Universitetet i Sørøst-Norge

Bacheloroppgave

SFHO3201 Karoline

Predefinert informasjon

Startdato:	05-05-2019 10:00	**Termin:**	2019 VÅR
Sluttdato:	23-05-2019 10:00	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Bacheloroppgave		
SIS-kode:	222 SFHO3201 1 BO 2019 VÅR		
Intern sensor:	Kiran Bylappa Raja		
Intern sensor:	Karoline Moholth Mcclenaghan		

Deltaker

Navn:	Emil Andre Elsetrønning
Kandidatnr.:	8
USN-id:	888614@usn.no

Gruppe

Gruppenavn:	6: Heimdall
Gruppenummer:	3
Andre medlemmer i gruppen:	Nils Erlend Heggem, Marte Kristoffersen, Tone Marie Løvseth Hognestad, Martin Farstad Evandt

Sensur av hovedoppgaver

Universitetet i Sørøst-Norge
Fakultet for teknologi og maritime fag

Prosjektnummer: **2019-06**
For studieåret: **2018/2019**
Emnekode: **SFHO3201-1 18H Bacheloroppgave**

Prosjektnavn
Adaptiv endepunktssikring
Adaptive endpoint protection

Utført i samarbeid med: Telenor Emergency Response Team (TCERT), Telenor Norge

Ekstern veileder: John Thomas Wiborg

Sammendrag:
Informasjonssikkerhet er et fagfelt i stadig endring, og det blir derfor viktigere enn noen gang for analytikere og sikkerhetsingeniører å detektere skadevare raskest mulig for å minske omfanget. Det er her kunstig intelligens kommer inn i bildet, og oppgaven til å gruppen er å se på muligheten for å bruke maskinlæring til dette. Dette gjøres via verktøy laget av gruppen, samt analyse av resultater som er generert.

Stikkord:
- Maskinlæring (ML)
- Endpoint Detection and Response (EDR)
- Cybersikkerhet

Tilgjengelig: JA

Prosjekt deltagere og karakter:

Navn	Karakter
Emil A. Elsetrønning	
Marte Kristoffersen	
Martin Farstad Evandt	
Nils Erlend Heggem	
Tone Marie L. Hognestad	

Dato: 14. juni 2019

_____	_____	_____
Kiran Raja	Karoline Moholth	Rune Hammersland
Intern Veileder	Intern Sensor	Ekstern Sensor

Universitetet i Sørøst-Norge

telenor

Adaptiv Endepunktssikring

Maskinlæring for deteksjon av skadevare og abnormal adferd innen Endpoint Detection & Response systemer

BACHELOROPPGAVE, UNIVERSITETET I SØR-ØST NORGE

E.A. ELSETRØNNING, M.F. EVANDT, N.E. HEGGEM,
T.M. HOGNESTAD, M. KRISTOFFERSEN

Abstract

Informasjonssikkerhet er et fagfelt preget av voldsomme endringer over korte tidsperioder. Med et trusselbilde i stadig endring er det viktigere enn noen gang for sikkerhetsingeniører å være klare over eksisterende trusler for å kunne motvirke datalekkasjer og sabotasje. Trusselaktører i alt fra vinningskriminelle grupper til statlige organisasjoner angriper privatpersoner, bedrifter og myndighetsorganer på tvers av landegrenser med liten til ingen konsekvens.

Løsepengevirus, valgfusk og IoT-baserte bot-nettverk som slår ut internettilgangen til hele regioner, tyder på en skarp og skremmende økning i avanserte angrep over kun de siste par årene. Å motvirke slike angrep beror på tidkrevende analyse av infeksjonsvektorer og angrepsteknikker, ofte i ettermælet av et gjennomført angrep. Kunstig intelligens kan bidra til å effektivisere denne identifiseringsprosessen ved hjelp av moderne teknikker.

Denne rapporten og arbeidet bak den illustrerer fordeler så vel som fallgruver og forbehold som må tas i henhold til datasett, dataprosessering, feature-valg og hyperparameteroptimalisering for å kunne benytte seg av denne teknologien i kontekst av data fra Endpoint Detection and Response systemer.

Ved å trene og teste en rekke ulike klassifiseringsmodeller på et datasett med flere simulerte angrep, viser dette prosjektet at maskinlæring kan bli et bærekraftig supplement i nestegenerasjons cybersikkerhets-arsenal.

`https://web01.usn.no/grupper/web-gr6-2019/`

Innhold

Figurer

Tabeller

Akronymer

AIDS Anomaly-based Intrusion Detection System.
APT Advanced Persistent Threat.
AV Antivirus.

CMS Cybermanufacturing systems.

EDR Endpoint Detection and Response.

IDS Intrusion Detection System.
IoT Internet of Things.

ML Machine Learning.

PST Politiets sikkerhetstjeneste.

TCERT Telenor Computer Emergency Response Team.
TSOC Telenor Security Operations Center.

Ordliste

BIAS En modells favorisering av visse typer klassifiseringer, ofte som følge av en feil i pipeline eller datasettet..

CLUSTERING Algoritmisk gruppering av data..

DECISION BOUNDARY Måten en klassifiseringsmodell skiller mellom klasser ved å partisjonere vektorrommet dataen ligger i. Alle datapunkter innenfor en decision boundary blir satt til én av to (eller fler) klasser..

EKSEMPEL Består av flere features, dette er totalen som sendes inn i modellen..

FEATURES Verdier som går inn i modellen. Dette kan være tall, ord eller mer komplekse objekter. Disse må prosesseres avhengig av hvilken modell de skal inn i. Dette er altså kolonnene i datasettet..

GRENSE En skalar som bestemmer størrelsen på området hvor eksempler blir kategorisert som usikre av feedback klassifikatorer..

HYPERPARAMETER Et parameter i en modell som må være satt før en modell begynner å trene..

HYPERPLAN Et høydimensjonalt plan som separerer eksempler..

KOVARIANS Et mål på den linære avhengigheten mellom to variabler..

LABEL Verdien det er ønskelig for modellen å bestemme, eller output. I bilder kan dette typisk være en beskrivelse av hva som befinner seg i bildet, f.eks hund, katt, fugl. I dette prosjektet er typisk label god og dårlig data..

LABELED EKSEMPEL er et eksempel med fastsatt label. Disse eksemplene brukes til å trene modellen, eller til å verifisere en gjetning fra modellen..

LOSS En funksjon som gir en absoluttverdi på differansen mellom det modellen forutser og de faktiske verdiene til et eksempel.. , 44

MASKINLÆRING Trening av modeller til å analysere data ut i fra data modellen tidligere er vist.

MODELL En bestemmelse av relasjonen mellom features og lables. Det er denne som trenes opp i ML-løpet. Det finnes typisk to kategorier av modeller; regresjonsmodeller og klassifiseringsmodeller. En regresjonsmodell prøver å beregne kontinuerlige verdier, mens en klassifiseringsmodell ønsker å bestemme i hvilken klasse eller kategori et eksempel tilhører..

PIPELINE Hvordan informasjonflyten inn i og ut av modellen går. Hvordan informasjon mates inn, hvordan den blir prosessert og presentert..

TERSKEL En skalar verdi som brukes for å bestemme hvilken klasse eksempelet tilhører ut ifra en sannsynlighet satt av modellen..

TRENING Opplæring av en modell i relasjonene mellom features og labels. Dette gjøres typisk med å vise modellen eksempler med labels..

1 Introduksjon

1.1 Bakgrunn

Gruppen består av fem dataingeniør-studenter, der alle har hatt grunnleggende datafag, samt valgfaget "Introduksjon til datasikkerhet". Ingen av gruppemedlemmene har tidligere erfaring med maskinlæring eller EDR-systemer.

1.2 Gruppemedlemmer

Emil A. Elsetrønning
Product Owner

- Sørger for at prosjektet går i riktig retning, og at de rette brukerhistoriene blir jobbet med.
- Hovedkommunikator med ekstern- og internveileder.

Marte Kristoffersen
Testansvarlig

- Ansvarlig for at alt testes, og sørger for at prosedyre for testing blir fulgt.

Martin Farstad Evandt
Dokumentasjonsansvarlig

- Ansvarlig for generering og formatering av teknisk dokumentasjon via Sphinx.
- Kontrollerer helheten i dokumentasjonen, og hindrer eventuelle mangler i dokumentasjonen.

Nils Erlend Heggem
Kodeansvarlig

- Sørger for at kodestandard følges, og at strukturen i alt som blir levert er korrekt.

Tone Marie L. Hognestad
Scrum Master

- Tar ansvar for at arbeidsflyten følges, spesielt komponentene fra Scrum-modellen.

1.3 Dokumentstruktur

Kapittel 2 - Problemdomene

Beskriver domenet oppgaven er gjennomført i, hvem oppdragsgiver er og hvilke behov de har, og hvordan disse behovene former konteksten i oppgaven.

Kapittel 3 - Metodologi

Hvordan prosessmetodologien, testprosedyrene og risikohåndteringen er lagt opp, og hvorfor. Videre et retrospektiv på hvordan prosessen ble gjennomført, hvilke utfordringer som ble støtt på og hvilke endringer som ble gjort. Øvrige detaljer om kommunikasjon med arbeidsgiver, samarbeid med intern veileder, stilguider, generering av teknisk dokumentasjon m.m.

Kapittel 4 - Teori

Teorikapittelet tar for seg teorien bak de forskjellige akademiske feltene det jobbes innenfor; informasjonssikkerhet, endepunktssystemer, maskinlæring - og hvorfor de har en nytte for prosjektet. Det diskuteres også hvilke eksterne rammeverk som skal benyttes.

Kapittel 5 - Krav

Dokumenterer hvilke krav som ble satt for analyse av maskinlæringsmodeller så vel som programpakken som ble skrevet for å oppnå data om disse modellene, bakgrunnen for disse kravene.

Kapittel 6 - Modellering

Viser hvordan programpakken ble modellert som et følge av kravene ved hjelp av UML m.m.

Kapittel 7 - Implementasjon

Implementasjonen av programpakken i kontekst av modellene og kravene som ble laget i de to foregående kapitler.

Kapittel 8 - Analyse

Analyse av arbeidet som ble gjort i kontekst av oppgaveteksten redegjort i kapittel 2.

Kapittel 9 - Konklusjon

En objektiv analyse av prosjektarbeidet, relaterte verker, gruppens bidrag til feltet, utfordringer og ev. resterende arbeide.

Kapittel 10 - Evaluering

En subjektiv analyse av prosjektarbeidet fra gruppen selv, hva de har lært, og hvordan de selv synes prosjektet har gått.

2 Problemdomene

2.1 Oppdragsgiver: Telenor Norge

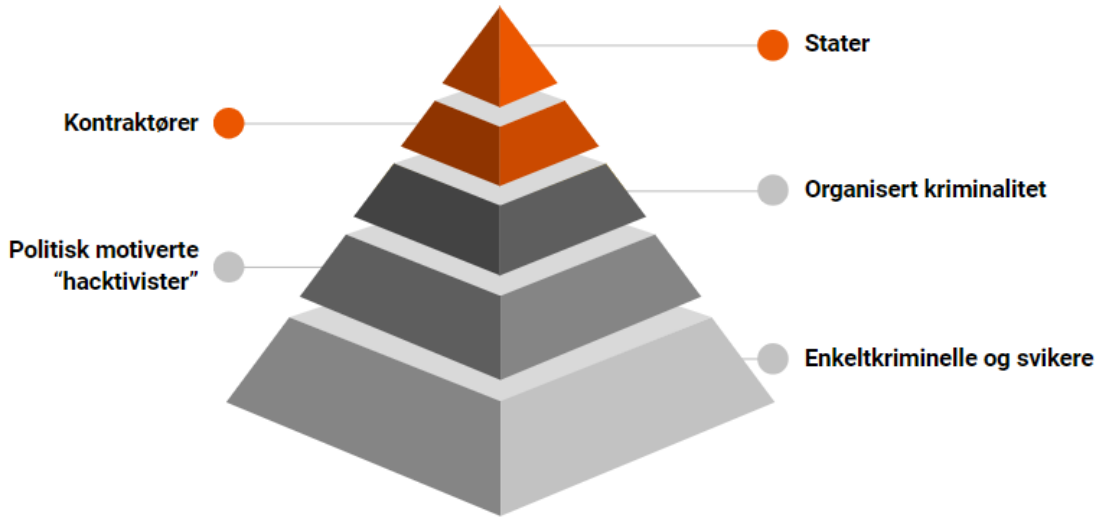
Som en ledende bedrift innen mobil, bredbånd og tv, er Telenorgruppen ansvarlig for over 174 millioner kunder verden over. Med over 20 000 ansatte over hele verden og en inntekt på over 110 milliarder i 2018 alene, er Telenorgruppen et telekommunikasjonsfirma i verdenstoppen. To verdensdeler er avhengige av tjenestene Telenorgruppen leverer; Europa og Asia. Nettet til Telenor Norge er det raskeste nettet i verden, godt over dobbelt så raskt som gjennomsnittet på verdensbasis. Telenor Norge eier store deler av kritisk, norsk informasjonsinfrastruktur, dvs. rundt 80% av all datatrafikk i Norge går over infrastruktur eid av Telenor Norge. Denne infrastrukturen er pålagt å være til disposisjon i krisesituasjoner og er en del av norsk beredskap. Det er derfor påkrevd et høyt nivå av sikkerhet [1]. Telenor Norge har inngått samarbeid med både det norske Cyberforsvaret og Nasjonal Sikkerhetsmyndighet, for å sørge for at Telenor lever opp til sitt samfunnsansvar. Internt er det to sikkerhetsavdelinger som forsvarer Telenor Norges infrastruktur; Telenor Security Operations Center (TSOC) og Telenor Computer Emergency Response Team (TCERT).

2.2 Scenariet

Norge er et land som ligger i fronten på telekommunikasjon og teknologi. Med stadig flere tjenester på nett genereres det store mengder informasjon, informasjon trusselaktører forsøker å få tilgang til ved å utnytte svakheter i disse tjenestene.På nasjon- og organisasjonsnivå er det såkalte Advanced Persistent Threats (APT) som er hovedaktørene. Disse aktørene kommer fra toppen av trusselpyramiden 2.1, hvor skaden og kompleksiteten av angrepene er størst. Typiske APT-er er etterretningstjenester fra nasjoner som Kina, Russland, Nord-Korea og Iran, eller betalte kontraktører.

I Politiets sikkerhetstjeneste (PST) sin årlige sikkerhetsvurdering er det for 2019 lagt stor vekt på statlig etterretningsvirksomhet. Statlige aktører vil, ved hjelp av urettmessig tilgang på en virksomhets nettverk, forsøke å skaffe sensitiv informasjon eller påvirke beslutninger [2]. Dette utgjør en stor trussel mot Norges nasjonale sikkerhet. Det er i særs statlige organisasjoner og bedrifter som jobber

med kritisk infrastruktur som er attraktive mål for slike angrep. Det er opp mot disse aktørene TCERT jobber.



Figur 2.1: Trusselpyramiden gir en oversikt over hvilke aktører som utgjør en trussel innenfor datasikkerhet. De mest alvorlige er listet i toppen.

2.3 Oppgaven

TCERT har gitt bachelorgruppen i oppgave å undersøke hvordan kunstig intelligens og maskinlæring kan brukes i sikkerhetssammenheng. Ved å undersøke effektiviteten av maskinlæring i Endpoint Detection and Response [3] systemer, eller EDR, håper TCERT å kunne redusere arbeidsmengden for sine analytikere, samt redusere sin responstid på eventuelle angrep. Bachelorgruppen opererer med et simulert datasett generert fra en øvelse med simulerte angrep. Håpet er at funn på dette datasettet skal kunne generaliseres, slik at det er mulig å nytte seg av samme fremgangsmåte i en reel situasjon. Telenor Norge ønsker et konkret svar på om maskinlæring innen EDR burde ses nærmere på. Oppgaveteksen oppsummert lyder slik:

> "Se på muligheten for å detektere skadevare og trusler ved hjelp av maskinlæring"

2.4 Relaterte arbeider

Bruk av kunstig intelligens i EDR-systemer er av noen spådd å være en av trendene i informasjonssikkerhet-sektoren i årene framover [4], og automasjon er ansett av

bl.a. IBM å være det neste store steget innen sikkerhet [5].

2.4.1 Kommersielle

Både EDR og kunstig intelligens er relativt nye felt, og det finnes få lignende prosjekter på hovedoppgavenivå. Derimot finnes det kommersielle løsninger som benytter seg av kunstig intelligens til sikring av endepunkter.

2.4.1.1 Cylance

Cylance er et amerikansk firma som lenge lå i fronten i å bruke kunstig intelligens i sine sikkerhetsløsninger. Det var det første firmaet som lanserte en signatur-løs antivirusløsning med sertifisering fra AV-TEST med produktet Cylance Protect i 2016 [6, 7]. Senere utvidet de denne teknologien til EDR-løsningen Cylance Optics [6]. Dette spranget i teknologi har ført til at Cylance per dags dato har en markedsandel på 4.19% når det kommer til endepunktsløsninger, større enn både McAfee og Microsofts løsninger, bare syv år etter at de ble grunnlagt i 2012 [8].Cylance har benyttet forspranget sitt til å dele kunnskap med resten av sikkerhetssektoren gjennom white papers, foredrag, og en gratis e-bok om bruk av kunstig intelligens i informasjonssikkerhet, som har vært til hjelp i denne oppgaven [9].

2.4.1.2 Deep Instinct

Deep Instinct ble grunnlagt i 2014, og benytter seg av deep learning - en underkategori av maskinlæring - til informasjonsikkerhetsformål deriblant endepunktsløsninger [10]. I uavhengige testresultater fra firmaet SE Labs oppnådde Deep Instinct en 100% prevensjonsrate og ingen falske positiver [11].Deep instinct ble i 2017 omtalt av NVIDIA som årets "most disruptive AI startup" [12], og ble samme året omtalt av World Economic Forum som en "technology pioneer" [13].

2.4.2 Akademiske

Maskinlæring som et akademisk felt er i vekst, og det publiseres ny forskning på det fortløpende. Mye av denne forskningen omfatter sikkerhetssektoren, men det er få prosjekter å finne på bruk av maskinlæring i EDR, særlig på hovedoppgavenivå. Derimot benyttes maskinlæring i flere andre områder av informasjonssikkerhet.

2.4.2.1 Advancing Neuro-Fuzzy Algorithm for Automated Classification in Large-scale Forensic and Cybercrime Investigation

Et prosjekt hvor maskinlæring blir brukt i deteksjon av skadevare, er i prosjektet [14]. Prosjektet går ut på å bruke neuro-fuzzy algoritme, som er en kombinasjon av fuzzy logic og nevrale nettverk, til å automere prosessen av å samle inn bevismateriale for rettsaker innen cyberkriminalitet, hvor mengden data er for stor for vanlig håndtering.

Normalt gir neuro-fuzzy algoritmer en stor mengde fuzzy-regler, noe som gjør det vanskelig å bruke for mennesker, og spesielt vanskelig å vise i rettsaker. I dette prosjektet forsøkes det å redusere mengden fuzzy-regler ned til en mer håndterbar

mengde. Forfatteren forteller at de har klart å nesten bevare treffsikkerheten, men en liten reduksjon fra 98.790% til 98.787%, mens antall regler ble redusert ned til 39 fra 10231.

Da dette prosjektet bruker maskinlæring til å gjøre analyse av skadevare, er det en indikasjon på at dette er mulig, selv om det ikke relaterer seg veldig til denne hovedoppgaven.

2.4.2.2 Detecting Cyber-physical Attacks in CyberManufacturing Systems With Machine Learning Methods

Dette prosjektet [15] jobber opp mot cybermanufacturing systems (CMS), også kjent som industrial internet of things (IIOT) [16]. CMS er som EDR et samlebegrep for en rekke nestegenerasjonsløsninger, i dette tilfellet industrielle cyber-fysiske løsninger som 3D-printing, laserkutting, etc.

Angreper på cyber-fysiske industrisystemer kan være katastrofale da de har fysiske følger. Ta Stuxnet-ormen som et sted mellom 2005 og 2010 forårsaket store materielle skader på det Iranske atomprogrammets turbiner [17].

Forfatterne av denne forskningsartikkelen ser på bruken av maskinlæring i denne konteksten. De monitorerer verts-maskiner og nettverkstilkoblinger i mellom dem etter spor på ondsinnet adferd, og rapporterer å ha en 96.1% nøyaktighet på klassifisering av angrep i 3D-printere, og 91.1% nøyaktighet på angrep mot CNC fresemaskiner.

2.5 Prosjektets bidrag til fagfeltet

Bruk av kunstig intelligens i antivirus-, IDS- og EDR-systemer er nestegenerasjons digitale sikkerhetstiltak. Dette prosjektet ser på sistnevnte system fra et ingeniørperspektiv i form av en hovedoppgave. Oppgaven vil bidra til å underbygge gjennomførbarheten og effektiviteten av kunstig intelligens i EDR ved å kartlegge hvor kompleks en implementasjon vil være, hvordan data fra et EDR-system kan behandles av en maskinlæringsmodell, hvilke verktøy som må utvikles, hvilke forbehold det er for å kunne gjennomføre det oppgaven setter ut for å gjøre, og til sist hvordan samspillet og kunnskapsnivået må være mellom maskinlæring og informasjonssikkerhet som respektive felter av ekspertise.

3 Metodologi

3.1 Oppgavens påvirkning på metodikken

Dette er en oppgave hvor omfang og vinkling kan endres fortløpende ut ifra funn. Prosjektet baserer seg på et tema som er nytt for både oppdragsgiver og prosjektgruppen. Det er derfor usikkert hvilken verdi som er ønsket av bedriften og hvilken verdi det er mulig å skape. Dette fører til at gruppen må jobbe på en tilpasningsdyktig måte, slik at fokuset i prosjektet kan snevres inn eller vides ut etter behov.

3.2 Agile, Scrum & Kanban

3.2.1 Smidig prosess

Grunnet oppgavens natur og behov for fleksibilitet vil metodikken bygges rundt agile, eller smidige, prosessutviklingsmodeller. Denne måten å jobbe på er basert på raske iterasjoner som tilrettelegger for endringer på kort varsel. Metodikken er utformet med basis i Scrum [18] og inneholder elementer av Kanban [19] tilpasset oppgavens behov.

3.2.2 Krav & brukerhistorier

Scrum har en vinkling til krav som kalles for en brukerhistorie. Formålet med en brukerhistorie er å se på et krav fra et brukerperspektiv for å sørge for at det som jobbes med er relevant for sluttproduktet. En vanlig måte å definere krav på i en brukerhistorie er ved skrive kravene ut på følgende måte:

> **SOM EN** stakeholder
> **VIL VI** utføre en handling
> **SLIK AT** et ønsket utfall skjer

Hensikten med denne øvelsen er å gi en kort og konkret beskrivelse av kravet; hvem stakeholderen er, hva kravet er, og hvilken verdi kravet har for stakeholderen. Brukerhistorier kan deles inn i deloppgaver, hvor hver deloppgave tar for seg et element av en brukerhistorie. Deloppgaver lar en brukerhistorie stykkes opp i

mindre oppgaver, der totalen av alle deloppgaver i en brukerhistorie utgjør hele funksjonaliteten til nevnte brukerhistorie.

3.2.3 Morgenmøter

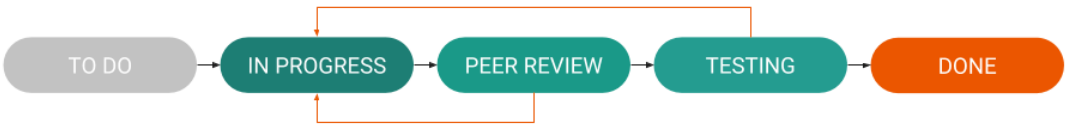
Hver morgen innkaller Scrum Master til morgenmøte, også kalt stand-up. Her står alle medlemmene i en ring og legger frem sin status. Statusen skal inneholde disse punktene:

> **Hva jeg gjorde i går.**
> **Hva jeg skal gjøre i dag.**
> **Utfordringer/problemer som har oppstått.**

Dette gjør gruppen for å få en grov oversikt over sprinten og muligheten for innspill på eventuelle problemer som har oppstått.

3.2.4 Sprinter

En sprint er et tidsintervall der det skal gjennomføres et gitt antall brukerhistorier. I starten av hver sprint bestemmes det hvor mange brukerhistorier som inngår i sprinten, og antall poeng hver brukerhistorie skal ha. Disse poengene forteller noe om estimert tidsbruk, der lav poengsum tilsier kort tidsbruk og høy poengsum lang tidsbruk. Poengsummen til en brukerhistorie bestemmes i felleskap av alle prosjektdeltakerne, der hver av dem foreslår en poengsum og medianen velges.



Figur 3.1: Arbeidsflyt

3.2.5 Retrospektiv

I enden av hver sprint, som varer i én uke, skal det gjennomføres et retrospektivt møte. I en retrospektiv diskuterer gruppen hva som har fungert bra og ikke fullt så bra i løpet av sprinten. Særskilt fokus legges på strukturen og størrelsen av brukerhistorier, antall brukerhistorier i sprintens backlog og prioriteringer. Formålet med en retrospektiv er å iterativt forbedre hvordan prosjektet gjennomføres, og å få bukt med problemer før de vokser seg for store.

3.2.6 Arbeidsflyt

Arbeidsflyten er hvordan en brukerhistorie går fra krav til verdi. Tradisjonelle Scrum-arbeidsflyter illustrerer bare hvilke brukerhistorier og oppgaver som skal gjennomføres i den aktuelle sprinten, ved bruk av tre kolonner: ”Ready” for alle oppgavene som skal gjennomføres i denne sprinten, ”In progress” for hvilke oppgaver det jobbes med i nåtid, og ”Done” for oppgavene som er ferdigstilte.I dette prosjektet ble det ansett som hensiktsmessig å implementere to ekstra kolonner som en del av arbeidsflyten: ”Peer review” og ”Test”, se figur 3.1. I denne fasen gjennomføres testing av programvare. .

3.3 Testing

For å sørge for at det i hver iterasjon skaper verdi fra en brukerhistorie må alt arbeidet etterprøves i en testprosess. Testing gjennomføres i to deler, white-box testing og black-box testing.

3.3.1 White-box & Peer review

White-box-testing, også kalt Glass-box-testing, er å se på programvaren med kunnskap om alle underliggende algoritmer og datastrukturer. Med andre ord vil dette si en nøye gjennomgang av selve kildekoden. Disse testene gjøres av et eller flere gruppemedlemmer andre enn den eller de som har skrevet koden, og bygger på følgende prinsipper:

Code coverage er hvor stor del av koden som har blitt testet. For at et program skal kunne anses å ha gjennomført god Peer review må det ha full ”code coverage”. Dette er en retningslinje for den som tester, og ikke en test i seg selv.

Condition testing er å analysere alle mulige utfall av logiske operatorer og setninger i et program, og sørge for at all kode som kjøres som et resultat av en logisk operasjon virkelig skal kjøres i den konteksten.

Data flow testing er å se på hvordan verdiene til variabler og data oppfører seg. Hensikten er å sørge for at deklarerte variabler brukes, at alle variabler er deklarert i scopet de brukes i, at en variabel ikke er definert flere ganger i samme scope, og at en variabel holdes innenfor tiltenkte parametre.

CONTROL FLOW TESTING er et makroperspektiv av koden, der man ser på et program på samme måte som man ville lest et flytdiagram og sørger for at sekvenser utføres i riktig rekkefølge. Hensikten er å sørge for at flyten i et program er riktig; at en while-løkke alltid når en base-case, at man ikke får tilgang på data uten å være autentisert, og lignende.

3.3.2 Black-box & endelig testing

Black-box skiller seg fra White-box da man ikke skal se på den underliggende logikken, men fra et system- og sluttbrukerperspektiv. I denne prosessen er det tre nøkkelord: formål, integrasjon og brukervennlighet. Black-box-testing gjøres av utvikleren med ansvar for brukerhistorien, samt et eller flere gruppemedlemmer.

FORMÅL er å analysere at koden som har blitt skrevet har oppfylt brukerhistorien eller bugrapporten den er et resultat av. At den ikke er ufullstendig eller overflødig.

INTEGRASJON er å se på om programmet som har blitt skrevet, endringene som har blitt gjort eller tilleggene til programmet interfacer med hele systemet. Om programmet har blitt utviklet på sin egen git-branch skal det i dette stadiet integreres med develop-branchen, først etter gjennomført "Black-box"-testing skal den ut i master.

BRUKERVENNLIGHET er å både se for seg grafiske interfacer fra et sluttbrukerperspektiv og å se på koden som en utvikler som skal jobbe direkte med den. I sistnevnte situasjon vil dette si å se at koden er tilstrekkelig dokumentert i den tekniske dokumentasjonen.

3.4 Testverifisering

For å opprettholde en sporbarhet i testprosedyrene skal alle tester dokumenteres i et felles regneark via Google Sheets. I dette regnearket er det forhåndsbestemte felter for informasjon om brukerhistorien slik som dets unike ID, forfatter og hvem som har testet det. Videre felter er sjekklister for alle de overnevnte test-typene fordelt på Black- og White-box. Hver gjennomført test som godkjennes skal hukes av i sitt respektive felt. Om en test feiler må dette rettes opp snarest, og i ytterste tilfelle må brukerhistorien tilbake til In progress i arbeidsflyten. Oppsummeringsvis skal en test inneholde følgende: condition-, data flow- og control flow-testing av kildekode, samt godkjenning av formål, integrasjon og brukervennlighet fra perspektivet til en sluttbruker og resten av systemet.

	Peer-review			Test		
Issue ID	Condition	Data flow	Control flow	Formål	Integrasjon	Brukervennlighet
Test-001	Godkjent	Godkjent	Godkjent	Godkjent	Godkjent	Godkjent
BG-78	Godkjent	Godkjent	Godkjent			

Figur 3.2: Skjermdump av testverifiseringsskjema, ekskludert forfatter og tester(e).

3.5 Risiko

Det er forskjellige meninger om hvordan risk burde håndteres innen smidige prosesser, men det som trekkes frem er en personlig tilnærming, der ansvarsroller holder styr på forskjellige risker [20]. Dette er en lettvektig tilnærming til riskanalyse, hvor mindre risker (som i en vanlig risikomatrise ville fått lav/middels rangering) ikke blir vurdert som reelle risker, da mindre risker blir eliminert hovedsakelig av iterasjoner. Derimot vil større risker bli håndtert på en skreddersydd måte.

3.5.1 Risikotyper

I sammenheng med Scrum, nevnes det ofte tre forskjellige risikonivåer i et prosjekt, økonomiske, tekniske eller forretningsrelaterte risker [20]. For dette bachelorprosjektet vil ikke økonomiske risker være aktuelle å se på, da det ikke inneholder økonomiske faktorer som utstyr, personell, lokaler osv. Derimot er forretningsrelaterte og tekniske risker aktuelle.

Forretnigsrelaterne risker Handler om hva kunden egentlig ønsker og risikoen for at produktet man lager ikke oppfyller kundens behov. Det er derfor viktig at Product Owner er et godt kommunikasjonsledd mellom oppdragsgiver og resten av gruppen.

Tekniske risker handler om produktet i seg selv. Kvalitetssikring, testing, validering og dokumentering er viktige faktorer for å unngå tekniske risker.

3.5.2 Håndtering av større risker

Siden mindre risker plukkes opp av iterasjonene, vil dokumentering og håndtering fokusere utelukkende på større risikoelementer i prosjektet. I hvert retrospektiv vil gruppen analysere uken som har vært og detektere eventuelle høynivårisker som kan komme som et resultat av ukens nye utfordringer. Når en høyrisiko oppstår legges denne i risikodokumentet. Risikodokumentet inneholder en tabell der man identifiserer, analyserer og fastslår tiltak til riskene.

3.6 Implementasjon av prosessen

I subseksjon 3.2 Agile, Scrum & Kanban blir prosessen innledet slik den var tiltenkt å være. I denne seksjonen går vi inn på gjennomføring, hvordan de ulike elementene i prosessen har fungert for gruppen, og hvilke endringer vi har gjort underveis i prosjektet.

3.6.1 Scrum

Gruppen har valgt å holde på én ukes sprinter, da det ble klart at hyppige retrospektiver ga mulighet for å bytte fokus relativt raskt der det ble ansett som nødvendig. Retrospektivene har også gitt mulighet for at alle i gruppen fritt har kunnet komme med innspill på et passende tidspunkt. Gruppens retrospektiver er vedlagt, se vedlegg B. Gjennom prosessen har gruppen også hatt god nytte av morgenmøter, da dette har gikk et raskt overblikk over hva resten av gruppen arbeider med. Gjennom morgenmøtene har det også kommet frem hvilke brukerhistorier som burde prioriteres, slik at medlemmene av gruppen har kunnet danne seg et bilde av hvilke oppgaver som kan gjøres dersom det har vært tvil om det.

3.6.2 Testing

Når en koderelatert brukerhistorie er ansett som ”ferdig” og klar for testing, har brukerhistorien blitt flyttet til kolonnen for testing. Her har hele gruppen tatt ansvar og sørget for å gå over og teste kode en annen har skrevet. Testoversikten har gjort det oversiktlig hvem som har skrevet kode, hvem som har testet, og hvilke krav til kode koden innfrir. Testoversikten har også vært behjelpelig i form av at testingen har blitt formelt gjennomført. Ved hjelp av konstruktive tilbakemeldinger har brukerhistorier fått status ”Reopened”, gått tilbake i In Progress, hvor tilbakemeldingene har kunnet blitt tatt hånd om på en smidig og effektiv måte. Testverifiseringsdokumentet er vedlagt ved siden av rapporten.

3.6.3 Risiko

I gjennom prosjektet har flere ufordelaktige muligheter dukket opp. De aller fleste av disse har kunnet blitt tatt hånd om i den iterative prosessen før det har blitt et behov for å kategorisere disse som større risker, likevel har flere overordnede risker blitt etablert.

Identifikator	Type	Analyse	Tiltak
Arbeidsgivers forventinger	Forretningsrelatert	Risiko for at vi leverer et produkt uten verdi for arbeidsgiver.	God kommunikasjon mellom Product Owner og Telenor Norge, samt nøye oppfølging og demonstrasjoner i samhold med Telenor Norge.
Manglende erfaring	Teknisk	Risiko for at mangel på erfaring kan føre til feilvurderinger av maskinlæringsalgoritmer og hvordan de brukes riktig.	Nøye research, samt rådføre med internveileder og verifisere resultater nøye.
Ubalansert datasett	Teknisk	Risiko for at ubalansert datasett kan føre til høy bias i modellene og dårlige resultater.	God research på mulige (ev. alternative) løsninger, samtidig ha i bakhodet at dette er en reel utfordring.

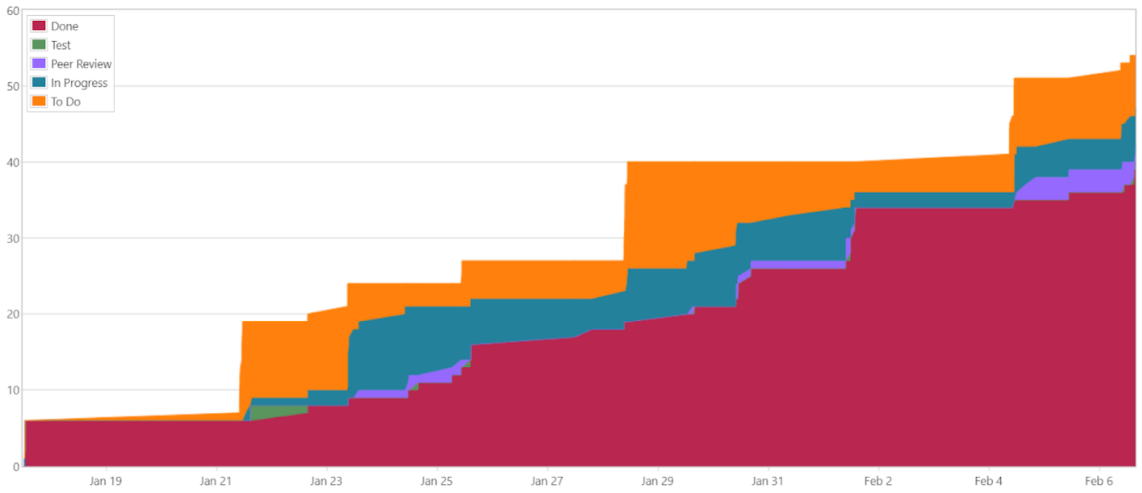
Tabell 3.1: Tabell over overordnede risker og deres tiltak

3.6.4 Endringer i prosessen

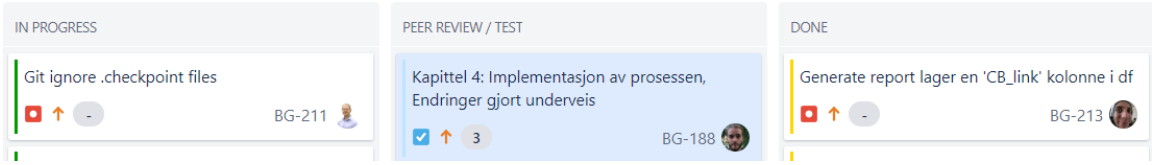
3.6.4.1 Test og Peer review

Når en brukerhistorie gikk over fra In progress til Peer review, der white-box testing skulle gjennomføres, ble det i samtlige tilfeller også gjennomført en black-box testing av brukerhistorien når man først var i gang. Black-box testing ble gjennomført i en egen kolonne, Test. Dette kom tydelig fram av det kumulative flytdiagrammet, der det så vidt var mulig å se når en brukerhistorie var innom Test-kolonnen, siden all testing gjerne ble gjennomført i én omgang, mens den lå i Peer review, og gikk derfor bare en rask tur innom Test før den havnet i Done.

Den opprinnelige hensikten med å holde dem adskilt, var at det var to forskjellige former for testing som ble gjennomført i hver kolonne. Dette var også en vanlig måte å gjennomføre testing på i en rekke Kanban-eksempler. Men siden de to testformene var såpass nært beslektet, og de ble gjennomført samtidig, ble det besluttet å slå de to kolonnene sammen til én, simpelthen kalt Peer review/Test.



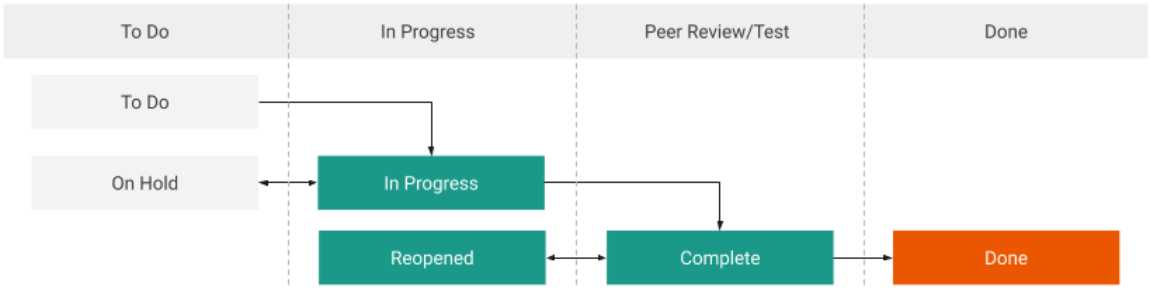
Figur 3.3: Kumulativt flytdiagram for perioden 18. januar - 7. februar. Peer review i lysest lilla, test i grønt.



Figur 3.4: Skjermdump av endringene i arbeidsflyten, den nye kolonnen er i midten.

3.6.4.2 Workflow - arbeidsflyt

Fra retrospektiv kom det frem at det var et behov for en status i arbeidsflyten som indikerte at en påbegynt brukerhistorie var satt på vent, slik at dette kom godt frem på tavla. Behovet for dette oppsto når vi i løpet av en sprint gjorde oss opp en mening om at fokuset for sprinten var feil, og vi ønsket og endre, samtidig som vi hadde oversikt over påbegynt arbeid på en oversiktlig måte.



Figur 3.5: En oversikt over arbeidsflyten inkludert ”On hold”.

3.6.4.3 Feature request issue

I starten av prosjektet opererte gruppen med tre issue-typer i backlogen: brukerhistorie, administrative oppgaver og bugs. Men over prosjektets gang oppstod behovet for å utvide eller endre allerede implementerte programmer og funksjoner. Det ble derfor opprettet en egen issue-type: Feature request.

Figur 3.6: Skjermdump av feature request issues

3.6.4.4 Risikohåndtering

I begynnelsen av prosjektet ble risikohåndtering gjort ved hjelp av en typisk riskmatrise, der risker ble vurdert etter hvor stort utfall de hadde og hva sannsynligheten for at de skulle skje var. Etter at prosjektet hadde pågått i et par måneders tid, innså gruppen at det var lite hensiktsmessig å gjøre det på denne måten grunnet bruken av en iterativ prosjektmodell. Særlig innså gruppen at de mindre riskene lett lot seg håndtere ved å opprette en brukerhistorie som tok stilling til eventuelle feil gjort under foregående sprinter. Det ble derfor bestemt at det kun skal sees på og taes stilling til større og alvorlige risker, som vist i seksjon 3.6.3.

3.6.4.5 Rollebytter

På retrospektiv etter andre presentasjon kom det frem at rollene innad i gruppen var lite strukturerte, og at de ble litt løse i form av hvem som tok ansvar for hva. Martin hadde gode tanker om rapportstruktur og hvordan denne kunne struktureres på en god måte, samtidig som Tone Marie gjerne tok initiativ til morgenmøter og hadde god oversikt over ulike funksjoner i Jira. Gruppen så det dermed som smidig at Tone Marie ble ny Scrum Master og Martin tok rollen som dokumentasjonsansvarlig.

Fra starten av var det også planer om å lage et interface arbeidsgiver kunne bruke for å gjøre nytte av programpakken, og det ble derfor bestemt at det var behov for en UX-ansvarlig, en rolle Marte tok på seg. Men samtidig som det ble gjort rollebytte for Martin og Tone Marie kom gruppen frem til at en egen UX-ansvarlig ikke var nødvendig da fokuset til gruppen endret seg. Marte tok derfor over den tidligere rollen til Tone Marie som testansvarlig. Disse endringene har fungert godt for gruppen, og ble et naturlig rollebytte etterhvert som prosjektet tok form.

3.7 Kommunikasjon med oppdragsgiver

Gruppen og oppdragsgiver har hatt daglig kontakt, først gjennom en felles Slack-kanal, og senere gjennom en lignende tjeneste, Webex Teams. Her har gruppen skrevet oppsummeringer av morgenmøtene, spurt tekniske spørsmål rundt problemdomenet, og oppdatert oppdragsgiver om interessante funn. Gruppen har også hatt regelmessige besøk av oppdragsgiver rundt annenhver fredag. Disse besøkene har vært en kombinasjon av uformelle møter og demonstrasjoner, der gruppen har tatt opp spørsmål, vist hva som har blitt gjort siden sist og oppdragsgiver har kommet med ønsker. Disse ønskene har umiddelbart blitt gjort til krav og brukerhistorier i backlogen. I slutten av hver sprint har gruppen også sendt retrospektivene fra sprinten til oppdragsgiver pr. e-post. Gjennom dette har gruppen holdt seg transparent og oppdragsgiver har kunnet holdt seg oppdatert på hvordan gruppearbeidet går.

3.8 Møter med internveileder

Gjennom prosjektet har gruppen vært veiledet av Kiran Raja, førsteamanuensis ved Universitetet i Sørøst-Norge. Internveileder har bakgrunn innen både maskinlæring og informasjonssikkerhet, felter gruppemedlemmene har begrenset eller ingen erfaring innenfor. Regelmessige møter har derfor vært essensielt for prosjektet og møtene har blitt holdt enten på grupperom eller over Skype. Under disse møtene har internveileder bistått gruppen med hvilke maskinlæringsalgoritmer det skal sees på, hvordan datasettet skal prosesseres og øvrige ting som clusteringalgoritmer, datahåndtering, funksjoner i programpakken, endringer i rapporten m.m. På samme måte som med oppdragsgiver har disse møtene resultert i krav og brukerhistorier som har blitt lagt inn i backlog enten under møtene, eller rett etter at møtene har konkludert.

3.9 Stilguide & navnekonvensjoner

Gruppen har benyttet seg av stilguiden PEP 8 [21], som er en standardisert stilguide laget av The Python Foundation.

3.10 Teknisk dokumentasjon & docstring

Den tekniske dokumentasjonen blir generert ved hjelp av verktøyet Sphinx. Gjennom dette kan den tekniske dokumentasjonen lagres som en HTML-nettside, som en .pdf-fil og i LaTeX format.

For den tekniske dokumentasjonen benyttes docstring-konvensjonen til Google [22] kombinert med markdown syntaks som tolkes av Sphinx. Dette lar funksjonsbeskrivelser lese både i en IDE, så vel som gjennom den genererte dokumentasjonen. Docstringene skrives på engelsk.

```
"""
**Description**:
    - Find the best classifier from clf_list generated from grid_search().
**Args**:
    - clf_list          (list): List of classifiers.
    - x_test            (pandas.DataFrame): A pandas DataFrame with the features for test.
    - y_test            (pandas.DataFrame): A pandas Dataframe with the labels for test.
    - scoring           (string): String containing the scoringtype.
    - n_jobs            (int, optional): Multithreading option. -1 if use of all cores. Default=None.
**Returns**:
    - current_best_clf   (sklearn classifier): The best classifier within clf_list.
    - current_best_score (float): The score of the best classifier.
    - all_scores         (list): All scores in a list.
"""
```

Figur 3.7: Et eksempel på docstring-syntaksen som benyttes i prosjektet.

heimdall.util.crossvalidation_gridsearch.best_clf(*clf_list, x_test, y_test, scoring='F1 Score', n_jobs=None*)

Description:

- Find the best classifier from clf_list generated from grid_search().

Args:

- clf_list (list): List of classifiers.
- x_test (pandas.DataFrame): A pandas DataFrame with the features for test.
- y_test (pandas.DataFrame): A pandas Dataframe with the labels for test.
- scoring (string): String containing the scoringtype.
- n_jobs (int, optional): Multithreading option. -1 if use of all cores. Default=None.

Returns:

- current_best_clf (sklearn classifier): The best classifier within clf_list.
- current_best_score (float): The score of the best classifier.
- all_scores (list): All scores in a list.

Figur 3.8: Den samme docstringen kompilert av Sphinx.

4 Akademisk Teori

4.1 Informasjonssikkerhet

Å redegjøre for hva informasjonssikkerhet er, hvordan det har fungert gjennom tidene og hvordan feltets fremtid ser ut er essensielt for å forstå konteksten i dette prosjektet, og behovet for nestegenerasjons teknologi. Informasjonssikkerhet er som navnet tilsier det å sikre informasjon, mer konkret, å sikre skjermingsverdig informasjon fra uvedkommende. I Lov om nasjonal sikkerhet § 5-1 [23] er skjermingsverdig informasjon definert følgende:

> "Informasjon er skjermingsverdig dersom det kan skade nasjonale sikkerhetsinteresser at informasjonen blir kjent for uvedkommende, går tapt, blir endret eller blir utilgjengelig."

I påfølgende kapittel under samme lov, § 6-1, defineres også informasjons*systemer* som skjermingsverdige om de inneholder informasjon som faller under de overnevnte omstendigheter. Ergo er informasjonssikkerhet et fagfelt hvis oppgave det er å forhindre uautorisert tilgang, bruk og manipulering av konfidensiell og kritisk informasjon.

4.2 Tradisjonelle informasjonssikkerhetstiltak

Behovet for å ha dedikert programvare til å beskytte systemer fra uautorisert tilgang går langt tilbake - til og med før internett ble lansert som et kommersielt produkt - og på den tiden var kjent som DARPANET. Programmet Creeper ble skrevet av Bob Thomas i 1971 [24], og var designet for å bevege seg mellom DEC-PDP-10 stormaskiner via DARPANET, og etterlate meldingen "I'M THE CREEPER: CATCH ME IF YOU CAN". Creeper ble senere modifisert av Ray Tomilson til å kopiere seg selv på tvers av maskiner, og er derfor anerkjent som den første data-ormen. Tomilson skrev programmet Reaper for å identifisere og slette Creeper fra maskiner [25].

Fra dette ser man at hele tjue år før internett-boomen på nittitallet, var dataforskere klare over farene ved skadevare. Over årene har mengeden skadevare økt eksponentielt, og tiltakene for å motarbeide dem likeså. Denne oppgaven faller innunder sistnevnte kategori: å motarbeide skadevare. Men for å forstå hvordan, og

ikke minst *hvorfor* maskinlæring kan og bør brukes til dette formålet, må man se på konteksten det jobbes innenfor. Spesifikt hvordan skadevare har blitt motvirket opp gjennom årene, og hvordan disse teknikkene former behovet for dette prosjektet.

4.2.1 Byte- & hash-matching

Matching går ut på å analysere et program (byte-matching), eller en hash generert av et program (hash-matching), tegn-for-tegn og sammenligne det med en database over kjente skadevarer og trusler. Dette har vært måten sikkerhetsorganisasjoner og antivirusleverandører har stoppet angrep og ondsinnede programmer siden de først dukket opp på åttitallet [26]. Moderne obfuskeringsteknikker [27, 28] som polymorfisme [29], der en skadevare endrer koden sin etter at den har blitt sluppet løs, kommer seg effektivt rundt disse tiltakene, da alt som trengs er at ett enkelt tegn endrer seg for å komme seg rundt matchingen.

4.2.2 Logikk & heuristikk

En logisk analyse av potensiell skadevare er å etablere et sett med regler for hvilken adferd det er akseptabelt for et program å ha. Dette kan være å for eksempel etablere en hviteliste med servere knyttet til visse programmer. Om et program da begynner å snakke med en server utenfor denne hvitelisten, er dette et tegn på at programmet kan være infisert med skadevare.

Heuristisk analyse tar logisk analyse litt lengre. Dette er en teknikk som kombinerer en rekke regler (i motsetning til én og én regel individuelt), og ser på hvordan et program ter seg i forhold til disse reglene. Fra dette kompileres det en poengsum basert på adferden til programmet, dernest settes det en terskel for hvilken poengsum som tilsvarer mistenkelig oppførsel. Denne teknikken forsøker å etterligne hvordan en menneskelig analytiker ville gått fram i møte med programvare som har atypiske adferdsmønstre [30].

4.2.3 Trussellandskapet endrer seg: Advanced Persistent Threats

De overnevnte signatur- og regelbaserte metodene beror på etterretning om skadevarer og trusler som eksisterer. Men mot midten av totusentallet dukket Advanced Persistent Threats, eller APT-er, opp [31]. Dette er en type trusselaktør med store midler og kompetanse, ofte stater eller kontraktører. Disse kontraktørene jobber for etteretningstjenester eller gjennomfører egne storskala angrep for økonomisk vinning. APT-er endret trusselbildet ved å introdusere svært sofistikerte angrep. Mange av dem benyttet opp til det punktet ukjente teknikker, infeksjonsvektorer og exploits, såkalte zero-day angrep [32]. Nye teknikker behøvdes for å skjerme informasjon fra denne nye trusselen.

4.2.4 Sandboxing

Med framveksten av høyt avanserte skadevarer og angrepsvektorer ble sandboxing introdusert [33]. En sandbox er et virtuelt miljø som tåler å bli angrepet. Hensikten

er å kjøre skadevare i et lukket miljø der den ikke får tilgang til resten av systemet [34]. Men som med både heuristikk og signaturer har trusselaktører begynt å iverksette tiltak som forhindrer sandboxing, såkalt sandbox evasion [35][28]. Dette kan være så enkelt som å benytte seg av tid; å legge skadevaren i "dvale" slik at den kommer seg rundt en sandboxing-prosess for så å aktivere seg når den er inne på skjermede systemer. Men skadevare har også muligheten til å oppdage om den ligger i et virtuelt miljø. VMWare, en leverandør av virtualiserings-software, har selv kartlagt flere mekanismer APT-er bruker for å oppdage dette i deres tjenester [36].

4.2.5 Problemer i tradisjonelle sikkerhetstiltak

Det digitale trussellandskapet har endret seg dramatisk siden starten av årtusenet. Spydspissen i sikkerhetsmiljøet har de siste årene vært å bruke etterretning om kjente trusler generert ved sandboxing, menneskelig analyse av endepunkters adferd og informasjonsdeling på tvers at sikkerhetsavdelinger. Trusselaktører og sikkerhetsoffiserer er låst i et våpenkappløp som best kan beskrives som en katt-og-mus-lek, der nye angrepsmetoder og skadevarer dukker opp daglig. Sikkerhetsinstituttet AV Test registrerer alene 350.000 nye skadevarer og Potensielt Uønskede Applikasjoner (PUA) *hver eneste dag* [37]. Med denne enorme mengden angrep, som har en stadig økende sofistikeringsgrad, behøves en ny state-of-the-art.

4.3 Endpoint Detection & Response

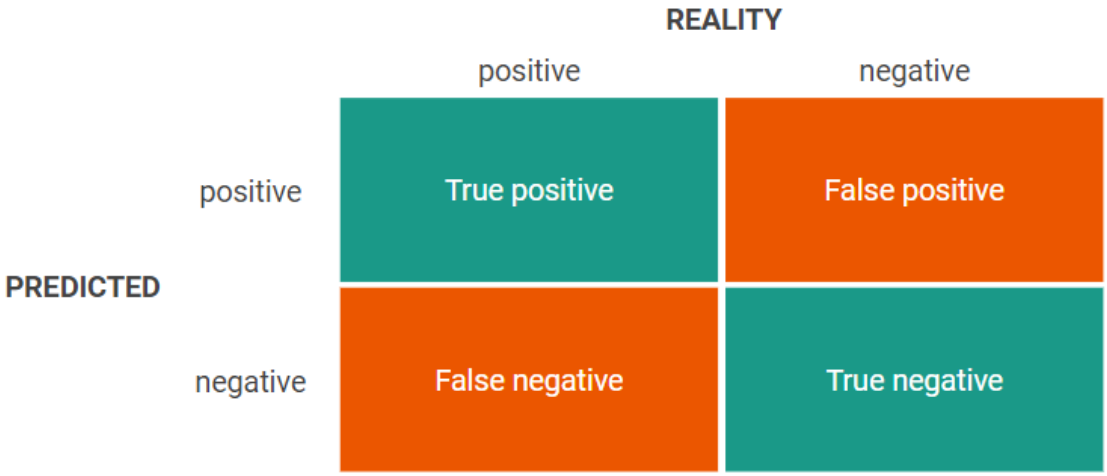
EDR er en relativt ny term innen sikkerhetssektoren. Termen var i 2013 Endpoint Threat Detection & Response (ETDR), og kom som et samlebegrep på verktøy som oppdager og undersøker mistenkelig adferd på endepunktsystemer, eller hostsystemer. I 2015 ble det omdøpt til EDR [38]. Disse systemene jobber med informasjon fra endepunktene og deres nettverkstrafikk, hvor informasjonen typisk sendes inn til en database for prosessering. EDR-systemer er ment til å hjelpe i beskyttelsen mot avanserte aktører, ved å tilby konstant overvåkning og mulighet for respons.

4.4 Maskinlæring

Maskinlæring (ML) er en del av fagfeltet kunstig intelligens, hvor modeller trenes til å analysere data ut i fra data modellen tidligere er vist [39]. Modellene ønsker å utføre enten en klassifisering eller en regresjon ut ifra ny data vist. Klassifiseringene kan være binære, positive/negative, hvor positive klasse typisk er det man ser etter, eller flerklasset; hund, katt, fugl. For å utføre treningen er det tre metoder utforsket i denne rapporten, om eksemplene modellen trener på har definerte labels, kalles dette supervised learning. Hvis lables derimot ikke er definert, vil metoden kalles en unsupervised metode [40]. Kombinasjoner av disse metodene kalles semi-

supervised learning, og er brukt for å høste styrkene fra begge metodene. Hvilken metode som er best egnet kommer an på konteksten. Generering av et lablet datasett krever store ressurser, men produserer forutsigbare resultater. Et datasett uten label derimot er raskt generert, men resultatene produsert av en modell trent på dette er ofte uforutsigbare. Det kan derfor ofte lønne seg og bruke en semi-supervised metode [41].

4.5 Analyse av trente modeller



Figur 4.1: Confusion matrix.

Når en maskinlæringsmodell er ferdig trent må den analyseres for å vurdere hvordan den utfører oppgaven. Ved å sette resultatene inn i en "confusion matrix" sammen med de virkelige klassifiseringene av dataen, analyseres klassifiseringene modellen har gjort [42]. Se figur 4.1. Med utgangspunkt i denne matrisen kan man gjøre numeriske beregninger, som gir informasjon om hvor god modellen er[43].

Vurdering av en hendelse		**Klassifisering**
Virkelighet	Forutsigelse	
Ondsinnet	Ondsinnet	True Positive
Ondsinnet	Godsinnet	False Negative
Godsinnet	Ondsinnet	False Positive
Godsinnet	Godsinnet	True Negative

Tabell 4.1: Beskrivelse av hva en klassifisering betyr i kontekst av sikkerhet

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total} \tag{4.1}$$

Accuracy, likning 4.1, er nøyaktighet, og gir informasjon om forholdet mellom antallet modellen gjettet riktig i forhold til totalen. Da nøyaktigheten representerer den underliggende klassedistribusjonen, kan denne typen beregning gi misledende resultater når distribusjonen er skjevfordelt, dette betegnes som et "Accuracy Paradox" [44].

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{4.2}$$

Presisjon, likning 4.2, er et mål på hvor mye modellen gjettet på som positiv klasse, som faktisk var en positiv klasse.

$$Recall / True\ Positive\ Rate / Sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{4.3}$$

Recall, også kalt True Positive Rate(TPR), likning 4.3, er en beregning av sann positiv i forhold til antallet faktiske positive verdier.

$$Specificity / False\ Positive\ Rate = \frac{False\ Positive}{False\ Positive + False\ Negative} \tag{4.4}$$

Relatert er False Positive Rate (FPR), liking 4.4, en beregning på hvor ofte modellen gjetter på positiv klasse, når klassen er negativ.

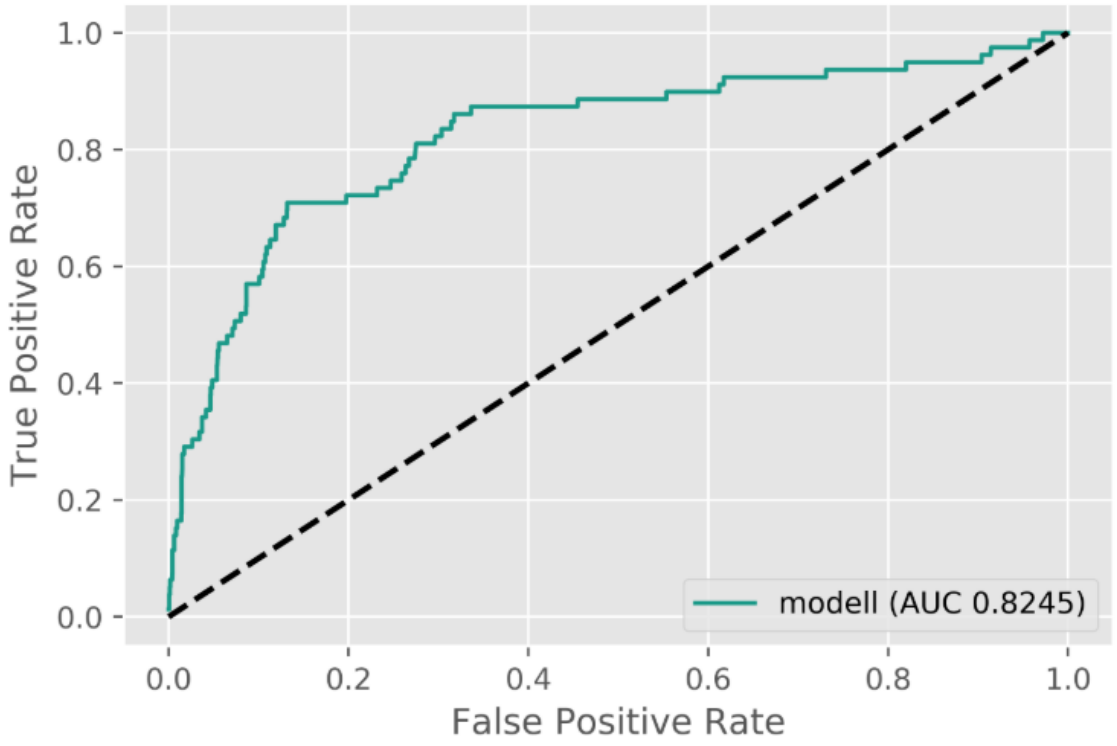
Recall og presisjon målt opp mot hverandre omtales som en tøff krig i Google's maskinlæringsmodell-introduksjon [45]. Dersom vi prøver å redusere mengden falske negativer vil modellen klassifisere flere hendelser som positive, derav er det er stor sjanse for at den klassifiserer negative hendelser som positivt. Omvendt vil en modell klassifisere flere hendelser som negativt for å minske falske positiver øke sannsynligheten for falske negativer. Som en løsning på dette trengs det en måte å se på disse to beregningene i forhold til hverandre, og en beregning for dette er "F1 Score" [46].

$$F1\ Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4.5}$$

Ved hjelp av F1 score, likning 4.5, vil man kunne beregne balansen mellom presisjon og recall. F1 score vil kunne gi et kort og konkret tall på hvor balansert og gode presisjon og recall er ved en gitt terskel, noe som kan brukes i en vurdering av hvor god en modell er i en setting hvor datasettet modellen opererer på er ubalansert. Disse likingene gir et overblikk over hvordan modellen gjør det ved en satt terskel. Om en ønsker å visuelt undersøke hvordan modellen ville gjort det ved forskjellige terkler kan "Receiver Operating Characteristic"-kurver (ROC) eller "Precision Recall" (PR)-kurver gi nyttig informasjon.

4.5.1 ROC-kurve

ROC-kurver var originalt utviklet for å analysere radarsignaler under andre verdenskrig. Disse kurvene har vist seg nyttige for å evaluere nøyaktigheten av statistiske modeller [47]. Kurvene viser sammenhengen mellom FPR og TPR for ulike terskeler, slik at man kan definere terskel ut ifra kravene til modellen, se figur 4.2.

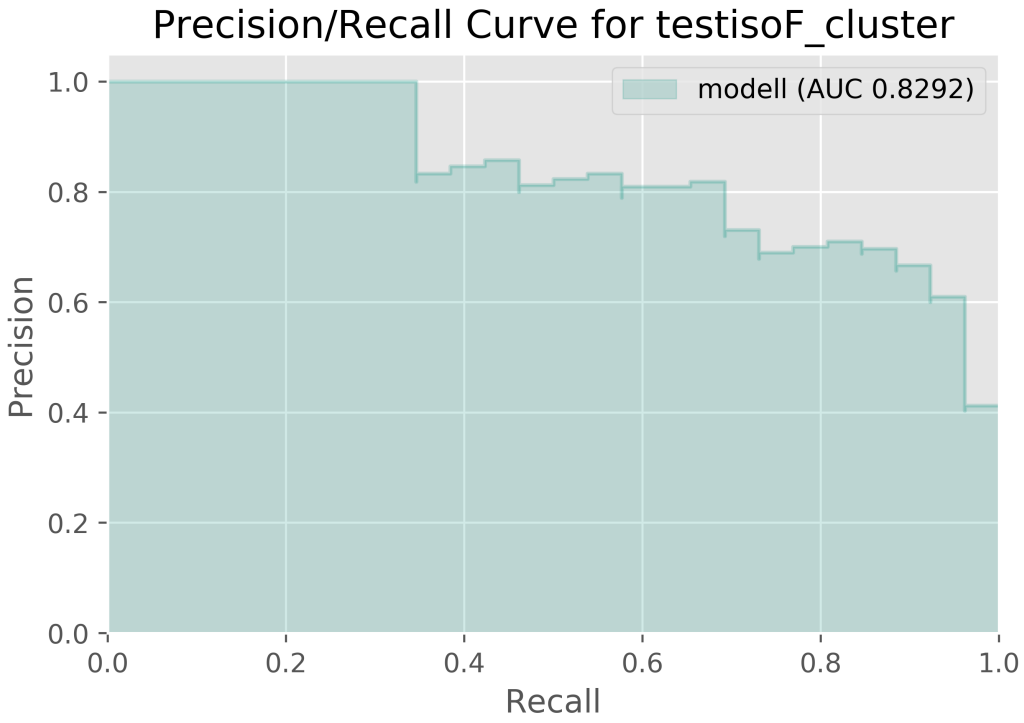


Figur 4.2: Illustrert eksempel - ROC-kurve for trent modell.

Problemet med ROC-kurver kommer når man bruker de på ubalanserte datasett, da ROC har vist seg gi et misvisende inntrykk av modellens ytelse [48]. PR-kurver derimot egner seg bedre for slike typer datasett, da de tar større hensyn til mengden sanne positiver [49].

4.5.2 PR-kurve

Ved å plotte forholdet mellom presisjon og recall ved ulike grenseverdier, har vi muligheten til å se hvordan disse opptrer i forhold til hverandre. Disse kurvene gir et nyttig innblikk, hvor alle grenseverdier er representert, selv med skjevfordelt data [49]. Figur 4.3 illustrerer dette. Figur 4.3 viser PR-kurven for tilsvarende modell som ROC-kurven ovenfor (Figur 4.2), der det er relativt lav recall dersom man øker presisjonen.



Figur 4.3: Illustrert eksempel - PR-kurve for trent modell.

4.6 Ubalanserte datasett

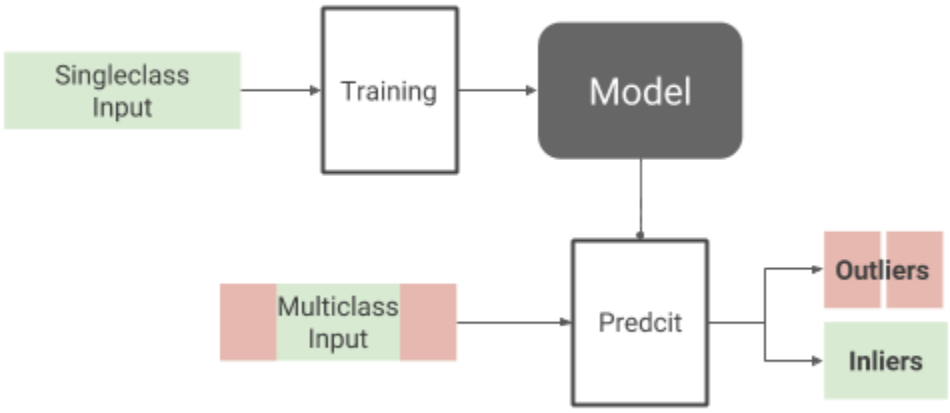
Hensikten med en maskinlæringsmodell er å tilpasse seg klassifiseringene i treningsdata, for så å foreta klassifiseringer på annen data. Hvis klassene i treningsettet er representative av klassene i datasettet modellen kommer til å utføre beslutninger på, vil modellen kunne gjøre de rette klassifiseringene. I tilfeller hvor én klasse dominerer datasettet, vil en modell generalisere ut ifra denne fordelingen, og derfor beslutte at de fleste av eksemplene tilhører majoritetsklassen [50]. Dette vil føre til en stor bias mot majoriteten. Innen konteksten informasjonssikkerhet vil det være ubalanse i datasettet. EDR-systemer genererer store mengder logger, hvorav det aller meste er godsinnede hendelser. Derfor må det opprettes en metode for å kunne operere innen denne konteksten, uten at modellene skaper en uønsket bias. Innen Intrusion Detection Systems (IDS), har anomaly detection metoder blitt brukt for å separere normal og uvanlig oppførsel [51]. Disse metodene har gitt gode resultater, selv om de opererer innen en kontekst der det er stor ubalanse.

4.7 Anomaly detection

Anomaly detection handler om å finne uforutsette, eller uvanlige hendelser [52] innen en kontekst. Disse teknikkene har muligheten til detektere zero day-angrep,

men er i en nettverkssammenheng kjent for å kunne generere en del falske positiver [53]. For å bestemme hva som er et avvik finnes det to kategorier.

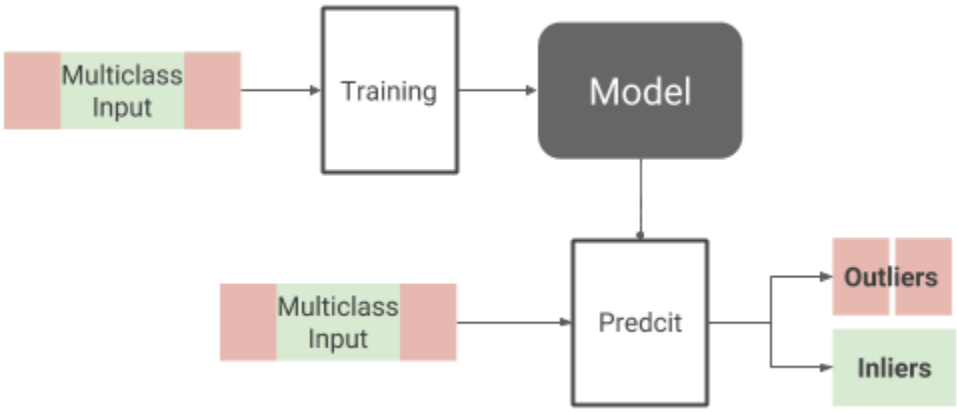
4.7.1 Novelty Detection



Figur 4.4: Generelt hierarki for novelty detection

Novelty detection modeller trener på kun en klasse data, slik at de kan bestemme om ny data hører til sammen med dataen den er trent på. Hvis det nye eksempelet hører til blir det klassifisert som en "inliner", men om det avviker blir det klassifisert som en "outlier" [54]. Dette er visualisert i fig 4.4. Siden modellen kun er trent på én klasse data kan man redusere store ressurser ved å slippe å klassifisere alle eksempler i treningsettet, så lenge man har tilgang på kjent normal data.

4.7.2 Outlier detection



Figur 4.5: Generelt hierarki for outlier detection

Outlier detection modeller trener på all data innen en kontekst og ser hvilke samples som skiller seg ut [55, 56]. Dette er visualisert i fig 4.5. Ved å trekke en grense for hvor mye et eksempel kan skille seg ut, en terskel, kan man bestemme hva modellene klassifiserer som en outlier. I disse modellene er det viktig å verifisere resultatene på treningen, slik at man kan sette en teskel som genererer fornuftige resultater.

4.8 Clustering

Clustering kan ses på som en unsupervised maskinlæringsmetode, som grupperer data i grupper (cluster) etter hvor ”like” de er [57]. Data tilhørende et cluster er på ett vis mer like enn data i andre clustere eller ikke-clustere. Antall clustere kan, avhengig av algoritme, enten bestemmes av algoritmen selv (eks. DBSCAN) eller forhåndsbestemmes (eks. K-means). Hvordan data havner i ulike clustere avhenger av cluster-type. Clustering blir gjerne brukt for å se underliggende sammenhenger i datasettet som ellers ikke kommer tydelig nok frem. To godt kjente clustering-algoritmer er K-Means og DBSCAN.

4.8.1 K-Means

K-means-clustering ligger i kategorien ”partitional clustering”, hvor man ser på avstanden mellom datapunktene for å avgjøre hvilket cluster de ulike datapunktene skal tilhøre. Typisk bruker man Euclidean Distance eller Manhattan distance for å avgjøre avstanden mellom punktene. Man bestemmer antall clustere k og man bestemmer (eller velger tilfeldige) ”centroids” som utgangspunkt for clustrene. Alle datapunktene blir tilegnet et cluster, for så å regne ut den gjennomsnittlige avstanden i clusteret. Dette vil generere et nytt centroid. Denne prosessen repeteres inntil det ikke skjer endringer [58].

4.8.2 DBSCAN

DBSCAN ligger i kategorien ”density-based clustering”, som handler om antall punkter i en sirkel omgitt av et punkt. På forhånd bestemmer man ikke antall clustere, men man bestemmer to parametere: ”eps”, som er radien på sirkelen, og ”minpts”, som er minimum antall punkter man ønsker å ha innenfor sirkelen. Hvis antallet punkter innenfor sirkelen er større enn minpts, er punktet et core-point. Hvis antallet er mindre, men det er core-points inne i sirkelen har man et border-point. Og hvis antallet er mindre og det ikke finnes core-points i sirkelen er det et noise-point [59].

4.9 Online learning

I ”online learning” blir ny data sendt inn i modellen over tid, uten å måtte trene på tidligere sett data på nytt. Online learning viser seg å være meget ressursnyttig, noe

som kommer godt frem når dataen blir tilgjengelig litt etter litt, som foreksempel i "streaming", eller med store mengder data. På denne måten kan modellen tilpasse seg, og minske loss for hver iterasjon [60]. Ved "Offline learning", eller statisk læring, blir modellen oppdatert etter at hele treningssettet er representert. Dette medfører at ny usett data ikke vil kunne bli med i læringsprosessen uten å trene modellen på nytt. Offline learning kan være hensiktsmessig på mindre datasett, men er svært utsatt for lokale minimum, og veldig ineffektivt når det kommer til store datasett [61].

En del av modellene som presenteres i denne rapporten støtter ikke inkrementell læring, grunnet restriksjoner av algoritmen selv. I denne rapporten introduseres to online algoritmer, "Passive Aggressive Algorithm" og "Stochastic Gradient Decent".

4.9.1 Active learning

Active learning er en revidert supervised online learning metode som aktivt fokuserer på å forutse de mest informative eksempelene ved å spørre om feedback. En vanlig inndeling når vi snakker om active learning er "pool-based learning" og "stream-based learning" [62]. I pool-based learning blir modellen vist et stor antall eksempler, hvor modellen velger selv de mest informative eksemplene. I stream-based learning tar modellen stilling til alle eksemplene, men det er kun de mest informative eksemplene modellen vil spørre om feedback for. For å velge de mest informative eksemplene krever en active learning algoritme en stategi for å velge disse, kalt "query selection strategy" [63]:

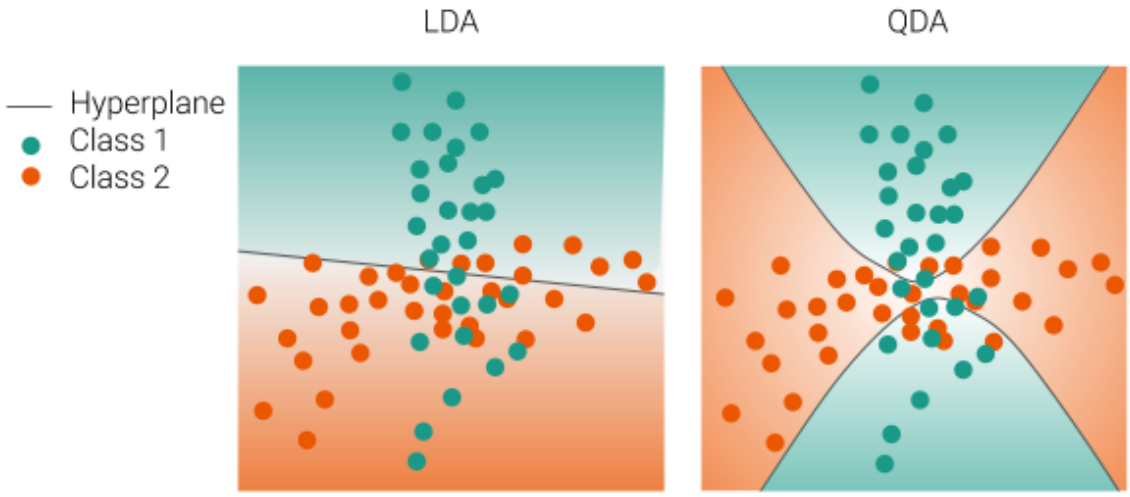
Uncertainty Sampling Avstand fra hyperplan

Query by Commitee En samling av modeller spør om eksempler modellene er mest uenige om.

4.10 Algoritmer

4.10.1 Linear/Quadratic Discriminant Analysis

Diskriminantanalyse forsøker å klassifisere fordelingen av data inn i grupper, for så å bruke Bayes' theorem til å estimere sannsynligheten for at X hører til i klasse Y [64]. Forskjellen på den linære og den kvadratiske tilnærmingen er at hvor den linære (LDA) estimerer en felles kovarians matrise for *alle* klassene, beregner en kvadrisk diskriminant (QDA) en matrise *per* klasse. Dette fører til at LDA får en høyere bias med lavere varians, mens QDA får en lavere bias med høyere varians [65]. Disse algoritmene er tradisjonelle supervised klassifikatorer. Typisk vil grensene visuelt se forskjellige ut, se figur 4.6.



Figur 4.6: Illustrasjonsbilde med pseudo-tilfeldig plasserte klasser. Illustrerer de forskjellige decision boundary-ene. LDA til venstre, QDA til høyre.

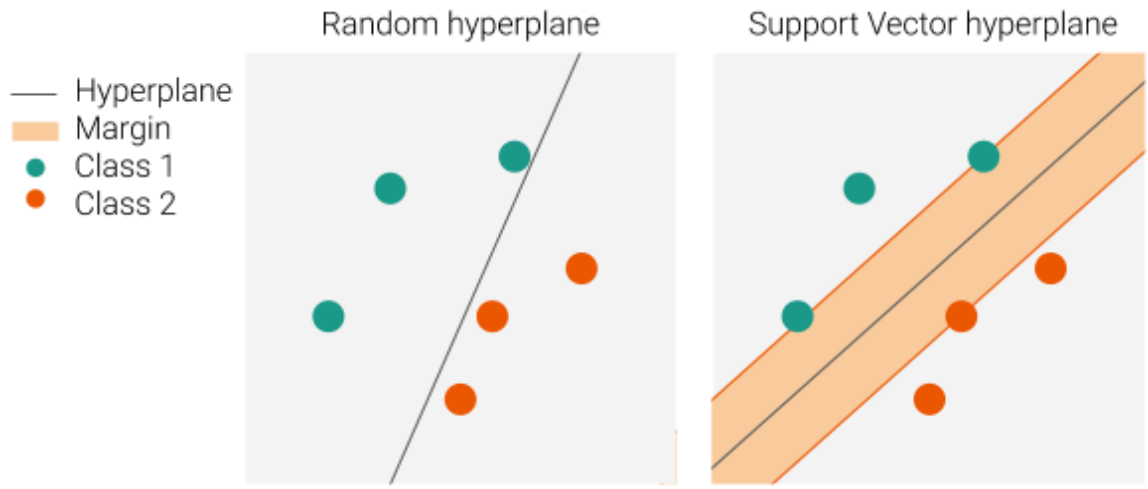
4.10.2 Support vector machines

SVM, eller support vector machines, er en maskinlæringsalgoritme som baserer seg på å øke dimensjonaliteten på input, slik at det kan konstrueres en lineær beslutningsflate i det høydimensjonale rommet. Algoritmen ønsker å definere en marg, der klasseskillet er høyest, slik at den best mulig kan generalisere læringen [66].

SVM kan også nyttes innen novelty detection ved å omstrukturere til en One-Class-SVM. Denne typen SVM ønsker å skape hyperplan rundt ønsket data, slik at alt som skiller seg ut fra dette blir klassifisert som en outlier [67].

4.10.3 Isolation forrest

Isolation forest (IsoF) er en outlier detection algoritme som forsøker å isolere outliers ut ifra prinsippet "few and different". Ved å anta at det er minoriteten som skiller



Figur 4.7: Illustrasjon av vilkårlig valgt hyperplan, opp mot et plan skapt ved SVM

seg ut, og at disse varierer stort fra normalen, bygger algoritmen en samling av isoleringstrær. Ut ifra denne samlingen finner algoritmen de punktene som i snitt har kortest vei ned i trærne [68].

4.10.4 Local Outlier Factor

Local Outlier Factor (LOF) har tilnærming til outlier detection der modellen estimerer i hvilken grad et eksempel er en outlier lokalt innen datasettet. Modellen sammenligner et eksempel med punktene rundt og beregner faktoren deretter [69]. Denne faktoren blir presentert som et tall, hvor høyere tall tyder på et større avvik.

4.10.5 Passive Aggressive Algorithm

PAA er en online algoritme, som holder seg passiv når den har en loss-verdi på 0, men går aggressivt inn for å rette på vektene om loss-verdien er over en satt grense. Denne algoritmen kan ha flere typer loss, samt flere varianter av innstramming ved feil.

Passive Aggressive Algorithm ønsker å separere eksemplene best mulig ved å danne et hyperplan med størst mulig margin. En vektor w brukes for å danne dette hyperplanet. Når et nytt eksempel, x med fasit $y(\pm 1)$, blir vist for algoritmen regner den ut graden av selvtillit ved å regne ut størrelsen $|w \cdot x|$. Dersom det er mulig ønsker algoritmen å danne en avstand 1 fra hyperplanet så langt dette er mulig. Dersom avstanden er mindre enn 1, eller eksemplet i verste fall er feilklassifisert får modellen et øyeblikkelig "loss". Algoritmen tar utgangspunkt i følgende "hinge loss":

$$\lambda(w;(x,y)) = \begin{cases} 0 & ,y(w \cdot x) >= 1 \\ 1 - y(w \cdot x) & ,\text{ellers} \end{cases} \tag{4.6}$$

Viktige elementer i PAA er at algoritmen krever at nåværende eksempel blir klassifisert korrekt, samtidig som w_{t+1} er så nærme w_t som mulig. Regelen for å oppdatere er derfor definert som følger for PAA:

$$w_{t+1} = w_t + \tau_t \cdot y_t \cdot x_t \Rightarrow \tau_t = \frac{\lambda_t}{\| x_t \|^2} \tag{4.7}$$

For at algoritmen ikke skal gå for aggressivt til verks dersom det er feil label eller ved støyende eksempler er det også innført et parameter C som tar høyde for nettopp dette. I PAA-1 blir regelen for å oppdatere vekt w definert på følgende måte:

$$w_{t+1} = w_t + \tau_t \cdot y_t \cdot x_t \Rightarrow \tau_t = \min\{C, \frac{\lambda_t}{\| x_t \|^2}\} \tag{4.8}$$

En oppsummering av algoritmen i korte trekk er gitt under:

— PAA i korte trekk. -
Input: C
Initialiser $w = (0, ..., 0)$
For alle eksempler t for t = (1,2...):

- Forutse: $\hat{y} = sign(w \cdot x_t)$
- Motta fasit y
- Kalkuler loss, se 4.6.
- Oppdater med en av reglene:
  - PAA, se 4.7
  - PAA-1, se 4.8

4.10.6 Stochastic Gradiant Decent

SGD beregnes som en særdeles effektiv optimaliseringsteknikk for lineære klassifiseringsmodeller. Oppdatering av vektene skjer inkrementelt slik at modellen skal ende opp med de vektene som gir lavest loss. Ved å regne ut gradienten til alle features kan man tilpasse disse vektene ved hjelp av små steg som blir kalkulert ut ifra en learning-rate. Learning-raten sørger for at stegene algoritmen tar blir store dersom algoritmen er langt unna, og minsker stegene når algoritmen nærmer seg et globalt minimum. Siden modellen er stokastisk blir eksemplene den arbeider med tilfeldig selektert ut ifra treningen som presenteres [70]. Eksempel på en lineær modell er SVM, og ved loss med ”hinge” tilpasser SGD seg en online versjon av SVM [71].

4.10.7 Ensemble learning

Et ensemble er en samling av forskjellige klassifiseringsmodeller hvis klassifikasjoner eller decision scores kan benyttes til å stemme fram en endelig klassifikasjon

[72]. Et ensemble er i praksis å anse som en klassifikasjonsmodell i seg selv, da den både kan trene og klassifisere data som en hvilken som helst annen maskinlæringsalgoritme. Det finnes flere ensemble-metoder, og i dette prosjektet benyttes to av dem, stemming og vektet stemming. Stemming vil simpelthen si å plusse sammen decision scoren fra hver modell for hvert datapunkt, og returnerer disse summene som decision scoren til ensemblet. La n være antallet klassifiseringsmodeller i ensemblet, og ds_n være decision score for ett enkelt datapunkt fra modell n:

$$ds_{total} = ds_1 + ds_2 + ... + ds_n$$

I vektet stemming tar ensemblet hensyn til hvor korrekt en klassifiseringsmodell klassifiserer data. Her ganges decision scoren med arealet under PR-kurven (tallet i høyre hjørne på figur 4.3), w_n.

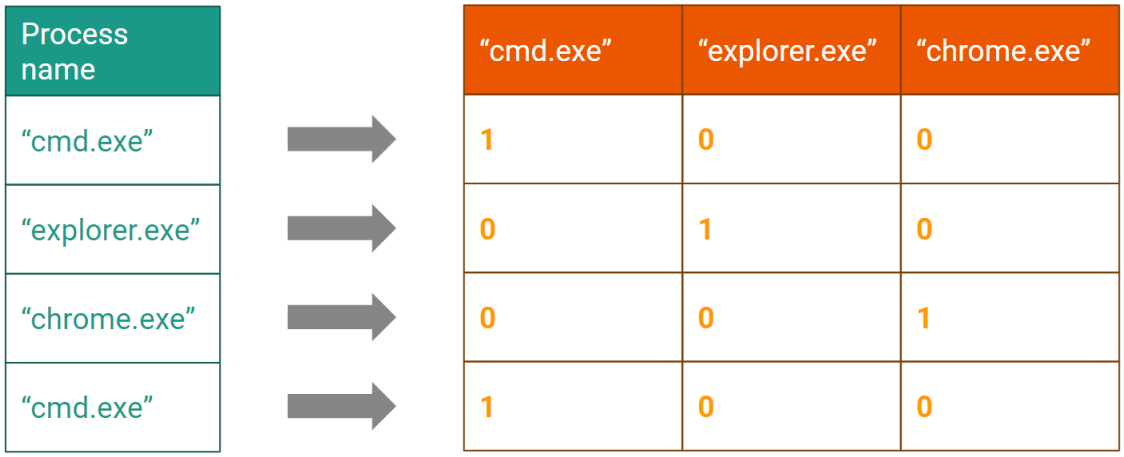
$$wds_{total} = w_1 ds_1 + w_2 ds_2 + ... + w_n ds_n$$

Om den nye decision scoren er større enn null, vil ensemblet klassifisere datapunktet som 1, og -1 for mindre enn eller likt null.

$$pred = \begin{cases} 1 & ds > 0 \\ -1 & ds \leq 0 \end{cases} \tag{4.9}$$

4.11 Preprocessing

Data kommer i alle typer og former. For at en maskinlæringsmodell skal kunne trene og forutse på denne dataen, må den først behandles [73, 74]. I maskinlæring kan data deles inn i forskjellige typer kategorier [75]. De typene vi vil fokusere på er numeriske og kategoriske, også kalt nominelle. Disse to typene må preprosesseres på forskjellige måter. Numeriske typer kan ha verdier som strekker seg over ulike områder. To forskjellige kolonner med numeriske verdier kan ha svært ulike skalaer. Det er derfor viktig å normalisere de numeriske kolonnene, slik at maskinlæringsmodellen får en raskere læringskurve, og ikke vekter dataen på en ubalansert måte. Alternativer for å normalisere dataen kan være min-maks ut ifra treningssettet, eller å passere alle verdier gjennom en logaritmefunksjon [74, 76]. Kategoriske typer vil i hovedsak si tekst, men kan også være lister eller andre ikke-numeriske typer. Den enkleste måten å representere kategoriske typer numerisk på, ville vært å gi hver unike instanse sin egen numeriske representasjon. Men da dette kan føre til feiltolkning av sammenhengen mellom verdiene av en maskinlæringsmodell, er det ofte foretrukket å gjøre det på alternative måter. One-hot-encoding er en vanlig måte å representere kategoriske typer numerisk på. Her lages det en ny kolonne for hver nye instans, hvor verdiene i kolonnene er 1 hvis rad det var som hadde denne verdien, og 0 på de andre [77, 76]. Se figur 4.8 for en demonstrasjon av one-hot-encoding.



Figur 4.8: One-hot-encoding.

En annen måte å representere kategoriske typer numerisk på, er ved "hash encoding". Her opprettes det et bestemt antall kolonner, for hver kategoriske kolonne i den ikke-preprosesserte dataen. Disse kolonnene representerer de ulike sifrene i et binært tall. Til sammen utgjør alle kolonnene til en spesifikk kategorisk type, outputen av den originale verdien sin hash, gjennom en spesifisert hashingfunksjon. På denne måten vil resultatet av å utføre hash encoding på en spesifikk type kategorisk data, alltid være den samme [78].

4.12 Cross validation

Cross validation er en metode for å vurdere og validere datasettet. Hovedsakelig går cross validation ut på at man splitter opp datasettet i deler (2 eller fler, avhengig av type/ønske), trening og testing, der man utfører læring på treningssettet og validerer på det usette testingsettet. Man bruker typisk er form for score som mål på hvor godt modellen gjør det på de ulike delene. Ved store varierende scoringer må datasettet som en helhet vurdereres og eventuelle tilpasninger må gjøres. Som nevnt er det flere måter å splitte datasettet på, men en av de vanligste metodene er "k-fold", der man velger k antall deler og tester på én av dem (eks. 10-fold, der 9 er trening og 1 er test), for så å gjøre dette for alle deler [79].

4.13 Grid search/Hyperparameteroptimalisering

Hyperparameter er en karakteristikk av modellen som ligger eksternt for modellen. Verdiene kan ikke beregnes fra dataen og må derfor settes på forhånd. Dette er parametre som C eller nu i SVM, eller n_neighbors i Local Outlier Factor. Grid search brukes til å finne de optimale hyperparametrene til en modell, for å gjøre

modellen mer ”nøyaktig”. De beste hyperparametrene brukes videre til trening. [80]

4.14 EDA

Exploratory Data Analysis (EDA) er et samlebegrep på analyser av datasett som baserer seg på interaktive hypoteser, gjerne ved hjelp av visuelle metoder. En EDA gir en god innsikt i datasettet, hvilke features som kan være nyttig og ha med, og hvilke features som er overflødige. Det finnes ulike teknikker for å utføre en EDA, beskrevet i [81]. Vanlige metoder er gjerne teknikker som lar oss studere én og én feature, kalt ”univariate visualization”. For numeriske verdier vil dette være spesielt interessant for å se om noen verdier skiller seg ut og er typisk for en bestemt klasse. En annen teknikk er ”multivariate visualization” som er et verktøy for å analysere flere features samtidig. Ved å analysere to og to features kan man luke ut features som representerer den samme informasjonen på ulike måter. Dette er overflødig informasjon, og gir ingen ekstra informasjon til modellen.

4.14.1 Dimensionality reduction

Dimensionality reduction handler om å redusere feature-settet, for å redusere kompleksiteten, samt irrelevante, villedende og overflødige features som kan gjøre læringsprosessen vanskeligere eller feilaktig. Nøyaktigheten og tidsbruken vil også vesentlig forbedres. Men å redusere dimensjonliteten for mye vil føre til mangel på data og informasjon som kan være viktig. Det er derfor viktig å utføre ulike algoritmer for å undersøke dataen i ulike features. Her fins det hovedsakelig to metoder: feature selection, hvor man velger et subsett av feature-settet, og feature extraction, hvor det genereres nye features ut i fra de featurene som allerede eksisterer [82]. Dette gjøres gjerne som en del av EDA.

4.15 Informasjonssikkerhet & maskinlæring i EDR-systemer

Dette kapittelet har tatt for seg tingenes tilstand i informasjonssikkerhets-sektoren, og gitt et overblikk over maskinlæringskonsepter som skal benyttes i prosjektet. Per i dag er det primært logikk, heuristikk, signatur-matching og sandboxing som benyttes for i sikre skjermingsverdig informasjon. Men felles for disse prosessene er at de beror på forutsigelser, prognoser og regler laget av mennesker, og felles for disse er at de i et stadig endrende landskap aldri kan være absolutte. Responssyklusen per i dag går ut på å observere en ny type angrep eller teknikk, analysere den, for så å endre sikkerhetstiltakene til å reflektere dette; scan og oppdater.

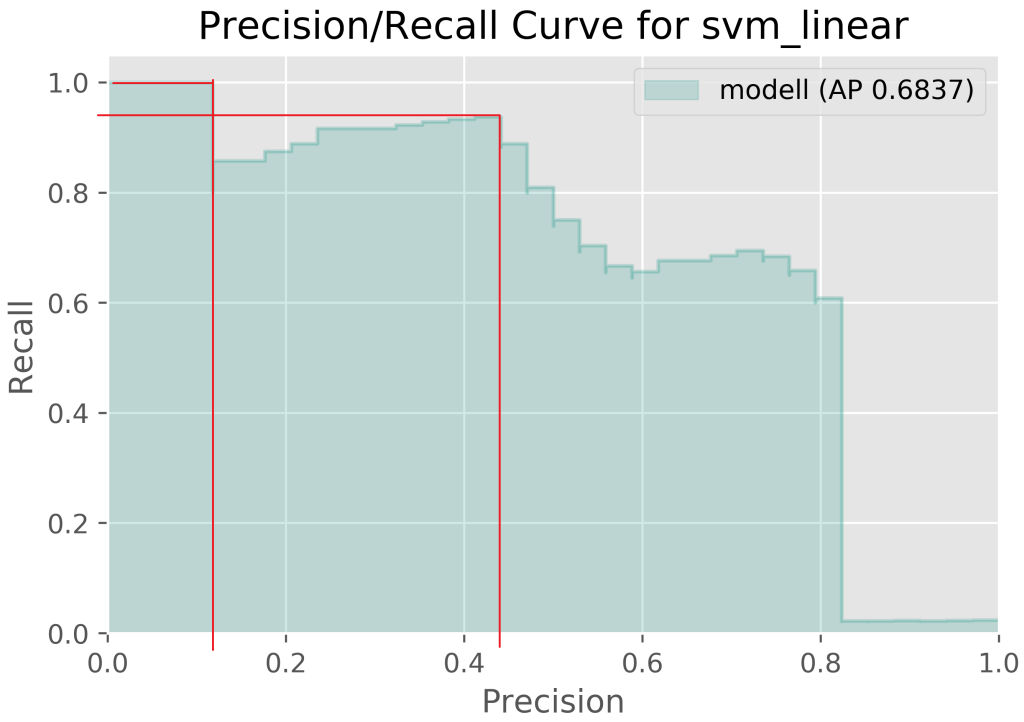
Men med store framskritt i maskinlæringsfeltet det siste tiåret, åpner muligheten for å høste en datamaskins overlegne matematiske egenskaper seg, egenskaper som kan bistå med å forutsi morgendagens angrep. Denne hovedoppgaven går ut på nettopp dette: å benytte seg av maskinlæring til å analysere data generert

av Endpoint Detection and Response systemer, og fra denne dataen identifisere ondsinnet adferd som benytter kjente og ukjente metoder.

5 Behov

5.1 Maskinlæringsmodeller

Når en maskinlæringsmodell er ferdig trent er det behov for å verifisere hvor godt modellen utfører oppgaven, men hvordan man vurderer dette kommer an på konteksten. I en treningskontekst vil vi ha tilgang på beregningene utledet i kapittel 4. Dette gir oss en mengde forskjellige verdier å vurdere modellene etter.



Figur 5.1: Visualisering av hvordan man kan bestemme grense ut i fra vekting av recall-verdi.

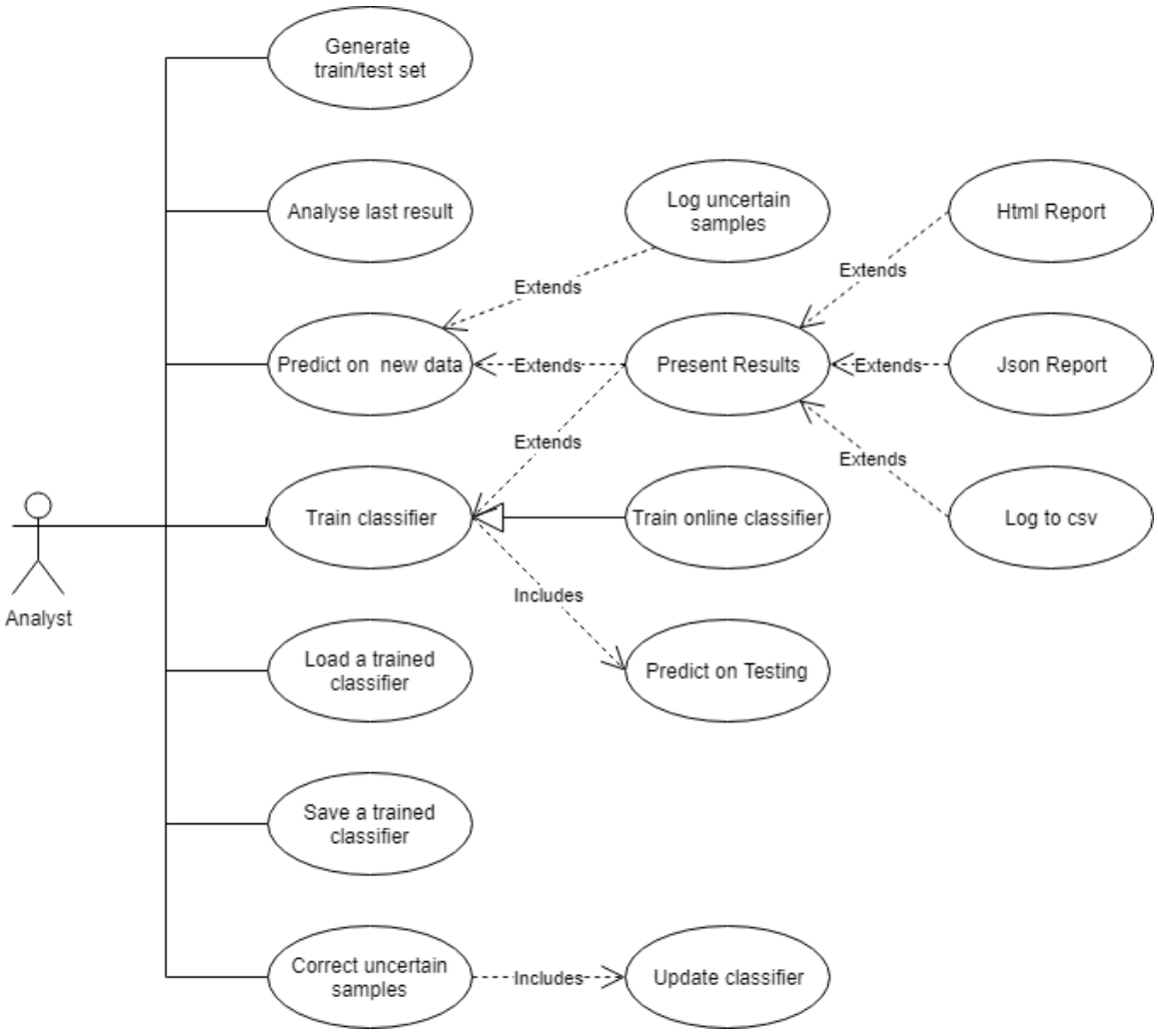
I kontekst av oppgaven, hvor målet er å detektere skadevare og trusler, vil det ønskelige tilfelle være at modellen oppdager alt som er uønsket, og flagger dette. Om den så først skulle gjøre feil, vil det være mindre skadelig om modellen flagger et godsinnet sample som ondsinnet enn om den flagger et ondsinnet eksempel

som godsinnet. Førstnevnte vil skape ekstra jobb, og ta opp arbeidstimer, mens sistnevnte vil kunne gjøre skade på systemet eller la en aktør komme inn i systemet. I vurderingen av modellen må det derfor være et stort fokus på å holde antallet falske negativer nede, dvs. ha en høy recall-verdi. Ut ifra likning 4.3 kan man se at recall vekter antallet sanne positiver i kombinasjon med antallet falske negativer. En recall-verdi på 1 vil derfor medføre at det ikke kan eksistere falske negativer, og dermed at ingen uønskede hendelser slipper gjennom. Recall på 1 er derfor ønskelig i dette scenariet.

For at modellen skal kunne utføre arbeid av verdi holder det ikke med 1 i recall, da modellen i teorien kan klassifisere alt som positivt og dermed ikke ha falske negativer. Dette vil føre til en ubrukelig modell da den ikke har noen kredibilitet i flaggene den oppretter, samtidig som den ville skapt store mengder ekstra arbeid for personellet som måtte undersøkt alle flaggene den kom med. Ved å kreve høyest mulig presisjon, med en recall opp mot 1, vil modellen ikke ha falske negativer, samt generere minst mulig falske positiver. Under disse begrensningene vil modellen kunne generere resultater av verdi uten at for mye av tiden går med på undersøkelse av unødvendige hendelser. Det blir derfor naturlig å legge mye vekt på PR kurver, om man ønsker å presentere den totale ferdigheten til modellen visuelt. Ved å bruke disse kurvene kan man visuelt identifisere ved hvilken grense modellen vil gi de beste resultatene, fig 5.1. Man kan så identifisere hvordan balansen mellom presisjon og recall er ved den satte grensen ved å bruke F1 score. I kontekst av cybersikkerhet, hvor data er svært ubalansert, vil tallet på denne balansen være en god måling på ferdigheten av modellen. For våre modeller betyr dette at vi ved hjelp av PR-kurvene vil sette den egnede grensen, for så å nytte F1 score til vår faste ferdighetsberegning.

5.1.1 Prioritering

1. Recall-verdi i umiddelbar nærhet av 1.
2. Bruke PR-kurvene for å analysere ved hvilken terskel modellen har best presisjon, med den krevde recall-verdien.
3. Vurdere F1 score ved satte terskel.



Figur 5.2: Usecase diagram for Heimdall pakken fra en analytikers perspektiv

5.2 Funksjoner i verktøyspakken

For å kunne gjennomføre maskinlæring på dataen fra EDR-systemet, og måle deres effektivitet i henhold til de overnevnte beregningene, må vi ha et verktøy som interfacer EDR-systemet med et maskinlæringsrammeverk. Dette verktøyet er dog ikke bare for å kunne gjennomføre analyseoppgaven, men også noe TCERT er interessert i å kunne gi til sin engineering-avdeling for videreutvikling. Ergo må innholdet i denne pakken kartlegges i henhold til både gruppens og TCERTs behov.

5.2.1 Scenarier

I møte med TCERT ble det fastsatt en rekke scenarier de så for seg at maskinlæringsmodeller ville bli brukt til på data fra EDR-systemet. Ut ifra disse scenariene kommer flere overordnede retningslinjer for funksjonalitet som vil påvirke imple-

mentasjonen av verktøyspakken. Disse er også generelt visualisert i figur 5.2.

5.2.1.1 On-site: Offline klassifisering av data

Et av scenariene som ble lagt fram for gruppen av oppdragsgiver, er en såkalt on-site situasjon. Dette vil si at en analytiker enten allerede befinner seg på eller reiser til en lokasjon der en sikkerhetshendelse finner sted. I dette scenariet vil analytikeren ha behov for å kunne benytte seg av en ferdigtrent maskinlæringsmodell til å klassifisere data for å forsøke å finne opphavet til et angrep. Dette vil si at innstillingene i en maskinlæringsmodell må kunne lagres til disk og lastes inn lokalt på en datamaskin. Det vil også være hensiktsmessig med både en supervised klassifikator som har blitt trent på ondsinnet data, så vel som en unsupervised klassifikator som har blitt trent på kun godsinnet data og leter etter avvik.

5.2.1.2 Langsiktig videretrening på ny data

En annen måte å klassifisere data på er over lang tid, der ny, tidligere usett data blir registrert av EDR-systemet og vist til modellen. I dette tilfellet vil man kunne ha en maskinlæringsmodell som lærer i kontekst av det den allerede kan, og ikke må trenes helt på nytt. I dette tilfellet er det online og feedback learning som skal benyttes (se seksjon 4.9).

5.2.2 Vanlige infeksjonsvektorer

I informasjonssikkerhet er det veldig vanlig for angripere å benytte seg av visse svakheter i bestemte systemer, eller bruke spesifikke teknikker. Å kunne se på kun visse prosesser, og samtidig utelukke andre, vil kunne gi et mer balansert datasett, og samtidig åpne for å ha spesialtilpassede modeller rettet mot visse scenarier. Programpakken må derfor ha muligheten til å enkelt redusere treningsettet ned til noen bestemte prosesser - som f.eks cmd.exe og powershell.exe - og samtidig ignorere prosesser som er å anse som trygge, men som genererer store mengder data i EDR-systemet.

5.2.3 Visualisering av data for analyseformål

For at en klassifikasjon av data skal kunne være nyttig for analyseformål, må hva modellen sier er godsinnet eller ondsinnet kunne presenteres på en ryddig måte. Dette vil inkludere informasjon om presisjon, recall og falske/sanne positive/negative, samt å plotte datapunktene i et grafisk grensesnitt. Datapunktene i dette grensesnittet må også være interaktive i den forstand at en analytiker skal kunne velge et spesifikt datapunkt og hente ut informasjon om dette. Denne informasjonen skal være relevant for hvorfor et datapunkt har blitt klassifisert. Ut ifra denne informasjonen skal også klassifikasjoner en feedback-modell er usikker på kunne gis tilbakemeldinger på.

5.2.4 Modularitet

I dette prosjektet har maskinlæringsrammeverket SciKit Learn (sklearn) blitt benyttet til maskinlæring. Det eksisterer en rekke andre rammeverk for maskinlæring, og det er derfor viktig at programpakken ikke hardkodes opp mot et bestemt system. Det må derfor lages en rekke submoduler som kan benyttes uavhengig av både maskinlæringsrammeverk og hverandre. Et eksempel på dette er å høste inn og preprosessere data ved hjelp av heimdall, gjennomføre maskinlæring i et helt annet rammeverk, men likevel kunne sende denne dataen gjennom analyseverktøyene og få den inn i grensesnittet.

5.2.5 Ønsket funksjonalitet oppsummert

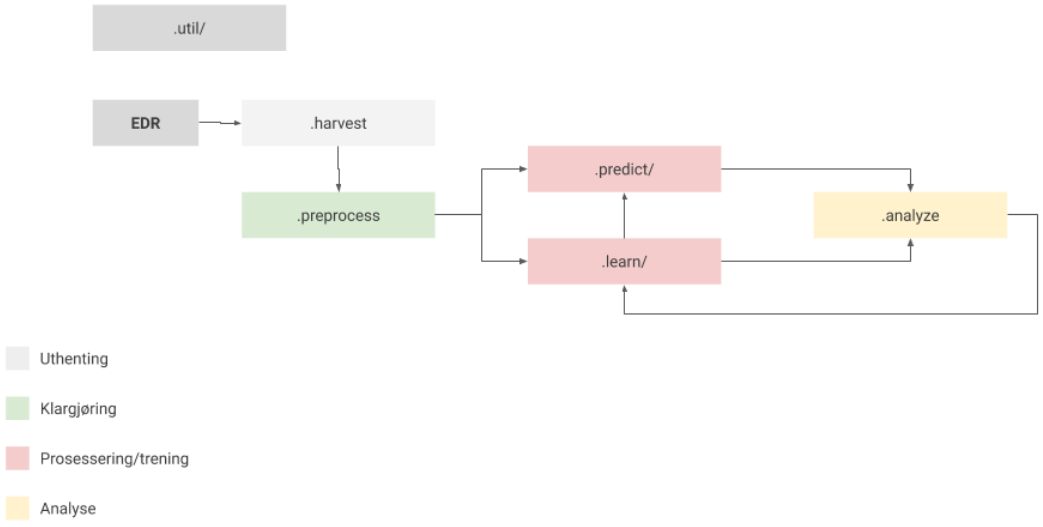
Ut ifra de overnevnte avsnitt, kommer følgende punkter fram som funksjoner TCERT vil at programpakken skal ha, og funksjoner som behøves for at gruppen skal kunne gjennomføre analyseoppgaven.

- Å effektivt kunne høste data fra EDR-systemet.
- Å kunne enkelt preprosessere data til et format håndterbart av en maskinlæringsmodell.
- Støtte for novelty/unsupervised trening.
- Støtte for online/feedback trening.
- Å enkelt kunne gi tilbakemeldinger til en online-modell.
- Muligheten for å lagre og laste inn trente modeller fra disk.
- Å enkelt kunne begrense datasettet ned til utvalgte prosesser, samt å kunne ignorere overflødige prosesser.
- Å kunne visualisere klassifiserings-resultater og å hente informasjon om én enkelt hendelse og dens klassifikasjon.
- Å kunne benytte seg av innhøsting, preprosessering og analyse uavhengig av maskinlæringsrammeverk.

6 Modellering

6.1 Heimdall-pakken

For å kunne gjennomføre maskinlæring på dataen fra EDR-systemet til Telenor Norge må en rekke verktøy utvikles. Disse verktøyene skal interface med EDR-systemet, hente data ut fra det, preprosessere denne daten, gjennomføre maskinlæring på den, ut ifra denne læringen forutsi om en hendelse er ondsinnet eller ikke, og til sist analysere effektiviteten av den valgte maskinlæringsmodellen. Navnet Heimdall kommer fra norrøn mytologi. Heimdall er guden hvis oppgave det er å våke over Bifrost, bruen inn til Åsgard, og blåse i Gjallarhorn om det var fare på ferde. Det ble sett til norrøn mytologi som et nikk i Telenor Norges retning, da de er en av de største aktørene i norden, og navnet Heimdall passer tematikken i oppgaven godt.

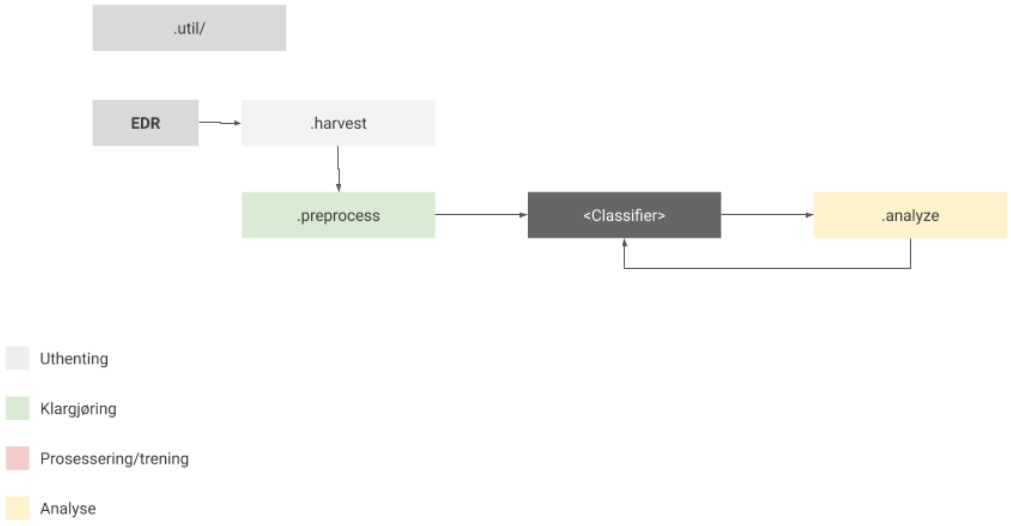


Figur 6.1: Initiell oversikt over pakkeflyt, samt sub-modulers navn og arbeidsoppgave.

Verktøyene skapt for prosjektet har blitt kategorisert i henhold til en flyt gruppen skapte tidlig i prosessen. Ved å organisere funksjonalitet inn i kategorier og separere

disse funksjonene ut i filer, skulle det skapes en ryddig pakke, med nyttige og spesifiserte sub-moduler. Figur 6.1 ble skapt i samarbeid med Telenor Norge, hvor navnekonvensjoner og intensjonen ved de ulike sub-modulene er nevnt.

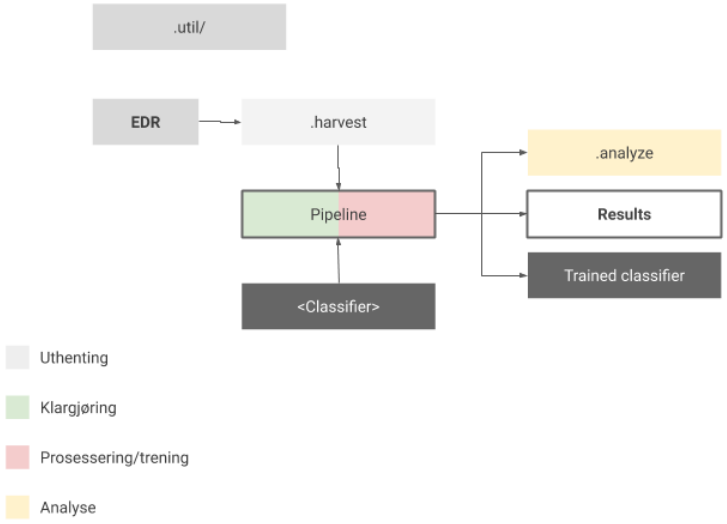
Tanken var å hente ut informasjon fra EDR-systemet med en harvest-modul, for så å klargjøre informasjonen for maskinlæring med en preprosess-modul. Den klargjorte informasjonen kunne så gå to veier, enten brukes til å trene nye modeller med learn-modulen, eller til nye forutsigelser med predict-modulen. Til slutt kan en bruker analysere resultatene, og derifra enten trekke konklusjoner eller endre innstillinger for trening av modellen. Denne flyten viste seg å ha sine feil, da det er ønskelig å kunne bruke en hvilken som helst modell, ikke bare de modellene definert i learn og predict. I praksis ble derfor flyten underveis i prosjektet tilnærmet det vist i figur 6.2.



Figur 6.2: Adoptert arbeidsflyt internt i gruppen for store deler av prosjektet.

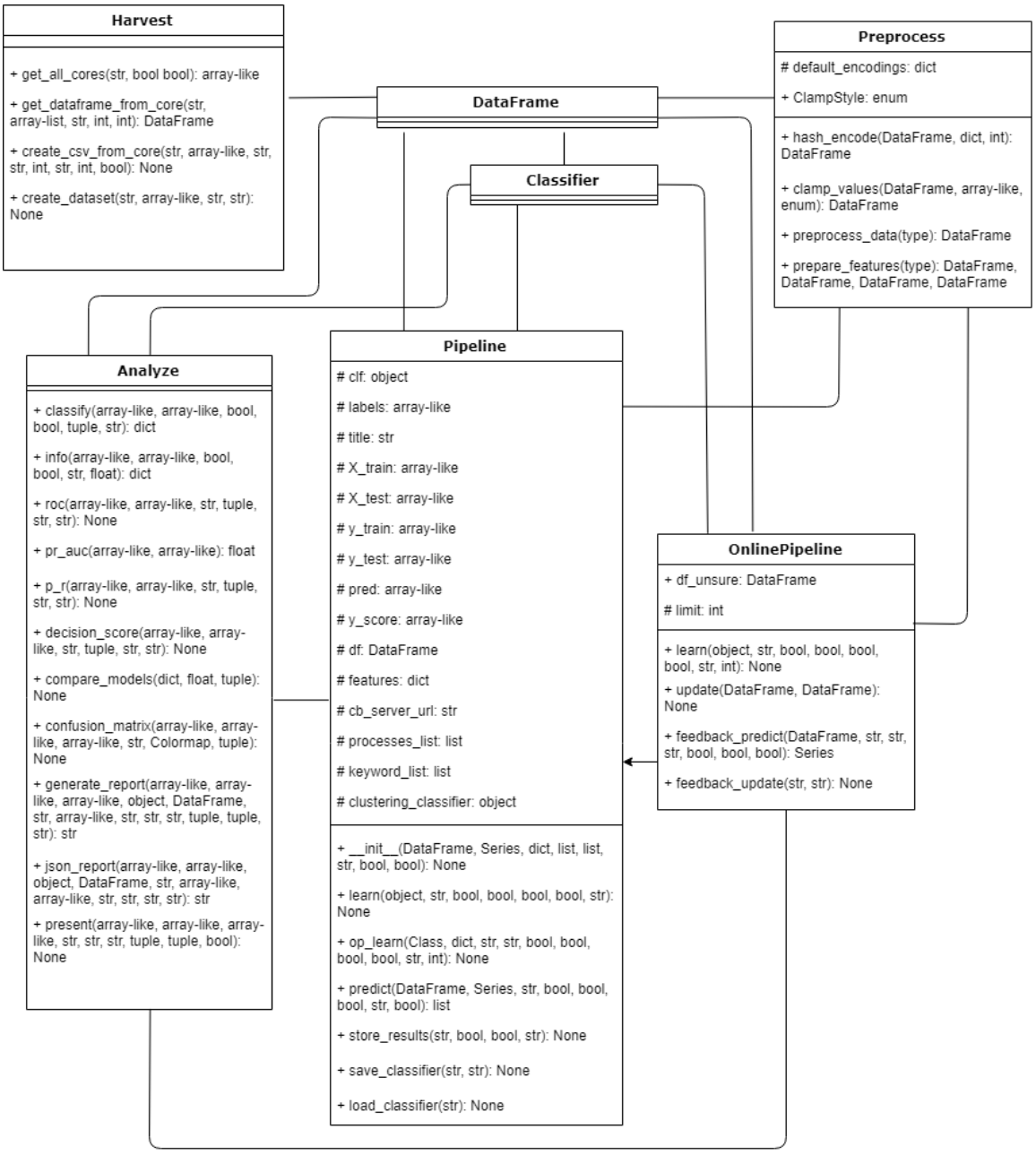
Denne flyten gjorde at prosesserings- og analyseringsverktøyene fikk generaliserte funksjoner, men det ble lite standard for hvordan funksjonskallene i en gitt test ble utført. Mangelen på standard førte til varierende løsninger på formatering og en generell mangel på brukervennlighet. Implementasjoner av online trening førte også til at feature-sett må være konsekvente, som ga flere variabler å holde kontroll på. Uten en overordnet kontroll på variablene, krever arbeidsflyten at en bruker holder kontroll på samtlige variabler, hvor ikke alle har muligheter for å endres. Dette fører til at testing av diverse klassifiseringer kan bli problematisk, da ulike klassifiseringer kan ha ulike krav til trenings-/testsett. Derfor ble det besluttet å generere en klasse for å standardisere fremgangsmåte, samt øke brukervennligheten av

systemet. Klassen standardiserer fremgangsmåten for prosessering av informasjon, samt gjør det mulig å gjøre operasjoner på en vilkårlig maskinlæringsmodell.



Figur 6.3: Arbeidsflyt med klasse for samling av moduler.

Flyten i dette nye oppsettet, vist i figur 6.3, har færre ledd og derav mindre systemer for en bruker å sette seg inn i. For å ta i bruk systemet er det også nå én enkelt import, ikke fire forskjellige. For å generere og bruke en modell krever det kun å sende inn uthentet informasjon og en form for modell-implementasjon. Med denne informasjonen genereres det trenings- og testsett, og modellen trenes på dette. Det er så mulig å få presentert beregninger ut ifra resultater på testsettet, hente ut den trente modellen, eller bruke den trente modellen til klassifisering av ny data.



Figur 6.4: Klasse diagram fra heimdall pakken

7 Implementasjon

7.1 Python-moduler

Verktøyene skapt for å utføre de oppgavene ønsket av Telenor Norge, samt svare på problemstillingen, er kodet i Python. Den tenkte arbeidsflyten modellert i kapittel 6, krever en smidig utviklingsflyt med store muligheter for videreutvikling. Når Python også er et veldig vanlig språk å bruke til ML, kom dette valget naturlig.

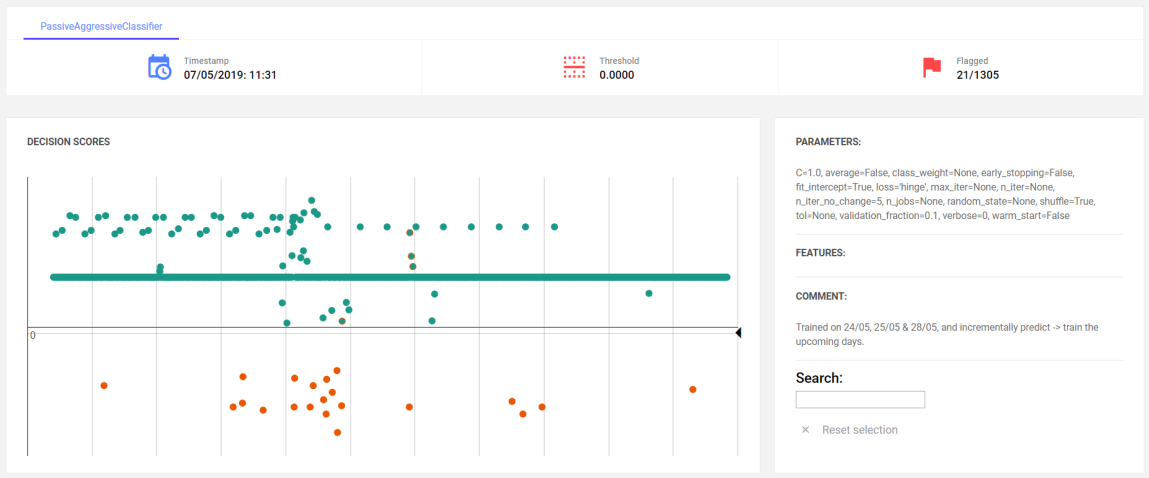
Da problemstillingen som skal besvares går ut på om det er mulig å bruke ML for EDR-systemer, faller det naturlig å implementere verktøypakken på modulær og adaptiv måte. Strukturen på pakken er vist i figur 6.4, hvor de ulike Python-modulene er koblet opp mot Pipeline-klassen, Pandas DataFrames og en vilkårlig implementasjon av klassifisering (classifier/klassifikator). Ved å nytte denne strukturen kan pakken gjenbrukes med nye klassifiseringsalgoritmer i fremtiden, om noen små forbehold taes. For å kunne bruke Pipeline-klassen, er det nødvendig at en klassifiseringsklasse (C) har implementert følgende metoder, med følgende signatur, for å nyttes på data X med label Y:

1. C.fit(X, Y(optional))
2. C.predict(X)
3. C.decision_function(X)

Dette er videre beskrevet i vedlegg A.4. Pipeline-klassen er laget for å samle modulene for preprosessering og analyse, samt standardisere hvordan en opererer med en implementasjon av en modell. Hensikten er å gjøre arbeidsflyten i verktøypakken enklere, men om det er ønskelig å nytte enkelt moduler alene er dette fullt mulig. Beskrivelser av du ulike modulene og verktøyene er beskrevet grundigere i vedlegg A.

7.2 Dashbord

Dashboardet er et verktøy som tolker JSON-filer generert av analyze.py A.3 og lar analytikere videre analysere resultatene. Dashboardet viser visuelt hvordan modellen klassifiserer de ulike eksemplene og tillater brukeren å se effekten av å endre terskelen. For et testing scenario, hvor label er gitt, vil også dette komme med i dashboardet, hvor klassen eksempelet tilhører er markert med en ring rundt noden, se figur 7.1. Fra dashboardet er det også lagt ved direktelink, til EDR-systemet, for hvert eksempel slik at det er mulig å undersøke nærmere dersom mer



Figur 7.1: Utklipp fra dashboardet.

informasjon er ønskelig. Søkefunksjonen innebygd i dashboardet er et regex-søk og gir brukeren mulighet for å søke opp spesifikke eksempler på en effektiv måte. Som beskrevet i A.4.1 online_pipeline-klassen er også active learning knyttet opp mot dashboardet, og gir brukeren mulighet for å titte på eksempler modellen er usikker på, og laste ned en fil med korreksjon for å oppdatere modellen.

Behovet for dashbordet oppsto etter ønske om å kunne se på fordelingen av samples på en smidig og effektiv måte. Da falske positiver og negativer er reelle og store problemer for klassifiseringsmodellene, vil et system som lar rask undersøking og prioritering av eksempler bidra til en økt nytte av systemet. I et eventuelt stadium hvor en modell skaper store deler falske positiver, kan man også raskt undersøke hvor disse usikkerhetene er, og derifra velge ved hvilket stadium en klassifisering skal taes alvorlig. Beslutningen om å bruke JSON-filer er tatt av to grunner; loggene kan enkelt nyttes i andre sammenhenger, og variabler vil beholde formatet sitt uten å omformatere til stringer. Flere visualiseringer i vedlegg G

Dashbordet er utseendemessig en tilpasning av Majestic Admin Template, med innsatt funksjonalitet for scatter-plotter, tabeller med sortering og søking og generell informasjon om modellen loggen er konstruert av.

7.3 Eksterne rammeverk & verktøy

7.3.1 Python-biblioteker

I prosjektet brukes følgende biblioteker:

7.3.1.1 Pandas

Pandas [83] er et bibliotek skrevet for håndtering og manipulering av data, og er veldig mye brukt i maskinlæring. En sentral datastruktur i Pandas er dataframes, en dictionary-aktig liste med høy ytelse ment for stordata-håndtering. Pandas brukes i

Bibliotek	Versjonsnummer
Pandas	0.23.0
Numpy	1.14.3
Xxhash	1.3.0
Joblib	0.13.2
Matplotlib	2.2.2
Scikit Learn	0.20.3

Tabell 7.1: Biblioteker og versjonsnummer

dette prosjektet til å lese inn og prosessere store mengder data fra Telenor Norges EDR-system.

7.3.1.2 Numpy

Numpy [84] er et mattebibliotek som flere biblioteker i dette prosjektet er avhengige av. Flere av datastrukturene i bl.a. Pandas er bygd på toppen av eksisterende strukturer i Numpy. I dette prosjektet benyttes også Numpy til logaritmeregning m.m.

7.3.1.3 Xxhash

Xxhash [85] brukes for å generere hashkoder på en mer stabil måte enn de innebygde hashing-funksjonene i Python. Disse hashkodene brukes i dette prosjektet til preprosessering av kategorisk data. Denne hashen gir ut et 64-bits tall.

7.3.1.4 Joblib

Joblib [86] inneholder funksjoner som tillater vedvaring av objekter fra minne ved å skrive disse objektene til disk. I dette prosjektet benyttes denne funksjonen av joblib til å lagre og laste inn ferdigtrente maskinlæringsmodeller.

7.3.1.5 Matplotlib

Matplotlib [87] er et grafisk bibliotek myntet på komputasjon og matematikk. Matplotlib brukes til å visualisere data og resultater fra maskinlæringsmodeller.

7.3.1.6 Scikit Learn

Scikit Learn [88] er et open-source maskinlæringsbibliotek som inneholder algoritmer for regresjon, klassifisering og clustering av data. Scikit brukes i dette prosjektet til trening av maskinlæring og klassifisering av data, samt clustering av data.

7.3.2 Øvrige verktøy

7.3.2.1 Atomic Red Team

Atomic Red Team [89] er et rammeverk for testing av sikkerheten i et system, ved å simulere angrep på systemet. Disse angrepene utføres ut ifra kategorier i Mitre's attack-matrise. Dette verktøyet vil tillate gruppen å utvide datasettet med flere eksempler på ondsinnet adferd.

7.3.2.2 Sphinx

Sphinx [90] er et verktøy som genererer teknisk dokumentasjon fra docstringer i Python, og vil brukes til nettopp dette formålet.

7.3.2.3 Jupyter Notebook

Jupyter Notebook [91] er et utviklingsmiljø så vel som programmeringsformat for Python. En notebook tillater asynkron eksekvering av kode separert i celler, samtidig som den bevarer deklarerte variabler og objekter i Python-kernelen den kjører på. Dette tillater at man f.eks. kan importere et datasett i én celle, men slipper å gjenta operasjonen om man må endre kode lengre ut i eksekveringssekvensen. Selve programpakken Heimdall er skrevet uavhengig av utviklingsmiljø (i dette tilfellet har samtlige gruppemedlemmer brukt Atom), men analysearbeidet som gjennomføres med programpakken gjøres i Jupyter Notebook nettopp på grunn av de overnevnte egenskapene.

8 Analyse

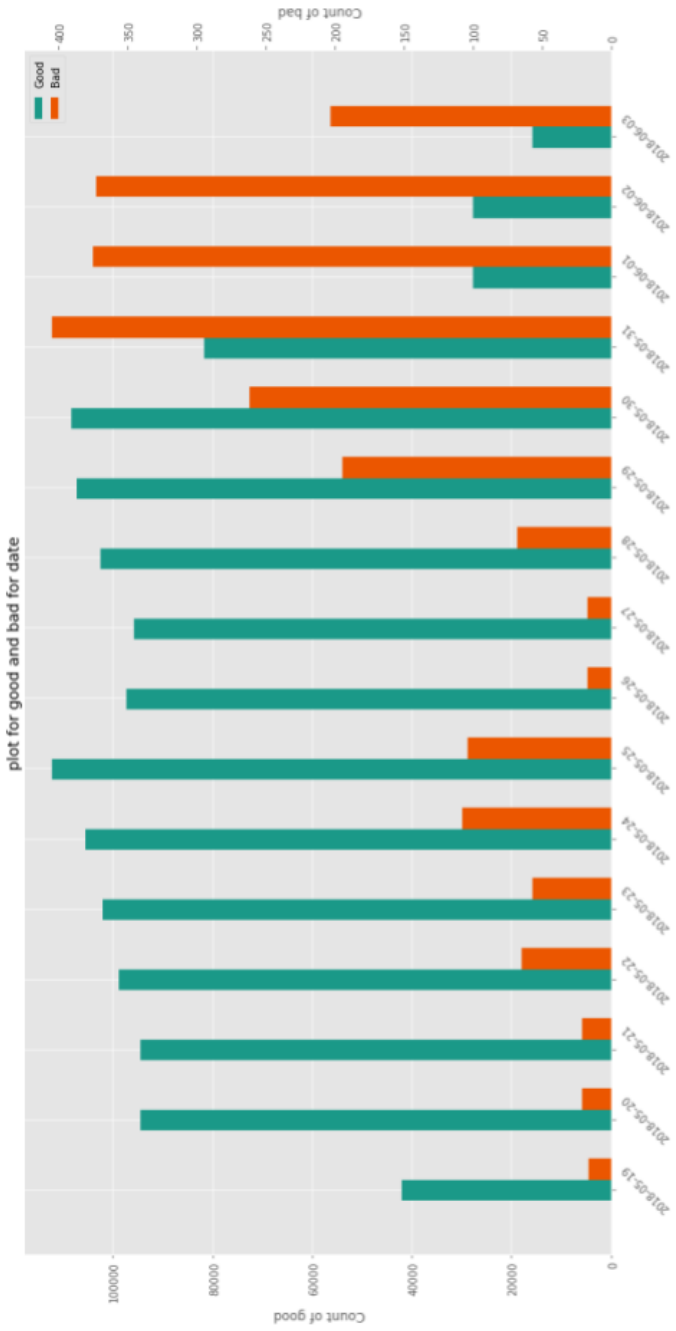
8.1 Utgangspunkt og datasett

I mai 2018 fant "Øvelse Bukkesprang" sted. Dette er en stor sikkerhetsøvelse organisert av TCERT, der 40 deltakere fra forskjellige norske sikkerhetsavdelinger over tre dager testet forsvarsevnene til en fiktiv bedrift. Denne bedriften presenterte et simulert miljø med egen verdikjede, datasystemer og brukere som aktivt interagerte med disse systemene. Et "Red Team" simulerte adferden til flere store trusselaktører fra hele verden, og gjennomførte angrep mot bedriften gjennom øvelsen. Samtlige maskiner i bedriften hadde EDR-sensorer installert på seg, og det er dataen fra disse sensorene som er basis for denne oppgaven.

Dette datasettet inneholder derfor både godsinnede og ondsinnede hendelser fra dagene øvelsen fant sted. I starten av prosjektet ble dataen lablet (forhåndsklassifisert) ved å knytte dataen opp mot hvorvidt den hadde aktivert en alarm i EDR-systemet eller ikke. Ut av 2 063 123 hendelser, ble kun 88 av dem klassifisert som ondsinnede med denne metoden, ca. 0.004% ondsinnet data. Men som diskutert i seksjon 4.6, har flere maskinlæringsmodeller et behov for et balansert datasett for å forhindre at den får et bias mot majoritetsklassen. Forholdet mellom godsinnet og ondsinnet data måtte derfor bedres.

8.1.1 Atomic Red

For å forsøke å øke mengden ondsinnet data, installerte gruppen sensorer på sine egne datamaskiner for så å kjøre simulerte angrep på maskinene ved hjelp av verktøyet Atomic Red (se seksjon 7.3.2.1). Men ved å generere data på denne måten, går mye av konteksten fra "organisk" registrert data tapt. Om man f.eks. kjører mange angrep på én enkelt klient, vil maskinlæringsmodellene oppdage et mønster fra den nevnte klienten, og klassifisere alle hendelser fra den som ondsinnet. Det ble derfor besluttet at denne metoden ville gjøre mer skade enn godt for å løse balanseringsproblemene i datasettet og ble derfor skrinlagt.



Figur 8.1: Beskrivelse av antall markerte ondsinnede hendelser per dag i det totale datasettet

8.1.2 Utvidet dataklassifisering

I samtaler med TCERT kom det fram at de 88 alarmene ikke var representativt for den ondsinnede adferden. For det første hadde flere av angrepene som ble gjennomført ikke generert en alarm i det hele tatt. Videre genererer EDR-systemet primært alarmer på prosessen som eksekverer skadevaren, og ikke de øvrige prosessene i prosesstreet. Det ble derfor gjort en gjennomgang av angrepsloggene ført av Red Team under øvelsen, og hendelser ble sammenliknet opp mot denne loggen for å korrekt klassifiseres. Forelder- og barneprosessene til en skadevares prosess ble også klassifisert som ondsinnet, da disse prosessene er indikativt på mistenkelig adferd.

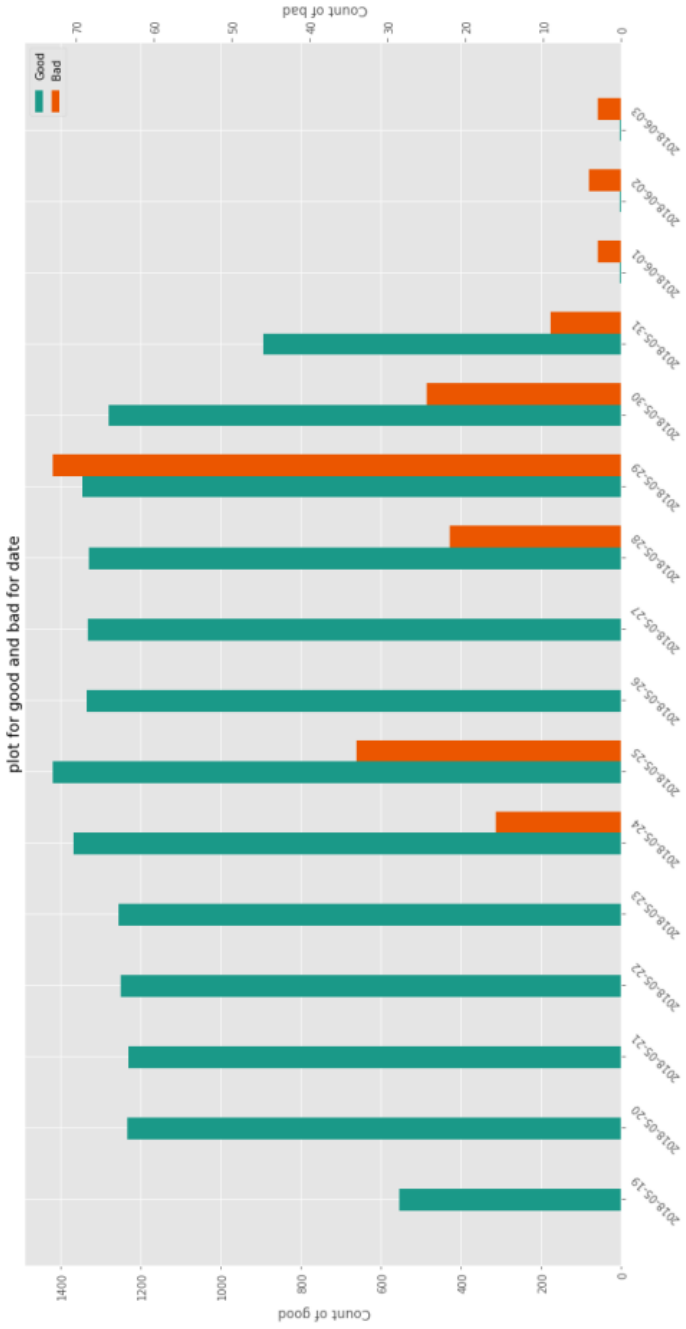
Datasettet gruppen opererte med inneholdte også logger fra fem dager, men siden øvelsen kun fant sted under tre dager, ble de overflødige to dagene fjernet fra datasettet. Den endelige fordelingen på datasettet ble etter dette 1 313 194 godsinnede og 2 307 ondsinnede, 0.17% ondsinnet data. En forbedring på 4250%, men fortsatt en uheldig fordeling.

8.1.3 Begrensning av prosesser

TCERT hadde fra starten av prosjektet lagt fram et ønske om å se hvordan maskinlæring gjorde det på kun kommandolinjeprosesser, Command Prompt og Powershell konkret. Samtidig kom det fram at prosesser som genererer store mengder data, men ikke vil inneholde ondsinnet adferd, kunne fjernes fra datasettet. Når datasettet ble begrenset ned til Command Prompt og Powershell, og lite relevante, støyende prosesser ble fjernet, var fordelingen i datasettet 15 837 godsinnede- og 189 ondsinnede hendelser, ca. 1.2% ondsinnet data.

8.1.4 Implikasjoner som følge av datasettet

Raten på 1.2% ondsinnet data er en økning på 30 000% fra de opprinnelige 0.004%. Det er en noe lav fordeling, men den lar seg jobbe med. Dog vil denne fordelingen ha noen implikasjoner for resultatene. Først og fremst vil algoritmer som benytter seg av supervised læring kunne bli veldig tilpasset dataen den blir trent på. For unsupervised læring vil dette ikke nødvendigvis være tilfellet, da den kan bli vist data uten labels, og selv forsøke å finne hvilke hendelser som skiller seg fra normalen.



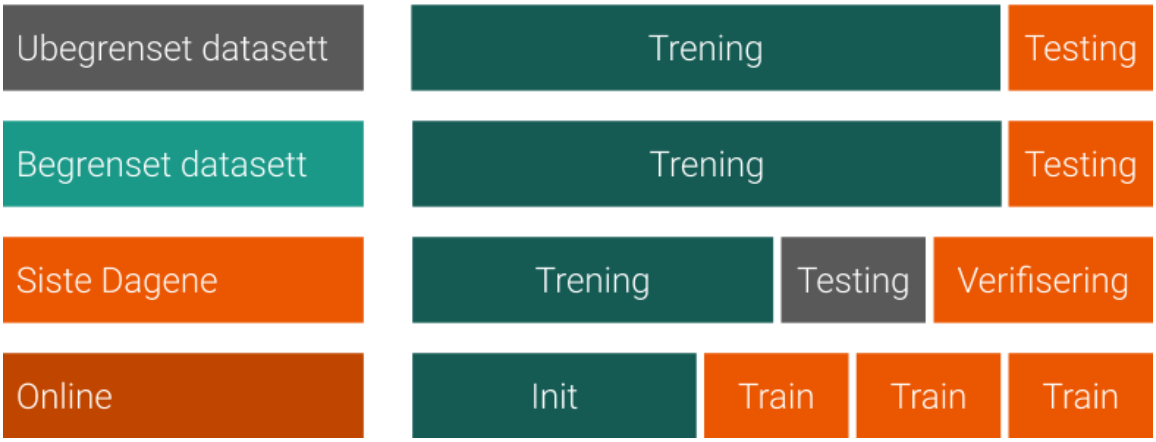
Figur 8.2: Beskrivelse av antall markerte ondsinnede hendelser per dag i det begrensede datasettet.

Et annet forbehold som må tas er at store deler av datasettet er lablet manuelt ut ifra angrepsloggene til Red Team. Data er også lablet ved hjelp av det grafiske grensesnittet som ble utviklet, da hendelser som ble klassifisert som ondsinnede men var lablet som godsinnede (falsk positiv), etter dypere analyse viste seg å faktisk være ondsinnede. Med andre ord bør det antas noen av hendelsene i datasettet kan være feilklassifisert.

8.2 Dimension reduction og EDA

Gruppen gjennomførte tidlig en EDA for å skaffe seg en oversikt over datasettet, og de ulike feature-ene. Ut ifra analysen fra den gjennomførte analysen, og i samråd med ekstern veileder kom gruppen frem til hvilke features som kunne egne seg til maskinlæring. I vedlegg D EDA and dimensionality reduction er det beskrevet gruppens observasjoner fra EDA.

8.3 Resultatgenerering



Figur 8.3: Fordeling av data i de ulike settende.

Resultater er i hovedsak generert med en fordeling av 75% data til trening og 25% til testing. Det er på de fleste settene ikke benyttet et verifiseringsett, da prioriteten har vært å nytte mest mulig data til trening. Resultatene på test settet brukes for å bedømme resultatene. Klassifiseringes modellene som er nyttet for resultatene har først blitt hyperparameter optimalisert med grid search for å kunne skape best mulig resultater. Det er viktig å presisere at denne optimaliseringen er gjort ved det begrensede datasettet, da denne metoden krever store ressurser og tid.

8.3.1 Cluster som feature

Figur 8.4: K-means clustering på begrensa datasett med tre clustere. Kryss representerer centroids.

Ved å sende inn et cluster som en feature på et sample, vil klassifiseringsmodellen i teorien ha tilgang på en eksisterende gruppering. Om denne grupperingen gir ekstra informasjon, vil det være mulig å skape nye assosiasjoner og koblinger, som igjen kan skape bedre resultater. Dette kommer på bekostning av kraften og tiden som brukes på å generere clusterene for samplesettet. I figur 8.4 kan man se hvordan datapunktene samler seg og blir gruppert. Denne representasjonen er generert ved å redusere dimensjonaliteten i featurespacet til 2D, slik at det er mulig å lage en enkel plot av data. Det skal bemerkes at i eksempelet over har all ondsinnet data havnet i cluster 1, sammen med en stor del godsinnet data.

DBSCAN er en annen clustering algoritme, som også be undersøkt til samme formål. Problemet med DBSCAN er at den for det første krever mer kraft enn K-means. Den krever også mer tid for å utføre clusteringen, da det ikke er forhåndsbestemt antall clustere algoritmen skal nytte. Den siste, og kanskje viktiste grunnen til at DBSCAN ikke er brukt som en feature, er at clusterene tilpasser seg konteksten algoritmen kjøres på. Hvis DBSCAN skulle vært nyttet, ville det vært nye antall og fordelinger av clustre per kjøring, som ville ført til at det ikke vil være noen sammenheng mellom clusterene for trening og clusterene for forutsigelse.

8.4 Resultater og analyse av algoritmer og modeller

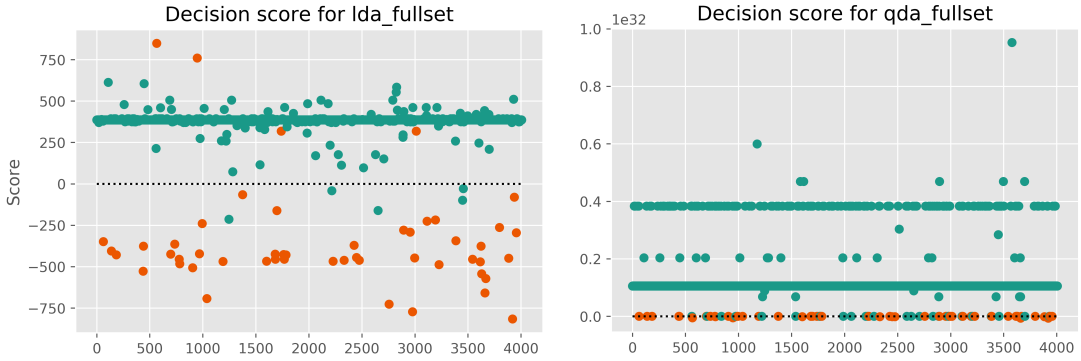
Det er igjennom oppgaven generert store deler resultater, men da disse resultatene er generert på grunnlag av data fra en øvelse holdt av Telenor Norge, vil nøyaktige resultater og statistikk forekomme i vedlegg F. I seksjonen under vil det være en oppsummering av modellenes ferdighet med eventuelle visualiseringer godkjent av Telenor Norge.

8.4.1 Offline modeller

Resultater for offline modeller er basert på ubegrenset og begrenset datasett, i en kombinasjon med ulike feature-sett. Dette er for å kunne se hvilken betydning de ulike feature-ene har for modellene på de ulike datasettene. Videre i denne underseksjonen gies det en kort oppsummering av prestasjonene til de ulike modellene, hvor det i vedlegg F utredes mer om resultatene for de ulike kjøringene.

Linear Discriminant Analysis

LDA tilpasser seg generelt godt til trenings-/testsett, og har gode resultater uavhengig av feature- og datasett. Opp igjennom er LDA den modellen som har gjort det best statistisk sett med de labels som har vært tilgjengelig, selv om det i datasettet er klassifiseringer som er feilaktig klassifisert som godsinnet. Dette tyder på at LDA kan tilpasse seg en godt lablet kontekst, men vil ikke nødvendigvis kunne korrekt klassifisere ny og tidligere usett data.



Figur 8.5: Sammenligning av fordeling og skala av decision score, for LDA og QDA

Quadratic Discriminant Analysis

QDA presterer generelt likt som LDA, men har generelt en større andel falske positiver enn nevnte. I flere tilfeller viser disse falske positivene seg å være faktiske positiver, kun med feil label, noe som taler i favør for å kunne bruke QDA i en semi-supervised setting. Man kan også se at QDA skiller seg godt fra LDA når det kommer til spredningen av decision score, da modellen får en helt annen skala på sine forutsigelser, se figur 8.5.

Isolation Forest

IsoF skaper en god spredning på samples den blir vist, men presterer ofte ikke over 50% presisjon med recall 100%. Modellen har ofte oppdaget samples som har vært feillablet i datasettet opp igjennom. Dette kommer nok i stor grad av at modellen ikke nytter lables for å foreta sin klassifisering, men heller beregner avvik. Om en vurderer grupperingene til IsoF, og ser bort ifra mengden falske positiver, kan ofte resultatene brukes for videre analyse.

Local Outlier Factor

LOF er den modellen som har skapt de generelt dårligeste resultatene. Modellen skaper forutsigelser på en meget stor skala, hvor fordelingen av positiv og negativ klasse har liten spredning. Dette fører til at modellen ligger på toppen med tanke på falske positiver, hvor det sjeldent viser seg at disse positivene er feilaktig labelet.

One-class-SVM

OC-SVM har blitt brukt med tre forskjellige kernels: linear, poly og rbf. Av de tre har den linære kjernen prestert best, men heller ikke denne har gjort det veldig godt. OC-SVM skaper en spredning hvor det er enklere å få en høy presisjon enn det er å få en høy recall-verdi. Modellene vil derfor ha problemer med å yte innen kravene definert innen denne oppgaven, men det må nevnes at også OC-SVM modellene har funnet faktiske positiver innen falske positiver.

8.4.2 Online algoritmer

For å utnytte egenskapene til online algoritmene har modellen blitt trent på begynnelsen av øvelsen Bukkesprang, hvor de videre forutser den kommende dagen, og deretter oppdateres med informasjon fra samme dag. Ved hjelp av grid search med prioritet på ulike beregninger (recall, F1 score, og arealet under PR-kurven) ble det generert resultater ut ifra disse modellene, i tillegg til en modell med default parameterne til sklearn. Dette var for å kunne se om det ble en merkbar forskjell. Grid search ga ikke kjempestore endringer, men litt varierende resultater genererte modellene likevel på de ulike scenariene..

Stochastic Gradient Descent

Generelt sett har SGD vist seg å ha god spredning mellom fordelingen av positiv og negativ klasse, og har fungert godt. Siden modellene er lineære og tilpasser seg det den blir vist, kommer dette ofte fram av resultatene som er generert. På den andre siden har SGD vist evnen til å tilpasse seg relativt raskt til ny informasjon.

Passive Aggressive Algorithm

For PAA har kontekst vist seg å være en viktig faktor da den hele tiden sørger for at siste eksempelet vist skal klassifiseres korrekt, med mulighet for å håndtere støyende eksempler ved å minske den maksimale steglengden C. I likhet med SGD har PAA også vist seg fra en god side, og har hatt gode resultater fra øvelsen.

8.4.2.1 Active learning

De to online algoritmene introdusert i denne rapporten er også testet i form av feedback, og her er det større varierende resultater. Dette avhenger av hvor sensitiv modellen er satt til å være, og hvordan de ulike modellene fordeler eksemplene i de ulike scenariene. Et gjennomgående tema for active learning er å velge en god query selection strategy. I denne rapporten er det brukt uncertainty sampling, hvor avstanden fra hyperplanet blir kalkulert ut i fra

9 Konklusjon

Å benytte maskinlæring til å detektere skadevare og trusler i et EDR-system er gjennomførbart, men effektiviteten til en modell avhenger av datasettet og hvordan modellen er konfigurert. Denne oppgaven har demonstrert at det er mulig å skape klassifiseringer uten falske negativer med et begrenset antall falske positiver, for både supervised og unsupervised maskinlæring innen en endepunktskontekst. Modellenes resultater forbeholder i en varierende grad at treningsettet inneholder korrekt klassifisert data, som i en sikkerhetskontekst med store mengder logger ikke alltid vil være en mulighet. Ved å nytte modeller som i større grad er uavhengig av forhåndsbestemte klassifiseringer vil det være lettere å ta en modell i bruk, men dette medfører en større grad av falske positiver. I slike tilfeller vil det være nyttig å se på høy sikkerhetsklassifiseringer.

Videre har maskinlæring allerede vist seg i stand til å oppdage nye, ukjente angrep der mange andre løsninger ikke var i stand til det, som vist i seksjon 2.4: Relaterte arbeider. Det er her maskinlæring skiller seg ut fra disse tradisjonelle løsningene - å i større grad kunne virke proaktivt istedenfor retroaktivt; å kunne bidra til å bryte syklusen av å patche systemer først etter at et angrep har funnet sted.

Maskinlæring er et verktøy som benyttes i større grad av selskaper i alle digitale sektorer, og informasjonsikkerhet er intet unntak. Maskinlæringsegenskaper til å automatisk oppdage sammenhenger mellom datapunkter vil komme til å bli en mer og mer uvurderlig ressurs i en verden mer og mer utsatt for digitale angrep. Det er dog ikke en erstatning for eksisterende løsninger, men et supplement til beredskaps- og responsmulighetene i den globale sikkerhetssektoren. Til dette formålet kan verktøyspakken utviklet i denne oppgaven brukes.

9.1 Fremtidig arbeid

Arbeidet utført er i et meget spennende fagfelt i stor vekst, så det er her store muligheter for videreutvikling. Følgende punkter har gruppen definert som mulige videreføringer av arbeidet så langt.

9.1.1 Kodeutvidelse i Pipeline-klassen

Pipelineobjektet (ref. seksjon A.4) er først og fremst utviklet opp mot SciKit Learn's (sklearn) maskinlærings- og clustering-rammeverk. Spesifikt er den hardkodet opp mot navnekonvensjoner i sklearn's objekter, $fit()$ for læring og $predict()$ for klassifisering. Disse navnekonvensjonene er industristandarden, men det kan finnes avvik. Samtidig vil clustering-algoritmer som DBSCAN ikke fungere på grunn av dette, da den ikke har fit- og predict-metodene, men én enkelt metode: $fit\_predict()$.

Pipeline.py bør derfor utvides til å kunne generalisere funksjonskallene i objektene som sendes til de offentlige funksjonene $learn()$ og $predict()$ på linje 94 og 123, respektivt. Dette kan oppnås ved å bruke den innebygde $getattr()$ [92] funksjonen i Python. Selve eksekveringen av funksjonene i objektene må gjøres i de beskyttede metodene $\_train()$, $\_train\_novelty()$ og $\_add\_cluster()$ på linjene, 241, 247 og 256, respektivt.

9.1.2 Interface mellom dashboard og programpakken med Django

Dashbordet ble utviklet mot slutten av prosjektet, og var en utvidelse av ønsket om grafisk presentasjon av data diskutert i seksjon 5.2.3. Før dashbordet ble skrevet, ble denne dataen presentert ved matplotlib og eksporterte HTML-filer som inneholdte informasjon om falske positiver og negativer. Som diskutert i seksjon 7.3.2.3, ble analysearbeidet gjennomført i Jupyter Notebook, og den visuelle presentasjonen av data ble derfor gjort rett i en notebook. Dashbordet ble utviklet etter at behovet for å kunne interagere med hvert enkelt datapunkt oppstod, og programpakken ble utvidet til å kunne eksportere dataen på JSON-format for så å leses inn av dashbordet.

Dashbordet opererer per nå som en frittstående HTML-fil som kan åpnes i nettleseren. Et logisk neste steg ville være å utvide dette med en webserver, der resultater skrives inn i en database og hentes ut av dashbordet automatisk (til forskjell fra å selv måtte navigere til JSON-filene eksportert fra analyze-modulen). Her vil vi anbefale å benytte rammeverket Django. Django kommer med en egen standalone webserver som kjører rett i en Python-kernel med databasefunksjoner, og tillater derfor databasespørringer, og skrives i Python. Videre vil dette kunne tillate at Django-serveren og maskinlæringen kan kjøre på samme kernel, som gjør kommunikasjon mellom disse trivielt da de har tilgang på hverandres variabler. Videre gjør dette at man kan starte webserveren/kernelen fra hvor som helst på maskinen, på samme måte som man nå kan åpne dashbordet fra hvor som helst på maskinen. Dette vil opprettholde brukervennligheten når man kjører analyse og klassifisering lokalt (som beskrevet i on-site scenariet i seksjon 5.2.1.1).

9.1.3 Trening og klassifisering av data i grafisk grensesnitt

En videreutvikling av det overnevnte punktet, så vel som en videreutvikling av tanken rundt hele Heimdall-pakken, er å kunne bevege arbeidet vekk fra å jobbe programmatisk i Python og å implementere samtlige funksjoner (og foreslåtte

utvidelser) i Heimdall i et grafisk grensesnitt. Dette arbeidet er utenfor scopet i oppgaveteksten vi har jobbet ut ifra, men vi foreslår at dette kunne være en bacheloroppgave for senere dataingeniørkull.

9.1.4 Deep learning

Deep learning er en underkategori av den bredere maskinlæringsfamilien der modellene opererer lagvis i et kunstig nevralt nettverk. Disse nettverkene er modellert etter de biologiske nevrale nettverkene i menneskehjernen, og øker en maskinlæringsmodells mulighet til å ta intelligente valg uten input fra en utvikler. Deep learning er den neste evolusjonen i feltet kunstig intelligens, og nestegenerasjons sikkerhetsprogramvare i både IDS, antivirus og EDR benytter seg av dette.

For eksempel i etterkant av angrepet mot Norsk Hydro der løsepengeviruset Lockergoga forårsaket store skader, ble en kopi av viruset lastet opp på Virustotal. Kun 25 av de 69 løsningene viruset ble testet på, oppdaget at det var en skadevare. Det som er interessant å merke seg, er at flere av disse løsningene benyttet seg av deep learning [93]. Deep learning er derfor et felt som bør sees på i kontekst av EDR. Kompleksiteten i disse teknikkene er dog så høy at det ikke var realistisk for oss å få testet det, da vi måtte lære oss grunnleggende maskinlæring fra bunnen av. Dette er noe vi foreslår kan gis som en masteroppgave.

10 Evaluering

10.1 Gruppens evaluering av prosjektarbeidet

Over det siste halvåret har vi jobbet med en virkelig krevende oppgave, ikke bare med tanke på oppgaveteksten, men med prosjektarbeid på en skala vi ikke tidligere har jobbet på. I dette avsnittet vil vi i gruppen komme med vårt subjektive retrospektiv på dette prosjektarbeidet og prosessen som har vært underlaget for gjennomføringen av oppgaven. Hva som har gått bra, hva som kunne vært gjort bedre og hva vi ville gjort annerledes med den kunnskapen og erfaringen vi nå har.

10.1.1 Hva vi kunne gjort bedre

- Gjennom prosjektet har det vært gjennomgående at poengsummen til en brukerhistorie har vært sprikende, ofte for lav. Dette er et tegn på at noen oppgaver undervurderes eller kunne vært delt opp i mindre historier.
- Brukerhistorier ble mindre beskrivende jo lengre ut i prosjektet vi kom. Noen ganger bare beskrevet med et stikkord, og uten å følge standardkonvensjonen til en brukerhistorie fra seksjon 3.2.2, men heller en pekepinn på hva som skal gjøres, og ikke et krav.
- Underveisarbeidet burde vært bedre dokumentert. Midlertidige resultater og koden som ledet til dem har blitt presentert for oppdragsgiver, men de har ikke blitt tatt vare på siden de ikke var en del av de endelige resultatene. Dette til tross for at de har en verdi når det kommer til å vise arbeidet som har blitt gjort, og hvordan vi har endt opp hvor vi er.
- Sprintoppstartene har ofte blitt et hastearbeid som har tatt mye tid. Ofte har brukerhistorier blitt laget under sprintoppstarten når de burde ha ligget i backlogen fra før av. Samtidig vil vi peke tilbake på lite beskrivende brukerhistorier som en faktor i dette. Mye tid har gått med på å forklare hensikten bak en brukerhistorie for at vi skal kunne vite hvilken poengsum vi ville gi den.
- Retningslinjer for bruk av Git ble ikke fulgt mot slutten av utviklingen av programpakken. Dette førte til flere problemer som kunne ha vært unngått.

10.1.2 Hva vi er fornøyde med

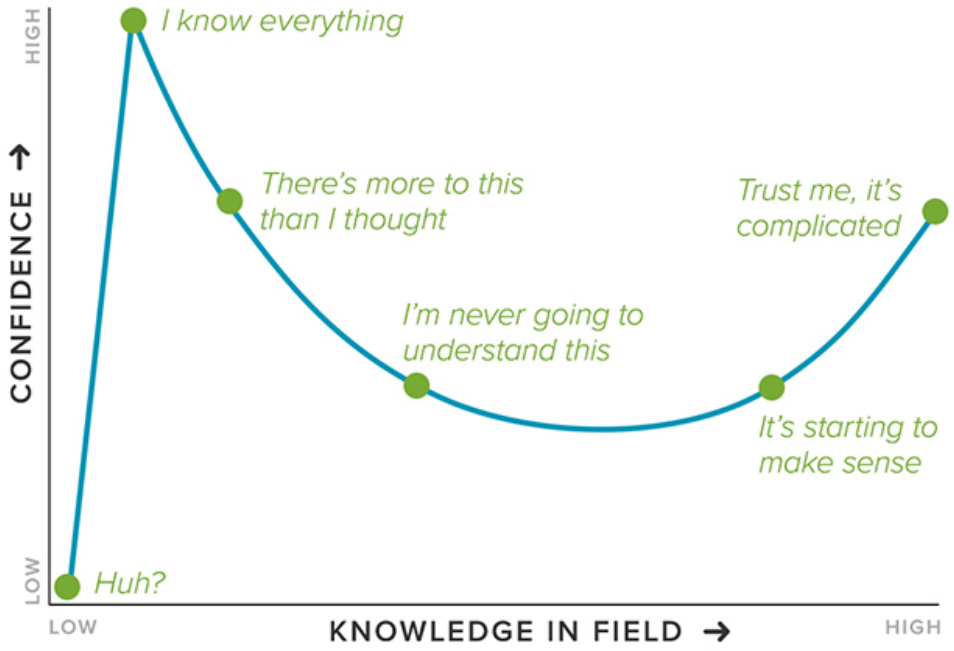
- Sprintene har i prosjektet blitt ført over én uke av gangen. Dette har tillatt oss den fleksibiliteten vi behøvde ved et prosjekt som dette.
- Vi har vært i stand til å tilpasse prosessmodellen under prosjektet til å være noe vi jobber hensiktsmessig med. Vi har gjort endringer der de trengs, og raskt tilpasset oss dem. Som f.eks. å endre hvordan risker håndteres m.m. (Ref. seksjon 3.6.4: Endringer i prosessen).
- Retrospektiver er et verktøy vi har benyttet flittig. Hver uke har det kommet tilbakemeldinger på forbedringspunkter og vedtak til endringer. Det er den delen av Scrum vi har dratt mest nytte av, særlig som studenter som ikke har jobbet på denne skalaen før, der det er fordelaktig å kunne iterere på hvordan vi jobber.
- Håndtering av bugs og utviding av funksjonalitet har blitt gjennomført på en god måte ved å opprette egne issue-typer som dekker begge deler. Feil har blitt fikset raskt, en faktor i dette er bruken av én-ukes sprinter.
- Vi har raskt gått fra idéer i samtale med veiledere internt og eksternt, og mellom oss selv, til en brukerhistorie, og til sist noe av verdi. Ting blir ikke liggende i backlogen særlig lenge før vi har noe håndfast.
- Kontakten mellom oss og oppdragsgiver har vært god. Vi har hatt en fast kommunikasjonskanal på nett, og gjennom dette fått god veiledning så vel som muligheten til å underrette om prosjektets gang.

10.1.3 Andre tanker rundt prosjektarbeidet

- Vi har gjennom prosjektet ført en uformell møtestruktur. Møtene har bestått av oss som viser fram hvor vi er i prosjektet, og får tilbakemeldinger på dette. Alle ønsker, research relaterte pekepinner og tilbakemeldinger har blitt lagt inn som en brukerhistorie i backlogen under møtet.
- Som diskutert i hva vi kunne gjort bedre, har brukerhistorier til tider vært for store. Dette er noe vi lærte oss å håndtere bedre i løpet av prosjektet. Størrelser på både brukerhistorier og backlogen for hver uke har vært en tilpasningsprosess, som vi nå har en god flyt på.
- Etter at vi snevret inn mye av arbeidet til å kun se på powershell og cmd, gikk alt mye fortere. Resultatene var bedre, og datakraften som krevdes for å håndtere dataen falt drastisk. Vi kunne snevret det inn tidligere, men igjen kom denne innsnevringen som et resultat av erfaringen vi ervervet oss i løpet av prosjektet.
- I utgangspunktet kunne vi veldig lite om problemdomenet. Derfor har vi sørget for at vi har hatt en god formidlingskultur innad i gruppen, der tavlen har blitt benyttet flittig for å dele kunnskap med hverandre. Vi har også sørget for å holde en arbeidsbalanse som tillatter alle i gruppen å jobbe med de forskjellige emnene, dette for å sørge for at vi har et universelt kunnskapsnivå.

10.2 Gruppens evaluering av det tekniske arbeidet

En oppgave av dette kaliberet har krevd mye av oss. Ingen i gruppen hadde drevet med maskinlæring tidligere, og vår erfaring med informasjonssikkerhet var kun en grunnleggende oversikt basert på ett enkelt fag i femte semester. Det primære fokuset har desidert vært på å lære oss maskinlæring, men en dypere forståelse av hvordan skadevare fungerer i kontekt av endepunktsikkerhet har også vært sentralt i å blant annet balansere datasettet. Denne læringsprosessen har vært tungt preget av Dunning-Kruger-effekten. Dunning-Kruger-effekten sier at jo mindre man vet om noe, jo enklere tror man at det er. Vice versa minsker oppfattelsen av egen evne jo mer kunnskap man får. Vi har uten tvil kjent denne effekten under prosjektet, noe som har ført til mange stressende fjes bak en dataskjerm med et par dusin nettsider åpne.



Spesielt toppet det seg rundt andre presentasjon, da det eneste vi turde å love å levere var verktøyspakken og noen retningslinjer slik at TCERT selv skulle kunne svare på oppgaveteksten. Etter dette holdte vi det vi kalte et "milepælsmøte". Dette var et større møte mellom gruppemedlemmer der veien framover ble kartlagt. Her landet vi på et "minimum viable product" i henhold til programpakken, og utviklingen av dette ble satt i gang. Det var under utvikling av dette at brikkene begynte å falle på plass: forståelsen for maskinlæring økte, familiariteten med datasettet ble dypere, og vi fikk implementert verktøy som exploratory data analysis, grid search og dimentionality reduction, noe som bidro til å hjelpe oss med valg av features og optimalisering av hyperparametre. Det store gjennombruddet kom derimot med en økt forståelse av hva som virkelig skulle lables som ondsinnet data, og ned-scopingen til kommandolinjeprosesser.

Resultater som tidligere var omtrent like nøyaktig som et myntkast førte nå til resultater som ikke kunne være tilfeldige. Samtidig hadde vi en forståelse for de forskjellige modellene som lot oss oppdage tegn på overtilpassing, og vi kunne utvikle forskjellige feature- og datasett som kunne underbygge teoriene vi hadde om modellenes resultater. Disse resultatene og deres forbehold er godt dokumentert og underbygget i både selve rapporten og vedleggene.

Til sist har vi også produsert en verktøyspakke som ikke bare kan brukes til å svare på selve oppgaven, men kan overleveres til TCERT og brukes til å både etterprøve våre resultater, og selv jobbe med maskinlæring i kontekst av EDR. Vi har laget nyttig programvare ved hjelp av godt ingeniørarbeid, og ikke minst svart på det oppgaven spør om og stiller oss bak konklusjonen i forrige kapittel.

10.3 Endringer i implementasjon underveis

10.3.1 Preprocess

I første utgave av preprosessoren, ble det brukt one-hot-encoding for å representere kategorisk data numerisk. Vi brukte da Pandas sin ”get_dummies” funksjon, som utfører one-hot-encoding for hele dataframen. Det var da viktig å gjøre preprosesseringen på hele datasettet, for deretter å dele det inn i trenings- og testsett, da å separere dem ville resultere i forskjellig struktur på treningssettet og testsettet. Dette ville uansett ha blitt et problem for helt usett data, så vi skjønte raskt etter vi kom i gang med maskinlæring for fullt, at vi trengte en annen løsning for kategorisk data. Det åpenbarte seg også at dette ville bli et problem for online og feedback trening. Vi trengte en måte å preprosessere som ville gi samme resultater på ethvert eksempel, uavhengig av hvor mange som blir sendt inn til enhver tid.

Det ble først vurdert om binær encoding ville fungere. Problemet her ville være at det fortsatt ville vært nødvendig med state, altså en oversikt over hva som tidligere er encodet. Det mest ideelle ville være noe state-løst. Vi fant så ut at hash-encoding ville fungere. Faren her er kollisjoner. Men ved å sette av nok plass, er sannsynligheten for en kollisjon relativt lav. Første utgave av hash-encoding brukte Pythons innebygde hashing-funksjon. Problemet var at denne lider av at resultatet av en hashing på strenger vil være forskjellig hver gang kernelen restartes. Dette vil gjøre at eventuelle trente modeller ikke vil fungere lenger, da den nye dataen vil få en annen hashkode på samme data, og vil derfor forvirre modellen. Som en løsning på dette, byttet vi til en ny type hashing, xxhash, som alltid produserer samme resultat.

10.3.2 Online pipeline

Første implementasjon av online pipeline tok utgangspunkt i at brukeren sendte inn en ønsket grense, som direkte sa avstanden et eksempel måtte ligge innenfor for å klassifiseres som usikker. Etterhvert kom gruppen fram til at en mer generell løsning ville være mer intuitiv for brukeren, da en direkte grense kan være veldig

liten for en modell, mens for en annen modell kan den være veldig stor. Dette kommer av at de ulike modellene plasserer eksemplene i forskjellig avstand fra hyperplanet. Dette var noe gruppen kom over da resultater skulle genereres for SGD og PAA. Fra oversikt over antall PAA og SGD var usikre på per kjøring kom det godt frem at en direkte grense med avstand til hyperplanet gjorde at SGD var usikker på så godt som ingen eksempler, mens PAA var usikker et normalt antall eksempler. Dette gjaldt for grenser mellom 1 og 3. For å tilpasse grensen ut ifra modellen som blir kjørt ble løsningen å sende inn en grensefraksjon som tilpasser seg treningssettet ved å regne differansen mellom høyeste decision score og laveste decision score og multiplisere dette med grensefraksjonen. En annen endring av implementasjon som ble sett på som hensiktsmessig var å droppe de identiske eksemplene, med hensyn på feature-settet, som ble sendt med inn i JSON-fila hvor de usikre eksemplene lagres. Fra kjøringer ble det klart at dersom modellen valgte å klassifisere mail som usikkert ble det et høyt antall identiske eksempler og dermed veldig mye jobb for en analytiker å klassifisere disse én og én.

Bibliografi

[1] *Hovedside Digital Sikkerhet*. URL: https://www.telenor.no/om/digital-sikkerhet/?fbclid=IwAR39en6aJqL9F5UizjdviKQHWlihCztbZOXDuhZAPsfF1EGP4thUXAU3UT4 (sjekket 07.02.2019) (se s. 20).

[2] *Trusselvurdering 2019*. Politiets Sikkerhetstjeneste. URL: https://www.pst.no/alle-artikler/trusselvurderinger/trusselvurdering-2019/ (sjekket 07.02.2019) (se s. 20).

[3] *Named: Endpoint Threat Detection & Response*. Anton Chuvakin. 26. jul. 2013. URL: https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/ (sjekket 11.02.2019) (se s. 21).

[4] *9 Endpoint Security Trends for 2019 and Beyond*. URL: https://ziften.com/9-endpoint-security-trends-for-2019-and-beyond/ (se s. 21).

[5] *IBM says automation is the next big step in cyber security*. URL: https://www.information-age.com/ibm-automation-cyber-security-123481699/ (se s. 22).

[6] *CylancePROTECT is the First Signature-less Next Generation Antivirus to be Certified by AV-TEST*. URL: https://threatvector.cylance.com/en_us/home/cylanceprotect-is-the-first-signature-less-next-generation-antivirus-to-be-certified-by-av-test.html (se s. 22).

[7] *The best antivirus software for Windows Client Business User - Windows 10: December 2015*. URL: https://www.av-test.org/en/antivirus/business-windows-client/windows-10/december-2015/ (se s. 22).

[8] *Cylance Market Share and Competitor Report*. URL: https://www.datanyze.com/market-share/ep/cylance-market-share (se s. 22).

[9] *Introduction to Artificial Intelligence for Security Professionals*. URL: https://www.amazon.com/Introduction-Artificial-Intelligence-Security-Professionals-ebook/dp/B07654CFFQ (se s. 22).

[10] *Deep Instinct Endpoint Protection*. URL: https://www.deepinstinct.com/endpoint-protection/ (se s. 22).

[11] *Deep Instinct Threat Prevention Evaluation*. URL: https://selabs.uk/download/enterprise/epp/2019/feb-2019-tpe-di.pdf (se s. 22).

[12] *NVIDIA Selects 5 Most Disruptive AI Startups*. URL: https://venturebeat.com/2017/04/23/nvidia-selects-5-most-disruptive-ai-startups/ (se s. 22).

[13] *Intorducing the Technology Pioneers 2017*. URL: https://widgets.weforum.org/techpioneers-2017/?sector=cyber (se s. 22).

[14] Andrii Shalaginov. "Advancing Neuro-Fuzzy Algorithm for Automated Classification in Largescale Forensic and Cybercrime Investigations: Adaptive Machine Learning for Big Data Forensic". I: (2018) (se s. 22).

[15] Mingtao Wu, Zhengyi Song og Young B. Moon. "Detecting cyber-physical attacks in CyberManufacturing systems with machine learning methods". I: *J Intell Manuf* (1. mar. 2019). URL: https://doi.org/10.1007/s10845-017-1315-5 (sjekket 13.05.2019) (se s. 23).

[16] Sabina Jeschke mfl. *Industrial Internet of Things: Cybermanufacturing Systems*. 1. jan. 2017 (se s. 23).

[17] R. Langner. "Stuxnet: Dissecting a Cyberwarfare Weapon". I: *IEEE Security and Privacy* (mai 2011) (se s. 23).

[18] Ken Schwaber. "SCRUM Development Process". I: *Business Object Design and Implementation*. Red. av Jeff Sutherland mfl. London: Springer London, 1997. ISBN: 978-3-540-76096-2 978-1-4471-0947-1. DOI: 10.1007/978-1-4471-0947-1_11. URL: http://link.springer.com/10.1007/978-1-4471-0947-1_11 (sjekket 11.02.2019) (se s. 24).

[19] Y. Sugimori mfl. "Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system". I: *International Journal of Production Research* 15.6 (jan. 1977), s. 553–564. ISSN: 0020-7543, 1366-588X. DOI: 10.1080/00207547708943149. URL: http://www.tandfonline.com/doi/abs/10.1080/00207547708943149 (sjekket 11.02.2019) (se s. 24).

[20] Jasper Alblas. *Managing risk*. 11. jul. 2018. URL: https://www.scrum.org/resources/blog/managing-risk (se s. 28).

[21] *PEP 8 – Style Guide for Python Code*. URL: https://www.python.org/dev/peps/pep-0008/ (se s. 33).

[22] *Google Style Python Docstrings*. URL: https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html (se s. 33).

[23] *Lov om nasjonal sikkerhet (sikkerhetsloven) - Kapittel 5. Informasjonssikkerhet - Lovdata*. URL: https://lovdata.no/dokument/NL/lov/2018-06-01-24/KAPITTEL_5#%5C%C2%5C%A75-6 (sjekket 12.03.2019) (se s. 35).

[24] *(PDF) The Evolution of Viruses and Worms*. URL: https://www.researchgate.net/publication/228869267_The_Evolution_of_Viruses_and_Worms (se s. 35).

[25] *Creeper & Reaper*. URL: http://corewar.co.uk/creeper.htm (se s. 35).

[26] M. Landesman. *Bootsector virus repair*. URL: https://web.archive.org/web/20110112024842/http://antivirus.about.com/od/securitytips/a/bootsectorvirus.htm (se s. 36).

[27] P. OKane, S. Sezer og K. McLaughlin. "Obfuscation: The Hidden Malware". I: *IEEE Security Privacy* (sep. 2011). URL: https://ieeexplore.ieee.org/abstract/document/5975134 (se s. 36).

[28] J. A. P. Marpaung og M. Sain and. "Survey on malware evasion techniques: State of the art and challenges". I: *2012 14th International Conference on Advanced Communication Technology (ICACT)*. Feb. 2012. URL: https://ieeexplore.ieee.org/abstract/document/6174775 (se s. 36, 37).

[29] David M Chess og Steve R White. "An Undetectable Computer Virus". I: (). URL: http://groups.csail.mit.edu/cis/crypto/classes/6.857/papers/ChessWhite.pdf (se s. 36).

[30] Heather Rosoff, Jinshu Cui og Richard S. John. "Heuristics and biases in cyber security dilemmas". I: (1. des. 2013). URL: https://doi.org/10.1007/s10669-013-9473-2 (se s. 36).

[31] Ping Chen, Lieven Desmet og Christophe Huygens. "A Study on Advanced Persistent Threats". I: 2014. URL: http://link.springer.com/10.1007/978-3-662-44885-4_5 (se s. 36).

[32] Leyla Bilge og Tudor Dumitraş. "Before We Knew It: An Empirical Study of Zero-day Attacks in the Real World". I: New York, NY, USA, 2012. URL: http://doi.acm.org/10.1145/2382196.2382284 (se s. 36).

[33] C. Greamo og A. Ghosh. "Sandboxing and Virtualization: Modern Tools for Combating Malware". I: (mar. 2011). URL: https://ieeexplore.ieee.org/abstract/document/5739643 (se s. 36).

[34] Michael Maass. "A Theory and Tools for Applying Sandboxes Eectively". I: (). URL: http://www.cs.cmu.edu/~mmaass/pdfs/dissertation.pdf (se s. 37).

[35] M. Mehra og D. Pandey. "Event triggered malware: A new challenge to sandboxing". I: des. 2015. URL: https://ieeexplore.ieee.org/abstract/document/7443327 (se s. 37).

[36] *Mechanisms to determine if software is running in a VMware virtual machine (1009458)*. URL: https://kb.vmware.com/s/article/1009458 (sjekket 13.03.2019) (se s. 37).

[37] *Malware Statistics*. URL: https://www.av-test.org/en/statistics/malware/ (se s. 37).

[38] *Named: Endpoint Threat Detection & Response*. Anton Chuvakin. 26. jul. 2013. URL: https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/ (sjekket 20.02.2019) (se s. 37).

[39] Axel Tidemann og Anne Cathrine Elster. *maskinlæring*. I: *Store norske leksikon*. 17. jan. 2019. URL: http://snl.no/maskinl%C3%A6ring (sjekket 20.02.2019) (se s. 37).

[40] Sotiris B Kotsiantis, I Zaharakis og P Pintelas. "Supervised machine learning: A review of classification techniques". I: *Emerging artificial intelligence applications in computer engineering* 160 (2007), s. 3–24 (se s. 37).

[41] Xiaojin Jerry Zhu. *Semi-supervised learning literature survey*. Tekn. rapp. University of Wisconsin-Madison Department of Computer Sciences, 2005 (se s. 38).

[42] Jason Brownlee. *What is a Confusion Matrix in Machine Learning*. Machine Learning Mastery. 17. nov. 2016. URL: https://machinelearningmastery.com/confusion-matrix-machine-learning/ (sjekket 16.03.2019) (se s. 38).

[43] Koo Ping Shung. *Accuracy, Precision, Recall or F1?* Towards Data Science. 15. mar. 2018. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (sjekket 16.03.2019) (se s. 38).

[44] Tejumade Afonja. *Accuracy Paradox*. Towards Data Science. 8. des. 2017. URL: https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b (sjekket 18.03.2019) (se s. 39).

[45] *Classification: Precision and Recall | Machine Learning Crash Course*. Google Developers. URL: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall (sjekket 17.03.2019) (se s. 39).

[46] Yutaka Sasaki mfl. "The truth of the F-measure". I: *Teach Tutor mater* 1.5 (2007), s. 1–5 (se s. 39).

[47] Kelly H Zou, A James O'Malley og Laura Mauri. "Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models". I: *Circulation* 115.5 (2007), s. 654–657 (se s. 40).

[48] Takaya Saito og Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". I: *PloS one* 10.3 (2015), e0118432 (se s. 40).

[49] Haibo He og Edwardo A Garcia. "Learning from imbalanced data". I: *IEEE Transactions on Knowledge & Data Engineering* 9 (2008), s. 1263–1284 (se s. 40).

[50] Jorge de la Calleja mfl. "Machine learning from imbalanced data sets for astronomical object classification". I: *2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR)*. 2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR). Dalian, China: IEEE, okt. 2011, s. 435–439. ISBN: 978-1-4577-1196-1 978-1-4577-1195-4 978-1-4577-1194-7. DOI: 10.1109/SoCPaR.2011.6089283. URL: http://ieeexplore.ieee.org/document/6089283/ (sjekket 19.03.2019) (se s. 41).

[51] A. Dawoud, S. Shahristani og C. Raun. "Deep Learning for Network Anomalies Detection". I: *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*. Des. 2018, s. 149–153 (se s. 41).

[52] K. Anand, J. Kumar og K. Anand. "Anomaly detection in online social network: A survey". I: *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*. Mar. 2017, s. 456–459 (se s. 41).

[53] P. M. Comar mfl. "Combining supervised and unsupervised learning for zero-day malware detection". I: *2013 Proceedings IEEE INFOCOM*. Apr. 2013, s. 2022–2030. DOI: 10.1109/INFCOM.2013.6567003 (se s. 42).

[54] Mohammad Sabokrou mfl. "Adversarially Learned One-Class Classifier for Novelty Detection". I: *CoRR* abs/1802.09088 (2018). arXiv: 1802.09088. URL: http://arxiv.org/abs/1802.09088 (se s. 42).

[55] Yixin Chen mfl. "Outlier detection with the kernelized spatial depth function". I: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2 (2009), s. 288–305 (se s. 43).

[56] R. Kumar Dwivedi, S. Pandey og R. Kumar. "A Study on Machine Learning Approaches for Outlier Detection in Wireless Sensor Network". I: *2018 8th International Conference on Cloud Computing, Data Science Engineering (Confluence)*. Jan. 2018, s. 189–192. DOI: 10.1109/CONFLUENCE.2018.8442992 (se s. 43).

[57] N. Boujemaa N. Grira M. Crucianu. *Unsupervised and Semi-supervised Clustering: a Brief Survey*. 15. aug. 2005. URL: http://cedric.cnam.fr/~crucianm/src/BriefSurveyClustering.pdf (se s. 43).

[58] C. Rong R. M. Esteves T. Hacker. *Competitive K-means*. 16. mar. 2014. URL: https://ezproxy2.usn.no:2160/stamp/stamp.jsp?tp=&arnumber=6753773&tag=1 (se s. 43).

[59] S. Li L. Zhang S. Deng. *Analysis of power consumer behavior based on the complementation of k-means and DBSCAN*. 4. jan. 2018. URL: https://ezproxy2.usn.no:2160/stamp/stamp.jsp?tp=&arnumber=8245490&tag=1 (se s. 43).

[60] S. Ertekin, L. Bottou og C. L. Giles. "Nonconvex Online Support Vector Machines". I: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.2 (feb. 2011), s. 368–381. ISSN: 0162-8828 (se s. 44).

[61] J. Wang mfl. "Online versus offline learning for spiking neural networks: A review and new strategies". I: *2010 IEEE 9th International Conference on Cyberntic Intelligent Systems*. Sep. 2010, s. 1–6. DOI: 10.1109/UKRICIS.2010.5898113 (se s. 44).

[62] Ran Wang mfl. "Fuzzy rough sets based uncertainty measuring for stream based active learning". I: *2012 International Conference on Machine Learning and Cybernetics*. Bd. 1. Jul. 2012, s. 282–288. DOI: 10.1109/ICMLC.2012.6358926 (se s. 44).

[63] Jordan Boyd-Graber. *Active Learning - Digging in to Data*. Apr. 2014 (se s. 44).

[64] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013 (se s. 45).

[65] Marco Peixeiro. *Classification (Part 2) — Linear Discriminant Analysis*. Towards Data Science. 11. des. 2018. URL: https://towardsdatascience.com/classification-part-2-linear-discriminant-analysis-ea60c45b9ee5 (sjekket 18.04.2019) (se s. 45).

[66] Corinna Cortes og Vladimir Vapnik. "Support-Vector Networks". I: *Machine Learning*. 1995, s. 273–297 (se s. 45).

[67] Hyun Joon Shin, Dong-Hwan Eom og Sung-Shick Kim. "One-class support vector machines—an application in machine fault detection and classification". I: *Computers & Industrial Engineering* 48.2 (2005), s. 395–408 (se s. 45).

[68] Fei Tony Liu, Kai Ming Ting og Zhi-Hua Zhou. "Isolation forest". I: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, s. 413–422 (se s. 46).

[69] Markus M Breunig mfl. "LOF: identifying density-based local outliers". I: *ACM sigmod record*. Bd. 29. 2. ACM. 2000, s. 93–104 (se s. 46).

[70] Tong Zhang. "Solving large scale linear prediction problems using stochastic gradient descent algorithms". I: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, s. 116 (se s. 47).

[71] D. Wang mfl. "Online Support Vector Machine Based on Convex Hull Vertices Selection". I: *IEEE Transactions on Neural Networks and Learning Systems* 24.4 (apr. 2013), s. 593–609. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2013.2238556 (se s. 47).

[72] Thomas G. Dietterich. "Ensemble Methods in Machine Learning". I: Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000. ISBN: 978-3-540-45014-6 (se s. 48).

[73] Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 5. apr. 1999. 564 s. ISBN: 978-1-55860-529-9 (se s. 48).

[74] *Data preprocessing for machine learning: options and recommendations | Solutions*. Google Cloud. URL: https://cloud.google.com/solutions/machine-learning/data-preprocessing-for-ml-with-tf-transform-pt1 (sjekket 02.05.2019) (se s. 48).

[75] Jeff Hale. *7 Data Types: A Better Way to Think about Data Types for Machine Learning*. Towards Data Science. 29. aug. 2018. URL: https://towardsdatascience.com/7-data-types-a-better-way-to-think-about-data-types-for-machine-learning-939fae99a689 (sjekket 05.05.2019) (se s. 48).

[76] Dhairya Kumar. *Introduction to Data Preprocessing in Machine Learning*. Towards Data Science. 25. des. 2018. URL: https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d (sjekket 02.05.2019) (se s. 48).

[77] Raheel Shaikh. *Choosing the right Encoding method-Label vs OneHot Encoder*. Towards Data Science. 9. nov. 2018. URL: https://towardsdatascience.com/choosing-the-right-encoding-method-label-vs-onehot-encoder-a4434493149b (sjekket 05.05.2019) (se s. 48).

[78] Jeff Hale. *Smarter Ways to Encode Categorical Data for Machine Learning*. Towards Data Science. 11. sep. 2018. URL: https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159 (sjekket 05.05.2019) (se s. 49).

[79] S. Shukla S. Yadav. *Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification*. 18. aug. 2016. URL: https://ezproxy2.usn.no:2160/stamp/stamp.jsp?tp=&arnumber=7544814&tag=1 (se s. 49).

[80] R. Joseph. *Grid Search for model tuning*. 30. des. 2018. URL: https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e (se s. 50).

[81] Tejas Nanaware mfl. "Exploratory Data Analysis Using Dimension Reduction". I: mai 2018 (se s. 50).

[82] S. Nasreen S. Khalid T. Khalil. *A survey of feature selection and feature extraction techniques in machine learning*. 9. okt. 2014. URL: https://ieeexplore.ieee.org/document/6918213 (se s. 50).

[83] *Pandas*. URL: https://pandas.pydata.org/r (se s. 62).

[84] *Intorducing the Technology Pioneers 2017*. URL: https://widgets.weforum.org/techpioneers-2017/?sector=cyber (se s. 63).

[85] *Numpy*. URL: https://www.numpy.org/ (se s. 63).

[86] *Joblib*. URL: https://joblib.readthedocs.io/en/latest/ (se s. 63).

[87] *Matplotlib*. URL: https://matplotlib.org/ (se s. 63).

[88] *Scikit Learn*. URL: https://scikit-learn.org/stable/ (se s. 63).

[89] *Atomic Red Team*. URL: https://atomicredteam.io/ (se s. 63).

[90] *Sphinx*. URL: http://www.sphinx-doc.org/en/master/ (se s. 64).

[91] *Jupyter Notebook*. URL: https://jupyter.org/ (se s. 64).

[92] *Python getattr()*. URL: https://www.programiz.com/python-programming/methods/built-in/getattr (se s. 75).

[93] *Virustotal - lockergoga*. URL: https://www.virustotal.com/en/file/c97d9bbc80b573bdeeda3812f4d00e5183493dd0d5805e2508728f65977dda15/analysis/1552999074/ (se s. 76).

A Pakkebeskrivelse

A.1 Uthenting: harvest-modulen

EDR-systemer genererer store mengder informasjon. Denne informasjonen er lagret i en database, hvor hver hendelse og informasjonen til denne hendelsen er indeksert. Når informasjonen er i databasen, er den godt strukturert, men spørringer til databasen skaper et ekstra ledd i behandlingen av informasjonen. Med flere millioner hendelser i datasettet vil et hvert ekstra ledd være en potensiell flaskehals. For å fjerne dette ekstra leddet blir informasjonen hentet ut av databasen og lagret i komma separerte filer (csv), slik at tiden på behandling av rådata kan reduseres. Det er to måter å hente informasjon ut av EDR systemet:

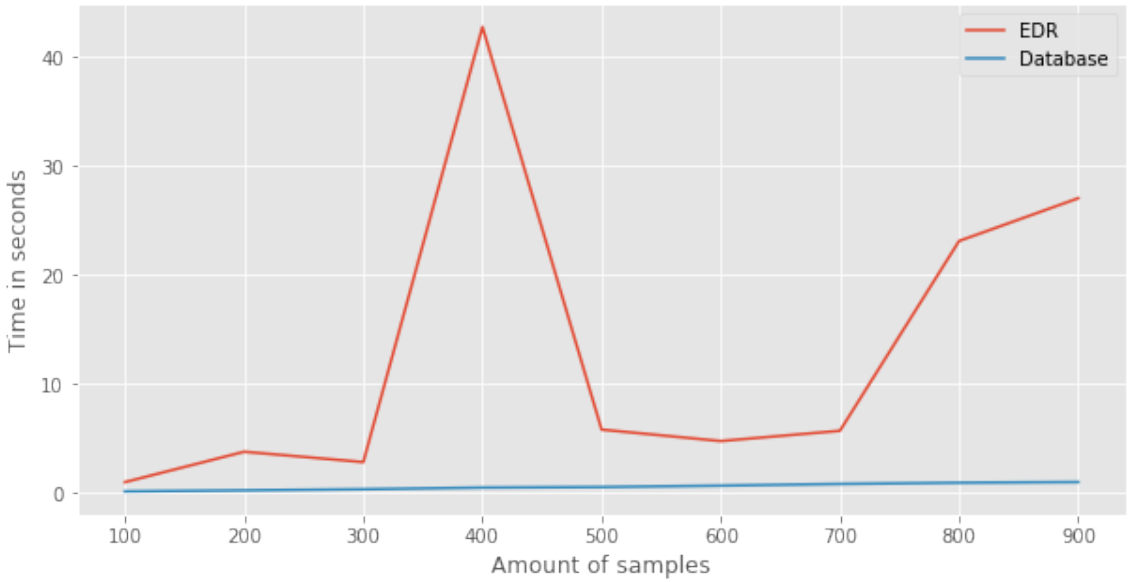
1. Hente informasjonen direkte fra databasen
2. Hente informasjonen via en API basert på REST API

Begge metodene tillater uthenting til Python-objekter og begrensning av informasjonen som hentes ut, men metodene er forskjellige når det kommer til tiden de bruker på spørringer. Ved små spørringer, 1-100 hendelser, er ikke forskjellen på metodene så store. Når man derimot øker antall spørringer bruker APIet mye lengre tid enn databasen, se figur A.1.

Beslutningen falt på å fortsette med direkte databasespørringer for å hente ut informasjonen. For å hente ut denne informasjonen velger man hvilke kolonner man vil ha, samt hvilke "shards" i databasen man ønsker å hente de ut fra. Hendelsene man da får ut blir lagret som en csv-fil på ønsket lokasjon på maskinen. Ved å bruke denne prosessen er det produsert et datasett basert på øvelsen utført av Telenor Norge.

A.2 Preprosessering: preprocess-modulen

Denne modulen blir brukt for å preprosessere dataen som kommer ut fra EDR-databasen. Dataen fra EDR-databasen vil være i numerisk eller kategorisk form. Kategorisk form vil si tekst, lister og lignende typer. Da vi behandler dataene som enten numeriske eller kategoriske typer, har preprosessoren funksjoner for å preprosessere disse to typene. Numerisk data blir normalisert slik at alle de numeriske typene ligger i det samme balanserte område. Kategoriske typer vil bli "hash encoded". En "encoding", som er en dictionary med nøklene int og category, inneholder en liste over navnene på kolonnene som tilhører denne typen. I category



Figur A.1: Plot av tidsbruk i sekunder per antall hendelser å hente ut.

sitt tilfelle, vil listen være en dictionary, hvor nøkkelen er navnet på kolonnene, og verdien er antall kolonner de forskjellige skal hash encodes til. Da ”xxhash” gir et 64-bits tall, vil det være nødvendig å benytte en annen hashing-algoritme hvis det er ønskelig med en category som krever mer enn 64 kolonner.

Prepare_features er en funksjon som automatisk gjør preprosessering og stokker datasettet, for så å dele det opp i et trenings- og testsett. Denne kan brukes om ønsket datasett kan tilfeldiggjøres, og ikke trengs å være delt opp på en spesifikk måte, da automatisering gjør omforming fra rådata til treningssett og testsett enklere.

A.3 Analyse: analyze-modulen

Analysemodulen er en rekke verktøy hvis formål det er å fastslå effektiviteten i en gitt forutsigelse, og består i hovedsak av to deler. Om ønskelig kan analysemodulen returnere beregninger som presisjon, nøyaktighet, recall samt ”confusion”-matriser, mm. Dersom det er ønskelig å se hvordan effektiviteten av modellen ved ulike terskler er, inneholder også modulen ROC-kurver og Precision-Recall-kurver. Disse konseptene er introdusert i seksjon 4.5 Analyse av trente modeller.

Den andre delen innebygd i analysemodulen er mulighet for å lagre resultatene til modellen. Dette kan gjøres ved å generere et HTML-dokument hvor man sender inn ønsket tittel på filen, informasjon om modell og resultatene til modellen, samt hva slags features som er brukt. Modulen støtter også muligheten for å generere en JSON-fil med samme informasjon som HTML-dokumentet. JSON-filen kan videre åpnes i dashboardet til Heimdall, beskrevet i 7.2 Dashbord.

A.4 Pipeline-klassen

Pipeline er en klasse som samler de sentrale verktøyene i Heimdall-pakken på ett sted. Det er også en objektorientering av maskinlæringsprosessen som automatiserer ofte gjentatte oppgaver så vel som å samle data og resultater på samme sted. Som diskutert i seksjon 6.1, var den opprinnelige hensikten å ha egne moduler for læring av modeller og klassifisering av data, henholdsvis learn- og predict-modulene. Men siden denne pakkestrukturen, vist i figur 6.1, ble ansett som lite hensiktsmessig, bevegde prosjektet seg over til å jobbe direkte med klassifikatorer som vist i figur 6.2.

Dette tillater at de forskjellige verktøyene som preprosessering, analyse, hyperparameteroptimalisering og datainnhøsting kan gjøres uavhengig av hvilket maskinlæringsrammeverk man jobber med. Det er en mer fleksibel måte å jobbe på, som er essensielt i et stadig endrende felt som informasjonssikkerhet.

Men noe som kom fram med denne arbeidsflyten var at en analytiker eller utvikler selv måtte preprosessere innhøstet data, sette opp trening- og testsett, trene modell, klassifisere data fra et eventuelt testsett, og analysere denne. Alle disse operasjonene måtte kjøres i rekkefølgen de er beskrevet, og alle behøver at forskjellige submoduler av Heimdall-pakken og øvrige maskinlæringsrammeverk importeres.

Et annet behov som ikke ble dekket av den nye arbeidsflyten var muligheten til å lagre trente modeller, og å laste dem inn igjen for kunne klassifisere ny, usett data. Det ble derfor bestemt å samle mange av verktøyene som hadde blitt utviklet i en pipeline som kunne automatisere mye av denne prosessen, og som kunne sentralisere verktøy og egenskaper i Heimdall i et objekt, som enkelt og greit ble kalt for et pipeline-objekt.

Med dette objektet trenger man kun å importere Heimdall-pakken og deklarere et Heimdall.Pipeline()-objekt. Objektets konstruktør lar det deklareres med et data- og feature-sett, som så vil automatisk preprosesseres og deles opp i trenings- og testsett. Man kan også legge ved øvrige innstillinger som hvilke prosesser det skal sees på, nøkkelord som skal ignoreres, om duplikater skal fjernes, og en ev. clustering klassifikator som så vil legge til clustere som features i treningen.

Videre har pipelinen støtte for å trene modeller, vise resultater og skrive resultatene til disk, enten som en HTML-fil eller som JSON-filene dashboardet bruker. Den har støtte for vanlige klassifikatorer, outlier klassifikatorer og novelty klassifikatorer, der den automatisk sorterer ut ”bad” data fra treningssettet. Når en klassifikator er ferdigtrent kan man lagre den til disk og laste den inn igjen for å kunne klassifisere ny data den ikke har sett før. I denne prosessen lastes også feature-sett, kolonne- og nøkkelordbegrensninger og ev. clustering klassifikator automatisk inn. Det eneste som bestemmes i dette tilfellet er stien den lagrede klassifikatoren ligger i og rådataen som den selv preprosesserer slik at dataen tilsvarer det klassifikatoren er trent på.

Til sist inneholder pipeline muligheten til å benytte seg av grid-search funksjo-

nene til Heimdall-pakken, slik at man kan finne optimale hyperparametere å trene en modell på, hente ut en kopi av det interne datasettet etter at det har redusert ned prosessene og fjernet nøkkelord, og sette innstillinger for interfacing med EDR systemet gjennom JSON-filene det genererer for dashbordet.

A.4.1 online_pipeline-klassen

Det er også utviklet en barneklasse som inneholder tilleggs-funksjonalitet for inkrementell læring. Initialiseringen av online pipeline-objektet tar imot de samme parameterne som pipeline-klassen, i tillegg til at man kan sette en ønsket grense. Denne grensen brukes til feedback trening og bestemmer selvstendigheten til modellen; med andre ord hvor sensitiv den skal være før den sender forespørsel om feedback. Grensen tar utgangspunkt i avstand fra hyperplanet som skiller de to ulike klassifiseringene, og spør om feedback dersom avstanden er innenfor grensen som er satt. Eksempler modellen er usikker på blir lagret til filer, og kan åpnes i dashboardet av en analytiker ved et passende tidspunkt for å korrigere disse. Korrigerte filer vil så kunne lastes inn i modellen, og modellen vil kunne trene på den gitte informasjonen. Dersom man skulle ønske å oppdatere modellen med ny informasjon uten å være avhenging av forespørsel fra modellen er det også mulighet for det.

Behovet for klassen ble til som et resultat av ønsket om å kunne utføre active learning ved hjelp av Heimdall-pakken, og for å kunne avgjøre en Query Selection Strategy for Stream-Based Active Learning (4.9.1 Active learning).

A.5 Øvrige verktøy: utils-modulen

Utils-modulen inneholder en rekke mindre verktøy og funksjoner som ikke er direkte tilknyttet de andre modulene. Disse rangerer fra å analysere prosesser - som for eksempel kommandolinjebruk - og generere preprosessert data fra dem, til å interface med EDR-systemet for å lage labels ut i fra hvilke alarmer EDR-systemet har generert.

A.5.1 alerts-submodulen

Utløpt

Denne modulen inneholder funksjoner for å koble alarmer generert av EDR-systemet opp mot prosesser. Modulen inneholder funksjoner for å koble alarmer opp mot sin respektive prosess, og returnerer en dataframe med alarmer og ev. informasjon om alarmene. Den er også mulighet for å lage label (good/bad) ut ifra om noe har generert en alarm eller ei.

Denne modulen var en av de første modulene som ble laget. Formålet var for å klassifisere data (gi en label). Denne modulen er nå (mer eller mindre) utløpt siden labels nå blir satt manuelt ved hjelp av eksternveilederes kunnskap om

dataen. Modulen vil fortsatt være nyttig ved bruk av andre datasett for å generere et utgangspunkt for labels.

A.5.2 binaries_to_process-submodulen

Ubrukt
Denne modulen inneholder en funksjon for å koble binærdata til prosesser og returnerer en dataframe med denne informasjonen. Denne modulen har ikke blitt brukt, men ble laget da Telenor Norge så det som nyttig å ha denne modulen. Det er likevel ikke brukt mer tid på dette, da fokuset gikk i en annen retning.

A.5.3 cmdline_args_as_features-submodulen

Utløpt
Denne modulen inneholder en funksjon som splitter kommandolinje-argumenter inn i onehot-encodede kolonner og returnerer en dataframe.

Denne modulen er utløpt, da måten man splitter kommandolinje-argumenter på ble gjort på en annen måte som var mer ønskelig å bruke.

A.5.4 crossvalidation_gridsearch-submodulen

Brukes
Denne modulen inneholder funksjoner for cross validation av datasettet og hyperparameteroptimalisering av modeller. Cross validation-funksjonen bruker sklearn sin implementasjon av ”cross_val_score” for å validere datasettet. Funksjonen returnerer scorene, sentralmål som gjennomsnitt og standardavvik, og største og minste verdi. Hvilken score som brukes kan velges selv, men for gruppens datasett vil gjerne PR AUC eller F1 score brukes som score. Grid search delen er fordelt over flere funksjoner. Funksjonene er basert på kildekoden fra sklearn sitt ”GridSearchCV”, men da denne krevde en spesifikk scores på estimator som ikke alle klassifikatorer har, kunne ikke denne brukes direkte. Grid search-funksjonen leter etter de beste hyperparameterne og den beste klassifikatoren funksjonen finner.

Det er hovedsakelig grid search som brukes, da cross validation ikke er så nyttig på online/feedback trening, men står fortsatt om det skulle være ønskelig å utføre ved offline trening.

A.5.5 dataframe_from_childproc_complete-submodulen

Ubrukt
Denne modulen inneholder en hovedfunksjon med flere mindre funksjoner for å lage en dataframe fra childproc_complete-kolonnen. Denne modulen ble laget da det ble sett et behov om å se nærmere på childproc_complete, men har ikke blitt brukt da fokuset gikk i en annen retning.

A.5.6 ensemble-submodulen

Brukes

Denne modulen inneholder funksjonaliteter for å lage en ensemble klassifikator (sammensetning av klassifikatorer). Ensemble klassifikator er laget med samme interface som sklearn klassifikatorer, med ”fit()”, ”partial_fit()”, ”predict()” og ”decision_function()”. I konstruksjonen av objektet sender man inn en dictionary, som inneholder de ulike klassifikatorene og en boolean, som representerer om denne klassifikatoren er en novelty klassifikator eller ikke, hvor True er novelty. På denne måten vil det internt bli håndtert at novelty klassifikatorer kun vil bli trent på én klasse. Konstruktøren tar også et parameter for om ensemble klassifikatoren skal bruke vekting eller stemming. Hvis vekting skal brukes, men ikke er kalibrert enda, vil stemming brukes, til kalibreringsfunksjonen blir kalt. Det er også mulig å få ut siste predictions, som sier noe om hvordan de ulike klassifikatorene gjorde det. Denne modulen ble laget for å utnytte de forskjellige klassifikatorenes egenskaper og kan brukes i pipelinen som en klassifikator.

A.5.7 multiprocess-submodulen

Brukes

Denne modulen inneholder funksjonaliteter for multithreading. Den lager et nivå av abstraksjon rundt Pythons multiprocessing-bibliotek sin ”map()” og ”starmap()” i ”Pool”. De to funksjonene i mulitprocess, som er kalt ”onemap()” og ”starmap()”(onemap bruker map(), men siden map er et reservert ord, måtte det kalles noe annet), hvor, henholdsvis, den første brukes hvis innsendt funksjon tar ett parameter. Den andre brukes der funksjonen tar flere parametre, og brukes på lik måte som funksjonen i Pool. Forskjellen er at de tar et ekstra parameter, ”n_jobs”, som beskriver hvor mange prosessorkjerner som skal brukes. Hvis 1 blir valgt, vil ikke multiprocessing biblioteket bli brukt, men heller Python sin innebygde map() funksjon.

Modulen ble laget da man så et behov for å kunne kjøre flere uavhengige prosesser på flere kjerner, og på denne måten redusere kjøringstiden. Noen klassifikatorer inneholder mulighet for multithreading (n_jobs), men ikke alle. Dette er noe de forskjellige modulene som bruker multiprocess tar hensyn til og kjører kun hvis nødvendig.

A.5.8 progress_bar-submodulen

Brukes

Denne modulen inneholder en klasse som printer en progressbar som sier noe om hvor lang tid det tar å prosessere noe. Progressbaren vises ved å stykke opp prosessen i biter, typisk i antall iterasjoner, og vises som ”-” når biten ikke fortsatt kjører og ”=” når den er ferdig kjørt. Modulen ble laget da noen prosesser bruker veldig lang tid på å kjøre og det kunne være hensiktsmessig å vite omtrentlig hvor lang tid en prosess bruker på å kjøre. Modulen brukes typisk inne i en annen modul/funksjon.

A.5.9 EDA/Dimension reduction.ipynb

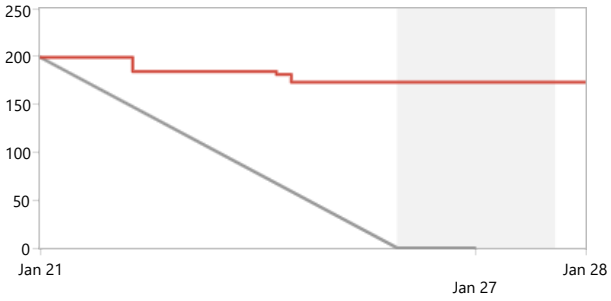
EDA-filen er en større fil med muligheter for å analysere og redusere data- og feature.settet. Her blir hele feature-settet tatt inn i utgangspunktet, men underveis blir feature-settet delt i numeriske og kategoriske features, og features med mange NaN-verdier blir fjernet. Man vil gjennom EDA kunne se på forholdet mellom good/bad, variansen, korrelasjon mm. Ved å sette ulike retningslinjer vil man kunne redusere feature-settet etter hva som er ønskelig. Siden EDA baserer seg veldig på det visuelle, som gjør det enklere for brukeren å forstå data- og feature-settet, er det derfor implementert en rekke plots for å se de ulike features-ene opp mot hverandre. Ut ifra resultatene kan brukeren avgjøre om ev. features bør fjernes. Modulen ble laget da det var nyttig å se sammenhengen mellom dataen og de ulike features-ene, samt å kunne redusere feature-settet basert på algoritmer og analyser.

B Retrospektiver og Sprintrapporter

Sprintrapport

Sprint - Uke 4

Lukket sprint, avsluttet av Martin Evandt 21/jan/19 11:34 AM - 28/jan/19 9:14 AM tilknyttet sider



Statusrapport

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (26)
BG-12	Definere testprosedyre	Historie	Medium	UTFØRT	3
BG-13	Lage et Scrum Roadmap	Historie	Medium	UTFØRT	2
BG-14	Konkretisere innholdet av første presentasjon	Historie	Medium	UTFØRT	13
BG-25	Lage en mappestruktur for git	Historie	Medium	UTFØRT	8

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (173)
BG-16	Hente data ut fra EPS	Historie	Medium	TIL UTFØRING	40
BG-17	Bygge en enkel maskinlæringsmodell	Historie	Medium	UNDER ARBEID	20
BG-18	Lage nettside for prosjektet	Historie	Medium	UNDER ARBEID	13
BG-34	Fullføre ML-kurset til Google	Historie	Highest	UNDER ARBEID	100

Retrospektiv - Uke 4

25/01/2019

Hva har gått bra?

- Færre konflikter.
- Bra balanse mellom teknisk og administrativt arbeide.

Hva kunne vi gjort bedre?

- Holde avtalte tider bedre (scrum møter, retrospektiver, etc.)
- Litt for teknisk når man forklarer felt som ikke er allmennkunnskap innad i gruppen.
- Litt mye avbryting under morgenmøtene.
- Noen user stories er for store.

Hinder

- Flere gruppemedlemmer føler de har kjørt seg fast med noen oppgaver i løpet av uken, og føler at de har kastet bort litt tid.

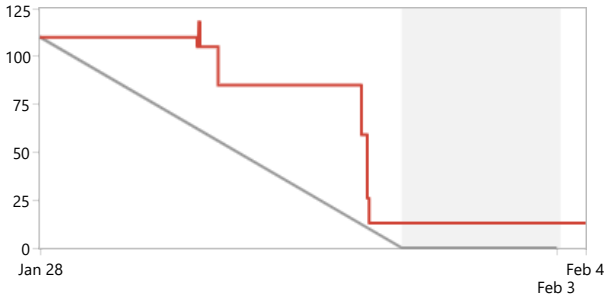
Handlinger

- Holde gruppen informert om utenforstående omstendigheter som kan påvirke oppmøte eller humør.
- Ha en felles kalender slik at alle i gruppen kan holdes oppdatert på hverandres timeplan.
- Overholde avtalte tidspunkt.
- Ordstyring under morgenmøter.
- Spørre seg selv om det man jobber med er særlig viktig, spesielt om man står fast med denne oppgaven. Kill your darlings.
- Lavere terskel for å spørre om hjelp.
- Bedre fordeling mellom administrative og tekniske oppgaver. Alle må opp på samme kunnskapsnivå både teknisk og administrativt.
- Stykke opp user stories og subtasks bedre.

Sprintrapport

Sprint - Uke 5

Lukket sprint, avsluttet av Martin Evandt 28/jan/19 10:29 AM - 04/feb/19 7:35 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (110)
BG-18	Lage nettside for prosjektet	Historie	Medium	UTFØRT	13
BG-34	Fullføre ML-kurset til Google	Historie	Highest	UTFØRT	20
BG-43	Teknisk dokumentasjon	Historie	Medium	UTFØRT	13
BG-49 *	Webapplikasjon - Django Prototyping	Historie	Medium	UTFØRT	13
BG-51	Diagrammer og retningslinjer for prototyping og programkartlegging	Historie	Medium	UTFØRT	13
BG-52	Lage retningslinjer for issues og bugs	Historie	Medium	UTFØRT	5
BG-53	Parser av csv er feilformatert	Feil	Medium	UTFØRT	-
BG-54	Skissere første presentasjon	Historie	Medium	UTFØRT	13
BG-56	Preprosessere data - prototyping	Historie	Medium	UTFØRT	20
BG-58	Forberede møte med internveileder	Oppgave	Medium	UTFØRT	-

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (13)
BG-55	Opprette en rapportmal	Oppgave	Medium	TIL UTFØRING	13

Saker fullført utenfor denne sprinten

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (40)
BG-16	Hente data ut fra EPS	Historie	Medium	UTFØRT	40

Retrospektiv - Sprint uke 5

01/02/2019

Hva har gått bra?

- Flere er i gang med utvikling og koding.

Hva kunne vi gjort bedre?

- Morgenmøtene kommer for sent i gang.
- Brukerhistoriebeskrivelsene må være tydeligere på hva man skal gjøre.

Hinder

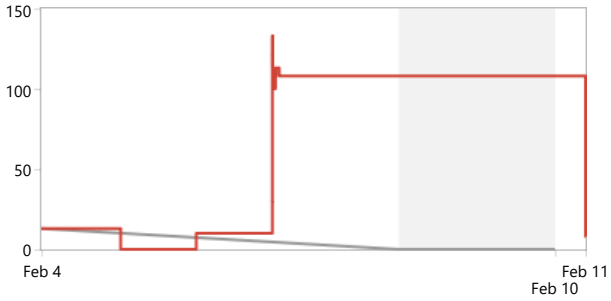
Handlinger og idèer

- Scrum master må tar mer initiativ til å starte morgenmøtene.
- Brukerhistoriebeskrivelsene må detaljeres bedre i sprintplanleggingen.

Sprintrapport

Sprint - Uke 6

Lukket sprint, avsluttet av Martin Evandt 04/feb/19 10:18 AM - 11/feb/19 9:19 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (13 → 151)
BG-55	Opprette en rapportmal	Oppgave	Medium	UTFØRT	13
BG-62	Lag en mal for godkjenning av testing og peer-review	Oppgave	Medium	UTFØRT	- → 13
BG-63	Forberede første presentasjon	Oppgave	Medium	UTFØRT	- → 20
BG-68	Ferdigstille første rapport	Oppgave	Medium	UTFØRT	- → 100
BG-79 *	Referater fra retrospektiver, med sprint info	Oppgave	Medium	UTFØRT	- → 5

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (- → 8)
BG-78 *	Automatisere pre-prosessering av data	Historie	Medium	COMPLETE	- → 8

Retrospektiv - Sprint uke 6

01/02/2019

Hva har gått bra?

- Morgenmøtene har gått etter planen

Hva kunne vi gjort bedre?

- Tempoet gikk ned mot slutten
- Man vet ikke alltid hva man skal gjøre

Spørsmål og idèer

- Kommentarfelt i testskjema
- Lage flere research-issues i backlog
- Produksjonskode (ikke prototyper) bør ut i master før det settes som done. Alt for mye ligger i dev-branch eller egen branch selv om userstorien er lukket.

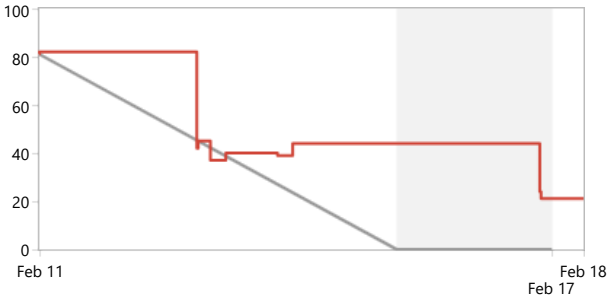
Tiltak

- Man kan jobbe med ting selv om det ikke står noe om det i backlog, frihet under ansvar

Sprintrapport

Sprint - Uke 7

Lukket sprint, avsluttet av Martin Evandt 11/feb/19 10:07 AM - 18/feb/19 9:33 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (28 → 32)
BG-78	Automatisere pre-prosessering av data	Historie	Medium	UTFØRT	8
BG-80	Første presentasjon	Oppgave	Medium	UTFØRT	20
BG-81	Dokumentasjonsrammeverk ut i produksjon	Historie	Medium	UTFØRT	- → 1
BG-83 *	Samkjøre timelister	Oppgave	Medium	UTFØRT	- → 3
BG-85 *	Preprocess endrer verdi av orginalt dataframe	Feil	Medium	UTFØRT	-

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (13 → 21)
BG-82	Kartlegge heimdall namespace	Historie	High	TIL UTFØRING	13
BG-84 *	Koble alerts opp mot prosesser	Historie	Medium	TESTING	- → 3
BG-86 *	Undersøke K-means clustering	Historie	Medium	UNDER ARBEID	- → 5

Saker fjernet fra sprint

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (40)
BG-35	Tagge data fra øvelsen	Historie	Medium	UNDER ARBEID	40

Retrospektiv - Sprint uke 7

15/02/2019

Hva har gått bra?

- Vi fikk god tilbakemelding på første presentasjon.
- Profesjonell opptreden i møte med sensorer og veiledere.

Hva kunne vi gjort bedre?

- Det må korrekturleses nøyere i e-post, og spesielt i dokumentasjon.
- Flyten i testfasene må forbedres. Oppgaver må testes fortere slik at de kan settes i done.
- Det må tas pauser oftere, ting går for tregt når folk blir slitne mot slutten av dagen.
- Føring av timelister har stort forbedringspotensiale.
- Giten har sklidd ut.

Spørsmål og idèer

- Gjennomgå forrige ukes tiltak om å kunne jobbe med småting uten user-story. Dette kan være noe som sklir ut, eller noe som resulterer i noe produktivt. Vi må ha dette i bakhodet.

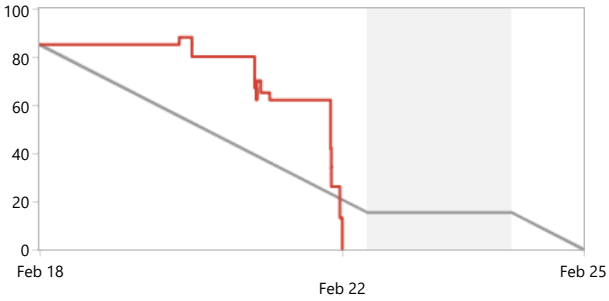
Tiltak

- Man må ta oftere initiativ til å få et team-medlem til å teste det man jobber med.
- En utvikler må selv legge ved et testsett, bruk gjerne cellefunksjonen i jupyter notebook.
- Alle må sørge for at all timebruk er loggført hver fredag før de går.
- Git repo må ryddes opp i førstkommende mandag.
    - Overveie ekstern sensors forslag om to levende brancher, og kjøre off-branch fra develop. Fjerne test.
    - Slett brancher du er ferdig med. Merge fjerner ikke brancher.
    - Personlige brancher skal ikke være langlevende.
    - Navngi brancher etter story-id, ikke eget fornavn.
    - Gitignore test-mappen
- Vær obs på at vi har gått over fra Slack til Webex.
- Prosesshåndboken må gjennomgås og oppdateres.

Sprintrapport

Sprint - Uke 8

Lukket sprint, avsluttet av Martin Evandt 18/feb/19 11:10 AM - 22/feb/19 3:54 PM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (93 → 96)
BG-82	Kartlegge heimdall namespace	Historie	High	UTFØRT	13
BG-84	Koble alerts opp mot prosesser	Historie	Medium	UTFØRT	3
BG-86	Undersøke K-means clustering	Historie	Medium	UTFØRT	5
BG-88	Undersøke atomic red for generering av bad data	Historie	High	UTFØRT	13
BG-89	Generere flere features ut i fra komplekse features	Historie	Medium	UTFØRT	20
BG-90	Første milepælsmøte	Oppgave	Medium	UTFØRT	5
BG-91	Begynne å skrive på kapittelet om maskinlæring	Oppgave	Medium	UTFØRT	8
BG-94	Implementere K-means clustering	Historie	Medium	UTFØRT	13
BG-95	Rydde opp github	Oppgave	Highest	UTFØRT	5
BG-96 *	Oppdatere kommentarer for å matche Googles standard	Oppgave	Medium	UTFØRT	- → 3
BG-99 *	Linke ML-analysert data tilbake til unique_id	Historie	Medium	UTFØRT	8

Retrospektiv - Sprint uke 8

22/02/2019

Hva har gått bra?

- God burndown.
- Medlemmer har tatt mye ansvar og jobbet godt.
- De nye git-retningslinjene fungerte bra.
- God balanse mellom å dokumentere og å kode.
- Sprinten ferdig før helg!

Hva kunne vi gjort bedre?

- Analyserte ikke risiko i sprintplanlegging.
- Prototypefasen sporer lett av.

Spørsmål og idèer

- Maskinlæringsbrukerhistorier bør stykkes opp mer.

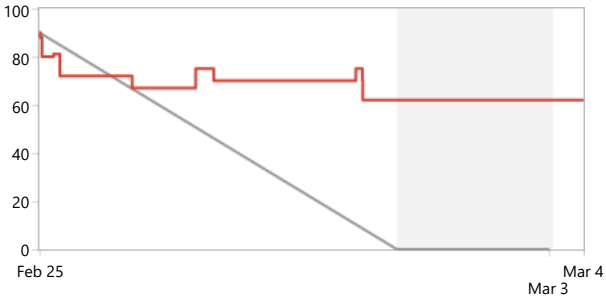
Tiltak

- Lage en plakat med prosessen og alle ting som skal gjennomføres i hvert ledd i en sprint oppsummert enkelt på en A4-side og henge opp.
- Emil må legge lokket på nugattiboksen sin før han går for dagen.

Sprintrapport

Sprint - Uke 9

Lukket sprint, avsluttet av Emil Elsetrønning 25/feb/19 10:06 AM - 04/mar/19 9:23 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (41 → 42)
BG-97	Oppdatere prosesshåndbok til å matche rapport	Oppgave	Medium	UTFØRT	5
BG-98	Gjennomgå rapport, etter tilbakemeldinger fra 1.presentasjon	Oppgave	Medium	UTFØRT	5
BG-101	Generere features fra childprocs	Historie	Medium	UTFØRT	8
BG-103	Konkretisere testrammer	Oppgave	Medium	UTFØRT	8
BG-104	Opprette resten av filstrukturen, (learn, predict, analyse)	Historie	Medium	UTFØRT	2
BG-106 *	Hente ut binary informasjon og koble den på et event	Historie	Medium	UTFØRT	8
BG-109	Rydde opp i SVM test	Feil	Medium	UTFØRT	5
BG-110 *	cmdline_as_args returnerer NaN og ikke 0	Feil	Medium	UTFØRT	- → 1

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (62)
BG-87	Utføre en EDA på datasettet	Historie	Medium	TIL UTFØRING	20
BG-102	Lage en notebook demo	Historie	Medium	COMPLETE	8
BG-105	Endre importer til lokale importer	Feil	Medium	UNDER ARBEID	3
BG-107	Lage benchmarkingrammeverk (ML)	Historie	Medium	UNDER ARBEID	13
BG-108	Generere bad data med atomic red	Historie	Medium	UNDER ARBEID	13
BG-114 *	Implementere flere funksjonaliteter i preprosessoren	Historie	Medium	UNDER ARBEID	5

Retrospektiv - Sprint uke 9

01/03/2019

Hva har gått bra?

- Riskanalyse ble gjennomført
- Raskere på administrativt arbeid

Hva kunne vi gjort bedre?

- Litt ufokuserte
- Lett avsporing

Spørsmål og idèer

- Fortsette med prosjekthåndbok?
- Strukturen på util-funksjonene

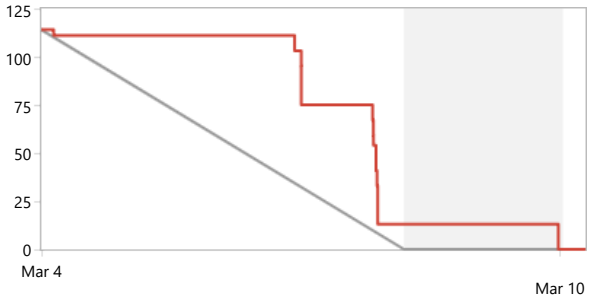
Tiltak

- Timebox jobbing

Sprintrapport

Sprint - Uke 10

Aktiv sprint 04/mar/19 10:29 AM - 10/mar/19 11:18 PM tilknyttet sider



Statusrapport

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (114)
BG-87	Utføre en EDA på datasettet	Historie	Medium	UTFØRT	20
BG-102	Lage en notebook demo	Historie	Medium	UTFØRT	8
BG-105	Endre importer til lokale importer	Feil	Medium	UTFØRT	3
BG-107	Lage benchmarkingrammeverk (ML)	Historie	Medium	UTFØRT	13
BG-108	Generere bad data med atomic red	Historie	Medium	UTFØRT	20
BG-111	Undersøke CrossValidation	Historie	Medium	UTFØRT	8
BG-112	Undersøke Ensamble approach	Historie	Medium	UTFØRT	8
BG-113	Undersøke LDA/QDA	Historie	Medium	UTFØRT	8
BG-114	Implementere flere funksjonaliteter i preprosessoren	Historie	Medium	UTFØRT	5
BG-115	Rapport - preprossering	Oppgave	Medium	UTFØRT	13
BG-116	Rapport - harvest	Oppgave	Medium	UTFØRT	8

Retrospektiv - Sprint uke 10

08/03/2019

Hva har gått bra?

- Prosjektet begynner å ta form.

Hva kunne vi gjort bedre?

- Ting går for fort gjennom testing.

Spørsmål og idèer

- Slå sammen “Test” og “Peer-review” kolonnene i Jira.

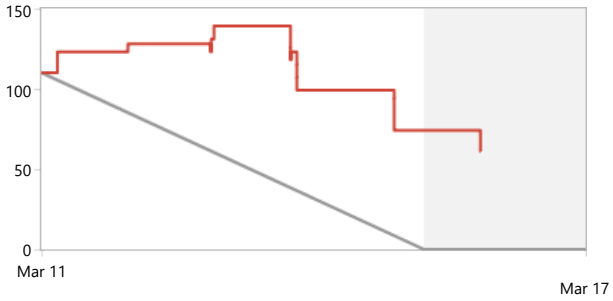
Tiltak

- Gjennomgå risikoanalyse rammeverket på mandag.

Sprintrapport

Sprint - Uke 11

Aktiv sprint 11/mar/19 9:45 AM - 17/mar/19 10:34 PM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (62 → 88)
BG-117	Hvorfor maskinlæring?	Oppgave	Medium	UTFØRT	8
BG-120	Generere feature-sets	Historie	Medium	UTFØRT	13
BG-131	Kartlegge andre presentasjon og dokumentasjon	Oppgave	Medium	UTFØRT	20
BG-137	Gjøre SVM på forskjellige kernels og dokumentere dem	Historie	Medium	UTFØRT	13
BG-140	Gjennomgå risikoanalyserammeverket	Oppgave	Medium	UTFØRT	8
BG-142 *	Sette opp server for maskinlæring	Oppgave	Medium	UTFØRT	- → 5
BG-144 *	Rapport - risk	Oppgave	Medium	UTFØRT	- → 8
BG-145 *	Standardiser presentasjon av modell analyse	Historie	Medium	UTFØRT	- → 8
BG-146 *	Visuell sammenligning av modell resultater	Historie	Medium	UTFØRT	- → 5
BG-147 *	Update feature set in prepare_data	Feil	Medium	UTFØRT	-

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (61)
BG-118	Teste classifiers	Historie	Medium	TIL UTFØRING	13
BG-119	Kartlegge feature-sets	Historie	Medium	TIL UTFØRING	8
BG-135	Research: Heuristic rules	Historie	Medium	PÅ VENT	8
BG-136	Research: Meta-heuristic based decisions	Historie	Medium	TIL UTFØRING	8
BG-138	Research: Grid-seach for optimal parameters	Historie	Medium	PÅ VENT	8
BG-139	Lage Facebook-event	Oppgave	Medium	UNDER ARBEID	3
BG-141 *	Implementere Cross-Validation	Historie	Medium	PÅ VENT	13

Retrospektiv - Sprint uke 11

15/03/2019

Større endringer

I løpet av uke 11 har det blitt gjennomført såpass mange store endringer at vi så behovet for å ha en egen seksjon for disse endringene denne uken.

- Test og peer-review kolonnene har blitt slått sammen, dette fordi test og peer-review gjerne ble gjort samtidig, og det derfor ikke var nødvendig å holde dem separerte.
- Brukerhistorier kan nå bli satt i en pause-kolonne. Dette tillater en brukerhistorie som ligger i en sprint å kunne gå over til neste ukes sprint, uten at den må slettes og re-opprettes i prosjektbacklogen.
- En ny issue type, feature-request, har blitt opprettet i Jira.
- Måten risiko dokumenteres på har blitt endret for å bedre reflektere smidige prosjektmodeller. Flere store endringer har blitt gjennomført i risikorammeverket.
- Strengere rammer rundt prototype-fasen skal implementeres. Det må skrives om algoritmer det jobbes med i rapporten, og resultater må taes vare på.
- Detaljer rundt endringene er allerede, eller skal i løpet av helgen, inn i dokumentasjonen som skal overleveres til universitetet og oppdragsgiver førstkommende tirsdag.

Hva har gått bra?

- Klart å endre fokus fra utvikling til rapport og presentasjon. Fikk gjennomført mye arbeid på kort tid.

Hva kunne vi gjort bedre?

- Research må dokumenteres bedre.
- Andre presentasjon kom for brått på. Vi ser ikke langt nok fram i tid, og jobber litt for mye fra uke til uke.

Spørsmål og idèer

Tiltak

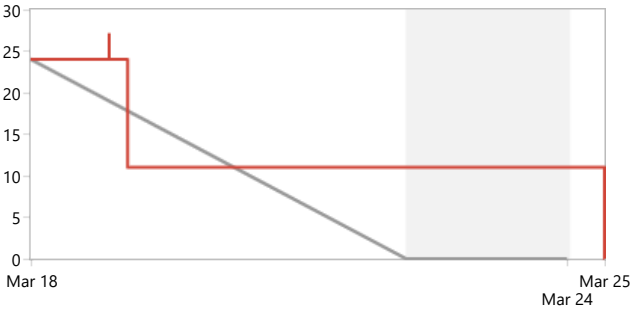
- Skrive om en maskinlæringsalgoritme i rapporten mens man jobber med den.

- Være tydeligere på når vi skal gjøre ting som f.eks retro, tavlegjennomgang av tema, etc. Gi varsel i forkant slik at folk kan gjøre seg ferdige med det de driver med.
- Visualisere tiden fram til milepæler. Telle ned uker på tavlen eller lignende.

Sprintrapport

Sprint - Uke 12

Lukket sprint, avsluttet av Emil Elsetrønning 18/mar/19 10:21 AM - 25/mar/19 10:03 AM tilknyttet sider



Statusrapport

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (24 → 27)
BG-67	Benchmark accuracy på OC-SVM	Historie	Medium	UTFØRT	13
BG-133	Fullføre andre presentasjon	Oppgave	Medium	UTFØRT	-
BG-134	Fullføre dokumentasjon	Oppgave	Medium	UTFØRT	-
BG-139	Lage Facebook-event	Oppgave	Medium	UTFØRT	3
BG-151	Skrive ferdig kapittel om klassifisering	Oppgave	Medium	UTFØRT	8
BG-152	Korrekturlesing	Oppgave	Medium	UTFØRT	-
BG-153	One hot encoding slide	Oppgave	Medium	UTFØRT	-
BG-155	Prosent <noe> inn i rapporten (vedlegg)	Oppgave	Medium	UTFØRT	-
BG-156	Sette inn confusion matrix	Oppgave	Medium	UTFØRT	- → 3
BG-157	Illustrasjon av algoritmer	Oppgave	Medium	UTFØRT	-

Retrospektiv - Sprint uke 12

22/03/2019

Hva har gått bra?

- Bra presentasjon
- Vi sa det samme i utspørringene, det er tydelig at vi alle er på samme bølgelengde i henhold til prosjektets gang.

Hva kunne vi gjort bedre?

- Manglende dokumentasjon, dette må utbedres.
- Vi har ikke skrevet om funn.
- Scrum ble ikke fulgt denne uken.

Spørsmål og idèer

- Ønske om mulighet til å jobbe individuelt når en driver med research tunge oppgaver er kommet. Gruppen vil åpne for å teste dette.
- Rollefordelingen har mistet sin tydelighet, skal gåes over på neste retrospektiv.

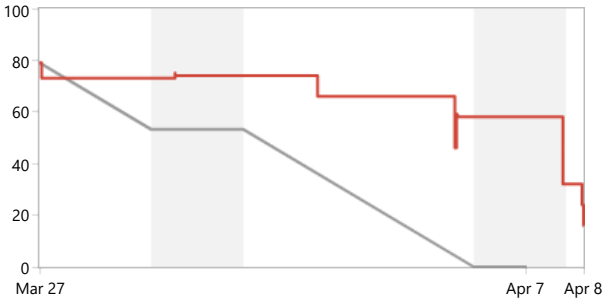
Tiltak

- Lage rammeverk og en rapportmal for funn. Samt prosedyre for generelt arbeid.

Sprintrapport

Sprint - Uke 13

Lukket sprint, avsluttet av Tone Marie Hognestad 27/mar/19 1:45 PM - 08/apr/19 9:07 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (63 → 77)
BG-138	Research: Grid-seach for optimal parameters	Historie	Medium	UTFØRT	8
BG-141	Implementere Cross-Validation: test/training	Historie	Medium	UTFØRT	13
BG-158	Lage et dokumentasjonsrammeverk	Oppgave	Highest	UTFØRT	20
BG-159	Endre input til analyseverktøy	Forbedring til kode	Medium	UTFØRT	8
BG-160	Lagre trente modeller	Historie	Medium	UTFØRT	3
BG-161	Bruke lagrede modeller i .predict	Historie	Medium	UTFØRT	3
BG-163	Undersøke feature korrolasjon	Historie	Medium	UTFØRT	8
BG-167 *	Prepare features: optional drop duplicates	Forbedring til kode	Medium	UTFØRT	- → 1
BG-170 *	Implementere Grid search	Historie	Medium	UTFØRT	- → 13

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (16)
BG-164	Undersøke online learning	Historie	Medium	UNDER ARBEID	8
BG-165	Undersøke feedback training	Historie	Medium	TIL UTFØRING	8

Retrospektiv - Sprint uke 13 & 14

05/04/2019

Hva har gått bra?

- Resterende arbeid er klarere, plan på dokumentasjonen er ferdig, minimum viable product er etablert.

Hva kunne vi gjort bedre?

- Strukturen sklidde ut grunnet eksamener.
- Kommunikasjonen med Telenor og veileder har vært veldig sporadisk etter andre presentasjon.

Spørsmål og idèer

-

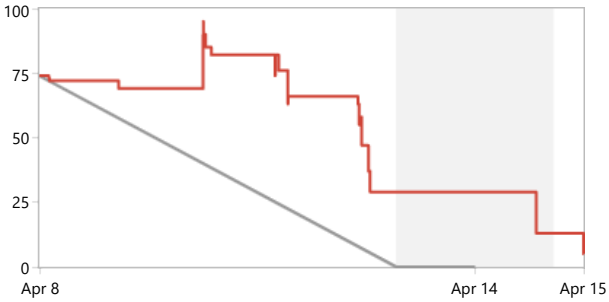
Tiltak

- Fast klokkeslett på retrospektiv; 14:00 - 15:00.

Sprintrapport

Sprint - Uke 14

Lukket sprint, avsluttet av Tone Marie Hognestad 08/apr/19 11:15 AM - 15/apr/19 9:08 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (101)
BG-164	Undersøke online learning	Historie	Medium	UTFØRT	8
BG-168	Klargjøre til USNExpo	Oppgave	Medium	UTFØRT	5
BG-171	Kapittel 1: Bakgrunn	Oppgave	Medium	UTFØRT	1
BG-172	Kapittel 1: Gruppemedlemmer	Oppgave	Medium	UTFØRT	1
BG-173	Kapittel 1: Dokumentstruktur	Oppgave	Medium	UTFØRT	3
BG-174	Kapittel 2: Telenor Norge	Oppgave	Medium	UTFØRT	1
BG-175	Kapittel 2: Scenariet	Oppgave	Medium	UTFØRT	3
BG-176	Kapittel 2: Oppgaven	Oppgave	Medium	UTFØRT	1
BG-177 *	Kapittel 3: Eksterne rammeverk & verktøy	Oppgave	Medium	UTFØRT	5
BG-178 *	Kapittel 3: Akademisk teori, cybersikkerhet	Oppgave	Medium	UTFØRT	1
BG-180 *	Kapittel 3: Akademisk teori, Endpoint Detection & Response	Oppgave	Medium	UTFØRT	3
BG-181 *	Kapittel 3: Scrum & Agile	Oppgave	Medium	UTFØRT	5
BG-182 *	Kapittel 3: Testing	Oppgave	Medium	UTFØRT	1
BG-183 *	Kapittel 3: Risiko	Oppgave	Medium	UTFØRT	1
BG-184 *	Kapittel 3: Programpakken Heimdall	Oppgave	Medium	UTFØRT	5
BG-186 *	Kapittel 4: Implementasjon av prosessen, Testing	Oppgave	Medium	UTFØRT	3
BG-203	False negatives og false positives i generate_rapport	Forbedring til kode	Medium	UTFØRT	3
BG-205	Undersøke alternativer til encoding av kategorisk data	Historie	Medium	UTFØRT	8
BG-206	Finne alt bad på cmd og powershell	Historie	Medium	UTFØRT	13
BG-207	Clustering på powershell	Historie	Medium	UTFØRT	8
BG-209	Preprosessere powershell datasett	Historie	Medium	UTFØRT	8
BG-210	Lag verktøy for powershell og cmd	Historie	Medium	UTFØRT	3

argumentuthenting

BG-214 *	Trene på dag 1, forutse dag 2.	Historie	Medium	UTFØRT	8
BG-215 *	Scatter plot av decision function i rapport	Forbedring til kode	Medium	UTFØRT	3

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (5)
BG-179 *	Kapittel 3: Akademisk teori, maskinlæring	Oppgave	Medium	TIL UTFØRING	5

Saker fjernet fra sprint

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (8)
BG-165	Undersøke feedback training	Historie	Medium	PÅ VENT	8

Retrospektiv - Sprint uke 15

12/04/2019

Hva har gått bra?

- Enorme framskritt denne uken
- Datasettet har blitt korrekt lablet
- Mange gode funn

Hva kunne vi gjort bedre?

- Ikke fulgt Scrum godt nok i alle tilfeller. Noe arbeide har blitt gjort uten en story.

Spørsmål og idèer

- Revurdere behovet for for .learn og .predict submodulene, vi merker at mye av maskinlæringen fungerer “out-of-the-box” fra SciKit.

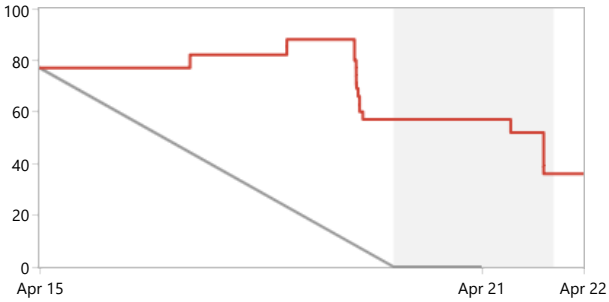
Tiltak

- Om noen spoiler en episode av Game of Thrones de neste ukene, blir vedkommende kastet ut av gruppen.

Sprintrapport

Sprint - Uke 16

Lukket sprint, avsluttet av Tone Marie Hognestad 15/apr/19 1:37 PM - 22/apr/19 8:59 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (52)
BG-165	Undersøke feedback training	Historie	Medium	UTFØRT	8
BG-179	Kapittel 3: Akademisk teori, maskinlæring	Oppgave	Medium	UTFØRT	5
BG-185	Kapittel 4: Implementasjon av prosessen, Scrum	Oppgave	Medium	UTFØRT	3
BG-187	Kapittel 4: Implementasjon av prosessen, Risiko	Oppgave	Medium	UTFØRT	3
BG-188	Kapittel 4: Implementasjon av prosessen, Endringer gjort underveis	Oppgave	Medium	UTFØRT	3
BG-189	Kapittel 4: Kommunikasjon med oppdragsgiver	Oppgave	Medium	UTFØRT	3
BG-190	Kapittel 4: Møter med intern veileder	Oppgave	Medium	UTFØRT	3
BG-193	Kapittel 4: Stilguide & navnekonvensjoner	Oppgave	Medium	UTFØRT	1
BG-194	Kapittel 4: Sphinx & teknisk dokumentasjon	Oppgave	Medium	UTFØRT	1
BG-213	Generate report lager en 'CB_link' kolonne i df	Feil	Medium	UTFØRT	-
BG-217	Generate_report krever av labels sendes inn som dataframe, mens det i beskrivelse står et array.	Feil	Medium	UTFØRT	-
BG-218	Kapittel 4: Implementasjon av prosessen, Testing	Oppgave	Medium	UTFØRT	3
BG-220	Implementere feature selection	Historie	Medium	UTFØRT	13
BG-221 *	Oppdater label datasett på drive	Feil	Medium	UTFØRT	-
BG-228 *	Kapittel 3: grid search, cross validation	Oppgave	Medium	UTFØRT	3
BG-229 *	Kapittel 3: Dimensjonreduksjon	Oppgave	Medium	UTFØRT	3

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (36)
BG-191	Kapittel 4: Research- & læringsprosess	Oppgave	Medium	TIL UTFØRING	5
BG-200	Kartlegge subseksjoner og innhold i kapittel 8: analyse	Oppgave	Medium	TIL UTFØRING	-
BG-208	Ensemble approach	Historie	Medium	UNDER ARBEID	13

BG-211	Git ignore .checkpoint files	Feil	Medium	UNDER ARBEID	-
BG-219	Vedlegg: dokumentere classifier-resultater	Oppgave	Medium	UNDER ARBEID	13
BG-222 *	Teste maskinlæring med cluster-feature	Historie	Medium	UNDER ARBEID	5

Retrospektiv - Sprint uke 16

18/04/2019

Hva har gått bra?

- Rapporten er bedre strukturert, vedleggsdokument er planlagt
- Første iterasjon av feedback-training begynner å ta form.
- Første iterasjon av ensemble approach begynner å ta form.
- Dashboard som henter inn JSON-filer og visualiserer resultatene på en god måte er på plass.

Hva kunne vi gjort bedre?

- Litt løs struktur på prosessen denne uken. Ventilasjonsanlegget på skolen har vært slått av i ferien, noe som har ført til dårlig luft og lite initiativ.
- Startet på mye forskjellig, uten å fullføre noe konkret.
- Løsningsorienterte, men lite framtidsretta i form av hvordan pakken er strukturert.

Spørsmål og idèer

- Vurdere brukervennligheten av programpakken.

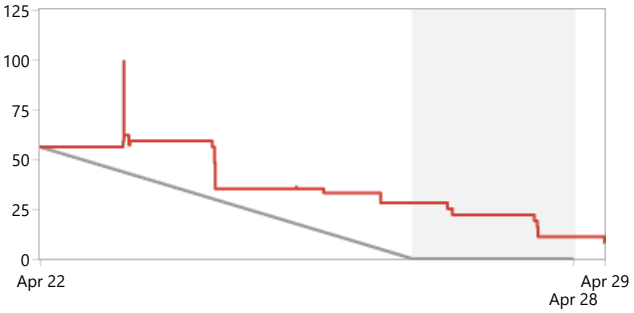
Tiltak

- Gå over rammeverk, gjøre hjelpefunksjoner til private.
- Følge scrumboard i JIRA for bedre struktur og kontroll på prosessen.
- Mer rapport fokus i parallell med annet arbeid.
- Gå over timelister.
- Kartlegge, og se på muligheten for å strukturere i klasser ved hjelp av UML.

Sprintrapport

Sprint - Uke 17

Lukket sprint, avsluttet av Tone Marie Hognestad 22/apr/19 10:35 AM - 29/apr/19 8:29 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (89 → 57)
BG-195	Kapittel 5: Krav til maskinlæringsmodeller	Oppgave	Medium	UTFØRT	3
BG-197	Kapittel 6: Heimdall-pakken	Oppgave	Medium	UTFØRT	3
BG-199 *	Kartlegge subseksjoner og innhold i kapittel 7: implementasjon	Oppgave	Medium	UTFØRT	- → 2
BG-200	Kartlegge subseksjoner og innhold i kapittel 8: analyse	Oppgave	Medium	UTFØRT	3
BG-208	Ensemble approach	Historie	Medium	UTFØRT	13
BG-211	Git ignore .checkpoint files	Feil	Medium	UTFØRT	3
BG-212 *	Skille mellom tekniske og forretningsrelaterte risikoer i Kapittel 4: Implementasjon av prosessen, risikoer	Oppgave	Medium	UTFØRT	- → 1
BG-222	Teste maskinlæring med cluster-feature	Historie	Medium	UTFØRT	5
BG-224	analyze.generate_report legger ikke til en backslash i slutten av path, analyze.json_report gjør det.	Feil	Medium	UTFØRT	1
BG-225	analyze.generate_report og analyze.json_report må lage pathen om den ikke eksisterer	Forbedring til kode	Medium	UTFØRT	3
BG-230	Endre analyseverktøy til -1 for positiv klasse og 1 for negativ klasse for å tilpasse standard	Forbedring til kode	Medium	UTFØRT	3
BG-231	Kapittel 3: Online Learning	Oppgave	Medium	UTFØRT	3
BG-232	Planlegge pipelineobjektet	Historie	Medium	UTFØRT	8
BG-237 *	analyze.generate_report bør ta label som en series	Forbedring til kode	Medium	UTFØRT	1
BG-238 *	Bytt ut modellene i teorikapittelet med egne visualiseringer	Oppgave	Medium	UTFØRT	40 → 3
BG-239 *	Preprocess bør droppe kolonner som ikke er i encodings.	Feil	Medium	UTFØRT	- → 2

BG-241 *	tittel kreves i .learn() selvom ikke rapport genereres	Feil	Medium	UTFØRT	-

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (8)
BG-196	Kapittel 5: Krav til verktøyspakken	Oppgave	Medium	TIL UTFØRING	3
BG-233	Implementere pipelineobjektet	Historie	Medium	TIL UTFØRING	-
BG-234	Fullføre feedback training	Historie	Medium	COMPLETE	5

Retrospektiv - Sprint uke 17

26/04/2019

Hva har gått bra?

- Fått gjennomgått mye kode, og rettet bugs.
- Interface mellom de ulike modulene faller på plass.
- Gruppen har oversikt over de resterende ukene, og hva som gjenstår.

Hva kunne vi gjort bedre?

- Klassen for å interface de ulike modulene har vært dårlig planlagt. Som et resultat av dette har det blitt jobbing parallelt i klasser, noe som har ført til at noe arbeid har måttet vente til annet er utført.
- Lite rapport.

Spørsmål og idèer

-

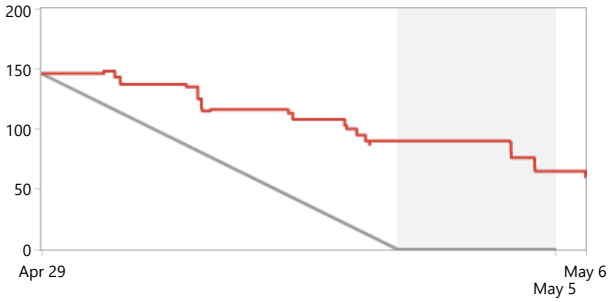
Tiltak

- Se på rapport i helgen. Få ferdigstilt kode iløpet av neste uke, for å ha fullt fokus på rapport frem før innlevering.

Sprintrapport

Sprint - Uke 18

Lukket sprint, avsluttet av Tone Marie Hognestad 29/apr/19 12:18 PM - 06/mai/19 8:53 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (88 → 92)
BG-196	Kapittel 5: Krav til verktøyspakken	Oppgave	Medium	UTFØRT	3
BG-234	Fullføre feedback training	Historie	Medium	UTFØRT	5
BG-242	_limit_to_processes(self, processes) returnerer ikke noe, setter kun den lagra _df'en	Feil	Medium	UTFØRT	-
BG-243	Pipeline: inkludere hyp. optimalisering som funksjon.	Forbedring til kode	Medium	UTFØRT	3
BG-246	Dashboard: når en node klikkes på markeres eksempel i lista	Forbedring til kode	Medium	UTFØRT	5
BG-252	Rydde i examples	Historie	Medium	UTFØRT	3
BG-257	Lage en standard for examples	Historie	Medium	UTFØRT	3
BG-258	Lag egne mapper for html/json/csv fra path	Forbedring til kode	Medium	UTFØRT	2
BG-259	Kapittel 6: UML-diagrammer - Use Case - Online Pipeline og Pipeline	Oppgave	Medium	UTFØRT	5
BG-261	Pipeline Demo	Historie	Medium	UTFØRT	8
BG-263	Util Demo	Historie	Medium	UTFØRT	8
BG-264	Harvest Demo	Historie	Medium	UTFØRT	3
BG-271	Oppdatere EDA og få inn feature selection	Historie	Medium	UTFØRT	5
BG-272	Kapittel 7: Utils	Oppgave	Medium	UTFØRT	5
BG-273	Kapittel 7: Preprocess	Oppgave	Medium	UTFØRT	5
BG-274	Kapittel 7: Pipeline	Oppgave	Medium	UTFØRT	5
BG-275	Kapittel 7: Online Pipeline	Oppgave	Medium	UTFØRT	3
BG-276	Kapittel 7: Ensemble Approach	Oppgave	Medium	UTFØRT	5
BG-277	Kapittel 3: Hashing og preprocessing	Oppgave	Medium	UTFØRT	5
BG-278	Kapittel 3: Ensemble Approach	Oppgave	Medium	UTFØRT	5
BG-280 *	Lese inn via json.load() for å generere json	Forbedring til kode	Medium	UTFØRT	2

BG-284 *	Docsstrings for pipeline.py	Feil	Medium	UTFØRT	- → 1
BG-286 *	Kapittel 7: EDA/Dimension reduction	Oppgave	Medium	UTFØRT	- → 3

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (60)
BG-233	Implementere pipelineobjektet	Historie	Medium	TIL UTFØRING	-
BG-260	Kapittel 6: UML-diagrammer - Class - Hele pakka	Oppgave	Medium	TIL UTFØRING	8
BG-262	Heimdall Demo	Historie	Medium	UNDER ARBEID	8
BG-265	Online Pipeline Demo	Historie	Medium	COMPLETE	8
BG-266	Generere resultater på totalt sett (train/test) for alle offline classifiers	Oppgave	Medium	ÅPNET PÅ NYTT.	5
BG-267	Generere resultater resultater med på de to siste dagene hver for seg for testing for alle offline classifiers	Oppgave	Medium	TIL UTFØRING	5
BG-268	Generere resultater på totalt sett (train/test) for alle offline classifiers (HELE SETTET!)	Oppgave	Medium	UNDER ARBEID	5
BG-269	Generere resultater på totalt sett (train/test) dag for dag på alle online classifiers.	Oppgave	Medium	ÅPNET PÅ NYTT.	5
BG-270	Generere resultater på totalt sett (train/test) dag for dag, og gi feedback på alle online classifiers.	Oppgave	Medium	ÅPNET PÅ NYTT.	8
BG-279	Kapittel 8: EDA og feature selection	Oppgave	Medium	UNDER ARBEID	8

Retrospektiv - Sprint uke 18

03/05/2019

Hva har gått bra?

- Precision/Recall kurver og ROC kurver har gitt misvisende resultater. Har fått endret opp i dette før de endelige resultatene genereres.
- Backloggen har vært mer fullkommen denne uka, noe som har gjort at den i stor grad har blitt fulgt

Hva kunne vi gjort bedre?

- Når endringer skal gjøres har det kommet frem at koden er lite generell, og at en endring kan skape følgefeil flere steder.

Spørsmål og idèer

-

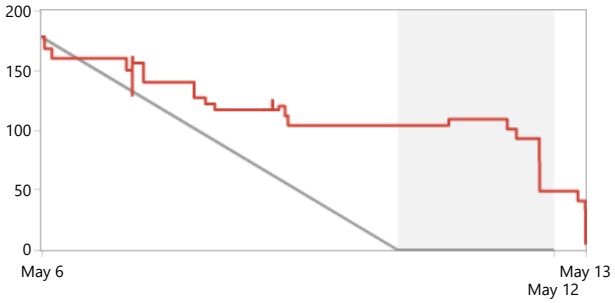
Tiltak

- Lese rapport i helga. Skaffe et overblikk over hva som mangler, og hva som kan gjøres bedre.
- Rydding i kode må skje etter rapport. Rapport er første prioritet mot innlevering av rapport for godkjenning hos bedrift.

Sprintrapport

Sprint - Uke 19

Lukket sprint, avsluttet av Tone Marie Hognestad 06/mai/19 10:45 AM - 13/mai/19 9:37 AM tilknyttet sider



Statusrapport

* Sak lagt til sprinten etter starttidspunktet

Fullførte saker

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (221)
BG-260	Kapittel 6: UML-diagrammer - Class - Hele pakka	Oppgave	Medium	UTFØRT	8
BG-262	Heimdall Demo	Historie	Medium	UTFØRT	8
BG-265	Online Pipeline Demo	Historie	Medium	UTFØRT	8
BG-266	Generere resultater på totalt sett (train/test) for alle offline classifiers	Oppgave	Medium	UTFØRT	5
BG-267	Generere resultater resultater med på de to siste dagene hver for seg for testing for alle offline classifiers	Oppgave	Medium	UTFØRT	5
BG-268	Generere resultater på totalt sett (train/test) for alle offline classifiers (HELE SETTET!)	Oppgave	Medium	UTFØRT	5
BG-269	Generere resultater på totalt sett (train/test) dag for dag på alle online classifiers.	Oppgave	Medium	UTFØRT	5
BG-270	Generere resultater på totalt sett (train/test) dag for dag, og gi feedback på alle online classifiers.	Oppgave	Medium	UTFØRT	8
BG-279	Kapittel 8: EDA og feature selection	Oppgave	Medium	UTFØRT	8
BG-283	Fjerne navn på EDR system under kap 7.1 harvest.py	Oppgave	Medium	UTFØRT	-
BG-285	Skrive om analyze.py	Oppgave	Medium	UTFØRT	5
BG-289	Fylle inn placeholder-tekster i rapporten	Oppgave	Medium	UTFØRT	-
BG-297	Bytte ut bilder i figurer	Oppgave	Medium	UTFØRT	-
BG-301	Kapittel 8: Datasettet	Oppgave	Medium	UTFØRT	5
BG-302	Kapittel 8: Offline modeller	Oppgave	Medium	UTFØRT	13
BG-303	Kapittel 8: Outlier detection modeller	Oppgave	Medium	UTFØRT	5
BG-304	Kapittel 8: Online Modeller	Oppgave	Medium	UTFØRT	13
BG-305	Kapittel 8: Novelty detection modeller	Oppgave	Medium	UTFØRT	8
BG-314	Kapittel 10 - Konklusjon	Oppgave	Medium	UTFØRT	20
BG-318	Kapittel 9 - Evaluering	Oppgave	Medium	UTFØRT	20

BG-321	Gå over docstrings i Heimdall-pakken	Historie	Medium	UTFØRT	8
BG-322	Oppdatere nettside	Historie	Medium	UTFØRT	5
BG-323	Kap 9 / Kap 10 / Vedlegg - Fremtidig arbeid	Oppgave	Medium	UTFØRT	8
BG-325	Kapittel 3: Oppdatere eksterne rammeverk lista	Oppgave	Medium	UTFØRT	8
BG-326 *	Resultater: cmd pwshell	Oppgave	Medium	UTFØRT	8
BG-327 *	Resultater Online	Oppgave	Medium	UTFØRT	8
BG-328 *	Resultater: pred_last	Oppgave	Medium	UTFØRT	8
BG-329 *	Resultater: alt	Oppgave	Medium	UTFØRT	8
BG-330 *	Resultater Online Feedback	Oppgave	Medium	UTFØRT	8
BG-331 *	Eksterne verktøy og rammeverk: Jupyter notebook	Oppgave	Medium	UTFØRT	3

Saker ikke fullført

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (- → 5)
BG-287	Rettskrivning av rapporten Kap 1-5	Oppgave	Medium	UNDER ARBEID	- → 5
BG-288	Rettskrivning av rapporten Kap 6-10 + vedlegg	Oppgave	Medium	TIL UTFØRING	-

Saker fullført utenfor denne sprinten

Se i saksnavigator

ID	Sammendrag	Sakstype	Prioritet	Status	Story Points (-)
BG-233	Implementere pipelineobjektet	Historie	Medium	UTFØRT	-