Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i ingeniørfag / Datateknologi	Vårsemesteret 2022 Åpen/~~Konfidensiel~~
Forfattere: Anders Wethal Pettersen, Erlend Eiring	
Fagansvarlig: Erlend Tøssebro Veiledere: Kjetil Klaussen, Eirik Junge Hess	
Tittel på bacheloroppgaven: Kartlegging av aktiviteter ved sjøbaserte oppdrettsanlegg Engelsk tittel: Mapping activities in aquaculture facilities	
Studiepoeng: 20	
Emneord: Data-analyse Geografisk kartlegging Datahåndtering	Sidetall: 62 + vedlegg/annet: 0 Stavanger 15. mai 2022

Innhold

Sammendrag

Bacheloroppgaven er laget og utgitt av Kontali som et pilotprosjekt. Kontali foreslo denne oppgaven som et konseptbevis, og planlegger utvidelser av systemet for globalt bruk.
Oppgaven har til hensikt å besvare problemstillingen:
Kan man gjøre antagelser på hva slags aktivitet som foregår ved en lokasjon, ved hjelp av geografiske- og AIS-data?

Analysen gjøres på offentlig tilgjengelig informasjon om posisjoner for forskjellige fartøy og maritime lokasjoner.
Oppdragsgiver stilte ingen krav om bruk av spesifikke teknologiske verktøy. Oppgaven ble derfor løst med de verktøyene som tilfredstilte kravene fra problemstillingen på best måte etter gruppens mening.

Rapporten begynner med en grunnleggende forklaring av prosjektets bakgrunn og grunnleggende informasjon om fiskeriindustrien.
Deretter følger informasjon om valg av teknologier, språk og rammeverk valgt for å konstruere applikasjonen. Det vil også bli forklart hva slags alternativer gruppen vurderte, og hvorfor valgene ble tatt.
Etter valg av utviklingsverktøy vil innsamling av data, konstruksjon av applikasjonen og endelige algoritmer for aktivitets antakelser bli gjennomgått.

Dette følges av en gjennomgang av hva slags revurderinger som har blitt tatt underveis, resultatene og refleksjon på prosjektets resultat mot oppgavens mål. Her vil det bli utdypet hvorfor noen av målene med oppgaven ikke ble nådd, og hva som kan gjøres for å oppnå de.

Rapporten avsluttes med konklusjonen, hvor det blir reflektert over hvor-

dan prosjektet gikk i sin helhet, og hvordan gjennomføringen kunne blitt forbedret.

Terminologi

- AIS: Automatisk identifikasjons system, et automatisert system for å si hvor et fartøy er på et gitt tidspunkt[1].
- API: API står for Application Programming Interface og er en kode som brukes for å utveksle data mellom to forskjellige systemer eller apper.[2]
- Breddegrad: En av 2 akser i jordens koordinatsystem, hvilket går parallelt med ekvator[3].
- Brønnbåt: En brønnbåt eller kvase er et spesialfartøy for transport av levende fisk [4].
- Euklidsk distanse: Innen euklidsk geometri er avstanden vanligvis definert ved en 2-norm, som tilsvarer den fysiske avstanden i planet og rommet[5].
- Fôringsraten: Hvor ofte/sjeldent fiskene får mat.
- GIS: En database som inneholder geografiske data, kombinert med programvare for å håndtere og visualisere dataene[6].
- IMO: Den Internasjonale maritime organisasjon. En organisasjon satt opp av FN for å følge opp og bedre kontrollere internasjonal maritim aktivitet[7].
- IMO nummer: Et unikt nummer for å registrere og følge opp individuelle båter[7].
- Lengdegrad: En av 2 akser i jordens koordinatsystem, hvilket går mellom Nordpolen og Sydpolen[8].

- Matfisk: Fisk som er i prosessen av å bli matet til de har en størrelse som er verdt å selge[9].
- Merd: Innhegning i sjøen for oppbevaring, fôring og stell av oppdrettsfisk[10].
- MMSI: Maritim mobil service identitet. Et 9-sifferet nummer brukt for å identifisere skipslokasjoner, kystlokasjoner og andre maritime lokasjoner[11].
- Parsing: Å parse også kalt syntaktisk analyse vil i denne oppgaven bli brukt for å beskrive oppdeling av html-kode, for å hente ut elementer nevnt på en nettside[12].
- Polygon: Ofte kalt mangekant er en geometrisk figur med x antall kanter, en trekant, firkant og femkant kan dermed betegnes som et polygon[13].
- Settefisk: Oppdrett av ung laks, ofte gjort på land før de når en størrelse hvor de kan settes ut på oppdrettslokasjoner[9].
- Sette ut smolt: Når mindre fisker avlet på smoltanlegg blir satt ut på oppdrettslokasjoner.
- Slaktemerd: En slaktemerd er et midlertidig oppholdssted for fisk, hvor fisken kan bli levert før de hentes for slakting[14].
- Smolt er betegnelsen på ungfisk av anadrome laksefisk (laks, ørret og røye) som er klare for utvandring fra ferskvann til saltvann[15].

Kapittel 1

Introduksjon

1.1 Problemstilling

Kan man ved hjelp av geo-koordinater for oppdrettsanlegg, smolt-anlegg og slakteri, samt AIS-data for skipstrafikk for et utvalg fartøy ekstrapolere om, når og hva slags oppdrettsaktivitet som foregår ved anlegget? For eksempel:

1. *Når det settes ut smolt.*
2. *Når fôringsraten er høy/lav.*
3. *Når det behandles for sykdom/lus.*
4. *Når det foretas annet vedlikehold.*
5. *Når fisk tas ut og slaktes.*

1.2 Beskrivelse av oppgaven

Oppgaven går ut på om man kan bruke informasjon om fartøy og dets posisjonshistorikk til å avgjøre hvilket arbeid det utfører.
Det må derfor lages en applikasjon, hvor dens viktigste funksjonalitet er:

1. Muligheten for å definere stopp.
2. Determinere om stoppet er ved en kjent lokasjon.
3. Finn type lokasjon.
4. Anslå aktivitet basert på tidligere stopp.
5. Sette ovennevnte punkter i et system, slik at dette kan implementeres på en større skala.

Hver del av applikasjonen vil bli utdypet og forklart iløpet av oppgaven.

1.3 Om bedriften

Kontali er et analyse-selskap som spesialiserer seg innen akvakultur og fiskeri, og har verdens mest omfattende proprietære database innen feltet[16][1]. Kontali bruker dataene de har samlet gjennom flere år til sine analyser og lager prognoser til den globale sjømatindustrien, samt tilbyr selskaper skreddersydde rapporter for optimal drift[16].

1.4 Grunnlag for oppgaven

Denne oppgaven ble laget av Kjetil Klaussen, CTO for Kontali. Kontali utleverte oppgaven som et konseptbevis, og vil bruke den resulterende applikasjonen videre for å lokalisere relevante lokasjoner for andre geografiske områder (Chile, UK, Canada, etc.) og andre arter (havabbor, havbrasme, tilapia, etc.), og bruke lokasjonene til å gjøre antakelser på aktivitetsforeløp. Dette kan da integreres i deres applikasjoner for å forbedre uthenting av data, og dermed levere flere og mer nøyaktige tjenester i regionene de opererer i.

Det ble satt som mål å se om dette var realiserbart, og å bruke denne informasjonen til å lage en enkel applikasjon Kontali kunne bruke i fremtiden.

[1] Kontali - hjemmeside. https://www.kontali.no/. Accessed: 2022- 01-15[16]

1.4 Grunnlag for oppgaven

For å kunne levere en tilfredstillende applikasjon både for oppgaven og for Kontali sin fremtidige implementering, må relevant informasjon samles inn og lagres for videre behandling. Renskingen av dataene fjerner unødvendig informasjon, noe som kan være forskjellig fra oppgavens mål og Kontali sin ønskede bruk. Derfor blir dataene lagret i de forskjellige formatene som kommer etter forskjellige behandlinger. Da kan minimale endringer av algoritmene endre hvilke data som brukes og hvilke data som forkastes.

Kontali vil benytte en karttjeneste ved egen implementasjon, og gruppen vil også lage et enkelt kart for bruk under utvikling for forståelsebygging og feilsøking. Dette vil også hjelpe Kontali se hvordan systemet er satt opp, og kan dermed implementere sin egen applikasjon utifra denne applikasjonens oppbygning.

Kapittel 2

Bakgrunn

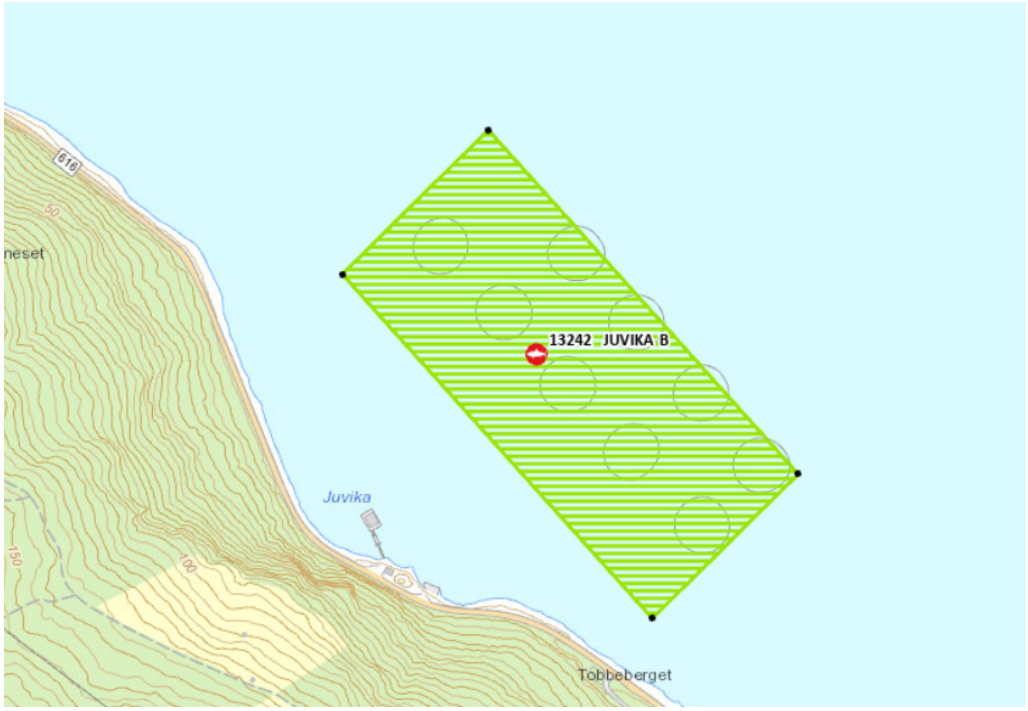
Dette kapittelet forklarer grunnleggende konsepter som må introduseres for å kunne forstå valgene som blir tatt i løpet av prosjekt.

2.1 Geografiske kunnskaper

Dette prosjektet tar for seg geografiske data for både oppdrettsanlegg, fartøy og slakteri. Alle lokasjonene vil ha datapunkter definert som et ukjent antall lengde- og breddegrader. Om et område har flere lengde- og breddegrader tilsier dette at området er et polygon, og alt innenfor dette området er en del av denne spesifikke lokasjonens driftsområde. Et eksempel på et typisk oppdrettslokasjons polygon[17] [1] er vist i figur 2.1. I denne figuren er det fire svarte punkter, disse punktene vil da være fire lengde- og breddegrader som definerer kantene i polygonet.

Området vist i figur 2.1 er et polygon for en oppdretsslokasjon, men en slaktemerd vil se tilsvarende ut.

[1] Fiskedirektoratet, akvakultur karttjeneste." https://portal. fiskeridir.no/portal/apps/webappviewer/index.html?id= 87d862c458774397a84. Accessed: 2022-01-20[17]



Figur 2.1: Polygon for en oppdrettslokasjon utenfor Juvika.[17]

2.2 Fiskerinæringen

Det er flere ledd i fiskerinæringen. De viktigste for denne oppgaven er oppdrettslokasjoner, slaktemerder, slakteri og smoltanlegg.

En oppdrettslokasjon vil være området hvor fisken avles, mens en slaktemerd vil være et polygon utenfor et slakteri, dette området vil holde fisken før den blir sendt videre til slakteriet. Et smoltanlegg har som oppgave å fostre opp fisken før den er stor nok til å bli satt ut i oppdrettslokasjonen sammen med de andre fiskene. En annen viktig del av prosjektet er fartøy. For å kunne bestemme aktiviteter vil flere av fartøyenes datapunkter være nødvendige: lokasjon, fart, båttype, og retning i grader for å se hvilken vei fartøyet er på vei.

Gjennom Kontali har det blitt utlevert historisk posisjonsdata for to fartøy som blir brukt som grunnlag for utbedringen av algoritmene. Disse fartøyene

er:

1. Biostar: Frakteskip for matleveranser.
2. Ronja Ocean: Brønnbåt som driver innen medikamenter, lusebehandling, frakt av smolt og slakteklar fisk.

Ved hjelp av de historiske posisjonsdataene kan man få kvalitetssikret algoritmene slik at de er robuste nok til å brukes i et sanntidssystem.

2.3 Datasamling

Ettersom Norge har åpne systemer med veldokumentert posisjons- og funksjonsopplysninger, var det mange forskjellige datasett tilgjengelig som ble brukt for gjennomføring av oppgaven. Valg av datasett ble en viktig del av oppgaven, ettersom valg av dårlige eller gamle data kunne gjort prediksjonsalgoritmene upålitelige.

Ettersom de åpne offentlige datasettene ville være komplette, var fiskeridirektoratet og mattilsynet gode kilder å begynne med. Etter en kort utveksling med mattilsynet ble det klargjort at all informasjonen de holdt vedrørende slaktemerdenes geografiske midtpunkt allerede var tilgjengelig gjennom fiskeridirektoratet sitt API[18]. [2]

Fiskeridirektoratet hadde også CSV-filer med alle oppdrettslokasjoner, men disse oppdrettslokasjonene manglet eksterne punkter. De eksterne punktene vil da være punktene som definerer polygonene til slaktemerdene eller oppdrettslokasjonene.

Det ble deretter klart at for å oppnå komplette datasett måtte de hentes ut direkte, for så å bruke fiskeridirektoratet sitt API for å finne de tilhørende datapunktene som ikke var tilgjengelig gjennom CSV-filene.

[2]Fiskeridirektoratet - api." https://api.fiskeridir.no/pub-aqua/ api/swagger-ui/index.html?configUrl=/pub-aqua/api/ api-docs/swagger-config#/site-resource/getBordersBySiteNr. Accessed: 2022-01-21[18]

Andre relevante data var åpne og ble funnet ved hjelp av Barentswatch[19] [3]. Barentswatch tilbyr digitale tjenester over tilstander og lokasjoner ute på havet, dette inkluderer informasjon om hvor et skip dro fra og til, mengden fiskesykdom på forskjellige lokasjoner, diverse overvåkede soner i Norge og alle offentlige slakterier[20][4]. Barentswatch holdt koordinatene for slakteriene, men for å koble disse opp mot slaktemerdene måtte det brukes et identifikasjonsnummer, for å så hente tilhørende lokasjoner ved hjelp av Barentswatch sitt API[21][5]. Konfigurasjoner og valg av datasett vil bli begrunnet videre i kapittel 4.

2.4 Målet med oppgaven

Gjennom denne oppgaven er målet først å se om det er mulig å bestemme aktivitet ved bestemte lokasjoner, gitt et utvalgt sett av fartøy. Om dette er mulig, vil informasjonen brukes til å lage en enkel applikasjon som kan sette dette i system, slik at Kontali kan integrere dette i sin egen plattform. Systemet må dermed hente all informasjon som kan være relevant for Kontali, samtidig som all informasjonen nødvendig for oppgaven blir renset og satt i fokus. Dette kan da leveres som et produkt Kontali kan basere sin egen applikasjon på, uten å måtte fjerne irrelevante oppsett og spesifikasjoner brukt i denne applikasjonen.

[3]"Barentswatch - hjemmeside." https://www.barentswatch.no/. Accessed: 2022-01-15.[19]

[4]Barentswatch - Åpne data." https://www.barentswatch.no/ artikler/apnedata/. Accessed: 2022-01-15.[20]

[5]"Barentswatch - aapne data og api." https://www.barentswatch.no/en/articles/open-data-via-barentswatch/. Accessed: 2022-02-03[21]

Kapittel 3

Valg av utviklingsverktøy

I dette kapittelet vil det bli forklart og utdypet hva slags programmeringsspråk, teknologier og utviklingsverktøy som har blitt valgt for å konstruere applikasjonen.

Valgene tatt iløpet av oppgaven har vært basert på flere faktorer, som kjennskap og preferanse, hva applikasjonen skal brukes til, og hvordan den kan bygges videre på etter oppgavens mål er nådd.

Grunnet oppgavens størrelse var det naturlig å velge kjente teknologier framfor å lære seg nye. Det var imidlertid nødvendig å lære noen nye verktøy for å gjennomføre oppgaven.

3.1 Arbeidsplattform: Github

Gjennom utdanningsløpet hos UiS har Github vært den mest brukte arbeidsplattformen, og har dermed gitt gruppen god erfaring med teknologien[22][1]. Github har en av de største brukerbasene i verden, og er en svært anvendelig plattform. Dette gjør bruk av Github i et prosjekt mer effektivt, gitt god dokumentasjon og at mange problemer som kan oppstå har blitt løst

[1]"Github wikipedia page." https://docs.github.com/en/ actions. Accessed: 2022-02-03[22]

tidligere av andre brukere.
En annen grunn til at valget falt på Github var at det var ønskelig at Kontali skulle ha innsyn i hvordan oppgaven gikk underveis.

Utviklermiljøet gruppen valgte å bruke har også god Github-integrering, som både viser til populariteten av Github og brukervennligheten som har utviklet seg gjennom mange år.

Github tilbyr mange forskjellige verktøy for utviklere, som er der for å gjøre integreringen av funksjoner enkel, trygg og automatisert. Et av disse verktøyene er Github Actions, et mye brukt verktøy for utviklerteam. Verktøyet forbedrer arbeidsflyten for grupper, med å automatisere deler av integreringsprosessen.
Gjennom tidligere semester og prosjekter på UiS har Github Actions vært mye brukt. Grunnet størrelsen på gruppen i denne oppgaven ble Github Actions valgt bort. Det er fordi Github Actions er et mer passende verktøy for større team av utviklere som jobber på forskjellig funksjonalitet parallelt. Dette vil ha mindre hensikt når teamet kun består av to personer, da mengden funksjoner som blir jobbet med på samme tidspunkt er begrenset[23][2].

3.2 Backend: Python

Ved valg av språk for backend var det tre valg som stod frem, disse var Python, Go og C#. Dette var grunnet tidligere erfaring gjennom UiS, åpen kildekode og god dokumentasjon. Ettersom frontend ikke var nødvendig, var valg av backend stort sett basert på den direkte effekten språket ville ha på oppgaven.
Frontend var ikke nødvendig ettersom oppgaven omhandler behandling av store datamenger, og trenger ikke brukergrensesnitt. Visualisering av dataene ble bare gjort for utviklingshensyn.

[2]"Github actions dokumentasjon." https://docs.github.com/en/ actions. Accessed: 2022-02-03[23]

3.2.1 Go

Go er et funksjonsbasert kompilert språk designet av Google. Go ble hovedsakelig laget for parallel-programmering som er en av grunnene til at den har bedre kjøretid enn Python[24][3]. Go hadde derimot problemer innen visualisering av data som ville bli nødvendig med tanke på kart.
Go har også relativt dårlig feilhåndtering og pakker for hjelp med koordinatmanipulering.

De ovennevnte grunnene gjorde at Go ikke ble vurdert videre[25][4].

3.2.2 C#

C# er et objektorientert språk designet av Microsoft. Språket ble originalt laget for å ha styrkene til både C++ og Java, samt at det skal være lett å lage webapplikasjoner med[26][5].
C# sine fordeler over Python i sammenheng med applikasjonen var hovedsakelig hastighet, men C# hadde diverse mangler som gjorde at valget falt på Python.

Disse manglene inkluderer visualiseringsverktøy, hjelpeverktøy for data-rensning og ekstensive bibliotek for forskjellige typer filbehandling[27][6].

3.2.3 Python

Python er et objektorientert og tolket språk. Syntaksten er lettlest og gjør koding i språket raskere og lettere å forstå enn de fleste andre språk[28][7].

[3]"Softkraft - golang vs python." https://www.softkraft.co/ golang-vs-python/. Accessed: 2022-01-18[24]

[4]Go - dokumentasjon." https://go.dev/doc/. Accessed: 2022-01-15.[25]

[5]D. Shappir, "Forbes - why did microsoft create c#." https://www.forbes.com/sites/quora/2018/03/02/ why-did-microsoft-create-c/. Accessed: 2022-01-18.[26]

[6]"Microsoft - C# dokumentasjon." https://docs.microsoft.com/ en-us/dotnet/csharp/. Accessed: 2022-01-18[27]

[7]"Bestcolleges.com - 6 easiest programming languages to learn." https://www.bestcolleges.com/bootcamps/guides/6-easiest-programming-languages-

Python har også gode rammeverk og innebygde metoder som gjør håndtering av store mengder data lett å gjennomføre[29][8].

Et område der Python er underlegen sammenlignet med språk som Go og C# er hastighet, men grunnet at applikasjonen kun er et konseptbevis, var ikke dette en viktig faktor[30][9][24][10].

3.2.4 Valg av Python

Når språkene ble sammenlignet var det flere variabler som gjorde at Python skilte seg ut. Når det ble klart hva applikasjonen måtte gjøre ble det tydelig at prosessering av store datasett, visualisering av data, dekonstruering (parsing) av nettsider og oppsett av databaser ble det viktigste.
Områdene nevnt gjennomfører Python veldig bra grunnet det ekstensive biblioteket Python har tilgjengelig. For prosesseringen av store datasett kan noen av Python sine innebygde pakker og metoder forenkle mye av arbeidet, som Pandas og CSV-reader.

Da hovedmålet med applikasjonen er å bevise at ekstrapolering av aktiviteter er mulig ved hjelp av diverse datasett, vil ikke kjøretid være i fokus.

Applikasjonen skal heller ikke være et fullstendig produkt, men heller en basis hvor Kontali kan hente ut de aspektene de ser som relevante, og se hva slags datasett som er nødvendig for å anslå aktivitetene. Dette gjorde det viktig for oppgaven å kunne bruke et språk som var representert hos Kontali, slik at det lettere kunne integreres i deres plattform. Python ble et klarere valg etterhvert som grunnlaget for applikasjonens funksjonalitet ble mer tydelig.

to-learn/. Accessed: 2022-01-20.[28]

[8]"Python - dokumentasjon." https://docs.python.org/3/. Accessed: 2022-01-15 [29]

[9]"Programming-language-benchmarks - python vs c#." https://programming-language-benchmarks.vercel.app/ python-vs-csharp. Accessed: 2022-01-18.[30]

[10]"Softkraft - golang vs python." https://www.softkraft.co/ golang-vs-python/. Accessed: 2022-01-18[24]

3.3 IDE: VS Code og Jupyter Notebook

Valg av IDE (integrated development environment / utviklermiljø) ble basert på tidligere erfaring, så vel som hensikten med oppgaven. Gruppen har erfaring og kjennskap til Visual Studio Code, som også støtter mange utvidelser. Ettersom VS Code ikke har gode visualiseringsverktøy for GIS (geographical information system), ble også Jupyter Notebook brukt.

3.3.1 Visual Studio Code

VS Code er et av de mest brukte utviklingsmiljøene i verden[31][11] , og har god dokumentasjon samt brukervennlighethet. Microsoft ga ut Visual Studio i 1997 og oppfølgeren Visual Studio Code i 2015. Orginalt var Visual Studio ment for utvikling av Windows programmer på .NET plattformen [32][12], mens etterfølgeren var mer rettet mot allsidig bruk. VS Code støtter og legger til rette for de fleste plattformer, og har integrert GitHub-funksjonalitet. VS Code tilbyr også et stort marked av utvidelser utviklet av tredjeparter, for økt funksjonalitet og personlig tilrettelegging.

VS Code har vært det mest brukte miljøet gjennom undervisningen på UiS, og gruppen har dermed god kjennskap til miljøet. Miljøet er kompatibelt med de fleste teknologivalgene gjort gjennom oppgaven, og ble dermed et naturlig valg.

Strukturen av brukergrensesnittet i VS Code har kodescript som hovedfokus, terminal og mappestruktur på siden[33][13].

[11]"Insightsstackoverflow - integrated development environment." https: //insights.stackoverflow.com/survey/2021. Accessed: 2022-01- 17[31]

[12]Wikipedia - visual studio." https://no.wikipedia.org/wiki/ Microsoft_Visual_Studio. Accessed: 2022-03-20[32]

[13]Vs code nettside." https://code.visualstudio.com. Accessed: 2022- 03-20.[33]

3.3.2 Jupyter Notebook

Et område hvor VS Code ikke møtte kravene for bruk, var gjennom oppsett av et interaktivt kart. Etter flere forsøk på å sette opp interaktive kart i VS Code ble ingen signifikant fremgang gjort, som førte til leting andre steder.

Jupyter Notebook stod frem som en optimal løsning på visualiseringsproblemene. Jupyter Notebook er et web-basert Python-miljø[34][14]. Siden det er ønsket å kunne vise et kart i nettleseren, var det mer passende å bruke et miljø gjennom nettleseren enn å laste ned pakker eller applikasjoner for å oppnå det samme.

Jupyter Notebook har fokus på visualisering, og ga full funksjonalitet til kartet som ble laget. Det krevde også lite ekstra konfigurasjon for å visualisere applikasjonen[34]

3.4 Database: SQLite

Valget av databasesystem stod mellom to systemer: SQLite og MySQL. Begge systemene har blitt mye brukt gjennom utdanningen, både gjennom emnet Databaser DAT220, og gjennom bruk i diverse andre skoleprosjekter. En avgjørende forskjell mellom disse er at SQLite kan integreres i andre programvarer, som gjør det lettere å komme i gang med og bruke.

3.4.1 SQLite

SQLite er et open source[35][15] databasesystem som egner seg bedre for mindre prosjekter. Det er enkelt å sette opp og bruke som en naturlig integrasjon av programmet man utvikler, siden SQLite er en serverløs motor [36][16]. Det har noen begrensninger i form av mengden datatyper, kapasi-

[14]"Jupyter notebook nettside." https://jupyter.org. Accessed: 2022- 01-24[34]

[15]Sqlite - copyright nettside." https://www.sqlite.org/copyright. html. Accessed: 2022-03-2[35]

[16]Sqlite - serverless nettside." https://www.sqlite.org/serverless. html. Accessed: 2022-02-22.[36]

tet og hastighet. Disse begrensingene er mest aktuelle når man tar i bruk store datasett eller mange operasjoner hvor kjøretid er i fokus[37][17]. Som et eksempel er det ingen datatype for DateTime, men datatyper er ikke påkrevd for tabellene i SQLite så det kan jobbes rundt. For mindre prosjekter med relativt enkle datasett, hvor kjøretid ikke er i fokus vil ikke disse begrensningene være et hinder[38] [18].

3.4.2 MySQL

MySQL er på mange måter et mer profesjonelt system, som krever mer hensyn for initialisering og bruk. Systemet brukes av de største teknologiselskapene i verden for blant annet håndtering av brukerkontoer[39][19]. MySQL ble utviklet i 1994, og ble kjøpt av og er eid av selskapet Oracle. MySQL støtter flere datatyper og har generelt større kapasitet og hastighet[40][20]. For prosjekter på større skala, vil MySQL være et bedre valg på grunn av mer funksjonalitet, og initialiserings-prosessen vil være verdt bryet sammenlignet med et mindre prosjekt.

MySQL stiller høyere krav til konfigurering, og dermed kunnskap til systemet, men dette kommer med flere fordeler. MySQL har muligheten til å definere tilgangsnivåer slik at en gruppe brukere kan ha høyere tilgang på datasett, dette gjør MySQL sikrere og mer passende til webapplikasjoner. Ettersom dette kun er en enkel applikasjon som ikke skal offentliggjøres over nett, vil ikke dette være nødvendig for oppgaven eller dets mål[41][21].

3.4.3 Valg av SQLite

Etter rådgivning med veileder og vurdering av systemets styrker og ulemper falt valget på SQLite. Selv om gruppen har erfaring med begge systemene, er SQLite den som tidligere har blitt mest brukt gjennom studiets forløp.

[17]"Sqlite - quirks nettside." https://www.sqlite.org/quirks.html. Accessed: 2022-02-22.[37]

[18]Sqlite - nettside." https://www.sqlite.org/index.html. Accessed: 2022-02-21[38]

[19]"Wikipedia - mysql." https://no.wikipedia.org/wiki/MySQL. Accessed: 2022-03-22..[39]

[20]"Mysql - dokumentasjon." https://dev.mysql.com/doc/. Accessed: 2022-03-22[40]

[21]"Mysql nettside." https://www.mysql.com. Accessed: 2022-03-21[41]

Datasettet er lite nok til at SQLite kan håndtere det uten problemer. Applikasjonen har heller ingen behov for de avanserte sikkerhetskonfigureringene som MySQL kan tilby. Kontali, som ønsker å bygge videre på applikasjonen etter den ønskede funksjonaliteten er nådd vil integrere applikasjonen inn i sine egne systemer, og dermed sine egne databaser. Databasen må da endres ved overlevering til Kontali. Valget av databasesystem vil hovedsakelig være for utviklingen av applikasjonen uten å trenge å ta hensyn til hva Kontali bruker selv. Dette fører til at den ekstra funksjonaliteten MySQL kan tilby ikke vil forbedre applikasjonens funksjonalitet, som gjør SQLite til det bedre valget.[38] [22].

3.5 Karttjeneste: Folium

Visualiseringen av datasettene var et aspekt av oppgaven som ble gjort for å forbedre forståelse og utvikling.

Det ble dermed ønskelig å vise et interaktivt kart, slik at Kontali kunne se hvordan disse dataene hadde blitt satt i system. På denne måten kan de på et senere tidpunkt lage et lignende system. Dette ble også til stor hjelp for gruppen, ettersom geografiske lengde- og breddegrader kan være komplisert å feilsøke.

Et interaktivt kart, som inneholder de relevante datasettene, kan gi et overblikk og vise sammenhenger mellom de forskjellige fartøysposisjonene og fiskerilokasjonene. Dette hjelper med forståelse, både under utvikling av applikasjonen og visualisering av dataene.
Dataene brukt for å opprette kartet er geografiske posisjoner og tidspunkt. Sammen danner det et fullstendig bilde som gir en bedre oversikt og forståelse.

[22] Sqlite - nettside." https://www.sqlite.org/index.html. Accessed: 2022-02-21[38]

3.5.1 Google Maps

Google Maps er den mest brukte karttjenesten for personlig bruk[42][23], og er også mye brukt for selskap og applikasjoner som bruker kart[43][24].

Det er en god karttjeneste som har blitt bygd opp gjennom mange år, og har muligheter for satellittbilder, gatekart, topologi, og andre karttyper. Et fremtidig problem med Google Maps er at tjenesten koster penger å bruke på om man oppdaterer kartet oftere enn 28500 ganger iløpet av en måned. Dette ville nok ikke påvirket prosjektet i denne fasen, men om Kontali valgte å bruke dette, ville antallet båtoppdateringer overskredet 28500. Dette er fordi applikasjonen må oppdatere hver individuelle båts posisjon ofte nok til å fange hver båts eventuelle stopp og posisjonsendring, og med et stort antall båter kan dette antallet fort overskrides[44][25].

3.5.2 ArcGIS

ArcGIS er et geografisk informasjonssystem, som legger til rette for enkel integrering og bruk av geografisk informasjon på en visuell måte. ArcGIS tilbyr interaktivitet på høyt nivå og med relativt lite krevende implementasjon. Det er forskjellige stiler og verktøy for kartet til ArcGIS, som hjelper brukere med analyse og oversikt[45][26].
ArcGIS var i utgangspunktet førstevalget for oppgaven, ettersom det hadde god dokumentasjon, var mye brukt blant utviklere og virket dermed som en god løsning.
Gjennom forsøkene på å sette opp et ArcGIS miljø i VS Code ble det ved flere situasjoner oppdaget problemer angående VS Code sine visualiseringsverktøy for GIS. Dette ledet til bruken av Jupyter Notebook. Det ble senere kjent at ArcGIS Pro var nødvendig for å kjøre karttjenesten, som gjorde at

[23]"Statista - most popular us mapping apps ranked by reach." https://www.statista.com/statistics/865419/ most-popular-us-mapping-apps-ranked-by-reach//. Accessed: 2022-01-17.[42]

[24]Mapsplatform - karttjeneste valg hos selskap." https: //mapsplatform.google.com/. Accessed: 2022-01-17..[43]

[25]"Google maps - prisliste." https://mapsplatform.google.com/ pricing/. Accessed: 2022-01-17.[44]

[26]Esri - arcgis." https://www.esri.com/en-us/arcgis/products/ arcgis-online/overview. Accessed: 2022-03-21.[45]

det ble problemer med betalingsbarrieren, som dermed førte til at ArcGIS ble valgt bort[45].

3.5.3 Mapbox

Mapbox er en karttjeneste som leverer god visualisering, samt diverse pakker med mye støtte og god dokumentasjon. Mapbox er hovedsakelig et frontend verktøy hvor man lager kartet med JavaScript og fyller inn med CSS, dette gjorde Mapbox mindre optimalt for prosjektet ettersom applikasjonen ikke skal ha frontend [46][27].
Mapbox kan brukes gjennom Plotly, som er en Python pakke, men dette krever mer konfigurering enn Folium.
Gruppen så derfor nærmere på Folium, ettersom kartet var ment for å gi en innføring i hvordan kartet kan realiseres for Kontali, og hjelpe gruppen under utviklingen[47][28].

3.5.4 Folium

Folium er en plattform som er mer egnet for mindre profesjonelle utviklere og selskap.
Folium tilbyr et godt samarbeid med andre typer utvidelser, og det er lett å ta i bruk[48][29]. Syntaksen er relativt enkel og gjør koden lett å følge.
Folium er en gratis plattform, noe som er avgjørende for oppgavens del ettersom kartet ikke er et krav for å løse oppgaven.
Et problem med Folium er at kartet er statisk, som gjør at oppdateringer av posisjoner er mer krevende. For oppgavens del er ikke dette en stor faktor, men det kan få større betydninger for større prosjekter.

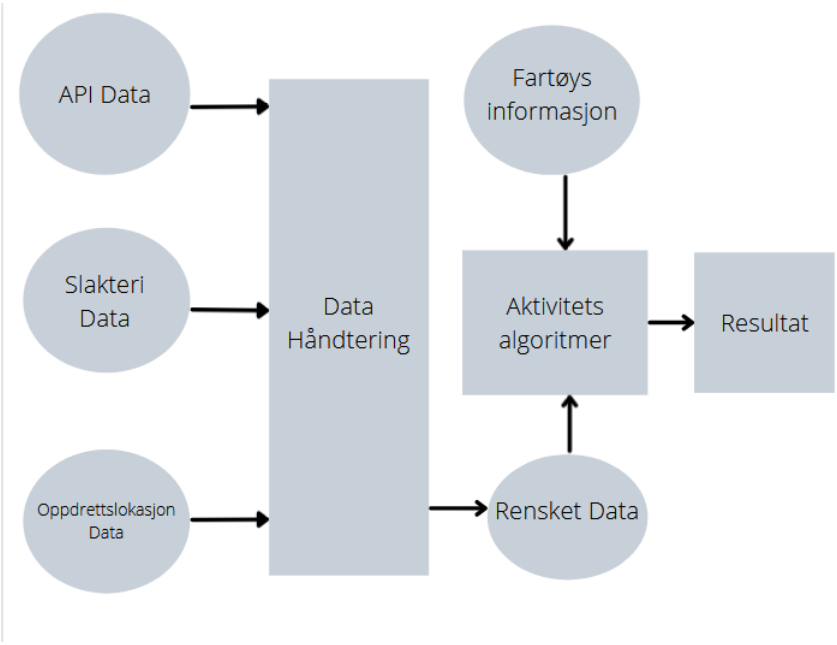
Valget falt på Folium ettersom det må være lett å lese for Kontali. Kontali kan da se over koden og se hva applikasjonen har tatt hensyn til, og dermed lage det selv uten å måtte søke for mye i dokumentasjonen[49][30].

[27]"Mapbox - dokumentasjon." https://docs.mapbox.com/. Accessed: 2022-01-18[46]
[28]"Plotly - dokumentasjon." https://plotly.com/python/. Accessed: 2022-01-18.[47]
[29]"Data2int - folium." https://data2int.com/Foliuml. Accessed: 2022- 01-16[48]
[30]"Folium dokumentasjon." https://autogis-site.readthedocs.io/en/latest/notebooks/L5/02_interactive-map-folium.html.Accessed: 2022-01-16. [49]

Kapittel 4

Konstruksjon

I dette kapittelet beskrives oppgavens informasjonssamling, datarensing og konstruksjon av applikasjonen. I figur 4.1 vises applikasjonens endelige struktur i et forenklet format.

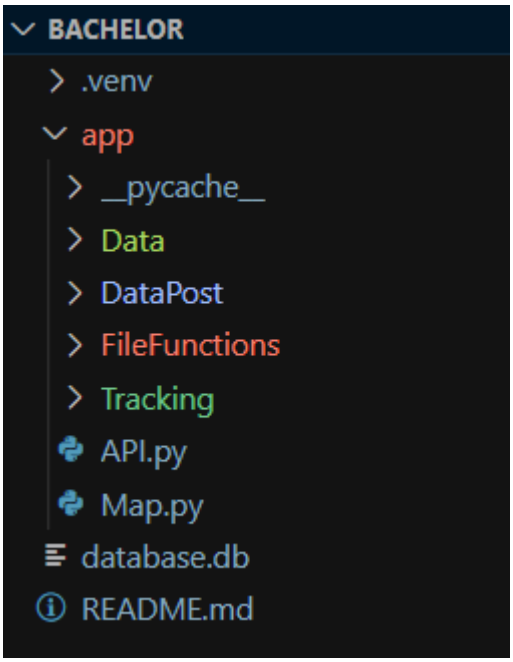


Figur 4.1: Endelig oppbygning av applikasjonens informasjonsflyt

4.1 Mappestruktur

Mappestrukturen er delt opp i fire forskjellige deler. Disse kan oppsummeres som følger:

1. Data: Data hentet direkte fra offentlige direktiv eller API.
2. DataPost: Data etter rensing.
3. FileFunctions: Kode for uthenting og rensing av data.
4. Tracking: Kode for database med tilhørende metoder, og algoritmer for behandling av data.



Figur 4.2: Mappestruktur brukt for å realisere prosjektet.

I mappestrukturen vist i figur 4.2 vil Tracking inneholde all kode brukt for etablering av database, geografisk funksjonalitet og algoritmer.

Denne inneholder hoveddelen av oppgaven, og har diverse metoder, funksjoner og utregninger for applikasjonens funksjonalitet. Her var filene for oppsett av database, innsetting av data, behandling av data, og konklusjoner om stopp og aktivitet.

Mappen 'Data' inneholder de første dataene hentet for oppgaven, disse er ikke renset og er som de var da de ble hentet.
DataPost mappen inneholder alle dataene etter rensing og har dermed dataene brukt i databasen og filene i Tracking.
FileFunctions inneholder alle metoder brukt for å rense datasettene, samt metoder for uthenting fra diverse API. Her var filene med metoder for lesing, datarensing og konstruksjon av tekstfiler.

4.2 Datavalg

Ettersom oppgaven gikk ut på dataanalyse var gode datasett veldig viktig for å kunne lage velfungerende algoritmer og nøyaktige prediksjoner.
Etter anbefalinger fra Kontali og mattilsynet ble grunnlaget for datasettene funnet hos fiskeridirektoratet[18][1] og Barentswatch[20][2].

Barentswatch har brukt diverse offentlige datasett for å bygge sin applikasjon, som hadde mye til felles med gruppens endelige applikasjon. Fiskeridirektoratet hadde åpne datasett som la grunnlaget for lokasjonene som ble brukt i applikasjonen, det viktigste av disse datasettene var akvakulturregisteret[50][3].

Datasettene funnet hos Barentswatch og fiskeridirektoratet var ikke fullstendige, og krevde bruk av deres respektive API for å hente ut all informasjon. Bruken av datasettene og deres API vil bli beskrevet videre i kapittel 4.3

Data for fartøy finnes gjennom AIS, som brukes i de fleste land, og det er

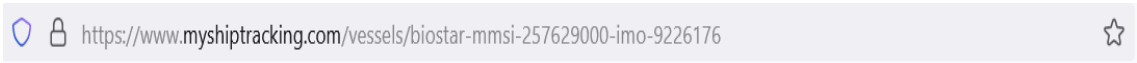
[1]"Fiskeridirektoratet - api." https://api.fiskeridir.no/pub-aqua/ api/swagger-ui/index.html?configUrl=/pub-aqua/api/ api-docs/swagger-config#/site-resource/getBordersBySiteNr. Accessed: 2022-01-21. [18]

[2]Barentswatch - aapne data og api." https://www.barentswatch.no/ en/articles/open-data-via-barentswatch/. Accessed: 2022-02-03[20]

[3]"Akvakulturregisteret - registre og skjema." https:// www.fiskeridir.no/Akvakultur/Registre-og-skjema/ Akvakulturregisteret. Accessed: 2022-01-15.[50]

flere nettsider som offentliggjør denne informasjonen[1] [4].
For å oppnå Kontali sine mål, hvor fartøyene ikke nødvendigvis var norske, måtte informasjonen hentes fra en side som inneholdt informasjon om skip både innenlands og utenlands. Dette ble oppnådd gjennom Myshiptracking.com ettersom siden har et lett håndterbart lenkeformat, samt at den inneholder informasjon om skip fra hele verden[51][5].

For å finne live informasjon om fartøy av interesse, kan deler av URL-lenken byttes ut med identifikasjonsnummer, i form av MMSI-[11][6] og IMO nummer[7][7].
Et eksempel på dette vises i figur 4.3, hvor båten Biostar blir søkt opp.



Figur 4.3: Lenke for BIOSTAR, vist med MMSI og IMO nummer.

Som vist i figur 4.3 vil MMSI nummeret være 25769000 mens IMO nummeret vil være 9226176, disse nummerene er offentlig informasjon, og vil dermed gjøre nettsiden lett tilgjengelig for alle båter registrert.
Dette kan automatiseres ved hjelp av en innsetting som vist i kodeblokk 4.1. Her vil navnet på fartøyet i lenken oppdateres ved nye MMSI og IMO nummer.

Kode 4.1: Behandlet lenke hvor MMSI og IMO nummer kan føres inn

```
1 url = f'
2     https://www.myshiptracking.com/vessels
3     /biostar-mmsi-{MMSI}-imo-{imo}
4 '
```

Lenken i kodeblokk 4.1 gjør det mulig å parse siden, for deretter å hente ut den nødvendige informasjonen for oppgaven.

[4]"Kystverket - ais beskrivelse." https://www.kystverket.no/en/ navigation-and-monitoring/ais/. Accessed: 2022-01-15[1]
[5]"Myshiptracking - nettside." https://www.myshiptracking.com/. Accessed: 2022-01-15.[51]
[6]Navcen - mmsi." https://www.navcen.uscg.gov/?pageName= mtmmsi. Accessed: 2022-01-15.[11]
[7]"Imo - hjemmeside." https://www.imo.org/. Accessed: 2022-01-15.[7]

4.2 Datavalg

Siden har et enkelt oppsett hvor skipenes informasjon i stor grad er holdt i tabeller. Dette gjorde det mulig å bruke Pythons parsing verktøy, BeautifulSoup[52][8], og søke gjennom tabellene for den nødvendige informasjonen. Denne uthentingen av data er vist i figur 4.2.
I denne kodeblokken vil nettsiden bli hentet med nødvendige kriterier, for deretter å bli parset slik at informasjonen kan behandles. Dette har blitt utført i andre prosjekter tidligere, og ga gruppen godt innsyn i hvordan det kunne gjøres for denne nettsiden[53][9]. Dataene vil deretter bli lagt til i en liste, før den blir lagt til i databasen etter alle skip er registrert. Selv om kun lengdegrad(Longitude) hentes ut i kodeblokken, vil resterende data hentes ut på tilsvarende vis.

Kode 4.2: Utdrag av kode for henting av AIS-data[53]

```
request = urllib.request.Request(url, None, headers)
        with urllib.request.urlopen(request) as site:
            page = site.read()
        html = BeautifulSoup(page, "html.parser")
        tables = html.findAll("table")
        for table in tables:
            if table.findParent("table") is None:
                for row in table.findAll('tr'):
                    elements = row.findAll('td')
                    try:
                        if elements[0].string == "Longitude":
                            Longitude = elements[1].string
                            #resterende data vil hentes ...
                                paa samme vis
                    except Error as e:
                        print(e)
        ships.append([float(latitude),float(longitude),
        name,speed,Course,Timestamp])

        return ships
```

I kodeutdraget 4.2 parses siden, for så å hente ut den nødvendige informasjonen. De viktigste elementene her er:

- Latitude: Breddegrad koordinat.

[8]"Crummy - beautifulsoup dokumentasjon." https://www.crummy.com/ software/-BeautifulSoup/bs4/doc/. Accessed: 2022-01-16.[52]

[9]Riccardo, "Digital-geography.com - vessel tracking the python way." https://digital-geography.com/ vessel-tracking-the-python-way/. Accessed: 2022-02-03.[53]

- Longitude: Lengdegrad koordinat.
- Name: Navn på det enkelte fartøyet.
- Speed: Fartøyets hastighet ved måling.
- Course: Retning fartøyet peker ved gjeldende måling.
- Timestamp: Tidspunkt med dato og klokkeslett for målingen.

Dette vil gi hvert skip en ny oppføring i skipsloggen som kan sees i figur 4.4. Hvis skipet står stille vil farten vises som 'None' grunnet manglende oppføring på nettsiden[51][10].

```
1   BIOSTAR,61.58474,4.99999,0.1 Knots,134°,2022-02-01 16:59
2   RONJA OCEAN,64.86081,11.24012,None,218°,2022-02-01 16:59
```

Figur 4.4: Utdrag fra skipslogg med oppføring av relevant informasjon.

For utviklingen av oppgavens applikasjon, ble historisk fartøysdata tildelt. Dette blir utdypet i neste delkapittel.

4.3 Datahåndtering

De nødvendige datasettene for gjennomføring av oppgaven var tilgjengelige gjennom forskjellige nettsider og API'er.
Det var lite sammenheng eller likhet mellom måten datasettene var bygd på, som gjorde at felles rensemetoder ikke kunne brukes.
De forskjellige datasettene brukte hovedsakelig tre formater, disse var:

- CSV
- Excel
- JSON

[10]Myshiptracking - nettside." https://www.myshiptracking.com/. Accessed: 2022-01-15.[51]

Rensing av CSV-filer kunne gjøres ved hjelp av Pythons innebygde metoder. Excel og JSON filer måtte derimot behandles ved hjelp av verktøy fra andre bibliotek.
Gjennom Barentswatch[20][11] og fiskeridirektoratet[50][12]var datasettene hovedsakelig tilgjengelig i CSV format, men fiskeridirektoratets tilhørende data for eksterne punkter var kun tilgjengelig gjennom deres API[18][13].

Her måtte det konstrueres en metode for behandling av dataene. Metoden gikk gjennom datasettene med oppdrettslokasjoner og slakterier og hentet identifikasjonsnummerene, for så å sette disse inn i API'et for videre datahenting. Dette gjorde det mulig å hente ut den relaterte lokasjonsinformasjonen, og legge den til i egne filer.

I de originale datasettene fra fiskeridirektoratet inneholdt datasettet for oppdrettslokasjoner to identifikasjonsnummer. Disse nummerene står oppført som TILL_NR og TILL_KOMNR, og er individuelle nummer for hvert oppdrettsanlegg. TILL_KOMNR var ikke et alternativ for bruk gjennom API'et, som gjorde TILL_NR til den optimale parameteren.

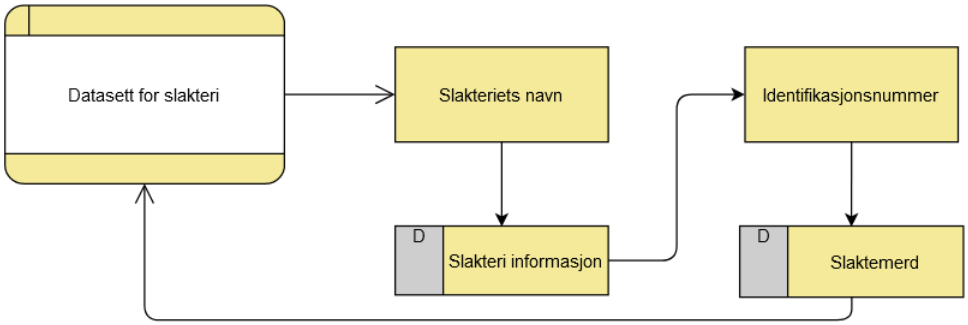
Slakteriene hadde derimot ikke et relevant identifikasjonsnummer innad i sine originale datasett. Dette krevde at slakteriets navn måtte brukes som identifikasjon for å kunne hente ut lokasjonens identifikasjonsnummer i API'et[21][14]. Dette identifikasjonsnummeret kunne da brukes videre i API'et for å hente ut slakteriets tilhørende slaktemerd.
I figur 4.5 vises informasjonsflyten ved henting av en slaktemerd.

[11] Barentswatch - Åpne data." https://www.barentswatch.no/ artikler/apnedata/. Accessed: 2022-01-15.[20]

[12] "Akvakulturregisteret - registre og skjema." https:// www.fiskeridir.no/Akvakultur/Registre-og-skjema/ Akvakulturregisteret. Accessed: 2022-01-15.[50]

[13] "Fiskeridirektoratet - api." https://api.fiskeridir.no/pub-aqua/ api/swagger-ui/index.html?configUrl=/pub-aqua/api/ api-docs/swagger-config#/site-resource/getBordersBySiteNr. Accessed: 2022-01-21[18]

[14] Barentswatch - aapne data og api." https://www.barentswatch.no/ en/articles/open-data-via-barentswatch/. Accessed: 2022-02-03.[21]



Figur 4.5: Informasjonsflyt for henting av et slakteri sin tilhørende slaktemerd.

De nye punktene ble presentert i et JSON-format, som gjorde de tidligere uthentingsmetodene unyttige, og nye metoder var dermed nødvendig for å håndtere de individuelle filene.

Etter datasettene var satt sammen, måtte de renses ytterligere for å kunne brukes i applikasjonen. Akvakulturregisteret inneholdt diverse duplikater, fordi hver lokasjon fikk en ny oppføring for hver fiskeart lokasjonen holdt. Lokasjonene som ble hentet ut måtte derfor renses for duplikater, mens fiskeartene måtte legges til på sine respektive lokasjoner.
Etter dataene ble samlet kunne oppdrettslokasjonene, smoltanleggene, slakteriene og slaktemerdene legges inn i sine endelige filer, for så å legges inn i databasen.

Informasjon om båtene som skulle brukes til å utbedre algoritmene, ble tilsendt fra Kontali og kom i form av Excel regneark.
De inneholdt den relevante posisjonsloggen, men også irrelevant informasjon som ikke var nødvendig for oppgavens formål. Den irrelevante informasjonen var eksempelvis logoer og tilleggsinformasjon om regnearket, og måtte fjernes før selve dataene kunne hentes. Regnearkene ble først behandlet gjennom Pandas, slik at de kunne konverteres til CSV-filer, for lettere håndteringen og nyttegjøring av allerede etablerte rense- og lesemetoder.

4.4 Kart

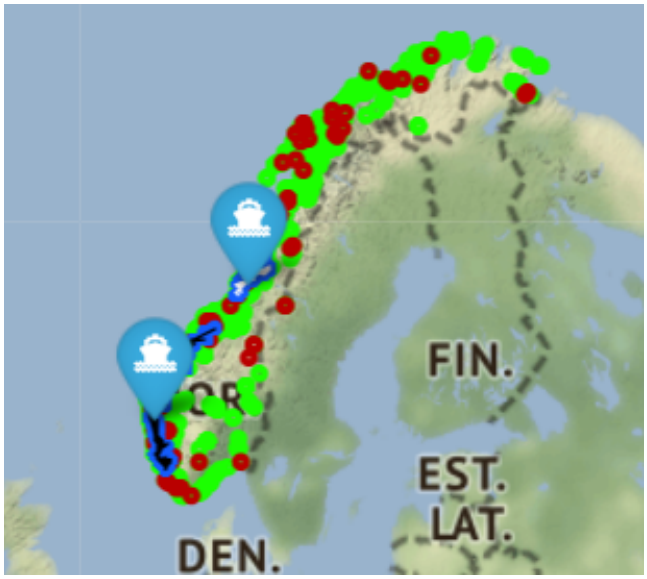
Kartet ble satt opp med basis i lengde- og breddegrad koordinatsystemet, slik at den innsamlede informasjonen kunne settes direkte inn i kartet uten reformatering.
Folium bruker statiske kart som gjorde at de eneste elementene som måtte oppdateres etter initialisering var skipenes posisjon og tilhørende informasjon.

Gjennom datahåndteringen ble all individuell informasjon separert og satt opp i sine egne spesialiserte filer. Metoden for kartoppdatering kunne dermed splittes opp til flere forskjellige metoder slik at oppdatering av kartet kunne bli så effektivt som mulig.
Med egne filer for slakteri, oppdrettslokasjoner og smolt-anlegg, og tidligere lokasjoner for RonjaOcean og Biostar, ble kartets oppsett konfigurert utifra uthenting av data fra de korresponderende filene.

Kartet kunne dermed konfigureres med grunnleggende pakker og innstillinger, for å deretter legge inn de forskjellige datapunktene individuelt. Ved å legge inn lokasjonene individuelt ble prosessen av å legge inn relatert informasjon enkelt å gjennomføre, ettersom hvert datasett hadde et tilnærmet likt oppsett. Det gjorde at hvert ikon på kartet kunne konfigureres med grunnleggende html kode.
I kodeblokk 4.3 vises hvordan informasjons-bokser generes for fartøy, men konfigurasjonen for andre informasjonsbokser vil i stor grad samsvare.

Kode 4.3: Kodeutdrag hvor informasjon blir lagt til for et fartøys informasjonsboks.

```
1     def generate_popup_boat(name, speed, course, date):
2         return f'''
3             <strong>Name:</strong> {name}<br>
4             <strong>Speed:</strong> {speed}<br>
5             <strong>Course:</strong> {course}<br>
6             <strong>Position Recuved:</strong> {date}<br> '''
```

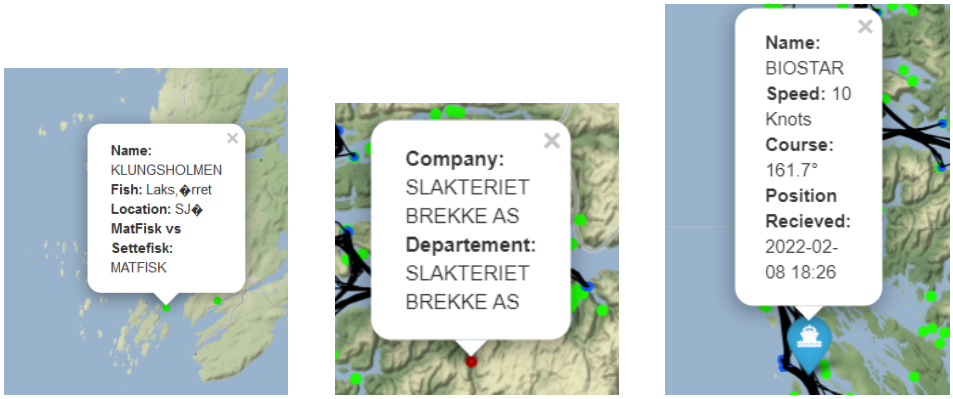


Figur 4.6: Kart over Norge med oppdrettslokasjoner, slakterier, fartøy og fartøyenes tidligere lokasjoner.

I figur 4.6 er hvert påførte element fargekodet.

- Grønn: Grønne markører er for oppdrettslokasjoner og smoltanlegg på land og på havet.
- Rød: Røde markører brukes for slakteri og slaktemerder.
- Fartøys ikon: Brukes for den siste lokasjonen til et fartøy.
- Blå: Blå markører brukes for tidligere lokasjoner fartøyene har blitt registrert, og legger dermed grunnlaget for posisjons linjene.
- Svart linje: Brukes for Biostar sine tidligere dokumenterte lokasjoner.
- Hvit linje: Brukes for Ronja Ocean sine tidligere dokumenterte lokasjoner.

Hvert ikon eller lokasjonselement inneholder informasjon om hva slags lokasjon dette er, samt hva som produseres og annen relevant data. Dette vises i figur 4.7a, 4.7b og 4.7c.



(a) Informasjon ved klikk på oppdrettslokasjon-ikon.

(b) Informasjon ved klikk på slakteri-ikon.

(c) Informasjon ved klikk på fartøy-ikon.

Figur 4.7: Informasjon tilgjengelig ved trykk på kartets ikoner.

Grunnet Folium sine statiske kart er oppdateringer relativt krevende. Fordelen med et statisk kart i denne sammenhengen er at det er lite ny informasjon å oppdatere, ettersom lokasjonene er faste og kun fartøy beveger seg.
Kartet blir derfor initialisert med alle tilgjengelige lokasjoner og tilhørende informasjon, og oppdaterer fartøyenes posisjon ved nye data hentinger. Den forminskede mengden data som oppdateres reduserer dermed behandlingstiden betraktelig.

4.5 Feilmarginer og feilmålinger

Gjennom oppgaven har det så langt blitt nevnt diverse datasett, som ved alle datasett inneholder disse datasettene feilmålinger. Dette er også datasett hentet ut fra målinger gjort på virkelige objekter, og det er stor sannsynlighet for at målinger ikke er helt eksakte. En del av oppgaven ble dermed å lage forsikringer mot eventuelle feilmålinger, som kunne gi et riktig resultat selv om datasettene ikke var feilfrie.
Feilmarginer ble derfor en viktig del av oppgaven, for å håndtere disse ble det utnyttet flere algoritmer. Når et fartøy stopper ved en slaktemerd eller en oppdrettslokasjon vil den ved flere anledninger ikke registreres som innenfor polygonet, men heller rett utenfor.

Dette kan for eksempel inntreffe om fartøyet er et frakteskip som skal levere mat til lokasjonen, og matbrønnene er ved utkanten av området som defineres som oppdrettslokasjonen. Ved hjelp av naturlige målinger og direkte utnyttelse av datasettene ville da fartøyet stoppe utenfor lokasjonen, for så å dra videre. For å unngå dette ble det laget en sikkerhetsmargin for fartøyene.
Ved å bruke Python-biblioteket Shapely[54][15] kunne det lages en logisk sirkel rundt båtens posisjon med en spesifisert radius. Denne radiusen måtte tilpasses de tilgjengelige dataene, og ble dermed endret underveis for å oppnå ønsket resultat, uten falske positive treff.

Å kun bruke en vanlig avstandsmetode var derimot ikke mulig, dette er fordi lengde- og breddegrads systemet vil få større avstander for hver lengde- og breddegrad unna ekvator målingene skjer. På grunn av dette brukes en mer komplisert formel enn euklidsk distanse[5][16] for å regne avstanden mellom to punkter, og formelen som stod frem som mest passende, var da haversine-formelen[55][17].
Denne formelen er laget for å kunne finne avstanden mellom 2 punkter på jordens overflate ved hjelp av lengde- og breddegrad koordinatene deres. Denne formelen vil ikke gi en helt nøyaktig måling, men en passende måling ettersom det er tatt høyde for feilmarginer og feilmålinger.

Ved hjelp av denne logiske sirkelen rundt båtens posisjon ville lokasjoner innenfor radiusen regnes som besøkt, selv om båten ikke var nøyaktig innenfor området til lokasjonen. Ettersom distansene mellom lokasjoner er relativt store, ble det mulig å basere sikkerhetsmarginen på en prosentandel av den totale avstanden mellom de to lokasjonene med kortest avstand.
Sikkerhetsmarginen kunne da være relativt stor uten å risikere at fartøyet var innenfor to forskjellige lokasjoner.
Dette sikret at sikkerhetsmarginen var stor nok til å kunne registrere et stopp ved en lokasjon selv om fartøyet ikke var innenfor lokasjonens område, uten å være så stor at andre lokasjoner også ble registrert.

[15]"Pypi - shapely hovedside." https://pypi.org/project/Shapely/. Accessed: 2022-03-06[54]

[16]"Wikipedia - avstand." https://no.wikipedia.org/wiki/Avstand.Accessed: 2022-03-15.[5]

[17]Geeksforgeeks - haversine formelen." https://www.geeksforgeeks. org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/. Accessed: 2022-03-23.[55]

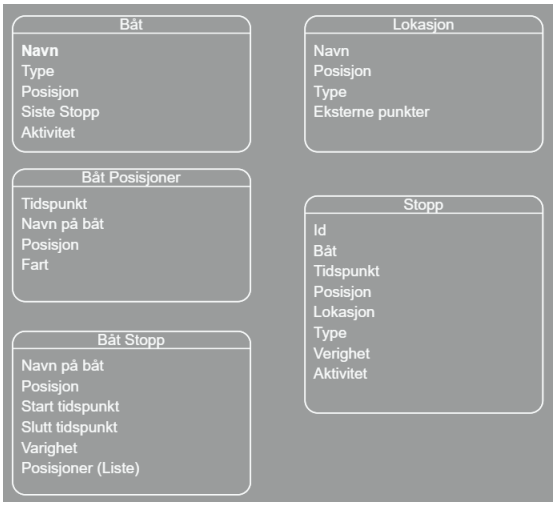
4.6 Database

Databasen som ble brukt for å lagre og hente informasjonen til algoritmene besto av flere tabeller, og hadde en samling metoder for tilgang til dataene. Databasen med alle tabeller ble bygget først, og så ble data skrevet inn når datasettene var renset. I figur 4.1 vil databasen stå i "Aktivitets algoritmer", hvor fartøys informasjon og renset data samles.

4.6.1 Tabeller

Innholdet i tabellene var renset data fra de samlede datasettene, og ble lagt til i sine respektive tabeller.
Båt- og lokasjonsinformasjonen var lagret i forskjellige tabeller, og inneholdt både informasjon hentet fra de forskjellige kildene og variabler som ble oppdatert underveis. For eksempel hadde tabellen for båtene en kolonne for siste stopp, som ble oppdatert etter stoppet ble kartlagt.

Det var også tabeller for posisjoner der båter hadde lav fart, som ble brukt for å finne og matche lokasjonstopp, som også ble lagret i en egen tabell. Disse stoppene, med båtstopp og lokasjoner, ble videre brukt for å bestemme sammenhengene mellom stoppene. Tabellene som ble brukt vises i figur 4.8:



Figur 4.8: Oppbyggingen av database tabellene

Figuren 4.8 viser tabellene som ble brukt for å lagre den relevante informasjonen om båtene, lokasjonene, posisjonene og stoppene.

Gjennom konstruksjonsprosessen av databasen ble det gjort mange endringer underveis, som forbedret logikken og gjorde det lettere å jobbe med.
Siden datamengden for både båtinformasjonen og lokasjonene var relativt store, ble bare en mindre del av denne datamengden brukt under konstruksjon og testingen av databasen.
Dette ble gjort for å spare tid for både innskriving av data og behandling av data. Den fulle databasen inneholdt over 50 000 datapunkter for båtposisjoner, og nært 2000 lokasjoner.

4.6.2 Metoder

Metodene som tilhørte databasen blir brukt til å hente, endre og skrive data til tabellene. Det er disse metodene som brukes for innføring av den rensede dataen fra datasamlingen. Lagringsmetodene la til båtposisjoner fra båtdatasettene, lokasjoner fra lokasjonesdatasettene, og de kartlagte stoppene for båtene.
Metoden for å legge til båtposisjoner ble også brukt for å legge til nye

datapunkter hentet fra Myshiptracking.com. Her ville data hentes ut fra Myshiptracking.com i 12 timer, før datasettet ble lagt til i databasen, for å deretter gjøre antakelser på hva fartøyet hadde foretatt seg i perioden.

Flere av metodene mottar argumenter og krever da spesiell syntaks for å bruke disse argumentene i SQL-spørringen.
Koden i blokk 4.4 viser en vanlig brukt syntaks, som i dette prosjektet ble brukt mest.

Kode 4.4: SQL-spørringen med argumenter

```
1      sql = ''' UPDATE boats
2                SET position_lat = ? , position_lon = ?
3                WHERE bname = ?'''
4      cur = conn.cursor()
5      cur.execute(sql, (lat, lon, boatname,))
```

Kodeblokk 4.4 viser den mest brukte syntaksen for argumentinnsetting av SQL-spørring, i dette prosjektet. Argumentene settes inn i rekkefølge for spørsmålstegnene i SQL-strengen.

En annen variant av syntaksen vises i kodeblokk 4.5, her blir samme argument brukt flere plasser i SQL-spørringen.

Kode 4.5: SQL-spørringen med argumenter som dictionary

```
1          sql = ''' SELECT *
2          FROM locations
3          WHERE
4              ABS(:lat - position_lat) < :distance
5          AND
6              ABS(:lon - position_lon) < :distance'''
7          cur = conn.cursor()
8          cur.execute(sql, {
9              'lat':boatpos_lat ,
10             'lon':boatpos_lon ,
11             'distance':distance}
12         )
```

Her vises den andre varianten av argumenthåndteringen som er brukt, hvor samme argument brukes flere plasser i SQL-strengen. For denne syntaksen dannes det et dictionary, bestående av nøkkelord og verdier, hvor argumen-

tene er verdiene. Nøkkelordene brukes i SQL-sprørringen og holder plassen til de variable argumentene som skal brukes.

Et området som krevde mer hensyn under innføring av datasettene, var de eksterne punktene for lokasjonene. For lokasjoner med eksterne punkter, ble disse punktene satt sammen til en tekst streng som ble lagret i databasen.

Etter databasen og relevante metoder for henting og behandling av data var komplett, var det åpent for å legge til fulle datamengder.

4.7 Algoritmer

Algoritmene for avgjørelser av stopp og sammenhenger tok nytte av dataene i databasen og variable parameter for justering. Forskjellige aspekter av logikken ble kjørt fra forskjellige filer, og behandlet data ble lagret i databasen. Applikasjonen kan brytes ned til disse trinnene:

- Behandle og lagre data for fartøyene
- Behandle og lagre data for lokasjonene
- Filtrer stopp for fartøy
- Sammenligne stoppene for fartøy med lokasjonene
- Undersøke sammenhengen mellom stoppene fartøy hadde ved lokasjoner

4.7.1 Stopp

Stoppene for båtene baserte seg på datapunktene hvor farten til båten var under en viss grense, og posisjoner i samme tidsrom ble gruppert sammen som et stopp for båten og lagret i databasen. For disse båtstoppene ble nærliggende lokasjoner hentet fra databasen og brukt til å konstruere polygoner. Selve posisjon for båten ble brukt som sentrum i en sirkel som representerte sikkerhetsmargin mot usikkerhet i målingene. Eventuelle overlapp mellom

polygonene for lokasjoner og sirkelen for båtstoppet ble lagret i en egen liste som ble sortert på avstand mellom båten og lokasjon. Med unntak av smoltanlegg, ble den lokasjon med lavest avstand valgt som sannsynlig stopp og lagret i databasen, smoltanlegg ble valgt om avstanden var mindre enn en satt margin. Unntaket for smoltanleggene ble implementert for å ta hensyn til at smoltanleggene ligger på land, og at båter ikke kommer nært de tilhørende koordinatene.

Et problem som måtte håndteres under utviklingen av programmet var polygoner som overlappet seg selv, dette ga feilmelding 'self-intersect'. De polygonene hvor dette var tilfellet regnes som ikke gyldig, eller 'not valid'. Shapely biblioteket har en metode for å gjøre polygoner gyldige, nemlig $make\_valid(polygon)$ [56][18]. Denne metoden omgjør polygonet til et multipolygon, en klasse polygon som inneholder flere polygoner, på samme måte som et polygon inneholder flere punkter. Denne metoden ble brukt for ugyldige polygoner, og vises i kodeblokk 4.6

Kode 4.6: Koden for gyldiggjøring av polygoner

```
1     # Lokasjonene som ligger nÃ¦rt stoppet blir lagret som ...
          polygoner i en list
2     for polygon in polygons:
3         try:
4             # Noen av polygonene har problemer med ...
                  self-intersect, at det overlapper seg selv
5             if polygon.intersection(boat):
6                 # Om polygonet til lokasjonen overlapper ...
                      med skipet sin posisjon returneres True
7                 return True
8             else:
9                 # Om polygonet ikke har overlapp, ...
                      fortsetter programmet til neste
10                continue
11        # Om polygonet har self-intersect kommer denne ...
              feiltypen
12        except shapely.errors.TopologicalError:
13            # Skaper gyldig polygonet, forvandler ...
                  polygonet til multipolygon
14            multiPolygon = make_valid(polygon)
15                # Sjekker da om multipolygonet overlapper ...
                      med skipet
16                if multiPolygon.intersection(boat):
```

[18]Shapely - dokumentasjon nettside." https://shapely.readthedocs. io/en/stable/manual.html. Accessed: 2022-03-22.[56]

```
17                          return True
```

Koden i blokk 4.6 viser hvordan problemet med ugyldige polygoner ble løst. Ved hjelp av en 'try / except' blokk ble eventuelle ugyldige polygoner omgjort til multipolygoner. Multipolygonene ble så sjekket mot båtens posisjon på samme måte som de gyldige polygonene.

4.7.2 Sammenheng

Ved å se på kartet vil man oppdage at rutene båtene tar kan virke tilfeldig, ettersom leveranser av mat og smolt ikke følger en bestemt rekkefølge, men er heller beslutninger som bestemmes etter behov. Den store utfordringen kom dermed med hvordan man skulle kartlegge aktivitet utifra posisjon. Den første faktoren måtte derfor baseres på hva de forskjellige båtene ble bygget for.

Datasettene utlevert fra Kontali inneholdt 2 fartøy som hadde hver sin hensikt. Ronja Ocean var en brønnbåt, og hadde et større potensielt oppgavesett enn Biostar som kun var et frakteskip for mat. Basert på informasjonen fra Kontali var dette typiske skip for sine respektive typer, og de båtene som var av størst interesse for oppgaven. Biostar sine posisjoner og eventuelle stopp kunne utelukkende sees på som enten henting av mat, eller leveranser av mat. Ronja Ocean derimot hadde flere typer oppgaver som kunne settes opp mot stoppene de hadde tatt. Dette førte til at å lage algoritmer for fartøyets stopp ble mer komplisert grunnet alle muligheter som kunne være resultatet av et stopp. For Biostar ble resulterende algoritme basert på hvor fartøyet hadde stoppet, og stoppets varighet. Dette gjorde at eventuelle stopp ved havn hvor mannskapet muligens skulle ha permisjon ikke ble satt opp som henting av mat, med mindre fartøyet dro direkte fra havnen til en oppdrettslokasjon. Deretter kan algoritmen gå som følger:

Matleveranse

- Om båten er et frakteskip for mat.
- Forrige stopp er ved en ukjent lokasjon eller ved en oppdrettslokasjon.
- Nåværende stopp er ved en oppdrettslokasjon eller et smoltanlegg.

- Stoppet er leveranse av mat.

For Ronja Ocean er konstruksjon av algoritmer mer komplisert, Ronja Ocean kan med datasettene brukt definere to forskjellige aktiviteter. Disse er da, når det settes ut smolt og når fisk tas ut og slaktes. Her måtte det dermed lages to forskjellige distinkte algoritmer, begge må defineres ut fra hva den mest sannsynlige aktiviteten er.
Informasjonen tilgjengelig gjorde de mest relevante algoritmene til:

Smolt-utsettelse

- Om båten er en brønnbåt
- Forrige stopp er ved smolt-anlegg
- Nåværende stopp er ved oppdrettslokasjon
- Stoppet er for utsettelse av smolt

Leveranse av slakteklar fisk

- Om båten er en brønnbåt
- Forrige stopp er ved oppdrettslokasjon
- Nåværende stopp er ved slakteri eller slaktemerd
- Stoppet er for leveranse av slakteklar fisk.

Ved hjelp av systemene laget tidligere ble implementasjonen av disse algoritmene relativt enkle. Resultatene algoritmene ga ble deretter satt opp mot handlingene fartøyene hadde gjort på Barentswatch[19][19].

Resultatene vil videre bli diskutert i kapittel 5.

[19]Barentswatch - hjemmeside." https://www.barentswatch.no/. Accessed: 2022-01-15.[19]

Kapittel 5

Resultater og diskusjon

I dette kapittelet vil valgene og vurderingene som ble tatt underveis i prosjektet – og hvordan disse valgene påvirker resultatene – diskuteres.
Resultatet for oppgaven vil deretter bli vist og forklart.
Etter resultatene vil det reflekteres over valgene som ble tatt, og hvilke valg som kunne hatt nytte av en revurdering.

5.1 Diskusjon

5.1.1 Vurdering av datasett

Dataene var den viktigste delen av prosjektet, og var dermed det området som hadde størst påvirkning på de endelige resultatene.

Gjennom uthentingen av data ble det klart hva slags ekstrapolering av aktiviteter som ville være mulig, og hva som ville være problematisk.
Gjennom datasettene nevnt tidligere ville leveranser av mat, utsetting av smolt og leveranser av slakteklar fisk være realistisk å gjøre antakelser på grunnet datasettene sine klare og bakoverkompatible datapunkter. Et område hvor algoritme-konstruksjon ble mer problematisk, var innen bestemmelse av aktivitet som utføres innen lus- og sykdomsbehandling, og generelt

vedlikehold på oppdrettslokasjoner.

Mangelfulle datasett

I begynnelsen av prosjektet var planen å basere oppdrettslokasjonene på akvakulturregisteret[50][1]. I løpet av en rask gjennomgang av datasettet innså vi at diverse lokasjoner i datasettet refererte til samme område, og måtte dermed renses på mer enn kun de tilhørende datapunktene for hver lokasjon.
Her kunne det da være så mye som syv posisjoner med samme identifikasjonsnummer som inneholdt de samme geografiske punktene.

Dette var i motsetning til vår første antakelse, om at de samme identifikasjonsnummerene inneholdt de forskjellige eksterne punktene for polygonene. De individuelle datasettene måtte dermed renses for duplikater og unyttige data.
Dette ledet gruppen videre til å utforske Barentswatch og fiskeridirektoratene sine API'er[18][2][21][3] som gjorde det mulig å definere de eksterne punktene.
Hvert API krevde også registrering av bruker og søking om tillatelse til bruk, som gjorde prosessen tregere, men førte til at de eksterne punktene ble hentet ut med alle oppdaterte posisjoner.

Selv med bruken av API var det visse problemer datasettene medførte. For eksempel var det ingen tilhørende havner eller leveringspunkter for smoltanlegg.
Smoltanleggene ligger på land, og vil derfor trenge en eventuell havn for å kunne gi bedre antakelser på når et skip har hentet smolt.
Denne informasjonen var heller ikke tilgjengelig gjennom kystdatahuset, og ville derfor kreve manuell gjennomgang av selskapers eiendeler, rapporter, eller å kontakte selskapet direkte for å kunne si hvor smolten blir sendt ut

[1] "Akvakulturregisteret - registre og skjema." https://www.fiskeridir.no/Akvakultur/Registre-og-skjema/ Akvakulturregisteret. Accessed: 2022-01-15.[50]

[2] "Fiskeridirektoratet - api." https://api.fiskeridir.no/pub-aqua/ api/swagger-ui/index.html?configUrl=/pub-aqua/api/ api-docs/swagger-config#/site-resource/getBordersBySiteNr. Accessed: 2022-01-21.[18]

[3] Barentswatch - aapne data og api." https://www.barentswatch.no/ en/articles/open-data-via-barentswatch/. Accessed: 2022-02-03.[21]

fra. Dette vil bli diskutert videre i kapittel 5.1.1.

Et annet problem var datasettet for fartøy utlevert av Kontali. Ettersom dataene for fartøy var satt til to spesifikke perioder, ble det mer komplisert å få AIS-data for fartøy Kontali ikke hadde avtaler med. Dette problemet viste seg når det ble nødvendig å bruke servicebåt-rederier sine fartøy for å kunne gjøre antakelser om lus- og sykdomsbehandling og generelt vedlikehold på oppdrettslokasjoner.
Dette problemet vil bli diskutert videre i kapittel 5.1.1.

Utilgjengelige datasett

Når oppgaven ble diskutert med Kontali, forventet vi store datasett med mange fartøy, og en direkte fasit slik at validiteten til resultatene kunne drøftes. Istedenfor mottok vi to forskjellige båter, og ingen fasit på hva båtene hadde drevet med på forskjellige tidspunkt.

Vi ble dermed bedt om å se på Barentswatch for å vurdere om stoppene ga mening i forhold til resultatene vi hadde fått gjennom algoritmene. Sammenligning med Barentswatch gir kun en antakelse på hva som har foregått uten noen direkte fasit, og gjorde arbeidet med å optimere algoritemene mer komplisert.
Dette blir diskutert videre i kapittel 5.2.2.

Mye av informasjonen innen fiskerinæringen som måtte letes frem var ikke offentlig tilgjengelig. Det ble åpenbart at dette ville skape problemer da vi så på lokasjonene til smoltanlegg. Smoltanleggene ligger på land, men de hadde ingen tilhørende punkter for hvilken havn varene ble sendt ut eller inn fra. Dette resulterte i at sikkerhetsmarginene for smoltanlegg måtte økes betraktelig, eller så måtte gruppen kontakte hvert smoltanlegg i datasettet. Dette kunne muligens gitt oss informasjon om hvilke havner hvert anlegg brukte, som deretter måtte legges inn manuelt.

Denne løsningen var ikke mulig med tiden vi hadde til rådighet, og ble dermed valgt bort. Det er derimot mulig ved utvidelser av prosjektet.
Et lignende problem oppstod i forbindelse med vedlikehold og lus- og sykdomsbehandling. Ettersom vedlikehold av lokasjoner og lus- og sykdomsbehandling krevde informasjon om fartøy Kontali ikke hadde informasjon om,

eller eksempler på – ble de delene av oppgaven urealistiske å oppnå.

Vedlikehold av lokasjoner

Å avgjøre om det som skjer er vedlikehold av lokasjoner var en oppgave som ikke virket komplisert gjennom beskrivelsen av hendelseforløpet. Et vedlikeholdsfartøy eid eller leid inn vil reise ut til oppdrettslokasjonen, utføre vedlikehold og reise tilbake til havnen hvor den har båtplass.
Problemene oppstod her hvor ikke alle selskapene som tok i bruk disse fartøyene hadde den relevante informasjonen offentlig.
Uten direkte kontakt med selskapene ble det dermed vanskelig å få brukt dataene for å gjøre prediksjoner.

Mange av oppdrettsselskapene vil også bruke servicebåtrederi[57][4], som gjør innsamlingen av denne informasjonen mer komplisert. Vi måtte her ha kontaktet hvert individuelle rederi for informasjon om fartøyene de hadde i flåten sin, for dermed å legge inn alle fartøyene manuelt, og skille de basert på forskjellig funksjonalitet[58][5].
Dette ble mer problematisk ettersom fartøyene utlevert av Kontali hadde posisjonsdata for flere måneder tidligere, som da gjorde at å motta flåtens IMO og MMSI-nummer ikke hadde vært nok. For å få brukt dataene måtte hvert selskap sende en fullstendig posisjonslogg for tidsrammene de to utleverte fartøyene hadde operert i.

Lus- og sykdomsbehandling

Problemene ved vedlikehold-antakelse oppstod også innenfor aktivitetsbestemmelse for lus- og sykdomsbehandling. Ifølge Kontali vil både en servicebåt og en brønnbåt være involvert i behandling av lus eller sykdom.
Et eksempel på en slik type behandling kan være en ferskvannsbehandling. Ved en ferskvannsbehandling vil fisken midlertidig pumpes inn i brønnbåten, før servicebåten vil fornye vannkilden. Dermed er det et problem å

[4] S. Olsen, "Ilaks.no - dette er de ti største servicebåtrederiene." https: //ilaks.no/dette-er-de-ti-storste-servicebatrederiene/. Accessed: 2022-04-03[57]

[5] "Follamaritime - type fartøy levert av servicebåtrederi." https:// follamaritime.no/vare-fartoy/#01-oppdrettsfartoy. Accessed: 2022-04-03[58]

finne ut om det har foregått lus eller sykdomsbehandling uten å samtidig kunne sortere ut posisjonene til alle servicebåter som i samme tidsrom stopper ved en oppdrettslokasjon. Uten dette vil alle antakelser være basert på mangelfulle datasett og på denne måten gjøre resultatene unyttige.

En annen måte dette kan bli løst på, er ved å hente ut fiskehelsedataene fra Barentswatch sitt API[21][6]. Her vil det være nødvendig informasjon angående mengden fiskelus i en gitt beholdning på et gitt tidspunkt[59][7]. Datasettene til Barentswatch kunne gitt et ønskelig resultat, men ville gi diverse problemer. Ettersom datasettet kun inneholder informasjon om mengden lus vil ikke datasettene kunne informere om andre typer behandlinger.

Behandlingene kan også skje ved hjelp av fartøy vi ikke har lokasjonen til, og på et tidspunkt vi ikke kan definere, ettersom måling av lus vil skje ukentlig, med mindre lokasjonen har fritak fra lusemåling i den nevnte uken[59][8]. Det kunne dermed blitt problematisk å definere når en lusebehandling foregikk ettersom nye data for lokasjonen ikke ville blitt oppdatert før uken etter, og manglende data til å bestemme når en eventuell servicebåt var ved lokasjonen. Den nærmeste løsningen hadde dermed vært å definere at en lusebehandling ble gjennomført i løpet av uken før.

Dette var derimot ikke en ønskelig løsning på Kontali's vegne. Kontali ønsker å implementere dette systemet i forskjellige regioner verden rundt, i motsetning til i Norge er ikke disse dataene offentlig tilgjengelig i alle land. Dette gjør dermed å hente informasjonen angående sykdom og lusenivåer gjennom Barentswatch til en lite hensiktsmessig løsning for deres prosjekt, de ba oss derfor om å heller fokusere på å optimalisere de andre algoritmene.

Problemer med datasett

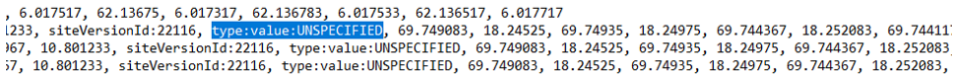
Datasettene som ble brukt, enten de ble utlevert av Kontali eller innsamlet av gruppen, trengte rensing og reformatering. Selv med dette, oppsto det

[6]"Barentswatch - aapne data og api." https://www.barentswatch.no/en/articles/open-data-via-barentswatch/. Accessed: 2022-02-03[21]

[7]"Barentswatch - kart over mengde fiskelus." https://www. barentswatch.no/fiskehelse/2022/14. Accessed: 2022-03-06[59]

[8]"Barentswatch - kart over mengde fiskelus." https://www. barentswatch.no/fiskehelse/2022/14. Accessed: 2022-03-06[59]

noen problemer underveis i prosessen. Blant annet ble det oppdaget at listen med eksterne koordinater for lokasjonene kunne innholde tekst som stanset metodene som behandlet dataene| på geografisk måte. Figur 5.1 viser et lite utdrag av tilfellene av den teksten.



Figur 5.1: Utklipp av eksterne koordinater med tekst blandet inn

Figuren viser tekstdata som en del av listen av geografiske posisjonsdata. Om dette var et problem fra API som leverte datasettet eller våre metoder som brukte datasettet, ble ikke fastslått. Problemet manifesterte seg som en verdifeil for konvertering av streng til flyttall, og løsningen var å ignorere de delene av koordinatparene som hadde feil innhold som i kodeblokk 5.1

Kode 5.1: Utdrag av kodeen for håndtering av eksterne punkter

```
1      # Traverserer gjennom listen av eksterne koordinater
2      for eksterne_koordinat in eksterne_koordinater:
3          try:
4              # Konverterer koordinatet fra streng til flyttall
5              point = float(eksterne_koordinat)
6          # Om innholdet ikke lar seg konvertere, hopp over
7          except ValueError:
8              # Continue hopper over resten av koden, og ...
                   bytter til neste element
9              continue
```

Koden i kodeblokk 5.1 er et eksempel på hvordan noen av problemene vi hadde ble løst. En lik framgangsmåte ble brukt for å håndtere et feiltilfelle hvor polygonene overlappet seg selv og returnerte en feilmelding, vises i kodeblokk 4.6.

5.2 Resultater

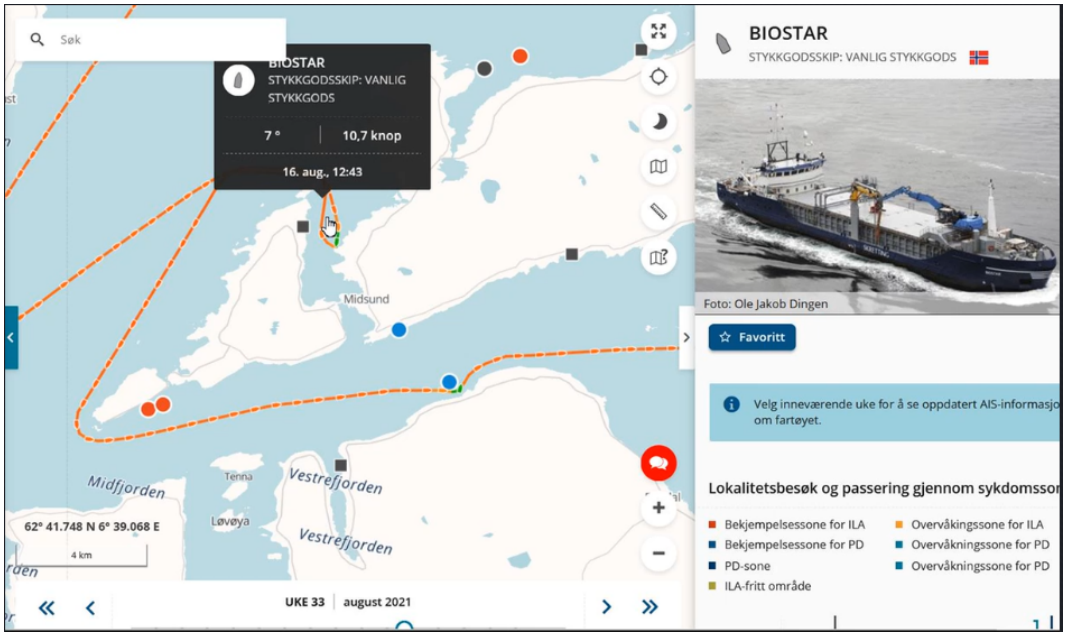
Resultatene fra oppgaven var antakelse om aktiviteten til fartøy basert på sammenhengen mellom stoppene fartøyet hadde hatt. Basert på interesseområdene fra Kontali og datasettene vi fikk tildelt, kan resultatet deles i fem deler: Mat-leveranser, smolt-leveranser, henting av fisk, leveranse av slakteklar fisk og kartets nøyaktighet.

5.2.1 Kartets nøyaktighet

Til å begynne med ble kartets nøyaktighet vurdert opp mot Barentswatch sitt kart, hvor fartøyets informasjon kunne sammenlignes mot deres informasjon. Det ble dermed klart at ruten båten hadde fulgt innenfor de forskjellige tidsperiodene stemte bra med resultatene vi hadde motatt gjennom kartet, men det var tilfeller hvor datasettene var ulike. Dette kan forklares ved feil i datasettene vi fikk utlevert. Ved et par tilfeller viste båtens fart seg som null på kartet vårt, selv om posisjonen endret seg betraktelig både på kartet til Barentswatch og vårt eget.
Dette var noe som måtte tas hensyn til når applikasjonen ble laget.

Farten til fartøy kunne ikke baseres på datasettene alene, men også fartøyets posisjonsendring. Et eksempel på dette vises i figur 5.2 og 5.3.

5.2 Resultater



Figur 5.2: Biostar fart og posisjon gjennom Barentswatch[60]

9

```
2021-08-16 12:31:54,62.649455,6.733187,0.0,,297.0,0.00454626687197121,109.668269706564
2021-08-16 12:47:03,62.649427,6.733265,0.0,,297.0,0.00273932223086166,109.671009028795
2021-08-16 13:06:44,62.649413,6.733132,0.0,,296.0,0.00377775427607614,109.674786783071
```

Figur 5.3: Biostars oppgitte fart i samme tidsrom som i figur 5.2, fart er her opplyst for lettere lesning.

Som vist i figur 5.2 er den reelle farten 10.7 knop, mens i datasettet som ble utlevert er farten 0.0 knop som vist i figur 5.3. Det måtte dermed konstrueres algoritmer som sammenlignet oppgitt fart mot reell lokasjon for å fastslå den reelle farten på fartøyet.
Etter dette var kartet riktig, og prosessen av å definere stopp ble mer presis.

[9]"Barentswatch - kart." https://www.barentswatch.no/fiskehelse/. Accessed: 2022-01-20.[60]

5.2.2 Algoritmer

Algoritmene vi laget hadde som funksjon å angi aktiviteten for et fartøy basert på sammenhengen mellom stoppene det gjorde. Dette var poenget med programmet vårt, og viste seg å være det enkleste aspektet av programmet å skrive kode for. Selve bestemmelsen av aktivitet ble gjort ved å analysere båttypen og lokasjonstypen for besøket. Det var derimot alt arbeidet som la til rette for disse analysene som var krevende.
Ideen for oppgaven holdt seg lik gjennom hele prosjektet, men detaljene endret seg betraktelig. Dette førte til endringer i de fleste aspektene av programmet, noen større enn andre. Frem til de viktigste detaljene var bestemt ble bare et utdrag av datasettet brukt for utvikling og testing, for å spare behandlingstid.

For flere av stegene i prosessen, førte bestemmelser om hvordan dataene skulle behandles til endringer i tidligere steg. Fremgangen ble da langsom, og dataene og filene vokste raskt i størrelse. Det var spesielt metodene som skulle angi om fartøy hadde stoppet hos kjente lokasjoner som ga problemer.

Noen av problemene var relatert til feil i datasettene, som måtte håndteres på forskjellige måte. Noen problemer var relatert til fremgangsmåten, som endret seg drastisk basert på resultatene fra metodene.

De forskjellige metodene tok feilmarginene som parameter, som ble justert som en del av de siste stegene av prosessen. Til slutt kom vi frem til et resultat som virket sannsynlig, og støttet hypotesen om at aktiviteten til et fartøy kan knyttes til stoppene det har gjort.

```
53  ronjaocean, 2021-10-14 12:46:10, 66.033735, 12.024503, S�R G�SV�R, Oppdrett, Slakteri, Levering av fisk for slakting, ,
54  ronjaocean, 2021-10-14 23:57:44, 64.85703, 11.248937, MAR�YA, Slakteri, Oppdrett, Henting av fisk, ,
55  ronjaocean, 2021-10-16 07:22:10, 66.468928, 12.96872, BREIVIKEN, Oppdrett, Slakteri, Levering av fisk for slakting, ,
56  ronjaocean, 2021-10-16 22:13:51, 64.85502, 11.24749, MAR�YA, Slakteri, Oppdrett, Henting av fisk, ,
57  ronjaocean, 2021-10-18 03:56:40, 64.956783, 11.718843, OSAVATNET, Oppdrett, Slakteri, Levering av fisk for slakting, ,
```

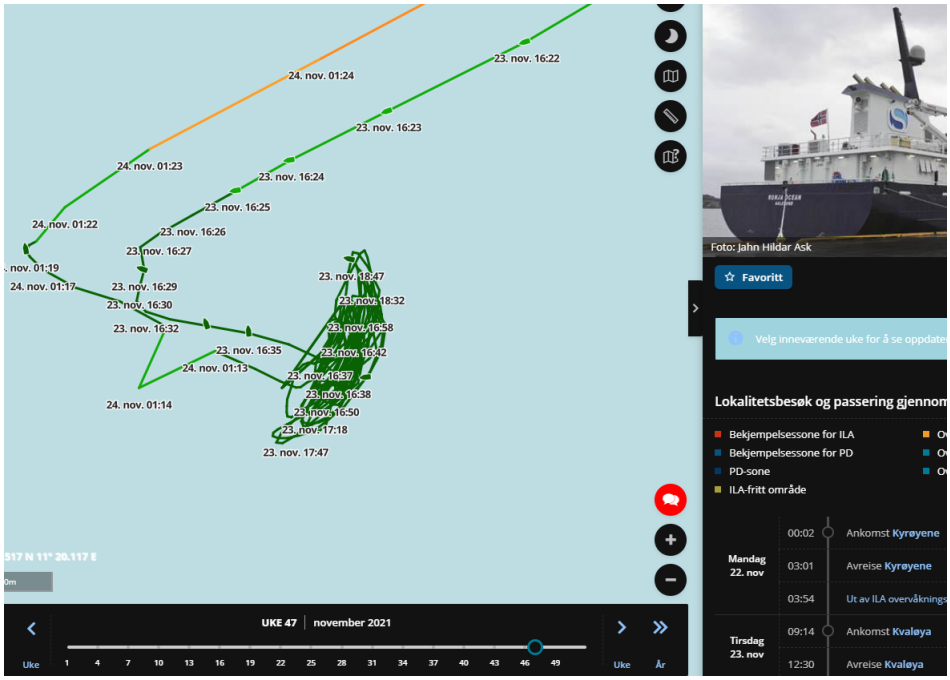
Figur 5.4: Ronja Ocean stopp og aktivitet

```
Boatname  Time  Lat  Lon  Location  Previous locationtype  Current Locationtype  Assumed activity
biostar, 2021-08-16 07:13:22, 62.47838, 6.227363, HOLSNESET, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-16 09:49:42, 62.687348, 6.659628, BJORNEREMSBUKTA, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-16 12:01:13, 62.649145, 6.733245, MYRANE, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-16 21:12:41, 62.990288, 7.330673, GAUSTAD, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-17 04:01:22, 63.455118, 8.950377, SL�TTHOLMEN, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-17 16:39:51, 63.174275, 8.621247, RENNDALEN, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-18 13:25:01, 62.53116, 6.422032, GJERSET V, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-18 17:42:34, 62.16512, 5.596197, BJ�RLYKKESTRANDA, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-19 06:28:53, 60.681563, 4.761197, VADHOLMEN, Oppdrett, Oppdrett, Matlevering, ,
biostar, 2021-08-19 10:53:00, 60.0833, 5.121, EIDHOLMEN, Oppdrett, Oppdrett, Matlevering, ,
```

Figur 5.5: Biostar stopp og aktivitet

Figurene 5.5 og 5.4 viser resultatene fra algoritmen basert på de stoppene vi hadde laget for båtene. Fra venstre er det informasjon om båten og stoppet den har hatt. Deretter kommer navnet på lokasjoner vi antar fartøyet har stoppet ved, samt hvilke typer lokasjoner. Basert på båten, typen lokasjon for et stopp og typen lokasjon for forrige stopp, kommer antatt aktivitet. Algoritmen undersøkte sammenhengen mellom to stopp om gangen, basert på rådgivning fra Kontali.

Figurene viser et utdrag av stoppene som ble funnet for båtene, både hos kjente lokasjoner og ukjente. For stopp ved ukjente lokasjoner har vi ingen måte å bedømme aktivitet på, båtene kan legge til ukjent kai eller ankre opp for natten, som i figur 5.6.



Figur 5.6: Ronja Ocean stopper for ukjent grunn[60]

10

Resten av aktivitetene diskuteres videre:

Mat-leveranser

For vårt datasett var matleveransene knyttet til en av de to båtene. Biostar hadde bare matleveranser, basert på informasjon fra Kontali. Figur 5.5 viser et utdrag av stoppene som ble funnet for Biostar hos kjente lokasjoner. Siden Biostar er et stykkgodsskip er det ikke mange mulige aktiviteter som kan antas.

[10]Barentswatch - kart." https://www.barentswatch.no/fiskehelse/. Accessed: 2022-01-20.[60]

Smolt-utsettelse

Utsetting av smolt er en viktig del av oppdrettsnæringen, men resultatet reflekterte ikke dette. Resultatene for Ronja Ocean inneholdt først ingen besøk hos smoltanlegg. Til å begynne med lignet dette på feil fra vår side, men videre manuell undersøkelse av datasettet og Barenswatch tydet på at smoltanleggene vi hadde ikke var tilstrekkelige for å bruke. Fra Barenswatch så vi at lokasjonene som ble hentet ikke hadde blitt besøkt av fartøyet, og siden anleggene lå på land var ikke dette så rart.
Vi endte dermed opp med å ha større marginer kun for smoltanlegg, slik at et stopp ved en smoltlokasjon kanskje ville bli hentet opp om et skip var i området til et smoltanlegg. Problemet med dette var at marginene måtte bli så store for å få treff for smoltanlegg, at det logisk ikke kunne stemme. Det ble derfor falske positive treff.

Vi konsulterte Kontali angående dette, og en av vurderingene vi tok var da om det ville hjulpet å hente lokasjonen til alle Norges havner fra kystdatahuset[61] [11]. Ettersom det ikke var mulig å trekke en direkte korrelasjon mellom en havn og henting av smolt ble dette valgt bort. Det er mulig å finne ut hvilke havner som tilhører hvilke smoltanlegg, eller hvilke smoltanlegg som bruker hvilke havner i de fleste tilfellene. Dette er derimot informasjon som måtte ha blitt hentet ut fra hvert enkelt selskap, eller selskapenes offentlige rapporter manuelt. Det hadde derfor tatt for mye tid uten å nødvendigvis lede til bedre resultater, ettersom å finne denne informasjonen ikke alltid ville være mulig.

Å inkludere havnene ville også føre til andre problemer grunnet mengden havner for kommersiell og privat bruk. Det ville ikke vært mulig å sortere de basert på bruksområde, og grunnet det store antallet kunne de skape støy for andre prediksjoner.
Dette kunne for eksempel forekomme når et fartøy leverte fisk til en slaktemerd. Her vil det ofte være en havn i nærheten av slakteriet, og havnens sikkerhetsmargin kan da overlappe med slaktemerdens. Å inkludere havner ga derfor ingen ekstra relevant informasjon, men kunne føre til feil antakelser og var derfor lite hensiktsmessig.

Uten havnene ble sikkerhetsmarginen økt betraktelig, selv da fikk vi ingen

[11] "Kystdatahuset, hjemmeside." https://kystdatahuset.no/. Accessed: 2022-03-20.[61]

resulterende stopp for leveranser av smolt. Vi kontaktet Kontali om problemet, som deretter forhørte seg med selskapet som utleverte skipsloggen for Ronja Ocean. Det kom da frem at Ronja Ocean ikke hadde foretatt seg noen leveranser av smolt i perioden.

Henting av slakteklar fisk

Hentingen av slakteklar fisk gjøres av brønnbåter som Ronja Ocean, og flere av stoppene vi fant ga denne aktiviteten. I figur 5.4 viser at Ronja Ocean har flere mulige aktiviteter og som bestemmes av typen lokasjon som er stoppet ved. For de sammenhengene hvor båten kom til oppdrettsanlegg, fra slakteri eller andre plasser, var henting av fisk en mulig aktivitet.

Leveranser av slakteklar fisk

Igjen var det brønnbåten Ronja Ocean som var involvert med leveransen av fisk til slakteriene. Den samme fisken som ble hentet opp hos oppdrettsanleggene, leveres til slakteri. Siden slakteri har få oppgaver, er det få grunner for et fartøy å stoppe hos slakteriet, annet enn å levere fisk til slakting. Andre typer båter kan nok være en del av den logistikken slakteriet har, men for brønnbåter er det kun én aktivitet som gir mening.

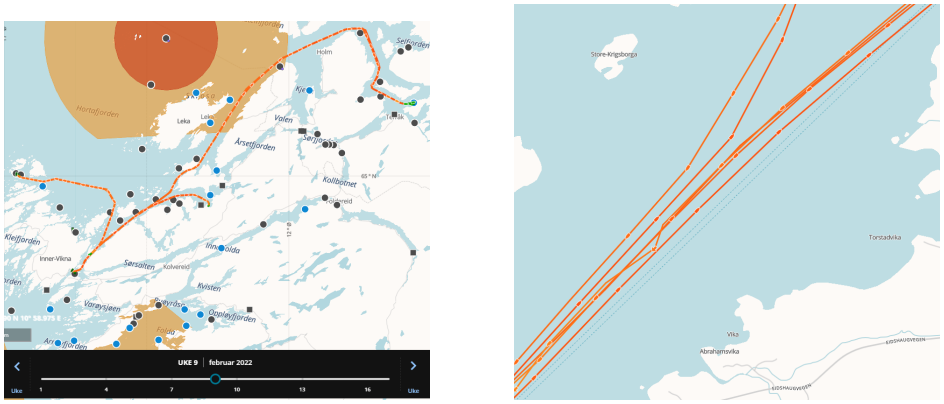
Selve bestemmelsen av aktivitet innebar ingen databehandling eller manipulering, men krevde rene og gode data som parameter. Dette var den største faktoren som påvirket resultatet til aktivitetsalgoritmen.

Sammenligning med Barenswatch

For å bekrefte resultatene vi fikk uten en tildelt fasit, ble strategien å bruke Barenswatch og undersøke om de stoppene vi kartla stemte overens med det Barenswatch hadde. Vi undersøkte muligheten for å automatisere denne prosessen, men fant ingen gode metoder for dette. Resterende metode var manuell navigering av tidspunkt og posisjon for enkeltbåter vi ville undersøke.

Den manuelle navigeringen av Barenswatch fungerer best for kortvarige undersøkelser, og ble mer problematisk for vårt bruksområde. Dette skyldes trolig den store mengden data som prosesseres av nettapplikasjonen. Nettsiden er lite responsiv og bruker lang tid på laste informasjonen man vil se.

Fartøyshistorikken vises med intervaller på en uke, og illustreres som fargekodete linjer. For å undersøke fartøyes posisjon for enkelte tidspunkt, ble vi nødt til å zoome inn på kartet og lokalisere posisjoner nært tidsrommet. Ved første posisjon som var nærliggende interessetidspunktet, måtte vi følge de markerte linjene i rett retning til vi fant rett tidspunkt.
Eksempler på hvordan dette ser ut er vist i figur 5.7.



(a) Barenswatch standardvisning for valgt uke[60]

(b) Posisjonslinjene for Ronja Ocean[60]

Figur 5.7: Barenswatch visning av fartøyshistorikk[60]

12

Figuren viser standardvisningen for en valgt uke, for et valg fartøy, på nettsiden Barenswatch. Figur 5.7b viser hvordan posisjonslinjen for Ronja Ocean illustreres, her ønsker vi å vise problematikken ved å finne presise posisjoner til visse tidspunkt. Navigeringen langs ved disse linjene, fra tilfeldige startpunkter til interesseområder, ble vanskeligere på grunn av de forskjellige linjene som overlapper og krysses langs de vanligste rutene for båten.

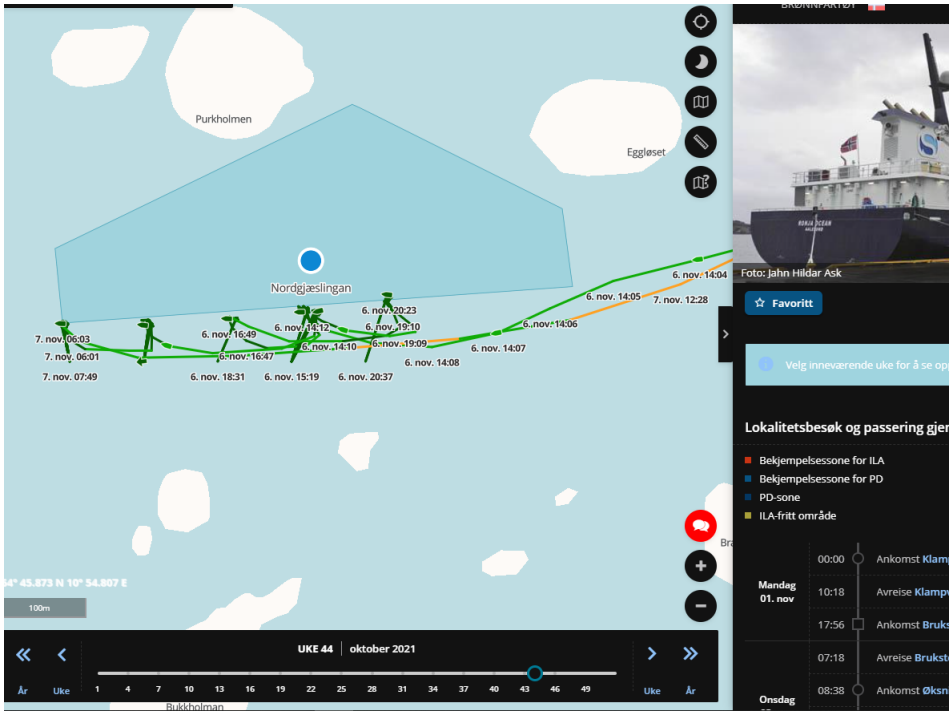
[12]Barentswatch - kart." https://www.barentswatch.no/fiskehelse/. Accessed: 2022-01-20.[60]

5.2 Resultater

Underveis i denne prosessen erfarte vi et annet problem med nettsiden Barenswatch; returnering til standardperspektiv. Ved det vi oppfattet som tilfeldige tidsintervall, ble nettsiden tilbakesatt til en standard visning av fartøyet for en valgt uke. Dette problemet oppsto flere ganger under navigeringen mot spesifikke tidspunkt vi ville undersøke, og nullstilte progresjon vi hadde oppnådd mot å finne ønsket informasjon.

Manuell navigering av Barenswatch var den eneste måten å sjekke stoppene til båten på, og var en tungvindt og vanskelig prosess. Et positivt aspekt ved Barenswatch er fargekoden på posisjoner basert på fart, noe som gjorde det lettere å finne og undersøke plasser hvor båten var loggført som stoppet hos Barenswatch. Dette kunne vi bruke til å sjekke stoppene til Barenswatch opp mot våre, men den motsatte prosessen ble for vanskelig.

Et av stoppene som ble funnet hos Barenswatch, men som ikke var lagret som et stopp i vår database er vist i figur 5.8.



Figur 5.8: Ronja Ocean stopper ved kjent lokasjon, utenfor fargelagt område[60]

Figuren viser hvordan et stopp ved en kjent lokasjon ser ut. Ronja Ocean vises som en grønn pil og strek rundt figuren som referer den kjente lokasjon "Nordgjæslingan".
Metoden for å bestemme om et fartøy har stoppet plukket opp disse posisjonene for Ronja Ocean og lagret dem som et stopp, men koblet ikke stoppet opp mot lokasjonen. Problemet var at noen av parameterene for sikkerhetsmarginene for algoritmen var for dårlig justert, de var for små. De parameterene ble justert slik at stoppet til slutt ble registert ved lokasjonen.
Den mest vesentlige variabelen for dette var nok marginen fra båten sin posisjon, som økte sjansen for overlapp med nærliggende lokasjoner.

5.3 Refleksjon

5.3.1 Teknologivalg

Teknologivalgene som ble tatt iløpet av oppgaven ble valgt for å gjøre oppgaven så lett gjennomførlig, vedlikeholdbar og leselig som mulig. Vi føler at teknologivalgene fungerte godt mot målene vi satt oss, og hva oppgaven var ment til å gjøre.

Python har fungert utmerket for utregningene og behandlingen av filene prosjektet inneholdt, og SQLite har hjulpet kjøretiden betraktelig. Selv om innføringstiden av datasett på størrelsen vi hadde er på flere minutter, blir denne tiden spart i bruken av dataene. Etter datapunktene er satt inn i databasen vil videre bruk og endring være mye raskere enn alternative metoder.
For større datasett ville nok optimalisering for kjøretid ha hjulpet mye med ytelsen, men for prosjekt av denne størreslsen er det ikke så mye å spare.
En del av oppgaven som kunne hatt godt av revurdering er valg av kart. Det statiske kartet fungerte godt for formålet sitt i denne oppgaven, ved å simulere og visualisere datasettene som hadde blitt hentet. Ettersom dataene som ble satt inn ikke krevde like mye omregning og analyse som vi først tenkte, ville nok Mapbox ha vært et bedre valg enn først tenkt. I et reellt tilfelle ville kartets statiske oppsett ha ført til for lang kjøretid, og dermed ikke vært optimalt for Kontali.

Teknologivalgene fungerte generelt bra, basert på grunnlaget de ble valgt på. For uten mindre endringer, som ville hatt liten eller ingen effekt på prosjektet, vil si oss fornøyde med valgene som ble tatt.

5.3.2 Endringer i prosjektet

Ettersom dette er et prosjekt hvor vi skulle se om oppgaven var gjennomførbar, ble det gjort noen endringer underveis grunnet revurdering av teorier, og muligheten for å se sammenhenger. En av disse endringene var da som nevnt i kapittel 5.1.1 ekskluderingen av lus- og sykdomsbehandling, og vedlikehold.
I begynnelsen var det mye informasjonshenting for å forstå hva de forskjellige stoppene innebærer. Denne informasjonen var ikke alltid tilgjengelig, og førte dermed til flere møter med Kontali, lesing av dokumenter for sjøfart og regler for oppdrettslokasjoner.

Til å begynne med virket det som de nødvendige dataene for å bestemme aktivitet rundt lusebehandling var tilstrekkelig for gjennomføring, ettersom informasjonen var tilgjengelig gjennom Barentswatch. Denne løsningen var ikke ønskelig grunnet at den ikke kunne implementeres i større system, da løsningen var avhengig av Barentswatch sine åpne data rundt norsk oppdrett.
Innsamlingen av lusedata måtte dermed fjernes fra systemet, og dato sammenligningsystemet som var under konstruksjon ble fjernet.

Andre endringer oppgaven gikk gjennom var ved bruk av kart. I begynnelsen stod ArcGIS frem som det åpenbare valget, det hadde god dokumentasjon, gode muligheter for ikon eller punkt avsetning, og ble derfor valgt i begynnelsen av prosjektet. Etter mye arbeid og feilsøking ble det klart at ArcGIS Pro var nødvendig for å kunne bruke karttjenesten. Betalingsbarrieren ble dermed for stor til at det var verdt det for dette prosjektet, ArcGIS konstruksjonen ble dermed lagt bort for andre valget, som var Folium.

En annen endring som ble gjort underveis var i forhold til databasen. Valget om å bruke en database for databehandling ble tatt tidlig i utviklingen av applikasjonen, etter vi erfarte hvor tungvindt det var å bruke tekstfiler til de forskjellige datasettene.
Underveis i utviklingen ble det også gjort endringer innad databasen, i form

av strukturen på tabeller og hvilke tabeller vi trengte. Dette kan skyldes en mindre gjennomført planlegging av databasen.

Kapittel 6

Konklusjon

Konklusjonen for oppgaven er at det er mulig å bruke sammenhengen mellom stoppene et fartøy har for å ekstrapolere hva fartøyet gjør.

Noen aktiviteter vil være lettere å anta enn andre, slik som at brønnfartøy som stopper hos slakteri skal mest sannsynlig levere fisk til slakting.
For smolt-utsettelse vil tilleggsinformasjon være nødvendig for å bekrefte resultatene.
For vedlikehold av oppdrettslokasjoner og lus- og sykdomsbehandling vil avtaler med servicebåtrederier eller oppdrettsselskap være nødvendig for å oppnå ønskede resultat.

En viktig faktor for en slik applikasjon er ren og god data, og god behandling av data. Siden reelle fartøy og lokasjoner ble brukt, må feilmarginer brukes for å kompensere for de unøyaktighetene som naturligvis vil forekomme.
Grunnet at flere av målene med oppgaven krever kontakt med individuelle selskaper, var ikke disse mulig å oppnå innen rammene vi jobbet innefor. Det kan derimot la seg gjøre om Kontali tar seg tiden til å finne dataene på egenhånd, eller lager avtaler med selskapene som kan ha denne informasjonen. Vi har dermed fått gjennomført de oppgavene som var realistiske, mens vi har funnet ut hvordan dette kan gjøres for Kontali når de skal lage sitt eget system for andre regioner.

Om vi skulle gjennomført oppgaven på nytt med kunnskapen vi har nå

og bedre tid, ville vi prøvd å opprette dialog med flere servicebåtrederier, oppdrettsselskaper og smoltanlegg. Vi kunne dermed fått et større datasett, og kunne muligens oppnådd bedre resultater på de resterende målene for oppgaven.

Som beskrevet i 5.1.1 var ikke mengden data utlevert nok til å kunne gjøre reelle antakelser på. Selv om det er gjort vurderinger for smoltanlegg, kan vi ikke bekrefte at de resulterende sikkerhetsmarginene er passende. Ettersom datasettet for Ronja Ocean ikke inneholdt smoltleveranser, er altså ikke den delen av oppgaven mulig å vurdere heller.
Datasettene hadde også flere mangler og feil, og var ikke tilstrekkelige for å gjøre vurderinger på. Gruppen prøvde derfor etter beste evne å gjøre opp for dette ved å lete frem nye datasett, men ettersom mange av dataene som var nødvendige ikke var tilgjengelige, som forklart i 5.1.1, var det ikke realistisk med tiden som var til rådighet.
Mangelen på en fasit gjorde også at resultatene ikke kunne bekreftes. Gruppen måtte derfor sette av mye tid på å manuelt gjennomsøke resultatene, for deretter å sammenligne med Barentswatch.
De resulterende aktivitetene så ut til å å samsvare med Barentswatch sine stopp, men dette kan ikke sies sikkert.

Datasettene utlevert utelukket derfor tre av de fem målene. I tillegg hindret de mangelfulle datasettene gruppen fra å optimalisere algoritmene på en tilfredstillende måte.

Gruppearbeidet kan derimot ansees som veldig bra. Ingen på gruppen hadde tidligere erfaring med fiskerinæringen, men brukte tiden godt og fikk god oversikt over de forskjellige aspektene innenfor industrien.
Gruppedynamikken fungerte godt som et resultat av bekjentskap og personlige erfaringer, og derav kjennskap til interne styrker og svakheter. Vi kunne dele oppgaver ut fra disse styrkene og svakhetene, og raskt komme frem til gode løsninger.
Det ble holdt kontinuerlig dialog innad gruppen, som førte til godt overblikk over arbeidsoppgaver og eliminerte behov for unødvendige forklaringer.

Prosjektet kan konkluderes som en suksess for vårt primære mål, nemlig at aktiviteter kan antas basert på sammenhengen mellom stoppene for et fartøy. For resterende mål relatert til fiskerilokasjoner kreves det bedre informasjonsinnsamling, mulig direkte kommunikasjon med eventuelle loka-

sjoner, flere fartøy og tid til å bygge opp en større applikasjon.

Bibliografi

[1] "Kystverket - ais beskrivelse." https://www.kystverket.no/en/navigation-and-monitoring/ais/. Accessed: 2022-01-15.

[2] T. Skjørten, "Visma - hva er api, spørsmål og svar." https://www.visma.no/blogg/hva-er-api-sporsmal-og-svar/. Accessed: 2022-03-15.

[3] K. Hofstad, "Store norske leksikon - breddegrad." https://snl.no/breddegrad. Accessed: 2022-01-19.

[4] J. P. Johnsen, "Store norske leksikon." https://snl.no/br\T1\onnbåt. Accessed: 2022-02-22.

[5] "Wikipedia - avstand." https://no.wikipedia.org/wiki/Avstand. Accessed: 2022-03-15.

[6] "Wikipedia - gis." https://en.wikipedia.org/wiki/Geographic_information_system. Accessed: 2022-01-25.

[7] "Imo - hjemmeside." https://www.imo.org/. Accessed: 2022-01-15.

[8] K. Hofstad, "Store norske leksikon." https://snl.no/lengdegrad. Accessed: 2022-01-16.

[9] "Store norske leksikon - fiskeoppdrett." https://snl.no/fiskeoppdrett. Accessed: 2022-01-15.

[10] B. Misund, "Store norske leksikon - merd." https://snl.no/merd. Accessed: 2022-01-19.

[11] "Navcen - mmsi." https://www.navcen.uscg.gov/?pageName=mtmmsi. Accessed: 2022-01-15.

[12] "Wikipedia - parsing." https://no.wikipedia.org/wiki/Parsing. Accessed: 2022-03-15.

[13] J. E. Vatne, "Store norske leksikon." https://snl.no/polygon. Accessed: 2022-02-22.

[14] "Kulturnav - slaktemerd." https://kulturnav.org/de5a191e-8ce1-45da-b4ac-48654afc318a. Accessed: 2022-01-19.

[15] A. Vøllestad, "Store norske leksikon." https://snl.no/smolt. Accessed: 2022-02-22.

[16] "Kontali - hjemmeside." https://www.kontali.no/. Accessed: 2022-01-15.

[17] "Fiskedirektoratet, akvakultur karttjeneste." https://portal.fiskeridir.no/portal/apps/webappviewer/index.html?id=87d862c458774397a84. Accessed: 2022-01-20.

[18] "Fiskeridirektoratet - api." https://api.fiskeridir.no/pub-aqua/api/swagger-ui/index.html?configUrl=/pub-aqua/api/api-docs/swagger-config#/site-resource/getBordersBySiteNr. Accessed: 2022-01-21.

[19] "Barentswatch - hjemmeside." https://www.barentswatch.no/. Accessed: 2022-01-15.

[20] "Barentswatch - Åpne data." https://www.barentswatch.no/artikler/apnedata/. Accessed: 2022-01-15.

[21] "Barentswatch - aapne data og api." https://www.barentswatch.no/en/articles/open-data-via-barentswatch/. Accessed: 2022-02-03.

[22] "Github wikipedia page." https://en.wikipedia.org/wiki/GitHub. Accessed: 2022-01-20.

[23] "Github actions dokumentasjon." https://docs.github.com/en/actions. Accessed: 2022-02-03.

[24] "Softkraft - golang vs python." https://www.softkraft.co/golang-vs-python/. Accessed: 2022-01-18.

[25] "Go - dokumentasjon." https://go.dev/doc/. Accessed: 2022-01-15.

[26] D. Shappir, "Forbes - why did microsoft create c#." https://www.forbes.com/sites/quora/2018/03/02/why-did-microsoft-create-c/. Accessed: 2022-01-18.

[27] "Microsoft - c# dokumentasjon." https://docs.microsoft.com/en-us/dotnet/csharp/. Accessed: 2022-01-18.

[28] "Bestcolleges.com - 6 easiest programming languages to learn." https://www.bestcolleges.com/bootcamps/guides/6-easiest-programming-languages-to-learn/. Accessed: 2022-01-20.

[29] "Python - dokumentasjon." https://docs.python.org/3/. Accessed: 2022-01-15.

[30] "Programming-language-benchmarks - python vs c#." https://programming-language-benchmarks.vercel.app/python-vs-csharp. Accessed: 2022-01-18.

[31] "Insightsstackoverflow - integrated development environment." https://insights.stackoverflow.com/survey/2021. Accessed: 2022-01-17.

[32] "Wikipedia - visual studio." https://no.wikipedia.org/wiki/Microsoft_Visual_Studio. Accessed: 2022-03-20.

[33] "Vs code nettside." https://code.visualstudio.com. Accessed: 2022-03-20.

[34] "Jupyter notebook nettside." https://jupyter.org. Accessed: 2022-01-24.

[35] "Sqlite - copyright nettside." https://www.sqlite.org/copyright.html. Accessed: 2022-03-22.

[36] "Sqlite - serverless nettside." https://www.sqlite.org/serverless.html. Accessed: 2022-02-22.

[37] "Sqlite - quirks nettside." https://www.sqlite.org/quirks.html. Accessed: 2022-02-22.

[38] "Sqlite - nettside." https://www.sqlite.org/index.html. Accessed: 2022-02-21.

[39] "Wikipedia - mysql." https://no.wikipedia.org/wiki/MySQL. Accessed: 2022-03-22.

[40] "Mysql - dokumentasjon." https://dev.mysql.com/doc/. Accessed: 2022-03-22.

[41] "Mysql nettside." https://www.mysql.com. Accessed: 2022-03-21.

[42] "Statista - most popular us mapping apps ranked by reach." https://www.statista.com/statistics/865419/most-popular-us-mapping-apps-ranked-by-reach//. Accessed: 2022-01-17.

[43] "Mapsplatform - karttjeneste valg hos selskap." https://mapsplatform.google.com/. Accessed: 2022-01-17.

[44] "Google maps - prisliste." https://mapsplatform.google.com/pricing/. Accessed: 2022-01-17.

[45] "Esri - arcgis." https://www.esri.com/en-us/arcgis/products/arcgis-online/overview. Accessed: 2022-03-21.

[46] "Mapbox - dokumentasjon." https://docs.mapbox.com/. Accessed: 2022-01-18.

[47] "Plotly - dokumentasjon." https://plotly.com/python/. Accessed: 2022-01-18.

[48] "Data2int - folium." https://data2int.com/Foliuml. Accessed: 2022-01-16.

[49] "Folium dokumentasjon." https://autogis-site.readthedocs.io/en/latest/notebooks/L5/02_interactive-map-folium.html. Accessed: 2022-01-16.

[50] "Akvakulturregisteret - registre og skjema." https://www.fiskeridir.no/Akvakultur/Registre-og-skjema/Akvakulturregisteret. Accessed: 2022-01-15.

[51] "Myshiptracking - nettside." https://www.myshiptracking.com/. Accessed: 2022-01-15.

[52] "Crummy - beautifulsoup dokumentasjon." https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Accessed: 2022-01-16.

[53] Riccardo, "Digital-geography.com - vessel tracking the python way." https://digital-geography.com/vessel-tracking-the-python-way/. Accessed: 2022-02-03.

[54] "Pypi - shapely hovedside." https://pypi.org/project/Shapely/. Accessed: 2022-03-06.

[55] "Geeksforgeeks - haversine formelen." https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/. Accessed: 2022-03-23.

[56] "Shapely - dokumentasjon nettside." https://shapely.readthedocs.io/en/stable/manual.html. Accessed: 2022-03-22.

[57] S. Olsen, "Ilaks.no - dette er de ti største servicebåtrederiene." https://ilaks.no/dette-er-de-ti-storste-servicebatrederiene/. Accessed: 2022-04-03.

[58] "Follamaritime - type fartøy levert av servicebåtrederi." https://follamaritime.no/vare-fartoy/#01-oppdrettsfartoy. Accessed: 2022-04-03.

[59] "Barentswatch - kart over mengde fiskelus." https://www.barentswatch.no/fiskehelse/2022/14. Accessed: 2022-03-06.

[60] "Barentswatch - kart." https://www.barentswatch.no/fiskehelse/. Accessed: 2022-01-20.

[61] "Kystdatahuset, hjemmeside." https://kystdatahuset.no/. Accessed: 2022-03-20.