DEN EUROPÆISKE UNION
Den Europæiske Socialfond
DEN EUROPÆISKE UNION
Den Europæiske Fond
for Regionaludvikling
Finansieret som et led i EU's reaktion
på COVID-19-pandemien
Vi investerer i din fremtid

Fyrtårnsprojekt - Offentlig tilgængelig dynamisk GIS Platform – Fase 1 - Vanddata

Forord

Produktet er udviklet i projektet "Offentligt tilgængelig dynamisk GIS platform – Fase 1" som en del af Erhvervsfyrtårn for vandteknologi, hvilket er baseret på REACT-EU-midler fra Danmarks Erhvervsfremmebestyrelse.

Projektet har indeholdt flere delprojekter og indeværende dokument omhandler alene det delprojekt, som har arbejdet med offentlig tilgængelige tidsserier for vanddata under arbejdspakke 3 "kvalitetssikring".

Overordnet formålsbeskrivelse for arbejdspakken: *"Metodeudvikling i forbindelse med kvalitetssikring og dokumentation af tidsseriedata"*

Ud over det bidrag i delprojektet, som er leveret af WatsonC (arbejdspakke 3) har NIRAS gennemført en brugerundersøgelse som har haft til formål at kortlægge aftageres ønsker og behov for tilgængelighed, kvalitetssikring og dokumentation. Derudover har Danmarks Miljøportal arbejdet med tilpasninger af de services, som modtager data fra dataleverandører vandområdet.

Indholdsfortegnelse

WATSONC
Tlf: +45 2843 5100 ● mail@watsonc.dk ● **www.watsonc.dk**

1 Automatisk kvalitetssikring af data:

Målet for WatsonC er at indkomne data så vidt muligt repræsenterer virkeligheden, således de er værdiskabende og kan danne handlingsgrundlag for os selv og vores kunder.
Data med mangler eller fejl er derfor direkte modstående i forhold til den vision. Derfor ønsker vi at identificere og rette mangler hurtigst muligt. Samtidig er det vigtigt at korrektionen er korrekt, troværdig og reversibel, således kunden har det fulde overblik over, hvordan deres data håndteres.

Det overordnede mål:

Først vil vi identificere behov, ambitioner og forsøge at etablere en vision for et ideelt automatisk kvalitetssikringsmodul.
Kernebehov:
Hurtigt, troværdigt/robust, gennemsigtigt, reversibelt, automatisk og veldokumenteret.

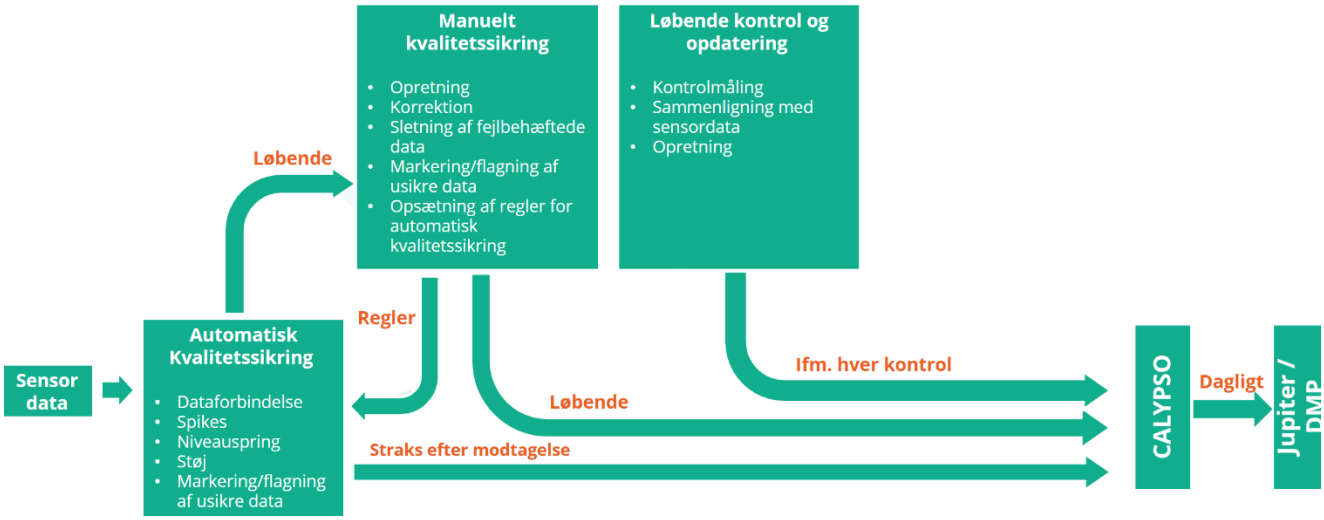
Ambitioner:
Værdiskabende igennem hele processen, modulært, mulighed for iterative forbedringer, at minimere behov for menneskelig intervention.

Typiske fejltyper:
Sensor out-of-range, niveauspring (sensor flyttes eller lignende), spikes (pludselig abnormal hændelse), støj, huludfyldning, fjernelse af barometer effekt.

1.1 Det ideelle resultat:

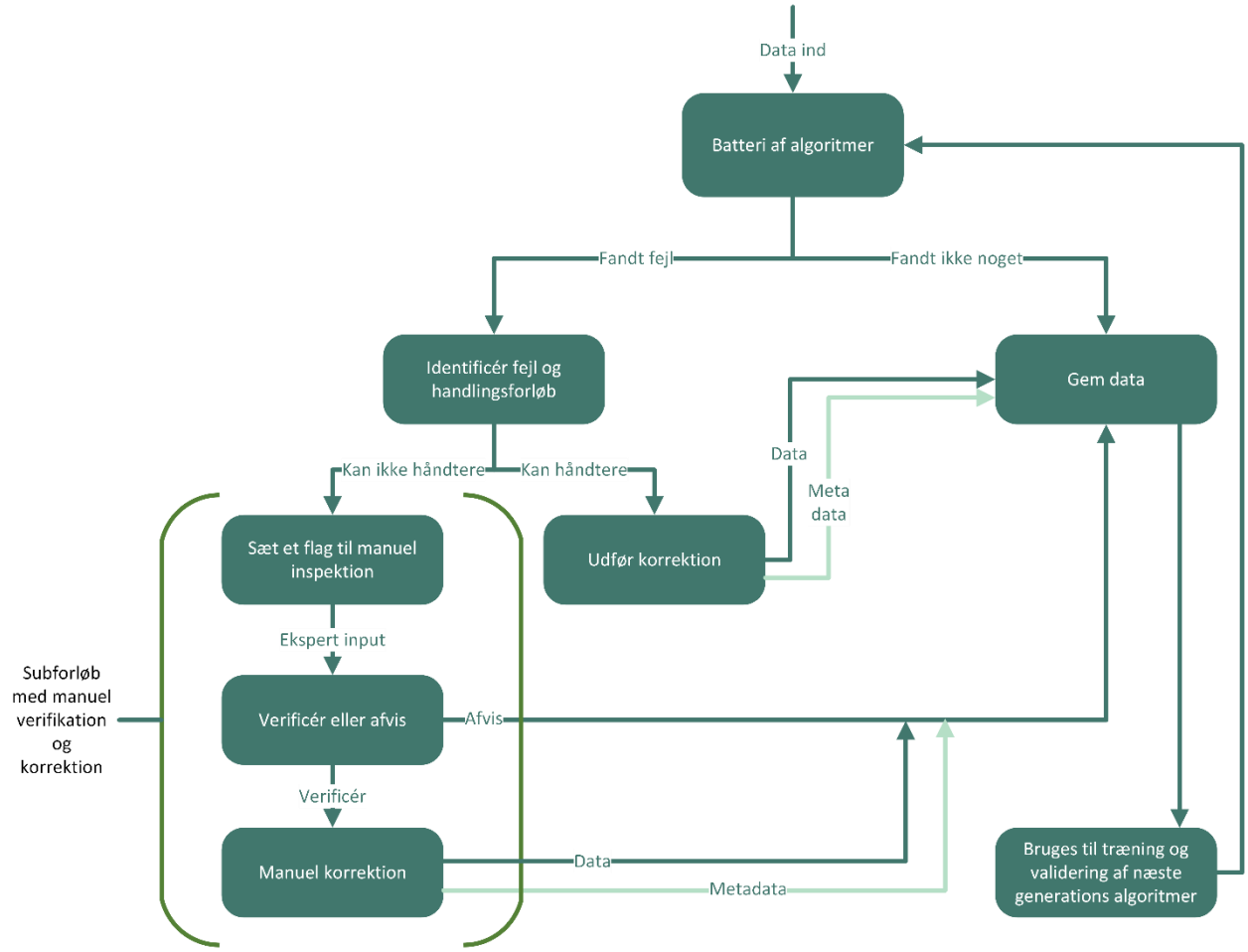
Vi vil forsøge at opstille, hvad vi opfatter som det ideelle resultat af udviklingsprocessen omkring automatisk kvalitetssikring og bruge det som fyrtårn.

Som set på figur 1.1 skal data igennem en automatisk kvalitetssikring inden visning eller videregivelse til øvrige aftagere. Her skal der foretages automatiserede korrektioner, markering af data af tvivlsom kvalitet med tilhørende usikkerhed og notificeres om eventuelle problemer med udstyr.
Manuel kvalitetssikring foregår løbende, så det sikres at outputtet af den automatiske kvalitetssikring er korrekt og troværdigt, samt vurderes om der kan dannes grundlag for nye tiltag i den automatiske kvalitetssikring. For hver korrektion, manuel eller automatisk, skal det noteres som metadata til den viste data.



Figur 1.1 Overordnet flow for en god kvalitetssikring

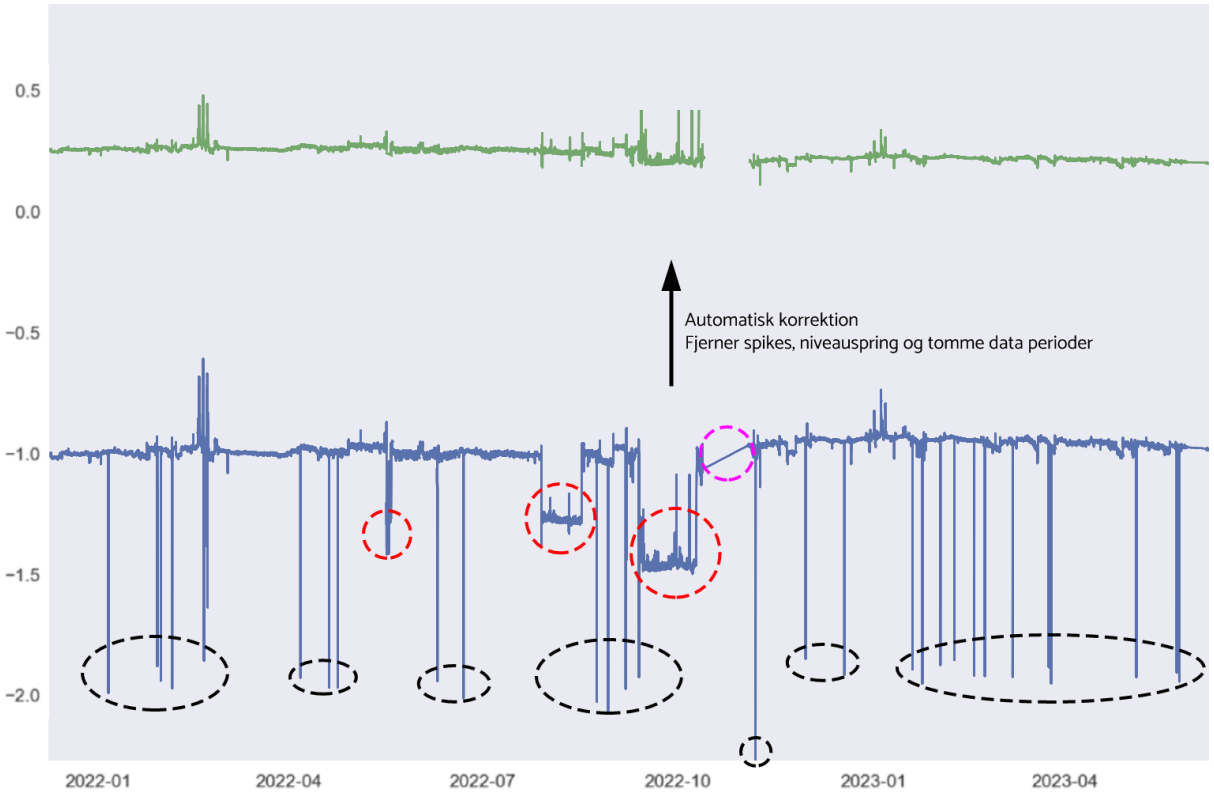
Det ideelle resultat indebærer, at den automatiske kvalitetssikring er robust og kræver et minimum af manuel korrektion. Derfor, hvis der kommer en given tidsserie ind i vores system, så skal systemet følge flowet på figur 1.2, uden at gå ned i subforløbet.
I et cirkulært flow udføres automatiske korrektioner, hvor data danner grundlag for næste generations algoritmer. Vi øger altså datagrundlaget og datakvaliteten kontinuerligt. Kernen i systemet er algoritmebatteriet, hvori der kan tilføjes og fjernes algoritmer afhængig af deres ydeevne og vores behov. Ved implementering af nye algoritmer skal der bruges tid på verifikation i subforløbet med ekspertviden.



Figur 1-.2 Overordnet kvalitetssikrings- og dataflow

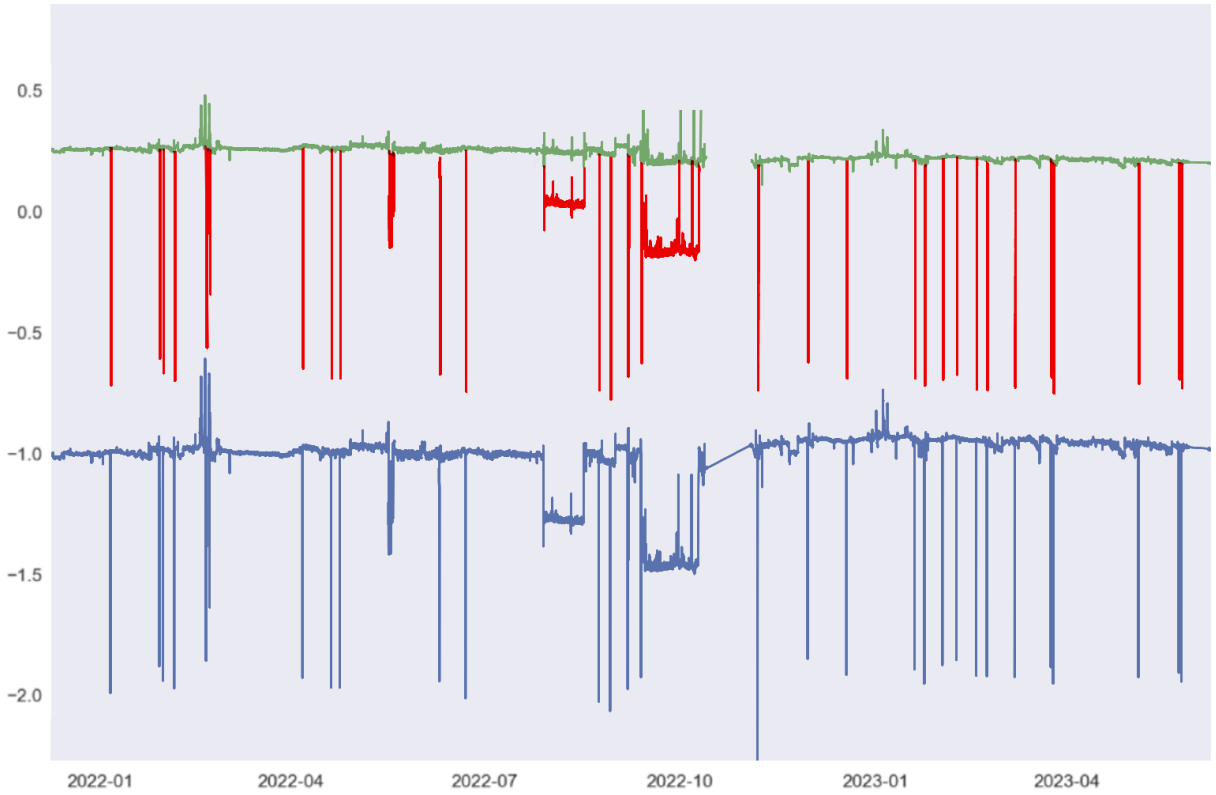
Hvis det kan opnås ikke at skulle ind i subforløbet, så er pipelinen fuldautomatisk. I realiteten er det et drømmescenarie og der vil opstå fejl, som kræver menneskelig intervention. Derfor er den konkrete opgave at mindske tiden vi bruger i subforløbet og sikre at pipelinen er så robust som muligt, så drift og vedligehold minimeres.
Konkret, så forestiller vi os, at den endelige løsning bliver en eller flere ML/AI baserede algoritmer (LSTM, CNN og måske endda en Transformer model). En forudsætning herfor er opbygning af et solidt datagrundlag, som vi opbygger løbende.

Et eksempel herunder (figur 1.3) viser hvordan outputtet af et ideelt KS system kunne se ud. Tydelige afvigelser i form af spikes og niveauspring er fjernet og manglende data er tydeliggjort.



Figur 1.3 Eksempel på et muligt output af automatisk detektion og korrektion, grøn = korrigeret tidsserie, blå = originalt signal (reference), sort = spikes, rød = niveauspring, lilla = manglende data

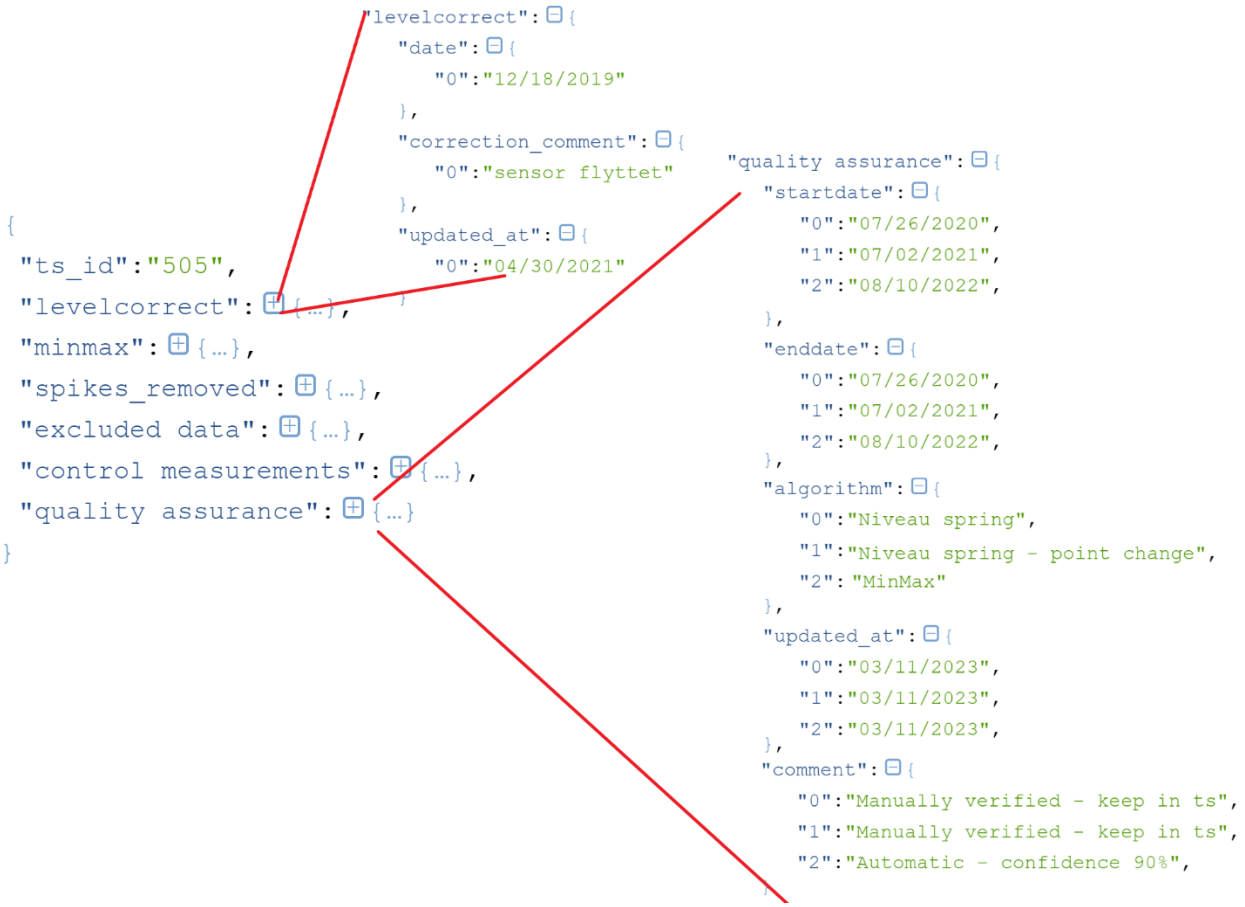
Det ønskes også at ændringer på en given tidsserie er gennemsigtige, hvorfor det er nødvendigt at man kan se forskellen på en given tidsserie og den ubehandlede version. Dette gøres gennem metadata, hvor alle hændelser kortlægges med dato, type og årsag m.m. Det er tiltænkt at kunne visualiseres som i figur 1.4.



Figur 1.4 Visning af korrigeret tidsserie. Grøn = korrigeret tidsserie, blå = originalt signal (reference) rød = udførte korrektioner.

Her er alle ændringer vist i rødt, men hver hændelsestype kunne fremhæves separat.

Både data og metadata opbevares i en SQL-database, hvor ændringer og korrektioner gemmes på hændelsesbasis og aggregeres i en json struktur, som set på figur 1.5. Her kan der tilføjes enkelte hændelser eller hændelsestyper, så det er simpelt at ændre metadataens struktur.



Figur 1.5 json struktur for tidsserie id = 505. To typer metadata, niveauspringskorrektion og kvalitetssikring, er foldet ud.

1.2 Hvordan skaber vi værdi på vej mod målet?

Vi ser overordnet fire store punkter for værdiskabende tiltag på vej mod et ideelt resultat:

1. Automatiseret fejlfinding med notifikation (udbygges løbende)
2. Manuel korrektion på baggrund af fejlfinding --> bedre datagrundlag og højere datakvalitet
3. Dokumentation og visning af de korrektioner, som er foretaget
4. Opbygning af pipelines til deployment og optimering

Den grundlæggende idé er at skrive 1. generations algoritmer, som udelukkende er assisterende i natur. De forsøger at identificere og notificere omkring fejl, men ikke udføre korrektion eller lignende. Herefter skal tidsseriedataen vurderes af eksperter, som bruger algoritmeoutputtet, som pejlemærke til at finde steder, der kræver korrektion, samt at notere, hvad der er naturligt, som algoritmen flagede forkert. Vi udvikler altså en form for Rapid Prototyping Algoritmer. Det forsøges at holde algoritmer til den konservative side i starten- vi vil hellere have falske negativer end falske positiver - da antagelsen er at det meste af tidsseriedataen er korrekt. Her skabes værdi ved at vi løbende får gennemgået tidsserier for fejl, hvor fejlene opdages hurtigere og mere fokuseret, samtidig med at korrektionen har høj troværdighed.

I takt med at der udføres flere korrektioner og findes fejl har næste generations algoritmer mere data at valideres ud fra, således, at de er mere præcise når der forekommer nye hændelser.

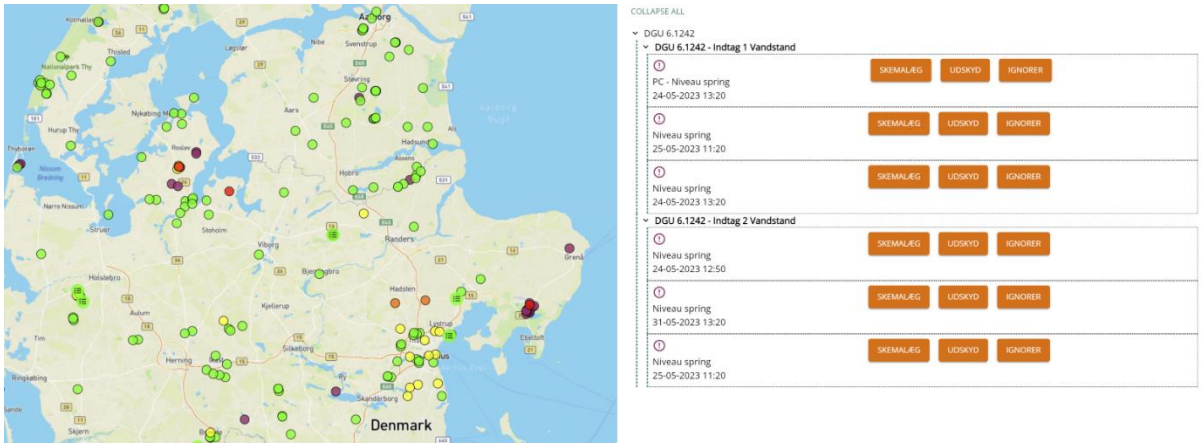
En del af vejen til det ideelle resultat er at opbygge flows eller pipelines, som kan bruges uafhængigt af algoritmetype, således, at systemet er modulært og man kan lave "plug'n'play" algoritmer. I et sådant system kan vi frit vælge de algoritmer som skal benyttes.
Vores nuværende løsning er et cloud-baseret flow, som kører med en fast frekvens og udfører en forudbestemt korrektion og en mere generel fejldetektion. I sidstnævnte tilføjer vi løbende nye fejltyper og forbedringer.
Det er vigtigt at få opbygget en pipeline til optimering (træning) af algoritmer, således de kan forbedres i takt med det stigende datagrundlag. Denne skal integreres med den overordnede pipeline.

Et andet værdiskabende element er udviklingen af et annoteringsværktøj, som gør det lettere for eksperter at udvælge data til korrektion. Det er et værktøj, som assisterer i subforløbet ovenfor, og hjælper til opbygning af datagrundlaget for bedre algoritmer.

Slutteligt skaber det værdi at etablere en datamodel for dokumentation af korrektioner og advarsler på de enkelte tidsserier. At have en intern historik over, hvordan en tidsserie er korrigeret, gør det lettere at sikre kvaliteten af tidsserien, da vi altid kan reversere korrektionen og sammenholde før og efter.

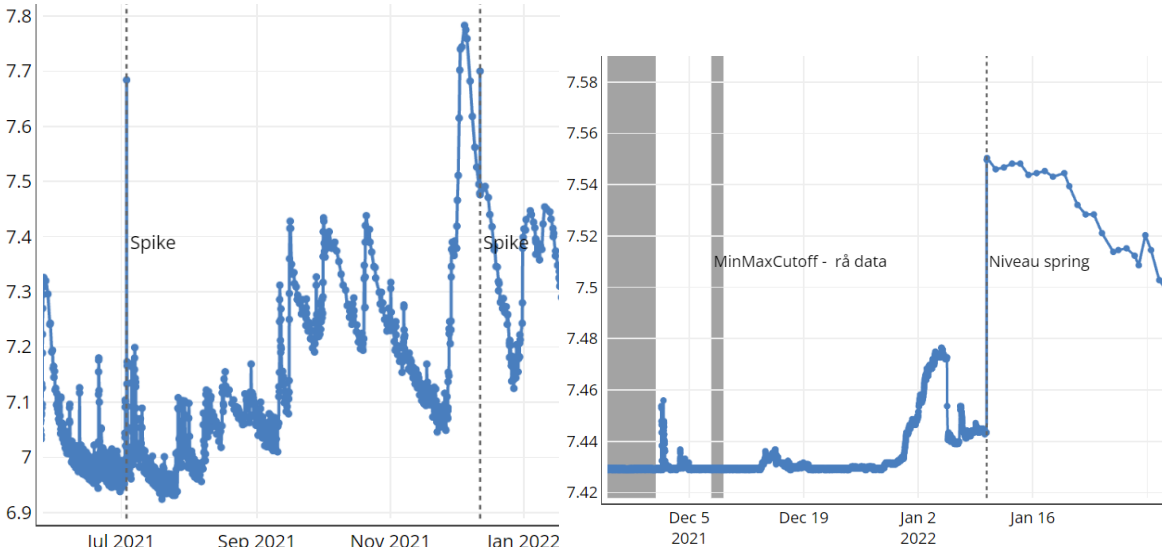
1.3 Konkrete tiltag

Der er opbygget en pipeline for udvikling og deployment af algoritmer, som allerede er løftet over i den webapplikation vi anvender til daglig drift - (sensor.watsonc.dk). Den kører primært i lambdafunktioner på AWS. Pipelinen er modulær og der er løbende udvikling af nye algoritmer, som skal sættes i søen. Det vil altså sige at vi allerede har et funktionelt kvalitetssikringssystem, inklusive notifikationssystem, hvor fejl bliver fremhævet, som kan udvides løbende. Et udsnit af den geografiske oversigt på notifikationssystemet kan ses i figur 1.6. Eksempler på fejludpegninger kan ses i figur 1.7.



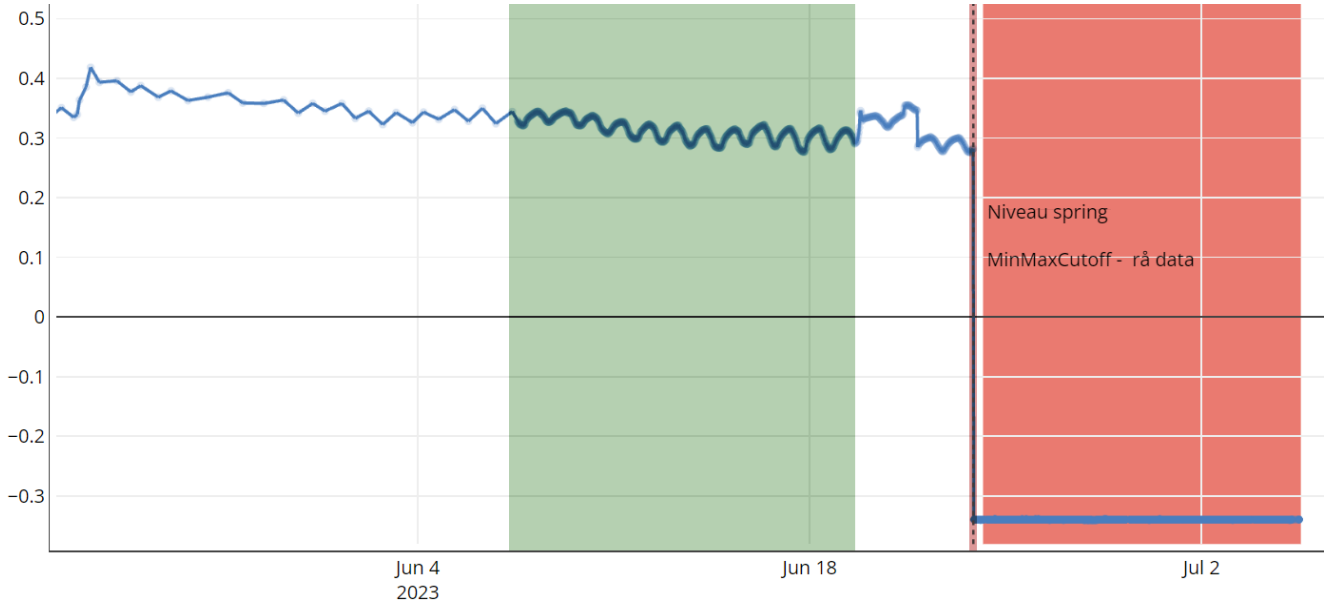
Figur 1.6 Notifikationsmodul - sensor.watsonc.dk. Eventuelle problemer i tidsserien fremhæves, hvorefter en ekspert kan danne sig et overblik og håndtere hver tidsserie.

Vi er ikke kommet til at algoritmer skal danne grundlag for en automatisk korrektion endnu, dertil kræver det stadig mere annotering og træning af nye algoritmer. Indtil videre markerer vi hændelser og lader eksperter foretage korrektionen. Den iterative optimering af algoritmer er klargjort, men ikke sat i gang endnu, da vi stadig ønsker et øget data grundlag.



Figur 1-1.7 Eksempler på automatiske fejludpegninger i tidsserier

Derudover er annoteringsværktøjet funktionelt og processen med at skabe datagrundlag er i gang. På figur 1.8 kan man se annoterede niveauspring og sensor out-of-range, samt et område som blev fremhævet, men var korrekt.



Figur 1-.28 Annoteringsværktøj – eksempler på fremhævning af dele af en tidsserie, rødt = anomalier i tidsserie, grønt = tidsserie ok.

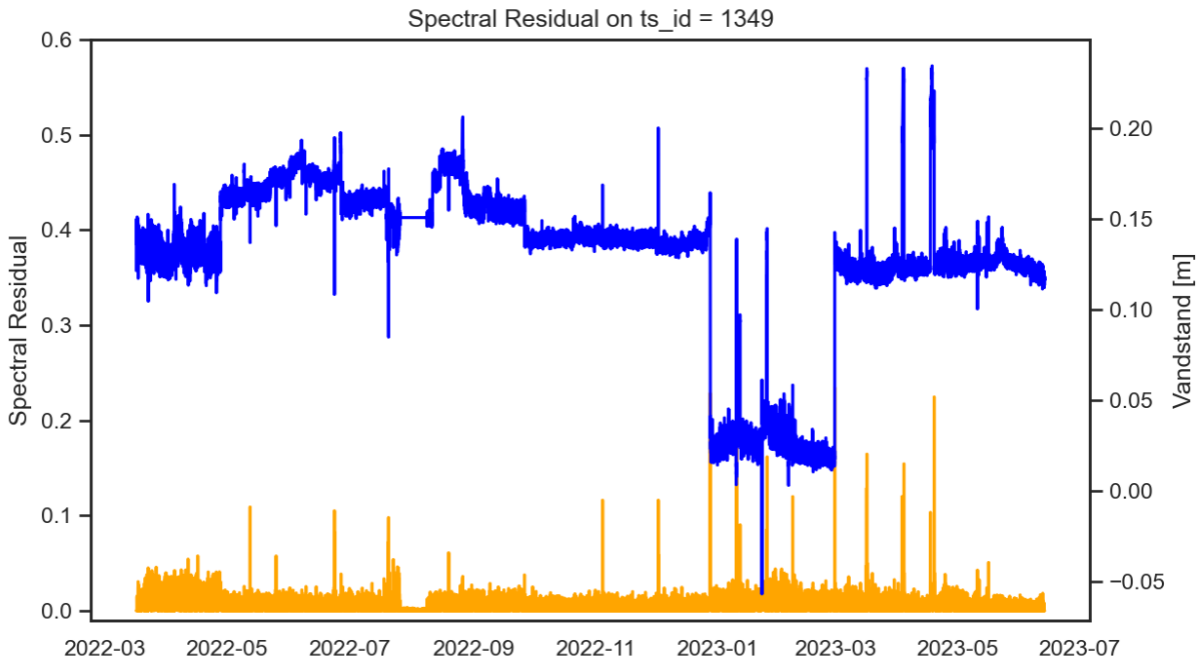
Status for typefejl:

Sensor out-of-range - Implementeret
Niveauspring - Implementeret (2 versioner)
Abnormale hændelse (En bredere kategori, som skal specificeres) - Implementeret
Spikes - Implementeret
Støj - Under udvikling
Huludfyldning - ikke implementeret
Barometer effekt - ikke implementeret

Der er lavet en notebook her: Notebook pdf , hvor der foretages en gennemgang af algoritmernes resultater. Deri kan man finde flere eksempler, både positive og negative, og figurer med enkelte hændelser. Koblet med en kort gennemgang af algoritmerne nedenfor giver det et godt indblik i funktionaliteten og der er referencer til videre læsning.

1.3.1 Spektral Residual baserede algoritmer – niveauspring og anomalier

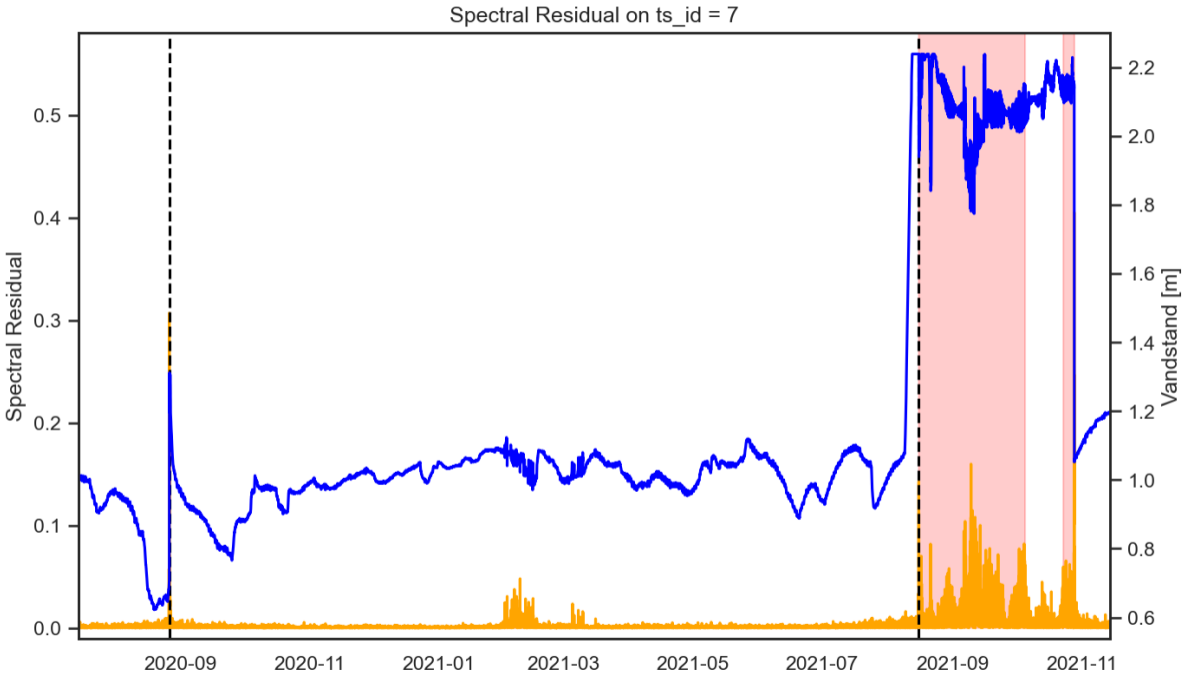
Spektral residual (SR) er en unsupervised teknik baseret på en Fourier transformation (fft), som har udmærket sig i det man kalder "visual saliency detektion", dvs. Den er god til at identificere hændelser som er "fremtrædne"[1], hvilket minder om, hvordan mennesker detekterer fejl. Overordnet udføres følgende skridt:
1) FFT for at få et log amplitudespektrum (frekvens), 2) udregning af SR, som er log amplitudespektrum minus et gennemsnitsfilter, 3) Invers Fourier Transform for at komme tilbage til det spatiale domæne.
SR kan anses for at være en komprimeret repræsentation af det originale signal, hvor afvigelser forstærkes. Derfor er det lettere at sætte en brugbar grænseværdi for at identificere outliers. Vi sætter en dynamisk defineret grænseværdi, baseret på gennemsnit og standardafvigelse, som har fastlagte min og maks værdier.



Figur 1.9 Et eksempel på outputtet af SR-beregningen, blå = originalt signal, gul =SR

[1] Time-Series Anomaly Detection Service at Microsoft

Herefter udvælges kandidathændelser baseret på indholdet i SR til enten at være en abnormalitet eller et niveauspring, afhængige af de respektive grænseværdier. Alle kandidathændelser gennemgås herefter af et postprocesseringsscript, som filtrerer hændelser fra og præciserer lokationen af korrekte hændelser, baseret på indhold omkring hændelsen.



Figur 1.10 SR output efter hændelses detektion og sortering. Sort stiplet = niveauspring, rød = anomali.

På sigt ønsker vi at eksperimentere med spektral signalet som input til en ML algoritme fremfor det originale tidsserie signal eller alternativt som supplerende input.

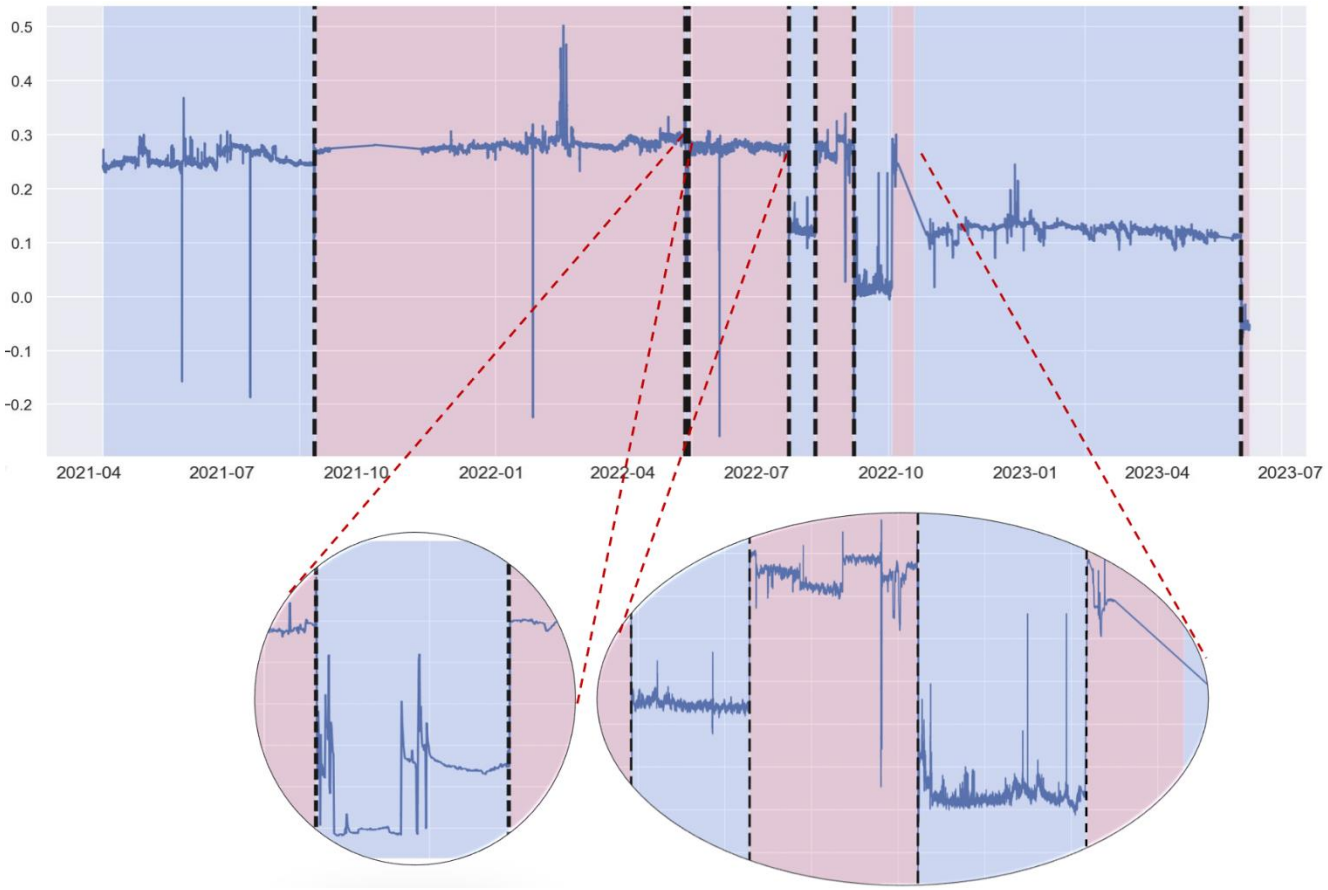
1.3.2 Change Point detektion (PELT-normal og Window-baseret)

Den næste implementerede unsupervised algoritme er baseret på konceptet Change Point detektion, hvori hypotesen er at der vil være punkter i tidsserier, hvor de statistiske mål vil skifte signifikant og derfor kan tidsserien inddeles på baggrund af disse. Ideen herfor er at identificere niveauspring i tidsserier, altså unaturlige forskydninger, som vi skal være opmærksomme på.

To versioner er implementeret, Pruned Exact Linear Time (PELT) og en version baseret på et vindue som køres hen over tidsserien.[2]
PELT er en udbygning af optimal partitioning[3], som grundlæggende forsøger at minimere en kost funktion og gør dette ved at kigge på det sidste changepoint i tidsserien og arbejde rekursivt gennem tidsserien hvor den sammenligner værdien for det optimale changepoint før inklusion af et nyt punkt med omkostningen af inklusionen af et nyt punkt og ved at minimerer værdien af kost funktionen vælger om punktet skal inkluderes. Kost funktionen i vores tilfælde er summen af segmenternes kvadrerede afvigelser (least squares deviation).

[2] Optimal detection of changepoints with a linear computational cost

[3] An Algorithm for Optimal Partitioning of Data on an Interval

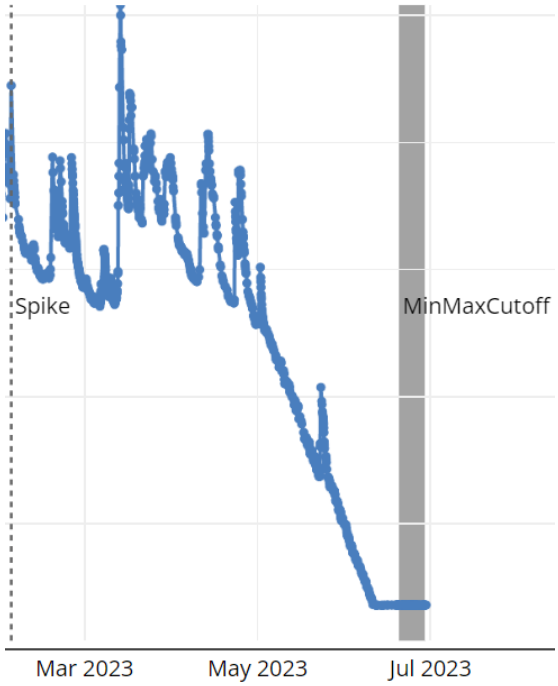


Figur 1.11 Resultatet af change point detektion – opdelte områder på basis af change points skiftevis rød og blå, stiplede linjer = registrerede niveauspring efter post processering

På begge metoder foretages der efterfølgende en post processering for at sortere yderligere i de fundne kandidater, eftersom Change Point detektion er en mere generel tilgang til at finde ændringer i signaler. Eksempelvis frasorteres det sidste change point i den højre fremhævning i figur 1.11.

1.3.3 Min-Max cutoff

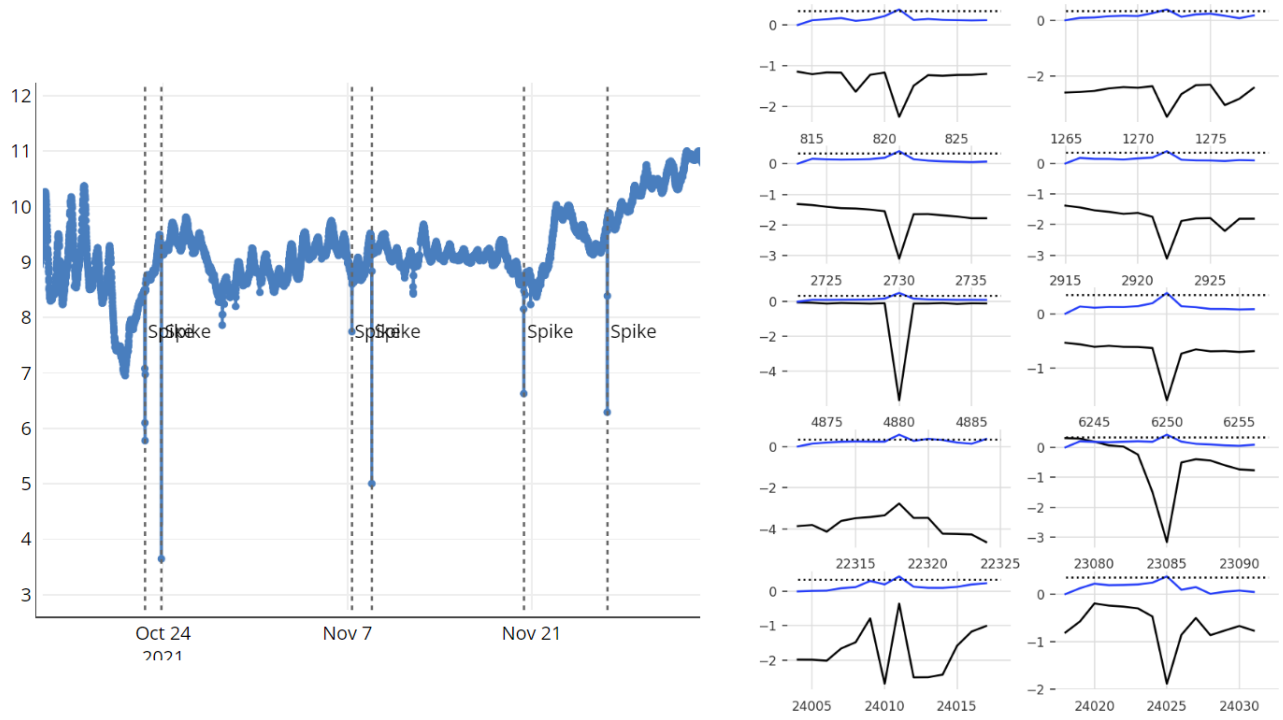
Min-Max cutoff omhandler at kunne detektere når sensorer flatliner fordi de rammer eller overskrider deres range. Dette er noget vi gerne vil reagere hurtigt på, da resultatet af sådan en hændelse er manglende data. Vores egne sensorer, hvor vi kender range bruger vi det som udgangspunkt til at undersøge alle punkter omkring eller over de værdier. For sensorer med en range, som vi ikke har i databasen, sensorer vi har overtaget eller lignende, kigger vi efter plateauer i dataen. Plateauer inkluderer små udsving, da det kan forekomme når sensoren rammer min eller maks.
Hvert plateau bliver vurderet på baggrund af varighed og indhold relativ til resten af tidsserien og bliver enten sorteret fra eller flaget.



Figur 1.12 Tidsseries sensor har ramt en minimumsværdi og bliver flaget til inspektion

1.3.4 Spike detektion

Spike detektion er implementeret som en lokal maksimum/minimum detektor, som sammenligner punkter med deres naboer og noterer om der er en signifikant ændring. Samtidig noteres det, hvor der er store forskelle i signalet selv samt signalet ift. en filtreret version. Punkter hvor der findes lokale maksima og minima og der er en stor forskel sammenlignet med det globale signal, udpeges som spikekandidater. Herefter analyserer vi Spektral Residualer omkring kandidatpunkter og frasorterer alle som ikke har en høj centreret aktivering (til højre på figur 1.13).



Figur 1.13 Til venstre eksempler på detekterede spikes, til højre et udsnit af spikes =sort, deres SR = blå og en cut-off linje = stiplet

1.4 Perspektivering

Vi fortsætter vores arbejde med at øge kvaliteten af vores egen data og den data vi bidrager til nationale databaser. Perspektiverne i det videre arbejde med kvalitetssikring af tidsseriedata er store set fra vores synspunkt. Skalerbarheden i vores forretning afhænger i høj grad af, at vi kan levere data af høj kvalitet til konkurrencedygtige priser. Vurderingen er, at priserne på IoT-sensorer (hardware) vil falde betydeligt og hvis kundernes pris også skal falde markant, så skal prisen på datahåndtering og kvalitetssikring minimeres ved automatisere kvalitetssikringen mest muligt. Så vi mener området er helt afgørende for vores fremtid som virksomhed der leverer data til vandsektoren.

For aftagerne af data bliver det også vigtigere og vigtigere at kunne afgøre om kvaliteten af data understøtter de analyser og beslutninger man ønsker at bygge ovenpå. Mængden af data vil vokse markant og det som fremad kan blive udfordringen, er at finde det rigtige data af den rigtige kvalitet.

Vi mener ikke der bør stilles generelle nøjagtighedskrav til data, som gøres offentlige via Danmarks Miljøportals services, men at der bør stilles krav til, at der medfølger en beskrivelse af graden af kvalitetssikring og gerne metadata, som beskriver detaljer om hvilke procedurer der er anvendt til kvalitetssikring.

Der skal laves et arbejde for at identificere områder (Noget af det arbejde er foretaget af Niras' undersøgelse), hvor den massive mængde data vi har til rådighed, kan være til gavn og hvordan flere aktører kan blive klar over, hvor vigtigt det er at levere gode data til fælles brug. Specielt kunne et samarbejde mellem SMV'er, kommuner og database ejere være gavnligt, så man kan blive enige om god praksis og undgå datasilo'er ved hver SMV. Man kan ikke tvinge virksomheder til at gøre deres data tilgængelig, men man kan opfordre til det. Det er nemmere at understrege vigtigheden, hvis man har helt konkrete cases at præsentere.