

MAE - Mean Absolute Error er summen av absoluttverdien av alle avvik delt på antall samples som vist i formel 4 [127]. Siden absoluttverdien av avviket er brukt, kansellerer ikke et negativt avvik et positivt avvik. En lav MAE indikerer en god ytelse.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}| \tag{4}$$

MSE - Mean Squared Error er summen av kvadratet av alle avvik delt på antall samples vist i formel 5 [67]. Denne KPI'en i likhet med MAE sørger for å unngå negative avvik ved å ta kvadratet av avviket. Siden det tas kvadratet av avviket vil MSE legge mer vekt på store avvik i prediksjonene. En lav MSE indikerer god ytelse.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{5}$$

RMSE - Root Mean Squared Error er kvadratroten av MSE, vist i formel 6. Dette gjør at denne KPI'en blir i sammen orden og enhet som MAE og kan dermed sammenlignes med hverandre. [67]

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2} \tag{6}$$

AIC -Akaike Information Criterion er en type KPI som ulikt MAE, MSE og RMSE straffer modellen for sin kompleksitet i tillegg til avvikene. AIC bruker "Maximum likelihood estimation", $\hat{L}$, sammen med antall parameter, k, for å finne den beste modellen med minst sett av parameter, gitt i formel 7 [3]. Denne metoden kan ikke brukes for å sammenligne ulike type maskinlærings-modeller, da de kan ha ulike sett av parameter, men brukes heller til å sammenligne modeller av samme type. En lav AIC indikerer god ytelse.

$$AIC = -2ln(\hat{L}) + 2k \tag{7}$$

3.4.5 Univariate og multivariate tidsserier

Univariate tidsserier er en tidsserie med en tidsavhengig variabel [102]. Se tabell 4a.

Multivariate tidsserier er en tidsserie som har flere enn en tidsavhengig variabel [102]. Variablene er ikke bare avhengig av sine tidligere verdier, men kan også ha avhengighet til andre variabler. Denne avhengigheten kan utnyttes for å predikere fremtidige verdier. Se tabell 4b.

Tid	**y**
12:00	1.2
13:00	1.5
14:00	1.4
15:00	1.6
16:00	1.5

(a) Univariate tidsserie.

Tid	**y**	$\boldsymbol{x^1}$	$\boldsymbol{x^2}$
12:00	1.2	5.6	3.5
13:00	1.5	5.1	3.2
14:00	1.4	4.8	3.6
15:00	1.6	5.0	3.2
16:00	1.5	5.2	3.1

(b) Multivariate tidsserie.

Tabell 4: Univariate- og multivariate tidsserier.

3.4.6 Singelsteg- og multistegprediksjon

Singelstegprediksjon (Single Step prediction, SSP) og multistegprediksjon (Multi Step prediction, MSP) er to metoder for å predikere frem i tid. Selv om disse eksemplene nedenfor viser bruken av tidligere verdier for å predikere fram i tid, er det også mulig å bruke verdier i fremtiden, $X_{t+...}$, som features. Slike features kan for eksempel være data fra en værmelding.

Singelstegprediksjon bruker tidligere samples og features til å predikere et tidssteg frem i tid, y_{t+1} [59]. Formel 8 viser et eksempel hvor det brukes J antall tidligere samples fra målverdien, y, pluss en matrise, X, med features M tilbake i tid til å predikere et tidssteg frem i tid.

$$\hat{y}_{t+1} = f([y_t ... y_{t-J}], [X_t ... X_{t-M}]) \tag{8}$$

Multistegprediksjon bruker tidligere samples og features til å predikere L tidssteg frem i tid [59]. Formel 9 viser et eksempel hvor det brukes J antall tidligere samples fra målverdien, y, pluss en matrise, X, med features M tilbake i tid til å predikere L tidssteg.

$$[\hat{y}_{t+1} ... \hat{y}_{t+L}] = f([y_t ... y_{t-J}], [X_t ... X_{t-M}]) \tag{9}$$

3.4.7 Eksogene og endogene variabler

Det er mulig å dele opp input dataen, X, noe mere for å beskrive de ulike features og forholdet mellom de. Gitt en maskinlæringsmodell lik:

$$y = f(A, B, C) \qquad A, B, C \in X \tag{10}$$

A og B er helt uavhengige, mens C er en variabel som er uthentet fra både A og B. Her vil A og B være eksogene variabler og C endogen.

Eksogene variabler er input variabler som påvirker systemet, men som ikke er påvirket av andre variabler i systemet [16].

Endogene variabler er input variabler som påvirker systemet, men som også er påvirket av andre variabler i systemet [16].

3.4.8 Grid Search

Grid search er en metode for å finne den beste modellen ut fra mulige sett med hyperparameter. Grid search-algoritmen setter opp alle mulige hyperparametersett og deretter itererer gjennom de mulige settene og tester modellen på valideringssettet [62]. Hver iterasjon får da et ytelsesmål på hvordan testen gikk. Den testen som har best ytelsesmål har de optimale hyperparameterene.

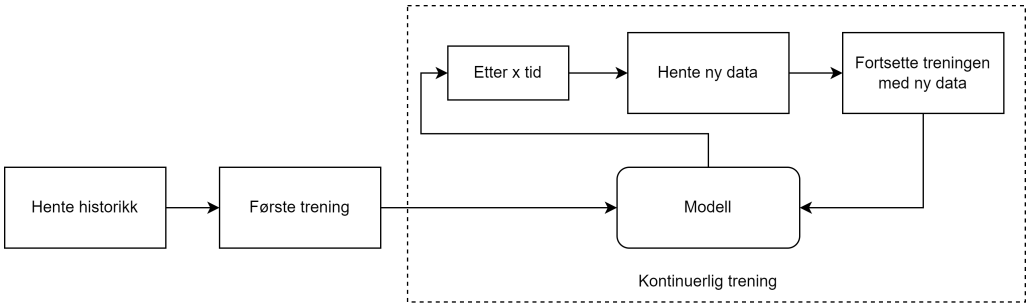
Selv om grid search automatiserer prosessen med å finne optimale hyperparametere kan prosessen være en tidkrevende jobb da antall tester, K, som må utføres er gitt etter formel 11. P er antall parameter og p_i er antall ulike verdier av parametere, i, som skal testes. Dersom det er fem parametere der hver parameter har fem forskjellige verdier som skal testes gir det $K = 5^5 = 3125$ tester.

$$K = \prod_{i=1}^{P} p_i \tag{11}$$

For å unngå mange tester kan ett alternativ være å bruke *Random Gridsearch*. Denne metoden tar ut tilfeldige hyperparametersett og tester [62]. Metoden vil nødvendigvis ikke finne det optimale settet under testingen, men kan få et godt nok resultat på mindre tidsforbruk.

3.4.9 Kontinuerlig trening

Kontinuerlig trening av modeller går ut på å iterativt fortsette treningen med ny data etter hvert som den blir oppdaget [89]. En maskinlæringsmodell innenfor veiledet læring går ut fra at fremtidig data vil ha likheter med dataen modellen allerede har trent på. Etter hvert som tiden går kan ny data endre seg i forhold til tidligere data. For å kontinuerlig ha en oppdatert modell som er trent på nyeste data anvendes kontinuerlig trening. Figur 6 viser et flytdiagram på hvordan kontinuerlig trening bli utført i praksis.

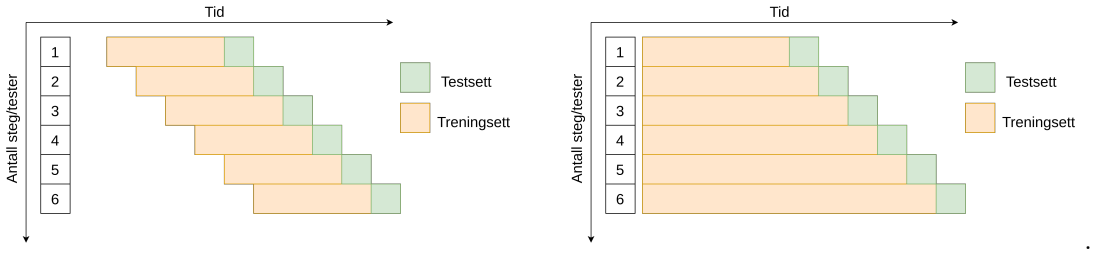


Figur 6: Flytdiagram av kontinuerlig trening.

3.4.10 Walk forward validation

Walk forward validation (WFV) er en metode som brukes for å beregne ytelsesmålene for modellen. I dette delkapittelet presenteres det hvordan WFV-metoden er bygd opp, basert på [118] og [46]. Metoden er anbefalt som en standard metode for å evaluere tidsseriemodeller [46]. Innenfor predikering er det vanlig å evaluere ytelsen på modellen ved å predikere frem i tid og deretter finne ytelsesmålene, for eksempel MAE. Dersom modellen trenes å testes på bare en prediksjon frem i tid vil ikke det gi godt nok grunnlag for å avgjøre om modellen er god eller ikke. Det er ønskelig å gjøre flere prediksjoner og deretter finne gjennomsnittet av alle testene. En metode for å gjøre nettopp dette er å bruke WFV. Metoden går ut på at modellen trenes og deretter predikerer x antall tidssteg frem i tid og finner ytelsesmål. Deretter blir den virkelige dataen over den tidshorisonten modellen predikerte lagt til i treningssettet og kontinuerlig trening brukes for å oppdatere modellen. Slik fortsetter prosessen helt til alle testene er utført og modellen har tatt WFV på validering- eller testsettet. Resultatet av at kontinuerlig trening kjører før det gjøres ny prediksjon er at modellen da vil være trent på den nyeste dataen i forhold til perioden den skal predikere.

Figur 7 viser to forskjellige metoder der WFV er implementert [30]. Figur 7a viser en WFV-metode som bruker glidende vindu for å evaluere modellen. Denne metoden legger til testsettet i treningssettet etter evaluering, men sletter også samme mengde fra den minst relevante treningsdataen, typisk den eldste dataen. Dette gir hver evaluering like mye treningsdata, men vil være avhengig av en god del tilgjengelig treningsdata. Figur 7b viser en metode av WFV som bruker ekspanderende vindu. I motsetning til figur 7a sletter ikke denne metoden gammel treningsdata, som gjør at hver evaluering vil få mer treningsdata. Denne metoden kan være aktuell å bruke dersom det finnes lite treningsdata. Når et nytt subsett blir lagt til er det ikke nødvendig å trene modellen på nytt med det tidligere treningssettet sammen med subsettet da de fleste modellene har mulighet for å fortsette treningen bare med det nye subsettet.



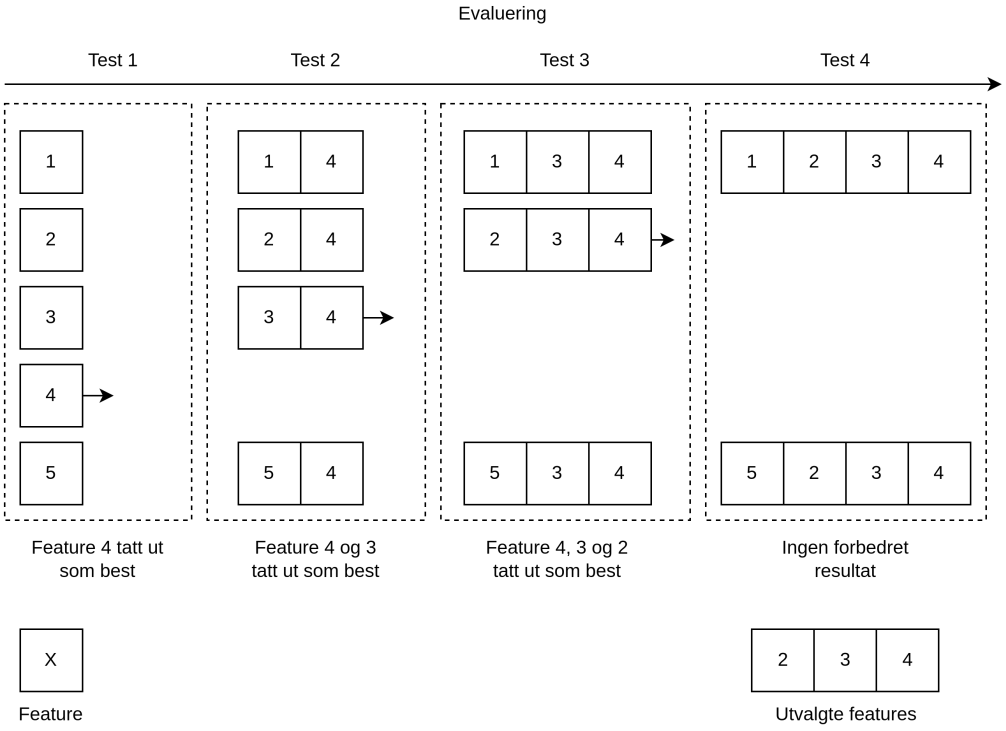
(a) WFV med glidende vindu.

(b) WFV med ekspanderende vindu.

Figur 7: Walk forward validation metoder. Figurer basert på [30]

3.4.11 Feature-utvelging

Feature-utvelging (feature extraction) er prosessen ved å velge ut de features som viser seg å være mest verdifulle for at modellen skal predikere med lavest mulig avvik [94][58]. Ved å velge ut de features som er mest aktuelle vil også kompleksiteten og tidsforbruket på trening gå ned [58]. Wrapper metoden velger iterativt subset av alle features og deretter tester modellens ytelse på valideringssettet. Forward selection er en wrapper metode der algoritmen starter med å teste en og en feature og finner ytelsesmål for hver feature. Deretter testes alle features sammen med den som var best i forrige test for å finne de to beste features. Slik fortsetter det til modellen ikke lenger får noe forbedret ytelsesmål ved å legge til flere features. Figur 8 viser hvordan forward selection kan fungerer som feature-utvelging.



Figur 8: Feature-utvelging.

Forward selection krever både mye tid og datakraft ved mange features, Z, da antall datasett som blir testet på modellen følger rekkeutviklingen i formel 12, som gir en kjøretid på $\mathcal{O}(n^2)$. Wrapper metoden er også en grådig algoritme da den tar det mest optimale valget i det bestemte øyeblikket [58]. Dette kan føre til at det globale optimumet ikke blir funnet.

$$Z + (Z-1) + (Z-2) + \cdots + 1 = \frac{Z(Z+1)}{2} \tag{12}$$

3.5 SARIMAX

SARIMAX er en utvidet modell av ARIMA som er en kjent prediksjonsmodell for tidsseriedata. SARIMAX er en prediksjonsmodell som har muligheten til å utnytte informasjonen i sesongvariasjoner i datasettet og eksogene variabler, derav S og X i navnet. For å presentere hvordan SARIMAX opererer er det enklest å starte med ARIMA.

ARIMA bli også kalt Box-Jenkis modellen siden det var George Box og Gwilym Jenkin som presenterte modellen i sin publikasjon i 1970 [109][13]. ARIMA er bygget opp av tre komponenter, AR (Autoregressive), I (Integrated) og MA (Moving Avarage). Beskrivelsen og forklaringen av modellen i dette delkapittelet er basert på [53].

AR er en autoregressiv modell og predikere ut fra p siste observerte verdier. Modellen kan settes opp slik som vist i formel 13. ϕ bestemmer vektingen på de siste observerte verdiene og c er en konstant verdi. ε_t er hvit støy.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t, \tag{13}$$

MA kan tenkes på som en modell som bruker det glidende gjennomsnittet for å modellere og predikere en verdi. Modellen kan uttrykkes som vektet gjennomsnitt av de tidligere, q, hvite støy verdiene. Formel 14 viser hvordan MA modelleres der θ er MA sin koeffisient.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}, \tag{14}$$

Stasjonær og ikke stasjonær tidsseriedata sier noe om egenskapene til datasettet. En stasjonær tidsserie betyr at dataen har de samme statistiske egenskapene uavhengig av tiden. Mer spesifisert betyr det at y_t er en stasjonær tidsseriedata dersom fordelingen (distribution) av $[y_t ... y_{t+M}]$, for alle M, ikke er avhengig av t. Dersom dataen ikke oppfyller dette kriteriet er den ikke stasjonær. Tidsserie som har sesong og/eller trend er typisk ikke stasjonære. Stasjonære data er viktig for mange statistiske prediksjonsmodeller, deriblant ARIMA. En metode for å gjøre ikke stasjonær data til stasjonær er å bruke differensiering. Differensiering ser på endringen mellom y_t og y_{t-1} og setter $y'_t = y_t - y_{t-1}$. Denne operasjonen fører til at trenden eller sesongen i dataen blir fjernet eller redusert. Dersom dataen fortsatt ikke er stasjonær etter å kjøre differensiering en gang, kan

denne operasjonen kjøres på nytt. Denne operasjonen er angitt som **I** i ARIMA og antall ganger differensieringen kjøres for å få tidsseriedataen stasjonær er angitt som parametere d.

Den fulle modellen av ARIMA kan skrives som:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{15}$$

y'_t beskriver den differensierte modellen. Implementasjonen av ARIMA blir ofte beskrevet som ARIMA(p, d, q) der p kommer fra AR komponenten, d fra I komponenten og q fra MA komponenten i modellen som er beskrevet ovenfor.

ARIMA begrenser seg til tidsseriedata uten sesonger. En utvidet ARIMA modell, SARIMA, har muligheten til å utnytte sesongegenskapene i datasettet også. Modellen blir notert som ARIMA$(p, d, q)(P, D, Q)_m)$ der $(P, D, Q)_m$ er lagt til for å ta hensyn til sesongdata. m beskriver hvor mange samples en sesong har. I et datasett med daglige sesonger og samplingstid på en time bør m settes til $m = 24$ for å beskrive sesongen i datasettet. P, D og Q opererer likt som p, d og q, men i stedet for å bruke $(t-1, t-2, ...)$ ser modellen heller på $(t-m-1, t-m-2, ...)$, altså tidligere observerte verdier fra forrige sesong.

Backshift-notasjonen er en notasjon som gjør det enklere å beskrive tidsserier med lag-verdier. Den defineres som [53]:

$$By_t = y_{t-1} \tag{16}$$

Denne notasjonen gjør notasjonen for differensiering enklere også da:

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1 - B)y_t \tag{17}$$

Som igjen kan skrives som en generell formel med d differensieringer som:

$$(1 - B)^d y_t \tag{18}$$

Med backshift-notasjonen kan modellen ARIMA$(1, 1, 1)(1, 1, 1)_4$ beskrives etter denne formelen (uten konstant) [53]:

$$(1 - \phi_1 B)\ (1 - \Phi_1 B^4)(1 - B)(1 - B^4)y_t = (1 + \theta_1 B)\ (1 + \Theta_1 B^4)\varepsilon_t. \tag{19}$$

Her representerer Φ og Θ koeffisientene i sesongimplementasjonen av AR og MA samme som ϕ og θ representerer koeffisientene i ikke-sesongimplementasjonen av AR og MA.

ARIMA modellen som har blitt gjennomgått frem til nå bruker bare tidligere verdier av y for å predikere fremtidige verdier. Det finnes også implementering av ARIMA som tar hensyn til eksogene variabler, og kan anvende disse i modellen for å få en bedre prediksjon. Det finnes flere måter å

implementere de eksogene variablene i modellen. En metode som er implementert av Statsmodel er å bruke regresjon med ARIMA avvik [99]. Med denne teknikken kan ARIMA modelleres med eksogene variabler. Nedenfor er det vist hvordan eksogene variabler kan implementeres i en ARIMA modell [116]. Eksogene variabler blir notert som x_t.

$$y_t = \beta x_t + n_t \tag{20}$$

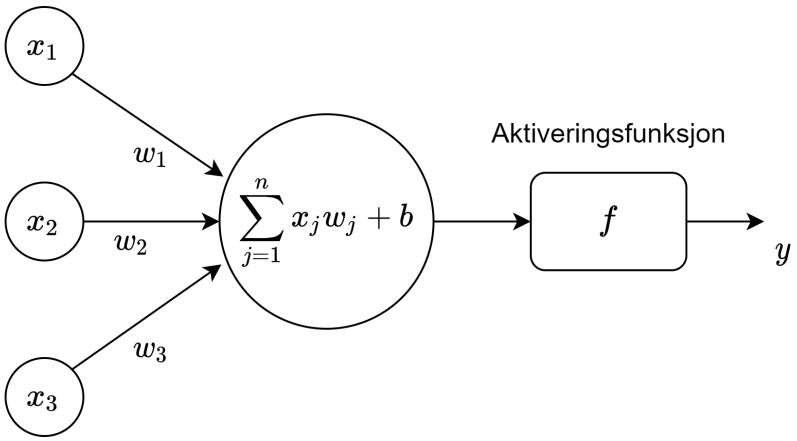
der n_t er beskrevet som:

$$n_t = \phi_1 n_{t-1} + \cdots + \phi_p n_{t-p} - \theta_1 \varepsilon_{t-1} - \cdots - \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{21}$$

Dersom komponentene for sesong og differensiering legges til eksempelet ovenfor er resultatet en SARIMAX-modell slik som Statsmodell presenterer den [99]. Denne modellen tar hensyn til både sesonginformasjon og eksogene variabler.

3.6 Nevrale nettverk

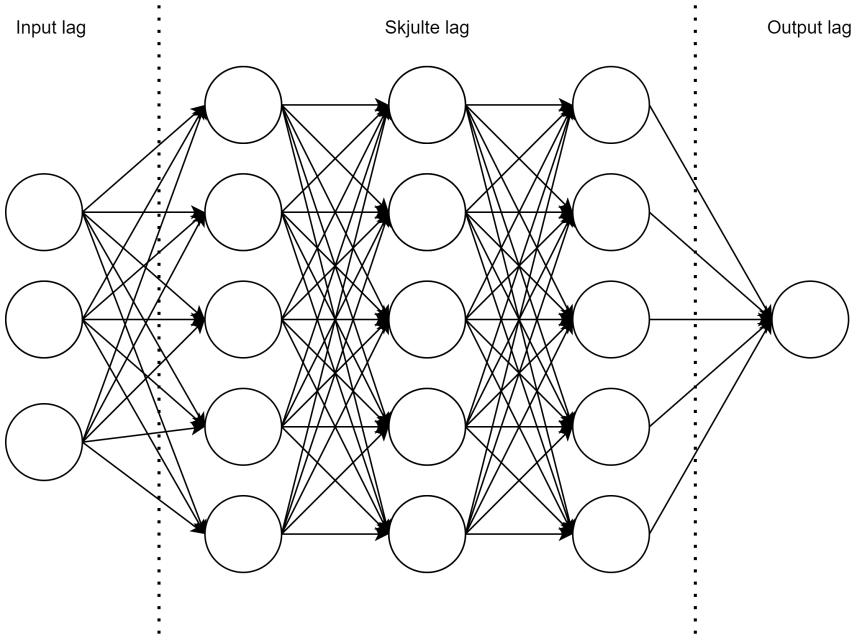
Kunstig nevralt nettverk er en kategori innenfor maskinlæring som kan brukes til å løse komplekse problemer. Nevrale nettverk er bygd opp av lagvise nevroner som sender informasjon mellom hverandre. Denne teknikken er inspirert av menneskehjernen og kan spores tilbake til forskning på perceptroner i 1958 [96]. Figur 9 viser hvordan et nevron er oppbygget. Nevronene i nettverket inneholder en aktiveringsfunksjon. Hva som avgjør om nevronet skal bli aktivert er dens egen aktiveringsfunksjon (f) og bias (b) samt de vektede input ($w_j x_j$) fra andre nevroner [121]. Store delere av teorien i dette delkapittelet er hentet fra boken *Deep Learning* [7].



Figur 9: Oppbygging og funksjonen av et nevron. Figur basert på [84].

Figur 10 viser hvordan et *feedforward* nevralt nettverk er bygd opp. Navnet feedforward kommer av at informasjonen flyter gjennom nettverket uten noe form for tilbakemelding. Til venstre består modellen av et input-lag som tar dataen inn i nettverket. Deretter består modellen av skjulte

lag med n antall nevroner som dataen går igjennom. Til slutt kommer resultatet ut av modellen gjennom output-laget.



Figur 10: Nevralt nettverk (Feedforward). Figur basert på [84].

Målet til et nevralt nettverk er å estimere en funksjon, f^*, som for eksempel kan være en funksjon mellom $\boldsymbol{y}$ og $\boldsymbol{x}$ lik $\boldsymbol{y} = f^*(\boldsymbol{x})$. Under trening prøver modellen å tilpasse en funksjon, $f(\boldsymbol{x};\boldsymbol{\theta})$, slik at den er lik $f^*(\boldsymbol{x})$ der $\boldsymbol{\theta}$ er et sett med parameter som gir den mest optimale funksjonen. Det er ønskelig å lage en funksjon $f(\boldsymbol{x};\boldsymbol{\theta})$, som beskriver det virkelige fenomenet $f^*(\boldsymbol{x})$. Dette kan gjøres ved å minimalisere en kostfunksjon, for eksempel MSE vist i formel 22.

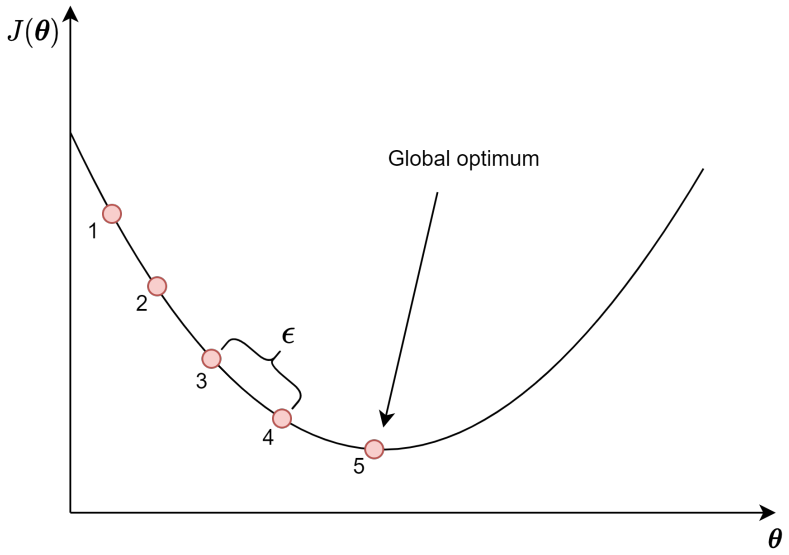
$$J(\boldsymbol{\theta}) = \frac{1}{4}\sum_{\boldsymbol{x}\in\mathbb{X}} \left(f^*(\boldsymbol{x}) - f(\boldsymbol{x};\boldsymbol{\theta})\right)^2 \tag{22}$$

Ved å minimalisere $J(\boldsymbol{\theta})$ vil funksjonen $f(\boldsymbol{x};\boldsymbol{\theta})$ optimaliseres slik at $f(\boldsymbol{x};\boldsymbol{\theta}) \rightarrow f^*(\boldsymbol{x})$ som kan anvendes til å mappe $\boldsymbol{y}$ og $\boldsymbol{x}$ gjennom $\boldsymbol{y} = f(\boldsymbol{x};\boldsymbol{\theta})$. $\boldsymbol{\theta}$ kan inneholde vekter og biaser som blir justert under treningen. For å minimalisere $J(\boldsymbol{\theta})$ brukes ofte *gradient descent*. Gradient descent er en iterativ optimaliseringsalgoritme for å finne det lokale minimum for en funksjon med variablene $\boldsymbol{\theta}$. For en funksjon med flere variabler vil gradienten $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ for funksjonen være i det punktet hvor funksjonen øker mest, altså det bratteste punktet. Gradient descent vil derfor følge den negative gradienten slik at den etter hvert ender opp i det punktet som er minimum for funksjonen. Algoritmen bruker formel 23 for å finne $\boldsymbol{\theta}_{k+1}$ som er neste steg sine parameter for å forsøke og minimalisere funksjonene. Gradienten i formel 23 kan bli funnet ved å bruke en annen type algoritme som heter *back-propagation*.

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_k) \tag{23}$$

Figur 11 viser hvordan funksjonen finner den optimale $\boldsymbol{\theta}$ ved å iterere gjennom formel 23. Gradient

descent finner lokalt optimum, men da figuren viser en kvadratisk funksjon vil den finne det globale optimum. Iterasjonene er vist med tall der ϵ er læringsraten, altså hvor lange steg algoritmen skal ta. Dersom ϵ settes for høyt kan det føre til at deepest descent hopper over det globale/lokale optimumet. En for lav ϵ vil føre til at algoritmen trenger flere steg for å finne det globale/lokale optimumet, som igjen kan føre til økt tid og ressursforbruk.



Figur 11: Gradient descent.

En *epoch* innenfor læringsalgoritmer referer til hvor mange ganger treningssettet har gått gjennom algoritmen. Enkelt forklart er det en løkke der treningssettet går gjennom læringsalgoritmen for et gitt antall ganger. For hver epoch oppdateres de indre modellparameterne.

3.7 Regularisering

Regularisering er prosesser og teknikker for å unngå at modellene blir overtilpasset (overfitted). Enkelt forklart betyr overtilpasset at modellen har tilpasset seg for godt til mønster og støy i treningssettet slik at modellen har problemer med ytelsen på nye, usette datasett [61]. For å unngå dette kan ulike type teknikker innenfor regularisering implementeres.

For nevrale nettverk er det mulig å oppdage overtilpasning dersom valideringsloss begynner å øke etter x antall epochs selv om treningsloss synker. Valideringsloss og treningsloss er et ytelsesmål på hvor godt modellen yter er på de ulike datasettene, og beregnes vanligvis for hver epoch. For å unngå overtilpasning kan det for eksempel implementeres tidlig stopp i treningen (early stopping). Dette kan gjøres ved å overvåke loss-verdiene gjennom treningen og deretter stoppe treningen dersom valideringsloss begynner å øke, eller at det er lite endringer i lossverdiene [7].

En annen metode er å implementere *dropout* lag i modellen. Under trening ignorerer dropout

tilfeldige nevroner i modellen. Effekten av dette er at de gjenværende nevronene må ta jobben, som tvinger nettverket til å lære seg mer robuste egenskaper i datasettet og fokuserer på disse [107]. Dette resulterer i at modellen for eksempel ikke vil tilpasse seg støy i datasettet som unngår overtilpasning.

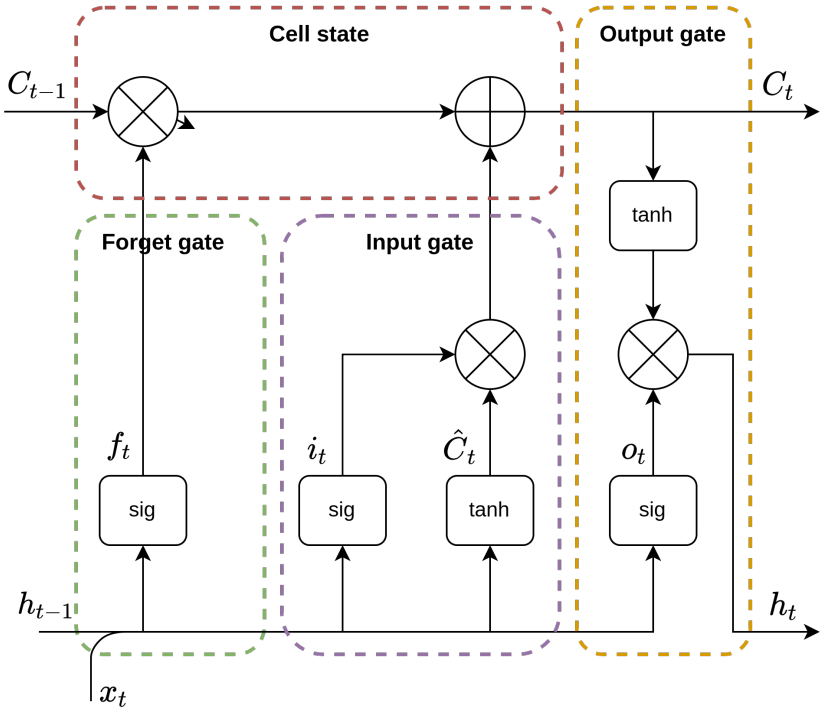
Det er også mulig å unngå overtilpasning ved å minke kompleksiteten til modellen [9].

3.8 LSTM

I kapittel 3.6 ble det vist i figur 10 et såkalt *feedforward network*. Denne typen er et godt utgangspunkt for innføring i nevrale nettverk. Det finnes også flere nettverk, blant annet *Recurrent Neural Network* (RNN), som i motsetning til feedforward er et nettverk med tilbakekobling. Dette fører til at slike nettverk har muligheten til å lagre eller huske informasjon fra input som kan påvirke resultatet på fremtidige output [41].

LSTM står for *Long short-term memory* og er en RNN arkitektur brukt i flere typer dyp læring. Opprinnelig ble LSTM utviklet for å unngå *The vanishing gradient problem* [44]. Et av konseptene bak en RNN modell er at den skal huske tidligere informasjon og kan se sammenhenger mellom nåtidsdata og tidligere gitt data. Når gapet mellom nåtid og tidligere data blir større kan RNN får problemer med å se sammenhenger mellom disse. Ved å bruke LSTM i stedet kan dette problemet unnvikes da LSTM sin hovedoppgave er å finne ut hvilken informasjon som er viktig å huske og hvilken informasjon modellen kan glemme [44]. Siden LSTM er effektiv til å fange opp langtidsforhold i dataen har modellen vært brukt i flere avanserte problemstillinger, blant annet skrift gjenkjenning, analyse av video og lyd til mer komplekse utfordringer innenfor tidsseriedata [41][125].

I likhet med andre nevrale nettverk inneholder også LSTM nevroner, men disse refereres heller til som en LSTM celle. Figur 12 viser en LSTM celle. LSTM cellen består av en *Forget Gate*, *Input Gate*, *Output gate* og *Cell State*. Teorien og beskrivelsen for de ulike modulene i LSTM cellen er hentet fra [37].



Figur 12: LSTM celle. Figur basert på [64].

Forget gate sin oppgave er å finne ut hvilken informasjon fra tidligere som er viktig å bevare og hvilken som kan ignoreres. Input, x_t, og input fra forrige hidden state, h_{t-1}, går gjennom en sigmoidfunksjon og bestemmer sammen med vektmatrisen, W_f og bias, b_f, viktigheten av informasjonen slik som formel 24 viser. Sigmoidfunksjonen genererer en verdi mellom 0 og 1, og er vist i formel 25 [7].

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{24}$$

$$\sigma = \frac{1}{1 + e^{-x}} \tag{25}$$

Input gate består av to deler. Første del bestemmer hvilken ny informasjon cellen ønsker å bevare. Den bruker samme formel som forget gate, men med andre vekter og bias som formel 26 viser. Sigmoidfunksjonen vil også her gi ut en verdi mellom 0 og 1 avhengig om informasjon bør bevares eller ikke.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{26}$$

Neste del av input gate bruker formel 27 som også tar i bruk x_t og h_{t-1} sammen med egne vekter og bias gjennom en tanh funksjon for å lage en vektor, $\hat{C}_t$. Denne vektoren hjelper til for å regulere nettverket.

$$\hat{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{27}$$

Cell State bruker informasjonen fra forget gate og input gate til å oppdatere den gamle cell state, C_{t-1}, til den nye cell state, C_t. Først multipliseres C_{t-1} med vektoren f_t fra forget gate for å glemme tidligere informasjon som ikke lenger er ansett som viktig. For å oppdatere cell state med ny informasjon bruker den i_t og $\hat{C}_t$ som punktvis adderes med $C_{t-1} * f_t$. Dette er vist i formel 28. Resultatet er en ny cell state, C_t, med oppdatert informasjon fra forrige cell state, C_{t-1}, og ny input x_t.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{28}$$

Output gate bestemmer verdien for neste h_t. Først, i formel 29, brukes x_t og h_{t-1} sammen med vekter og bias gjennom en sigmoidfunksjon for å danne o_t, vist i formel 29.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{29}$$

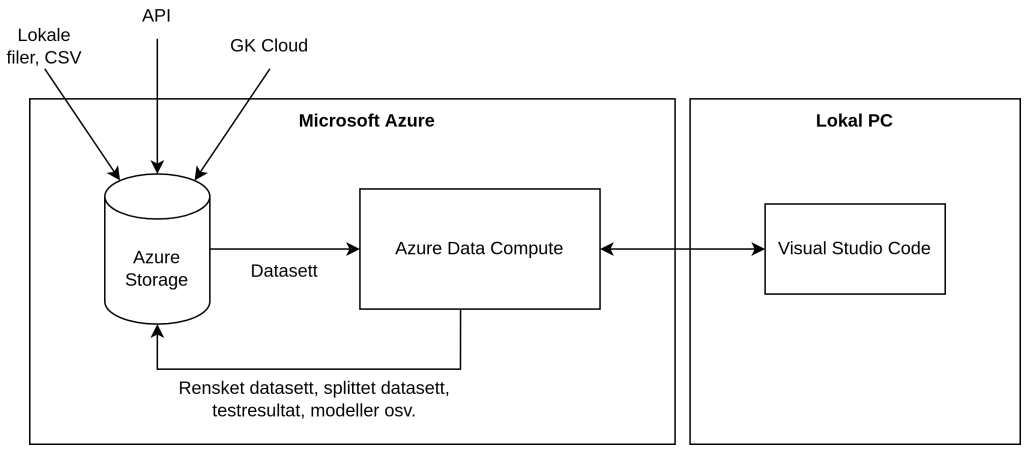
Etter at o_t er kalkulert multipliseres den sammen med tanh funksjonen av den nye cell state, C_t, for å finne h_t som vist i formel 30.

$$h_t = o_t * \tanh\left(C_t\right) \tag{30}$$

Hidden state, h_t, fungerer som korttidshukommelsen til modellen, mens cell state, C_t, blir ofte referert til som langtidshukommelsen.

4 Utviklingsmiljø

Innenfor maskinlæring er det viktig å ha et godt utviklingsmiljø da prosessen for å finne en god modell er avhengig av god struktur i arbeidet og dataen som anvendes og lagres. Siden GK ikke har noe slikt utviklingsmiljø ble det brukt litt tid på å finne et godt system. Etter en del undersøkelse ble Microsoft Azure tatt i bruk. Azure er en skyplattform laget av Microsoft for å erstatte lokal infrastruktur med skybaserte løsninger ved å blant annet leie lagringsplass og datakraft [49]. Inne i Azure ble det leid inn skybasert datakraft, en *compute instance* med 14GB RAM og 4 kjerner som ble koblet opp mot en Azure Datastore. Azure Datastore sin funksjon er å lagre all data i Azure skyen. Det er mulig å bruke en IDE i Microsoft Azure Machine Learning Studio, men da den har noe manglende funksjoner ble det heller tilkoblet Visual Studio Code fra lokal PC til Azure skyen. Visual Studio Code brukes som grensesnittet for å programmere i Python med datakraft fra Azure Data Compute. Figur 13 viser en flyt diagram over utviklingsmiljøet.



Figur 13: Oversikt over utviklingsmiljøet.

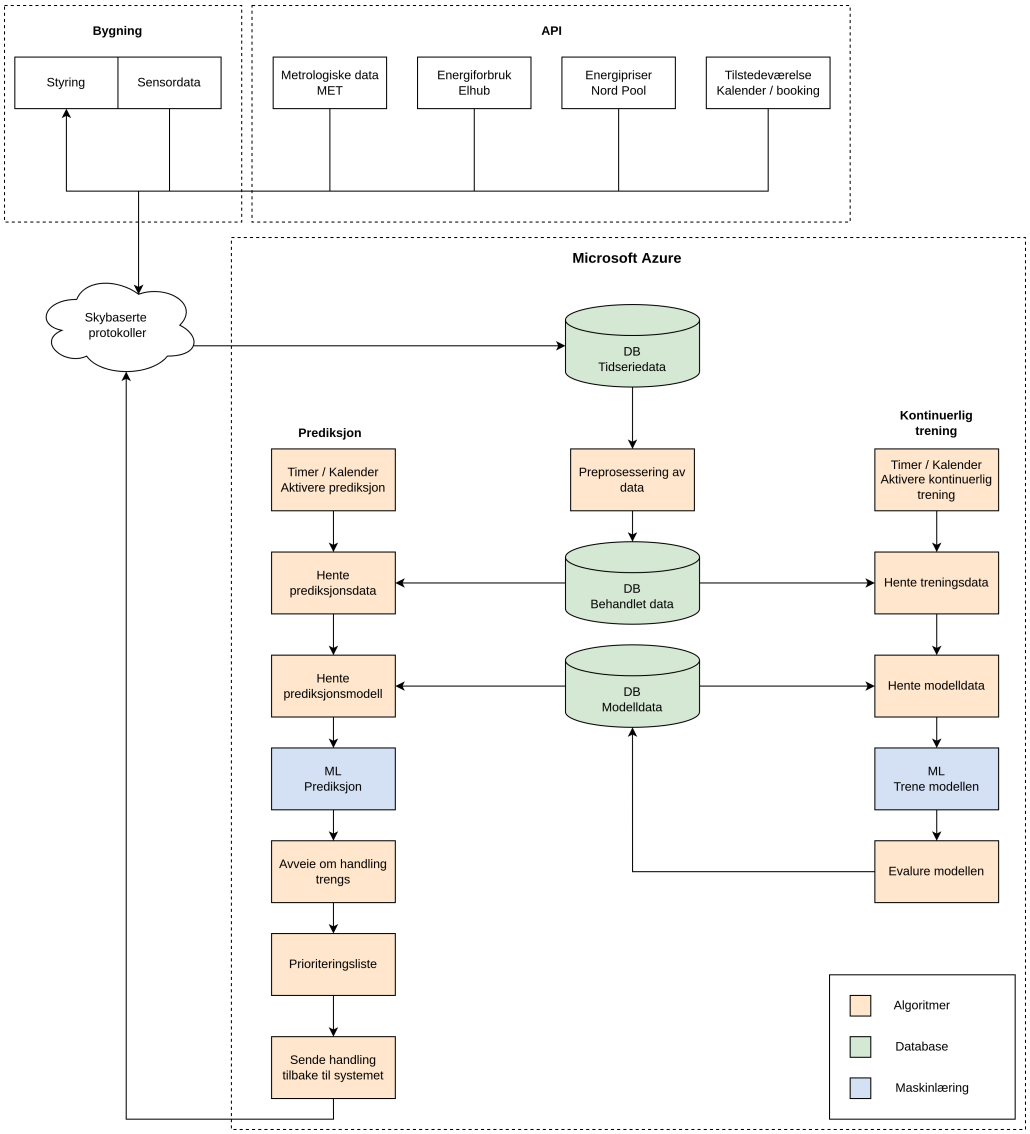
Som programmeringsspråk ble det brukt Python versjon 3.8.1 sammen med Jupyter Notebook. For å gjøre pakke- og versjonshåndtering enklere ble Poetry tatt i bruk i Python. For visualisering av data brukes i hovedsak pakkene Seaborn, Matplotlib og Plotly. I tillegg ble det laget en lokal loggetjeneste som logger hver kjøring av maskinlæringsmodellene. Loggen inneholder tidsstempel, type datasett, hyperparameter, ytelsesmål og tidsforbruk. Tabell 5 viser en oversikt over de mest anvendte pakkene og programvarene.

Pakkenavn	**Versjon**	**Bruksområde**
python3	3.8.1	System
poetry	1.1.13	System
azure-appconfiguration	1.1.1	System
jupyter-notebook	6.4.5	System
pandas	0.25.3	Databehandling
numpy	1.18.5	Databehandling
sklearn	1.7.0	Databehandling
statsmodels	0.10.2	Maskinlæring
tensorflow	2.3.0	Maskinlæring
matplotlib	3.2.1	Visualisering
seaborn	0.11.2	Visualisering
plotly	5.4.0	Visualisering

Tabell 5: Liste over mest anvendte pakker og programvare.

5 Produksjonsmiljø

I forrige kapittel ble det presentert hvordan et utviklingsmiljø for maskinlæring kan utformes og hvorfor det er viktig med god struktur under utviklingen. Når selve modellen er utviklet og skal settes i produksjon er det nødvendig med et miljø å kjøre modellen i; et produksjonsmiljø. Dette vil være spesielt viktig for GK da skalerbarhet av modellene vil være avgjørende med flere tusen bygg og systemer. Et slikt produksjonsmiljø for maskinlæring kalles for *machine learning operations* eller ML Ops. ML Ops er et system tatt i bruk for å modellere, implementere og kjøre i gang maskinlæringsmodeller pålitelig og effektivt [77]. For å illustrere hvilket behov modellene for dette prosjektet vil ha i et produksjonsmiljø er det satt opp en skisse. Figur 14 viser flyten av dataen i ett produksjonsmiljø for modellene i dette prosjektet. Microsoft Azure har ML Ops i sin skybaserte Azure Machine Learning. Det er derfor lagt fokus på detaljene i den stiplede firkanten rundt Microsoft Azure og ikke på hvordan de ulike dataene hentes.



Figur 14: Prosess- og dataflyt i produksjonsmiljøet.

Prosessen vist i figur 14 baserer seg på at det finnes en modell med optimaliserte hyperparameter som er klar for å settes i produksjon. Flyten i figur 14 vil gjelde både for langtidsprediksjons- og korttidsprediksjonsmodellene. Den produksjonsklare modellen vil da bli plassert i en database som lagrer de ulike modellene hver for seg og versjonerer hver modell. Et utrykk som ofte blir brukt i ML Ops er *pipeline*. Pipeline er en serie med operasjoner som påføres dataen fra dataens kilde til dataens endelige mål [77]. Blokkene under prediksjon i figuren vil da være en pipeline. En pipeline aktiveres typisk av en annen algoritme eller en timer. Første blokk under prediksjon er en timer som gir beskjed når pipelinen skal aktiveres og en modell må predikere. Eksempelvis kan langtidsprediksjonsmodellen predikerer hver time, mens korttidsprediksjonsmodellen predikerer hvert femte minutt. Neste steg er å hente dataen den trenger for å predikere samtidig som den henter den aktuelle prediksjonsmodellen. Deretter predikerer modellen x steg fremover før den analyserer prediksjonen og avveier om det er nødvendig å gjøre en handling på systemet. Et eksempel kan være at langtidspredikeringen predikerer veldig høyt forbruk samtidig som algoritmen ser at strømprisen blir dyr neste dag. Algoritmen vil da sende informasjon videre i pipelinen at en handling trengs for å unngå høyt energiforbruk ved høy strømpris. En handling kan da være at algoritmen ser i en prioriteringsliste for å finne ut hvilke effekter den enten kan kutte eller forskyve, avhengig av type modell som kjøres. Til slutt vil den sende en handling ned til systemet på bygget. Når pipelinen er ferdig, vil den stå og vente på neste aktivering.

Det er ønskelig at modellen skal bruke kontinuerlig trening som ble presentert kapittel 3.4.9 [89]. Denne funksjonen gjør at modellen hele tiden er trent på nyeste data. Spesielt viktig vil funksjonen være dersom det allerede er lite treningsdata tilgjengelig. Kontinuerlig trening er også satt opp som en pipeline som aktiveres av en timer. Et forslag her kan være at langtidsprediksjonsmodellen kjører kontinuerlig trening hver uke mens korttidsprediksjonsmodellen kjøre funksjonen hver andre dag. Etter hvert som modellen får mer data tilgjengelig kan det være gunstig å sette et maksvindu på hvor mye data modellene skal trenes på, for eksempel siste to årene. Etter at en modell har kjørt kontinuerlig trening må modellene evalueres før de blir lagt inn som en ny modell med nytt versjonsnummer. Det kan også være smart å implementere en algoritme i pipelinen som sjekker om den nye treningsdataen ser normal ut før modellen trenes, for eksempel ved hjelp av utligger deteksjon [36].

6 Datasett

I de neste delkapitlene tar rapporten for seg dataen som skal brukes for å trene og evaluere modellene. Datautvalget er basert på samtaler med energieksperter for bygg og tilgjengeligheten av data. Enkelte data som er ønskelig å ha med var ikke tilgjengelig under prosjektperioden, for eksempel solstråling. Det har derfor blitt forsøkt å ta ned andre typer data som kan erstatte disse. Det er også mulig at det er tatt ned data som ikke blir brukt i den endelige modellen. Dette blir avgjort i feature-utvelgingen senere i rapporten. Da de to prediksjonsmodellene som skal bygges bruker samme datakilde er det laget felles datasettkapittel.

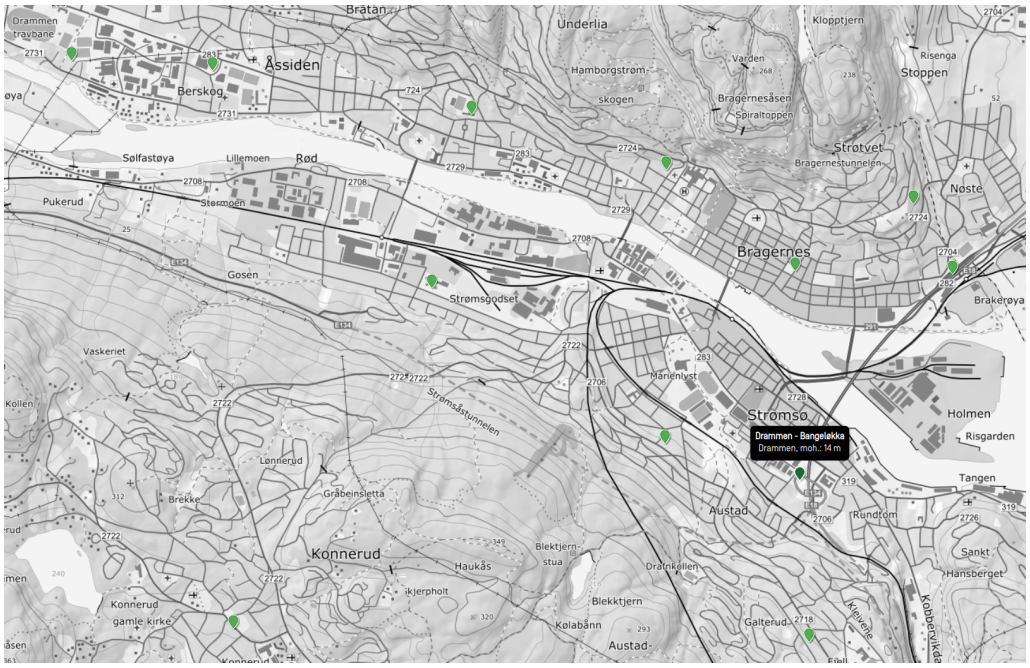
6.1 Datauthenting

Den første delen av datasettet for prediksjonsmodellen er hentet fra GK Cloud gjennom et API. Bangeløkka er valgt som testbygg siden det er et moderne bygg som har god samplingstid på dataen, relativt mye historikk og er godt utrustet med sensorikk. Selv om Bangeløkka er utrustet med flere sensorer ble det oppdaget at historikken på noen av sensorene var mangelfull. Den valgte perioden er fra 1.september 2018 til 31. desember 2020.

Tidsseriedataen som uthentes fra GK Cloud inneholder tre verdier; elektrisk effektforbruk i bygget, utetemperatur og antall personer tilstede. Det er også ønskelig å hente ut meteorologiske data slik som vindhastighet, vindretning, fuktighet, skydekke og nedbør, da dette har vist seg å være viktige features i tidligere arbeid [91][20][97][119].

Det ble videre sett på metoder for å ta ned meteorologiske data gjennom eksterne API, deriblant MET. Først ble det sett på API'et Frost som blant annet tilgjengeliggjør MET sin historiske data. Dette API'et kan gi ut ønskede data, men har bare mulighet til å gi ut daglig, månedlig eller årlig historikk [35]. Samplingstid på en dag er for stort for prediksjonsmodellene som skal utvikles i dette prosjektet og derfor ble det ikke hentet ut noe data fra Frost. En annen mulighet er å hente dataen gjennom Norsk klimaservicesenter (KSS) [57]. KSS tilbyr historikk fra ulike værstasjoner plassert rundt i Norge. Fordelen med KSS er at de tilbyr historikk med timesverdier. Bygget som ble valgt som testbygg, Bangeløkka, er plassert i Drammen, og det ble derfor undersøkt om værstasjoner i Drammen kunne tilby meteorologiske data.

Stedet Bangeløkka har en værstasjon med flere sensorer, men kun lufttemperatur var fullstendig nok. Dette var allerede uthentet gjennom GK Cloud. I tillegg til værstasjonen på Bangeløkka er det plassert ut 12 andre værstasjoner i Drammen. Siden værstasjonene var såpass nærme hverandre, ble det undersøkt om de andre stasjonene kunne gi mer informasjon. Figur 15 viser plasseringen av de ulike værstasjonene, markert i grønn, i forhold til Bangeløkka.



Figur 15: Kart over værstasjonene plassert rundt Bangeløkka [57].

En av de nærmeste stasjonene hadde en nokså fullstendig tidsserie av luftfuktighet som ble hentet ned. Vindstyrke ble sjekket og var tilgjengelige på noen av nabostasjonene litt lenger unna. Da tidsserien fra stasjonene ble hentet ned, ble det oppdaget at dataen var mangelfull. I tillegg er stasjonene plassert høyere i terrenget enn bygget Bangeløkka, som kan føre til ulik påvirkning av vinden. Derfor ble ikke dataen tatt ned.

Det er ønskelig med noe data som kan beskrive solstråling mot bygget da dette kan påvirke energibehovet. En stasjon hadde en variabel; skydekke, som var beskrevet til å variere fra 0-4 for grad av skydekke. Når denne ble tatt ned ble det oppdaget at tidsserien hadde en fast verdi på 1 for hele perioden, og ble derfor ikke brukt videre. En annen variabel som i noen grad beskriver solstråling er nedbør. Det kan antas at det ved nedbør ikke er sol, men det trenger nødvendigvis ikke å være sol ved null nedbør. Først ble nedbør hentet ut av værstasjonene på Bangeløkka, men her var også dataen noe mangelfull. Fordi det å beskrive solstrålingen kan være en viktig feature, ble det utehentet nedbør for alle stasjonene i Drammen, og samlet kan disse trolig gi et fullstendig datasett for nedbør.

Det ble også undersøkt andre meteorologiske data fra værstasjonene rundt Bangeløkka, men de var enten mangelfulle, hadde ikke data for riktig periode, frosne verdier eller manglet sensordata, selv om KSS beskrev at værstasjonene hadde slike sensorer. All data fra KSS ble uthentet med samplingstid på 1 time, som var den laveste tilgjengelige samplingstiden. Uthentingen av dataen var en tidskrevende jobb, da KSS hadde problemer med å hente ut flere datapunkt samtidig for flere stasjoner. Dette førte til at tidsseriene måtte hentes individuelt for hver stasjon.

Det er forventet at energiforbruket er lavere i et bygg dersom der er feriedager. For å hente ut

feriedagene i Norge brukes biblioteket *holidays* i Python [45]. Denne funksjonen gir ut helligdager med navn for gitt tidsperiode. Tabell 6 viser dataen som er uthentet fra API.

Variabel	**Beskrivelse**	**Type**	**Enhet**	**Samplingstid**	**API**
Effektforbruk	Gjennomsnittlig effektforbruk i bygget	Flyt	kW	5 min og 1 time	GK Cloud
Utetemperatur	Temperatur utenfor bygget	Flyt	$°C$	5 min og 1 time	GK Cloud
Tilstedeværelse	Antall personer i bygget	Int	stk	5 min og 1 time	GK Cloud
Luftfuktighet	Relativ luftfuktighet	Tekst	%	1 time	KSS
Nedbør (13 stasjoner)	Nedbør i time	Tekst	mm	1 time	KSS
Helligdager	Navn på helligdag og dato	Tekst	-	-	holidays (Python)

Tabell 6: Oversikt over data uthentet.

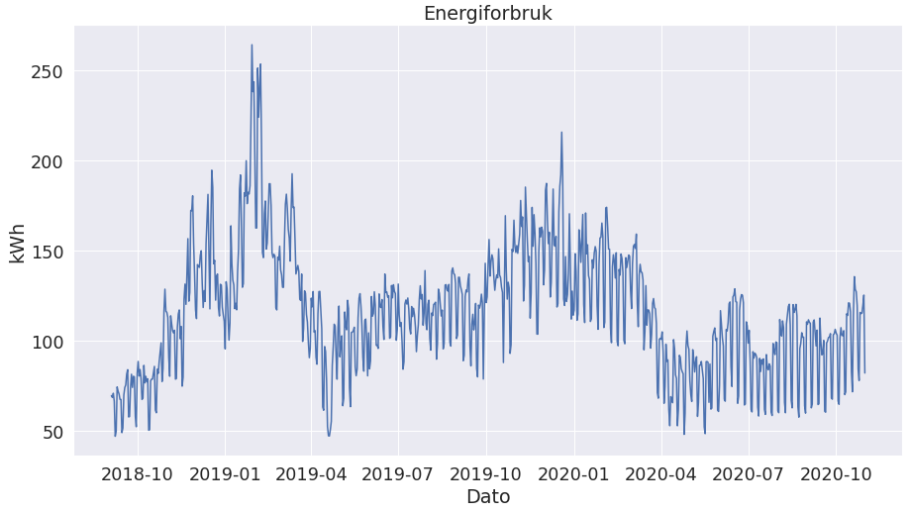
Etter at dataen er uthentet kan det være viktig å få et forhold til dataen ved å analysere den grundig [110]. Dette vil gjøre det enklere å se hvilke nye features som kan være viktig å hente ut. Videre i dette kapittelet skal derfor noe av dataen analyseres nærmere.

Det er viktig å skille mellom energi og effekt. Målet til modellene er at de skal predikere energi, men fra GK Cloud er det uthentet effekt fordi dette var tilgjengelig. Forholdet mellom effekt og energi er gitt i formel 31, der t beskriver hvor lenge effekt, P, har blitt brukt.

$$E = P \cdot t \quad [kWh] \tag{31}$$

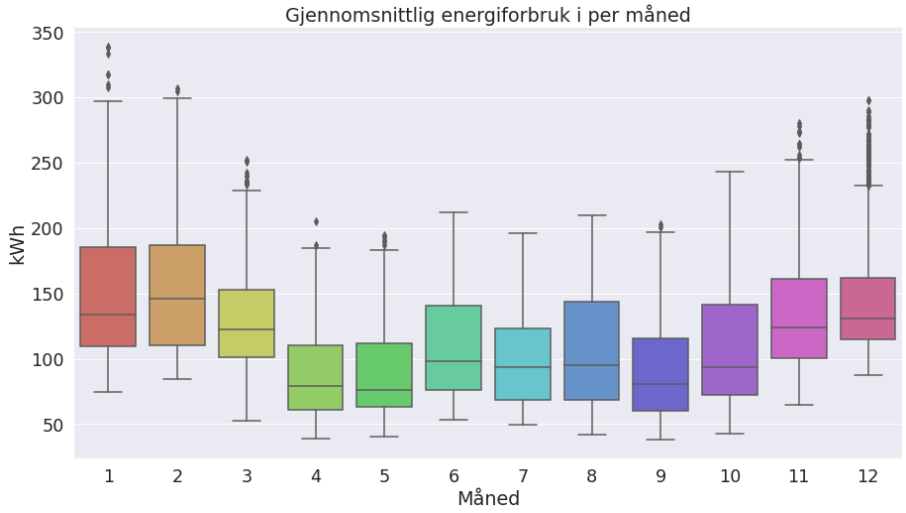
Siden samplingstiden på det ene datasettet er en time kan det etter formel 31 sies at energien er omtrentlig lik det gjennomsnittlige effektforbruket, $E \approx \bar{P}$. Analysen videre i kapittelet bruker datasettet med samplingstid på en time, og anvender derfor energi som benevnelse. Hvordan energien for datasettet med fem minutter samplingstid beregnes blir vist senere.

Energiforbruk er verdien som skal predikeres og er derfor målverdien. Figur 16 viser at energiforbruket er høyest i vintersesongen da bygget må varmes opp, men også sommersesongen har høyt forbruk når det er behov for kjøling.



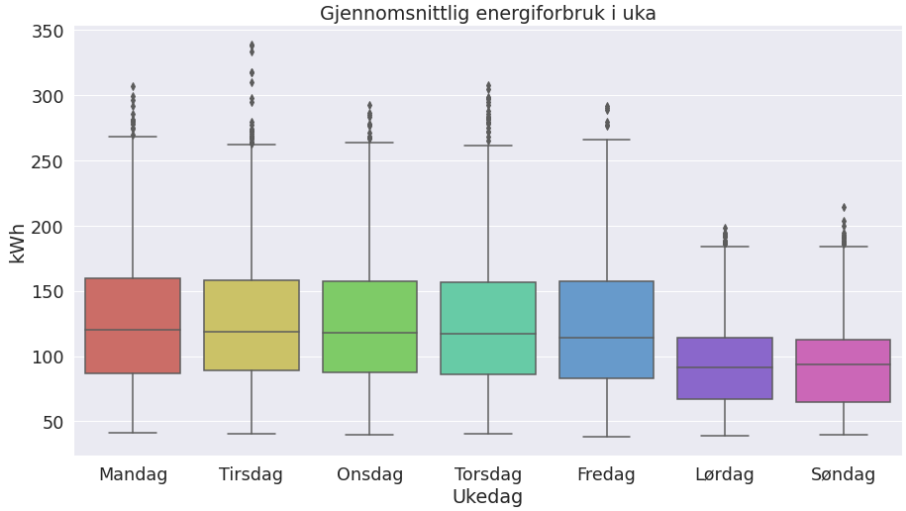
Figur 16: Energiforbruk for valgt periode.

Dersom forbruket analyseres hver måned, viser graf 17 at forbruket er størst i vintersesongen. Sommermåneder har også noe høyt forbruk utenfor juli. Juli har lite energiforbruk på grunn av lite aktivitet på kontorbygget i ferietider.



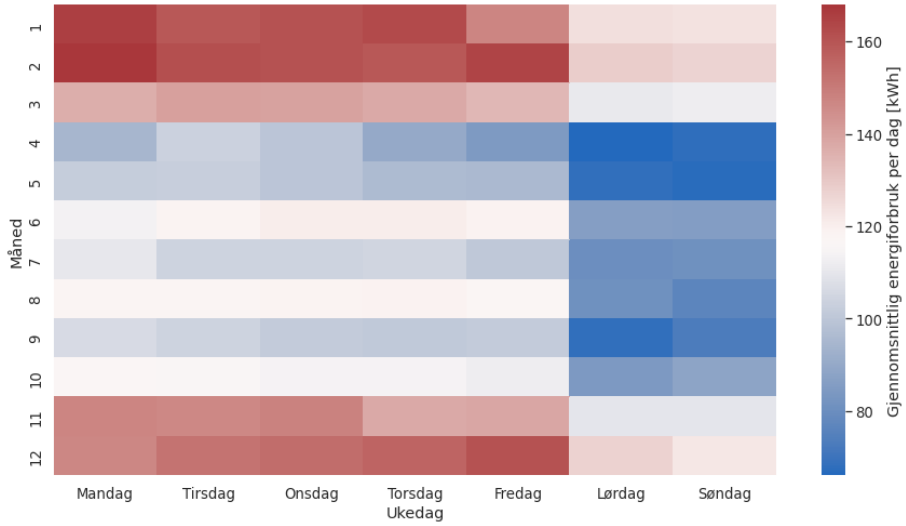
Figur 17: Gjennomsnittlig energiforbruk per måned.

Som forventet viser figur 18 at det gjennomsnittlige forbruket er høyere på ukedager enn i helger.



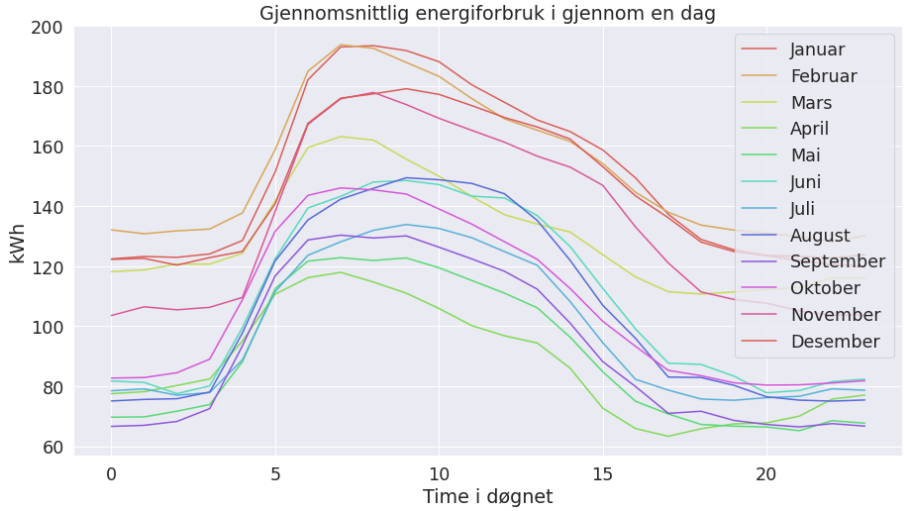
Figur 18: Gjennomsnittlig energiforbruk i uka.

Figur 19 er en sammensetning av figur 17 og figur 18. Denne figuren viser hvilken ukedag i hvilken måned det er høyest energiforbruk. De røde punktene i varmekartet viser de dagene med høyest forbruk.



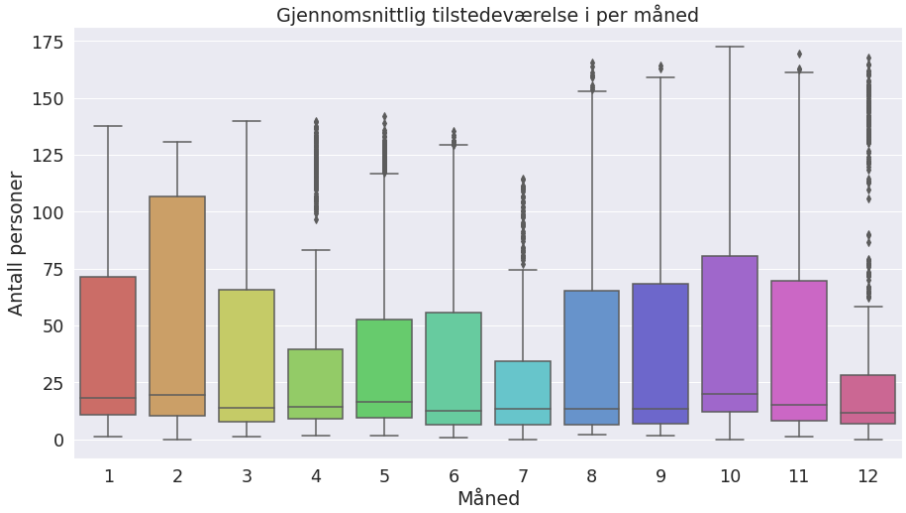
Figur 19: Energiforbruk ukedag i måned.

Figur 20 viser et typisk mønster i forbruket gjennom et døgn. Rundt klokken 05 starter ventilasjonssystemene etter nattmodus. Det er mellom klokken 05 og 15 at det vil være størst sjanse for en effekttopp ifølge denne grafen.



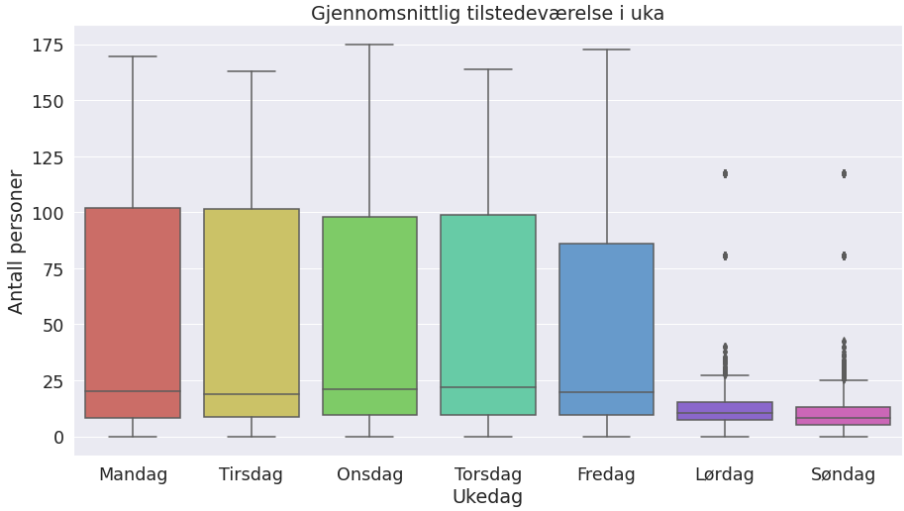
Figur 20: Gjennomsnittlig energiforbruk gjennom en dag.

Tilstedeværelse gir innblikk i aktivitetsnivået på bygget og har en god korrelasjon med forbruket. Figur 21 viser hvordan ferietider påvirker tilstedeværelsen i bygget. Påske-, sommer- og juleferie er typisk i månedene april, juli og desember, og dette påvirker aktiviteten i bygget.



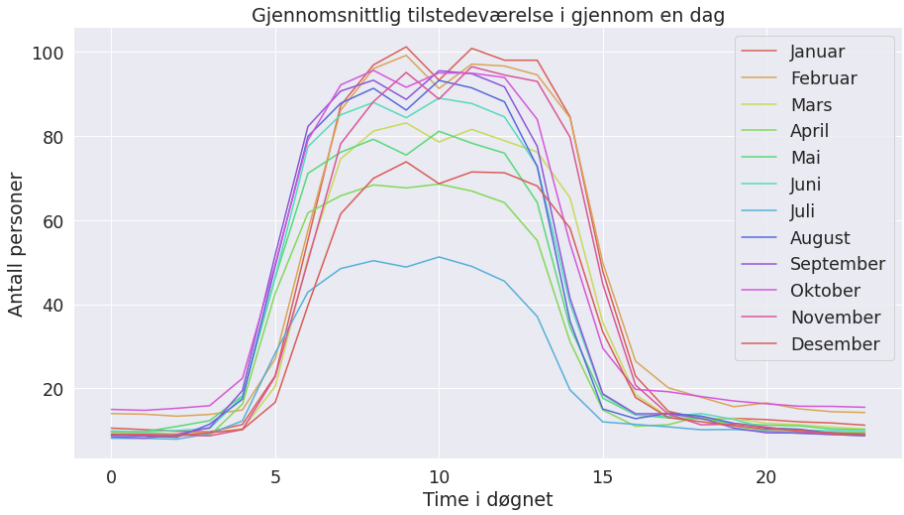
Figur 21: Gjennomsnittlig tilstedeværelse i per måned.

Tilstedeværelsen i likhet med forbruket er også veldig lavt i helgene slik figur 22 viser.



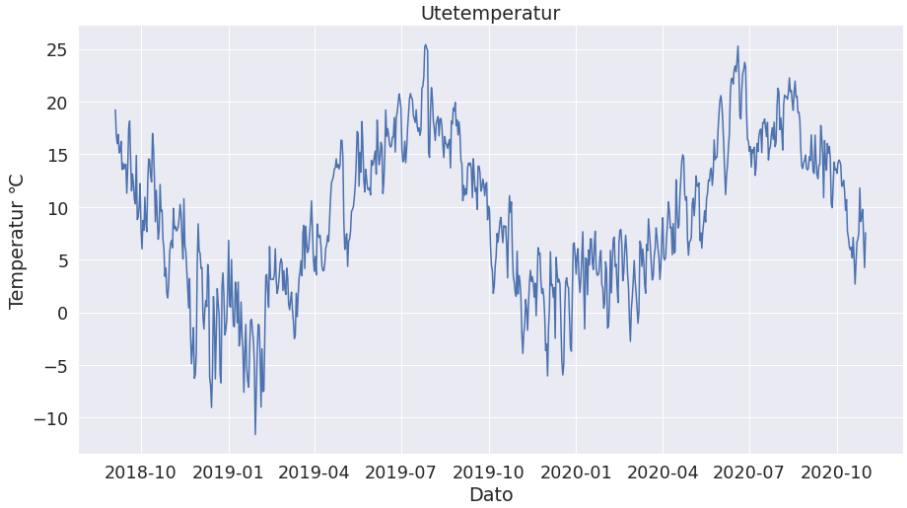
Figur 22: Gjennomsnittlig tilstedeværelse i uka.

Figur 23 viser tilstedeværelse i bygget opp mot tid i døgnet. Sammenlignes figur 23 og 20 er det korrelasjon mellom tilstedeværelse og energiforbruk.



Figur 23: Gjennomsnittlig tilstedeværelse gjennom en dag.

Utetemperatur for Bangeløkka i perioden som er valgt er vist i figur 24. Utetemperatur har sesongvariasjon gjennom et år, der måneden beskriver sesongen. Naturlig vil denne dataen være viktig for modelleringen.



Figur 24: Utetemperatur for gitt periode.

Fra figurene vist i analysen kan det konkluderes med at tidsstempelet på dataen vil være en viktig feature i datasettet. Figurene viser at både tid i døgnet, hvilken ukedag og hvilken måned er viktig informasjon for å predikere energiforbruket.

6.2 Datarensing

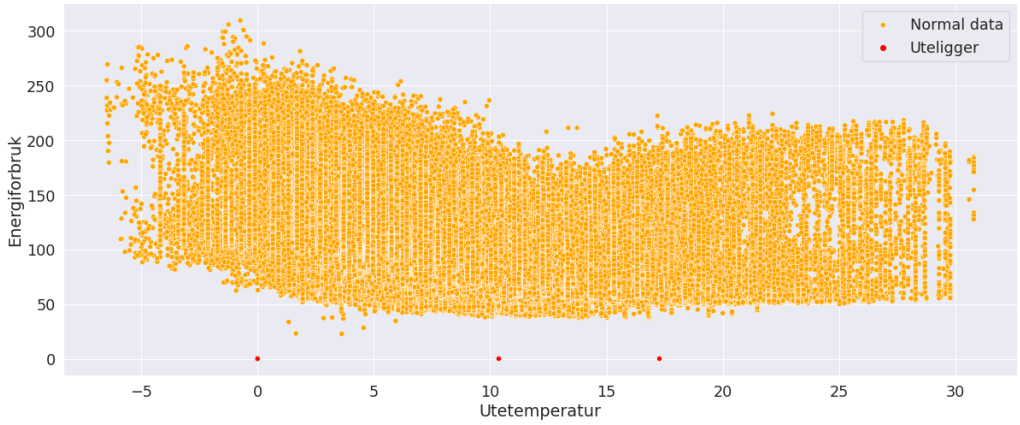
Datarensing er ofte den største tidstyven innenfor modellering av AI [24]. Det er mulig å se over datasettet i en graf for å avgjøre om datasettet trenger noen form for rensing, men for å unngå mye manuelt arbeid ble det laget noen hjelpefunksjoner. Først ble det laget en funksjon som kontrollerer at datasettet ikke inneholder NaN-verdier, har rett datatype og at datasettet ikke inneholder hull. NaN-verdier er uheldig for modelleringen. Først og fremst for at de ikke gir noen verdi eller feil verdi i analysen, men det fører også til at noen modeller, slik som nevrale nettverk, kan få problemer under treningen [33][126]. Ved behandling av tidsseriedata er det viktig å ha kontroll på hvordan NaN-verdier behandles. Ofte bruker modeller innenfor tidsserieprediksjon sekvenser mellom $t - S$ og t når de predikerer $\hat{y}_{t+1}$. Dersom det fjernes noen verdier mellom t og $t - S$ vil dette gi en sekvens som er mangelfull og fører til at sekvensen blir ugyldig. Det er derfor en bedre løsning innenfor tidsserieprediksjon å erstatte NaN-verdier [58]. En annen mulighet kan være å slette hele sekvensen som inneholder NaN-verdier, men dette kan føre til at mye av dataen blir ekskludert dersom det er flere enkeltpunkter som ødelegger mange sekvenser [48].

Hull i datasettet sjekkes ved å finne samplingstiden (T) og dermed iterere gjennom tidsstemplene for å sjekke om $t_2 - t_1 = T$, hvis ikke blir det avvik. Frosne verdier er ikke uvanlig i byggautomasjonsbransjen. Dette kan komme av at en sensor har låst seg på en verdi eller at signalet har hengt seg opp i selve PLS'en. For korte tidsperioder kan det være vanskelige å oppdage frosne verdier visuelt, derfor ble det laget en funksjon for å finne disse verdiene. Funksjonen går gjennom

alle features i datasettet for å først sjekke om verdien er boolsk (0-1) eller en flytverdi. Deretter sjekker funksjonen om verdiene har samme verdi, gitt antall verdier etter hverandre. Dersom den har mange like verdier etter hverandre vil funksjonen gi ut et avvik. Funksjonen sjekker bare flytverdier.

Før dataen sendes inn i modellene er det gunstig å sjekke om dataen inneholder utliggere [36]. Utliggere er definert som unormale verdier i forhold til det gitte datasettet [32]. DBSCAN implementeres for å finne utliggere i datasettet da denne modellen har vist et godt resultat tidligere som utliggerdeteksjon [115][18]. DBSCAN står for *Density-based spatial clustering of applications with noise* og er en dataklynge-algoritme lagt frem av Martin Ester, Hans-Peter Kriegel, Jörg Sander og Xiaowei Xu i 1996 [28]. Algoritmen bruker egenskaper som tetthet i datasettet for å finne riktige klynger. I dette tilfellet er det mer interessant å finne punkter som ikke klynger seg, da disse mest sannsynlig er utliggere. I denne problemstillingen må det med forsiktighet klassifiseres hva som er utliggere i datasettet. For eksempel kan unormalt høyt energiforbruk bli klassifisert som utligger. Dette er svært uheldig da dette er viktig data for modellen, siden systemets oppgave skal unngå høyt energiforbruk ved å predikere det unormalt høye energiforbruket. På bakgrunn av dette settes det derfor opp en deteksjonsmodell som detekterer de punktene som med stor sannsynlighet er utligger. Samtidig blir en manuell analyse gjort av resultatet. Teorien og implementasjonen er lik arbeidet som er gjort i fordypningsprosjektet, *Vannlekkasjedeteksjon ved bruk av anomaly detection* [125].

Når unormale verdier slik som frosne, NaN- og utliggerverdier har blitt funnet, må de erstattes på en god måte. Det ble gjort en grundig jobb og undersøkelse av ulike bygg for å finne godt datasett før prosjektet, og allerede da var det fokus på at datasettet ikke skulle inneholdt hull, NaN- eller frosne verdier. Forarbeidet som ble gjort gjorde at det ikke ble funnet noen NaN-verdier, hull i datasettet eller frosne verdier. Etter å ha kjørt DBSCAN ble det funnet noen utliggere. Figur 25 viser et eksempel der DBSCAN har blitt kjørt på utetemperatur og effekt. I figur 25 er tre av punktene definert som utliggere.



Figur 25: Resultat av utliggerdeteksjon med DBSCAN på energiforbruk og utetemperatur.

For å erstatte verdiene finnes det flere teknikker. Dersom det er enkeltstående punkter, f.eks $x_2, x_{135}, x_{222}...$ som må erstattes, kan de for eksempel erstattes med interpolering [32]. Dersom en større del av sammensatte punkter mangler, f.eks. $[x_{77}...x_{155}]$ bør det undersøkes andre teknikker enn interpolering, da det vil ødelegge sesongen i dataen. En mulighet er å sette opp prediksjonsmodeller for å erstatte slik data, for eksempel SARIMA [33]. I dette tilfellet var det enkeltstående punkter som måtte erstattes slik at interpolering ble brukt for å erstatte utliggere.

6.3 Feature-uthenting

Fra datasettet som er uthentet, beskrevet i kapittel 6.1, er det mulig å hente flere features som kan brukes som variabler i prediksjonen. Siden det er ønskelig å teste ut forskjellige typer prediksjonsmodeller vil det være lurt å gjøre noe manuell feature-uthenting. Dersom oppgaven bare hadde inneholdt modeller som bruker dyp læring kan noe av feature-uthentingen være unødvendig, da slike typer modeller kan gjøre noe feature-uthenting selv [23].

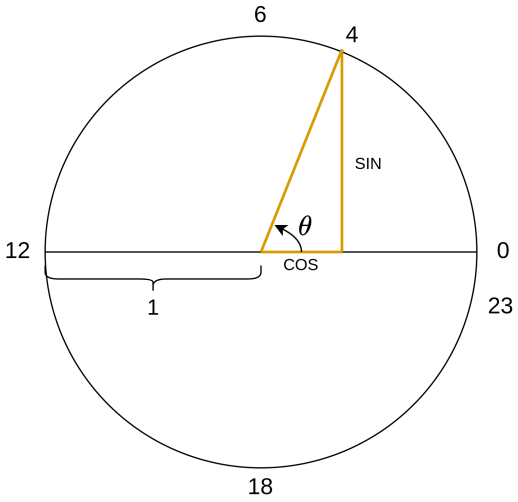
Tidsstempeldata: Funksjonen *holidays* gir ut en liste med dato og navn på helligdagene for en gitt tidsperiode. Ved bruk av andre funksjoner i Python ble det satt inn en kolonne i datasettet som gir 1 for helligdag og 0 for ikke helligdag.

For å gi modellen en indikasjon på at det for eksempel helg eller arbeidsdag, blir datostemplingen i datasettet utnyttet. Ved analysen av datasettet, kapittel 6.1, ble det observert at det er høyere forbruk i ukedagene enn i helgene. Det er derfor nyttig å gi modellen beskjed på hvilken av samplene som gjelder for ukedag og helg. Dette gjøres ved å legge til en ny kolonne i datasettet som har 1 for arbeidsdag og 0 for helg. I tillegg til fridager på helg, er det også fri andre dager; slik som helligdager. Innhenting av helligdager ble vist i kapittel 6.1, og sammen med den nye kolonnen som nettopp ble opprettet ble det laget en feature for å definere arbeidsdag eller fridag. Helligdager ble fjernet fra datasettet etterpå.

Figurene i kapittel 6.1 viste også forskjell i forbruket i forhold til hvilken tid på døgnet det er. Derfor legges det til en variabel som forteller hvilken time og minutt når den aktuelle verdien er samplet med en tallverdi fra 0-23 for time og 0-59 for minutt. I tillegg til å vise hvilken time det er, legges det til en variabel som beskriver om det er arbeidstid eller ikke i form av 0 og 1. Arbeidstiden er satt til å være 07:00-16:00. Det legges også til en ekstra variabel for dag i uken, dag i måneden og hvilken måned, med en tallverdi 1-7 for ukedag, 1-(28-31) for dag i måneden og 1-12 for måned.

En annen metode for å representere tidsstempel-features er å bruke sykluser. Ved å bruke kategoriske features for time, slik som 0-23, kan modellen få problemer med å forstå at det er seks timer mellom 09:00 og 03:00. Dette kommer av at modellen ikke nødvendigvis forstår at 23:00 og 00:00 er like nærme hverandre som 22:00 og 23:00 da den numeriske forskjellen på den første er 23, men den siste er 1. Det å bruke tidsstempel-features som en kategori, slik som det er gjort frem til nå, trenger nødvendigvis ikke å være dårligere, men krever mer treningsdata [65]. En metode for å

unngå denne problematikken er å representere disse features som sykluser med x og y koordinater i en sirkel [34]. Dette gjøres ved å bruke trigonomiske funksjoner som sinus og cosinus. Figur 26 viser, ved hjelp av et koordinatsystem og enhetssirkelen, hvordan time 23 og 00 kan beskrives som like nære verdier som 22 og 23.



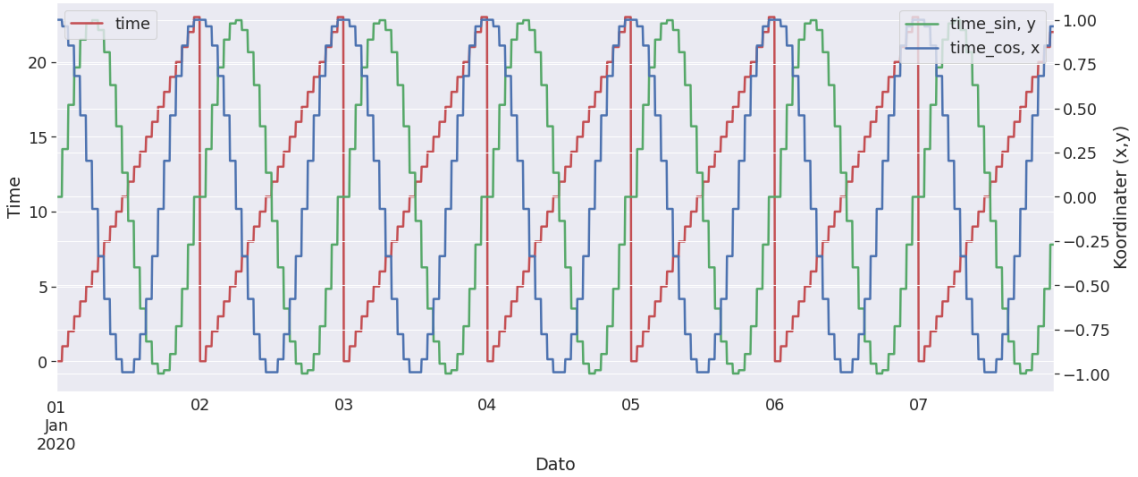
Figur 26: Koordinatsystem som beskriver tidssempelet med x og y koordinater.

For å gjøre denne omformingen brukes formlene 32 og 33. Her er det vist hvordan time feature blir gjort om fra kategorisk til syklus feature. Dette fører til at en feature blir til to.

$$y = time_{sin} = sin(x^{time} \cdot 2\pi \frac{1}{24}) \tag{32}$$

$$x = time_{cos} = cos(x^{time} \cdot 2\pi \frac{1}{24}) \tag{33}$$

Figur 27 viser feature *time* sammenlignet med $time_{sin}$ og $time_{cos}$. Figuren viser at $time = 0$ er representert av $x = 1$ og $y = 0$ som forklarer posisjonene i sirkelen fra figur 26. Dette gjøres for alle tidsstempel-features. Alle features uthentet av tidsstempelet er vist i tabell 7.



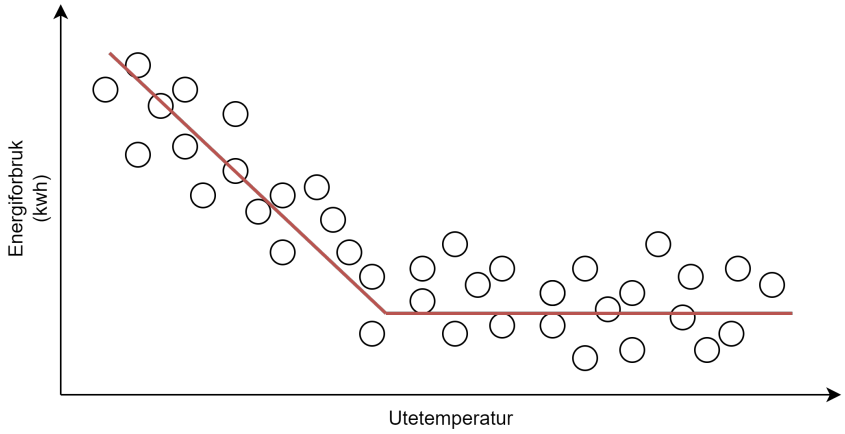
Figur 27: Graf av time feature (venstre y-akse) sammen med sinus og cosinus for time feature (høyre y-akse).

Variabelnavn	**Beskrivelse**	**Verdi**
minutt	Hvilken minutt i time	0 - 59
time	Hvilken time i døgnet	0 - 23
ukedag	Hvilken dag i uken	1 - 7
måned	Hvilken måned i året	1 - 12
månedsdag	Hvilken dag i måned	1 - (28-31)
arbeidsdag	Arbeidsdag eller ikke	0 - 1
arbeidstid	Arbeidstid eller ikke	0 - 1
minutt_{sin}	Sinusfunksjon med periode time	-1 - 1
minutt_{cos}	Cosinusfunksjon med periode time	-1 - 1
time_{sin}	Sinusfunksjon med periode dag	-1 - 1
time_{cos}	Cosinusfunksjon med periode dag	-1 - 1
ukedag_{sin}	Sinusfunksjon med periode uke	-1 - 1
ukedag_{cos}	Cosinusfunksjon med periode uke	-1 - 1
månedsdag_{sin}	Sinusfunksjon med periode måned	-1 - 1
månedsdag_{cos}	Cosinusfunksjon med periode måned	-1 - 1
måned_{sin}	Sinusfunksjon med periode år	-1 - 1
måned_{cos}	Cosinusfunksjon med periode år	-1 - 1

Tabell 7: Oversikt over features uthentet fra tidssempelet.

Energi/Temperatur data: Det er mulig å gi noe prediksjon på forbruket dersom den fremtidige utetemperaturen er gitt ved å se på tidligere forhold mellom energiforbruk og utetemperatur. Energiforbruket har en korrelasjon med utetemperatur. Dette blir ofte visualisert innenfor energido-

menet som energi-/temperaturkurve også kalt ET-kurve. ET-kurve er altså en grafisk framstilling som viser energiforbruket til en gitt utetemperatur [50]. Figur 28 viser et eksempel på en ET-kurve. Her er det tatt en lineær regresjon frem til knekkpunktet. Knekkpunktet forteller ved hvilken utetemperatur det ikke lenger er nødvendig med oppvarming. ET-kurver er ofte oppbygd av to-tre sammensatte regresjonslinjer, denne figuren viser to linjer.

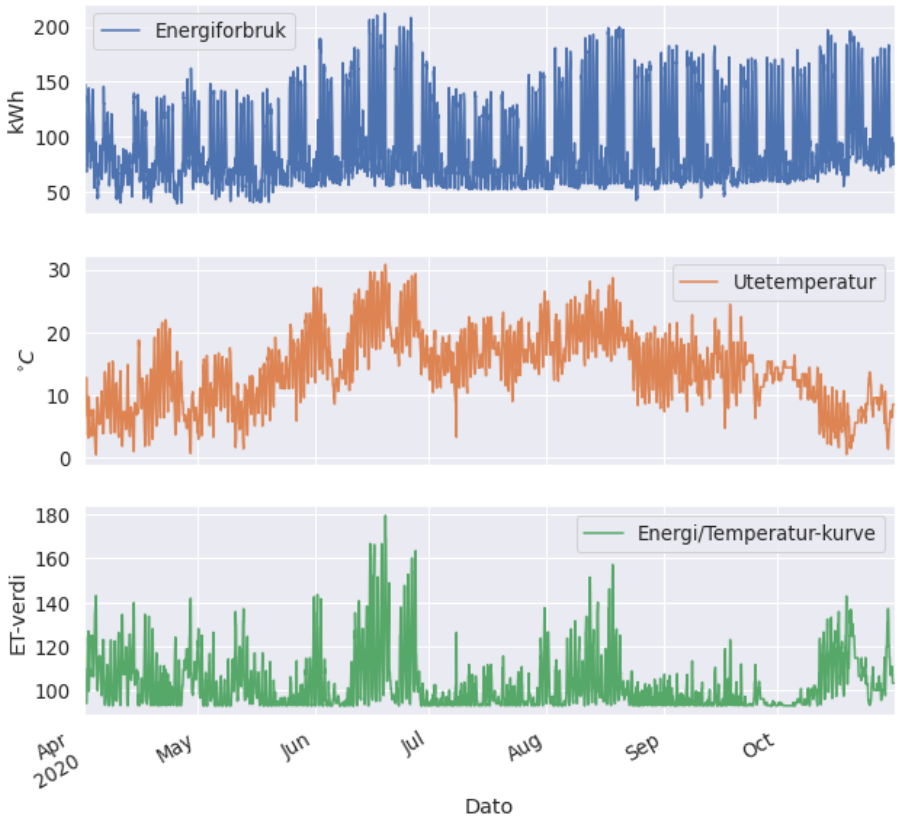


Figur 28: Eksempel på ET-kurve.

Informasjonen som ET-kurven gir kan være et viktig input til modellen da det gir en form for prediksjon av energiforbruket ved gitt utetemperatur. ET-kurven vil lage en funksjon mellom utetemperatur og energiforbruket. Polyfit i Python lager en polynomfunksjon ut av x og y slik som formel 34 viser. I dette tilfellet er x utetemperatur og y energiforbruk.

$$y(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \cdots \tag{34}$$

I dette datasettet er det ingen lineær sammenheng mellom utetemperatur og energiforbruket, da en lav utetemperatur vil øke energiforbruket, men det vil også en høy utetemperatur. En polynomfunksjon med grad tre vil beskrive dette forholdet. Figur 29 viser hvordan den nye variabelen ET-verdien korrelerer mer med energiforbruket en hva utetemperatur gjør. Legg merke til at ET-verdien øker både ved lav og høy utetemperatur, altså ved oppvarming og nedkjøling av bygget. Ut fra denne grafen kan det også observeres at bygget bruker minst energi rundt ca. 17 $°C$



Figur 29: Resultat av ET-kurve.

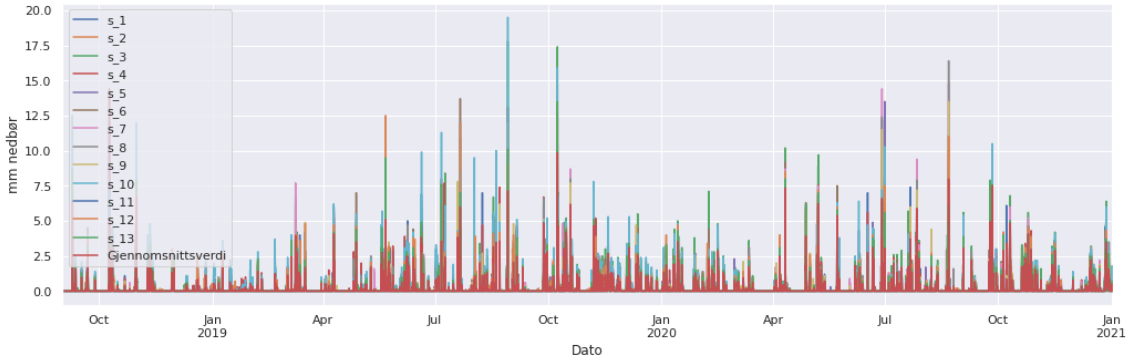
Lag temperatur: En annen egenskap som er vanlig å hente ut av datasettet er forskjøvede verdier heretter kalt lag-verdier. Dette er verdier som er forskjøvet gitte antall samplingssteg. Utetemperaturen vil ikke direkte påvirke øyeblikksforbruket, men gjennom en tidsforsinkelse, da det tar tid før utetemperaturen påvirker innetemperaturen, som deretter påvirker energiforbruket. Dersom det tar to timer før innetemperaturen påvirkes, bør det legges inn en lag-verdi av utetemperaturen som er forskjøvet to timer fremover slik at den har en sterkere korrelasjon i forhold til energiforbruket. Det finnes flere metoder for å finne riktig lag-verdi [111]. Innenfor denne anvendelsen vil det være naturlig å legge på lag-verdier på utetemperatur da det er en variabel som påvirker energiforbruket med en tidsforsinkelse. Etter samtale med domeneeksperter ble det valgt å legge inn utetemperatur med lag-verdier fra en til fem timer. Det er mulig å legge inn lag-verdier for alle andre features også, men i første omgang testes det på temperatur da den er antatt å være den mest aktuelle meteorologiske feature.

Nedbør: En feature som viser om det er nedbør eller ikke, er ikke antatt å være en viktig feature i seg selv for denne problemstillingen. Men som nevnt tidligere er det ønskelig å bruke denne feature som erstatning for mangelfull data på solstråling. Verdien 1 på nedbør vil si at det regner og det mest sannsynlig ikke er sol. 0 på nedbør vil si at det er oppholdsvær, men sier ikke nødvendigvis at det er sol. For å danne en slik feature kreves det litt arbeid med datasettet fra KSS. Det ble i kapittel 6.1 beskrevet at det ble hentet nedbørsdata fra total 13 forskjellige værstasjoner i samme

by. Tanken bak å ta ned data fra alle stasjonene var at de sammen kunne settes opp som et fullstendig datasett som kunne beskrive nedbøret i området. For å vise noe av arbeidet som kreves i feature-uthenting er det satt opp en mer detaljert liste nedenfor for denne feature.

1. Nedbør ble uthentet gjennom KSS fra 13 stasjoner i en CSV-fil.
2. CSV-filen inneholder datasettene fra alle stasjonene lagt nedenfor hverandre radvis. Dette er ikke optimalt da det er ønskelig med en kolonne som har tidsserie fra t_0 til t_N, der N er antall samples i datasettet og en kolonne for hver stasjon. Datasettet ble derfor delt opp i 13 tabeller, en for hver stasjon.
3. Etter oppdeling ble det oppdaget at en av stasjonene hadde NaN-verdier og en annen stasjon hadde veldig få verdier. Disse ble fjernet.
4. Av de resterende stasjonene hadde alle forskjellige lengde, men forskjellene var små. Den som hadde størst N ble satt som hovedramme.
5. En funksjon ble laget som går igjennom alle tabellene.
   (a) Fjerner alle kolonnene bortsett fra tidsstempelet og verdien i tabellen.
   (b) Siden tidsstempelet er tekst, blir dette gjort om til datatypen tidsstempel.
   (c) Tidsstempelet blir satt som indeks i tabellen.
   (d) Endrer navn på kolonnen med verdiene til hvilken stasjon det gjelder, f.eks: nedbør_2 for stasjon to.
   (e) Siden verdien også kommer som tekst blir komma byttet ut med punktum for å klargjøre for neste steg.
   (f) Verdiene blir gjort om til numeriske verdier.
   (g) Fjerner desimaler slik at verdiene har maks to desimaler.
6. Etter at hver tabell har gått igjennom funksjonen ovenfor ble alle tabellene satt sammen med tidsstempelet til hovedrammen som indeks. Dette resulterer i en tabell med kolonne for hver stasjon og tidsstempel som indeks i tabellen.
7. En ny kolonne ble laget der gjennomsnittet av alle kolonnene fra de ulike værstasjonene ble funnet på hver sampel. Gjennomsnittet regner ikke med NaN-verdier i datasettet.

Prosedyren ovenfor viser hvordan alle værstasjonene er brukt til å få et fullt datasett med nedbør. Figur 30 viser hvordan de ulike stasjonene sammen bygger opp et fullstendig datasett. Gjennomsnittsverdiene av alle værstasjonene som ble funnet vil ikke beskrive det korrekte nedbøret, men vil gi noe informasjon om det er nedbør i området eller ikke. I tillegg til å ha en feature som viser den gjennomsnittlige nedbøren, ble det laget en feature som sier om det er nedbør eller ikke.



Figur 30: mm nedbør av hver stasjon sammen med gjennomsnittsverdien for alle stasjoner.

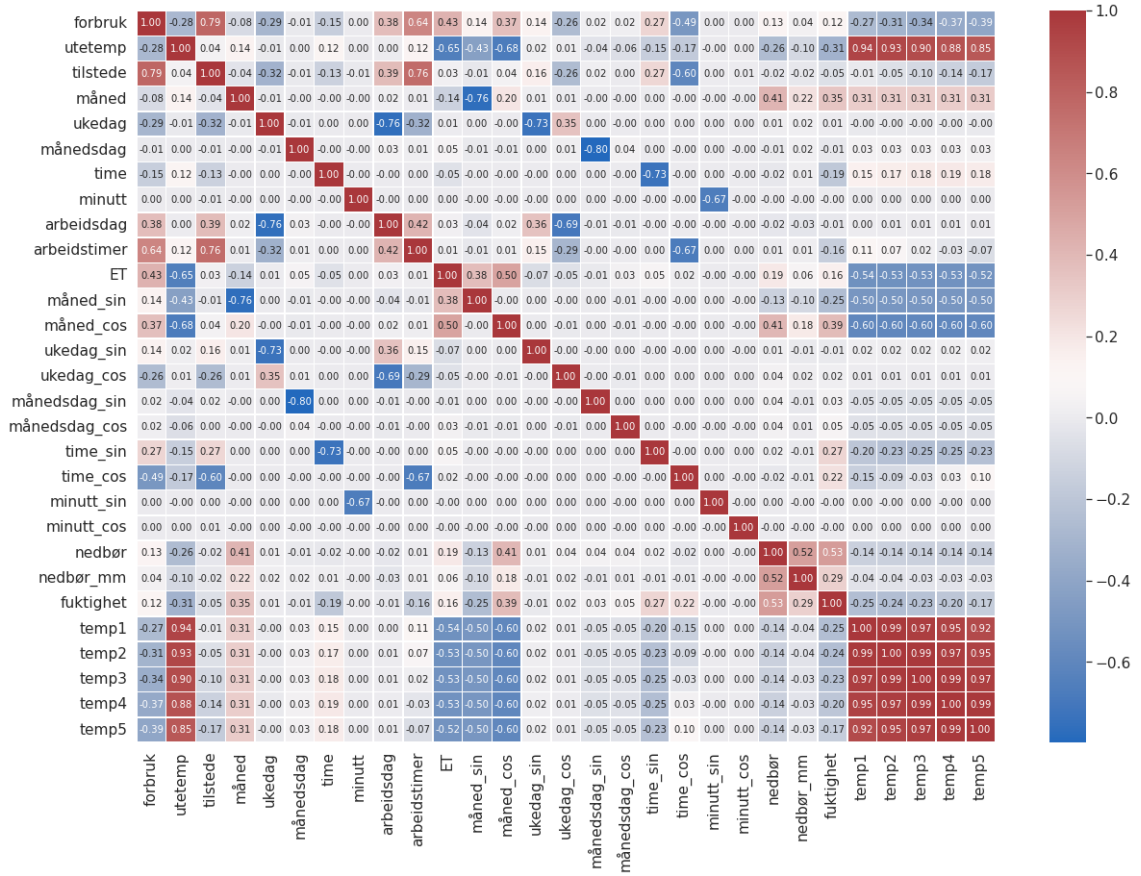
Resultatet av kapittel 6.1, 6.2 og 6.3 er en tabell med tidsstempel som indeks og features lik tabell 8.

Variabelnavn	Beskrivelse	Verdiområde
forbruk	Energiforbruk i bygg	-
utetemp	Utetemperatur for bygg	-
tilstede	Tilstedeværelse i bygg	-
minutt	Hvilken minutt i timen	0 - 59
time	Hvilken time i døgnet	0 - 23
ukedag	Hvilken dag i uken	1 - 7
måned	Hvilken måned	1 - 12
månedsdag	Hvilken dag i måned	1 - (28-31)
minutt_sin	Sinusfunksjon med periode time	-1 - 1
minutt_cos	Cosinusfunksjon med periode time	-1 - 1
time_sin	Sinusfunksjon med periode dag	-1 - 1
time_cos	Cosinusfunksjon med periode dag	-1 - 1
ukedag_sin	Sinusfunksjon med periode uke	-1 - 1
ukedag_cos	Cosinusfunksjon med periode uke	-1 - 1
månedsdag_sin	Sinusfunksjon med periode måned	-1 - 1
månedsdag_cos	Cosinusfunksjon med periode måned	-1 - 1
måned_sin	Sinusfunksjon med periode år	-1 - 1
måned_cos	Cosinusfunksjon med periode år	-1 - 1
arbeidsdag	Arbeidsdag eller ikke	0 - 1
arbeidstid	Arbeidstid eller ikke	0 - 1
ET	ET verdi for temperatur	-
temp1	Utetemperatur forskjøvet 1 time	-
temp2	Utetemperatur forskjøvet 2 time	-
temp3	Utetemperatur forskjøvet 3 time	-
temp4	Utetemperatur forskjøvet 4 time	-
temp5	Utetemperatur forskjøvet 5 time	-
fuktighet	Relativ luftfuktighet	-
nedbør_mm	Gjennomsnittlig nedbør i mm	-
nedbør	Nedbør eller ikke	0 - 1

Tabell 8: Oversiktstabel over all dataen uthentet.

For å analysere dataen opp mot hverandre kan et korrelasjonsplot brukes. Et korrelasjonsplot viser korrelasjonen mellom kvantitative variabler med en korrelasjonskoeffisient som viser hvor sterk korrelasjon det er mellom variablene [58]. Plottet er et godt verktøy for å få en bedre forståelse av dataen og kan også gi en indikasjon av viktigheten på ulike features. Figur 31 viser korrelasjonsplot mellom alle datapunktene. Figuren viser for eksempel at feature ET har høyere korrelasjon med

energiforbruket en hva utetemperatur har.



Figur 31: Korrelasjonsplot mellom all dataen.

6.4 Skalering og splitting

Før dataen skal anvendes er det nesten alltid en fordel å skalere dataen [9]. Skaleringen gjør at alle verdiene for ulike features blir plassert innenfor samme skala. Når skaleringen er utført vil alle features ha samme maksimum- og minimumverdier. Dette gjøres for å unngå at magnituden av ulike features påvirker modellen ulikt [1]. For eksempel i denne sammenhengen, med data fra sensorer vil forskjellige enheter (kWh, $°C$) ha forskjellige maks- og minimumsverdi.

En vanlig måte å skalere datasettet er ved hjelp av *Z-score* [1]. Z-score skalerer datasettet slik at alle features har middelverdi 0 og standardavvik 1. Skaleringen gjøres ved å følge formlene nedenfor der x_i er hver sampel og $\bar{x}$ er gjennomsnittsverdien [125]. Etter modellene har predikert, er det normalt å skalere tilbake verdiene for å få rett skalering på ytelsesmålene.

$$Middelverdi: \mu = \frac{1}{N}\sum_{i=1}^{N}(x_i) \tag{35}$$

$$Standardavvik : \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{36}$$

$$Skalering : z = \frac{x - \mu}{\sigma} \tag{37}$$

Etter skaleringen ble datasettet splittet i trenings-, valideringsog testsett. Innenfor maskinlæring er ønskelig å splitte datasettet slik at settene er mest mulig homogene [58]. *Cross validation* er en vanlig valideringsmetode for maskinlæringsmodeller. Cross validation krever at datasettet deles opp i k antall sett, der settene omstokkes og blir til slutt plukket ut til trenings-, valideringsog testsett [7]. I tidseriedata er ikke dette nødvendigvis en god fremgangsmåte, da det kan føre til at modellen trenes på data i fremtiden og testes på data i fortiden [8]. I tillegg krever noen modeller en datastruktur med tidssemplene i rekkefølge, noe som gjør at tilfeldig omstokking av settene ikke er gunstig. En anbefalt metode for validering av modellen som ikke krever å stokke om på datasettet er *Out Of Sample validation*; OOS [19]. OOS gjør at den første andelen av datasettet blir brukt til trening, deretter validering og til slutt testsett.

Prosentandelen for hvert sett ble satt til 70% trening, 15% validering og 15% test som er en av de anbefalte splittingene av datasettet [27].

7 Testforbredelser

Før prosessen med valg av modell starter, er det viktig å definere en fremgangsmåte for å velge ut den beste modellen. I tillegg er det også viktig å velge en god valideringsmetode og en effektiv og presis metode for optimalisering av hyperparameterene. Alt dette er viktig for at den endelige avgjørelsen skal være rettferdig i forhold til at alle modellene har blitt testet likt, og har hatt samme forutsetninger til å vise sin ytelse. Samtidig som det er ønskelig å teste alle kombinasjonene for å finne den optimale modellen må det settes noen avgrensinger slik at tidsforbruket satt av til uttesting er innenfor rimelige rammer.

7.1 Fremgangsmåte for valg av modeller

Når det skal velges hvilken modell som skal anvendes finnes det flere mulige fremgangsmåter. Optimalt bør alle ulike modeller testes, med ulike parameterkonfigurasjoner på alle mulige datasettkombinasjoner. Dette fører til stor kompleksitet som krever mye tid. Siden tiden er noe begrenset vil det i dette kapittelet heller bli foreslått en fremgangsmåte basert på Dennis Soemers fremgangsmåte. Han påpeker også at en slik optimal fremgangsmåte ofte ikke er mulig på grunn av kompleksiteten og ressursforbruket. Fremgangsmåten foreslått av Dennis Soemers er som følgende [105] :

1. Velg ut et subsett med features fra datasettet som kan antas å ha en viktig påvirkning for prediksjonen.
2. Bruk grid search eller lignende til å justere hyperparameterene til modellen med subsettet av datasettet.
3. Kjør feature-utvelging med den beste modellen funnet i forrige steg etter justerte hyperparameter.
4. Finjuster modellens hyperparameter til de features som ble plukket ut i feature-utvelgingen.

Det nevnes ikke noe om valg av maskinlæringsmodell, også kalt modelltype, men det vil i dette tilfellet være naturlig å velge modelltype etter første hyperparamterjustering, altså mellom steg to og tre [58]. For å vurdere modellene er det valgt å bruke ytelsesmålene RMSE og MAE, og tiden modellene bruker på å trene. RMSE vil være den viktigste faktoren og hovedytelsesmålet siden den vektlegger store avvik mer enn MAE. Større avvik er uheldig for disse modellene, da det kan føre til økonomiske tap ved anvendelse av modellene for styring av effekter. Derfor er det viktig å bruke et ytelsesmål som vektlegger større avvik, slik som RMSE. Fremgangsmåten for å finne riktig modell er vist som flytdiagram i figur 32 og blir da som følger:

Steg en innebærer uthentingen av data. Her vil data bli hentet ut fra ulike API'er og samlet til et datasett.

Steg to tar for seg preprosesseringen av dataen, som er et viktig steg før modelleringen [110]. Først vil dette steget rense dataen med blant annet utligger deteksjon og andre funksjoner [36]. Videre gjøres feature-uthenting ved hjelp domeneekspertise og god kjennskap til dataen. Til slutt i dette steget skaleres dataen.

Steg tre er implementert for å få ned ressurs- og tidsforbruket av modelleringen [58] [105]. Et subsett av datasettet blir dannet for bruk i utvikling og selektering av modeller. Subsettet er valgt ut med de features som er ansett som viktigst. Modellen som blir utviklet ved hjelp av dette subsettet vil senere bli brukt i feature-utvelging på det fullstendige datasettet.

Steg fire deler dataen opp i trening, validering og test. Merk her at testsettet blir ikke tatt i bruk under modellselekteringen. Testsettet skal være helt isolert til siste test for den endelige modellen [43]. Trenings- og valideringsdataen blir deretter tatt med videre til steg fem.

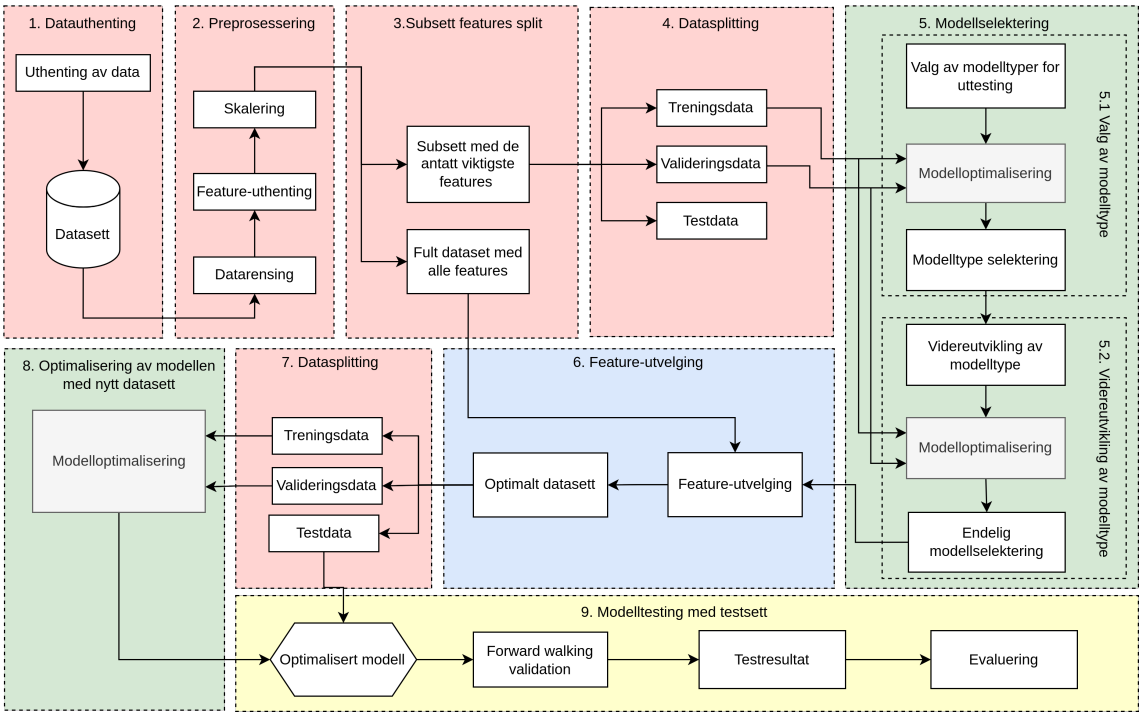
Steg fem er det mest tidskrevende steget. Først, i steg 5.1, vil ulike maskinlæringsmodeller, videre kalt modelltyper, bli valgt ut for testing basert på det tidligere arbeidet som har blitt gjort. En modelltype kan for eksempel være ARIMA eller LSTM. De valgte modelltypene optimaliseres ved hjelp av grid search og walking validation. Modelloptimaliseringen er vist i figur 33. Valg av modelltype gjøres for å unngå kompleksitet og tidsforbruk ved å videreutvikle flere modelltyper parallelt. Når en modelltype har blitt valgt vil det i steg 5.2 bli brukt tid på å videreutvikle denne modelltypen. Dersom for eksempel LSTM blir valgt som modelltype i 5.1 vil det i 5.2 utforskes teknikker som vil lage ulike LSTM-modeller, altså viderutvikle modelltypen. De videreutviklede modellene blir også hyperparameteroptimalisert før den endelige avgjørelsen blir tatt på modellselekteringen. Det å først se på ulike modelltyper og deretter sette opp ulike modeller av den valgte maskinlæringsmodellen er presentert som en løsning i *Applied Predictive Modeling* [58].

Steg seks tar for seg feature-utvelgingen. Her brukes den optimaliserte modellen valgt i steg fem med feature-utvelgingsmetoden *forward selection* for å velge det nye datasettet ut fra det fullstendige datasettet [94]. Resultatet vil gi et godt datasett med de mest aktuelle features for prediksjonsmodellen.

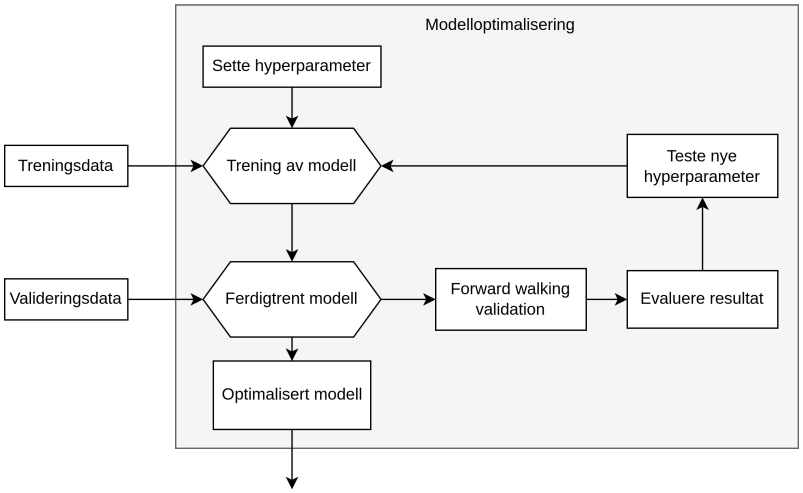
Steg syv splitter opp det nye datasettet fra steg seks. Trenings- og valideringssettet vil bli brukt for en siste modelloptimalisering og testsettet blir brukt på den endelige testen på den valgte modellen.

Steg åtte bruker som nevnt trening- og testdataen for å optimalisere modellen som ble valgt i steg fem på nytt, da datasettet mest sannsynlig har endret seg etter steg seks.

Steg ni tar for seg den endelige evalueringen av modellen. Testsettet har frem til nå vært isolert fra modellen og derfor vil testen gi et grunnlag for hvorvidt modellen er generalisert [43].



Figur 32: Flytdiagram for valg av modell.



Figur 33: Modelloptimalisering.

7.2 Valideringsmetode

I følge [19] anbefales det for validering av modeller innenfor tidsseriedata å bruk OOS-metoder med flere tidsperiode. For å validere resultatene og finne ytelsesmålene implementeres det en funksjon som bruker walking validation, som er en *Out Of Sample validation* metode beskrevet i kapittel 3.4.10. Denne vil for validerings- og testsettet bruke prediksjonsmodellen til å predikere $\hat{y}$ L steg fremover, altså $[\hat{y}_{t+1}...\hat{y}_{t+L}]$. Deretter vil funksjonen sammenligne $\hat{y}$ med den virkelige verdien y for å finne ytelsesmålene. Til slutt vil funksjonen sørge for at modellen trenes på $[y_{t+1}...y_{t+L}]$ og

$[X_{t+1}...X_{t+L}]$ før den vil predikere neste steg, nemlig $[\hat{y}_{t+1+L}...\hat{y}_{t+2L}]$. Slik fortsetter funksjonen til den er gjennom validerings- eller testsettet, avhengig av hvilket sett som brukes. Når metoden har kjørt gjennom validerings- eller testsettet vil den regne ut de gjennomsnittlige ytelsesmålene for alle de predikerte periodene, p, der hver periode blir $[\hat{y}_{t+1+(Lp)}...\hat{y}_{t+(L+Lp)}]$. For et datasett med lengde N vil antall perioder bli $p = \lfloor \frac{N}{L} \rfloor$ og p starter på $p = 0$.

7.3 Hyperparameteroptimalisering

Under hyperparameteroptimaliseringen er det ønskelig å bruke et verktøy for å unngå mye manuelt arbeid. Det finnes flere metoder som automatisk tester forskjellige hyperparametere i modellen. I dette prosjektet implementeres grid search, presentert i kapittel 3.4.8. Denne funksjonen tar alle mulige hyperparametere som er ønskelig å teste, som input. Deretter lager funksjonen en løkke der modellfunksjonen kalles. Input til i modellfunksjonen er de ulike settene av hyperparametere. Funksjonen er satt til å ta opp tiden modellen bruker for å trene, og tidene modellen bruker for å kjøre walking validation. Hver iterasjon i løkken sender inn ett nytt sett med hyperparametere og får ut nye ytelsesmål slik som RMSE, MAE og tid. Ytelsesmålene samt hyperparametersettene lagres i en CSV-fil, slik at hver enkel test blir lagret og det er mulig å gå tilbake i historikken for å sammenligne tester. Resultatet fra loss-funksjonen blir også lagret i en CSV-fil for hver test. Dette blir først og fremst gjort for å bevare alt av testresultat, men også for å filtrere ut overtilpassede modeller.

For å filtrere ut overtilpassede modeller brukes læringskurven for loss-funksjonen. Denne kurven dannes av at modellen finner loss-funksjonsverdien for hver epoch til både trening og validering. Ut fra læringskurven kan det observeres om modellen blir overtilpasset. Kjennetegnet til en overtilpasset modell er at læringskurven for trening minker, samtidig som læringskurven for validering, etter å ha minket noe, begynner å øke etter x antall epochs [7]. Dette er egenskaper som kan brukes for å filtrere ut overtilpassede modeller. Funksjonen er satt opp slik at den overvåker læringskurven. Dersom valideringskurven begynner å øke i x antall epochs etter ha minket i over y antall epochs vil modellen bli flagget som overtilpasset. Dette filteret gjør jobben med å analysere testresultatet enklere.

Under hyperparameteroptimalisering er det viktig å sette *random seeds* . Random seed er en initialverdi til modeller som genererer tilfeldige verdier. Ofte brukes nåtid i millisekunder som initialverdien. Dersom random seed ikke blir satt til en fast verdi kan det oppstå problemer med å reprodusere resultatet, da flere modelltyper brukere tilfeldige nummer i optimaliseringen [47].

7.4 Oppdeling av datasett

I dette prosjektet er det to modeller som skal utvikles. Begge modellene har forskjellige tidshorisont på prediksjonene. Langtidsmodellen skal predikere 48 timer frem i tid, og korttidsmodellen skal predikere maks 60 minutter frem i tid. På grunn av dette opprettes to datasettet. Først et datasett med samplingstid på en time som skal brukes til langtidsmodellen, og et datasett med samplingstid på fem minutter for korttidsmodellen. Features som bare har samplingstid på en time blir ikke brukt i korttidsmodellen.

Før det skal velges hvilke features som blir tatt med i det endelige datasettet til den beste modellen, må det først modelleres en noe optimalisert modell som kan brukes i feature-utvelging, slik som fremgangsmåten er beskrevet i kapittel 7.1. For å velge mellom modelltypene, brukes det derfor bare et subsett av alle features, da dette sparer kjøretid og det er usikkert hvilke features som er gode eller dårlige for prediksjonen. Flere features trenger nødvendigvis ikke å bety bedre prediksjoner [110]. Subsettet er valgt med en antagelse av hvilke features som er viktig for å predikere energiforbruket, i samarbeid med domeneeksperter. Features som er valgt som subsett er vist i tabell 9. Feature *minutt* vil bare være aktuelt for modellen som bruker en samplingstid lavere enn en time.

Variabelnavn	**Beskrivelse**	**Verdi**
forbruk	Gjennomsnittlig effektforbruk i bygg	-
utetemp	Utetemperatur for bygg	-
tilstede	Tilstedeværelse i bygg	-
måned	Hvilken måned	1-12
ukedag	Hvilken dag i uken	1-7
time	Hvilken time i døgnet	0-23
minutt	Hvilken minutt i timen	0-59
arbeidsdag	Arbeidsdag eller ikke	0-1
arbeidstid	Arbeidstid eller ikke	0-1

Tabell 9: Oversikt over features valgt i subsettet.

8 Langtidspredikering

Dette kapittelet beskriver hvordan prediksjonsmodellen som skal predikere over lengre tid er utviklet. Først presenteres en beskrivelse av modellen. For å finne en god modell skal to ulike modelltyper evalueres. Etter at den beste modelltypen er valgt, blir denne modellen videreutviklet og brukt til feature-utvelging.

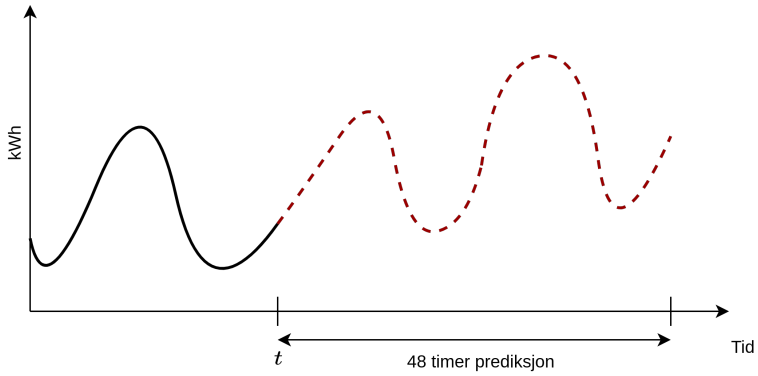
8.1 Beskrivelse av modellen

Som nevnt i problemstillingen er det ønskelig å lage en modell som kan brukes for å predikere energiforbruket eller energibehovet lengre frem i tid. Dette er fordi en slik modell kan brukes til å se om det vil være høyt forbruk ved høye strømpriser. Dersom det er tilfellet, vil denne informasjonen brukes til å forskyve energiforbruket til perioder med mindre forbruk. GK har også i senere tid sett på bruken av batteri i bygg. Batteriene er tiltenkt som en ekstra energikilde som kan settes inn ved effekttopper eller høye strømpriser. For å estimere når batteriene bør være fulladet og klare for å ta over som energikilde, kan en slik modell anvendes.

Langtidsprediksjonsmodellen er en modell som predikerer energiforbruket i bygget flere timer frem i tid. Som diskutert i kapitel 6.1 er det gjennomsnittlige effektforbruket med samplingstid en time tilnærmet lik energiforbruket, slik at målverdien, y, er gitt i formel 38. Derfor blir y i dette kapittelet beskrevet som energiforbruket.

$$y = E \approx \bar{P} \cdot 1h \quad [kWh] \tag{38}$$

Som et krav fra GK er det satt ett predikeringsvindu opptil 48 timer frem i tid. Modellen kjører derfor 48 timers prediksjonsvindu som heretter blir kalt L. Modellen skal bruke features som værmelding og fremtidig tilstedeværelse i bygget til å øke presisjonen på prediksjonene. Værmeldingen kan skaffes fra ulike API'er. Antall personer i bygget kan for eksempel hentes fra en annen prediksjonsmodell, kalender API eller bookingoversikt hos hoteller. For å utvikle modellen blir det brukt historiske data og ingen prognosedata som for eksempel værmelding. Dette er gjort for å unngå at usikkerheten fra prognosedataen påvirker modellen i dette prosjektet, som igjen vil gjøre det vanskelig å skille ut om prognosedataen eller modellen er dårlig [69]. Det vil si at modellen bruker virkelig historiske verdier under trening, men i et produksjonsmiljø vil den bruke prognosedata for å predikere frem i tid. Modellene som blir vist i dette kapittelet vil derfor vise testresultat basert på at prognosedataen er helt korrekt. Til slutt utforskes det nærmere hvilken usikkerhet som kan forventes i prognosedataen. Figur 34 viser at ved t skal modellen predikere energiforbruket 48 timer frem i tid.



Figur 34: Prediksjon av energiforbruk 48 timer frem i tid.

I dette prosjektet er det valgt ut et testbygg, men det er et ønske fra GK å lage en generell modell som kan implementeres for alle typer bygg. Prediksjonsproblemet kan beskrives som multivariant-multistep-multisesong prediksjon med eksogene variabler. Disse utrykkene er beskrevet i kapittel 3.4. Ut fra det tidligere arbeidet som er gjort ble det valgt å teste ut modelltypene SARIMAX og LSTM. Etter at en av modellene er valgt, blir den valgte modelltypen brukt for videreutvikling og eksperiment.

8.2 SARIMAX-modell

8.2.1 Implementasjon

For å implementere SARIMAX ble biblioteket *SARIMAX* fra Statsmodels brukt i Python. Når dette biblioteket er importert, kan SARIMAX-modellen implementeres med noen få linjer kode. I motsetning til LSTM kan SARIMAX ta inn dataramme direkte, som gjør at det ikke er nødvendig med ekstra databehandling før dataen brukes, da det allerede er gjort i preprosesseringen av dataen.

8.2.2 Testing av hyperparameter

En anbefalt metode for å finne parameterne for SARIMAX er å teste ut flere modeller og deretter evaluere disse med ytelsesmål [75]. Ut fra testene som er utført har SARIMAX en god del lengre modelleringstid en LSTM. I tillegg krever kontinuerlig trening som brukes i walking validation mer prosessorkraft enn kontinuerlig trening for LSTM. På bakgrunn av dette og at tilgjengelig RAM er begrenset, er det valgt å kjøre *random grid search* for SARIMAX. Denne metoden er beskrevet i kapittel 3.4.8. SARIMAX krever også at parametere m som sier hvor lang en sesong er, blir satt. I dette tilfellet har datasettet samplingstid på en time og fra tidligere analyse er det observert at en sesong er 24 timer for energiforbruket. Derfor blir $m = 24$. Resultatet med de beste topp fem sortert etter RMSE er vist i tabell 10. Beste resultatet av random gridsearch sortert etter RMSE er SARIMAX(0,0,0)(1,1,1)24.

En annen fremgangsmåte for å finne hyperparameter, presantert i [53], er å bruke Adfuller-test for å finne ut om dataen er stasjonær, eller om den må differensieres. Deretter kan ACF og PACF grafene benyttes for finne AR (p,P) og MA (q,Q). Denne metoden er effektiv og kan finne et godt parametersett på kort tid. Pakken *autoarima* i Python bruker disse metodene for å finne hyperparameterne. Autoarima finner ut at SARIMAX(2,0,0)(2,0,1)24 er den beste modellen ved å bruke AIC som ytelsesmål. Hvis modellen testes på valideringssettet og finner RMSE gjennom walking validation gir det et resultat på $RMSE = 15.68$. Dette tyder på en dårlige modell enn det random grid search fant, og derfor ble ikke denne metoden brukt for å finne hyperparameterene. Grunnen for at disse to metodene finner forskjellige hyperparameter kan være de ulike ytelsesmålene og fremgangsmåten de bruker for å finne den beste modellen. Autoarima bruker AIC og random gridsearch er satt opp til å bruke RMSE.

	Hyperparameter							**Evalueringsmål**			
Test	**p**	**d**	**q**	**P**	**D**	**Q**	**S**	**AIC**	**RMSE**	**MAE**	**Tid (s)**
72	0	0	0	1	1	1	24	133038.09	14.21	10.65	696.1
80	0	0	1	1	1	1	24	124629.52	14.29	10.77	790.4
76	0	0	1	0	1	1	24	124928.36	14.67	11.08	730.0
78	0	0	1	1	0	1	24	125053.53	14.70	11.21	213.5
70	0	0	0	1	0	1	24	133699.15	14.78	11.23	249.7

Tabell 10: Resultat av hyperparameteroptimalisering SARIMAX.

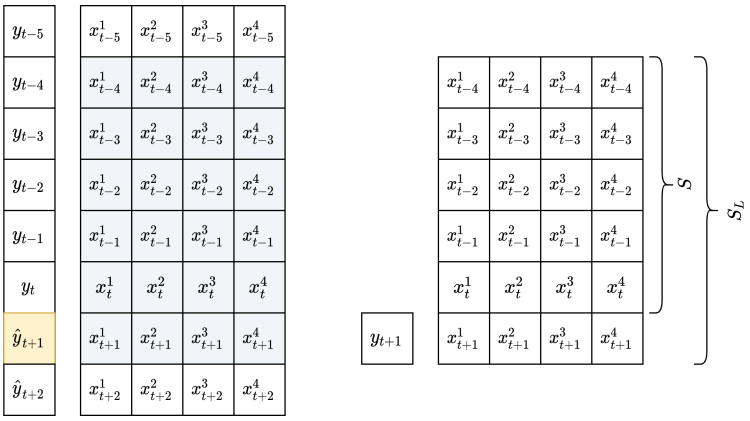
8.3 LSTM-modell

8.3.1 Implementasjon

For å implementere LSTM må det i tillegg til å sette opp modellen, omforme datarammen slik at LSTM kan utnytte informasjonen. I kapittel 3.8 ble det beskrevet at LSTM er oppbygget slik at den kan memorere sekvensiell data. En metode for å utnytte denne egenskapen er å sette opp dataen slik at hver sampel inneholder en sekvens. Dersom $\hat{y}_{t+1}$ skal predikeres vil features for $t+1$ anvendes, men også data fra tidligere samples. S defineres som antall siste samples som brukes i sekvensen og S_L definerer lengden av sekvensen. Lengden vil bli $S_L = S + 1$ siden sekvensen blir S antall siste verdier pluss neste verdi. Formel 39 viser hvordan S siste samples blir brukt for alle features, X, sammen med X_{t+1} for å predikere $\hat{y}_{t+1}$.

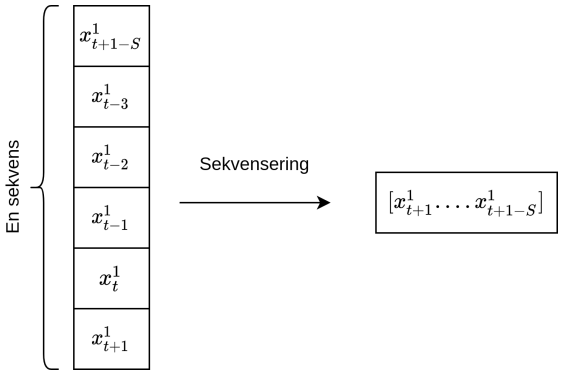
$$\hat{y}_{t+1} = f([[X_{t+1}...X_{t+1-S}] \tag{39}$$

Datarammen i venstre bilde nedenfor, figur 35, viser dataen som trengs for å predikere $\hat{y}_{t+1}$ som grå celler.



Figur 35: Dataen som trengs for å predikere $\hat{y}_{t+1}$.

Slik LSTM er bygd opp må dataen som skal brukes for å predikere $\hat{y}_{t+1}$ være på samme rad som $\hat{y}_{t+1}$. Dette er ikke tilfellet nå, da $\hat{y}_{t+1}$ er bare en rad og $[X_{t+1}...X_{t+1-S}]$ er S_L rader. For å gjøre dette hentes en sekvens av lengde S_L for hver feature og transponerer disse radene til en vektor. Figur 36 viser hvordan S_L radene ble tatt ut for feature x^1 og transponert til en vektor.



Figur 36: Fra S_L rader til vektor med lengde S_L.

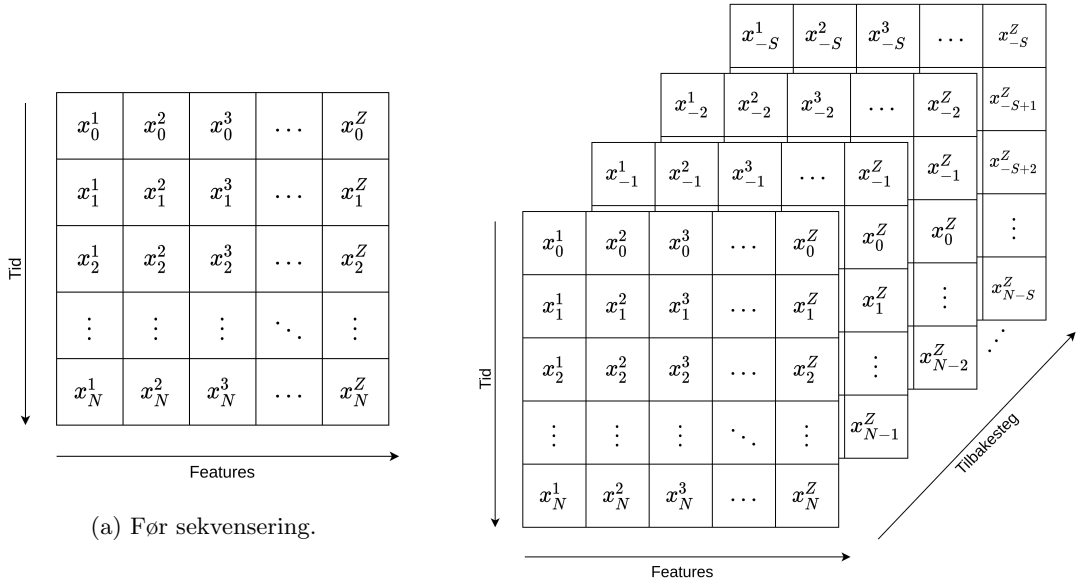
Etter at dataen er sekvensert må sekvensene settes inn igjen i samme rad som $\hat{y}_{t+1}$. Figur 37 viser resultatet av sekvensering for $\hat{y}_{t+1}$ og $\hat{y}_{t+2}$

$\hat{y}_{t+1}$	$[x^1_{t+1} \dots . x^1_{t+1-S}]$	$[x^2_{t+1} \dots . x^2_{t+1-S}]$	$[x^3_{t+1} \dots . x^3_{t+1-S}]$	$[x^4_{t+1} \dots . x^4_{t+1-S}]$
$\hat{y}_{t+2}$	$[x^1_{t+2} \dots . x^1_{t+2-S}]$	$[x^2_{t+2} \dots . x^2_{t+2-S}]$	$[x^3_{t+2} \dots . x^3_{t+2-S}]$	$[x^4_{t+2} \dots . x^4_{t+2-S}]$

Figur 37: Resultat av sekvenseringen.

Denne operasjonen må gjøres for hele datasettet før modellen trenes. Figur 38a viser en generell matrise av datasettet. x^z_t er en sample der z beskriver hvilken feature det er snakk om. Z beskriver totalt antall features og N er lengden på datasettet, altså hvor mange samples det er i datasettet. For hver sampel i hver feature er det ønskelig å legge til en sekvens. Resultatet av sekvenseringen

på hele datasettet er en tredimensjonal matrise vist i figur 38b. x_{t+1} blir her notert som x_1 for å spare plass i figuren.



(a) Før sekvensering.

(b) Etter sekvensering.

Figur 38: Sekvensering av data.

For å utføre sekvenseringen ble de laget en generisk funksjon som kan ta inn ulike verdier for N, Z og S. Denne delen hører egentlig til forberedelse av datasettet, kapittel 6, men siden SARIMAX ikke trenger samme type input blir det beskrevet her. Datasettet vil miste de S første verdiene på grunn av at disse verdiene ikke har S verdier før seg, og kan derfor ikke danne en sekvens. Første verdi som har S verdier bak seg er verdi nummer S_L i datasettet. Dimensjonen etter sekvensering på hele datasettet vil derfor bli med dimensjonene $(N - S) \times S_L \times Z$. For å finne en optimal S ble LSTM-modellen tatt i bruk der $S = [1...10]$ ble testet og evaluert med RMSE. Den S-verdien som ga best resultat var $S = 7$.

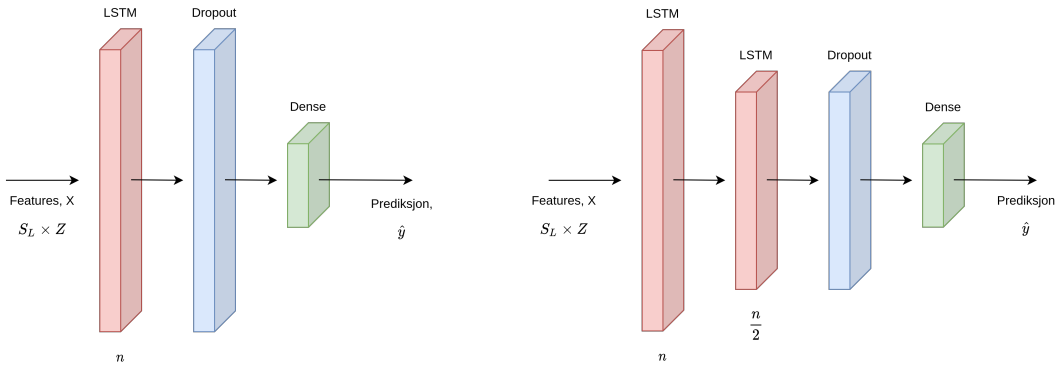
For denne modellen vil verdiene predikeres basert på features det finnes fremtidig data på, slik som værmelding, ukedag og tidspunkt i døgnet. Det betyr at ved $t = 0$, har modellen dataen på alle features L prediksjoner fremover. Begrensningen på prediksjonsvinduet er typisk avgrenset av dataen, for eksempel hvor lang tidshorisont det er på værmeldingen. Formelen for prediksjonen blir lik formel 40. Her er det mulig å predikere hele perioden $[\hat{y}_{t+1}...\hat{y}_{t+L}]$ i tidspunktet t, da dataen på høyre side av likhetstegnet er tilgjengelig ved t.

$$[\hat{y}_{t+1}...\hat{y}_{t+L}] = f([[X_{t+1}...X_{t+1-S}]...[X_{t+L}...X_{t+L-S}]]) \tag{40}$$

Når et nevralt nettverk skal settes opp, finnes det ingen konkret oppskrift på hvordan dette skal gjøres [92]. Ofte kreves det unike modeller for gitt problemstilling og datasett. Derfor trenes et

gitt antall modeller ved bruk av for eksempel grid search, og deretter evalueres disse for å finne den beste modellen. For å unngå å teste alle mulige verdier på hyperparameterne finnes det noen tommelfingerregler som kan gi en god initialverdi under hyperparameteroptimaliseringen [92][93]. Som en generell tommelfingerregel innenfor tidsserieanalyse sies det at en modell med ett lag løser de enkle problemene, mens modeller med to lag kan hente ut mer komplekse features fra datasettet, og deretter løse mer komplekse problemer. Det anbefales å bruke *relu* som aktiveringsfunksjon, og som optimaliserer anbefales *Adam*. Dersom det er flere lag i modellen er det vanlig å halvere antall nevroner per lag.

Det er også ønskelig å legge til dropout-lag for å unngå overtilpasning dersom det er fare for det. Dropout-laget ble lagt til etter LSTM lagene og før dense-laget som også er presentert som en løsning av Jason Brownlee [15]. Som tommelfingerregler ble initialverdiene til dropout avgrenset til å teste verdier under 50% [14] og learning rate ble testet med ulike verdier med en faktor på 10 [17]. På bakgrunn av dette settes det opp to LSTM-modeller med ulike antall lag, se figur 39. Modellen vil ut fra en feature-matrise med dimensjonene $S_L \times Z$ predikere $\hat{y_{t+1}}$. Verdiene under LSTM lagene beskriver antall nevroner. Modellene er satt opp ved bruk av biblioteket Keras og Tensorflow i Python.



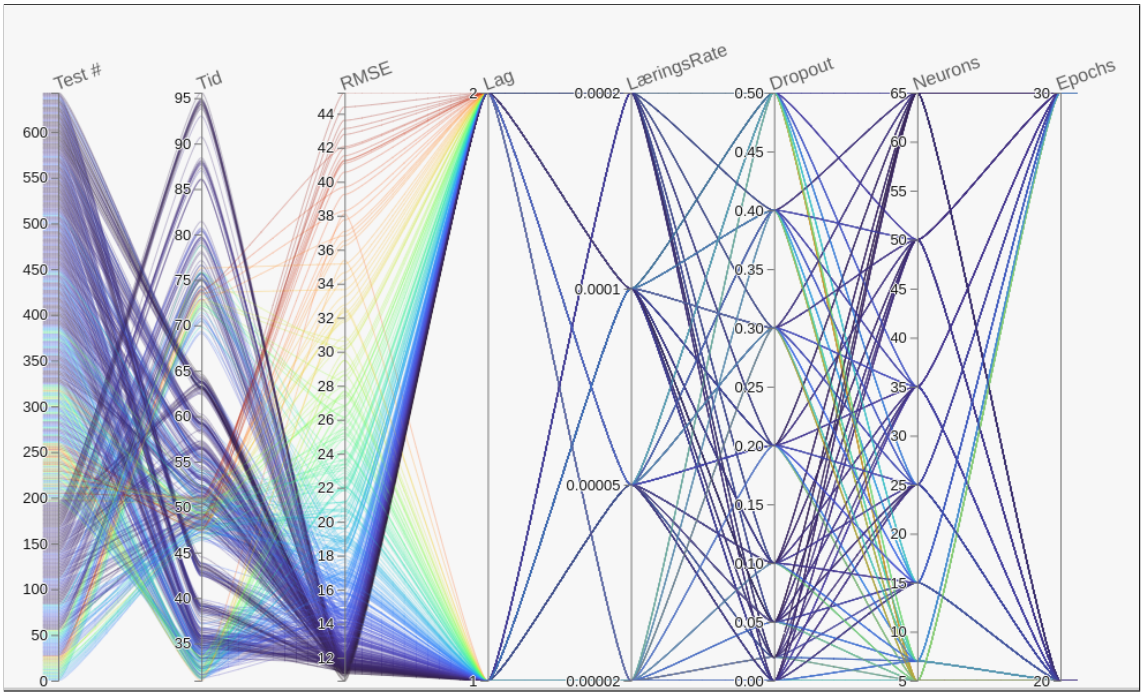
(a) LSTM-modell med ett lag.

(b) LSTM-modell med to lag.

Figur 39: LSTM-modellen brukt under testing.

8.3.2 Testing av hyperparameter

For å finne hyperparameterne for LSTM-modellen brukes grid search. Etter en runde med grid search er det mulig å spesifisere hyperparameterne enda mere før en ny grid search kjøres. Figur 40 viser et såkalt *Parallel coordinates plot* over de forskjellige hyperparameterne og ytelsesmålene for et subsett av utførte tester. Hver linje i dette plottet indikerer en test der linjene har farger etter RMSE-verdien på testen. Dette plottet er satt opp i Python til å være interaktivt og er et godt verktøy til å analyseres testene. Ut fra det statiske bilde er det mulig å se at for eksempel nevroner under 15 gir dårlig resultat.



Figur 40: Paralellkordinatplot for noen av testene utført.

Tabell 11 viser resultatet av grid search sortert etter RMSE. Tabellen viser at modellen ikke nødvendigvis trenger dropout-laget for noen konfigurasjoner. Det er flere parametere som kan påvirke overtilpasningen på modellen, slik som å endre på kompleksiteten til modellen eller datasettet, som kan gjøre at noen modeller ikke trenger dropout-laget.

	Hyperparameter					**Evalueringsmål**		
#	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	**RMSE**	**MAE**	**Tid (s)**
329	15	15	0	2	0.0001	11.63	8.64	49.6
57	20	15	0	2	0.0001	11.66	8.70	75.7
337	15	15	0.02	2	0.0001	11.70	8.85	52.4
518	20	50	0	1	0.00005	11.76	9.02	38.8
174	30	65	0.05	1	0.00005	11.90	9.03	63.8

Tabell 11: Resultat av hyperparameteroptimalisering LSTM.

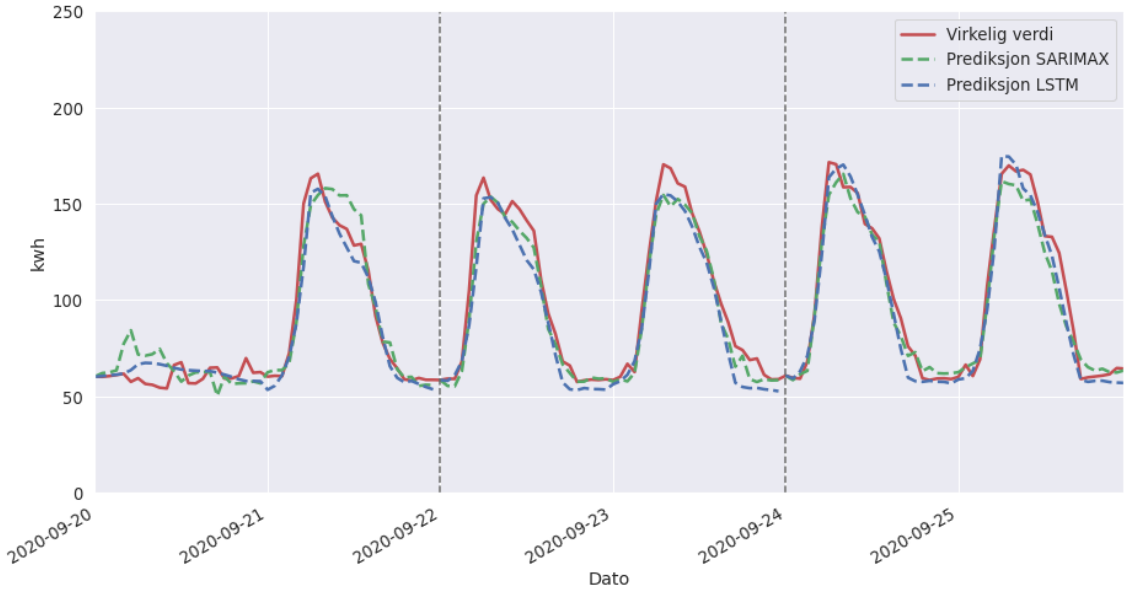
Modell nr. 329 ga det beste resultatet etter grid search og bruker oppbygningen vist i figur 39b, der dropout-laget ikke brukes da det parameteret er satt til null.

8.4 Valg av type modell

8.4.1 Sammenligning av LSTM- og SARIMAX-modeller

Analyseres resultatene fra hyperparameteroptimaliseringen ved hjelp av valideringssettet, viser resultatet fra tabell 10 at SARIMAX har $RMSE = 14.21$ og fra tabell 11 at LSTM har $RMSE = 11.63$.

Dersom prediksjonene på valideringssettet hentes ut, blir resultatene lik figur 41. Denne figuren viser et tilfeldig utklipp av valideringssettet med prediksjonene av begge modellene. Den faste røde linjen er det virkelige energiforbruket i bygget. Den grå, vertikale stiplede linjen forteller når modellene predikerer 48 timer frem i tid. Før neste prediksjon ved neste grå linje trenes modellene på den siste dataen, slik som walking validation er implementert. Dette plottet viser derfor tre eksempler der modellen ved hver grå vertikal strek predikerer 48 timer frem i tid.



Figur 41: Sammenligning av prediksjon med SARIMAX og LSTM på testsettet.

Det er kjent at SARIMAX-modellen originalt bare kan predikere på data med en sesong [31]. Ved å introdusere eksogene variabler som kan beskrive flere sesonger, kan modellen modifiseres til å predikere på multisesonger. Figur 41 viser både ukentlige og daglige sesonger. Da SARIMAX ble implementert ble den satt opp til å bruke sesongen for en dag. Ved å introdusere feature *ukedag* gir det resultatet fra figur 41 som viser at den kan predikere i helgene også, derav predikere med ukentlige og daglige sesonger. En mulig bedre løsning er å introdusere variablene *ukedag_sin* og *ukedag_cos* for å beskrive den ukentlige sesongen [31]. Etter implementering av de to nye features gir det en $RMSE = 13.43$ for SARIMAX, som er en forbedring, men fortsatt dårligere en LSTM-modellen.

Begge modellene gir gode resultater med stort sett tilfeldige avvik. Ytelsesmålene tyder på at LSTM-modellen har en bedre prediksjon. SARIMAX har for dette datasettet en treningstid 10 ganger høyere enn LSTM, og bruker også lengre tid på kontinuerlig trening. Selv om det bare er snakk om minutter i differanse mellom de to modellene, kan det utgjøre en stor forskjell dersom modellene skal implementeres på tusen bygg, der hvert bygg har behov for flere prediksjoner til enhver tid. På bakgrunn av dette velges LSTM-modellen for videre utvikling.

8.5 Videreutvikling LSTM-modell

I forrige delkapittel ble det valgt å gå videre med LSTM-modellen. Videre er det mulig å se om LSTM-modellen kan forbedres.

8.5.1 ML1

Modellen som ble satt opp i 8.3 blir videre kalt ML1. Som nevnt bruker modellen S siste samples for alle features, X, sammen med X_{t+1} for å predikere $\hat{y}_{t+1}$, slik som formel 41 viser.

$$\begin{aligned} \hat{y}_{t+1} &= f([X_{t+1}...X_{t+1-S}]) \\ &\vdots \\ \hat{y}_{t+L} &= f([X_{t+L}...X_{t+L-S}]) \end{aligned} \tag{41}$$

8.5.2 ML2

Foreløpig har modellen bare brukt features som er isolert fra energiforbruket, y, eller den predikerte verdien, $\hat{y}$. Det er også mulig å bruke y og $\hat{y}$ som en feature. En metode går ut på å bruke de siste y-verdiene som en feature. Dette er enkelt å implementere dersom bare ett tidssteg skal predikeres frem i tid. Ved å legge til y-verdien med lag-verdi forskjøvet fremover et tidssteg som feature, altså y_t er en feature for prediksjonen $\hat{y}_{t+1}$, kan singelsteg predikering implementeres. Det er også i dette tilfellet ønskelig å se på de S siste y-verdiene slik at formelen blir som følger.

$$\hat{y}_{t+1} = f([y_t...y_{t-S}], [X_{t+1}...X_{t+1-S}]) \tag{42}$$

Implementasjonen blir noe mer komplisert dersom flere tidssteg (L) skal predikeres frem i tid. En fremgangsmåte er å predikere et tidssteg frem i tid, og deretter bruke denne prediksjonen i neste prediksjon som en historisk verdi. Formel 43 viser hvordan første prediksjon, $\hat{y}_{t+1}$, bruker tidligere virkelige verdier, y, for å predikere. Videre viser formelen at for å predikere $\hat{y}_{t+2}$ og utover brukes tidligere predikerte verdier, $\hat{y}$, sammen med y som en feature. Denne metoden er en multistegsprediksjon, men av måten den er bygd opp på er den også kjent som *Multi-stage*

Prediction [21] eller *Multi-step Rolling forecasts* [120]. For videre diskusjon i dette kapittelet får modellen navnet ML2. Største forskjellen mellom ML1 og ML2 er at ML1 har alle features, mens ML2 må lage sine egne features som er $\hat{y}$.

$$\begin{aligned}
\hat{y}_{t+1} &= f([y_t ... y_{t-S}], [X_{t+1} ... X_{t+1-S}]) \\
\hat{y}_{t+2} &= f([\hat{y}_{t+1}, y_t ... y_{t+1-S}], [X_{t+2} ... X_{t+2-S}]) \\
\hat{y}_{t+3} &= f([\hat{y}_{t+2}, \hat{y}_{t+1}, y_t ... y_{t+2-S}], [X_{t+3} ... X_{t+3-S}]) \\
&\vdots \\
\hat{y}_{t+L} &= f([\hat{y}_{t+(L-1)} ... \hat{y}_{t+(L-1)-S}], [X_{t+L} ... X_{t+L-S}])
\end{aligned} \tag{43}$$

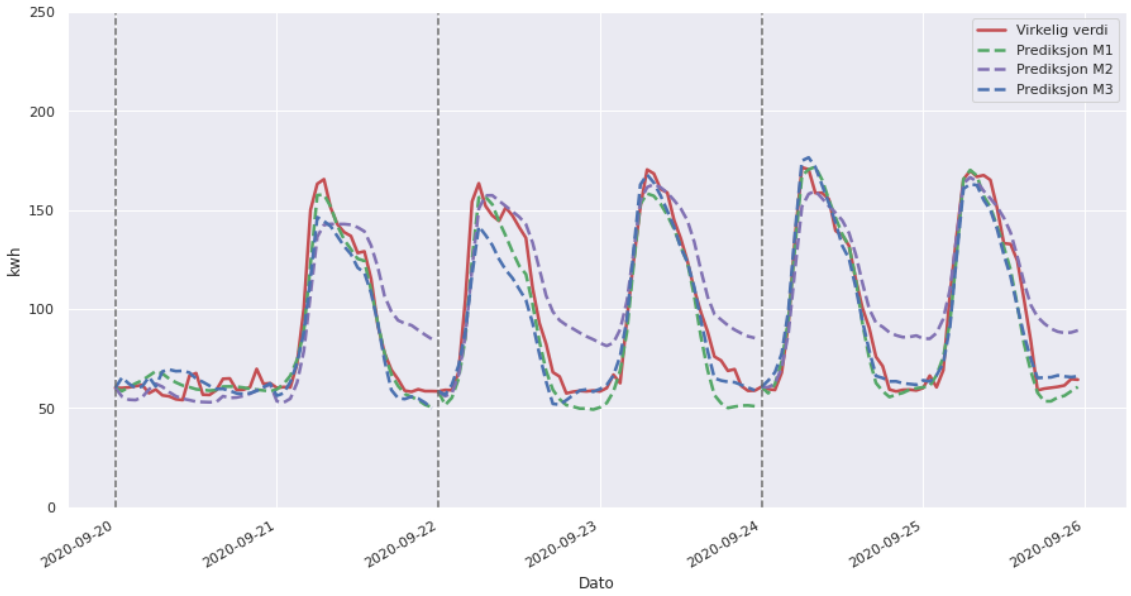
8.5.3 ML3

En annen teknikk er å bruke den virkelige verdien, y, som en feature med L lag-verdier bakover. Dersom samplingstiden er en time, og det er ønskelig å predikere 48 timer frem i tid, $L = 48$, legges de siste 48 timene inn som en feature. Forskjellen mellom ML2 og ML3 er at ML3 kun bruker y som en feature, hvor ML2 bruker y og $\hat{y}$ som feature. Funksjonen for en slik modell blir lik formel 44 og modellen får navnet ML3 for videre diskusjon.

$$\begin{aligned}
\hat{y}_{t+1} &= f([[y_{t-L} ... y_{t-L-S}], [X_{t+1} ... X_{t+1-S}]]) \\
&\vdots \\
\hat{y}_{t+L} &= f([[y_t ... y_{t-S}], [X_{t+L} ... X_{t+L-S}]])
\end{aligned} \tag{44}$$

8.5.4 Evaluering av nye LSTM-modeller

Graf 42 sammenligner LSTM-modellen som ble valgt i kapittel 8.3, ML1, med de videreutviklede LSTM-modellen fremlagt i dette kapittelet, ML2 og ML3. De grå linjene viser når multistegsprediksjonen starter.



Figur 42: Sammenligning av prediksjon med ulike teknikker.

Figuren viser at ML2 har dårlige prediksjoner etter noen samples inn i prediksjonen. Dette kommer mest sannsynlig av at når modellen trenes, vektlegger den tidligere y-verdier høyt siden det beskriver neste verdi godt. Når modellen etter hvert bruker $\hat{y}$ som en feature og den prediksjonen har avvik, vil neste prediksjon få enda større avvik. Denne teknikken vil antagelig fungerer bedre på kortere multistegsprediksjoner. ML3 viser noe dårligere resultat på dette utklippet enn ML1, men tabell 12 viser at RMSE-verdien er nokså lik for ML1 og ML3, og at de har nesten like gode resultater. Det er også her brukt walking validation for å finne RMSE. Siden GK ønsker en modell som kan predikere opp mot 48 timer frem i tid forkastes ML2. Forskjellen mellom ML1 og ML3 er at ML3 har med en ekstra feature; tidligere energiforbruk. I neste kapittel skal det undersøkes hvilken features som er viktig å ha med i modellen, derfor blir den nye feature i ML3 tatt med videre til neste kapittel.

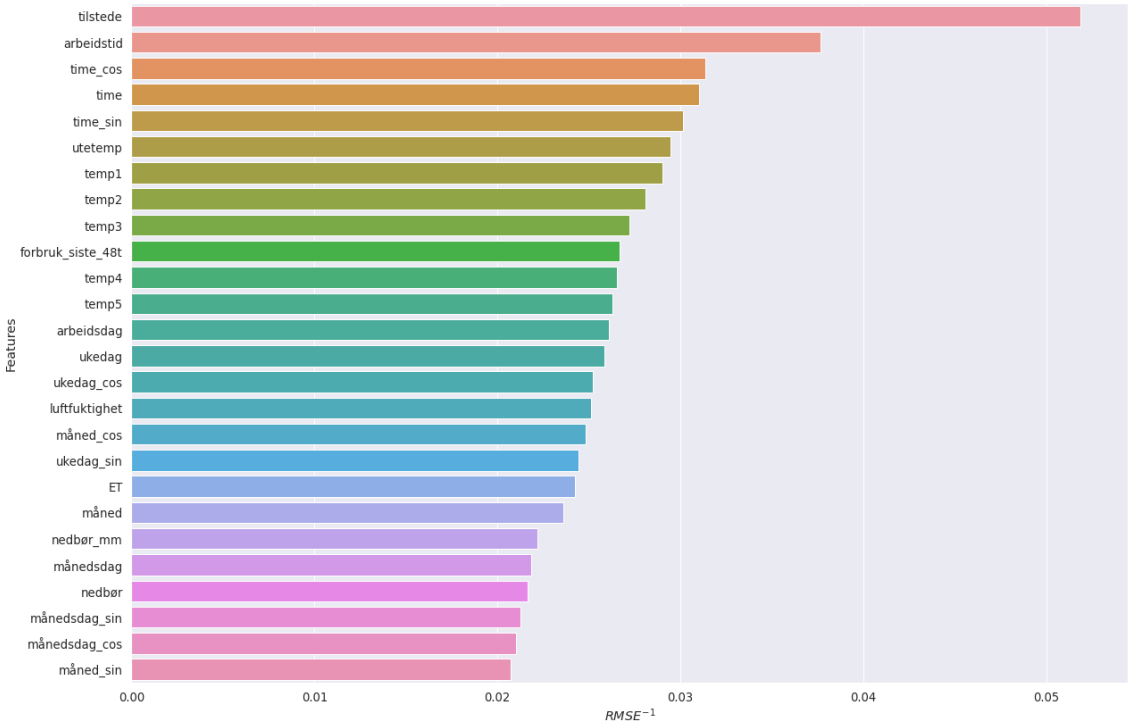
	Hyperparameter					**Evalueringsmål**	
Modell	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	**RMSE**	**Tid (s)**
ML1	15	15	0	2	0.0001	11.63	49.6
ML2	30	15	0	2	0.0001	15.99	404.2
ML3	17	15	0	2	0.0001	11.27	55.5

Tabell 12: Resultat av hyperparameteroptimalisering på modellene

8.6 Feature-utvelging

Det er ønskelig å bruke litt tid på feature-utvelging for å unngå features som motvirker prediksjonen, og for å minke modelleringstiden. I kapittel 3.4.11 ble *forward selection* presentert, som er

en *wrapper* metode. Denne algoritmen brukes i dette kapittelet for å velge ut features. Algoritmen ble implementert fra bunnen av, og går gjennom alle features. Første iterasjon i algoritmen går gjennom en og en feature og evaluerer hver enkel. Figur 43 viser hvilken data modellen anser som viktigst for prediksjonene, dersom modellen bare skal bruke en feature. Det er ikke nødvendigvis de øverste som er best å velge i datasettet, selv om de er ansett som viktige ifølge denne søylegrafen. For eksempel de to øverste features *tilstede* og *arbeidstid* har en høy korrelasjon, som gjør at dersom modellen velger *tilstede* vil ikke *arbeidstid* være like viktig for modellen. Dette er fordi modellen mest sannsynlig kan finne mye av den samme informasjonen i de to ulike features, og modellen vil heller velge en annen feature som kan gi ny informasjon. Verdiene er angitt i inverse av RMSE; en høy verdi betyr at den er viktig. *forbruk siste 48t* som ikke ble presentert i kapittel 6.3 er lag-verdien 48 timer bak i tid som ble foreslått når ML3 ble undersøkt i kapittel 8.5.



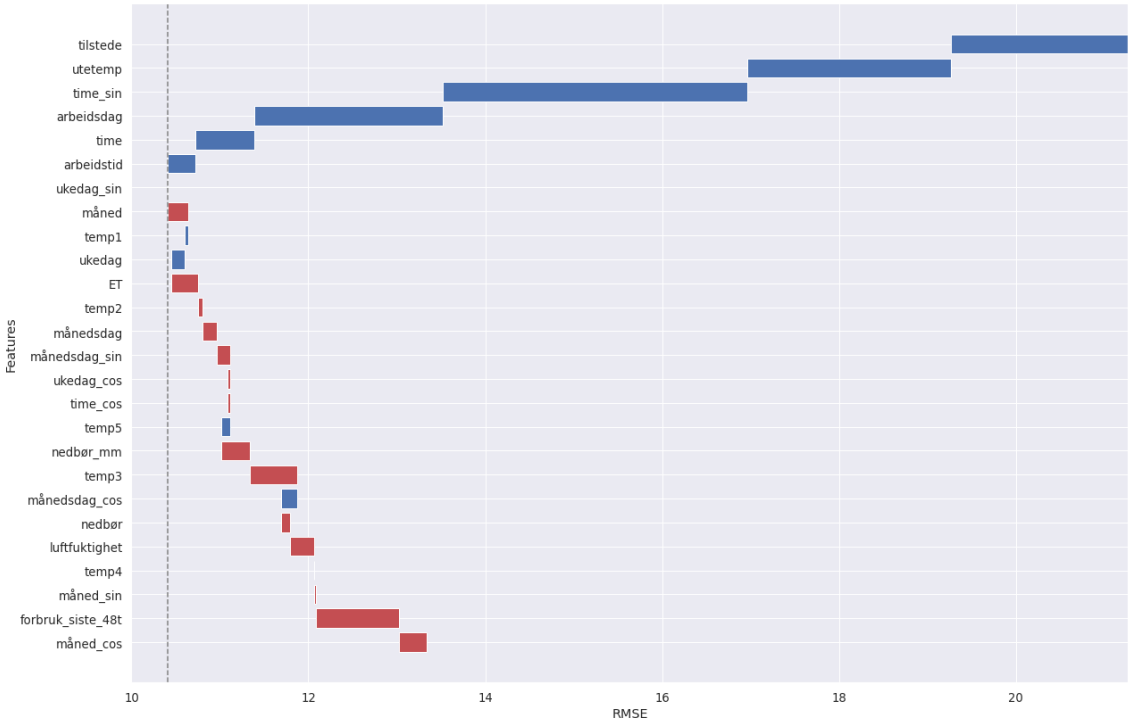
Figur 43: Viktighet av features ved analyse på en og en feature.

Som figur 43 viser velger modellen tilstedeværelse som den viktigste feature. Hvilke andre features den ønsker å legge til i treningssettet er vist i tabell 13, som viser hvordan en og en feature har blitt plukket ut, med de som gir best resultat først.

	Features																										
#	tilstede	utetemp	time_sin	arbeidsdag	time	arbeidstid	ukedag_sin	måned	temp1	ukedag	ET	temp2	månedsdag	månedsdag_sin	ukedag_cos	time_cos	temp5	nedbør_mm	temp3	månedsdag_cos	nedbør	luftfuktighet	temp4	temp4	forbruk_siste_48t	måned_cos	**RMSE**
1	x																										19.27
2	x	x																									16.96
3	x	x	x																								13.52
4	x	x	x	x																							11.39
5	x	x	x	x	x																						10.72
6	x	x	x	x	x	x																					**10.41**
7	x	x	x	x	x	x	x																				10.41
8	x	x	x	x	x	x	x	x																			10.64
9	x	x	x	x	x	x	x	x	x																		10.60
10	x	x	x	x	x	x	x	x	x	x																	10.45
11	x	x	x	x	x	x	x	x	x	x	x																10.75
12	x	x	x	x	x	x	x	x	x	x	x	x															10.80
13	x	x	x	x	x	x	x	x	x	x	x	x	x														10.96
14	x	x	x	x	x	x	x	x	x	x	x	x	x	x													11.12
15	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x												11.09
16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x											11.12
17	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x										11.02
18	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x									11.34
19	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								11.87
20	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x							11.69
21	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x						11.79
22	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					12.07
23	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				12.07
24	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			12.09
25	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		13.03
26	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	13.34

Tabell 13: Feature-utvelging ved bruk av wrapper-metoden forward selection.

Tabell 13 viser at det optimale datasettet er datasett seks, med en RMSE-verdi på 10.41. Datasett syv har også samme RMSE-verdi, men det er ønskelig å bruke minst mulig data for best resultat da det vil bruke mindre datakraft. Derfor velges datasett seks som det beste. Datasett seks inneholder features *tilstede*, *utetemp*, *time_sin*, *arbeidsdag*, *time* og *arbeidstid*. Det er verdt å nevne at dette er det beste datasettet for akkurat denne LSTM-modellen funnet i kapittel 8.3 og dette datasettet. Noe endring på modellen, mer eller mindre data, eller annen samplingstid kan føre til at andre features gir et optimalt datasett. Forward selection er også en grådig algoritme så datasettet er muligens bare et lokalt optimum og ikke et globalt optimum. Figur 44 viser hvordan verdiene påvirker positivt mot den laveste RMSE-verdien (blå) og hvordan noen features etter hvert påvirker resultatet negativt (rød).



Figur 44: Feature sin påvirkning av modellen ved forward selection.

8.7 Resultat - Endelig modell

Av fremgangsmåten som ble presentert i kapittel 7.1 er neste steg etter feature-utvelging å kjøre en ny hyperparameteroptimalisering. Når en ny hyperparameteroptimalisering kjøres ved hjelp av grid search blir den beste modellen, som ikke er overtilpasset, en modell med RMSE-verdi på 10.23. Hyperparameterne av den nye optimaliseringen er vist i tabell 15. Det tidligere forbruket, *forbruk_siste_48t*, ble etter feature-utvelgingen ikke ansett som en viktig feature, noe som betyr at ML1 er den beste modellen. Det endelige datasettet som ble valgt etter feature-utvelging er vist i tabell 14. Resultatet med testsettet er vist i tabell 15. Dette er et godt resultat og RMSE_{val} er mindre enn RMSE_{test}, noe som er forventet siden testsettet er uobservert for modellen fra tidligere. Avviket mellom RMSE_{val} og RMSE_{test} er også nokså lavt, som antyder at modellen ikke er overtilpasset og at testsettet representerer valideringssettet godt.

Variabelnavn	Beskrivelse
tilstede	Tilstedeværelse i bygg
utetemp	Utetemperatur for bygg
time_sin	Sinus med periode dag
arbeidsdag	Arbeidsdag eller ikke
time	Hvilken time i døgnet
arbeidstid	Arbeidstid eller ikke

Tabell 14: Oversikt over endelige valgt features.

	Hyperparameter					Evalueringsmål		
Modell	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	**RMSE**$_{val}$	**RMSE**$_{test}$	**Tid (s)**
ML1	20	15	0	2	0.0001	10.23	11.18	40.4

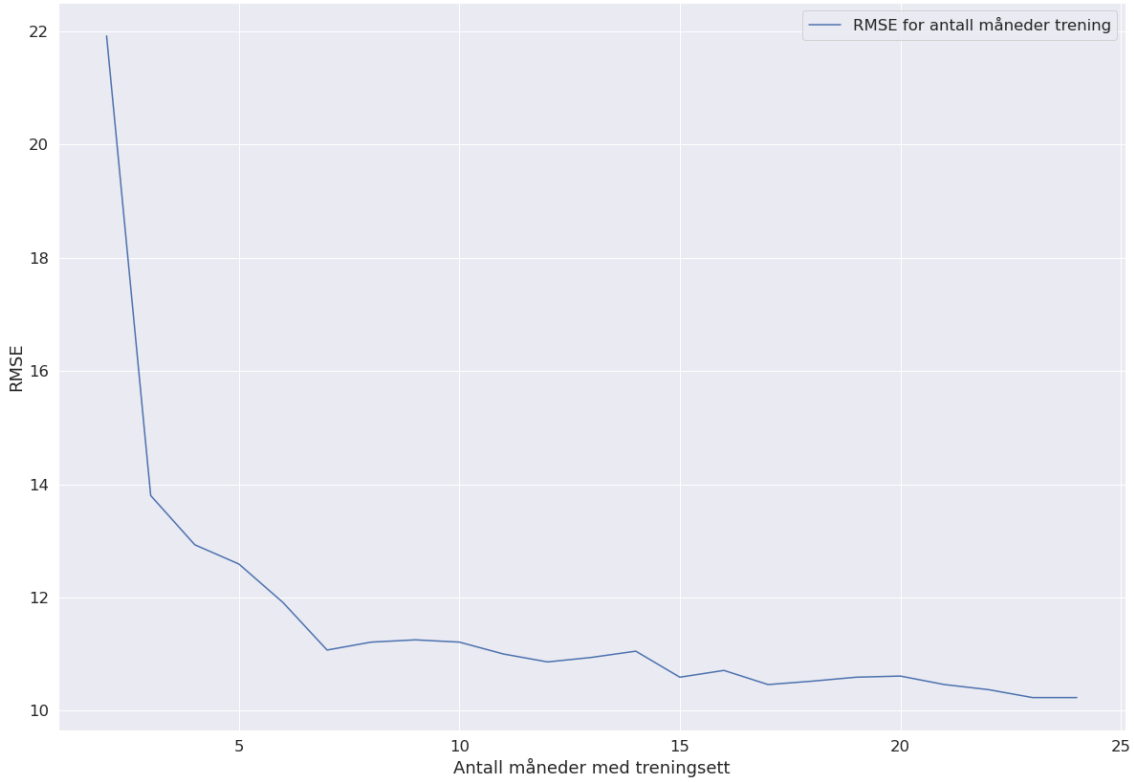
Tabell 15: Resultat av ML1 på testsettet.

8.8 Eksperiment

For GK sin del er det interessant å kjøre flere eksperimenter og tester på hvordan ulike betingelser kan påvirke modellen. Ulike bygg har ulik mengde med data og ulik sensorikk. Ettersom modellen skal være generell, vil noen eksperimenter gi grunnlag for å si noe om forventet resultat med andre betingelser.

8.8.1 Mengde av treningsdata

Mange av byggene GK har i dag har vært brukt i flere år, og har dermed en god del historikk lagret. Det som kan være interessant å utforske, er dersom GK setter i gang et nytt bygg; hvor lenge bør det bygget brukes før bygget har nok data til en god prediksjonsmodell? Ved igangkjøring av nytt bygg brukes det ofte en igangkjøringsfase der det er satt av en periode på å kjøre inn anlegget. Dvs. justere reguleringen, oppgrader underdimensjonerte komponenter osv. Denne perioden bør være utelatt fra datasettet da det ikke er normal kjøring. Figur 45 viser treningsdata i forskjellige perioder fra 2 måneder til 24 måneder. Det er tydelig at RMSE-verdien synker med større datasett, noe som er forventet resultat. Det er også mulig å se at kurven begynner å flate ut etter ca. et år med data. Dette er også forventet da det er først ved et år med data at modellen har sett alle årstidene. Trenden på kurven tyder på at noe mer data også kan føre til enda lavere RMSE.



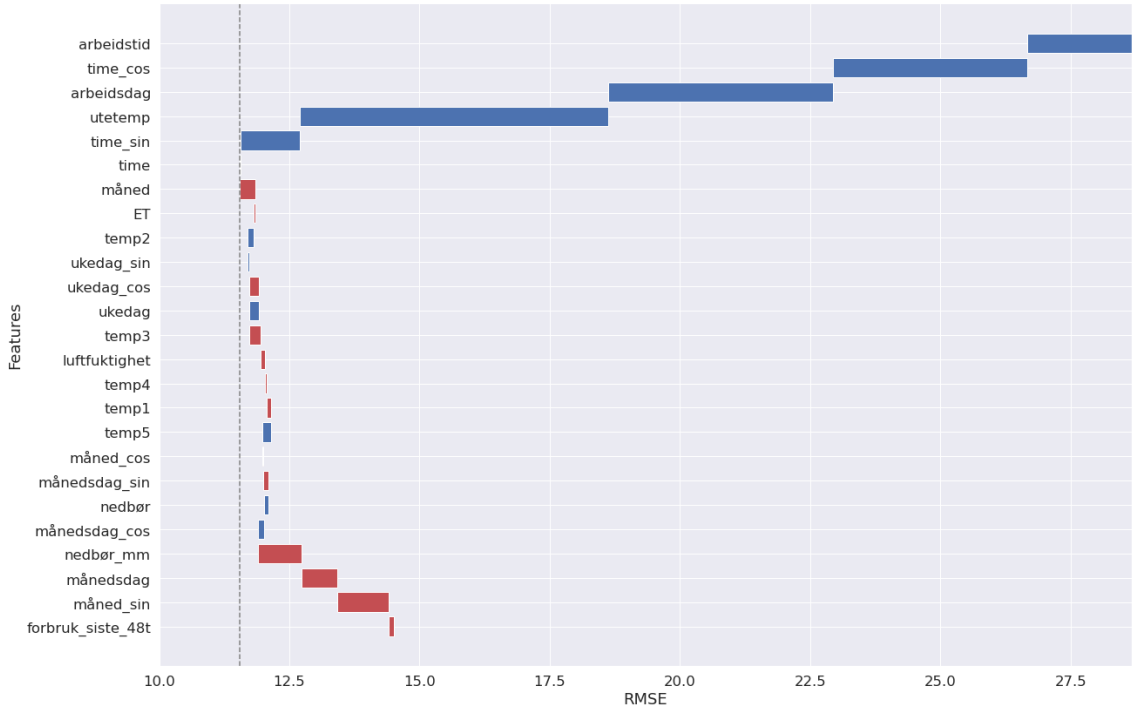
Figur 45: RMSE-verdi ved forskjellig lengde på treningssettet.

8.8.2 Manglende tilstedeværelse

De fleste nye bygg i dag har målinger på tilstedeværelse. Modellen som er utviklet så langt bruker tilstedeværelse som en eksogen variabel til å predikere. GK ønsket å bruke tilstedeværelse som en eksogen variabel selv om det i dag ikke finnes noe data på fremtidig tilstedeværelse i bygget. Tanken bak dette var å se hvor viktig tilstedeværelse i bygget er for prediksjonen. I kapittel 8.6 ble det vist at dette var en veldig viktig feature. Informasjonen er viktig for GK da de kan begynne å se på hvordan de kan hente ut fremtidig tilstedeværelse fra for eksempel kalendere eller booking. Eventuelt kan mulighetene for å lage en prediksjonsmodell for tilstedeværelse undersøkes. Siden de fleste byggene ikke har mulighet til å hente denne informasjonen per dags dato, er det ønskelig å se hvordan modellen presterer uten tilstedeværelse som en feature.

En mulighet for å sjekke hvordan mangel på tilstedeværelse påvirker modellen, kan være å fjerne tilstedeværelse som en feature fra datasettet som ble valgt i kapittel 8.6, og deretter se hvordan modellen presterer. Dette er ikke en optimal fremgangsmåte da informasjonen som forsvinner ved fjerning av tilstedeværelse muligens kan erstattes med andre features som ble valgt bort i feature-utvelgingen. Derfor blir en ny feature-utvelging kjørt uten tilstedeværelse for å se hvilke features modellen foretrekker. Uten tilstedeværelse velger feature-utvelgingen *arbeidstid*, *time_cos*, *arbeidsdag*, *utetemp*, *time_sin* og *time* som det beste datasettet. Dette datasettet har en RMSE-verdi på 11.55. Sammenlignes dette datasettet med datasettet valgt i kapittel 8.6 viser det at feature-

utvelgingen tar med flere features som er forbundet med tidsstempelet for å erstatte mangelfull informasjon. Figur 46 illustrerer hvordan de ulike features påvirker RMSE-verdien på datasettet.



Figur 46: Feature-utvelging uten tilstedeværelse.

Etter at tilstedeværelse er fjernet kan det igjen være nyttig å kjøre en ny hyperparameteroptimalisering ved hjelp av grid search. Tabell 16 viser at justering av hyperparameterene gir en RMSE-verdi på 10.94. Dette er ikke fullt så godt resultat sammenlignet med tilstedeværelse (RMSE=10.23) som er forståelig siden modellen har mistet en viktig feature.

	Hyperparameter					**Evalueringsmål**		
Modell	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	**RMSE**	**MAE**	**Tid (s)**
ML1	13	65	0	2	0.00005	10.94	8.54	90.1

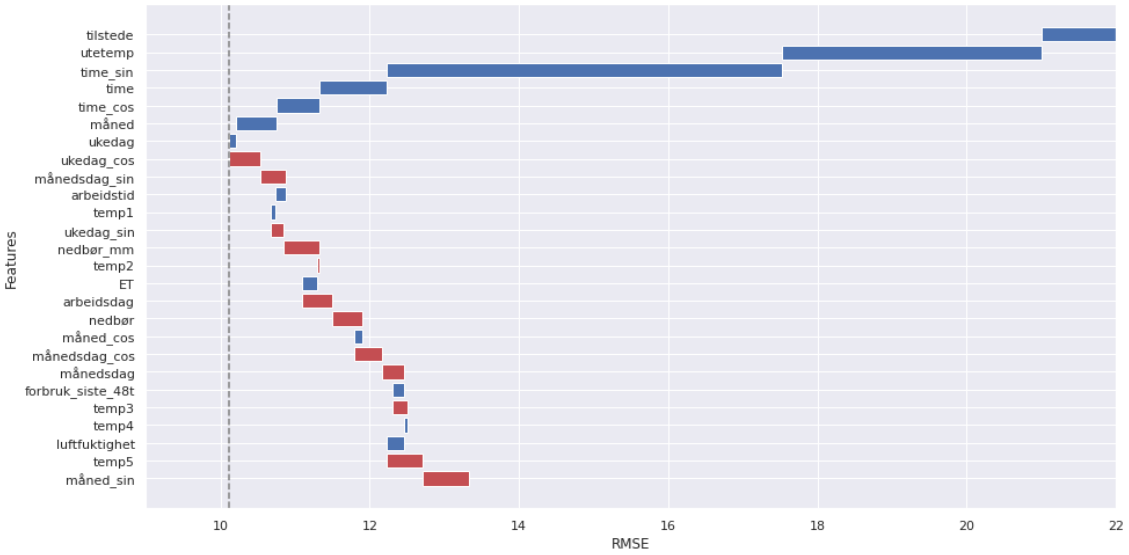
Tabell 16: Resultat av hyperparameteroptimalisering etter tilstedeværelse er fjernet i datasettet.

8.8.3 Prediksjon med og uten helg

Frem til nå har modellene blitt trent og testet på hele testsettet som inneholder både ukedager og helg. Som nevnt i kapittel 8.1 er dette gjort for å lage en generell modell som kan passe for alle bygg og ukedager. For Bangeløkka vil det være størst behov å redusere energi i ukedager, da det er et kontorbygg og har størst forbruk i ukedagene. På bakgrunn av dette vil det være interessant å utforske hvordan resultatet av modellen påvirkes av å bare predikere i ukedager. Konsekvensen av å redusere datasettet til bare ukedager, er at det vil redusere antall sesonger i datasettet fra en

uke og 24 timer til bare 24 timer (sett bort fra ett år som sesong) samtidig som mengde data vil bli redusert. Siden en sesong blir fjernet fra datasettet vil det minke kompleksiteten på problemet som forhåpentligvis gjør det enklere for modellen å predikere.

For predikere bare i ukedagene ble alle lørdager og søndager fjernet fra det originale datasettet. I kapittel 6.2 ble det tatt opp bekymringer rundt fjerning av data i tidsserier. Det ble i dette tilfellet fjernet all data i tidsrommet 00:00 lørdag til 00:00 mandag, altså all data for lørdag og søndag. Dette gjør at det blir en unaturlig overgang fra 23:59 fredag til 00:00 mandag. Derfor ble sekvensene som inneholder denne overgangen fjernet fra datasettet. Resultatet er da et datasett med bare ukedager (man-fre) uten ugyldige sekvenser. En slik endring minker datasettet med ca. 30%. Siden datasettet er endret må det kjøres en ny feature-utvelging. Det nye utvalget av features er som vist i figur 47: *tilstede*, *utetemp*, *time_sin*, *time*, *time_cos*, *måned*, *ukedag*. Dette nye datasettet har en RMSE på 10.12.



Figur 47: Feature-utvelging med bare ukedager i datasettet.

Etter at det nye datasettet er valgt kjøres det igjen et nytt grid search for å finne de beste hyperparameterne for dette datasettet. Resultatet av nye hyperparametere er vist i tabell 17. Tabellen viser at RMSE-verdien er 10.06 som er et bedre resultat i forhold til datasettet med både ukedager og helg (RMSE=10.23). Resultatet påvirkes av et mindre datasett, siden det er redusert med ca. 30%, men har også et enklere datasett da det nå bare inkluderer en sesong på 24 timer og ikke sesongen for hver uke.

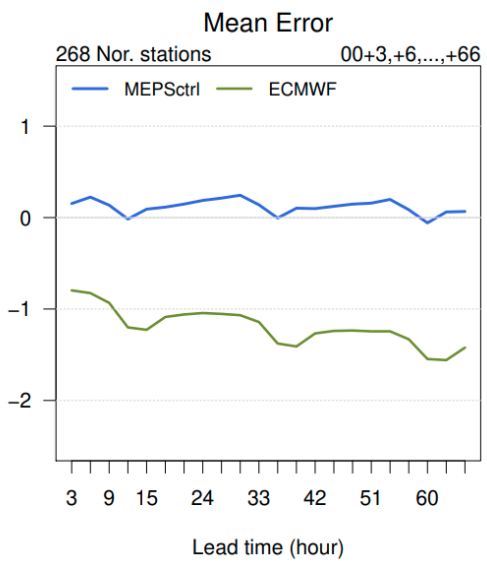
	Hyperparameter					**Evalueringsmål**		
Modell	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	**RMSE**	**MAE**	**Tid (s)**
ML1	30	15	0	2	0.0001	10.06	7.98	36.8

Tabell 17: Resultat av hyperparameteroptimalisering med bare ukedager i datasettet.

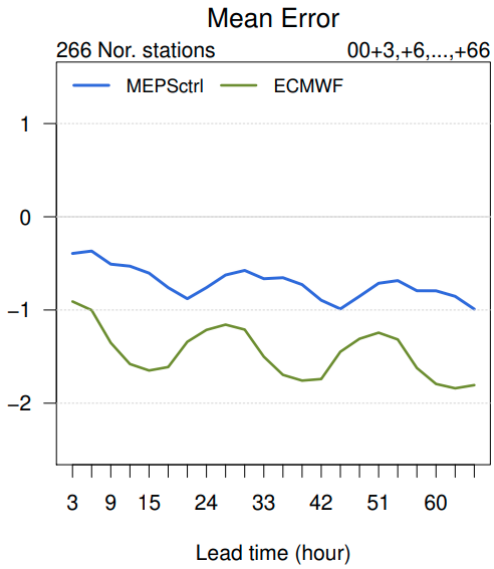
8.8.4 Påvirkning av usikkerheten i værmelding

I modelleringen av prediksjonsmodellen er det brukt virkelig utetemperatur på bygget. I virkeligheten finnes ikke denne verdien L samples frem i tid ved t, bare nåverdien og historikk på de virkelige verdiene er tilgjengelig. Dersom modellen blir satt i produksjon vil den bruke værmelding fra for eksempel MET som feature for å predikere energiforbruket. Det er ikke ønskelig å bruke slik data i selve modelleringen da usikkerheten fra prediksjonsmodellen til f.eks. MET vil påvirke modelleringen av prediksjonsmodellene i dette prosjektet. Dersom resultatet da er dårlig, vil det være vanskelig å vite om det er dataen fra MET som er dårlig predikert, eller om det er modellen i dette prosjektet som er dårlig modellert [69]. Det vil uansett være interessant å utforske hvordan bruken av data fra MET og dens usikkerhet kan påvirke modellen.

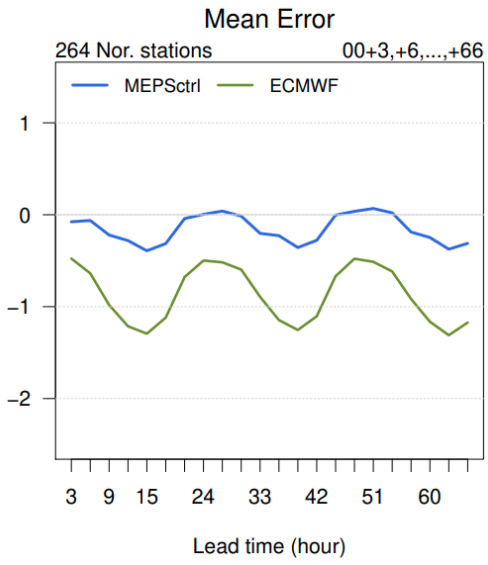
Etter samtale med Eldbjørg Moxnes i MET ble det gitt tilgang til rapportene *Verification of Operational Weather Prediction Models*. Dette er kvartalsrapporter som går gjennom ytelsen av prediksjonmodellen MET bruker for værmeldingsdata. Rapportene inneholder en god del statistikk, ytelsesmål og informasjon om de ulike modellene. Grafene i figur 48 viser avviket på prediksjonen frem i tid for ulike sesonger. MEPSctrl er samme modell som YR.no bruker til meteorologiske prediksjoner 2-3 dager frem i tid, men korrigerer dataen noe for bedre lokale prediksjoner [103]. ECMWF er en modell fra European Centre for Medium-Range Weather Forecasts som blir sammenlignet med MEPS [70]. Grafene viser at ved bruk av modellen MEPS kan det forventes et gjennomsnittlig avvik mellom ca. +0.2 $°C$ og -1 $°C$ for $L = [3...66]$ timer. Denne usikkerheten vil også direkte påvirke LSTM-modellen dersom denne dataen anvendes.



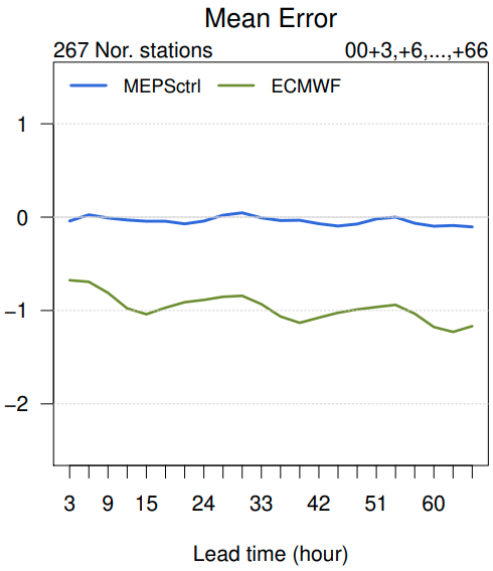
(a) Gjennomsnittlig avvik $°C$ des 2020 - feb 2021 [70].



(b) Gjennomsnittlig avvik $°C$ mars - mai 2021 [72].



(c) Gjennomsnittlig avvik $°C$ juni - aug 2021 [71].



(d) Gjennomsnittlig avvik $°C$ sep - nov 2021 [73].

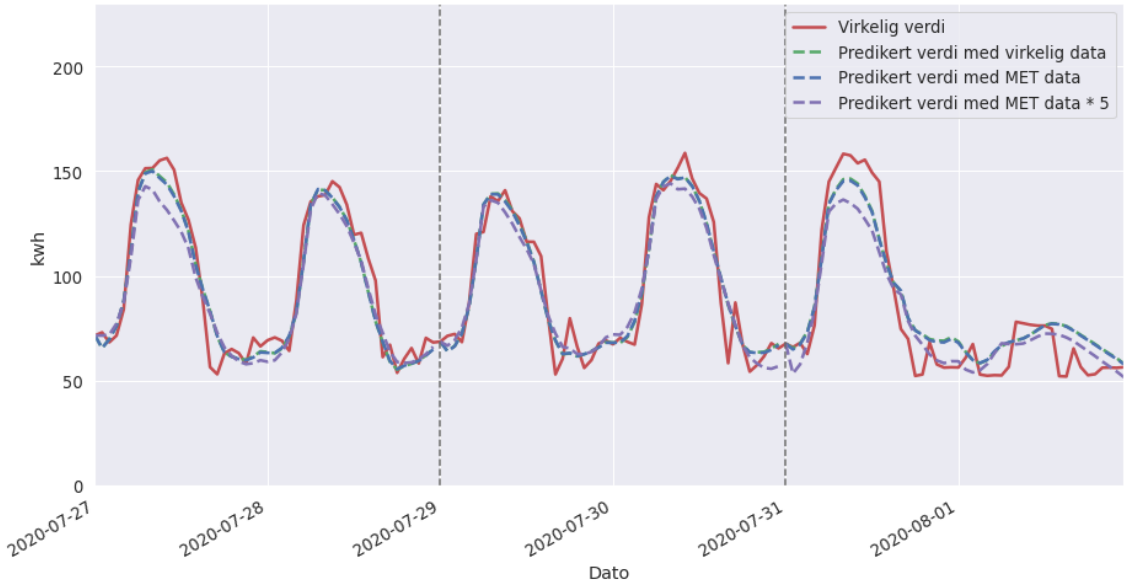
Figur 48: Gjennomsnittlig avvik i $°C$ på MetCoOp EPS (MEPS) modellen over ulike sesonger.

Ved å hente ut verdiene Mean Error fra grafene og deretter legge dette avviket på den virkelige utetemperaturen, kan det analyseres hvordan modellen påvirkes av usikkerheten i værmeldingen. Treningsdataen bruker fortsatt virkelige, historiske verdier for utetemperatur, men valideringsdataen bruker historiske verdier for utetemperatur addert med gjennomsnittlig avvik fra grafene. For å hente ut det gjennomsnittlige avviket ble grafen for mars-mai brukt siden det har størst avvik. Etter at den dataen var uthentet fra grafen ble det for hver periode (48 timer) i valideringsdataen lagt på det gjennomsnittlige avviket for den virkelige utetemperaturen. Figur 49 viser hvordan tidsserien for den nye utetemperatur blir etter at avviket er lagt til. Ved de grå, vertikale stiplede linjene, der modellen skal predikere $L = 48$ steg fremover, er det null avvik. Figuren viser også

at utover i perioden blir avviket større, noe som er naturlig siden det er vanskeligere å predikere lengre fram i tid og usikkerheten blir større.

Figur 49: Utetemperatur påvirket av gjennomsnittlig avvik fra MEPSctrl-modellen.

Etter at den nye dataen er lagt inn, trenes modellen som nevnt på den virkelige utetemperaturen mens valideringsdataen bruker utetemperaturen pluss avviket fra MEPSctrl-modellen. Figur 50 viser prediksjonen av energiforbruket med den virkelige utetemperaturen (grønn), sammenlignet mot virkelige forbruket (rød), og prediksjonen med utetemperatur pluss avviket fra MEPSctrl-modellen (blå). Figuren viser at det er minimalt med avvik mellom grønn og blå graf. Siden det er et gjennomsnittlig avvik som brukes vil det ikke reflektere store pluss og minus avvik da de kan utjevne hverandre. På grunn av dette er det i tillegg lagt til en ekstra graf som viser prediksjonen med utetemperatur pluss avviket fra MEPSctrl-modellen multiplisert med fem (lilla). Dette gjøres for å se hvordan modellen påvirkes av mer ekstreme avvik. Ut fra graf 50 er det tydelig at det gir et større avvik.



Figur 50: Prediksjon med og uten gjennomsnittlig avvik fra MEPSctrl-modellen.

Sammenlignes RMSE-verdiene for disse prediksjonene viser tabell 18 at modellen nesten ikke påvirkes av det gjennomsnittlige avviket, men påvirkes noe mer av ekstreme avvik, som er naturlig. Resultatet kan si noe om den forventede påvirkningen på modellen av usikkerheten i værmeldingen. Avviket som er oppgitt på MEPSctrl-modellen er lavt, og fra figur 50 påvirker det heller ikke modellen noe stort.

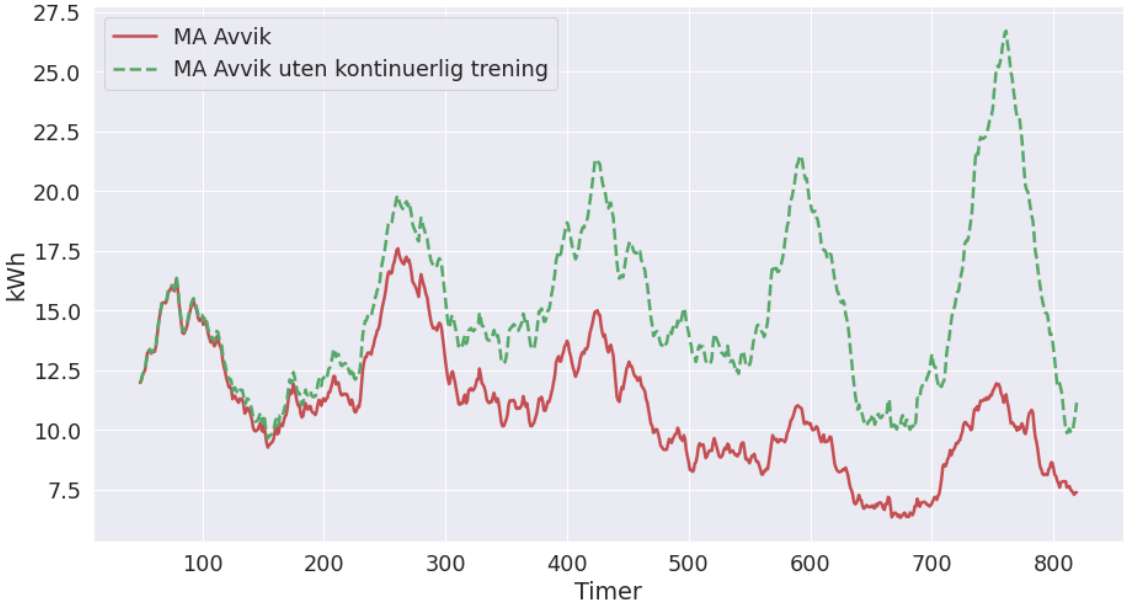
Testdata	**RMSE**
Predikert verdi med virkelig data	10.23
Predikert verdi med MET data	10.33
Predikert verdi med MET data · 5	13.42

Tabell 18: Resultat av prediksjon med forskjellige utetemperaturer.

8.8.5 Prediksjon uten kontinuerlig trening

Kontinuerlig trening er implementert for å forsikre at modellen hele tiden er trent på nyeste data. Dette vil forhåpentligvis gjøre at den kan predikere sin neste periode bedre, sammenlignet med uten kontinuerlig trening, da modellen har trent på den siste dataen som er mest relevant for neste prediksjon. I tillegg vil det forhåpentligvis gjøre at modellen i helhet blir bedre siden den vil få mer treningsdata tilgjengelig etter hvert som tiden går. For å teste om dette er tilfellet er det satt opp to modeller, en med kontinuerlig trening og en uten. Figur 51 viser et glidende gjennomsnitt (moving average) på avviket, $|y - \hat{y}|$, for de to modellene. Figuren viser at avviket er noe likt på begge i starten av kjøringen, men etter ca. 200 timer vises effekten av kontinuerlig trening. Den grønne grafen øker sitt avvik etter ca. 200 timer, men den røde minker sitt avvik. Dette bekrefter begge

hypotesene nevnt tidligere. Modellen gjør bedre prediksjoner for neste periode med kontinuerlig trening sammenlignet med uten kontinuerlig trening, samtidig som modellen gradvis blir bedre fordi den får mer treningsdata tilgjengelig. Grafen viser også at behovet for ny trening ligger rundt ca. 200 timer for denne modellen.



Figur 51: Glidende gjennomsnittlig avvik fra modell med og uten kontinuerlig trening.

8.9 Diskusjon

Prosessen gjennom dette kapittelet har vært iterativt. Det har blitt tatt valg gjennom kapittelet som har formet neste kapittel. Dette har ført til at mye av diskusjonene har blitt tatt inne i hvert delkapittel. For eksempel i første del av dette kapittelet ble det valgt å gå for en LSTM-modell etter å ha sammenlignet SARIMAX og LSTM. Dette valget ble tatt på bakgrunn av utførte tester og sammenligningen av ytelsesmålene. LSTM viste et bedre resultat på alle ytelsesmålene, samt at den hadde lavere kjøretid og mindre ressursforbruk. Dette førte til at SARIMAX-modellen ble lagt vekk, og fokuset ble på LSTM-modellen. Optimalt burde alle mulige kombinasjoner av modeller, hyperparameter, treningssett og valg av features blitt testet før det endelige valget ble tatt. Dette viste seg å være en stor utfordring på grunn av kompleksiteten av alle kombinasjonene. Ressursene tilgjengelig i dette prosjektet strakk ikke til for en slik operasjon, derfor ble prosessen en iterativ prosess, slik som det ble presentert i kapittel 7.1.

Gjennom denne delen av rapporten har de fleste avgjørelsene blitt tatt på bakgrunn av ytelsesmålene, der RMSE har blitt brukt som det avgjørende ytelsesmålet. Som nevnt tidligere straffer RMSE større avvik hardere enn MAE, noe som gjør den til et fornuftig ytelsesmål i denne sammenhengen, da ekstreme avvik vil være uheldig for modellens hensikt. Når det er sagt, har alle ytelsesmålene nevnt i kapittel 3.4.4 i tillegg til modelleringstiden og ressursforbruk blitt vurdert

for å kunne ta avgjørelsene.

Under testingen av modellene ble det brukt walking validation som valideringsmetode. Som vist i kapittel 3.4.10 oppdaterer den modellen med nyeste data gjennom testingen. I en produksjonssammenheng vil det være mer naturlig å bruke samme teknikk, men ha et fast vindu som treningsdata; for eksempel de to siste årene. Dette fører til at modellen ikke får for mye data eller data som er utdatert. Under utvikling av modellen i dette kapittelet var tilgjengelig data noe begrenset, derfor ble all data brukt på trening under walking validation.

Andre punkter som er verdt å tenke på med et treningssett, er hvordan vedlikehold eller eventuelle endringer på anlegget påvirker modellen. Slike hendelser vil naturlig påvirke dataen modellen vil bruke for trening. Derfor bør det finnes muligheter for å markere data der anlegget har blitt utsatt for endringer eller vedlikehold, slik at den kan ekskluderes i et treningssett.

En modell som kun bruker ukedager ble testet. Denne modellen viste seg å ha bedre resultat, noe som var forventet siden problemet blir enklere å løse med bare en sesong i dataen. Selv om modellen ble bedre, er den ikke generell da det må spesifiseres at den ikke kan brukes på helger, noe som kan være tilfellet for noen bygg. Det var et ønske fra GK å ha en generell modell i første omgang, men ut fra undersøkelsene og testene gjort i dette kapittelet kan det være lurt å ha flere modeller for ulike bygg og bruksmønster.

8.10 Konklusjon

For langtidsprediksjonsmodellen ble det valgt å teste modelltypene LSTM og SARIMAX basert på tidligere arbeid som har blitt gjort. LSTM ble valgt som modelltype da den beste LSTM-modellen hadde bedre resultat på samtlige ytelsesmål sammenlignet med den beste SARIMAX-modellen. LSTM-modellen ble videreutviklet til tre nye modeller der den beste modellkonfigurasjonen ble brukt for å finne de mest aktuelle features i feature-utvelgingen. Til slutt ble det gjort flere eksperimenter som er av interesse for GK, blant annet hvordan mengde med treningsdata påvirker prediksjonene, prediksjoner på spesielle dager i uken, og hvordan manglende data påvirker modellen. Dette oppfyller kravene fra GK der det først og fremst var ønskelig å utforske om det var mulig å lage en langtidsprediksjonsmodell med dataen tilgjengelig og deretter modellere en slik modell. Den tilgjengelige dataen er utforsket og analysert, modellen er bygd, og resultatet tilfredsstiller behovet til GK nevnt i problemstillingen.

9 Korttidspredikering

9.1 Beskrivelse av modellen

Kortidsprediksjonsmodellen skal som nevnt brukes for å unngå effekttopper. Effekttopper defineres som det høyeste energiforbruket innenfor en time i den aktuelle måneden. For denne modellen vil det være viktig å vite hvor mange minutter den er innenfor den aktuelle timen. For å holde kontroll på dette defineres H:M, der H:M angir klokkeslettet for nåtiden, t. H angir antall timer inn i en dag og M antall minutter innenfor timen H. For eksempel om tiden er H:10 og med samplingstiden fem minutter vil y_t beskrive det gjennomsnittlige effektforbruket fra H:05 til H:10.

Målet til modellen er å predikere energien brukt i den aktuelle timen. For å approksimere energiforbruket med et diskret signal kan Riemann sum anvendes [108]. E vil videre i rapporten brukes som benevning for det virkelige energiforbruket i den aktuelle timen, og $\hat{E}$ vil bli brukt om det predikerte energiforbruket for den aktuelle timen. For å approksimere E for en periode D ved hjelp Riemann sum kan formel 45 brukes.

$$E \approx p_1 T_1 + p_2 T_2 + ... + p_{D-1} T_{D-1} + p_D T_D \qquad [kWh] \tag{45}$$

Effektforbruket, p, er hentet ut fra GK Cloud med samplingstid på fem minutter. Når dataen blir hentet med en samplingstid på fem minutter vil disse verdiene bli gjennomsnittet for de siste fem minuttene. Da det er gjennomsnittsverdier, kan p omdefineres til $\bar{p}$. $\bar{p}$ brukes for å regne ut E etter formel 45. Siden det er $\bar{p}$ som er den ønskede predikerte verdien, defineres $y = \bar{p}$. T er lik for alle samples siden det er fast samplingstid, $T = \frac{5}{60}$. For å approksimere energiforbruket for den aktuelle timen, kan formel 46 brukes ved timeslutt, H:60 = H+1:00. Approksimasjonen vil bli mer presis ved lavere samplingstid, T.

$$E_t \approx \frac{y_{t-1} + y_{t-2} + ... + y_{t-11} + y_{t-12}}{12} \tag{46}$$

Det er ønskelig at den endelige modellen for hver sampling predikerer den forventende effekttoppen innenfor den aktuelle timen. Det er altså E som det til slutt er ønskelig å finne. Jo nærmere modellen kommer slutten på en time vil $\hat{E} \to E$ siden det da ikke er nødvendig å predikere hele timen. For eksempel dersom den har kommet 30 minutter inn i en time, er det aktuelt å bare predikere $\hat{y}_{t+1}...\hat{y}_{t+6}$ siden den da vet $y_t...y_{t-5}$. Prediksjonen, $\hat{E}$, for den aktuelle timen, H, som modellen predikerer ved H:30 kan da skrives lik formel 47.

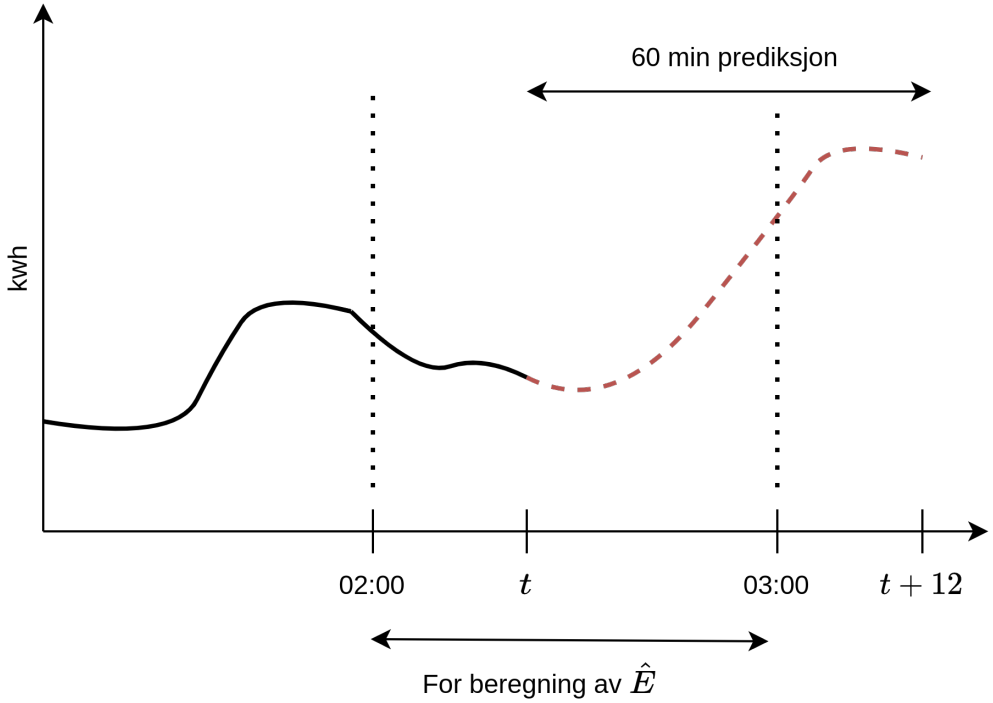
$$\hat{E}_t = \frac{y_{t-5}...y_t + \hat{y}_{t+1}...\hat{y}_{t+6}}{12} \tag{47}$$

Selv om det i eksempelet ovenfor ved H:30 bare er nødvendig å predikere $\hat{y}_{t+1}...\hat{y}_{t+6}$ vil modellen

hele tiden predikere en time frem i tid for hvert tidssteg. Det betyr at ved H:30 predikerer modellen $\hat{y}_{t+1}...\hat{y}_{t+12}$, men den vil bare bruke $\hat{y}_{t+1}...\hat{y}_{t+6}$ sammen med akkumulerte verdier av y i den aktuelle timen for å finne $\hat{E}$. Ved å bruke H:M som ble definert tidligere, kan det settes opp en generell formel for prediksjon av $\hat{E}$ med samplingstid, $T = \frac{5}{60}$, vist i formel 48. Det er viktig å huske at samplingstiden er fem minutter slik at M vil bare ha verdiene $A = [5, 10, ...55, 60]$.

$$\hat{E}_t = \frac{\sum_{i=-(\frac{M}{T\cdot 60}-1)}^{0} y_{t+i} + \sum_{j=1}^{12-\frac{M}{T\cdot 60}} \hat{y}_{t+j}}{12} \qquad M \in A \tag{48}$$

Figur 52 viser hvordan modellen for hvert tidssteg predikerer en time frem i tid, men bare verdiene innenfor timen brukes for beregning av $\hat{E}$.



Figur 52: Illustrasjon av hvordan modellen brukes for å finne $\hat{E}$.

Ut fra det tidligere arbeidet som er gjort ble det valgt å teste ut modelltypene ARIMAX og LSTM. Det vil bli implementert modeller som predikerer $\hat{y}$, men også modeller som direkte kan predikere $\hat{E}$. For å sammenligne disse er det nødvendig med et felles ytelsesmål. Derfor defineres $RMSE_{\hat{y}}$ og $RMSE_{\hat{E}}$. $RMSE_{\hat{E}}$ er et felles ytelsesmål mellom modellene, mens $RMSE_{\hat{y}}$ kan brukes som sammenligningsgrunnlag mellom de modellene som bare predikerer $\hat{y}$. Det er mindre beregningstid på $RMSE_{\hat{y}}$ i forhold til $RMSE_{\hat{E}}$ på grunn av at formel 48 ikke anvendes, derfor er $RMSE_{\hat{y}}$ et foretrukket ytelsesmål. Som nevnt vil noen av modellene direkte predikere $\hat{E}$ som gjør at det må finnes et ytelsemål som $RMSE_{\hat{E}}$. Begge ytelsesmålene er presentert i formel 49 og 50.

$$RMSE_{\hat{y}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2} \tag{49}$$

$$RMSE_{\hat{E}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(E_i - \hat{E}_i)^2} \tag{50}$$

Ut fra det tidligere arbeidet som er undersøkt på korttidsprediksjon er det ønskelig å utforske modelltypene ARIMAX og LSTM. Etter at en av modelltypene er valgt, blir den valgte modelltypen brukt for videreutvikling og eksperiment.

9.2 ARIMAX-modell

9.2.1 Implementasjon

Implementasjonen av ARIMAX er lik implementasjonen av SARIMAX i kapittel 8.2.1, bortsett fra at sesongkomponentene i SARIMAX, $(P, D, Q)_m$, ikke er tatt i bruk.

9.2.2 Testing av hyperparameter

Tabell 19 viser resultatet av de fem beste modellkonfigurasjonene ved bruk av grid search sortert etter RMSE. På bakgrunn av det som ble diskutert i kapittel 8.2.2 er det ikke ønskelig å bruke AIC som det avgjørende ytelsesmålet. Fra tabell 19 er modellen ARIMAX(3,1,1) tatt ut for videre sammenligning.

	Hyperparameter			**Evalueringsmål**			
Test	**p**	**d**	**q**	**AIC**	**RMSE$_{\hat{y}}$**	**MAE**	**Tid (s)**
54	3	1	1	507550.41	12.24	6.96	815.4
53	3	1	0	509305.96	12.25	6.91	807.8
58	3	2	1	509331.74	12.25	6.93	908.3
6	0	1	1	520720.89	12.27	6.9 2	771.9
11	0	2	2	520737.68	12.28	6.93	893.7

Tabell 19: Resultat av hyperparameteroptimalisering ARIMAX.

9.3 LSTM-modell

9.3.1 Implementasjon

Implementasjonen av LSTM-modellen er lik implementasjonen vist i 8.3.1 bortsett fra at det i dette tilfellet er samplingstiden $T = \frac{5}{60}$ time. For å finne en optimal S ble det brukt LSTM-modellen der $S = [1...12]$ ble testet og evaluerte med RMSE. Den S-verdien som ga best resultat var $S = 9$.

9.3.2 Testing av hyperparameter

Tabell 20 viser resultatet av grid search sortert etter RMSE. Test nr. 116 av LSTM-modellen gir det beste resultatet. Denne modellkonfigurasjonen velges derfor for videre sammenligning.

	Hyperparameter					**Evalueringsmål**		
#	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	**RMSE**$_{\hat{y}}$	**MAE**	**Tid**
116	30	65	0.5	1	0.00005	11.39	7.98	504.7
114	30	65	0.4	1	0.00005	11.43	7.96	517.8
99	30	65	0.3	1	0.00005	11.46	7.97	468.7
102	30	50	0.2	1	0.00005	11.50	7.95	548.6
101	30	50	0.2	1	0.0001	11.58	8.14	482.9

Tabell 20: Resultat av hyperparameteroptimalisering LSTM.

9.4 Valg av type modell

9.4.1 Sammenligning av LSTM- og ARIMAX-modeller

Testingen av modelltypene LSTM og ARIMAX har gitt den beste LSTM-modellen, nr. 116, med RMSE$_{\hat{y}}$=11.39 og RMSE$_{\hat{y}}$=12.24 for den beste ARIMAX-modellen. ARIMAX har et tidsforbruk på 815.4 sekunder mens LSTM-modellen kun bruker 504.7 sekunder. Siden LSTM har lavere tidsforbruk og RMSE$_{\hat{y}}$ velges LSTM-modellen for videre utvikling.

9.4.2 MK1

For å benchmarke modellene er det ønskelig å bruke en enkel matematisk modell for ha noe sammenligningsgrunnlag. Dette gjøres først og fremst på grunn av en antagelse av at de features som er tilgjengelige vil ha en liten påvirkning på prediksjonen når prediksjonsvinduet er såpass lite. Det er ikke ønskelig å bruke avanserte modeller, slik som nevrale nettverk om det ikke er nødvendig, da en enkel matematisk modell vil kreve mindre ressursforbruk og modellerings-, implementerings-, og vedlikeholdstid.

MK1 er ikke en maskinlæringsmodell, men en matematisk modell. Modellen går ut fra at det aktuelle effektforbruket ved t, y_t, fortsetter ut timen. Med denne forutsetningen er det mulig å predikere $\hat{E}$ ved å bytte ut det predikerte leddet fra formel 48 med formel 51. Her er T samplingstiden og M antall minutter inn i timen.

$$\sum_{j=1}^{12-\frac{M}{T\cdot 60}} \hat{y}_{t+j} = y_t \cdot (12 - \frac{M}{T \cdot 60}) \qquad M \in A \tag{51}$$

9.5 Videreutvikling av LSTM-modell

Nå som LSTM-modellen er valgt som modelltype skal det utforskes muligheter for å videreutvikle denne modelltypen til ulike modeller.

9.5.1 MK2

MK2 er modellen som er implementert i implementasjonskapittelet, 8.3.1, og den modellen som ble sammenlignet og testet med ARIMAX i kapittel 9.3.2. Denne modellen bruker en sekvens av features S bakover for det aktuelle tidssteget, sammen med X_{t+1}, for å predikere $\hat{y}_{t+1}$ frem til $t+L$ steg fremover. Denne modellen er også gitt som ML1 i kapittel 8.5. Modellen er presentert i formel 52.

$$\begin{aligned} \hat{y}_{t+1} &= f([X_{t+1}...X_{t+1-S}]) \\ &\vdots \\ \hat{y}_{t+L} &= f([X_{t+L}...X_{t+L-S}]) \end{aligned} \tag{52}$$

9.5.2 MK3

Denne modellen er forholdsvis lik MK2, men i tillegg til X brukes også tidligere verdier for y. Siden prediksjonsvinduet her er 60 minutter, ser den på verdiene i siste time for å predikere en time fremover. Denne modellen er også forklart som ML3 i kapittel 8.5.3. Modellen er gitt i formel 53.

$$\begin{aligned} \hat{y}_{t+1} &= f([[y_{t-L}...y_{t-L-S}], [X_{t+1}...X_{t+1-S}]]) \\ &\vdots \\ \hat{y}_{t+L} &= f([[y_t...y_{t-S}], [X_{t+L}...X_{t+L-S}]]) \end{aligned} \tag{53}$$

Når denne modellen trenes må lagverdier for y legges inn som en ny feature. I dette tilfellet vil det bli lagverdi på 12 samples som utgjør 60 minutter.

9.5.3 MK4

Denne modellen er detaljert forklart som ML2 i kapittel 8.5.2. Kort forklart predikerer den neste tidssteg som blir brukt som en feature for videre prediksjoner. Modellen er vist slik:

$$\begin{aligned}
\hat{y}_{t+1} &= f([y_t ... y_{t-S}], [X_{t+1} ... X_{t+1-S}]) \\
\hat{y}_{t+2} &= f([\hat{y}_{t+1}, y_t ... y_{t+1-S}], [X_{t+2} ... X_{t+2-S}]) \\
\hat{y}_{t+3} &= f([\hat{y}_{t+2}, \hat{y}_{t+1}, y_t ... y_{t+2-S}], [X_{t+3} ... X_{t+3-S}]) \\
&\vdots \\
\hat{y}_{t+L} &= f([\hat{y}_{t+(L-1)} ... \hat{y}_{t+(L-1)-S}], [X_{t+L} ... X_{t+L-S}])
\end{aligned} \tag{54}$$

Denne modellen, ulikt de andre modellene, må for hvert tidssteg legge til nye features-verdier, siden det er forrige tidsstegs prediksjon som skal brukes for neste predikering. Det at modellen må predikere for hvert tidssteg, samt legge inn nye features for hvert tidssteg, gjør at denne modellen krever vesentlig mer kjøretid. Når denne modellen trenes må lag verdier for y legges inn som en ny feature. I dette tilfellet vil det bli lag-verdi på en sample.

9.5.4 MK5

I stedet for å predikere $\hat{y}$ vil denne modellen direkte predikere $\hat{E}$. Modellen vil i tillegg til å bruke features som er nevnt kapittel 7.4 utnytte en feature som sier hvor mye energi som har blitt brukt i den aktuelle timen frem til t. Denne feature kan implementeres lik formel 55. Formelen vil summere forbruket brukt frem til H:M. Deretter deles summen på antall samples som har vært frem til t. Dette vil da gi et mål på hva energiforbruket for den aktuelle timen er frem til M i timen.

$$x_t^E = \sum_{i=-(\frac{M}{T \cdot 60}-1)}^{0} y_{t+i} \cdot T \qquad M \in A \tag{55}$$

Funksjonen for prediksjonsmodellen er da som følger:

$$\begin{aligned}
\hat{E}_{t+1} &= f([x_t^E ... x_{t-S}^E], [X_{t+1} ... X_{t+1-S}]) \\
\hat{E}_{t+2} &= f([x_{t+1}^E ... x_{t+1-S}^E], [X_{t+2} ... X_{t+2-S}]) \\
&\vdots \\
\hat{E}_{t+L} &= f([x_{t+(L-1)}^E ... x_{t+(L-1)-S}^E], [X_{t+L} ... X_{t+L-S}])
\end{aligned} \tag{56}$$

9.5.5 Evaluering av nye LSTM-modeller

Etter at alle modellene er definert og implementert skal de testes. Det er nå ikke lenger mulig å bruke ytelsemålet med bare y og $\hat{y}$ siden det er implementert en modell som direkte predikerer $\hat{E}$.

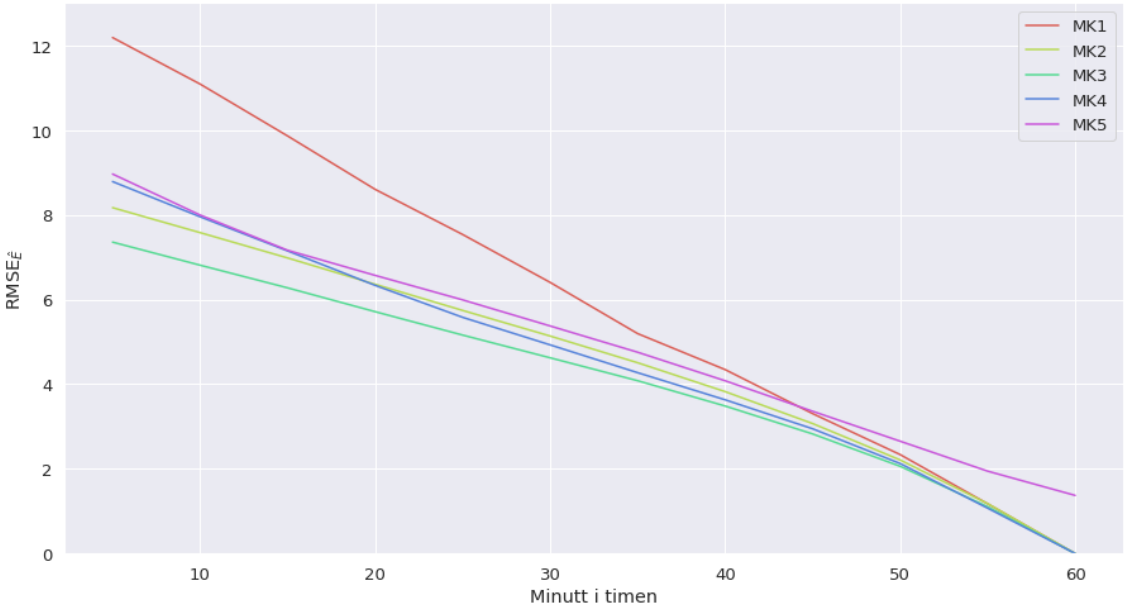
Det er mulig å finne $\hat{E}$ for MK2, MK3 og MK4, etter å ha funnet $\hat{y}$, ved bruk av formel 48.

Ved å bruke grid search blir alle modellene, bortsett fra MK1, testet med flere hyperparameter-konfigurasjoner. For å regne ut $\text{RMSE}_{\hat{E}}$ blir formel 50 brukt. Etter at en modellkonfigurasjon er testet, blir den sjekket for overtilpasning. Alle modellene under er derfor klarert for overtilpasning. Tabell 21 viser de beste modellkonfigurasjonene sortert etter $\text{RMSE}_{\hat{E}}$ på valideringssettet.

	Hyperparameter					**Evalueringsmål**	
Modell	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	$\textbf{RMSE}_{\hat{E}}$	**Tid (s)**
MK1	-	-	-	-	-	5.21	2.2
MK2	30	65	0.5	1	0.00005	4.98	504.7
MK3	30	75	0.5	1	0.00005	4.35	503.9
MK4	20	25	0	1	0.00005	4.56	971.7
MK5	20	15	0	1	0.00005	5.02	422.8

Tabell 21: Resultat av hyperparameteroptimalisering på modellene.

Grupperes prediksjonsavviket på hver modell etter minutter i timen er resultatet plottet i figur 53. Her er $\text{RMSE}_{\hat{E}}$ regnet ut for hvert minutt for alle modellene, i stedet for hele perioden. Plottet viser at at alle modellene, bortsett fra MK5, går mot null $\text{RMSE}_{\hat{E}}$. Tanken bak MK5 var at modellen med feature x_t^E skulle ha nok informasjon om hva forbruket så langt i timen var, og deretter kunne nærme seg null avvik mot timeslutt. Sammenlignet med de andre modellene som regner ut $\hat{E}$ ved hjelp av y og $\hat{y}$ har MK5 et avvik ved timeslutt. Ut fra plottet er MK3 den beste modellen gjennom hele timen.



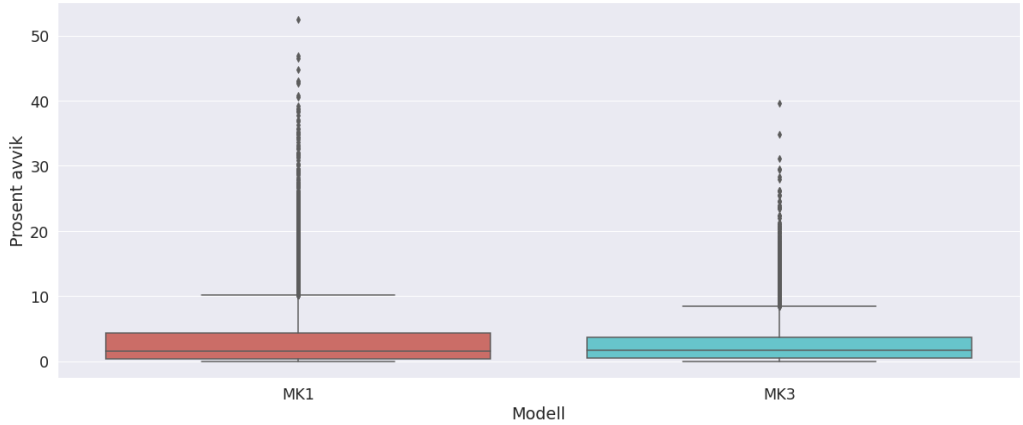
Figur 53: $\text{RMSE}_{\hat{E}}$ for hvert minutt innenfor en time for de ulike modellene.

Dersom modellenes treningstid sammenlignes, bruker naturligvis MK1 minst tid, siden den ikke trenger noen form for trening. På bakgrunn av dette vil det være interessant å se om forskjellen i presisjonen på modell MK1 og MK3 overgår tidsforskjellen mellom de. For å få et mer konkret mål beregnes prosentavvik for begge modellene ut fra formel 57.

$$prosentavvik = \frac{|E - \hat{E}|}{E} \tag{57}$$

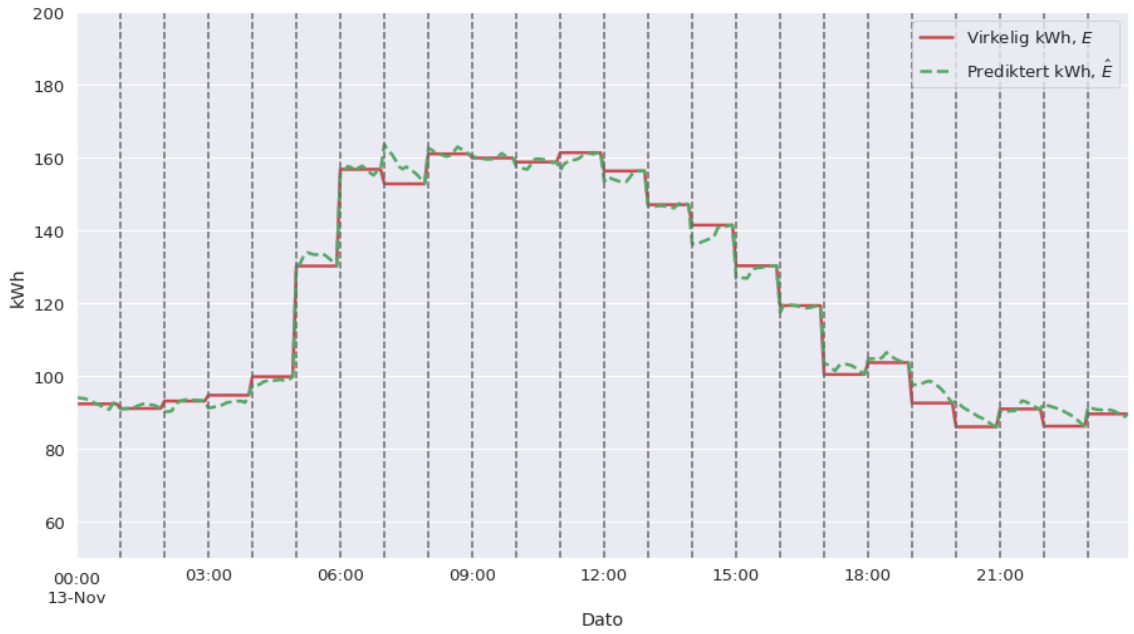
Det gjennomsnittlige prosentavviket er på 3.31% for MK1 og 2.68% for MK3. Ut fra gjennomsnittsverdiene ser det ved første øyekast ikke ut som det er store forskjeller mellom modellene, og at MK1 heller burde velges som endelig modell. Ved å sammenligne det gjennomsnittlige prosentavviket for begge modellene, har MK1 23.55% større gjennomsnittlig prosentavvik en hva MK3 har. Analyseres prosentavviket mellom de tidspunktene det er størst sjanse for en effekttopp, 05:00-15:00, har MK1 44.97% større gjennomsnittlig prosentavvik sammenlignet med MK3.

Gjennomsnittverdier er en dårlig metode for å representere utliggere, noe som er svært kritisk i denne sammenhengen. Ved å analysere et boksplot, vist i figur 54, viser det helt klart at MK1 har flere og større utliggere i prosentavviket. Plottet viser at MK1 har prosentavvik på over 50%. Det er ugunstig om modellen har et avvik på 50% på en effekttopp, siden det er den største effekttoppen i måneden som ofte blir brukt i utregningen av effektleddet. Det betyr i praksis at en slik utligger i prediksjonen kan sette prisen på effektleddet for hele måneden. Utliggere på prediksjonen er derfor noe som i større grad bør unngås, og er i denne sammenhengen mye viktigere enn gjennomsnittlig prosentavvik og tidsforbruk. Bakgrunn av dette og figur 53 velges derfor MK3 som den endelige modellen.



Figur 54: Boksplot av prosent-avviket for MK1 og MK3.

Figur 55 viser en tilfeldig dag der MK3 har blitt brukt til å predikere effekttopper. Figuren viser at det ofte starter med et større avvik i starten av timen, da modellen må predikere $[\hat{y}_{t+1}...\hat{y}_{t+11}]$. Mot timeslutt konvergerer prediksjonen mot det virkelige energiforbruket da modellen har verdiene $[y_t...y_{t-12}]$ som kan finne det korrekte energiforbruket for timen.



Figur 55: Prediksjon av $\hat{E}$ (grønn) for hvert tidssteg sammenlignet E (rød).

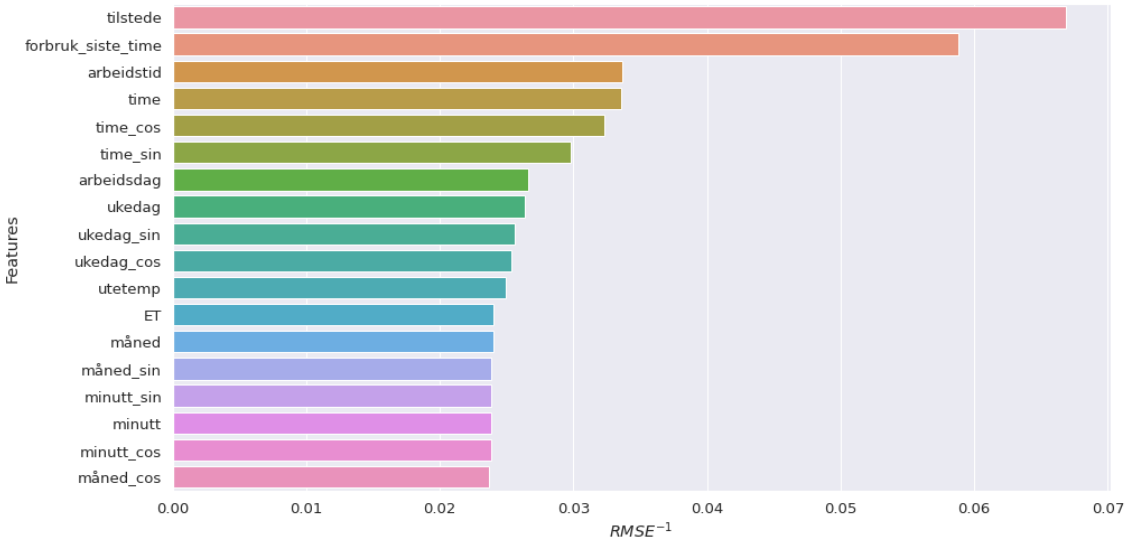
9.6 Feature-utvelging

Nå som en modell er valgt vil neste steg være å finne de riktige features for modellen. Som nevnt tidligere er feature-utvelging en tidskrevende jobb. Sammenlignet med modelleringen for langtidsprediksjon er det lengre modelleringstid grunnet lavere samplingstid og dermed større datasett. For å begrense tidsforbruket noe er det forsøkt å utelukke noen features fra tabell 8 i kapittel 6.3 som kan antas å ikke ha stor påvirkning på prediksjonen i denne modellen. Features som *fuktighet*, *nedbør_mm* og *nedbør* har samplingstid på bare en time. Det er mulig å resample disse verdiene til fem minutter for å bruke de i denne modellen, men det er antatt at disse features med den samplingstiden de har, ikke har stor påvirkning på prediksjonen for denne modellen. I tillegg er det antatt at features som *temp1* - *temp5* heller ikke har stor påvirkning på prediksjonen i tidsrommet som denne modellen predikerer. De blir derfor heller ikke med i feature-utvelgingen. Disse antagelsene kommer fra arbeidet som har blitt gjort tidligere i denne rapporten og samtaler med domeneeksperter på energi og bygg. *månedsdag* viste seg i kapittel 8.6 som en lite relevant feature. Siden denne modellen har kortere prediksjonsvindu blir *månedsdag* også ansett som ikke relevant for denne modellen. De ulike features som modellen da kan velge er vist i tabell 22.

Variabelnavn	Beskrivelse	Verdiområde
forbruk siste time	Effektforbruk i bygg siste time	-
utetemp	Utetemperatur for bygg	-
tilstede	Tilstedeværelse i bygg	-
minutt	Hvilken minutt i timen	0 - 59
time	Hvilken time i døgnet	0 - 23
ukedag	Hvilken dag i uken	1 - 7
måned	Hvilken måned	1 - 12
måned_sin	Sinus med periode år	-1 - 1
måned_cos	Cosinus med periode år	-1 - 1
ukedag_sin	Sinus med periode uke	-1 - 1
ukedag_cos	Cosinus med periode uke	-1 - 1
time_sin	Sinus med periode dag	-1 - 1
time_cos	Cosinus med periode dag	-1 - 1
minutt_sin	Sinus med periode time	-1 - 1
minutt_cos	Cosinus med periode time	-1 - 1
arbeidsdag	Arbeidsdag eller ikke	0 - 1
arbeidstid	Arbeidstid eller ikke	0 - 1
ET	ET verdi	-

Tabell 22: Oversiktstabell over all dataen uthentet.

For å få ett overblikk over hvor viktig hver feature er, brukes en og en feature i modellen, MK3, og sammenligner RMSE-verdien, vist i figur 56. Siden MK3 blir brukt, er RMSE-verdiene dette kapittelet utregnet av $\hat{y}$ og ikke $\hat{E}$. Dette er gjort på grunn av at det kreves noe mer beregningstid og finne $\hat{E}$ ut av $\hat{y}$ og y og modellen i seg selv predikerer $\hat{y}$. Ved å se på inverse av $RMSE_{\hat{y}}$-verdien vil den med høyest verdi være viktigere for modellen dersom bare en feature skal brukes. Ikke overraskende er *tilstedeværelse* og *forbruket siste time* viktige features. Noe mer overraskende er at meteorologiske data som ET og *utetemp* ikke er ansett som viktige features.



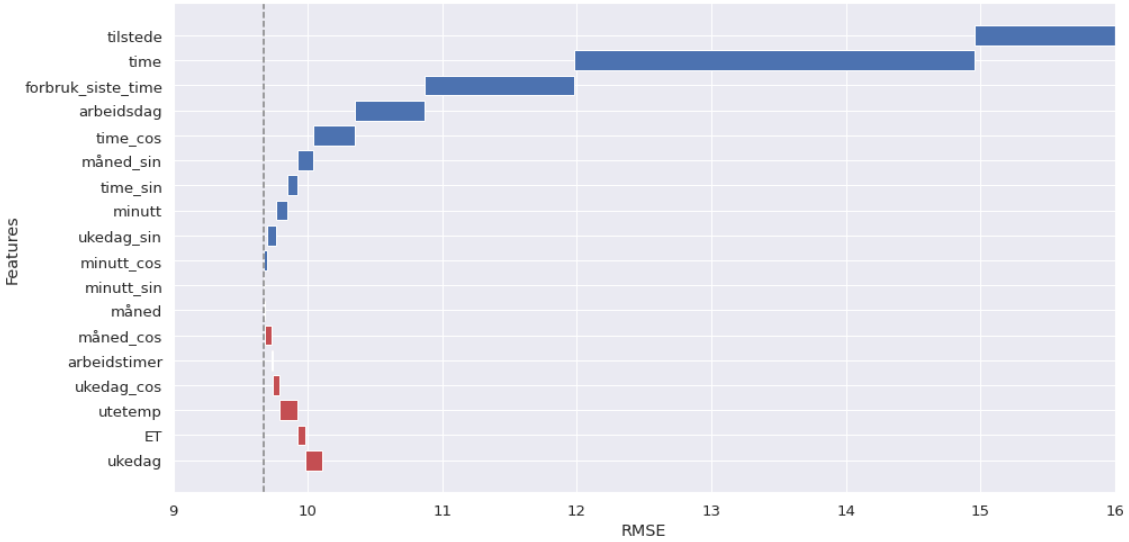
Figur 56: Viktighet av en og en feature for korttidspredikering. Ytelsesmålet $RMSE_{\hat{y}}$ er anvendt i denne figuren.

Som nevnt i kapittel 8.6 er det ikke nødvendigvis de øverste i figur 56 som danner det beste datasettet da flere features kan beskrive det samme. For å finne et godt datasett brukes feature-utvelgingsmetoden *forward selection* presentert i kapittel 3.4.11. Denne metoden er som nevnt tidligere en grådig algoritme, så den vil nødvendigvis ikke finne det optimale datasettet, men med tanke på kjøretiden som kreves for en optimal metode er det ansett som godt nok resultat. Dersom *forward selection* kjøres med datasettet vist i tabell 22 og med modell MK3 blir resultatet for valg av features vist i tabell 23 og visualisert i figur 57.

	Features																		
#	tilstede	time	forbruk_siste_time	arbeidsdag	time_cos	måned_sin	time_sin	minutt	ukedag_sin	minutt_cos	minutt_sin	måned	måned_cos	arbeidstimer	ukedag_cos	utetemp	ET	ukedag	$RMSE_{\hat{y}}$
1	x																		14.96
2	x	x																	11.98
3	x	x	x																10.87
4	x	x	x	x															10.35
5	x	x	x	x	x														10.04
6	x	x	x	x	x	x													9.92
7	x	x	x	x	x	x	x												9.85
8	x	x	x	x	x	x	x	x											9.76
9	x	x	x	x	x	x	x	x	x										9.70
10	x	x	x	x	x	x	x	x	x	x									**9.67**
11	x	x	x	x	x	x	x	x	x	x	x								9.67
12	x	x	x	x	x	x	x	x	x	x	x	x							9.68
13	x	x	x	x	x	x	x	x	x	x	x	x	x						9.73
14	x	x	x	x	x	x	x	x	x	x	x	x	x	x					9.74
15	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				9.79
16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			9.92
17	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		9.98
18	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	10.11

Tabell 23: Feature-utvelging ved bruk av wrapper metoden forward selection.

Figur 57 viser hvordan datasettet påvirker RMSE-verdien når en og en feature blir lagt til i datasettet etter forward selection metoden. Etter at *minutt_cos* er lagt til i datasettet viser figur 57 at RMSE-verdien blir verre, og det er da ikke nødvendig å legge til flere features med hensyn på RMSE-verdien. Den grå, stiplede linjen representerer den laveste RMSE-verdien.



Figur 57: Viktighet av en og en feature for korttidspredikering. X-aksen er $RMSE_{\hat{y}}$.

Som vist i figur 57 er utetemperatur og ET noen av de minst viktigste features å ha med i datasettet da de blir valgt nesten sist. Det kan være flere forklaringer for det, blant annet at forbruket siste time kan beskrive nåværende forbruksnivå i forhold til været såpass bra at utetemperaturen ikke er nødvendig. I tillegg vil ikke utetemperaturen variere noe stort innenfor en time, som er maksimum på prediksjonsvinduet.

9.7 Resultat - Endelig modell

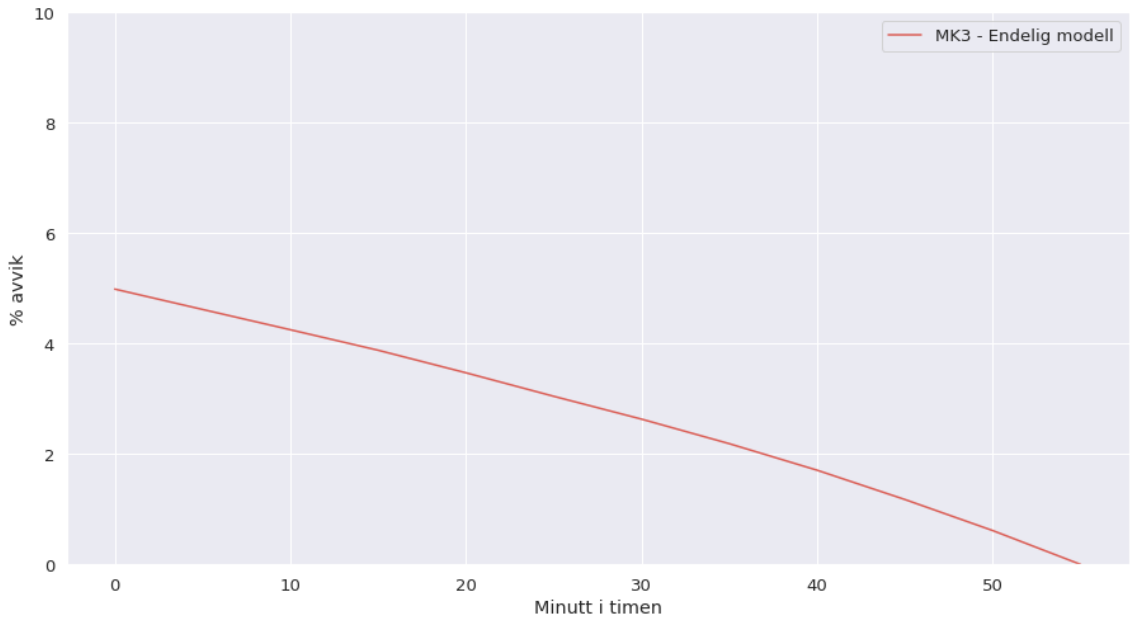
Etter feature-utvelging optimaliseres modellen en siste gang med nye hyperparametere. Dette gjøres ved å kjøre en ny grid search med datasettet i tabell 24. Beste parametere for denne modellen er vist i tabell 25. Som siste evaluering er modellen testet på testsettet der $\text{RMSE}_{\hat{E},test}$=4.40 som er noe dårligere enn $\text{RMSE}_{\hat{E},val}$=3.78. Dette er forventet resultat og viser at modellen generaliserer seg godt. Figur 58 viser at modellen har et gjennomsnittlig avvik på ca. 5% på prediksjonen av den aktuelle timens energiforbruk i starten på timen, før den gradvis går mot 0% avvik i slutten på timen.

Variabelnavn	Beskrivelse
tilstede	Tilstedeværelse i bygg
time	Hvilken time i døgnet
forbruk siste time	Effektforbruk i bygg siste time
arbeidsdag	Arbeidsdag eller ikke
time_{cos}	Cosinus med periode dag
måned_{sin}	Sinus med periode år
time_{sin}	Sinus med periode dag
minutt	Hvilken minutt i timen
ukedag_{sin}	Sinus med periode uke
minutt_{cos}	Cosinus med periode time

Tabell 24: Det beste datasett for korttidsprediksjon.

	Hyperparameter					Evalueringsmål		
Modell	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	$\textbf{RMSE}_{\hat{E},val}$	$\textbf{RMSE}_{\hat{E},test}$	**Tid (s)**
MK3	30	40	0.2	1	0.00005	3.78	4.40	540.2

Tabell 25: Resultat av MK3 på testsettet.



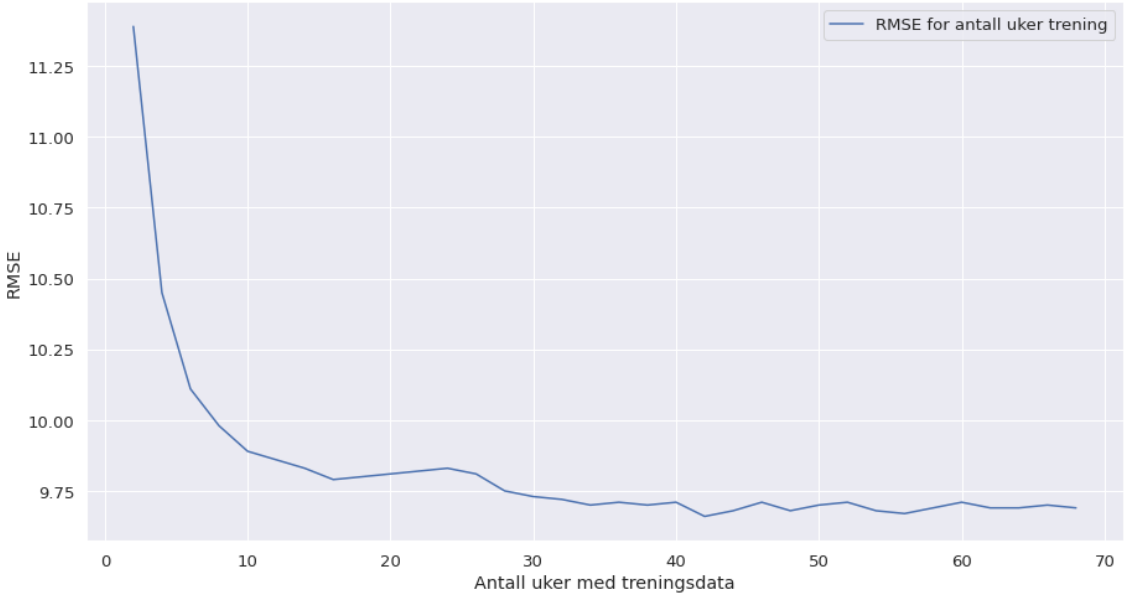
Figur 58: Prosent avvik ved prediksjon av energiforbruket i den aktuelle timen for den endelige modellen.

9.8 Eksperiment

I likhet med langtidsprediksjonsmodellen vil det være nyttig å kjøre noen eksperimenter på denne modellen også. Siden det er de samme eksperimentene som blir kjørt for begge modellene, vil beskrivelsen og bakgrunnen for eksperimentene i dette kapittelet være det samme som i kapittel 8.8 og er derfor ikke utdypet noe vesentlig her.

9.8.1 Mengde av treningsdata

Figur 59 viser $\text{RMSE}_{\hat{y}}$ predikert av MK3 og datasett med ulike mengder treningsdata. Sammenlignet med langtidsprediksjonsmodellen krever kortidsprediksjonsmodellen mindre treningsdata før prediksjonene er gode. Ved ca. 40 uker med treningsdata begynner ytelsesmålene å flate ut og mer treningsdata gjør minimale ytelsesforbedringer.



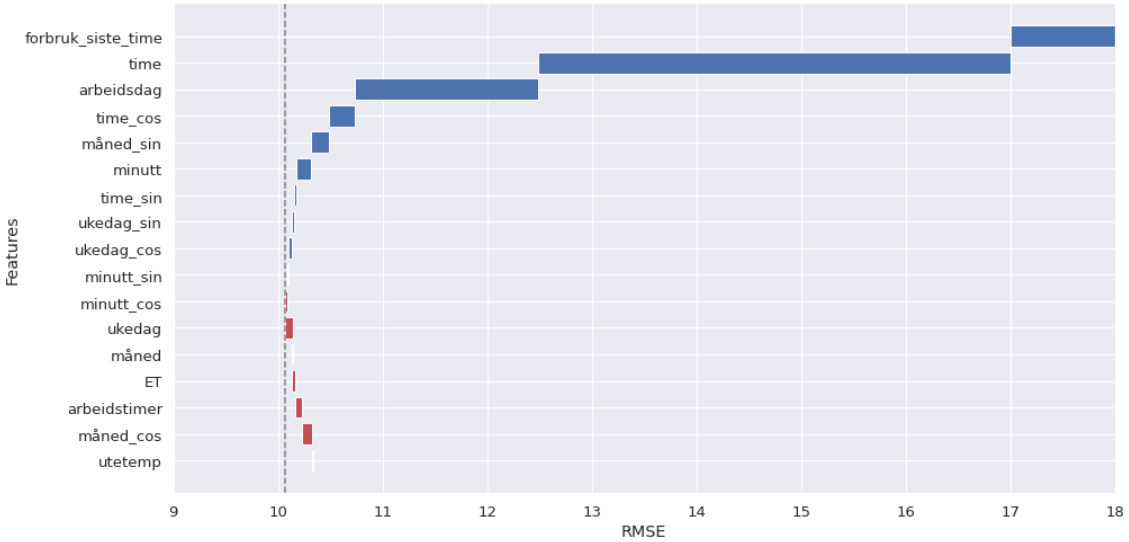
Figur 59: Hvordan ulik mengde treningsdata påvirker ytelsen av modellen. y-aksen er $\text{RMSE}_{\hat{y}}$.

9.8.2 Manglende tilstedeværelse

Uten tilstedeværelse velger feature-utvelgingen features lik tabell 26. Hvordan hver feature påvirker $\text{RMSE}_{\hat{y}}$ når de legges inn i datasettet er vist i figur 60. Det nye datasettet gir en $\text{RMSE}_{\hat{y}} = 10.09$ og $\text{RMSE}_{\hat{E}} = 3.97$ som er noe dårligere sammenlignet med datasettet med tilstedeværelse ($\text{RMSE}_{\hat{E}} = 3.78$). En hyperparameteroptimalisering av modellen gir ikke noen endringer da den velger den samme modellen som allerede er optimalisert. Modellkonfigurasjonen som ga dette ytelsesmålet er gitt i tabell 27.

Variabelnavn	Beskrivelse
forbruk siste time	Effektforbruk i bygg siste time
time	Hvilken time i døgnet
arbeidsdag	Arbeidsdag eller ikke
time_{cos}	Cosinus med periode dag
måned_{sin}	Sinus med periode år
minutt	Hvilken minutt i timen
time_{sin}	Sinus med periode dag
ukedag_{cos}	Cosinus med periode uke
ukedag_{sin}	Sinus med periode uke
minutt_{sin}	Cosinus med periode time

Tabell 26: Det beste datasett for korttidsprediksjon.



Figur 60: Feature-utvelging uten tilstedeværelse.

	Hyperparameter					Evalueringsmål	
Modell	**Epoch**	**Nevroner**	**Dropout**	**Lag**	α	$\textbf{RMSE}_{\hat{E}}$	$\textbf{RMSE}_{\hat{y}}$
MK3	30	40	0.2	1	0.00005	3.97	10.09

Tabell 27: Hyperparameteroptimalsering på MK3 med datasettet uten tilstedeværelse.

9.9 Diskusjon

Det har gjennom hele dette delkapittelet blitt diskutert og tatt avgjørelsen som har ført til neste steg i kapittelet. Selv om mye av diskusjonene allerede er tatt er det noen punkter som krever

ekstra oppmerksomhet.

Først og fremst bør tilgjengeligheten på gode features for en slik modell diskuteres ytterligere. Uten tilstedeværelse er det bare forbruk siste time som er en eksogen feature for å finne $\hat{E}$. Det hadde vært nyttig å ha flere eksogene features tilgjengelig som kan beskrive effektforbruket innenfor et kortere prediksjonsvindu. I kapittel 9.6 ble det funnet ut at utetemperatur ikke hadde stor innvirkning for prediksjonene av $\hat{E}$. For å finne bedre features for denne modellen burde det blitt satt av tid til analyse av forbruket på et bygg, og se hva som virkelig skjer da en effekttopp oppstår. Dette er utenfor avgrensningene for denne oppgaven, men noen undersøkelser ble likevel gjort. Etter samtale med domeneeksperter hos GK ville kalenderstyrt og tidsstyrte komponenter på bygg antagelig vært et godt sted å starte. Dersom det er faste perioder på styringene av disse komponentene, kan det være at modellen klarer å finne disse tidene selv, ved bruk av tidsstempel-features. Dersom disse komponentene blir uregelmessig styrt, vil ikke modellen vite når komponentene bruker energi, og dermed vil heller ikke klare å predikere dette forbruket uten noe mer informasjon.

Det er viktig å tenke på hva som skjer dersom et system skal styres etter informasjonen gitt fra en modell som predikerer energien. Problemet oppstår når det skal brukes ny treningsdata som den selv har påvirket. Et eksempel kan være at modellen sender informasjon til systemet om at en effekttopp kommer kl. 07:00, og dette skjer flere dager på rad. Systemet vil dermed begynne å kutte effekter kl. 07:00. Når modellen senere får historiske energidata som treningsdata vil modellen se at forbruket kl. 07:00 er lavere enn tidligere, og vil dermed trenes slik at den forventer et lavere forbruk kl. 07:00. Modellen kan derfor påvirke sin egen treningsdata, noe som er uheldig. En metode for å unngå dette er å markere dataen som har blitt styrt av systemet, på grunn av informasjon fra modellen, og ekskludere denne dataen fra treningsdataen. Mengden av data som blir markert kan antas å være minimal, siden det er lite sannsynlig at en effektopp oppstår ofte.

Tidligere i dette kapittelet ble det satt opp en enkel matematisk modell for å sammenligne LSTM-modellene. Selv om modellen var svært enkel, presterte den relativt godt. Det hadde vært interessant å arbeide noe mer med den matematiske modellen for å se på mulighetene for å forbedre den. Det ville ha blitt mindre arbeid både innenfor vedlikehold og implementasjon dersom en matematisk modell kunne ha erstattet en maskinlæringsmodell. Denne antagelsen bygger også på at MK3 uten tilstedeværelse bare har en eksogen variabel som er det tidligere forbruket. I denne rapporten blir det ikke gjort noe mer arbeid rundt den matematiske modellen da oppgaven er avgrenset til å teste maskinlæringsmodeller.

9.10 Konklusjon

I dette kapittelet har muligheten til å predikere energiforbruket for en time for hvert tidssteg innenfor den aktuelle timen blitt sett på. Først ble det testet og evaluert to modelltyper, LSTM og ARIMAX. LSTM-modellen viste seg å ha best prediksjon, samt lavere treningstid. Det ble vi-

dereutviklet fire LSTM-modellkonfigurasjoner med ulik oppbygging og datasett. Her ble modellen MK3 evaluert som den beste. Denne modellen ble brukt for feature-utvelging der resultatet er vist i kapittel 9.6. Etter en siste modelloptimalisering med nye features ble modellen brukt i to eksperimenter. Det ene eksperimentet testet hvordan modellen ble påvirket av mengde treningsdata. Her ble det funnet ut at ca. 40 uker med treningsdata er nødvendig for å få gode prediksjoner med denne modellen. I tillegg ble modellen testet med datasettet uten tilstedeværelse, der resultatet viste seg å være noe dårligere. Den endelige modellen er testet på et testsett der det viser noe dårligere resultat, som er forventet. Samtidig viser testresultatet at modellen generaliserer godt og at den ikke er overtilpasset. Det konkluderes med at det er bygd en modell som kan, med tilgjengelig data gjennom GK, predikere energiforbruket for den aktuelle timen i hvert tidssteg. Modellen gir et resultat som tilfredsstiller GK sitt behov nevnt i problemstillingen.

10 Hovedkonklusjon

GK hadde et ønske om å utvikle to prediksjonsmodeller. Den første modellen er en langtidsprediksjonsmodell som predikerer energibehovet to dager frem i tid. Denne modellen kan anvendes til å predikere behovet for energiforskyving og energilagring. I tillegg ønsket de en kortidsprediksjonsmodell som predikerer energiforbruket for den aktuelle timen. Sistnevnte modell kan sammen med et styresystem anvendes for å unngå effekttopper. Det er gitt delkonklusjoner for begge disse modellene i deres respektive kapitler.

Rapporten har tatt for seg en grundig analyse av den tilgjengelige dataen og brukt denne analysen for feature-uthenting og datarensing. Modelltypene har blitt valgt ut fra en analyse av det tidligere arbeidet som har blitt gjort på dette feltet. Over 6000 modellkonfigurasjoner har blitt testet gjennom en systematisk og ryddig fremgangsmåte, bestående av teknikker innenfor teori og praktisk anvendt modellering av maskinlæringsmodeller. De valgte modellene har blitt optimalisert, samtidig som et optimalt datasett har blitt funnet. Disse modellene har igjen blitt brukt til eksperiment i samarbeid med GK for å vise ytelsen, fleksibiliteten og begrensingene til modellen. I tillegg har det blitt skrevet et kapittel om hvordan disse modellene bør implementeres som en pipeline i et system. Rapporten viser at med dataen GK har tilgjengelig i dag er det mulig å utvikle begge modellene som GK ønsker. Modellene er utviklet og testet. Fra testene konkluderes det med at modellene kan anvendes til prediksjon av energiforbruk. Etter samtale med GK oppfyller modellene de ønskelige kravene som ble gitt og konklusjonen er dermed at problemstillingen og oppgaven er løst.

Selv om oppgaven var å modellere disse to modellene bør det nevnes at prosjektet har gitt verdi på flere felt. GK mener at verdien i å se fremgangsmåten av hvordan modellene utvikles, i tillegg til å se nytten av god data har vært svært viktig. Som en følge av dette har GK sett på mulighetene for å ta opp mer av dataen de allerede har tilgjengelig på bygget, samt ta i bruk smartere sensorer for riktig datafangst.

10.1 Videre arbeid

Rapporten viser hvordan det er mulig å sette opp modeller for både langtids- og kortidspredikering. Dersom det er aktuelt å jobbe videre med dette prosjektet er det flere punkter som kan nevnes som videre arbeid av denne rapporten.

Noe mer undersøkelse rundt tilgjengelig data fra ulike bygg fra GK sin portefølje ville vært hensiktsmessig å bruke noe tid på. I denne rapporten ble Bangeløkka valgt som bygg da det har GK Cloud som tilgjengeliggjør dataen i bygget. GK Cloud blir stadig implementert i nye bygg som muligens har mer sensorikk, og dermed mer data. I tillegg har GK, parallelt med skriving av denne rapporten, fått prosjekter der det er fokus på sensorikk og datainnsamling. Disse nye prosjektene

har sensorikk som for eksempel solstyrke gjennom solcellepanel. Dette er en feature som hadde vært veldig interessant å ha med i dette prosjektet. Det har også kommet frem underveis i arbeidet med denne rapporten at flere bygg har kalender- eller tidsstyrt HVAC system. Denne dataen ligger i hovedsak på PLS-nivå og er ikke sendt opp til GK Cloud, noe som har gjort denne type data utilgjengelig i dette prosjektet. Noe mer arbeid her ville også vært gunstig for å se om slik type data kan være gode features.

GK er i gang med å ta i bruk eSight, som er et energioppfølgingssystem. eSight kategoriserer energimålene etter bruksområde slik at det er mulig å se spesifikt hvilke type komponenter som bruker energi i bygget. Å tilgjengeliggjøre denne dataen for modellering er viktig arbeid som bør undersøkes.

Tilstedeværelse har vist seg å være en viktig feature for begge modellene. GK bør derfor se på mulighetene for å få tak i forventet tilstedeværelse frem i tid. Dette kan hentes fra booking, kalendere, eller ved å se på mulighetene for å lage en prediksjonsmodell for tilstedeværelse i bygget. Tilstedeværelse er innhentet i systemet gjennom komponenter fra Lindinvent. Lindinvent er et selskap som lager komponenter for behovsstyrt ventilasjon. Det viser seg at Lindinvent også har et API med mer tilgjengelig informasjon enn det som er tatt inn i systemet. Det ville vært et nyttig bidrag å få tilgang på det API'et, og se hvilken data som er tilgjengelig der.

Modellen er ikke implementert i et produksjonsmiljø i dette prosjektet. Oppgaven begrenser seg til modelleringen av maskinlæringsmodellene. Det ble i kapittel 5 skrevet om hvordan modellene kan og bør implementeres i et system. Dette er arbeidet som ikke er gjort og blir derfor nevnt her som videre arbeid.

Referanser

[1] Dean Abbott. *Applied predictive analytics: Principles and techniques for the professional data analyst*. John Wiley & Sons, 2014.

[2] *About Us — Energy Management Software — eSight Energy*. https://www.esightenergy.com/about/. (Accessed on 05/31/2022).

[3] Hirotugu Akaike. «A new look at the statistical model identification». I: *IEEE transactions on automatic control* 19.6 (1974), s. 716–723.

[4] Yuichiro Anzai. *Pattern recognition and machine learning*. Elsevier, 2012.

[5] *Azure Machine Learning – maskinlæring som en tjeneste — Microsoft Azure*. https://azure.microsoft.com/nb-no/services/machine-learning/. (Accessed on 03/08/2022).

[6] *Basic Feature Engineering With Time Series Data in Python*. https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/. (Accessed on 05/04/2022).

[7] Yoshua Bengio, Ian Goodfellow og Aaron Courville. *Deep learning*. Bd. 1. MIT press Massachusetts, USA: 2017.

[8] Christoph Bergmeir, Rob J Hyndman og Bonsoo Koo. «A note on the validity of cross-validation for evaluating autoregressive time series prediction». I: *Computational Statistics & Data Analysis* 120 (2018), s. 70–83.

[9] Christopher M Bishop mfl. *Neural networks for pattern recognition*. Oxford university press, 1995, s. 332.

[10] *Bli kjent med GK — GK*. https://www.gk.no/om-oss. (Accessed on 02/24/2022).

[11] André B Bondi. «Characteristics of scalability and their impact on performance». I: *Proceedings of the 2nd international workshop on Software and performance*. 2000, s. 195–203.

[12] Salah Bouktif mfl. «Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches». I: *Energies* 11.7 (2018), s. 1636.

[13] G.E.P. Box, G.M. Jenkins og WISCONSIN UNIV MADISON Dept. of STATISTICS. *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing. Holden-Day, 1970. ISBN: 9780816210947.

[14] Jason Brownlee. *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks*. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. (Accessed on 02/08/2022). Des. 2018.

[15] Jason Brownlee. *Dropout Regularization in Deep Learning Models With Keras*. https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/. (Accessed on 02/08/2022). Jun. 2016.

[16] Jason Brownlee. *Taxonomy of Time Series Forecasting Problems*. https://machinelearningmastery.com/taxonomy-of-time-series-forecasting-problems/. (Accessed on 01/30/2022). Aug. 2019.

[17] Jason Brownlee. *Understand the Impact of Learning Rate on Neural Network Performance*. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/. (Accessed on 02/08/2022). Jan. 2019.

[18] Mete Çelik, Filiz Dadaşer-Çelik og Ahmet Şakir Dokuz. «Anomaly detection in temperature data using dbscan algorithm». I: *2011 international symposium on innovations in intelligent systems and applications*. IEEE. 2011, s. 91–95.

[19] Vitor Cerqueira, Luis Torgo og Igor Mozetič. «Evaluating time series forecasting models: An empirical study on performance estimation methods». I: *Machine Learning* 109.11 (2020), s. 1997–2028.

[20] Young Tae Chae mfl. «Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings». I: *Energy and Buildings* 111 (2016), s. 184–194.

[21] Haibin Cheng mfl. «Multistep-ahead time series prediction». I: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2006, s. 765–774.

[22] Delson Chikobvu og Caston Sigauke. «Regression-SARIMA modelling of daily peak electricity demand in South Africa». I: *Journal of Energy in Southern Africa* 23.3 (2012), s. 23–30.

[23] Giuseppe Ciaburro og Gino Iannace. «Machine Learning-Based Algorithms to Knowledge Extraction from Time Series Data: A Review». I: *Data* 6.6 (2021), s. 55.

[24] CrowdFlower. «Data Scientis Report 2017». I: (2017).

[25] Herui Cui og Xu Peng. «Short-term city electric load forecasting with considering temperature effects: an improved ARIMAX model». I: *Mathematical Problems in Engineering* 2015 (2015).

[26] Torgeir Ericson Dag Spilde Synne Krekling Lien Ingrid Magnussen. «Energibruk i Norge mot 2035». I: *Norges vassdrags- og energidirektorat* (2018).

[27] Rachel Draelos. *Best Use of Train/Val/Test Splits, with Tips for Medical Data*. https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/. (Accessed on 01/30/2022). Sep. 2019.

[28] Martin Ester mfl. «A density-based algorithm for discovering clusters in large spatial databases with noise». I: AAAI Press, 1996, s. 226–231.

[29] *Feature – Wikipedia*. https://no.wikipedia.org/wiki/Feature. (Accessed on 03/08/2022).

[30] *Forecasting at Uber: An Introduction*. https://eng.uber.com/forecasting-introduction/. (Accessed on 04/03/2022).

[31] *Forecasting Time Series Data with Multiple Seasonal Periods.* https://tanzu.vmware.com/content/blog/forecasting-time-series-data-with-multiple-seasonal-periods. (Accessed on 05/12/2022).

[32] *Forecasting: Principles and Practice (2nd ed) - 12.9 Dealing with missing values and outliers.* https://otexts.com/fpp2/missing-outliers.html. (Accessed on 04/26/2022).

[33] *Forecasting: Principles and Practice (3rd ed) - 13.9 Dealing with outliers and missing values.* https://otexts.com/fpp3/missing-outliers.html. (Accessed on 04/26/2022).

[34] *From Neutrinos to Data Science.* http://blog.davidkaleko.com/feature-engineering-cyclical-features.html. (Accessed on 04/05/2022).

[35] *Frost API.* https://frost.met.no/index.html. (Accessed on 02/22/2022).

[36] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2019.

[37] Felix A Gers, Jürgen Schmidhuber og Fred Cummins. «Learning to forget: Continual prediction with LSTM». I: *Neural computation* 12.10 (2000), s. 2451–2471.

[38] *Get to Know Azure — Microsoft Azure.* https://azure.microsoft.com/en-us/overview/. (Accessed on 05/11/2022).

[39] *GK Gloud — Smart styring av bygg — GK.* https://www.gk.no/fag-og-tjenester/gk-cloud. (Accessed on 05/11/2022).

[40] Knut Gluggvasshaug. «Reduksjon av energibruk i bygninger knyttet til nær-eller fjernvarmesystemer ved hjelp av et intelligent IKT-verktøy». Masteroppg. NTNU, 2018.

[41] Klaus Greff mfl. «LSTM: A search space odyssey». I: *IEEE transactions on neural networks and learning systems* 28.10 (2016), s. 2222–2232.

[42] *Grådig algoritme – Wikipedia.* https://no.wikipedia.org/wiki/Gr%C3%A5dig_algoritme. (Accessed on 03/08/2022).

[43] Trevor Hastie mfl. *The elements of statistical learning: data mining, inference, and prediction.* Bd. 2. Springer, 2009.

[44] Sepp Hochreiter og Jürgen Schmidhuber. «Long short-term memory». I: *Neural computation* 9.8 (1997), s. 1735–1780.

[45] *holidays · PyPI.* https://pypi.org/project/holidays/. (Accessed on 02/22/2022).

[46] *How To Backtest Machine Learning Models for Time Series Forecasting.* https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/. (Accessed on 03/02/2022).

[47] *How to Get Reproducible Results with Keras.* https://machinelearningmastery.com/reproducible-results-neural-networks-keras/. (Accessed on 04/26/2022).

[48] *How to Handle Missing Timesteps in Sequence Prediction Problems with Python*. https://machinelearningmastery.com/handle-missing-timesteps-sequence-prediction-problems-python/. (Accessed on 05/06/2022).

[49] *Hva er Azure – Microsoft-skytjenester — Microsoft Azure*. https://azure.microsoft.com/nb-no/overview/what-is-azure/. (Accessed on 05/06/2022).

[50] *Hva er en ET-kurve?* https://www.miljolare.no/meis/et_kurve.php. (Accessed on 02/16/2022).

[51] *Hva er U-verdi? - Bygg og Bevar*. https://www.byggogbevar.no/pusse-opp/vindu-doer/artikler/hva-er-u-verdi. (Accessed on 05/27/2022).

[52] *HVAC Definition & Meaning - Merriam-Webster*. https://www.merriam-webster.com/dictionary/HVAC. (Accessed on 05/27/2022).

[53] Robin John Hyndman og George Athanasopoulos. *Forecasting: Principles and Practice*. English. 2nd. Australia: OTexts, 2018.

[54] *Intro to data structures — pandas 1.4.1 documentation*. https://pandas.pydata.org/docs/user_guide/dsintro.html. (Accessed on 03/08/2022).

[55] Rishee K Jain mfl. «Forecasting energy consumption of multi-family residential buildings using support vector regression: Investigating the impact of temporal and spatial monitoring granularity on performance accuracy». I: *Applied Energy* 123 (2014), s. 168–178.

[56] Tae-Young Kim og Sung-Bae Cho. «Predicting residential energy consumption using CNN-LSTM neural networks». I: *Energy* 182 (2019), s. 72–81.

[57] *Klimaservicesenter*. https://klimaservicesenter.no/kss/om-oss/om-kss. (Accessed on 02/22/2022).

[58] Max Kuhn, Kjell Johnson mfl. *Applied predictive modeling*. Bd. 26. Springer, 2013.

[59] Víctor Manuel Landassuri-Moreno mfl. «Single-step-ahead and multi-step-ahead prediction with evolutionary artificial neural networks». I: *Iberoamerican Congress on Pattern Recognition*. Springer. 2013, s. 65–72.

[60] Ingrid Jæger Landsnes. «Muligheter med kontinuerlig energikostnadsminimering basert på prisprediksjon og lastpåvirkning». Masteroppg. NTNU, 2021.

[61] David J Leinweber. «Stupid data miner tricks: overfitting the S&P 500». I: *The Journal of Investing* 16.1 (2007), s. 15–22.

[62] Petro Liashchynskyi og Pavlo Liashchynskyi. «Grid search, random search, genetic algorithm: a big comparison for NAS». I: *arXiv preprint arXiv:1912.06059* (2019).

[63] Nengbao Liu, Vahan Babushkin og Afshin Afshari. «Short-term forecasting of temperature driven electricity load using time series and neural network model». I: *Journal of Clean Energy Technologies* 2.4 (2014), s. 327–331.

[64] *LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras — by Ryan T. J. J. — Analytics Vidhya — Medium*. https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2. (Accessed on 05/30/2022).

[65] *machine learning - Encoding features like month and hour as categorial or numeric? - Data Science Stack Exchange*. https://datascience.stackexchange.com/questions/17759/encoding-features-like-month-and-hour-as-categorial-or-numeric. (Accessed on 04/05/2022).

[66] *Machine Learning Mastery*. https://machinelearningmastery.com/. (Accessed on 03/07/2022).

[67] Spyros Makridakis og Michele Hibon. «The M3-Competition: results, conclusions and implications». I: *International journal of forecasting* 16.4 (2000), s. 451–476.

[68] *Makridakis Competitions - Wikipedia*. https://en.wikipedia.org/wiki/Makridakis_Competitions#cite_note-m3-1-2. (Accessed on 02/18/2022).

[69] Deyslen Mariano-Hernández mfl. «A Data-Driven Forecasting Strategy to Predict Continuous Hourly Energy Demand in Smart Buildings». I: *Applied Sciences* 11.17 (2021), s. 7886.

[70] Frank Thomas Tveter Mariken Homleid Gunnar Noer og Lene Østvand. «Verification of Operational Weather Prediction Models December 2020 to February 2021». I: *Meteorologisk institutt* (2021).

[71] Frank Thomas Tveter Mariken Homleid Gunnar Noer og Lene Østvand. «Verification of Operational Weather Prediction Models June to August 2021». I: *Meteorologisk institutt* (2021).

[72] Frank Thomas Tveter Mariken Homleid Gunnar Noer og Lene Østvand. «Verification of Operational Weather Prediction Models March to May 2021». I: *Meteorologisk institutt* (2021).

[73] Frank Thomas Tveter Mariken Homleid Gunnar Noer og Lene Østvand. «Verification of Operational Weather Prediction Models September to November 2021». I: *Meteorologisk institutt* (2021).

[74] *Masteroppgave - NTNU*. https://i.ntnu.no/masteroppgave. (Accessed on 02/27/2022).

[75] Andrew V Metcalfe og Paul SP Cowpertwait. *Introductory time series with R*. Springer, 2009.

[76] Tom Mitchell. «Machine learning». I: (1997).

[77] *MLOps: Machine Learning Engineering — Towards Data Science*. https://towardsdatascience.com/ml-ops-machine-learning-as-an-engineering-discipline-b86ca4874a3f. (Accessed on 02/24/2022).

[78] *nan - C++ Reference*. https://www.cplusplus.com/reference/cmath/nan-function/. (Accessed on 05/27/2022).

[79] Vladimir Nasteski. «An overview of the supervised machine learning methods». I: *Horizons. b* 4 (2017), s. 51–62.

[80] *Nettleie for forbruk - NVE*. https://www.nve.no/reguleringsmyndigheten/regulering/nettvirksomhet/nettleie/nettleie-for-forbruk/. (Accessed on 01/29/2022).

[81] *Nettleie, priser og avtaler - Tensio Trøndelag Sør*. https://ts.tensio.no/kunde/nettleie-priser-og-avtaler. (Accessed on 01/29/2022).

[82] Ngoc-Tri Ngo. «Early predicting cooling loads for energy-efficient design in office buildings by machine learning». I: *Energy and Buildings* 182 (2019), s. 264–273.

[83] *Ny strømprisrekord i 1. kvartal*. https://www.ssb.no/energi-og-industri/energi/statistikk/elektrisitetspriser/artikler/ny-stromprisrekord-i-1.kvartal. (Accessed on 05/23/2022).

[84] Varun Kumar Ojha, Ajith Abraham og Václav Snášel. «Metaheuristic design of feedforward neural networks: A review of two decades of research». I: *Engineering Applications of Artificial Intelligence* 60 (2017), s. 97–116.

[85] Marie Sveen Olsen. «Reduksjon av effekttopper i kontorbygg». Masteroppg. NTNU, 2018.

[86] *Om Meteorologisk institutt*. https://www.met.no/om-oss/om-meteorologisk-institutt. (Accessed on 05/11/2022).

[87] *Oversikt over referansestil - IEEE*. https://www.scribbr.no/sitere-kilder/oversikt-over-referansestil/. (Accessed on 06/01/2022).

[88] *Peak Shaving — What it is & how it works*. https://www.next-kraftwerke.com/knowledge/what-is-peak-shaving. (Accessed on 01/29/2022).

[89] Ioannis Prapas mfl. «Continuous Training and Deployment of Deep Learning Models». I: *Datenbank-Spektrum* 21.3 (2021), s. 203–212.

[90] *Programmeringsgrensesnitt – Wikipedia*. https://no.wikipedia.org/wiki/Programmeringsgrensesnitt. (Accessed on 03/08/2022).

[91] Aowabin Rahman, Vivek Srikumar og Amanda D Smith. «Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks». I: *Applied energy* 212 (2018), s. 372–385.

[92] Chitta Ranjan. *Rules-of-thumb for building a Neural Network — by Chitta Ranjan — Towards Data Science*. https://towardsdatascience.com/17-rules-of-thumb-for-building-a-neural-network-93356f9930af. (Accessed on 02/08/2022). Jul. 2019.

[93] Chitta Ranjan. *Understanding deep learning: Application in rare event prediction*. Connaissance Publishing, 2020.

[94] Matthias Reif og Faisal Shafait. «Efficient feature size reduction via predictive forward selection». I: *Pattern Recognition* 47.4 (2014), s. 1664–1673.

[95] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[96] Frank Rosenblatt. «Perceptron simulation experiments». I: *Proceedings of the IRE* 48.3 (1960), s. 301–309.

[97] Seunghyoung Ryu, Jaekoo Noh og Hongseok Kim. «Deep neural network based demand side short term load forecasting». I: *Energies* 10.1 (2017), s. 3.

[98] *Sampling (signal processing) - Wikipedia*. `https://en.wikipedia.org/wiki/Sampling_(signal_processing)`. (Accessed on 03/08/2022).

[99] *SARIMAX: Introduction — statsmodels*. `https://www.statsmodels.org/stable/examples/notebooks/generated/statespace_sarimax_stata.html`. (Accessed on 02/26/2022).

[100] Minglei Shao mfl. «Prediction of energy consumption in hotel buildings via support vector machines». I: *Sustainable Cities and Society* 57 (2020), s. 102128.

[101] Caston Sigauke og D Chikobvu. «Prediction of daily peak electricity demand in South Africa using volatility forecasting models». I: *Energy Economics* 33.5 (2011), s. 882–888.

[102] Aishwarya Singh. *Multivariate Time Series — Vector Auto Regression (VAR)*. `https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/`. (Accessed on 01/30/2022). Sep. 2018.

[103] *Slik lager vi værvarslene på Yr – Yr hjelp og informasjon*. `https://hjelp.yr.no/hc/no/articles/360004008874`. (Accessed on 02/18/2022).

[104] «SN-NSPEK 3031:2020 - Bygningers energiytelse — Beregning av energibehov og energiforsyning». I: (2016).

[105] Dennis Soemers. *How should Feature Selection and Hyperparameter optimization be ordered in the machine learning pipeline?* Accessed on 02/04/2022. URL: `https://stats.stackexchange.com/q/323899`.

[106] Xuanyi Song mfl. «Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model». I: *Journal of Petroleum Science and Engineering* 186 (2020), s. 106682.

[107] Nitish Srivastava mfl. «Dropout: a simple way to prevent neural networks from overfitting». I: *The journal of machine learning research* 15.1 (2014), s. 1929–1958.

[108] Callahan Stade. *Calculus in Context - 4.3 Riemann Sums*. `https://math.colorado.edu/~stade/CLS/Section_4_3.pdf`. (Accessed on 05/20/2022).

[109] Eric Stellwagen, Len Tashman mfl. «ARIMA: The models of Box and Jenkins». I: *Foresight: The International Journal of Applied Forecasting* 30 (2013), s. 28–33.

[110] Abdulhamit Subasi. *Practical Machine Learning for Data Analysis Using Python*. Academic Press, 2020.

[111] Ola Surakhi mfl. «Time-Lag Selection for Time-Series Forecasting Using Neural Network and Heuristic Algorithm». I: *Electronics* 10.20 (2021), s. 2518.

[112] Pruethsan Sutthichaimethee og Danupon Ariyasajjakorn. «Forecasting energy consumption in short-term and long-term period by using arimax model in the construction and materials sector in thailand». I: *Journal of Ecological Engineering* 18.4 (2017).

[113] *Tabell (datastruktur) – Wikipedia*. `https://no.wikipedia.org/wiki/Tabell_(datastruktur)`. (Accessed on 03/08/2022).

[114] Agostino Tarsitano og Ilaria L Amerise. «Short-term load forecasting using a two-stage sarimax model». I: *Energy* 133 (2017), s. 108–114.

[115] Tran Manh Thang og Juntae Kim. «The anomaly detection by using dbscan clustering with multiple parameters». I: *2011 International Conference on Information Science and Applications*. IEEE. 2011, s. 1–5.

[116] *The ARIMAX model muddle — Rob J Hyndman*. `https://robjhyndman.com/hyndsight/arimax/`. (Accessed on 02/26/2022).

[117] *Towards Data Science*. `https://towardsdatascience.com/`. (Accessed on 03/07/2022).

[118] Thanh Ngoc Tran, Dang Thi Phuc mfl. «Grid search of multilayer perceptron based on the walk-forward validation methodology». I: *International Journal of Electrical and Computer Engineering* 11.2 (2021), s. 1742.

[119] Stylianos I Vagropoulos mfl. «Comparison of SARIMAX, SARIMA, modified SARIMA and ANN-based models for short-term PV generation forecasting». I: *2016 IEEE International Energy Conference (ENERGYCON)*. IEEE. 2016, s. 1–6.

[120] *Variations on rolling forecasts — Rob J Hyndman*. `https://robjhyndman.com/hyndsight/rolling-forecasts/`. (Accessed on 02/25/2022).

[121] Sandra Vieira, Walter Pinaya og Andrea Mechelli. «Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications». I: *Neuroscience & Biobehavioral Reviews* 74 (jan. 2017). DOI: `10.1016/j.neubiorev.2017.01.002`.

[122] *Visual Studio Code - Wikipedia*. `https://en.wikipedia.org/wiki/Visual_Studio_Code`. (Accessed on 03/08/2022).

[123] Shalika Walker mfl. «Accuracy of different machine learning algorithms and added-value of predicting aggregated-level energy performance of commercial buildings». I: *Energy and Buildings* 209 (2020), s. 109705.

[124] Jian Qi Wang, Yu Du og Jing Wang. «LSTM based long-term energy consumption prediction with periodicity». I: *Energy* 197 (2020), s. 117197.

[125] Peder Ward. ≪Vannlekkasjedeteksjon ved bruk av anomaly detection≫. I: *NTNU* (des. 2021).

[126] *What's the best way to handle NaN values? — by Vasile Păpăluță — Towards Data Science.* `https://towardsdatascience.com/whats-the-best-way-to-handle-nan-values-62d50f738fc`. (Accessed on 05/06/2022).

[127] Cort J Willmott og Kenji Matsuura. ≪Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance≫. I: *Climate research* 30.1 (2005), s. 79–82.

Peder Enes Ward

NTNU Masteroppgave - Kybernetikk og robotikk

NTNU
Norwegian University of
Science and Technology