Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i ingeniørfag, Datateknologi Data	Vårsemesteret 2022 Åpen
Forfatter(e): Daniel Havstad, Danny Vo, Sindre Reidar Mohr	
Fagansvarlig: Mina Farmanbar Veileder(e): Mina Farmanbar & Aida Mehdipour	
Tittel på bacheloroppgaven: Design og implementering av et interaktivt webgrensesnitt for prognoser om energiforbruk Engelsk tittel: Design and Implementation of an interactive web interface for energy consumption forecasting	
Studiepoeng: 20	
Emneord: AI, Deep Learning, web-interface, REACT, Dashboard, strømforbruk	Sidetall: 96 + vedlegg/annet: Stavanger 15. mai 2022

Innhold

Innhold **i**

1 Introduksjon **1**

1.1 Problemstilling 2

1.2 Arbeidsflyt 2

2 Bakgrunn **4**

2.1 Kunstig Intelligens 4

2.2 Python 12

2.3 Tensorflow og Scikit-learn 13

2.3.1 Tensorflow 13

2.3.2 Scikit-learn 13

2.4 Iterativ kodeutvikling 14

2.5 Web-applikasjon 15

2.5.1 Backend 15

2.5.2 Frontend 17

3 Exploratory Data Analysis (EDA) 18

3.1 Data utforskning 18

3.2 Husdata 19

3.3 Halvtimelig datasett 21

3.4 Værdata 26

3.5 Korrelasjoner 27

3.6 Data prosessering 28

3.7 Omgjøring av data 31

4 Design og konstruksjon av programvare 35

4.1 Design av modellene 35

4.1.1 Utgangspunkt 37

4.1.2 Test oppsett 37

4.1.3 Beslutningstre 39

4.1.4 Perceptron, SLP og MLP 40

4.1.5 LSTM 43

4.2 Dashboard 45

4.2.1 Databasestruktur 46

4.2.2 Funksjonalitet 48

4.2.3 Brukergrensesnitt 49

5 Resultater og diskusjon **62**

5.1 En enkel modell 62

5.2 Lineær regresjon 64

5.3 Beslutningstre modell 66

5.4 Perceptron (lineær modell) 68

5.5 SLP 69

5.6 LSTM 71

5.7 Sammenligning 72

5.8 Bruk av værdata 74

5.9 Evnen til å spå daglige topper 76

6 Sammendrag og Konklusjon **79**

6.1 Sammendrag 79

6.2 Konklusjon 80

6.3 Utfordringer 81

6.4 Videreutvikling 81

Bibliografi **84**

Vedlegg **84**

A Programlisting **85**

Figurer

2.1 Visualisering av sammenhengen mellom Kunstig Intelligens, Maskinlæring og Dyplæring 5

2.2 Beslutningstre for om en skal bruke strøm eller ikke 6

2.3 Visualisering av neurale nettverk 7

2.4 Recurrent Neural Network 10

2.5 Visualisering av tilbakevendende neurale nettverk 12

3.1 Acorn kategorier 19

3.2 Prosentvis av gjorte målinger gruppert på Acorn gruppe 20

3.3 Boksplot av strømdata fra 10 hus 22

3.4 Data fra MAC000150 over hele perioden 22

3.5 Data fra enkelte dager 23

3.6 Data fra enkelte uker 24

3.7 Data fra enkelte måneder 24

3.8 MAC000002 Household Visualization 25

3.9 MAC003305 Household Visualization 25

3.10 Første fem radene i timelig værdata 26

3.11 Temperaturmålinger for london i 2012 27

3.12 Korrelasjoner med energi målinger for husene 28

3.13 Manglende målinger i Halfhourly datasett 33

3.14 Manglende målinger i daily datasett 33

4.1 Model input tidsetterslep 38

4.2 MSE optimalisert til venstre, MAE optimalisert til høyre (tatt fra en lstm modell), spådde verdier i oransj 39

4.3 trening på MSE (perceptron) 41

4.4 trening på MSE 44

4.5 database skjema 46

4.6 Sidekartet viser navigasjon på siden fra topp til bunn. 48

4.7 Forside før bruker laster in prosjekt 50

4.8 Forside med lastet prosjekt 51

4.9 Oversikt over prosjekter 52

4.10 Liste over tilgjengelig data 53

4.11 Oversikt over prosjektdata 54

4.12 Dataplot ved bruk av Plotly 55

4.13 Verktøylinje plot 55

4.14 Utforskning av plot 56

4.15 Datautforskning med plot 57

4.16 Datautforskning med tabeller 58

4.17 Endring av modelparametere 59

4.18 Kjøring av prosjekter 60

4.19 Utklipp fra eksportert PDF 61

5.1 prognosen, plottet mot test data for den enkle modellen 62

5.2 Testdata for MAC000168 64

5.3 Linje for lineær regresjon 64

5.4 Plot av spådommer på testdata 66

5.5 Prognose for en uke fram i tid 66

5.6 Plot av spådommer på testdata 68

5.7 Plot av spådommer på testdata en uke i framtiden 68

5.8 Plot av spådommer på testdata 69

5.9 Plot av spådommer på testdata en uke i framtiden 70

5.10 Plot av spådommer på testdata 71

5.11 Plot av spådommer på testdata 71

5.12 Plot av topper, spådommer i oransj 76

5.13 Plot av forskjellen mellom toppene og prognosen 77

Tabeller

4.1 Perceptron trening 41

4.2 antall nerveceller 42

4.3 testing av MLP med forskjellige lag 42

4.4 antall nerveceller 44

4.5 testing av LSTM med flere lag 45

5.1 resultater fra baseline modell (halvtime) 63

5.2 resultater fra baseline modell (time) 63

5.3 resultater fra lineær regresjons modell (time) 65

5.4 resultater fra DT modell (halvtime) 67

5.5 resultater fra DT modell (time) 67

5.6 resultater fra lineær modell (halvtime) 69

5.7 resultater fra SLP (halvtime) 70

5.8 resultater fra LSTM (halvtime) 72

5.9 Forskjell fra regresjonsmodell 73

5.10 Forskjell fra SLP modell 74

5.11 resultater fra DT 75

5.12 resultater fra Perceptron 75

5.13 resultater fra SLP med 30 enheter 75

5.14 resultater fra LSTM et lag med 50 enheter 75

5.15 Resultater for daglige topper for hver modell 78

Kapittel 1

Introduksjon

For at en strømprodusent skal kunne produsere strøm i takt med behovene til trengs det prediksjoner av forbruk. Manglende samsvar mellom produksjon og forbruk vil kunne resultere i spenningsvariasjon, og også store forskjeller i pris på strøm. En overproduksjon er dessuten sløsing av ressurser, om det f.eks produseres i vannkraftanlegg. Tidligere har prediksjoner av strømforbruk måttet baseres på et målt forbruk på et større segment av strømnettet. De individuelle husholdningene i nettverket har ikke kunnet måles, og en bedre forståelse av strømforbruket har ikke vært mulig.

Med en kraftig fremvekst av smarte strømmålere finnes det det nå en overflod av målinger fra husholdnings nivå. Med disse målingene kan en lage prognoser for strømforbruk, som vil kunne hjelpe å balansere belastningen på strømnettet. En bedre forståelse av hvordan forbruksmønstrene til husholdninger ser ut kan også bidra til opplæring av forbrukere på hvilke tidspunkter de burde kjøre elektriske apparater. De siste årene har det også vært en stor fremvekst av el-bil andel på veiene, og mange forbrukere velger å lade bilen hjemme. Med så store variasjoner i de forskjellige husholdningene er det smarte strømmålere som vil kunne gi oss et bedre overblikk over helheten av strømnettet.

For strømforbruk nytter det sjeldent å se veldig langt frem i tid, da værforhold er en stor

påvirkningsfaktor. Derfor er prediksjoner kort tid i fremtiden det som vil kunne si mest om forbruket. Det er heller ikke veldig interessant å komme med prediksjoner på det jevne forbruket, da dette ikke er veldig kritisk. Det som er essensielt for å holde produksjon så nært forbruk som mulig er å se på toppen av forbruket. Det er i toppen vi best kan oppleve under/overproduksjon som skaper store variasjoner i spenningsnivået.

1.1 Problemstilling

Det finnes mange forskjellige modeller som blir brukt til å lage prediksjoner. Ved hjelp av deep learning modeller kan vi trene opp en modell til å lage predikasjoner frem i tid basert på tidligere målinger. I dette prosjektet vil vi lage et load forecasting system som vil lage prediksjoner for den neste timen. Vi vil lage ulike modeller for load forecasting av strømforbruket til et hushold og finne ut hvilken av algoritmene som vil gi best resultat. Dette vil så bli integrert og visualisert ved hjelp av et dashboard.

1.2 Arbeidsflyt

Arbeid med oppgaven bar preg av sosial-distanserings rutiner fra pandemien, til tross for oppheving av smittevernsrestriksjoner. Dette førte til at det ikke ble fysiske møter hverken innad i gruppen eller med veileder. Det ble holdt møter på Zoom, Microsoft Teams, Google Hangouts og Discord. Innad i gruppen var det spesielt Discord som ble benyttet, ettersom behovet var utelukkende for audio-kommunikasjon og ikke video-streaming. Muligheten for lavterskel audio-kommunikasjon på Discord åpnet for hyppigere sammarbeidsøkter for å fikse mindre problemer som stoppet arbeidsflyten.

For å få til et godt samarbeid er rask deling av endringer i koden essensielt. Derfor valgte vi å bruke Git sammen med GitHub for versjonkontroll av koden. Alle på gruppen har erfaringer med tjenesten fra før, noe som sparte tid og lot oss starte rett på prosjektet. Git issues ble benyttet for å holde styr på hvilke oppgaver som måtte bli gjort og var en måte

for oss å fordele oppgaver innad i gruppen. Denne funksjonaliteten har ikke vært benyttet i like stor grad i tidligere prosjekt, og er noe vi som gruppe var svært fornøyd med.

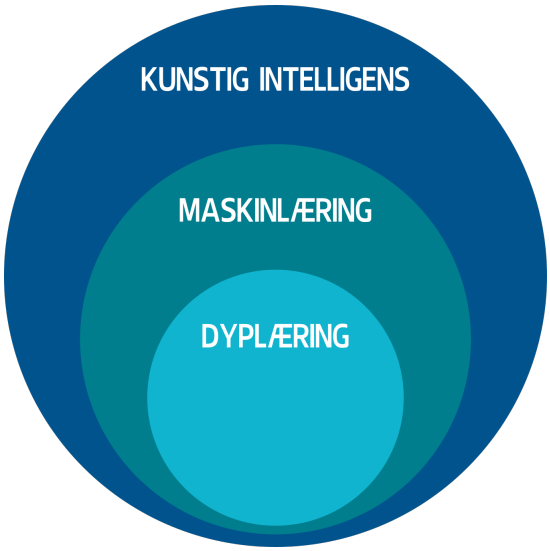
I tillegg til GitHub benyttet vi oss av Visual Studio Live Share for å skrive kode sammen. Bruken av Live Share gjorde det mye lettere å få en oversikt over hva andre har gjort og var et effektivt verktøy i feilsøkeprosessen. Etterhvert som prosjektet ble større, støttet vi på problemer med Live Share som antakeligvis kunne vært forhindret med segmentering av prosjektet. Jobbing med prosjektet foregikk som oftest fra mandager til torsdager der vi satt sammen i discord og jobbet. Fredager ble ofte brukt til å holde møter med veileder når det var nødvendig.

Kapittel 2

Bakgrunn

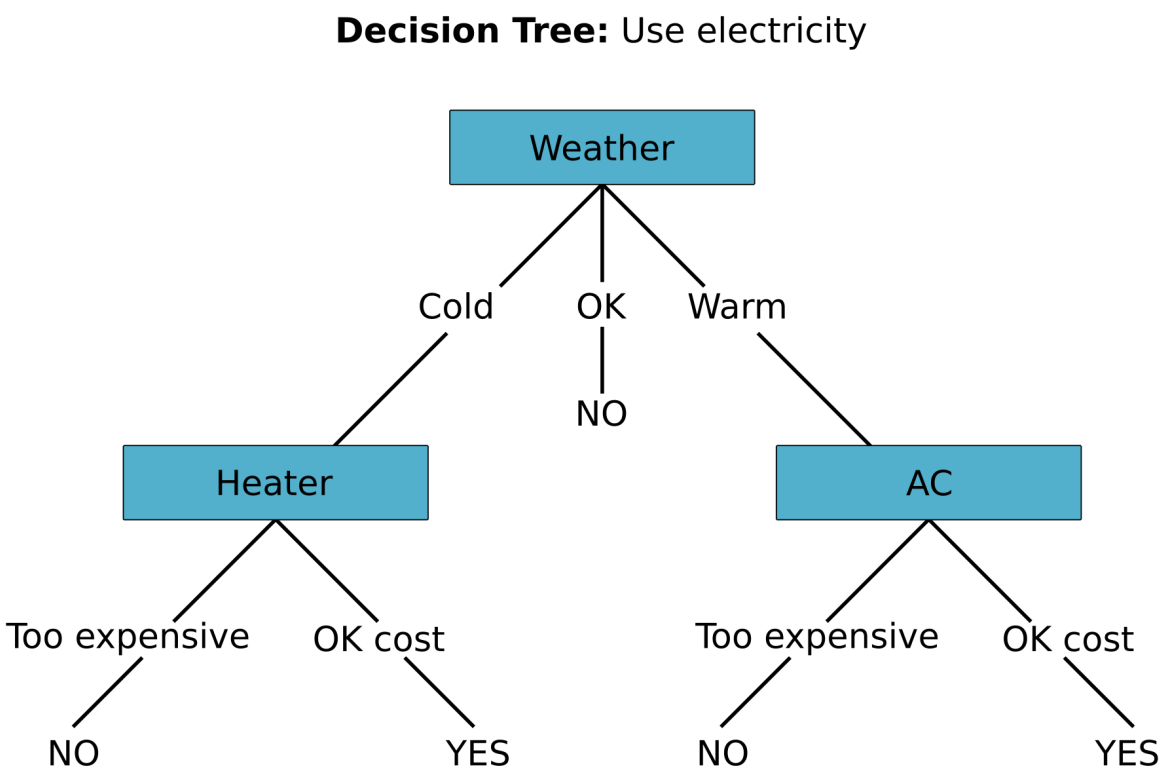
2.1 Kunstig Intelligens

Kunstig intelligens er programvare som justerer sin egen oppførsel, og på den måten framstår som intelligent. Med andre ord kan vi si at det er teknikker som tillater maskiner å utføre oppgaver likt det et menneske ville løst det. Maskinlæring bygger videre på kunstig intelligens, og er mer spesifikt algoritmer som tillater den å lære fra input slik at prediksjoner kan forbedres.



Figur 2.1: Visualisering av sammenhengen mellom Kunstig Intelligens, Maskinlæring og Dyplæring

Et beslutningstre eller Decision tree (DT) på engelsk er til å begynne med en type flytdiagram. Der hver node i treet er en test, og hver kant eller grein er et tenkelig resultat av testen. Er det grunn til å bruke strøm? Først kan en ta en avgjørelse basert på temperaturen, er denne tilstrekkelig lav bør en vurdere å bruke strøm til oppvarming. Er temperaturen høy kan en vurdere aircondition. Slik utvikler beslutningstreet greiner, se figur 2.2. bladnodene i treet gir den endelige beslutningen. For en beslutningstremodell kan et tre læres ved å splitte en opprinnelig mengde i delmengder. Dette basert på en test på en egenskap en er interessert i. En splitter deretter delmengdene igjen rekursivt. Prosessen avsluttes når delmengden ved en node har samme verdi som variablen en var ute etter, eller dersom en videre oppsplitting ikke gir bedre prediksjon. En beslutningstre modell skal være god til å ta imot støyfull data, og kan håndtere både kontinuerlige data, og kategoriske data. Men er ikke godt egnet estimerings oppgaver [2].

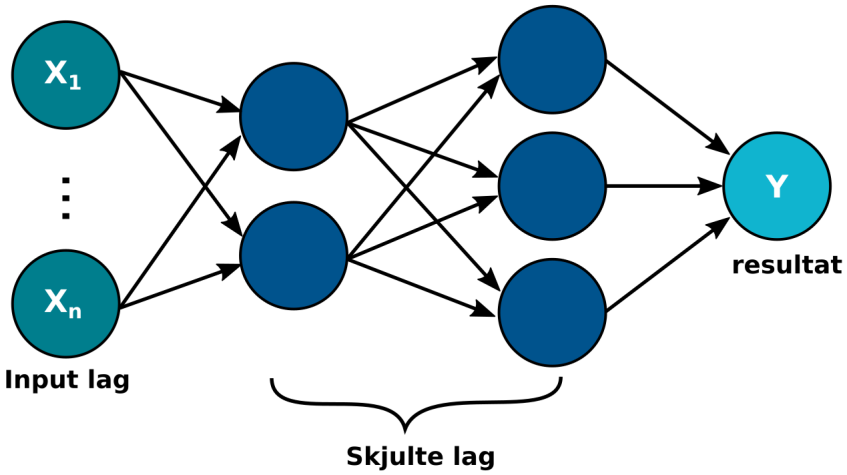


Figur 2.2: Beslutningstre for om en skal bruke strøm eller ikke

Kunstige neurale nettverk (ANN) er maskinlæring som er inspirert av det neurale nettverket i hjernen til mennesker. Denne typen læring kalles dyplæring. Neurale nettverk er modeller som kan brukes til å lage prediksjoner. Modellen trenes på datasett fra tidligere målinger, hvor datasettene deles inn i trenings- og testdatasett. Treningssettet brukes for å finne mønster, og testdatasettet brukes for å validere nøyaktigheten av modellen.

Det neurale nettverket består først av et Input lag, der nervecellenes oppgave er kun å ta inn informasjonen fra et datasett. Tenk nervecellene i hjernen som tar imot impulser fra sansecellene i f.eks. øret. Deretter i modellen har en skjulte lag som oppgave er å tenke seg fram til et resultat. Si at en får oppgaven å regne ut 3 ganger 5 uten å ha allerede

lært svaret. Kanskje gjetter en først vilkårlig på 8, men etter nok ganger der en får vite at svaret er 15, kommer nervecellene i hjernen alltid til å fyre av slik at en kommer fram til nettopp 15. Du kan tenke på nervecellen i modellen som en node som holder på et tall. Hver celle fra input laget blir sendt inn i hver celle i det første skjulte laget, se figur 2.3.



Figur 2.3: Visualisering av neurale nettverk

For hver tilkobling fra input laget til cellen er det koblet en vekt w, denne vekten sier noe om hvordan inngangen vil korrelere med utgangen til nervecellen. Altså om den skal fyre av eller ikke. En positiv vekt vil tilsi at den skal fyre av, mens en negativ vekt betyr det motsatte. Den vektede summen liggende i nervecellen blir da:

$\sum_{i=1}^{n} w_i x_i$

I tillegg kan en legge til en bias b, i den vektede summen som en ekstra vekt eller buffer for akkurat når denne nervecellen skal fyre av eller aktiveres.

$(\sum_{i=1}^{n} w_i x_i) - b$

For å gjøre den vektede summen mer tilpasset nettverket trengs en aktiveringsfunksjon.

Vanlig brukte aktiveringsfunksjoner er: relu, en sigmoid funksjon, eller hyperbolsk tangens. ReLU (Rectified Linear Unit), har lineær form og uttrykkes som:

$f(x) = max(0, x)$

En sigmoid funksjon kjennetegnes av en S formet kurve og kan være:

$\sigma(x) = \frac{1}{1+e^{-x}}$

Til slutt hyperbolsk tangens:

$tanh(x)$

Aktiveringen til en enkelt celle blir da med relu:

$f((\sum_{i=1}^{n} w_i x_i) - b)$

For hvert lag i nettverket kan vi kalkulere aktiveringen med å legge alle cellene i laget før i en vektor, dens vekter i en matrise, gange disse sammen for en resulterende matrise med de vektede summene. Deretter kan en justere disse med biasene og gå gjennom aktiveringsfunksjonen[6]:

$$f(\begin{bmatrix} w_{1,1} & w_{1,2} & ... & w_{1,n} \\ w_{2,1} & w_{2,2} & ... & w_{2,n} \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ w_{n,1} & w_{1,2} & ... & w_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_n \end{bmatrix})$$

Nå for hvordan nettverket lærer. En sammenligner resultatet av nettverket med den faktiske verdien, og spør hva tapet i avviket mellom disse verdiene er. For dette bruker nettverket en loss (tap) funksjon. Disse forteller oss hvor mye av verdiene i datasettet prognosen taper. Kjente loss funksjoner er:

Gjennomsnittlig kvadratisk avvik MSE

$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2$

Kvadratrota av MSE, RMSE:

$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2}$

Gjennomsnittlig absolutt avvik MAE

$MAE = \frac{1}{n}\sum_{i=1}^{n}(|y_i - y_i'|)$

Prosentvis MAE, MAPE:

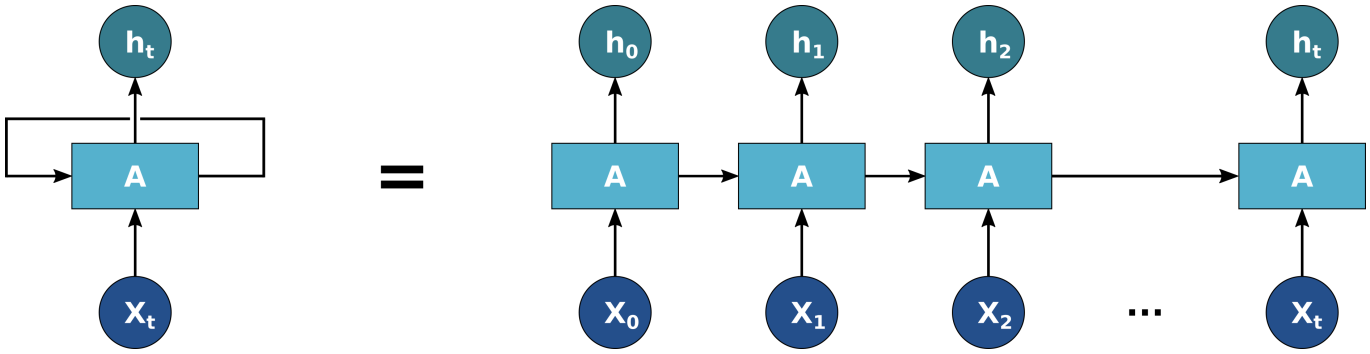
$MAPE = \frac{1}{n}\sum_{i=1}^{n}(\frac{|y_i - y_i'|}{y_i} * 100\%)$

Det er ønskelig å minimere dette tapet. Ved å finne den negative gradienten finner en retningen som reduserer funksjonen raskest. Derfor så lenge du følger den negative gradienten og kalkulerer den for hvert steg vil du etter hvert nærme deg et minimum. Den negative gradientsvektoren vil fortelle deg om input vektoren, vektene og biasene til nodene i nettverket, skal justeres opp eller ned basert på fortegnet, og hvor viktig enhver vekt er basert på størrelsen til tallet i gradient vektoren. Å lære nettverket er å finne ut hvordan en skal endre vektene og biasene til å minimere loss funksjonen. Disse blir da justert tilbakevirkende imot resultatet (Backpropagation).

Et neuralt nettverk med kun et skjult lag kalles: single layer perceptron, legger en til flere lag blir nettverket et multi layer perceptron (MLP). MLP er den vanligste form for kunstige neurale nettverk.

Modeller kan trenes supervisedeller "unsupervised". Det vil si om modellen får ferdige set med in- og utverdier, som er eksempel på ønsket resultat.

LSTM er en versjon av Recurrent Neural Network (RNN) og ble laget som en løsning til



Figur 2.4: Recurrent Neural Network

RNN's short term memory problem. Hvis en sekvens er for lang, vil RNN ha vanskeligheter med å overføre informasjon fra tidligere til senere tidssteg. Dette fører til at RNN vil kunne utelate viktig informasjon fra begynnelsen. LSTM derimot har en innebygd mekanisme som blir kalt for gates. Disse kan lære hvilke data i en sekvens som er viktig å beholde, og hvilke som kan bli kastet vekk. Den kan lære å beholde relevant data eller informasjon for å lage predikasjoner, og glemme data som ikke er viktig.

LSTM blir brukt iblant annet stemmegjenkjenning, syntetisk tale og språkgenerering. For å vite hvordan LSTM fungerer må vi ta en titt på RNN. RNN har et input lag, et skjult lag og et output lag. Input laget tar inn en input, behandler det og sender det til det midterste laget som består av skjulte lag. Hvert skjulte lag har sine egne vekter. Hvis vi har en sekvens som sendes inn til RNN, vil RNN behandle sekvensen en etter en. Under behandlingen vil den sende det forrige skjulte laget til det neste steget av sekvensen. Skjulte lag vil fungere som nettverkets minne og vil holde informasjon om det som nettverket har sett før.

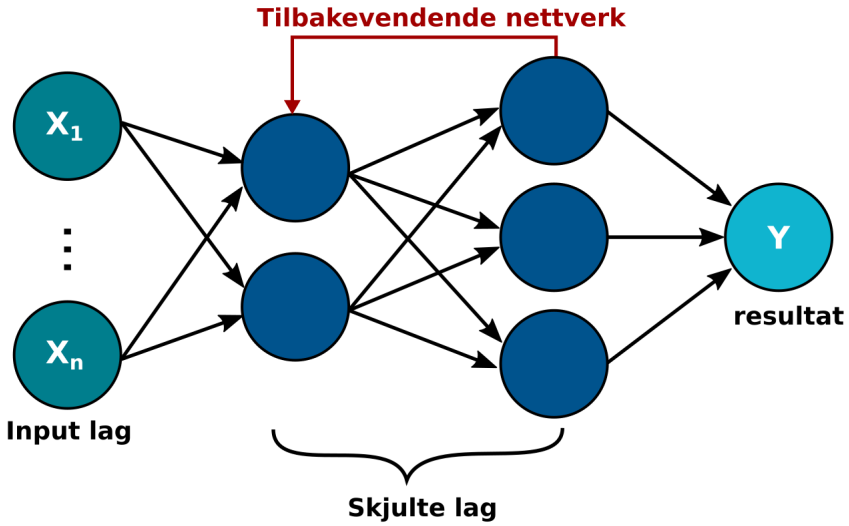
Boksen i figur 2.4 representerer et neural nettverk og pilene representerer minne eller outputten som vil bli sendt til neste input. Tenk for deg at vi har en serie med tall fra 1 til 5. Disse tallene vil bli sendt inn en etter en fra t=0, X_0 vil da bli 1. På t=1 vi $X_1 = 2$ bli gitt som input sammen med inputten fra forrige tidsskritt som har blitt lagret. Outputten vi får nå er 12, dette vil igjen bli lagret og sendt som input sammen med $X_2 = 3$ i t=2. Vi ender da opp med 123, sånn vil det gå helt til vi kommer til 12345. I en RNN vil forrige

input bli sendt til neste input like mange ganger tidsskritt per input vi har.

LSTM virker nesten helt lik som en RNN celle, men istedet for neurons bruker LSTM nettverk minne blokker som er koblet sammen ved hjelp av lag. Hver LSTM celle inneholder 3 forskjellige komponenter som blir kalt for gates.

- **Forget Gate:** Bestemmer hvilke data som blir glemt basert på betingelser.
- **Input Gate:** Bestemmer hvilke verdier fra inputtet som blir lagret i minne basert på betingelser.
- **Output Gate:** Bestemmer hva som blir outputtet basert på det som ble inputtet og hva som ligger i minne.

Akkurat som i en enkel RNN, har LSTM også en skjult tilstand hvor h_{t-1} representerer den skjulte tilstanden i det forrige tidsskrittet og der h_t er den skjulte tilstanden i nåværende tidsskritt. Det som skiller seg ut hos LSTM er celle tilstanden for forrige tidsskritt C_{t-1} og nåværende tidsskritt C_t.Den skjulte tilstanden er short term memory, mens celle tilstanden er long term memory.



Figur 2.5: Visualisering av tilbakevendende neurale nettverk

2.2 Python

Python er et høynivå programmeringsspråk som krever lite boilerplatekode for at applikasjoner skal kunne kjøre. Tar vi et klassisk eksempel som en HelloWorld applikasjon vil det kun kreve én linje med kode, noe som vil kreve minimum 5 linjer i Java for å oppnå samme resultat. Dette gjør at det tar kortere tid å nå et MVP, eller minste brukbare produkt. Python er et tolket språk, som vi si at koden ikke må kompileres til et ferdig program for å kjøre, men at det leses og utføres av et annet program. Selv om Python vil kunne ta kortere tid å skrive en applikasjon i vil kjøretiden ha problemer med å holde følge med kompilerte språk som Java. Ettersom koden i prosjektet er noe begrenset vil ikke kjøretiden være et for stort problem. Det faktum at koden ikke må kompileres hver gang det skal kjøres gir en fordel i gruppearbeid, hvor det skjer små endringer kontinuerlig, gjerne i en debug-prosess. Da kan koden raskt testes for hver lille endring i koden.

2.3 Tensorflow og Scikit-learn

2.3.1 Tensorflow

Tensorflow er en open source bibliotek brukt til å lage og trene maskinlærings modeller. Tensorflow er designet for å gjøre det lettere å lage neurale nettverk for maskinlæring ved hjelp av keras API. Keras er en dyplærings API som blir brukt på toppen av maskinlærings platformer, tensorflow har sin egen implementasjon av denne, kalt tf.keras, som vil støtte noen mer avanserte tensorflow funksjoner.

De viktigste data strukturene i keras er layers og models. Layers er byggesteinene i et neural nettverk. Vi kan tenke på layers som et filter for data. Ved å sende data gjennom disse filtrene vil vi få dem ut på en mer brukbar form. Keras sequential modell er den enkleste modellen som er en stabel av layers. Hvis vi trenger en mer kompleks model kan Keras Functional API bli brukt istedet. Når vi lager en modell og skal gjøre den klar for trening må vi velge ut 3 parametere:

- **En optimizer:** Mekanisme som modellen vil bruke til å oppdatere seg selv basert på trenings dataen som den blir tilført.
- **En loss funksjon:** Hvordan modellen vil måle sin fremgang på treningsdataen, og styre seg selv i riktig retning.
- **Metrics for å observere under trening og testing:** Her blir bare nøyaktigheten fokusert.

2.3.2 Scikit-learn

Scikit-learn er et veldig enkelt python bibliotek å bruke. Den har flere effektive verktøy som implementerer maskinlærings algoritmer, noe som gjør den et veldig godt startpunkt

for å lære maskinlæring. Scikit-learn er skrevet i python og er bygget på NumPy, SciPy og Matplotlib.

Scikit-learn ble utviklet av David Cournapeau som et Google summer of code prosjekt i 2007 og ble originalt kalt for scikits.learn. Senere i 2010 tok Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort og Vincent Michel fra INRIA opp prosjektet og publiserte denne 1. februar[11].

Scikit-learn sitt bibliotek fokuserer på modellering av data istedet for manipulering, og lasting av data. Noen av de mest populære modellene fra Scikit-learn er:

- **Clustering** - En modell brukt for å gruppere objekter som er like inn i mengder.
- **Dimensionality reduction** - Reduserer nummeret av tilfeldige variabler å ta i betraktning.
- **Regression** - Forutsi en kontinuerlig verdi assosiert med et objekt.
- **Classification** - Identifisere hvilken kategori et objekt tilhører.

Vi bruker det først og fremst for deres implementasjon av beslutningstre.

2.4 Iterativ kodeutvikling

I prosjektet har en tatt i bruk python notebooks ofte i stedet for et tradisjonelt python-script. Ipynb består av to elementer, en blokk som kan brukes til ren tekst eller notering, og en blokk som er kjørbar kode. Tradisjonelt er dette et nyttig læringsverktøy, ettersom eleven kan lese teorien i tekst blokken, og deretter få kjørt et kode eksempel rett etter på. I tillegg kan personen eksperimentere med eksempelet for å få bedre forståelse av hva som foregår. Ulempen med et tradisjonelt python skript der kjøretiden tross alt er raskere, er at du må kjøre gjennom hele skriptet med en gang, eller bruke tid på å definere klasser

og funksjoner, for å ta vare på og lett presentere arbeidet sitt. Alle variablene vil da stort sett være lokale, og ha levetid kun så lenge den koden kjører, som innebærer at den tunge operasjonen med å hente dataen ut fra fil må gjøres flere ganger. I en python-notebook vil dataen kun måtte hentes en gang. Det i koden som skal printes eller tegnes på skjermen kommer rett under kodeblokken de utføres fra, slik kan en holde styr på mange figurer og utskrifter, uten å miste tid eller oversikt. Utskriftene er også lagret til ipynb fila etter utførelse slik at en lett kan lese av resultater i etterkant uten å gi CPUen mer å jobbe med.

2.5 Web-applikasjon

For å enklere kunne benytte seg av datasett og modeller utvikles det en web-applikasjon hvor brukere kan få tilgang til kode og data i form av et enkelt brukergrensesnitt. En web-applikasjon som skal tilby mer enn en statisk nettside trenger kode som tilsier hvordan den skal respondere på interaksjon fra brukeren. Dette kan være endringer på nettsidens utseende og innhold, eller henting/oppdatering av data på serversiden. De visuelle, og gjerne umiddelbare endringene på nettsiden er tatt hånd om av et frontend rammeverk. Frontend vil være det brukeren har direkte tilgang på. Alt som vises i nettleser, er det frontend rammeverket som tilbyr. For å kunne kjøre en web-applikasjon for maskinlæring må det være tilgang til data og funksjoner / klasser. Disse befinner seg i backend systemet. Dette systemet vil ikke være direkte synlig for brukeren, men det er her all data blir prosessert og sendt videre til frontend rammeverket. Under backend finner vi også databasesystemet, som er den mest hensiktsmessige måten å oppbevare informasjonen som er brukt i applikasjonen.

2.5.1 Backend

I prosjektet vårt har vi valgt å benytte oss av programmeringsspråket python, ettersom det er det et språk som krever lite ekstra kodestruktur for å komme i gang. Det er også mest vanlige språket innenfor kunstig intelligens. For å lett kunne benyttes oss av koden skrevet

i python er det mest hensiktsmessig å også benytte seg av et backend rammeverk som er basert på python. Her ble det valgt å bruke Flask, som er et lettvekts mikrorammeverk. Flask har ikke objektsrelasjonshåndtering, men er strukturert for å håndtere URL-ruting på en effektiv måte. Dette passer bra med kravene våre til backendsystemet, da det i hovedsak vil foregå som API-kall fra frontend til backend. API (Application Programming Interface) er måte å koble to forskjellige systemer / programmer sammen. Ettersom frontend og backend systemene i prosjektet ikke er skrevet i samme språk vil det være hensiktsmessig å bruke en generell metode for API-kall slik at språket er irrelevant.

For å sende informasjon mellom to forskjellige språk vil det benyttes et format som er akseptert uavhengig av språk. Her er det enkleste, og mest utbredte formatet JSON. JSON står for JavaScript Object Notation og er et oppsett hvor data representeres som et objekt, det vil si i form av en nøkkel og en verdi. Verdien kan være tall, tekst, liste, eller et nytt objekt. I prosjektet vårt har vi valgt en struktur med mange små API-kall som benyttes når brukeren etterspør innhold. Dette gjør at det ikke er merkbart tregt å begynne en økt i applikasjonen, men at brukeren vil oppleve små forsinkelser innimellom. Dette er fordelaktig for bruker, da unødvendige mengder data ikke tar opp plass i nettleseren.

Til lagring og uthenting av data benyttes en SQL-database, hvor data settes inn i tabeller. SQL står for Structured Query Language, og er et spesialisert språk for håndtering av relasjonsdatabaser. En relasjonsdatabase er en database som benytter seg av prinsippene i relasjonsmodellen. For dette prosjektet brukes det en SQLite database for datahåndtering. SQLite er en lettvekts database som er lett å sette opp og som er enkel å flytte. Det SQLite ikke tilbyr er en pålitelig håndtering av flerbruker skriving. Flerbrukerskriving vil si at flere brukere oppdaterer og leser fra databasen samtidig, noe som krever strengere kontroll av hvordan innhold skrives. Dette er noe som ikke er relevant med mindre applikasjonen publiseres på en webserver hvor en større mengde brukere vil benytte seg av den samtidig.

2.5.2 Frontend

Ettersom det ikke er behov for segmentering av applikasjon i flere sider, er det mulig å gjennomføre den som en SPA. En SPA, eller Singel Page Application, benytter seg av komponenter istedenfor flere sider. For SPA er React et godt alternativ. React er et verktøy brukt for å lage brukergrensesnitt og er ikke et rammeverk men er et JavaScript-bibliotek. React er designet for å dele applikasjoner inn i komponenter. Et React komponent kan være alt i en web applikasjon, som for eksempel en knapp eller tekst. Fordelen med React er at JavaScript da kan benyttes på klientsiden. Klientside JavaScript betyr at vi kjører Javascript kode i nettleseren.

Dashboardet vil kreve en kontinuerlig oppdatering av komponenten slik at en kan benytte dataendringer i sanntid. Ved hjelp av Reacts Virtual DOM kan det implementeres server side rendering uten å måtte oppdatere hele siden etter hver oppdatering. Virtual DOM vil kunne oppdatere enkelte komponenter uten å berøre andre. Dette vil føre til raskere re-rendering av komponenter og det vil ikke kreve vider oppfølging etter vi sender dataen gjennom en API.

Kapittel 3

Exploratory Data Analysis (EDA)

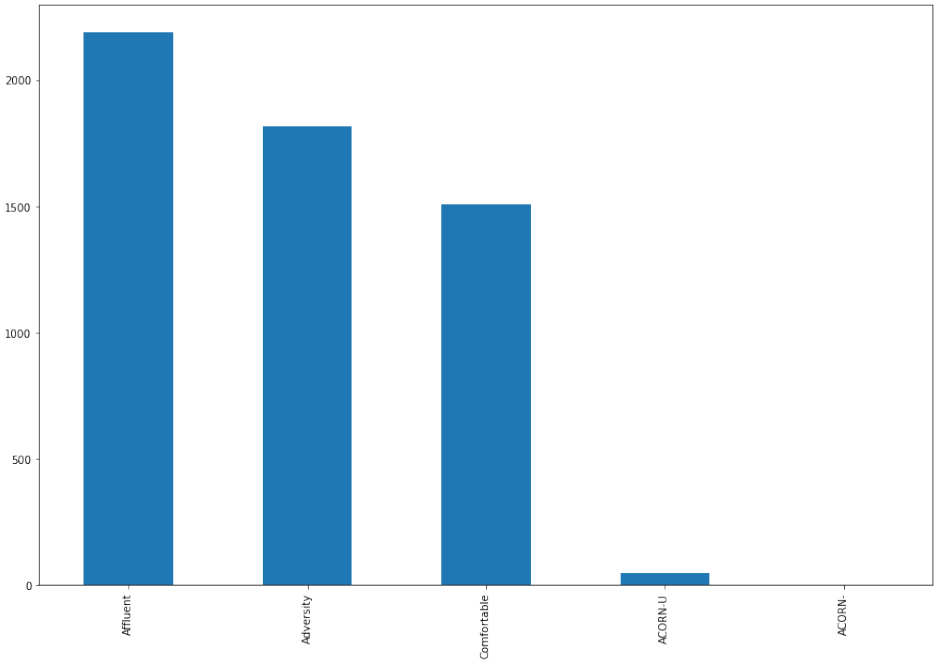
3.1 Data utforskning

For å kunne gjøre prediksjoner av et scenario er en avhengig av at en har tilgang på relevant data. Data som blir brukt må derfor være representativt, og en sentral del av maskinlæring er derfor å ha tilgang til datasett som samsvarer med oppgaven. En kan også kombinere flere datasett med målinger som samsvarer på lokasjon og tid, for å kunne benytte seg av flere parametre i utregninger. Dette kan være bruk av et hoveddatasett som en supplerer med f.eks værdata, som kan åpne muligheter for mer nøyaktige prediksjoner.

Dataen vi har funnet kommer fra målinger fra smartmetere i London installert i 5566 husstander mellom november 2011 og februar 2014 [8]. Målinger av strømforbruk er tatt hver halvtime i Kilowattime og det er totalt 167 817 021 målinger. Dataen for hver dag er oppsummert i et daglig datasett, med maksimumsverdi, minimumsverdi, gjennomsnitt, median, standardavvik og antall målinger. Det forekommer informasjon om de forskjellige husene i et eget datasett. Værdata i perioden fra darksky api er også tatt med. Målingene er gjort av UK power networks [5].

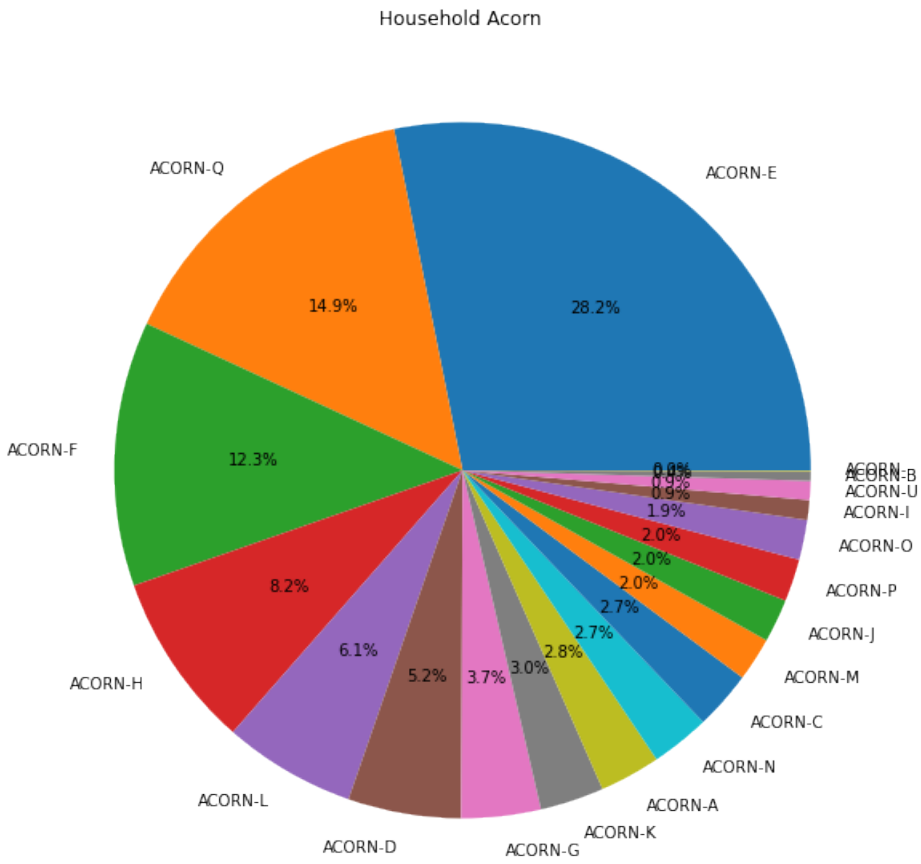
3.2 Husdata

Husene er delt inn i Acorn. ACORN er et segmenteringsverktøy som kategoriserer populasjonen i Storbritannia inn i demografiske kategorier. Husholdninger er segmentert inn i 5 kategorier, 17 grupper og 59 typer. I vårt datasett får en vite om huset tilhører kategoriene: Affluent, Comfortable, Adversity, ACORN- , ACORN-U [1].



Figur 3.1: Acorn kategorier

Innholdet i disse kategoriene er noe innlysende. I affluent holder de rike til, i comfortable har en mennesker som fortsatt klarer seg med god margin, og i adversity er det mennesker som sliter litt mer. Navnene her stemmer dessverre ikke overens med kategoriene fra acorn direkte. Kategori 1: Affluent achievers og kategori 2: Rising Prosperity, er begge del av Affluent kategorien i datasettet fordi gruppa Acorn D: City sophisticates del av Rising Prosperity er linket med Affluent, likedan med gruppe E [1].



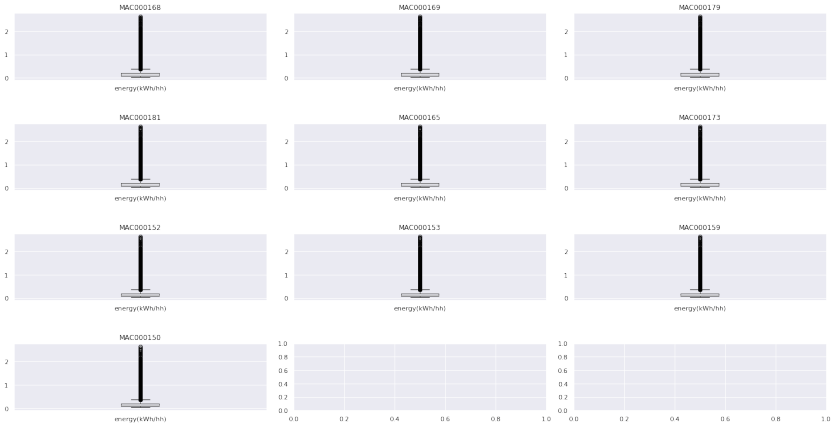
Figur 3.2: Prosentvis av gjorte målinger gruppert på Acorn gruppe

Ettersom kategorien er definert noe vagt i datasettet, og fordi det er grunn til å tro at gruppene innad i en kategori varierer stort i strømbruk, ser en på gruppene direkte. Datasettet gir oss dessverre ikke ytterlige informasjon om hvilke typer husstand det er. Acorn-A: Lavish lifestyles, har en blant annet valgt å se bort fra siden den ikke representerer folk flest, og er nok i tillegg svært varierende fra hushold til hushold. Vi har dermed tatt for oss gruppen Acorn E: "Career climbers", i kategorien Rising Prosperity". Det er av denne gruppen det er flest målinger av 28.2 %. Denne gruppen inneholder karriæredrevne unge familier, førstegangskjøpere i mindre - men moderne - boliger, samt leiligheter i storby

områder. I gruppen finner vi single, ektepar og småbarnsfamilier. Boligene er for det meste enten leid eller kjøpt med lån, og felles for majoriteten er at de bor i urbane områder, nært sentrum av London. Prisene på bolig, og lånekostnadene for de som leier, er over det gjennomsnittlige for landet. På generell basis er utdanningen i den gruppen høy, som også reflekteres i at gruppen har høyere lønn enn medianen i landet. Denne gruppen beskrives som mer selvsikker når det gjelder teknologi, som muligens forklarer at det er mange av de med smartmeter slik at de er en betydelig del av datasettet. Forhåpentligvis kan dette i praksis bety mer komplett data. Det inngår tre typer husstander i gruppen, og det vil bety ytterligere variasjon hvis en ser på hus fra forskjellige typer. I denne gruppen virker forskjellen mellom typene ganske liten. Og ettersom dette er en demografisk kartlegging, er antagelig forskjellen på vanene til menneskene i gruppen så minimale de kan være.

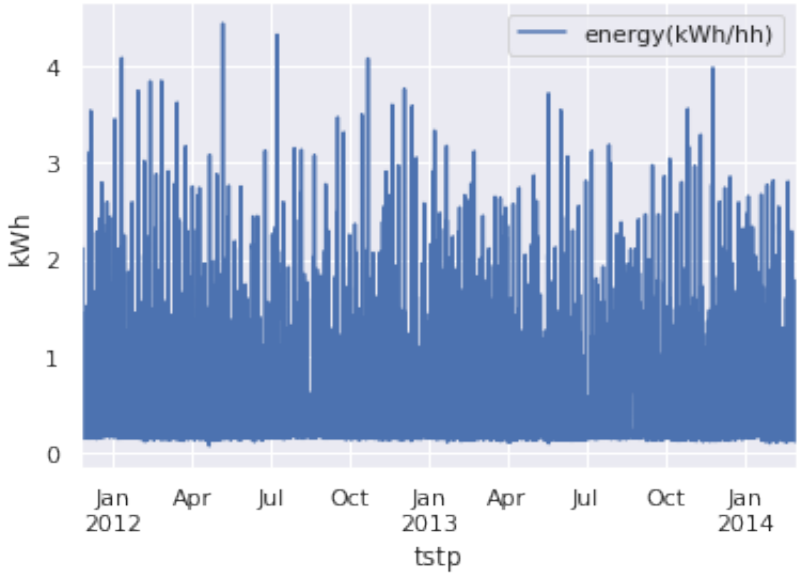
3.3 Halvtimelig datasett

Av husholdene i Acorn-E plukket en ut 10 av dem med mest målinger, dermed er dataen som blir brukt til maskinlæringen over så lang tid i perioden som mulig. Av disse er huset med identifikasjonen MAC000150 den med flest målinger 39720 totalt. Strømmålingene er i kwh, og er tatt hver halvtime. De fleste målingene som vist i boksplottet under i figur 3.3, ligger like over null. Dette er for det meste hjemmets hvileenergi, altså den strøm som blir brukt uansett til kjøleskap, internett, osv. Data over den tredje kvartilen representerer i større grad når folk er hjemme og husstanden er i aktivt bruk.

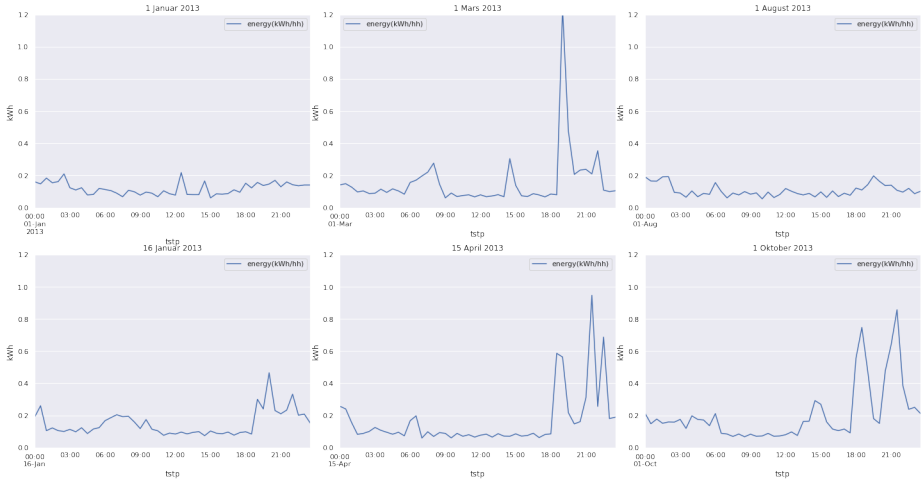


Figur 3.3: Boksplot av strømdata fra 10 hus

De videre diagrammene er alle basert på målinger fra MAC000150.

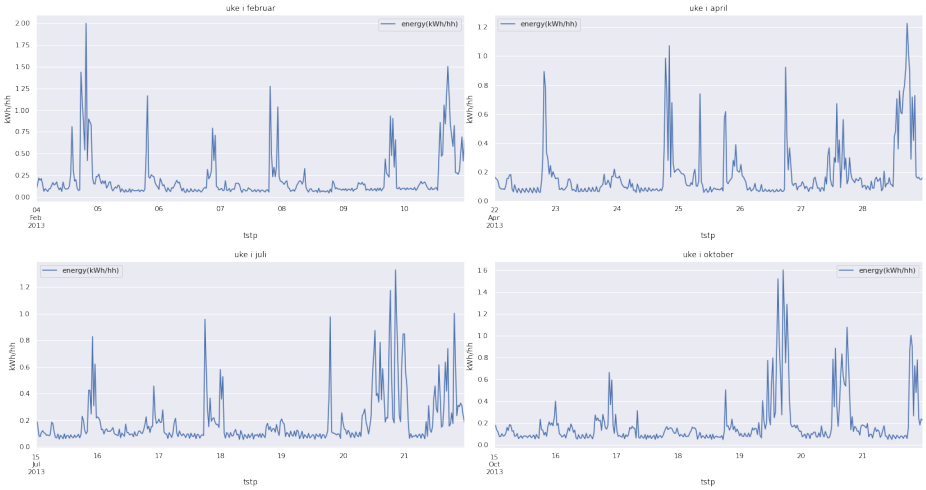


Figur 3.4: Data fra MAC000150 over hele perioden



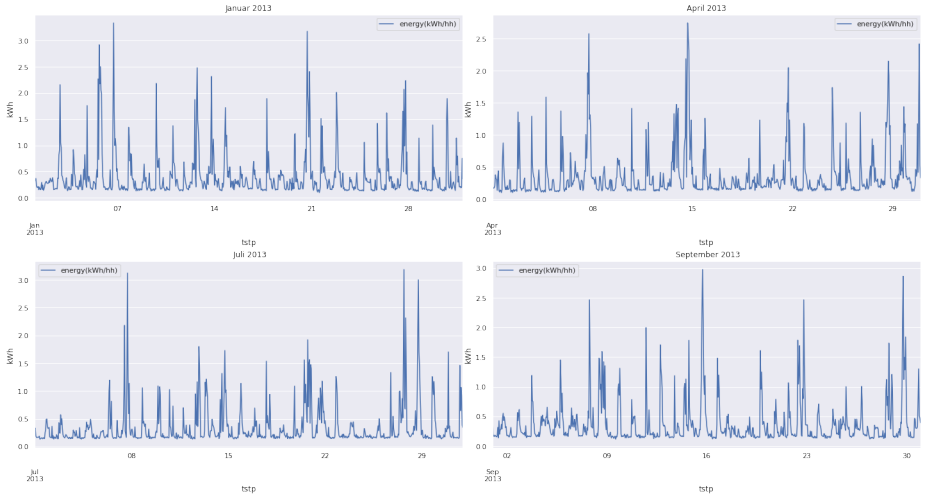
Figur 3.5: Data fra enkelte dager

Figur 3.5 viser strømforbruket for noen utvalgte dager til hushold MAC000150. Vi kan se en trend som går igjen i alle grafene, strømforbruket begynner å øke fra kl 18:00. Dette skyldes mest naturlig at familien kommer hjem fra jobb og at det er på denne tiden av døgnet folk bruker mest strøm. Mandag den første januar er første nyttårsdag, og er et eksempel på en helligdag. Det ser derimot ut som at vedkommende ikke feiret nyttår hjemme, og heller ikke særlig tegn til aktivitet for resten av dagen. Noe lignende kan sies om første august, kanskje de tok seg ferie. Men det er ikke konklusivt ettersom en ikke skulle trenge strøm på oppvarming i august, og en ser antydning til bruk av strøm klokka sju.



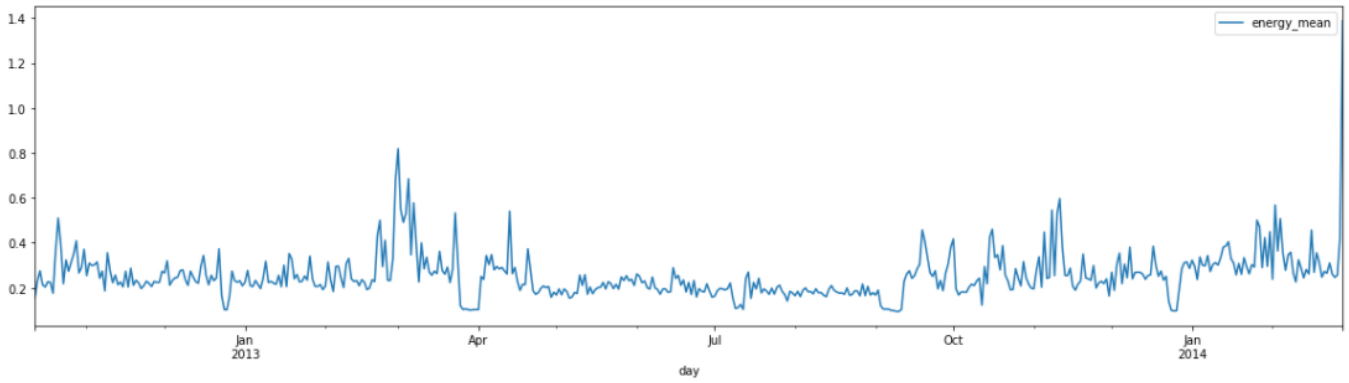
Figur 3.6: Data fra enkelte uker

For oversikten over hele uken kan en fortsette å se denne trenden, gjerne med økte topper på strømforbruket i slutten av uka ved helgen.

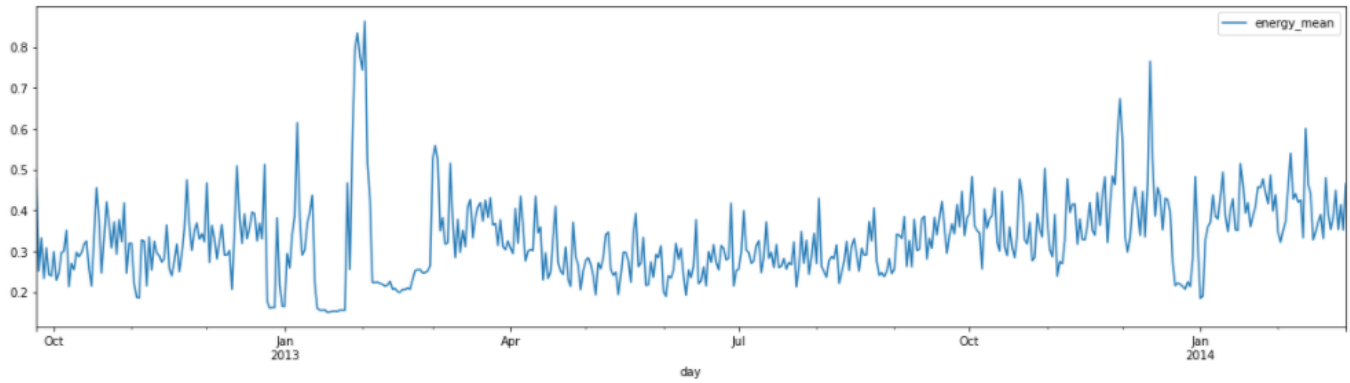


Figur 3.7: Data fra enkelte måneder

Utvidet til hver måned.



Figur 3.8: MAC000002 Household Visualization



Figur 3.9: MAC003305 Household Visualization

Ved å plotte dataen vi har kan vi få en oversikt over dataen som er tilgjengelig. I figur 3.8 kan det observeres at det ble brukte mest strøm i mars. Det kan også observeres at strømbruken går opp under vintertid, noe som er forventet ettersom det vil kreve mer strøm for å holde varme i et hus når temperaturen ute er lavere. Hus med ID MAC003305, se fig 3.9, har en mer forventet graf. Strømforbruket er høyest i vinter og er lavere i de andre årstidene. Hvis en ser bort fra strømforbruket i mars for MAC000002 kan det observeres at ved kaldere temperatur vil strømforbruket gå opp. Det lave strømforbruket i juni-august kan tenkes å være en konsekvens av at familien er vekkreist på ferie eller at det ikke er

nødvendig å bruke mye strøm grunnet varmere vær.

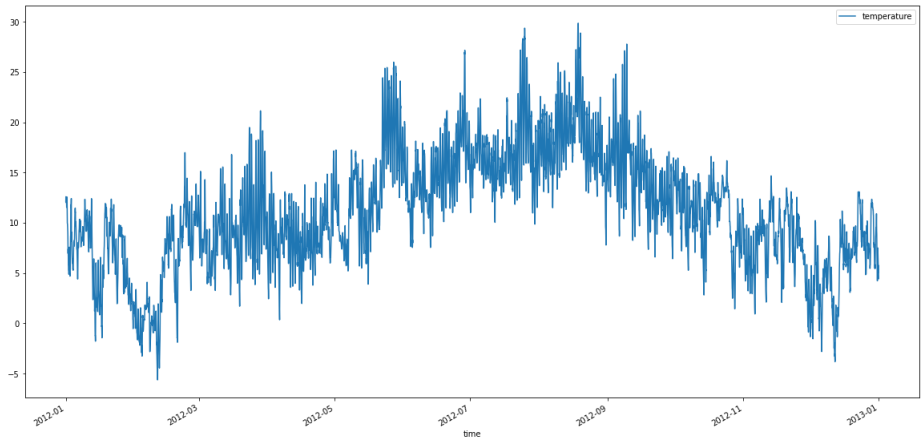
3.4 Værdata

Værdataen fra darksky api går fra 11-11-2011 til 15-02-2014. Enkelte av strømmålingene går lenger enn 15 februar 2014, dermed er det ikke direkte sammenlignbart, og vil ikke kunne bidra til modellering for data etter dette. For værdataen er det også to datasett. Ett på times basis, og et som er på daglig basis. Der det daglige gir en oppsummering av informasjonen i det med målinger hver time. I det daglige er det blant annet registrert når på dagen maks og minimumstemperaturen er registrert.

	visibility	windBearing	temperature	time	dewPoint	pressure	apparentTemperature	windSpeed	precipType	icon	humidity	summary
0	5.97	104	10.24	2011-11-11 00:00:00	8.86	1016.76	10.24	2.77	rain	partly-cloudy-night	0.91	Partly Cloudy
1	4.88	99	9.76	2011-11-11 01:00:00	8.83	1016.63	8.24	2.95	rain	partly-cloudy-night	0.94	Partly Cloudy
2	3.70	98	9.46	2011-11-11 02:00:00	8.79	1016.36	7.76	3.17	rain	partly-cloudy-night	0.96	Partly Cloudy
3	3.12	99	9.23	2011-11-11 03:00:00	8.63	1016.28	7.44	3.25	rain	fog	0.96	Foggy
4	1.85	111	9.26	2011-11-11 04:00:00	9.21	1015.98	7.24	3.70	rain	fog	1.00	Foggy

Figur 3.10: Første fem radene i timelig værdata

Darksky api har målt blant annet vindstyrke og temperaturer, alle kolonnene kan en se i figur 3.10. Vi har ikke tatt i bruk den kategoriske dataen ifra kolonnene: summary, icon og precipType.



Figur 3.11: Temperaturmålinger for london i 2012

Temperaturen i London er sjelden langt under null, på grunn av at byen ligger på kysten kan en anta en noe mer stabil temperatur i forhold til mer indre strøk. Ettersom vann tar både lengre tid å varme opp og kjøles ned enn jord.

3.5 Korrelasjoner

For å se hvilke data en eventuelt knytter opp mot en læringsmodell fant en korrelasjoner med værdata og dag/måned. Aller korrelasjonene er pearson R.

	LCLid	tempCorr	apparentTempCorr	windBearingCorr	windSpeedCorr	day	month	weekend
0	MAC000150	-0.033222	-0.036636	-0.017303	0.020423	0.166705	0.221778	0.221778
1	MAC000152	-0.058749	-0.067210	0.002200	0.076439	-0.019000	-0.009517	-0.009517
2	MAC000153	-0.423154	-0.421131	-0.048212	-0.069210	0.022852	0.025668	0.025668
3	MAC000165	0.092253	0.076671	-0.020507	0.068883	0.037624	0.055287	0.055287
4	MAC000169	-0.175923	-0.172782	-0.000892	-0.043566	0.098453	0.108026	0.108026
5	MAC000168	-0.194417	-0.198495	-0.027372	-0.018244	0.012557	-0.004077	-0.004077
6	MAC000159	-0.012752	-0.025763	-0.028490	0.066948	0.006883	-0.003540	-0.003540
7	MAC000173	-0.116428	-0.127590	-0.001374	0.070423	0.029237	0.042574	0.042574
8	MAC000179	-0.007468	-0.008194	0.012307	0.007076	0.065984	0.110316	0.110316
9	MAC000181	0.011728	-0.002104	-0.033900	0.099597	0.017970	0.030138	0.030138

Figur 3.12: Korrelasjoner med energi målinger for husene

For å få til korrelasjoner med dag og måned ble dagene fra mandag-søndag gitt verdier fra 1-7, og januar-desember ble gitt verdiene 1-12. En hverdag mandag-fredag ble representert med 0, mens en helg lørdag-søndag er representert med 1. Korrelasjonen for weekend tar også høyde for om det er en helligdag, ved å ta i bruk UK Bank Holiday datasettet. En helligdag er også representert med 1. For å kunne sammenligne værdataen er energimålingene gjort om fra hver halvtime til hver time. Av denne grunn er dataen brukt videre i modelleringen også i timeformat.

En kan se at dataen som omhandler vind er uaktuelt ettersom korrelasjonen for alle hus er < 0.1. Samtidig kan en legge merke til at korrelasjonene varierer mellom husstand, det tidligere presenterte huset MAC000150, er blant de eneste som ser ut til å påvirkes av ukedag, måned, og om det er helg eller ikke. Dette huset har svært dårlig korrelasjon med temperaturen, mens huset MAC000153 har en korrelasjon på -0.42.

3.6 Data prosessering

Data preprossesering er en viktig prosess når vi driver med datasett. Datasett inneholder ofte støy og mangler som oftest data. Derfor kan vi ikke sende datasettet rett til modellen

vår, og må rydde opp i det ved å fjerne eller gjøre om disse radene/kolonnene. Bruker vi dårlig data til å trene modellen vår, vil vi ende opp med en dårlig trent modell som ikke er relevant for analysen vi gjør. Data preprossesering er en teknikk vi bruker for å rydde opp og transformere data i et datasett om til et format som kan bli sendt til modellen. Lettere sagt er data preprossesering en metode for å pusse opp og rengjøre data vi har.

Datasett er oftest tilgjengelig i rå form. Det representerer målinger fra virkeligheten, som kan være utsatt for feil i måleinstrumenter og menneskelige feil i oppsett av registrering. Det forekommer derfor ofte usannsynlige målte verdier, manglende verdier, eller målinger i et format som ikke stemmer overens med det en modell kan benytte seg av.

Interpolasjon er det å beregne verdier imellom allerede kjente verdier. Dersom datasettet har manglende verdier, kan en interpolere disse. Interpolasjon forutsetter at de manglende datapunktene ligger jevnt opp mot hverandre. Lineær interpolasjon er vanligst, men en har også Cubic Spline, Lagrange og Newton's polynom interpolasjon. Lineær interpolasjon for funksjoner ser ut som følger [3]:

$f(x) \approx f(x_1) + \frac{x-x_1}{x_2-x_1}(f(x_2) - f(x_1))$

For et datasett er funksjonsverdien de gitte målingene en interpolerer mellom, mens x-ene vil være tidspunktene eller indeksen de ligger på. Hvis f(x1) er startverdien kan x1 være 0, f(x2) være 2, gitt at en kun mangler en måling, som gjør at x = 1. Brøken er der kun for å beskrive den relative forskjellen mellom punktene.

Data preprossesering inneholder visse steg:

1. **Data rensing (data cleaning):** Fyller inn manglende verdier og fikser eller fjerner støyfull data
2. **Data integrasjon (data integration):** Hente data fra forskjellige kilder og slår dem sammen (for at det skal bli lettere å jobbe med?)
3. **Data transformasjon (data transformation):** Normalisere og aggregere data etter hva du vil ha ut av datasettet?

4. **Data reduksjon (data reduction):** Vi kan redusere dataen, men bare med tanke på at den reduserte dataen vil gi samme resultat som den opprinnelige/orginale dataen.

Data rensingen hjelper med å oppdage feil og uregelmessigheter i dataen, noe som vil øke kvaliteten. Dette steget er også det viktigste steget i preprossesering av data fordi det vil garantere at dataen du har er klar til senere bruk. Det finnes mange måter å korrigere manglende data på, men de mest vanlige er å ignorere raden eller kolonnen med manglende data. Denne metoden blir ofte brukt på store datasett der noen få ignorerte verdier ikke vil ha noe å si for analysen. Den andre metoden som ofte blir brukt er å manuelt fylle inn for manglende data. Dette kan være tungvint, men er nødvendig i mindre datasett siden du ikke har mange verdier å gå utifra.

For manglende eller usannsynlige verdier er det to mulige valg:

1. Fjerne urealistiske målinger.
2. Estimere hva verdien kunne vært, om den hadde vært målt riktig

Formatering av data for kunne brukes i en modell går ut på å konvertere målinger til tallformat. Modeller har ikke mulighet til å bruke tekst til å lage prediksjoner, så disse må konverteres til tall som har tilhørende tekst verdi. En lager da et kart mellom Data rensing går også ut på å fikse støyfull data. Dette kan være data som inneholder unødvendige verdier eller verdier som er vanskeligere å gruppere sammen. Vi kan sortere data i et datasett inn i mindre grupper med like verdier, dette blir kalt for data binding og blir brukt for å analysere demografi. Vi kan for eksempel gruppere inntekt etter hvor mye som blir tjent. Regresjon er en annen metode som blir brukt for å bestemme hvilke variabler som vil bli brukt i analysen din. Regresjon blir brukt til å jevne ut store mengder med data. Dette vil hjelpe til å fjerne unødvendig data. Klyngeanalyse brukes ofte til å kartlegge strukturen til ukjente data og blir ofte brukt i unsupervised learning når vi ikke vet så mye om relasjonene i datasettet.

Data transformasjon er hvor vi gjør om formatet på dataen sånn at vi kan bruke dem i modellen og analysen. Dette gjør vi ved hjelp av å aggregere, normalisere og velge ut features. Data aggregering vil kombinere dataen vår inn i et uniform format. Ved at vi normaliserer dataen vi har inn i en bestemt rekkevidde, gjør at vi kan sammenligne den mer nøyaktig. Når vi velger ut features bestemmer vi hvilke variabler som er viktigst for analysen vår. Disse features vil bli brukt til å trene modellen vår.

Data reduksjon er en måte for oss å kutte ned på datasettet når du har mer data enn du trenger. Mer data kan føre til at det blir vanskeligere å analysere selv etter å ha renset og transformert det. Data reduksjon gjør datasettet lettere å analysere, det blir mer nøyaktig og i tillegg kutter vi ned på lagringen. Metoder som blir brukt er attribute selection, numerosity reduction og dimensionality reduction. Attribute selection kombinerer features og vil tilpasse dataen din i mindre porsjoner. Numerosity reduction er en data reduksjons teknikk som erstatter den originale dataen med mindre data som kan representere den originale. Dette hjelper med datalagring og dataoverføring. Dimensionality reduction er transformasjonen av data fra ett høyere dimensjon til et lavere dimensjon der dataen kan representere og i tillegg beholde viktige egenskaper til den originale dataen.

3.7 Omgjøring av data

I pandas finnes det flere ulike hjelpemidler som vi kan bruke for å få en god oversikt over dataen vi jobber med. For å sjekke for manglende verdier bruker vi pandas *DataFrame.isnull().sum()*. Dette vil liste opp alle manglende verdier i datasettet vårt. Når vi leser inn datasettet bruker vi et script som sjekker for akkurat dette, hvis det er en rad med manglende verdier vil denne kunne bli droppet basert på hva brukeren setter variabelen drop row til, se kode 3.1. Siden datasettet vårt inneholder store mengder data vil det ikke ha en noen innvirkning om vi dropper enkelte rader. Det vil derimot gjøre det lettere å Benytte seg av dataen.

For å droppe en rad bruker vi DataFrame.dropna(axis=0, inplace=True). En forklaring av

paramterne her er som følger:

1. axis, eller akse, er hvilken retning metoden skal utføres. Den går enten på kolonner eller rader, her
2. inplace er pandas sin syntaks for å bestemme om endringene skal påvirke det opprinnelige objektet eller om endringene må lagres i en ny variabel. Pandas dataframe er et mutable objekt, som vil si at det kan oppdateres direkte, dermed må inplace spesifiseres.

Kode 3.1: Sjekker for manglende verdier

```
def data_preprocessing(dataset_option, drop_row=True):
    df = load_data_into_dataframe(dataset_option)

    # Checking for empty cells
    missing_values = df.isna().sum()                                    # ...
    Counts number of empty cells
    missing_values_percentage = missing_values/len(df)*100              # ...
    Percentage of cells empty
    print(missing_values_percentage)




    if drop_row:
        df.dropna(axis=0, inplace=True)                                 # ...
    Removing rows with missing data
    elif not drop_row:
        pass # Interpolating?
    return df
```

Utdrag fra pandas med oversikt over manglende data

Kolonne	Manglende målinger %
LCLid	0
tstp	0
energy(kWh/hh)	0

Figur 3.13: Manglende målinger i Halfhourly datasett

Kolonne	Manglende målinger %
LCLid	0
day	0
energy_median	0.000855
energy_mean	0.000855
energy_max	0.000855
energy_count	0
energy_std	0.322781
energy_sum	0.000855
energy_min	0.000855

Figur 3.14: Manglende målinger i daily datasett

Som vi kan observere fra tabellen er det svært lav forekomst av manglende data, slik at en reduksjon basert på manglende verdier vil være tilnærmet ubetydelig for datasettet.

Det opprinnelige formatet som dato var lagret som i datasettet var inkompatibelt med modellen vår. Dette gjorde at vi måtte endre formatet til et som var kompatibelt, ved hjelp av en python-funksjon. Denne funksjonen tar in verdien fra det opprinnelige datasettet og splitter det opp, for så å kunne behandle de individuelle delene av strengen. Datoen settes til slutt sammen på en bedre egnet måte og lagres i det behandlede datasettet vårt. 3.2.

Kode 3.2: Konverterer til datetime

```
def unix_time(timestamp):
    values = timestamp.split(" ")
    date = values[0].split("-")
    year = int(date[0])
    month = int(date[1])
    day = int(date[2])
    date = datetime.date(year, month, day)
    clock = values[1].split(":")
    hour = int(clock[0])
    minute = int(clock[1])
    second = int(0)
    clock = datetime.time(hour, minute, second)
    return datetime.datetime.combine(date, clock)
```

Med det redigerte datasettet i output.csv filen kan vi starte på transformasjons delen. Verdiene i datasettet vårt var av typen string, noe som ikke er direkte kompatibelt med pandas. For å kunne bruke dataen måtte det konverteres til float og datoene måtte endres til datetime. Dette ble gjort ved å bruke pandas.DataFrame.astype funksjon, se kode 3.3

Kode 3.3: Konverterer data til bruk i modell

```
df['tstp'] = [datetime.strptime(x, '%Y-%m-%d %H:%M:%S') for x in df['...
    tstp']]
df["energy(kWh/hh)"] = pd.to_numeric(df["energy(kWh/hh)"], downcast="...
    float", errors="coerce")
```

Kapittel 4

Design og konstruksjon av programvare

4.1 Design av modellene

Etter preprosessering blir dataen brukt til å tilpasse modeller. Modellene er utviklet som en samling av klasser som i praksis følger et grensesnitt. Selv om det strengt tatt ikke er implementert som det, ettersom det ikke er naturlig å bruke interface i python. Ideen om å plassere maskinlæringsmodellen i en intuitiv klassestruktur med all nødvendig funksjonalitet kommer fra Eligijus Bujokas. Vi har utvidet og modifisert hans arbeid for LSTM til å være lettere å videreutvikle, og til å passe for andre modeller [7]. Grensesnittet er som følger, der (m) er for metode, og (p) for egenskap:

- **data (p)**
  Klassene får datasettet inn som en Pandas dataframe

- **train_test_split (p)**
  Skal ha et tall mellom 1 og 0 som bestemmer hvor stor del av dataen som er trening og test. Hvis verdien på denne egenskapen er 0.2 blir 80% av dataen brukt som trening, mens 20% er til å teste på.

- **init_model (m)**
  Metode for å sette opp og trene modellen.

- **predict (m)**
  Metode for å få spådommene av test datasettet

- **predict n ahead (m)**
  Metode for å få spådommer utover datasettet. Denne metoden bruker tidligere spådommer til å spå enda lengre inn i framtiden.

- **evaluator (p)**
  Klassen skal inneholde en implementasjon av en evaluator. Dette objektet skal kunne brukes til å bedømme treffsikkerheten til modellene, og har metoder for å gi resultatet av tapsfunksjonene MSE, RMSE, MAE, MAPE. I tillegg til andre evaluering vi ønsker å ta i bruk som å se på forskjellene mellom daglige topper for spådom og virkelighet. Dette har blitt et eget objekt istedet for å tilhøre hver enkel modell, fordi koden ellers ville vært repetert mange ganger, og vil da være vanskeligere å vedlikeholde.

I tillegg er modell-klassene satt opp til å bruke en egen klasse for data omforming: NN_data_creator.py, som kan erstattes. Det er utviklet to varianter, en som bruker temperaturen fra værdataen, og en uten. Denne klassen har som ansvar å segmentere datasettet til et treningsdatasett og et testdatasett. I tillegg til å lage inngangsvektorer i mengden X, og fasit i mengden Y.

Maskinlæringsklassene ligger fra prosjektrota i mappen models/ML_classes, og er gjort til en python pakke. Dette gir python notebook skriptene som ligger i forelder mappen tilgang

til klassene. Disse skriptene er ansvarlig for å kjøre modellene, og framstille resultatene av dem.

4.1.1 Utgangspunkt

Den første modellen er en enkel modell, der en velger å spå framtidens verdi som verdien ved forrige tidssteg. Denne modellen kan dermed sies å være god for kun en spådom om en halvtime fram i tid, ettersom spådommen om enda en halvtime vil være den samme som utgangspunktet. Denne modellen selv om den er naiv kommer veldig nærme testdataen ettersom den kun ligger et tidssteg bak, og strømbruken endrer seg ikke mye for hver halvtime.

Et mer passende utgangspunkt i mange tilfeller for å vurdere om maskinlæringsmodellene presterer er å tilpasse en lineær regresjon på dataen. Dermed er den andre modellen en lineær regresjonsmodell. Det er brukt scikit learn sin implementasjon av regresjon, og en har utviklet både enkel regresjon med x-aksen, og multippel regresjon med bruk av tidsetterslep. Men vi bruker kun resultater fra den enkle regresjonen.

4.1.2 Test oppsett

For inngangen til resten av modellene brukes tidsetterslep. Altså for hver måling lages det en inngangsvektor med n antall forrige tidssteg.

```
deep_learner.x_train[0]
```

```
array([0.03560697, 0.08248197, 0.04417067, 0.0640024 , 0.07932692,
       0.03771033, 0.22641225, 0.16000601, 0.14708534, 0.09705529,
       0.07587139, 0.07692308, 0.06385216, 0.04371995, 0.02208534,
       0.03245192, 0.02884615, 0.02824519, 0.02764423, 0.02914663,
       0.02779447, 0.02223558, 0.02118389, 0.0688101 ])
```

```
deep_learner.y_train[0]
```

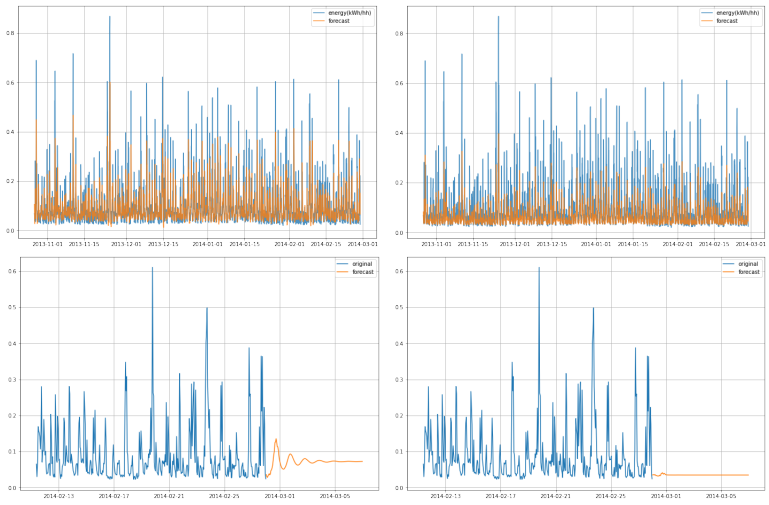
```
0.03485576750864912
```

Figur 4.1: Model input tidsetterslep

Hvis en har et tidsetterslep på 1, blir data for forrige time brukt til å forutsi en time framover. Hvis tidsetterslepet er på 24 timer, blir data fra et døgn tilbake i tid brukt for prognosen osv. I figur 4.1 ser en et eksempel på innganger sammen med den tilsvarende fasiten.

Verdiene inn og ut til modellen er normaliserte ved å dele hver måling på datasettets høyeste verdi, verdiene blir da mellom 0 og 1. Hvis du utvider ideen om tidsetterslep til å gjelde for alle parametere som er sendt inn, kalles det også et tidsvindu. Her er verdien om hvor lang tid det er mellom hver verdi i vinduet kodet inn i datasettet. I figur 4.1 hentet vi dataen ut fra et datasett med en times mellomrom fra målingene. Med vårt datasett kan vi på det meste spå for hver halvtime, for at spådommene fortsatt er gjort basert på målinger fra forrige døgn må inngangsvektoren holde på 48 tidligere målinger.

Dyplærings modellene er laget ved bruk av Tensorflow keras versjon 2.8 [4]. TF keras gjør det lett å endre modellparametere som hvilke aktiveringsfunksjon som er brukt for hvert lag. Hvilken tapsfunksjon modellen skal trenes etter, og til slutt hvilken optimaliserer modellen bruker. Optimalisereren som er brukt kalles adam. Vi velger å trene mot MSE som kostnadsfunksjon i alle relevante tilfeller. Alternativet MAE er ikke sensitiv nok mot topper i datasettet siden den ikke straffer store feil. Bruk av MAE gir for flat linje for spådommene i vårt tilfelle, se figur 4.2, men er nyttig dersom uteliggerne i datasettet er støyfulle målinger en ikke vil ta for mye hensyn til. En vil derimot prøve å filtrere ut støyen før en mater dataen inn i en modell.



Figur 4.2: MSE optimalisert til venstre, MAE optimalisert til høyre (tatt fra en lstm modell), spådde verdier i oransj

Om ingenting annet er sagt er aktiveringsfunksjonen relu tatt i bruk for skjulte lag. Antallet av epoker å trene dataen på er funnet ved eksperimentering, mens antall målinger i hver batch er konstant 256.

Evaluering

Modellene blir evaluert ved å se resultatet av de forskjellige kostnadsfunksjonene MSE, RMSE, MAE og MAPE. RMSE blir trukket fram foran MSE ettersom rota av MSE gir et resultat som er lettere for mennesker å lese, RMSE blir større hvis $1 > \text{MSE} < 0$, og hvis $\text{MSE} > 1$ mindre. MAPE oppgis i prosent som de aller fleste har et forhold til, og vil derfor bli diskutert mest.

4.1.3 Beslutningstre

Kode 4.1: DT

```
1       # Defining the model
2          model = tree.DecisionTreeRegressor()
3          model = model.fit(X_train, Y_train)
```

Med tree fra sklearn kan en enkelt lage et beslutningstre.

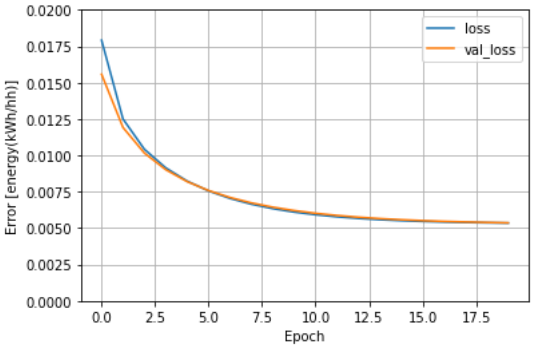
4.1.4 Perceptron, SLP og MLP

Perceptron, SLP og MLP, kan alle implementeres gjennom samme klasse, ettersom alle er vanlig ANN kun med varierende antall tf.keras Dense lag og nerveceller. En må ha et Input lag for å kunne korrekt oversette inngangsvektorene.

Kode 4.2: Perceptron

```
1          # Defining the model
2          model = Sequential()
3          model.add(Input(shape=(len(X_train[0])),))
4          model.add(Dense(1))
5          model.compile(optimizer='adam', loss='mse')
```

En har likevel valgt å skille ut perceptron tilfellet i en egen klasse LinearModel.py siden den har en spesiell oppførsel. Perceptronet er nemlig også en implementasjon av lineær regresjon bare ved bruk av et neuralt nettverk. I den forstand blir modellen en multivariabel regresjon på tidsetterslepet av energimålinger.



Figur 4.3: trening på MSE (perceptron)

Av figuren over, og tabellen under kan en se at det er tilstrekkelig å trene et sted mellom 10-20 epoker. Vi har observert at alle modellene også MLP er omtrent ferdig trent etter 20 epoker, og bruker dermed å trene for 20 epoker i resultatet.

epochs	MAPE (%)	mse
1	137.28	0.0125
5	81.44	0.0064
10	54.45	0.0048
20	45.67	0.0046
30	44.78	0.0046
40	44.8	0.0046

Tabell 4.1: Perceptron trening

For å dynamisk bygge en vilkårlig SLP eller MLP blir en liste med antall nerveceller i hvert lag sendt inn som parameter i klassens konstruktør. For et nettverk med 2 skjulte lag, det første med 10 nerveceller, og det andre med 5, blir listen slik: [10,5].

Kode 4.3: SLP og MLP

```
1       # Defining the model
2          model = Sequential()
3          model.add(Input(shape=(len(X_train[0])),))
4          #building layers to specification.
5          for j in range(self.layer_count):
```

```
6                 model.add(Dense(self.layer_depths[j], activation="relu"))
7             model.add(Dense(1))
8             model.compile(optimizer='adam', loss='mse')
```

Testing av antall nerveceller i SLP gir ingen betydelige resultater, vi har vilkårlig valgt å bruke et lag med 30 nerveceller.

units	rmse	mape
10	0.0680	45.52
20	0.0671	45.166
30	0.0674	49.93
40	0.6750	49.35
50	0.0667	47.34
60	0.0673	48.78
70	0.0674	51.83
80	0.0673	48.26
90	0.0674	46.16
100	0.0670	47.833

Tabell 4.2: antall nerveceller

Utifra tabellen under ser det ikke ut som at en kan få ytterligere utbytte ved å bruke en MLP modell.

units	MAPE (%)	rmse
30 (controll)	50.59	0.0674
10,10	49.55	0.0676
20,10	45.22	0.0674
20,20	49.43	0.0677
30,10	47.19	0.0676
30,20	51.31	0.0673
30,30	48.12	0.0674
10,10,10	48.05	0.0677
20,15,10	46.87	0.0675
20,10,5	48.88	0.0674
30,20,10	48.58	0.0673
30,15,10	50.59	0.0679
40,20,10	49.21	0.0675
50,30,10	50.26	0.0674

Tabell 4.3: testing av MLP med forskjellige lag

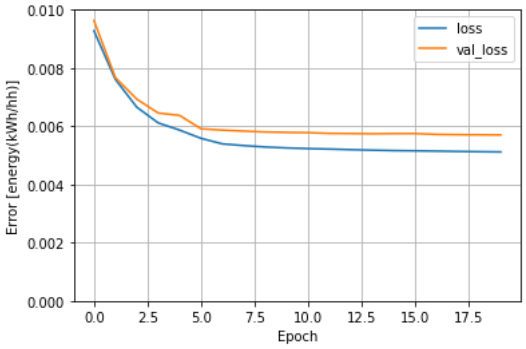
Derfor har vi valgt å ikke gjennomføre videre undersøking av MLP

4.1.5 LSTM

LSTM klassen har også evnen til å ha et vilkårlig antall med LSTM-lag ved å få inn en liste som parameter. Koden fra tensorflow for LSTM lag er derimot mindre robust, som er grunnen til den mer fiklete if-setning strukturen i koden under. Skal en ha enda et lag på toppen av et LSTM lag må dette laget ha: return_sequences=True. Men kan ikke ha return_sequences=True, dersom det er det siste laget i nettverket.

Kode 4.4: LSTM

```
1         # Defining the model
2          model = Sequential()
3          if len(self.LSTM_layer_depths) > 1:
4
5              model.add(LSTM(self.LSTM_layer_depths[0], ...
                   activation='relu', input_shape=(len(X_train[0]), 1), ...
                   return_sequences=True))
6              for idx, dephts in enumerate(self.LSTM_layer_depths[1:]):
7                  if idx+1 == len(self.LSTM_layer_depths)-1:
8                       model.add(LSTM(dephts, activation='relu'))
9                  else:
10                      model.add(LSTM(dephts, ...
                            activation='relu',return_sequences=True))
11
12         else:
13             model.add(LSTM(self.LSTM_layer_depths[0], ...
                   activation='relu', input_shape=(len(X_train[0]), 1)))
14         model.add(Dense(1))
15         model.compile(optimizer='adam', loss='mse')
```



Figur 4.4: trening på MSE

Etter 20 epoker er lstm modellen også ferdigtrent.

Etter 20 nerveceller i LSTM laget, ser en ingen særlig forskjell i resultater.

LSTM layer depth	MAPE (%)	rmse
1	44.4	0.0959
10	43.8	0.0728
20	48.8	0.0714
30	51.2	0.0695
40	48.2	0.0705
50	46.9	0.0698
60	49.2	0.0701
70	45.8	0.0701
80	44.7	0.0707
90	48.3	0.0694
100	47.0	0.0697
110	46.6	0.0695
120	49.7	0.0696
150	45.2	0.0697
200	48.82	0.0693

Tabell 4.4: antall nerveceller

flere lag

Å utvide LSTM modellen til flere lag ser ikke ut til å nødvendigvis øke presisjonen til modellen. Konfigurasjonen i tre lag med (40,30,10) og (50,30,10) ser noe lovende ut, men ikke nok til å kompensere for økt kjøretid i vår mening

units	MAPE (%)	rmse
50 (controll)	48.98	0.0675
10,10	51.44	0.0683
20,10	45.25	0.0675
20,20	45.18	0.0682
30,10	47.83	0.0682
30,20	48.22	0.0676
30,30	48.30	0.0673
40,10	44.45	0.0680
40,20	53.30	0.0676
40,40	56.89	0.0675
50,10	49.84	0.0673
50,30	52.06	0.0672
50,50	45.87	0.0680
10,10,10	46.39	0.0679
20,10,5	46.98	0.0680
30,20,10	45.68	0.0685
40,20,10	47.24	0.0670
40,30,20	50.11	0.0674
50,30,10	51.78	0.0670

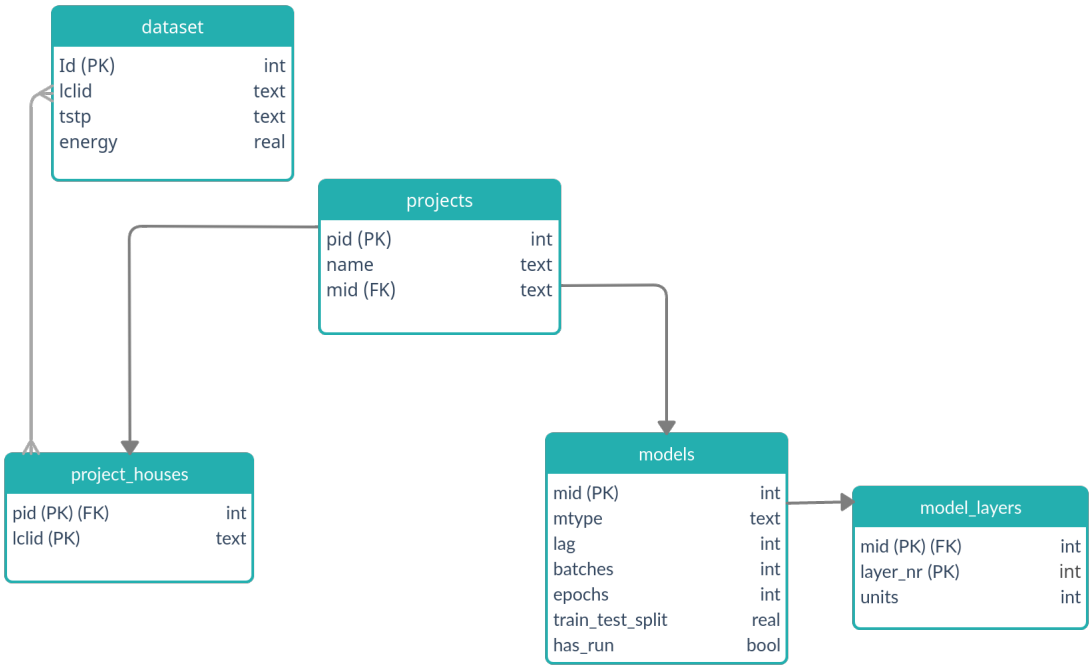
Tabell 4.5: testing av LSTM med flere lag

4.2 Dashboard

Kodestrukturen bak dashboardet er som tidligere nevnt strukturert i to deler, server og klient. Serversiden er en flask API med tilgang til database-funksjoner og klassene for maskinlæringsmodellene. Klient siden er en React-app som håndterer det brukeren direkte kan benytte seg av.

4.2.1 Databasestruktur

Databasestrukturen er basert rundt en Python-fil kalt setup_db.py. Denne filen har kode for generering av en SQLite database (database.db) samt alle funksjonene som direkte kommuniserer med databasen. Funksjonene er skrevet i Python, men benytter seg av SQL-kode i form av strenger for å kunne påvirke databasen.



Figur 4.5: database skjema

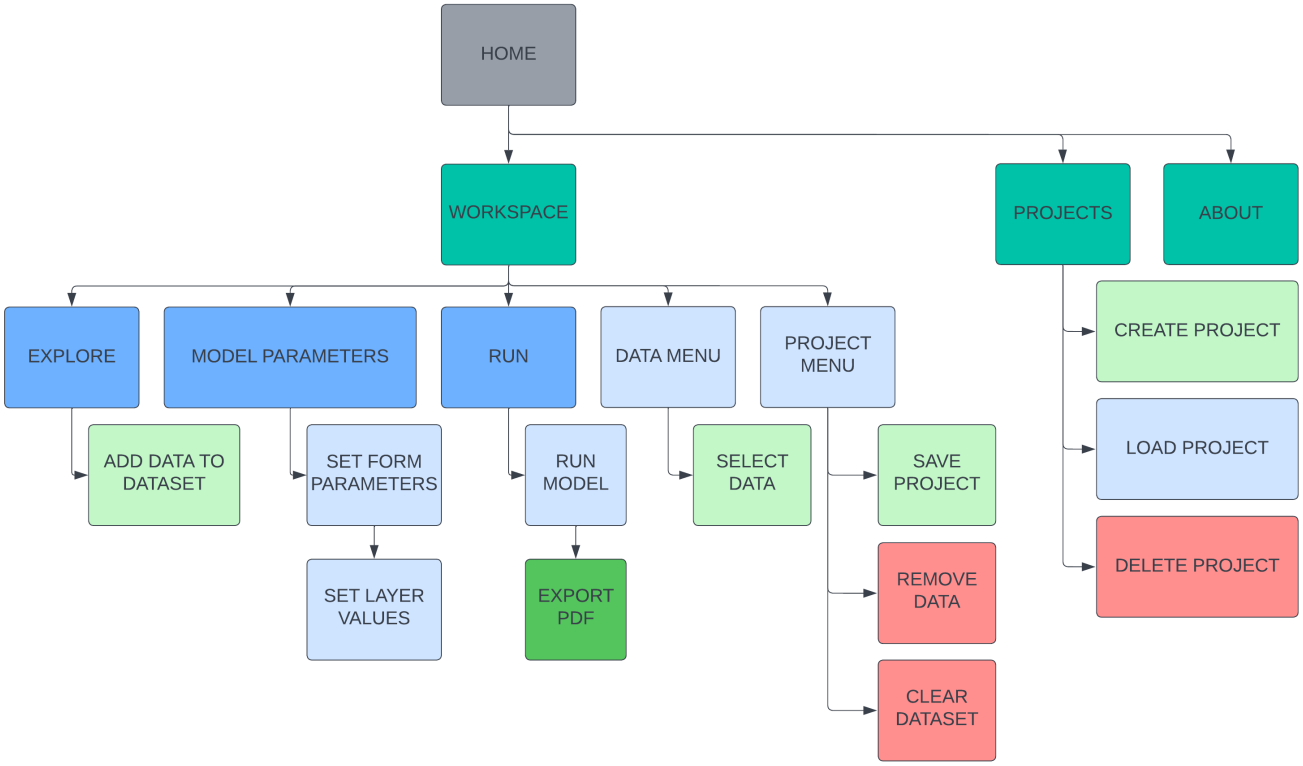
I figur 4.5, betyr pilene at tabellen tilhører pilens opprinnelse. I dataset, har vi lagret informasjonen i det halvtimelige datasettet. En var inne på å lagre en ekstra tabell for datasettet som ga informasjon om husholdet, dette har ikke blitt gjennomført ettersom det var unødvendig for utviklingen av maskinlæringsfunksjonalitet. Men med tanke på distribusjon vil det være en enkel endring å gjennomføre slik at brukeren kan se fra hva slags hushold dataen kommer fra. Dette henger sammen med at vi kun lastet inn de ti

tidligere selekterte husene inn i databasen i første omgang, dette tok likevel i overkant av en time å utføre. Å bruke tid på laste inn flere av de 5566 husene ble sett på som bortkastet tid under utviklingen. Likevel valgte vi å plassere datasettet i databasen for å kunne tjene på raskere oppslagstid. Prosjektet er objektet nærmest brukerens grensesnitt. Project_houses lagrer en liste av hvilke hus brukeren har valgt å ta med i prosjektet sitt, dermed kan en hente ut dataen fra dataset. Dette er en mer økonomisk løsning med tanke på lagringsplass, men krever litt mer prossessor ressurser for å hente ut dataen, enn å lagre prosjektets data på nytt i tabellen i stedet. En kan argumentere at lagringsplass er en billig ressurs, men det argumentet blir fort overskygget med tanke på hvor mange tunge skriveoperasjoner serveren må gjøre for å legge hver måling i databasen. Et prosjekt kan ha en modell bundet til seg.

lagring av maskinlæringsmodellen

For dashboardet ønsket vi å kunne lagre modellen for neste gang brukeren laster inn prosjektet med en kjørt modell. Modell parameterne kan lagres i databasen (fig. 4.5), men ikke selve modellen. Heldigvis for oss har tensorflow en innebygd funksjon for lagring og lasting av modellobjektet. Ved bruk av disse funksjonene lagrer vi modellen på filsystemet basert på mid som navn på mappen der den skal ligge. For beslutningstre modellen bruker en implementasjonen til scikit-learn. SK-learn har ingen slik lagringsfunksjon, men en har brukt python-biblioteket pickle for tilsvarende funksjonalitet.

4.2.2 Funksjonalitet



Figur 4.6: Sidekartet viser navigasjon på siden fra topp til bunn.

Sidekartet viser en oversikt over hvilke sider og funksjoner som er tilgjengelig i applikasjonen. Applikasjonen begynner på Home, og laster in Workspace som hovedside. Det kan navigeres til de 3 hovedsidene i toppmenyen. Hovedfunksjonene til applikasjonen befinner seg i Workspace, hvor det er en egen meny for navigasjon. Der er Explore, Model Parameters og Run. I tillegg til dette finner vi to sidemenyer, én på hver side. Funksjonene som er tilgjengelig i workspace er:

- Datamenyen har funksjoner for å velge data, mens prosjektmenyen har mulighet for

lagring av prosjekt, fjerning av enkeltdata, og tømming av hele datasettet.

- Explore siden lar brukere legge til data i datasettet.
- Model Parameters lar brukeren endre parametere for modellen, som også lagrer dem til prosjektet.
- Run er hvor brukeren kan kjøre modelen med datasettet og valgte parametre. Det er også en funksjon for lagring av resultatet som PDF.

I tillegg til hovedsiden finnes det en prosjektside hvor de tilgjengelige funksjonene er:

- Oppretting av prosjekt.
- Lasting av prosjekt.
- Sletting av prosjekt.

Til slutt er det en side med kortfattet informasjon om applikasjonen.

4.2.3 Brukergrensesnitt

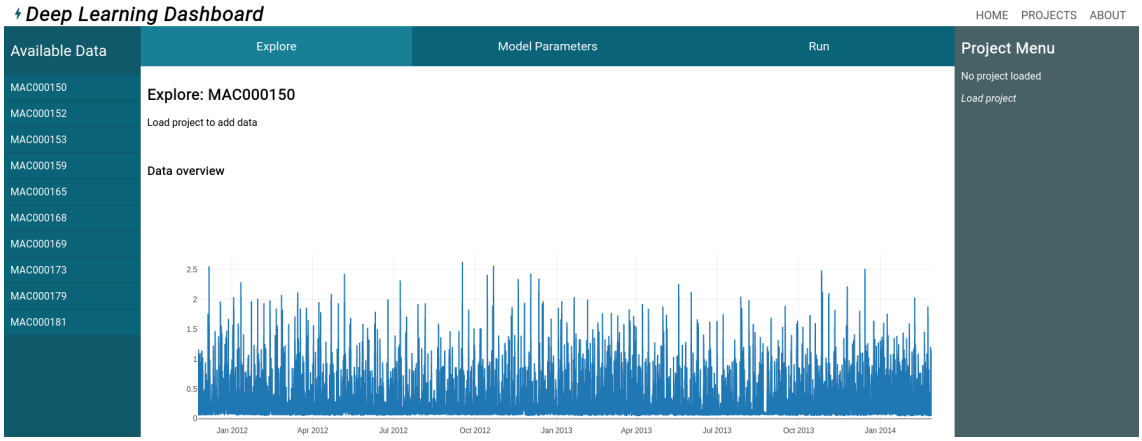
Det ble bestemt tidlig i prosjektet at applikasjonen utelukkende skulle utvikles for bruk på datamaskiner, og den er derfor ikke tilpasset mobiltelefoner eller nettbrett. Brukergrensesnittet ble utviklet ved bruk av skjermer med oppløsningen 1920px * 1080px, men er testet på skjermer opp til 3840px * 2160px uten at det ble oppdaget problemer. Den minimale bredden på skjermen før det blir utfordringer å bruke applikasjonen er 1400px.

Brukergrensesnittet er forsøkt holdt så enkelt som mulig uten å fjerne nødvendig funksjonalitet. Det er brukt ikoner fra Google Fonts [9] som skal gi en raskt overblikk over funksjonalitet uten at brukere trenger å lese for mye tekst. Alle ikoner har for brukervennlighet tilhørende title"attributter med en kort forklaring av funksjonaliteten.

I fargevalget er det forskjellige blånyanser som gir et avslappet inntrykk. Hovedfargen er rgb(16, 89, 106). For å indikere funksjonalitet på ikoner og knapper er det brukt CSS egenskapen `:hover` på elementer med skiftende farge på tekst og/eller bakgrunn. Dette er en enkel måte for brukere å oppfatte funksjonalitet.

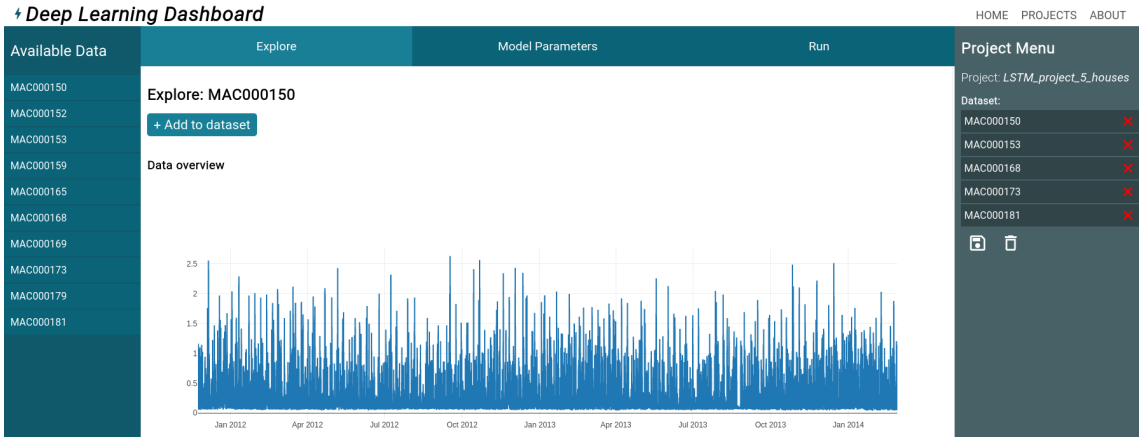
Forside

Det første brukeren møter er forsiden, som automatisk går til datautforskningsiden. For å tydeliggjøre om brukeren har lastet inn et prosjekt eller ikke vil det være betinget formatering som tydeliggjør dette.



Figur 4.7: Forside før bruker laster in prosjekt

I figur 4.7 vises siden før prosjekt er lastet inn. Brukeren vil her ikke kunne legge til data i datasett, og det vises heller ikke noe prosjektdata. Det er allikevel mulig å utforske data på datautforskningssiden. For å tydeliggjøre at det må lastes inn et prosjekt er det her gjentatte ganger vist informasjon som tilsier dette.

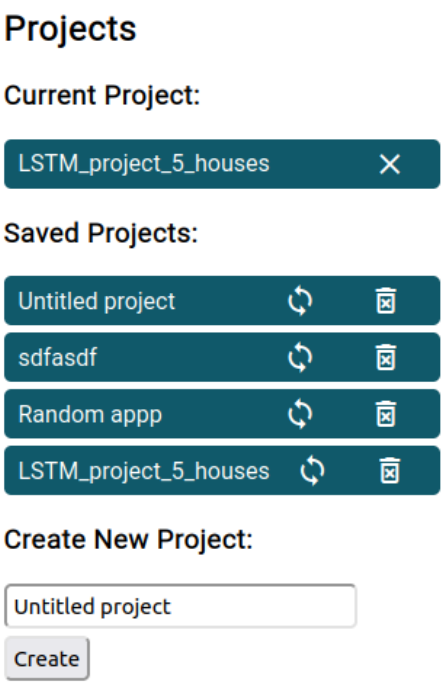


Figur 4.8: Forside med lastet prosjekt

Etter at et prosjekt har blitt lastet inn kan en bruker benytte seg av all funksjonalitet i applikasjonen, se sidekartet i figur 4.6. I figur 4.8 vises applikasjonen etter at et prosjekt er lastet inn. Nå tilbys mer informasjon, og det gis nå et helhetlig inntrykk.

Prosjekthåndtering

Som nevnt tidligere er Dashboardet utviklet med tanken om at en bruker lager og forvalter forskjellige prosjekter. Prosjekter identifiseres med navn som er synlig for brukeren.



Figur 4.9: Oversikt over prosjekter

I prosjekt siden, se figur 4.9, kan en lage og navngi et nytt prosjekt. En kan laste inn lagrede prosjekter. Det nåværende prosjektet er vist øverst i oversikten. Sletting av et prosjekt vil også slette all tilhørende data for modellen i databasen og i filsystemet.

Her er det verdt å merke seg at det benyttes ikoner for det forskjellige funksjonene til prosjektene.

- Et kryss for lukking av prosjekt
- Søppelbøtte-ikon som indikerer sletting av prosjekt
- Sirklende piler for å indikere lasting av prosjekt

Tilgjengelige data

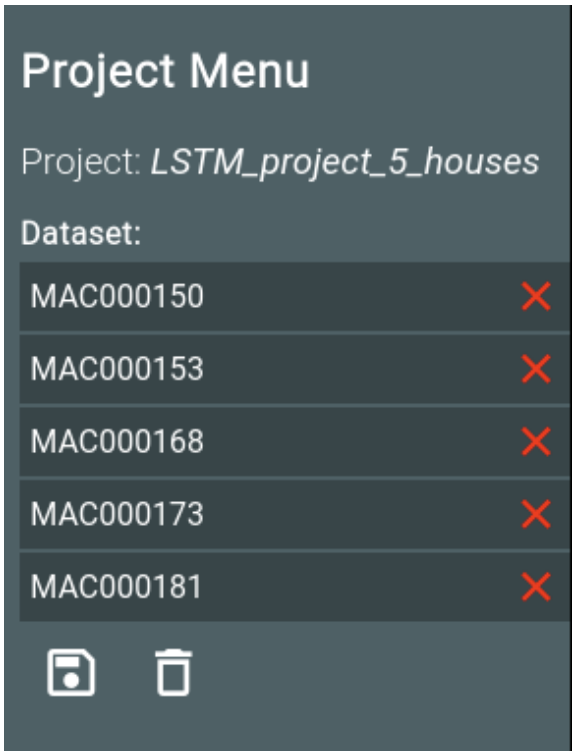
Alle tilgjengelige data er listet opp på venstre side av applikasjonen som vist i figur 4.8. Her benyttes CSS egenskapen `overflow: scroll;` som tilater datamenyen å være av en betydelig lengde uten at den skaper problemer for sideoppsettet.

Figur 4.10: Liste over tilgjengelig data

Med et nærmere blikk på datamenyen ser en i figur 4.10 at hvert data-element er i et vertikalt listeoppsett, med behagelig avstand til andre elementer. Elementene har en annen fargenyanse rundt bakgrunnen for å tydeliggjøre området som reagerer på trykk til hvert enkelt element. All tilgjengelig data er i tillegg vist til brukeren i sortert rekkefølge, noe som gjør det enklere å finne frem til spesifikke elementer.

Prosjektmeny

Prosjektmenyen er som vist i figur 4.8 plassert på høyre side, og har en annen farge enn resten av innholdet. Dette er gjort for å skape et visuelt skille mellom generell data og det som er spesifikt for prosjektet. Plasseringen av prosjektmenyen på høyre side er inspirert av andre dashbord-applikasjoner, hvor det virker til å være normen.



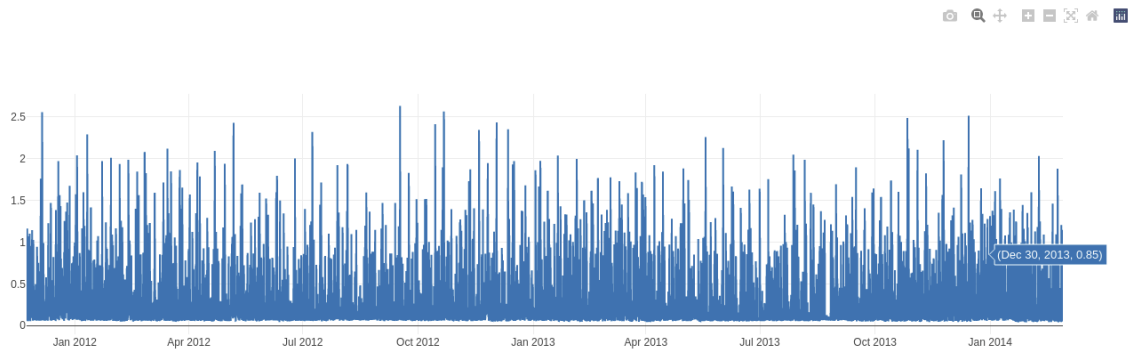
Figur 4.11: Oversikt over prosjektdata

Prosjektmenyen viser prosjektnavn øverst og benytter seg også av en vertikalt listeoppsett for å vise prosjektdata. Den betydelige forskjellen, som vist i figur 4.11 er at elementene her har et tilhørende kryss, for fjerning av data fra datasettet. Krysset har rød farge, som bidrar til synlighet, samt indikerer at det er en advarsel knyttet til det. I tillegg til mulighet for manipulering av prosjektdata på hvert enkelt element er det benyttet et

søppelbøtteikon som er knyttet til tømming av all prosjektdata. Dette befinner seg på siden av et lagringsikon (floppydisk) som gjør det mulig å lagre all prosjektdata til prosjektet.

Plot

Gjennomgående i applikasjonen er bruk av plot. Plottene lages ved bruk av Plotly [12] som tilbyr flere forskjellige typer plot, sammen med verktøy for brukerinteraksjon.

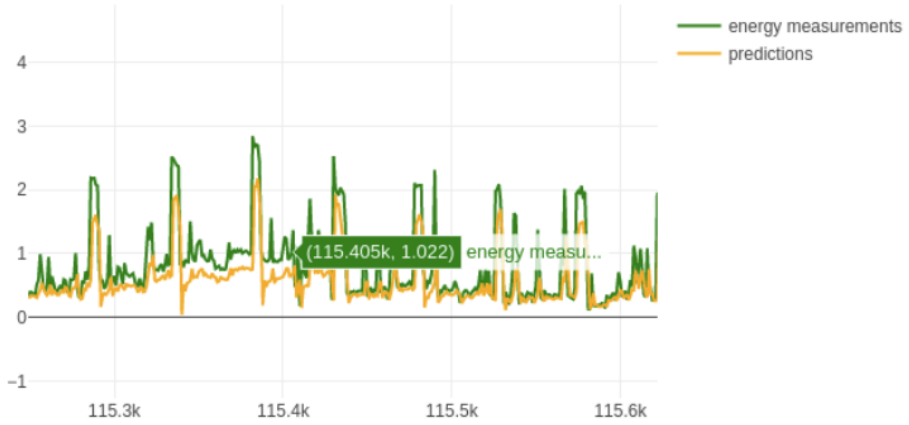


Figur 4.12: Dataplot ved bruk av Plotly

I figur 4.12 vises et utklipp fra datautforskning, hvor en kan se tilgjengelige data som hoveddel, med en verktøylinje øverst til høyre.

Figur 4.13: Verktøylinje plot

De tilgjengelige funksjonene i verktøylinjen er vist i figur 4.13 og er, fra venstre til høyre: øyeblikksbilde som png, zoom, flytt, zoom inn, zoom ut, autoskalering, tilbakestilling av akser, plotly-informasjon. I tillegg til disse knappene kan mye utforskning utføres ved bruk av bare musepekeren, hvor det å zoome inn gjøres ved å holde inne og dra over ønsket område, og tilbakestilling gjøres med et dobbelttrykk.

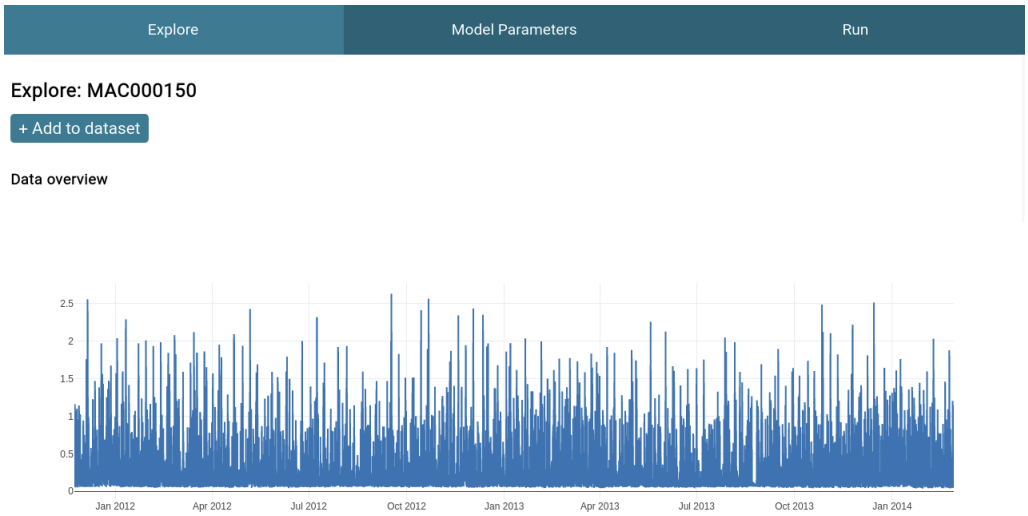


Figur 4.14: Utforskning av plot

Figur 4.14 viser et utklipp av hvordan mye arbeid med de tilgjengelige plottene foregår. Her er det zoomet inn for å vise omtrent én uke med målinger, sammen med prediksjonene for denne uken. Det kan raskt utforskes områder av interesse, uten at det krever bruk av input-felt, eller søk.

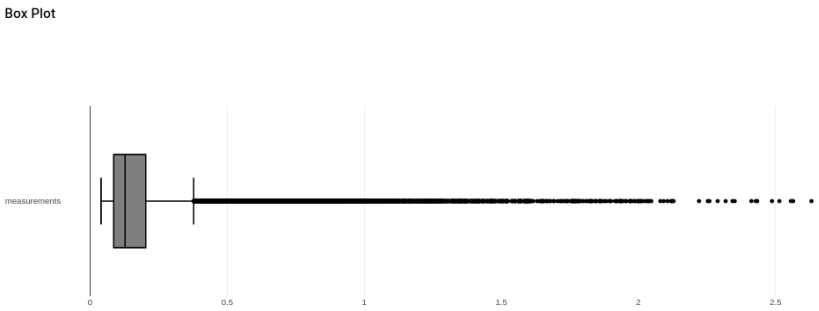
Datautforskning

Mesteparten av tid brukt i applikasjonen kan tenkes å være datautforskningen, derfor tilbys det informasjon i prioritert rekkefølge for at brukeren for skal kunne gjøre en vurdering før det bestemmes om det er verdt å utforskes nærmere.



Figur 4.15: Datautforskning med plot

Den umiddelbare informasjonen om valgt data er i form av et plot med alle strømmålingene til husstanden som vist i figur 4.15. Dette er plottet med KwH på Y-aksen og tidssteg på X-aksen.



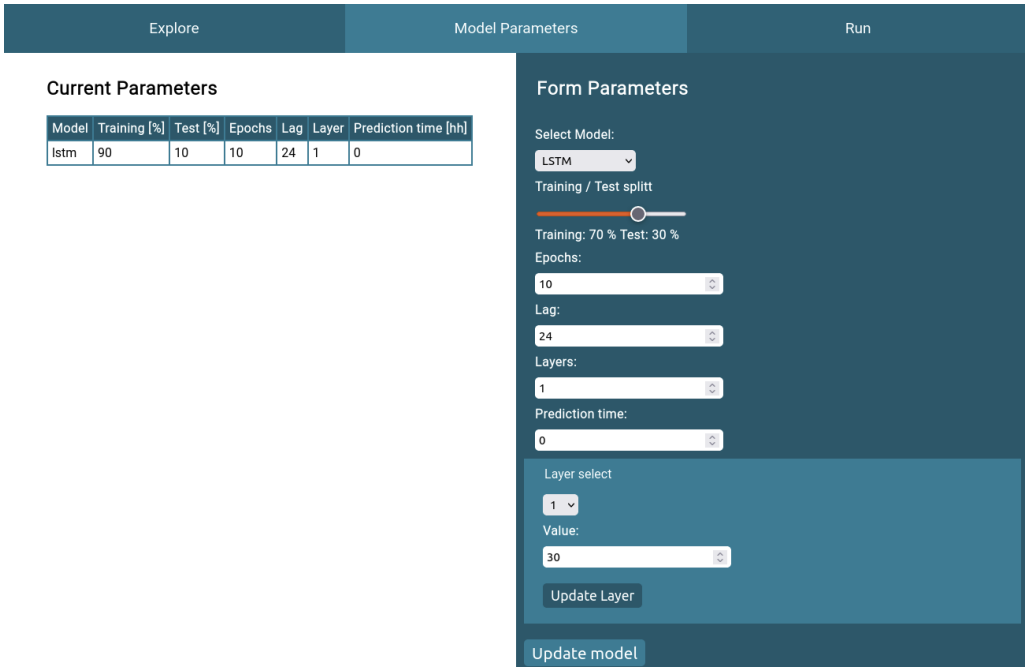
Column	Count	Mean	Std.	Min.	Max.
Energy	39719	0.20339927992396648	0.23203260226243758	0.04	2.631
Date	39719	2012-6-9 3:00:00	2012-6-9 3:00:00	2011-11-23 10:00:00	2014-02-28 00:00:00

Dataset head:			
id	LCLID	tstp	Energy
356834	MAC000150	2011-11-23 10:00:00	0.301
356835	MAC000150	2011-11-23 10:30:00	0.127
356836	MAC000150	2011-11-23 11:00:00	0.201
356837	MAC000150	2011-11-23 11:30:00	0.564
356838	MAC000150	2011-11-23 12:00:00	0.159
356839	MAC000150	2011-11-23 12:30:00	0.059
356840	MAC000150	2011-11-23 13:00:00	0.096
356841	MAC000150	2011-11-23 13:30:00	0.075
356842	MAC000150	2011-11-23 14:00:00	0.064
356843	MAC000150	2011-11-23 14:30:00	0.318

Figur 4.16: Datautforskning med tabeller

I tillegg til å plotte tidslinje diagrammet en ser i figur 4.15, inneholder explore også et boxplot, rådataen, og et sammendrag. Sammendraget forteller om antall målinger, gjennomsnitt, standardavvik, minimumsverdi, og maksimumsverdi, se figur 4.16. Ved å trykke: Add MAC000168 to dataset, blir dataen lagt inn i prosjektmenyen til høyre i figur 4.11. Fra denne prosjektmenyen kan en fjerne dataen hvis en er misfornøyd eller lagre den til senere.

Model parametre

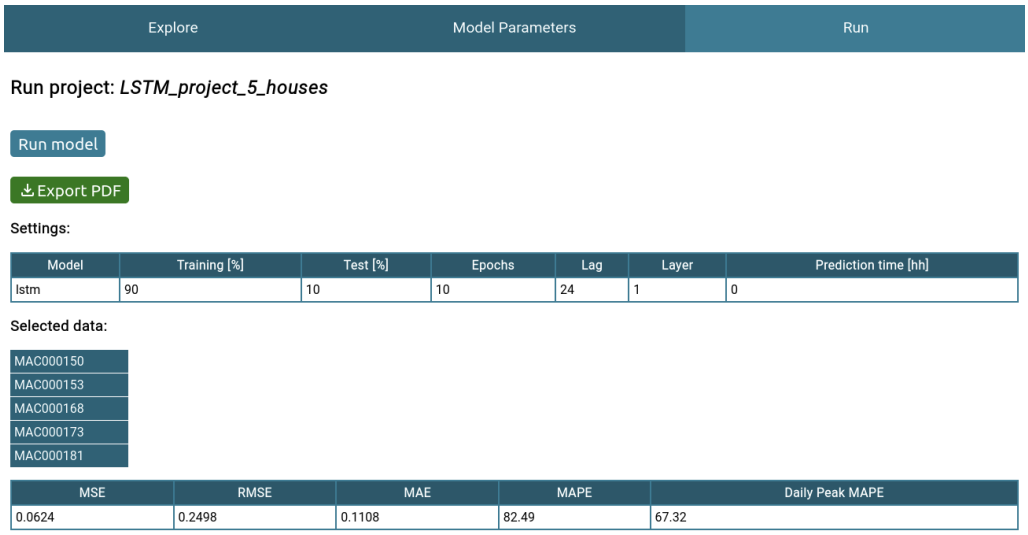


Figur 4.17: Endring av modelparametere

Når en har studert og lagt inn all prosjektdataen en har tenkt å bruke, er det tid for å konfigurere hvilken modell en har tenkt å bruke. De modellene som til nå er implementert i nettsiden er DT, LSTM og MLP. I figur 4.17 er LSTM valgt. For LSTM og MLP kan en i tillegg til de andre konfigurasjonene vist i figuren velge antall lag og enheter.

Kjør

Siden for å kjøre prosjektet er tilgjengelig med lastet prosjekt.



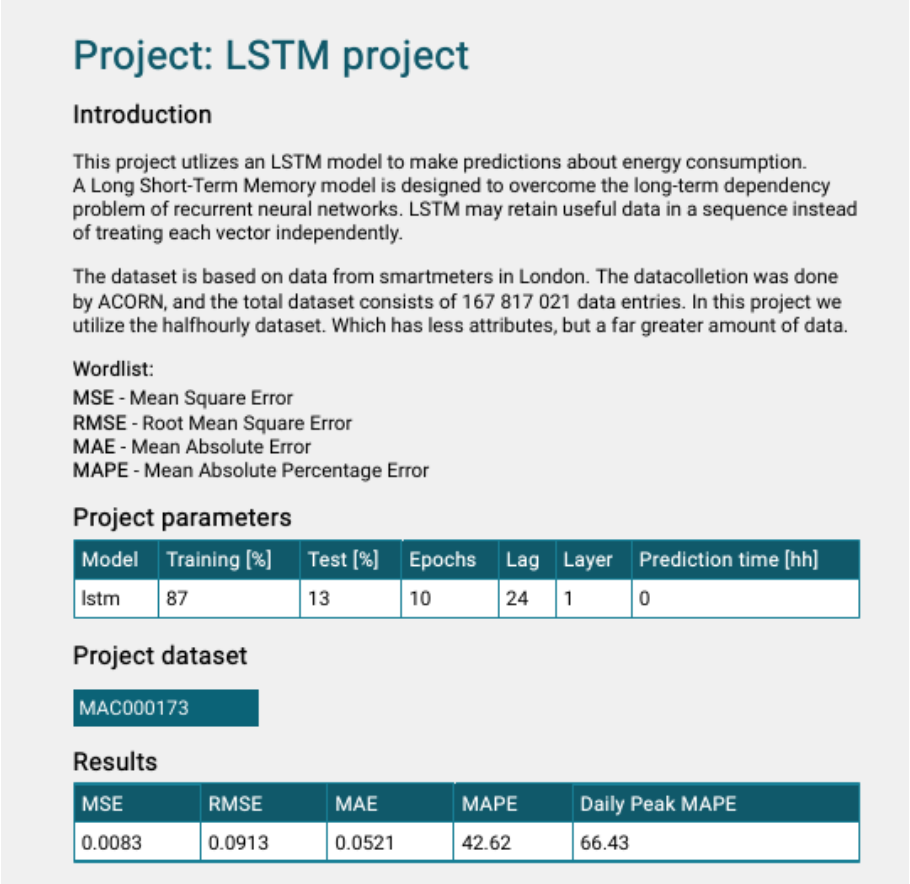
Figur 4.18: Kjøring av prosjekter

På denne siden vil det basert på lagrede hendelser i prosjektet kunne presenteres resultat fra tidligere modeller. Det vil uansett være mulig å kjøre modellen på nytt, eller kjøre den for første gang. Her vises nok en gang valgte data, samt parametere for modellen, slik at en slipper å bla tilbake til andre sider for å finne denne informasjonen. Som vist i figur 4.18 vil et prosjekt som er ferdig å kjøre presentere mulighet for kjøring på nytt, eller eksportering av resultat. Her er det valgt en grønn farge på eksport-knappen, som kan assosieres med lagring. I tillegg til bruk av passende farge er det igjen brukt visuelle hjelpemiddel, her i form av et klassisk nedlastningsikon.

Export av resultat

For eksportering av resultat ble det benyttet ReactToPrint [10], som muliggjør eksportering av definerte elementer/områder fra HTML koden. ReactToPrint baserer seg på nettleserens innebygde print egenskaper, som inneholder mulighet for lagring som PDF. Seksjonen som eksporteres er for brukeren ikke synlig da den har CSS egenskapen `display: none;` på beholder, som ved hjelp av CSS-regelen for printing, funnet under `@media print`. Her endres

`display` til å være `block`, slik at den synliggjøres for print.



Project: LSTM project

Introduction

This project utlizes an LSTM model to make predictions about energy consumption. A Long Short-Term Memory model is designed to overcome the long-term dependency problem of recurrent neural networks. LSTM may retain useful data in a sequence instead of treating each vector independently.

The dataset is based on data from smartmeters in London. The datacolletion was done by ACORN, and the total dataset consists of 167 817 021 data entries. In this project we utilize the halfhourly dataset. Which has less attributes, but a far greater amount of data.

Wordlist:

MSE - Mean Square Error
RMSE - Root Mean Square Error
MAE - Mean Absolute Error
MAPE - Mean Absolute Percentage Error

Project parameters

Model	Training [%]	Test [%]	Epochs	Lag	Layer	Prediction time [hh]
lstm	87	13	10	24	1	0

Project dataset

MAC000173

Results

MSE	RMSE	MAE	MAPE	Daily Peak MAPE
0.0083	0.0913	0.0521	42.62	66.43

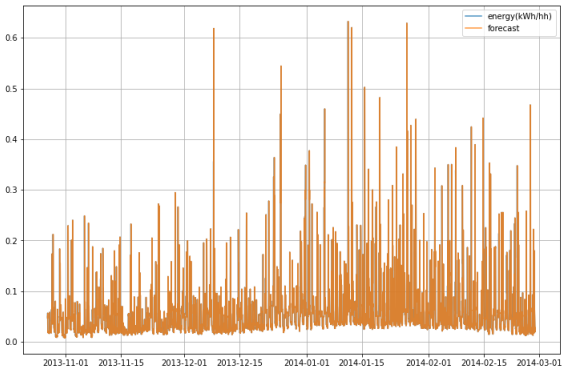
Figur 4.19: Utklipp fra eksportert PDF

I figur 4.19 vises et utklipp av hvordan eksporterte resultater vises frem. I tillegg til informasjonen vist i figuren legges det ved plot, som er like i utforming til de som presenteres på resultatsiden. Ettersom rapporten må kunne fungere selvstendig er det skrevet korte introduksjoner for de forskjellige modellene, som basert på hvilken modell som er kjørt legges til på toppen. Det gis også informasjon om datasettet, samt en ordliste som forklarer forkortelser brukt i rapporten. Navnet på den eksporterte rapporten blir ved hjelp av variabelbruk i React satt til å være likt prosjektnavnet.

Kapittel 5

Resultater og diskusjon

5.1 En enkel modell



Figur 5.1: prognosen, plottet mot test data for den enkle modellen

Selv om dette er en veldig enkel modell, er også modellen svært godt tilpasset treningsdataen, ettersom strømforbruket sjelden endrer seg svært mye for hver halvtime, dette kan en se tydelig i figur 5.1. Denne modellen er kun god for en spådom utenfor innsamlet data, men for den ene spådommen er modellen vanskelig å slå.

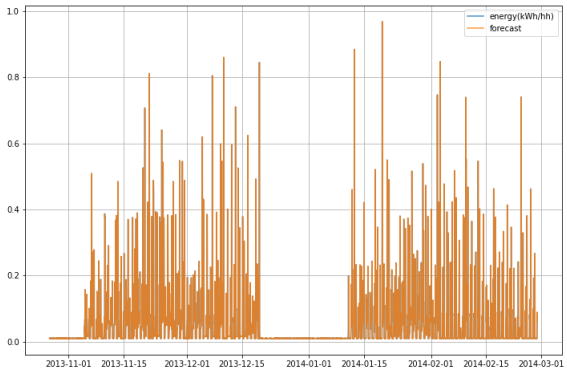
Hus	MSE	RMSE	MAE	MAPE (%)	merknader
MAC000150	0.0060	0.0778	0.0394	43.18	
MAC000152	0.0833	0.0833	0.0402	59.26	
MAC000153	0.0078	0.0888	0.0491	40.27	
MAC000165	0.0077	0.0879	0.0461	22.55	
MAC000169	0.0044	0.0667	0.0314	37.60	
MAC000168	0.0060	0.0779	0.0241	60.59	Perioder med veldig lite strømbruk
MAC000159	0.0029	0.0540	0.0254	28.56	
MAC000173	0.0090	0.0950	0.0548	47.70	
MAC000179	0.0033	0.0576	0.0297	45.08	
MAC000181	0.0020	0.0453	0.0181	26.07	

Tabell 5.1: resultater fra baseline modell (halvtime)

Modellen blir noe dårligere hvis vi spår for hver time, tabell 5.2. Hus MAC000168 har størst økning i MAPE med 29%, målingene for dette huset er spesielt vanskelig å forutse med mange og uregelmessige perioder med lavt strømbruk imellom perioder med tilsynelatende normal oppførsel.

Hus	MSE	RMSE	MAE	MAPE (%)	merknader
MAC000150	0.0066	0.0816	0.0458	43.23	
MAC000152	0.0092	0.0960	0.0494	66.18	
MAC000153	0.0154	0.1242	0.0712	47.67	
MAC000165	0.0105	0.1025	0.0611	29.01	
MAC000169	0.0052	0.0721	0.0360	38.20	
MAC000168	0.0070	0.0840	0.0314	89.55	Perioder med veldig lite strømbruk
MAC000159	0.0053	0.0730	0.0396	31.34	
MAC000173	0.0145	0.1208	0.0775	52.48	
MAC000179	0.0052	0.0726	0.0385	43.01	
MAC000181	0.0040	0.0638	0.0296	32.66	

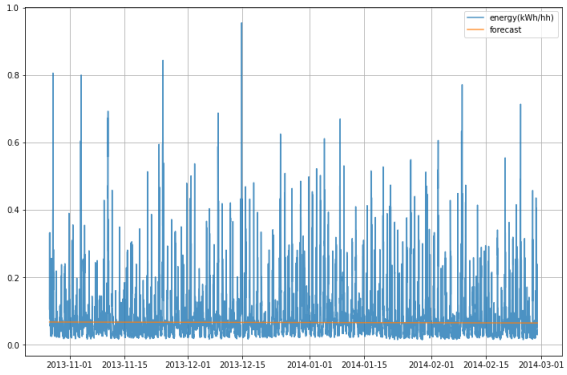
Tabell 5.2: resultater fra baseline modell (time)



Figur 5.2: Testdata for MAC000168

Siden tapsfunksjone for dette huset er litt mer følsomt, er det interessant å følge med på det når en evaluerer framtidige modeller.

5.2 Lineær regresjon



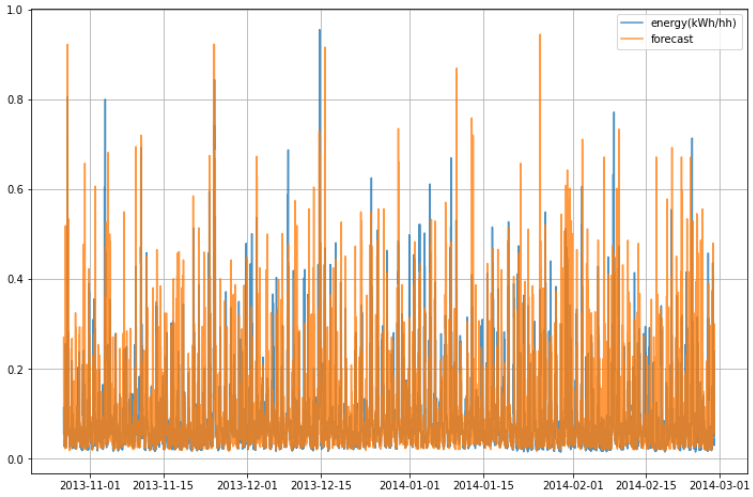
Figur 5.3: Linje for lineær regresjon

Hus	MSE	RMSE	MAE	MAPE (%)
MAC000150	0.0088	0.0939	0.0504	64.34
MAC000152	0.0107	0.1035	0.0581	137.45
MAC000153	0.0176	0.1328	0.0850	71.47
MAC000165	0.0202	0.1422	0.1102	96.98
MAC000169	0.0099	0.0996	0.0654	89.30
MAC000168	0.0080	0.0896	0.0430	171.44
MAC000159	0.0030	0.0556	0.0403	66.19
MAC000173	0.0131	0.1145	0.0762	69.95
MAC000179	0.0053	0.0730	0.0388	56.65
MAC000181	0.0030	0.0551	0.0324	84.04

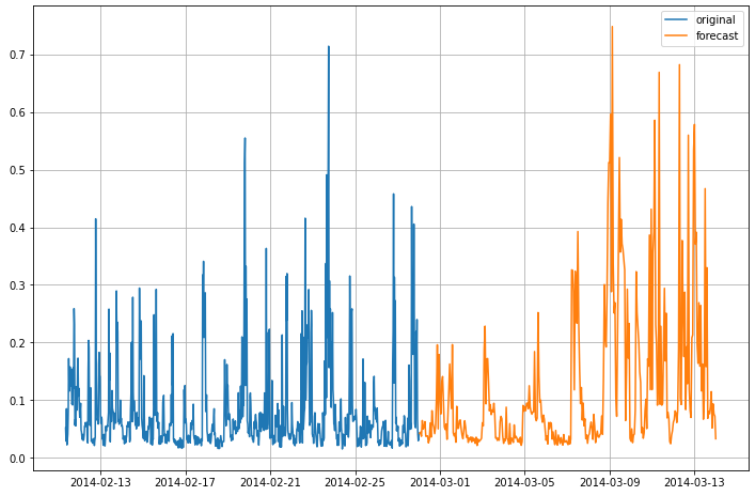
Tabell 5.3: resultater fra lineær regresjons modell (time)

En annen enkel modell er å tilpasse en linje på datasettet ved bruk av lineær regresjon med x aksen. Som en ser fra figur 5.3, blir denne linjen trukket ned siden det vanligvis ikke er mye strømforbruk. Denne modellen er farlig å bruke ettersom den alltid kommer til å gi en lavere verdi av strømforbruket enn det en i virkeligheten vil oppleve. Dette er et gjennomgående problem for alle modellene ettersom det er høydene i grafen som er vanskelig å spå. Med lineær regresjon vet en iallefall hva som er en dårlig modell, og hva det gjelder å slå.

5.3 Beslutningstre modell



Figur 5.4: Plot av spådommer på testdata



Figur 5.5: Prognose for en uke fram i tid

Spådommene er mer varierende enn for de andre modellene, og ved å bruke de forrige spådommene til å spå lengre ut i framtiden flater ikke beslutningstre modellen ut som de

andre modellene, se figur 5.5. Og vil fortsatt gi noe som ved øyemål ser ut som et organisk resultat.

Hus	MSE	RMSE	MAE	MAPE (%)
MAC000150	0.0110	0.1048	0.0557	73.52
MAC000152	0.0112	0.1060	0.0576	112.80
MAC000153	0.0083	0.0915	0.0583	53.16
MAC000165	0.0131	0.1146	0.0641	33.06
MAC000169	0.0101	0.1008	0.0525	71.69
MAC000168	0.0098	0.0991	0.0364	102.80
MAC000159	0.0039	0.0623	0.0328	40.83
MAC000173	0.0127	0.1131	0.0628	60.32
MAC000179	0.0066	0.0812	0.0399	60.72
MAC000181	0.0033	0.0578	0.0253	41.84

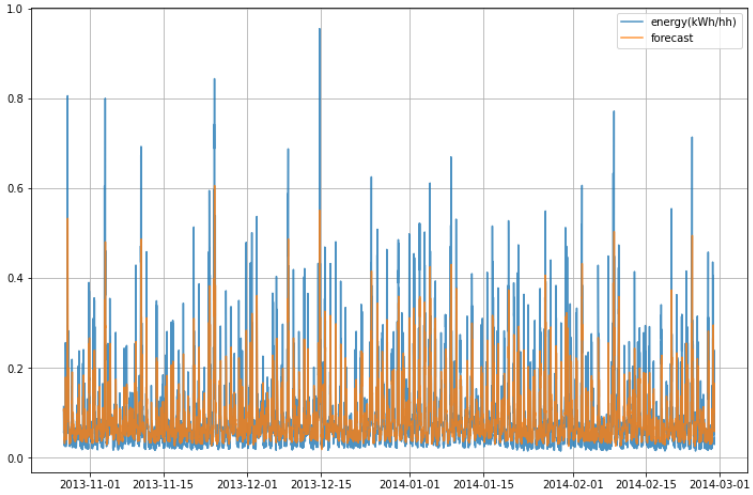
Tabell 5.4: resultater fra DT modell (halvtime)

Modellen blir mindre presis for trening og prognose for data hver time.

Hus	MSE	RMSE	MAE	MAPE (%)
MAC000150	0.0133	0.1155	0.0662	73.76
MAC000152	0.0149	0.1224	0.0697	110.45
MAC000153	0.0119	0.1093	0.0731	55.26
MAC000165	0.0130	0.1149	0.0703	34.61
MAC000169	0.0128	0.1132	0.0658	120.05
MAC000168	0.0101	0.1008	0.0388	123.39
MAC000159	0.0084	0.0918	0.0552	51.99
MAC000173	0.0159	0.1264	0.0773	51.92
MAC000179	0.0099	0.0996	0.0548	70.77
MAC000181	0.0066	0.0813	0.0379	47.76

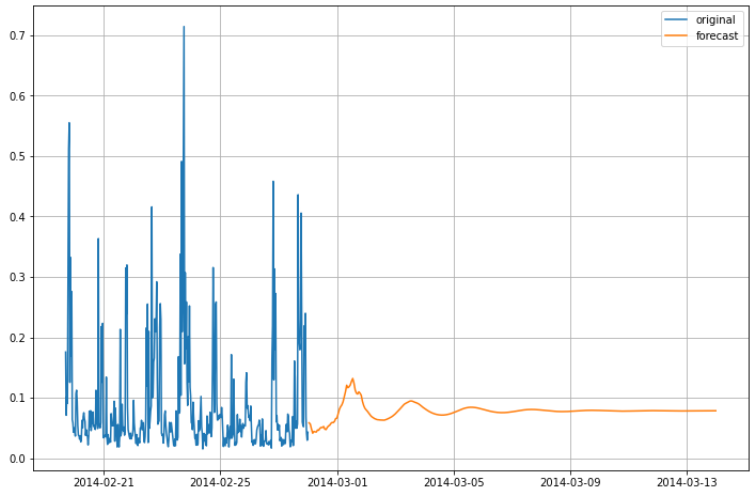
Tabell 5.5: resultater fra DT modell (time)

5.4 Perceptron (lineær modell)



Figur 5.6: Plot av spådommer på testdata

Modellen klarer ikke å få med seg målingstoppene.



Figur 5.7: Plot av spådommer på testdata en uke i framtiden

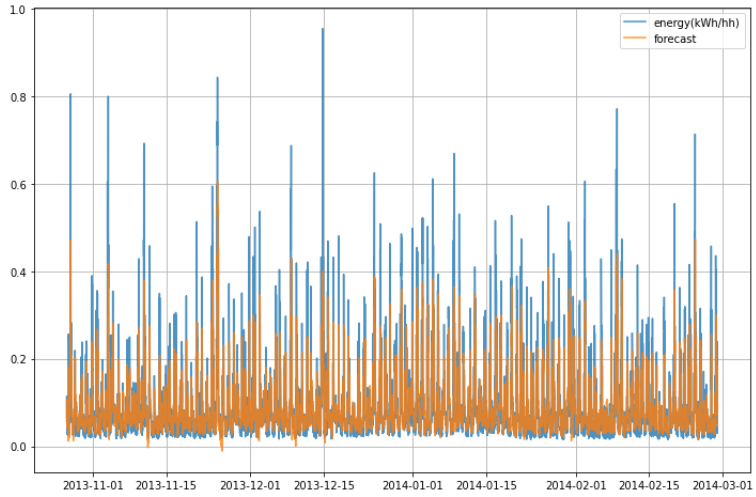
Målingene vil etterhvert flates ut.

Hus	MSE	RMSE	MAE	MAPE (%)
MAC000150	0.0046	0.0679	0.0366	45.99
MAC000152	0.0056	0.0753	0.0392	72.40
MAC000153	0.0055	0.0745	0.0496	44.12
MAC000165	0.0065	0.0777	0.0441	24.12
MAC000169	0.0039	0.0624	0.0339	48.05
MAC000168	0.0047	0.0691	0.0284	98.89
MAC000159	0.0022	0.0472	0.0251	31.15
MAC000173	0.0062	0.0792	0.0467	45.44
MAC000179	0.0027	0.0521	0.0286	50.09
MAC000181	0.0017	0.0416	0.0190	34.21

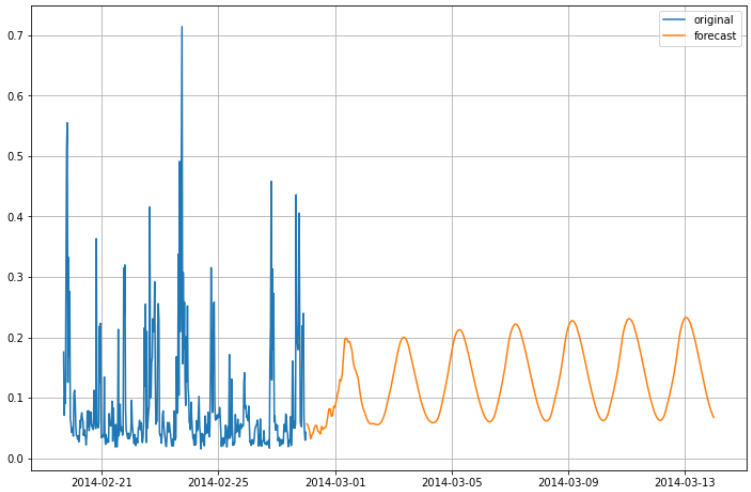
Tabell 5.6: resultater fra lineær modell (halvtime)

Denne modellen presterer bedre enn den enkle lineære regresjonen, og bedre enn den enkle modellen i mse/rmse, ettersom den er trent for å minimere denne. Selv om den også slår beslutningstre modellen så vil spådommene fort flates ut.

5.5 SLP



Figur 5.8: Plot av spådommer på testdata



Figur 5.9: Plot av spådommer på testdata en uke i framtiden

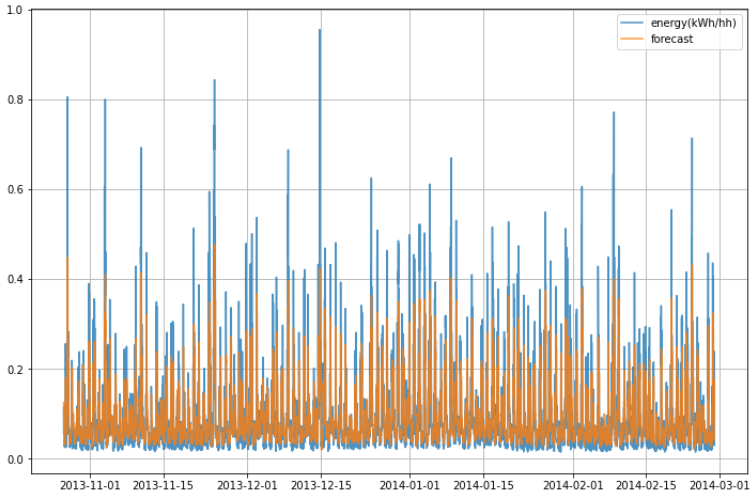
Hus	MSE	RMSE	MAE	MAPE (%)
MAC000150	0.0045	0.0673	0.0381	50.19
MAC000152	0.0057	0.0755	0.0411	84.69
MAC000153	0.0044	0.0665	0.0444	41.16
MAC000165	0.0056	0.0753	0.0553	26.64
MAC000169	0.0038	0.0616	0.0315	43.15
MAC000168	0.0048	0.0693	0.0296	103.88
MAC000159	0.0020	0.0457	0.0236	28.48
MAC000173	0.0059	0.0773	0.0456	46.15
MAC000179	0.0026	0.0515	0.0278	46.97
MAC000181	0.0018	0.0426	0.0206	34.79

Tabell 5.7: resultater fra SLP (halvtime)

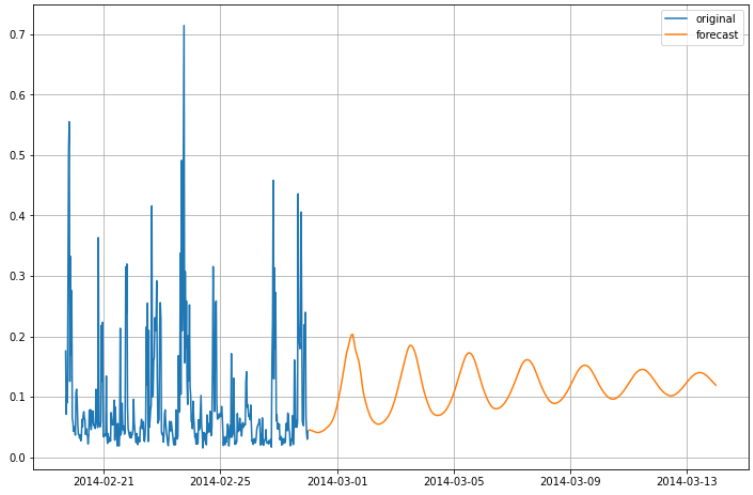
SLP gir tilsvarende resultater på testdata som for perceptronet, faktisk er perceptonet litt bedre ettersom det greide under 100% MAPE på MAC00168.

5.6 LSTM

Resultatene i tabell 5.8, er funnet ved 20 epoker og 50 nerveceller.



Figur 5.10: Plot av spådommer på testdata



Figur 5.11: Plot av spådommer på testdata

Hus	MSE	RMSE	MAE	MAPE (%)	merknader
MAC000150	0.0047	0.0687	0.0368	46.75	
MAC000152	0.0055	0.0746	0.0401	82.45	Støyfult data
MAC000153	0.0071	0.0848	0.0518	41.73	
MAC000165	0.0058	0.0763	0.0429	22.74	
MAC000169	0.0038	0.0620	0.0300	37.29	
MAC000168	0.0047	0.0687	0.0251	74.90	Perioder med veldig lite strømbruk
MAC000159	0.0021	0.0466	0.0226	24.50	
MAC000173	0.0061	0.0787	0.0458	44.34	
MAC000179	0.0020	0.0519	0.0287	52.56	
MAC000181	0.0017	0.0519	0.0175	25.34	

Tabell 5.8: resultater fra LSTM (halvtime)

5.7 Sammenligning

I tabell 5.9 har en regnet ut for hver modell forskjellen i resultatet i RMSE og MAPE med lineær regresjon. Hvor større negativ verdi, vist med blå bakgrunnsfarge, hvor mye bedre er modellen enn med enkel lineær regresjon. En positiv verdi (gult), betyr at regresjonsmodellen var bedre. Dette gjelder kun beslutningstre på enkelte hus derimot. Selv om dette er tilfellet anser vi beslutningstreet som mer relevant siden den har en evne til å spå topper i strømforbruk.

Modell	hus	RMSE forskjell	MAPE forskjell
Enkel modell	MAC000150	-0.0161	-21.16
Enkel modell	MAC000152	-0.0202	-78.19
Enkel modell	MAC000153	-0.0440	-31.20
Enkel modell	MAC000165	-0.0543	-74.43
Enkel modell	MAC000169	-0.0329	-51.70
Enkel modell	MAC000168	-0.0117	-110.85
Enkel modell	MAC000159	-0.0016	-37.63
Enkel modell	MAC000173	-0.0195	-22.25
Enkel modell	MAC000179	-0.0154	-11.57
Enkel modell	MAC000181	-0.0098	-57.97
Perceptron	MAC000150	-0.0260	-18.35
Perceptron	MAC000152	-0.0282	-65.05
Perceptron	MAC000153	-0.0583	-27.35
Perceptron	MAC000165	-0.0645	-72.86
Perceptron	MAC000169	-0.0372	-41.25
Perceptron	MAC000168	-0.0205	-72.55
Perceptron	MAC000159	-0.0084	-35.04
Perceptron	MAC000173	-0.0353	-24.51
Perceptron	MAC000179	-0.0209	-6.56
Perceptron	MAC000181	-0.0135	-49.83
DT	MAC000150	0.0109	9.18
DT	MAC000152	0.0025	-24.65
DT	MAC000153	-0.0413	-18.31
DT	MAC000165	-0.0276	-63.92
DT	MAC000169	0.0012	-17.61
DT	MAC000168	0.0095	-68.64
DT	MAC000159	0.0067	-25.36
DT	MAC000173	-0.0014	-9.63
DT	MAC000179	0.0082	4.07
DT	MAC000181	0.0027	-42.20
SLP	MAC000150	-0.0266	-14.15
SLP	MAC000152	-0.0280	-52.76
SLP	MAC000153	-0.0663	-30.31
SLP	MAC000165	-0.0669	-70.34
SLP	MAC000169	-0.0380	-46.15
SLP	MAC000168	-0.0203	-67.56
SLP	MAC000159	-0.0099	-37.71
SLP	MAC000173	-0.0372	-23.80
SLP	MAC000179	-0.0215	-9.68
SLP	MAC000181	-0.0125	-49.25
LSTM	MAC000150	-0.0252	-17.59
LSTM	MAC000152	-0.0289	-55.00
LSTM	MAC000153	-0.0480	-29.74
LSTM	MAC000165	-0.0659	-74.24
LSTM	MAC000169	-0.0376	-52.01
LSTM	MAC000168	-0.0209	-96.54
LSTM	MAC000159	-0.0090	-41.69
LSTM	MAC000173	-0.0358	-25.61
LSTM	MAC000179	-0.0211	-4.09
LSTM	MAC000181	-0.0032	-58.70

Tabell 5.9: Forskjell fra regresjonsmodell

En videre sammenligning mellom akkurat SLP og LSTM virker nødvendig. I RMSE er SLP stort sett best, mens LSTM gir noe bedre MAPE. LSTM oppfører seg bedre på MAC000168.

LSTM	MAC000150	0.0014	-3.44
LSTM	MAC000152	-0.0009	-2.24
LSTM	MAC000153	0.0183	0.57
LSTM	MAC000165	0.0010	-3.90
LSTM	MAC000169	0.0004	-5.86
LSTM	MAC000168	-0.0006	-28.98
LSTM	MAC000159	0.0009	-3.98
LSTM	MAC000173	0.0014	-1.81
LSTM	MAC000179	0.0004	5.59
LSTM	MAC000181	0.0093	-9.45

Tabell 5.10: Forskjell fra SLP modell

5.8 Bruk av værdata

For husene som ved bruk av pearsons R har vist litt korrelasjon med temperaturen har en også sett på modeller med bruk av temperatur data i inngangsvektoren. Likedan som med energimålingene er 24 timer med temperaturmålinger tatt med. Vi har kun værdata i time format, dermed spår modellene for hver time i første omgang. For å unngå negative verdier ble tok en kvadratet av temperaturmålingen, og så normaliserte de mellom 0 og 1. Huset MAC000153 har dobbel så stor korrelasjon som de andre, og en forventer å se at temperaturen påvirker dette huset.

MAC000153 med beslutningstre modellen er 1.4% (55.26-53.87) bedre i MAPE, men ellers tilsvarende for modellen uten temperatur data. For MAC000168 har modellen forverret seg, men MAC000169 har blitt betydelig forbedret med 34.5%, nå er det verdt å nevne at DT er meget dårlig på MAC000169 sammenlignet med andre modeller. MAC000173 har forbedret seg med 0.40% som ikke er spesielt betydelig, noe som er forventet ettersom denne har lavest korrelasjon.

Hus	korrelasjon	mse	rmse	mae	mape
MAC000153	-0.42	0.0119	0.1093	0.0730	53.87
MAC000168	-0.19	0.0105	0.1026	0.0401	127.98
MAC000169	-0.18	0.0990	0.0584	0.0584	85.52
MAC000173	-0.12	0.1267	0.0779	0.0779	51.52

Tabell 5.11: resultater fra DT

MAC000153 har forverret seg med 1.18%, MAC000168 med 11.26%, og MAC000169 med 9.86%. MAC000173 har forbedret seg med 5.49%.

Hus	korrelasjon	mse	rmse	mae	mape
MAC000153	-0.42	0.0084	0.0920	0.0655	48.83
MAC000168	-0.19	0.0052	0.0723	0.0364	149.30
MAC000169	-0.18	0.0050	0.0710	0.0447	58.89
MAC000173	-0.12	0.0101	0.1005	0.0647	42.69

Tabell 5.12: resultater fra Perceptron

MAC000153 har forverret seg med 9.00%, MAC000168 med 9.52%, og MAC000169 med 3.30%. MAC000173 har forbedret seg med 0.48%.

Hus	korrelasjon	mse	rmse	mae	mape
MAC000153	-0.42	0.0064	0.0802	0.0571	46.16
MAC000168	-0.19	0.0049	0.0704	0.0325	113.63
MAC000169	-0.18	0.0044	0.0667	0.0408	54.84
MAC000173	-0.12	0.0091	0.0957	0.0605	39.60

Tabell 5.13: resultater fra SLP med 30 enheter

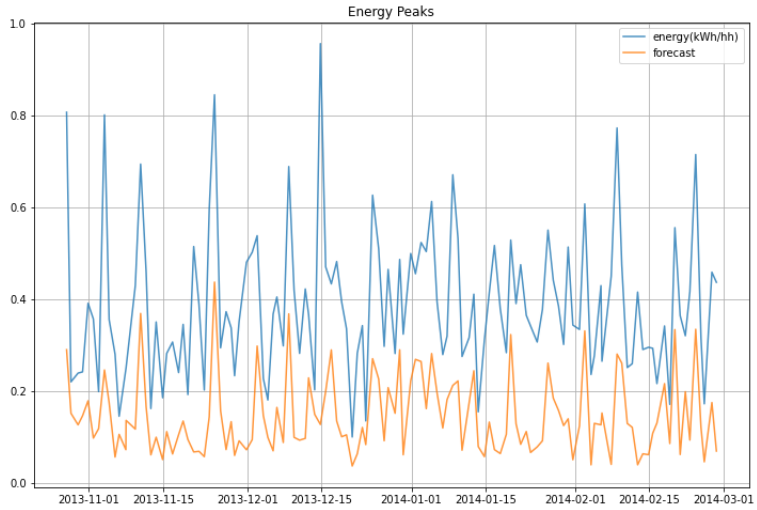
MAC000153 har forverret seg med 24.94%, MAC000168 med 115.99%, MAC000169 med 44.14%, og MAC000173 med 1.69%. LSTM modellen har størst avvik.

Hus	korrelasjon	mse	rmse	mae	mape
MAC000153	-0.42	0.0206	0.1438	0.0948	66.67
MAC000168	-0.19	0.0065	0.0808	0.0436	190.89
MAC000169	-0.18	0.0066	0.0816	0.0555	81.43
MAC000173	-0.12	0.0158	0.1260	0.0815	46.03

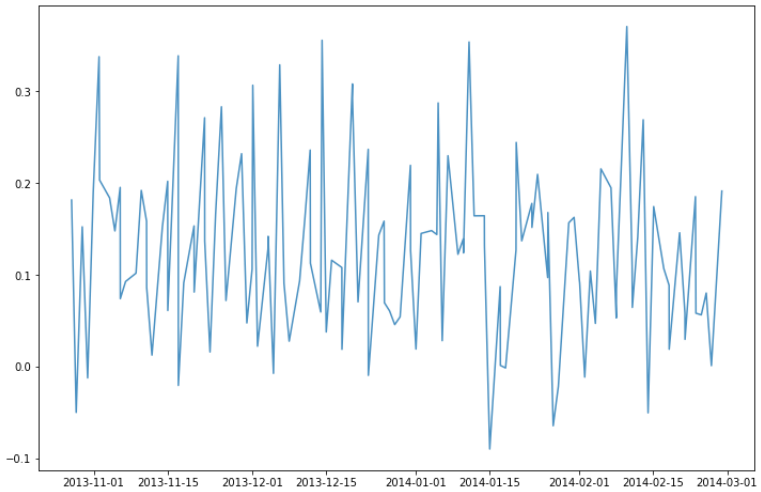
Tabell 5.14: resultater fra LSTM et lag med 50 enheter

Utifra denne testingen, er det unødvendig å ta i bruk data om temperatur slik vi har gjort. MAC000173 er derimot huset som er minst påvirket av temperatur dataen som gir mening ettersom det har minst korrelasjon, de andre husene blir dessverre negativt påvirket. Det er mulig det finnes andre måter å formatere dataen og/eller modellene på som gjør det mer betydningsfullt.

5.9 Evnen til å spå daglige topper



Figur 5.12: Plot av topper, spådommer i oransj



Figur 5.13: Plot av forskjellen mellom toppene og prognosen

Fra figur 5.13, kan en se at en høyere topp vil tilsvare et større avvik mellom prediksjon og måling. vil det i tilsvarende grad være større forskjell mellom spådom og måling.

For energitoppene hver dag, regnet en ut MAPE på disse og sammenlignet med MAPE funnet på hele testdataen. En negativ differanse betyr at modellen er bedre på toppene enn for dataen totalt sett. Prognosen på toppene ligger dessverre fortsatt lavere enn strømmålingene. Som antatt er det regresjonen som er dårligst til å få med seg de daglige toppene, mens DT konkurrer omtrent like godt LSTM og SLP, med det desidert beste resultatet på MAC000153. LSTM fikk et ukarakteristisk dårlig resultat på akkurat dette huset, men presterer omtrent jevngod med SLP.

Modell	hus	Topper MAPE	MAPE	differanse
Enkel modell	MAC000150	55.81	43.19	12.62
Enkel modell	MAC000152	55.31	59.27	-3.96
Enkel modell	MAC000153	36.19	40.27	-4.08
Enkel modell	MAC000165	38.08	22.55	15.53
Enkel modell	MAC000169	41.38	37.60	3.78
Enkel modell	MAC000168	43.32	60.59	-17.27
Enkel modell	MAC000159	57.32	28.57	28.75
Enkel modell	MAC000173	52.67	47.70	4.97
Enkel modell	MAC000179	51.67	45.09	6.58
Enkel modell	MAC000181	49.36	26.07	23.29
Regresjon	MAC000150	80.61	64.34	16.27
Regresjon	MAC000152	84.02	137.45	-53.43
Regresjon	MAC000153	75.51	71.47	4.04
Regresjon	MAC000165	67.63	96.98	-29.35
Regresjon	MAC000169	76.67	89.30	-12.63
Regresjon	MAC000168	122.52	171.44	-48.92
Regresjon	MAC000159	64.49	66,20	-1.71
Regresjon	MAC000173	79.51	69.95	9.56
Regresjon	MAC000179	76.20	56.65	19.55
Regresjon	MAC000181	66.31	84.04	-17.73
DT	MAC000150	59.24	73.70	-14.46
DT	MAC000152	59.90	113.46	-53.56
DT	MAC000153	19.33	53.15	-33.82
DT	MAC000165	41.99	33.57	8.42
DT	MAC000169	52.44	73.23	-20.79
DT	MAC000168	51.23	100.92	-49.69
DT	MAC000159	58.18	39.99	18.19
DT	MAC000173	57.52	59.93	-2.41
DT	MAC000179	60.40	62.45	-2.05
DT	MAC000181	63,77	41.71	22.06
SLP	MAC000150	59.49	47.47	12.02
SLP	MAC000152	61.76	73.69	-11.93
SLP	MAC000153	23.84	41.37	-17.53
SLP	MAC000165	38.73	24.98	13.75
SLP	MAC000169	45.03	42.64	2.39
SLP	MAC000168	51.99	98.93	-46.96
SLP	MAC000159	64.17	28.13	36.04
SLP	MAC000173	55.96	45.94	10.02
SLP	MAC000179	57.09	42.37	14.72
SLP	MAC000181	56.54	32.88	23.66
LSTM	MAC000150	60.24	51.56	8.68
LSTM	MAC000152	59.28	74.37	-15.09
LSTM	MAC000153	46.35	42.77	3.58
LSTM	MAC000165	39.40	23.58	15.82
LSTM	MAC000169	48.33	36.56	11.77
LSTM	MAC000168	55.03	63.75	-8.72
LSTM	MAC000159	48.33	36.56	11.77
LSTM	MAC000173	60.15	44.83	15.32
LSTM	MAC000179	57.46	50.96	6.50
LSTM	MAC000181	53.10	30.31	22.79

Tabell 5.15: Resultater for daglige topper for hver modell

Kapittel 6

Sammendrag og Konklusjon

6.1 Sammendrag

Prosjektet bruker modeller for prediksjoner av strømforbruket til hushold i London. Det finnes mange algoritmer som en kan bruke i maskinlærings modeller, de vi har valgt å se nærmere på er LSTM, Perceptron, SLP, MLP, beslutningstre og lineær regresjon. Bruk av forskjellige modeller gjør det enklere å sammenligne prediksjoner, og finne ut hvilke som er mest nøyaktig. Målingene av strømforbruk i oppgaven kommer fra smartmetere i london (5566 husstander), mellom 2011 og 2014. Målingene er gjort hver halvtime og måles i kilowattimer. Husene er delt inn i forskjellige demografiske kategorier og grupper, og i dette prosjektet har vi valgt hus i gruppen Career Climbers, som utgjorde den største andelen av husstandene.

Maskinlæring delen av prosjektet er skrevet i python og benytter seg av Tensorflow og Scikit-learn bibliotekene. Python er et populært og moderne programmeringsspråk som er brukervennlig. Python er spesielt mye brukt i maskinlæring prosjekter, da de ofte er iterative.

En webapplikasjon er en applikasjon som kjøres i nettlesere hos brukere, og leveres av servere hvor koden er lastet opp. Python ble også benyttet for å skrive backenden til web applikasjonen, det ga kort vei fra modellene til bruk i applikasjonen. Mer spesifikt ble det skrevet med Flask, som er et mikro-rammeverk for webapplikasjoner i python. Server kommuniserer med klient via API, som er en måte å sende informasjon mellom backend og frontend. Frontend ble skrevet i React.js med html og css. React er et verktøy brukt for å lage brukergrensesnitt og er ikke et rammeverk, men er et JavaScript-bibliotek. For lagring av informasjon ble det skrevet en database basert på SQLite.

6.2 Konklusjon

Av maskinlæringsmodellene er ikke beslutningstreet spesielt treffsikkert, men presterer likevel bra på de daglige toppene. Det er lite å skille mellom resultatene for SLP og LSTM, og hvilke en bruker kan avhenge av husstanden, likevel vurderer vi LSTM litt høyere ettersom denne presterte bedre på det mer uregelmessige huset MAC000168. Med tanke på trening derimot tar LSTM mye lenger tid enn SLP, så for effektivitetens skyld skårer SLP høyere. Ettersom kortsiktig punktprediksjon er en enkel situasjon for maskinlæring, er det ikke store marginer å finne i resultatene. Vi fant ikke noen økt presisjon med å bygge mer avanserte flerlagsmodeller både for MLP og LSTM. Bruk av temperatur ga heller ikke bedre spådommer, men det er rom for å jobbe med preprosessering av data slik at det likevel kan gi utslag.

Webapplikasjonen gir et overblikk over tilgjengelige data og gjør endring av parametere enklere. Kun det mest nødvendige er tatt med her, slik at det skal kunne gå kort tid mellom interesse til resultat. Dette åpner opp for at brukeren ikke skal trenge en dypere forståelse for hvordan modellene virker for å kunne kjøre dem. Valg av data med visuelle input gjør det også enkelt for brukere å forstå forskjellene i datasett og velge hva som er mest aktuelt å benytte seg av.

6.3 Utfordringer

Prosjektet ble laget uten bruk av strukturerte tester som førte til en god del debugging for å finne ut av hva som feilet. Ettersom dette var vårt første møte med tensorflow og machine learning ble store deler av tiden brukt for å forsøke å forstå de tilgjengelige verktøyene. Med bedre grunnlag ville en god del av kode-utfordringene vi møtte på kunnet blitt løst raskere, da vi kunne hatt flere referanser å støtte oss på.

En stor utfordring var å forsøke å forbedre normalisering under preprosessering. Dette ble forsøkt for å gjøre modellene bedre. Dette blir særlig relevant med tanke på å trekke inn data med en viss korrelasjon som temperaturen.

Som del av studieløpet på bachelor i datateknologi har vi hatt faget DAT310, Webprogrammering, hvor det ble skrevet applikasjoner med Vue.js, som er lignende på React.js. Det er derimot mye av den forenklede strukturen i Vue-prosjekt som vi lette etter i React som ikke finnes der. Det var mye kode i frontend vi først skrev på tungvinte og ineffektive måter før vi etterhvert lærte hvordan det er mest hensiktsmessig å løse i React.

6.4 Videreutvikling

Tester ville ha gjort det mye enklere å debugge og teste koden og ville ha spart oss en god del tid. Testene vil også hjelpe med tanke på videre utvikling av modellene eller dashboardet ved at det blir enklere å se hva som feiler. Testene kan skrives ved hjelp av unittest biblioteket til python.

Modifisering av datasettet i dashboardet er noe vi tenkte å få implementert, men ville ha tatt alt for lang tid. For at dette skal kunne fungere kreves det funksjoner for oversetting av data i lister opp mot hverandre. En mulighet her kunne vært å bruke OrderedDict fra python biblioteket collections. Da kunne en hatt en mer håndterlig måte å bearbeide målingene til hvert hus.

Det er mulighet for flere funksjoner og mer visualisering i dashboardet sånn at brukere med ingen bakgrunns erfaring vil kunne leke seg rundt og forstå dataen bedre. Dette krever bedre planlegging, slik at mengden funksjoner ikke gjør opplevelsen mer forvirrende for brukeren. Vi gjorde en avveining sett opp mot tilgjengelig tid, og bestemte oss for at det ikke ville la seg gjøre på en gjennomført måte. Det er skrevet langt flere funksjoner i backend enn hva som er tilgjengelig per nå på klient siden. En implementasjon for bruk av disse er dermed mulig å gjennomføre i fremtiden. Det er blant annet funksjonalitet for å bytte navn på prosjekttene i backend. Databasen er satt opp for å kunne bli endret og oppdatert, det som mangler er å skrive koden for å tilgjengeliggjøre disse funksjonene for brukerne i frontend.

Vi har brukt en SQlite database som ikke er velegnet for produksjon ettersom den ikke tillater flerbruk skriving. SQlite fungerer også dårlig med store datasett siden databasen er begrenset til 281 terrabytes, SQLite lagrer hele databasen i en disk fil og mange filsystemer har en mindre maksimal grense for filer enn dette. Andre backend databaser som kan bli brukt istedet kan være postgres.

Slik applikasjonen er skrevet nå er den skrevet med Engelsk som språk, dette ble gjort for at den skal kunne forstås av et større publikum. En mulig videreutvikling ville vært å oversette siden til Norsk, eller integrert en oversetter slik at en ikke trenger gode engelskkunnskaper for å kunne bruke applikajsonen.

Bibliografi

[1] Acorn - understanding consumers and communities. https://acorn.caci.co.uk/what-is-acorn#acc-1.

[2] Decision tree. https://www.geeksforgeeks.org/decision-tree/.

[3] interpolasjon (matematikk). https://snl.no/interpolasjon__matematikk.

[4] Tf keras documentation. https://www.tensorflow.org/api_docs/python/tf/keras.

[5] Uk power networks. https://www.ukpowernetworks.co.uk/.

[6] 3Blue1Brown. But what is a neural network? | chapter 1, deep learning. https://www.3blue1brown.com/lessons/neural-networks.

[7] Eligijus Bujokas. Energy consumption time series forecasting with python and lstm deep learning model. https://towardsdatascience.com/energy-consumption-time-series-forecasting-with-python-and-lstm-deep-learning-model-7952e2f9a796, 2020.

[8] Jean-Michel D. Smart meters in london. https://www.kaggle.com/jeanmidev/smwart-meters-in-london.

[9] Google. Google fonts icons. https://fonts.google.com/icons.

[10] gregn. Reacttoprint. https://www.npmjs.com/package/react-to-print.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] Plotly. Plotly javascript. https://plotly.com/javascript/.

Vedlegg A

Programlisting

Github: https://github.com/SindreMohr/Bachelor_thesis