Norges miljø- og biovitenskapelige universitet

Masteroppgave 2020 30 STP
Handelshøyskolen
Veileder: Dag Einar Sommervoll

Process Mining & Maskinlæring i den Prehospitale Klinikk

- Big Data analyse av prosesser tilknyttet ambulansetjeneste i New York

Erlend Oliver Eriksen
Jonas Michael Jensen
Master i Økonomi og Administrasjon. Spesialisering: Business Analytics.

Abstract

The health service is an important part of society. The much discussed wave of the elderly will increase the need for health services, and when unexpected pandemics occur, the health service is a vulnerable part of society. The need for efficiency is often a discussed topic, and data analysis techniques can be part of the solution. If data analysis methods have potential - such methods must be tested in the context of the health sector, and with health related data. This study aims to retrieve and study information about processes and activities - in New York, Manhattan. The data used in the study is from New York's ambulance service. This involves all activities from when a citizen calls to the emergency center for help, and to the time when the citizen is possibly brought to the hospital. The data addresses cardiac arrest patients. The study uses Process Mining and Machine Learning methods to retrieve information about the data. Using the Process Mining algorithm, alpha plus, a process model (Petri net) was produced. The Petri net model aimed to highlight the activities associated with ambulance calls, during the period 2008 to 2019. Targets in the study were used to assess model quality. The target figures used were precision, generalization, fitness and simplicity. The result communicated that the model had largely captured much of the important information in the data. The average score was 90.38 %, of a maximum of 100 %.

When using Machine Learning, several algorithms were used to create a model that was intended to estimate the total time spent on a case (throughput time). The same data was used for Machine Learning as in Process Mining. The best algorithm for prediction of throughput time was the model based on XGBoost. The model was trained using repeated cross validation and random search was used to find the optimal hyperparameters. XGBoost received an MAE of 627.

Sammendrag

Helsetjenesten er en viktig del av samfunnet. Den mye omtalte eldrebølgen vil øke behovet for helsetjenester, og når uventede pandemier oppstår, så er helsetjenesten en utsatt del i samfunnet. Behovet for effektivisering er ofte et diskutert tema, og datanalyse metoder kan være en del av løsningen. Om dataanalyse metoder har potensiale, må slike metoder utprøves i kontekst av helsesektoren, og da helse relatert data. Studien tar til sikte på å uthente, og studere informasjon om prosesser og aktiviteter - i New York, Manhattan. Dataene som brukes i studien er fra New York sin ambulansetjeneste. Dette innebærer alle aktiviteter fra når en borger ringer inn til nødsentralen for hjelp, og til det tidspunkt når borgeren eventuelt blir brakt med til sykehuset. Dataene omhandler hjertestans pasienter. Studien bruker metodene Process Mining og Maskinlæring for å uthente informasjon om dataene. Ved bruk av Process Mining algoritmen, alfa pluss, ble en prosessmodell (Petri net) produsert. Petri net modellen tok til sikte på å fremheve aktivitetene som er tilknyttet ambulanseutrykninger, i tidsperioden 2008 til 2019. Måltall i studien ble brukt til å vurdere modellkvalitet. Måltallene som ble brukt var precision, generalization, fitness og simplicity. Måltallene i studien kommuniserte at modellen hadde i stor grad fanget opp mye viktig informasjon i dataene, der gjennomsnitt score til måltallene ble 90.38%, av maksimalt 100%.

Ved bruk av Maskinlæring ble flere algoritmer brukt for å lage en modell som hadde til hensikt å estimere den totale tidsbruken til én case (gjennomstrømstid). Samme datagrunnlag som ble brukt for Process Mining ble brukt i Maskinlæring. Den beste algoritmen for prediksjon av gjennomstrømstid var modellen basert på XGBoost. Modellen ble trenet ved hjelp av repetert kryssvalidering, der tilfeldig søk ble andvendt for å finne de optimale hyperparameterene. XGBoost fikk en MAE på 627.

Forord

Vi vil takke alle kjente og kjære som har vært med på å støtte oss med oppgaveskrivingen. Takk til dere som har hjulpet til med både stort og smått.

Med vennlig hilsen

Jonas Michael Jensen og Erlend Oliver Eriksen

Innholdsfortegnelse

List av Symboler og Forkortelser

Symboler

α = Alpha. Hyperparameter for regularisering algoritmer eller konstantledd i multippel regresjon

β = Beta = Modell parameter / Koeffisient

ε = Restledd/Feilledd/Residual

λ = Lambda. Hyperparameter for regulariserings algoritmer

L = Hendelsesloggen

R^2 = Determinasjonskoeffisient

Forkortelser

ABT = Analytisk Base Tabell

AI/KI = Artificial intelligence/Kunstig intelligens

BPM = Business process managment

BPMN = Business process model notation

CRAN = Comprehensive R Archive Network

EDA = Eksplorativ Data Analyse

FDNY EMS = New York City Fire Department Bureau of Emergency Medical Services

FEMS = District of Columbia Fire and Emergency Medical Services Department

HIPA = Health Insurance Portability and Accountability Act

IOT = Internet of things

IT = Informasjonsteknologi

MAE = Mean absolute error

ML = Maskinlæring

MLR = Multivariat lineær regresjon

MSE = Mean squared Error

NA = Not available/missing value

NFPA = National Fire Protection Association

NYC = New york city

PM = Process mining

RMSE = Root mean square error

Liste av Figurer

Liste av Tabeller

Kapittel 1

Innledning

Data er den nye vinen i business analytics. I helsesektoren så har det i nyere tid blitt observert flere mulige måter en kan ta i bruk data på. Primært for å forbedre lønnsomhet, effektivitet og forskning. Noen eksempler er følgende; Ved bruk av analytiske verktøy, så kan helsemyndigheter estimere utfallet ved den årlige influensa med stor presisjon (Jank, 2011, s.2). I det amerikanske helsevesen er det implementert deteksjon metoder for å kunne estimere hvem som urettmessig mottar stønad (Davenport & Harris, 2017, s.82). Dataanalyse har i USA bidratt til effektiv ressursallokering[1], i helsesektoren. Med historiske data kan tiden for sykehusopphold bli predikert (Mans, Aalst & Vanwersch, 2015, s.5). Rettet mot helseforskning, brukes maskinlæring med stor suksess til å kunne, blant annet, predikere sykdomsforløp[2], samt estimering av forsikringspremier[3].

Demokratisering av IT verktøy og økt prosessorkraft er primærkilden for teknologiske fremskritt (Westerman, Bonnet & McAfee, 2014, s.1-3). Hvor Moores law[4] har vist seg å stemme, samt ha stor påvirkning på hvordan organisasjoner utfører sitt arbeid (Davenport, 2017, s.8). I kjølvannet av denne ekspansive utviklingen innad i IT, så har det gitt uante muligheter for organisasjoner til å utnytte nye teknologier, som ikke har kunne blitt nyttiggjort tidligere. Effektiv utnyttelse av IT drevne verktøy vil være av stor nytte, da dette er med på å bedre kundetjenester (Westerman et al., 2014, s.29).

Gradsoppgaven er strukturert i seks deler og benytter seg av ambulansedata fra New York city. Målet er å analysere ambulansedata. Metodene i fokus vil omhandle analyse av prosesser og aktiviteter, ved bruk av process mining (PM) og maskinlæring (ML). PM brukes for å lage prosessmodeller, som gir oversikt over aktiviteter, og hvordan prosesser i dataene foregår. Prediksjonsmodeller i maskinlæring, konstrueres for å estimere den prehospitale tidsbruken, dette er tidsbruken forbundet med de ulike i aktivitetene tilknyttet ambulanseutrykninger. Tidsbruken for alle aktiviteter kalles for total gjennomstrømstid[5].

[1] Se Davenport (2017, s.101).
[2] Se Davenport (2017, s.83).
[3] Se Davenport (2017, s.121).
[4] Prosessorkraft vil doble seg hver attende mnd
[5] Er tilknyttet syv aktiviteter som utgjør gjennomstrømstid. Alle variablene vil bli belyst i resultater. Se variablene i **Vedlegg I3.** (POSIXct variabel kategori).

Første del av studien beskriver relevant bakgrunnsinformasjon og kontekst til oppgaven. Problemstillinger utledes. Andre del tar for seg relevant teori. Tredje del, metode, er praktisk bruk av teori. Fjerde del er resultatene fra metoden og i femte del vil resultatene diskuteres og konkluderes. Videre forskning og begrensninger fremmes i sjette del.

1.1 Bakgrunn

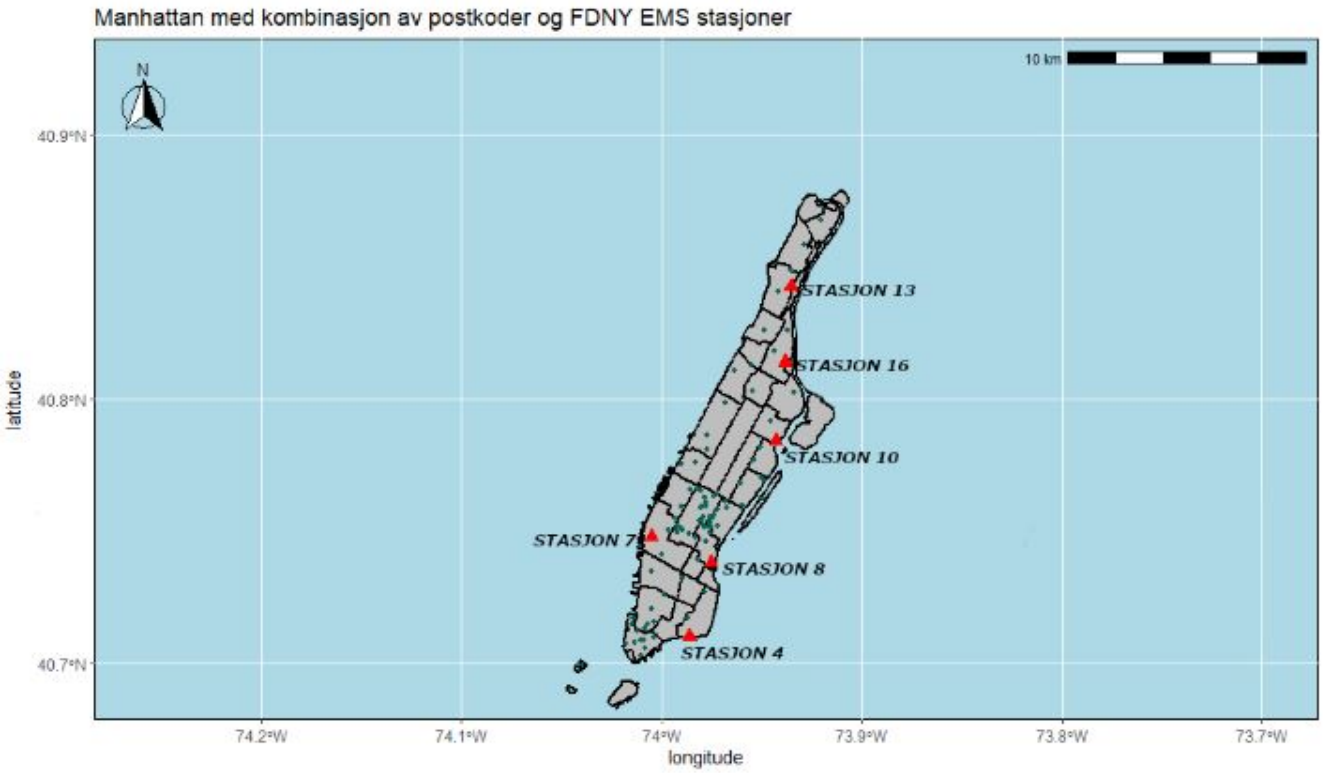
Prehospitalt medisinsk arbeid omhandler akuttmedisinsk tjenesteytelse før pasienten har ankommet akuttmottak. Slike prosesser foregår utenfor sykehus, hvor begrensede ressurser og liten tid spiller en stor rolle. Studiens fokus er delstaten New York, USA. Der oppgavens avgrensing er bydelen Manhattan. Sykdomsforløp avgrenses til hjertestans. Tidsperioden er fra 01.01.2008 til 31.12.2019.

1.1.1 New York City

Manhattan er en av de fem bydelene i metropolen New York City[6] (Nordlie, 2019). New York City er en av de byene i USA, med høyeste befolkningstetthet. Der bydelen Manhattan har høyeste befolkningstetthet (World Population Review, 2020). Det er estimert at i 2019, så rommer Manhattan en befolkning på 1,628706 millioner mennesker, hvor bydelen strekker seg over 59,13 kvadratkilometer. Dette gir en befolkningstetthet på 27 826 innbyggere per kvadratkilometer (United States Census Bureau, 2020).

New York City Fire Department Bureau of Emergency Medical Services (FDNY EMS) står for alt av operasjonell drift av systemet tilknyttet ambulanser i New york city. FDNY EMS er assosiert med, og drifter nødnummeret 911. Ulike oppdrag blir løst av ambulansepersonell og akuttmedisinske spesialister (FDNY EMS, u.å). FDNY EMS operer i snitt med 418 ambulanser daglig, hvor den totale flåten er på 608 ambulanseenheter (NYC Mayor's Office of Operations, 2020). Opptil 70 % av ambulansene er kommuneenheter tilknyttet FDNY EMS, hvor resterende 30 % er sykehusenes egne ambulanser. FDNY EMS strekker seg over seks divisjoner, hvor en divisjon er Manhattan. Innad i divisjonen for Manhattan er det totalt seks stasjoner (FDNY EMS, u.å).

[6] De resterende er Bronx, Brooklyn, Queens og Staten Island.



Figur 1.1: De ulike FDNY EMS Stasjonene i Manhattan. Figur produsert i R. Informasjon om Manhattan, Kilde: FDNY EMS (u.å.).

Kartet viser sammensetning av alle FDNY EMS stasjonene i Manhattan (Rød prikker) og de ulike lokasjonene for hvor ulykker har tatt plass (grønne prikker). De grønne prikkene viser hvor de ulike hendelsene har inntruffet på basis av postkodene[7].

1.1.2 Hastegrad og Prioritering

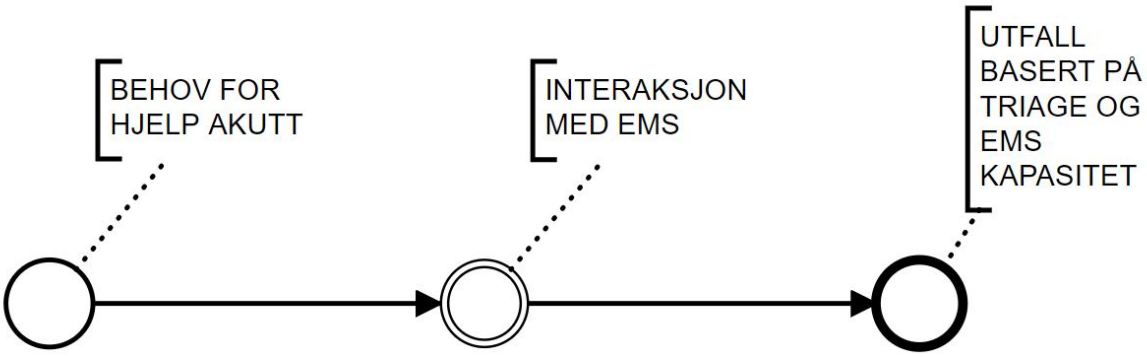
Fra Norges Legeforening tidsskrift (Engebretsen, Røise & Ribu, 2013, s.285) nevnes det at hastegrad og prioritering, er et konsept som har kommet fra det franske ordet triage. Triage omhandler prosesser tilknyttet utvelging, sortering, utvalg og utskilling, for å gi rett helsehjelp. Systemet har røtter tilbake til Napoleons krigene, hvor triage ble brukt som et inndelingssystem for allokering av medisinske ressurser til skadet infanteri. Inndelingene var; akutte hendelser, kan ikke behandles og kan vente. Det nevnes fra tidskriftet at triage fremdeles er aktuelt i dag, der det blir brukt i katastrofesituasjoner, militære hendelser og prehospital arbeid som akuttmottak (Engebretsen, et al., 2013, s.285).

FDNY EMS operer i dag med hastegrader (triage system) fra 1 til 8. Hvor 1-3 går under klassifiseringen livstruende medisinske nødsituasjoner og 4-8 er klassifisert som ikke livstruende medisinske nødsituasjoner. Hjertestans tilegnes livstruende medisinske nødsituasjoner (1 til 3) (NYC 911 reporting, 2020). I denne oppgaven reflekteres triage med variabelen(e) severity level code.

[7]Observasjoner er basert på postkoder, med hensyn på anonymisering av data og personvern.

1.1.3 Den Akuttmedisinske Kjede

> *"Den akuttmedisinske kjeden omfatter de tiltak og tjenester som er etablert for å yte nødvendig medisinsk hjelp ved akutt, behandlingstrengende sykdom."* (NOU 1998:9, s. 14)



Figur 1.2: Forenkelt Oversikt av den Akuttmedisinske Kjede.

Reduksjon i responstid[8] og gjennomstrømstid [9]vil kunne redusere antall dødsfall, tap av funksjonsnivå, lidelse og antall gjenværende leveår for pasienten. En av fire dødsfall i USA er relatert til hjertesykdom ifølge CDC (u.å.). Hjertesykdommer kostet USA 219 milliarder kroner fra 2015 til 2016. Denne summen innebærer kost fra medisiner, ulike helsetjenester, samt effekten av produktivitetstap (CDC, u.å.). Reduksjon av responstid og total gjennomstrømstid er begge drivere som påvirker sannsynligheten for om pasienten overlever (Helsedirektoratet, 2019).

> *"Overlevelse etter prehospital hjertestans avhenger blant annet av hvor raskt medisinsk redningspersonell når pasienten"*(Sunde, Fremstad, Furuheim & Steen, 2001, s.900).

For uventet hjertestans er det beregnet at for hvert minutt som går uten tilgang til hjertestarter, vil overlevelsesraten reduseres med 10 %. Dette er tilfellet når hjertestans oppstår utenfor sykehuset (Helsedirektoratet, 2019). Omtrent 535 000 hjertestans oppstod i USA i året 2015. Av disse 535 000 ble 61 % hendelser identifisert til å være utenfor sykehuset, mens resterende 29% var mens pasient var på sykehuset (Kronick et al, 2015). Hjertestans blir vanligere å få jo eldre man blir (NHLBI, u.å.). Flere mennesker har en gjennomsnittlig høyere forventet levealder enn tidligere, i USA. I 1880 var forventet levealder 39.4 år, hvor i 2019, var forventet alder på 82.4 år (Roser, Ortiz & Ritchie, 2013). Desto viktigere blir ressursoptimalisering i prehospital tjeneste, som følge av økt antall eldre mennesker, og at majoriteten av hjertestans i USA skjer på utsiden av sykehuset.

[8]Responstid omhandler tiden fra hendelsen er registrert i system til ambulansen er ankomet pasienten. Se **Vedlegg I** figur **I.2.**

[9]Denne variabelen kalles throughput time på engelsk se **Vedlegg I** figur **I.2.**

1.1.4 Vekst av ny Teknologi og Datavolum

Som følge av en nærliggende eksponentielle utviklingen av CPU kraft basert på moores law, har analyse av store mengder data blitt en mulighet (Davenport & Harris, 2017, s.8). Maskiner har mer prosessorkraft, og mengden enheter som strømmer data, også kalt Internet of things (IoT), har økt. Fra 15.41 milliarder enheter i 2015 til estimert 75.44 milliarder enheter i 2025 (Statista, 2016). Med rikelig mengde på data, vil teknologiske (statistiske) verktøy som PM og ML kunne hjelpe til med å høste informasjon i store datavolum. Det vil være av interesse å studere om ambulansetjenester kan få mer innsikt i virksomhetens prosesser[10]. I delkapittel 1.3 utledes problemstillinger for å utprøve disse metodene.

1.2 Hendelsesdata fra New York City

Data hentes fra databasen NYC Open Data (2020). Data loggføres av brannvern departementet (FDNY) (NYC OpenData, 2020). Datasettet er offentliggjort, og anonymisert. Dette er gjort som et tiltak for demokratiseringen av data. Gjennom datadeling kan offentlige tjenester forbedres, fordi allmennheten kan bidra med dataanalyse (NYC Opendata, 2019, s.5-8). Beskrivelse av de ulike variablene tilknyttet datasettet er å finne i **Vedlegg I** figur **I.2**.

> *"The EMS Incident Dispatch Data file contains data that is generated by the EMS Computer Aided Dispatch System. The data spans from the time the incident is created in the system to the time the incident is closed in the system. It covers information about the incident as it relates to the assignment of resources and the Fire Department's response to the emergency. To protect personal identifying information in accordance with the Health Insurance Portability and Accountability Act (HIPAA), specific locations of incidents are not included and have been aggregated to a higher level of detail"* (NYC Opendata, 2020).

1.2.1 NYC Statistikk

I datasettet finnes to kategorier med data, derav faktor og numeriske variabler[11]. Her presenteres relevante figurer og tabeller, som belyser overordnet informasjon tilknyttet hendelsesdataene.

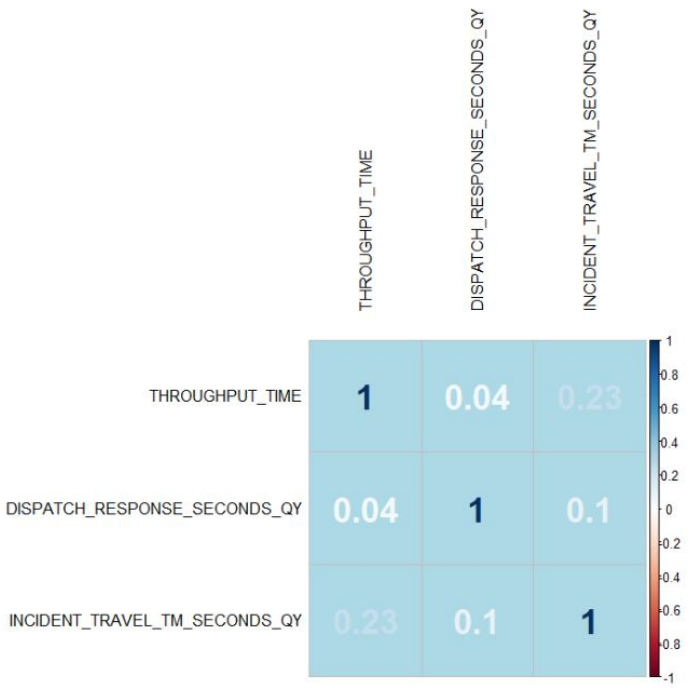
[10] PM & ML utledes i ytterligere i teori kapittelet.

[11] Datasettet som her er utgangspunkt for eksplorativ analyse, er basert på det ferdige datasett, som fremkommer i metodekapittelet. Dette heter i metode ABT - Analytisk Base Tabell, se figur **3.2**.

Korrelasjonsmatrise

Figur **1.3** viser korrelasjonskoeffisienten mellom de ulike numeriske variablene. Korrelasjonskoeffisienten belyser den lineære sammenhengen mellom to variabler. Korrelasjonskoeffisienten kan ligge i rommet mellom +1 til -1. En sterk positiv lineær sammenheng er +1, og -1 er sterk negativ lineær sammenheng. En korrelasjon på 0 belyser at det er ingen samvariasjon mellom variablene.

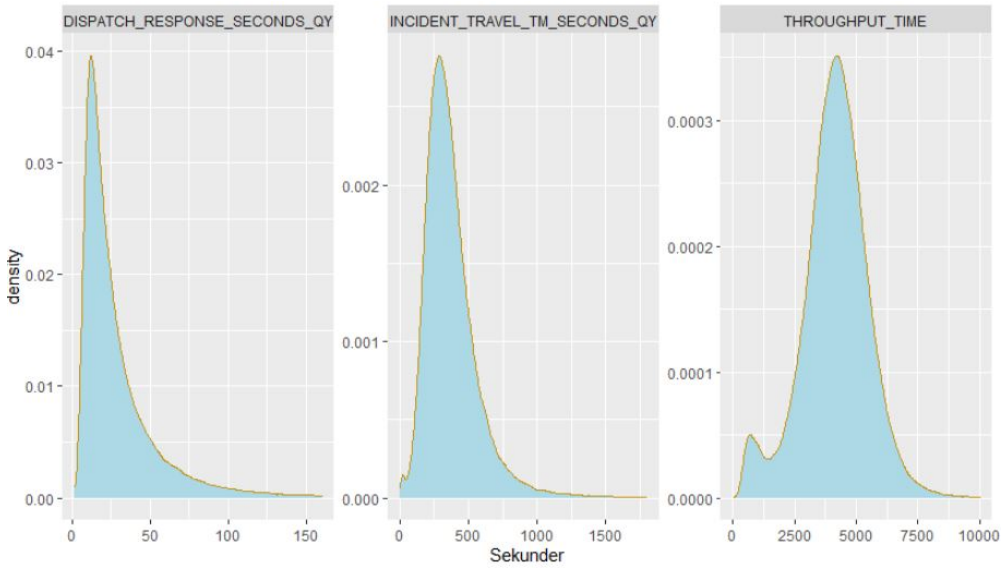
Fra figur **1.3** fremkommer det liten grad av korrelasjon. Det eneste måltall som kan peke i retning av en viss samvariasjon er mellom gjennomstrømstid (*THROUGHPUT_TIME*) og reisetiden til pasient (*INCIDENT_TRAVEL*), med 0.23. Reisetid er en mindre komponent av gjennomstrømstid. Responstiden (*DISPATCH_RESPONS*) korrelerer bare med 0.04 til den totale tid.



Figur 1.3: Korrelasjonsmatrise.

Distrubisjon

Density distribusjon for de tre numeriske variablene, er målt i sekunder. Her fremvises distribusjonen til de tre numeriske variablene i datasettet. Det er tendenser mot normalfordeling i noen av variablene, likevel med variasjon. En observerer først at *THROUGHPUT_TIME* utviser stor grad av normalfordeling, men i det nedre kvantil har en ujevnhet. INCIDENT_TRAVEL utviser grad av normalfordeling, men mindre enn *THROUGHPUT_TIME. DISPATCH_RESPONSE* utviser størst skjevfordeling av alle, med høyreside-skjevhet. Det er ingen antydninger til venstreside-skjevhet.



Figur 1.4: Density Distrubisjon Plott.

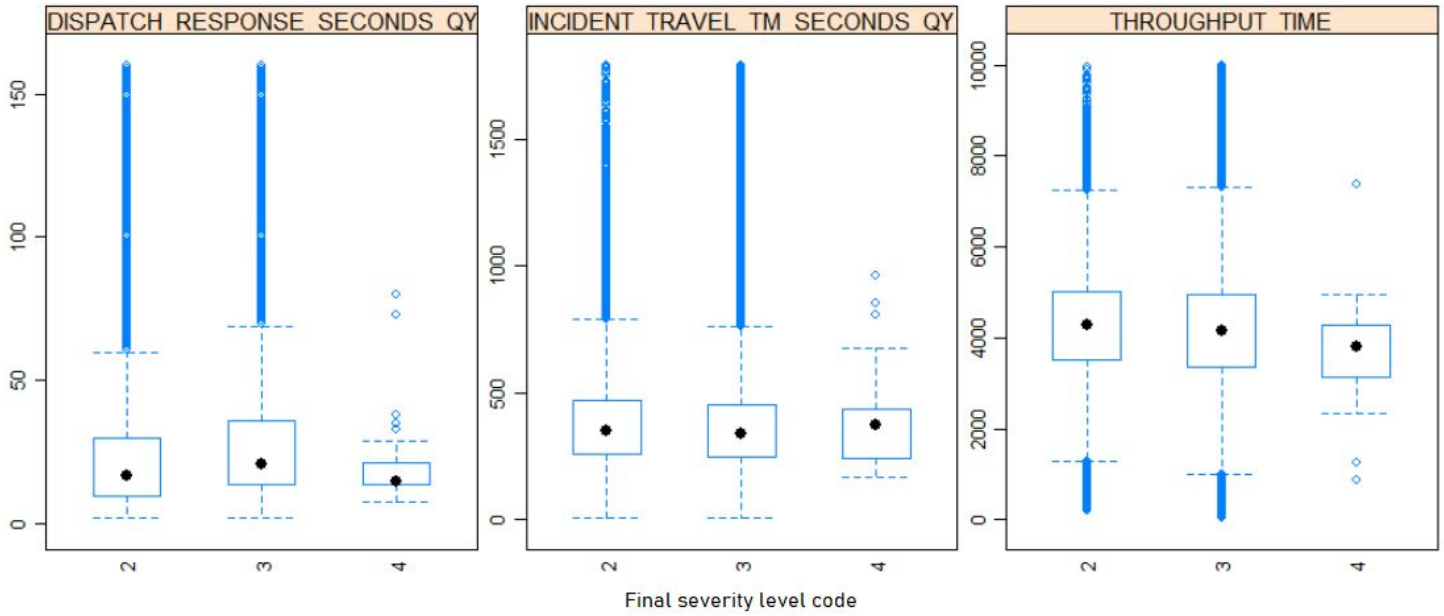
Boksplott - Alvorlighetsgrad

Boksplott har mye til felles med distribusjonene i figur **1.4**. Figur **1.5** viser i tillegg tidsbruken til de ulike numeriske tidsvariabler, differensiert på alvorlighetsgraden til én case. Av den endelige fastsatte alvorlighetsgraden (severity level), er kategori 1-3 livstruende. Kategori 4 og utover, er av mindre alvorlig sort (NYC 911 reporting, 2020). *DISPATCH_RESPONSE* variabelen viser størst spredning i kategori 2-3. Dette knyttes opp mot alvorlighetsgraden til casene, da de er relatert til hjertestans. I kategori 2-3 beveger 50%[12] av dataene seg i et mindre område. Det fremkommer at kategori 2 bruker relativt mindre tid enn kategori 3. Fra ytterkanten ved den horisontale stiplede linje, observeres det "outliers", som er representert ved blå sirkler. Disse dataene utgjør mindre enn 0,7%[13] av det totale datagrunnlag.

INCIDENT_TRAVEL bruker relativt lik tid i kategori 2-3. Dette skiller seg fra *DISPATCH*, der kategori 2 hadde lavere tidsbruk enn kategori 3. Kategori 4 har også her et mindre spenn i denne variabelen, enn kategori 2-3. I *INCIDENT_TRAVEL* beveger 50% av dataene seg i et mindre spenn, fra ca 300-480 sekunder. Variabelen *THROUGHPUT_TIME*, som er den totale tid, utviser i stor grad samstemthet i kategori 2-3. Det eneste som skiller seg ut, er at det nå eksisterer outliers under nederste horisontale linje. I likhet med de to andre variablene så beveger 50% av dataene seg i et mindre rom, der den normative tidsbruk er snaut over 4000 sekunder. *THROUGHPUT* har outliers både over og under normativ tid. Dette observeres også fra figur **1.4**, der den totale tid utviser en grad av normalfordeling.

[12]Kvadraten (50%) kalles for "interkvantilen", området mellom nedre (25) og øvre (75) kvantil. Kilde: Box Plot (u.å.)

[13]Se figur **7** i følgende artikkel. Kilde: Box Plot (u.å.).

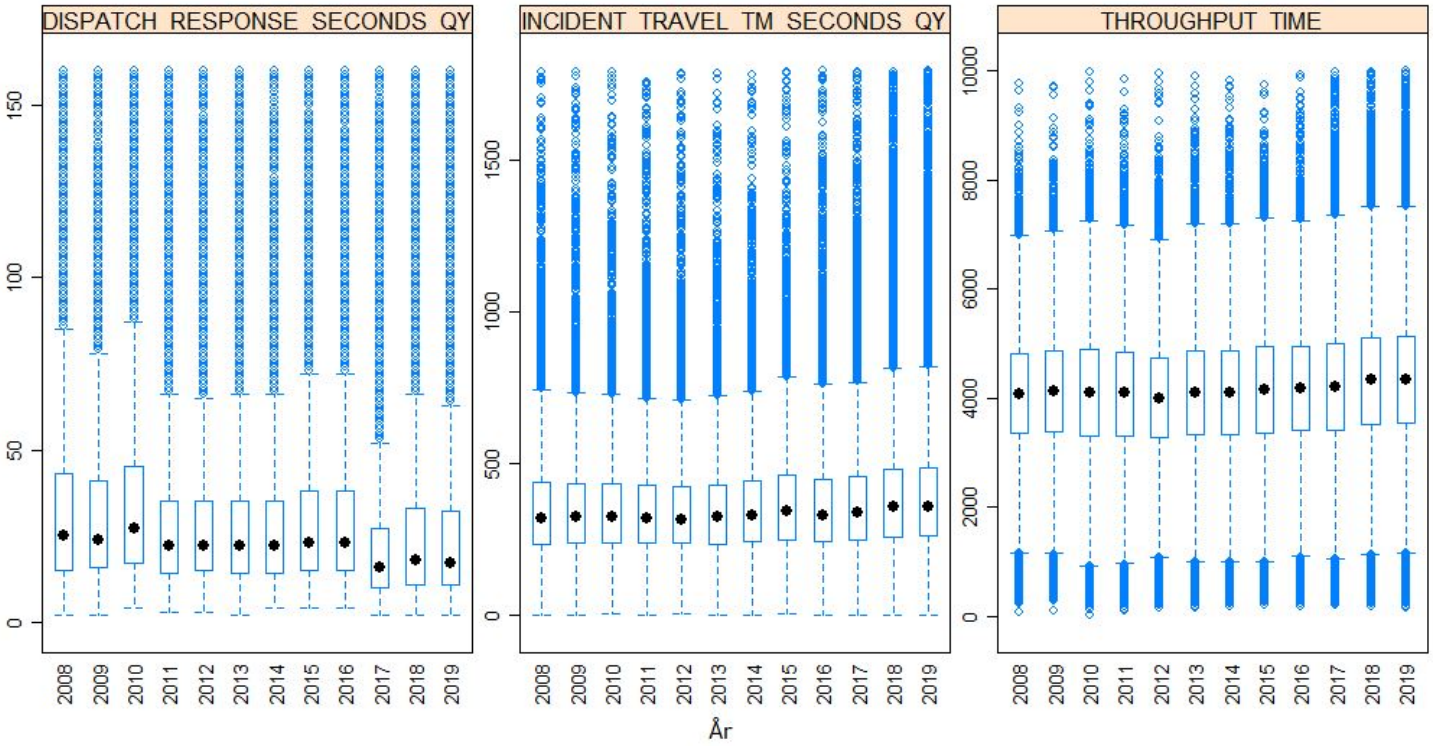


Figur 1.5: Hastegrad & Tidsbruk.

Boksplott - Utvikling over tid

Figur **1.6** viser til utvikling i tidsbruk over hele perioden for studien. I figuren er det et boksplott for de numeriske variabler, med år på x-aksen. Boksplottet kan vise effekter av f.eks, rutine- og arbeidsoppgave-endringer, til demografiske endringer. Fra *DISPATCH_RESPONSE* antydes det en nedadgående trend i outliers. Det samme gjelder interkvantilen. Tidsbruk i *INCIDENT_TRAVEL* er nokså jevn over hele dataens tidsperiode, uten noen sterke trender. Det som kan nevnes er en marginal trend (økning) for outliers og interkvantilen - fra 2015 til 2019. Den samme trend gjelder også for *THROUGHPUT_TIME*, der det indikeres en svak økning i tidsbruk. For *INCIDENT* og *THROUGHPUT*, indikeres det en konsolidering av outliers i perioden 2017 - 2019. *THROUGHPUTs* nedre outliers ser ut til å ha holdt seg stabil over hele perioden.

Da *DISPATCH* indikerer en svak trend ned, i tidsbruk, og de to andre variablene det motsatt, så er det relevant å vurdere disse funn i lys av korrelasjonsmatrisen. I figur **1.3** observeres det blant annet at *DISPATCH_RESPONSE* har tilnærmet ingen samvarians med den totale tid. *INCIDENT* hadde en noe høyere, endog svak samvarians, noe som potensielt kan være med på å forklare hvorfor *INCIDENT* og *THROUGHPUT* har en svak trend oppover. Dette betyr ikke at forholdet har kausalitet, men dataene viser likevel antydning til trender og mønstre.



Figur 1.6: Utvikling i Gjennomsnittlig Tid.

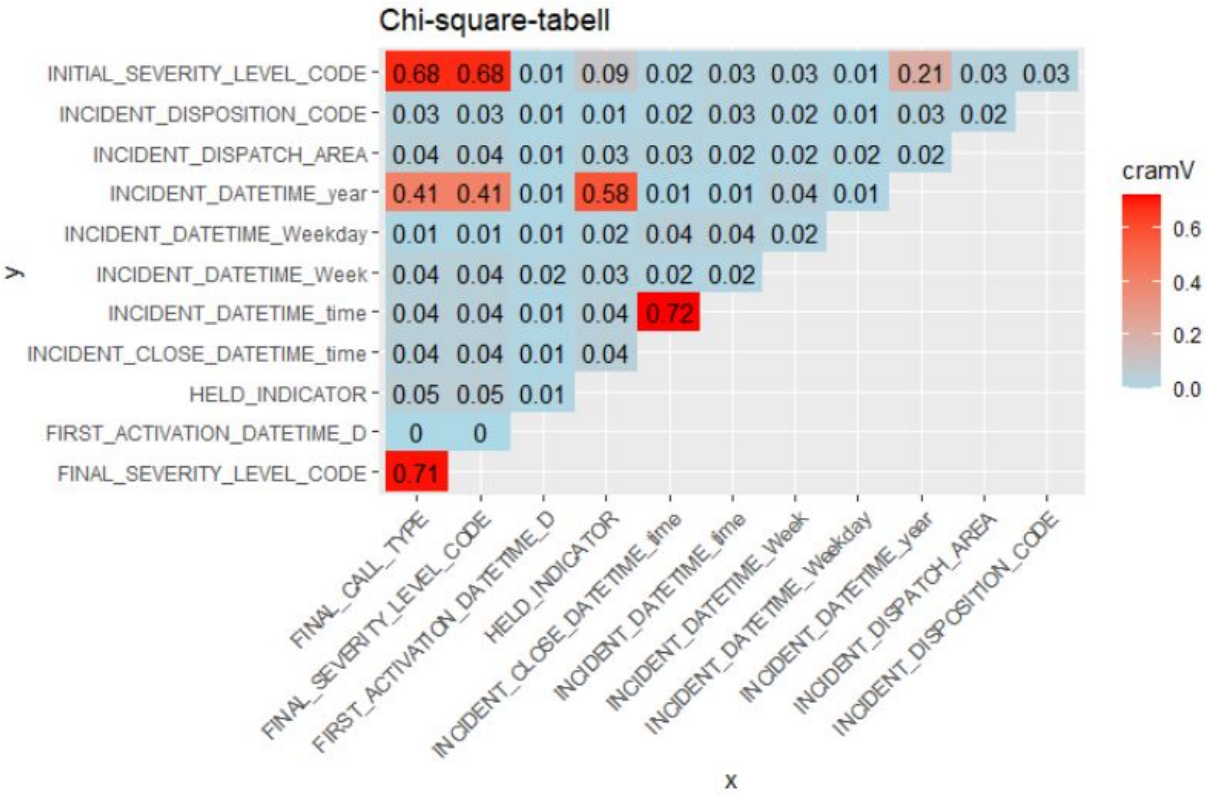
Faktor Variabler

Faktorvariabelene belyses her i Chi-square matrisen, figur **1.7** (Cramers-V, u.å).

Chi-square

Fra figur **1.7** så observeres en majoritet av variabler med lav samvariasjon[14]. Fra figuren fremkommer det i rødt de variabler som har sammenheng, der majoritet av variablene ikke har samvariasjon. De lysere farger som tenderer mot rødt indikerer en høyere grad samvariasjon, der 1 indikerer perfekt samvariasjon. Noen variabler skiller seg ut fra majoriteten. Blant annet variabel *INITIAL_SEVERITY* utviser stor samvariasjon med variabel *FINAL_SEVERITY* og *FINAL_CALL_TYPE*. Det samme gjelder *INCIDENT_DATETIME_time* og *INCIDENT_CLOSE_DATETIME_time*

[14]**Vedlegg I** og metode beskriver datavask. En av kriteriene lagt til grunn for å bruke en variabel var; ikke ha samvariasjon (Cramers-V) over 0,8.



Figur 1.7: Cramer-V test basert på Chi-square.

1.3 Eksisterende Litteratur

Her vil det presenteres relevant eksisterende litteratur knyttet opp mot PM og ML i kontekst av helsesektoren.

1.3.1 Process Mining i Helsesektoren

PM kan brukes til å uthente prosessinformasjon, analysere avvik, finne flaskehalser og komme med forslag til forbedringspotensial - relatert til aktiviteter og prosesser (Mans et al., 2015, s.5-6). PM kan gi informasjon om hvordan aktiviteter gjennomføres, hvilke personell som utfører ulike oppgaver, informasjon om tidsbruk og informasjonen om aktiviteters påvirkningsgrad på hverandre (Mans et al., 2015, s.8). Slik informasjon kan PM identifisere, da PM er en dataanalyse-metode, som særskilt ser på prosesser i data[15]. PM i helse-kontekst kan være interessant å studere. Eksempelvis, kan et scenario ha en case, der subjekt er en pasient. Første aktivitet er tilknyttet innleggelse, og den siste aktivitet er tilknyttet utskrivelse fra sykehus. Under hele sykehusoppholdet, vil det være mange aktiviteter som blir gjennomført, relatert til en pasient (se Figur **1.3** i Mans et al., 2015, s. 6). PM kan nyttiggjøres for å skaffe prosess informasjon fra "call centere",

[15]PM utledes i teorikapittelet.

som er relatert til akutt pre-hospital management (se Lamine, Fontanili, Mascolo & Pingaud, 2015). Ambulanse aktiviteter i den pre-hospitale tjeneste har også blitt studert i lys av PM, i asiatisk kontekst. Med den hensikt for å bedre pre-hospital tjeneste (se Badakhshan & Alibabaei, 2018). I Italia har PM blitt brukt til å predikere påfølgende aktiviteter tilknyttet en pasient, på basis av historisk data (se Duma & Aringhieri, 2020).

1.3.2 Maskinlæring og Process Mining

PM er en samlebetegnelse for ulike metoder som kan skaffe til veie kunnskap om prosesser, i organisasjoner (Mans et al., 2015, s.5). Ett aspekt ved PM er det som kalles operasjonellstøtte. I operasjonellstøtte vil man bruke data for å kunne bidra til å gi informasjon til organisasjonens daglige drift. Data som brukes er historiske data, som er loggført. Dette utgjør datagrunnlaget til ML, der en modell trenes for å estimere, eksempelvis tidsbruk. Relatert til sanntidsprediksjon, så har en organisasjon løpende inngående data, tilknyttet ufullstendige caser[16]. Aalst (2016, s.305) kaller disse ufullstendige caser for partial traces. Der betydningen er den, at en case med et gitt antall historisk aktiviteter, har et gitt antall fremtidige aktiviteter - som enda ikke har blitt loggført. Disse ufullstendige caser, innehar data om hva som hittil har hendt. Dette datagrunnlag vil da kunne bli brukt i operasjonellstøtte, for reelle sanntidsestimasjon (Aalst, 2016, s.305). For estimasjon nevner Aalst (2016, s.306) tre muligheter i operasjonellstøtte; deteksjon, prediksjon og anbefalinger. Et alternativ, ifølge Aalst (2016, s.304), til operasjonellstøtte er ren revisjon. Revisjon bruker historiske data, for å revidere prosesser opp mot en intern standard, eller benchmark.

Hvordan PM og ML kan brukes for dataanalyse, har blitt adressert tidligere. I et bidrag til den 51. CIRP konferansen[17], gjennomførte Lingitz et al., (2018) en studie relatert til prediksjon av ledetid, i en produksjonsbedrift. Den endelige modellen med lavest MAE og RMSE[18] var en random forest[19] (Lingitz et al., 2018, s.1054-1055). Dongen et al (2008) gjennomførte en studie basert på data fra en nederlandsk kommune. Hensikten var å predikere syklustiden det tok for å behandle en henvendelse fra innbyggerne, til kommunen. I studien ble det brukt parametrisk regresjon ved bruk av "lokale gjennomsnitt[20]", for å modellere eventuelle ikke lineære sammenhenger (Dongen et al., 2008, s.4). Funnene ble presentert ved å sammenligne MSE, der en parametrisk regresjon som brukte kombinerte estimatorer (variabler), utviste lavest grad av feil prediksjon (Dongen et al., 2008, s.15-16).

[16] En case er en hendelse i dataene. Eksempel ambulanseutrykning.
[17] Konferanse for produksjonssystemer.
[18] MAE og RMSE (MSE), er presisjons-måltall for hvor god en modell er til å predikere.
[19] Random forest er en maskinlæringsalgoritme.
[20] "Lokale gjennomsnitt" er en glattingsmetode for estimering av data (Dongen et al., 2008, s.4).

1.4 Forskningsdesign & Problemstilling

Dette er en retrospektiv studie, der intensjonen er å studere hendelsesdata (fra NYC) - fra tidsperioden 2008 til 2019. Hendelsesdata hentes fra den åpne databasen, *Open Data* (2019).

Gradsoppgaven har som mål å studere PM og ML, i kontekst av ambulansedata. Datagrunnlaget klargjøres for PM og ML, ved databearbeiding (datavask). Ulike aspekter i datavask er; eliminering av manglende verdier, konstruksjon av nye variabler, eliminasjon av outliers[21] og variabel transformasjon. Resultatet av alle de andvendte metodene med hensyn på datavask, danner grunnlaget for PM og ML. PM anvendes for produksjon av prosessmodeller, der resultatene i PM kan gi et analytisk grunnlag for ML. I ML vil prediksjonsmodeller for gjennomstrømstid bli produsert, der en endelig modell blir utvalgt, på basis av prediksjonskraft[22]. Den endelige beste modell vil bli analysert. Resultater blir diskutert i kontekst av problemstillingene. Avslutningsvis, i kapittel seks, vil vi komme med overordnede refleksjoner over hva prosessinformasjonen (fra resultatkapittelet) kan brukes til i ambulansetjenesten.

Problemstilling 1: Vil process mining kunne lage en god prosessmodell av NYC hendelsesdata?

For å adressere den første problemstillingen vil vi nyttiggjøre oss av alfa pluss algoritmen (Aalst, 2016, s.176). En PM algoritme [23], produserer en prosessmodell (Petri Net, som utledes i teori) fra hendelsesdata. For å prøve problemstillingen vil ulike statistiske måltall bli brukt. Det er primært fire måltall der modellens egnethet skal bli vurdert ut ifra. Det er fitness, simplicity, precision og generalization (Aalst, 2016, s.189). Disse måltallene tar for seg fire perspektiver, om hvorvidt en prosessmodell, er en egnet representasjon av dataene. Det er forstått at en avveining av alle måltallene er ønskelig (Aalst, 2016, s.189). En god prosessmodell er kontekstavhengig[24], fordi fire måltall skal balanseres.

Problemstilling 2: Vil process mining og maskinlæring kunne predikere gjennomstrømstid[25], i NYC hendelsesdata?

For å adressere den andre problemstillingen vil vi nyttiggjøre oss av flere metoder. På basis av eksisterende litteratur så vil det være av interesse og bruke PM og ML. Disse to metodene kan sees i lys av hverandre[26]. Strukturen i besvarelsen av problemet er sådan: først gjennomfører PM som utledet i metodekapitlet (problemstilling 1), for så

[21] Unormalt store verdier, eller ulogiske verdier (eksempel, -10 sekunder.
[22] Måltall blir ytterlige beskrevet i kapitel 3
[23] Alfa pluss utledes i teori.
[24] Dette er beviselig fra det ståsted at noen organisasjoner er mer komplekse enn andre. Sammenligne romfart med brødbaking.
[25] Total tid for en case.
[26] PM bygger på to pilarer, ifølge Aalst (2016, s.89), og disse er PM modeller og data mining.

å gjennomføre ML. Den primære funksjon PM har for ML, er å supplere informasjon til hvordan ML modellens resultat skal analyseres. Hendelsesdata er grunnlag for prediksjon av gjennomstrømstid til en case[27]. Der ML brukes for revisjon av historiske data, slik Aalst (2016, s.304)[28] beskrev rollen til *auditing* (revisjon) av data.

Hva som henvises til i problemstilling 2, med å "kunne predikere", er knyttet opp mot ML modellens presisjon. Modellens preisjon vil bli vurdert opp mot usett testdata[29]. Ulike statistiske måltall som MAE og RMSE vil bli nyttiggjort i evalueringen av modellens egnethet for prediksjon. MAE og RMSE er statistiske måltall som forklarer prediksjonevnen til en modell. Måltallene forklarer hvor god en modell er på å predikere usett data, som modellen ikke er trent og bygget på (Jank, 2011, s.140) . Måltallene kommuniserer forskjellen mellom observerte, historiske data og de predikerte data. Generelt kan det sies; hvor lavere MAE og RMSE er, hvor mer presis er modellen. Hvor mer presis modellen er, hvor mer informasjon kan høstes.

[27]En case innehar informasjonen om alle aktiviteter tilknyttet ett menneske som trenger akutt medisinsk hjelp. Der den totale tiden er summen av tidsbruken til alle aktiviteter, primært fra når noen ringer til EMS, og til når vedkommende er avlevert av ambulanse til akutten. I datasettet er variabelnavn for total tid, THROUGHPUT_TIME.

[28]I kilden henvises det til revisjon av data ved bruk av PM, vi vil gjøre det samme, men også ved bruk av ML.

[29]Data som er reservert for diagnose av ML modell.

Kapittel 2

Teori

Teori, fra det greske ordet theoria ($\Theta\varepsilon\omega\rho\iota\alpha$), har kommet til å bety i den vitenskapelige metode "vel attesterte antakelser om virkeligheten"[1]. I kapittelet utledes de rådende teorier, som vil understøtte hvilke metoder vi skal bruke for å besvare problemstillingene.

2.1 Process Mining

PM har til hensikt til å skaffe prosessinformasjon gjennom å analysere hendelseslogger med prosessdata (Aalst et al., 2012, s.12). PM finner sin plass mellom prosessvitenskap og datavitenskap (Aalst, 2016, s.18). Korrekt representert prosessinformasjon fra hendelseslogger ved bruk av PM og datavitenskap, vil kunne generere innsikt, tilknyttet organisasjoners operasjonelle prosesser. Det er flere bedrifter som nyttiggjøre seg av PM, blant annet ABB og Chemours, dette for dataanalyse hensikter (Spanyi & Davenport, 2019). PMs økte popularitet blant organisasjoner kommer fra PMs evne til å kunne representere aktiviteter og prosesser, på en god måte (Aalst, 2016, s.193). Alt er likevel ikke rosenrødt, feil implementering kan gi feil representasjon av virkeligheten, og eller promotere uheldige endringer av nåværende prosesser. Dette på basis av analyserte a-priori data (Aalst, 2016, s.448). Modeller er også utsatt for konseptdrift. Konseptdrift skjer når atferd endres ved oppdagelse av ny informasjon, fra PM (Aalst, 2016, s.450). Hvor den nye atferden blir fanget opp i ny fremtidig data, a posteriori. Gammel modell blir da utdatert grunnet ny atferd, og må eventuelt skrotes eller oppdateres (Zliobaite, Pechenizkiy & Gama, 2016, s.2). Fallgruvene tatt i betraktning, ved korrekt implementasjon, indikeres det fra litteraturen at PM vil kunne i ulik grad representere prosesser fra virkeligheten (Aalst, 2016, s.193).

2.1.1 Process Mining og Data Mining

PM er ansett å være en "missing link" mellom Business process management (BPM) og dataanalyse (Aalst et al., 2012, s.4). Der dataanalyse er å finne mønster og forhold internt

[1]Se artikklene, Theoria (2020) og Theory (2020).

i dataene, med den hensikt å kommunisere disse funnene (Aalst et al., 2012, s.4). Har BPM til hensikt å studere de interne prosesser og aktiviteter i en organisasjon (Aalst, 2016, s. 16). Når en kombinerer disse to, så fungerer PM som en datadrevet metode for å analysere prosesser.

2.1.2 Data og Hendelseslogger

PM nyttiggjør seg av hendelsesdata. Slik data er et strukturert i hendelseslogger. Fra den teorien, indikeres det at en hendelseslogg skal fremheve fire perspektiver; organisasjon-, case-, tid- og aktivitetsperspektiv (Aalst et al., 2012, s.4). Disse perspektivene belyses i form av variabler. Organisasjonsperspektiv er med på å belyse hvilke ressurser som er tilknyttet en prosess, tidsperspektiv belyser tidsbruk ved aktiviteter, caseperspektiv sier hva som skal analyseres og aktivitetsperspektiv tar for seg hyppighet, sortering og modellering av aktiviteter. Ifølge Aalst et al (2012, s.5), er disse perspektiver med på å belyse de ulike elementene ved en prosess.

PM som en disiplin har kravspesifikasjoner til data og dataenes struktur. For at data skal kunne være kompatibel for PM, må dataene representeres i en hendelseslogg. Algoritmene som blir brukt til å generere prosess informasjon, er avhengig av kompatibel loggdata. Den enkleste hendelseslogg har tre attributter, derav en variabel for aktiviteter, tid da aktivitet fant sted, og en unik case ID. Disse variablene er minimumskrav til en hendelseslogg. I tillegg kan en ha flere variabler (utover disse tre), f.eks kostnad (tilknyttet aktivitet), bruker ID (gjøreren av aktivitet), samt flere andre variabler av ulik natur[2] (Aalst, u.å, s.9). Hendelsesloggens minstekrav er ment for å representere kjernen av alle prosesser tilknyttet aktiviteter. Dette gir en grov simplifisering av virkeligheten.

En logg kan eksempelvis sees slik ut. Hendelsesloggen = L.

$$L = [(a, b, y)., (a, z, y).^3, (a, x, y).]. \tag{2.1}$$

Loggen har start aktivitet a, og henholdsvis tre alternative aktiviteter, før den siste avsluttende aktivitet y. I denne hendelsesloggen er det tre traces[3], der to av dem har info om en enkelt case. Trace (a,z,y)[3] har info om tre caser, dvs at tre caser har fulgt samme vei (Aalst, 2016, s.171).

2.1.3 Process Mining Variabler

Case ID

Denne variabelen er unik for en unik case. I en hendelseslogg, vil en case ID kunne fremkomme flere ganger i datasettet. Vis så, indikerer det at en case innehar flere ak-

[2]Denne listen er ikke utelukkende, da det ikke bør være et problem å kunne tilpasse en hendelseslogg, gitt spesifikke behov.

[3]Aktivitetskombinasjoner tilknyttet en case.

tiviteter. En case kan også ha gjentagelser av de samme aktiviteter som tidligere hadde truffet sted (Aalst, 2016, s.129-132). En case kan også pågå samtidig med en annen case.

Tid

Tidsvariabelen fanger opp tiden når en aktivitet finner sted. For økt validitet har de fleste hendelseslogger loggført helt ned på minutt nivå, eller lavere (Aalst, 2016, s.129). Total tidsbruk for alle aktivitetene tilknyttet en case er gjennomstrømtiden.

Aktivitet

Pasient	Aktivitet	Timestamp(tid).	Satus	Aktivitets-instans	Ressurs
Ola Nordmann	Registrering	2018-08-08-10:00	Ferdig	1	Åse
Ola Nordmann	Utkjøring	2018-08-08-11:00	Ferdig	2	Mathilde
Ola Nordmann	Hentes Ambulanse	2018-08-08-12:00	Ferdig	3	Kari
Ola Nordmann	Transport	2018-08-08-13:00	Ferdig	4	Pål
Ola Nordmann	Fremme Sykehus	2018-08-08-14:00	Ferdig	5	Per
Ola Nordmann	Avlevert	2018-08-08-15:00	Ferdig	6	Reodor
Ola Nordmann	Case Ferdig	2018-08-08-16:00	Ferdig	7	Jonathan

Tabell 2.1: Hendelseslogg. Kilde: Tilpasset fra Aalst (2016, s.129).

En hendelseslogg består av x antall aktiviteter. Om en har syv mulige aktiviteter en case kan være utsatt for, så betyr det 7! = 5040 mulige aktivitetskombinasjoner[4] for en case gitt n aktiviteter. Det er likevel liten sannsynlighet for at alle disse scenarioene noensinne vil se dagens lys (Aalst et al., 2012, s.12). Sortering av aktivitetenes rekkefølge i en case er kontekstbetinget.

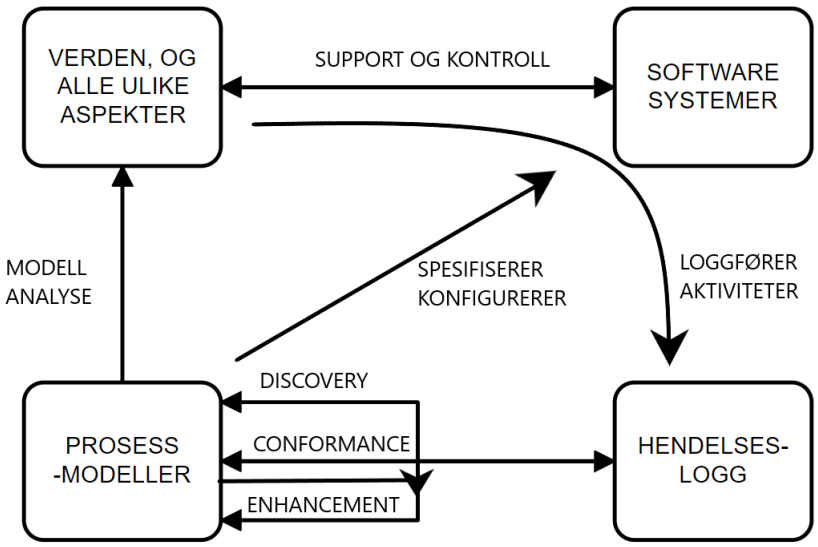
Supplerende Variabler

Utover de nødvendige variabler, så er det flere andre variabler som kan brukes for å supplere beskrivelsen av prosesser. Primært kostnad-, sortering-, transaksjonsinfo- og ressursvariabel (Aalst, 2016, s.152). Kostnad (eller inntekt) omhandler en pengesum tilknyttet en aktivitet, dette kan eksempelvis være verdiskapning, eller lønn. Sorteringsvariabel søker og sorterere aktivitetenes rekkefølge i en unik case, dette er hensiktsmessig da flere caser kan overlappe hverandre og en unik sorterings variabel kan være med på å skille mellom ulike caser[5]. Resursvariabel indikerer hvilke ressurs som har blitt brukt til å gjennomføre en aktivitet. Der variabelen kan ha ulike nivåer, for eksempel individ-/gruppenivå.

2.1.4 Hovedmetodene i Process Mining

I PM er det tre hovedmetoder for dataanalyse: Discovery, conformance og enhancement (Aalst, 2016, s.276).

[4]Traces i PM terminologi.
[5]Dette er aktivitets-instans variabelen i Tabell **2.1**.



Figur 2.1: De Ulike Metodene i Process Mining. Kilde: Tilpasset fra Aalst (2016, s.276).

Process Discovery

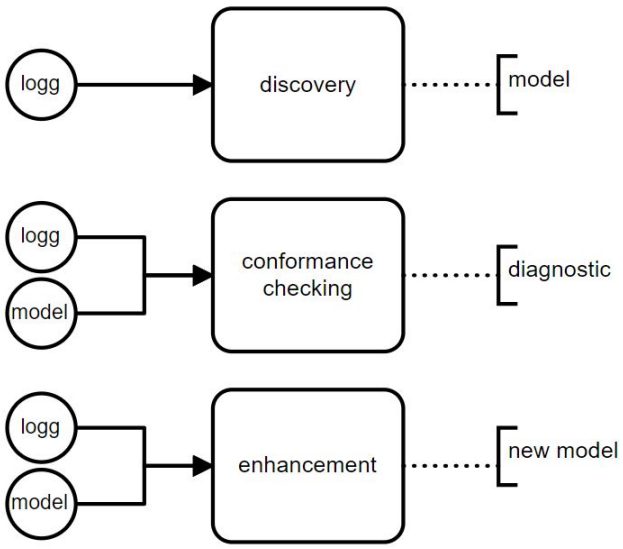
Process discovery er en kombinasjon av data utforskning (discovery), samt control-flow perspektivet (Aalst, 2016, s.163). Control-flow perspektivet fokusere på hvordan aktiviteter oppstår i caser. Målet med et slikt perspektiv er å identifisere alle mulige retninger en case kan ta (Aalst et al., 2012, s.4).

> "The control-flow description is the backbone of any process model." (Aalst et al., 2012, s.8).

En process discovery algoritme lager en prosessmodell. En slik algoritme er en funksjon som søker å lage en modell fra en hendelseslogg. Formålet er å skape en modell som representerer atferden observert i hendelsesloggen (Aalst, 2016, s.164). Process discovery er den viktigste grenen innad PM, da den skaper fundamentet for videre analyse (Aalst et al., 2012, s.4).

> *"Noise and incompleteness make process discovery a challenging problem."* (Aalst et al., 2012, s.12).

Prosessmodeller kommer i mange ulike formater, etter hva en ønsker å oppnå. Slike prosessmodeller kan være i form av blant annet business process model notation (BPMN). eller Petri nets (Aalst, 2016, s.163). En av de største utfordringene i process discovery fasen, er å finne hvilke process discovery algoritme man skal ta i bruk. Dette for å skape en mest mulig representativ prosessmodell (Aalst, 2016, s.163). Der en representabel modell skal inkludere relevant atferd fra hendelsesloggen (Aalst, 2016, s.166).



Figur 2.2: De Tre Metodene i Process Mining og Deres Output. Kilde: Tilpasset fra Aalst (2012, s.4).

Conformance Checking

Conformance checking innebærer å analysere om hendelsesloggen er i samsvar med prosessmodellen. Det er en to-veis prosess, som innebærer at man ikke bare ser om hendelsesloggen samsvarer med modellen, men også om modellen samsvarer med hendelsesloggen (Aalst et al., 2012, s.14).

Conformance checking avhenger av forholdet mellom en hendelseslogg og modell. Hensikten er å simulere[6] hendelsesloggen, oppå prosessmodellen. Conformance kan identifisere avvik og fellestrekk, mellom hendelsesloggen og prosessmodellen. Dette kan f.eks være tilfellet hvor atferd observert i logg, ikke blir reflektert i process modellen (Aalst et al., 2012, s.8-9). Resultat fra conformance checking gir diagnostiske måltall. Der det er primært fire ulike diagnose kriterier, som søker å adressere prosessmodellen og loggen. Disse er "fitness", "precision", "generalization" og "simplicity" (Aalst, 2016, s.166)[7]. Det er ønskelig å balansere disse fire. Der målet er at en algoritme skal produsere en prosessmodell (eksempelvis et Petri net[8]), på basis av loggen. For å regne ut måltall, brukes prosessmodellen for simulasjon av loggen. Dataene "kjøres" gjennom modellen. Dette kalles "token-replay/-game", der en "token" (token er data fra loggen) blir simulert i modellen (Aalst, 2016, s.243-246). Hvis det eksisterer en trace (aktivitetskombinasjon), som uproblematisk "kommer" seg gjennom modellen, så påvirker det måltallet "positivt" (eksempelvis økt fitness). Hvis en trace ikke klarer å komme seg igjennom prosessmod-

[6] I litteraturen kalt replay.

[7] For ytterligere diskusjon og utledning av måltall, kan følgende kilder være nyttige; Aalst (2016) kapittel åtte, Evaluation Log-Model (u.å.) og Rozinat (2008). Måltall blir også tatt opp i metode.

[8] Petri net blir utforsket i seksjonen om alfa pluss. Oppkalt etter Carl A. Petri.

ellen, betyr det at simuleringen vil kunne "negativt" påvirke måltallene[9].

PM som et fagfelt er under utvikling[10]. Dette observeres blant annet fra følgende sitater, som taler om at det er fortsatt mye arbeid som gjenstår, knyttet til evalueringen av prosessmodeller.

> *"In the domain of process mining the evaluation of models (i.e., "How can we measure the quality of a mined process model?"). is still subject to ongoing research..."* (Rozinat, Veloso & Aalst, 2008, s.1).

Der målet er:

> *"The ultimate vision for process model evaluation would then be to have a methodology that assists in selecting the "right" metric for the "right" situation, and based on the goal of the evaluation"* (Rozinat et al, 2008, s.7).

Model Enhancement

Model Enhancement (modell forbedring) er den siste av de tre grenene innad PM. Fra tidligere om conformance checking hvor formålet var å måle avviket mellom modell og hendelseslogg, søker enhancement å forbedre prosessmodellen (Aalst et al., 2012, s.4). Som vist i figur **2.2** er input fremdeles en hendelseslogg og modell, men output i form av en ny modell(Aalst et al., 2012, s.4). Prosessmodellen kan repareres via output fra conformance checking "diagnostic" (Aalst, 2016, s.243). Hvis fitness er lav, så betyr det at få traces er tatt med i modellen, en løsning da er å manuelt legge ved flere alternativer for traces, i én prosessmodell.

2.1.5 Prosessmodellen

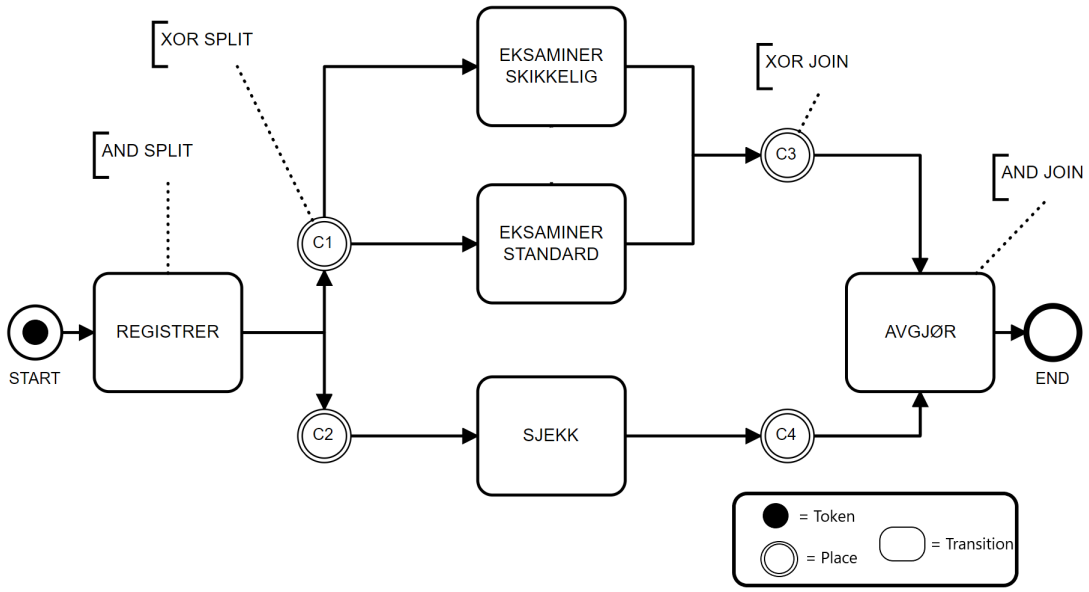
En algoritme produserer en grafisk prosessmodell (Aalst, 2016, s.55), der en søker å plotte hendelsesloggen i en modell. Noen av de mer kjente modellene som en gitt algoritme kan produsere er; Petri net, simpel net, casual net, workflow net, fuzzy og BPMN[11]. Notasjon er prosessmodellens eget språk. Ulike modeller har ulike symboler (språk), som er designet for å kommunisere ulike konsepter og scenarioer. Eksempelvis; BPMN er en modell som søker og fremheve bedriftsorienterte aspekter ved prosesser (Aalst, 2016, s.56). Fuzzy

[9]Det som menes med, "ikke klarer å komme seg igjennom prosess modellen", betyr at en case i loggen, ikke er representert i prosessmodellen. Dette resulterer i at en simulering, så vil denne casen ikke klare og komme seg fra "A til Å". Eksempel: Logg har en case med 3 aktiviteter. En modell blir konstruert av logg data. Den konstruerte modell har bare 2 aktiviteter. Den nevnte case blir ikke representert.

[10]Dette innebærer blant annet ulike meninger om måltallenes egnethet. Dette tas opp i følgende kilde (Evaluation Log-Model, u.å).

[11]Se diskusjon i Aalst (2016, kapittel 6).

model søker å sekvensielt vise, eksplisitt, bare aktivitetene i en logg (Fluxicon, u.å, s.3). Korrekt valg av algoritme påvirker presisjon og resultat (Aalst, 2016, s. 195).



Figur 2.3: Petri Net og Dens Notasjoner. Kilde: Tilpasset fra Aalst (2016, s.60).

Notasjon i Prosessmodellen

Et eksempel på notasjon finner en i prosessmodellen Petri net. Dette eksemplifiseres her med alfa algoritmen[12]. Alfa algoritmen er en relativt enkel og mye brukt prosess mining algoritme, da den klarer å håndtere hendelseslogg data med aktiviteter som skjer samtidig om hverandre (Aalst, 2016, s.167). Alfa algoritmen produserer en grafisk representasjon, en prosessmodell. Denne modellen kalles også for Petri net. Petri net har en spesifikk notasjon[13]. Notasjonen er de ulike momenter som en algoritme søker å grafisk vise i en prosessmodell, ved bruk av symboler.

I notasjonen eksisterer AND (split, join) og XOR (split, join) (Aalst, 2016, s.60). Fra figur **2.3** fremkommer et XOR veiskille (split), der en case kan velge mellom alternative aktiviteter. Det samme gjelder for innsnevring (join). For AND observeres et veiskille (split), der en prosess må gjennom begge veiene, og må møtes på AND destinasjon (join). AND tilknyttes transition (kvadratene), der en observerer piler til eller fra kvadraten. XOR tilknyttes places (sirklene) på lik linje som AND, der XOR har piler til eller fra sirkelen[14] (Aalst, 2016, s.60).

[12] Alfa algoritmen utledes i metode.
[13] Se XOR, AND i figur **2.3** for eksempel på notasjon.
[14] Token fra firguren kan leses om i Petri Net Management (u.å.). Token er en utregningsmessig nødvendighet i et petri net, der tokens er nødvendig for fyringsekvensene relatert til traces. Det menes da med hvordan caser (fra loggen) beveger seg i modellen. Tokens vil ikke være gjenstand for videre diskusjon. Places og transition vil være i fokus for analyse og diskusjon i **kapittel 4.**

2.2 Maskinlæring

> *"While artificial intelligence (AI) is the broad science of mimicking human abilities, machine learning is a specific subset of AI that trains a machine how to learn."* (SAS, 2020).

ML og kunstig intelligens er ord som ofte brukes synonymt, men forskjellene ligger i detaljene. Kunstig intelligens (AI eller KI) omfavner alle momenter som innebærer hvordan en kan imitere menneskelige egenskaper, der maskinlæring er en gren av kunstig intelligens (SAS, 2020). ML dreier seg om bruk av algoritmer, som gjør det mulig for en datamaskin å lære. Maskinen gjør så avgjørelser basert på hva den har lært, uten form for eksplisitt programmering (Aalst, 2016, s.13). Der tradisjonell koding søker et programmert ønsket output, er formålet i ML å produsere en ML-modell.

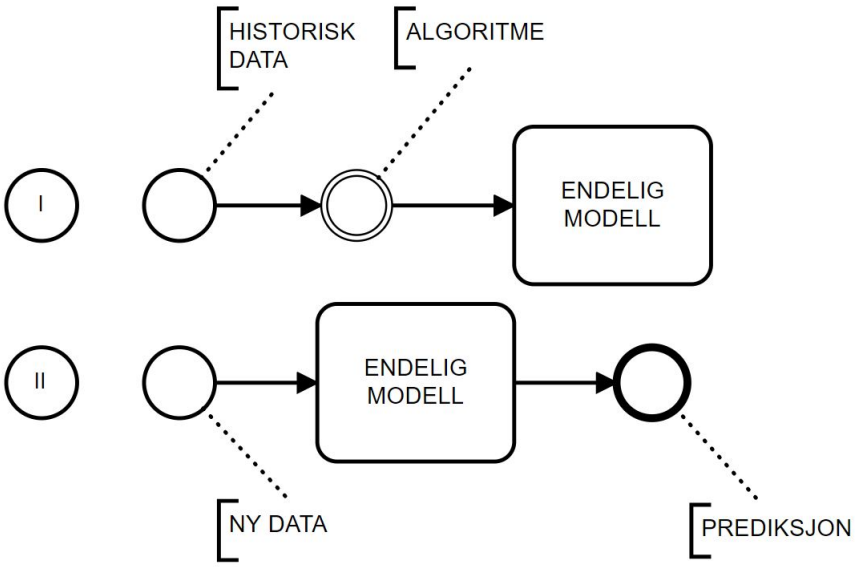
ML kan deles inn i flere overordnede metoder for hvordan maskinen skal lære. Disse metodene er supervised learning, unsupervised learning, reinforcement learning og semi-supervised learning (Krzyk, 2018). Fokus i denne oppgaven vil ligge i supervised learning, der resterende teori vil ha fokus på denne grenen av maskinlæring. Det alle metodene har til felles er en underliggende læringsalgoritme (Aalst, 2016, s.13).

2.2.1 Maskinlæringsmodell

En ML-modell har egenskapen til å predikere verdier fra nye observasjoner (Hastie, Tibshirani & Friedman, 2017, S.2). Modellen er resultatet av læringsalgoritmen, som er anvendt (Aalst, 2016, S.13). Enhver modell er unik tilpasset datagrunnlaget, der algoritmen er det teoretiske rammeverket for hvordan en modell skal produseres (Krzyk, 2018). En ML-modell har en avhengig variabel, som skal predikeres[15]. De uavhengige variablene er grunnlag for trening av modellen, dette for å predikere den avhengige variabel.

ML-modellen innehar modellparametere og hyperparametere. Modellparametere, er parametere som læres direkte av datasettet (ved en læringsalgoritme). Et eksempel er koeffisienter i lineær regresjon. Hyperparametere skiller seg ut ved at de ikke er lært direkte av datasettet. Disse er spesifisert før modellen blir trent. Hyperparametere reflekterer algoritmens interne struktur, ved å spesifisere hvordan ML-modellen skal konstrueres. Eksempel på hyperparametere er lambda (λ) i ridge og lasso regresjon[16]. Lambda har til hensikt å straffe/redusere koeffisienter, før koeffisientene (modellparametere) er endelig konstruert (Hastie et al., 2017, s.61-63). Ved hjelp av hyperparametere, kan man forbedre algoritmens presisjon, og derav få modellparametere som minimerer feil prediksjoner.

[15] I denne oppgaven er den avhengige variabel, gjennomstrømstid.
[16] Ridge og lasso brukes senere i oppgaven, for prediksjon.



Figur 2.4: Representasjon av en Prediktiv Modell.

2.2.2 Supervised Learning

> *"For each example the goal is to use the inputs to predict the values of the outputs. This exercise is called supervised learning."* (Hastie et al., 2017, s.17).

I supervised læring antas den avhengige variabelens utfall å være definert for alle observasjoner[17]. Det er med andre ord spesifisert utfallet av hva prediksjonen kan være (Aalst, 2016, s.92). En slik prediksjon kan gjennomføres ved å gi læringsalgoritmen X variabler (Hastie et al., 2017, s.18). Supervised læring brukes ofte når en har en antagelse om at fenomener, kan forklares på basis av historisk data (SAS, 2020). Supervised læring kan deles opp i to kategorier. Den første er å predikere en kontinuerlig variabel (regresjon), noe som er fokus i denne studien. Den andre er klassifisering av utfall (klassifisering), eksempelvis salg/ikke salg, svindel/ikke svindel. Den endelige modellen kan predikere utfallet på ny ukjent data, for å komme med prediksjoner. Dette fremkommer fra figur **2.4**

2.2.3 Bias og Varians

Målet med supervised maskinlæring er å predikere fremtiden (Jank, 2011, s.126). På historiske data, kan ML modellen forklares av R kvadrat (R^2). R kvadrat gir et prosentvis måltall, som forklarer forklart varians i modellen (Jank, 2011, s.127). På historisk data gir dette innsikt i modellens prediksjonsevne, men i prediksjon av fremtidige usett data så er dette ikke tilstrekkelig. Måltallet fanger ikke opp effekten av prediksjon på fremtidig

[17]Derav navnet supervised læring, ettersom utfallsvariabelen er definert av menneskelig veilder (Krzyk, 2018).

data, grunnet at måltallet er basert på historisk data. For å takle dette problemet finner vi i teorien, konseptet om treningsdata og testdata. Disse data brukes for å vurdere bias/varians og generaliserbarhet (Hastie et al., 2017, s.222-223 og Jank, 2011, s.134).

For å kunne gjøre meningsfylte vurderinger om hvorvidt én modell vil kunne predikere på fremtidig data, vil det være hensiktsmessig å inndele historisk data i treningsdata og testdata. Dette gjøres for å begrunne om hvorvidt modellen har generaliserbarhet (Hastie et al., 2017, S.219). Treningsdata er datagrunnlaget for hva modellen er konstruert på. Testdata, er data som er reservert for diagnostisk om prediksjonsevnen. Hvis testdata utviser god evne for prediksjon, vil en kunne ekstrapolere at modellen evner å predikere på usett fremtidig data (Jank, 2011, s.126). Blant de egnede måltall for å vurdere evne til prediksjon, eksisterer MAE og RMSE (Jank, 2011, s.140), som respektivt måler feilledd i absolutte verdier (MAE), og kvadratroten av kvadrerte feilledd (RMSE).

Bias og varians er uønskede fenomener i modellen, og gir upresis prediksjon på testdata. Mer presist så er høy bias, det som gir underfitting, og høy varians gir overfitting. Begge fenomener i modellen gir uegnet generaliserbarhet (Hastie et al., 2017, s.225-226). En modell med høy bias vil i lav grad fange opp mønstrene i treningsdata. Dette skyldes primært mangler på variabler (utelatelsesbias). Dette leder til lav varians (Hastie et al., 2017, s.223). En annen årsak for høy bias er også valg av en algoritme som ikke tilstrekkelig adresserer mønstrene i dataene. Det reverserte gjelder for høy varians, da vil det være lav bias. Dette fordi modellen klarer å fange opp mye av mønstrene i treningsdataene. Likevel, høy varians vil være problematisk da slike forekomster utviser såkalt overfitting, noe som leder til lav generaliserbarhet på testdata. Hovedårsaken til dette fenomen er ofte tilknyttet inkluderingsbias, da en har inkludert problematiske variabler[18], eller valgt en uegnet algoritme som har overfitting (Hastie et al., 2017, s.223).

Den gode modell er den som klarer å balansere dette tveeggede sverd, der en søker å minimere bias og varians[19] og derav kunne generalisere på usett data. En modell som klarer å finne dette vinningsoptimum, mellom bias og varians, har da bare gjenstående ufravikelig støy i modellen. Dette er støy som ikke kan fjernes gitt dataenes egenskaper, eller mangelen derav (Hastie et al., 2017, s.223).

$$Totalfeil = Bias^2 + Varians + UfravikeligStøy \tag{2.2}$$

[18]Problematiske variabler, kan f.eks være varialer som har høy korrelasjon med hverandre.

[19]Bias og variance er støy man kan fjerne som kalles fjernbar støy (reducible), men støy vi ikke kan gjøre noe med, som følge av mangler på uavhengige variabler kalles ufravikelig støy (irreducible).

Kapittel 3

Metode

I dette kapittelet blir metode for PM og ML presentert. I PM utledes alfa pluss algoritmen, som skal brukes i besvarelsen av problemstilling 1. I ML utledes fire algoritmer som skal brukes i besvarelsen av problemstilling 2.

3.1 Data

Data er hentet fra den åpne databasen, NYC OpenData (2020). Navnet på datasettet er "EMS Incident Dispatch Data" (NYC Opendata, 2020). Dette datasettet inneholder utrykninger i ambulansetjenesten, tilknyttet NYC. For å kunne utføre PM og ML bruker vi det statistiske programmet R, med relevante tilleggspakker[1].

Den totale databasen inneholder 16.4 millioner observasjoner og 31 variabler[2]. Observasjonene i dataene strekker seg over perioden 1/1 2008 til 31/12 2019. Dette er alle observasjonene tilgjengelig fra databasen, tidspunktet da oppgaven ble skrevet. Oppgavens avgrensning strekker seg til kardiologiske diagnoser (hjertestans) og utrykninger tilknyttet bydelen Manhattan[3]. Dette uttrekket sikrer at kun får hjertestans og ambulansereiser innad Manhattan.

3.1.1 Preprosessering før ML og PM

For å arbeide med ML og PM kreves korrekt inndeling av variabler med ulike datatyper (f.eks. variabler som kategoriske og kontinuerlige). NYC OpenData databasen klassifiserer ikke variablenes datatyper. Det må dermed klassifiseres[4] ved innlasting til R. Inndelingene for hvilke datatyper disse variablene har blitt spesifisert med, er belyst i **Vedlegg I.3**

I preprosessering av data er det tatt særskilt hensyn til det faktum at det er hendelsesdata

[1] R versjonene som brukes er 3.5.3/3.6. En pakke er et objekt som kan lastes ned til R for eks. statistisk analyse. Pakker lastes ned fra CRAN, som er utgiver av R (CRAN, u.å.).

[2] Se **Vedlegg I** figur **I.2** for variabel beskrivelse.

[3] Fra databasen spesifiserte vi vårt uttrykk med tre kriteria (se **Vedlegg I** figur **I.1** Utdrag database).

[4] Datavask vil bli brukt på datasettet etter variablene er klassifisert.

fra helsesektoren. Slike data bør behandles med stor respekt. Metodkapitlet søker å ivareta en slik respekt for dataene. Dette gjøres primært ved å bruke konservative datavask metoder[5]. Dette valg underbygges også av rådende retningslinjer, her fra District of Columbia Fire and Emergency Medical Services Department (FEMS).

> *"If any single vehicle's incident arrival time is missing, the comparative measure calculation for all vehicles cannot be completed. As such, incidents with "incomplete time value data" are excluded from benchmark time goal comparisons"* (FEMS, 2016)[6].

All preprosessering og datavask beskrives i Vedlegg I seksjon I.3 og seksjon I.4.

3.2 Process Mining

Her beskrives PM metoden. Data som brukes i PM er beskrevet i **Vedlegg I** seksjon **I.4**, note 1-5[7].

3.2.1 Preprosessering av Data for PM

I PM må data behandles i tråd med kravene til en hendelseslogg. Dette innebærer å ha et minimum av variabler. Dette minimumskrav er representert i figur **3.1**, og innebærer at dataene må inneholde ID, tid og aktiviteter. NYC data er hendelsesdata, og er således skikket til å kunne brukes i PM, men må transformes til logg format. Dette skjer fordi dataene ikke er loggført som hendelseslogg[8]. Fra databasen OpenData, så får en det som kalles ifølge Aalst, "flate tabeller" (2016, s.153-154). I flate tabeller er en unik case, en rad. I en hendelseslogg så er en rad, en aktivitet, der en unik case kan ha flere observasjoner (rader), avhengig av antall aktiviteter utført. I PM blir dataene transformert i likhet med figur **3.1**. Dette belyses i **Vedlegg II.1**, der hendelsesloggen blir utledet og forklart.

3.2.2 Hendelseslogg Variabeler

Hendelsesloggen består av fire variabler. Variablene som brukes er tid, aktivitet, case ID og aktivitetinstans ID. Disse variablene er egnet for å lage en PM modell (Aalst, u.å, s.9).

[5]Innebærer blant annet at vi ikke vil estimere verdier for manglende (NAs, not avilable, missing values) observasjoner. Metoden heter imputation (tilegnelse) av manglende observasjoner, der bruk av regresjon kan gi et estimat for den manglende observasjon. Dette brukes ikke.

[6]For ytterligere spesifikasjon av benchmarks se NFPA standard 1710.

[7]Noter henviser til beskrivelse av datavask.

[8]Fra teorien fremkommer det at PM er i nødvendighet av hendelseslogger, hvor hendelseslogger inneholder hendelsesdata. Det finnes altså hendelsesdata som ikke er i hendelseslogg format, som er nødvendig for at algoritmene skal kunne blir brukt i PM.

id	aktivitet x	aktivitet y
1	11:00	12:00
2	13:00	14:00
3	15:00	16:00

id	tid	aktivitet
1	11:00	x
1	12:00	y
2	13:00	x
2	14:00	y
3	15:00	x
3	16:00	y

Figur 3.1: Transformasjon av Rådata til Hendelseslogg.

3.2.3 Process Discovery

I process discovery vil vi nyttiggjøre oss av to algoritmer, alfa pluss og fuzzy. Vi bruker det samme datasett og hendelseslogg for begge algoritmene. Algoritmene brukes for prosessforståelse. Hovedfokus med PM er å lage en prosessmodell, som kan brukes i problemstilling 1. Prosessmodellen som brukes for besvarelse av problemstilling 1 er alfa pluss. Fuzzy brukes bare for eksplorativ dataanalyse (EDA).

Fuzzy algoritmen

Fuzzy algoritmen er blant de mer intuitive algoritmene for å forstå store datasett, da algoritmen har en meget simpel grafisk notasjon (symboler), i prosessmodellen (TUe, 2010). Algoritmen har til hensikt å skape en prosessmodellen. Prosessmodellen viser all data fra hendelsesloggen. Fuzzy modellerer all aktivitet, uten å gjøre avveininger, slik Petri net gjør[9]. Prosessmodellen fra fuzzy er som et kart, da den viser alle traces fra hendelsesloggen (Anne, 2010). Strukturen til modellen er slik at en aktivitet alltid følger en annen, med hensyn på aktivitetene sin rekkefølge. Diagnose måltall vil derfor ikke være av relevans for denne algoritmen, da den ikke søker å gjøre avveininger.

Fuzzy modellene som blir presentert i kapittel fire, inneholder informasjon om tidsbruk (snitt) mellom aktivitetene, og informasjon om antall caser, som har vært innom en spesifikk aktivitet.

Alfa pluss algoritmen

Alfa algoritmen er en mye brukt PM algoritme. En av styrkene ved alfa algoritmen, er evnen til å takle aktiviteter som skjer samtidig om hverandre (Aalst, 2016, s.167). I litteraturen er det likevel påpekt flere svakheter ved alfa algoritme. Når det gjelder alfa algoritmen, så har den problemer med å representere gjentakende aktiviteter (loops) (Aalst, 2016, s.167-176). Som et svar til dette, har alfa pluss algoritmen blitt introdusert, da denne evner bedre og modellere gjentagende aktiviteter (Aalst, 2016, s.176). Utover

[9] Alfa pluss gjør avveininger, som måles i måltallene.

dette, så er rammeverket for alfa pluss algortimen det samme som for alfa algoritmen[10]. For analyse, vil alfa pluss produsere en prosessmodell. Denne prosessmodellen kan inneholde; XOR JOIN, XOR SPLIT, AND SPLIT og AND JOIN symboler.

XOR SPLIT er når en case står framfor et valg mellom flere aktiviteter, og må velge (enten eller) for å komme videre til en annen fremtidig aktivitet. Etter dette valget er tatt, vil casen nyttiggjøre seg av XOR JOIN. XOR JOIN er en betegnelse for at en case tidligere har tatt et valg mellom flere aktiviteter, og har passert. AND SPLIT er når en case må gå i flere retninger, for å kunne passerer videre i prosessene. Dette kan eksemplifiseres fra figur **2.3**, der første aktivitet er registrering, for at casen skal videre gjennom en sjekk, der denne sjekken er enten (A) STANDARD, eller (B) SKIKKELIG (mer utfyllende). Fra eksempelet (i figuren) skal alle caser gjennom en sjekk, men som det fremkommer fra figuren, så er det et valg mulighet mellom (A) og (B). Etter AND SPLIT må casen møtes i en AND JOIN, for videre progresjon.

3.2.4 Conformance Checking

For evaluering av prosessmodellen, Petri net brukes *evalueringstall*. Måltallene er fitness, simplicity, precision og generalization. Disse måltallene kan vise om prosessmodellen er konform (conformance checking) med hendelsesloggen[11]. Måltallene tar på seg en verdi fra 0 til 1. I studien som Buijs et al (2012) gjennomførte, scoret alfa algoritmen høyt på presisjonsmåltall[12]. Loggen som ble brukt hadde syv aktiviteter, og var basert på lånesøknader (Buijs et al, 2012, s. 316-322). Det er like mange aktiviteter som er å finne i dataene tilknyttet denne studien (se **Vedlegg II**.1.).

Generalization

Generalization omhandler overfitting av loggen. En prosessmodell som overfitter klarer ikke generalisere, men reflekter kun hva som har skjedd i selve hendelsesloggen. Det vil si at en prosessmodell som har lav form av generalisering vil ikke kunne ta høyde for ny[13] atferd, som ikke er fanget opp i hendelsesloggen. Hvis verdi er lik 1, så betyr det maks generalisering.

Fitness

Fitness måltallet representerer antall traces (aktivitetskombinasjoner) som er å finne i hendelsesloggen (Evaluation Log-Model, u.å). Om en prosessmodell har god fitness, tilsvarer

[10] Modell produseres i R. Se **Vedlegg II** II.2 for detaljer.

[11] Alle måltallene fremkommer fra utskriftene i R, der pakken PM4PY brukes (Evaluation Log-Model, u.å).

[12] Algoritmen fikk følgende resultater. Fitness: 0.992. Simplicity: 1.000. Generalization: 0.889. Precision: 0.995 (Buijs et al., 2012, s.305-310).

[13] Dette kan være tilfelle i fenomenet "concept drift", som dreier seg om at observert atferd kan endres umiddelbart eller over tid. Dette kan være som følge av at det er ønskelig å handle ulikt fra standardiserte rutiner. Grunnet blant annet økt konkurranse eller sesong effekter (Aalst et al., 2012, s.14).

dette at mesteparten av aktivitetene (med tilhørende traces) i loggen, fremstilles i modellen (Aalst, 2016, s.167). En prosessmodell, ifølge Aalst et al (2012, s.14), sies å ha perfekt[14] fitness hvis den tillater alle traces (aktivitetskombinasjon) i hendelsesloggen, til å bli gjenspeilet i prosessmodellen. Hvis verdi er lik 1, så betyr det perfekt fitness.

Precision

Precision taler om underfitting, hvor hendelser som avviker markant i forhold til hva som er observert i logg, blir fanget opp i prosessmodellen (Aalst, 2016, s.167). Det er ikke ønskelig med traces i prosessmodellen, som ikke eksisterer i hendelsesloggen. Dette skaper en villedende prosessmodellen (Aalst et al., 2012, s.12). Hvis verdi er lik 1, så betyr det maks grad av precision.

Simplicity

Simplicity belyser om en prosessmodell inkluderer unødvendig mange hendelser, men bare de mest relevante som skal til for å forklare atferden i hendelsesloggen. Det blir nærmest umulig å representere alle traces i en logg[15], for meningsfull analyse (Aalst et al., 2012, s.189).

De presenterte måltallene brukes for å teste problemstillingen 1 vedrørende om: "process mining vil kunne lage en god prosessmodell av NYC hendelsesdata?"

3.3 Maskinlæring

Her beskrives ML metode, og hvordan hendelsesdata skal brukes i prediksjon. Først presenteres datavask, så konstrueres ML modeller.

3.3.1 Konstruksjon av Variabler

Den avhengige variabel

Den avhengige variabel i analysen er gjennomstrømstid. Variabelen for gjennomstrømstid skal predikeres i ML modellen[16]. Gjennomstrømstid er differansen mellom start tidspunkt og slutt tidspunkt.

Dummyvariabler

[14] Perfekt i og med at modellen fanger opp alt, betyr likevel ikke at modellen er god.
[15] Betinget på at logg er kompleks. Hvis få aktiviteter, så er ikke dette nødvendigvis et problem. Måltallet er inspirert av Occams barberkniv.
[16] Fremkommer i datasett som THROUGHPUT_TIME.

For ML lages dummyvariabler[17], som indikatorer for manglende observasjoner (NAs) i aktivitetsvariablene[18]. Målet er å gi informasjon til ML modellen, om hvorvidt én case har manglende observasjoner. Dette kommuniserer til ML modellen hvilke, og hvor mange aktiviteter som er tilknyttet én case. I tillegg til dette kommuniseres det hvilke aktiviteter som ikke er tilknyttet en case. Slike variabler er også brukt i andre studier. Dongen et al (2008, s.6) presenterte en variabel for hvor mange aktiviteter som var tilknyttet en unik case.

Tidsvariabler

Tidsperspektiv inkluderes i ML modellen. Dette gjelder når en case har inntruffet. Det er nærliggende å tro at eksempelvis, tid på døgnet vil påvirke gjennomstrømstid, samt andre attributter. Vil det være flere hendelser på formiddag fremfor ettermiddag? I aktivitetsvariablene er det registrert tidspunkt (timestamps) for når en aktivitet har funnet sted. Tidspunkt innehar potensielt viktig informasjon som kan bidra til å forbedre ML modellene. I studien Lingnitz et al (2018, s.1055) gjennomførte, var to tidsvariabler (ankomst og uke) blant de ti viktigste variablene. For å best representere slike tidsdata vil det bli produsert seks variabler, som er basert på aktivitetsvariablene[19].

3.3.2 Datavask Tilknyttet ML

Eliminasjon av Numeriske Variabler

Korrelasjonsmatrise i **Vedlegg I.6** ble brukt for å identifisere hvilke numeriske variabler som ble eliminert. Korrelasjon viser den lineære sammenhengen mellom to numeriske variabler. Korrelasjon er en verdi som strekker seg fra verdiene -1 til +1, hvor -1 er perfekt negativ lineær samvariasjon og +1 er perfekt positiv lineær samvariasjon. Variabler med stor korrelasjon, kan føre til multikollineæritet. Da begge variablene er med på å forklare samme variasjon[20].

Eliminasjon av Diverse Variabler

Det eksisterer variabler som forklarer minimal variasjon i den avhengige variabelen, disse fjernes[21]. Andre variabler er utenfor gjeldende avgrensninger, og derav overflødige[22]. Tidsvariablene er i uegnet dataformat og er substituert med dummy variabler[23].

Eliminasjon av Kategoriske Variabler

[17] Informasjon om dummyvariabler, se **Vedlegg I.4**, note 7-10.
[18] Se **Vedlegg I**.3, POSIXct variabler.
[19] For utdypende informasjon av konstruksjon av tidsvariabler, se **Vedlegg I**: I.4, note 11-16.
[20] Hvilke numerisk variabler som ble eliminert som følge av korrelasjonsplotet belyses i **Vedlegg I**.6 og I.4 note 49.
[21] **Vedlegg I.4**, note 24.
[22] **Vedlegg I.4**, note 25.
[23] **Vedlegg I.4**, note 17-23.

For å belyse de mange faktor variabler[24] så er det lagd en figur basert på "Cramers V" (Cramers V, u.å). Resultatene er presentert i **Vedlegg I.5**, og vil her bli brukt for å utvelge hvilke variabler som skal brukes i modellen[25]. Cramers-V funksjonene resulterer i et chi square, som gir et måltall mellom 0 og 1. Dette tallet viser samvariasjon mellom to faktorvariabler. Tolkningen er tilnærmet det samme som for korrelasjon, 0 er ingen samvariasjon, hvor 1 indikerer at variablene har perfekt samvariasjon. Ved hjelp av chi-square, er det identifisert flere faktorvariabler med perfekt korrelasjon 1. Det er valgte å sette en terskel på maksimum 0.8 korrelasjon[26], variabler som faller under denne terskelen beholdes[27].

Eliminasjon av Observasjoner

Hendelsesdataen inneholder flere anomaliteter. Dette er tilfellet hvor kategorier av faktor variabler inneholder svært få observasjoner. Et annet eksempel er når kategorier i faktor variablene er registrert i hendelsesdata, men ikke henvist til i forklaringen av hendelsesdata (se **I.2** forklaring av EMS variabler). Der en kategori er registrert i hendelsesdata, men ikke gitt noen forklaring av FDNY EMS, anses slike kategorier som spuriøse, og fjernes. Kategorier som har ekstremt få observasjoner vil kunne skape problemer ved trening av ML modellen. Dette fordi svært få observasjonen kun blir fanget opp i testing av modell og ikke trening (eller vice versa). For å unngå dette problemet, blir variabler med ekstremt få observasjoner eliminert[28].

Manglende observasjoner (NAs) tilhørende gjennomstrømstid vil bli eliminert. Hvis det er en manglende observasjon i den avhengige variabel, betyr det at enten den første aktivitet, eller den siste, ikke har skjedd, eller ikke er loggført. Det avgrenser til komplette caser[29].

Unormalt store verdier blir eliminert[30]. Dette er foranket i fremgangsmetodikken til FEMS EMS.

> *"If this time value results in a vehicle "Travel Time" duration exceeding 1,800 seconds (30 minutes), there is high likelihood the time value was inaccurately recorded"* (FEMS, s.47, 2016).

Det endelige datasettet for ML (ABT) fremkommer i figur 3.2.

[24]Faktor er ekvivalent med kategorisk. Faktor er begrepet R bruker.
[25]I R modellerte vi ved hjelp av krysstabeller de nødvendige data for å kunne bruke Cramers-V funksjonen.
[26]Gjøres for å unngå mulig multikollinearitet.
[27]**Vedlegg I.5** og I.4, note 27-43.
[28]**Vedlegg I.4**, note 47-48.
[29]**Vedlegg I.4**, note 26.
[30]**Vedlegg I.4**, note 44-46.

```
-- Data Summary ------------------------
                           Values
Name                       ABT
Number of rows             207346
Number of columns          15
___________________________
Column type frequency:
  factor                   12
  numeric                  3
___________________________

-- Variable type: factor ---------------------------------------------------------------------------------------------
# A tibble: 12 x 6
   skim_variable               n_missing complete_rate ordered n_unique top_counts

 1 INITIAL_SEVERITY_LEVEL_CODE         0             1 FALSE          8 3: 143339, 2: 41215, 6: 12102, 4: 6182
 2 FINAL_CALL_TYPE                     0             1 FALSE          3 CAR: 165916, CAR: 41390, CAR: 40
 3 FINAL_SEVERITY_LEVEL_CODE           0             1 FALSE          3 3: 165917, 2: 41394, 4: 35, 6: 0
 4 HELD_INDICATOR                      0             1 FALSE          4 fal: 109120, N: 95942, tru: 1274, Y: 1010
 5 INCIDENT_DISPOSITION_CODE           0             1 FALSE          9 82: 178920, 93: 18956, 90: 4925, 96: 2476
 6 INCIDENT_DISPATCH_AREA              0             1 FALSE          9 M3: 39427, M2: 34062, M9: 25703, M1: 25286
 7 FIRST_ACTIVATION_DATETIME_D         0             1 FALSE          2 1: 207134, 0: 212
 8 INCIDENT_DATETIME_year              0             1 FALSE         12 201: 33370, 201: 31620, 201: 25131, 201: 14364
 9 INCIDENT_DATETIME_time              0             1 FALSE         24 12: 13094, 11: 12937, 13: 12453, 10: 12368
10 INCIDENT_CLOSE_DATETIME_time        0             1 FALSE         24 13: 12988, 14: 12817, 12: 12794, 15: 12201
11 INCIDENT_DATETIME_Weekday           0             1 FALSE          7 3: 31810, 2: 31704, 4: 31638, 5: 31158
12 INCIDENT_DATETIME_Week              0             1 FALSE         53 18: 4292, 50: 4271, 37: 4228, 20: 4226

-- Variable type: numeric --------------------------------------------------------------------------------------------
# A tibble: 3 x 11
  skim_variable               n_missing complete_rate   mean     sd    p0   p25   p50   p75  p100 hist

1 DISPATCH_RESPONSE_SECONDS_QY         0             1   28.6   24.2     2    13    21    35   160 ▇▂▁▁▁
2 INCIDENT_TRAVEL_TM_SECONDS_QY        0             1  375.   196.      1   247   337   456  1796 ▇▇▂▁▁
3 THROUGHPUT_TIME                      0             1 4142.  1340.     27  3405  4192  4968 10000 ▁▃▇▂▁
```

Figur 3.2: Datasett for Maskinlæring.

3.3.3 Anvendt Maskinlæring

Her vil ulike algoritmer som er andvendt for ML bli beskrevet. Videre vil eksersisen for konstruksjon og kalibrering av modell bli fremvist.

Regresjonsanalyse

For å predikere den totale tid for en hendelse, så er regresjon en mye brukt metode. I denne studien vil lineære modeller bli anvendt. Dette anses som tilstrekkelig, for å gi en indikator på interaksjon mellom avhengig og uavhengig variabel (Hastie et al., 2017, s.43).

Lineær Regresjon

Lineær regresjon er en av algoritmene som andvendes for å predikere gjennomstrømstid. Modellparametrene blir konstruert ved å bruke minste kvadraters metode. Dette minimere total støy ved å innføre en kurvetilpasning (Gripsrud, Olsson & Silkoset, 2016, s.302). Denne kurvetilpasningen gjør at modellen minimerer residualkvadratsummen, med hensyn på den avhengige variabel (Sucarrat, 2016, s.52). Den lineær regresjonen kan fremstilles slik.

$$Y = \alpha + \beta X + \epsilon \tag{3.1}$$

Y er den avhengige variabel gjennomstrømstid. α er konstanten i ligningen og gir oss den

estimerte verdien på y, gitt at x er lik null. β er den estimerte koeffisient til variabelen, på basis av utvalgte populasjonsdata. Flere beatakoeffesienter vil bli introdusert for å estimere gjennomstrømstid. Disse koeffisientene kommer fra multivariat lineær regresjon (MLR). β gir oss tolkningen for hvordan den avhengige variabel blir påvirket av den uavhengige variabel. Restleddet (feilledd) ε gir oss info om uforklart varians, som X ikke evner å forklare, med hensyn på estimeringen av Y.

I multivariat lineær regresjon (MLR) bruker vi minste kvadraters metode for å predikere gjennomstrømstid. Alle de uavhengige variabler som beskrevet i det foregående kapittel, tas med i modellen.

Ridge og Lasso regresjon

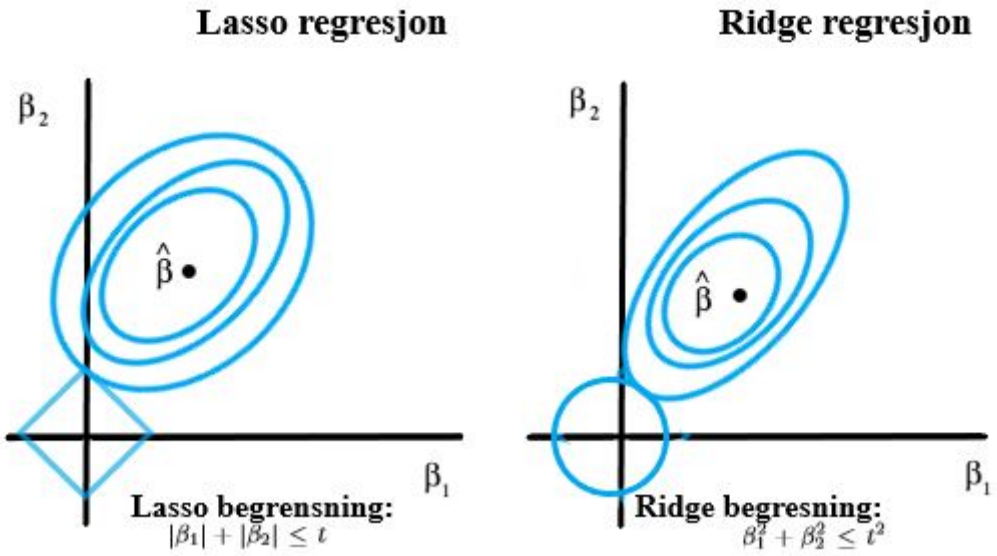
Ridge Regresjon og Lasso gir lineære modeller, ved bruk av en krympe-metode på koeffisientene. Koeffisientene krympes ved en straffutmåling basert på hyperpameteren lambda. Lambda, λ, krympingen baserer seg på koeffisient størrelsen og residualkvadratsummen. Desto større lambda, desto større krympe effekt (Hastie et al., 2017, s.63). Lambda krymper koeffisienter med hensikt på å redusere varians[31] (Hastie et al., 2017, s.63).

$$\beta_1^2 + \beta_2^2 \leq t^2 \tag{3.2}$$

$$\mid \beta_1 \mid + \mid \beta_2 \mid \leq t \tag{3.3}$$

Funksjonen som viser til krympeeffekten sess i formel **3.2** for Ridge begrensning (sirkel) og formel **3.3** er Lasso begrensning (kvadrat) (Hastie et al, 2017, s.71). I figur **3.3** fremkommer elliptiske sirkler fra minste kvadraters feil (Hastie et al, 2017, s.71). I lasso og ridge blir koeffisientene determinert der den elliptiske sirkel treffer den angitte begrensning. Vi ser da fra figuren at det er to vidt forskjellige begrensninger, der den ene er en sirkel og den andre er et kvadrat. Lasso er annerledes enn ridge, da koeffisienten kan få verdi null om beta treffer en av aksene. Dette skjer på bakgrunn av lasso (kvadrat) begrensningen (Hastie et al., 2017, s.72). Med flere koeffesienter (betaer) vil en kunne risikere å få flere verdier mot null (Hastie et al., 2017, s.72).

[31] Som kan være et resultat av høyt korrelerte variabler (Hastie et al., 2017, s.63).



Figur 3.3: Illustrasjon av Lasso og Ridge - Regresjon. Kilde: Tilpasset fra Hastie et al., (2017, s.72).

XGBoost

XGBoost er en algoritme basert på ensemble metodikken kalt boosting. Ensemble i ML innebærer bruken av flere separate modellers utfall/prediksjon. Boosting er en av flere ensemble teknikker (Elite data science, u.å.). Boosting fokuserer på å andvende flere "svake"[32] modeller for å konstrure en endelig modell (ensemble) (Hastie, 2017, s.337). En svak modell (learner), ses i kontekst av hvor restriktiv modellen er[33].

Boosting er en iterativ prosess, dette innebærer at informasjon fra hver svake modell som blir konstruert, blir tilegnet den neste modellen i sekvensen. For hver sekvens ender man opp med en modell med høyere forklaringskraft, da modellen har iterativt lært av feil den tidligere modellen har gjort (Hastie et al, 2017, s.338). På en slik måte mister tidligere svake modeller sin innflytelse, da ny informasjon blir tillegnet den neste modellen. Denne sekvensen gjentas slik at man til slutt står igjen med en helhetlig modell, og derav navnet ensemble (Hastie, 2017, s.337). Ensembele modellen har så større forklaringskraft, enn en svak modell isolert sett. Ved tilfellet av XGBoost, anvender vi en lineær variant som måler det lineær forholdet mellom avhengig og uavhengige variabler. Den additive prosessen ved å inkludere flere svake modeller til et ensemble (kalt boosting) er henvist til i funksjon **3.4** (2017, s.341).

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m) \tag{3.4}$$

[32]Dette kalles også for "weak learnes"
[33]dette kan f.eks være max dybde i hendelsestrær (Elite data science, u.å.))

Der antall iterasjoner betegnes som M sekvenser[34]. Etter hver sekvens (M) blir modellen krympet og kontrollert, ved hjelp av hyperparametrene alfa og lambda (Hastie, 2017, s.364). Krymping av modellparametrene andvendes for å motvirke overfitting[36]. De nevnte hyperparameterene er med på å kalibrere læringsprosessen, slik at modellen vil kunne gjøre mer pressise prediksjoner (James, 2013, s.322)[37].

3.3.4 Variabel Transformasjon

Det andvendes min-max feature scaling for normalisering av variabler. Ved min-max feature scaling redusere man de numeriske variablene til en rekkevidde fra 0-1 (Feature scaling, u.å.). Denne metoden er andvendt på alle de numeriske uavhengige variablene. Hensikten med normalisering av de numeriske variablene er å behandle variabelen på lik skala. Reisetid er på en større skala (tidsbruk snitt 375 sek), enn responstid (tidsbruk snitt 28 sek). Se figur 3.2.

Normalisering er særdeles viktige ved bruk av Lambda (λ) hyperparameteren. Om ikke normalisering blir brukt på de uavhengige numeriske variablene, vil regulariseringsparameteren lambda straffe variabler mer enn andre, ettersom de ikke operer i samme rekkevidde (James et al , 2013, s. 217)[38]. Normalisering blir andvendt under treningsfasen i repetert kryssvalidering[39]. Dette gjøres for å unngå datalekk. Datalekk er et fenomen der informasjon utenom treningsdata, blir brukt til treningen av modellen. Dette medfører til bias i modellen. Datalekk vil kunne lede til optimistiske modell resultater, da modellen har hatt innsikt i data, som direkte forklarer fenomenet man ønsker å predikere (Schutts, O'Neil, 2013, s.313). Sannsynligheten for datalekk reduseres ved å bruke normalisering etter splitt av data, i trening og test. Dette medfører at modellen ikke har lært fordelingen (min max transformasjon) i testdata, og derav tar med seg denne kunnskap i trening av modellen.

3.3.5 Trening av Modell

Her vil det forklares hvilke metoder som ble andvendt for å trene de ulike modellene. Modellene som blir trent er basert på tidligere utledede algoritmer. Modellene blir trent[40] ved hjelp av å kalibrere de ulike algoritmene, slik at sammensetningen av hyperparametere

[34]ved bruk av XGBoost er Nrunder andvendt istedet for M, dette ser man fra figur **4.2**. Læringsraten (ETA) er en hyperparameter som bestemmer boosting prosedyrens sekvensielle fremdrift[35](XGBoost, u.å.)

[36]Dette innebærer å balansere bias/varians. Kilde: Hastie (2017, s.364).

[37]Krympe parameterne er noe som skiller XGBoost fra andre boosting algoritmer, da dette er særegent for XGBoost. Kilde: Gradient Boosting (u.å.).

[38]James bruker en normaliseringstype kalt standardisering. Dette krever normalfordeling. Derfor velges heller min max normalisering, fordi det ikke krever normalfordelte variabler.

[39]Blir beskrevet i 3.3.5. Brukes på hvert fold.

[40]Pakken i R, kalt Classification and Regression Training (CARET), ble andvendt for trening av maskinlæringsmodellen (Kuhn, 2019).

gir optimale modellparametere. Disse modellparametere resulterer i best treningsscore og implisitt lavest total feil.

Trening og Testdata

Datasettet ble splittet opp med 80 % allokert til treningsdata, hvor resterende 20 % ble allokert til testdata. Data ble splittet tilfeldig, der seed ble satt til 13678[41] [42]

Siste 20 % av datasettet, testdata, vil ikke bli tatt i bruk før modellen er trent. Testdata settes til side, slik at man kan revidere den endelige modellens prediksjonskraft, og om den overfitter eller underfitter. Målet er at modellen skal kunne generalisere som nevnt i kapittel 2, om bias og varians.

Repetert K-fold Kryssvalidering

Det vil her spesifiseres hvordan repetert K-fold kryssvalidering ble andvendt for kalibrering av hyperparametrene. Repetert K-fold kryssvalidering ble brukt i tandem med grid søk eller tilfeldig søk, avhengig av algoritmen.

Ved K-fold kryssvalidering splitter man *treningsdata (80%)* opp i K - 1 antall treningsfold (Hastie et al, 2017, s.242). Hvor resterende siste fold(1) blir holdt til side for testing. K i K-fold kryssvalidering spesifiserer antall treningsfold, som dataen splittes opp i. Det er andvendt K=10 i trening av modellen, som innebærer at treningsdata splittes opp i 10 - 1 = 9 like store deler. Siste testfold blir holdt til side for å teste modellen som er trent på de 9 resterende treningsfold.

Ved repetert kryssvalidering spesifiseres hvor mange ganger K-fold kryssvalidring skal gjentas (Kuhn, 2019). Ved vårt tilfelle har er det valgt å spesifisere antall repetisjoner til 10.

Grid søk og Tilfeldig søk

Det er andvendt grid-søk på lasso og ridge regresjon[43]. Ved grid-søk spesifiseres et grid, over en rekke av verdier (Kuhn, 2019). Lengden på gridet er satt til å være 10^{-2} til 10^{2}, som tilsier verdier 0,001 til 100. Innad i gridet har vi spesifisert at algoritmen kan velge 20 verdier, hvor ønske er å finne hvilke hyperparameter som minimerer/maksimere et spesifisert mål. *I denne studien ønskes MAE å minimeres.* For hvert set av hyperparametere[44]

[41] Tilfeldig plukke-orden (sampling), som er reproduserbar. Det vil si, når et script eventuelt kjøres senere, genereres likt svar, som tidligere.

[42] CreateDataPartition i CARET sikrer balanserte proporsjoner mellom klassene i faktor variabler, tilknyttet testdata og treningsdata. Dette medfører at data vi trener på blir representativt til hva vi tester på. Dette kan leses mer om under funksjonen createDataPartition i CARET dokumentasjonen, kapitel 4.1 (Kuhn, 2019).

[43] Tilfeldig søk er andvendt på XGBOOST, fordi denne algoritmen består av 4 ulike hyperparametere. Grid for hver hyperparameter er meget intensivt for CPU, og ble derfor ikke brukt.

[44] Lasso og ridge regresjon har kun en hyperparameter, som er lambda, (λ).

gjennomføres repetert 10-fold kryssvalidering.

Ved algoritmer som inneholder mer enn en hyperparameter (XGBoost) andvendes tilfeldig søk. Dette innebærer at det blir valgt arbitrære[45] sammensetninger av hyperparametere som søker å minimere MAE (Kuhn, 2019). Det er spesifisert at ved tilfeldig søk er terskel for antall sammensetninger av hyperparameter kombinasjoner satt til 10. Dette gjelder hyperparameterne som ble nevnt i utleggelsen av XGBoost algoritmen. For hvert set av hyperparameter sammensetninger, blir så repeterte 10-fold kryssvaldiering gjennomført.

Etter repetert K-fold kryssvalidering er gjennomført, vil man sitte igjen med en vinner-modell for alle de valgte sett av hyperparametere. Av disse modellene selekteres den modellen som har en hyperparameter samensetning, som har oppnåd lavest gjennomsnittlig MAE over sine 10 kryssvaliderings iterasjoner. De beste hyperparametere blir så trenet på 80 % av treningsdata. Dette gjøres for alle algoritmene, som resulterer i optimale modellparametere, for den endelige modell.

```
1  Definer set av hyperparametere man ønsker å evaluere
2  For hvert set av hyperparametere gjør
3      For hver  resampling repetisjon gjør
4          Hold til side K-1 fold for trening av model
5          Tilføy pre prossesering av data (Normalisering)
6          Fit/Tren modellen
7          Prediker på Fold som er satt til side
8      Slutt
9      Kalkuler gjennomsnitt presisjon over alle fold som er holdt til side
10 Slutt
11 Avgjør det optimale set av hyperparametere
12 Fit/tilpass den endelige modelen på hele treningsdata på 80 % med optimale parametere
```

Figur 3.4: Repetert K-fold Kryssvalidering. Kilde: Tilpasset fra Kuhn (2019, CARET dokumentasjon kapitel 5.1).

Figur **3.4** viser strømlinjen for hvordan repetert K-fold blir andvendt for trening av de ulike modellene[46] [47]

Hensikten med K-fold repetert kryssvalidering, er å motvirke overfitting i den trenede modellen. Om en ikke andvender en kryss validerigsteknikk, og kun trener modellen på 80 % av data, for så å teste opp mot 20 %, vil man ikke være like sikker på om modellen klarer å generalisere. Dette fordi modellen kun har blitt testet en gang, og kan derav heller ikke gi en indikator om modellen presterer godt ved ny usett data. K-fold gir en

[45] Verdier kan gjenproduseres med lik seed, se Kuhn (2019, seksjon 10).

[46] Preprosessering av data (steg 5) er andvendt for hver iterasjon av K-fold, dette er for å sikre at man minimerer datalekk ved variabel transformasjon.

[47] Det eneste som blir endret på ved bruk av grid eller tilfeldig søk er steg 1 i figur **3.4**, definering av sett av hyperparametere. For visualisering av disse hyperparametersett se vedlegg **III.1.**

pekepinn på om modellen klarer å generalisere (Hastie et al, 2017, s.242-244).

3.3.6 Modellvalg Kriterium

For å evaluere modellene som algoritmene produserer (på basis av treningsdata), bruker vi ulike presisjonsmåltall[48] knyttet opp mot de usette testdata. De to presisjonsmåltall er følgende; MAE og RMSE[49]. Evaluering av modellen på dette vis, vil sikre forskningens robusthet og validitet, da problemstillingen blir utprøvd ved å evaluere data som er urørt (testdata). Modellen brukes først for å utregne prediksjoner på testdata. Hvor en så sammenligner historiske data, med hva modellen estimerer. Differansen mellom estimater og historiske data, kalles for residualer. Disse residualene er utgangspunktet for å estimere RMSE og MAE.

MAE som måltall bruker de absolutte verdier (av feilledd), for å utregne snittverdien av disse.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{3.5}$$

RMSE som måltall bruker de kvadrerte verdier (av feilledd), for å utregne snittverdien av disse, sammen med å ta kvadratroten.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2} \tag{3.6}$$

MAE gir en absolutt fremvisning av feilledd. RMSE i større grad straffer verdier på basis av hvor store de er, som en naturlig konsekvens av kvadratrot (Jank, 2011, s.140). Hvis eksempelvis MAE/RMSE er hundre, så betyr det at modellen predikerer i snitt med hundre sekunder feil (på avhengig variabel i testdata)[50].

Problemstillingen om å predikere gjennomstrømstid i NYC hendelsesdata, vil bli testet ved disse måltall. Den beste modell er den med lavest MAE.

[48] Dette kalles testsettscore.

[49] MAE = Mean absolute error, RMSE = Root mean square error.

[50] Lignitz (2018, s.1054) endte opp med beste MAE = 390 minutter (se fig. 4.). I et slikt tilfelle er tolkningen annerledes for MAE, avhengig av den avhengige variabel (minutter eller skunder).

Kapittel 4

Resultat

Her blir resultatene fra metoden presentert. Vi tar for oss PM og de utvalgte metoder, for å belyse ambulanse relaterte aktiviteter i dataene. I seksjonen for ML presenteres modell kandidatene, hvor beste modell for prediksjon av gjennomstrømstid kåres. Resultater vil være gjenstand for diskusjon i kapittel 5.

4.1 Process Mining

Hendelsesloggen som er beskrevet i vedlegg II.1, brukes i PM. Her blir prosessmodellene fra fuzzy og alfa algoritmen presentert.

4.1.1 Prosessmodell

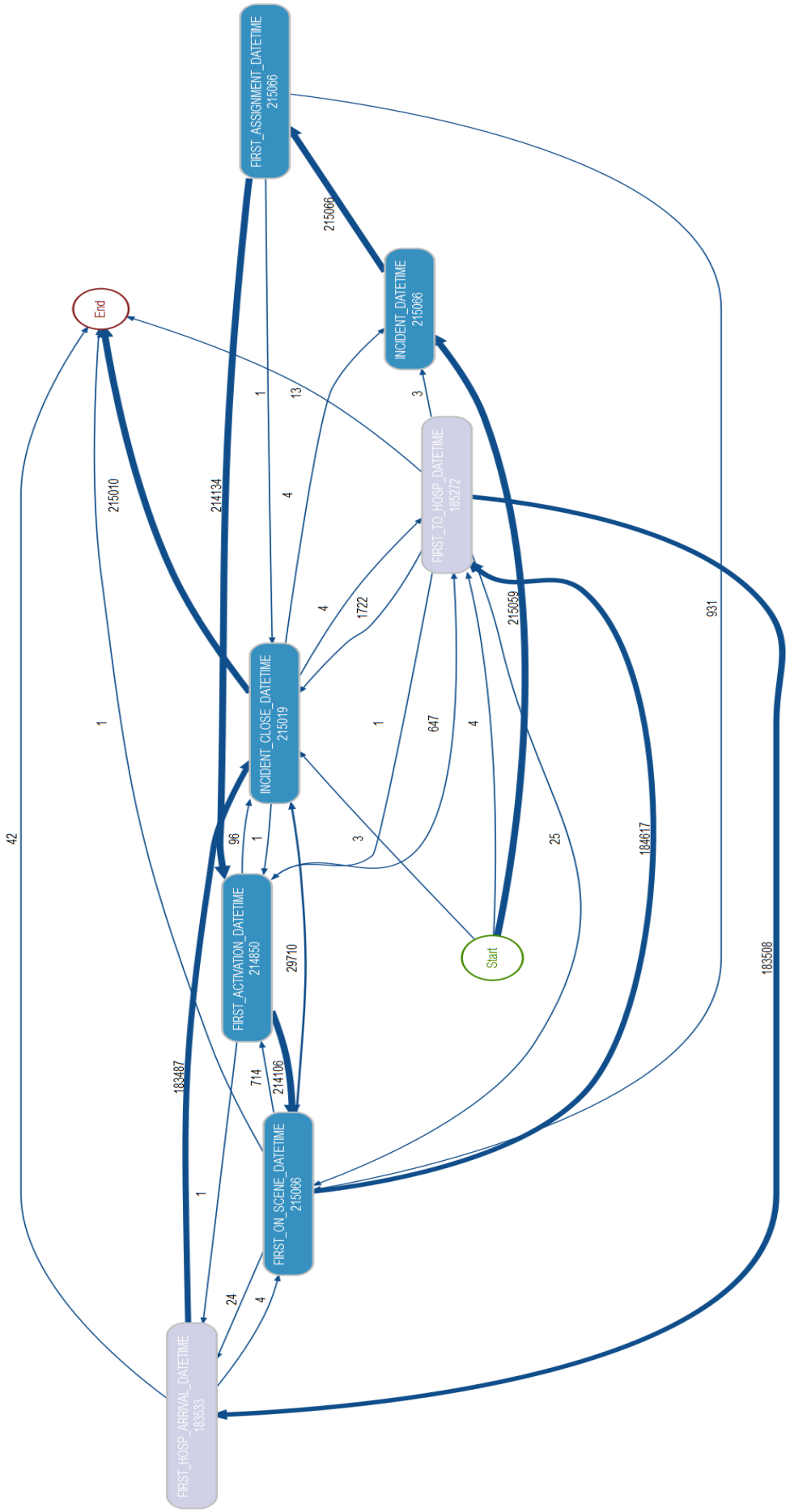
Fuzzy prosessmodell - Case perspektiv

Figuren **4.1** viser stor samstemthet[1] i hvordan aktiviteter blir behandlet i møte med akutt helsetjenesten. De fleste caser er innom alle de syv aktivitetene. Det fremkommer i majoritetsrute[2] der det er en logisk sekvensiell anordning av aktiviteter. Det fremkommer at det er normalt at en aktivitet følger en annen. Det er ikke tilfeldig hvilke rekkefølge aktivitetene er anordnet etter. Dette vet vi fra datagrunnlaget, og dette kan sees i **Vedlegg I.3**, der vi observerer syv aktivitetsvariabler.

I figuren **4.1** er det avbildet et prosessmodell basert på fuzzy algoritmen som vi har beskrevet i teori. I dette tilfelle, så vises alle mulige veier fra loggen, og representasjonen er 100 % av loggen. Det observeres start og endepunkt, dette er ikke aktiviteter, men en grafisk fremstilling for å vise hva den første aktiviteten har vært, og hva den siste aktiviteten har vært - for en case.

[1] Motstykket er uoversiktlige "spaghetti modeller", se Aalst (2016) kap 14. Med samstemthet menes en kompakt oversiktlig prosessmodell.

[2] Majoritetsruten er i uthevet i blå farge, samt at gjennomsnitt trace er på 6.7 aktiviteter (**Vedlegg II.1**). Tatt i betraktning at det er maks syv antall aktiviteter, så er det fornuftig å anta en slik majoritets-rute.



Figur 4.1: Absolutte Antall Gjennomstrømning Målt i Unike Caser.

I figuren **4.1** ser vi at en majoritetruten har begynt med aktivitet INCIDENT_DATETIME, og en minoritetrute har begynt med FIRST_TO_HOSP_DATETIME og INCIDENT_CLOSE _DATETIME. Blant de siste aktiviteter en case har tatt, så har vi fire alternativer, FIRST_ON_SCENE_DATETIME, INCIDENT_CLOSE_DATETIME, FIRST_TO_HOSP _DATETIME og FIRST_HOSP_ARRIVAL _DATETIME. Aktiviteten for når en case blir åpnet og avsluttet er INCIDENT_DATETIME OG INCIDENT_CLOSE_DATETIME.

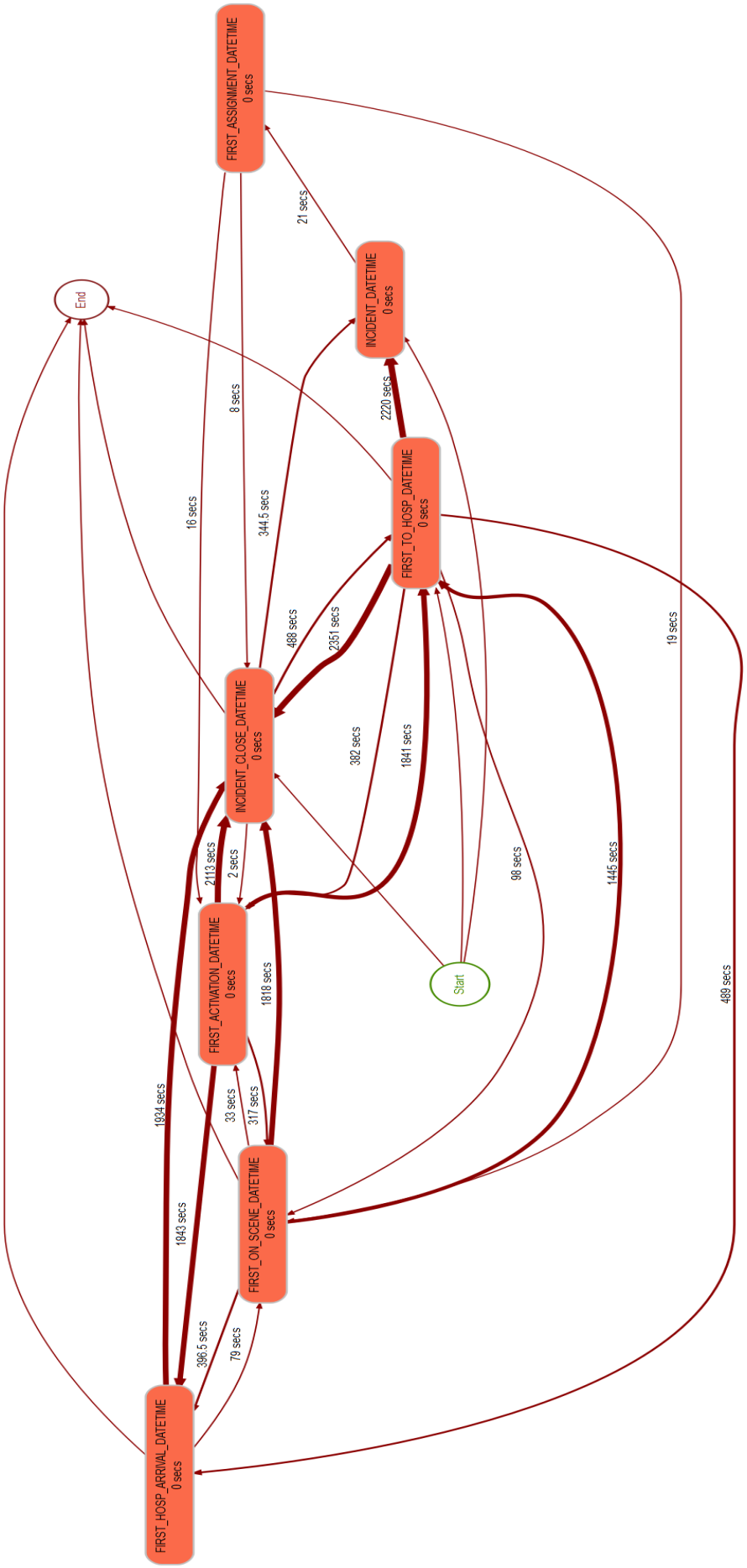
Det fremkommer fra figur **4.1** at for hver aktivitet, så ankommer det maks én uthevet strek, og det utgår maks en uthevet strek. Alle uthevede streker er majoritetsruten. Alle de tynne strekene er minoritetsruter, som indikerer mulig avvik fra prosedyre eller feilregistrering i system. Et problematisk eksempel vi ser fra figuren er for eksempel flere caser som går fra aktiviteten, INCIDENT_CLOSE_DATETIME, til andre aktiviteter. Ideelt sett så skulle det bare vært en rute for INCIDENT_CLOSE_DATETIME, til "End". Dette er bevis for at det har vært en form for feilregistrering av tid, da denne aktivitetens rolle er å registrere det siste tidspunkt for en case. En ser også at det er tre caser som går fra start til den nevnte aktivitet.

Når det gjelder FIRST_HOSP_ARRIVAL_DATETIME, som indikerer når en ambulansen er fremme på sykehus, observeres to minoritetsruter fra aktiviteten. Den ene ruten går direkte til slutt (end), den andre ruten går til aktiviteten FIRST_ON_SCENE_DATETIME, som indikerer tid for oppmøte der en pasient er lokalisert. I dette tilfellet så er dette feil rekkefølge, da FIRST_ON_SCENE_DATETIME logisk burde komme først. I det andre tilfellet så har 42 caser blitt fullført, uten å ha blitt registrert i den siste aktiviteten. Dette er med på å underbygge observasjonen om at dataene indikerer en grad av feilregistrering. Relatert til minoritetsruter, ser det ut til å være fravik fra majoritetsruten. Majoritetsruten har ikke blitt fulgt i alle tilfeller. Det er interessant at visse ruter fremtoner seg på en ulogisk måte. I hendelsesloggen eksisterer det en aktivitet, der dens eneste hensikt er å registrere når en case er ferdig (INCIDENT_CLOSE_DATETIME). Det har seg slik at denne aktiviteten ikke er sist i alle tilfeller, da det fremkommer flere ruter til end-symbolet.

Fuzzy prosessmodell - Tidsperspektiv

Det fremkommer fra figur **4.2** en prosessmodell, der hendelsesruten for aktivitetene er markert med sekunder. Dette er gjennomsnittstiden til en aktivitet fra hendelsesloggen. Sammenligner vi med første (blå) prosessmodell, så er det lite som overlapper med hensyn på majoritetsruter. Fra de to prosessmodellene ser det ut til å være en reversering av majoritetsrutene. Majoritetsruten i den blå prosessmodellen, er den ruten som er minoritet i den røde modellen[3]. Minoritsetrutene i den blå modell, er de ruter med høyest tidsbruk,

[3]Majoritetrute i røde modell er uthevet rute, som indikerer majoritet av tidsbruk. Minoritet er lavest tidsbruk.



Figur 4.2: Fuzzyminner - Tidsperspektiv.

i den røde modellen. Majoritetsrute i den røde modellen, ser ut til å være de rutene i den blå modellen som var av minoritet.

Alfa Pluss

Figur **4.3** viser et petri net, som alfa algoritmen har produsert i BupaR pakken fra R. Petri net prosessmodellen skal tolkes ovenfra og ned. Figuren starter på start og stanser på end. Et objekt (Petri nets ulike notasjoner) følger et annet. Det første som bør poengteres er at alle aktiviteter har blitt inkludert i algoritmen, der aktiviteter er representert av figuren som firkanter. Det er totalt tjueen punkter i modellen. Fra Petri net observerer vi tretten sirkler, disse sirklene beskriver hvilken vei en case kan ta med hensyn på aktiviteter. De ulike veiene fra notasjonen er enten split eller joins, med hensyn på enten XOR eller AND.

Fra Petri net observerers bare AND notasjon. Dette sees da det ikke forekommer en splitt ut ifra en sirkel (place), hvor en sirkel har maks et utløp til en transition (Aalst, 2016, s.169). Det observeres også AND split (til sirkel), men da er det utelukkende fra transition. Det er fravær av XOR joins/splits, noe som kan indikerer mangel på valgmuligheter, enten eller scenarioer. Det er ikke tegn til at alfa algoritmen har registrert slik type aktivitet i hendelsesloggen[4].

4.1.2 Conformance Checking

Her presenteres resultatene fra conformance checking. Der måltallene skal belyse forholdet mellom hendelsesloggen og den produserte Petri net modellen, i figur **4.3**.

Alpha Pluss - Conformance Checking

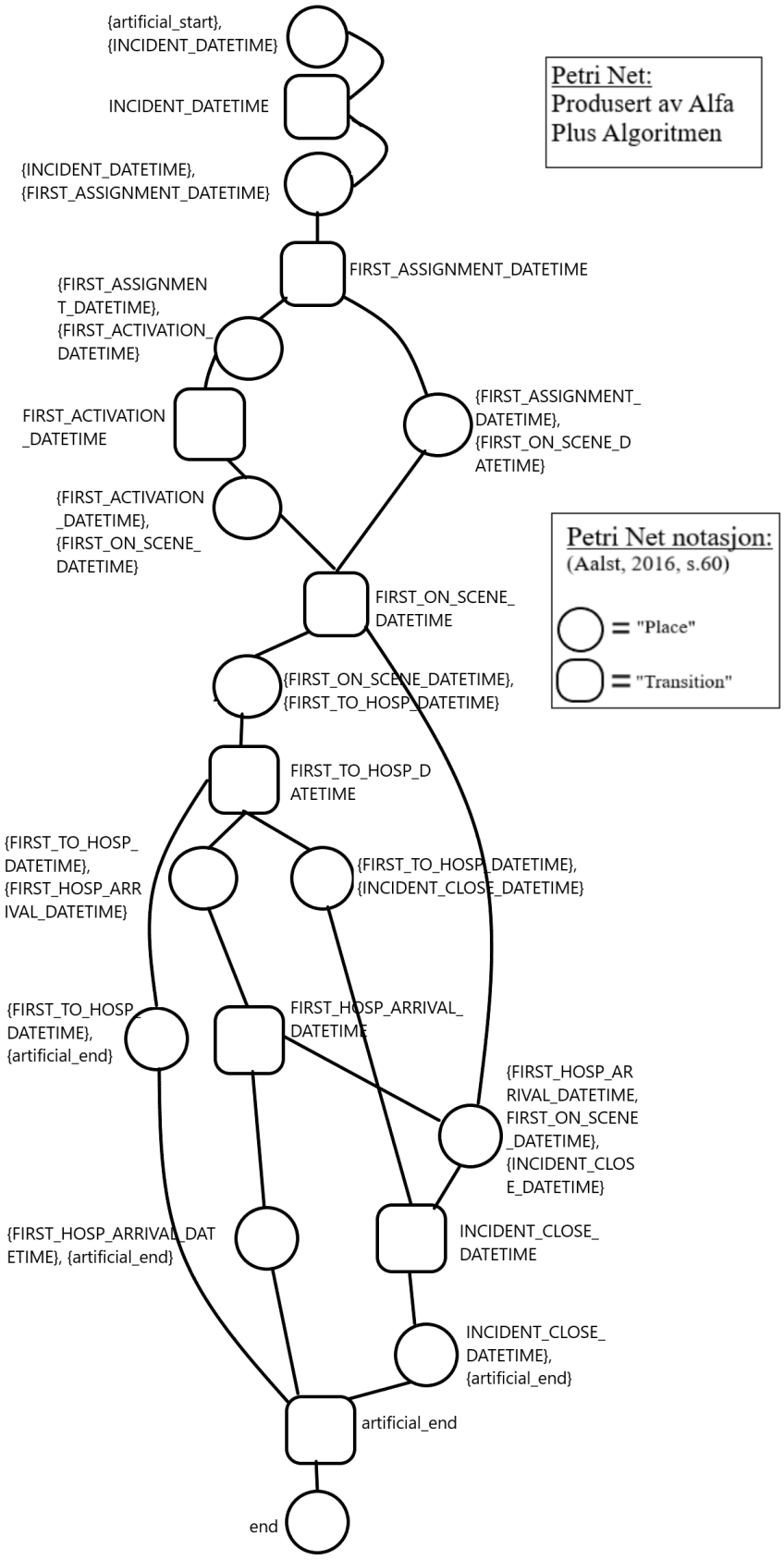
Fitness (average trace)	Generalization	Precision	Simplicity	Gjenomsnitt
0,893	0,998	1,000	0,724	0.9038

Tabell 4.1: Presisjonsmåltall - Alfa Pluss.

Fitness

Det fremkommer fra måltallet, fitness, at Petri net modellen klarer å forklare 89.3%

[4]Dette behøver ikke å være problematisk, men er forståelig i lys av data grunnlaget. Aktiviteter i prehospital-klinikk er strømlinjeformede, og en aktivitet følger sekvensielt en annen. Det lave antall mulige aktiviteter underbygger dette, samt resultatene fra fuzzy algoritmen.



Figur 4.3: Petri Net fra Alfa Algoritme Pluss.

av alle traces. Forklaringskraften til prosessmodellen er høy[5] når det kommer til dette måltallet. 100% Betyr at alle traces hadde vært inkludert i prosessmodellen, som alfa pluss algoritmen har produsert.

Precision

Måltallet som taler om presisjon, endte opp på 100%. fra teorien vil dette bli forstått som om at modellen ikke har underfitting.

Simplicity

Simplisitets kriteriet gir en indikator på hvor simpel prosessmodellen er. I dette tilfellet observeres det en score på 72,4%. Relativt til alle de andre måltallene, så er måltallet noe mindre, uten at det nødvendigvis er direkte sammenlignbart.

Generalization

Relatert til modellens evne å generalisere, observeres det et tilnærmet perfekt resultat på over 99,8%. Forklaringskraften 99,8%, relatert til å generalisere hendelsesdata, kan sees i lys av hvor uniforme dataene ser ut til å være, særlig i kontekst av resultatene i fuzzy prosessmodell. En observerer i fuzzy prosessmodellen en liten andel som divergerte fra minoritetsrute. Fra teorien (Aalst, 2016, s.272) tilsier resultatet fra prosessmodellen (Petri net), at modellen evner å kunne generalisere, på fremtidig data. Det indikeres at det ikke er noe overfitting.

Modellen totalt sett virker robust i lys av måltallene, med noe lavere simplicity. Modellen har klart å fange opp alle aktiviteter. Snitt score på 90.38% virker heller ikke uvanlig høyt, gitt det faktum at loggen har bare har syv aktiviteter som en case kan ta[6]. Fra figur **4.3** eksisterer bare AND veier/muligheter, som en case kan ta. En case må innom alle aktiviteter. Dette ser ut til å korrespondere bra med tanke på at en case i snitt har brukt 6.7 aktiviteter av totalt 7 mulige[7]. Likevel betyr det at noen caser ikke blir representert i prosessmodellen (Petri net). Det har blitt gjort en avveining fra alfa pluss, med å ikke inkludere alle mulige traces.

4.2 Maskinlæring

Her presenteres ML modellene som ble utledet i metode.

[5]Ikke nødvendigvis ensbetydende med bra.
[6]Buijs studien fikk snitt score på godt over 90%, med syv aktiviteter i logg (2012).
[7]**Vedlegg II.1**

4.2.1 Trening av Modeller

Ridge	
Hyperparameter	Verdi
Lambda	37.9269

(a) Ridge.

Lasso	
Hyperparameter	Verdi
Lambda	0.04281332

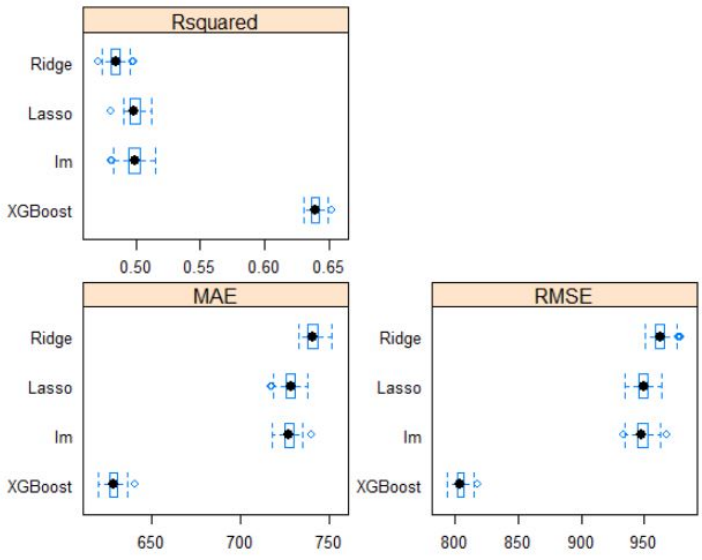
(b) Lasso.

XGBOOST	
Hyperparameter	Verdi
Alpha	0.03879761
Eta	0.7056409
Lambda	0.3648002
Nrunder	92

(c) XGboost.

Tabell 4.2: Optimale Hyperparametere.

Tabell **4.2** viser den optimale hyperparameter sammensetningen i de ulike modellene[8]. Som nevnt tidligere ble repetert 10-fold kryssvalidering brukt i tandem med grid og eller tilfeldig søk. Dette for å finne de optimale hyperparameterene. Fra figur **4.4** ser en hvordan presisjonesmålene under trening av modellen[9], gitt de optimale hyperametrene har beveget seg [10]. Det fremkommer ved trening av XGboost, at alle presisjonsmålene er bedre enn de andre tre modellene[11]



Figur 4.4: Presisjonsmåltall fra Repetert Kryssvalidering.

[8]Merk at MLR ikke er i tabellen, det er som følge av at MLR anser konstantledd som hyperparameter. Konstantleddet er holdt konstant over hele modell treningsfasen. Variasjonene som fremkommer i figur **4.4** angående MLR er kun refleksjon av tilfeldig varians, gitt ulike observasjoner i testfold under trening.

[9]Man ser fra Figur **4.4** at måltallene beveger seg i et lite spenn, som styrker antagelsen om at observasjonene er proporsjonalt fordelt i testdata. Dette ble nevnt avsnitt 3.3.5 under datafordeling.

[10]Dette innebærer 10*10 = 100-testfold resultater(resamples).

[11]Tolkningen av boksplott er det samme som forklart tidligere.

4.2.2 Modell Kandidater

Tabell **4.3** viser til testsdata resultater for de ulike modellene. I tabellen fremkommer måltallene MAE, R^2 og RMSE. Fra tabellen er alle modellene som er produsert presentert.

Lineær regresjon	
Fitstat	Sum
MAE	722.30
RMSE	939.79
R^2	0.5059

(a) Baseline (MLR).

XGBOOST	
Fitstat	Sum
MAE	627.15
RMSE	799.92
R^2	0.6422

(b) XGboost.

Ridge	
Fitstat	Sum
MAE	735.35
RMSE	952.78
R^2	0.4931

(c) Ridge.

Lasso	
Fitstat	Sum
MAE	722.76
RMSE	939.75
R^2	0.5060

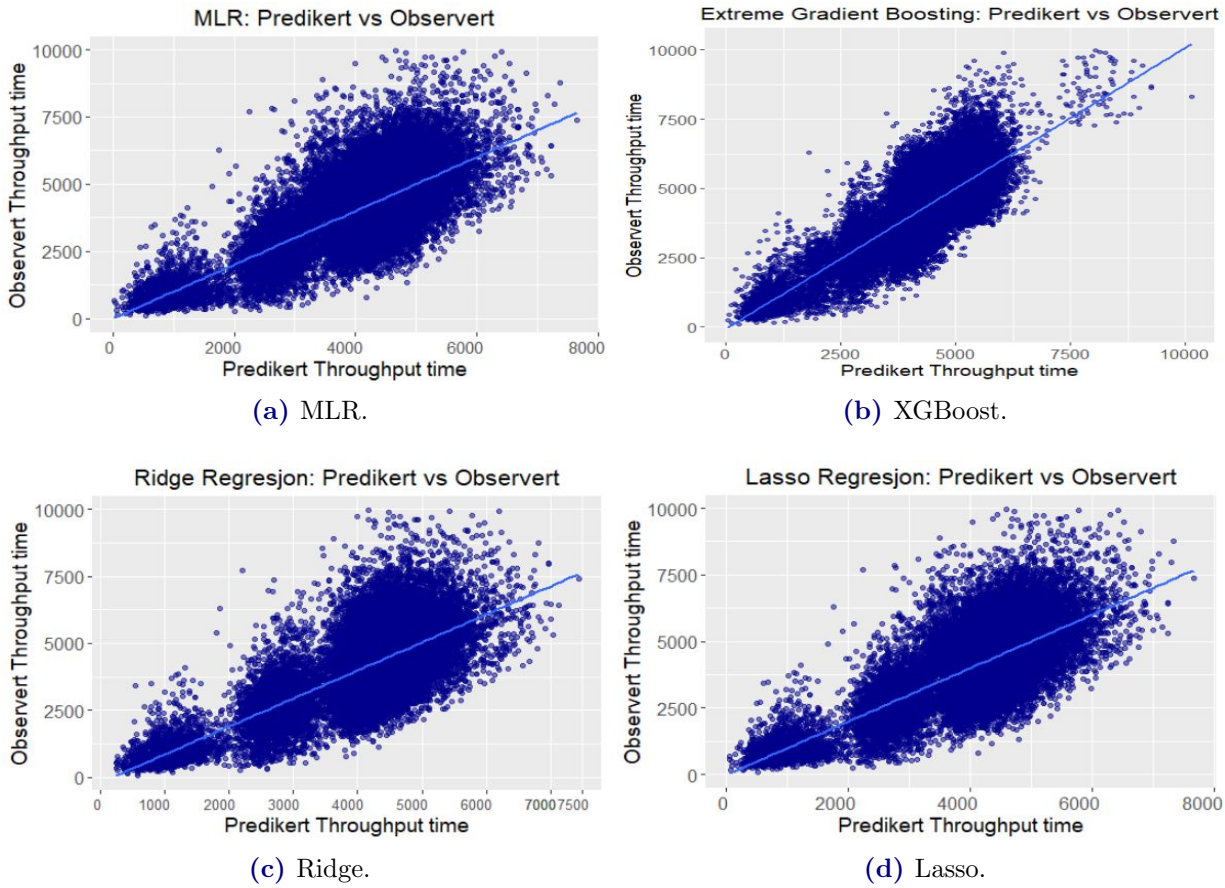
(d) Lasso.

Tabell 4.3: Presisjonsmål til Modellene.

- MAE: Hovedkriteriet for å velge den beste modell er MAE. Vinner modellen er XGBoost, med lavest MAE på 627.15. De tre andre modellene har tilnærmet like verdier, der ridge har høyest med 735.35. Lasso og LM (MLR), har nesten lik MAE, rundt 722.
- RMSE: RMSE er det kvadrerte gjennomsnittlig avvik. RMSE er lavest i XGBoost, med 799.92. Måltallet er størst i ridge. I Lasso og MLR, er RMSE tilnærmet like store med 939.80.
- R^2 : XGBoost forklarer mest av variasjonen med ca R^2 på 64%. Lasso, MLR og ridge er rundt 50%.

4.2.3 Modell Statistikk

Her presenteres statistikk opp mot modellkandidatene, med fokus på vinner modellen - XGBoost.



(a) MLR. **(b)** XGBoost.

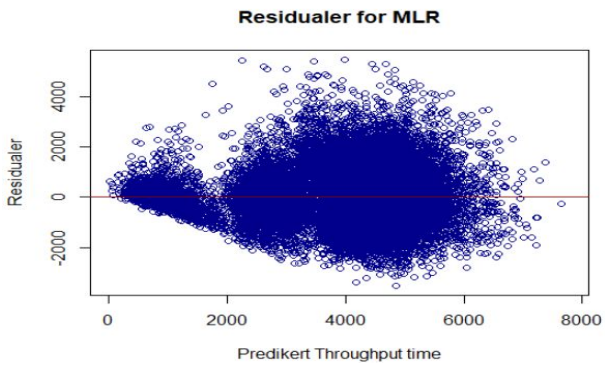
(c) Ridge. **(d)** Lasso.

Figur 4.5: Observert vs Predikert Verdier.

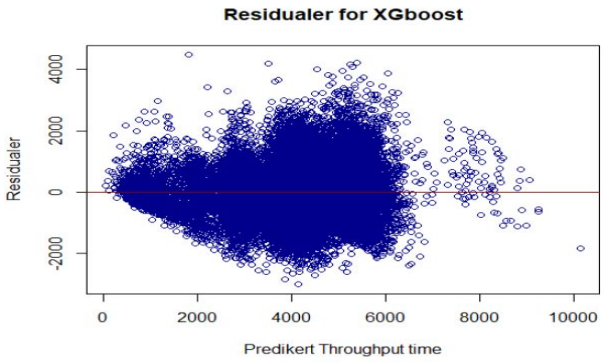
Fra figur **4.5** ser vi blå prikker som symboliserer de predikerte verdier for gjennomstrømstiden. Dette er sett opp mot faktiske observerte gjennomstrømstid[12]. Y-aksen symbolisere faktiske observerte, der X-aksen symboliserer predikerte verdier[13]. Det kan ses fra figuren at når x-aksen (prediksjoner) er eksakt lik med Y-aksen (virkelige verdier), vil man treffe den blå linjen. Den blå linjen symbolisere den optimale modellen. Den Optimal modell er hva man strekker seg etter, den referer til den modellen som predikerer 100 % riktig, og fungere som et referansepunkt for hvor god modellen er. Fra figur **4.3**, fremkommer det at MAE er minst for XGboost, dette ser man tydelig fremkommer i figur **4.5**, hvor man observere langt mindre spredning rundt den optimale blå linjen.

[12]Gjennomstrømstid i testsettet som var på 20 % av totale datasettet.

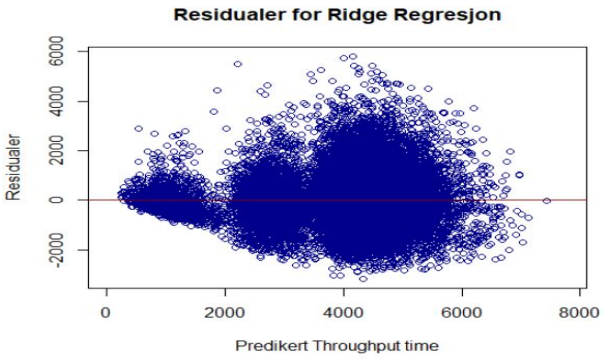
[13]Man ser fra Y-aksen at den strekker seg fra verdi 0-10000, som er nøyaktig hvilke rekkevidde som ble spesifisert for gjennomstrømstid ifra datavask.



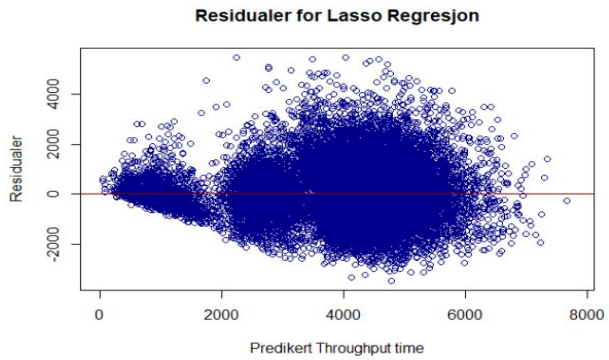
(a) MLR.



(b) XGBoost.



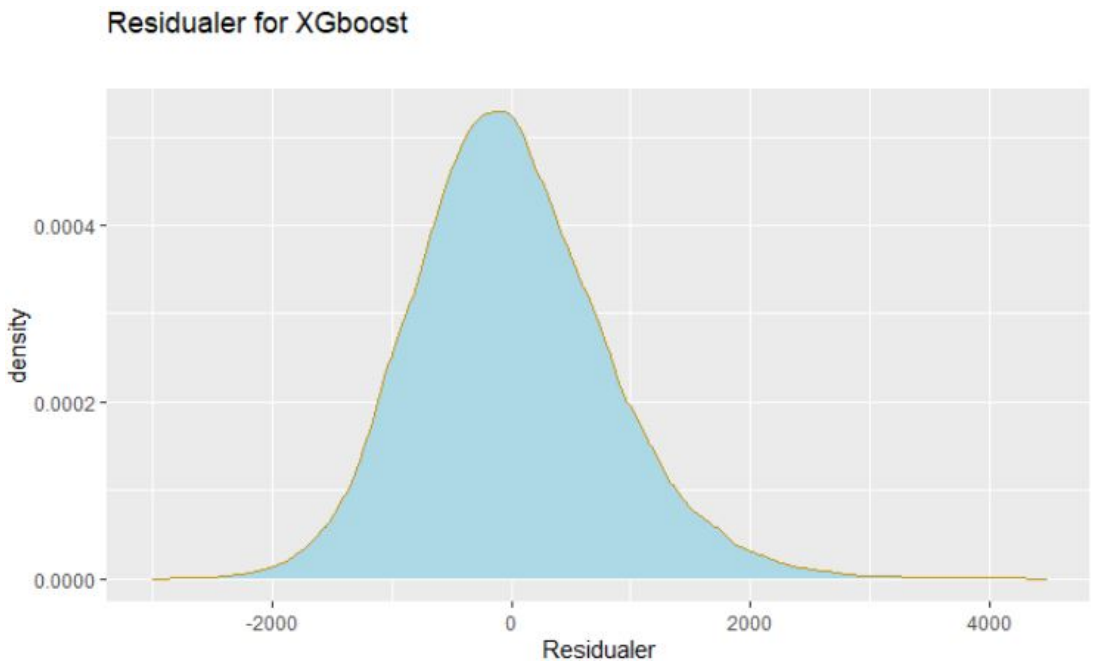
(c) Ridge.



(d) Lasso.

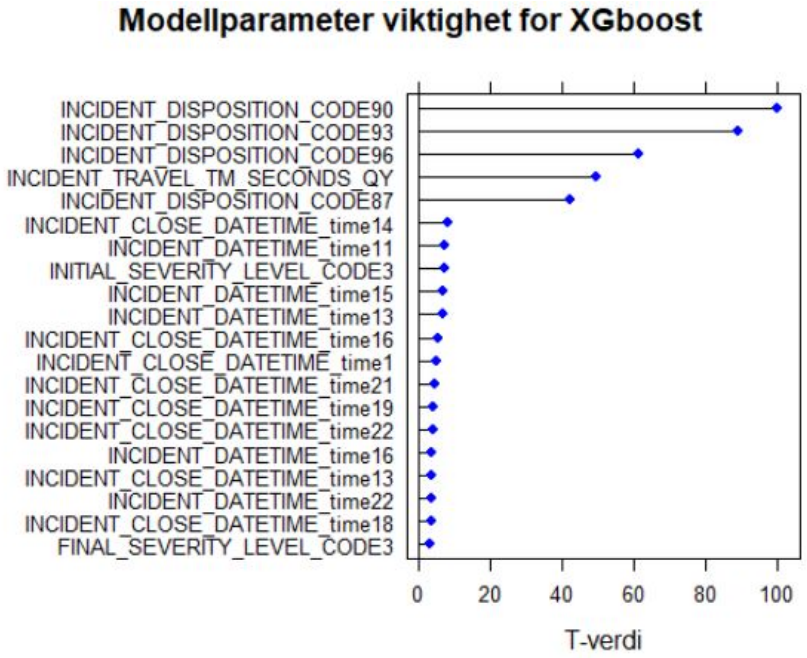
Figur 4.6: Residualer fra Testdata.

Residualene i figur **4.6** viser grad av feil estimering opp mot gjennomstrømstid. Størrelsen til residualene er gitt på y-aksen. Residualen er differansen mellom historiske og estimerte data. De predikerte data er på x-aksen. Hvor større spredning er i residualene, hvor dårligere er modellen på å predikere. Fra de fire figurene observeres stor likhet mellom MLR, ridge og lasso, i relasjonen mellom residualer og predikert gjennomstrømstid. Fra disse modellene kan det sees ut som om at det er klynger i dataene fra 0 til 2000 sekunder. Dette fenomenet er ikke like tydelig i XGBoost, der dataene er mer tettpakket. XGBoost er også alene i å predikere verdier over 8000 sekunder, noe som kan indikere at modellen er bedre i å fange opp unormalt høye verdier. Utover dette så ser modellene ut til å ha relativt like spredning av residualer, over og under verdien null på y-aksen.



Figur 4.7: Densityplot for XGboost Residualer.

En ser fra figur **4.7** at residualene er nærliggende normaltfordelt. Det indikeres at det er en marginal høyere andel residualer som er negative. Av de største residualen, så er de positive, da x-asken viser positive reisdualer opp mot 4000 sekunder.



Figur 4.8: Modellparameter Viktighet for Vinner Modell.

Figur **4.8** viser til topp tyve modellparametere fra XGboost. Parametrene er rangert etter viktighet. Modell parametrenes viktighet er kalkulert ut ifra T-verdi. T-Verdi forteller om hvor statistisk signifikans modellparametrene er. Høy T-verdi heller over mot høy grad av viktighet og statistisk signifikans (Sucarrat, 2016, s.21-23). Det som er viktig å påpeke,

er at statistikk signifikans forklarer bare om modellparameteren har effekt (signifikans) på den totale gjennomstrømstid. Den viser ikke til hvor stor effekten er for prediksjon.

Fra figuren fremkommer det at INCIDENT_DISPOSITION_CODE variabelen innehar flest modellparametere med høy signifikans. Fra vedlegg I.8 ser en kodenes betydning i variabelen, variabelen viser utfallet til en case. Kode 90 fra vedlegget betyr "ubegrunnet", noe som kan forstås i kontekst av at en ambulanseenhet ikke får gjennomført oppdraget. 93 betyr at pasient motstår hjelp. 96 betyr at pasient er ikke tilstede når ambulanse møter opp. 87 betyr at oppdrag er avbrutt. INCIDENT_TRAVEL er også ansett som viktig, dette omhandler reisetid. Av de resterende er tidsvariabelen for tid på døgnet ansett som viktig. Disse funn belyses i diskusjon.

Kapittel 5

Diskusjon & Konklusjon

5.1 Process Mining & Problemstilling

Problemstilling 1 for PM ble formulert slik: **Problemstilling 1: Vil process mining kunne lage en god prosessmodell av NYC hendelsesdata?** Problemstillingen nevner først PM, som er metoden i studien. Videre i problemstillingen er ønsket å belyse om prosessmodellen er god. Dette i Konteksten av NYC hendelsedata.

For å kunne belyse om hvorvidt modellen er god eller ikke, ble måltallene fitness, simplicity, precision og generalization utregnet. Måltallene estimerer ulike aspekter mellom loggdata og den produserte prosessmodell. Der den produserte prosessmodell er et Petri net. I innledningen ble det påpekt at en god prosessmodell er kontekstavhengig[1]. Dette belyses primært ved bruk av de nevnte måltall, men også ved en visuell vurdering av et Petri net. I denne studien må det også understrekes viktigheten av å forstå datagrunnlaget. Dataene var ikke originalt i hendelseslogg format, men i tverrsnitt format. Dette innebærer at vi ikke har direkte sammenligningsgrunnlag med andre studier, som har en hendelseslogg som utgangspunkt[2]. Tidligere studier vil kunne bli brukt indirekte for referanse, hvis det er skjellig grunn for det.

Datagrunnlaget til modellen var en hendelseslogg med aktiviteter og caser. En case kan ta på seg maks syv aktiviteter. I dataene var det loggført 215 066 caser og 1 443 872 aktiviteter. Loggen var begrenset til fire variabler. Dette var en begrensning i dataene før loggen ble produsert. Med flere variabler kunne mer informasjon om aktivitetene blitt belyst.

Petri net & Måltall

Det ble produsert prosessmodeller ved å bruke fuzzy og alfa algoritmen, der bare Petri

[1]Kontekst kan eksempelvis forstås i lys av bransje, arbeidsoppgave kompleksitet, algoritme valg og datagrunnlag.

[2]Metoden ugyldiggjøres ikke av dette, da dataene som er brukt for PM, er fullt kompatible med metoden. Se **Vedlegg II**.1.

net med tilhørende måltall brukes i besvarelsen av problemstillingen[3]. Petri net modellen ble produsert ved hjelp av alfa pluss algoritmen. I prosessmodellen ble alle syv aktiviteter visuelt representert. Dette er en styrke ved prosessmodellen, da viktig informasjon som aktiviteter ikke har blitt utelatt. Videre viser prosessmodellen bare overganger (transitions) i form av AND notasjoner. AND notasjon kommuniserer at en case må utføre alle aktiviteter før videre progresjon. Motsetningen er XOR, som kommuniserer at en case har valgmulighet mellom to eller flere aktiviteter. XOR notasjon er ikke å finne i modellen.

Petri net har kun brukt AND notasjonen. På basis av datagrunnlaget kan det se ut til å være et rett valg, fra alfa pluss algoritmen. Dette fordi aktivitetene er logisk anordnet[4]. Dataene har til hensikt å vise når en aktivitet har inntruffet. I stor grad skjer en aktivitet etter en annen. På basis av dette er det få caser som utviser annen atferd, enn hva majoriteten av caser utviser. Slik atferd sees også i fuzzy prosessmodellene, der det fremkommer at majoritet av caser følger den samme rute. Denne ruten er også moddelert i Petri net[5]. Petri net har klart å modellere hvilke aktiviteter som majoritet av caser er tilknyttet. Dette er en viktig egenskap. Petri net har også klart å utelukke minoritetruter (som ble fremhevet blant annet i fuzzy prosessmodellen)[6]. Dette er positivt fordi en ønsker at prosessmodellen skal kunne generalisere på fremtidige data. Dette blir fanget opp i måltallet generalization.

Petri net sin evne til å modellere majoriteten av atferd tilknyttet casene, underbygges av måltallenes presisjon. Måltallene har snitt score på 90.38%. Modellen utviste særdeles høy precision, og maksimal generalization, med henholdsvis 99.8% og 100%. Fitness og simplicity var noe mindre med respektive 89.3% og 72.4%. Hvorvidt dette er et godt resultat er ikke sikkert, selv om tallene er i snitt høye (snitt score 90.38%.). Fra den visuelle vurderingen så kan en observere at modellen har fanget med mange viktige elementer fra dataene. Angående måltallene, er det lite å sammenligne med når det gjelder studier med tilsvarende datagrunnlag. En studie (Buijs et al., 2012) brukte en hendelseslogg med syv aktiviteter, som en case kunne være tilknyttet. Det er like mange aktiviteter som eksisterer i hendelsesloggen for denne studien. Snittscore i Buijs et al (2012) studien var på 96.9%. Dette er noe høyere enn resultatet for alfa pluss algoritmen i denne studien.

[3]Prosessmodellene som ble produsert ved hjelp av fuzzy algoritmen var brukt for eksplorativ data-analyse.

[4]Det er bare syv aktiviteter å velge mellom, der majoriteten av aktivitetene har brukt alle syv tilgjengelige. Se også vedlegg angående aktivitetenes anordning i prosessene. **Vedlegg II**.1.

[5]Ruten som følges er, *INCIDENT_DATETIME, FIRST_ASSIGNMENT, FIRST_ACTIVATION, FIRST_ON_SCENE, FIRST_TO_HOSP, FIRST_HOSP_ARRIVAL, INCIDENT_CLOSE.* Dette fremkommer også i variabel beskrivelsen, i **Vedlegg I**.2.

[6]I hendelsesloggen eksisterer det en aktivitet, der aktivitetens eneste hensikt er å registrere når en case er ferdig *(INCIDENT_CLOSE_DATETIME)*. Fra fuzzy prosessmodellen ser en at dette ikke var faktum, men at også andre aktiviteter hadde vært registrert som siste aktivitet i en case. *INCIDENT_CLOSE_DATETIME* er altså ikke blitt brukt konsistent for å signalisere at én case er ferdig. Det ser ut til at Petri net har gjort en korrekt generalisering i dette tilfellet, da slike fenomener ikke eksisterer i prosessmodellen.

Buijs et al (2012) brukte standard varianten av alfa algortimen[7]. Det er likevel utfordrende å sammenligne resultater, da dataene var basert på lånesøknader (Buijs et al, 2012).

Tilnærmet 100% ble resultatet for precision, og 100% for generalization. Disse resultatene innebærer at prosessmodellen utviser liten grad av overfitting/underfitting. Resultatet ser ut til å korrespondere bra med det vi ser fra Petri net modellen. I Petri net modellen er de mest høyfrekvente veiene, som en case kan ta, blitt inkludert. Det ser ut til at prosessmodellen evner å generalisere. Dette fremkommer når en sammenligner Petri net, med fuzzy modellen. I fuzzy modellen ble lite frekvente veier en case kan ta, modellert. Det indikerer at Petri net har gjort en korrekt avveining i lys av disse to måltallene, der ikke for mye (overfitting), eller for lite (underfitting) er tatt med i modellen. Selv om Petri net har ekskludert ett fåtall av aktivitetskombinasjoner (traces), så har likevel Petri net modellen inkludert 89.3% (fitness) av alle traces. Dette innebærer at modellen har liten overfitting og underfitting, uten å ekskludere mer en cirka 11.7% av alle traces. Høy fitness virker å være logisk gitt antall aktiviteter, og hvor strømlinje formede arbeidsoppgavene i ambulansetjenesten er. Måltallet simplicity skiller seg noe ut, da det er det eneste måltallet med score under 80%, med respektive 72.4%. Måltallet er noe lavere enn snitt score, der det indikerer at modellen har potensiale til å kunne fremstille dataene enklere[8].

5.1.1 Konklusjon

Modellen scorer i snitt 90.38%, noe som indikerer at prosessmodellen har fanget opp mye av dataene i hendelsesloggen. Relatert til valg av notasjon (symboler) i Petri net, indikeres det at valg av bare AND notasjon, virker til å være et godt valg gitt datagrunnlaget. Petri net modellen indikeres samtidig å ha potensiale til å forenkles. I besvarelsen av problemstillingen indikeres det at prosessmodellen har fanget opp flere viktige elementer i dataene. Der modellen utviser flere gode trekk, når det gjelder fremvisning av aktiviteter i ambulansetjenesten. Manglende sammenligningsgrunnlag av andre studier gjør det vanskelig å trekke flere konklusjoner enn dette.

5.2 Maskinlæring & Problemstilling

Problemstilling 2 for PM og ML ble formulert slik: **Problemstilling 2: Vil process mining og maskinlæring kunne predikere gjennomstrømstid, i NYC hendelsesdata?** Problemstillingen nevner først PM og ML, som er metodene i studien. Videre i problemstillingen er ønsket å belyse om PM og ML kan predikere gjennomstrømstid. Konteksten er NYC hendelsedata.

[7]Ikke alfa pluss algoritmen, men den ordinære alfa algoritmen
[8]Modellen kan muligens forenkles ved å bruke en annen algoritme.

5.2.1 Datagrunnalg, Datavask og Produksjon av Variabler

Data som ble brukt i oppgaven kommer fra den offentlige databasen OpenData. Dette er data som har blitt loggført av FDNY EMS, helsevesenet i NYC. Datagrunnlaget er med på å øke troverdigheten til dataanalysen. Datavask før ML, ble gjennomført i tråd med det ble nevnt i metode. "I preprosessering av data er det tatt særskilt hensyn til det faktum at det er hendelsesdata fra helsesektoren. Slike data bør behandles med stor respekt". Fjerning av outliers ble gjennomført i tråd med gjeldende normer fra (Columbia, FEMS.)[9]. Videre ble kategorier med minimalt få observasjoner fjernet (eksempel en observasjon)[10]. NAs ble eliminert med hensyn på FEMS (2016, s.45-47) retningslinjene[11]. Det ble valgt å ikke tilegne (imputaition) nye verdier i tilfellet av NAs, grunnet ønsket om å gjennomføre datavask, med det hensyn til at det arbeids med sensitiv helsedata. Om tilegnelse av NAs hadde blitt andvendt kunne mer av datasettet informasjon blitt bevaret, noe som kunne ha hjulpet ML modellene å finne flere mønstre i datasettet. En negative konsekvens, ligger i feilaktig tilegnelse av NAs. Der feilaktig tilegnelse vil kunne forsterke feil i den endelige modellen.

I produksjon av nye variabler tok vi utgangspunkt i allerede eksisterende variabler som var i dataene. Variablene som ble produsert søkte å fremheve tid[12] når aktiviteter inntraff, og hvilke aktiviteter som var tilknyttet en unik case[13]. Dummy variabler for hvilke aktiviteter som var tilknyttet en case ble inkludert i lys av PM. Der PM har til hensikt å bringe frem prosessinformasjon om aktiviteter, hvor vi søkte å inkludere tid i ML modellen. Vi anser variabel produksjonen som korrekt i lys av hvor orientert oppgaven er rundt aktiviteter og prosesser.

Årsaken til at variabler ble fjernet, var på grunn av høy korrelasjon mellom to eller flere variabler. Dette ble belyst i chi-square ved hjelp av Cramers-V test[14] og korrelasjonsmatrisen[15]. Rasjonale for å bruke disse metodene var for å fjerne variabler som kunne forårsake multikollinearitet. Om ikke dette hadde blitt gjort, ville det kunne ha uheldige konsekvenser, da modellen ville kunne få en perfekt samvariert variabel. Dette kan inkludere støy i modellen. Hvor støy vil kunne føre til overfitting av modellen.

5.2.2 Bias og Varians Trade-Off

Som nevnt tidligere (se formel **2.2**) kan total feil til en ML modell måles i fravikelig feil og ufravikelig feil. Der fravikelig feil er satt sammen av elementene bias og varians. Fravikelig

[9]Se vedlegg I.4. Note 44-46.
[10]Se vedlegg I.4. Note 47-48.
[11]Se vedlegg I.4. Note 1-5.
[12]Se vedlegg I.4. Note 11-16.
[13]Se vedlegg I.4. Note 6-10.
[14]Se vedlegg I.4. Note 27-43.
[15]Se vedlegg I.4. Note 49.

feil er feil det er mulig å minimere, der ufravikelig feil ikke kan minimeres. Først drøftes synspunkter opp mot fravikelig feil og hva som mulignes kunne bedre modellen. Siste del legger vekt på synspunkter angående ufravikelig feil, og om hvorvidt nye variabler kunne vært inkorporert, som prediktorer, for å bedre modellens presisjonskraft.

K-fold repetert kryssvalidering ble brukt i tandem med grid og eller tilfeldig søk. Hensikten for bruk av disse metodene var for å minimere varians (overfitting), slik at den endelige modellen kan kunne generalisere på ny usett data. Det kom tydelig frem i resultatet av de ulike modellene som MLR, Ridge og Lasso ga nærliggende like pressisjonsmåltall[16]. Rasjonale for dette ligger i at hyperparameteren var Lambda tilnærmet lik null[17]. Dette indikerer at modellen har lav grad av kompleksitet, ettersom at nesten ingen krympeeffekt har blitt andvendt. Det ser ut til å være lite som trengs å bli gjort, for å motarbeide overfitting. Lav lambda reflekter derav en mulig indikator på lav varians i treningsdata.

I tilfellet av vinner modellen XGboost ser man fra figur **4.4** at det heller ikke er tilfellet av høy varians. Det sees fra over 100 testiterasjoner[18], da det ikke er et stort spenn i de ulike testfold resultatene[19]. Dette belyser at modellen gjør en god jobb på å generalisere. Som man kan se er resultat mellom test (tabell **4.3**) og trening (figur **4.4**) tilnærmet lik. Modellen har lært mønstrene i datasettet, uten at den inkluderer for mye støy fra treningssettet. Dette signaliserer at høy kompleksitet som følge av for mange modell parametere ikke er et problem. Modellen klarer dermed å generalisere funn på tvers av testdata.

Det ser ut til at det eksisterer en grad av bias i modellene, med minst bias i XGBoost (lavest feilprediksjon i XGBoost.). Med bias henvises det til at modellene ikke lært mønstrene i treningsdata godt nok. Modellens gjennomsnittlige prediksjoner er skjev iforhold til de faktiske sanne observerte verdier[20]. Dette medfører at modellen oversimplifiserer, og har i større grad underfitting. Flere apsekter kan drøftes for årsaken til høy bias. Det kan for eksempel skyldes av for få modellparametere. Algoritmene som er andvendt kan også være en årsak, som følge av at forholdet i datasettet ikke er lineært[21]. En annen årsak, kan ligge i at datasettet er for lite.

Den siste type feil er ufravikelig feil. Årsaken kan være mangel på variabler i datasettet. Det kan tenkes at flere variabler kunne vært introdusert som gode modellparametere, for å ytterligere forklare det underliggende fenomenet. Hvilke variabler dette kunne ha vært er spekulativt. Det å forstå hvilke variabler som skal inkluderes for å forklarer den avhengige

[16]Om krympeparameteren lambda er 0, gir MLR like svar som ridge og lasso.

[17]Lambda har som hensikt å krympe koeffisienter.

[18]Over disse test iterasjonene, er de optimale hyperparametrene valgt.

[19]Som nevnt tidligere, ble all preprosessering av data (Variabel transformasjon) gjort på hvert individuelle fold, slik at datalekk er minimert, og dermed også optimaliseringsbias. Dette sikrer troverdige funn i henhold til varians og overfitting.

[20]Se figur **4.5**.

[21]Alle algoritmene som er andvendt måler lineære forhold mellom den avhengig variabelen og de uavhengige. En mulighet hadde vært å prøve ikke-lineære algoritmer.

variabel, er en av de største utfordringene i ML. Desto viktigere blir bransjeekspertise.

5.2.3 Måltall og Modell Statistikk

Måltall som ble valgt for å velge den endelige modell, var MAE. Fordelen med MAE er enkeltheten ved tolkning. Ved trening av modell ble optimaliseringskriteriumet satt til MAE, der modellen skulle minimere MAE. Den endelige modell som ble utvalgt på basis av MAE, var XGBoost. XGBoost fikk en MAE på 627.15. Betydningen av MAE er at modellen estimerer feil med ca 627 sekunder i snitt. Alle de andre modellene hadde en MAE mellom 722 og 735. Dette betyr at XGBoost har cirka 100 sekunder lavere feil estimering. Vi bruker ikke RMSE, da den straffer outliers i større grad enn MAE. Vi ønsker ikke å vektlegge en case fremfor en annen. Alle residualene behandles derfor likt ved bruk av MAE. Dette mener vi er ønskelig i bruk opp mot ambulanse data. Modellens optimaliseringskriterium er basert på MAE, da vi mener at det å behandle residualene likt, kan skape troverdige modellparametere. Det å bruke MAE som optimaliseringskriterium mener vi er egnet løsning for å predikere total gjennomstrømstid. Derfor ble også MAE brukt som hovedkriterium i utvelgelsen av vinner modell XGBoost.

Residualene til vinner modellen er mer sentrert enn de tapende modeller. XGBoost utviser størst grad prediktiv evne fra MAE. Det kan sees fra figur **4.5** og **4.6**. Fra spredningen i resdual plottene er XGBoost den beste modell, med tettest spredning i residualene. Dette gir mening i lys av MAE, der XGBoost klarte å predikere gjennomstrømstid med minst feil. XGBoost i residual plottene utviser størst grad av konstant varians.

T-verdien belyser en modellparameter sin statistiske signifikans. I modellen framkom det at INCIDENT_DISPOSITION_CODE variabelen hadde flere statistisk signifikante parametere. Blant de med høyest signifikans, var modell parameteren "unfounded" (kode 90). Denne parameteren omhandler caser som av ulike årsaker ikke kunne ferdigstilles. Flere andre koder som var høyt signifikante omhandlet å motstå hjelp, ikke tilstede og oppdrag avbrutt. Hovedobservasjonen fra et slikt resultat er at disse situasjonene ikke er ønsket arbeidssituasjon, og bør derav motvirkes.

5.2.4 Konklusjon

Den beste modell for å besvare problemstillingen var en XGBoost. Der XGBoost scoret best på alle presisjonsmål, både under trening av modell og testing av modell. XGBoost endte opp med en med MAE på 627.25, som er cirka 100 sekunder lavere enn nest beste modell som hadde MAE på 722. Dette var det best resultat av alle modellene som ble produsert. I treningen av modellen ble repetert kryssvalidering andvendt, dette for å motvirke overfitting. XGBoost utviser lav grad av overfitting, med lite tap av prediksjonsevne fra treningsdata til testdata. Angående bias og varians trade-off, er svakheten

i modellen vinkeltt mer mot bias (underfitting) og mindre grad varians (overfitting). Vi konkluderer med at XGBoost modellen adresserer dataene best med lavest MAE. Overfitting ser ikke ut til å være et stort problem, men modellen har et potensiale til forbedring. Da ved å fokusere mer på å minimere bias.

Kapittel 6

Begrensninger & Videre Forskning

6.1 Begrensninger

I studien er det flere begrensninger. Sammenligningsgrunnlag for resultater er en begrensning som er viktig å påpeke. Vi fant ikke studier som hadde nyttiggjort PM og ML, på aktiviteter i ambulansetjenesten. Tidligere studier har nyttiggjort PM og ML, men ikke på samme datagrunnlag - til vår kunnskap. En konsekvens av dette er mindre sikkerhet i å kunne stadfeste en konklusjon til problemstillingene. Det er vanskelig å vite pr dagsdato hva gode presisjonsmåltall for ML og PM er under disse forutsetningene.

En begrensning i PM er at vi bare brukte alfa pluss algoritmen. Ved bruk av flere algoritmer, slik vi gjorde i ML, ville det interne sammenligningsgrunnlaget har vært større. Dette kunne ha ledet til økt reliabilitet i studien.

I studien fokuserte vi på kun bruk av supervised learning i ML. Flere teknikker forbundet med unsupervised learnig kunne vært andvendt for å høste mer informasjon om datasettet. I unsupervised learnig kunne eksempelvis automatisk variabel seleksjon (ved hjelp av prinsipal komponent analyse) blitt brukt. En slik teknikk ville automatisk funnet mønstre i data til bruk som modellparametere. En annen begrensning var avgrensningen vi valgte når det gjaldt hva vi skulle gjøre med manglende observasjoner. Vi valgte å ikke bruke imputation, på basis av hvor alvorlig potensiell feilbehandling av data kan ha. Likevel er et alternativ hadde vært å bruke algoritmer til å predikere manglende verdier, slik at mer informasjon hadde blitt bevart.

Flere algoritmer kunne vært andvendt for predikering av gjennomstrømstid. Der vi kunne ha brukt algoritmer som måler mer en kun lineære forhold. Ikke lineære algoritmer, som tar nytte av eksempelvis ensemble algoritmer (da f.eks en ikke lineær variant av XGBoost) kunne vært andvendt, for å bedre adressere mønstre i dataene. CPU kraft og tid var en drivende faktor for at mindre komplekse algoritmer ble andvendt.

6.2 Videre Forskning

I fremtidig forskning angående PM, vil særlig implementasjonen av prosessmodeller være av interesse (eksempelvis et Petri net, lik det studien presenterte). Vi vil foreslå at fremtidige studier vil se på forholdet mellom interne arbeidsprosedyrer og prosessmodeller. Dette vil være av interesse og knytte opp i mot modell enhancement (som nevnt kort i teori). Modell enhancement går ut på å forbedre en prosessmodell, ved å sammenligne arbeidsprosedyre protokoller med en prosessmodell.

Et annet alternativ til supervised ML prediksjon er transition system (Aalst, 2016, s. 312). Transition System er en PM metode for prediksjon. I denne metoden brukes uferdige caser, for å predikere gjenstående tid. Dette innebærer at en ambulanse tjeneste kan få prediksjon på gjenstående tid, på basis av løpende akkumulert historisk data. Det hadde vært av interesse å sammenligne resultatene fra denne metoden med ML. Særs i lys av om slike verktøy kan bedre koordinering og allokering av ressurser.

Conformance er én PM metode som vil kunne høste verdi for organisasjoner. I denne studien var det hovedsakelig fokus på måltall, for å sammenligne modell med logg. I organisasjoner vil også bruken av conformance opp mot spesifikke mål være av interesse å studere. Ett eksempel i ambulansetjenesten er å bruke comformance for å detektere unormal tidsbruk, og flaskehalser. Der en spesifiser et forhåndsdefinerte kriterium for unormal tidsbruk, som en skal søke etter i loggen (Aalst, 2016, s. 33). Vi anser det interessant å studere om hvorvidt et slikt verktøy kan brukes i daglig drift, i ambulansetjenesten.

Ved videre forskning i kontekst av ML, kunne et kvalitativ intervju med ambulansepersonell bli gjennomført. Dette kunne muligens hjulpet til med å redusere ufravikelig feil, ved å inkludere flere variabler ved hjelp av bransjeekspertise. Ny variabler inkludert, vil kunne ytterligere hjelpe til med å forklare gjennomstrømstid. En variabel av stor interesse ville vært variabel som forklarer ambulanses lokasjonen ved utrykning[1].

En annen ting av interesse hadde vært å studere om hvorvidt kunnskap fra prosessmodellen kan andvendes for å lage nye variabler for ML[2]. Der eksempelsvis bruk av uvanlige traces, eller flaskehalser observert i prosessmodellen, kunne vært brukt som utgangspunkt for variabel konstruksjon.

[1] Grunnet anonymisering av data, var ikke denne variabelen tilgjengelig

[2] Med dette menes kunnskap for å supplere feature engineering/variabel konstruksjon. Variabel konstruksjon er en av mest krevene og viktigste oppgavene for pressise ML modeller.

Bibliografi

Aalst, W. V. D. (u.å). Event Logs What kind of data does process mining require? Hentet 16. April. 2020 fra: http://www.processmining.org/_media/presentations/event_logs_the_input_for_process_mining.pdf

Aalst, W. V. D. et al. (2012) Process Mining Manifesto. In: Daniel F., Barkaoui K., Dustdar S. (eds) Business Process Management Workshops. BPM 2011. Lecture Notes in Business Information Processing, vol 99. Berlin, Heidelberg: Springer. https://www.win.tue.nl/ieeetfpm/downloads/Process%20Mining%20Manifesto.pdf

Aalst, W. V. D. (2016). *Process Mining Data Science in Action* (2. utg.) Berlin, Heidelberg: Springer

Anne, R. (2010, 18. Oktober) which mining algorithm should you use? Hentet fra: http://fluxicon.com/blog/2010/10/prom-tips-mining-algorithm/

Areal connect (2020, u.å.). Hentet 1. Juni. fra: https://manhattanny.areaconnect.com/zip2.htm?city=Manhattan&search=zip

Badakhshan, P & Alibabaei, A. (2018). Using Process Mining for Process Analysis Improvement in Pre-Hospital Emergency. Innlegg presentert ved EMENA 2018. Proceedings of the Middle East North Africa Conference for Information Systems, Paris, Frankrike.

Box plot. (u.å). I wikipedia. Hentet 13.mai 2020 fra: https://en.wikipedia.org/wiki/Box_plot

Buijs, J.C.A.M., Aalst, V.D. W., Dongen, B.F. (2012). On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. 7565. 305-322. Hentet fra: http://www.padsweb.rwth-aachen.de/wvdaalst/publications/p688.pdf

BupaR-event logs. (u.å.). Creating event logs. Hentet 17. April 2020 fra: https://www.bupar.net/creating_eventlogs.html

BupaR-PM4Py. (u.å.). Connecting to PM4Py. Hentet 17. April 2020 fra: https://www.bupar.net/pm4py.html

CDC. (u.å.). Heart Disease Facts. Hentet 16.april fra https://www.cdc.gov/heartdisease/facts.htm

Cramers V. (u.å). I wikipedia. Hentet 26.mai 2020 fra: https://en.wikipedia.org/wiki/Cram%C3%A9r%27s_V

CRAN (u.å.) The R Project for Statistical Computing. Hentet 21. Mai. 2020 fra: https://www.r-project.org

Davenport, T. H., Harris, J. G. (2017). *Competing on analytics: The new science of winning.* Boston, Massachusetts: Harvard Business Review press.

Dongen, B.F.V., Crooy, R.A., Aalst, V.D. W. (2008). Cycle Time Prediction: When Will This Case Finally Be Finished? In: Meersman R., Tari Z. (Red.). On the Move to Meaningful Internet Systems: OTM 2008. OTM 2008. Lecture Notes in Computer Science, vol 5331. Springer, Berlin, Heidelberg

Duma, D & Aringhieri, R. (2020). An ad hoc process mining approach to discover patient paths of an Emergency Department. Flex Serv Manuf J 32, 6–34. https://doi.org/10.1007/s10696-018-9330-1

Elite data science. (u.å.). Hentet fra https://elitedatascience.com/model-training)

EMS WORLD. (2004). EMS Response Time Standards. Hentet fra:https://www.emsworld.com/article/10324786/ems-response-time-standards

Evaluation Log-Model. (u.å). Evaluation Log-Model. Hentet fra: http://pm4py.pads.rwth-aachen.de/documentation/conformance-checking/evaluation-log-model/

FDNY EMS. (u.å). I wikipedia. Hentet 23.april 2020 fra: https://en.wikipedia.org/wiki/New_York_City_Fire_Department_Bureau_of_EMS

Feature scaling. (u.å.). I wikipedia. Hentet 30. Mai 2020 fra: https://en.wikipedia.org/wiki/Feature_scaling

FEMS. (2016, Oktober). Hentet 17. April. 2020: https://fems.dc.gov/sites/default/files/dc/sites/fems/FY%202016%20Response%20Time%20Performance%20Measures_0.pdf

Fluxicon. (u.å). Disco Tour. Hentet 16. April. 2020 fra: https://fluxicon.com/disco/files/Disco-Tour.pdf

Gradient Boosting. (u.å.). Gradient Boosting. Hentet 29. Mai. 2020 fra: https://bradleyboehmke.github.io/HOML/gbm.html#xgboost

Gripsrud, G., Olsson, U. H & Silkoset, R. (2016). *Metode og dataanalyse: beslutningsstøtte for bedrifter ved bruk av JMP, Excel og SPSS* (utg. 3). Oslo: Cappelen Damm.

Hastie, T., Tibshirani, R. & Friedman, J. (2017). *The elements of statistical learning: Data mining, inference and prediction (2 utg). Berlin: Springer.* (Det er flere utgaver tror jeg)

Helsedirektoratet. (2019, 9. mai). Tid fra AMK varsles til ambulanse er på hendelsessted. Hentet 8.januar 2020 fra:https://www.helsedirektoratet.no/statistikk/kvalitetsindikatorer/akuttmedisinske-tjenester-utenfor-sykehus/tid-fra-amk-varsles-til-ambulanse-er-p%C3%A5-hendelsessted

James, G., Hastie, T., Witten, D., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R.* New York: Springer.

Jank, W. (2011). *Business analytics for managers.* New York: Springer.

Janssenswillen, G., Depaire, B., Swennen, M., Jans, M., & Vanhoof, K. (2019). bupaR: Enabling reproducible business process analysis. Knowledge-Based Systems, 163, 927-930.

Krzyk, K. (2018, 25. juli). Coding Deep Learning For Beginners. Hentet fra https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d

Kuhn, Max. (2019, 27.mars). The caret Package. Hentet fra : https://topepo.github.io/caret/index.html

Lamine E., Fontanili F., Di Mascolo M & Pingaud H. (2015). Improving the Management of an Emergency Call Service by Combining Process Mining and Discrete Event Simulation Approaches. Innlegg presentert ved PRO-VE 2015. IFIP Advances in Information and Communication Technology, Albi, Frankrike.

Lingitz, L., Gallinaa, V., Ansaria, F., Gyulai, D., Pfeifferc, A., Wilfried, S & Monostori, L. (2018). Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer. Procedia CIRP 72, 1051-1056.

Mans, S. R., Vanwersch, J.B. R., Aalst, V.D. W. (2015). *Process Mining in Healthcare Evaluating and Exploiting Operational Healthcare Processes.* Nederland: Springer

NHBLI. (u.å.). Sudden Cardiac Death. Hentet 18.05 fra: https://www.nhlbi.nih.gov/health-topics/sudden-cardiac-arrest

Nordlie, H. (2019). New York. Thuesen, N. P. (red.).Store norske leksikon. Hentet 16. april 2020 fra: https://snl.no/New_York

NOU 1998: 9. (1998). Hvis det haster..... Faglige krav til akuttmedisinsk beredskap. Hentet fra https://www.regjeringen.no/contentassets/8087d548c0a04059aa88f416fe19f3cc/no/pdfa/nou199819980009000dddpdfa.pdf

NYC 911 reporting. (2020, 16.april). Definitions. Hentet fra https://www1.nyc.gov/site/911reporting/reports/definitions.page

Nyc mayor's office of operations. (2020, 16.april). New York City Fleet Daily Service Report. Hentet fra https://www1.nyc.gov/site/operations/performance/fleet-report.page

NYC Opendata. (2019). The Next Decade of Open Data 2019: Open Data for All Report. Hentet fra: https://opendata.cityofnewyork.us/wp-content/uploads/2019/09/2019_OpenDataForAllReport.pdf

NYC Open data. (2020, 20.februar). Hentet fra: https://data.cityofnewyork.us/Public-Safety/EMS-Incident-Dispatch-Data/76xm-jjuj

Petri Net Management. (u.å.). Petri Net Management. Hentet 16. Mai 2020 fra:http://pm4py.pads.rwth-aachen.de/documentation/petri-net-management/

Roser, M., Ortiz, E-O. & Ritchie, H. (2013). Life Expectancy. Hentet 15. April fra https://ourworldindata.org/life-expectancy

Reticulate. (u.å.). R Interface to Python. Hentet 17. April 2020 fra: https://rstudio.github.io/reticulate/

Rozinat. A, Veloso. M & Aalst, W.V.D. (2008). Evaluating the quality of discovered process models. 2nd Int. Workshop on the Induction of Process Models.

SAS. (2020, u.å.). Machine Learning What it is and why it matters. Hentet fra https://www.sas.com/en_us/insights/analytics/machine-learning.html

Schutts, R. & O'Neil, C. (2013). *Doing Data Science: Straight Talk from the Frontline.* California: O'Reilly Media

Spanyi, A. & Davenport, T. H. (2019, 13, April). What Process Mining Is, and Why Companies Should Do It. Hentet fra: https://hbr.org/2019/04/what-process-mining-is-and-why-companies-should-do-it

Statista. (2016, 27.november). Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025. Hentet fra https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

Stine, E., Røise,O., Ribu,L. (2013). Bruk av triage i norske akuttmottak,Tidsskr Nor Legeforen 133(3), 285 –289. Hentet fra https://tidsskriftet.no/2013/02/originalartikkel/bruk-av-triage-i-norske-akuttmottak

Sucarrat, G. (2016). *Metode og Økonometri en moderne innføring.* Bergen: Fagbokforlaget

Sunde, K., Fremstad, K.O., Furuheim, J. & Steen, P.A. (2001). Ambulance response intervals during cardiac arrest in Oslo, Norway. Tidsskr Nor Lægeforen,121 (8),900–903. Hentet fra: https://tidsskriftet.no/sites/default/files/pdf2001--900-3.pdf

Sunday. (u.å). I wikipedia. Hentet 17. April 2020 fra: https://en.wikipedia.org/wiki/Sunday

Theoria. (2020). I Lexico online dictionary. Hentet 16. April. 2020 fra: https://www.lexico.com/definition/theoria

Theory. (2020). I Merriam-Webster's online dictionary. Hentet 16. April. 2020 fra: http://www.m-w.com/dictionary/heuristic

TUe. (2009, 17.juni). Fuzzy Miner. Hentet fra: http://www.processmining.org/online/fuzzyminer

United states census bureau. (2020, 16.april). QuickFacts. Hentet fra https://www.census.gov/quickfacts/fact/table/newyorkcitynewyork,bronxcountybronxboroughnewyork,kingscountybrooklynboroughnewyork,newyorkcountymanhattanboroughnewyork,queenscountyqueensboroughnewyork,richmondcountystatenislandboroughnewyork/PST045219

XGBoost. (u.å.). XGBoost. Hentet 29. Mai. 2020 fra: https://xgboost.readthedocs.io/en/latest/parameter.html

Zliobaite, I., Pechenizkiy, M., & Gama, J. (2016). An overview of concept drift applications. Hentet fra: https://www.win.tue.nl/~mpechen/publications/pubs/CD_applications15.pdf

Westerman, G., bonnet, D. & McAfee, A. (2014). leading digital: turning technology into business transformation. Boston, Massachusetts: Harvard Business Review press.

WHO. (u.å.). WHO Emergency Care System Framework Infographic. Hentet fra: https://www.who.int/emergencycare/emergencycare_infographic/en/

World Population Review. (2020,16.april). Manhattan Population 2020. Hentet fra: https://worldpopulationreview.com/boroughs/manhattan-population/

Kronick et al. (2015). Part 4: Systems of Care and Continuous Quality Improvement, 2015 American Heart Association Guidelines Update for Cardiopulmonary Resuscitation and Emergency Cardiovascular Care. American heart association,Volume 132, Issue 18 supply 2, 3 November 2015, side 397-413. Hentet 26. Mai. 2020 fra: https://www.ahajournals.org/doi/full/10.1161/cir.0000000000000258

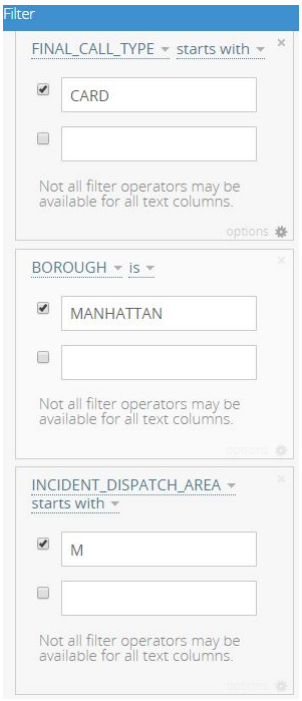
Vedlegg

Vedlegg I

Datavask

I.1 Utdrag NYC Database

For å spesifisere uttrekk av data, gjøres dette i OpenData[1]. På OpenData kan en spesifisere ulike filtre av databasen.



Figur I.1: Utdrag fra NYC Opendata Database.

Det spesifiserte uttrekket gir 222 724 Rader og 31 variabler i csv format. Variabel beskrivelse ses i figur **I.2**. Databasen vi lastet ned data fra (27 februar, 2020) ble sist oppdatert av FDNY den 20 februar 2020.

[1]Nettsiden er: `https://data.cityofnewyork.us/Public-Safety/EMS-Incident-Dispatch-Data/76xm-jjuj`

I.2 EMS Variabler

Forkortelse navn (bruk i tekst)	Variabeler fra Opden Data database, orginalt navn	Beskrivelse
CAD_INCIDENT_ID	CAD_INCIDENT_ID	An incident identifier comprising the julian date and a 4 character sequence number starting at 1 each day.
INCIDENT_DATETIME	INCIDENT_DATETIME	The date and time the incident was created in the dispatch system
INITIAL_CALL_TYPE	INITIAL_CALL_TYPE	The call type assigned at the time of incident creation.
INITIAL_SEVERITY	INITIAL_SEVERITY_LEVEL_CODE	The segment(priority) assigned at the time of incident creation.
FINAL_CALL_TYPE	FINAL_CALL_TYPE	The call type at the time the incident closes.
FINAL_SEVERITY	FINAL_SEVERITY_LEVEL_CODE	The segment(priority) assigned at the time the incident closes.
FIRST_ASSIGNMENT	FIRST_ASSIGNMENT_DATETIME	The date and time the first unit is assigned.
VALID_DISPATCH	VALID_DISPATCH_RSPNS_TIME_INDC	Indicates that the components comprising the calculation of the DISPATCH_RESPONSE_SECONDS_QY are valid.
DISPATCH_RESPONSE	DISPATCH_RESPONSE_SECONDS_QY	The time elapsed in seconds between the incident_datetime and the first_assignment_datetime.
FIRST_ACTIVATION	FIRST_ACTIVATION_DATETIME	The date and time the first unit gives the signal that it is enroute to the location of the incident.
FIRST_ON_SCENE	FIRST_ON_SCENE_DATETIME	The date and time the first unit signals that it has arrived at the location of the incident.
VALID_INCIDENT	VALID_INCIDENT_RSPNS_TIME_INDC	Indicates that the components comprising the calculation of theINCIDENT_RESPONSE_SECONDS_QY are valid.
INCIDENT_RESPONSE	INCIDENT_RESPONSE_SECONDS_QY	The time elapsed in seconds between the incident_datetime and the first_on_scene_datetime.
INCIDENT_TRAVEL	INCIDENT_TRAVEL_TM_SECONDS_QY	The time elapsed in seconds between the first_assignment_datetime and the first_on_scene_datetime.
FIRST_TO_HOSP	FIRST_TO_HOSP_DATETIME	The date and time the first unit gives the signal that it is enroute to the hospital.
FIRST_HOSP_ARRIVAL	FIRST_HOSP_ARRIVAL_DATETIME	The date and time the first unit signals that it has arrived at the hospital.
INCIDENT_CLOSE	INCIDENT_CLOSE_DATETIME	The date and time the incident closes in the dispatch system.
HELD_INDICATOR	HELD_INDICATOR	Indicates that for some reason a unit could not be assigned immediately
INCIDENT_DISPOSITION	INCIDENT_DISPOSITION_CODE	A code indicating the final outcome of the incident. See incident dispositions.
BOROUGH	BOROUGH	The borough of the incident location.
INCIDENT_DISPATCH	INCIDENT_DISPATCH_AREA	The dispatch area of the incident.
ZIPCODE	ZIPCODE	The zip code of the incident.
POLICEPRECINCT	POLICEPRECINCT	The police precinct of the incident.
CITYCOUNCILDISTRICT	CITYCOUNCILDISTRICT	The city council district.
COMMUNITYDISTRICT	COMMUNITYDISTRICT	The community district.
COMMUNITYSCHOOLDISTRICT	COMMUNITYSCHOOLDISTRICT	The community school district.
CONGRESSIONALDISTRICT	CONGRESSIONALDISTRICT	The congressional district.
REOPEN_INDICATOR	REOPEN_INDICATOR	Indicates that at some point the incident was closed but then reopened.
SPECIAL_EVENT_INDICATOR	SPECIAL_EVENT_INDICATOR	Indicates that the incident was a special event such as the NYC Marathon.
STANDBY_INDICATOR	STANDBY_INDICATOR	Indicates that the units were assigned to stand by incase they were needed.
TRANSFER_INDICATOR	TRANSFER_INDICATOR	Indicates that the incident was created for the transportation ...- of a patient from one facility (ie a hospital or nursing home) to another.
Forkortelse navn (bruk i tekst)	**Egen produserte Variabler**	**Beskrivelse**
Vi henviser til dummyfisering av...	FIRST_ACTIVATION_DATETIME_D	Dummy av variabel: FIRST_ACTIVATION_DATETIME
	FIRST_TO_HOSP_DATETIME_D	Dummy av variabel: FIRST_TO_HOSP_DATETIME
..data fra originale variablene	FIRST_HOSP_ARRIVAL_DATETIME_D	Dummy av variabel: FIRST_HOSP_ARRIVAL_DATETIME
	INCIDENT_CLOSE_DATETIME_D	Dummy av variabel: INCIDENT_CLOSE_DATETIME
YEAR	INCIDENT_DATETIME_year	Variabler for når en case inntraff med henyn på: YEAR
TIME_START	INCIDENT_DATETIME_time	Variabler for når en case inntraff med henyn på: TIME_START
TIME_CLOSE	INCIDENT_CLOSE_DATETIME_time	Variabler for når en case inntraff med henyn på: TIME_CLOSE
WEEKDAY	INCIDENT_DATETIME_Weekday	Variabler for når en case inntraff med henyn på: WEEKDAY
MONTH	INCIDENT_DATETIME_Month	Variabler for når en case inntraff med henyn på: MONTH
WEEK	INCIDENT_DATETIME_Week	Variabler for når en case inntraff med henyn på: WEEK
THROUGHPUT_TIME	THROUGHPUT_TIME	Totale gjennomstrømstid fra kontakt med helsevesen til evt innleggelse (eller annen løsning)

Figur I.2: EMS Variabel Forklaring. Kilde: NYC OpenData (2020).

I.3 Preprossesering av Data

Alt som tas opp her finnes i vedlagt R kode.

- NYC OPEN data er av typen csv format. Dette lastes inn i R.
- Klassifiserer ulike koloner på basis av om variabelen er faktorvariabel (kategorisk), numeriskvariabel (kontinuerlig) og tidsvariabel (POSIXct)
- Omgjør alle manglende observasjoner til NAs.
- Dette gir følgende datasett med variabler:

Faktor Variabler	**Numerisk Variabler**
CAD_INCIDENT_ID	DISPATCH_RESPONSE_SECONDS_QY
INITIAL_CALL_TYPE	INCIDENT_RESPONSE_SECONDS_QY
INITIAL_SEVERITY_LEVEL_CODE	INCIDENT_TRAVEL_TM_SECONDS_QY
FINAL_CALL_TYPE	
FINAL_SEVERITY_LEVEL_CODE	
VALID_DISPATCH_RSPNS_TIME_INDC	
VALID_INCIDENT_RSPNS_TIME_INDC	
HELD_INDICATOR	
INCIDENT_DISPOSITION_CODE	
BOROUGH	**POSIXct**
INCIDENT_DISPATCH_AREA	INCIDENT_DATETIME
ZIPCODE	FIRST_ASSIGNMENT_DATETIME
POLICEPRECINCT	FIRST_ACTIVATION_DATETIME
CITYCOUNCILDISTRICT	FIRST_ON_SCENE_DATETIME
COMMUNITYDISTRICT	FIRST_TO_HOSP_DATETIME
COMMUNITYSCHOOLDISTRICT	FIRST_HOSP_ARRIVAL_DATETIME
CONGRESSIONALDISTRICT	INCIDENT_CLOSE_DATETIME
REOPEN_INDICATOR	
SPECIAL_EVENT_INDICATOR	
STANDBY_INDICATOR	
TRANSFER_INDICATOR	

Tabell I.1: Variabel Inndeling.

I.4 Datavask av Ambulansedata

Utganspunkt for datavask, er resultatet av det pre prosesserte datasettet beskrevet i **Vedlegg I, seksjon I.3.**

Variabel	**Handling**	**Note**		**Datasett for Metode**
CONGRESSIONALDISTRICT	Fjerne NAs	1		
INCIDENT_RESPONSE_SECONDS_QY	Fjerne NAs	2		
INCIDENT_DISPOSITION_CODE	Fjerne NAs	3		
COMMUNITYSCHOOLDISTRICT	Fjerne NAs	4		
ZIPCODE	Fjerne NAs	5	→	Datasett for bruk i PM
Preparere data for ML				
THROUGHPUT_TIME	Konstruksjon variabel	6		
FIRST_ACTIVATION_DATETIME_D	Konstruksjon variabel	7		
FIRST_TO_HOSP_DATETIME_D	Konstruksjon variabel	8		
FIRST_HOSP_ARRIVAL_DATETIME_D	Konstruksjon variabel	9		
INCIDENT_CLOSE_DATETIME_D	Konstruksjon variabel	10		
INCIDENT_DATETIME_year	Konstruksjon variabel	11		
INCIDENT_DATETIME_time	Konstruksjon variabel	12		
INCIDENT_CLOSE_DATETIME_time	Konstruksjon variabel	13		
INCIDENT_DATETIME_Weekday	Konstruksjon variabel	14		
INCIDENT_DATETIME_Month	Konstruksjon variabel	15		
INCIDENT_DATETIME_Week	Konstruksjon variabel	16		
INCIDENT_DATETIME	Fjerne variabel	17		
FIRST_ASSIGNMENT_DATETIME	Fjerne variabel	18		
FIRST_ACTIVATION_DATETIME	Fjerne variabel	19		
FIRST_ON_SCENE_DATETIME	Fjerne variabel	20		
FIRST_TO_HOSP_DATETIME	Fjerne variabel	21		
FIRST_HOSP_ARRIVAL_DATETIME	Fjerne variabel	22		
INCIDENT_CLOSE_DATETIME	Fjerne variabel	23		
CAD_INCIDENT_ID	Fjerne variabel	24		
BOROUGH	Fjerne variabel	25		
THROUGHPUT_TIME	Fjerne NAs	26		
REOPEN_INDICATOR	Fjerne variabel	27		
VALID_DISPATCH_RSPNS_TIME_INDC	Fjerne variabel	28		
VALID_INCIDENT_RSPNS_TIME_INDC	Fjerne variabel	29		
SPECIAL_EVENT_INDICATOR	Fjerne variabel	30		
TRANSFER_INDICATOR	Fjerne variabel	31		
STANDBY_INDICATOR	Fjerne variabel	32		
POLICEPRECINCT	Fjerne variabel	33		
CITYCOUNCILDISTRICT	Fjerne variabel	34		
COMMUNITYDISTRICT	Fjerne variabel	35		
COMMUNITYSCHOOLDISTRICT	Fjerne variabel	36		
CONGRESSIONALDISTRICT	Fjerne variabel	37		
ZIPCODE	Fjerne variabel	38		
INCIDENT_CLOSE_DATETIME_D	Fjerne variabel	39		
FIRST_TO_HOSP_DATETIME_D	Fjerne variabel	40		
FIRST_HOSP_ARRIVAL_DATETIME_D	Fjerne variabel	41		
INITIAL_CALL_TYPE	Fjerne variabel	42		
INCIDENT_DATETIME_Month	Fjerne variabel	43		
INCIDENT_TRAVEL_TM_SECONDS_QY	Fjerne outliers	44		
DISPATCH_RESPONSE_SECONDS_QY	Fjerne outliers	45		
THROUGHPUT_TIME	Fjerne outliers	46		
INCIDENT_DISPOSITION_CODE	Fjerne outliers	47		
FINAL_SEVERITY_LEVEL_CODE	Fjerne outliers	48		
INCIDENT_RESPONSE_SECONDS_QY	Fjerne variabel	49	→	Datasett for bruk i ML

Tabell I.2: Datavask.

Note 1-5 resulterer til datasettet som ble andvendt i PM. Note 1-49 resulterer til datasett andvendt i ML

- Note 1 : Den første variablene som er valgt å eliminere NAs fra, er CONGRESSIONALDISTRICT. Det andvendes ikke noe form for tilegnelse(imputation) ved en slik variabel, da dette er for kritisk , i kontekst av helsedata[2]. Etter eliminering av alle NAs tilknyttet variablene CONGRESSIONALDISTRICT går man fra 222.724 observasjoner til 218.828.

- Note 2: Det er valgt å eliminere alle NAs tilknyttet variabelen INCIDENT_RESPONSE _SECONDS _QY, der majoriteten av NAs skyldes at flere NAs er assosiert med aktivitetsvariabelen FIRST_ON_SCENE _DATETIME. Det er dermed også heller ikke noe grunnlag for imputation, da INCIDENT_RESPONSE_SECONDS_QY er direkte tilknyttet FIRST_ON_SCENE_DATETIME. Alle NAs tilknyttet FIRST_ON_SCENE _DATETIME blir eliminert ved å fjerne alle NAs tilknyttet INCIDENT_RESPONSE _SECONDS_QY. Etter denne elimineringen står man igjen med 215.124 observasjoner.

- Note 3 : Den tredje variabelen som det elimineres NAs i, er INCIDENT_DISPOSITION _CODE, her igjen, ses det ikke nok grunnlag for imputation, gitt resonnementet i metodekapittelet. Etter denne elimineringen står man igjen med 215.089 observasjoner.

- Note 4 : Fjerde variabel det elimineres NAs i er COMMUNITYSCHOOLDISTRICT. Resterende observasjoner blir så 215 067.

- Note 5 : Den siste variabelen som elimineres tilknyttet NAs er ZIPCODE, igjen blir det for vagt med imputation ved en slik variabel. gjenværende Observasjoner blir så 215.066. Den eneste hensikten ved å endre på ZIPCODE variabelen, ligger i at denne variabelen brukes til produksjon av 1.1. Blant alle de ulike zipcodene er det 148 unike koder. Dette er et skjevt bilde da flere input feil gjør opp for mange av disse unike variablene. Input feil som at flere av radene inneholdt verdier som 10,036,10,037 og 10,038 osv. Dette anses som feil, da zipcoder assosiert med Manhatten strekker seg fra 10 001 til 10 286 (Areal connect, 2020). Det ble observert at 57 av observasjonene inneholdt input feil, og endte derfor opp med 91 unike verdier i stedet for 148.

- Note 6 : Produsert gjennomstrømstid, avhengig variabel.

- Note 7-10 : Produserer dummyvariabel for å modellere NAs i aktivitetesvariabelen. det introduseres dummyvariabler på, DATETIME, aktivitetsvariablene, som har

[2]Det er mulig via CARET pakken i R (preprosess funksjon) og spesifisere ulike algoritmer for predikering av mulig verdier man kan tilegne, på basis av trening/test splitt i datasett. Tilegne nye verdier med median, eller snitt ved tilfelle av numerisk verdier, er høyt spekulativt. Gitt få NAs i variabelen velges den heller å elimineres.

manglende observasjoner. Disse fire variablene innehar samme navn som før, med tillegget "D", som indikerer dummy. Med disse fire nye dummyvariablene vil variablene også samtidig kunne representere manglende observasjoner i datasettet. Dette er fordelaktig for algoritmene, da manglende data observasjoner, indirekte, kan være et viktig datapunkt i seg selv. Info om "missingness" kan gi bedre modeller.

- Note 11 : INCIDENT_DATETIME_year variabelen tar utgangspunkt i den originale variabelen ved samme navn utenom "_year". Som utganspunkt fra den originale variabelen, tilegnes det en ny variabel som oppgir hvilket år én case har skjedd. Dette vil i modellen representeres som en kategorisk faktor variabel.

- Note 12 og 13 : INCIDENT_DATETIME_time og INCIDENT_CLOSE_DATETIME _time er variabler tar utgangspunkt i de originale variablene ved samme navn, utenom "_time". Fra de originale variablene henter vi ut hvilke time når den første aktivitet har inntruffet, og når den siste aktivitet har inntruffet. Variablene kan ta verdien 0 til 23. Denne variabelen produseres for å enklere representerer tidsaspektet vedrørende start og stopp ved en unik case. Dette ses til å korrespondere med process mining litteraturen. Det velges å ha primært disse to variablene, da om vi skulle ha flere variabler som relfekterer tid, vil dette kunne bringe multikollinearitet til modellen. Dataene anses ikke som tidsseriedata, da vi har unike caser. De konstruerte variablene også registrert som kategoriske faktor variabler.

- Note 14 : INCIDENT_DATETIME_Weekday variabelen tar på seg en verdi mellom 1-7. 1 Er søndag, da søndag regnes som dag 1. Ukedagene, i statene, er fra hebraisk kalender (Sunday, u.å). Det vil likevel ikke utgjøre noe forskjell i analysen, selv når verdiene er forskjøvet. Ukedag vil kunne fange opp effekter av for eksempel arbeidsdager, kontra helg.

- Note 15 : Månedsvariabelen INCIDENT_DATETIME_Month, returnerer en verdi fra 1-12. Verdiene henviser til alle måneder fra januar til desember. Denne variabelen anses som hensiktsmessig å inkorporere, da den kan fange opp variasjon i den avhengig variabelen, som er total gjenomstrømstid. Det er en mulig hypotese å anta at sykdomsforløp vil være annerledes fra en måned til en annen. Effekter av årstider kan være nyttig å fange opp. Dette studiet dreier seg ikke om helseforskning, men det er naturlig å tro at en medisinske tilstander vil kunne bli statistisk representert fra en måned til en annen. Bruddskader når det er is og glatt på gaten er et eksempel.

- Note 16 : Ukesvariabelen INCIDENTDATETIME_Week, tar på seg verdier fra 1-53. 53 Indikerer skuddår. Ukesvariabelen tas med, da denne variabelen gir mer informasjon vedrørende endring over tid, fremfor månedsvariabel.

- Note 17-23: Eliminerer aktivitetsvariabler, da dummyer er skapt. Disse variab-

lene reflekterer de syv aktivitets variabler, i form av timestamps[3]. Det ses som unødvendig å ha med disse variablene, da tidselementer blir allerede fanget opp i de konstruerte dummyvariablene forklart i note 7-10. Posixct er uegnet format, da det ikke er kompatibel beregningsmessig med algoritmer.

- Note 24 : CAD_INCIDENT_ID elimineres, da denne variablene ikke forklarer noe variasjon i datasettet, da denne variabelen kun er en unik identifikator for et case.

- Note 25 : Variabelen BOROUGH reflekterer hvilken bydel hendelsen inntraff, den ble eliminert. Grunnen til dette var ettersom det kun fokuseres på Manhattan. Denne variabelen vil derfor ikke forklare noe variasjon i datasettet, da alle observasjoner er i Manhattan.

- Note 26 : Det elimineres 47 NAs tilhørende den konstruerte avhengige variabelen THROUGHPUT_TIME. Rasjonale for dette ligger i at det kun ønskes å studere fullførte hendelser.

- Note 27-43 : Notene er tilknyttet chi-square, med tilhørende underpunkter. Cut-off for eliminering var en samvariasjon mellom variablene på 0.8. Alle variabler med en samvariasjon på mindre en 0.8 beholdes.

  Flere av indikator variablene fra NYC EMS (se figur vedlegg **I.2**) ble eliminert. Da flere av indikator variablene hadde perfekt korrelasjon med hverandre. Det ble avgjort å kun beholde en av indikator variabelne, da multikollinearitet vil kunne oppstå. Variabelen HELD_INDICATOR ble beholdt av indikator variablene. Dette fordi, denne variabelen skapte minst problemer med henhold til korrelasjon mellom de resterende variabler (se figur vedlegg **I.3**).

  Overflødige variabler assosiert med lokasjonen til hvor hendelsen hadde tatt plass ble eleminert. Det gjenstår[4] syv variabler i datasettet som har som hensikt å lokalisere hvor hendelsen inntraff. Disse variablene ble eliminert, der maksimum terskelen for korrelasjon ble brutt. POLICEPRECINCT, CITYCOUNCILDISTRICT, COMMUNITYDISTRICT, COMMUNITYSCHOOLDISTRICT og CONGRESSIONALDISTRICT var variablene som ble eliminert. Av de to gjenstående høyt korrelerte lokasjonsvariablene, var ZIPCODE og INCIDENT_DISPATCH_AREA. ZIPCODE variabelen ble fjernet. Rasjonale for dette var at ZIPCODE inneholdt flere kategorier, som kan forårsake problemer ved trening av ML modellen[5]. INCIDENT_DISPATCH_AREA er en variabel som forklarer lokasjon på et mer aggregert nivå, og hadde som følge av dette færre kategorier. Dette introdusere færre problemer ved trening av ML modellen, og INCIDENT_DISPATCH_AREA ble derfor behold blant lokasjonsvariablene.

[3]Posixct format i R.
[4]Ettersom BOROUGH ble fjernet fra tidligere
[5]Der observasjoner blir inkludert i trenigssett, men ikke i testsett, grunnet for få observasjoner.

INCIDENT_CLOSE_DATETIME_D, ble eliminert. Da det fra tidligere ble fjernet alle NAs i THROUGHPUT_TIME. Som følge av dette, gjenstod det ingen variasjon i dummy-variabelen. Ettersom dummy variabelen nå kun innehar verdien 1 for alle observasjoner. Den vil da ikke forklare noe variasjon og ble som følge av dette eliminert.

Videre ble det valgte å eliminere variablene FIRST_TO HOSP_DATETIME_D og FIRST _HOSP ARRIVAL_DATETIME_D). Det var et trilemma mellom disse 2 variablene og INCIDENT_DISPOSITION_CODE, da disse utviste stor samvariasjon fra chi-square. Det ble derfor valgte å beholde den originale variabelen fra Open data, og fjernet derfor heller de to egenproduserte dummy variablene.

Neste steg var å avgjøre om INITIAL_CALL_TYPE eller INITIAL_SEVERITY_LEVEL _CODE skulle elimineres, da de hadde en samvariajson på 0.97. Det ble valgte å beholde INITIAL_SEVERITY LEVEL_CODE da denne variabelen var på et mer aggregert nivå. samt inneholdt variabelen informasjon om differansen mellom hva initial utfall ble, fremfor endelig utfall (FINAL SEVERITY LEVEL CODE)

Den siste faktor variabelen som ble eliminert var INCIDENT_DATETIME_Month, der den hadde over 0.8 korrelasjon med variabelen INCIDENT_DATETIME_week. Det ble valgte å beholde uke variabelen, da denne kan adressere sesong elementer på et akseptabelt aggregert nivå. Der kunnskap om uker kan forklare måneder, og ikke motsatt. Samt at uker kan forklare, eks 4 juli og andre høyaktivitets-perioder i USA, som ikke strekker seg over en månedlig periode.

- Note 44-46 : Eliminerer outliers i INCIDENT_TRAVEL_TM_SECONDS_QY, der det er satt en øvre terskel på maksimum 1800 sekunder/30 minutter. Grunnen til dette, var som følge av at dette er vanlig praksis i EMS departementet for Columbia (FEMS, 2016, s.47). Det antas at denne fremgangsmåten kan overføres til NYC kontekst. Fra den samme kilden henvises det til en benchmark for håndteringstid. Der 99 % av alle håndteringstider skal være under 2 min og 40 sekunder (FEMS, 2016, s.8)[6]. For DISPATCH_RESPONSE_SECONDS_QY fjernes derfor alt av observasjoner som tar over 2min og 40 sek. Ved THROUGHPUT_TIME fjerner vi alle negative verdier, da dette anses som lite meningsfylt å inkludere. Alle observasjoner som tar totalt mer enn 10000 sek/2,78 timer i THROUGHPUT_TIME elimineres.

- Note 47-48 : Rasjonale for eliminering av outliers tilknyttet faktorvariablene, var foranket i at flere av kategoriene innad faktor variablene inneholdt ekstremt få

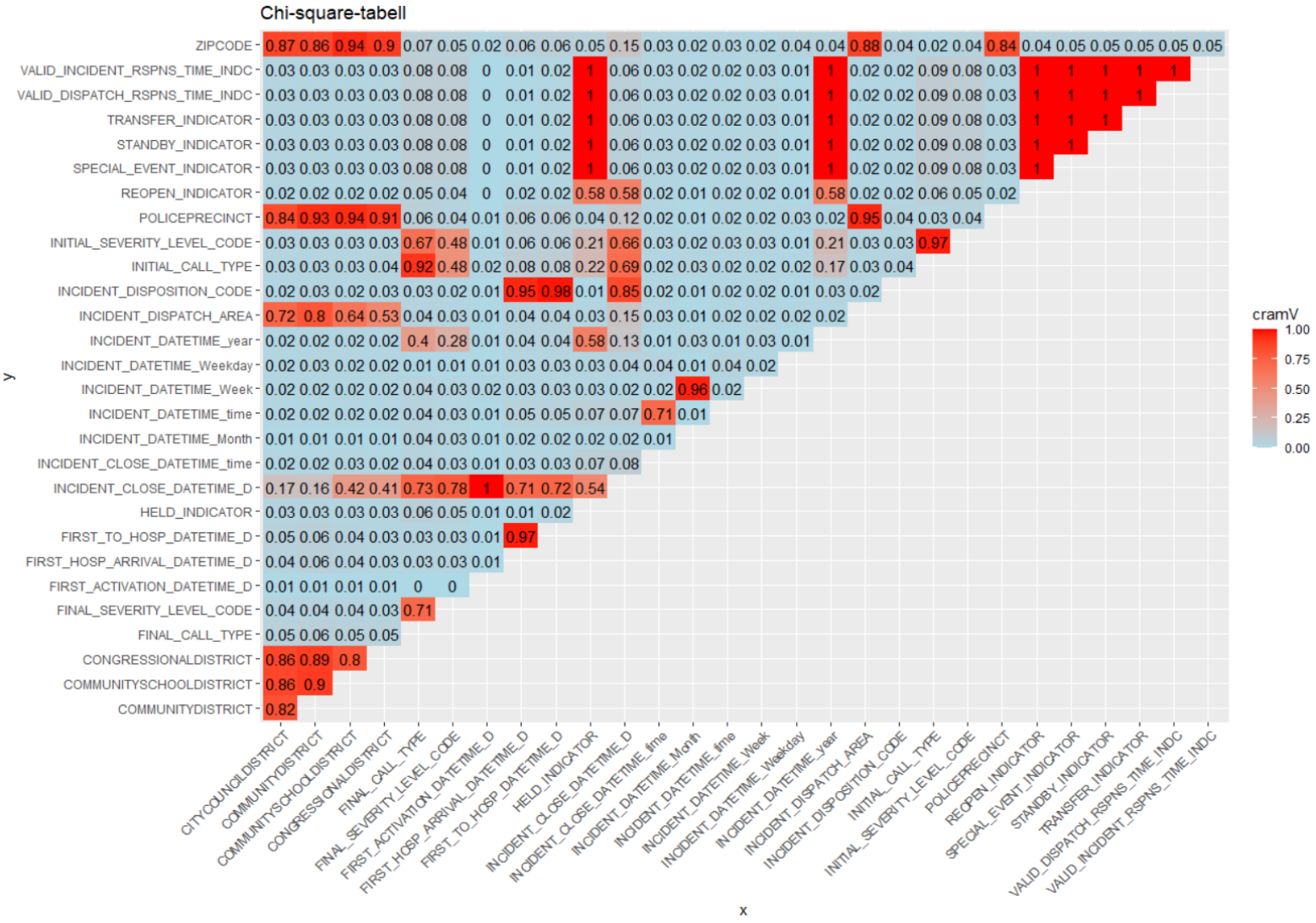
[6]Det kan reises et argument om at eliminering av outliers før splitt av test og trening vil introduserer datalekk. Vi argumenterer for at dette er en uunngåelig konsekvens, i og med at retningslinjene fra FEMS tilsier at visse verdier er av en størrelsesorden som gjør de "uegnet" for prediksjon. *"If this time value results in a vehicle "Travel Time" duration exceeding 1,800 seconds (30 minutes), there is high likelihood the time value was inaccurately recorded"* (FEMS, s.47, 2016).

observasjoner. I tilleg er det antatte registreringsfeil tilknyttet variabelen INCIDENT_DISPOSITION_CODE, der kode 86 ikke er registrert i EMS description (se vedlegg I.8), men fremdeles er observert i datasettet. Denne observasjonen ble derfor eliminert. Klassen 6 og 7 i FINAL_SEVERITY LEVEL_CODE ble eliminert da de begge inneholdt kun 1 observasjon. INCIDENT_DISPOSITION_CODE klasse 92 ble også eliminert da den inneholdt særdeles få observasjoner.

- Note 49 : Fra figur **I.4** ser man at de numeriske variablene har liten korrelasjon, utenom en instans av 0.99 korrelasjon mellom INCIDENT_RESPONSE_SECONDS _QY og INCIDENT_TRAVEL_TM_SECONDS_QY. Dette på grunn av variabelen INCIDENT_RESPONSE _SECONDS_QY innehar to komponenter, som er en sammenslåing av INCIDENT_TRAVEL TM_SECONDS_QY + DISPATCH_RESPONSE SECONDS_QY. INCIDENT_RESPONSE SECONDS_QY velges derfor å fjernes, for å unngå perfekt multikollinearitet. De 2 resterende numeriske variabler beholdes for å bruke i ML-modellen.

I.5 Chi-Square

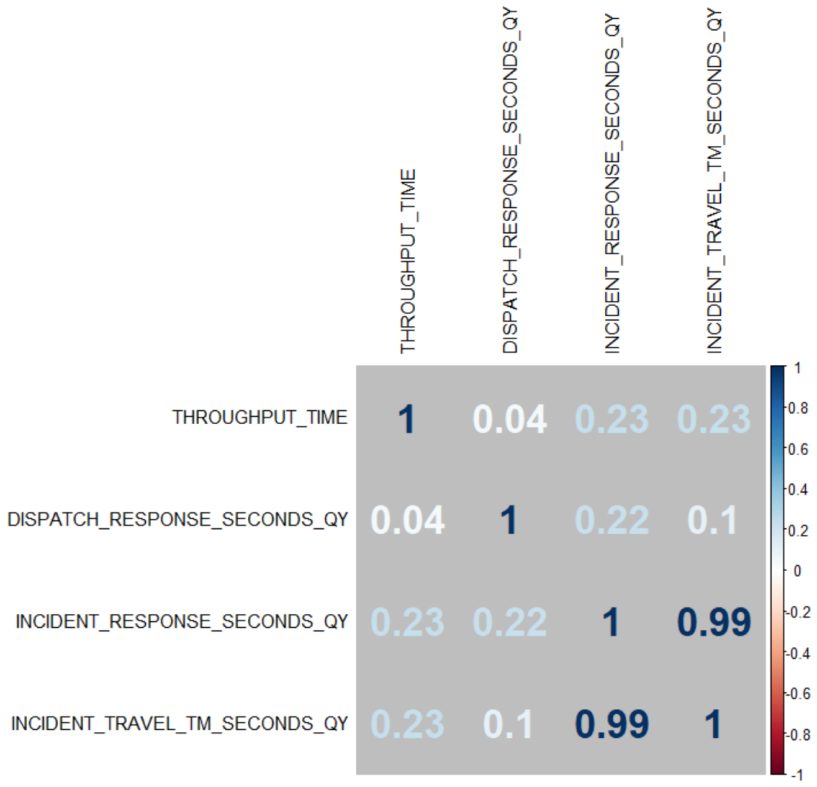
Chi square for datavask tilknyttet ML



Figur I.3: Chi-Square.

I.6 Korrelasjonsmatrise

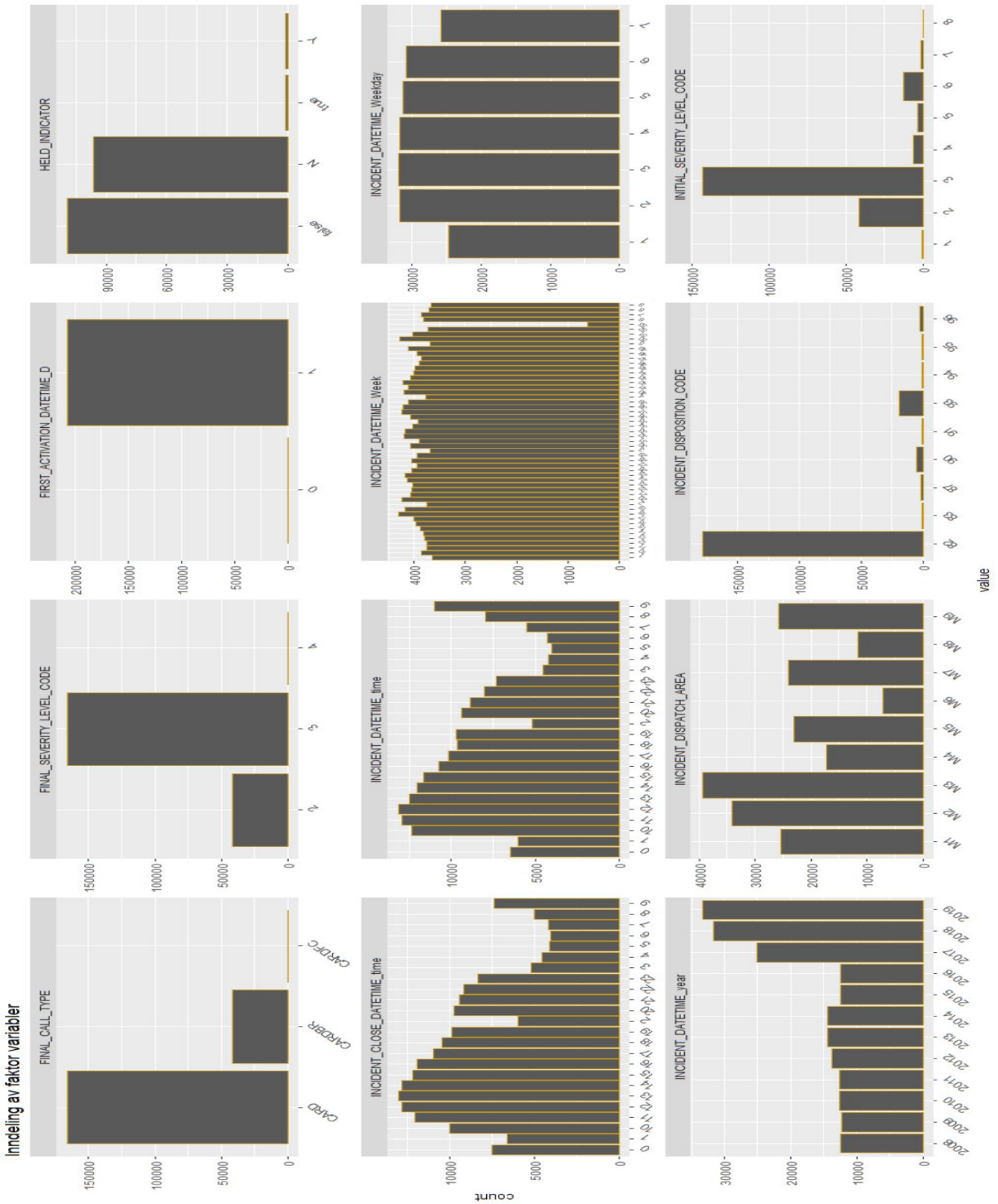
Korrelasjonsmatrise for datavask tilknyttet ML



Figur I.4: Korrelasjonsmatrise.

I.7 Inndeling av Faktor Variabler

For supplerende statistikk tilknyttet (ABT) se figur **3.2**



Figur I.5: Inndeling av Faktor Variabler (ABT).

I.8 Variabel Beskrivelse av Incident Disposition Code

Figur **I.6** henviser til forklaring av de ulike kodene tilknyttet variabelen INCIDENT DISPOSITION CODE.

INCIDENT DISPOSITION CODE	Description
82	Transporting patient
83	Patient pronounced dead
87	Cancelled
90	Unfounded
91	Condition corrected
92	Treated not transported
93	Refused medical aid
94	Treated and transported
95	Triaged at scene no transport
96	Patient gone on arrival
CANCEL	Cancelled
DUP	Duplicate incident
NOTSNT	Unit not sent
ZZZZZZ	No disposition

Figur I.6: Forklaring av Disposisjonskoder. Kilde: NYC OpenData (2020).

Vedlegg II

PM

II.1 Informasjon om Hendelseslogg & Produksjon

Hendelseslogg Det originale datasett har etter preprosessering 215 066 observasjoner og syv aktivitetsvariabler. Dataene blir gruppert på case ID for å lage hendelseslogg. Dette innebærer flere nye rader, som vist i figur **3.1**. Etter omstrukturering til hendelseslogg, innehar loggen 1 505 462[1] observasjoner. Dataene blir oppskalert med syv gangen, da det er syv aktivitetsvariabler[2]. Hendelsesloggen som brukes i analysen innehar 1 443 872 observasjoner. Årsaken til dette frafall fra 1 505 462, er grunnet manglende observasjoner i det originale datasett. En manglende observasjon antas i orginale dataene til å indikere at hendelse ikke har inntruffet da tid ikke er registrert. Etter gruppering på case ID blir disse tomme observasjonene tatt med, disse slettes, da de antas å ikke inneholde en reell aktivitet. Dette gir en logg med 1 443 872. Gjennomsnitt trace lengde er 6.7[3].

Variabler

Variabelen aktivitetsinstans ID produseres. Variabelen strukturerer aktivitetene til en case sekvensielt, på basis av tid (bupaR-event logs, u.å.).

Kommentar til Hendelsesloggen

Særskilt bør det utmerkes hvor sekvensiell dataene er, se **Vedlegg I** seksjon **Figure I.2**. Alle aktiviter er fanget opp i DATETIME variablene. Fra beskrivelsen er det en klar logikk at en aktivitet kommer før en annen. Vi anser dette som normativt, uten å nødvendigvis trekke konklusjoner om "regelbrudd", eller "feilregistrering", om vi skulle se noe som går mot ren intuisjon og sunn fornuft. Eksempelvis, om en case har første aktivitet INCIDENT_CLOSE, så er det ulogisk. Aktiviteten burde være den siste. Vi vil påpeke eventuelle slike funn, men vi har ikke grunnlag for å vite hvorfor. Vi kan bare fremheve plausible hypoteser. Aktivitetenes logiske rekkefølge, som er antatt i

[1] 215066*7.

[2] Se **Vedlegg I.3**, variablene er de med navn datetime. Et registrert tidspunkt indikerer at en aktivitet har inntruffet.

[3] 1.443 872/1.505 462*7. Snitt aktiviteter pr case.

oppgaven, fra første til siste aktivitet; INCIDENT_DATETIME, FIRST_ASSIGNMENT, FIRST_ACTIVATION, FIRST_ON_SCENE, FIRST_TO_HOSP, FIRST_HOSP_ARRIVAL, INCIDENT_CLOSE. Dette fremkommer i variabel beskrivelsen.

II.2 Alfa Pluss Algoritmen

Algoritmen produseres i bupaR ved pakkene pm4py og petrinetR (Janssenswillen, G., Depaire, B., Swennen, M., Jans, M., & Vanhoof, K., 2019). Pm4py er en pakke som gjør at man kjøre python funksjonalitet i R, der Python er et statistisk program, som også har PM funksjonalitet ved pakken pm4py (bupaR-PM4Py, u.å). Ved bruk av R pakken reticulate, så kan man kjøre python funksjonalitet I R (reticulate, u.å). Pakken pm4py har muligheten for å bruke alfa pluss algoritmen fra python, i R. Algoritmen andvendes for å produsere en prosessmodell (Petri net).

II.3 PM Script

Fullstendig Script for PM. R script testet og fungerer i versjon R. 3.5.3

```r

#------------------Hendelseslogg er produsert i R versjon 3.5.3---------------

library(bupaR)
library(tidyr)
library(readr)
library(tidyverse)
library(skimr)
library(pm4py)
library(petrinetR)
library(processmapR)
library(lubridate)


EMS_Incident_Dispatch_Data_endelig_222k <- read.csv("data")

##Skim
EMS_Incident_Dispatch_Data_endelig_222k %>%
  skim()

### Fikse opp Na`S
M_MANHATTAN_CARD_NA<-EMS_Incident_Dispatch_Data_endelig_222k %>% mutate_all(na_if,"")

M_MANHATTAN_CARD_NA %>%
  skim()

###kodeER for å fjærne Na`S [endre på xxx]
# remove na in r - remove rows - na.omit function / option

NA_SELECTED1<-M_MANHATTAN_CARD_NA[complete.cases(M_MANHATTAN_CARD_NA$CONGRESSIONALDISTRICT),
NA_SELECTED2<-NA_SELECTED1[complete.cases(NA_SELECTED1$INCIDENT_RESPONSE_SECONDS_QY),]
NA_SELECTED3<-NA_SELECTED2[complete.cases(NA_SELECTED2$INCIDENT_DISPOSITION_CODE),]
NA_SELECTED4<-NA_SELECTED3[complete.cases(NA_SELECTED3$COMMUNITYSCHOOLDISTRICT),]
NA_SELECTED5<-NA_SELECTED4[complete.cases(NA_SELECTED4$ZIPCODE),]

NA_SELECTED5 %>%
  skim()

## Det endelig datasett prepp
EMS_Incident_Dispatch_Data <-NA_SELECTED5


#-----------------eventlogg----------------------#----------------------------

#SORTERER CASE, AKTIVITET og TIMESTAMP
#fetch id samt 7x aktivitetsvariabler fra open data - data
newdata <- EMS_Incident_Dispatch_Data[c(1,2,7,10,11,15,16,17)]
#lag nye variabler aktivitet og tid i nytt datasett newdata
test<-gather(newdata, aktivitet, tid, -1)

#sortering activity id

df2<-as.data.frame(test)

#sorter på tid, PRIMÆR SORTERING ER PÅ ID, SKUNDÆR ER PÅ TID,
#DETTE FOR Å FÅ KORREKT AKTIVITY ID FOR BUPAR KOMPITABILITET
#VI HAR EN ENKEL AKT ID DA EN CASE ER FERDIG NÅR DEN ER FERDIG OG...
#VI HAR DA BARE EN ENKEL TALLREKKE FOR TRACES 1,2,3,4 ETC FOR HVER UTFØRTE CASE.
```

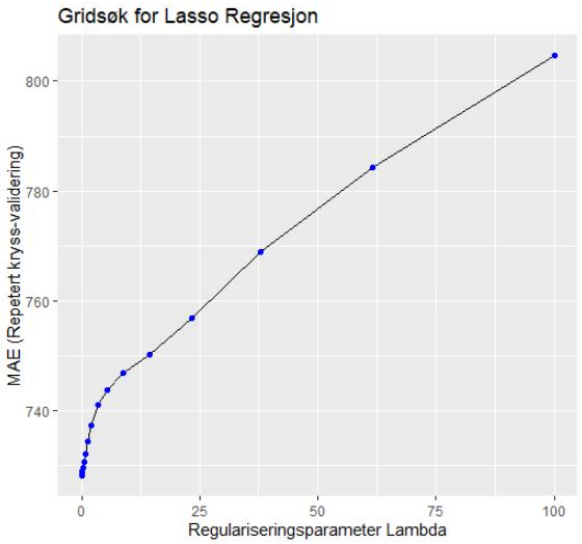
```r
df2<-df2[order(df2$CAD_INCIDENT_ID),] #TID SEKVENSIElT SORTERT UNDER HVER ENKELT UNIKE ID

#legg til activity ID
df2$ID <- rownames(test)
#---------------------------------------------- TRANSFORMER TIL TIDSFORMAT----
df3<-mdy_hms(df2$tid)

eventlogg<-df2

eventlogg$tid <- df3

#----------------------------------------------  PRODUSER EVENTLOGG -----------
#-------------------------------------------------------------------------------
#EVENTLOG#
summary(eventlogg)

eventlogg<-eventlogg[complete.cases(eventlogg$tid),] #fjern nas

skim(eventlogg)

log<-eventlogg %>%
  mutate(status = "complete",
         activity_instance = 1:nrow(.)) %>%#mangler transitional lifecycle
  mutate(resource = NA) %>% #mangler ressurs variabel
  eventlog(
    case_id = "CAD_INCIDENT_ID",
    activity_id = "aktivitet",
    timestamp = "tid",
    activity_instance_id = "ID",
    lifecycle_id = "status",
    resource_id = "resource"
  )

mapping(log)
log %>% summary

#----------------------- STATISTIKK-----------------------------------------

#traces
log %>%
  trace_coverage("trace") %>%
  plot()
#boxplot gjenstrømtid
log %>%
  throughput_time(level = "log", units = "hours") %>%
  plot
#fuzzy absolutte tall
log %>%
  process_map()
#fuzzy sekunder avg
log %>%
  process_map(performance(median, "secs"))

#--------------------ALFA ALGORITME-----------------------------------------

#alpha plus

discovery_alpha(log, variant = variant_alpha_plus()) -> PN
PN$petrinet %>% render_PN()

PN %>% str

evaluation_all(log, PN$petrinet, PN$intial_marking, PN$final_marking)


#-----------------------SLUTT-----------------------------------------------
```

Figur II.1: PM Script.

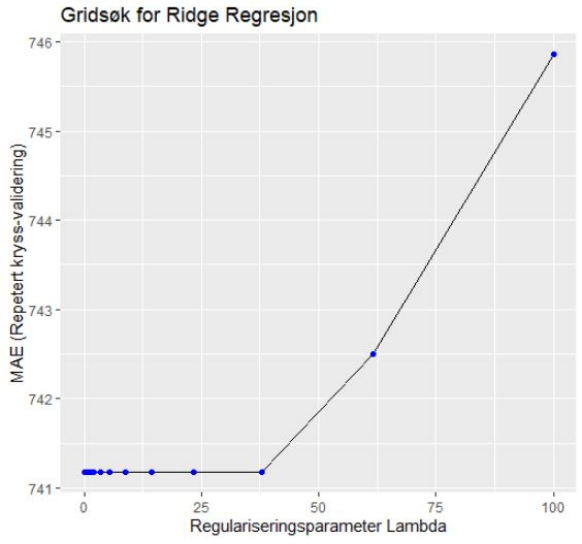
Vedlegg III

ML

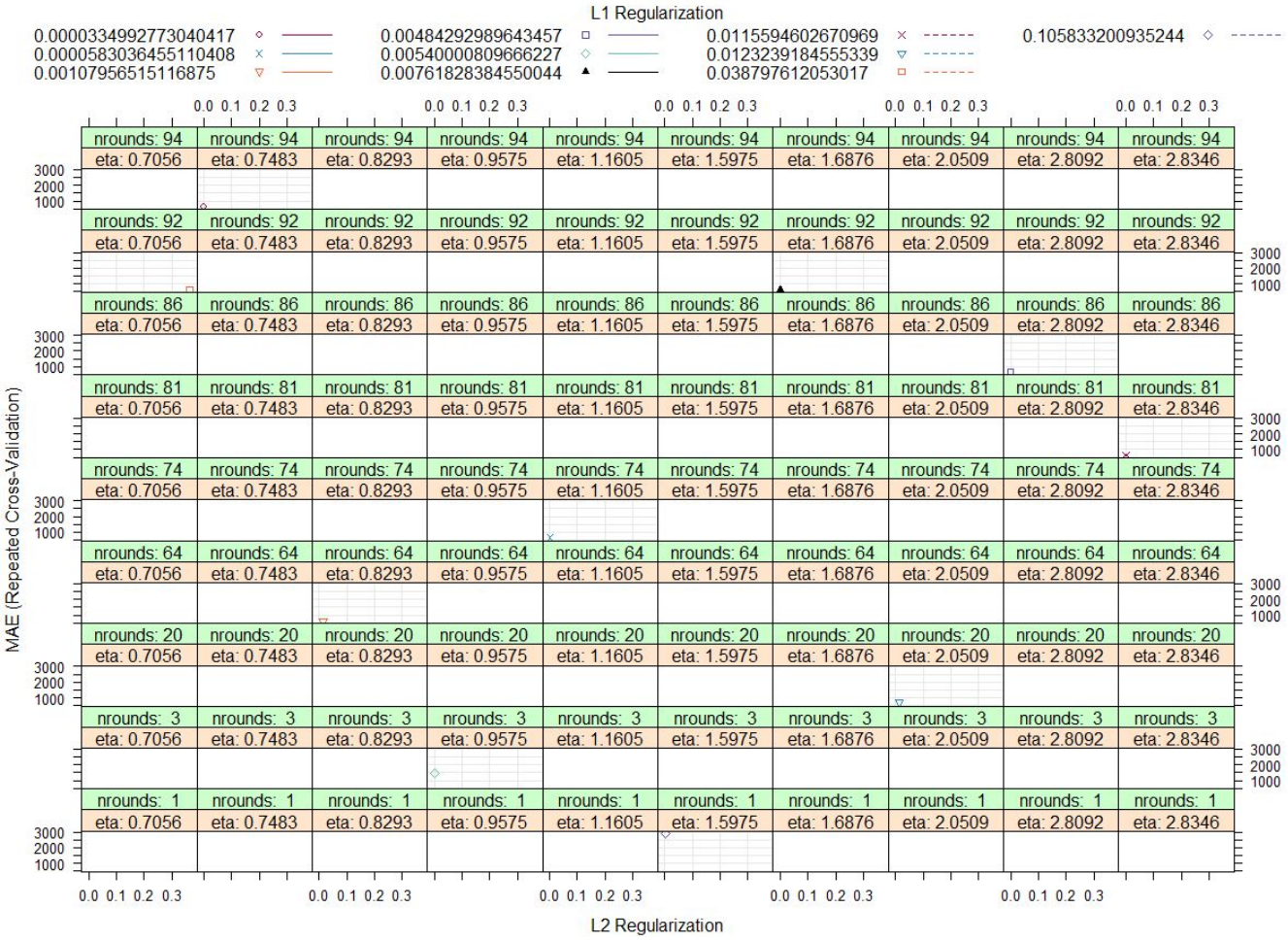
III.1 Grid & Tilfeldig Søk



Figur III.1: Grid Søk for Lasso.



Figur III.2: Grid Søk for Ridge.



Figur III.3: Tilfeldig Søk for XGboost.

III.2 ML Script

Fullstendig Script for ML. R script testet og fungerer i versjon R-3.6.1

```r
# Bibliotek og innlastning av data ----
#Pacman: Laster inn alle pakker og installerer.
pacman::p_load(pacman, bupaR, readr, skimr, dplyr, corrplot, purrr, tidyr, tidyverse, hrbrthemes, viridis, lsr, plotrix,
               car, sjPlot, glmnet, caret, moments, e1071, tigris, leaflet, sp, ggmap, maptools,
               lubridate, corrr, broom, httr, rgdal, zipcode, sf, rio, ggspatial, ggforce, maps, ggthemes, psycho, tibble,
               plyr, ggplot2, GGally, mlbench, RCurl, AppliedPredictiveModeling)

#Denne funksjonen lar R gi oss tall uten E format.
options(scipen = 999)

#Laste inn CSV file
M_MANHATTAN_CARD <- read_csv("C:/Users/Jonas Jensen/Desktop/R/Datasett/M_MANHATTAN_CARD.csv",
                             col_types = cols(CAD_INCIDENT_ID = col_character(),
                                              CITYCOUNCILDISTRICT = col_character(),
                                              COMMUNITYDISTRICT = col_character(),
                                              COMMUNITYSCHOOLDISTRICT = col_character(),
                                              CONGRESSIONALDISTRICT = col_character(),
                                              DISPATCH_RESPONSE_SECONDS_QY = col_number(),
                                              FINAL_SEVERITY_LEVEL_CODE = col_character(),
                                              INCIDENT_DISPOSITION_CODE = col_character(),
                                              INCIDENT_RESPONSE_SECONDS_QY = col_number(),
                                              INCIDENT_TRAVEL_TM_SECONDS_QY = col_number(),
                                              INITIAL_SEVERITY_LEVEL_CODE = col_character(),
                                              POLICEPRECINCT = col_character(),
                                              ZIPCODE = col_character()))
#Oversikt av data
skim(M_MANHATTAN_CARD)

# Dataprepp ------------------------------------------------------------------

# Gjør om alle char til factor variabler.
M_MANHATTAN_CARD <- as.data.frame(unclass(M_MANHATTAN_CARD))

# Ordner  med NA.
M_MANHATTAN_CARD_NA<-M_MANHATTAN_CARD %>% mutate_all(na_if,"")

###Datavask - Fikse NAs
NA_SELECTED1<-M_MANHATTAN_CARD_NA[complete.cases(M_MANHATTAN_CARD_NA$CONGRESSIONALDISTRICT),]
NA_SELECTED2<-NA_SELECTED1[complete.cases(NA_SELECTED1$INCIDENT_RESPONSE_SECONDS_QY),]
NA_SELECTED3<-NA_SELECTED2[complete.cases(NA_SELECTED2$INCIDENT_DISPOSITION_CODE),]
NA_SELECTED4<-NA_SELECTED3[complete.cases(NA_SELECTED3$COMMUNITYSCHOOLDISTRICT),]
NA_SELECTED5<-NA_SELECTED4[complete.cases(NA_SELECTED4$ZIPCODE),]

## Datasett prepp, for tidsvariabel transformasjon. Gir Datasettet passende navn.
Generell_data_prep <-NA_SELECTED5


# Gjøre om factor til tidselementer.

###1
tiddata1<-mdy_hms(Generell_data_prep$INCIDENT_DATETIME)
Generell_data_prep$INCIDENT_DATETIME<-tiddata1

###2
tiddata2<-mdy_hms(Generell_data_prep$FIRST_ASSIGNMENT_DATETIME)
Generell_data_prep$FIRST_ASSIGNMENT_DATETIME<-tiddata2
```

```r

###3
tiddata3<-mdy_hms(Generell_data_prep$FIRST_ACTIVATION_DATETIME)
Generell_data_prep$FIRST_ACTIVATION_DATETIME<-tiddata3

###4
tiddata4<-mdy_hms(Generell_data_prep$FIRST_ON_SCENE_DATETIME)
Generell_data_prep$FIRST_ON_SCENE_DATETIME<-tiddata4

###5
tiddata5<-mdy_hms(Generell_data_prep$FIRST_TO_HOSP_DATETIME)
Generell_data_prep$FIRST_TO_HOSP_DATETIME<-tiddata5

###6
tiddata6<-mdy_hms(Generell_data_prep$FIRST_HOSP_ARRIVAL_DATETIME)
Generell_data_prep$FIRST_HOSP_ARRIVAL_DATETIME<-tiddata6

###7
tiddata7<-mdy_hms(Generell_data_prep$INCIDENT_CLOSE_DATETIME)
Generell_data_prep$INCIDENT_CLOSE_DATETIME<-tiddata7


# Fikse opp i kommafeil mtp zipkodene -> Da dette anses som input error.
# For å få riktig kartdata, har ikke effekt på PM og ML.
ZIPCODE <- gsub(',', '', Generell_data_prep$ZIPCODE)
Generell_data_prep$ZIPCODE <-ZIPCODE
Generell_data_prep <- as.data.frame(unclass(Generell_data_prep))



# Datasett med rikitg datakoloner
Endeligpreppa_riktig_datakol<-Generell_data_prep


# Ferdig med pre prossesering
Endeligpreppa_riktig_datakol %>%
  skim()


# Konstruksjon av nye features/variabler ----

#Konstruksjon av Y totaltid
Endeligpreppa_riktig_datakol$THROUGHPUT_TIME <- difftime(Endeligpreppa_riktig_datakol$INCIDENT_CLOSE_DATETIME,
                                                         Endeligpreppa_riktig_datakol$INCIDENT_DATETIME, units = "secs")

Endeligpreppa_riktig_datakol$THROUGHPUT_TIME<-as.numeric(Endeligpreppa_riktig_datakol$THROUGHPUT_TIME)

#Dummy for NAs i FIRST_TO_HOSP_DATETIME ----- FIRST_HOSP_ARRIVAL_DATETIME (Fjerne tid varibaler, NAs substituert og forklart i dummy)
Endeligpreppa_riktig_datakol$FIRST_ACTIVATION_DATETIME_D <- ifelse(is.na(Endeligpreppa_riktig_datakol$FIRST_ACTIVATION_DATETIME), 0, 1)
Endeligpreppa_riktig_datakol$FIRST_TO_HOSP_DATETIME_D <- ifelse(is.na(Endeligpreppa_riktig_datakol$FIRST_TO_HOSP_DATETIME), 0, 1)
Endeligpreppa_riktig_datakol$FIRST_HOSP_ARRIVAL_DATETIME_D <- ifelse(is.na(Endeligpreppa_riktig_datakol$FIRST_HOSP_ARRIVAL_DATETIME), 0, 1)
Endeligpreppa_riktig_datakol$INCIDENT_CLOSE_DATETIME_D <- ifelse(is.na(Endeligpreppa_riktig_datakol$INCIDENT_CLOSE_DATETIME), 0, 1)
```

```r

#Lager faktor variabel for basis av posixct variabler
Endeligpreppa_riktig_datakol$FIRST_ACTIVATION_DATETIME_D <- factor(Endeligpreppa_riktig_datakol$FIRST_ACTIVATION_DATETIME_D)
Endeligpreppa_riktig_datakol$FIRST_TO_HOSP_DATETIME_D <- factor(Endeligpreppa_riktig_datakol$FIRST_TO_HOSP_DATETIME_D)
Endeligpreppa_riktig_datakol$FIRST_HOSP_ARRIVAL_DATETIME_D <- factor(Endeligpreppa_riktig_datakol$FIRST_HOSP_ARRIVAL_DATETIME_D)
Endeligpreppa_riktig_datakol$INCIDENT_CLOSE_DATETIME_D <- factor(Endeligpreppa_riktig_datakol$INCIDENT_CLOSE_DATETIME_D)

#Tilegner datasett x for enkelt arbeid.
x<-Endeligpreppa_riktig_datakol

x$INCIDENT_DATETIME_year<-as.POSIXlt(tiddata1)$year + 1900
x$INCIDENT_DATETIME_time<-as.POSIXlt(tiddata1)$hour
x$INCIDENT_CLOSE_DATETIME_time<-as.POSIXlt(tiddata7)$hour
x$INCIDENT_DATETIME_Weekday<-wday(x$INCIDENT_DATETIME)
x$INCIDENT_DATETIME_Month<-month(x$INCIDENT_DATETIME)
x$INCIDENT_DATETIME_Week<-week(x$INCIDENT_DATETIME)

##Faktor variabel transformasjon
x$INCIDENT_DATETIME_year<-factor(x$INCIDENT_DATETIME_year)
x$INCIDENT_DATETIME_time<-factor(x$INCIDENT_DATETIME_time)
x$INCIDENT_CLOSE_DATETIME_time<-as.factor(x$INCIDENT_CLOSE_DATETIME_time)
x$INCIDENT_DATETIME_Weekday<-factor(x$INCIDENT_DATETIME_Weekday)
x$INCIDENT_DATETIME_Month<-factor(x$INCIDENT_DATETIME_Month)
x$INCIDENT_DATETIME_Week<-factor(x$INCIDENT_DATETIME_Week)


# Fjerner variabler som ikke forklarer variasjon ----

#Eliminerer resterende Posixct variabler, da dummyer er skapt.
x <- select(x, -INCIDENT_DATETIME,-FIRST_ASSIGNMENT_DATETIME,-FIRST_ACTIVATION_DATETIME,
            -FIRST_ON_SCENE_DATETIME,-FIRST_TO_HOSP_DATETIME,
            -FIRST_HOSP_ARRIVAL_DATETIME,-INCIDENT_CLOSE_DATETIME)

#Eliminerer CadID og Borough, da disse ikke forklarer noe variasjon i datasettet.
x <- select(x, -CAD_INCIDENT_ID,-BOROUGH)

#Dropper NAs i througput time, da vi ikke ønsker missing values i avhengig varibaler.

x<-x[complete.cases(x$THROUGHPUT_TIME),]

#Redusert til 33 variabler og 215019 observasjoner
skim(x)


# Eliminering av variabler ved hjelp av chi-square ----

#Subsetter for å isloere  kategoriske variabler
z <- select(x, -DISPATCH_RESPONSE_SECONDS_QY,-INCIDENT_RESPONSE_SECONDS_QY,
            -INCIDENT_TRAVEL_TM_SECONDS_QY,-THROUGHPUT_TIME)


#Z er dataset

#Lapply transformasjon for ikke å få problemer med NAs.
z[] <- lapply(z,as.integer)

```

```r
# Funksjon for å få cramers v (Chi square)
f = function(x,y){
  tbl = z %>% select(x,y) %>% table()
  cramV = round(cramersV(tbl), 2)
  data.frame(x, y, cramV, cramV)}

# Lage kombo datasett av variabel navn
# Sorter for å få bedre grafikk
df_comb = data.frame(t(combn(sort(names(z)), 2)), stringsAsFactors = F)


# Bruke funksjon for hver varibelnavn kombo for cramer v
df_res = map2_df(df_comb$X1, df_comb$X2, f)

# Plot resultater
df_res %>%
  ggplot(aes(x,y,fill=cramV))+
  geom_tile()+
  geom_text(aes(x,y,label=cramV))+
  scale_fill_gradient(low="lightblue", high="red")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  ggtitle("Chi-square-tabell")

#Eliminering av variabeler som er 80 % terskel mtp chi-square

z <- select(z, -REOPEN_INDICATOR,-VALID_DISPATCH_RSPNS_TIME_INDC,-VALID_INCIDENT_RSPNS_TIME_INDC,
            -SPECIAL_EVENT_INDICATOR,-TRANSFER_INDICATOR,-STANDBY_INDICATOR)

z <- select(z, -POLICEPRECINCT,-CITYCOUNCILDISTRICT,-COMMUNITYDISTRICT,
            -COMMUNITYSCHOOLDISTRICT,-CONGRESSIONALDISTRICT)

z <- select(z, -ZIPCODE)

z<- select(z, -INCIDENT_CLOSE_DATETIME_D)

z<- select(z, -FIRST_TO_HOSP_DATETIME_D,-FIRST_HOSP_ARRIVAL_DATETIME_D)

z<- select(z, -INITIAL_CALL_TYPE)

z<- select(z, -INCIDENT_DATETIME_Month)



# Y, fjerner basert på hva vi fant i chi square. Dette gjøres på datasett Y
#For å redusere arbeid mtp transformasjoner fra tidligere.
y<-x

y <- select(y, -REOPEN_INDICATOR,-VALID_DISPATCH_RSPNS_TIME_INDC,-VALID_INCIDENT_RSPNS_TIME_INDC,
            -SPECIAL_EVENT_INDICATOR,-TRANSFER_INDICATOR,-STANDBY_INDICATOR)

y <- select(y, -POLICEPRECINCT,-CITYCOUNCILDISTRICT,-COMMUNITYDISTRICT,-COMMUNITYSCHOOLDISTRICT,-CONGRESSIONALDISTRICT)


### OM ML eller PM skal andvendes fjern denne variabelen, om kart skal produseres, la denne variabelen stå.
y <- select(y, -ZIPCODE)
####
```

```
221
222 y<- select(y, -INCIDENT_CLOSE_DATETIME_D)
223
224 y<- select(y, -FIRST_TO_HOSP_DATETIME_D,-FIRST_HOSP_ARRIVAL_DATETIME_D)
225
226 y<- select(y, -INITIAL_CALL_TYPE)
227
228 y<- select(y, -INCIDENT_DATETIME_Month)
229
230 ##Oversikt
231 skim(y)
232
233
234 # Eliminering av Outliers ----
235
236 #Navnlegger datasett
237 Outliers_Elimnere<-y
238
239
240 ##Først eliminere Numeriske outliers.
241 Outliers_Elimnere <- subset(Outliers_Elimnere, INCIDENT_TRAVEL_TM_SECONDS_QY<=1800)  #obs: 214213
242 Outliers_Elimnere <- subset(Outliers_Elimnere, DISPATCH_RESPONSE_SECONDS_QY<=160)    #obs: 208680
243 Outliers_Elimnere <- subset(Outliers_Elimnere, THROUGHPUT_TIME>0)                    #obs: 208676
244 Outliers_Elimnere <- subset(Outliers_Elimnere, THROUGHPUT_TIME<=10000)               #obs: 208512
245
246
247 ###Eliminere verdier med 0 og negative verdier.
248 Outliers_Elimnere <- subset(Outliers_Elimnere, DISPATCH_RESPONSE_SECONDS_QY>0)   #207606
249 Outliers_Elimnere <- subset(Outliers_Elimnere, INCIDENT_TRAVEL_TM_SECONDS_QY>0)  #207350
250
251
252 #86 Eksistere ikke i systemet.
253 Outliers_Elimnere<-Outliers_Elimnere[!(Outliers_Elimnere$INCIDENT_DISPOSITION_CODE=="86"),] #obs: 207349
254
255
256 ## Eliminerer klasser med 1 observasjon
257 Outliers_Elimnere<-Outliers_Elimnere[!(Outliers_Elimnere$FINAL_SEVERITY_LEVEL_CODE=="6"),]  #obs: 207348
258 Outliers_Elimnere<-Outliers_Elimnere[!(Outliers_Elimnere$FINAL_SEVERITY_LEVEL_CODE=="7"),]  #obs: 207347
259 Outliers_Elimnere<-Outliers_Elimnere[!(Outliers_Elimnere$INCIDENT_DISPOSITION_CODE=="92"),] #obs: 207346
260
261
262 ##Korplott, arguemtasjon for å fjerne INCIDENT_RESPONSE_SECONDS_QY
263 Outliers_Elimnere_corr<-cor(Outliers_Elimnere[c("THROUGHPUT_TIME",
264                                                 "DISPATCH_RESPONSE_SECONDS_QY",
265                                                 "INCIDENT_RESPONSE_SECONDS_QY",
266                                                 "INCIDENT_TRAVEL_TM_SECONDS_QY")])
267
268 corrplot(Outliers_Elimnere_corr, method = "number" , number.cex = 2.10, tl.col = "black", bg = "gray")
269
270
271 #Det ses fra  korrplott: Høy korrelasjon INCIDENT_RESPONSE_SECONDS_QY
272 Outliers_Elimnere<- select(Outliers_Elimnere, -INCIDENT_RESPONSE_SECONDS_QY)
273
274
275 #Analytisk basetabel(ABT)/Datasett klar for test og trening splitt.
276 ABT<-Outliers_Elimnere  ##obs: 207436
277
```

```r
skim(ABT)


# EDA på ABT datasett ----

##Faktor variabel statistikk

#Faktor variabel distrubisjon
ABT %>%
  keep(is.factor) %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +
  geom_bar(color="#C4961A") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Inndeling av faktor variabler ")


# For 1 Enkelt faktor variabel
ggplot(data=ABT, aes(x=INCIDENT_DISPOSITION_CODE)) +
  geom_bar(fill = '#C4961A') +
  geom_text(stat='count', aes(label=..count..), position=position_dodge(width=3), size=3.5, hjust = 0) +
  theme(
    legend.position="none",
    plot.title = element_text(size=11)) +
  ggtitle("Sparse classes for factor variabel")


#Chi Square
#Subsett for å kategorisere variabler
zz <- select(ABT, -DISPATCH_RESPONSE_SECONDS_QY,-INCIDENT_TRAVEL_TM_SECONDS_QY,-THROUGHPUT_TIME)

#ZZ er datasett

#Lapply transformasjon.
zz[] <- lapply(zz,as.integer)

# Funksjon for å få cramers v
f = function(x,y){
  tbl = zz %>% select(x,y) %>% table()
  cramV = round(cramersV(tbl), 2)
  data.frame(x, y, cramV, cramV)}

# Lage kombo datasett av variabel navn
# Sorter for å få beder grafikk
df_comb = data.frame(t(combn(sort(names(zz)), 2)), stringsAsFactors = F)


# Bruke funksjon for hver varibelnavn kombo for cramer v
df_res = map2_df(df_comb$X1, df_comb$X2, f)
```

```
328
329 # Plot resultater
330 df_res %>%
331   ggplot(aes(x,y,fill=cramV))+
332   geom_tile()+
333   geom_text(aes(x,y,label=cramV))+
334   scale_fill_gradient(low="lightblue", high="red")+
335   theme(axis.text.x = element_text(angle = 45, hjust = 1))+
336   ggtitle("Chi-square-tabell")
337
338
339 #Numerisk variabel statistikk#
340
341 #Korrelasjonsmatrise
342 ABT_corr<-cor(ABT[c("THROUGHPUT_TIME","DISPATCH_RESPONSE_SECONDS_QY","INCIDENT_TRAVEL_TM_SECONDS_QY")])
343
344 corrplot(ABT_corr, method = "number" , number.cex = 2.10, tl.col = "black", bg = "lightblue")
345
346 ###Density
347 ABT %>%
348   keep(is.numeric) %>%
349   gather() %>%
350   ggplot(aes(value)) +
351   facet_wrap(~ key, scales = "free") +
352   geom_density(color="#C4961A",fill="lightblue") +
353   xlab("Sekunder")
354
355
356 #Konstruerer feature boxplot for FINAL_SEVERITY_LEVEL_CODE [Bytt ut etter $ med ønsket variabel]
357 featurePlot(x = select(ABT,c(4:5, 9)),
358             y = ABT$FINAL_SEVERITY_LEVEL_CODE,
359             plot = "box",
360             ## bwplot() alternativ.
361             scales = list(y = list(relation="free"),
362                           x = list(rot = 90)),
363             layout = c(3,1 ),
364             auto.key = list(columns = 2))
365
366
367
368 #Spatial data/Kart data statistikk#
369
370
371 #Fjerne zipcodes som ikke ligger i Manhattan.
372 ABT_ZIP<-Outliers_Elimnere[!(Outliers_Elimnere$ZIPCODE=="11224"),]  #obs: 207345
373 ABT_ZIP<-ABT_ZIP[!(ABT_ZIP$ZIPCODE=="10301"),]  #obs: 207344
374
375
376 ## 89 unike zipcoder, som stemmer etter de 2 outlierne er fjernet (91-2)
377 skim(ABT_ZIP)
```

```
378
379
380 ###Trekk kun ut unike zipcoder.
381 ABT_ZIP<-unique(ABT_ZIP$ZIPCODE)
382
383 ABT_ZIP<-as.data.frame(ABT_ZIP)
384
385
386 ###Rename for merge match
387 names(ABT_ZIP)[1] <- "zip"
388
389 ###Merger - 1 observasjoner forsvant
390 ZC = data.frame(ABT_ZIP)
391
392 ## Data registeret for zip (må selv inserte den som mangler her, som var zip 10000)
393 data(zipcode)
394
395 # Data Frame - Steg 1
396 registerinnput <- data.frame(10000, "New York", "NY", 40.74754, -73.98336) #Long og Lat til zip 10000
397
398 #Navngir koloner i  Data Frame - Steg 2
399 names(registerinnput) <- c("zip", "city", "state", "latitude", "longitude")
400
401 #Bruker rbind() function for å sette inn observasjoner.
402 zipcodenew <- rbind(zipcode, registerinnput)
403
404 ## Redusere dimensjoner
405 zipcodenew1 <- subset(zipcodenew, zip>=10000)
406
407 zipcodenew2 <- subset(zipcodenew1, zip<11225)
408
409 done <-merge(ZC, zipcodenew2)
410
411 endeliginputkart<-done[c(4,5)]
412
413 attach(ZC)
414
415 ### Legger til data for de ulike NYC EMS stasjonene i Manhatten.
416 ## Stasjonenes koordinater funnet via google maps.
417 # Data Frame - Steg 1
418 FDNY_EMS_Stasjoner <- data.frame("4", 40.710091, -73.986206)
419
420 #Navngir koloner i Data Frame - Steg 2
421 names(FDNY_EMS_Stasjoner) <- c("StasjonsNummer", "latitude", "longitude")
422
423
424 #Konstruere
425 nRow7 <- data.frame(StasjonsNummer = "7", latitude = 40.747750, longitude = -74.005016)
426 nRow8 <- data.frame(StasjonsNummer = "8", latitude = 40.738126, longitude = -73.975211)
427 nRow10 <- data.frame(StasjonsNummer = "10", latitude = 40.784418, longitude = -73.942853)
428 nRow13 <- data.frame(StasjonsNummer = "13", latitude = 40.842594, longitude = -73.935350)
429 nRow16 <- data.frame(StasjonsNummer = "16", latitude = 40.813915, longitude = -73.938140)
```

```
430
431 #Kombinere
432 FDNY_EMS_Stasjoner <- rbind(FDNY_EMS_Stasjoner,nRow7)
433 FDNY_EMS_Stasjoner <- rbind(FDNY_EMS_Stasjoner,nRow8)
434 FDNY_EMS_Stasjoner <- rbind(FDNY_EMS_Stasjoner,nRow10)
435 FDNY_EMS_Stasjoner <- rbind(FDNY_EMS_Stasjoner,nRow13)
436 FDNY_EMS_Stasjoner <- rbind(FDNY_EMS_Stasjoner,nRow16)
437
438
439 FDNY_EMS_StasjonerX <- FDNY_EMS_Stasjoner
440
441
442 ####Nå over til å skape kartet
443
444 ### Last in shapefile
445 ##Shapefile: https://data.cityofnewyork.us/Transportation/Citywide-Mobility-Survey-Survey-Zones/4xfb-z29j
446 shape_file <- st_read("C:/Users/Jonas Jensen/Desktop/R datasett/spatial/Citywide Mobility Survey
447                       - Survey Zones/geo_export_d19aef61-2104-49d0-9cec-51d30b046e14.shp")
448
449
450 ##Lage kart
451 ggplot() +
452   geom_sf(data = shape_file, size = 1, color = "black", fill = "grey") +
453   ggtitle("Manhattan med kombinasjon av postkoder og FDNY EMS stasjoner") +
454   geom_point(data = endeliginputkart, aes(x = longitude, y = latitude), size = 1.2,
455              shape = 19, color = "blue") +
456   geom_point(data = FDNY_EMS_StasjonerX, aes(x = longitude, y = latitude), size = 3,
457              shape = 17, color = "red") +
458   annotation_north_arrow(location = "bl", which_north = "true",
459                          pad_x = unit(0.1, "in"), pad_y = unit(9, "in"),
460                          style = north_arrow_fancy_orienteering) +
461   annotation_scale(location = "tr", width_hint = 0.3) +
462   theme(panel.background = element_rect(fill = 'lightblue', colour = 'black'))
463 coord_sf()
464
465
466
467
468 # Splitting og transformering av data ----
469
470 # ML er utført via CARET pakken.
471
472 #Splitting av data til test og trening. Set seed for randomisert trekk.
473 # creatdatapartion vil stratifisere aoutomatisk (Proporsjonalitet).
474 set.seed(13678)
475 trainIndex <- createDataPartition(ABT$THROUGHPUT_TIME, p = .8,
476                                   list = FALSE,
477                                   times = 1)
478
479
480 trainingdata <- ABT[ trainIndex,]
481 testdata  <- ABT[-trainIndex,]
```

```r

# Pre-arbeid av ulike modeller ----

#Oppsett på hvordan modellen skal trene/fit [Bruk av grid]
Fitting <- trainControl(method = "repeatedcv",
                                number = 10,
                                repeats = 10,
                                search = "grid")

##Grid oppsett.
# Vi spesifiserer at vi skal teste 20 lambda verdier i rekkevidden 10^-2 til 10^2 = [0,001 til 100]
lambdagrid <- 10^seq(-2, 2, length = 20)
plot(lambdagrid)


# Trening av modeller ----

# Multiple linear regression (MLR)
MLR <- train(THROUGHPUT_TIME ~ ., data = trainingdata, method = "lm",
             trControl = trainControl("repeatedcv", number = 10 , repeats=10),
             preProcess = c("range"), # Spesifierer normalisering av numeriske verider. [0-1]
             metric = "MAE")


#Model koeffesienter.
coef(MLR$finalModel)


#Optimal Hyperparameter.
MLR


# Gjør prediksjoner (Fitting data)
predictionsMLR <- predict.train(object= MLR, newdata = testdata)

# Fitstats (Testsettscore)
data.frame(
  RMSE = RMSE(predictionsMLR, testdata$THROUGHPUT_TIME),
  Rsquare = R2(predictionsMLR, testdata$THROUGHPUT_TIME) ,
  MAE = MAE(predictionsMLR, testdata$THROUGHPUT_TIME))




#RIDGE med grid search.
Ridge <- train(
  THROUGHPUT_TIME ~., data = trainingdata, method = "glmnet",
  trControl = Fitting, # Her spesifiserer vi grunnlag for hvordan modellen skal konstrueres/fit
  tuneGrid = expand.grid(alpha = 0, lambda = lambdagrid), ##Alpha 0 = Ridge
  preProcess = c("range"), # Spesifierer normalisering av numeriske verider. [0-1]
  metric = "MAE") ## Spessifisere hvilke metrik vi optimaliserer etter
```

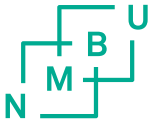
```r

## Visualisering av Grid for Ridge mtp Lambda seleksjon.
ggplot(Ridge) + geom_point(color = "blue") + #Viser til hyperparameterenes kombinasjon og MAE.
  labs(title= "Grid søk for Ridge regresjon",
       y="MAE (Repetert kryss-validering)", x = "Regulariseringsparameter Lambda")


# #Model koeffesienter.
coef(Ridge$finalModel, Ridge$bestTune$lambda)

#Optimal Hyperparameter.
Ridge


# Gjør prediksjoner (Fitting data)
predictionsRIDGE <- predict.train(object= Ridge, newdata = testdata)

# Fitstats (Testsettscore)
data.frame(
  RMSE = RMSE(predictionsRIDGE, testdata$THROUGHPUT_TIME),
  Rsquare = R2(predictionsRIDGE, testdata$THROUGHPUT_TIME) ,
  MAE = MAE(predictionsRIDGE, testdata$THROUGHPUT_TIME)
)


#Lasso regression med grid search
Lasso <- train(
  THROUGHPUT_TIME ~., data = trainingdata, method = "glmnet",
  trControl = Fitting,
  preProcess = c("range"),
  tuneGrid = expand.grid(alpha = 1, lambda = lambdagrid), # Alpha 1 = Lasso
  metric = "MAE")

## Visualisering av Grid for Lasso mtp Lambda seleksjon.
ggplot(Lasso) + geom_point(color = "blue") + #Viser til hyperparameterenes kombinasjon og MAE.
  labs(title= "Grid søk for Lasso regresjon",
       y="MAE (Repetert kryss-validering)", x = "Regulariseringsparameter Lambda")

# Model coefficients gitt optimale lambda
coef(Lasso$finalModel, Lasso$bestTune$lambda)


#Optimal Hyperparameter.
Lasso


# Gjør prediksjoner
predictionsLASSO <- predict.train(object=Lasso, newdata = testdata)

# Fitstats (Testsettscore)
data.frame(
  RMSE = RMSE(predictionsLASSO, testdata$THROUGHPUT_TIME),
  Rsquare = R2(predictionsLASSO, testdata$THROUGHPUT_TIME) ,
  MAE = MAE(predictionsLASSO, testdata$THROUGHPUT_TIME)
)
```

```r

###XGBOOST med tilfeldig søk
XGBlinear <- train(
THROUGHPUT_TIME ~., data = trainingdata, method = "xgbLinear",
trControl = trainControl(method = "repeatedcv",
                         number = 10,
                         repeats = 10,
                         search = "random"),
tuneLength = 10 ,
preProcess = c("range"),
metric = "MAE")


## Tilfeldig søk oversikten av hyperparameterset seleksjon.
trellis.par.set(caretTheme())
plot(XGBlinear, metric = "MAE")

#Optimal Hyperparameter.
XGBlinear


# Gjør prediksjoner
predictionsXGB <- predict.train(object=XGBlinear, newdata = testdata)


# # Fitstats (Testsettscore)
data.frame(
  RMSE = RMSE(predictionsXGB, testdata$THROUGHPUT_TIME),
  Rsquare = R2(predictionsXGB, testdata$THROUGHPUT_TIME) ,
  MAE = MAE(predictionsXGB, testdata$THROUGHPUT_TIME)
)




# Model evaluering ----


##Model evaluering av treningsfasen.
modelscompare <- resamples(list(Ridge = Ridge, Lasso = Lasso, lm = MLR, XGBoost = XGBlinear))

summary(modelscompare)

#Boxplots for evaluering.
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(modelscompare, scales=scales)


## Variabel viktighet for XGBOOST.
varimpxgb<-varImp((XGBlinear))
plot(varimpxgb, top=20, main="Variable viktighet i henhold til T-verdi for XGboost",
     xlab = "T-verdi", col="blue")

```

```
#XGBOOST prediksjoner vs observerte (Test data)


#Plotting Faktiske vs Predikerte verdier
options(repr.plot.width=8, repr.plot.height=4)
my_dataXGB = as.data.frame(cbind(predicted = predictionsXGB,
                                 observed = testdata$THROUGHPUT_TIME))

### Graf
ggplot(my_dataXGB,aes(predicted, observed)) + geom_point(color = "darkblue", alpha = 0.5) +
  geom_smooth(method=lm)+
  ggtitle("Extreme Gradient Boosting: Predikert vs Observert") +
  xlab("Predikert Throughput time") + ylab("Observert Throughput time") +
  theme(plot.title = element_text(color="black",size=16,hjust = 0.5),
        axis.text.y = element_text(size=12), axis.text.x = element_text(size=12,hjust=.5),
        axis.title.x = element_text(size=14), axis.title.y = element_text(size=14))


##Residular XGBOOST. (Testdata)
residualValues.XGB <- testdata$THROUGHPUT_TIME - predictionsXGB

# Predikerte verdier vs residualer
plot(predictionsXGB, residualValues.XGB, ylab = "Residualer",
     xlab = "Predikert Throughput time", col = "darkblue", main="Residualer for XGboost")
abline(h = 0, col = "darkred", lty = 1)




#MLR prediksjoner vs observerte
options(repr.plot.width=8, repr.plot.height=4)
my_dataMLR = as.data.frame(cbind(predicted = predictionsMLR,
                                 observed = testdata$THROUGHPUT_TIME))

### Graf
ggplot(my_dataMLR,aes(predicted, observed)) + geom_point(color = "darkblue", alpha = 0.5) +
  geom_smooth(method=lm) +ggtitle("MLR: Predikert vs Observert") +
  xlab("Predikert Throughput time") + ylab("Observert Throughput time") +
  theme(plot.title = element_text(color="black",size=16,hjust = 0.5),
        axis.text.y = element_text(size=12), axis.text.x = element_text(size=12,hjust=.5),
        axis.title.x = element_text(size=14), axis.title.y = element_text(size=14))


##Residular  (Testdata)
residualValues.MLR <- testdata$THROUGHPUT_TIME - predictionsMLR

# Predikerte verdier vs residualer
plot(predictionsMLR, residualValues.MLR, ylab = "Residualer",
     xlab = "Predikert Throughput time", col = "darkblue", main="Residualer for MLR")
abline(h = 0, col = "darkred", lty = 1)
```

```
691
692  #RIDGE prediksjoner vs observerte (Test data)
693  #Plotting Faktiske vs Predikerte verdier
694  options(repr.plot.width=8, repr.plot.height=4)
695  my_dataRIDGE = as.data.frame(cbind(predicted = predictionsRIDGE,
696                                     observed = testdata$THROUGHPUT_TIME))
697
698  ### Graf
699  ggplot(my_dataRIDGE,aes(predicted, observed)) + geom_point(color = "darkblue", alpha = 0.5) +
700    geom_smooth(method=lm) + ggtitle("Ridge Regresjon: Predikert vs Observert") +
701    xlab("Predikert Throughput time") + ylab("Observert Throughput time") +
702    scale_x_continuous(breaks = seq(from = 0, to = 7500, by = 500)) +
703    theme(plot.title = element_text(color="black",size=16,hjust = 0.5),
704          axis.text.y = element_text(size=12), axis.text.x = element_text(size=12,hjust=.5),
705          axis.title.x = element_text(size=14), axis.title.y = element_text(size=14))
706
707
708
709  ##Residular for Ridge. (Testdata)
710  residualValues.RIDGE <- testdata$THROUGHPUT_TIME - predictionsRIDGE
711
712  # Predikerte verdier versus residualer
713  plot(predictionsRIDGE, residualValues.RIDGE, ylab = "Residualer",
714       xlab = "Predikert Throughput time", col = "darkblue",
715       main="Residualer for Ridge Regresjon")
716  abline(h = 0, col = "darkred", lty = 1)
717
718
719
720
721  #Lasso prediksjoner vs observerte (Test data)
722  #Plotting Faktiske vs Predikerte verdier
723  options(repr.plot.width=8, repr.plot.height=4)
724  my_dataLASSO = as.data.frame(cbind(predicted = predictionsLASSO,
725                                     observed = testdata$THROUGHPUT_TIME))
726
727  ### Graf
728  ggplot(my_dataLASSO,aes(predicted, observed)) + geom_point(color = "darkblue", alpha = 0.5) +
729    geom_smooth(method=lm) +ggtitle("Lasso Regresjon: Predikert vs Observert") +
730    xlab("Predikert Throughput time") + ylab("Observert Throughput time") +
731    theme(plot.title = element_text(color="black",size=16,hjust = 0.5),
732          axis.text.y = element_text(size=12), axis.text.x = element_text(size=12,hjust=.5),
733          axis.title.x = element_text(size=14), axis.title.y = element_text(size=14))
734
735
736  ##Residular (Testdata)
737  residualValues.LASSO <- testdata$THROUGHPUT_TIME - predictionsLASSO
738
739  # Predikerte verdier vs residualer
740  plot(predictionsLASSO, residualValues.LASSO, ylab = "Residualer",
741       xlab = "Predikert Throughput time", col = "darkblue",
742       main="Residualer for Lasso Regresjon")
743  abline(h = 0, col = "darkred", lty = 1)
744  #Script Ferdig.
745
746
747
```

Figur III.4: ML Script.



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

Postboks 5003
NO-1432 Ås
Norway