Bacheloroppgave

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

Simen Refsland
Oscar Urfjell Brøndelsbo

Prediksjon av drivstoffpriser

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Tom Røise

Mai 2022

NTNU
Kunnskap for en bedre verden

Simen Refsland
Oscar Urfjell Brøndelsbo

Prediksjon av drivstoffpriser

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Tom Røise
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Abstract

DrivstoffAppen is an app used to check fuel prices on gas stations i Norway, where the purpose is to find the cheapest petrol station. In connection with this, the owners of DrivstoffAppen wanted to examine whether it is possible to predict future gas prices, so that users of the app can refuel at the cheapest point in time. The group has created a solution that uses a machine learning model, which uses a data set consisting of historical fuel prices. The final machine learning model can predict prices at all petrol station in Norway every day where good data is available. In addition an API has been developed to send the predictions to and from the database on behalf of the parts in the system, and expanded the existing Android application to visualize predictions in tabular form, as well as graphically.

Sammendrag

DrivstoffAppen er en app som brukes til å sjekke drivstoffpriser på bensinstasjoner i Norge, hvor formålet er å finne billigste stasjon. I forbindelse med dette ville eierne i Drivstoffappen undersøke om det er mulig å forutse fremtidige drivstoffpriser, slik at forbrukerene kan fylle drivstoff når det er billigst. Gruppen har laget en løsning som benytter seg av en maskinlæringsmodell, som bruker et datasett bestående av historiske drivstoffpriser. Den endelige maskinlæringsmodellen kan produsere predikerte priser på alle bensinstasjoner i Norge hver dag der det foreligger god data. I tillegg har det har blitt utviklet et API for å sende prediksjonene til og fra databasen for partene i systemet, og utvidet den eksisterende Android-applikasjonen til å visualisere prediksjoner i tabellform og grafisk.

Forord

Denne bacheloroppgaven er skrevet av Simen Refsland og Oscar Urfjell Brøndelsbo, som er studenter i det avsluttende semesteret i dataingeniørutdanningen ved NTNU i Gjøvik.

Vi ønsker å takke personene som har hjulpet oss gjennom oppgaven. Først vil vi gi en spesiell takk til Vegard Ølstad Dalberg, tidligere kontaktperson i Headit, som har brukt deler av sin fritid på å veilede oss gjennom oppgaven, og har vært til stor hjelp under utvikling av maskinlæringsmodell. Videre vil vi takke kontaktperson i Headit, Martin Holthe Rønningen, som har vært til god hjelp gjennom hele prosjektperioden, og sørget for jevnlige statusoppdateringer og fremdrift. Vi vil også takke Syver Orhagen og Sander Helgesen i DrivstoffAppen, som har gitt oss en spennende oppgave og mulighet.

Til slutt vil vi takke Tom Røise, som har vært vår veileder gjennom prosjektet. Tom har gitt oss gode innspill, og vært til god hjelp både gjennom prosjektfasen generelt og under rapportskrivingen.

Innhold

Figurer

Tabeller

Kodelister

Akronymer

API Application Programming Interface.

CI/CD Continous Integration / Continuous Deployment.

CNN Konvolusjonelle nevrale nettverk.

CRUD Create, Read, Update, Delete.

GPU Graphical Processing Unit (norsk: skjermkort).

HTTP Hypertext Transfer Protocol.

JSON JavaScript Object Notation.

LSTM Long short-term memory (norsk: lang korttidsminne).

MAE Mean Absolute Error.

MAPE Mean Absolute Percentage Error.

ML Maskinlæring.

MSE Mean Squared Error.

MVP Minimum Viable Product (norsk: minimumløsning).

ReLU Rectified linear unit.

REST Representational State Transfer.

RNN Rekurrente nevrale nettverk.

Ordliste

dependency injection Designmønster hvor et objekt mottar objekter det er avhengig av, i stedet for å konstruere disse selv.

feature engineering Teknikk i maskinlæring for å generere nye variabler basert på ekstern kunnskap og/eller metoder.

hyperparameter Parameter som ikke blir bestemt av maskinlæringmodellens trening. Kan være læringsrate, antall skjulte lag, antall noder etc.

overtilpasning Fenomen hvor maskinlæringsmodellens tilpasning til treningsdataen er god, men har høy generaliseringsfeil for ny data som modellen ikke har sett.

pipeline En serie automatiserte prosesser som utføres for å bygge, teste og/eller levere programvare.

sekvensmodellering Modeller som bruker sekvenser av input eller gir en sekvens som output.

Kapittel 1

Introduksjon

1.1 Bakgrunn

I det avsluttende semesteret av bachelor i ingeniørfag, data, har det blitt gjennomført en bacheloroppgave som har blitt utført på vegne av en oppdragsgiver, som i vårt tilfelle er den eksterne bedriften Headit. Headit er en bedrift med tverrfaglig kompetanse innen UX, data science, forretnings- og systemutvikling.[1]

Med dagens drivstoffpriser har det blitt viktig for mange å følge med på drivstoffprisene og fylle når eller hvor det er billig. Headit har inngått et samarbeid med utviklerne av DrivstoffAppen.[2] DrivstoffAppen er en app hvor brukere kan legge inn, oppdatere og bekrefte drivstoffpriser på bensinstasjoner rundt om i Norge. Siden oppdateringene på drivstoffpriser kommer fra brukernes observasjoner, er det ikke alltid gitt at alle prisene er oppdaterte. DrivstoffAppen ønsker derfor å se på en løsning der sannsynlig predikert pris blir beregnet ved hjelp av en maskinlæringsmodell og hvor resultatet av prediksjonen blir plottet inn i en fullscreen-kart som gir en visuell fremstilling til brukeren.

Formålet med denne oppgaven er å gi innsikt og erfaring i identifisere og løse problemer i et ingeniørfaglig relevant område, samt å kunne planlegge, gjennomføre og dokumentere prosjektet på en god måte.

1.2 Prosjektbeskrivelse

I korte trekk har oppgaven gått ut på å utvide mobilapplikasjonen DrivstoffAppen. I hovedsak har det vært et ønske om å utvikle (om mulig) og implementere en maskinlæringsmodell mot DrivstoffAppens eksisterende database, med tilsvarende frontend-komponent for visualisering av modellens bensinprediksjoner. Studentene har stått fritt til å velge hvordan prediksjonene visualiseres og hvordan påfølgende «features» skulle se ut.

[1] Headit: `https://headit.no/hjem`

[2] DrivstoffAppen: `https://drivstoffappen.no`

Hovedfokuset i oppgaven har vært på maskinlæring og å lage en modell som gir gode prediksjoner på drivstoffpris, og som er meningsfulle. Dette har innebært å se på data som brukerne har rapportert inn, eventuelt se på trender i oljepris og andre mulige faktorer som har spiller inn.

For å se orginal oppgavebeskrivelse, henvises det til vedlegg B.

1.3 Mål og rammer

Prosjektmål

Hovedmålet med prosjektet er å implementere en løsning hvor DrivstoffAppens innrapporterte priser fra brukere blir supplert med predikerte priser fra maskinlæringsmodeller.

Resultatmål

Det resulterende produktet skal ha en maskinlæringsmodell som henter data fra DrivstoffAppens database og predikerer priser for forskjellige stasjoner og deretter legger disse inn i en prediksjonstabell. Maskinlæringsmodellen må kunne gi brukeren en prediksjon som har tilstrekkelig nøyaktighet, slik at den gir meningsfull informasjon til brukerne.

Grensesnittet mellom DrivstoffAppen og maskinlæringen sine prediksjoner skal være definert gjennom et REST API som henter data fra databasen og sender prediksjoner gjennom HTTP protokollen. Løsningen skal ha et frontend-komponent som visualiserer prediksjonene i et fullscreen-kart for hver bensinstasjon.

Effektmål

Ved å supplere innrapporterte priser for drivstoff med predikerte priser, vil brukeren av applikasjon ha et bredere grunnlag for å avgjøre når det er billigst å fylle drivstoff, og kan planlegge deretter. Den ønskede effekten er derfor at brukeren har mindre utgifter til drivstoff enn det som er realiteten den dag i dag.

Rammer

Den generelle tidsrammen har vært fra 11. januar til 20. mai. Bacheloroppgaven skulle være levert på Inspera innen 20. mai kl. 12:00, hvor løsning skulle være ferdig utviklet og rapport ferdig skrevet. Det er ikke medregnet noen økonomiske kostnader knyttet direkte til prosjektgruppen. DrivstoffAppen har tilgang til en eksisterende infrastruktur som tilbyr skytjenester. I tillegg tilbyr en del skyleverandører en «free tier», altså en gratisplan som eventuelt kunne brukes.

1.4 Omfang

Fagområde

I drivstoffmarkedet er det mange faktorer som spiller inn prisnivået for drivstoff. Det mest åpenbare er prisen på råolje, som har en relativt sterk korrelasjon med drivstoffprisen. I handel av råolje og resulterende produkter blir amerikanske dollar ofte brukt. Av denne grunn vil kronekursen også gi en påvirkning på prisnivået ved at en svakere krone mot dollaren vil føre til høyere priser. I tillegg kommer diverse avgifter, som veibruksavgifter og CO2-avgifter også til å spille en rolle [1]. Lokale konkurranseforhold mellom bensinstasjoner kan også ha betydning på prisen.

I tillegg må det nevnes det at korrelasjonen mellom oljepris og drivstoffpris ikke nødvendigvis er positiv. For sommeren 2015 fant Statistisk sentralbyrå at drivstoffprisene økte med 6 prosent fra mai til juni og 1,6 prosent fra juni til juli, mens råoljeprisen falt med nesten 6 prosent i samme periode [2]. Foruten dette spiller mer komplekse sammenhenger inn som er vanskelige å kvantifisere eller oppdage. Her kan kunstig intelligens potensielt spille en rolle.

Avgrensning

API-et som har blitt utviklet vil i utgangspunktet ikke være tilgjengelig for allmennheten, derfor var det viktig å sikre at tilgangen blir begrenset til de korrekte partene. Implementasjonen av frontend omfatter kun Android-versjonen av applikasjonen. Grunnet et kommende visuell overhaling av applikasjonen som ikke har blitt påbegynt i skrivende stund, vil utvikling av brukergrensesnittet ikke bli lansert til offentligheten. Det vil med andre ord bare bli utviklet for oppgavens skyld, eventuelt være til inspirasjon for DrivstoffAppen.

1.5 Terminologi

Det er tatt utgangspunkt i at leseren har en grunnleggende innsikt i informatikk og matematikk, og ordlister og akronymer reflekterer i stor grad dette. Fagområdet oppgaven omfatter har en klar preferanse for bruk av mange engelske ord og utrykk, og gruppen har i den grad det er hensiktsmessig prøvd å oversette en del av disse. Unntaksvis brukes engelske uttrykk hvor det ikke er en passende oversettelse, eller hvor det engelske uttrykket dominerer selv på norsk.

1.6 Gruppens bakgrunn

Begge medlemmene går siste år på dataingeniørutdanningen ved NTNU i Gjøvik. Utdanningsløpet har i stor grad vært likt, og omfatter her emner som omhandler statistikk, matematikk, systemutvikling, skyteknologier og mobilprogrammering. I tillegg har et av gruppemedlemmene tatt videre fordypning i matematikk som

valgemne, hvor lineær algebra kunne være av særlig relevans i forbindelse med maskinlæring.

På grunn av dette hadde gruppen en forutsetning for å kunne utvikle REST API og Android-applikasjoner. I maskinlæring hadde gruppen i stor grad kun fått en grunnleggende innføring i kunstig intelligens og maskinlæring. Med bakgrunn i matematikk, var det derimot gode forutsetninger for å tilegne seg kunnskaper knyttet til feltet, som bruker statistiske metoder i stor grad.

Hovedtyngden av det som gruppen måtte øke kunnskapen sin om ble med andre ord maskinlæring, med særlig vekt på nevrale nettverk, og andre nettverk som er avledet fra dette. En betydelig del av tidsbruken har gått til å sette seg inn i og lære om forskjellige arkitekturer innen maskinlæring.

1.7 Prosjektorganisering

På grunn av gruppens størrelse og oppgavens art, har det følgelig blitt mye overlapp i forbindelse med ansvarsområder og arbeidsoppgaver. Likevel har det blitt tildelt overordnede ansvar innen utvikling og administrativt.

Ansvarsforhold og rutiner

Administrativt

- Prosjektleder: Oscar Urfjell Brøndelsbo
- Prosjektleders ansvarsforhold:
    - Prosjektgjennomføring
    - Fordeling av oppgaver og rammer
    - Planlegging av sprinter
    - Kvalitetssikring av prosess
- Dokument-ansvarlig: Simen Refsland
- Dokument-ansvarlig ansvarsforhold:
    - Møteinkallinger
    - Møtereferater
    - Dokumentasjon (kode, rapport o.l.)

Innen utvikling

I utviklingsdelen har begge gruppemedlemmene jobbet med det samme i relativt stor grad. Ansvaret har derfor i større grad vært å tilegne seg kunnskap rundt hver del og å kunne formidle denne videre.

- Maskinlæring- og backendansvarlig: Simen Refsland
- Frontendansvarlig: Oscar Urfjell Brøndelsbo

1.8 Rapportstruktur

Kapittel 1 - Introduksjon beskriver bakgrunn og målet med prosjektet, samt litt om gruppens bakgrunn og organisering.

Kapittel 2 - Teori går gjennom relevant teori, særlig i forbindelse med maskinlæring. Det forutsettes at leseren har en viss innsikt innen informatikk og matematikk.

Kapittel 3 - Teknologier beskriver hva slags teknologier som har blitt brukt i forbindelse med implementasjon av prosjekt.

Kapittel 4 - Kravspesifikasjon går gjennom kravene til de forskjellige kravene som stilles til Androidapplikasjon, API og maskinlæringsmodul, samt bruksmønsterdiagram.

Kapittel 5 - Utviklingsprosess går gjennom gruppens prosess rundt prosjektstyring og systemutvikling. Bruk av verktøy, risikovurdering, valg av systemutviklingsmodell, tidsbruk, planlegging og gjennomføring gås gjennom.

Kapittel 6 - Design og arkitekturer viser gruppens overordnede arkitektur og struktur av de forskjellige modulene, samt samspill mellom disse.

Kapittel 7 - Implementasjon går gjennom programutviklingsfasen, og beskriver prosessen og valg tatt samt går i nærmere detalj rundt visse deler, spesielt utvikling av maskinlæringsmodell. I dette kapitlet vil kapittel 2 og kapittel 3 være til hjelp dersom noe er uklart.

Kapittel 8 - Diskusjon diskuterer resultatene oppnådd gjennom perioden og knytter disse opp mot læringsmålene samt effekt- og resultatmålene definert innledningsvis. I tillegg vil det drøftes mulige punkter som kan tas med i videre arbeid, samt gi en kritikk av oppgaven.

Kapittel 9 - Konklusjon gir en oppsummering av hovedfunnene i rapporten, samt de viktigste resultatene av arbeidet.

Kapittel 2

Teori

For å forstå rapportens innhold, vil sannsynligvis noe kunnskap om emnene nedenfor være påkrevd. Dette kapitlet gir en innføring i grunnleggende teori, med særlig vekt på maskinlæring. Denne delen blir knyttet opp mot implementasjonsdelen i kapittel 7.

2.1 Maskinlæring

Maskinlæring er en gren av kunstig intelligens som går ut på å la datamaskiner bruke statistiske metoder for å finne mønstre i datamengder. Datamaskinen analyserer datamengden, og vi sier at modellen lærer, eller blir trent opp. Måten modellen lærer kan deles opp i veiledet læring, ikke-veiledet læring og forsterket læring [3]. I denne oppgaven er det veiledet læring som blir brukt. I veiledet læring får modellen et sett med merkede treningsdata, og predikerer hva utgangsverdien(e) skal være [4, s. 6]. Ved å sammenlikne med av treningsdataens merkede «sanne» verdi, kan *tapet* i nevrale nettverk regnes ut.

Kunstig intelligens blir ofte forvekslet med maskinlæring, forskjellen er at maskinlæring blir regnet som en underart av kunstig intelligens. Kunstig intelligens er en samlebetegnelse for alle intelligente systemer. Man skiller gjerne mellom regelbaserte og datadrevne modeller. Maskinlæring er en datadreven modell [5]. Dette betyr at det er ingen regler som er programmert inn på forhånd, og at modellen lærer slike regler på egenhånd.

Inndeling av data

Ofte vil maskinlæringen foregå ved at treningsdataen blir inndelt i *trenings-, valideringsog testsett*. Treningssettet blir brukt til å tilpasse modellen innledningsvis. Videre vil valideringssettet bli brukt til å gi en nøytral vurdering av modellens tilpasning, samtidig som modellens *hyperparametre* blir justert. Til slutt vil testsettet gi en nøytral evaluering av modellens endelige tilpasning [6, s. 1]. Et vanlig scenario ved *overtilpasning* er at feilraten for testsettet er høyere enn feilraten for treningsog/eller valideringssettet. Formålet med denne inndelingen er med andre ord å

redusere generaliseringsfeilen, det vil si å redusere feilraten på ny data modellen ikke har sett. I denne oppgaven vil testsettet bestå av den nyligste dataen, og valideringssettet av den nest nyligste dataen.

Dyplæring

Dyplæring er en type maskinlæring som går ut på å trene opp nevrale nettverk, og det er en sentral metode innenfor maskinlæring. Ideen er at maskinen skal lære seg kunnskap den ikke vet fra før basert på et datasett som den leser av [7].

Nevrale nettverk

Kunstige nevrale nettverk

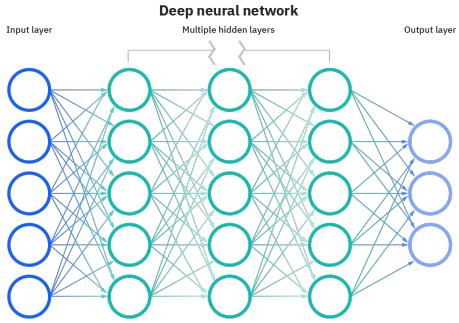
Et nevralt nettverk beskriver datastrukturer som er inspirert av nervecellenes struktur i hjernen. Nevrale nettverk er bygget opp av *nevroner*. Nevroner kan ta imot tallverdier og sende tallverdier videre. I et lag inngår n antall nevroner. Et nevralt nettverk består et inngangslag, skjulte lag og et utgangslag. Inngangslaget henter inn eksterne verdier i starten, skjulte lag brukes for å oppdage mer nyanserte trekk og utgangslaget produserer den endelige prediksjonen. Nevronene i lagene er koblet sammen slik at hver tilkobling har en vekt. I tillegg har de ofte en skjevhet eller bias [8].

For å regne ut et nevrons utgangsverdi i lag n etter inngangslaget, summeres alle verdiene til nevronene i lag $n-1$ er tilkoblet nevronen som multiplisert med sin vekt. Deretter adderes biasen, og summen av dette legges inn i en aktiveringsfunksjon (e.g. ligning (2.2)), som produserer nevronens utgangsverdi. En aktiveringsfunksjon er en matematisk funksjon som typisk introduserer en ikkelinearitet inn i nettverket. Behovet for dette kommer av at det ønskes at nettverket er dynamisk og kan tilegne seg kompleks informasjon fra inngangsdataen [9]. Biasen kan anses for å være et konstantledd som flytter aktiveringsfunksjonen til venstre eller høyre for å tilpasse seg dataen bedre.

For et gitt nevron med inngangsverdier $x_1, x_2, ...x_n$ og tilsvarende vekter $w_1, w_2, ...w_n$ og bias b, kan utgangsverdien av nevronen beskrives slik:

$$a = \phi\left(\left(\sum_{i=1}^{n} x_i \cdot w_i\right) + b\right) \tag{2.1}$$

Hvor ϕ er den gitte aktiveringsfunksjonen.

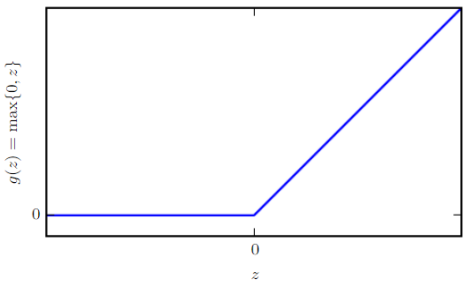


Figur 2.1: Skisse som viser kunstige nevrale nettverks struktur [10]

En svært relevant aktiveringsfunksjon i dette prosjektet er rectified linear unit (ReLU):

$$\phi(x) = x^+ = max(0, x) \tag{2.2}$$

ReLU sørger for at alle inngangsverdier mindre eller lik null forblir null som utgangsverdi, og er lineær for alle verdier større enn null. Sammenliknet med sigmoide aktiveringsfunksjoner, lider denne i mindre grad av problemet med forsvinnende gradienter [11].



Figur 2.2: Graf for RELU-funksjonen [12, s. 170]

En annen mye brukt aktiveringsfunksjon, blant annet i LSTM, er en sigmoid funksjon, som eksempelvis kan være:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Hvor utgangsverdien vil være på et intervall mellom 0 og 1.

For å bedømme hvor god modellen er gitt treningsdataen, brukes en kostnadsfunksjon for å beskrive tapet, eller feilmarginen mellom sannhetsverdien og predikert verdi. Innen regresjon er særlig *mean absolute error* (MAE), *mean squared error* (MSE) og *mean absolute percentage error* (MAPE) relevante:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \tag{2.4}$$

Hvor y_i er predikert verdi og x_i er faktisk verdi. MAE regner altså ut den gjennomsnittlige absoluttfeilen for alle predikerte verdier [13].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.5}$$

Hvor Y_i er faktisk verdi og $\hat{Y}_i$ er predikert verdi. Her vil større feilmarginer resultere i en disproporsjonal høyere MSE-verdi.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \tag{2.6}$$

Hvor A_i er faktisk verdi og F_i er predikert verdi. I motsetning til MAE, regnes MAPE ut den gjennomsnittlige prosentandelen den absolutte feilen mellom predikert og faktisk verdi utgjør i forhold til den faktiske verdien [14].

I starten er nettverket sine vekter tilfeldig justert, og den predikerte verdien(e) vil naturligvis være mindre gode i starten. Nettverket forbedrer seg ved å justere på disse vektene. Når disse vektene justeres, brukes en algoritme kalt *backpropagation*. I backpropagation-algoritmen beregnes gradienten til kostnadsfunksjonen med hensyn på hver vekt via kjerneregelen. Dette forsetter videre fra siste laget i nettverket [12, s. 200-220].

Denne gradienten brukes videre ved å ta den negative gradienten for å få retning mot et minimum av kostnadsfunksjonen. Ved å bevege seg i små steg i retning den negative gradienten og beregne gradienten på nytt, vil vi nå et (lokalt) minimum [12, s. 80-82]. Det vil vise seg at valg at steglengde, også kalt læringsrate, kan ha stor påvirkning på hvilket minimum som nås og konvergens. Det er ingen garanti for det globalet minimumet av funksjonen nås.

Videre blir det gjort rede for noen spesielle varianter av kunstige nevrale nettverk.

Konvolusjonelle nevrale nettverk

Konvolusjonelle nevrale nettverk er en type kunstige nevrale nettverk. Arkitekturen er i stor grad lik nevrale nettverk, bortsett fra at konvolusjon blir brukt i stedet for matrisemultiplikasjon i minst ett av lagene. Konvolusjonsoperasjonen er som regel skrevet slik:

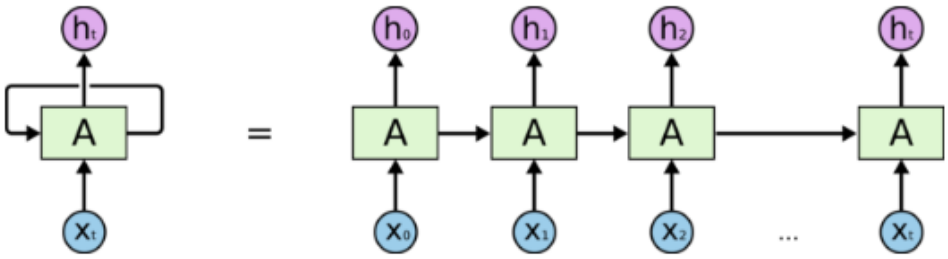
$$s(t) = (x * w)(t) \tag{2.7}$$

Hvor x typisk kalles inngangsverdien og w kalles filteret, eller *kernel*. Motivasjonen bak å bruke konvolusjon er å redusere antall parametre og antall steg i beregningen av utgangsverdiene i modellen. Den økte effektiviteten er som regel signifikant [12, s. 326-330].

Selv om rekurrente nevrale nettverk typisk blir ansett for å være de beste arkitekturene for sekvensmodellering (slik som å forutse tidsrekker), har noen studier vist at enkle CNN-arkitekturer kan utkonkurrere RNN-arkitekturer, slik som LSTM, i problemer relatert til sekvensmodellering. Basert på dette, er konvolusjonelle nevrale nettverk et naturlig startpunkt for denne typen problem [15]. Et annet studie argumenterer for at denne typen nettverk er lettere å trene enn RNN-nettverk, ettersom at det ikke lider av problemer som forsvinnende eller eksploderende gradienter, i tillegg til å være mer effektive [16].

Rekurrente nevrale nettverk

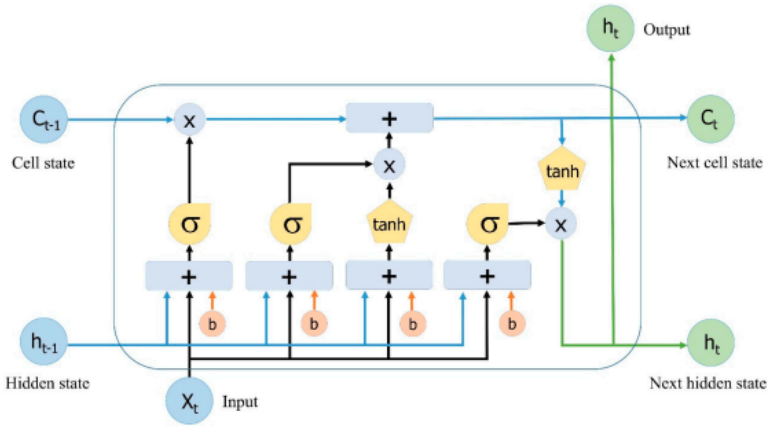
Rekurrente nevrale nettverk er en type kunstig nevralt nettverk som har sekvensiell eller tidsrekkedata som inngangsverdi. De kjennetegnes ved å ha kjedelignedende strukturer hvor det brukes en minnemodul til å lagre viktig informasjon fra tidligere steg. Dette tillater nettverket å akseptere en sekvens av inngangsverdier. Dette betyr at steg $t-1$ har innflytelse over resultatet av steg t [17]. Denne typen nettverk blir typisk brukt til problemer som naturlig språkbehandling, stemmegjenkjenning, prediksjoner av tidsrekker etc. Long short-term memory er en spesiell RNN-arkitektur som blir brukt i dette prosjektet. Denne blir forklart i nærmere detalj i neste del.



Figur 2.3: Skisse som viser RNNs struktur [18]

LSTM

Long Short Term Memory (lang korttidsminne) er en type rekkurent nevralt nettverk som har en arkitektur som ulikt RNNer kan lære seg langtidsinformasjon, noe som tradisjonelle RNN har problemer med på grunn av eksploderende eller forsvinnende gradienter. Et LSTM-nettverk består typisk av minnebrikker kalt celler. To tilstander blir overført til neste celle, celletilstanden og den skjulte tilstanden. Celletilstanden reflekterer langtidsminnet, og kan påvirkes via porter som bestemmer hvilken del av den gamle tilstanden som skal glemmes. Den skjulte tilstanden reflekterer i større grad korttidsminnet [17].



Figur 2.4: Skisse som viser en LSTM-celle sin struktur [17]

2.2 Prediksjoner

Oppgaven handlet i stor grad om prediksjoner, og det å lage prediksjoner med høy nøyaktighet. En prediksjon er en antagelse eller prognose som gjelder et fremtidig forhold, og er basert på sannsynlighet [19]. Ordet er latinsk og er todelt, *pre* betyr før"og *diction* betyr å snakke. ordet betyr altså å snakke før, som er akkurat det en prediksjon er, en prognose hva som kommer til å skje før det har skjedd.

For å få høyere nøyaktighet på prediksjonene, ble det foreslått av veileder at prediksjonene kunne være tallintervaller. Altså at predikert pris kunne for eksempel være [20.37 - 20.57], for å gjøre det lettere å være helt nøyaktig. Likevel ønsket DrivstoffAppen at prediksjonene skal være èn enkel tallverdi, og ikke et intervall av tallverdier. Av denne årsaken blir det vanskeligere å få en høy nøyaktighetsrate, men det vil i gjengjeld bli lettere å tolke verdien til prediksjonen. Det ble gjort som DrivstoffAppen ønsket, og enkle verdier blir produsert som prediksjoner på det ferdige produktet.

Ekspertbaserte prediksjoner

Ekspertbaserte prediksjoner er prediksjoner fra eksperter innen et fagområde. Ved å være ekspert, har man mye kunnskap og informasjon om feltet. Dermed er det logisk å anta at en ekspert kan utføre en bedre prediksjon innenfor sitt fagområde, enn en tilfeldig person. Man gir prediksjonene til eksperter innenfor et fagområde mer kredibilitet enn man gjør med en person man ikke kjenner.

Analogibaserte prediksjoner

Prediksjoner som baserer seg på sammenlikninger og likheter mellom to ting, kalles analogibaserte prediksjoner. La oss si at fly *A* flyr Oslo-Trondheim klokken 13:00, og lander på Gardermoen klokken 14:35. Fly *B* flyr Oslo-Trondheim

klokken 15:00, men vi vet ikke når det landet på Gardermoen. Med analogibaserte prediksjoner, vil det være logisk å predikere at fly *B* lander på Gardermoen kl 16:35. Dette er fordi flyreisene er helt like bortsett i fra starttid, så man regner med at reisen tar like lang tid.

Kapittel 3

Teknologier

For å få gode prediksjoner som kan leveres raskt kreves det gode og pålitelige teknologier. Noen av teknologiene som har blitt brukt i prosjektet har etter prosjektplanleggingsperioden blitt bestemt av oppdragsgiver, slik at likevel har blitt kjørt en prosess på valg av teknologier i forkant. Det har blitt brukt et bredt spekter av verktøy, ettersom det har blitt jobbet i flere felter innenfor datateknologi. Det har blitt utviklet tre forskjellige typer programvare i tre forskjellige utviklingsmiljøer.

3.1 Android

Videreutvikling av appen har blitt gjort i Android Studio, som er det offisielle utviklingsmiljøet for Android-utvikling. En vesentlig fordel med dette er muligheten for å kjøre apper i Android Emulator, som er en virtuell representasjon av en gitt Android-enhet, og utviklingsverktøy for Android.

Kotlin

Kotlin er et statisk skrevet programmeringsspråk som støtter både objectorientert og funksjonell programmering. Kotlin er designet for å ha interoperabilitet med Java, slik at biblioteker fra Java kan benyttes i Kotlin [20]. Sammenliknet med Java, reduserer Kotlin i stor grad *boilerplate code*, som er kode som gjentas mange ganger med liten variasjon.

I Java er NullPointerException svært vanlig, og Kotlin reduserer sannsynligheten for dette inntreffer ved å forby tilgang til en nullreferanse uten å bruke en not-null assertion operator (!!) [21].

Kotlin har siden 2019 vært Googles foretrukne programmeringsspråk for Android-utvikling, og det er også tilfellet i dette prosjektet. [1]

[1] `https://developer.android.com/kotlin/first`

3.2 Valg API

Av hensyn til Drivstoffappens eksisterende arkitektur, ble Spring Boot valgt som rammeverk for utvikling av API. Siden dette ikke var kjent i planleggingsperioden, ble det likevel kjørt en prosess på valg av mulige kandidater.

Go

Go er et programmeringsspråk med sterk typetildeling som er bygget for å lage samtidig (engelsk: concurrent) programvare [22]. Go er raskt, og samtidighet og parallellisme kan håndteres ved hjelp av *goroutines*. I tillegg har standardbiblioteket støtte for webutvikling, derunder støtte for HTTP-funksjonalitet. Fordelen som fremgår av dette er redusert nødvendighet av eksterne tredjeparts avhengigheter (engelsk: dependency).

Spring Boot

Spring Boot[2] er et webrammeverk som kan brukes til å lage REST API. Sammen med Spring-rammeverket som helhet tilbys testrammeverk som kan etterlikne (engelsk: mock) objekter, *dependency injection*, samt tilby abstraksjoner for transaksjons-API som JDBC, Hibernate og JPA.

Spring Boot gjør det i tillegg enkelt å sette opp en lokal testdatabase i hovedminnet og konfigurasjon, slik at eksempeldata enkelt kan innsettes og slettes etter hver testiterasjon.

Github Actions

Github Actions er en plattform for kontinuerlig integrasjon og kontinuerlig levering (CI/CD), tilbyr automatisering for bygge-, test og distribusjonspipeliner. Disse pipelinene kan utføres ved hver pull request eller hvert push til en gitt branch [23].

3.3 Valg maskinlæring

Python

Python er et tolket, objektorientert programmeringsspråk, opprinnelig startet av Guido van Rossum. [3] Python er et åpenbart valg når det kommer til maskinlæring. Python er et av verdens mest populære programmeringsspråk [24], har en enkel og konsis syntaks og har god støtte fra tredjeparts biblioteker, slik som biblioteker for databehandling og visualering av data. Her nevnes eksempelvis pandas[4],

[2] Spring Boot: `https://spring.io/projects/spring-boot`
[3] Python: `https://www.python.org/`
[4] pandas: `https://pandas.pydata.org/`

sklearn[5], seaborn[6], numpy [7] og matplotlib[8].

I tillegg tilbyr maskinlæringsrammeverk som Tensorflow og PyTorch et Python API som kan brukes til å bygge maskinlæringsmodeller, samtidig som det abstraherer bort kompleksiteten til mange algoritmer involvert i treningen.

En annen tungtveiende grunn faller på muligheten til å bruke Jupyter Notebooks[9]. I løpet av treningen av maskinlæringsmodell vil kodelinjer endres på kontinuerlig. Ettersom noen få kodelinjer blir kjørt mange ganger, vil det spares tid ved å kunne kjøre enkeltceller med kode som Jupyter Notebook tilbyr.

TensorFlow

TensorFlow er et programvarebibliotek fra laget av Google Brain, som er en samling av åpen kildekode for maskinlæring, og blir brukt til å utvikle maskinlæringsmoduler. [10] I tillegg tilbys det støtte for bruk av Nvidia GPU under trening gjennom grensesnittet CUDA, som reduserer tiden som kreves å trene modeller [25]. TensorFlow har blitt brukt for å utvikle maskinlæringsmodulen for prediksjon av drivstoffpriser.

PyTorch[11] er også et høyt ansett maskinlæringsrammeverk, men ble valgt bort til fordel for TensorFlow på grunn av at TensorFlow har god dokumentasjon på prediksjoner av tidsrekker som er relativt nærliggende det som har blitt gjort i denne oppgaven.

Kodeliste 3.1: Eksempel på oppbygging av en lagvis modell i Tensorflow

```
import tensorflow as tf

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

Optuna

Optuna[12] er et rammeverk for å automatisk optimalisere hyperparametre for maskinlæring [26]. Enkelt forklart vil rammeverket trene og evaluere en modell n antall ganger, samtidig som forskjellige definerte hyperparametre endres på. Dette kan være aktiveringsfunksjon, antall lag, antall nevroner, læringsrate etc.

[5] sklearn: `https://scikit-learn.org/stable/`
[6] seaborn: `https://seaborn.pydata.org/`
[7] NumPy: `https://numpy.org/`
[8] Matplotlib: `https://matplotlib.org/`
[9] Jupyter: `https://jupyter.org/`
[10] Tensorflow: `https://www.tensorflow.org`
[11] PyTorch: `https://pytorch.org/`
[12] Optuna: `https://optuna.org/`

Motivasjonen bak bruken av dette rammeverket hviler på at maskinlæring i sin helhet består av prøving og feiling. En typisk prosess som foregår under læringen er nettopp å justere disse parameterene. Optuna gjør at dette blir langt mindre tidkrevende, samtidig som at denne prosessen kan gjøres i bakgrunnen uten input fra utvikler.

Kapittel 4

Kravspesifikasjon

Formålet med dette kapitlet er å beskrive hvilke krav som har blitt fremlagt innledningsvis og under utvikling.

4.1 Funksjonelle krav

Gjennom diskusjon med kontaktpersonene i Headit og DrivstoffAppen har følgende funksjonelle kravene blitt fremsatt for løsningen:

- Brukeren skal kunne velge en gitt stasjon og se prediksjoner for denne stasjonen.
- Brukeren skal kunne se prediksjoner i en tabell form.
- Brukeren skal kunne se prediksjoner som en graf eller diagram.

4.2 Ikke-funksjonelle krav

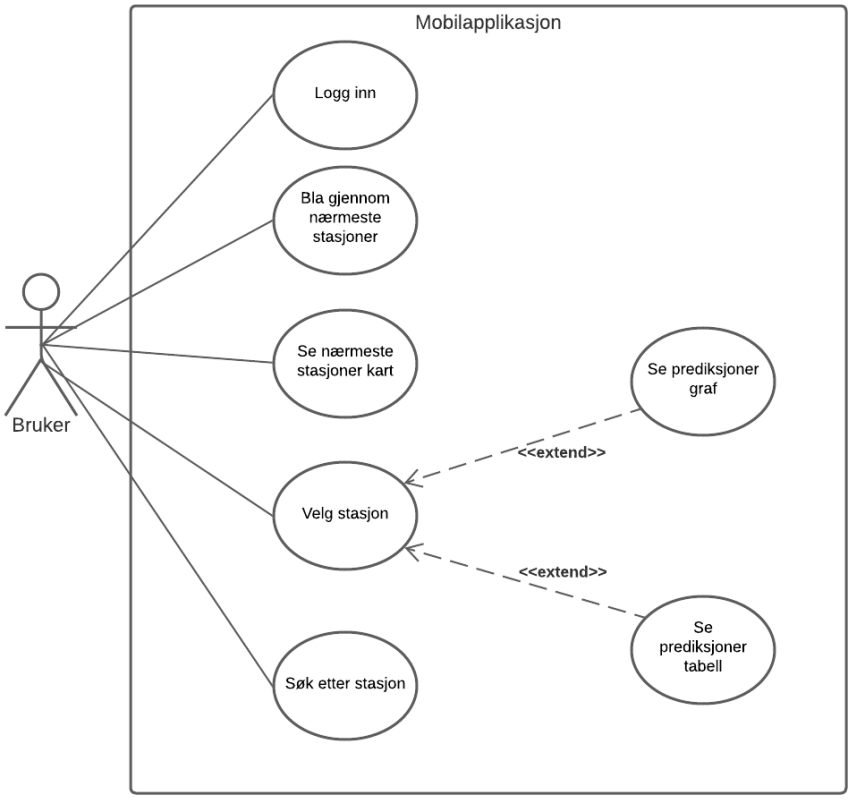
I tillegg til funksjonelle krav, er det blitt fremsatt en rekke ikke-funksjonelle krav. Disse har delvis oppstått som følge av dialog med Headit og DrivstoffAppen, og delvis gjennom diskusjon i gruppen.

- Implementasjonen skal foregå sømløst slik at løsningen ikke trenger input fra bruker/administrator.
- Det skal være tydelig og godt opplyst i brukergrensesnittet at prisen er en prediksjon og ikke manuelt lagt inn av brukere.
- Løsningen skal ha en fleksibel struktur, slik at modeller og komponenter enkelt kan byttes ut.
- Løsningen bør legge til rette for enkel feilsøking og kvalitetskontroll. I maskinlæringsmodeller bør det implementeres en evalueringsparameter som kan avdekke feil.
- Løsningen skal ikke føre til økt opplevd treghet i DrivstoffAppen.
- Implementasjonen skal foregå sømløst slik at løsningen ikke trenger input fra bruker/administrator.

- Koden skal ha tilstrekkelig dokumentasjon, med særlig vekt på kommentering av ikke-trivielle funksjoner. Eksempler på trivelle funksjoner er gettere/settere i modellklasser. Vedrørende API skal dokumentasjon foreligge slik at alle endepunkt er beskrevet, samt instruksjoner på hvilke spørringsparametre og request body som kan/må være inkludert i hver forespørsel.
- API skal ha tilstrekkelig testdekning, med et minimum på 80 prosent. Dette er for å sikre at funksjonaliteten til backend er robust.
- Lokalisering (språk) i frontendløsning skal som minumum omfatte engelsk og norsk. Det vil også være en fordel å lokalisere til svensk og dansk.

4.3 Bruksmønsterdiagram

Det er satt opp et bruksmønsterdiagram for å vise hvordan det er tenkt at brukeren vil benytte applikasjonen for å få informasjon om predikerte priser for en gitt stasjon. Det understrekes at de fleste av disse bruksmønsterene allerede er implementert, bortsett fra *Velg stasjon*, *Se prediksjoner graf* og *Se prediksjoner tabell*, som er beskrevet i de funksjonelle kravene. Nedenfor beskrives disse tre bruksmønsterene i nærmere detalj.



Figur 4.1: Bruksmønsterdiagram for mobilapplikasjon

Bruksmønster	Velg stasjon
Aktør	Bruker
Formål	Velg spesifikk stasjon for mer informasjon
Forutsetning	Brukeren må være logget inn
Beskrivelse	Brukeren kan velge en spesifikk stasjon for å få informasjon om dagens drivstoffpriser. Her kan også brukeren få informasjon om prediksjoner. Stasjon kan velges enten ved å søke eller finne stasjon på kart.

Tabell 4.1: Bruksmønster 1: Velg stasjon

Bruksmønster	Se prediksjoner graf
Aktør	Bruker
Formål	Se prediksjoner for stasjon visualisert
Forutsetning	Brukeren må være logget inn og må ha valgt en stasjon
Beskrivelse	Gitt at brukeren har valgt en stasjon, kan prediksjonene visualiseres i en graf eller diagram hvor fremtidige datoer er på x-aksen og pris på y-aksen. Brukeren vil da lettere se utviklingen i pris.
Variasjoner	Brukeren kan velge om priser fra drivstofftype diesel eller blyfri 95 skal vises.

Tabell 4.2: Bruksmønster 2: Se prediksjoner graf

Bruksmønster	Se prediksjoner tabell
Aktør	Bruker
Formål	Se prediksjoner for stasjon i tabellform
Forutsetning	Brukeren må være logget inn og må ha valgt en stasjon
Beskrivelse	Gitt at brukeren har valgt en stasjon, kan prediksjonene vises i en tabell, med dag og pris, hvor billigste pris vises markert i grønn tekst.
Variasjoner	Brukeren kan velge om priser fra drivstofftype diesel eller blyfri 95 skal vises

Tabell 4.3: Bruksmønster 3: Se prediksjoner tabell

Kapittel 5

Utviklingsprosess

5.1 Systemutviklingsmodell

I starten av prosjektperioden ble gruppen enige om å velge Scrum som systemutviklingsmodell. Grunnen til dette var i stor grad fordi det var antatt at et svært enkelt produkt kunne lages innledningsvis. Videre så gruppen for seg inkrementelle leveranser som var stadige forbedringer til det som allerede eksisterte. I tillegg var det vanskelig å planlegge langsiktig hvilke aktiviteter som skulle påbegynnes. Med Scrum kan disse aktivitetene planlegges mer fortløpende. Sluttmålet var også relativt løst definert, og ved å definere sprinter blir det lettere å dokumentere og kommunisere fremgang i prosjektet med alle de involverte partene. Kanban og Scrumban ble også vurdert, men ble valgt bort på grunnlag av at Scrum gir bedre oversikt over gruppens progresjon og mål for ekstern oppdragsgiver.

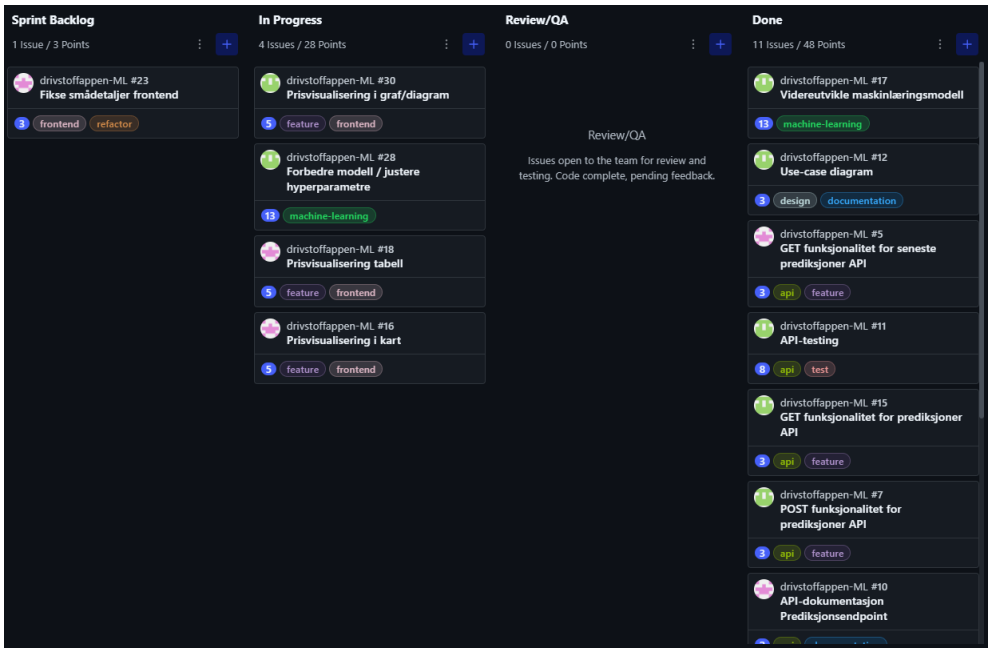
Når Scrum ble brukt, var kommunikasjon særs viktig, slik at gruppen og oppdragsgiver hele tiden har vært oppdaterte på hvilke oppgaver som ble gjort først. *The Scrum Guide* definerer fem Scrum-hendelser: The Sprint, Sprint Planning, Daily Scrum, Sprint Review og Sprint Retrospective [27]. På slutten av hver sprint ble neste sprint planlagt. I tillegg har det vært daglige oppdateringer (arbeidsdager) holde gruppen internt oppdatert på hva som skulle gjøres og hva som hadde blitt gjort. Opparbeidet kunnskap ble også formidlet videre. Før neste sprint har blitt planlagt, har det blitt gjort en evaluering på den forbigående sprinten og om tilpasninger måtte gjøres. Dette kunne for eksempel være å dele større oppgaver inn i mindre deler, eller omfordele oppgaver. Til slutt har gruppen sett på måter å forbedre samarbeidet og kommunikasjon i neste sprint. Her ble eventuelle utfordringer tatt opp, eksempelvis motivasjon og informasjonsflyt.

5.2 Scrum-brett

For å organisere arbeidsoppgavene for hver sprint, ble et issue-brett opprettet. Hvert *issue*, eller kort, representerte en spesifikk oppgave av et visst omfang. Hvert kort har en poengskår som sa noe om hvor omfattende oppgaven var. I tillegg

hadde alle kortene merkelapper som gjorde det enkelt å se hvilket område hver oppave omfattet, slik som dokumentasjon, testing, kode etc. Ved å klikke på kortene, kom en mer detaljert beskrivelse av oppgaven. Formålet med brettet var å konkretisere og dele større oppgaver inn i mindre deler. I tillegg hindret det gruppemedlemmene å begynne på samme oppgave, samtidig som det informerte om hvem som gjorde hva. Det ble også enklere å holde styr på hver sprint sine gjøremål.

Brettet hadde flere kolonner som beskrev oppgavenes status og prioritet. *Sprint Backlog* beskriver hvilke oppgaver som skulle gjøres i den gjeldende sprint. *Icebox* inneholdt planlagte oppgaver som skulle gjøres på ubestemt tid i fremtiden. *Product backlog* beskrev oppgaver som var planlagt i nærliggende sprinter. Når oppgaver i sprinten var ferdige og gjennomgått, ble disse flyttet til *Done*.



Figur 5.1: Issue-brett for gruppen

5.3 Statusmøter og beslutningspunkter

Gruppen har hatt jevnlige veiledningsmøter, innledningsvis en gang i uka. Med oppdragsgiver har det blitt kalt inn demomøte omtrent på slutten av hver sprint, hvor resultatene for gjeldende sprint har blitt presentert, i tillegg til at plan for ny sprint har blitt diskutert. Arbeidsgiver har ofte kommet med innspill på hvilke oppgaver som burde prioriteres. I tillegg har det vært fire møter i planleggingsfasen av prosjektet for å gå gjennom problemstillingen, forventninger, få innføring i maskinlæring, arbeidsmetoder og generelt hvordan prosjektet skal utføres. Det har også vært jevnlig kontakt med eierne av Drivstoffappen. For å se møterefera-

ter, henvises det til vedlegg F.

Internt i gruppen har det på mange arbeidsdager vært samtaler (daily Scrum), hvor følgende punkter har blitt diskutert:

- Hvilke oppgaver som har blitt gjort
- Hvilke utfordringer som hadde oppstått
- Hvordan disse kan løses
- Hvilke oppgaver som burde prioriteres
- Hvilke oppgaver som burde samarbeides på
- Status på oppnåelse av sprintens mål

5.4 Sammendrag av sprinter

I løpet av bachelorprosjektet har det blitt gjennomført en prosjektplansperiode og 7 sprinter. I utgangspunktet var det beregnet en jevn arbeidsflyt fra 1. februar til 20. mai, men på grunn av at et annet fag ble lagt frem som en kortere intensiv periode fra 24. januar til 24. mars, har arbeidet med bacheloroppgaven pågått mest intensivt etter denne perioden.

I utgangspunktet skulle sprintene avsluttes med demonstrasjoner av det som har blitt gjort, av hensyn til kolliderende timerplaner og ulike årsaksforhold, har noen av disse blitt flyttet til starten av den neste sprinten.

Prosjektplan (12. jan - 31. jan)

Denne perioden markerte starten på bacheloroppgaven. Store deler av denne tiden gikk til å lage prosjektplanen som ga grunnlaget for videre arbeid i perioden etter. I tillegg gikk også en del av tiden til å vurdere og lese seg opp på forskjellige teknologier, samt å få en grunnleggende forståelse av maskinlæring og nevrale nettverk.

Sprint 1, 1. feb - 11. feb

Denne sprinten startet med en innføring i maskinlæring og behandling av data. Gruppen skulle egentlig ferdigstille et grunnleggende API, det ble overført til neste sprint på grunn av det viste seg at gruppen skulle jobbe videre på et eksisterende API, som tok litt tid å få tilgang til, samt sette seg inn i.

Sprint 2, 14. feb - 25. feb

Her fortsatte utviklingen av API. En svært enkel modell ble laget som kunne lese priser og predikere nye priser. Det ble også forsøkt å sette opp et testmiljø for API, men av ulike grunner var dette vanskelig å implementere.

Sprint 3, 28. feb - 11. mars

På grunn av et høyt arbeidstrykk i et annet fag, ble videre utvikling i stor grad bortprioritert i løpet av denne perioden. Litt tid ble brukt på å sette seg inn i kodebasen til Android-applikasjonen.

Sprint 4, 14. mars - 25. mars

Starten av denne sprinten ble igjen nedprioritert som følge av eksamen og rapportinnlevering i annet fag. Når dette ble ferdig, ble det jobbet videre på maskinlæringsmodellen, hvor CNN- og LSTM-arkitekturer ble utprøvd. I tillegg ble oljepriser forsøkt integrert inn i modellen i håp om å øke nøyaktigheten. Frontenddelen av applikasjonen ble også startet for fullt.

Sprint 5, 28. mars - 8. apr

Første del av denne sprinten gikk til å ferdigstille en MVP for Android-appen, med tabell og graf for visualisering av predikerte priser. Videre ble API refaktorert med ny arkitektur og det ble skrevet tester. I tillegg ble CI-arbeidsflyter lagt til og hyperparametre med Optuna ble introdusert.

Sprint 6, 11. apr - 22. apr

Denne sprinten gikk i stor grad til å forsøke å forbedre maskinlæringsarkitekturene som var introdusert med hyperparameterjustering. I tillegg ble frontend forbedret basert på tilbakemelding fra Headit og Drivstoffappen.

Sprint 7, 25. apr - 20. mai

Siste sprint sin hovedtyngde gikk på å ferdigstille rapporten, som krevde mange arbeidstimer. I tillegg ble det foretatt noen siste forbedringer og justeringer på maskinlæringsmodellen i håp om å forbedre resultatene ytterligere.

5.5 Verktøy/programvare

I styring av prosjektet har det vært viktig å velge verktøy som fremmer effektivitet rundt kommunikasjon og planlegging, samt automatisering så fremt det er mulig. Under er det listet verktøy som har blitt brukt gjennom utviklingsprosessen.

Toggl Track har blitt brukt til å registrere timelister i gruppen. Fordelen med dette er mulighet til å generere rapporter over tidsbruk over et tidsrom, slik at det er enkelt å se om tidsmålet har blitt nådd. I tillegg unngås manuell innskrivning i Excel-ark o.l.

Overleaf har blitt brukt i all hovedsak til å skrive forprosjektplan og rapport i LaTeX. Dette har lettet arbeidet med samskriving og samle på kilder.

Github har blitt brukt som versjonskontroll til maskinlæringsmodellen og refaktorert API.

Bitbucket har blitt brukt som versjonskontroll vedrørende kode knyttet til Drivstoffappen sin kodebase.

Discord og Teams har blitt brukt til kommunikasjon og oppsett av møter.

Zenhub har blitt brukt til organisering av backlog og issues. Kan enkelt integreres med Github.

Onedrive/Sharepoint har blitt brukt til å dele filer som typisk er lite hensiktsmessig å inkludere i versjonskontroll og ikke skrives i Overleaf. Eksempelvis møtereferat og andre uformelle notater.

TeamGantt har blitt brukt for å lage og oppdatere Gantt-diagram. Dette diagrammet har gitt gruppen et mer langsiktig perspektiv på gjøremål.

LucidChart har blitt brukt til å lage use case diagram og liknende.

5.6 Tidsbruk

Registering av tidsbruk

I løpet av prosjektet har det blitt registert timebruk i Toggl Track, hvor timefordelingen har blitt generelt inndelt i forskjellige kategorier. Denne inndelingen vises på figur 5.2.

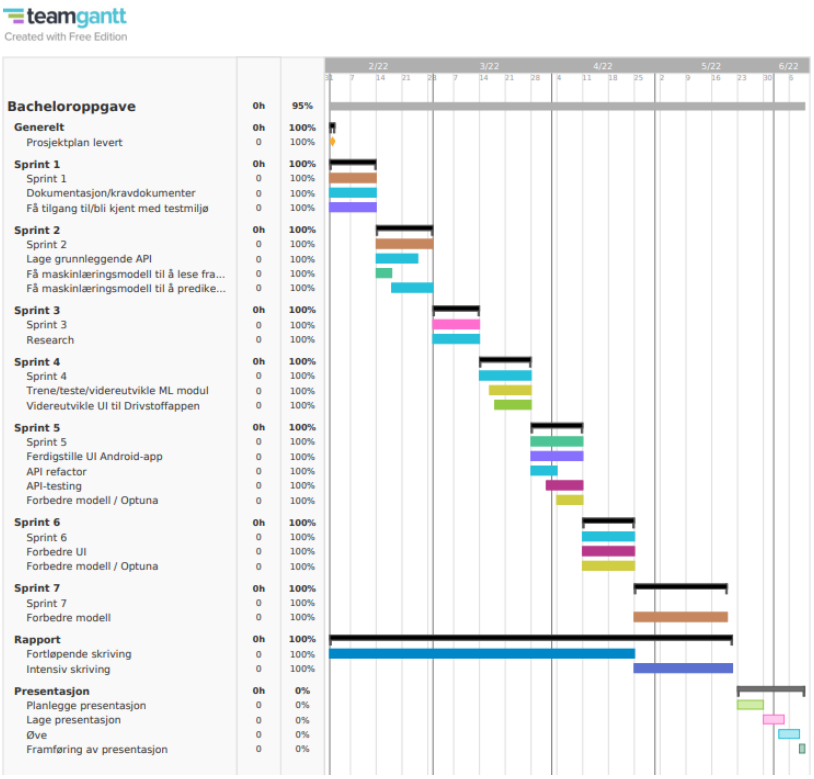
Generelt går inn under møter, dokumentasjon, diverse oppgaver. Utvikling er relatert til utvikling av de tre modulene. Prosjektplan refererer til arbeid gjort før 1. februar. Research refererer til tid brukt til å sette seg inn i eksisterende kodebase og lære ny teknologi og metoder. Rapport refererer til rapportskriving. Noen timeregistreringer inngår i flere kategorier, og reflekterer dermed ikke helt nøyaktig tidsbruken. Det er antatt at spesielt kategoriene generelt og research har flere timer enn registrert. Samtidig er det klart at utvikling og rapportskriving har opptatt mest tid.

Figur 5.2: Tidsbruk per kategori mot slutten av prosjektet

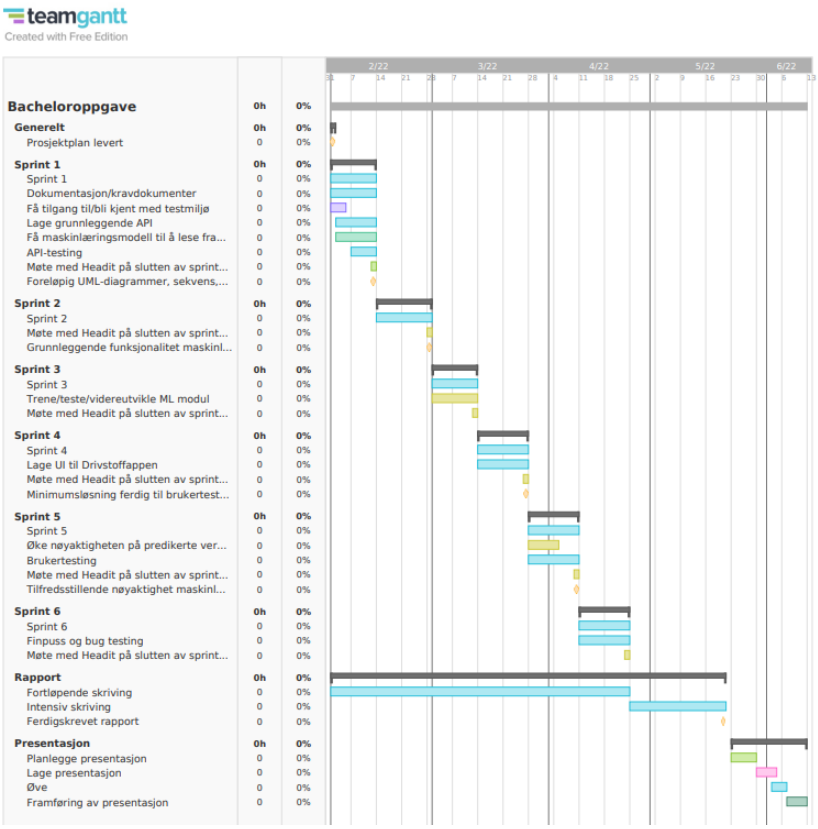
Gantt-diagram

Et Gantt-diagram ble laget i prosjektplanleggingsfasen, som i stor grad baserte seg på antagelser. I løpet av prosjektet har noen av disse punktene endret seg, og noen aktiviteters omfang har vært større enn antatt. Generelt sett har den opprinnelige planen i stor grad blitt fulgt fra prosjektstart til slutt.

De mest vesentlige endringene har omhandlet refaktorering og testing av API, i tillegg har maskinlæringsdelen inngått under flere sprinter enn planlagt, samt startet på videreutvikling litt senere enn planlagt.



Figur 5.4: Siste versjon av Gantt-diagram før innlevering



Figur 5.3: Første versjon av Gantt-diagram 31. januar

5.7 Risikoanalyse

Det har blitt foretatt en risikoanalyse med de punktene som synes relevant for gruppen. Her listes det potensielle fare/problemer som kunne ha oppstått, hvilke konsekvenser disse hadde hatt og tiltak for å redusere denne risikoen. Alle risikomomenter har fått sannynlig- og konsekvenspoeng som sier noe om hvilke problemer som må prioriteres. Her følger poengene risikomatrisen som vist under, hvor 2-4 er grønt nivå, 5-7 gult nivå og 8-10 rødt nivå.

Noen av disse momentene inntraff uansett, slik som *Sykdom og andre forhold*, som er vanskelig å forebygge, men gruppen opplevde at å identifisere risikomomenter har bidratt til å redusere forekomsten/alvorlighetsgraden av disse, og har således vært en nyttig aktivitet i planleggingen av prosjektet.

Sannsynlighet	Konsekvens: Ubetydelig 1	Mindre alvorlig 2	Betydelig 3	Alvorlig 4	Kritisk 5
Svært sannsynlig 5					
Meget Sannsynlig 4			#11		
Sannsynlig 3			#5	#2	
Lite sannsynlig 2		#3	#4, #8, #9	#1	
Usannsynlig 1			#10	#7, #12	#6

Figur 5.5: Risikomatrise

NR	HVA	MULIGE KONSEKVENSER	RISIKO			REDUSERENDE TILTAK
			SANNSYNLIGHET	KONSEKVENS	RISIKOINDEKS	
1	Oppgaven er for omfattende/kompleks	Ikke får ønsket resultat. Mye tidsbruk.	2	4	6	Fokuser på delinnleveringer som fungerer. Bruk til å lære viktige ting. Realistisk målsetning.
2	For lite data til treningssett	Unøyaktig/lite brukbare prediksjoner	3	4	7	Se etter alternative kilder til drivstoffdata (oljepris for eksempel).
3	Tap av kode som ikke er lagt til i repository	Ekstra tidsbruk	2	2	4	Commit og push til Github hyppig slik at koden er flere steder.
4	Sykdom og andre forhold	Resterende gruppemedlemmer må ta på seg flere oppgaver.	2	3	6	Alle gruppemedlemmer burde ha innsikt i hvordan en løser alle oppgaver som kommer opp.
5	Konflikter innad i gruppa	Redusert arbeidsinnsats og effektivitet. Forverring av arbeidsmiljø.	3	3	6	Definer gode regler for arbeid og væremåte. Ha god kommunikasjon og bevissthet rundt dette.
6	Tap av hele kodebasen	Alt arbeid lagt fremt til da forsvinner. Må starte på nytt.	1	5	6	Tilse at kode er oppdatert både lokalt og remote hos begge gruppemedlemmer
7	Rapport ikke ferdig til tidsfristen	Ufullstendig innlevering, dårligere karakter.	1	4	5	Jobb rutinert og med mål om å bli ferdig i god tid for tidsfrist. Legg ting inni rapporten under veis.
8	Sluttbrukeren forstår ikke funksjonalitet	Misnøye med applikasjonen, redusert tillit.	2	3	5	Utfør brukertester fra forskjellige demografiske grupperinger slik at en får et helhetlig bilde av brukernes intuisjon av brukergrensesnittet.
9	Komponentene har høy kopling/lite abraksjon	Komponentene er avhengige av hverandre. Vanskeligere å vedlikeholde.	2	3	5	Ha fokus på en god arkitektur og vær obs på dette tidlig. Definer hver komponent sin rolle.
10	Oppgaven er for lite omfattende	Produktet blir for lite i forhold til forventet arbeidsmengde.	1	3	4	Spør oppdragsgiver om forslag/krav om utvidelser, mer funksjonalitet o.l.
11	Motivasjonen forsvinner i løpet prosjektet	Redusert arbeidsinnsats, mål for prosjektet blir ikke nådd.	4	3	7	Spille hverandre gode, bryte problemet ned i mindre deler, sett delmål.
12	Levert løsning svarer ikke til oppdragsgivers forventninger	Ferdig produkt blir ikke slik oppdragsgiver har tenkt seg.	1	4	5	Ha møter etter sprinter hvor utført arbeid blir gjennomgått. Klar kommunikasjon mellom studenter og oppdragsgiver.

Figur 5.6: Risikovurdering gjort innledningsvis i prosjektet

Kapittel 6

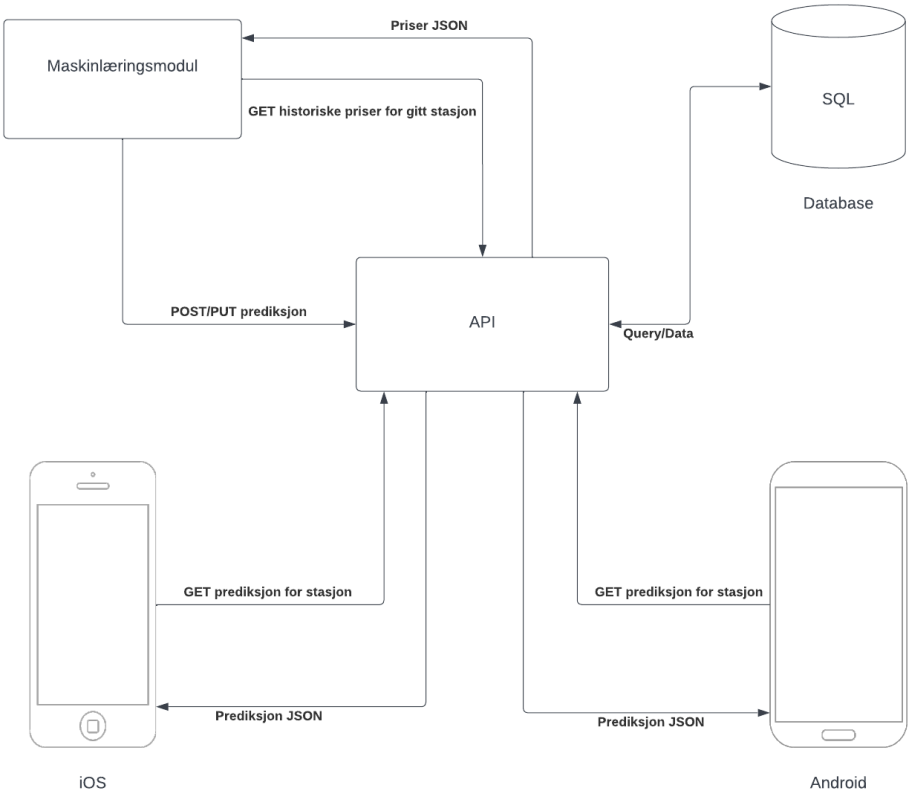
Design og arkitektur

6.1 Overordnet design

Når den overordnede strukturen ble bestemt, var det fokus på å separere tre komponenters virkemåte fra hverandre: database, mobilenhetene (iOS og Android) og maskinlæringsmodulen. Dette har blitt oppnådd ved å introdusere et REST API, som håndterer alle CRUD (Create, Retrieve, Update, Delete) operasjoner fra og til databasen. En illustrasjon av denne strukturen kan ses på figur 6.1

Formålet med denne strukturen er å hindre at backend-funksjonalitet sin implementasjon blir implementert flere ganger. Ved å definere et grensesnitt som bestemmer hvordan en ressurs skal lages, hentes, oppdateres eller slettet, vil endringer i backend ikke føre til endringer i de komponentene som benytter seg av API-et. Eksempelvis ville en endring fra en SQL-database til en NoSQL-database, som MongoDB, begrenset seg til hvordan APIet samhandler med denne nye databasen. Alternativet hadde vært å oppdatere alle komponentenes implementasjon. ML-modulen og Android- og iOS-appen trenger som følge av dette kun funksjonalitet for å kalle API og modellklasser for responsen. Dette er spesielt viktig dette tilfellet hvor mange komponenter er involvert.

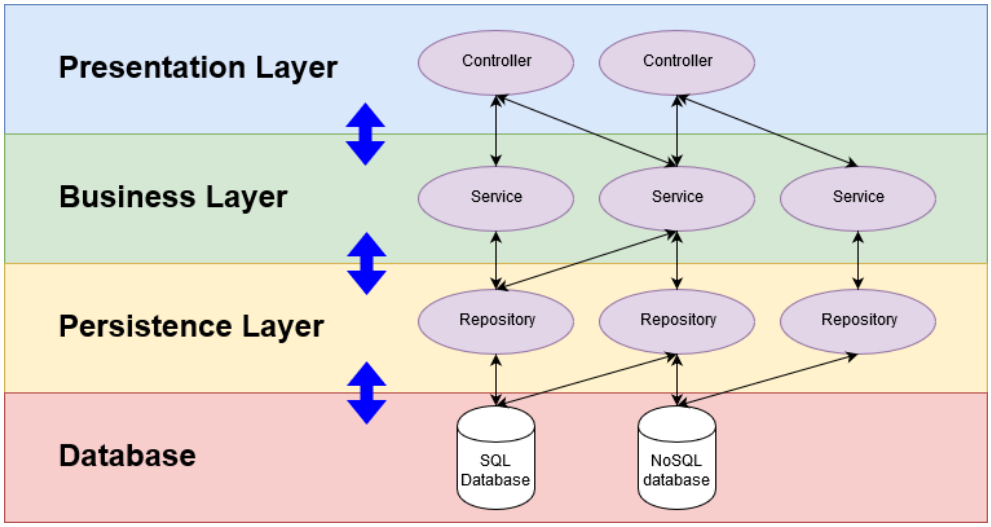
En annen fordel er en redusert avhengighet av tredjeparts bibliotek. Foruten biblioteker for HTTP-forespørsler, er API-et er den eneste komponenten som er avhengig av dette, og hadde vært vanskeligere å vedlikeholde hvis de andre komponentene også hadde slike avhengigheter.



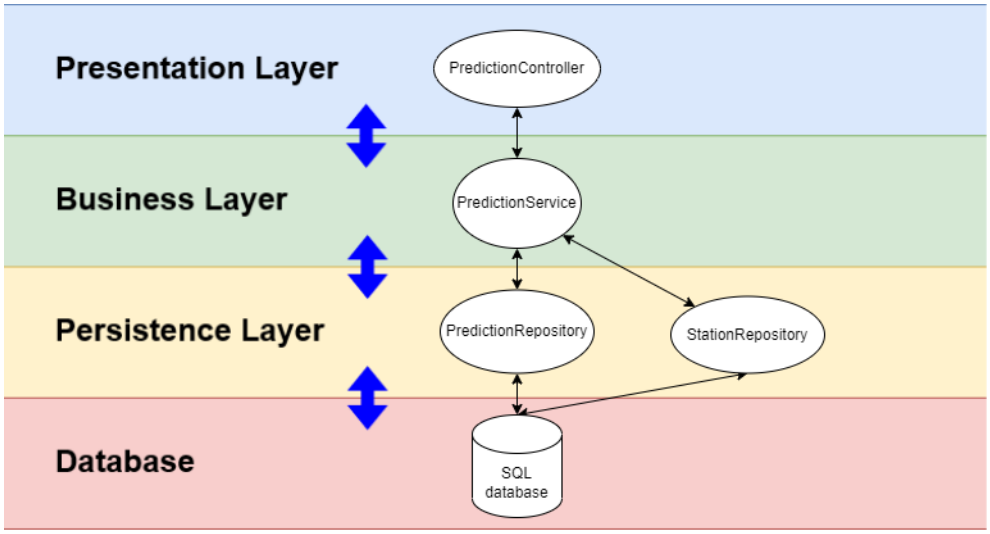
Figur 6.1: Overordnet arkitektur for modulene

6.2 Design API

Når arkitekturen for API-et ble valgt, var det fokus på å ha en struktur hvor Single Responsibility-prinsippet ble fulgt, som sier at «En klasse burde kun ha et ansvarsområde, et enkelt formål.» [28, s. 95]. Arkitekturen følger controller-service-repository-mønsteret, hvor controller-laget eksponerer grensensnittet for ekstern interaksjon, service-laget håndterer forretningslogikken og repository-laget er ansvarlig for selve datahåndteringen. Ved å ha disse lagene, kan hvert lag testes enkeltvis ved å etterlikne (engelsk: mock) de andre lagene det er avhengig av. I tillegg er det enklere å bytte ut komponenter og øke gjenbruk. For API-er som er avhengig av spørringer til og fra databaser, som dette API-et er, er dette svært viktig for å kvalitetssikre applikasjonen med enhet- og integrasjonstester.



Figur 6.2: Eksempel på hvordan de forskjellige lagene kan samhandle



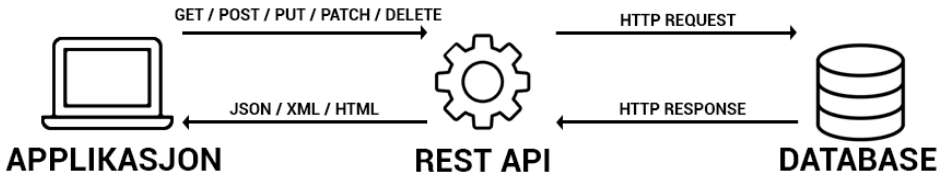
Figur 6.3: Arkitektur relatert til prediksjon i API-et

Operasjoner

REST API gjør det lettere å utvikle et API, siden det inneholder all CRUD funksjonalitet. REST API bruker også HTTP metoder, siden de er passer all CRUD funksjonalitet. APIet bruker altså CRUD for å kommunisere med appen og HTTP for å kommunisere med databasen. Under er det gitt en forklaring på de forskjellige operasjonene i HTTP og hva som tilsvarer det samme i CRUD [29].

HTTP	CRUD	Forklaring
GET	Read	Operasjon til å hente data fra databasen
POST	Create	Operasjon for å legge til ny data i databasen
PUT	Update/Replace	Operasjon for å oppdatere en allerede eksisterende ressurs
PATCH	Modify	Operasjon for å delvis oppdatere en ressurs
DELETE	Delete	Operasjon for å slette data fra databasen

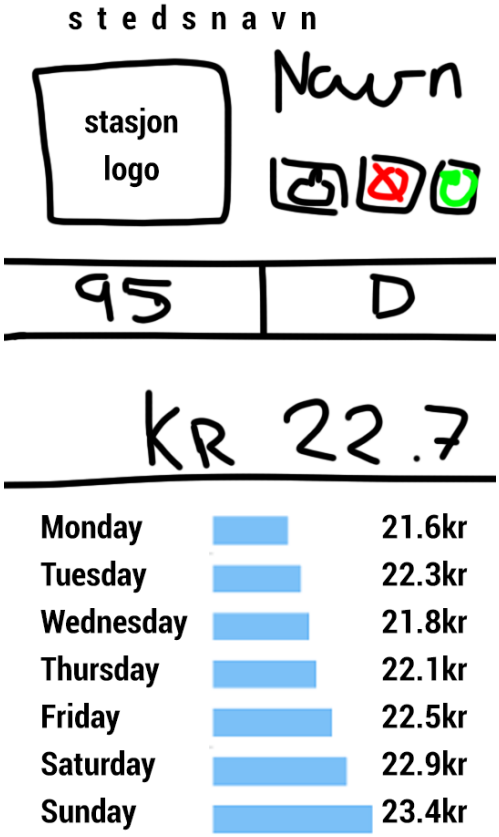
De operasjonene som er mest relevante i dette tilfellet er GET og POST. Mesteparten av funksjonaliteten innebærer å legge til nye prediksjoner for drivstoffpriser i databasen (POST), eller å hente prediksjoner fra databasen (GET). PUT blir også nyttig for å oppdatere predikert pris på en prediksjon. Hver dato vil få en oppdatering i prediksjon hver 24. time, og prediksjonene blir justert mer nøyaktig. Det er lettere å predikere prisen for en dato som er nærme i tid.



Figur 6.4: Modell for hvordan APIet jobber med de andre komponentene

6.3 Brukergrensesnitt

Et brukergrensesnitt eksisterer for å gi brukeren en kontaktflate som gjør det mulig å kommunisere med backend. Hensikten til et brukergrensesnitt er å gjøre appen mer brukervennlig, forståelig og oversiktlig for brukeren. En gjennomsnittlig person ser på en applikasjon i kort tid før personen har gjort seg en mening om appen, man må altså fange interessen til brukeren det første sekundet. Det er derfor viktig at kommunikasjonen i appen er tydelig og lett å forstå, slik at det ikke er noe tvil om hva all informasjonen og alle knappene i brukergrensesnittet er til. Samtidig er det viktig at appen er visuelt tilfredstillende, meningsfullt design og at det føles intuitivt hvor man finner de forskjellige funksjonalitetene. Under vises en tidlig modell av hvordan vi tenkte å fremstille prediksjonene i brukergrensesnittet.



Figur 6.5: Tidlig skisse av brukergrensesnittet med prediksjoner

Siden det har blitt utviklet en funksjon til en allerede eksisterende app, måtte designet på brukergrensesnittet stå i stil til resten av appen. Dette er fordi det er viktig at designet i applikasjonen er konsistent, slik at det føles sammenhengende. Av den grunn ble det ikke tatt så mange valg med tanke på designet til appen, men i større grad prøvd å lage brukergrensesnitt som står i stil til resten av applikasjonen. Designet på brukergrensesnittet var derfor preget av begrensninger. Likevel har Donald Norman's prinsipper for design blitt brukt [30] [31].

1 Synlighet

Det er viktig at alle funksjonalitetene er tydelig synlige og at det er selvforklarende hva som er funksjonaliteten. Grunnen til at dette er viktig er at appen skal være intuitiv og enkel å bruke. Man skal ikke trenge å bli forklart hvordan det fungerer før man bruker appen. Siden en bruker naturlig trykker inn på en stasjon for å se en pris, vil prediksjonene være synlig og lett tilgjengelig for brukeren.

2 Tilbakemeldinger

Tilbakemeldinger er for å fortelle brukeren at de har gjort en handling, det kan være en lyd, et lys, tekst på skjermen, eller et annet signal. Det handler om god kommunikasjon mellom applikasjonen og brukeren. Det aller viktigste er å gi brukeren tilbakemelding når applikasjonen tar input fra brukeren. Det trenger ikke være avansert i det hele tatt, så lenge det blir kommunisert på en god måte til brukeren at endringen i appen er registrert. Et eksempel fra DrivstoffAppen er når det prisen manuelt oppdateres på en stasjon, så får man en melding tilbake som sier *prisen er blitt oppdatert*.

3 Råd

I Donald Norman's seks prinsipper, handler råd om forholdet mellom hvordan noe ser ut og hva det brukes til. En knapp burde se ut som en knapp, og det burde være intuitivt at det er en knapp. For eksempel, inne på en stasjon i appen er det tydelig at det store rød feltet hvor det står *oppdater*, er en knapp som man kan oppdatere prisen på.

4 Mapping (Overføring)

Mapping eller Overføring forteller noe om sammenhengen mellom kontrollere og deres effekt. Det må være så at det er logisk og intuitivt. Det beste eksemplet på mapping er volumkontrollere, knappen for øking av volumet er alltid over knappen for senking av volumet. Et eksempel på dette fra appen er at man drar skjermen til siden for å bytte mellom dieselpriser og bensinpriser.

5 Begrensninger

Begrensninger er grensen til en interaksjon eller brukergrensesnitt, noen begrensninger er åpenbare, for eksempel skjermstørrelse. Man kan ikke ha noe på skjermen som er større en skjermstørrelsen. det er et eksempel på en fysisk begrensning. De fleste andre begrensninger er mer nyansert og abstrakte. En begrensning vi fikk av DrivstoffAppen var at prediksjonene kunne maksimalt ha to desimaler. En annen type begrensning fra appen er at det fungerer bare i Norge, den dag i dag. Den har altså en funksjonalitet som bare fungerer innenfor de norske grensene.

6 Kontinuitet

Kontinuitet handler om at den samme handlingen må forårsake den samme reaksjonen, hver eneste gang handlingen blir utført. Dette gjelder både for funksjonaliteter og interaksjoner, men også på design, detaljer og alt brukergrensesnitt. Hvis det ikke er kontinuitet i appen, vil den være lite brukbar. Hvis man ser for seg at *update-knappen* i appen gjør at appen lukkes. Altså, like elementer bør ha

like funksjoner, og oppførselen deres bør være bestemt. I design delen, er det mer fordi at brukeren skal få inntrykket av at det er en godt sammensatt app, med et tema. Hvis alle bokser og layout skal være forskjellig farge og størrelse på, blir det nok ikke noe særlig god applikasjon.

Kapittel 7

Implementasjon

I dette kapitlet vil implementasjonsprosessen bli beskrevet. Ettersom det er definert tre forskjellige moduler i den overordnede arkitekturen, deles denne inn i tre deler: utvikling av API, utvikling av brukergrensesnitt og utvikling av maskinlæringsmodul.

7.1 API

Utvikling av API

Utvikling av REST API var noe av det første som skjedde i utviklingsperioden. Dette ble prioritert ettersom de resterende modulene er avhengig av dette grensesnittet. Innledningsvis trodde gruppen at et eget API skulle utvikles, men det viste seg at et eksisterende API skulle utvides. Dette førte til at gruppen måtte sette seg inn i en eksisterende arkitektur, og fullførte denne aktiviteten litt senere enn planlagt.

Grunnet arkitekturen brukt i det gamle API-et, viste det seg å være vanskelig å teste, og av tidshensyn valgte gruppen å lage et ekstra API i tillegg, som gjorde det enklere å teste og vedlikeholde. Dette API-et følger controller-service-repository-arkitekturen, som er svært vanlig i applikasjoner som bruker Spring Boot. I tillegg til å være enklere å teste, er repository-laget sin funksjonalitet abstrahert bort og implementeres av Spring Boot automatisk ved hjelp av *dependency injection*. Dette gjør at alle grunnleggende CRUD-operasjoner er implementert, i tillegg kan mer kompliserte spørringer defineres i et grensesnitt, hvor bare SQL-query, parametre og returtype blir spesifisert. Dette er tidsbesparende, samt reduserer antall tester som må skrives. Hensikten med denne endringen var å gi et forslag på hvordan et testbart og skalerbart API kan se ut, skulle det eksisterende API-et bli refaktorert.

Det er også laget en Docker-fil for applikasjonen. Motivasjonen bak dette er å fasilitere en enkel deployering til ulike systemer ved hjelp av virtualisering, samt tilrettelegge for orkestrering av konteinere, slik som Kubernetes eller Docker Compose.

For dokumentasjon på bruk, og hvilke endepunkter som er tilgjengelige i API-et, henvises det til vedlegg D.

Kodeliste 7.1: SQL kommando for å opprette databasetabell for prediksjoner

```
CREATE TABLE ‘predictionlog‘ (
  ‘station_id‘ int NOT NULL,
  ‘prediction_date‘ date NOT NULL,
  ‘gas_type‘ varchar(45) NOT NULL,
  ‘price_lower_margin‘ float DEFAULT NULL,
  ‘price_upper_margin‘ float DEFAULT NULL,
  ‘price‘ float DEFAULT NULL,
  PRIMARY KEY (‘station_id‘,‘prediction_date‘,‘gas_type‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

ALTER TABLE ‘predictionlog‘
ADD FOREIGN KEY (‘station_id‘) REFERENCES station(‘ID‘);
```

7.2 Brukergrensesnitt

Som nevnt i avsnitt 1.4, vil det kun bli lagd brukergrensesnitt til Android, og ikke iOS. Drivstoffappens kommende visuelle overhaling gjør at utviklingen av brukergrensesnitt blir bare for oppgavens skyld, og ikke noe Drivstoffappen vil bruke i appen sin.

Designet på brukergrensesnittet var så og si forhåndsbestemt, da det må stå i stil til resten av applikasjonen. Drivstoffappen har et tema i designet sitt, og det eneste som ga mening var å bruke dette temaet i brukergrensesnittet som blir utviklet for fremvisning av predikasjoner. Slik at det blir et kontinuerlig design på hele appen, Av visuell fremvisning er det blitt utviklet en tabell og et søylediagram for predikerte drivstoffpriser. Det kan leses av predikert pris direkte fra tallverdiene i tabellen, eller bla til høyre for en mer visuell sammenligning av prisene i søylediagrammet

Utvikling av brukergrensesnitt

Oppbyggingen til layoutfilen til stasjonsvisningssiden er blitt bygd opp på nytt. Dette ble gjort fordi det var vanskelig å få inn en switch tab med både tabellen og søylediagrammet inni, sånn som brukergrensesnittet var bygd opp før. Det ble da byttet fra et vertical layout til et constraint layout. Måten dette ble gjort på var å opprette en ny layoutfil, også forsøke å lage den så lik som mulig til den originale layoutfilen. Når den nye layoutfilen så mer eller mindre identisk ut til den originale filen, ble den originale filen erstattet. Etter det ble switch taben lagt inn i aktiviteten, og pristabellen og søylediagrammet fikk hver sin plass i switch taben.

Pristabell Pristabellen ble utviklet før ideen om en mer grafisk fremstilling av prisene dukket opp. Tabellen er bygd opp av vertical layouts og horizontal layouts, som er plassert inni hverandre. Det ble prøvd mange forskjellige fonter, fontstørrelser, tekstplassering og farger i utviklingen. Etterhvert var designet på plass. Prisen i tabellen er en variabel som viser prediksjoner fra databasen, og som også

holder på en dato for den spesifikke prediksjonen. Datoen blir brukt til å fremvise hvilken ukedag prisen gjelder. Den prisen i tabellen som har lavest verdi, alstå den prisen som er billigst, blir markert med grønn farge. Dette er fordi det kjapt kan undersøkes hvilken av de kommende dagene det er billigst drivstoff. Det var tenkt at diesel priser skulle markeres med grå, og bensinpriser med grønn. Dette ble droppet da den billigste prisen ble markert med grønn farge.

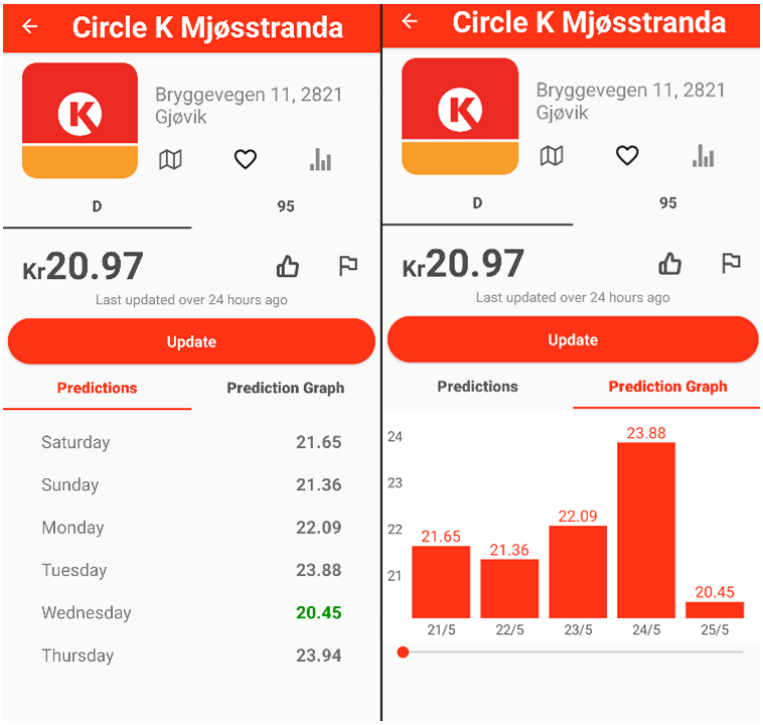
Grafisk fremstilling Etterhvert kom ideen om en mer grafisk fremstilling av prisene, og det ble bestemt at prediksjonene skulle bestå av tabellen og en grafisk fremstilling, i form av en graf eller et diagram, som enkelt kunne byttes mellom. Først ble det utviklet en graf, som viste prediksjonene i form av punkter på et kart hvor x-aksen var datoene og y-aksen var prisene.

Til slutt ble det konkludert med at et søylediagram var en bedre grafisk fremstilling. Søylediagrammet som er blitt utviklet bruker et bibliotek som heter *MPAndroidChart*, som var det samme biblioteket som grafen kom fra. Det var ved bruk av dette bilioteket enkelt å bytte fra grafen til et søylediagram, samt å style diagrammet så det passet temaet til appen. *MPAndroidChart* tar prediksjoner og fremstiller de grafisk i søylediagrammet.

Ikoner Det ble endret størrelse på alle ikonene som blir brukt, dette fordi selv om størrelsen er relativ og de blir skalert så de er like store, er kan man fortsatt se forskjell i proporsjonene på ikonet. Det er også god praksis å ha en standardisert form av ikonene som skal brukes.

Overflødig informasjon Tittelen på Stasjon-aktiviteten var satt til å vise navnet til stasjonen, men det sto allerede øverst på aktiviteten. så resultatet ble at det sto navnet på stasjonen to ganger helt på toppen av aktiviteten. Det ble valgt å fjerne tittelen, slik at navnet bare står der en gang. Det var dårlig kommunikasjon at det sto der to ganger, og det kunne ha skapt forvirring. I tillegg ser det mer visuelt tilfredstillende og stilrent ut med bare et navn.

Lokalisering Som tidligere spesifisert, var det et krav om å lokalisere teksten til engelsk og norsk, gjerne også dansk og svensk. Den ferdige løsningen er lokalisert til alle de skandinaviske språkene, i tillegg til engelsk. Drivstoffappen har i etterkant utvidet stasjonsområdet til både Danmark og Sverige, og det er derfor hensiktsmessig å oversette til disse språkene.



Figur 7.1: Endelig versjon av brukergrensesnitt

7.3 Maskinlæringsmodul

Datasett

Hovedtyngden av dataen brukt i treningen av maskinlæringsmodulen baserer seg på brukeres innrapporterte priser på forskjellige bensinstasjoner. Under treningen ble fortrinnsvis data fra bensinstasjoner som har blitt mest registrert brukt. Før dataen har blitt brukt, har det vært behov for å bearbeide denne i noen steg:

- Konvertere Unix-tid til datoformat
- Fjerne feilregistreringer, det vil si ekstremt høye eller lave priser
- Omdanne datasett til daglig frekvens, data på samme dag tas gjennomsnitt på
- Bruke lineær interpolasjon til å erstatte datoer som mangler registeringer

Datasettet har blitt delt opp i trenings-, validerings- og testsett, hvor 70 % er treningsdata, 20 % er valideringsdata og 10 % er testdata. Det finnes alternativer til andre fordelinger, for eksempel 60/20/20, men ble valgt bort på grunn av at modellen avhenger av en god tilpasning av data nær nåtid (testsettet består av den nyligste dataen). Datasettet har også blitt normalisert, ved å subtrahere hver verdi med gjennomsnittet og dele på standardavviket til dataen. Motivasjonen bak dette er å balansere hver variabel sin påvirkning, slik at høye verdier ikke har en skjev påvirkning.

I et forsøk på å hjelpe modellen med å finne karakteristikker i datasettet, har forskjellige metoder blitt utprøvd, som går under *feature engineering*. Alle dataoppføringer har en boolsk verdi som sier om datoen er helg eller ikke. I tillegg har nummeret på ukedagen (0-6) blitt hentet ut. Andre karakteristikker har også blitt utprøvd, slik som *lag features*, som forskyver data n antall dager, eller *moving averages*, som kan eksempelvis være gjennomsnittspris siste 3 dager. Disse ble utelatt ettersom de viste seg å ha en neglisjerbar effekt på feilraten, men kan muligens forbedre modellen med andre verdier.

I et forsøk på å øke mengden data som modellen benytter seg av, har oljeprisen per fat på nordsjøolje blitt utprøvd, som er hentet fra indeksen Brent Crude. [1] Fra dette kunne informasjon om åpnings- og sluttkursen på olje hentes, i tilegg til volum, minimal og maksimal pris. Grunnen til at dette ble utprøvd er på grunn av korrelasjonen mellom drivstoffpriser og oljepriser. Under treningen viste det seg at feilraten ble i liten eller ingen grad forbedret, og ble derfor tatt bort fra den endelige versjonen. Selv om resultatene ikke ble forbedret under treningen er det ikke utenkelig at informasjonen kan være til nytte ved å bruke *moving averages* eller *lag features*.

Utvikling av maskinlæringsmodell

Innledningsvis var det fokus på å få på plass en grunnleggende modell som kunne gi enkle prediksjoner. Her ble Prophet, et prediksjonsrammeverk i Python brukt som et utgangspunkt. Videre ble forskjellige kunstige nevrale nettverk utprøvd, i håp om å forbedre resultatet fra den enkle modellen.

Utviklingen av maskinlæringsmodellene har i stor grad basert seg på en modifisert versjon av en Jupyter Notebook laget av Tensorflow. Grunnen til at denne har blitt brukt er fordi dette sparer betraktelig tid ved å introdusere metoder for å preprosessere data og gjøre det klart for trening av modeller. I tillegg blir en klasse for å dele dataen inn i såkalte «vinduer» definert. Datavinduer gjør det mulig å sende en rekke data inn i kronologisk rekkefølge, samt justere hvor mange datapunkter som blir sendt inn [32].

Etter å ha lest seg opp på ulike nevrale nettverk, var spesielt LSTM og CNN aktuelle kandidater. Alle modeller brukt til læring bruker *Mean Squared Error* som tapsfunksjon, og *Mean Absolute Error* som evalueringsfunksjon. Hensikten med dette er å forsterke store feil under trening, og at feilraten er intuitiv når resultatene fremlegges.

For å sammenlikne forskjellige modeller, har en baseline-modell blitt brukt til å sammenlikne. Denne baseline-modellen baserer seg på antagelsen at neste dags pris er lik den forrige. Med andre ord gjentas forrige dags pris. Naturligvis vil denne modellen verken lære noe av datasettet, eller gi spesielt gode prediksjoner. Derfor burde de mer avanserte modellene slå denne med god margin.

I tillegg til denne svært enkle modellen, har det blitt implementert en enkel lineær modell, som er en lineær transformasjon mellom inngangsverdiene og ut-

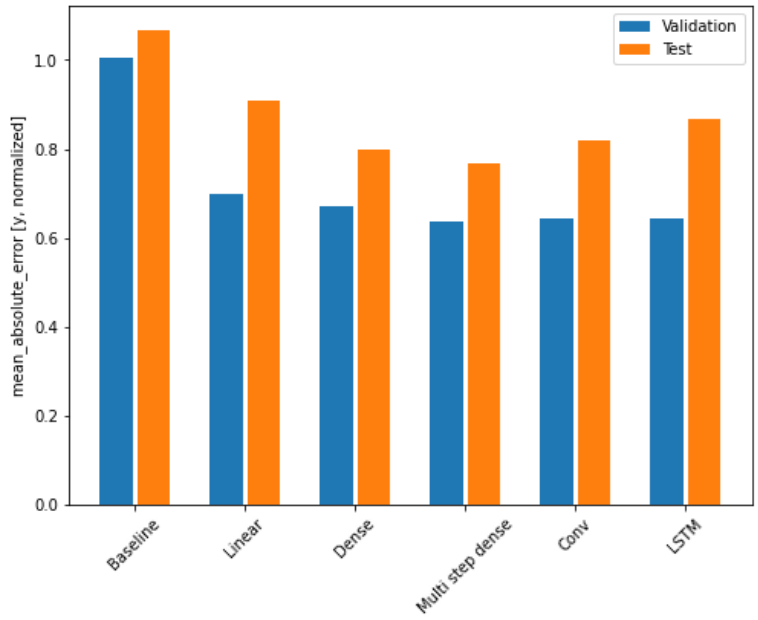
[1] `https://www.nasdaq.com/market-activity/commodities/bz:nmx/historical`

gangsverdien, hvor det i stor grad bare er vektene mellom inngangsnodene og utgangsnoden som justeres. Denne modellen kan ikke tilegne seg ikke-lineære sammenhenger, og er den enkleste trenbare modellen. De mer komplekse modellene burde også kunne slå denne modellen.

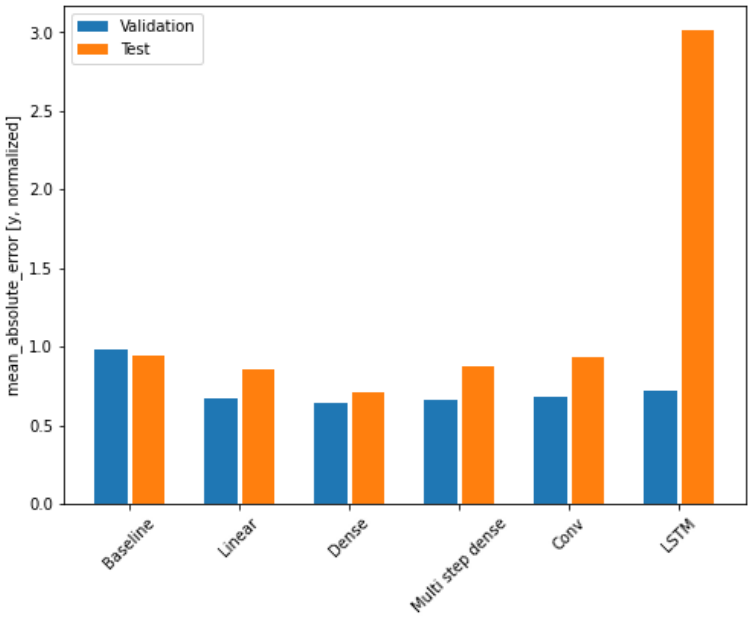
Videre har det blitt utviklet en såkalt dense-modell, som vil si en modell med fullt tilkoblede lag. Dense-modellen skiller seg fra den lineære modellen ved at den har skjulte lag mellom inngang- og utgangslaget, samt at den introduserer aktiveringsfunksjoner. Generelt sett har denne prestert bedre enn den lineære modellen.

CNN-modellen som har blitt utviklet likner på dense-modellen, men har et endimensjonalt konvolusjonslag før dense-lagene. Formålet med dette er å fange opp karakteristikker ved dataen, og for dette datasettet har det gjort modellen enkel å trene. På bakgrunn av god prestasjon av CNN-modellen, ble denne valgt til å forbedres ytterligere.

Før maskinlæringen startet, var det spesielt store forventninger om at LSTM ville prestere bedre enn andre arkitekturer. Dette fordi den både har korttidsminne og langtidsminne, som i teorien burde være aktuelt i denne problemstillingen. Det viste seg at denne modellen slet med overtilpasning når den ble evaluert med testsettet, som kan ses på figur 7.3. Når den nyligste dataen ble tatt bort, presterte den omtrent jevngodt med de andre modellene. Forskjellige hyperparametre har hatt liten effekt på feilraten, og sannsynligvis vil LSTM passe bedre for større datasett.



Figur 7.2: Valideringss- og testfeil for generiske modeller før 24. februar

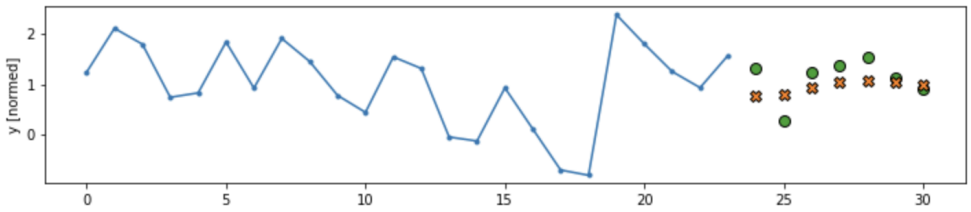


Figur 7.3: Valideringss- og testfeil for generiske modeller etter 24. februar

For å optimalisere hyperparameterne, har Optuna blitt brukt. I treningen av modellen, har det blitt kjørt 100 iterasjoner, også kalt *trials*, hvor forskjellige hyperparametre har blitt testet. Hver trial tester automatisk ut forskjellige hyperparametre. I utviklingen er det i hovedsak antall dense-lag og nevroner som har blitt justert.

Kodeliste 7.2: Utdrag av Optuna-studie

```
n_layers = trial.suggest_int("n_layers", 1, 6)

model = tf.keras.Sequential()
model.add(tf.keras.layers.Lambda(lambda x: x[:, -CONV_WIDTH:, :]))
model.add(tf.keras.layers.Conv1D(256, activation='relu', kernel_size=(CONV_WIDTH)))


early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
    patience=3,
    mode='min')

    for i in range(n_layers):
        num_hidden = trial.suggest_int(
            "n_units_l{}".format(i), 2, 128, log=True)

        model.add(
            tf.keras.layers.Dense(
                num_hidden,
                activation='relu'
            )
        )
```



Figur 7.4: CNN-modell sin tilpasning for 7 dager for et tilfeldig vindu

Resultater

Resultatene under trening av maskinlæringsmodell følger under. Resultatene er fremsatt som en tabellarisk sammenlikning mellom de forskjellige arkitekturene. Feilraten er oppgitt som normalisert *Mean Absolute Error*.

De forskjellige modellene sin prestasjon er testet på henholdsvis 1, 5 og 7 dager frem i tid. Jevnt over kan det ses at CNN-modellen utkonkurrerer de andre modellene, samt at LSTM-modellen sliter med høy generaliseringsfeil.

Modell	**Normalisert MAE Valideringssett**	**Normalisert MAE Testsett**
Baseline	0.963	0.943
Lineær	0.774	1.042
Dense	0.683	1.110
CNN	0.680	0.543
LSTM	0.809	3.003

Tabell 7.1: Resultater for prediksjon en dag frem i tid

Modell	**Normalisert MAE Valideringssett**	**Normalisert MAE Testsett**
Lineær	0.817	1.297
Dense	0.807	1.266
CNN	0.727	0.762
LSTM	1.288	4.897

Tabell 7.2: Resultater for prediksjoner fem dager frem i tid

Modell	Normalisert MAE Valideringssett	Normalisert MAE Testsett
Lineær	0.806	1.269
Dense	0.832	1.381
CNN	0.738	0.886
LSTM	1.325	5.017

Tabell 7.3: Resultater for prediksjon syv dager frem i tid

Utvikling av maskinlæringsmodul

For å sette den ferdige arkitekturen ut i produksjon, har det blitt utviklet en enkel modul som kan predikere drivstoffpriser fra stasjoner med tilstrekkelig data ved faste intervaller. Denne modulen trener en modell for hver stasjon og gjør en prediksjon basert på data fra de 24 siste dagene.

Sannsynligvis vil noen av modellene kunne predikere priser for andre stasjoner, men dette har ikke blitt undersøkt av tidshensyn. Spesielt vil dette gjelde stasjoner med liknende sannsynlighetsfordeling.

Kapittel 8

Testing og kvalitetssikring

8.1 Testing

Backend

I prosjektet har det vært fokus på at backend-funksjonaliteten skal være robust. For økt kvalitetssikring har det vært skrevet enhet/integrasjonstester som gjør at koden har 100 % testdekning.

Som testdatabase har H2[1] blitt brukt, som er en database som lever i hovedminnet. Denne er konfigurert til å initialiseres hver gang testene blir kjørt og nullstilt i etterkant. Dette gjør det enkelt å skrive enhetstester, ettersom det er samme utgangspunkt ved hver kjøring.

For å teste hvert lag, har Mockito[2] blitt brukt til å etterlikne responsen fra hver avhengighet hvert lag har. Spring Boot sitt testrammeverk har blitt brukt til å initialisere avhengighetene ved hver test ved hjelp av *dependency injection*. Det har også blitt brukt til å etterlikne REST-kontrolleren, slik at HTTP-forespørsler kan simuleres.

Som følge av at hvert lag testes en og en, er naturlig å tenke at lagene kanskje ikke fungerer sammen. Ettersom repository ikke har noen avhengigheter som må etterliknes, vil service-laget fungere dersom repository fungerer og controller-laget dersom service-laget fungerer. På denne måten sikres det at lagene fungerer sammen.

Kodeliste 8.1: Applikasjonskonfigurasjon for testing av API

```
spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.url=jdbc:h2://mem:db;DB_CLOSE_DELAY=-1
spring.datasource.username=sa
spring.datasource.password=sa
spring.datasource.driver-class-name=org.h2.Driver
spring.jpa.show-sql=true
```

[1] H2: `https://www.h2database.com/html/main.html`

[2] Mockito: `https://site.mockito.org/`

Brukertesting

Gjennom utvikling av frontend har gruppen fått kontinuerlige tilbakemeldinger på brukergrensesnittet. Dette har i hovedsak skjedd gjennom møter med oppdragsgiver og Drivstoffappen. Med andre ord har det vært en uformell evaluering av brukerens opplevelse av grensesnittet. I utgangspunktet var det planlagt mer omfattende brukertesting, men av hensyn til tid ble dette ikke gjennomført. Dette ble prioritert bort ettersom at løsningen krever lite brukerinteraksjon, samt at handlingsrommet for å gjøre store endringer i designet er forholdsvis begrenset.

Gruppen er likevel klar over at dette er en svakhet i implementasjon i brukergrensesnittet, og ville sannsynligvis prioritert dette høyere hvis denne funksjonaliteten skulle implementeres på den nye versjonen av Android-appen.

8.2 Kvalitetssikring

Dokumentasjon

API-ets tilgjengelige endepunkter og spørringsparametere er laget dokumentasjon på. I tillegg er all produksjonsklar kode kommentert, slik at alle ikke-trivielle funksjoner er kommentert, samt ytterligere forklaringer i koden der det er behov for dette.

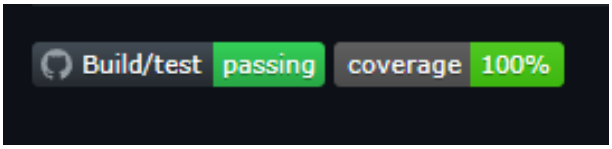
Kontinuerlig integrasjon

For å garantere at koden kompilererer og alle testene er suksessfulle, samt generere rapport om testdekning, brukes Github Actions til å lage arbeidsflyter som automatisk sjekker dette ved å kalle Gradle-kommandoer. Disse arbeidsflytene blir trigget hver gang en pull request blir opprettet, samt ved push til master-grenen i kodebasen. Dette betyr at man kan være sikker på at koden er robust før grener flettes sammen.

Kodeliste 8.2: Arbeidsflyt for build og test av API

```
name: Build/test

on:
  push:
    branches: [ master ]
  pull_request: {}

permissions:
  contents: read

jobs:
  build:

    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ ubuntu-latest, windows-latest, macOS-latest ]
```

```
    steps:
    - uses: actions/checkout@v3
    - name: Set up JDK 11
      uses: actions/setup-java@v3
      with:
        java-version: '11'
        distribution: 'temurin'
    - name: Grant execute permission for gradlew
      run: chmod +x gradlew
    - name: Build with Gradle
      uses: gradle/gradle-build-action@0d13054264b0bb894ded474f08ebb30921341cee
      with:
        arguments: build
```



Figur 8.1: Skilt generert av en Github Actions arbeidsflyt som viser at build er suksessfull og testdekning er 100 %

Kapittel 9

Diskusjon

9.1 Utviklingsprosess

Scrum som systemutviklingsmodell har etter gruppens syn fungert godt og vært hensiktsmessig å bruke. Utviklingen har i stor grad vært inkrementell med stadige forbedringer, med jevnlige demonstrasjoner og statusoppdateringer, og smidig metodikk har lagt godt til rette for dette. Siden kravene og sluttmålet var relativt vagt, var mer kortsiktig planlegging logisk. Ved å ha daglige statusmøter internt, og møter på ca. slutten av hver sprint, har planlegging av nye sprinter og statusoppdateringer blitt mer overkommelig.

Gruppen ble opprinnelig enige om å bruke 30 timer i uka frem til prosjektinnlevering, som skulle bli omtrent 500 timer totalt. Dette viste seg å være vanskelig på grunn av balansering av tidsbruk med annet fag. Dette resulterte i at flere timer måtte legges inn de siste to månedene. I retrospekt var det urealistisk å bruke 30 timer i uka mens arbeidsmengden til et annet fag var stor, beregningen av tidsbruk per uke kunne dermed vært bedre. Med bedre planlegging kunne tidsbruken vært jevnere. Registrering av tidsbruk har foregått i Toggl Track, og har fungert veldig godt. Gruppen har registert timebruk i relativt generelle kategorier, som kanskje kunne vært mer konkret, for eksempel kunne utvikling blitt brutt ned til de tre forskjellige modulene.

Opprinnelig var det planlagt at gruppen skulle skrive rapport kontinuerlig fra prosjektstart til prosjektslutt, med økende intensitet mot slutten av prosjektperioden. Denne planen ble fulgt i mindre grad enn ønsket, og førte til større arbeidsbelastning knyttet til rapport mot slutten. Dette er en klart forbedringsmoment, ettersom kontinuerlig skriving hadde gjort prosessen mer effektiv ved å fortløpende skrive ned ting gruppen kommer på.

Issue-brettet i ZenHub har bidratt til å få bedre oversikt internt i gruppen. En utfordring som oppsto i forbindelse med dette var å bryte visse kompliserte oppgaver ned i mer konkrete deloppgaver. Maskinlæringskortene var spesielt vage, ettersom gruppen særlig innledningsvis få formeninger om hvordan denne oppgaven skulle løses. Forøvrig har kortene gjort det enkelt å se hva det andre gruppemedlemmet jobbet på, og har bidratt til bedre kommunikasjon.

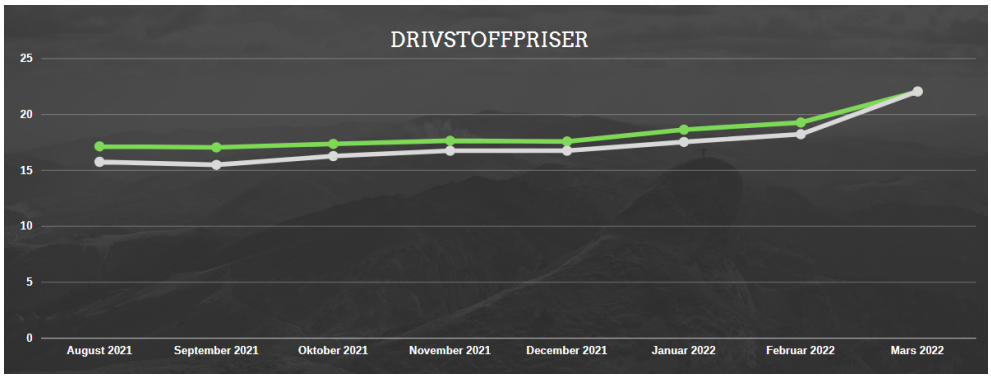
9.2 Maskinlæring

Brukeropplevelse

For at brukeren av applikasjonen skal ha en positiv brukeropplevelse av prediksjonfunksjonaliteten, må nøyaktigheten på prediksjonene være gode. Det antas at en feilaktig prediksjon vil virke mer negativt inn på brukeren enn en riktig prediksjon. Per nå er prediksjonene forholdsvis gode tilnærminger, men klarer i liten grad å forutse plutselig opp- eller nedganger i pris. Slike endringer er vesentlige for brukeren å få informasjon om, ettersom dette er dager hvor det enten er mer eller mindre fordelaktig å fylle drivstoff.

Ytre faktorer

Grunnet internasjonale hendelser, deriblant Russlands invasjon av Ukraina, har drivstoffprisene i Norge blitt generelt høyere, men også mer volatile. Det gjør at utviklingen grafen til drivstoffprisen blir mer uforutsigbar, som vil si at det blir vanskeligere å predikere prisen på drivstoff. I vanlige tilstander vil . Det er utallig mange faktorer som spiller inn på kursen til drivstoffprisen, og mange av disse er ikke mulig å kvantifisere. Derfor må vi bruke de mønstrene som ses som verktøy for predikering av prisene. Under kan det ses en graf over drivstoffprisen de siste 8 månedene. Denne grafen blir brattere og brattere i de første månedene i 2022, mens i perioden August - Oktober 2021 er grafen mer stabil.

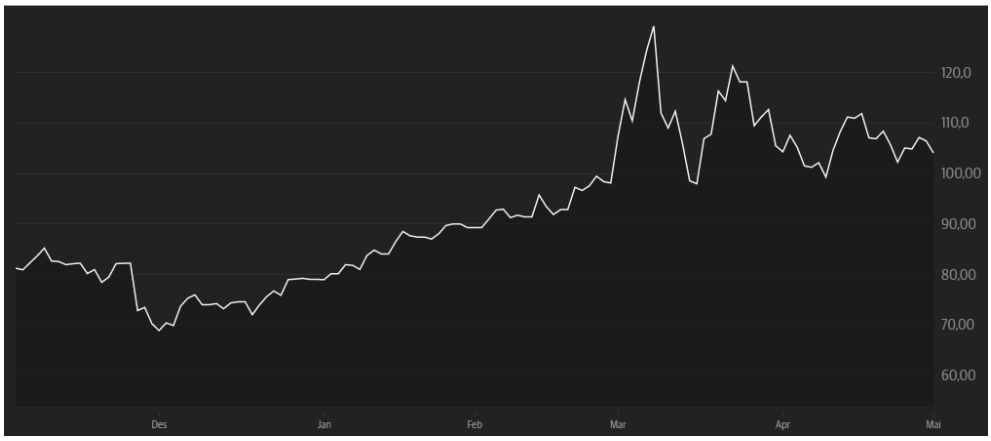


Figur 9.1: Graf som viser drivstoffpris for 95 og Diesel de siste 8 månedene [33]

Nærmere forklart gjør dette at testdataen har vist et mønster som treningsdataen i mindre grad. Det vil si at ML-modulen som har blitt utviklet er blitt trent på de mest nylige trendene. Av denne grunn har det vært en utfordring å lage en modell som får en akseptabel score på testdataen. Dette har blitt et mindre problem et par måneder senere, fordi det foreligger mer treningsdata som har liknende mønstre.

Under vises en graf over prisen på råolje. Det kan ses på figuren at grafen stiger betraktelig i månedsskifte mellom februar og mars. I perioden mars og april er

grafen ustabil og volatil, og det har vært vanskeligere å predikere i denne perioden i forhold til for eksempel perioden mellom 1. desember og 31. januar, hvor grafen er mer stabil.



Figur 9.2: Graf som viser kursen for råolje de siste 6 månedene [34]

Det blir vanskeligere å predikere når grafen er volatil, fordi det er større svingninger og variasjon i prisen. Intervallet over tall som prisen kan utvikle seg til, er større når prisen er volatil. Det gir mening fordi det er flere verdier som er sannsynlig. Under vanlige tilstander vil intervallet være mindre, og det er av den logikk lettere å predikere hvilken av verdiene i intervallet som blir faktisk pris.

9.3 Videreutvikling i eksisterende prosjekt

Da vi startet oppgaven fikk vi tilgang til oppbevaringsstedet for all kildekode for DrivstoffAppen, og vi måtte på selvstendig vis sette oss inn i et allerede eksisterende utviklingsmiljø med en stor kildekode. Siden det ikke er kontaktpersonene våre i DrivstoffAppen selv som har programmert applikasjonen, og det ikke har vært noe form for kontakt eller kommunikasjon med utviklerne av applikasjonen i det hele tatt, har det vært litt tidskrevende å sette seg inn i koden og utviklingsmiljøet. Det er ikke nødvendigvis negativt, men betyr at det tok ekstra tid å sette oss inn i kildekoden til prosjektet. Det som er positivt med dette er at man får mer erfaring på å lese udokumentert kode.

9.4 Ferdigheter i prosjekt

Høyt nivå på kommunikasjon er et viktig punkt, god kommunikasjonsflyt gjør at prosjektet blir mer oversiktlig og systematisk, som igjen gjør det mer givende å

jobbe på prosjektet, og resultatet blir at flere oppgaver blir gjort, altså at effektiviteten stiger. Når målene effektiviteten er høy, blir mange oppgaver fullført. Dette gir gruppedeltagerne igjen mer motivasjon og energi, som gjør at effektiviteten stiger enda mer.

Andre ferdigheter som har vært verdifulle i oppgaven inkluderer blant annet samarbeid, tålmodighet, handlingsdyktighet og selvstendighet. samarbeid er åpentbart viktig da man jobber i et team som skal gjøre en oppgave sammen, og man må forholde seg til gruppemedlemmene sine. tålmodighet er en evne som gjør jobben mye letter. Det er mye sakte arbeid, som pirking, research og feilsøking i et prosjekt som dette, og hvis man ikke har tålmodigheten kan det bli tungt. Handlingsdyktighet er et annen evne som man burde ha i prosjekt. Det handler om å ta handling når det trengs handling. Det motsatte begrepet er handlingslammet, som da er å ikke være i stand til å ta handling når det trengs. Selvstendighet er evnen til å jobbe alene. Selv om det har vært prosjekt, og man har gruppemedlemmer, er det likevel en svært god egenskap å kunne jobbe selvstendige med oppgaver. Mange oppgaver krever kun at en person jobber på dem. For å holde effektiviteten oppe, burde man bytte mellom å jobbe selvstendig, og å jobbe i sammen på oppgaver i prosjektet.

I de aller fleste prosjekter er det en kunde, altså en part som mottar en tjeneste eller et produkt på slutten av prosjektet. Det er viktig å behandle kunden pent og med respekt. Man må prøve å være så serviceinnstilt man kan, og å gjøre som kunden ønsker. Samtidig må man bruke sine profesjonelle kunnskaper til å styre retningen på produktet/tjenesten. Poenget er at det er viktig at kunden får det slik de vil, slik at de er fornøyde med produktet eller tjenesten.

Kapittel 10

Konklusjon

10.1 Måloppnåelse

I løpet av prosjektet har gruppen klart å fremstille en maskinlæringsmodell, et API og utvidet en eksisterende Android-applikasjon til å inkludere prediksjoner. Den fleksible strukturen gjør at komponenter enkeltvis kan byttes ut ved behov. API-et har god testdekning og er skalerbart. Den beste maskinlæringsmodellen har en CNN-arkitektur og har oppnådd en *mean absolute error* på 0.543, 0.762 og 0.886 for henholdsvis 1, 5 og 7 dager frem i tid evaluert på testsettet til den mest oppdaterte stasjonen. Hvorvidt denne modellen er klar til å settes ut i produksjon må vurderes ytterligere, men legger et godt fundament for videre utvikling. Android-applikasjonen oppleves av oppdragsgiver som intuitiv, og lett å skille mellom predikert pris og faktisk pris.

Gruppen har gjennom prosjektet tilegnet seg ny kunnskap rundt maskinlæring, og kombinert dette med eksisterende kunnskap til å anvende denne kunnskapen til å lage en ny løsning på vegne av en ekstern oppdragsgiver.

10.2 Videre arbeid

På slutten av prosjektperioden har gruppen gjort seg noen tanker om hvordan løsningen kan forbedres og utvides ytterligere. Av tidsmessige hensyn har visse tilnærminger og idéer ikke blitt realisert, og denne delen beskriver noen av disse momentene.

Maskinlæring

For å forbedre maskinlæringsmodellen, kan det forsøkes å generere nye variabler basert på tidsrekkedataen ved hjelp av *feature engineering*. Eksempelvis kan dette være hvorvidt dataen er på en helligdag, gjennomsnitt basert på forbigående dager (f.eks. tre eller syv dager), som kan oppnås ved hjelp av rullende vinduer og lag features, som forsinker dataen med n antall dager. Per nå har hvorvidt det er helg og ukedag blir hentet ut.

I tillegg er det mange variabler som kan justeres som ikke har blitt utprøvd i vesentlig grad. Inputvinduet kan justeres for å inkludere flere eller færre dager. Av alternative arkitekturer som ikke er utprøvd, står Gated Recurrent Unit (GRU) som en potensiell kandidat. Denne likner LSTM-arkitekturen, men har mindre kompleksitet, og kan potensielt passe mindre datasett.

Brukergrensesnitt

Samlet sett har brukergrensesnittet funksjonaliteten den trenger for å utføre oppgaven det ble designet til. Altså fremvise prediksjonene på en brukervennlig måte. Likevel er det noen funksjonaliteter som har blitt tenkt på, men som ikke har blitt implementert av forskjellige grunner. Man får fort mange ideer om hvordan appen kan utvides når man jobber så tett på den som det er blitt gjort i prosjektet.

Det var planlagt å implementere visuell fremstilling av pris på alle stasjoner i kartet. Det er allerede implementert en markør for alle stasjonene, som prisen ville ha blitt plassert inni. Designet ville vært penere å se på, da dette bare er en enkel modell. Det måtte også ha blitt utviklet funksjonalitet så ikke boksene som viser prisen hadde lagt seg over hverandre. Dette hadde vært et stort problem der det ikke er langt i mellom stasjonene, for eksempel Oslo. Under vises det en enkel modell om hvordan funksjonaliteten ville ha sett ut

Figur 10.1: Enkel skisse av prisfremstilling i kartet

Dette har ikke blitt gjort på grunn av prioritering av oppgaver i prosjektet. men også fordi Drivstoffappen har en kommende grafisk oppdatering av hele appen. Dette ble nevnt i avsnitt 1.4. All oppdatering av brukergrensesnitt vil av den grunn være kun for oppgaven, og det ble derfor lav prioritering av denne funksjonen.

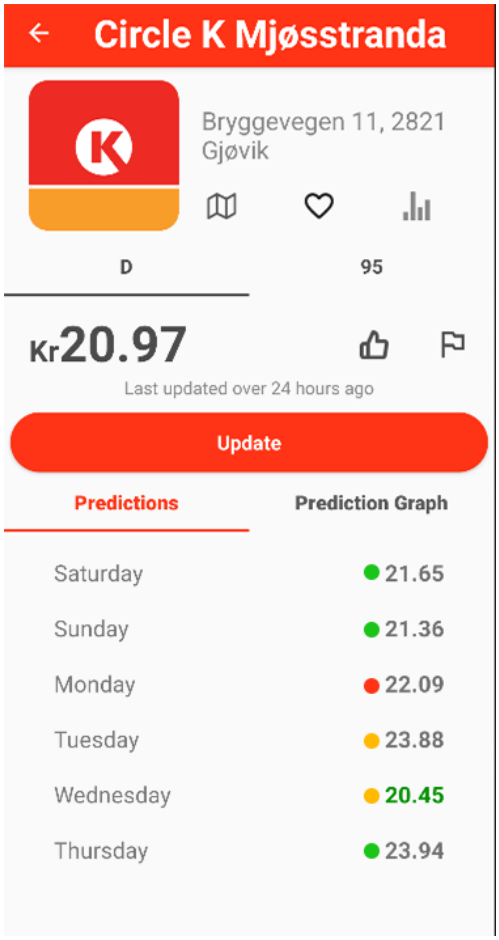
Sikkerhetsnivå på prediksjon kan bli utviklet for å fortelle noe om hvor sikker prediksjonen er, altså hvor sannsynlig det er for at den stemmer med virkeligheten. Gamle prediksjoner kan sjekkes opp mot faktisk pris den dagen for å få data som kan brukes til å forbedre systemet for gradering av sikkerhet rundt en prediksjon.

Et enkelt system for å klassifisere prediksjonene kan for eksempel være fargekoding. Her kan det brukes den svært kjente kombinasjonen grønn, gul, rød (trafikklys). Hvor grønn er sikker, gul er passe sikker, og grønn er usikker.Et mer avansert og nøyaktig system kan være i prosent. Med det menes at hver prediksjon har en sikkerhetsprosent. Hvis sikkerhetsprosenten er 90%, vil det si at det er 90 prosent sannsynlighet for at den faktiske prisen vil stemme med prediksjonen. Dette systemet er mer nøyaktig, men mindre brukervennlig.

Hvis man tar inn i betraktning at alle type mennesker skal bruke funksjonen

i appen, er nok fargekoding mer forståelig og gir god nok indikasjon til brukeren om hvor sikker prediksjonen er. Under vises en modell over hvordan sikkerhetssystemet for prediksjonene kunne sett ut.

Farge	Betydning
GRØNN	prediksjonen er sikker
GUL	prediksjonen er noe sikker
RØD	prediksjonen er usikker



Figur 10.2: Alternativt hvordan fargekoding på prediksjonene kunne blitt seende ut.

Bibliografi

[1] Skatteetaten. «Veibruksavgift på drivstoff.» (), adresse: https://www.skatteetaten.no/bedrift-og-organisasjon/avgifter/saravgifter/om/veibruksavgift/. (hentet: 24.04.2022).

[2] A. Korlyuk, G. Henriksen og B. Bleskestad. «Langt mer enn oljeprisen påvirker bensinprisene - SSB.» (), adresse: https://www.ssb.no/energi-og-industri/artikler-og-publikasjoner/langt-mer-enn-oljeprisen-pavirker-bensinprisene. (hentet: 24.04.2022).

[3] A. Tidemann og A. C. Elster, *maskinlæring – Store norske leksikon*. adresse: https://snl.no/maskinl%C3%A6ring, (hentet: 26.04.2022).

[4] M. Mohri, A. Rostamizadeh og A. Talwalkar, *Foundations of Machine Learning Second Edition*. The MIT Press, 2018.

[5] A. Tiedemann. «kunstig intelligens – Store norske leksikon.» (), adresse: https://snl.no/kunstig_intelligens. (hentet: 25.04.2022).

[6] B. Genç og H. Tunc, «Optimal training and test sets design for machine learning,» *TURKISH JOURNAL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES*, årg. 27, s. 1534–1545, mar. 2019. DOI: 10.3906/elk-1807-212.

[7] A. Tiedemann. «dyplæring – Store norske leksikon.» (), adresse: https://snl.no/dypl%C3%A6ring. (hentet: 25.04.2022).

[8] H. Dvergsdal, *nevralt nettverk – Store norske leksikon*. adresse: https://snl.no/nevralt_nettverk, (hentet: 25.04.2022).

[9] S. Sharma, S. Sharma og A. Athaiya, «Activation functions in neural networks,» *towards data science*, årg. 6, nr. 12, s. 310–316, 2017.

[10] *What are Neural Networks? | IBM*. adresse: https://www.ibm.com/cloud/learn/neural-networks, (hentet: 25.04.2022).

[11] *ReLu Definition | DeepAI*. adresse: https://deepai.org/machine-learning-glossary-and-terms/relu, (hentet: 25.04.2022).

[12] I. Goodfellow, Y. Bengio og A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[13] C. Willmott og K. Matsuura, «Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,» en, *Climate Research*, årg. 30, s. 79–82, 2005, ISSN: 0936-577X. DOI: 10.3354/cr030079. adresse: http://dx.doi.org/10.3354/cr030079.

[14] A. de Myttenaere, B. Golden, B. L. Grand og F. Rossi, «Mean Absolute Percentage Error for regression models,» *Neurocomputing*, årg. 192, s. 38–48, jun. 2016. DOI: 10.1016/j.neucom.2015.12.114. adresse: https://doi.org/10.1016%2Fj.neucom.2015.12.114.

[15] S. Bai, J. Z. Kolter og V. Koltun, «An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling,» *CoRR*, årg. abs/1803.01271, 2018. arXiv: 1803.01271. adresse: http://arxiv.org/abs/1803.01271.

[16] R. Mittelman, *Time-series modeling with undecimated fully convolutional neural networks*, 2015. DOI: 10.48550/ARXIV.1508.00317. adresse: https://arxiv.org/abs/1508.00317.

[17] X. H. Le, H. Ho, G. Lee og S. Jung, «Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting,» *Water*, årg. 11, s. 1387, jul. 2019. DOI: 10.3390/w11071387.

[18] A. Mittal, *Understanding RNN and LSTM. What is Neural Network? | by Aditi Mittal | Medium*. adresse: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e, (hentet: 27.04.2022).

[19] C. Nilstun. «prediksjon.» (), adresse: https://snl.no/prediksjon. (hentet: 28.04.2022).

[20] *Kotlin overview*. adresse: https://developer.android.com/kotlin/overview, (hentet: 02.05.2022).

[21] *Null safety in Kotlin*. adresse: https://kotlinlang.org/docs/null-safety.html, (hentet: 02.05.2022).

[22] *Go language*. adresse: https://developers.google.com/learn/topics/go, (hentet: 01.05.2022).

[23] *Understanding GitHub Actions - GitHub Docs*. adresse: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions, (hentet: 02.05.2022).

[24] *Stack Overflow Developer Survey 2020*. adresse: https://insights.stackoverflow.com/survey/2020/#most-popular-technologies, (hentet: 02.05.2022).

[25] *GPU support*. adresse: https://www.tensorflow.org/install/gpu, (hentet: 02.05.2022).

[26] *Optuna: A hyperparameter optimization framework*. adresse: https://optuna.readthedocs.io/en/stable/, (hentet: 02.05.2022).

[27] K. Schwaber og J. Sutherland. «The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game.» (), adresse: https://scrumguides.org/scrum-guide.html. (hentet: 27.01.2022).

[28] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003. adresse: http://dl.acm.org/citation.cfm?id=515230.

[29] L. Gupta. «Http metoder - RESTful API.» (), adresse: https://restfulapi.net/http-methods/#summary. (hentet: 25.04.2022).

[30] Enginess. «Principles of design.» (), adresse: https://folk.ntnu.no/baldurk/skolearbeid/MMI/6%20-%20Designprinsipper.html. (hentet: 20.05.2022).

[31] Baldurk. «Principles of design norsk.» (), adresse: https://www.enginess.io/insights/6-principles-design-la-donald-norman. (hentet: 10.05.2022).

[32] *Time series forecasting*. adresse: https://www.tensorflow.org/tutorials/structured_data/time_series, (hentet: 01.05.2022).

[33] SSB. «Drivstoffpriser- SSB.» (), adresse: https://www.ssb.no/statbank/table/09654/. (hentet: 16.05.2022).

[34] D. Investor. «oljepriser.» (), adresse: https://investor.dn.no/#!/Ravare/C1/BrentSpot. (hentet: 02.05.2022).

Vedlegg A

Prosjektavtale

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Institutt for datateknologi og informatikk
Veileder ved NTNU: Tom Røise e-post og tlf. 61135192 tom.roise@ntnu.no
Ekstern virksomhet: Headit Ekstern virksomhet sin kontaktperson, e-post og tlf.: Martin Holthe Rønningen Tlf: 97122845 E-post: martin.holthe.ronningen@headit.no
Student: Simen Refsland Fødselsdato: 28. september 1997
Student: Oscar Brøndelsbo Fødselsdato: 21. februar 1997

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	x
Prosjektoppgave	
Annen oppgave	

Startdato: 11. februar 2022
Sluttdato: 20. mai 2022

Oppgavens arbeidstittel er: Predikerte priser på drivstoff

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[1]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

x	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

[1] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

x	Oppgaven skal være offentlig

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i

denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder: Dato:
Veileder ved NTNU: Dato:
Ekstern virksomhet: Dato:
Student: Simen Refsland Dato: 28.01.2022
Student: Oscar U. Brandelsbo Dato: 28.01.2022

Vedlegg B

Prosjektoppgave

Oppdragsgiver

Oppdragsgiver:	Headit AS
Kontaktperson:	Rune Kollstrøm
Adresse:	Løvstadvegen 7, 2312 Ottestad
Telefon:	625 10 052
E-post:	rune.kollstrom@headit.no

Kontaktperson oppgave: Vegard Ølstad Dalberg, vegard.dalberg@headit.no

Bakgrunn

Headit AS utvikler unike fag- og innsiktsløsninger som hjelper deg til å jobbe enklere, og ta riktige beslutninger. Vi er et stort og innflytelsesrikt miljø med tverrfaglig kompetanse innen UX, data science, forretnings- og systemutvikling. Headit er et selskap som jobber sammen med våre kunder om å realisere strategi og mål. Hver dag skriver vi videre på vår unike historie. Vi har hovedkontor på Hamar, og er i dag 33 ansatte.

Lytte, forstå og løse!

Drivstoffpriser er blitt av stadig større viktighet for økonomien til mange, og Headit har inngått et samarbeid med utviklerne av Drivstoffappen. Appen er basert på brukernes egne observasjoner, men ikke alle priser oppdateres like ofte. Derfor har vi utviklet en maskinlæringsmodell, som skal beregne sannsynlig pris for brukerne, også der prisen ikke er blitt oppdatert nylig. I forbindelse med dette ønsker vi å utvikle en løsning for kundene å ta i bruk denne tjenesten.

Oppgaven – Predikerte priser på drivstoff

Behovet for en slik løsning er følgende:

- **Integrering av maskinlæringsmodell**, som viser priser for de aktuelle stasjonene for betalende kunder.
- **Backend-tjeneste**
  - Lese inn data fra databasen og inn til ML-modell
  - Utføre predikasjon
  - Publisere predikasjoner fra modell og inn i kart (se frontend-tjeneste)
- **Frontend-tjeneste**
  - Fullscreen-kart som plotter posisjoner på stasjoner med pris
    - Det skal være enkelt å se hvor billigste besinpris er til enhver tid, og det skal være tydelig om det er en predikasjon eller en faktisk pris som vises
  - Visuell fremstilling av sikkerhet/feilmargin på predikasjoner
  - Visuell fremstilling av fremtidig pris innenfor gitte tidsrammer, og forslag til når og hvor man bør fylle innenfor gitte tidsintervaller
    - 1, 3, 7 dagers alternativer
  - Mulighet for å skru av predikasjoner

Headit vil bistå prosjektet i gjennomføringen av oppgaven. Jevnlige møter og oppgavefordelinger avtales underveis.

Oppgaven er egnet for to til fire utviklere, noe som også vil bidra til en innføring i Scrum-metodikk og team-samarbeid, og vil gi innsikt innen følgende områder:

- Utvikling av app
- Database
- Java/Rest/Frontend
- Visualisering i kart
- Utviklingsverktøy
- Implementering av maskinlæringsmodeller

Vedlegg C

Prosjektplan

IDATG2900
Predikerte priser på drivstoff
Prosjektplan

Simen Refsland, Oscar Urfjell Brøndelsbo

Januar 2022

Innhold

Forkortelser

ML - Maskinlæring

REST - Representational State Transfer

API - Application Programming Interface

UX - User Experience

HTML - Hypertext Transfer Protocol

UI - User Interface

Ordforklaringer

Repository - kodebase for kildekode

Branch - gren i kildetreet i Git

Bug - feil i programmet

Code coverage - hvor mange linjer av koden som er dekket av automatiske tester

1 Mål og rammer

1.1 Bakgrunn

I det som er det avsluttende semesteret av Bachelor i ingeniørfag, data, skal det gjennomføres en bacheloroppgave som utføres på vegne av en oppdragsgiver. I vårt tilfelle er dette den eksterne bedriften Headit. Formålet med oppgaven er å gi innsikt og erfaring i identifisere og løse problemer i et ingeniørfaglig relevant område, samt å kunne planlegge, gjennomføre og dokumentere prosjektet på en god måte. Headit er en bedrift med tverrfaglig kompetanse innen UX, data science, forretnings- og systemutvikling.[1]

Med dagens drivstoffpriser har det blitt viktig for mange folk å følge med på drivstoffprisene og fylle når eller hvor det er billig. Headit har inngått et samarbeid med utviklerne av Drivstoffappen.[2] Drivstoffappen er en app hvor brukere kan legge inn, oppdatere og bekrefte drivstoffpriser på bensinstasjoner rundt om i Norge. Siden oppdateringene på drivstoffpriser kommer fra brukernes observasjoner, er det ikke alltid gitt at alle prisene er oppdaterte. Drivstoffappen ønsker derfor å se på en løsning der sannsynlig/predikert pris blir beregnet ved hjelp av en maskinlæringsmodell og hvor resultatet av prediksjonen blir plottet inn i en fullscreen-kart som gir en visuell fremstilling til brukeren.

1.2 Prosjektmål

Hovedmålet med prosjektet er å implementere en løsning hvor Drivstoffappens innrapporterte priser fra brukere blir supplert med predikerte priser fra maskinlæringsmodeller.

1.2.1 Resultatmål

Det resulterende produktet skal ha en maskinlæringsmodell som henter data fra Drivstoffappens database og predikerer priser for forskjellige stasjoner og deretter legger disse inn i en prediksjonstabell. Maskinlæringsmodellen må kunne gi brukeren en prediksjon som har tilstrekkelig nøyaktighet, opplyst med sikkerhetsmargin, slik at den gir meningsfull informasjon til brukerne.

Grensesnittet mellom Drivstoffappen og maskinlæringen sine prediksjoner skal være definert gjennom et REST API som henter data fra databasen og sender prediksjoner gjennom HTTP protokollen. Løsningen skal ha et frontend-komponent som visualiserer prediksjonene i et fullscreen-kart for hver bensinstasjon.

1.2.2 Effektmål

Hensikten med å videreutvikle Drivstoffappen med predikerte priser er for å gi meningsfull informasjon om drivstoffpriser til brukerne selv om prisene ikke har blitt nylig oppdatert.

[1] Headit: `https://headit.no/hjem`
[2] Drivstoffappen: `https://drivstoffappen.no`

1.3 Rammer

1.3.1 Tidsramme

Den generelle tidsrammen er fra 11. januar til 20. mai. Bacheloroppgaven skal være leveres på Inspera innen 20. mai kl 12:00, hvor løsning skal være ferdig utviklet og rapport ferdig skrevet.

1.3.2 Generelle rammer for ferdig produkt

- Det skal være tydelig og godt opplyst i brukergrensesnittet at prisen er en prediksjon og ikke manuelt lagt inn av brukere.
- Løsningen skal ha en fleksibel struktur, slik at modeller og komponenter enkelt kan byttes ut.
- Løsningen bør legge til rette for enkel feilsøking og kvalitetskontroll. I maskinlæringsmodeller bør det implementeres en evalueringsparameter som kan avdekke feil.
- Løsningen skal ikke føre til økt opplevd treghet i Drivstoffappen.
- Implementasjonen skal foregå sømløst slik at løsningen ikke trenger input fra bruker/administrator.

2 Omfang

2.1 Fagområde

Relevant fagområde omfatter å forutse tidsrekker, som omfatter områder som å predikere priser, værmeldinger etc. I vårt tilfelle gjelder dette oljepriser og bensinpriser på bensinstasjoner. Å forutse dette kan være til nytte for brukere opptatt av forbrukerøkonomi.

2.2 Avgrensning

API-et som utvikles vil i utgangspunktet ikke være tilgjengelig for allmennheten, så det vil være begrenset fokus på autentisering/autorisasjon. Implementasjonen av frontend vil i første omgang kun omfatte Android-versjonen av applikasjonen, men kan implementeres i iOS-versjonen hvis situasjonen/tiden tillater dette.

2.3 Oppgavebeskrivelse

Oppgaven belager seg på å videreutvikle mobilapplikasjonen Drivstoffappen. I hovedsak er det et ønske om å utvikle (om mulig) og implementere en ML-modell mot Drivstoffappens eksisterende database, med tilsvarende frontend-komponent for visualisering av modellens bensin prediksjoner. Studentene står

fritt til å velge hvordan prediksjonene visualiseres og hvordan påfølgende «features» skal se ut.

Hovedfokuset i oppgaven på maskinlæring og å lage en modell som gir gode prediksjoner på drivstoffpris og som er meningsfulle. Dette innebærer å se på data som brukerne har rapportert inn, eventuelt se på trender i oljepris og andre mulige faktorer som spiller inn.

3 Prosjektorganisering

Siden prosjektgruppen er på to personer, antas det at det blir mye overlapp i arbeidsområder og oppgaver.

3.1 Ansvarsforhold og rutiner

3.1.1 Administrativt

- Prosjektleder: Oscar Urfjell Brøndelsbo
- Prosjektleders ansvarsforhold:
  - Prosjektgjennomføring
  - Fordeling av oppgaver og rammer
  - Planlegging av sprinter
  - Kvalitetssikring av prosess
- Dokument-ansvarlig: Simen Refsland
- Dokument-ansvarlig ansvarsforhold:
  - Møteinkallinger
  - Møtereferater
  - Dokumentasjon (kode, rapport o.l.)

3.1.2 Innen utvikling

Sannsynligvis vil begge gruppemedlemmene jobbe stort sett innen de samme områdene, likevel har det blitt listet opp noen hovedansvar.

1. ML/backend-ansvarlig: Simen Refsland
2. Frontend-ansvarlig: Oscar Urfjell Brøndelsbo

3.2 Rutiner og grupperegler

For å forhindre konflikter og øke produktiviteten og motivasjonen i gruppen er det viktig å etablere noen regler som gruppemedlemmene etter beste evne prøver å følge.

3.2.1 Regler for arbeid

1. Det skal jobbes minst 30 timer hver uke under normale omstendigheter, med mindre noe spesielt oppstår (familierelaterte hendelser, sykdom o.l.)
2. Hvert gruppemedlem skal totalt jobbe minst 500 timer hver i prosjektet.
3. Hvert gruppemedlem prøver etter beste evne å opprettholde en felles arbeidstid fra 12:00 til 16:00 mandag til fredag med mindre dette skjer på bekostning av forelesninger eller andre skolerelaterte aktiviteter. Dette er for at det skal bli enklere å avtale samarbeid, jobbe rutinert og klare 30 arbeidstimer i uka. Dette gir rom for 2+ timer arbeid utenom.
4. Hvert gruppemedlem er selv ansvarlig for å utfylle sine timelister.
5. Gruppen skal rekke viktige tidsfrister og bli ferdig i god tid før fristen utløper om mulig.
6. Hvert gruppemedlem har ansvar for å fordele arbeidsmengden riktig, slik at alle har noe å jobbe med og at det blir rettferdig.
7. Når arbeidet pågår, skal det være fokus på oppgaven og distraksjoner skal fjernes om mulig.
8. Gruppen skal ha jevnlige stand-upmøter hvor progresjon og utfordringer diskuteres.

3.2.2 Regler for væremåte

1. Hvert gruppemedlem viser initiativ når oppgaver skal løses.
2. Hvert gruppemedlem prøver å ha en positiv innstilling for å bidra til et hyggelig arbeidsmiljø.
3. Hvert gruppemedlem prøver å motivere og oppmuntre hverandre hvis man ser den andre sliter.
4. Hvert gruppemedlem ber om hjelp hvis man sitter fast med oppgaven man har.
5. Hvert gruppemedlem prøver å være åpen og tolerante for ulike synspunkt og holdninger.

Eventuelle konflikter som oppstår blir forsøkt løst internt i første omgang, hvis ikke dette hjelper kan veileder bli involvert hvor det forsøkes å finne en løsning.

4 Planlegging, oppfølging og rapportering

4.1 Systemutviklingsmodell

Scrum har blitt valgt som systemutviklingsmodell ettersom at gruppen ser for seg mulighet for å levere produkt i inkrementelle versjoner, med fleksibilitet til krav i hver sprint. Sluttmålet er også relativt løst definert, og ved å definere sprinter blir det lettere å dokumentere og kommunisere fremgang i prosjektet med alle de involverte partene. Kanban og Scrumban ble også vurdert, men ble valgt bort på grunnlag av at Scrum gir bedre oversikt over gruppens progresjon og mål for ekstern oppdragsgiver. I tillegg er prosjektets overordnede mål godt definert, som taler i favør av Scrum.

Når Scrum blir brukt, er kommunikasjon særs viktig, slik at gruppen og oppdragsgiver er oppdaterte på hvilke oppgaver som blir gjort først. *The Scrum Guide* definerer fem Scrum-hendelser: The Sprint, Sprint Planning, Daily Scrum, Sprint Review og Sprint Retrospective [6]. På slutten av hver sprint skal neste sprint planlegges. Dette blir i lys av hvilke oppgaver som har blitt fullført i forrige sprint. I tillegg vil daglige oppdateringer holde gruppen internt oppdatert på hva som skal gjøres og hva som har blitt gjort. Før neste sprint planlegges, vil det gjøres en evaluering på den forbigående sprinten og om tilpasninger må gjøres. Til slutt ser gruppen på måter å forbedre prosjektarbeidet på i neste sprint. Dette gjøres ved å se på hva som gikk bra og hvilke punkter som har forbedringspotensial.

4.2 Statusmøter og beslutningspunkter

Vi har veiledningsmøte med veileder hver torsdag kl 12:30 på NTNU i Gjøvik, med mindre noe annen informasjon blir gitt. Med arbeidsgiver innkalles det til møte på slutten av hver sprint, hvor resultatene for gjeldende sprint blir presentert, samt. plan for ny sprint. Arbeidsgiver kan her komme med innspill på hvilke oppgaver som burde prioriteres. I tillegg har det vært fire møter med de i planleggingsfasen av prosjektet for å gå gjennom problemstillingen, forventninger, motta ML-modul, arbeidsmetoder, og generelt hvordan prosjektet skal utføres. Vi vil også være i kontakt med eierne av Drivstoff Appen, Sander og Syver. Det er de det til syvende og sist blir laget ML-modulen for, så det er viktig at vi har kommunikasjon med de også. Det har også blitt enighet om å stille spørsmål gjennom Skype, også kan de svare når de har tid til det.

Innad i gruppen planlegges det å holdes uformelle møter hver dag mandag til fredag hvor følgende vil bli diskutert:

- Hvilke oppgaver som har blitt gjort
- Hvilke utfordringer man står overfor
- Hvordan utfordringene kan løses
- Hvilke oppgaver man skal gjøre videre

- Hvilke oppgaver som bør bli gjort i samarbeid
- Foreløpig status på prosjektet

5 Organisering av kvalitetssikring

5.1 Dokumentasjon, standarder, konfigurasjonsstyring, verktøy

5.1.1 Dokumentasjon

Ettersom at produktet er en leveranse til en ekstern tredjepart, er det spesielt viktig å fokusere på god dokumentasjon og en konsekvent kodestil slik at kildekoden enkelt kan forstås og videreutvikles ved behov. API-et trenger en klar definisjon på hvilke endpoints som finnes og hva de gjør, hvilke queries som er mulig og hvilke typer HTTP-forespørsler som er mulig.

I tillegg er det viktig å lage modeller som sekvensdiagram, use-case-diagram, UML-diagram etc. for å kommunisere programmenes struktur og funksjonalitet.

5.1.2 Standarder

Høy kvalitet av kode innebærer at det følges visse standarder. Derfor har vi valgt å følge visse standarder som enkelt kan brukes i autoformaterings-verktøy.

For Python:

- PEP 8 - Style Guide for Python Code [5]
- PEP 257 - Docstring conventions [3]

For Kotlin og Android-utvikling vil vi få tilgang til Drivstoffappens repository, her vil vi rette oss etter standardene brukt der, slik at kvaliteten bevares.

5.1.3 Konfigurasjonsstyring

Kildekoden vil være lagret på Github, hvor master-branchen er forbeholdt lanseringsklar kode. Videre vil en typisk Git-workflow følges, med develop, feature, release og hotfix branches [2].

Når det kommer til versjonen til prosjektet, er det logisk å bruke *Major.Minor.Patch*-formatet, hvor endring i versjon deles inn i major (stor inkompatibel endring), minor (mindre kompatible endringer) og patch (bugfixes som er kompatible) [4]. Prosjektet vil i utgangspunkt leve i 0.x.y versjonen innledningsvis og første lanseringsklare versjon vil være 1.0.0.

Alle oppgaver som oppstår skal ha et issue. Hver pull-request skal forklare hvilke endringer som er gjort før merge.

5.1.4 Verktøy/programvare

Overleaf - brukes i all hovedsak til å skrive rapport i LaTeX.

Github - brukes til versjonskontroll.

Discord/Teams - brukes til kommunikasjon og oppsett av møter.

Zenhub - brukes til organisering av backlog og issues. Kan enkelt integreres i Github.

Toggl Track - brukes til å registere timelister på en enkel måte. Kan også skrive ut statusrapporter fra gitte tidsrom.

Visual Studio Code - brukes som primært utviklingsmiljø.

Android Studio - standard utviklingsmiljø for Android-programmering.

TeamGantt - brukes til å lage Gantt-diagram.

Onedrive/Sharepoint - brukes til å dele filer internt i gruppen som ikke inngår i repository på Github.

GoLand - brukes til skriving av API.

5.2 Plan for inspeksjoner og testing

For å lage robuste program med minimalt med bugs er det viktig med omfattende testing før lansering. I tillegg er det viktig å verifisere produktet står til oppdragsgiverens forventninger. For at produktet skal fungere som det skal, må også brukeren intuitivt vite hvordan det fungerer. For å møte disse utfordringene må det lages forskjellige tester.

Brukertester planlegges fordi det er viktig å verifisere at gjennomsnittsbrukeren vet hvordan man benytter seg av prediksjonstjenesten. I tillegg er det svært viktig at brukeren forstår at funksjonen bare predikerer prisen, hvis dette blir tatt for å være den sikre prisen, kan det oppstå misnøye med tjenesten hvis det viser seg å være feil. I tillegg må forståelse av eventuelle tilleggsfunksjoner (sikkerhetsmargin, prediksjoner frem i tid o.l.) testes hvis dette skal implementeres. For å sikre at produktet fungerer i henhold til sluttbrukerens krav, planlegges det akspetansetester. Disse kan utformes ved hjelp av brukerhistorier.

I tillegg er det viktig å teste at programvaren også fungerer som det skal. API-tester sikrer at innholdet fra responsen har riktig statuskode og riktig innhold. Unit testing sikrer at funksjonene returnerer riktige verdier. I planleggingsfasen er det tatt høyde for at det vil være minst tre individuelle moduler (maskinlæring, API og mobilapplikasjon). Derfor er det viktig å utføre integrasjonstester for å se at modulene fungerer som forventet når de kombineres.

Andre tester som kan vurderes er automatiserte UI tester som kan simulere brukerinteraksjoner. Dette kan potensielt spare tid og/eller avdekke feil som

ikke vanligvis ville blitt oppdaget. Android Jetpack tilbyr flere APIer som har denne funksjonaliteten, deriblant Espresso [1].

Unit-tester og API-tester kan implementeres fortløpende i hver sprint, brukertester vil være hensiktsmessig i slutten av mars, da det forhåpentligvis har blitt implementert et grunnleggende brukergrensesnitt. Alternativt kan mockups og prototyper lages. Integrasjonstester vil være fornuftig når modulene fungerer hver for seg.

5.3 Risikoanalyse

Det har blitt foretatt en risikoanalyse med de punktene som synes relevant for gruppen. Her listes det potensielle fare/problemer som kan oppstå, hvilke konsekvenser disse har og tiltak for å redusere risikoen. Alle risikoer har fått sannynlig- og konsekvenspoeng som sier noe om hvilke problemer som må prioriteres. Her følger poengene risikomatrisen som vist under, hvor 2-4 er grønt nivå, 5-7 gult nivå og 8-10 rødt nivå.

		Konsekvens				
		Ubetydelig **1**	Mindre alvorlig **2**	Betydelig **3**	Alvorlig **4**	Kritisk **5**
Sannsynlighet	Svært sannsynlig **5**	6	7	8	9	10
	Meget Sannsynlig **4**	5	6	7	8	9
	Sannsynlig **3**	4	5	6	7	8
	Lite sannsynlig **2**	3	4	5	6	7
	Usannsynlig **1**	2	3	4	5	6

Figur 1: Risikomatrise

NR	HVA	MULIGE KONSEKVENSER	RISIKO			REDUSERENDE TILTAK
			SANNSYNLIGHET	KONSEKVENS	RISIKOINDEKS	
1	Oppgaven er for omfattende/kompleks	Ikke får ønsket resultat. Mye tidsbruk.	2	4	6	Fokuser på delinnleveringer som fungerer. Bruk til å lære viktige ting. Realistisk målsetning.
2	For lite data til treningssett	Unøyaktig/lite brukbare prediksjoner	3	4	7	Se etter alternative kilder til drivstoffdata (oljepris for eksempel).
3	Tap av kode som ikke er lagt til i repository	Ekstra tidsbruk	2	2	4	Commit og push til Github hyppig slik at koden er flere steder.
4	Sykdom og andre forhold	Resterende gruppemedlemmer må ta på seg flere oppgaver.	2	3	6	Alle gruppemedlemmer burde ha innsikt i hvordan en løser alle oppgaver som kommer opp.
5	Konflikter innad i gruppa	Redusert arbeidsinnsats og effektivitet. Forverring av arbeidsmiljø.	3	3	6	Definer gode regler for arbeid og væremåte. Ha god kommunikasjon og bevissthet rundt dette.
6	Tap av hele kodebasen	Alt arbeid lagt fremt til da forsvinner. Må starte på nytt.	1	5	6	Tilse at kode er oppdatert både lokalt og remote hos begge gruppemedlemmer
7	Rapport ikke ferdig til tidsfristen	Ufullstendig innlevering, dårligere karakter.	1	4	5	Jobb rutinert og med mål om å bli ferdig i god tid før tidsfrist. Legg ting inni rapporten under veis.
8	Sluttbrukeren forstår ikke funksjonalitet	Misnøye med applikasjonen, redusert tillit.	2	3	5	Utfør brukertester fra forskjellige demografiske grupperinger slik at en får et helhetlig bilde av brukernes intuisjon av brukergrensesnittet.
9	Komponentene har høy kopling/lite abraksjon	Komponentene er avhengige av hverandre. Vanskeligere å vedlikeholde.	2	3	5	Ha fokus på en god arkitektur og vær obs på dette tidlig. Definer hver komponent sin rolle.
10	Oppgaven er for lite omfattende	Produktet blir for lite i forhold til forventet arbeidsmengde.	1	3	4	Spør oppdragsgiver om forslag/krav om utvidelser, mer funksjonalitet o.l.
11	Motivasjonen forsvinner i løpet prosjektet	Redusert arbeidsinnsats, mål for prosjektet blir ikke nådd.	4	3	7	Spille hverandre gode, bryte problemet ned i mindre deler, sett delmål.
12	Levert løsning svarer ikke til oppdragsgivers forventninger	Ferdig produkt blir ikke slik oppdragsgiver har tenkt seg.	1	4	5	Ha møter etter sprinter hvor utført arbeid blir gjennomgått. Klar kommunikasjon mellom studenter og oppdragsgiver.

Figur 2: Risikovurdering for prosjektet

6 Plan for gjennomføring

I og med at prosjektet gjennomføres ved hjelp av en smidig systemutviklingsmodell, vil det være ugunstig å planlegge noe i detalj langt frem i tid. Sprintenes form og innhold vil hovedsaklig være basert på den forrige sprintens resultater, samt potensielle endringer i krav og innhold. Sprintene har planlagt med utgangspunkt i 10 arbeidsdagers lengde, og planen i hver sprint er løst beskrevet. Likevel er det listet opp noen antatte aktiviteter og milepæler. Planen vil høyst sannsynlig endres i takt med problemer og utfordringer som dukker opp.

Sprint 1, 1. feb - 11. feb Hovedfokuset i sprint 1 er å få tilgang til maskinlæringsmodell, innledende design og dokumentasjon og å lage et grunnleggende API. Milepæl: grunnleggende dokumentasjon og design (klassediagram

Sprint 2, 14. feb - 25. feb Sprint 2 er forbeholdt eventuelle aktiviteter som ikke ble ferdig i sprint 1, samt å få maskinlæringsmodell til å fungere. Milepæl: grunnleggende funksjonalitet for maskinlæringsmodell.

Sprint 3, 28. feb - 11. mars Sprint 3 er det planlagt å videreutvikle maskinlæringsmodellen, eventuelt lage ny. Muligens begynne på brukergrensesnitt i tillegg.

Sprint 4, 14. mars - 25. mars Sprint 4 er forbeholdt å lage UI og ferdigstille minimumløsning slik at brukertesting kan begynne. Milepæl: minimumsløsning.

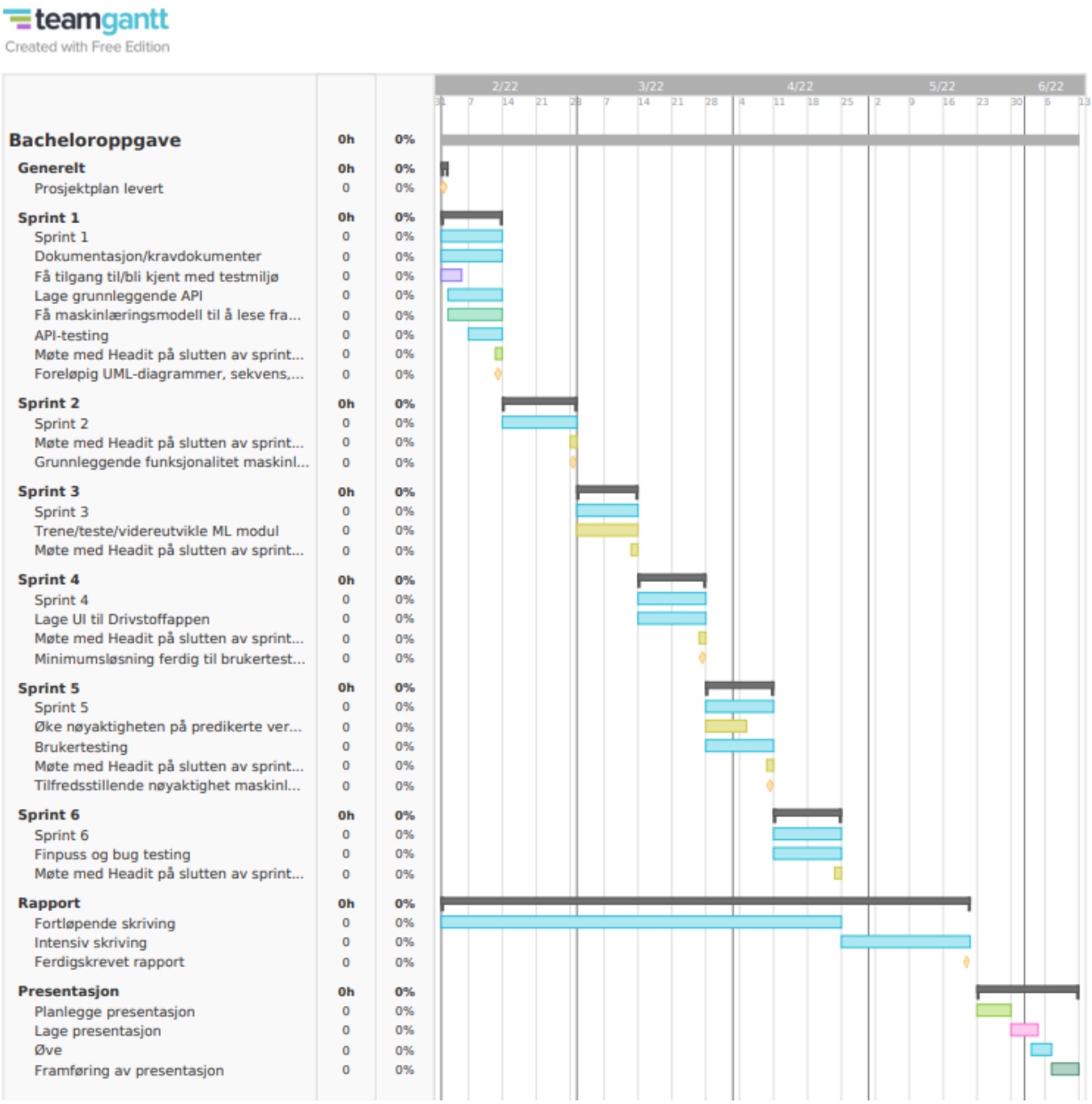
Sprint 5, 28. mars - 8. apr I sprint 5 vil brukertestingen begynne, samt å prøve å forbedre/optimalisere maskinlæringsmodellen. Resultatene fra brukertestingen kan medføre endringer i design av brukergrensesnittet.

Sprint 6, 11. apr - 22. apr Sprint 6 vil finpuss og feiltesting være i fokus slik at det blir verifisert at løsningen er robust og av høy kvalitet.

Rapportskriving, 1. feb - 19. mai Rapportskriving vil i første periode av prosjektet være en liten del av arbeidsoppgavene, og vil oppta større plass etter hvert i prosjektet. Etter sprint 6 vil rapportskriving være hovedfokuset og vil være den mest intensive perioden for dette.

Presentasjon 23. mai - 10. juni Første del av presentasjonsfasen vil være å planlegge presentasjonens form og innhold, med andre ord, hvilke aspekter av prosjektet som skal inkluderes. Videre vil presentasjonen bli laget og øvd på, slik at gruppen er klar i perioden 7. - 10. juni.

6.1 Gantt-diagram



Figur 3: Gantt-diagram

6.2 Aktiviteter

I løpet av prosjektet er det blitt planlagt/tatt høyde for følgende aktiviter i forbindelse med utvikling av programvare. Disse fungerer som en generell rettesnor for hovedmålene i forbindelse til prosjektet.

1. Maskinlæring
    - 1.1 Få tilgang til data
    - 1.2 Integrere ferdiglagd modell
    - 1.3 Videreutvikle/lage ny maskinlæringsmodell
    - 1.4 Trene modell
    - 1.5 Testing av nøyaktighet
    - 1.6 Lagre resultat i database
2. API
    - 2.1 Lese og sende prediksjonsdata
    - 2.2 Endepunkter for seneste prediksjonsdata, historikk, fremtidig prediksjon
3. Brukergrensesnitt (UI)
    - 3.1 Visualisering i kart
    - 3.2 Prediksjon med sikkerhetsmarginer
4. Dokumentasjon
    - 4.1 Kravdokumenter
    - 4.2 Design (UML, sekvens o.l.)
    - 4.3 API-dokumentasjon

Referanser

[1] Android. *Automate UI tests*. URL: https://developer.android.com/training/testing/instrumented-tests/ui-tests. (hentet: 27.01.2022).

[2] Atlassian Bitbucket. *Gitflow Workflow*. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow. (hentet: 26.01.2022).

[3] David Goodger og Guido van Rossum. *PEP 257 – Docstring Conventions*. URL: https://www.python.org/dev/peps/pep-0257/#multi-line-docstrings. (hentet: 26.01.2022).

[4] Tom Preston-Werner. *Semantic Versioning*. URL: https://semver.org/. (hentet: 26.01.2022).

[5] Guido van Rossum, Barry Warsaw og Nick Coghlan. *PEP 8 – Style Guide for Python Code*. URL: https://www.python.org/dev/peps/pep-0008/. (hentet: 26.01.2022).

[6] Ken Schwaber og Jeff Sutherland. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. URL: https://scrumguides.org/scrum-guide.html. (hentet: 27.01.2022).

Vedlegg D

API-dokumentasjon

API-dokumentasjon

API-rot:
http://localhost/api

Få siste prediksjoner fra gitt stasjon

Request

`Metode: GET`
`URI: /predictions/latest/{stationId}`

Eksempel på request: `/predictions/latest/101`

Response

- Content type: application/json
- Status kode: 200 hvis suksessfull, feilmelding hvis ikke.

Eksempel på response body:


```
[
  {
    "stationId": 101,
    "predictionDate": "2022-02-18",
    "gasType": "95",
    "priceLowerMargin": 15.22,
    "priceUpperMargin": 17.1,
    "price": 16.32
  },
  {
    "stationId": 101,
    "predictionDate": "2022-02-18",
    "gasType": "D",
    "priceLowerMargin": 14.77,
    "priceUpperMargin": 17.23,
    "price": 16.2
  },
  {
    "stationId": 101,
    "predictionDate": "2022-02-19",
    "gasType": "D",
    "priceLowerMargin": 15.22,
    "priceUpperMargin": 17.1,
    "price": 16.7
  },
  {
    "stationId": 101,
```

```
    "predictionDate": "2022-02-19",
    "gasType": "95",
    "priceLowerMargin": 12.64,
    "priceUpperMargin": 25.65,
    "price": 23.44
  }
]
```

Legg inn ny prediksjon

Request

`Metode: POST`
`URI: /predictions/{stationId}`

- Content type: application/json

Request body må inneholde:

- `stationId`, id på stasjonen som blir predikert
- `predictionDate`, dato prediksjonen gjelder
- `gasType`, type drivstoff
- `priceLowerMargin`, lavere estimat på pris
- `priceUpperMargin`, høyere estimat på pris
- `price`, predikert pris

Eksempel på request: `/predictions/101` Eksempel på request body:

```
{
  "stationId": 101,
  "predictionDate": "2022-02-18",
  "gasType": "95",
  "priceLowerMargin": 14.77,
  "priceUpperMargin": 17.23,
  "price": 16.20
}
```

Response

- Content type: `application/json`
- Status kode: 201 hvis suksessfull, feilmelding hvis ikke.

Legg inn prediksjoner

Request

`Metode: POST`
`URI: /predictions`

- Content type: application/json

Request body må inneholde:

- `stationId`, id på stasjonen som blir predikert
- `predictionDate`, dato prediksjonen gjelder
- `gasType`, type drivstoff
- `priceLowerMargin`, lavere estimat på pris
- `priceUpperMargin`, høyere estimat på pris
- `price`, predikert pris

Eksempel på request body:

```
[
  {
    "stationId": 101,
    "predictionDate": "2022-02-18",
    "gasType": "D",
    "priceLowerMargin": 14.77,
    "priceUpperMargin": 17.23,
    "price": 16.20
  },
  {
    "stationId": 101,
    "predictionDate": "2022-02-19",
    "gasType": "D",
    "priceLowerMargin": 15.22,
    "priceUpperMargin": 17.10,
    "price": 16.70
  }
]
```

Response

- Content type: `application/json`
- Status kode: 201 hvis suksessfull, feilmelding hvis ikke.

Få prediksjoner for gitt stasjon

Request

`Metode: GET`
`URI: /predictions/{station_id}`

Valgfrie query parametre:

- `startDate`, startdato for prediksjoner
- `endDate`, sluttdato for prediksjoner

Eksempel på request: `/predictions/688?startDate=2022-04-28&endDate=2022-04-29`

Response

- Content type: application/json
- Status kode: 200 hvis suksessfull, feilmelding hvis ikke.


```
[
  {
    "stationId": 688,
    "predictionDate": "2022-04-28",
    "gasType": "95",
    "priceLowerMargin": 18.55,
    "priceUpperMargin": 22.66,
    "price": 19.66
  },
  {
    "stationId": 688,
    "predictionDate": "2022-04-28",
    "gasType": "D",
    "priceLowerMargin": 16.17,
    "priceUpperMargin": 22.23,
    "price": 19.22
  },
  {
    "stationId": 688,
    "predictionDate": "2022-04-29",
    "gasType": "95",
    "priceLowerMargin": 16.64,
    "priceUpperMargin": 22.21,
    "price": 18.44
  },
  {
    "stationId": 688,
    "predictionDate": "2022-04-29",
    "gasType": "D",
    "priceLowerMargin": 15.75,
    "priceUpperMargin": 20.77,
    "price": 17.75
  }
]
```


Slett en prediksjon

Request

`Metode: DELETE`
`URI: /predictions/{stationId}`

- Content type: application/json

Request body må inneholde:

- `stationId`, id på stasjonen som er blitt predikert
- `predictionDate`, dato prediksjonen gjelder
- `gasType`, type drivstoff

Eksempel på request: `/predictions/101` Eksempel på request body:

```
{
  "stationId": 101,
  "predictionDate": "2022-02-18",
  "gasType": "95"
}
```

Response

- Content type: `application/json`
- Status kode: 200 hvis suksessfull, feilmelding hvis ikke.

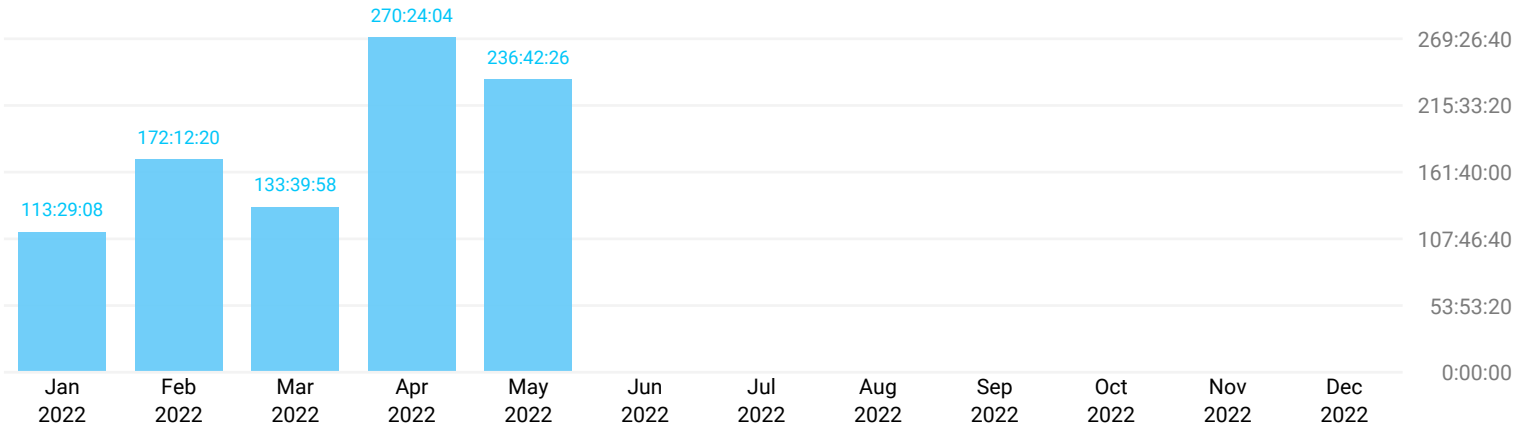
Vedlegg E

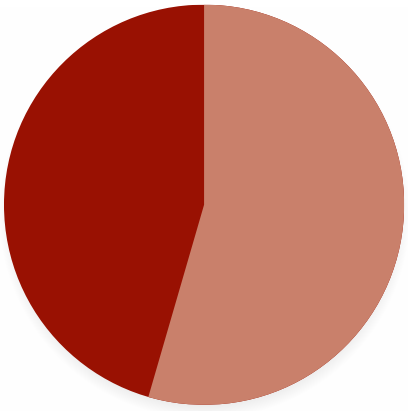
Arbeidslogg

Summary Report

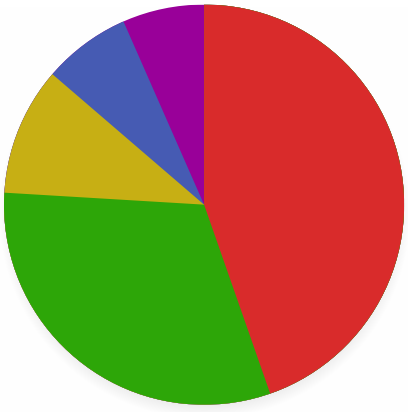
01/01/2022 – 12/31/2022

TOTAL HOURS: 926:27:56





USER	DURATION
SI Simen_refsland	504:44:55
OS osci	421:43:01



PROJECT	DURATION
Utvikling	414:08:12
Rapport	289:10:21
Research	96:30:11
Generelt	65:34:17
Prosjektplan	61:04:55

USER - PROJECT	DURATION
OS osci	421:43:01
Generelt	33:34:59
Prosjektplan	24:49:17
Rapport	151:11:08
Research	58:45:58
Utvikling	153:21:39
SI Simen_refsland	504:44:55
Generelt	31:59:18
Prosjektplan	36:15:38
Rapport	137:59:13
Research	37:44:13
Utvikling	260:46:33

Detailed Report

01/01/2022 – 12/31/2022

TOTAL HOURS

926:27:56

TIME ENTRY	TAGS	USER	DURATION	TIME
Første møte med arbeidsgiver ● Generelt	Annet	osci	1:00:00	12:00 PM-01:00 PM 01/12/2022
Møte med oppdragsgiver introduksjon ● Generelt	Annet	Simen_refsland	0:30:11	12:00 PM-12:30 PM 01/12/2022
Oppsett av timelisteprogram, referat fra møte, TensorFlow, oppsett av programvare ● Generelt	Annet	Simen_refsland	3:08:53	12:30 PM-03:38 PM 01/12/2022
Bli-kjent møte med Tom Røise ● Generelt	Annet	osci	0:43:47	11:30 AM-12:14 PM 01/13/2022
Introduksjonsmøte med veileder + skriving av referat ● Generelt	Annet	Simen_refsland	1:00:00	12:30 PM-01:30 PM 01/13/2022
Møte med Sander og Syver 1 ● Generelt	Annet	osci	0:28:32	09:13 AM-09:42 AM 01/14/2022
Møte med Sander og Syver 2 ● Generelt	Annet	osci	0:33:16	09:56 AM-10:29 AM 01/14/2022
Møte med Drivstoffappen-utviklere ● Generelt	Annet	Simen_refsland	1:44:09	10:00 AM-11:45 AM 01/14/2022
Lære TensorFlow introduksjon ● Research	Research	Simen_refsland	4:52:14	12:00 PM-04:53 PM 01/17/2022
Mer TensorFlow ● Research	Research	Simen_refsland	4:30:05	12:00 PM-04:30 PM 01/18/2022

Mer TensorFlow Research	Research	Simen_refsland	5:10:33	12:00 PM-05:11 PM 01/19/2022
Jobbe med forprosjektplan Prosjektplan	Prosjektplan	Simen_refsland	1:27:13	05:30 PM-06:57 PM 01/20/2022
Veiledningsmøte 2 Generelt	Annet	osci	0:42:43	10:00 AM-10:43 AM 01/21/2022
Møte med veileder + referat Generelt	Annet	Simen_refsland	1:38:54	11:00 AM-12:39 PM 01/21/2022
Python Research	Research	osci	8:30:47	12:36 PM-09:07 PM 01/21/2022
Tensorflow tutorial Research	Research	osci	2:15:58	09:34 AM-11:50 AM 01/24/2022
Jobbe med prosjektplan Prosjektplan	Prosjektplan	Simen_refsland	5:00:22	10:00 AM-03:00 PM 01/24/2022
gruppemøte Generelt	Annet	osci	0:46:28	12:06 PM-12:53 PM 01/24/2022
Forprosjektplan Prosjektplan	Prosjektplan	osci	1:33:39	12:53 PM-02:27 PM 01/24/2022
Møte med headit Generelt	Annet	osci	0:33:45	01:56 PM-02:29 PM 01/24/2022
Møte med oppdragsgiver + referat Generelt	Annet	Simen_refsland	1:06:17	03:00 PM-04:06 PM 01/24/2022
tensorflow testing Research	Research	osci	2:09:23	05:06 PM-07:15 PM 01/24/2022

Utforme standardavtale ● Generelt	Prosjektplan	Simen_refsland	0:56:35	11:24 AM-12:21 PM 01/25/2022
Forprosjektplan ● Prosjektplan	Prosjektplan	osci	6:12:16	11:40 AM-05:53 PM 01/25/2022
Jobbe med prosjektplan ● Prosjektplan	Prosjektplan	Simen_refsland	5:18:42	12:21 PM-05:40 PM 01/25/2022
Forprosjektplan ● Prosjektplan	Prosjektplan	osci	2:15:42	09:05 AM-11:21 AM 01/26/2022
Jobbe med prosjektplan ● Prosjektplan	Prosjektplan	Simen_refsland	3:52:15	10:30 AM-02:22 PM 01/26/2022
github ● Generelt	Research	osci	1:29:06	11:31 AM-01:01 PM 01/26/2022
Zenhub ● Generelt	Research	osci	0:32:42	01:21 PM-01:54 PM 01/26/2022
Forprosjektplan ● Prosjektplan	Prosjektplan	osci	2:17:00	03:03 PM-05:20 PM 01/26/2022
Risikoanalyse prosjektplan ● Prosjektplan	Prosjektplan	Simen_refsland	3:01:13	03:59 PM-07:01 PM 01/26/2022
risikomatrise ● Prosjektplan	Prosjektplan	osci	0:56:41	06:40 PM-07:37 PM 01/26/2022
Forprosjektplan ● Prosjektplan	Prosjektplan	osci	1:49:34	10:23 AM-12:13 PM 01/27/2022
Jobbe med prosjektplan ● Prosjektplan	Prosjektplan	Simen_refsland	1:50:33	10:59 AM-12:50 PM 01/27/2022

Veiledningsmøte 3 ● Generelt	Annet	osci	1:29:36	12:13 PM-01:43 PM 01/27/2022
Møte med veileder + skrive referat ● Generelt	Annet	Simen_refsland	0:55:51	12:50 PM-01:46 PM 01/27/2022
Forprosjektplan ● Prosjektplan	Prosjektplan	osci	2:30:45	01:47 PM-04:17 PM 01/27/2022
Prosjektplan, Risikovurering, Gantt-skjema ● Prosjektplan	Prosjektplan	Simen_refsland	6:59:14	10:10 AM-05:10 PM 01/28/2022
Tensorflow tutorial ● Research	Research	osci	3:27:36	10:11 AM-01:39 PM 01/28/2022
Forprosjektsplan ● Prosjektplan	Prosjektplan	osci	1:49:46	01:58 PM-03:48 PM 01/28/2022
tensorflow testing ● Research	Research	osci	2:06:52	09:31 AM-11:37 AM 01/31/2022
Ferdigstilling av prosjektplan + ny standardavtale ● Prosjektplan	Prosjektplan	Simen_refsland	3:05:05	10:03 AM-01:08 PM 01/31/2022
gantt-diagram ● Prosjektplan	Prosjektplan	osci	0:28:25	11:42 AM-12:11 PM 01/31/2022
Forprosjektsplan ● Prosjektplan	Prosjektplan	osci	4:55:29	05:19 PM-10:15 PM 01/31/2022
Ferdigstilling prosjektplan ● Prosjektplan	Prosjektplan	Simen_refsland	5:41:01	06:00 PM-11:41 PM 01/31/2022
Gjennomgang maskin læring ● Research	Research	osci	1:37:46	01:59 PM-03:37 PM 02/01/2022

Møte, ML-gjennomgang Generelt	Research	Simen_refsland	2:11:33	02:54 PM-05:05 PM 02/01/2022
Tensorflow tutorial Research	Research	osci	2:35:14	04:48 PM-07:23 PM 02/01/2022
Tensorflow tutorial Research	Research	osci	2:32:30	09:57 AM-12:30 PM 02/02/2022
Tensorflow tutorial Research	Research	osci	2:44:47	12:31 PM-03:16 PM 02/02/2022
tensorflow testing Research	Research	osci	1:31:44	10:56 AM-12:27 PM 02/03/2022
Møte med veileder Generelt	Annet	Simen_refsland	2:09:47	12:55 PM-03:05 PM 02/03/2022
Veiledningsmøte Generelt	Annet	osci	1:41:51	01:28 PM-03:10 PM 02/03/2022
Statistikk i maskin læring Research	Research	osci	1:03:43	10:05 AM-11:09 AM 02/04/2022
maskin læring algoritmer Research	Research	osci	1:18:39	11:09 AM-12:28 PM 02/04/2022
bli kjent i drivstoffappen kildekode Research	Research	osci	2:55:31	02:44 PM-05:40 PM 02/04/2022
Sette seg inn i Drivstoffappens kodebase Research	Research	Simen_refsland	1:05:02	11:03 AM-12:08 PM 02/07/2022
Lese DrivstoffAppen kode Research	Research	osci	6:25:58	11:22 AM-05:48 PM 02/07/2022

Sette seg inn i Drivstoffappens kodebase ● Research	Research	Simen_refsland	0:56:06	05:15 PM-06:11 PM 02/07/2022
Spring boot intro + oppsett av lokal database ● Utvikling	Utvikling	Simen_refsland	5:50:42	12:30 PM-06:21 PM 02/08/2022
maskin læring algoritmer ● Research	Research	osci	1:57:36	02:07 PM-04:05 PM 02/08/2022
Lese DrivstoffAppen API ● Research	Research	osci	5:57:41	08:59 AM-02:57 PM 02/09/2022
Oppsett av lokal database for koding ● Utvikling	Utvikling	Simen_refsland	3:58:46	12:16 PM-04:14 PM 02/09/2022
Sette opp lokalt miljø ● Utvikling	Utvikling	osci	0:53:22	10:15 AM-11:08 AM 02/10/2022
Oppsett lokal API + møte med veileder ● Utvikling	Utvikling	Simen_refsland	4:10:27	10:20 AM-02:31 PM 02/10/2022
Veiledningsmøte ● Generelt	Annet	osci	2:29:49	11:08 AM-01:38 PM 02/10/2022
API ● Utvikling	Utvikling	osci	6:02:47	01:38 PM-07:41 PM 02/10/2022
Endring av lokalt oppsett + ML ● Utvikling	Utvikling	Simen_refsland	5:44:13	11:18 AM-05:02 PM 02/11/2022
Oppdatere API ● Utvikling	Utvikling	osci	0:00:07	11:43 PM-11:44 PM 02/13/2022
API-design, møte med Headit, prediksjonstabell ● Utvikling	Utvikling	Simen_refsland	5:04:17	10:59 AM-04:03 PM 02/14/2022

Endpoint design ● Utvikling	Utvikling	osci	0:44:03	12:16 PM-01:00 PM 02/14/2022
møte m/ headit ● Generelt	Annet	osci	0:30:11	01:00 PM-01:30 PM 02/14/2022
prediksjonstabell ● Utvikling	Utvikling	osci	1:48:34	01:50 PM-03:39 PM 02/14/2022
database oppfriskning ● Generelt	Annet	osci	3:23:02	08:06 AM-11:29 AM 02/15/2022
Prediksjonstabell i databasen ● Utvikling	Utvikling	osci	3:54:14	12:56 PM-04:50 PM 02/15/2022
API dokumentasjon, ORM for prediksjonstabell ● Utvikling	Utvikling	Simen_refsland	4:16:11	03:01 PM-07:17 PM 02/15/2022
HTTP og API oppfriskning ● Utvikling	Utvikling	osci	6:17:07	07:08 AM-01:25 PM 02/16/2022
API veiledning ● Generelt	Annet	Simen_refsland	2:14:46	12:02 PM-02:17 PM 02/17/2022
GET funksjonalitet for seneste prediksjoner API ● Utvikling	Utvikling	osci	6:49:22	09:30 PM-04:19 AM 02/17/2022 - 02/18/2022
PredictionDao.java ● Utvikling	Utvikling	osci	5:34:23	09:30 AM-03:04 PM 02/18/2022
PredictionDao.java ● Utvikling	Utvikling	osci	0:30:04	04:48 PM-05:18 PM 02/18/2022
POST-funksjonalitet API ● Utvikling	Utvikling	Simen_refsland	2:30:51	01:34 PM-04:05 PM 02/20/2022

POST-funksjonalitet API ● Utvikling	Utvikling	Simen_refsland	2:23:46	02:33 PM-04:57 PM 02/21/2022
POST-funksjonalitet API ● Utvikling	Utvikling	Simen_refsland	2:58:23	01:54 PM-04:53 PM 02/22/2022
GET-funksjonalitet API + Møte med veileder ● Utvikling	Utvikling	Simen_refsland	4:12:29	10:01 AM-02:13 PM 02/24/2022
PredictionLogDao.java - getLatestPredictions ● Utvikling	Utvikling	osci	5:46:04	11:42 AM-05:28 PM 02/24/2022
veiledningsmøte ● Generelt	Annet	osci	1:00:01	12:00 PM-01:00 PM 02/24/2022
GET-funksjonalitet API ● Utvikling	Utvikling	Simen_refsland	1:36:00	03:00 PM-04:36 PM 02/24/2022
GET-funksjonalitet API ● Utvikling	Utvikling	Simen_refsland	1:48:08	06:40 PM-08:28 PM 02/24/2022
PredictionLogDao & PredictionController ● Utvikling	Utvikling	osci	6:01:40	09:30 AM-03:31 PM 02/25/2022
API testing ● Utvikling	Utvikling	Simen_refsland	4:54:39	10:57 AM-03:52 PM 02/25/2022
API testing ● Utvikling	Utvikling	Simen_refsland	1:17:36	03:58 PM-05:16 PM 02/25/2022
API testing ● Utvikling	Utvikling	Simen_refsland	5:25:40	12:04 PM-05:30 PM 02/26/2022
Integrasjon ML-modell ● Utvikling	Utvikling	Simen_refsland	5:55:24	12:27 PM-06:22 PM 02/27/2022

API demo og kode dokumentasjon ● Utvikling	Utvikling	osci	5:17:35	02:37 PM-07:55 PM 02/27/2022
Integrasjon ML-modell ● Utvikling	Utvikling	Simen_refsland	8:07:00	07:14 PM-03:21 AM 02/27/2022 - 02/28/2022
Integrasjon ML-modell + møte med Headit ● Utvikling	Utvikling	Simen_refsland	3:55:09	11:16 AM-03:11 PM 02/28/2022
Lynkurs 2 ● Generelt	Annet	osci	3:15:55	01:39 PM-04:55 PM 03/02/2022
Lynkurs rapportskriving ● Generelt	Annet	Simen_refsland	1:21:03	02:15 PM-03:36 PM 03/02/2022
Møte veileder ● Generelt	Annet	Simen_refsland	0:30:02	01:00 PM-01:30 PM 03/03/2022
Sette seg inn i Android-repository ● Research	Research	Simen_refsland	2:21:02	01:05 PM-03:26 PM 03/05/2022
Møte + oppsett av Overleaf prosjekt ● Generelt	Annet	Simen_refsland	1:31:44	01:55 PM-03:27 PM 03/08/2022
front end tabell ● Utvikling	Utvikling	osci	3:34:29	11:56 AM-03:30 PM 03/14/2022
Signere fortrolighetsavtale ● Generelt	Annet	Simen_refsland	0:33:02	03:36 PM-04:09 PM 03/16/2022
Tensorflow maskinlæring CNN, LSTM ● Utvikling	Utvikling	Simen_refsland	7:56:28	08:30 AM-04:27 PM 03/21/2022
Tensorflow utprøving flere features + Android UI start ● Utvikling	Utvikling	Simen_refsland	7:56:18	08:34 AM-04:30 PM 03/22/2022

Frontend tabell ● Utvikling	Utvikling	osci	3:34:00	01:01 PM-04:35 PM 03/23/2022
Android frontend ● Utvikling	Utvikling	Simen_refsland	2:29:01	06:33 PM-09:02 PM 03/23/2022
Android frontend ● Utvikling	Utvikling	Simen_refsland	0:31:33	11:00 PM-11:32 PM 03/23/2022
Android frontend + møte med Headit ● Utvikling	Utvikling	Simen_refsland	6:09:18	08:53 AM-03:02 PM 03/24/2022
frontend plantegning ● Utvikling	Utvikling	osci	5:36:00	09:16 AM-02:52 PM 03/24/2022
Veiledningsmøte med Tom ● Generelt	Annet	osci	0:30:27	09:00 AM-09:30 AM 03/25/2022
Frontend tabell ● Utvikling	Utvikling	osci	5:34:00	09:30 AM-03:04 PM 03/25/2022
Møte veileder + hjelp til Android oppsett ● Generelt	Annet	Simen_refsland	2:46:24	09:56 AM-12:42 PM 03/25/2022
LSTM/ML research ● Research	Research	Simen_refsland	6:12:01	09:53 AM-04:05 PM 03/26/2022
ML research ● Research	Research	Simen_refsland	6:01:57	09:50 AM-03:52 PM 03/27/2022
Android API kall + ML research ● Research	Research	Simen_refsland	6:35:13	08:37 AM-03:12 PM 03/28/2022
Frontend tabell ● Utvikling	Utvikling	osci	1:02:49	12:13 PM-01:16 PM 03/28/2022

Frontend tabell ● Utvikling	Utvikling	osci	5:29:00	01:16 PM-06:45 PM 03/28/2022
Android Frontend Graph View ● Utvikling	Utvikling	Simen_refsland	6:02:13	08:38 AM-02:40 PM 03/29/2022
Frontend tabell ● Utvikling	Utvikling	osci	8:11:00	10:27 AM-06:38 PM 03/29/2022
Android frontend ● Utvikling	Utvikling	Simen_refsland	10:09:41	08:30 AM-06:40 PM 03/30/2022
frontend tabell ● Utvikling	Utvikling	osci	4:05:51	10:45 AM-02:51 PM 03/30/2022
prediksjoner switch tab ● Utvikling	Utvikling	osci	5:04:02	02:51 PM-07:55 PM 03/30/2022
Android frontend ● Utvikling	Utvikling	Simen_refsland	4:47:46	07:52 PM-12:40 AM 03/30/2022 - 03/31/2022
Prediksjonstabell og graf i Android + demo med Headit ● Utvikling	Utvikling	Simen_refsland	5:11:39	09:36 AM-02:48 PM 03/31/2022
Frontend design ● Utvikling	Utvikling	osci	8:36:00	10:17 AM-06:53 PM 03/31/2022
Maskinlæring + møte med Vegard ● Utvikling	Utvikling	Simen_refsland	5:40:05	08:34 AM-02:14 PM 04/01/2022
YouTube om LSTM ● Research	Research	osci	0:53:14	11:03 AM-11:56 AM 04/01/2022
ML gjennomgang med Vegard ● Generelt	Annet	osci	6:22:00	11:57 AM-06:19 PM 04/01/2022

Frontend design ● Utvikling	Utvikling	osci	10:19:00	08:38 AM-06:57 PM 04/02/2022
Spring boot refactoring ● Utvikling	Utvikling	Simen_refsland	0:42:54	11:20 AM-12:03 PM 04/03/2022
Spring boot refactoring ● Utvikling	Utvikling	Simen_refsland	0:13:20	12:35 PM-12:48 PM 04/03/2022
Spring boot refactoring ● Utvikling	Utvikling	Simen_refsland	0:43:36	02:06 PM-02:50 PM 04/03/2022
Orientering i prosjekt ● Generelt	Utvikling	osci	1:30:01	08:00 AM-09:30 AM 04/04/2022
Spring boot API refactoring ● Utvikling	Utvikling	Simen_refsland	7:44:18	08:46 AM-04:31 PM 04/04/2022
Frontend navngi prediksjons kolonner ● Utvikling	Utvikling	osci	3:37:08	09:35 AM-01:12 PM 04/04/2022
Neural Networks forskning ● Research	Research	osci	1:47:26	01:13 PM-03:00 PM 04/04/2022
Spring boot API refactoring ● Utvikling	Utvikling	Simen_refsland	0:17:56	05:08 PM-05:26 PM 04/04/2022
Frontend design og planlegging ● Utvikling	Utvikling	osci	7:24:21	08:19 AM-03:43 PM 04/05/2022
Spring boot API testing + CI ● Utvikling	Utvikling	Simen_refsland	8:54:28	08:40 AM-05:34 PM 04/05/2022
Neural Networks læring ● Research	Research	osci	3:44:53	09:31 AM-01:16 PM 04/06/2022

CI + Optuna ML Utvikling	Utvikling	Simen_refsland	6:41:19	10:00 AM-04:41 PM 04/06/2022
Neural Networks læring Research	Research	osci	3:08:40	02:24 PM-05:33 PM 04/06/2022
CI + Optuna ML Utvikling	Utvikling	Simen_refsland	0:47:09	05:05 PM-05:52 PM 04/06/2022
Møte med Tom Generelt	Annet	osci	0:59:28	09:00 AM-10:00 AM 04/07/2022
Optuna ML + møter Utvikling	Utvikling	Simen_refsland	6:52:06	09:30 AM-04:22 PM 04/07/2022
Bytte fra RelativeLayout til ConstraintLayout i Station Details Utvikling	Utvikling	osci	3:08:39	10:00 AM-01:09 PM 04/07/2022
ML gjennomgang med Vegard Generelt	Annet	osci	0:36:37	01:09 PM-01:46 PM 04/07/2022
Frontend detaljer Utvikling	Utvikling	osci	0:58:44	01:46 PM-02:45 PM 04/07/2022
Frontend detaljer Utvikling	Utvikling	osci	1:33:02	03:08 PM-04:41 PM 04/07/2022
API testing + Docker Utvikling	Utvikling	Simen_refsland	8:14:44	09:02 AM-05:17 PM 04/08/2022
Lese rapport fra NTNU open Rapport	Rapport	osci	1:51:01	08:06 AM-09:57 AM 04/11/2022
Docker + Rapportskrivning start Utvikling	Utvikling	Simen_refsland	7:14:48	09:28 AM-04:43 PM 04/11/2022

rapport mal ● Rapport	Rapport	osci	1:55:48	10:23 AM-12:19 PM 04/11/2022
rapport TEORI ● Rapport	Rapport	osci	1:59:42	12:20 PM-02:20 PM 04/11/2022
Orientering i prosjekt ● Generelt	Annet	osci	0:18:29	02:21 PM-02:39 PM 04/11/2022
FrontEnd justering og Graf oppdatering ● Utvikling	Utvikling	osci	5:57:34	09:17 AM-03:14 PM 04/12/2022
Forbedre frontend + API testing ● Utvikling	Utvikling	Simen_refsland	6:57:33	10:31 AM-05:29 PM 04/12/2022
Optuna + Demo + Dokumentasjon ● Utvikling	Utvikling	Simen_refsland	6:53:15	09:28 AM-04:21 PM 04/13/2022
FrontEnd Detaljer ● Utvikling	Utvikling	osci	7:18:27	10:46 AM-06:04 PM 04/13/2022
Jupyter Notebook lokalt setup + Optuna ● Utvikling	Utvikling	Simen_refsland	7:44:53	09:30 AM-05:15 PM 04/15/2022
Optuna ML ● Utvikling	Utvikling	Simen_refsland	2:38:44	02:30 PM-05:08 PM 04/16/2022
Orientering i prosjekt ● Generelt	Annet	osci	1:56:26	05:18 AM-07:14 AM 04/19/2022
RAPPORT: Maskinlæring research ● Rapport	Rapport	osci	4:11:52	07:15 AM-11:26 AM 04/19/2022
Maskinlæring Optuna ● Utvikling	Utvikling	Simen_refsland	8:52:47	09:00 AM-05:52 PM 04/19/2022

ML multi-step ● Utvikling	Utvikling	Simen_refsland	7:54:52	09:15 AM-05:09 PM 04/20/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	7:52:48	09:30 AM-05:22 PM 04/21/2022
Rapportskriving + Frontend finpuss ● Rapport	Utvikling	Simen_refsland	8:08:58	08:30 AM-04:39 PM 04/23/2022
frontend kart ● Utvikling	Utvikling	osci	4:28:59	10:44 AM-03:13 PM 04/23/2022
Frontend finpuss + rapportskriving ● Utvikling	Utvikling	Simen_refsland	8:07:40	08:45 AM-04:53 PM 04/24/2022
Rapportskriving teori ● Rapport	Rapport	Simen_refsland	8:25:51	08:40 AM-05:06 PM 04/25/2022
RAPPORT: Teori ● Rapport	Rapport	osci	7:00:47	09:47 AM-04:48 PM 04/25/2022
Rapport teori + møte med Headit ● Rapport	Rapport	Simen_refsland	8:03:54	08:47 AM-04:51 PM 04/26/2022
Gruppemøte ● Generelt	Annet	osci	0:40:47	11:00 AM-11:41 AM 04/26/2022
RAPPORT: Teori ● Rapport	Rapport	osci	7:02:07	11:41 AM-06:43 PM 04/26/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	5:41:53	07:47 PM-01:29 AM 04/26/2022 - 04/27/2022
RAPPORT: frontend ● Rapport	Rapport	osci	7:57:22	10:03 AM-06:00 PM 04/27/2022

RAPPORT ● Rapport	Rapport	osci	9:09:41	09:00 AM-06:09 PM 04/28/2022
Møte med veileder + rapportskriving ● Generelt	Annet	Simen_refsland	6:42:15	10:30 AM-05:12 PM 04/28/2022
Møte med headit ● Generelt	Annet	Simen_refsland	0:57:52	05:21 PM-06:19 PM 04/28/2022
RAPPORT: research ● Rapport	Rapport	osci	6:33:00	08:10 AM-02:43 PM 04/29/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	5:49:22	09:56 AM-03:46 PM 04/29/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	0:59:29	11:44 AM-12:44 PM 04/30/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	4:59:41	11:57 AM-04:57 PM 05/01/2022
RAPPORT: lesing ● Rapport	Rapport	osci	5:23:00	01:06 PM-06:29 PM 05/01/2022
RAPPORT: Diskusjon ● Rapport	Rapport	osci	6:08:26	08:08 AM-02:17 PM 05/02/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	7:43:13	09:15 AM-04:58 PM 05/02/2022
RAPPORT: diskusjon ● Rapport	Rapport	osci	4:27:02	02:27 PM-06:54 PM 05/02/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	3:20:03	09:18 AM-12:38 PM 05/03/2022

app ferdigstilling ● Utvikling	Utvikling	osci	8:09:12	09:18 AM-05:27 PM 05/03/2022
RAPPORT: brukergrensesnitt ● Rapport	Rapport	osci	7:51:00	08:08 AM-03:59 PM 05/04/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	3:47:42	10:23 AM-02:11 PM 05/04/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	2:02:55	02:51 PM-04:54 PM 05/04/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	3:23:02	09:04 AM-12:27 PM 05/05/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	3:04:35	01:58 PM-05:03 PM 05/05/2022
RAPPORT: Diskusjon ● Rapport	Rapport	osci	7:20:05	02:32 PM-09:52 PM 05/05/2022
RAPPORT: Diskusjon, utviklingsmiljø ● Rapport	Rapport	osci	8:28:00	08:27 AM-04:55 PM 05/06/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	3:10:05	09:22 AM-12:32 PM 05/06/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	1:06:40	01:04 PM-02:11 PM 05/06/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	6:29:29	09:57 AM-04:26 PM 05/07/2022
Dokumentasjon kode + rapportskrivning ● Rapport	Rapport	Simen_refsland	6:00:11	10:00 AM-04:00 PM 05/08/2022

RAPPORT: Diskusjon & Research til avsnittemner ● Rapport	Rapport	osci	7:28:02	08:37 AM-04:05 PM 05/09/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	7:57:43	09:02 AM-05:00 PM 05/09/2022
RAPPORT: diskusjon ● Rapport	Rapport	osci	0:55:59	04:06 PM-05:02 PM 05/09/2022
RAPPORT: Implementering ● Rapport	Rapport	osci	9:23:21	09:18 AM-06:42 PM 05/10/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	0:44:22	10:20 AM-11:04 AM 05/10/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	2:28:30	12:48 PM-03:17 PM 05/10/2022
Rapportskriving ● Rapport	Rapport	Simen_refsland	5:14:47	09:06 PM-02:21 AM 05/10/2022 - 05/11/2022
RAPPORT: Prediksjoner ● Rapport		osci	7:17:02	09:27 AM-04:44 PM 05/11/2022
Rapportskriving + ML ● Rapport	Rapport	Simen_refsland	3:16:26	12:30 PM-03:46 PM 05/11/2022
RAPPORT: Implementasjon ● Rapport		osci	8:33:01	08:21 AM-04:54 PM 05/12/2022
ML + Rapportskriving + Møte ● Rapport	Rapport	Simen_refsland	7:08:20	09:40 AM-04:49 PM 05/12/2022
Jobbe med tilbakemeldinger fra Tom ● Rapport		osci	6:36:00	09:21 AM-03:57 PM 05/14/2022

Rapportskriving ● Rapport	Simen_refsland	6:28:28	10:16 AM-04:44 PM 05/14/2022
Forbedring ML ● Utvikling	Simen_refsland	2:05:24	05:46 PM-07:52 PM 05/14/2022
Forbedring ML ● Utvikling	Simen_refsland	5:57:06	10:52 AM-04:49 PM 05/15/2022
brukergrensesnitt i kapittel 10 ● Rapport	osci	7:24:00	07:37 AM-03:01 PM 05/16/2022
Forbedring ML ● Utvikling	Simen_refsland	2:09:40	12:21 PM-02:31 PM 05/16/2022
Forbedring ML ● Utvikling	Simen_refsland	2:20:55	03:14 PM-05:35 PM 05/16/2022
Forbedring ML ● Utvikling	Simen_refsland	1:04:36	07:27 PM-08:31 PM 05/16/2022
Forbedring ML ● Utvikling	Simen_refsland	6:01:19	12:10 PM-06:11 PM 05/17/2022
Forbedring ML ● Utvikling	Simen_refsland	3:15:51	08:32 PM-11:48 PM 05/17/2022
Rapport konklusjon og diskusjon ● Rapport	osci	7:19:32	09:49 AM-05:08 PM 05/18/2022
Forbedring ML ● Utvikling	Simen_refsland	4:38:34	11:06 AM-03:44 PM 05/18/2022
ML forbedring ● Utvikling	Simen_refsland	4:33:03	05:36 PM-10:09 PM 05/18/2022

Rapport ferdigskriving ● Rapport	Simen_refsland	7:17:00	09:43 AM-05:00 PM 05/19/2022
Siste dag på bacheloroppgave! ● Rapport	osci	8:55:18	10:27 AM-07:23 PM 05/19/2022
Rapport ferdigskriving ● Rapport	Simen_refsland	2:31:42	06:37 PM-09:09 PM 05/19/2022
Rapport ferdigskriving ● Rapport	Simen_refsland	4:42:04	09:09 PM-01:51 AM 05/19/2022 - 05/20/2022

Vedlegg F

Møtereferater

Vedlagt ligger de fleste møtereferater skrevet i prosjektperioden.

Møtereferat 12.01.2022

Tid/sted: 12.01.2022 12:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Vegard Ølstad Dalberg, Martin Holthe Rønningen

Referent: Simen Refsland

Agenda

Sak 1/2022: Introduksjon

- Gjennomgang av oppgaven stilt av oppdragsgiver, forklare vår oppfatning av oppgaven
- Sett på Drivstoffappen, som gruppa skal samarbeide med.
- Tidligere relevant erfaring (tidligere emner, prosjekter etc.)

Sak 2/2022 og Sak 3/2022: Roller/forståelser av roller og forståelse av oppgaven, scope/skalering

- Skal videreutvikle funksjonalitet som Drivstoffappen allerede har.
- TensorFlow, predikere drivstoffpriser.
- Tilgang til database.
- Skal i første omgang klare å hente data fra databasen, bruke dette til å utføre prediksjon og deretter oppdatere database med predikert pris.
- Planlegger nytt møte hvor utviklerne fra Drivstoffappen er med.

Sak 4/2022: Ønske om faste møter, oppfølging evnt. neste møtepunkt

- Har bestemt at møter innkalles ved behov, mindre spørsmål oppfordres til å stilles fortløpende.
- Det blir sannsynligvis nytt møte i begynnelsen av neste uke med Drivstoffappen-utviklerne.

Sak 5/2022: Eventuelt

- Kontrakt mellom studentene, veileder og oppdragsgiver må signeres, blir sannsynligvis sendt ila. uka. Skal tas opp med veileder torsdag 12:30.

Møte avsluttet 12:30

Møtereferat 13.01.2022

Tid/sted: 13.01.2022 12:30, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 6/2022: Introduksjon

- Alle fortalte litt om seg selv, bakgrunn, interesser etc. Informasjon om hvilke emne som har blitt tatt / kompetanse, samt sterke / svake sider med tanke på emnene som er tatt.

- Oppfordring om å melde seg på emne PROG2051 – Kunstig Intelligens, ettersom vi skal predikere priser. Skal vurderes.

Sak 7/2022: Oppdatering fra oppdragsgiver

- Oppdatering fra møtet med oppdragsgiver 12.02.2022. Melde ifra om bytte av kontaktperson, som nå er Martin Holthe Rønningen.

Sak 8/2022: Prosjektplan

- Tips om hvordan gruppa kan jobbe med prosjektplanen. Anbefaler å ta noen punkter for hvert medlem og så bytte og komme med forbedringer.
- Har bestemt å bruke Toggl som verktøy for å føre timelister, ettersom at dette er mindre tungvint enn å bruke regneark, samt at verktøyet kan enkelt generere rapporter fra gitte tidsintervaller.

Sak 9/2022: Diverse spørsmål

- Spørsmål om kontrakt, fikk beskjed om hvordan denne skulle utformes. Vanligvis vil standardkontrakten være tilstrekkelig, kan bruke ekstra kontrakt om konfidensialitet om oppdragsgiver ønsker dette. Fikk beskjed om at lukket kildekode kan bli fjernet fra offentlig publisering.
- Spørsmål om bruk av systemutviklingsverktøy, ble anbefalt å spørre oppdragsgiver om hva de ønsker at gruppa skal bruke.

Møtet avsluttet 13:10

Møtereferat 14.01.2022

Tid/sted: 14.01.2022 10:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Syver Orhagen, Sander Helgesen, Vegard Ølstad Dalberg, Martin Holthe Rønningen

Referent: Simen Refsland

Agenda

Sak 10/2022: Introduksjon

- Introduksjon med utviklerne med Drivstoffappen. Oppdatering og gjennomgang av oppgaven.
- Informasjon om utviklermiljø og programmeringsspråk Drivstoffappen bruker, bruker Kotlin for Android, Swift for iOS-appene.

Sak 11/2022: Diskusjon av oppgaven

- Enighet om å fokusere på selve drivstoffpredikasjonen i første omgang, lage API for å predikere priser.
- Alle får ila. kort tid en oppdatert oppgavebeskrivelse og ser om alle parter er enig / forstår denne.

- Usikkerhet rundt hvorvidt gruppa skal ta på seg å integrere tjenesten i selve appen eller om utviklerne i Drivstoffappen skal gjøre dette, blir sannsynligvis klarere etter hvert.

Møtet avsluttet 11:00

Møtereferat 21.01.2022

Tid/sted: 21.01.2022 11:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 12/2022: Oppdatering etter møte med HeadIT

- Oppdatering etter møte med HeadIT og Drivstoffappen-utviklerne. Ga veileder tilgang på oppdatert tentativ oppgavebeskrivelse.
- Usikkerhet rundt frontend-biten av prosjektet, ga beskjed om at dette forhåpentligvis oppklares på møte mandag 24. januar.
- Har fått forslag fra oppdragsgiver om bruk av ZenHub som integreres i GitHub, vil sannsynligvis bruke dette.

Sak 13/2022: Diverse spørsmål rundt prosjektplan

- Spørsmål rundt bruk av systemutviklingsmodell, veileder anbefaler å bruke Scrum, slik at oppdragsgiver får jevnlige oppdateringer og kan gi input på hva som skal fokuseres på i hver sprint.
- Spørsmål rundt Gantt-diagram, får beskjed om at sprintene kan defineres, men ikke innhold, dette blir avklart etter hvert avhengig av hva som trengs. Aktivitet utenfor sprintene kan også defineres. Kan definere planlagt aktivitet de første ukene nærmere.

Sak 14/2022: Eventuelt

- Foreløpig plan er å starte første sprint 1. februar etter innlevering av forprosjektsplan.
- Plan om å sende inn utkast til prosjektplan ett døgn før neste møte 27. januar for å få input før levering.
- Viktig å sette seg inn i utviklermiljø/rammeverk/teknologi vi skal bruke før sprinten starter, da spesielt med tanke på TensorFlow i Python.

Møtet avsluttet 11:35

Møtereferat 24.01.2022

Tid/sted: 24.01.2022 15:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Martin Holthe Rønningen, Vegard Ølstad Dahlberg

Referent: Simen Refsland

Agenda

Sak 15/2022: Oppdatering

- Første sprint er bestemt at skal være fra 1. februar til 14. februar.
- Jobber med prosjektplanen frem til 31. januar.
- Har tenkt til å bruke ZenHub som scrum-verktøy, dette integreres i Github, som blir repository for gruppa.

Sak 16/2022: Oppklaring og spørsmål

- Spørsmål om gruppa skal få en maskinlæringsmodell innledningsvis. Vegard har laget en modell og har avtalt å gi en innføring i dette i et nytt Teams-møte 1. februar 15:00.
- Gruppa har ikke fått tilgang til Drivstoffappens repository, har fått oppklart at dette er nødvendig for å implementere frontend-løsning. Gruppa vil i utgangspunktet implementere frontend-løsningen for Android-versjonen av Drivstoffappen skrevet i Kotlin.
- Oppklaring om API, APIet skal bare brukes til å hente predikerte priser som maskinlæringsmodulen legger inn i databasen, kan dermed bruke noe annet enn Python til å lage API, f.eks. Node.js.
- Gruppa står selv frie til å sette sprintmål, vil i første sprint fokusere på å få satt opp et API og få innføring i maskinlæringsmodellen.

Møtet avsluttet 15:20

Møtereferat 27.01.2022

Tid/sted: 27.01.2022 12:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 17/2022: Gjennomgang av prosjektplan / tips til utbedring

- Diskusjon av systemutviklingsmodell, gruppa har blitt enige om en sprintlengde på 10 arbeidsdager som begynner på mandager og slutter på fredager.
- Gruppa burde bruke siste sprintdag til å gå gjennom hva som har blitt oppnådd og hva som skal gjøres i neste sprint med oppdragsgiver.
- Issues i backlogen burde kategoriseres grovt inn i estimert tidsbruk (small, medium, large) for å kunne bedre planlegge hvor mange oppgaver som gruppa kan ta på seg i hver sprint.
- Lag en testplan som gir et grovt estimat av når disse skal foregå. For eksempel å begynne med akseptanse- og brukertester etter sprint 6 eller liknende.
- Fagområde burde ikke gjelde bruk av teknologi, men hvilket fagområde vår løsning har nytte av.

Møte avsluttet 13:25

Møtereferat 01.02.2022

Tid/sted: 01.02.2022 15:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Vegard Ølstad Dahlberg, Martin Holthe Rønningen

Referent: Simen Refsland

Agenda

Sak 18/2022: Gjennomgang ML-modell

- Gruppen har fått tilgang til Jupyter-notebook i tillegg til en serialisert maskinlæringsmodell. I notebooken er det en innføring i hvordan en burde forberede data til maskinlæringsmodellen hentet fra database. Den viser også teknikker for maskinlæring, slik som CNN (Convolutional Neural Network), LTSM (long short-term memory).
- Gruppen har fått noen lenker som kan være nyttige (lagret internt for senere lesing) og anbefaling av en bok *Deep Learning* av Ian Goodfellow, Yoshua Bengio og Aaron Courville, som gir en mer teoretisk innsikt til maskinlæring.
- Gruppen har fått større innsikt i hvordan en modell kan trenes ved hjelp av dataene fra databasen.

Sak 19/2022: Plan for sprinten

- Gruppen har plan om å få integrert maskinlæringsmodellen som er gitt av Vegard i første omgang og få den til å funke sammen med API. Senere kan det ses på enklere rammeverk, slik som Prophet, som brukes til prediksjon. Antas at det må allokeres tid til å lære en del om maskinlæring.

Møte avsluttet 16:08

Møtereferat 03.02.2022

Tid/sted: 03.02.2022 13:05, NTNU Gjøvik K205B

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 20/2022: Tilbakemelding på prosjektplan

- Prosjektplan er godkjent.
- Ikke gå for langt på hva som må forklares i akronymer og ordforklaringer (HTML er på grensen).
- Effektmål kunne vært mer spesifikk på hvilket konkurransefortrinn Drivstoffappen får av å ha en implementert prediksjonstjeneste.
- Ikke forveksle rammer og krav, en del av det som står som rammer er fint å ha som krav i en annen del. Eksempel på generelle rammer er valg av programmeringsspråk o.l.
- Kan utdype samarbeidet mellom Headit og Drivstoffappen.

- Kan konkretisere hvordan bruker av appen opplever tjenesten. Ikke oppgi eksakt predikert pris eller begrep som konfidensintervall, men noe alle brukere forstår og har nytte av.
- Konkretiser hvilke deler av Scrum-hendelser gruppen skal bruke og hvordan. Hvordan blir dette i så fall praktisert? Knytt dette opp mot "teorien" forklart. Forklar hvorfor visse sprintlengder velges.
- Bli mer klar på hvordan brukertestingen skal foregå.
- Når det kommer til grupperegler, vær mer klar på hvordan konflikter i gruppen skal håndteres når noen ikke gjør som avtalt.
- Fagområde i omfang burde heller handle om drivstoffmarkedet, hva som påvirker prisen, oljepris, relevans for forbrukere og liknende.
- Risikomatrise er kanskje ikke nødvendig. Risikoer markert grønn trenger kanskje ikke forebyggende tiltak.

Sak 21/2022: Eventuelt

- Møter med veileder fortsetter hver torsdag rundt kl. 12.

Møte avsluttet 13:50

Møtereferat 10.02.2022

Tid/sted: 10.02.2022 13:00, NTNU Gjøvik K205B

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 22/2022: Oppdatering

- Gruppen har nå fått tilgang til repository for API og Android applikasjon. Testdatabase er også satt opp.
- Begge repository mangler vesentlig dokumentasjon, så gruppen har brukt litt tid på å sette seg inn i og forstå API'et.

Sak 23/2022: Videre råd

- Oppfordring om å skrive tester før koden blir skrevet. Fordelen med dette er at koden da forhåpentligvis blir skrevet uten større feil første gangen.
- Tenk på hvordan predikasjonen skal bli fremstilt. Mulig å fremstille hvor det er predikert billigst i et større geografisk område, slik som Hamar.
- Samle data om treffsikkerheten geografisk. Per nå vil modellen sannsynligvis treffe bedre i Oslo-området enn ellers. Sammenlikne mellom predikert og oppdaterte verdier etterpå.

Møte avsluttet 13:30

Møtereferat 14.02.2022

Tid/sted: 14.02.2022 14:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Martin Holthe Rønningen

Referent: Simen Refsland

Agenda

Sak 24/2022: Status sprint

- Tilgang på repo har tatt litt tid, og gruppen har brukt tid på å sette seg inn i Spring Boot-rammeverket. Gruppen hadde ikke fått info om at et allerede eksisterende API skulle brukes, derfor tok det litt tid å sette seg inn i koden og få satt opp prosjektet i utviklingsmodus.
- Oppgavene som tilhørte forrige sprint videreføres til neste sprint, med mål om en demonstrasjon av fungerende endpoint på prediksjon på slutten av neste sprint (28. februar). Ideellt sett vil ML-modell da kunne predikere automatisk etter «scheduling» på alle stasjoner.

Sak 25/2022: Eventuelt

- Gruppen anbefales å begynne å tenke på design og diagrammer som viser arkitektur i prosjektet for å få bedre oversikt og lette oppgaveskrivingen senere.
- Det blir prioritert å få backenden på plass i starten, slik at ML-modellen kan videreutvikles og frontend-implementasjonen kan begynne.

Møte avsluttet 14:20

Møtereferat 24.02.2022

Tid/sted: 24.02.2022 14:00, A217 NTNU Gjøvik

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 26/2022: Status

- Gruppen antar å kunne vise en demo som viser API'et og at maskinlæringsmodellen automatisk gjør nye prediksjoner for hver dag.
- Videre tenkes det å designe og implementere brukergrensesnittet, samt prøve å videreutvikle maskinlæringsmodellen.
- Det anbefales å jobbe systematisk med forskjellige tilnærmingsmåter og reflektere over hvorfor én «approach» fungerer bra eller dårlig når vi prøver å forbedre modellen.
- Det er viktig å fokusere på brukernes forventninger når en lager brukergrensesnittet, vurdere prismarginer i stedet for konkrete priser.

Møte avsluttet 14:23

Møtereferat 28.02.2022

Tid/sted: 28.02.2022 14:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Martin Holthe Rønningen, Vegard Ølstad Dalberg

Referent: Simen Refsland

Agenda

Sak 27/2022: Demo

- Gruppen har vist frem en demo av det som har blitt gjort foreløpig, som vil si et funksjonelt API og en grunnleggende modell som predikerer priser.
- Hovedmålet i neste sprint blir å videreutvikle Android-appen, slik at prediksjonene kan vises og visualiseres. Dette for å kunne fokusere mer på maskinlæringsdelen frem mot deadline samt kunne gjøre endringer underveis.
- Gruppen anbefales å se mer på LSTM-modeller (Long short-term memory) for å forbedre resultatene.

Møte avsluttet 14:25

Møtereferat 24.03.2022

Tid/sted: 24.03.2022 14:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Martin Holthe Rønningen, Vegard Ølstad Dalberg

Referent: Simen Refsland

Agenda

Sak 28/2022: Demo

- På grunn av problemer med lokalt oppsett (utdatert database), har ikke gruppen fått ferdigstilt frontend til Android foreløpig, nytt møte har blitt satt opp neste torsdag. Nytt møte i forbindelse med maskinlæring er satt opp påfølgende fredag.
- Siden nytt design ikke har blitt påbegynt av Drivstoffappen, blir UI implementert i gammel versjon.

Sak 29/2022: Innspill til utforming av UI

- På grunn av begrenset plass for å visualisere prediksjoner, anbefales det å se på alternativer hvor enkelt komponenter kan forminskes for å frigi plass.
- Det viktigste for bruker vil være å vite hvilken dag som er billigst. Et alternativ er å implementere en scroll-pane hvor brukeren kan bla gjennom priser for ulike dager. Et annet alternativ er å lage en graf som visualiserer priser hvor x-aksen viser dager og y-aksen viser priser.
- Et lovende kompromiss er å introdusere to faner som viser henholdsvis scroll-panen og grafen. Slik kan brukeren få info på to måter.

Sak 30/2022 Maskinlæring

- Siden sist er det funnet en datakilde for historiske oljepriser i dollar per fat. Motivasjonen bak dette er å øke antall features som modellen kan dra nytte av.

- Mulige endringer som kan utprøves er endring i aktiveringsfunksjoner, forskjellige antall lag og forskjellig antall noder. Kan også utforske metoder for interpolasjon for å fylle inn manglende data.

Møte avsluttet 14:45

Møtereferat 25.03.2022

Tid/sted: 25.03.2022 10:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 31/2022: Oppdatering

- Det blir informert om at det ikke har vært en spesielt stor fremgang siden sist, på grunn av en intensiv periode med INGG2300, men at gruppa er fristilt til å jobbe hundre prosent med bachelor nå.
- Det har vært litt forvirring med Drivstoffappen, siden en grafisk oppdatering skulle vært utrullet for en stund siden. På grunn av tidshensyn, skal det i stedet jobbes med å oppdatere den gamle versjonen.

Sak 32/2022: Rapportskriving

- Frem til neste uke skal det jobbes utelukkende med utvikling av frontend og maskinlæringsmodell, men etter dette planlegges det å gi rapportskriving mer rom per uke.
- Et utkast av rapporten planlegges å leveres innen 13. april, hvor det tenkes at visse del(er) vil være 80 prosent ferdige, mens andre deler vil være mer en skisse.

Sak 33/2022: Råd/tips

- I forbindelse med frontendutvikling, blir det anbefalt å se på teori og «best practices» for brukersentrert design. Dette vil gi et bedre grunnlag både rapporten og for å vise læringsutbyttet av prosessen.
- Det samme gjelder maskinlæringsmodellen, kan være lurt å se på andre fremgangsmåter og metoder, og sammenlikne, samtidig som en argumenterer for hvorfor en bestemt fremgangsmåte velges.

Sak 34/2022: Eventuelt

- Nytt møte er blitt satt til torsdag 7. mars.

Møte avsluttet 10:20

Møtereferat 31.03.2022

Tid/sted: 31.03.2022 13:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Martin Holthe Rønningen

Referent: Simen Refsland

Agenda

Sak 35/2022: Demo

- Det har blitt i stor grad ferdigstilt brukergrensesnitt for Android-appen. Appen kan nå vise prediksjonene for en gitt stasjon i en tabellvisning, i tillegg til en grafvisning. Selve designet vil sannsynligvis forbedres noe, og grafen vil også modifiseres.
- Det har blitt anbefalt å vise i kart hvor billigste prediksjon for en bensinstasjon er, slik at det er lett for bruker å se hvor en bør fylle. Kan for eksempel være billigste stasjon i en mils radius eller liknende.
- Det anbefales å begynne å fokusere mer på maskinlæringsmodellen og begynne å skrive på rapporten. Dette er viktig fordi det er lett å havne bakpå hvis det ikke finnes en balanse mellom disse to. Forslag til arbeidsflyt er å fordele oppgavene i en 3/2-fordeling i en uke, hvor 3 av dagene går til maskinlæringsmodell og 2 av dagene går til rapportskriving.
- Nytt møte blir sannsynligvis innkalt etter møte mellom Headit og Drivstoffappen førstkommende mandag.

Møte avsluttet 13:12

Møtereferat 01.04.2022

Tid/sted: 01.04.2022 13:00, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Vegard Ølstad Dalberg

Referent: Simen Refsland

Agenda

Sak 36/2022: ML-gjennomgang

- Gruppen har vist fremgangen på maskinlæringsdelen hittil. Modeller som har blitt utprøvd hittil er en lineær modell, convolutional neural network (CNN), long short term memory (LSTM) og dense neural network (DNN). Hittil har en del av modellene hatt tendenser med overtilpasning når testdataene brukes.
- I forbindelse med dette er det usikkerhet rundt hvor mange lag og nevroner i hvert lag som er gunstig i nevrale nettverk for tilsvarende problem.
- Bensinprisene har også blitt mer volatile i forbindelse med nylige hendelser i verden som påvirker olje og gass.
- Vegard har sent ut et skript som tester forskjellige hyperparametre (antall noder, lag, aktiveringsfunksjoner etc.). Med dette er håpet at modellene kan forbedres ved hjelp av utprøving av forskjellige hyperparametre. Dette skriptet drar nytte av Optuna, som er et rammeverk for optimalisering av hyperparametre.
- Neste møte er avtalt onsdag 6. april.

Møte avsluttet 14:00

Møtereferat 07.04.2022

Tid/sted: 07.04.2022 10:30, Teams-møte

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 37/2022: Status

- Det har blitt kjørt en demo med Headit, hvor foreløpig frontend har blitt vist, som hadde god respons.
- Møtet med Vegard var også nyttig, hvor det ble sendt et skript som automatisk justerer hyperparameterne i en maskinlæringsmodell.
- Ny frist for innlevering av rapportutkast er nå 26. april.
- Neste møte blir 28. april kl. 10:30, hvor rapportutkastet vil gås gjennom og mulige forbedringer blir diskutert.
- Gruppa har signert konfidensialitetsavtale, som gjør at kildekoden antageligvis må unntas publikasjon, men rapporten vil sannsynligvis være mulig å publisere. Det blir opplyst at visse deler av rapporten kan redigeres bort.

Møte avsluttet 10:59

Møtereferat 28.04.2022

Tid/sted: 28.04.2022 10:30, NTNU Gjøvik

Til stede: Simen Refsland, Oscar Brøndelsbo, Tom Røise

Referent: Simen Refsland

Agenda

Sak 38/2022: Tilbakemelding på utkast av rapport

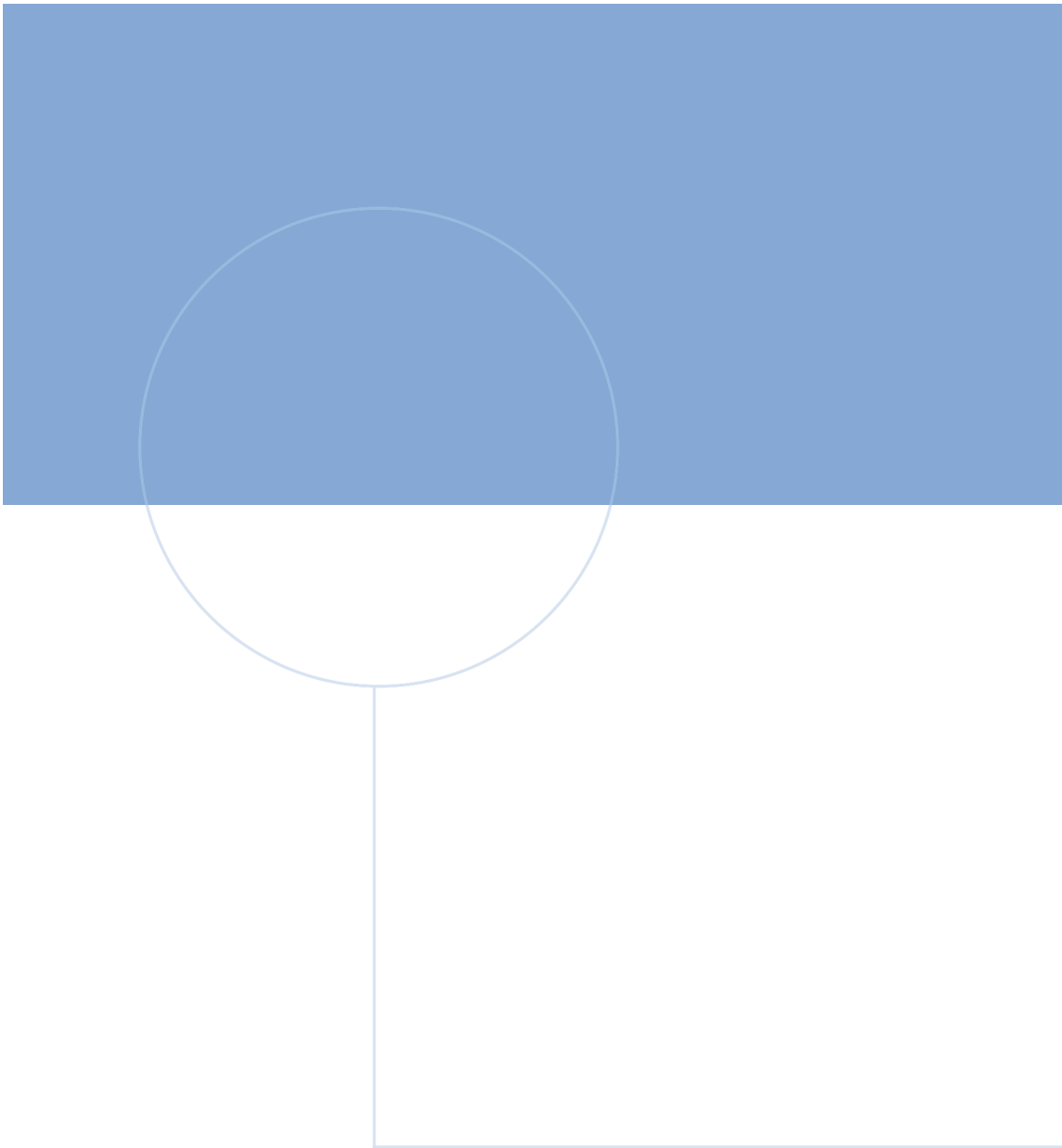
- Generelt
    - Unngå muntlig språk i rapporten, prøv å opprettholde akademisk språk.
    - Unngå å skrive «vi skal gjøre» og heller skrive «vi har gjort».
    - Legg til design og arkitektur før implementasjon.
    - Les tidligere gode rapporter.
- Introduksjon
    - Jobb med å «selge» oppgaven i større grad enn til nå, tenk at leseren har lite kontekst å forholde seg til på forhånd.
    - Vurder å endre på nummering i inndeling (bruk * i visse seksjoner slik at nummering ikke dukker opp).
    - Resultatmål før prosjektbeskrivelse, muligens legge på en figur som illustrerer prosjektets bakgrunn.
    - Om fagområde, skriv mer om hvem som er målgruppa, hvorfor folk er opptatt av drivstoffpriser, konkurransesituasjon o.l.

- Teori
    - Utdyp mer, det blir litt for kort og konsist på noen punkter. Eventuelt kan det vinkles mer mot hvorfor det er relevant i prosjektet.
    - Mer om hvorfor ReLU er relevant for eksempel.
    - Ha en struktur hvor nettverkstypene henger logisk sammen. F. eks. gi en innledning som forklarer at videre nettverkstyper er spesialisering av det mer generelle.
    - Kanskje si noe om prediksjoner.
    - Hvis API skal stå der, gjør det mer personlig. Evt. kan dette flyttes til implementasjon.
- Kravspesifikasjon
    - Ikke gi en beskrivelse av de use-casene som allerede finnes, gi informasjon om hvilke use-caser som faktisk implementeres i prosjektet. Å vise de andre gir kontekst av hvordan brukeren skal benytte seg av appen, og er derfor ført opp.
    - Gi mer detaljerte beskrivelser av use-caser.
    - Si mer om hvordan ML-modulen fungerer, hvordan blir prediksjoner oppdatert, hvor ofte, er det avhengig av brukerinput etc.
    - Mulig med sikkerhetskoder av prediksjoner basert på mengde data, situasjon i verden.
- Utviklingsprosess
    - Ikke forklar hva scrum er, argumenter om hvorfor det blir brukt i prosjektet, hvordan det fungerte i praksis, hvordan sprinter ble definert og hvorfor.
    - Gi en oppsummering av hva som skjedde hver sprint.
- Diskusjon
    - Dra inn faktorer som bidrar til utfordringer i prosjektet, f.eks. hendelser som krigen i Ukraina.
    - Kommunikasjon mellom partene.

Sak 39/2022: Eventuelt

- Nytt utkast av rapport skal leveres til vurdering i løpet av 10. mai, og vil bli gjennomgått i nytt møte torsdag 12. mai 14:00.

Møte avsluttet 11:45



NTNU

Kunnskap for en bedre verden