Norges miljø- og biovitenskapelige universitet

Masteroppgave 2021 30 stp
Fakultet for realfag og teknologi

Utforskende analyse av hyperspektral data i NIR og SWIR til diskriminering av gulrust og bladflekksykdommer på hveteblad

Exploratory analysis of hyperspectral data in NIR and SWIR to discriminate yellow rust and blotch on wheat leaves

Svein Jakob Kristoffersen
Miljøfysikk og fornybar energi

Forord

Morten Lillemo foreslo sommeren 2020 å forsøke å identifisere og kvantifisere gulrustutbrudd i hveteåkre ved hjelp av dronebilder. Det ble derfor nødvendig å analysere hveteblad og sykdommen nærmere med hyperspektralt kamera for å danne grunnverket for videre arbeid, noe som resulterte i denne studien.

Jeg vil takke min veileder, Ingunn Burud, og biveileder, Sahameh Shafiee, for all hjelp til utforming, analyse og tolking av resultater, Morten Lillemo, Min lin og Andrea Ficke for all hjelp rundt innsamling av hveteblader og informasjon rundt sykdommene, samt Marija Vukovic for opplæring i bruken av hyperspektral kamera og utforming av kode nødvendig for analyse av hvetebladene.

Jeg vil takke min mor og min kjære Sanne for motivasjon og støtte.

Svein J. Kristoffersen
Trondheim, mai 2021

Sammendrag

Hyperspektrale bilder i NIR- og SWIR-området (950 nm – 2500 nm) ble fanget av hveteblad for å utforske evne til diskriminering av gulrust og bladflekksykdommer. Spektrene ble representert med ulike korrigeringer og deriverte. Gjennom PCA og kryssvalidering av tre ulike maskinlæringsmodeller, støttevektorklassifikator (SVC), tilfeldig skogklassifisator (RFC) og partial least squares-discriminant analysis (PLS-DA), ble EMSC identifisert som den beste korrigeringen for spektrene. Deretter ble sensitive bølgelengder valgt ut basert på en PCAs komponenters- og en RFCs bølgelengdevektlegging, og ble identifisert i hele spekteret med størst vekt på vannovertonene sentrert rundt 1450nm og 1930nm. SVC trent på hele spekteret oppnådde en nøyaktighet på $92.0 \pm 2.7\%$, RFC trent på 10 sensitive bånd oppnådde $90.7 \pm 2.2\%$, mens RFC trent på 8 sensitive bånd utenfor vannovertonene oppnådde $88.9 \pm 2.7\%$. Det ble ikke identifisert noen direkte tilknytning til kjente absorpsjonsområder til nitrogen i NIR eller SWIR, selv om noen sensitive bølgelengder lå nært disse. Vannrelaterte bølgelengder ble studert i mer detalj i et aquagram og det ble oppdaget store forskjeller i absorpsjon mellom sunt og sykt blad, og for sunt blad og blotch chlorosis ble bølgelengder knyttet til C1-C3 identifisert som viktige. NIR/SWIR-området virker egnet for sykdomsdiskriminering og burde brukes sammen med VIS, men det er nødvendig med videre forskning på både planter og sykdommene, deres bestanddelers unike absorpsjonsmønstre, samt hvordan vannabsorpsjon påvirker målinger i SWIR. Framtidige studier bør inkludere romlig og temporal informasjon, i tillegg til den spektrale.

Abstract

Hyperspectral images in the NIR and SWIR range (950 nm - 2500 nm) were captured of wheat leaves to explore the ability to discriminate yellow rust and leaf spot diseases. The spectra were represented with different corrections and derivatives. Through PCA and cross-validation of three different machine learning models, support vector classifier (SVC), random forest classifier (RFC) and partial least squares-discriminant analysis (PLS-DA), EMSC was identified as the best correction for the spectra. Sensitive wavelengths were selected based on a PCA's component and an RFC's wavelength weighting (feature selection), and were identified throughout the spectrum with greatest emphasis on the water overtones centered around 1450nm and 1930nm. SVC trained on the whole spectrum achieved an accuracy of $92.0 \pm 2.7\%$, RFC trained on 10 sensitive bands achieved $90.7 \pm 2.2\%$, while RFC trained on 8 sensitive bands outside the water harmonics achieved $88.9 \pm 2.7\%$. No direct association with known absorption features of nitrogen in NIR or SWIR was identified, although some sensitive wavelengths laid close. Water-related wavelengths were studied in more detail in an aquagram and large differences in absorption between healthy and diseased leaf were detected, and for healthy leaf and blotch chlorosis, wavelengths associated with C1-C3 were identified as important. The NIR / SWIR area seems suitable for disease discrimination and should be used together with VIS, but further research is needed on both plants and the diseases, their unique absorption features and patterns of constituents, and how water absorption affects measurements in SWIR. Future studies should include spatial and temporal information, in addition to spectral.

Innhold

1 Introduksjon

Hvete har blitt benyttet som mat i over tolv tusen år og er en av de tidligst domestiserte planteartene [1]. I dag er hvete er en av verdens viktigste matforsyninger sammen med ris og mais og er en nødvendig del av kostholdet til rundt en tredjedel av verdens befolkning. Sykdommer og insektangrep på hveteplantene kan forårsake store avlingsreduksjoner som har økonomiske konsekvenser og kan i alvorlige tilfeller påvirke et lands, eller hele verdens, matproduksjon. I bekjempelsen av disse er planteforedling det beste tiltaket, men dette tar lang tid og for å raskt redde avlinger fra soppangrep eller insekter brukes fungicider eller pesticider [2]. Hyppig bruk av slike midler kan føre til utvikling av fungicidresistens hos soppene, som kan være katastrofalt for matproduksjon. Begrensing av bruk av fungicider er altså svært viktig og kan gjøres ved å identifisere soppangrep tidlig. Problemet her ligger i den fortsatt utbredte bruken av tradisjonell, visuell inspeksjon av hveteplantene for å bedømme plantehelsen. Dette må gjøres av personer med kunnskap om sykdommene, noe som tar lang tid og er subjektivt [3]. Det har i mange studier vært forsøkt å oppdage og kartlegge utbrudd av sykdommer i hveteåkre ved hjelp av fjernmåling [4, 5, 6, 7, 8] og resultatene tyder på gode muligheter for bruk av for eksempel drone. Resultatene fra studier som benytter noen få enkelt-bølgelengder (multispektral) og studier som benytter mange kontinuerlige bølgelengder (hyperspektral) er svært gode, men det er få som benytter det elektromagnetiske bølgelengdeområdet SWIR (short-wave infrared) for diskriminering av sykdommer. Denne studien tar for seg en soppsykdom vanlig i nordisk klima, gulrust [9], og utforsker evne til diskriminering av denne, bladflekksykdommer og sunt blad gjennom hyperspektrale bilder fanget i NIR- (near infrared) og SWIR-området. Det skal isoleres sensitive bølgelengder knyttet til klassifiseringen og det forsøkes å knytte disse til stoffer og bestanddeler av hveteplanten.

2 Teori

2.1 Gulrust og bladflekksykdommer

Gulrust (yellow rust/stripe rust) oppstår i hovedsak på hvete og rughvete, og er forårsaket av soppen *Puccinia striiformis* f.sp. *tritici*. som overlever på levende planter [2]. Symptomene i sykdommens tidlige fase (inkubasjon) er milde, til ikke merkbare [3], og kan forårsake klorotiske flekker på bladvevet. Bladvevet utvikler først gul-oransje sporehoper som kan opptre enkeltvis, men etter hvert også mellom bladnervene i striper. Sporehopene inneholder flere tusen sporer [5] som spres effektivt i vinden til naboplanter og til andre hveteåkre over store avstander. Hvetesorter har ulik resistans mot sykdommer, og forsøk gjennomført av NIBIO i 2019 med vanlige, norske hvetesorter viser at sorten Bjarne er minst resistent mot gulrust, etterfulgt av sorten Zebra [10]. Avlingsreduksjonene som følge av gulrustutbrudd varierer med temperatur, fuktighet, hvetesort og vekststadie, men kan bli så høye som 70-80%.

Bladflekksykdommer (blotch) er en fellesbetegnelse på sykdommene hveteaksprikk (*Parastagonospora nodorum*), hvetebladprikk (*Zymoseptoria tritici*) og hvetebrunflekk (*Pyrenophora tritici-repentis*). Disse kan overleve i dødt plantevev og framstår på sunt blad som mørkebrune, runde eller ovale flekker, med lyse kanter. Det er små forskjeller i symptomene mellom sykdommene og de opptrer ofte sammen, noe som kan gjøre dem vanskelig å skille fra hverandre. I denne studien benyttes det engelske navnet på bladflekksykdommer: blotch.

Nekrose er død av plantenes celler og forårsakes av for eksempel sykdom, mens klorose (gulning av blad) oppstår når klorofyll brytes ned i blad, som i enkelte sorter kan komme som et resultat av plantens forsvarsmekanismer mot sykdom. I denne studien benyttes noen steder de engelske navnene for nekrose og klorose: necrosis og chlorosis.

2.2 Hyperspektrale bilder

Digitale bilder består typisk av kombinasjoner av tre farger, rød, grønn og blå (RGB), som kombinert skaper farger tilnærmet slik vi ser dem. Det øyet vårt fanger opp er egentlig elektromagnetisk stråling ved ulike bølgelengder i området synlig lys fra ca. 400 - 750 nm (VIS) [11]. Fargene rød, grønn og blå ligger jevnt spredt i dette bølgelengdeintervallet og kan dermed kombineres additivt for å danne alle

fargene. Det finnes mange flere bølgelengder kortere og lengre enn de vi kan se, som kan benyttes til for eksempel røntgenbilder og nattsynsbriller, men også i radiokommunikasjon og mikrobølgeovner.

Bølgelengder kan grupperes i korte intervaller, såkalte bånd. Multispektrale bilder består av flere bånd, typisk 3-10 [12], mens hyperspektrale bilder består av flere, gjerne hundre kontinuerlige bånd. Vanlige RGB-bilder består som nevnt av tre farger, eller bånd, og danner derfor et bilde med en høyde og bredde, men også en dybde som tilsvarer de tre båndene. Bildet kan dermed sees på som en kube. På samme måte danner et hyperspektralt bilde en såkalt hyperkube, med den romlige informasjonen, den fysiske størrelsen på det avbildet, fanget av de to første dimensjonene, mens den tredje representerer alle båndene.

Elektromagnetisk stråling inneholder ulik mengde energi, som er relatert med strålingens bølgelengde [13]. Det elektromagnetiske spektrum er delt inn i kategorier, eller brede opptaksbånd (bølgelengdeintervaller), etter strålingens ulike egenskaper, og de relevant for denne studien er: synlig lys (VIS), nær-infrarød (NIR) ca. 700 - 1400 nm og kortbølget-infrarød (SWIR) ca. 1400 - 3000 nm [14]. Disse grensene, og de for andre kategoriene i det elektromagnetiske spektrum, varierer i ulike kilder. VIS består som nevnt av synlig lys og dets bånd benyttes ofte sammen med bånd fra NIR for å undersøke plantehelse. De vanligste vegetasjonsindeksene (forholdstall som skiller ulike planteegenskaper basert på refleksjonsverdiene til bølgelengder) benytter bølgelengder fra disse intervallene, for eksempel normalisert differanse-vegetasjonsindeks (NDVI), grønn-indeks (GI) og forbedret vegetasjons index (EVI). Det er derimot få vegetasjosindekser som benytter bølgelengder fra SWIR-området, og de få som gjør det bruker også VIS/NIR-området. I arbeidet knyttet til denne studien ble det funnet én vegetasjonsindeks som kun benyttet bølgelengder fra SWIR-området: bladmasse-indeks (LMA), som benytter bølgelengdene 1368 og 1722 nm for å måle hvor mye blad (gram) plantene danner per kvadratmeter, og er funnet å være positivt korrelert med pigmenter i bladene [15].

SWIR-området er sterkt knyttet til vannabsorpsjon og er derfor vanskelig å benytte i praksis da atmosfærens vanninnhold skaper forstyrrelser på målinger. Det ligger fire kjente vannabsorpsjonsområder i NIR og SWIR: to svake rundt 970 nm, 1200 nm og to sterke rundt 1450 nm og 1930 nm [16], hvor de to sistenevnte kalles første- og andre vannovertone, respektivt. Mange blad- og skogtakstudier velger å fjerne første og andre vannovertone for å unngå støy og finne sensitive bølgelengder som kan anvendes i praksis [17].

2.3 Preprosesseringsteknikker

For å korrigere for fysiske effekter relatert til lyskilde og sensor kan ulike preposesseringsmetoder benyttes. For å fjerne sensorutslag forårsaket av bakgrunnstråling kan man korrigere for kameraets mørkestrøm. Dette gjøres ved hjelp av et mørkt bilde, oppnådd ved å dekke til kameralinsen, for så å gjennomføre korrigeringen med likning

$$I_{korrigert} = I_{original} - I_{mørk}, \tag{1}$$

hvor $I_{korrigert}$ er det mørkekorrigerte bildet, $I_{original}$ er det originale bildet og $I_{mørk}$ er mørkestrømbildet (det mørke bildet). For å korrigere for endringer i lyskilden kan man bruke en hvitreferanse som har høy reflektanse. Korrigeringen på et bilde kan gjennomføres med likning

$$I_{korrigert} = \frac{I_{original}}{M_{W.R.}}, \tag{2}$$

hvor $I_{korrigert}$ er det korrigerte bildet, $I_{original}$ er det originale, mens $M_{W.R.}$ er median-spekteret til strålingen fra hvitreferansen. Spredning av strålingen grunnet ulike størrelser på partikler er uønsket. Dette kan korrigeres på flere måter og to vanlige er standard normal variat (SNV) og multiplicative scatter correction (MSC) eller extended multiplicative scatter correction (EMSC) når modellen inkluderer ledd av høyere grad. SNV korreksjonen gjennomføres for hver enkelt spektrum (piksel) med likning

$$S_{SNV} = \frac{S - \mu_S}{\sigma_S}, \tag{3}$$

hvor S_{SNV} er det korrigerte spekteret, S er det originale, mens μ_S og σ_S er spektrumets gjennomsnitt og standardavvik. EMSC er en utvidelse av MSC som er en modell basert på Lambert–Beers

lov [18]. Modellen til MSC uttrykker absorbansespektrum som summen av et referansespektrum med et multiplikativt ledd, et konstant ledd og et residualledd. EMSC utvider denne modellen med muligheten til å representere ulineære grunnlinjer (baselines) ved å introdusere flere ledd bestående av en konstant multiplisert med spektrene opphøyd i høyere orden. Modellen for EMSC av andre orden kan dermed beskrives som

$$A = x \cdot b_1 + w \cdot b_2 + w^2 \cdot b_3 + a + e, \tag{4}$$

hvor A er absorbansspekteret, x er et referansespektrum som er mest mulig uten støy og forstyrrelser og kan oppnås som gjennomsnittet av alle spektrum, a er det additive leddet, e er residualleddet mens $b_{1,2,3}$ er konstanter som tilnærmes gjennom minste kvadraters metode. Når konstantene er tilnærmet kan korreksjonen for hvert spektrum gjennomføres med likning

$$A_{korr} = \frac{A - w \cdot b_2 - w^2 \cdot b_3 - a}{b_1}, \tag{5}$$

hvor residualleddet neglisjeres. Utnyttelse av 1., 2. og høyere ordens deriverte kan tydeliggjøre et spektrums mer komplekse informasjon og gjøre det lettere for modeller å oppdage mønstre. 1. deriverte kan for eksempel nøytralisere alle grunnlinjeforskyvinger (baselineshifts) i spektrum forårsaket av lyskildevariasjon, noe som ofte er ønskelig, men som også betyr at derivasjon fører til tap av informasjon [19]. Beregning av neste ordens deriverte gjøres iterativt langsmed spektrenes x-akse med likning

$$\frac{\Delta y}{\Delta x}_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \tag{6}$$

hvor $\frac{\Delta y}{\Delta x}_i$ er den deriverte (stigningstall) for et datapunkt, y_i er reflektansen og x_i er datapunktets bånd. Datapunktene er $i = 1, 2, 3...N$, hvor $N = 288$. Dette kan gjøres for hvert punkt på grafen bortsett fra det siste som dermed resulterer i en forkortelse av grafen med ett datapunkt for hver orden deriverte. Likning (7) benyttes flere ganger for å beregne høyere ordens deriverte. Det antas at grafens punker er jevnt fordelt med konstant avstand $x_{i+1} - x_i = 1$ så likning (6) forenkles til

$$\frac{\Delta y}{\Delta x}_i = y_{i+1} - y_i. \tag{7}$$

Et derivert spektrum inneholder mer støy jo høyere orden det er, noe som kan reduseres med et Savitzky-Golay filter (S-G filter). Dette filteret glatter ut grafer ved å ta for seg en liten del av de om gangen (vindu) for deretter å tilpasse et polynom av bestemt grad til denne delen gjennom minste kvadraters metode.

En enkelt ikke-veiledet gruppering av data er K-means, som benytter et bestemt antall sentroider som datapunkter blir tilnærmet mot ved hjelp av kvadratsum, for så å oppdatere sentroidenes posisjon. Algoritmen gjennomføres flere ganger og stopper når endringen i sentroidenes posisjon ikke endres mer (eller endringen er tilstrekkelig liten). Algoritmen er relativt enkel og kan være utsatt for avvik i data (outliers).

Det velges ofte å omdanne refleksjonsspektrum til absorpsjonsspektrum, noe som kan gjøres med likning

$$A = log_{10}(\frac{1}{R}), \tag{8}$$

hvor A er absorpsjon og R er refleksjon.

2.4 Veiledet og ikke-veiledet komponentanalyse

Hyperspektral data inneholder mange egenskaper, eller attributter (features), i form av bølgelengder eller bånd, og det er ofte ønskelig å redusere dette antallet for å unngå støy som kan minke modellers nøyaktighet. Kombinasjonen av få prøver (her som spektrum) og mange egenskaper kan føre til Hughes fenomen, som innebærer en minking i modeller nøyaktighet når mengden egenskaper blir for stor [17]. En ofte brukt multivariat dimensjonsreduserende teknikk er prinsipal komponent analyse (PCA). Denne baseres på prinsipalkomponenter som fanger variasjon i data og dermed kan erstatte

de originale egenskapene ved å kartlegge disse ned på komponentene. Prinsipialkomponentene har som egenskap at de er matematisk ortogonale vektorer, det vil si ukorrelerte, og dannes ved å benytte egenvektorene i kovariansmatrisen (co-variance matrix). Prinsipialkomponentene blir altså valgt ut som den retningen i data med mest variasjon, hvor den første komponenten fanger mest variasjon, den andre fanger mest variasjon i data i en retning ortogonal til den første komponenten, osv. Antall komponenter kan være opptil antall egenskaper i originaldata og fanger mer variasjon jo flere komponenter som brukes, men det er ofte kun nødvendig med noen få komponenter for å fange mesteparten av variasjonen. Lineær diskriminant analyse (LDA) er en tilsynelatende veldig liknende dimensjonsreduserende teknikk, men skiller seg fra PCA ved at dens komponenter har som mål å finne den vektleggingen av egenskaper som skaper mest variasjon mellom klassene og minst variasjon innad i klassene [20], og den kan benyttes som en klassifiseringsmodell. LDA benytter altså klassene for å skille data og er dermed en veiledet, eller overvåket (supervised), analysemetode. PCA benytter i motsetning ikke klassene, altså ikke-veiledet (unsupervised) og har derfor større potensial til å oppdage usette sammenhenger i data. LDA baserer seg på antagelsene om at hver klasse er gaussianfordelt og at hver egenskap har lik varianse.

PCA og LDA brukes altså ofte til å transformere data lineært ned på komponenter som fanger mye av variasjonen. Man kan dermed visualisere data i rommet dannet av komponentene som akser. Her velges det å visualisere data kartlegget på to komponenter om gangen gjennom et 2D bilde, eller plan, av alle datapunktene med komponentene som akser. Dette vil bli referert til som et komponentrom her. Identifiseres det tydelige grupper eller klynger av en enkelt klasse i komponentrommet vil de komponentene som rommet består av inneholde viktig informasjon for å kunne skille klassene. Dermed kan man undersøke komponentenes vektlegging av de originale egenskapene for å identifisere viktige bølgelengder knyttet til diskriminering av klasser.

2.5 Bekreftende klassifisering

I denne studien benyttes flere ulike maskinlæringsmodeller for å klassifisere og finne sensitive bølgelengder i data. Partial least squares-discriminant analysis (PLS-DA), en variant av partial least squares regression (PLSR), er et multivariat dimensjonsreduserende verktøy og kan benyttes til egenskap-vektlegging (feature selection) og klassifisering [21]. I motsetning til PLSR, som anvendes for kontinuerlige klasser, benyttes PLS-DA for kategoriske klasser. PLS-DA danner, i likhet med PCA, ortogonale prinsipalkomponenter (latente variabler) gjennom lineær regresjon ved hjelp av klassene (veiledet), men har som mål å fange så mye kovarians som mulig. PLS-DA har stor evne til å benytte data med høyt korrelerte egenskaper og støy, slik som spektral data.

Støttevektorklassifikator (support vector classifier, SVC) er en klassifiseringsmetode fra metodegruppen støttevektormaskiner (support vector machines, SVM-er). Dette er maskinlæringsmetoder som baseres på statistisk læring og dimensjonsteori og er velegnet til å håndtere datasett med mange egenskaper og få prøver [22]. SVC har som mål å skille data lineært, ved hjelp av det som kalles støttevektorer (linjer, plan eller hyperplan), men kan også håndtere ulineære problemer ved å benytte ulike kjernefunksjoner som kan transformere egenskapene til en høyere dimensjon hvor klassene kan skilles lineært. Dette kalles kjernetrikset (kernel trick), en metode som transformerer egenskapene i høyere dimensjoner uten å eksplisitt vite transformasjonen [23], noe som er beregningsmessig effektivt. En lineær kjerne finner lineære løsninger på problemet, en polynomisk kjerne finner ulineære løsninger med en polynom av bestemt grad (en økning i egenskapdimensjonene tilsvarende polynomgraden), mens en radial basis kjerne (radial basis kernel, RBF) finner løsninger på problemet i et uendelig antall dimensjoner.

Tilfeldig skogklassifikator (random forest classifier, RFC) er en ensemble av mange beslutningstrær (decision trees) som alle forsøker å klassifisere (eller regressere) mindre deler av totaldata [24]. Beslutningstrær får navnet fra deres struktur som likner på et opp-ned tre med noder (nodes) og grener (branches), hvor hver node tilsvarer en beslutning som splitter data i to eller flere grener som representerer attributter. Treet velger noder fra data basert på statiske måleenheter som for eksempel informasjonsgevinst (information gain), entropi eller Gini indeks, og plasserer hver node i treet basert på dens viktighet for klassifiseringen, med de viktigste nodene øverst (først). Ved klassifisering sendes data fra rot-noden, gjennom alle nodene og grenene og ender til slutt opp som en klasse. RFC benytter mange slike beslutningstrær som hver trenes på en liten, tilfeldig del av

data og deres hyppigste klassifisering settes som sluttklassifiseringen.

Før man trener en modell må dens hyperparametre bestemmes. Dette er bestemte egenskaper ved modellen som bestemmer hvordan treningen skal foregå og er unike for hver modelltype og hvert datasett. Dette kan være for eksempel være hvor mange iterasjoner en modell skal trenes eller hvor stor del av treningsdata som skal brukes for hver iterasjon (batch-size). For å finne optimale hyperparametre for en modell kan man benytte GridSearch, et verktøy i scikit-learn pakken [25]. Dette er en algoritme som finner en modells beste parametre for et gitt datasett og gitte hyperparameters intervaller, ved å teste alle mulige kombinasjoner av de gitte verdiene for hyperparametrene.

For å validere modellene kan kryssvalidering benyttes. Dette innebærer å dele et datasett i et bestemt antall like store deler, bruke én av delene som valideringssett og resten som treningssett, lagre modellens nøyaktighet (eller andre statistiske måleenheter) basert på enkeltvalideringen med valideringsettet og deretter gjenta dette med neste del av data som valideringssett og de andre som treningssett. Gjennomsnittet av alle nøyaktighetene velges som den beste representasjonen av modellens nøyaktighet.

2.6 Aquagram

Aquafotonikk er en relativt ny måte å analysere stoffer med mye vanninnhold gjennom deres absorpsjonsspekter i VIS/NIR-området [26] [27]. Mye av absorpsjonen i SWIR skylles som sagt vannmolekyler, som naturligvis inneholder hydrogenbindinger, og disse blir igjen påvirket på ulike måter av andre stoffer, som for eksempel ved tilstedeværelsen av et metallion som vannmolekyler danner et hydreringsskall (hydration shell) rundt. På denne måten kan absorpsjonen til vann og hydrogenbindinger reflektere forstyrrelser forårsaket av kjemiske eller fysiske årsaker og kan dermed være en rask og effektiv måte å oppnå molekylær informasjon. Bølgelengder som gjennom multivariat analyse tilknyttes bestemte påvirkninger på vannmolekyler uavhengig av forstyrrelsen kalles vann-matrise koordinater (water matrix coodinates) eller WAMACS, og sammensettingen av disse til spesifikke spektrale mønstre kalles vannabsorpsjonsmønstre (water absorbance pattern) eller WAPS. Ved hjelp av WAMACS og WAPS jobber aquafotonikk med oppbyggingen av en database av informasjon tilknyttet spesifikke system som kan benyttes til bedret forståelse og videre forskning. De 12 WAMACS i SWIR-området vises i tabell 1 og for å tolke data gjennom disse framstilles de i et aquagram, en stjernegraf (star-chart), med normaliserte absorpsjonsverdier. Aquagram har blitt benyttet i flere studier, blant annet for å bedømme om en hunn-panda var i brunst basert på endringene i WAMACS av dens urin [28]. En studie som benyttet aquafotonikk, SVM og NIR spektrum av blod for å identifisere tidlige symptomer på diabetes type 2 fant gjennom aquagrammet et tydelig forskjell i antall hydrogenbånd og fremmer dette som gode biomarkører for tidlig diagnose [22].

Tabell 1: Bølgelengdeintervallene til kjente WAMACS og deres tilknytning til vannmolekyler, samt de bølgelengdene benyttet for hver enkelt WAMACS i denne studien. I tilknytningene er følgende: v_1: symmetrisk strekking av den første overtonen av vann, v_2: bøying av første overtone i vann, v_3: asymmetrisk strekking av første overtone av vann og $S_{0-4} : (H_2O)_{1-5}$. Alle intervall og tilknyttinger hentet fra [22]

WAMACS	Bølgelengde-intervall [nm]	Benyttet bølgelengde [nm]	Tilknytning
C1	1336-1348	1345	v_3
C2	1360-1366	1367	Hydreringsskall (Hydration shell)
C3	1370-1376	1372	$v_1 + v_3$
C4	1380-1390	1383	Hydreringsskall (Hydration shell)
C5	1398-1418	1399	S_0
C6	1420-1428	1410	Vann hydrering (Water hydration)
C7	1434-1444	1437	S_1
C8	1448-1454	1443	$v_1 + v_3$
C9	1460-1468	1465	S_2
C10	1472-1472	1475	S_3
C11	1482-1495	1492	S_4
C12	1506-1516	1519	Sterkt bundet vann eller v_2

3 Metode

3.1 Eksperimentell metode

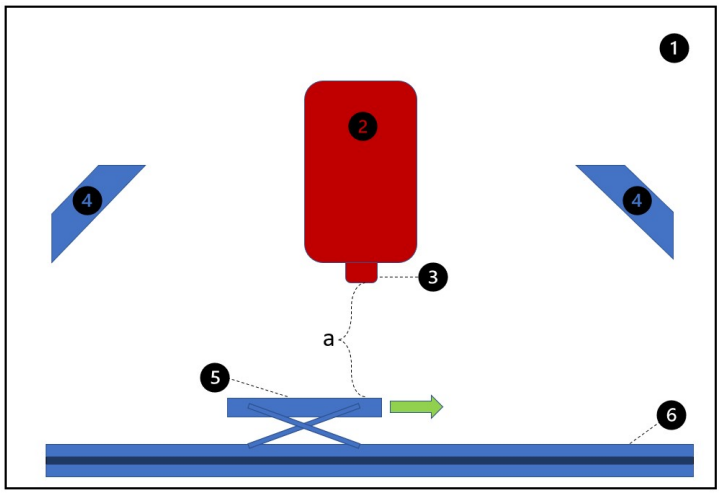
3.1.1 Oppsett av hyperspektralt kamera

Stativ med kameraoppsettet er plassert på et bord i bilderom med tilhørende PC. Kameraoppsettet vises i figur 1 (PC ikke vist) og består av stativ, hyperspektralt kamera, halogenpærer, metallskinne, platform og avbildningsplate. Stativet er drapert med lystette stoffgardiner og hyperspektralt kamera er av typen HySpex SWIR-384 montert med 30 cm linse. Metallskinnen er sammen med kamera koblet til PC som gjennom programvare forflytter platform med avbildningsplate ettersom kamera tar bilder (push-broom). Platformen er høydejusterbar.

3.1.2 Innsamling og fotografering av blader

Hveteplantene benyttet til analysen vokser på et forskningsfelt (koordinater: 59.67216643795494, 10.77089840268966) knyttet til NIBIO i Ås kommune, en del av Viken fylke, Norge. Hveteplantene var av flere sorter, deriblant Zebra, Seniorita og Bjarne, og ble sådd 16.04.2020. Spirer inokulert med gulrust ble plantet 28.05.2020.

Totalt 35 hveteblad ble plukket 16.07.2020 (vekststadie 75-80) fra hvetesorten Bjarne og plasseres i tre lukkede plastposer etter klassene gulrust (13 blad), blotch (9 blad) og sunt blad (13 blad). Det ble også plukket et testsett den 23.07.2020 (vekststadie 82-87) fra hvetesorten Zebra, bestående av 9 blad, 3 fra hver klasse. Bladene ble fraktet til bildelab for hyperspektral bildefanging rett etter plukking, hvor de ble festet til A4 papir med klebemasse på blad-ende, nummerert og fotografert med mobilkamera for RGB-bilde. Utvalgte RGB-bilder tatt før hypserspektral bildefangst vises i figur 2. Et test-blad ble plassert på avbildningsplaten (se kameraoppsett figur 1) og benyttet for å kalibrere kameraets integrasjonstid (lukkerhastighet) og avstand mellom kamera-aperture og avbildningsplate (a i figur 1). Deretter ble alle bladene flyttet forsiktig over til avbildningsplate etter tur og fotografert sammen med hvitreferanse med hyperspektralt kamera. All håndtering av blader foregikk med latekshansker. Tid fra plukking til bildetaking lå mellom én time til én og en halv time basert på hvilken klasse bladene tilhørte. Gulrustbladene ble fotografert først, deretter sunne og blotch-bladene. Halogenpærene førte til høy temperatur inni kamerastativet, men grad av tørke i bladene var tilsynelatende lav, bortsett fra langs noen bladers kanter hvor det oppstod krumming.



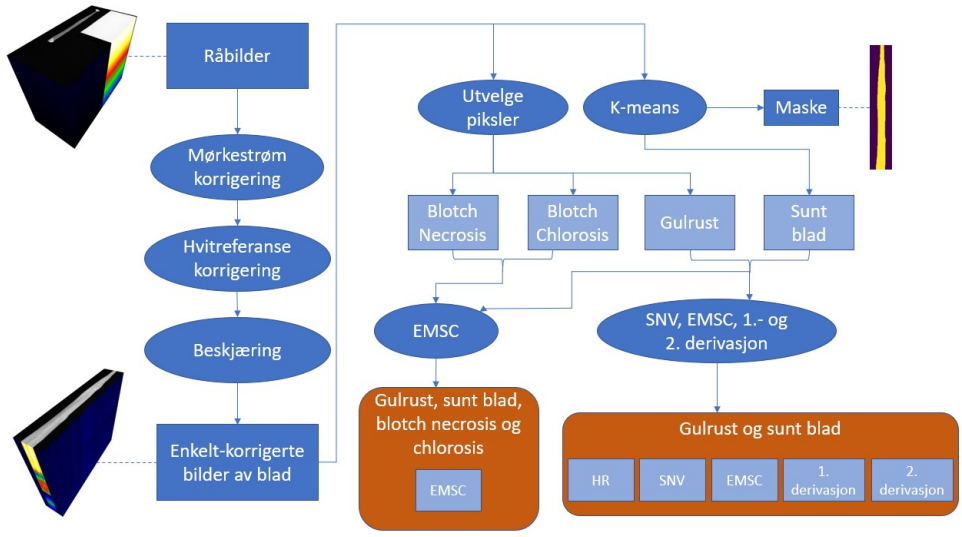
Figur 1: Oppsett for hyperspektralt kamera. 1: stativ med lystette stoffgardiner, 2: hyperspektralt kamera, 3: kamera-aperture, 4: halogenpærer, 5: forflyttende avbildningsplate på platform, 6: metallskinne, a: avstand mellom avbildningsplate og kamera-aperture

(a) Sunne blad. **(b)** Blad infisert med gulrust. **(c)** Blad infisert med blotch.

Figur 2: RGB-bilder av utvalg av bladene før hyperspektral bildetaking.

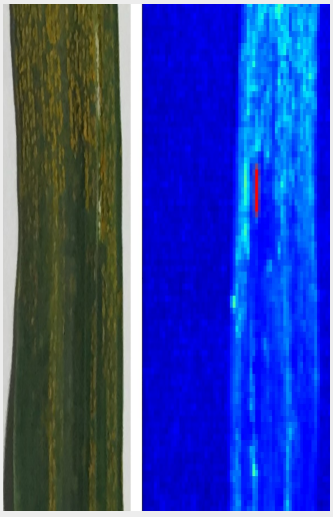
3.1.3 Preprosessering

Framgangsmåte for preprosesseringen vises skjematisk i figur 3. Alle prosesser etter mørkestrømkorrigeringen ble gjennomført med Python 3.8 i Pycharm 2020.1.5, all kode vises i vedlegg 3. Råbildene ble automatisk korrigert for mørkestrøm gjennom det hyperspektrale kameraets programvare og likning (1). Deretter korrigert med hvitreferansen etter likning (2) hvor $M_{W.R}$ er medianspektrumet til et beskjært rektangulært område innenfor hvitreferansen. Bildet ble så beskjært slik at hvitreferanse og klebemasse på blad-endene utelates og bildet kun inneholdt bladet.

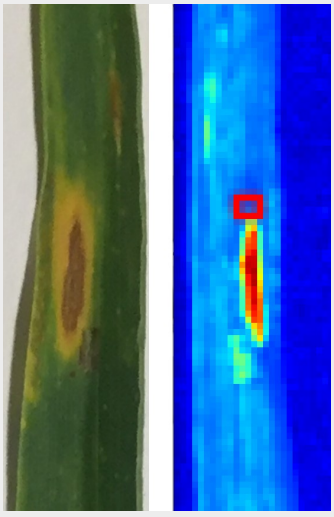


Figur 3: Framgangsmåte for preprosesseringen. Råbildene korrigeres for mørkestrøm, med hvitreferansen og beskjæres, deretter plukkes klasse-pikslene ut ved hjelp av enten K-means (sunt blad) eller valgt fra bildene for hånd. Alle klassepiksler korrigeres deretter og ender opp i to grupper markert med oransje: én gruppe med kun gulrust og sunt blad-piksler korrigert etter alle datatypene HR-korrigert, SNV-korrigert, EMSC og 1. og 2. derivasjon, og én gruppe med alle klassene gulrust, sunt blad, blotch necrosis og blotch chlorosis korrigert med EMSC. Spektrum i den første gruppen benyttes i metode 1, mens den andre gruppen benyttes i metode 2.

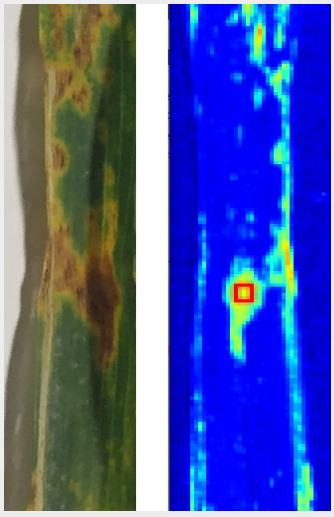
Det ble så valgt ut områder (rektangler som innrammet en mengde piksler) på bladene som skulle representere de ulike klassene rust, blotch chlorosis og blotch necrosis. Eksempel på framgang for utvelging vises i figur 4. For klassen sunt blad velges alle pikslene i de sunne bladene, noe som oppnås med to-gruppes K-means gruppering. Denne grupperingen benyttes også for å danne en maske av bladene. Av alle pikslene var 82 fra gulrust, 104 blotch necrosis, 100 blotch chlorosis og 100 sunne.



(a) Områdeutvelging av gulrust. Markert område fanger en tynn stripe gulrust.



(b) Områdeutvelging av blotch chlorosis. Markert område ligger innenfor et tydelig chlorosis-område.



(c) Områdeutvelging av blotch necrosis. Markert område ligger innenfor et tydelig mørkt blotch necrosis-område

Figur 4: Utvelging av piksel-områder som skal representere klassene. Hver underfigur er med to bilder; venstre: RGB-foto av blad med tydelige klasse-områder, høyre: gjennomsnitt av utvalgte bånd på hyperkuben som belyser ønskede områder godt. Områder markert eller omkretset av rødt rektangel viser det utvalgte området.

Antall piksler tilhørende blotch chlorosis og sunt blad ble nedsamplet fra henholdsvis 205 og 134640 piksler for å bevare balansen i datasettet. Det bør tydeliggjøres at en piksel tilsvarer et spektrum, og at benevningene brukes mye om hverandre her. Spektrene til klassene gulrust og sunt blad ble korrigert med SNV med likning (3), EMSC-prosessert med likning (5) og endret til 1. og 2. deriverte med likning (7), mens klassene blotch necrosis og blotch chlorosis kun ble EMSC-prosessert. For referansespekteret nødvendig for EMSC-prosessering ble gjennomsnittsspekteret til de originale 134640 sunne blad spektra brukt. Dette referansespekteret blir også benyttet for EMSC-prosessering av spektrum i testsettet i klassifiseringsdelen for å unngå informasjonslekasje. Derivasjonsdataen ble glattet ut gjennom et S-G filter. Figur 5 viser alle spektrene til klassene gulrust og sunt blad korrigert til alle datatypene, sammen med klassene blotch necrosis og blotch chlorosis prosessert med EMSC. Variasjonen i data er stor for HR-korrigerte spektrum, mens den er lavere for SNV-korrigerte og EMSC-prosesserte. Derivasjonsdata inneholder tilsynelatende mye støy, selv etter glatting med S-G filter. Overlapping mellom klassene er stor for derivasjons- og EMSC-prosessert data, mens den for HR- og SNV-korrigert data er lavere og tilsynelatende enklere å klassifisere.

3.2 Metode for analyse

3.2.1 Metode 1: utforskende analyse

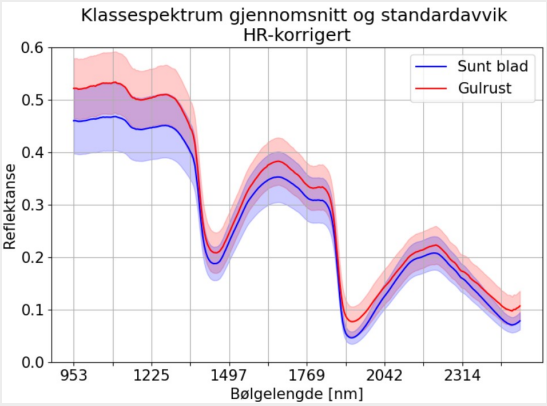
Skjematisk framgangsmåte vises i figur 6. For klassene gulrust og sunt blad blir alle datatypene analysert etter tur gjennom PCA (5 komponenter), som hentes fra python pakken scikit-learn [25]. Dataene plottes i komponentrommene for å identifisere evne til gruppering. Deretter trenes og valideres de tre modellene, SVC, PLS-DA og RFC med de ulike datatypene i tillegg til PCA transformert data. Alle modellene hentes fra scikit-learn pakken, med unntak av PLS-DA, som er modifisert med utgangspunkt i scikit-learn sin PLSR-modell (regressjonsmodell), for å kunne håndtere kategorisk klassifisering. Treningen gjennomføres med fem gangers kryssvalidering, hvorav valideringssettet står for 20% av datsettet for hver fold. Kryssvalideringen gir hver modell trent på hver datatype en nøyaktighet og standardavvik for nøyaktigheten.

Til slutt trenes modellene på hele datasettet for deretter å klassifisere piksler på bilder av blader utenfor datasettet. Dette skal representere praktisk testing av modellene. Her består testsettet av bilder som hyperkuber og modellene klassifiserer hvert enkelt spektrum tilhørende hver enkelt piksel på bildet. Siden modellen som benyttes til klassifisering er trent med en preprosessert datatype, sendes spektrene som skal klassifiseres gjennom den samme preprosesseringen.

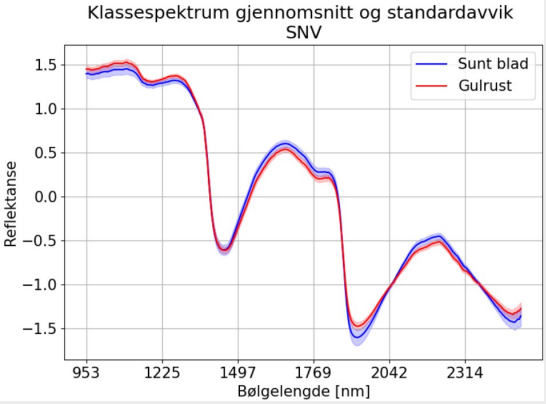
3.2.2 Metode 2: sensitive bølgelengder og full klassifisering

Skjematisk framgansmåte vises i figur 7. Her benyttes alle fire klassene representert gjennom EMSC-prosessert data. PCA, LDA og trening av en RFC-modell blir gjennomført og det undersøkes hvilke bølgelengder som er sensitive ovenfor klassifiseringen. For PCA og LDA blir data plottet i komponentrommene og grad av gruppering bestemmer hvor mye hver enkelt komponents bølgelengdevektlegging skal prioriteres. Basert på dette velges det ut sensitive bølgelengder.

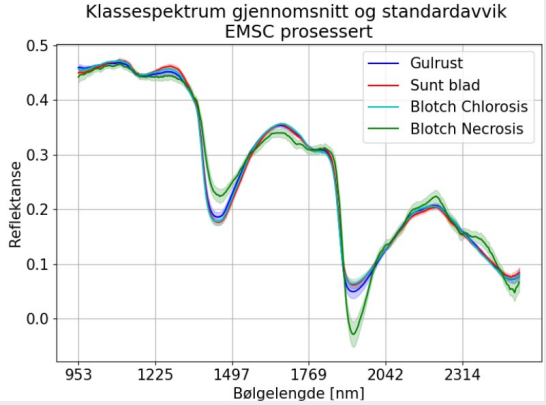
Modeller blir først trent på hele spektrum og deretter utvalgte sensitive bølgelengder for å sammenlikne klassifiseringsevne. Sensitive bølgelender brukes også til å trene modeller som benyttes til å klassifisere piksler på bilder av bladene. Til slutt undersøkes de viktigste båndene rundt bølgelengder knyttet til absorpsjon av vann ved hjelp av aquagram.



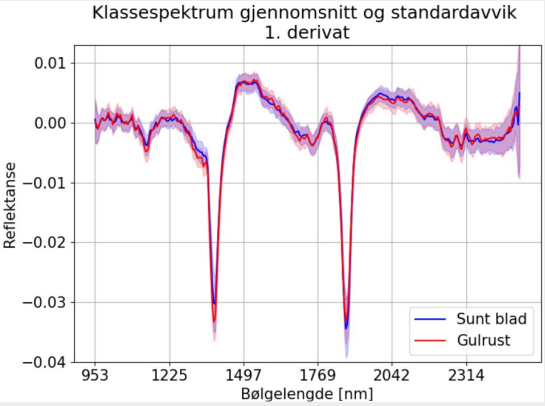
(a) HR-korrigert spektrum.



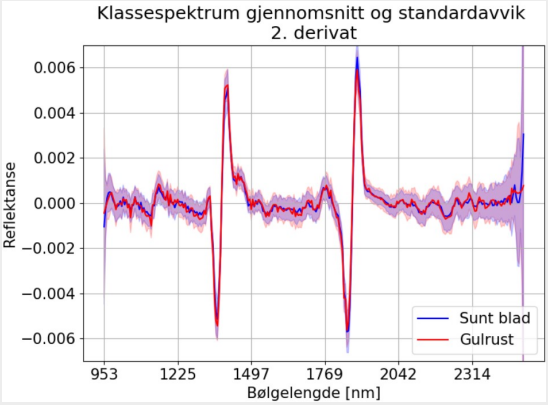
(b) SNV beregnet med likning (3).



(c) Alle klassespektrene prosessert med EMSC beregnet med likning (5).

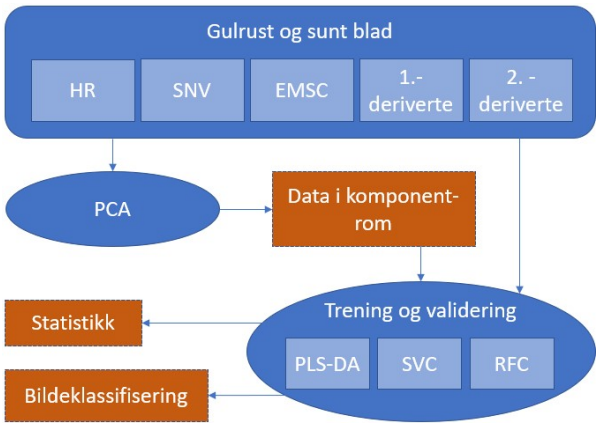


(d) 1. deriverte beregnet med likning (7) og glattet ut med et SG-filter med vindusstørrelse 7 og polyorder 5

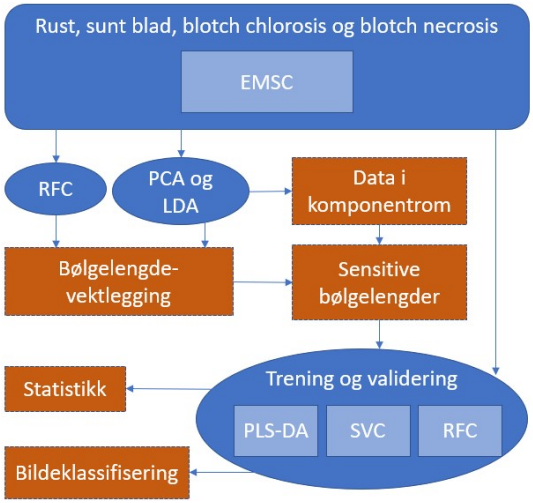


(e) 2. deriverte beregnet med likning (7) og glattet ut med et SG-filter med vindusstørrelse 11 og polyorder 3

Figur 5: Spektrene til klassene gulrust og sunt blad korrigert med HR, SNV, EMSC-prosessert og endret til 1. og 2. derivasjon, sammen med klassene blotch necrosis og blotch chlorosis prosessert med EMSC. Standardavvik vises som fargeregion med tydelig gjennomsnittslinje i midten.



Figur 6: Framgangsmåte for metode 1: utforskende analyse. Klassene gulrust og sunt blad representert gjennom fem datatyper (HR-korrigert, SNV-korrigert, EMSC-prosessert og 1. og 2. deriverte) og analyseres gjennom PCA og plottes i analysens komponentrom. De fem datatypene og data fra komponentrommene benyttes etter tur til å trene og validere PLS-DA, SVC og RFC. Valideringen gir statistikk og modellene benyttes videre til bildeklassifisering. Trinn markert med oransje er framstilt som resultater.



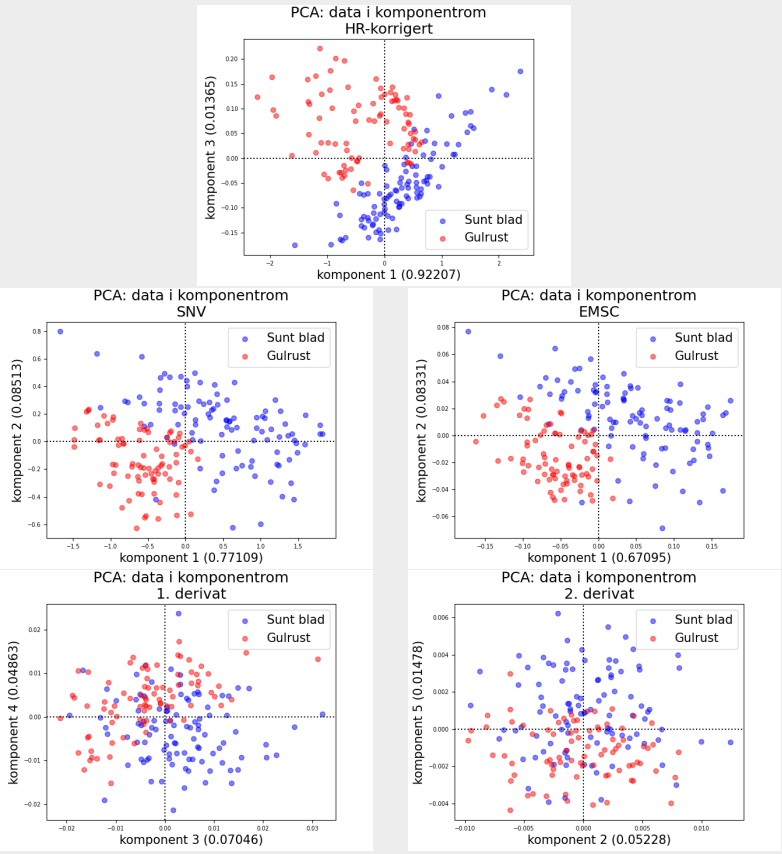
Figur 7: Framgangsmåte for metode 2: sensitive bølgelengder og full klassifisering. Klassene gulrust, sunt blad, blotch chlorosis og blotch necrosis representeres som EMSC-prosessert data og analyseres gjennom PCA, LDA og trenes på en RFC. Fra disse fås en bølgelengdevektlegging og data kan plottes i analysenes komponentrom. Fra dette velges det ut sensitive bølgelengder som benyttes til å trene og validere PLS-DA, SVC og RFC. Valideringen gir statistikk og modellene benyttes videre til bildeklassifisering. Trinn markert med oransje framstilles som resultater.

4 Resultater

4.1 Metode 1: utforskende analyse

4.1.1 PCA

Klassene gulrust og sunt blad ble for hver datatype analysert gjennom PCA med fem komponenter for å undersøke evne til klassegruppering. Data plottet i alle komponentrommene vises i vedlegg 1, mens figur 8 viser noen utvalgte komponentrom. Her kommer det fram at 1. og 2. derivasjons-spektrum analysert gjennom PCA har lav evne til klassegruppering. Mengde variasjon fanget av en enkeltkomponent er høyest for komponent 1 fra PCA trent på HR-korrigert spektrum, hvor den står for 92.2% av den totale variasjonen. Den er lavere for komponent 1 fra PCA trent på SNV-korrigert og EMSC-prosessert spektrum, med henholdsvis 77.1% og 67.1%.



Figur 8: Data plottet i utvalgte komponentrom til PCA trent på datatypene. Det ble valgt ut de komponentrommene som viste best gruppering for hver datatype.

4.1.2 Modellvalidering og pikselklassifisering

Alle datatypene ble etter tur brukt til å trene og validere modellene SVC, PLS-DA og RFC med 5 gangers kryssvalidering. For å finne hver modells beste parametre for hver datatype ble GridSearch benyttet. Resultatet vises i tabell 2. Modeller trent på 1. og 2. derivasjons spektrum oppnådde lavere nøyaktighet enn andre datatyper. De to beste modellene var de med høyest nøyaktighet: RFC trent på PCA transformert HR-korrigerte spektrum, og den med beste korrigeringen med høy nøyaktighet og lavt standardavvik: SNV trent på EMSC-prosessert spektrum. Av modellene var det SVC som presterte best, og dens beste kernel for alle datatyper, bortsett fra PCA transformerte, var polynomisk og tydet på at klassifiseringsproblemet var ulineært. De store fordelene rundt korrigering av data og EMSC-prosesseringens høyere diskrimineringsevne enn SNV-korrigering resulterte i beslutningen om å kun benytte EMSC-prosessert data i metode 2.

Tabell 2: Modellene SVC, PLS-DA og RFC sine valideringsresultater basert på fem gangers kryssvalidering når trent på datatypene HR-korrigert, SNV, EMSC, 1. deriverte, 2. deriverte og PCA transformert data. Modellenes beste hyperparametre ble funnet gjennom GridSearch. Statistikken viser nøyaktighet og standardavvik i prosent.

Datatype	**SVC**	**PLS-DA**	**RFC**
HR-korrigert	$95.6 \pm 6.5\%$	$93.4 \pm 6.2\%$	$88.4 \pm 8.3\%$
SNV	$89.0 \pm 4.1\%$	$85.8 \pm 5.7\%$	$91.8 \pm 5.6\%$
EMSC	$95.0 \pm 5.1\%$	$88.4 \pm 4.8\%$	$94.0 \pm 2.1\%$
1. deriverte	$80.3 \pm 9.8\%$	$82.0 \pm 7.3\%$	$77.0 \pm 9.2\%$
2. deriverte	$71.6 \pm 10.0\%$	$70.0 \pm 7.0\%$	$70.4 \pm 3.1\%$
Trans. HR-korrigert	$92.8 \pm 7.3\%$	$92.8 \pm 6.0\%$	$97.3 \pm 3.5\%$
Trans. SNV	$94.6 \pm 4.5\%$	$94.5 \pm 4.5\%$	$91.9 \pm 8.2\%$
Trans. EMSC	$93.4 \pm 5.1\%$	$93.9 \pm 2.8\%$	$90.7 \pm 7.0\%$
Trans. 1. deriverte	$68.6 \pm 3.4\%$	$72.5 \pm 6.1\%$	$69.2 \pm 6.3\%$
Trans 2. deriverte	$63.7 \pm 11.4\%$	$65.4 \pm 3.9\%$	$61.5 \pm 6.0\%$

Effekten av å transformere data gjennom PCA med fem komponenter vises også i tabell 2. Her kommer det fram at transformasjon av 1. og 2. derivasjonsdata resulterte i en lavere nøyaktighet. Transformasjon av EMSC-prosessert data gav høyere nøyaktighet for PLS-DA, men ikke SVC og RFC. Transformasjon av SNV-korrigert data resulterte i høyere nøyaktighet for alle modeller, mens det for original HR-korrigert data gav lavere nøyaktighet for SVC og PLS-DA, men den høyeste nøyaktigheten totalt for RFC på $97.3 \pm 3.5\%$.

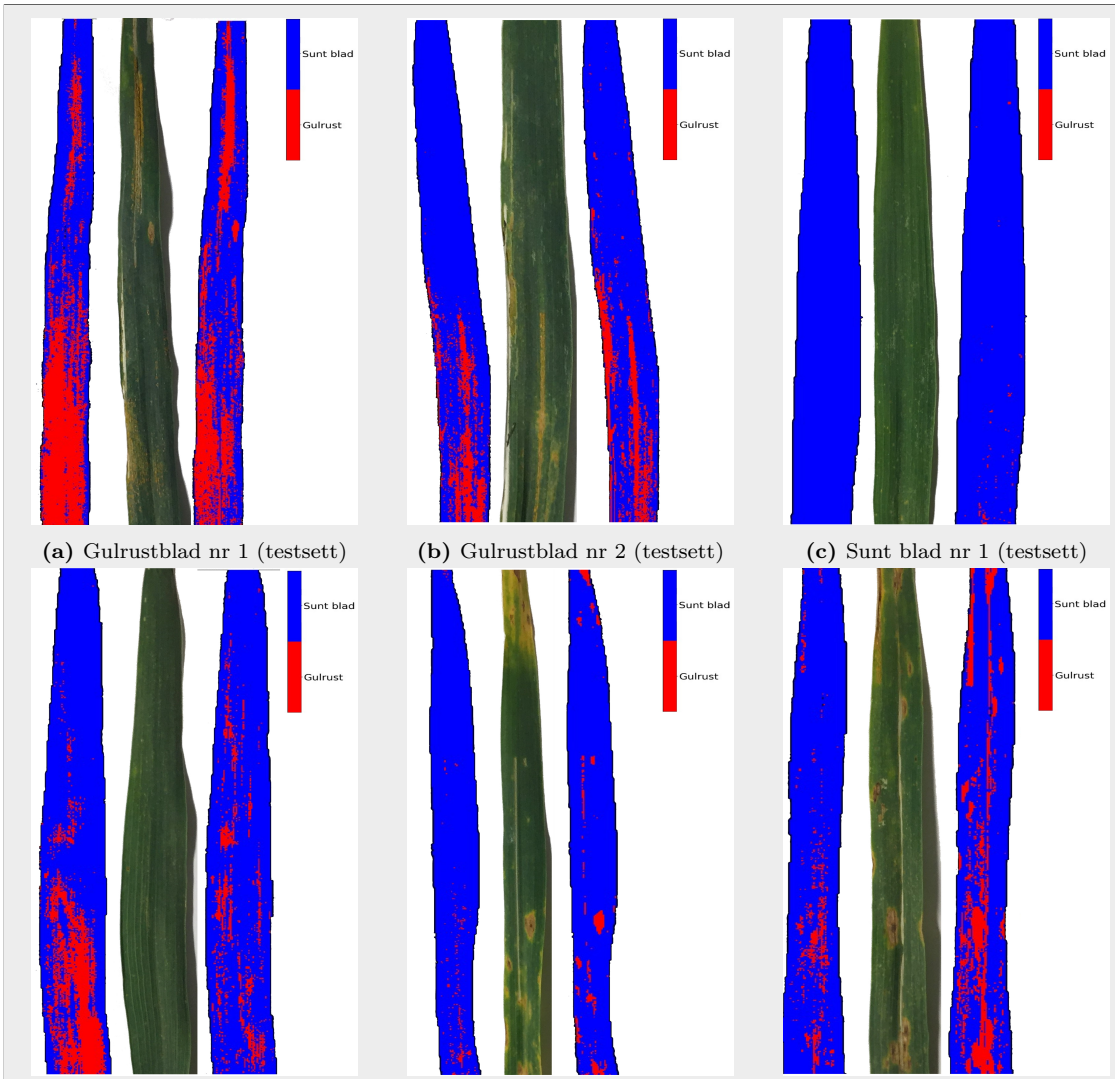
De to beste modellene, SVC og RFC, trent med sin beste datatype, henholdsvis EMSC-prosessert og HR-korrigert + PCA-transformert, ble videre brukt til å klassifisere piksler på bilder i testsettet. Resultatene vises i figur 9. For gulrustblad nr. 1, figur 9a, er klassifiseringen god, men den feilklassifiserer noen tilsynelatende sunne og blotch-piksler som gulrust. Det samme gjelder for gulrustblad nr. 2, figur 9b, hvor noen chlorosispiksler feilklassifiseres som gulrust. Sunt blad nr. 1, figur 9c, har god klassifisering, mens den for sunt blad nr. 2, figur 9d, gjenkjenner flere tilsynelatende sunne piksler som gulrust for begge modellene. På begge blotch-bladene, figur 9e og 9f, klassifiseres blotch necrosis og noen chlorosis piksler som gulrust. Det er tydelig fra resultatene at disse modellene ikke kan skille mellom gulrust, chlorosis og blotch necrosis. RFC klassifiserer flere blotch- og necrosispiksler som gulrust enn SVC, i alle blader bortsett fra sunt blad nr 2.

4.2 Metode 2: sensitive bølgelengder og full klassifisering

4.2.1 Dataforenklende analyser og bølgelengdevektlegging

PCA og LDA ble gjennomført for alle fire klassene representert gjennom EMSC-prosessert data. Komponentrommene til komponent 1-3 for PCA vises i figur 10 sammen med alle komponentrom for LDA. PCA skiller enkelt ut blotch necrosis, men har en del overlapp mellom blotch necrosis, gulrust og sunt blad. LDA skiller alle klassene effektivt, slik man forventer, og de viktigste komponentene er 2 og 3.

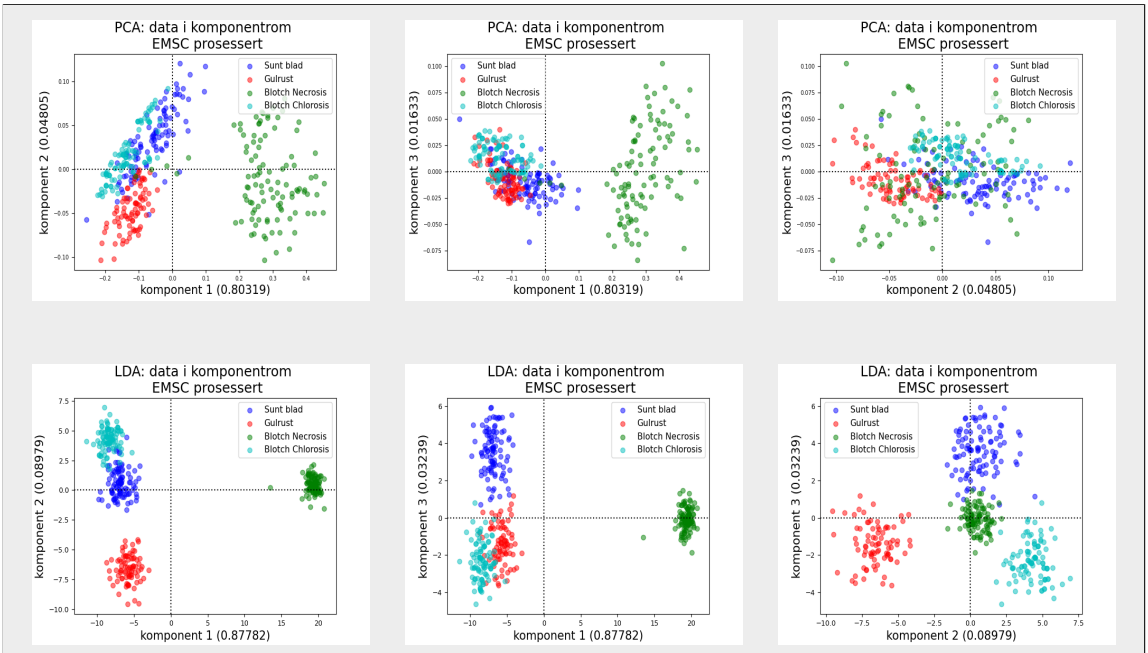
Komponent 1-3 sine vektlegginger av bølgelengder for PCA og de for LDA, vises øverst og nederst i figur 11. LDA vektlegger første overtone og området rundt mest, mens PCA har litt høyere og mer varierende vektlegging av andre overtone. Siden vannovertonene har stor betydning, men ofte ikke kan benyttes i praksis, ble det gjennomført enda en PCA trent på samme data, men kun på bølgelengder utenfor vannovertonene. Disse utvalgte bølgelengdene er markert som lilla områder med ulik fargegradering øverst i figur 11. Dens komponenters bølgelengdevektlegging vises i midten i figur 11 og vektlegger bølgelengder jevnt over alle områder. Bølgelengdevektleggingen til LDA-komponentene var originalt veldig oscillerende og det ble nødvendig å benytte et S-G filter, noe som endret den originale vektleggingen i mistenkelig høy grad, resultatet vises nederst i figur 11. Den resulterende vektleggingen til LDA-komponentene blir derfor ikke benyttet til utvelging av sensitive bølgelengder.



(a) Gulrustblad nr 1 (testsett) **(b)** Gulrustblad nr 2 (testsett) **(c)** Sunt blad nr 1 (testsett)

(d) Sunt blad nr 2 (testsett) **(e)** blotch-blad nr 1 (testsett) **(f)** blotch-blad nr 2 (testsett)

Figur 9: Bildeklassifisering av testsett med SVC og RFC. Figurene a-f viser klassifisering av bladene gulrust 1 og 2, sunt blad 1 og 2 og blotch-blad 1 og 2 fra testsettet. På hver figur er bladets RGB-bilde i midten, til venstre er pikselklassifiseringen til SVC trent på EMSC-prosessert data og til høyre er den til RFC trent på PCA transformert HR-korrigert data.



Figur 10: Data plottet i utvalgte komponentrom til PCA og LDA trent på EMSC-prosessert data. Det ble valgt ut de komponentrommene som viste best gruppering for hver datatype.

En RFC ble også trent på all data og dens innebygde algoritme egenskap-viktighet (Feature importance) ble benyttet for å finne dens vektlegging av bølgelengder, vist i figur 12. Igjen ble vannovertonene vektlagt mye, og det ble derfor nødvendig å trene enda en RFC på spektrum med bølgelengder utenfor disse. RFC har mest vektlegging av andre overtone, og når bølgelengdene i vannovertonene ekskluderes vektlegger RFC området mellom overtonene ca. 1650 nm til 1730 nm som svært viktig, mens de lengste bølgelengdene over 2014 nm vektlegges lite.

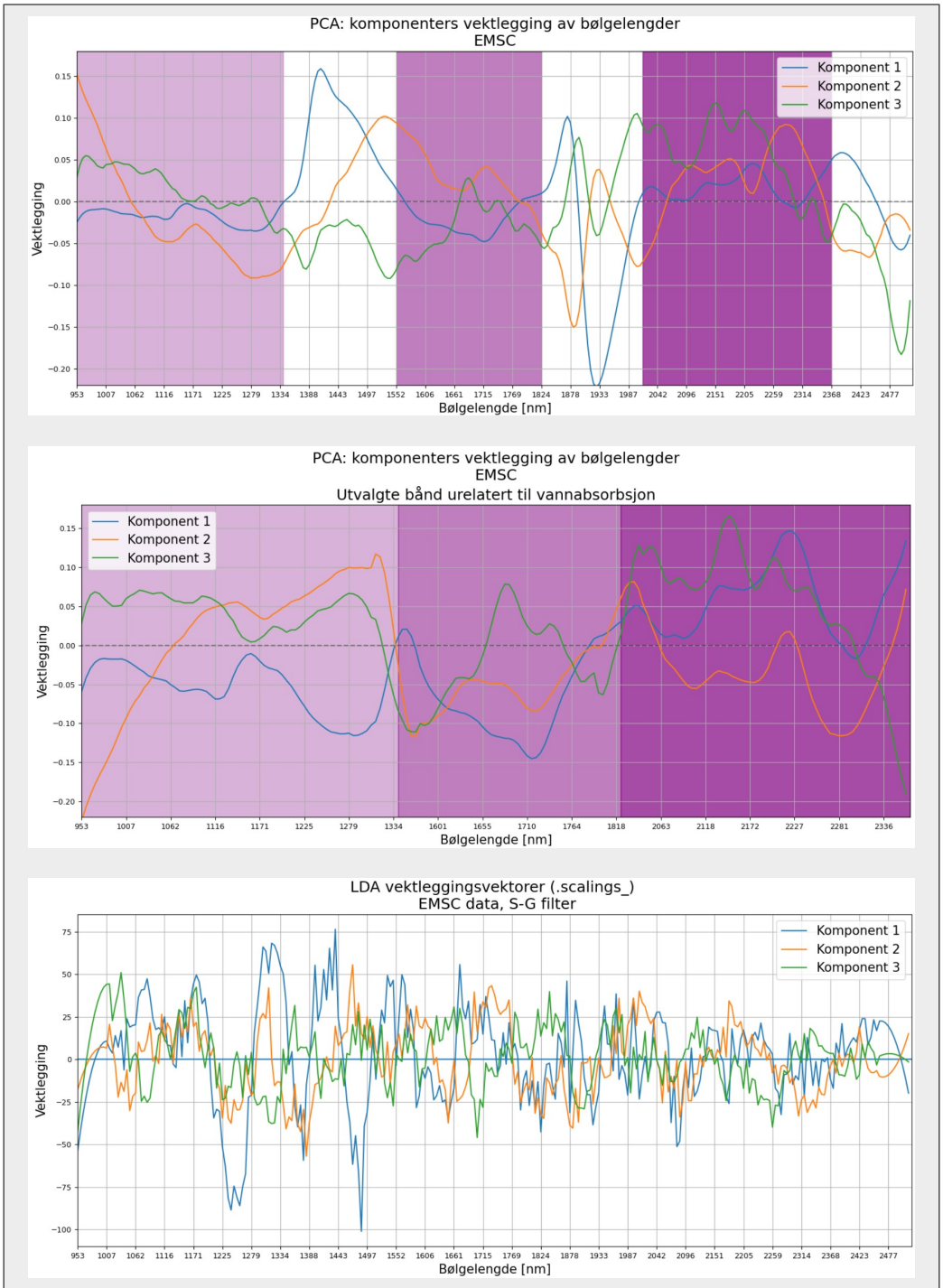
4.2.2 Sensitive bølgelengder

Basert på bølgelengdevektleggingen til PCA og RFC, både med og uten vannabsorpsjonsbåndene, ble det valgt ut fire sett med sensitive bølgelengder som vist i tabell 3. Påliteligheten til bølgelengdevektleggingen til LDA sees på som lav og brukes derfor ikke til å produsere et sett med sensitive bølgelengder, men dens mest vektlagte bølgelengder kan bemerkes, rundt 1030 nm og 1470 nm (komponent 2 og 3). Sett 1, basert på PCA trent på data med alle bånd, inneholder 8 bånd fra de høyeste toppene til dens komponenters bølgelengdevektlegging øverst i figur 11. Sett 2 består av 9 bånd fra vektleggingen til figuren i midten, mens sett 3 og 4 består av henholdsvis 10 og 8 bånd valgt basert på grafene i figur 12.

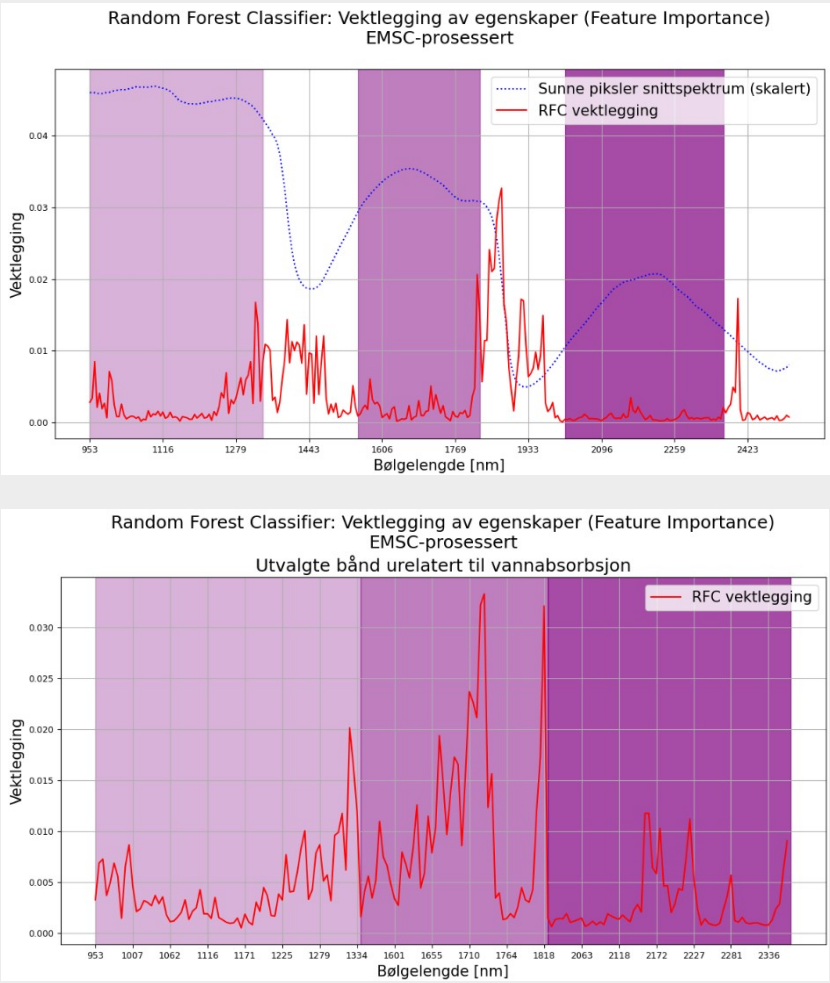
Noen bemerkelser rundt settene bør gjøres. De basert på PCA inneholder bølgelengder jevnere fordelt på spektrumet enn de basert på RFC. Sett 3 basert på RFC har majoriteten av bølgelengdene fokusert rundt første og andre vannovertone, mens sett 4 basert på RFC trent på bølgelengder utenfor vannovertonene inneholder flere bånd som ligger nært vannovertonene, i hovedsak opp mot grensene 1330 nm og 1824 nm. Sensitive bølgelengder valgt på grunnlag av PCA-vektlegging har flere lengre bølgelengder over 2000 nm enn de for RFC.

4.2.3 Modellvalidering og pikselklassifisering

De sensitive bølgelengdene ble benyttet til trening og validering av modellene SVC, PLS-DA og RFC. Resultatet vises i tabell 4 og er oppgitt som nøyaktighet og dens standardavvik. Siden problemet er fler-klasses burde statistikken også innebære begrepene presisjon, sensitivitet og F1-score, men det ble oppdaget at disse verdiene var tilnærmet like verdiene for nøyaktighet, noe som sammen med den aksepterende balansen i datasettet resulterte i utelatelsen av disse verdiene i tabell 4.



Figur 11: Vektlegging av bølgelengder for PCA og LDA. Øverst: PCA trent på alle bølgelengder, fargede områder tilhører bølgelengder ikke knyttet til vannabsorpsjon. Midten: PCA trent på bølgelengder innenfor fargede områder. Nederst: LDA trent på alle bølgelengder med de resulterende vektleggingsgrafene glattet ut med S-G filter.



Figur 12: Vektlegging av bølgelengder for to RFC-modeller. Øverst: RFC trent på alle bølgelengdene, nederst: RFC trent på bølgelengder utenfor vannovertonene

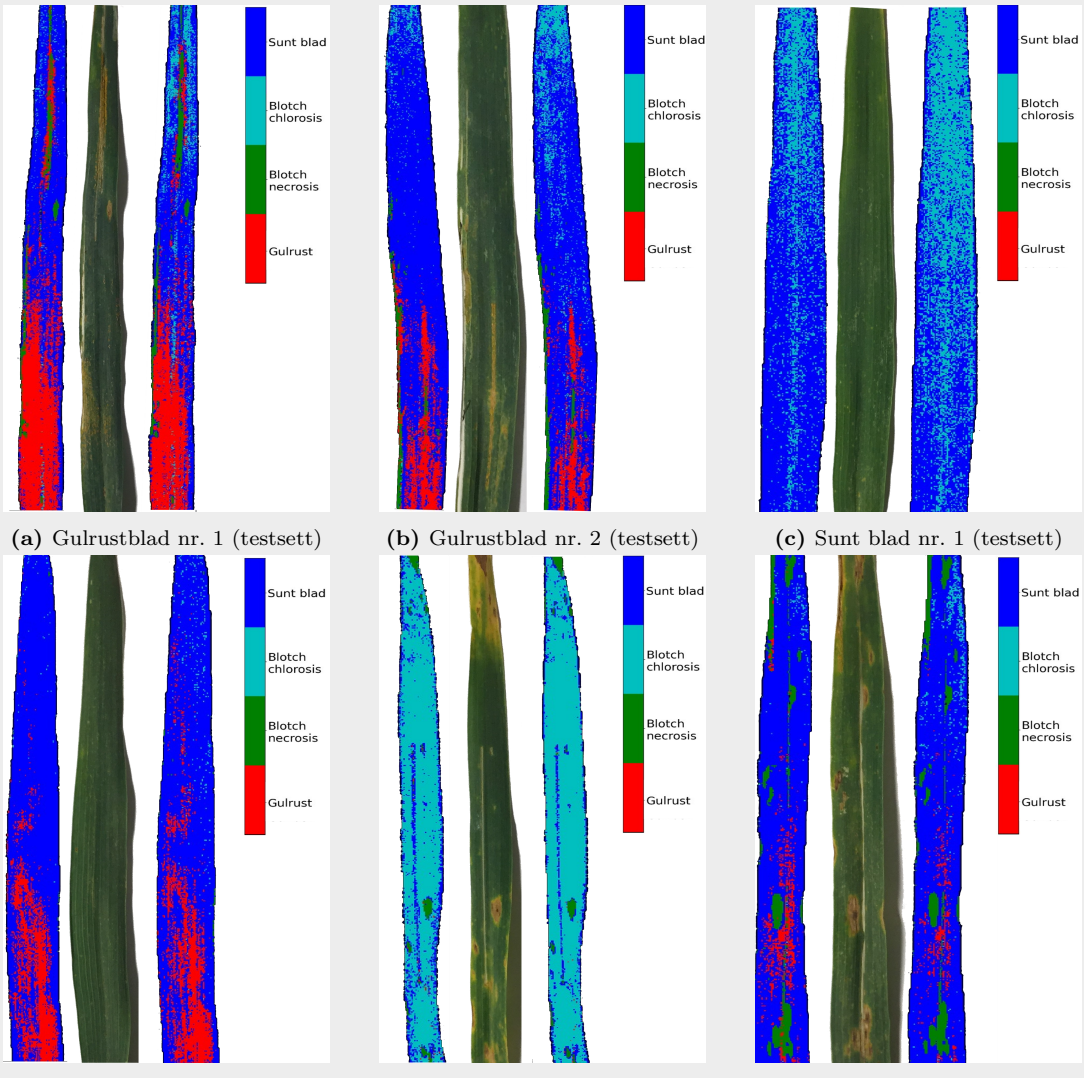
Tabell 3: Sensitive bølgelengder valgt fra bølgelengdevektlegging til RFC og komponentene til PCA, begge trent på spektrum med og uten vannovertonene resulterende i fire sett med sensitive bølgelengder.

Sett	Grunnlag	Sensitive bølgelengder [nm]
1	PCA	953.3, 1280.0, 1432.4, 1530.4, 1878.8, 1927.8, 2004.0, 2287.1
2	PCA u/ vannabs. bånd	953.3, 1285.4, 1312.6, 1682.8, 1715.5, 2031.2, 2129.2, 2221.8, 2363.3
3	RFC	1002.3, 1323.5, 1345.3, 1405.2, 1421.5, 1824.4, 1851.6, 1868.0, 1928.0, 1966.0
4	RFC u/ vannabs. bånd	1002.3, 1258.2, 1329.0, 1623.0, 1693.7, 1715.5, 1726.4, 1818.9

Tabell 4: Statistikk fra modellene SVC, PLS-DA og RFC trent på ulike bånd fra EMSC-prosessert data. Settene er de fra 3

Bånd	SVC	PLS-DA	RFC
Alle	$92.0 \pm 2.7\%$	$44.3 \pm 6.3\%$	$91.4 \pm 3.4\%$
Sett 1	$87.6 \pm 7.1\%$	$37.3 \pm 3.8\%$	$87.6 \pm 4.3\%$
Sett 2	$84.0 \pm 3.1\%$	$42.0 \pm 6.4\%$	$82.4 \pm 3.5\%$
Sett 3	$90.2 \pm 3.5\%$	$43.3 \pm 4.4\%$	$90.7 \pm 2.2\%$
Sett 4	$88.1 \pm 2.2\%$	$40.9 \pm 5.3\%$	$88.9 \pm 2.7\%$

De beste modellene, SVC trent med alle bånd og RFC trent med bølgelengder i sett 3, ble videre brukt til å klassifisere piksler på bilder i testsettet. Resultatene vises i figur 13.



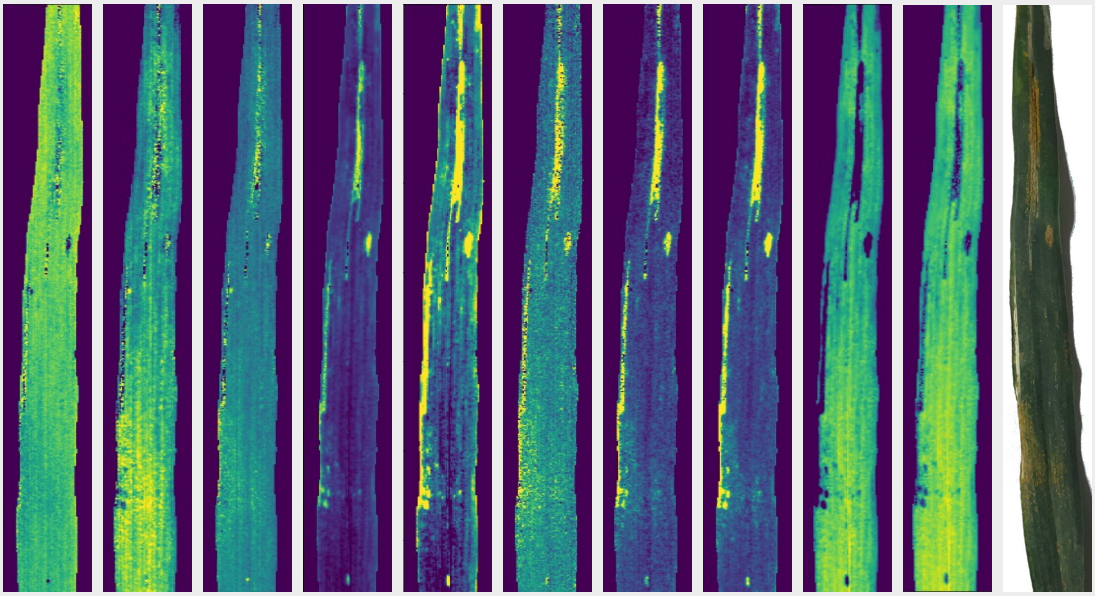
(a) Gulrustblad nr. 1 (testsett) **(b)** Gulrustblad nr. 2 (testsett) **(c)** Sunt blad nr. 1 (testsett)

(d) Sunt blad nr. 2 (testsett) **(e)** blotch-blad nr. 1 (testsett) **(f)** blotch-blad nr. 2 (testsett)

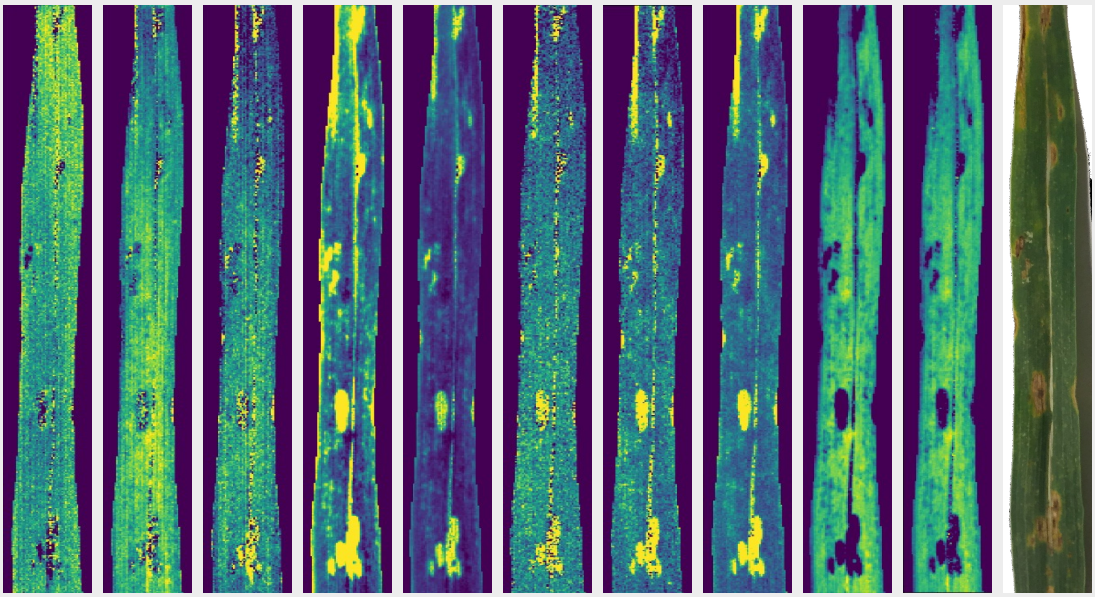
Figur 13: Bildeklassifisering av testsett med SVC og RFC. Figurene a-f viser klassifisering av bladene gulrust 1 og 2, sunt blad 1 og 2 og blotch-blad 1 og 2 fra testsettet. På hver figur er bladets RGB-bilde i midten, til venstre er pikselklassifiseringen til SVC trent på EMSC og til høyre er den til RFC trent på 10 sensitive bølgelengder fra sett 3.

Klassifiseringen av gulrustblad nr. 1 og 2 i figur 13a og 13b ser vellykket ut, da den fanger de åpenbare områdene med gulrust, men identifiserer disse områdene som større enn de på RGB-bildene. Pikslene ligger også her langsmed bladnervene, slik som i pikselklassifiseringen i figur 9a og 9b. Det identifiseres noe blotch necrosis på gulrustbladene, på midten i et tydelig blotch-område av blad nr. 1, men også langsmed gulrustlinjen på øvre del av blad nr. 1. For sunt blad nr. 1 og 2 i figur 13c og 13d klassifiseres det tilsynelatende dårlig. Mange av pikslene klassifiseres som sunt blad av begge modellene, men for blad nr. 1 klassifiseres mange som blotch chlorosis, noe som tilsynelatende er feil når bladet ikke har noe blotch på seg. Mange av disse ligger tydelig langsmed bladets midtnerve. For blad nr. 2 klassifiseres det flere områder som gulrust på nedre del av bladet, av begge modellene, tilsynelatende langsmed bladnervene. Klassifiseringen av blotch-blad nr. 1 og 2 figur 13e og 13f er både god og dårlig. For blad nr. 1 klassifiseres tilnærmet hele bladet som blotch chlorosis, noe som stemmer for øvre del og rundt de tydelige blotch necrosis-områdene, men ikke for resten av bladet som er sunt blad. Begge modellene klassifiserer piksler rundt kanten av bladet og de tilhørende en stripe forårsaket av insektskade som sunt blad. For blad nr. 2 er områder av blotch necrosis korrekt klassifisert, men alle mangler den omkretsende ringen med blotch chlorosis. Store deler av bladets tilsynelatende sunne piksler er korrekt klassifisert, med unntak av de klassifisert som gulrust ved bladets nedre del.

Siden sett 3 oppnår høyest nøyaktighet er det ønskelig å knytte dets enkelte bølgelengder til hver klasse, noe som kan oppnås ved å visualisere bildet av bladet for hver av disse bølgelengdene. Dette gir muligheten til å kvalitativt utpeke de viktigste båndene, samt knytte disse til hver klasse. Resultatet vises i figur 14. Her kommer blotch-områdene tydelig fram i alle bølgelengdene høyere enn 1345.3 nm som enten høy eller lav verdi (lys eller mørk farge), men de er mindre tydelige i de kortere bølgelengdene. På gulrustbladet er områdene som ble klassifisert som gulrust i figur 13a svært tydelige i bildet av bølgelengden 1323.5 nm og det kan virke som denne bølgelengden vektlegges sterkt av modellene. Dette gjelder også for blotch-blad nr. 2 som også klassifiserer noen gulrustområder i figur 13f. De lengre bølgelengdene 1928.0 nm og 1966.0 nm viser også tegn på tilknytting til gulrust, men ikke like sterkt. Bølgelengdene 1851.6 nm og 1868.0 nm virker tilknyttet klassifiseringen av blotch chlorosis ved nøye inspeksjon av mønsteret til blotch chlorosis-området klassifisert av RFC modellen i figur 13f. Korrelasjonen mellom bølgelengdene vises også godt, og det kan tenkes at settet kunne vært redusert til fire ukorrelerte bølgelengder sentrert rundt 1323.5 nm, 1405.2 nm, 1851.6 nm og 1966.0 nm.



1002.3nm 1323.5nm 1345.3nm,1405.2nm 1421.5nm 1824.4nm 1851.6nm 1868.0nm 1928.0nm 1966.0nm RGB

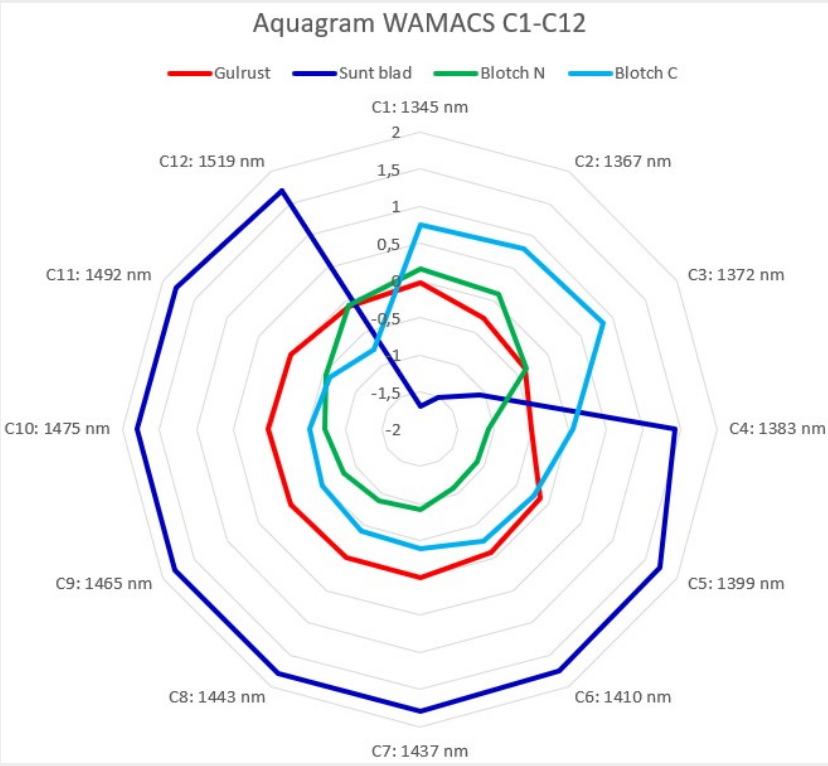


1002.3nm 1323.5nm 1345.3nm 1405.2nm 1421.5nm 1824.4nm 1851.6nm 1868.0nm 1928.0nm 1966.0nm RGB

Figur 14: Bølgelengdene tilhørende sensitive bølgelengder i sett 3 fra de hyperspektrale bildene til testbladene gulrust nr. 1 (øverst) og blotch nr. 2 (nederst).

4.3 Aquagram

Av resultatene kom det fram at vannovertonene spiller en vesentlig rolle i klassifiseringen. Derfor blir absorpsjonsverdiene til vannrelaterte bølgelengder undersøkt nærmere i et aquagram vist i figur 15. Figuren viser at sunt blad-klassen har lave verdier rundt C1-C3, mens de er for C4-C12 jevnt høye. For klassen blotch chlorosis finner vi den samme men motsatte sensitiviteten, høyere verdier i C1-C3 og lavere for C4-C12. Klassen gulrust viser et stabilt absorpsjonsmønster med bare litt lavere verdier i C2 og C3. Klassen blotch necrosis viser også et stabilt absorpsjonsmønster, men mindre enn for gulrust. Klassen har noe høyere verdier i C1, C2, C3 og C12, mens det for C4 og C5 er litt lavere.



Figur 15: Aquagram av normaliserte absorpsjonsverdier (gjennomsnitt) for klassene ved utvalgte bølgelengder (WAMACS)

5 Diskusjon

5.1 Innsamling og fotografering av blader

Ved innsamling av bladene oppsto det i noen poser fukt som følge av kondensering av luft og/eller fordamping av vann i bladene. Som bølgelengdevektleggingen i figur 11 og 12 viser er bølgelengdene knyttet til vannabsorpsjon viktige for klassifiseringen, noe som også er bekreftet av den lavere nøyaktigheten for modeller trent på sensitive bølgelengder utenfor vannovertonene, vist i tabell 4. Derfor vil den usikre graden av redusert vanninnhold i bladene ha påvirket analysen, både gjennom endret vanninnhold i bladene, men også gjennom endringen i bladets fasong og tekstur. Det kan argumenteres for at bladene tørket like mye før bildetaking og at feilen da er systematisk, men tiden det tok fra første til siste blad var fotografert var betydelig i forhold til den totale tiden for plukking og bildefangst. Det har også blitt knyttet positiv korrelasjon mellom et blads tørking og dets opprinnelige vannmengde [29], som derfor tilsier større grad av tørke hos de sunne bladene, som naturligvis har høyere vanninnhold. For å minke tørke av bladene burde fangst av hyperspektrale bilder skje så raskt som mulig etter plukking, samt fraktes og oppbevares kjølig.

Testsettet ble som nevnt plukket fra hvetesorten Zebra, altså en annen hvetesort enn den benyttet

for å danne treningssettet (Bjarne). Dette kan medføre en svekket tiltro til konklusjonene trukket gjennom testsettet, men samtidig en sterkere evne til å benytte disse konklusjonene generelt for flere hvetesorter.

5.2 Preprosesseringen

Utvelging av piksler for å representere klassene var utsatt for feilvalg, altså velging av piksler som ikke tilhørte den ønskede klassen. For gulrust ble de små postulene, tydelige symptomer på bladene som følge av viruset, fokusert (se figur 4a) og de utvalgte områdene var dermed svært små og antall gulrustspektrum ble derfor lavere enn for andre klasser. Denne klassen var derfor mest utsatt for feilvelging. For klassen blotch chlorosis var områdene store og tydelige på RGB-bildene av bladene, men for de hyperspektrale var de tilnærmet usynlige (se figur 4b). Dette resulterte i en utvelging av områder basert på deres posisjon i forhold til andre, gjenkjennelige områder, i hovedsak områder knyttet til blotch necrosis som var svært tydelig på hyperspektrale bilder (se figur 4c). Klassen blotch necrosis var derfor minst utsatt for feilvelging. For klassen sunt blad ble det valgt å danne en maske av bladet gjennom en enkel K-means gruppering, for deretter å velge ut alle pikslene i bladet innenfor denne masken. Grunnen til å bruke denne utvelgingsmetoden for sunt blad var den allerede nødvendige produksjon av masker av bladene noe som førte til tidsbesparelse. Sunt blad klassen skiller seg ut fra de andre ettersom hele bladet kategoriseres som sunt, og ikke kun et lite område. En enklere og raskere metode for å danne masker av bladene er å benytte en terskelverdi for spektrumsverdier for å skille bakgrunn og blad [3].

Gulrust viser en høyere refleksjon enn sunt blad i NIR-området (950 nm-1300 nm) i figur 5a, noe som strider i mot den generelle beskrivelsen av sunt blad i forhold til dødt blad: refleksjonen til sunt blad er som generelt høyere i NIR-området på grunn av multiplikativ spredning mellom cellene og luft i bladvevet [6, 8]. Resultatene her kan begrunnes med at gulrustspektrene tilhører sporehopene på bladene, og ikke nødvendigvis uttørkede, nedbrytende blader, men det kommer fram av alle de hyperspektrale bildene at også de uttørkede bladområdene har høyere refleksjon i dette området. Spektrum til gulrust ligger jevnt over sunt blad i figur 5a, noe som kan tyde på en differanse forårsaket av en jevn forskyving. Spektrene krysser flere ganger og differansen blir mindre når spektrene blir SNV-korrigert og EMSC-prosessert, noe viser viktigheten av korrigering. Zhifeng et al. [3] benytter hyperspektrale bilder i området 400 nm - 1000 nm for å oppdage tidlig utvikling av gulrust på vinterhvete og har også høyere refleksjon for de infiserte og syke bladene i NIR-området. Studien ønsket å ivareta karakteristikkene til spektrene og valgte derfor kun et S-G filter for preprosessering, noe som kan ha ført til større differanse mellom sunt og infisert blad. Årsaken til resultatenes høyere refleksjon i dette området for uttørkede blad er ikke kjent, men kan komme av uvitende effekter ved preprosesseringen eller ukjente påvirkninger ved bildefangst.

Ved endt preprosessering oppbevares spektrene til de utvalgte pikslene i lister og mister dermed den originale romlige informasjonen. Som nevnt utvikler gulrustsymptomer seg ofte mellom og langsmed bladvevet og har derfor et sterkt geometrisk uttrykk. En modell som utnytter denne informasjonen vil derfor være bedre til å identifisere gulrust. Det kan argumenters for at utnyttelsen av romlig informasjon muligens minker sjansen for å finne bølgelengder sensitive til gulrust, at modellene baseres seg mer på den romlige informasjonen, men målet om å oppnå enkel og rask gulrust-, og annen sykdomskartlegging i hveteåkre overgår målet om å forstå sykdommene og hvilke stoffer de består av. Dype nevrale nettverk av typen Convolutional Neural Network (CNN) er en modell som kan benytte den romlige informasjonen, og ble brukt av Zhang et al. [4] hvor gulrust ble forsøket kvantifisert i en åker ved hjelp av dronebilder. Studien benyttet også temporal data, og oppnådde høy nøyaktighet ved alle utviklingsfaser til sykdommen, med høyest nøyaktighet i de seneste fasene. Det kan argumenteres at bladet lenge har vært utsatt for gulrust når symptomene med romlig informasjon oppstår og at metoder som oppdager viruset i tidligere stadier er mer ønskelig.

Det kan argumenteres for at tilstedeværelsen av klassen blotch chlorosis også ville krevet bruken av en klasse for klorose forårsaket av gulrust, men siden klorose (og nekrose) er en reaksjon fra planten, vil det bli vanskelig å skille mellom årsakene til disse kun ved hjelp av spektral informasjon (etter samtale med Morten Lillemo, Professor i plantevitenskap og leder av prosjektet masteroppgaven er en del av). Det kommer fram av figur 13 at flere chlorosis-områder på gulrustblad klassifiseres som blotch chlorosis, noe som tyder på at modellene ikke skiller mellom sykdomsspesifikk klorose. Det

kan derfor muligens benyttes en fellesklasse for all klorose på bladene.

5.3 Metode 1

5.3.1 Datatypene og modellvalideringen

Det kommer tydelig fram av figur 5 at 1. og 2. derivasjonsspektrum inneholder mye støy, selv etter glatting med S-G filter. Dette kombinert med lav nøyaktighet på validering på tvers av modellene, tabell 2, gjør disse datatypene uegnet til klassifiseringen. Videre viser figur 5 hvordan både SNV-korrigeringen og EMSC-prosesseringen fører til redusert variasjon i data. Av tabell 2 kommer det fram at EMSC-prosessert data gir høyere nøyaktighet og lavere nøyaktighetsvariasjon enn SNV-korrigert ved kryssvalidering av modellene.

For å validere om data i testsettet kan sees på som lik den i treningssettet, selv om de er fanget ved ulike tidspunk og muligens varierende lysmengder, kunne en uavhengig t-test vært benyttet for å teste ulikhetene i bølgelengdeverdier [30].

5.3.2 Pikselklassifiseringen

Gjennom klassifiseringen vises noen problemer tilknyttet modellene. Det mest åpenbare er de sunne pikslene som klassifiseres som gulrust, noe som mest sannsynlig er en feilklassifisering, men som også kan forklares med dårlig håndtering av blader som førte til introduksjon av fremmedstoffer eller tørke. Et annet problem er at gulrustområder er identifisert som mye større på klassifiseringen enn de på RGB-bildene. Dette kan også være et resultat av bladhåndteringen, muligens gulrustsporer fra andre blader som har havnet på dette bladet under transport. Men det bør påpekes at klassifiseringen av gulrust opptrer langsmed bladnervene, noe som er vanlig for disse symptomene. Derfor er det mulig at modellene enten forveksler bladnerver med gulrust, eller at de faktisk oppdager symptomene før de er synlige på RGB-bildene, noe som også gjelder for tilfellet med sunne piksler klassifisert som gulrust. For å undersøke dette nærmere er det nødvendig å se på sykdomsforløpet på infiserte blader. Et annet tydelig problem ved klassifiseringen er den åpenbare klassifiseringen av blotch, og noe chlorosis som gulrust. Dette tyder på at spektrene til disse klassene er relativt like, eller i hvertfall tilstrekkelig ulike fra sunt blad-klassen. Dette er uønsket av en modell som kun skal identifisere gulrust, da det kan føre til unødvendig bruk av fungicider dersom modellen identifiserer andre sykdommer som gulrust, og det er altså nødvendig å inkludere blotch-områdene som en klasse i treningsdata, for at modellene skal kunne skille dem fra gulrust. Zhang et al. [4] benyttet klassene sunt blad, gulrust og "andre" for å ikke gruppere andre sykdommer sammen med gulrust.

5.4 Metode 2

5.4.1 Dataforenklende analyse og bølgelengdevektlegging

Klassen blotch necrosis er den som basert på spektrum i figur 5c er enklest å skille fra de andre, som kan være grunnen til at PCA i figur 10 har større overlapp mellom de andre klassene. LDA skiller naturligvis klassene godt, men har til gjengjeld en svært oscillerende og ikke tydelig vektlegging av bølgelengdene i figur 11. Det kan argumenteres for at antagelsene som LDA baseres på ikke stemmer for studiens datasett, og dermed at dens gruppering og bølgelengdevektlegging ikke kan benyttes, men det har vært funnet god diskrimineringsevne av LDA som har benyttet data som ikke oppfylte antagelsene fullstendig, mens bruk av LDA som klassifiseringsmodell er strengere i dens krav om oppfylte antagelser [20, 31]. Det kommer tydelig fram av vektleggingene at begge vannovertonene er viktige noe som sannsynligvis kommer av minking av vanninnholdet i infisert bladvev.

5.4.2 Sensitive bølgelengder

Det er god kjent at SWIR-området er relatert til vannabsorpsjon og det benyttes ofte for å undersøke tørke i planter [32]. Krishnaa et al. [33], som forsøkte å identifisere sensitive bølgelengder for gulrust på hveteblader, isolerte bølgelengden 1399 nm fra SWIR i kombinasjon med bølgelengdene 428 nm og 672 nm fra VIS ved hjelp av PLSR og en multi-lineær regresjonsmodell. Denne bølgelengden kan tilnærmet være representert i sett 3 som 1405.2 nm. Huang et al. [6] kombinerte

hyperspektrale bilder av VIS, NIR og SWIR med målinger av klorofyll- og nitrogeninnhold i bladene og benyttet bølgelengdene 704 nm, 1423 nm og 1926 nm for å konstruere en indeks som kvantitativt målte gulrust. Begge disse studiene bruker bølgelengder som er sentrale i vannovertonene og det kan tenkes at de dermed kun representerer vanninnhold i bladet. Apan et al. [34] benyttet bølgelengder fra VIS og bølgelengden 1660 nm for å identifisere appelsinrust på sukkerrør ved hjelp av hyperspektrale bilder fra EO-1 satellitten. Denne bølgelengden ligger utenfor vannovertonene, men refereres til i studien som fuktighetsrelatert og knyttes altså også til vanninnhold. Bølgelengdene i settene som ligger nærmest denne er 1682.8 nm fra sett 2 og 1623.0 nm og 1693.7 nm fra sett 4. RFC bølgelengdevektleggingen nederst i figur 12 identifiserte nettopp dette området som av stor betydning for klassifiseringen, men det er uvisst om det er grunnet fuktighet eller sykdomsrelatert.

Herrmann et al. [35], en studie på nitrogeninnhold i potetåkre, fant nitrogeninnhold knyttet til klorofyllinnhold i potetplantene gjennom hyperspektrale bilder, og finner en direkte sammenheng mellom bølgelengden 1510 nm og nitrogeninnhold. Ingen av settene inneholder denne bølgelengden, med sett 1 med den nærmeste, 1530.3 nm. Evne til å utnytte bølgelengder relatert til stoffer på kryss av plantearter er usikker. Ferwerda et al. [36] undersøkte nitrogeninnhold i flere arter og fant mye variasjon i høyest korrelerte bølgelengde, men at kombinasjonen av bølgelengdene 693 nm og 1770 nm muligens gir den beste relateringen til nitrogeninnhold. Schmidt et al. [37] undersøkte spektral diskriminering av ulike vegetasjonstyper og fant bortimot alle vannovertonene i SWIR, samt bølgelengdene rundt 1700 nm - 1820 nm og spektrumstoppen rundt 2000 nm av stor betydning. Dette tyder på ulikheter i disse bølgelengdene mellom ulike vegetasjonstyper og dermed også muligens mellom plantearter, noe som kan gjøre det vanskelig å finne en bestemt nitrogensensitiv bølgelengde. Kokaly [38] undersøker nitrogeninnhold i tørt løvverk og identifiserer bølgelengdene 2055 nm og 2172 nm som relatert til nitrogeninnhold og knytter den til nitrogenbindingene til amidbånd til proteiner, men den spektrale forskjellen på sunt eller sykt blad, som benyttet i denne studien, og tørt blad er trolig stor. Disse nitrogensensitive bølgelengdene var ikke direkte representert i settene, men området rundt 2172 nm har en betydelig vektlegging i figur 12 som ikke ble inkludert i sett 4 på grunn av dens relativt lavere vektlegging enn andre bølgelengder. Sett 1 og 2 inneholdt bølgelengder i områdene rundt 2000 nm og 2200 nm som kan ha hatt sammenheng med nitrogeninnhold. Settene presterte dårligere enn sett 3 og 4, noe som kan bety at sykdommene ikke kan gjenkjennes basert på nitrogeninnhold, men det kan også ha kommet av økt støy i disse lengre bølgelengdene.

5.4.3 Modellvalideringen

Nøyaktigheten er lik den oppnådd av andre studier som benytter VIS/NIR-områdene for klassifisering [7, 32]. Den høyere nøyaktigheten til modellene trent på alle bånd tyder på at de sensitive bølgelengdene i settene fører til tap av viktig informasjon, men siden nøyaktigheten ikke er mer enn 1.3 prosentpoeng høyere enn det for det beste settet viser at dette informasjonstapet er svært lite. Nøyaktigheten til PLS-DA er dårlig og mye lavere enn for de andre modellene. Dette kan skyldes feil i modifiseringen av PLSR-modellen, men også at modellen er uegnet for fler-klassen klassifisering. Settene 1 og 3 som inkluderer bølgelengder fra vannovertonene oppnår høyere nøyaktighet enn sett 2 og 4, men det bør merkes at RFC trent og validert med bølgelengder i sett 4 har oppnådd $88.9 \pm 2.7\%$, altså en fortsatt høy nøyaktighet. Dette tyder på at det er mulig å benytte bølgelengder i SWIR-området utenfor vannovertonene til diskriminering av gulrust, blotch chlorosis og blotch necrosis.

5.4.4 Pikselklassifiseringen

Introduksjonen av flere klasser gir bedre og mer detaljert klassifisering av bladpikslene, men noen problemer er fortsatt tydelige. Gulrustområdene oppdaget av modellene er i likhet med pikselklassifiseringen i figur 9 større enn de på RGB-bildene, og opptrer fortsatt langsmed bladnervene. Det identifiseres fortsatt gulrust på sunt blad nr. 2 i figur 13d, noe som forsterker muligheten for at modellene identifiserer sykdom ikke synlig på RGB-bildene, men også muligheten for dårlig kvalitet på data, grunnet bladhåndtering eller tørke. Identifiseringen av blotch chlorosis langsmed bladets midtnerve for sunt blad nr. 1, figur 13c, kan komme av at midtnervens tykkelse, og dermed dens spektrum, skiller seg fra resten av bladet. Det kan også tenkes at bladet nylig er infisert med blotch og at modellene oppdager dette. Om det stemmer at store deler av bladet utvikler blotch chlorosis ved tilstedeværelsen av blotch, slik som i sunt blad nr. 1 og blotch blad nr. 1, figur 13e, og

at modellene oppdager dette, burde også blotch blad nr. 2 hatt liknende blotch chlorosis-områder identifisert. Siden dette ikke er tilfellet kan det heller virke som modellene feilklassifiserer. Det faktum at klassifiseringene viser ingen blotch necrosis-områder omkretset av blotch chlorosis-områder svekker tilitten til denne klassens diskriminering enda mer, og dermed også modellenes nøyaktighet.

Introduksjonen av klassen blotch necrosis var altså vellykket og gav pikselklassifiseringen en evne til å skille gulrust og blotch, mens klassen blotch chlorosis var mindre vellykket. Pikselklassifiseringen ble tilsynelatende påvirket av små variasjoner i spektrene, for eksempel midtnerven i sunt blad nr. 1, noe som skapte litt varierende resultat. Dette gir mening da data er av høy kvalitet og fanger svært mye informasjon fra hvert blad, men det kan tenkes at disse små variasjonene i spektrene jevnes ut og dempes ved datafangst fra for eksempel drone. Modellene oppdager områder tydelig knyttet til sykdom og skade, men oppdager også områder tilsynelatende uten sykdom, og det vil derfor kreve videre undersøkelser knyttet til sykdommenes forløp på bladene for å kunne avkrefte om modellene faktisk oppdager sykdommer før symptomene vises på RGB-bilder, eller om modellene feilklassifiserer.

Tilknyttingen av bølgelengder i sett 3 til klassene i figur 14 utpeker viktige bølgelengder for gulrust og blotch chlorosis og viste at blotch necrosis, som tydelig gjennom den åpenbare utskillelsen av klassen i figur 5c, lett kan identifiseres gjennom de fleste av bølgelengdene. Identifiseringen av bølgelengde 1323.5 nm som svært viktig for gulrust kan begrunnes med at den ligger nært første vannovertonen og man kan forvente en viss korrelasjon, men refleksjonen endrer seg raskt i dette området så korrelasjonen bør isåfall være liten. Bølgelengdene 1928 nm og 1966 nm er også viktige for gulrust, men ligger i andre vannovertone noe gjør praktiske målinger vanskelige. Man ser også at disse bølgelengdene gir en høyere refleksjon for gulrustområder, mens blotch necrosis får svært lav, noe som kan være grunnlag for å kunne skille disse klassene. Sensitivitet for gulrust ved 1323.5 nm, mulig skillelse mellom gulrust og blotch necrosis i 1928 nm og 1966 nm, sammen med sensitivitet for blotch chlorosis rundt 1851 nm, kan gi grunnlag for god diskrimineringsevne av sykdommene, men kan ikke enkelt knyttes til bestemte stoffer i planten uten videre undersøkelser.

Det identifiseres altså ingen direkte tilknytning til bestemte stoffer i bladet med kjente absorpsjonsområdet i NIR eller SWIR, selv om noen av de sensitive bølgelengdene ligger nært bølgelengder knyttet til nitrogeninnhold, og det er nødvendig med videre forskning på både planter og bladvev, men også deres bestanddelers unike absorpsjonsmønstre, samt hvordan vannabsorpsjon påvirker målinger i SWIR. Utenfor vannovertonene peker denne studien ut bølgelengden 1323.5 nm, og området rundt, som muligens av betydning for gulrust.

5.5 Aquagrammet

Fra aquagrammet i figur 15 ser man en sterkere absorpsjon for suntblad enn for sykt blad i C4-C12, noe som tyder på god diskrimingeringsevne. Gulrustklassens stabile absorpsjonsmønster viser at den har liten eller ingen større tilknytting til en bestemt WAMACS. Den lille endringen i C2 og C3 som i følge tabell 1 er knyttet til hydreringsskall og symmetrisk og asymmeterisk strekking av den første overtonen av vann, er så liten at den kan skyldes størrelsen på datasettet. Med tanke på sykdomsforløpet til blotch kan man forvente en gradvis endring i absorpsjonsverdier fra sunt blad til blotch chlorosis og til slutt til blotch necrosis, men for C1-C3 kan det virke som utviklingen starter med svak absorpsjon for sunt blad, øker kraftig for klassen blotch chlorosis, for deretter å synke igjen til en verdi høyere enn sunt blad. Dette skal på grunn av aquagrammets teori reflektere molekylære egenskaper i planten, men siden klassen blotch chlorosis har resultert i en mindre vellykket klassifisering og dens betydning virker tvilsom, svekkes denne reflekteringen. Aquagrammet viser ikke variasjonen for hver klasse, kun gjennomsnittet, noe som kan gjøre utvelgingen av bestemte, sensitive WAMACS mindre korrekt dersom klassene har stor variasjon, eller føre til oversette sensitive WAMACS dersom gjennomsnittet ikke endrer seg, men klassene har distinkte fordelinger [22]. Generelt virker den største forskjellen mellom sunt og sykt blad som absorpsjon i C4-C12, mens området C1-C3 muligens innehar mer detaljert informasjon om sykdomsforløpet, og det er nødvendig med videre forskning og bedre forståelse av molekylene og stoffene i bladene.

5.6 Kritikk mot metoden

Datasettet er relativt lite, for eksempel brukte Anting et al. [7] et datasett av 96 blad, og får da mindre nøyaktig refleksjon av virkelige verdier. Bladene benyttet til trening var plukket fra hvetesorten Bjarne, noe som også minker resultatenes refleksjon av alle hvetesorter, selv med testing på en annen hvetesort (Zebra). Det burde heller vært benyttet flere blader fra ulike hvetesorter til både trening, validering og testing, slik at man kan utelukke feilaktig funn av sensitive bølgelengder og ha et sterke grunnlag for å knytte sensitive bølgelengder til sykdommene.

Studien benytter hyperspektral data samlet fra blader under kontrollerte omgivelser i bilderom, noe som kan gi praktisk uanvendelige resultater for det meste grunnet den sterke vannabsorpsjonen i SWIR-området. Her vil det være nødvendig med en bedret forståelse av hvordan vannabsorpsjon endrer eller påvirker andre stoffers absorpsjon i SWIR-området, noe oppbyggingen av en database innen aquafotonikk kan bidra til.

Som nevnt ble kun nøyaktighet og dens standardavvik benyttet i valideringen for å bedømme hver modell og datatypes klassifiseringsevne. Når klassifiseringen er fler-klasses er det ofte en fordel å bruke presisjon, sensitivitet og F1-score for å få et bedre inntrykk av modellers prestasjon og har blitt benyttet i flere studier [32]. En forvirringsmatrise (confusionmatrix) hadde også vært en fordel, for å se hvordan modellene klassifiserte, hvilke klasser de klassifiserte godt og dårlig og om noen klasser ofte ble feilklassifisert som en annen. Dette kommer til en viss grad fram kvalitativt gjennom klassifiseringene i figur 13, men en kvantitativ bestemmelse hadde vært foretrukket.

5.7 Videre arbeid

Modellenes identifisering av sykdom ikke synlig på RGB-bildene i figur 13 kan kun valideres ved å undersøke sykdommenes utvikling på bladene. Zhang et al. [4] og Su et al. [5] benyttet både temporal, romlig og spektral informasjon når de fanget data av hveteåkre med drone, altså er de svært knyttet til den mulige praktiske anvendelsen av videre forskning. Likevel kan det også være fordelaktig å fange mer detaljert data av bladene, som hyperspektrale bilder av enkelte blad, over tid, noe som kan gi en sterkere evne til å knytte funn av sensitive bølgelengder til bladenes og sykdommenes biologi, fysiologi og molekylære innhold. Ved å benytte temporal og biokjemisk informasjon kan man knytte sensitive bølgelengder til sykdommers spesifikke symptomer ved hvert utviklingsstadie, noe som dermed kan gi mønstre for hver sykdom og lettere føre til identifisering [33, 39]. Det er ønskelig å identifisere sykdomsutbrudd så tidlig som mulig, men som forklart i Zhang et al. [30] vil de tidligste symptomene på gulrust ikke forårsake noen endring i plantenes fysiologiske og biokjemiske egenskaper og identifisering av gulrust kan dermed bli vanskelig. Videre studier bør altså fokusere på å øke vår kunnskap om plantene og sykdommene, deres spektrale mønstre og utvikling, men også praktisk anvendelighet av resultatene er viktig, spesielt med tanke på vanskelighetene rundt bildefangst i SWIR-området.

6 Konklusjon

Evne til diskriminering av gulrust og bladflekksykdommer i hveteblad gjennom hyperspektrale bilder i NIR og SWIR-området er god og bør utforskes ytterligere for å identifisere flere sensitive bølgelengder og mønstre. Fra bølgelengdevektlegging til PCA og RFC kommer det fram at vannovertonene er viktige for klassifiseringen. SVC trent på hele spekteret oppnår en nøyaktighet på $92.0 \pm 2.7\%$, RFC trent på 10 sensitive bånd oppnår $90.7 \pm 2.2\%$, mens RFC trent på 8 sensitive bånd utenfor vannovertonene oppnår $88.9 \pm 2.7\%$. Resultater tyder på at 1323.5 nm, 1928.0 nm og 1966.0 nm kan knyttes til gulrust, men dette vil kreve videre forskning. Det identifiseres ingen direkte tilknytning til kjente absorpsjonsområder til nitrogen i NIR eller SWIR, og det er nødvendig med videre forskning på både planter og sykdommene og deres bestanddelers unike absorpsjonsmønstre, samt hvordan vannabsorpsjon påvirker målinger i SWIR. Framtidige studier bør inkludere romlig og temporal informasjon, i tillegg til den spektrale.

Referanser

[1] Ann Katrin Holtekjølen mfl. *hvete*. Store norske leksikon. URL: https://snl.no/hvete#-Beskyttet_norsk_produksjon (sjekket 28.05.2021).

[2] Guro Brodal og Andrea Ficke. *Gulrust*. NIBIO Plantevernleksikonet. 21. jul. 2017. URL: https://www.plantevernleksikonet.no/l/oppslag/1235/ (sjekket 28.05.2021).

[3] Zhifeng Yao, Yu Lei og Dongjian He. "Early Visual Detection of Wheat Stripe Rust Using Visible/Near-Infrared Hyperspectral Imaging". I: *Sensors* 19.4 (2019). ISSN: 1424-8220. DOI: 10.3390/s19040952. URL: https://www.mdpi.com/1424-8220/19/4/952.

[4] Xin Zhang mfl. "A Deep Learning-Based Approach for Automated Yellow Rust Disease Detection from High-Resolution Hyperspectral UAV Images". I: *Remote Sensing* 11.13 (2019). ISSN: 2072-4292. DOI: 10.3390/rs11131554. URL: https://www.mdpi.com/2072-4292/11/13/1554.

[5] Jinya Su mfl. "Spatio-temporal monitoring of wheat yellow rust using UAV multispectral imagery". I: *Computers and Electronics in Agriculture* 167 (2019), s. 105035. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2019.105035. URL: https://www.sciencedirect.com/science/article/pii/S0168169918318520.

[6] Lin-Sheng Huang mfl. "Hyperspectral Measurements for Estimating Vertical Infection of Yellow Rust on Winter Wheat Plant". I: *International Journal of Agriculture and Biology* 17 (nov. 2015). DOI: 10.17957/IJAB/15.0034.

[7] Anting Guo mfl. "Identification of Wheat Yellow Rust Using Spectral and Texture Features of Hyperspectral Images". I: *Remote Sensing* 12.9 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12091419. URL: https://www.mdpi.com/2072-4292/12/9/1419.

[8] Lin-Sheng Huang mfl. "Identifying and Mapping Stripe Rust in Winter Wheat using Multi-temporal Airborne Hyperspectral Images". I: *International Journal of Agriculture Biology* 14 (5 2012), s. 697–704. ISSN: 1560-8530.

[9] Alan P. Roelfs, Singh R. P. og Saari E. E. *Rust Diseases of Wheat: Concepts and Methods of Disease Management*. CIMMYT, 1992. ISBN: 968-6127-47-X. URL: shorturl.at/bkrL9.

[10] *Jord- og Plantekultur 2020*. Bd. 6. 1. 2020. ISBN: 978-82-17-02481-1. URL: https://hdl.handle.net/11250/2658926.

[11] Johannes Skaar. *lys*. Store norske leksikon. URL: https://snl.no/lys (sjekket 29.05.2021).

[12] Roger Birkeland. *multispektrale opptak*. Store norske leksikon. URL: https://snl.no/multispektrale_opptak (sjekket 29.05.2021).

[13] Emma Linnea Wiström, Jostein Riiser Kristiansen og Øyvind Grøn. *elektromagnetisk spektrum*. Store norske leksikon. URL: https://snl.no/elektromagnetisk_spektrum (sjekket 29.05.2021).

[14] Trygve Holtebekk. *infrarød stråling*. Store norske leksikon. URL: https://snl.no/infrar%C3%B8d_str%C3%A5ling (sjekket 29.05.2021).

[15] Xi Yang mfl. "Seasonal variability of multiple leaf traits captured by leaf spectroscopy at two temperate deciduous forests". I: *Remote Sensing of Environment* 179 (2016), s. 1–12. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2016.03.026. URL: https://www.sciencedirect.com/science/article/pii/S0034425716301201.

[16] Raymond F. Kokaly mfl. "Characterizing canopy biochemistry from imaging spectroscopy and its application to ecosystem studies". I: *Remote Sensing of Environment* 113 (2009). Imaging Spectroscopy Special Issue, S78–S91. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2008.10.018. URL: https://www.sciencedirect.com/science/article/pii/S0034425709000790.

[17] Andrew Hennessy, Kenneth Clarke og Megan Lewis. "Hyperspectral Classification of Plants: A Review of Waveband Selection Generalisability". I: *Remote Sensing* 12.1 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12010113. URL: https://www.mdpi.com/2072-4292/12/1/113.

[18] Nils Kristian Afseth og Achim Kohler. "Extended multiplicative signal correction in vibrational spectroscopy, a tutorial". I: *Chemometrics and Intelligent Laboratory Systems* 117 (2012), s. 92–99. ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2012.03.004.

[19] Anthony J. Owens. "Uses of Derivative Spectroscopy, Application Note". I: Agilent Technologies, 5963-3940E (1995).

[20] Mitsunori Ogihara Tao Li Shenghuo Zhu. "Using discriminant analysis for multi-class classification: an experimental investigation". I: *Knowledge and Information Systems* 10 (2006), s. 453–472. DOI: 10.1007/s10115-006-0013-y.

[21] Daniel Ruiz-Perez mfl. "So you think you can PLS-DA?" I: *BMC Bioinformatics* 21.2 (2020). DOI: 10.1186/s12859-019-3310-7.

[22] Yuanpeng Li mfl. "Early Diagnosis of Type 2 Diabetes Based on Near-Infrared Spectroscopy Combined With Machine Learning and Aquaphotomics". I: *Frontiers in Chemistry* 8 (2020), s. 1133. ISSN: 2296-2646. DOI: 10.3389/fchem.2020.580489.

[23] G. Baudat og F. Anouar. "Kernel-based methods and function approximation". I: *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*. Bd. 2. 2001, 1244–1249 vol.2. DOI: 10.1109/IJCNN.2001.939539.

[24] Jafar Alzubi, Anand Nayyar og Akshi Kumar. "Machine Learning from Theory to Algorithms: An Overview". I: *Journal of Physics: Conference Series* 1142.012012 (nov. 2018). DOI: 10.1088/1742-6596/1142/1/012012.

[25] "Scikit-learn: Machine Learning in Python". I: *Journal of Machine Learning Research* 12 (2011), s. 2825–2830.

[26] Roumiana Tsenkova. "Aquaphotomics: Water in the biological and aqueous world scrutinised with invisible light". I: *SpectroscopyEurope* 22.6 (2010). URL: http://www.spectroscopyasia.com/system/files/pdf/NIR-22_6.pdf.

[27] Lidija Matija mfl. "Aquagrams: Water Spectral Pattern as Characterization of Hydrogenated Nanomaterial". I: *FME Transactions* 40 (jan. 2012), s. 51–56.

[28] Kodzue Kinoshita mfl. "Spectral pattern of urinary water as a biomarker of estrus in the giant panda". I: *Scientific Reports* 2 (2012). DOI: 10.1038/srep00856.

[29] Kevyn J. Juneau og Catherine S. Tarasoff. "Leaf Area and Water Content Changes after Permanent and Temporary Storage". I: *PLOS ONE* 7.8 (aug. 2012), s. 1–6. DOI: 10.1371/journal.pone.0042604. URL: https://doi.org/10.1371/journal.pone.0042604.

[30] Jingcheng Zhang mfl. "Using in-situ hyperspectral data for detecting and discriminating yellow rust disease from nutrient stresses". I: *Field Crops Research* 134 (2012), s. 165–174. ISSN: 0378-4290. DOI: https://doi.org/10.1016/j.fcr.2012.05.011. URL: https://www.sciencedirect.com/science/article/pii/S0378429012001839.

[31] Jerome Friedman Trevor Hastie Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science, 2001. ISBN: 978-1-4899-0519-2. DOI: 10.1007/978-0-387-21606-5.

[32] Jinya Su mfl. "Wheat yellow rust monitoring by learning from multispectral UAV aerial imagery". I: *Computers and Electronics in Agriculture* 155 (2018), s. 157–166. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2018.10.017. URL: https://www.sciencedirect.com/science/article/pii/S0168169918312584.

[33] Gopal Krishnaa mfl. "Assessing wheat yellow rust disease through hyperspectral remote sensing". I: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL-8 (des. 2014), s. 1413–1416. DOI: 10.5194/isprsarchives-XL-8-1413-2014.

[34] Armando Apan mfl. "Detecting sugarcane 'orange rust' disease using EO-1 Hyperion hyperspectral imagery". I: *Int. J. Remote Sens* 25 (jan. 2004). DOI: 10.1080/01431160310001618031.

[35] I. Herrmann mfl. "SWIR-based spectral indices for assessing nitrogen content in potato fields". I: *International Journal of Remote Sensing* 31.19 (2010), s. 5127–5143. DOI: 10.1080/01431160903283892.

[36] Jelle G. Ferwerda, Andrew K. Skidmore og Onisimo Mutanga. "Nitrogen detection with hyperspectral normalized ratio indices across multiple plant species". I: *International Journal of Remote Sensing* 26.18 (2005), s. 4083–4095. DOI: 10.1080/01431160500181044.

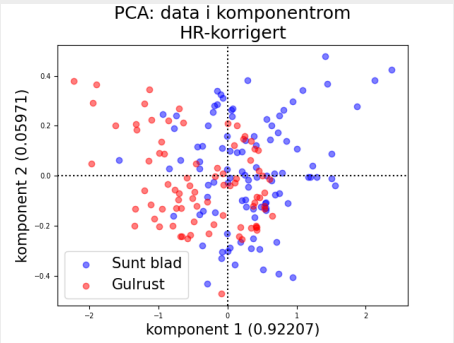
[37] K.S. Schmidt og A.K. Skidmore. "Spectral discrimination of vegetation types in a coastal wetland". I: *Remote Sensing of Environment* 85.1 (2003), s. 92–108. ISSN: 0034-4257. DOI: https://doi.org/10.1016/S0034-4257(02)00196-7.

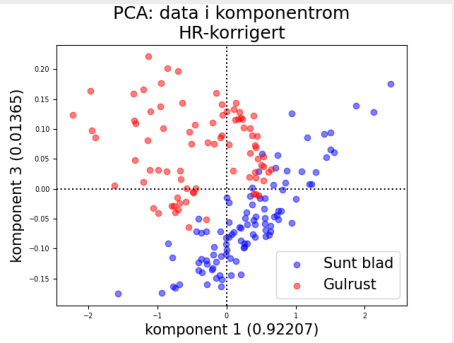
[38] Raymond F Kokaly. "Investigating a Physical Basis for Spectroscopic Estimates of Leaf Nitrogen Concentration". I: *Remote Sensing of Environment* 75.2 (2001), s. 153–161. ISSN: 0034-4257. DOI: https://doi.org/10.1016/S0034-4257(00)00163-2. URL: https://www.sciencedirect.com/science/article/pii/S0034425700001632.

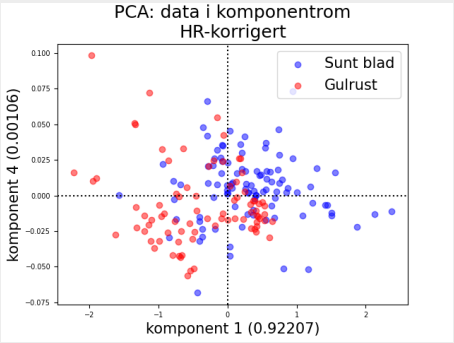
[39] Jingcheng Zhang mfl. “Monitoring plant diseases and pests through remote sensing technology: A review”. I: *Computers and Electronics in Agriculture* 165 (2019), s. 104943. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2019.104943. URL: https://www.sciencedirect.com/science/article/pii/S016816991930290X.

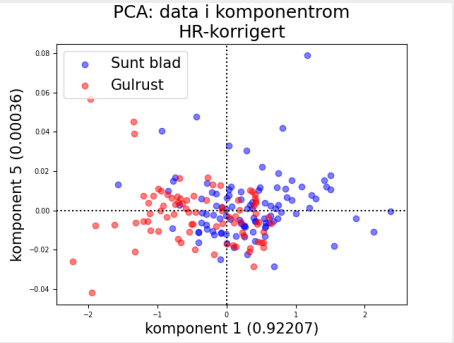
Vedlegg 1

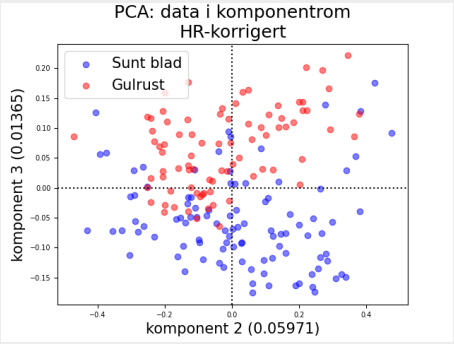
HR-korrigert: Data i komponentrommet til PCA trent på HR-korrigerte spektrum. Aksenavn viser til komponent og mengde variasjon komponenten representerer i desimaltall

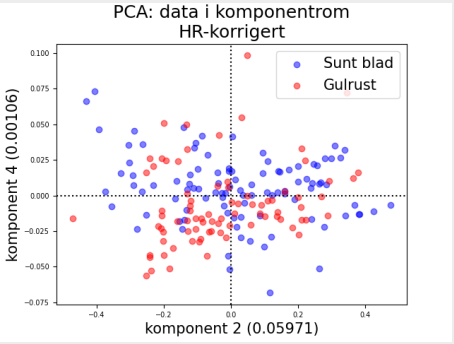


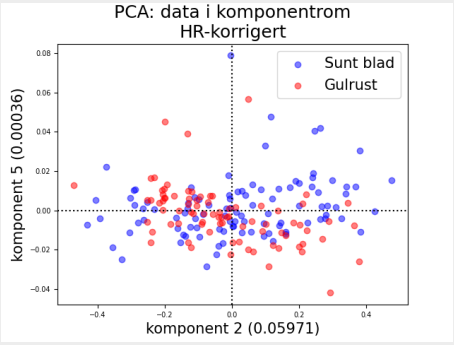


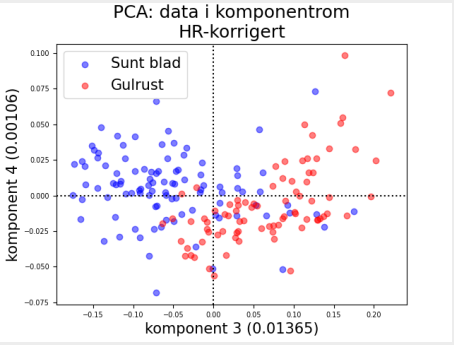


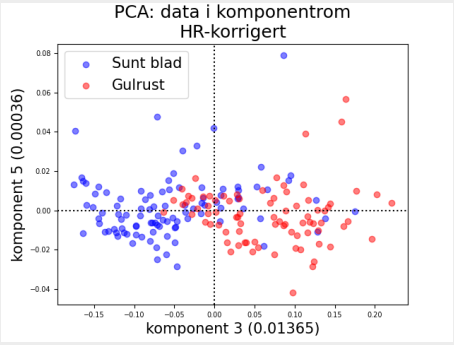


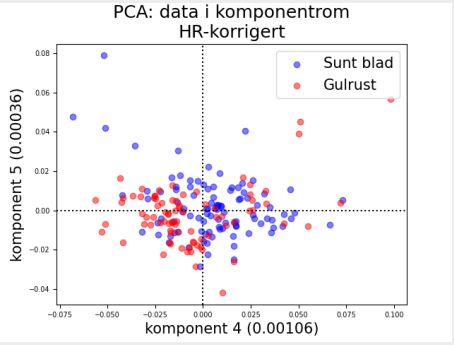




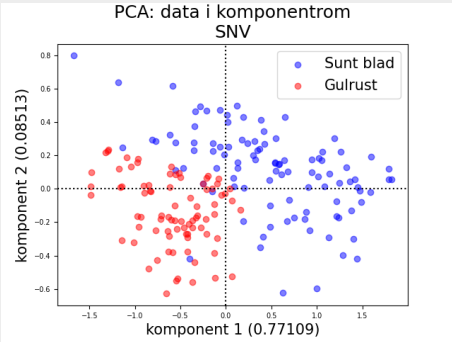


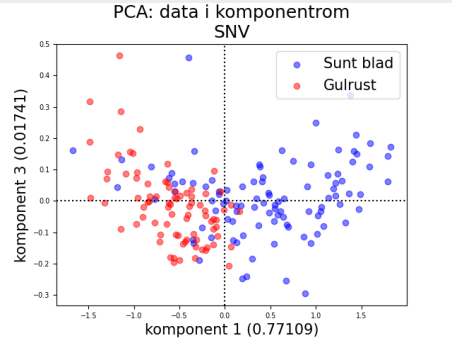


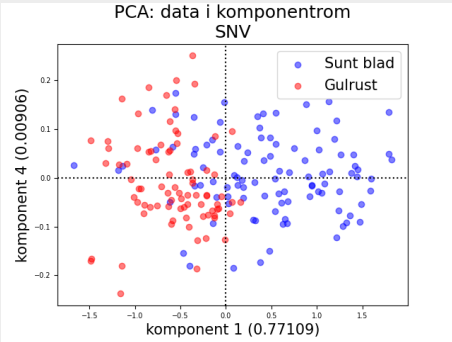


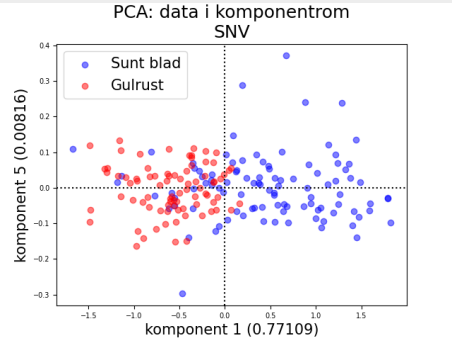


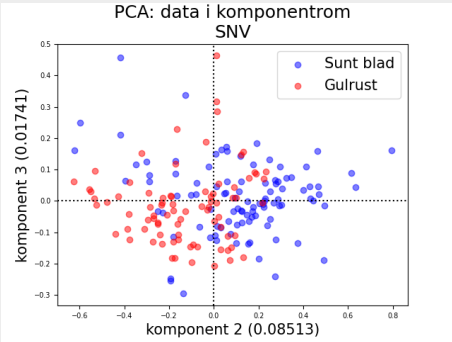
SNV Data i komponentrommet til PCA trent på SNV spektrum. Aksenavn viser til komponent og mengde variasjon komponenten representerer i desimaltall

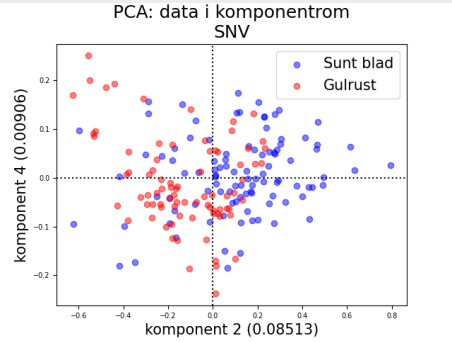


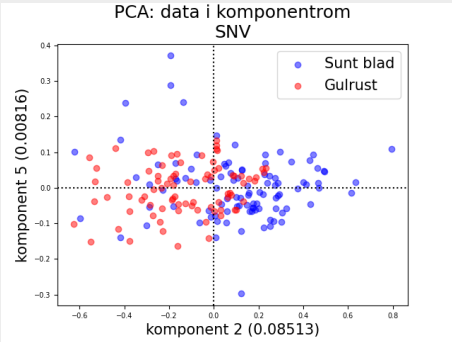


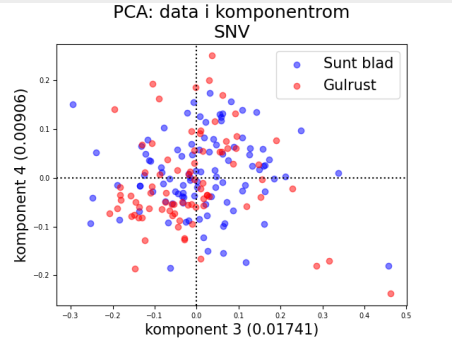


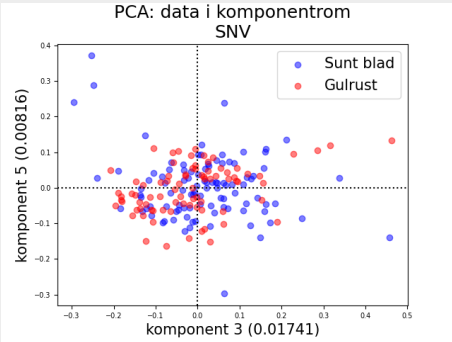


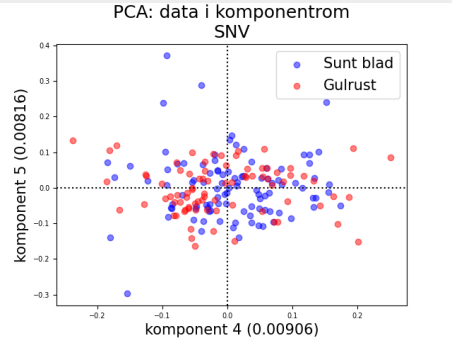




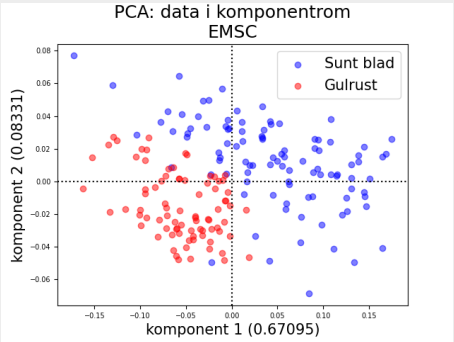


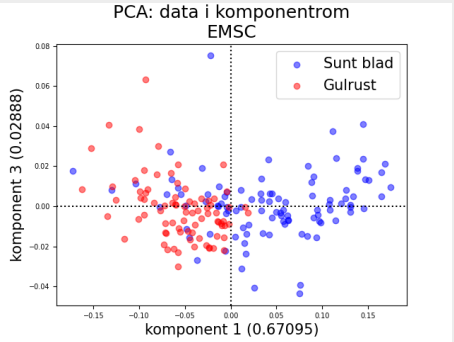


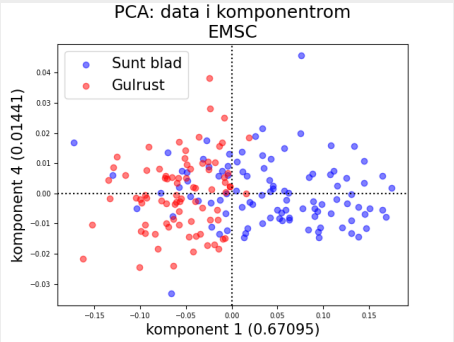


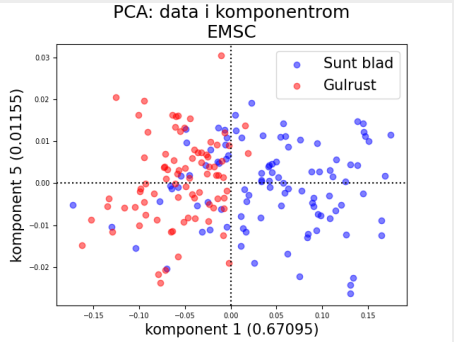


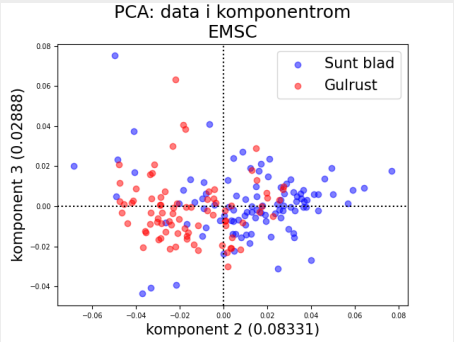
EMSC: Data i komponentrommet til PCA trent på EMSC spektrum. Aksenavn viser til komponent og mengde variasjon komponenten representerer i desimaltall

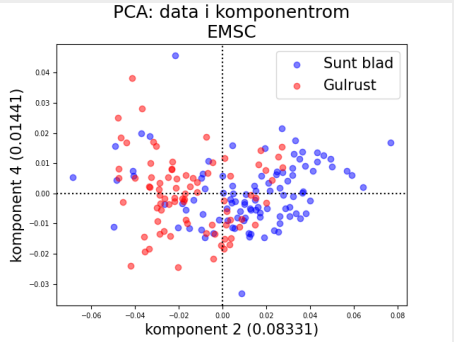


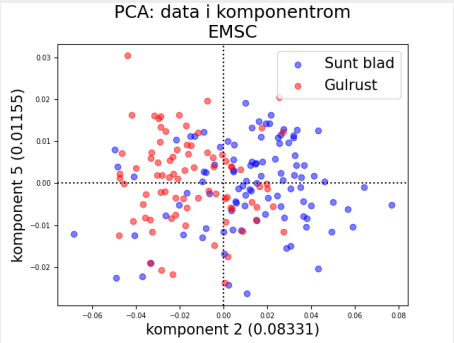


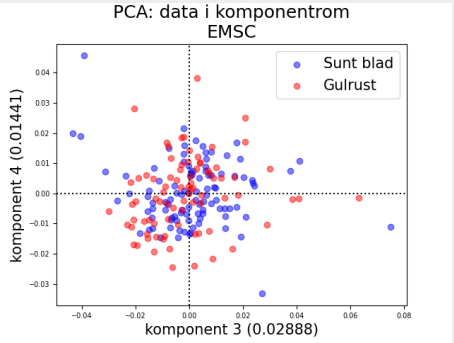


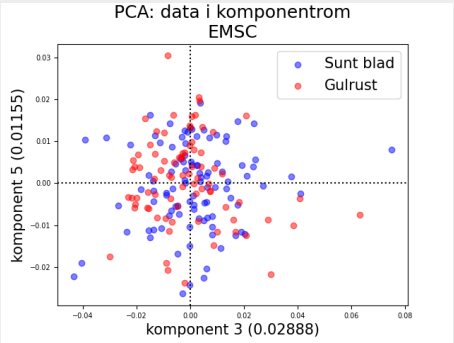


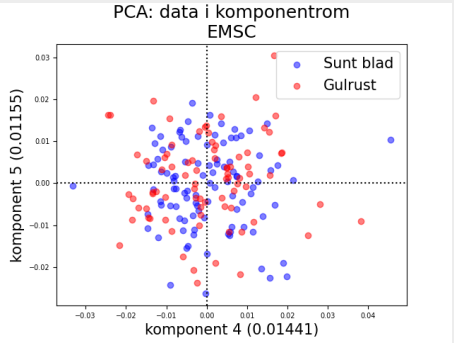




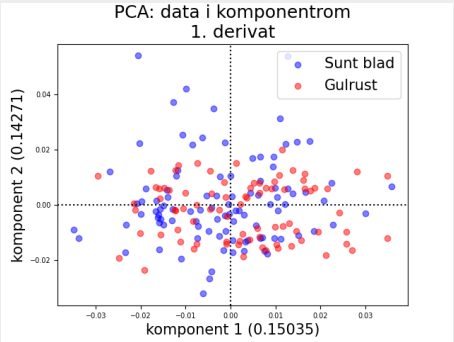


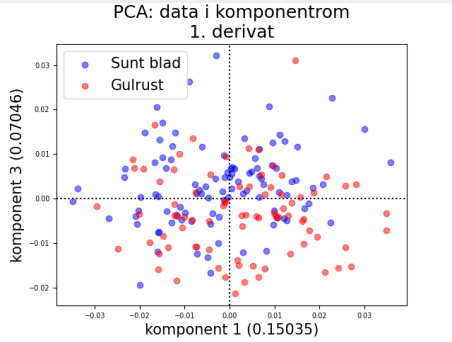


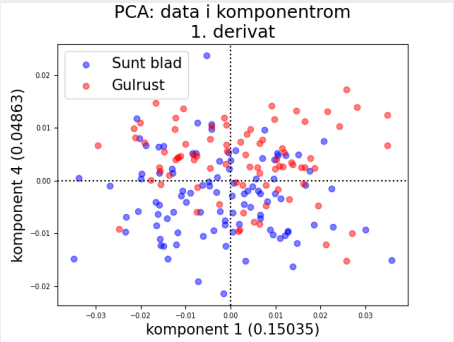


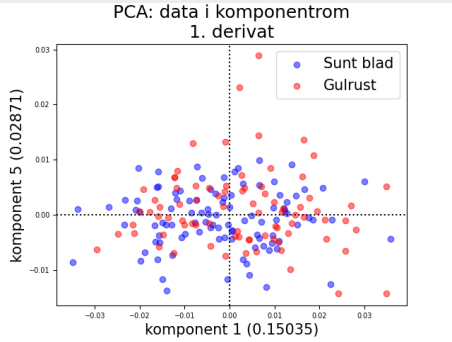


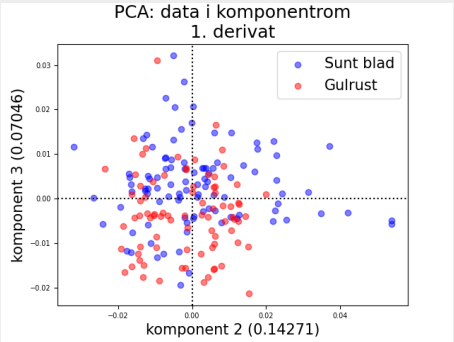
1. derivat: Data i komponentrommet til PCA trent på 1. derivat spektrum. Aksenavn viser til komponent og mengde variasjon komponenten representerer i desimaltall

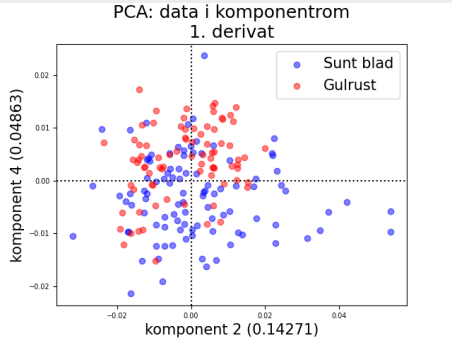


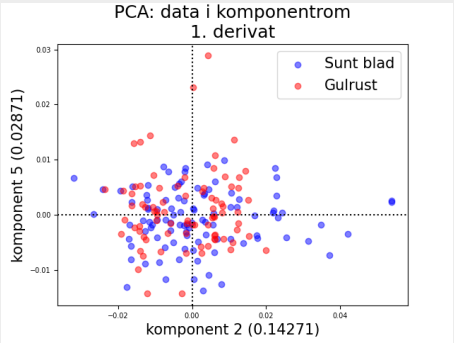


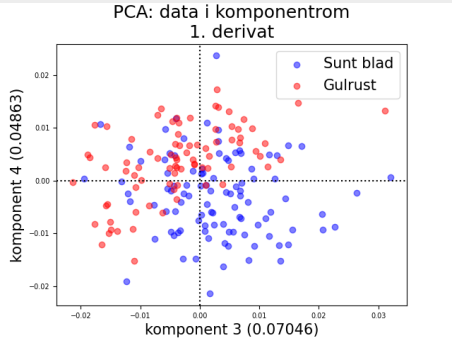


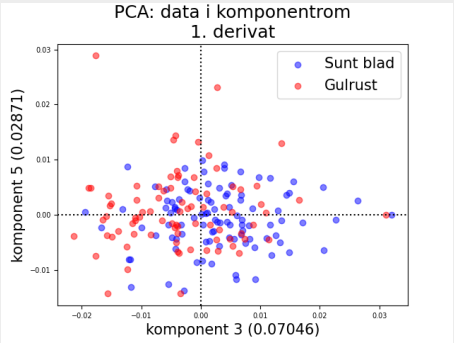


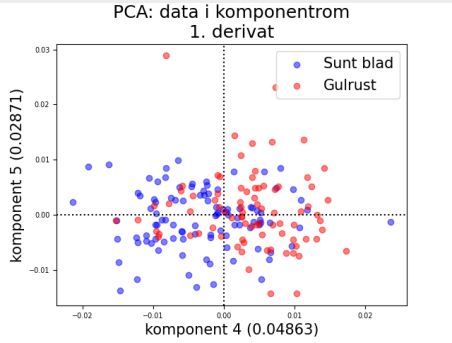




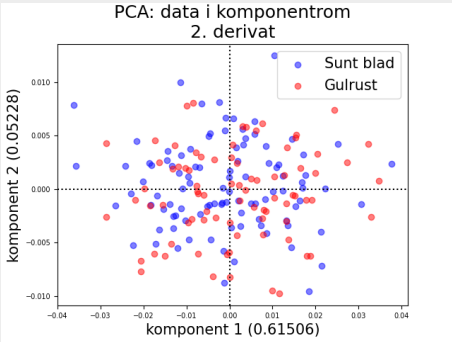


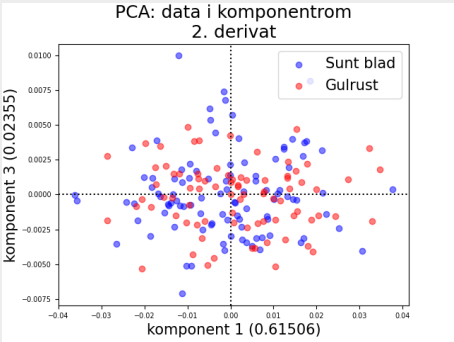


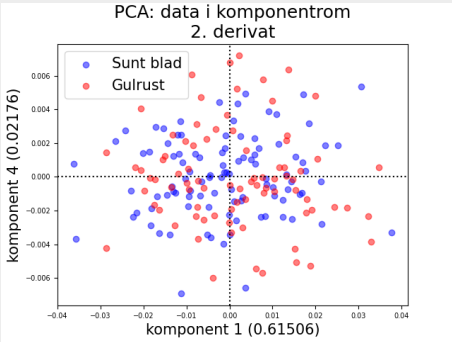


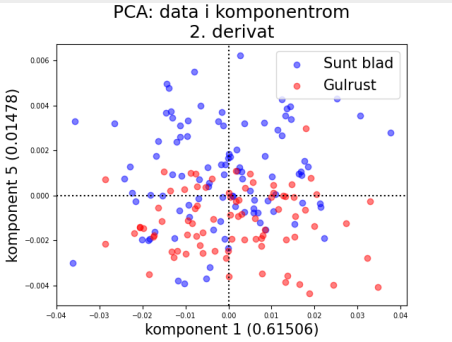


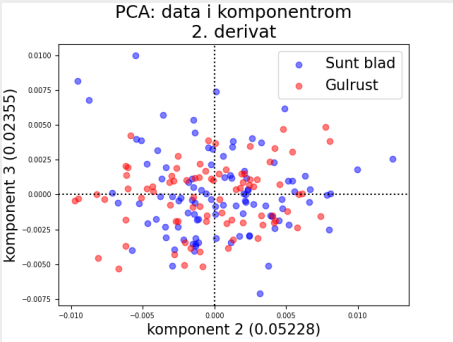
2. derivat: Data i komponentrommet til PCA trent på 2. derivat spektrum. Aksenavn viser til komponent og mengde variasjon komponenten representerer i desimaltall

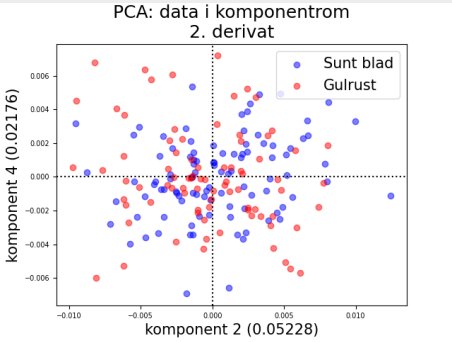


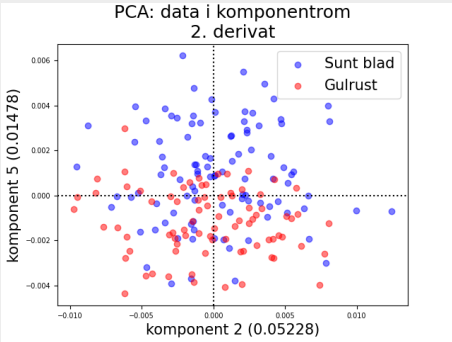


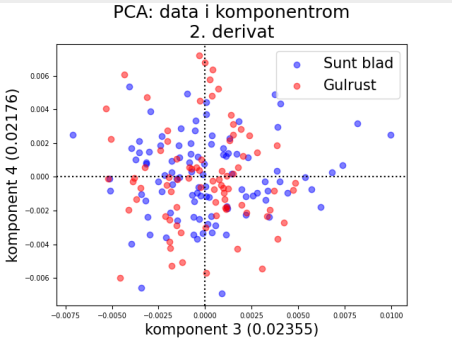


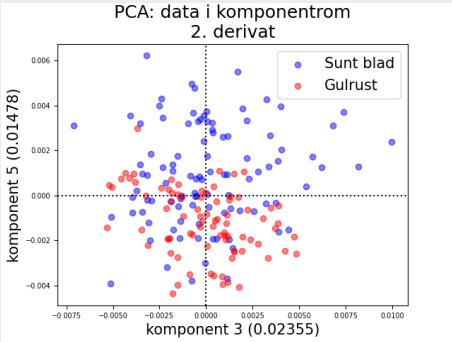


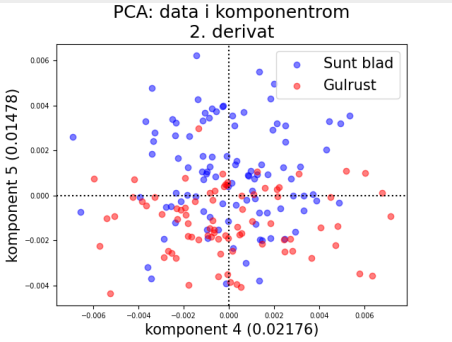






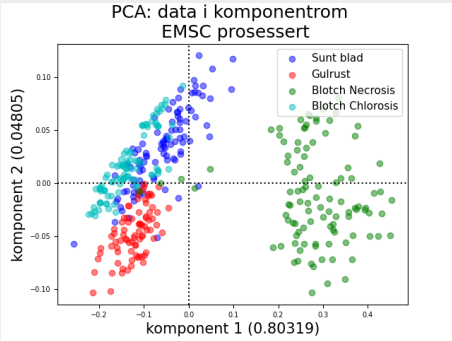


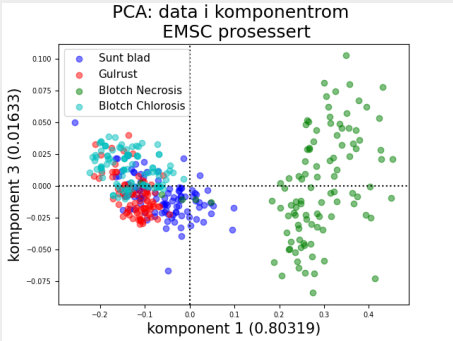


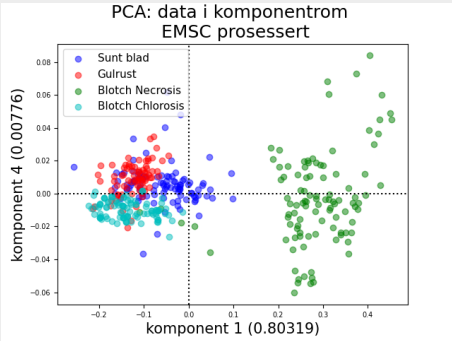


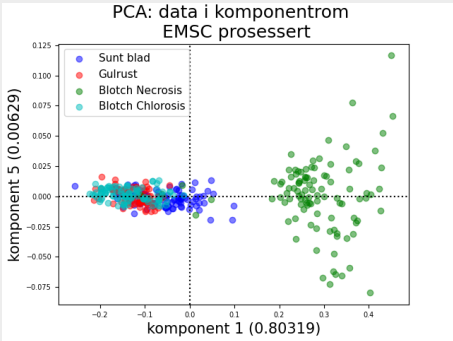
Vedlegg 2

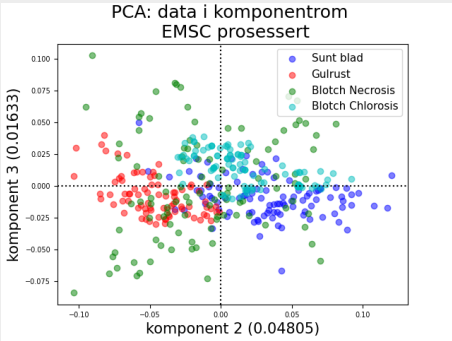
EMSC-prosessert: Data i komponentrommet til PCA trent på EMSC-prosessert spektrum. Aksenavn viser til komponent og mengde variasjon komponenten representerer i desimaltall

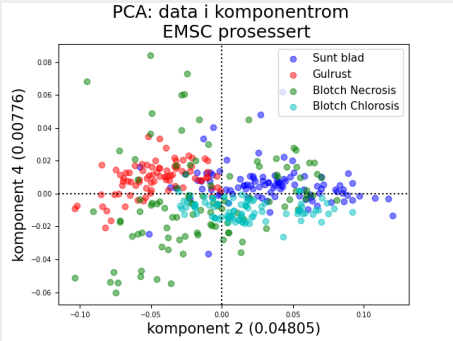


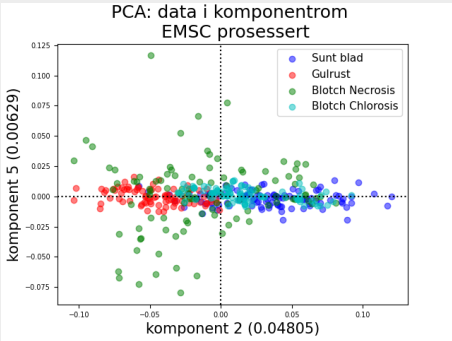


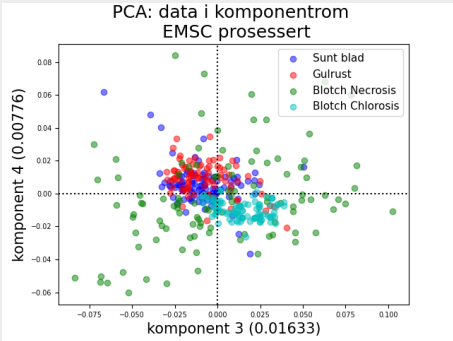


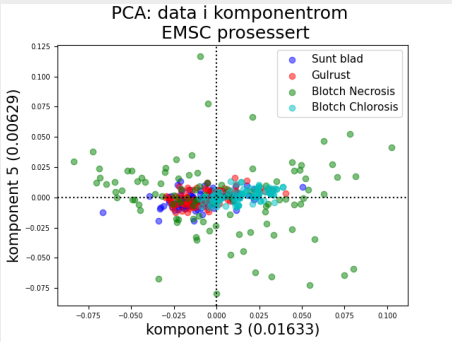


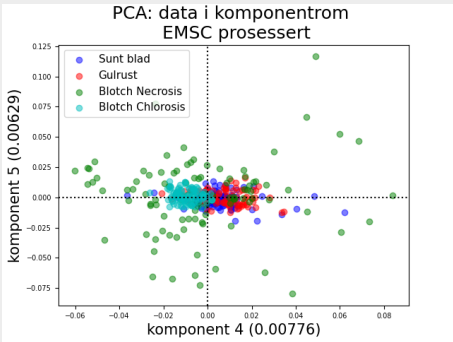




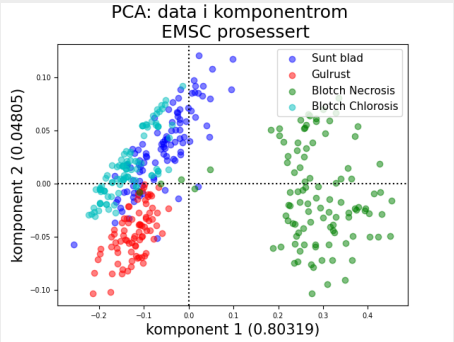


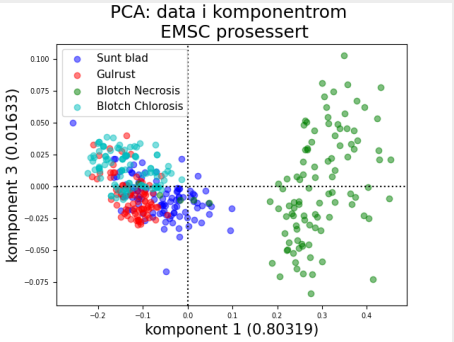


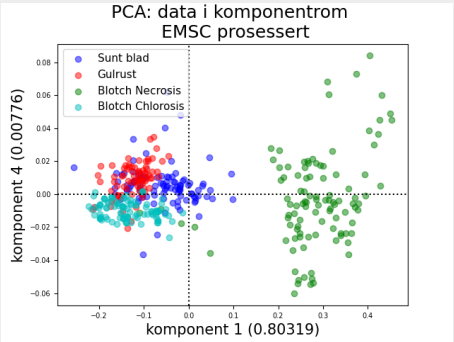


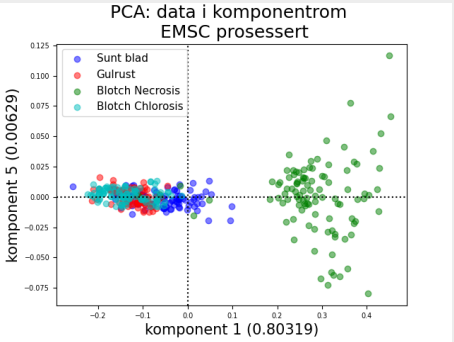


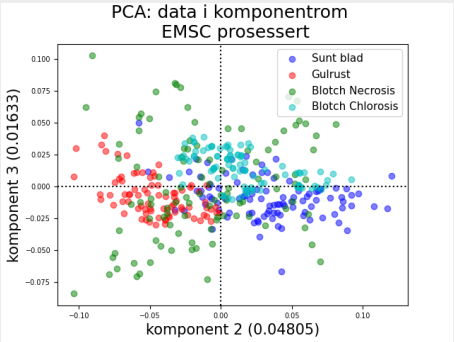
EMSC-prosessert: Data i komponentrommet til PCA trent på EMSC-prosessert spektrum. Aksenavn viser til komponent og mengde variasjon komponenten representerer i desimaltall

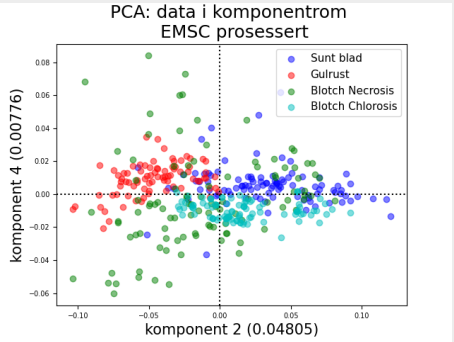


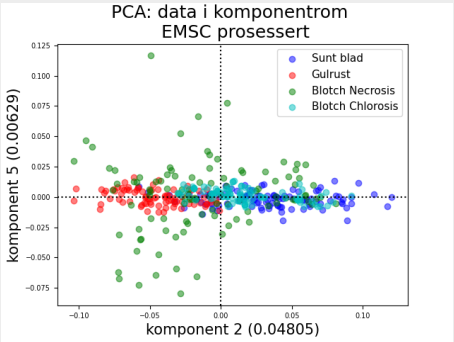


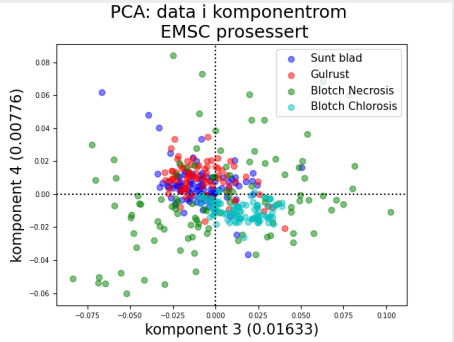


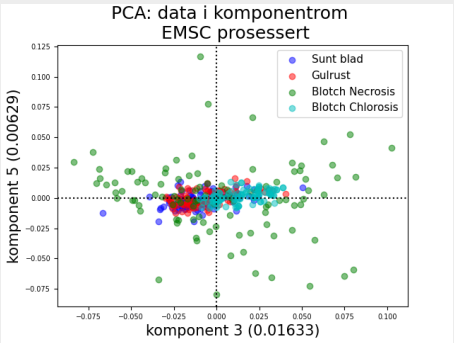


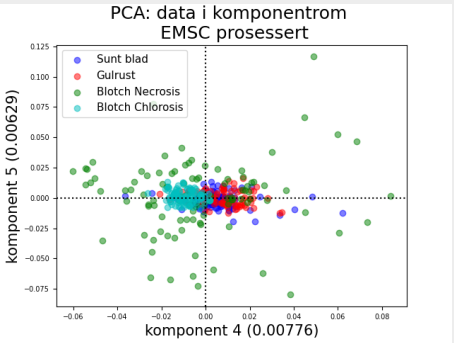












Vedlegg 3

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 13 2021 13:26

@author: Svein Jakob Kristoffersen (svein.jakob.kristoffersen@nmbu.no,
                                    sveinjakobkristoffersen@gmail.com)

Code to preprocess the leaf data for my master thesis. Some code from Maria
Vukovic.
"""

import os
from emsc import emsc
from math import log10
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import numpy as np
from spectral import envi
from spectral import kmeans, principal_components
from scipy.signal import savgol_filter
import pandas as pd
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, \
    recall_score, precision_score
from sklearn.model_selection import StratifiedKFold

path = os.path.dirname(__file__)
os.chdir(path)


def read_image_file(img_path, hdr_path):
    """Reads and returns and image and hdr file"""
    # Read in the image and the headerfile
    img_read = envi.open(hdr_path, img_path).load()
    hdr_read = open(hdr_path, 'r').read()  # open the hdr file
    return img_read, hdr_read


def white_reference_correction(img, area):
    """
    Perform correction with the white reference, given its area in image.

    ---- input -----
    img: spectral image
        Image with a white reference area in it
    area: string
        Area of the white reference in the image. Ex. '250:270, 400:470'
    ---- returns -----
    img_corr: spectral image
        The corrected image.
    """
    # Splitting area string into y- and x-ranges (strings)
    area_split = area.split(',')  # '250:270', '400:470'
    # Splittig the individual ranges into start and end values
    y_range_string = area_split[0].split(':')  # '250', '270'
    x_range_string = area_split[1].split(':')  # '400', '470'

    # Reaching start and end values and making slices
    y1, y2 = int(y_range_string[0]), int(y_range_string[1])
    x1, x2 = int(x_range_string[0]), int(x_range_string[1])
    y_range = slice(y1, y2)
    x_range = slice(x1, x2)

    # Slicing the area containing the white reference
```

```
    white_ref = img[y_range, x_range, :]
    # Median of the white reference
    white_ref_median = np.median(white_ref, axis=(0, 1))

    # Carry through white reference correction
    img_corr = img / white_ref_median
    return img_corr


def crop_to_area(img, area):
    """
    Crops image and returns cropped area.

    Will be used before k-means grouping to crop out the white reference for
    better groupig.

    ----- input -----
    img: spectral image
        Image to be cropped
    area: string
        Area of image to be cropped
    ----- returns -----
    img_cropped: spectral image
        The cropped image.
    """
    # Splitting area string into y- and x-ranges (strings)
    area_split = area.split(',')  # '250:270', '400:470'
    # Splittig the individual ranges into start and end values
    y_range_string = area_split[0].split(':')  # '250', '270'
    x_range_string = area_split[1].split(':')  # '400', '470'

    # Reaching start and end values and making slices
    y1, y2 = int(y_range_string[0]), int(y_range_string[1])
    x1, x2 = int(x_range_string[0]), int(x_range_string[1])
    y_range = slice(y1, y2)
    x_range = slice(x1, x2)

    img_cropped = img[y_range, x_range, :]
    return img_cropped


def group_with_kmeans(img):
    """
    Groups the pixels in image in two groups for leaf-background segmentation.

    To be used on cropped image containing leaf and background

    ----- input -----
    img: spectral image
        Image containing leaf and background to be segmented
    ----- returns -----
    leaf_pixels: list
        List containing all the pixels from the leaf
    mask: matrix
        matrix contaning the mask for leaf-background segmentation
    c: array(?)
        centers found with kmeans.
    """
    (mask, c) = kmeans(img, 2, 20)

    leaf_pixels = []
    for i in range(np.shape(mask)[0]):
        for j in range(np.shape(mask)[1]):
            if mask[i, j] == 1:
                leaf_pixels.append(img[i, j, :])
    leaf_pixels = np.array(leaf_pixels)
```

```
    return leaf_pixels, mask, c


def crop_out_area_pixels(img, areas):
    """
    Function to crop out the chosen areas.

    ----- input -----
    img: spectral image
        Image containing the rust pixels to crop out
    area: string
        Areas in image to crop out, format [ymin, ymax, xmin, xmax]
    ----- returns -----
    areas_: list
        Contains list of all areas' pixels from the image.
    """
    areas_ = []
    for area in areas:
        area_pixels = []
        ymin = area[0]
        ymax = area[1]
        xmin = area[2]
        xmax = area[3]

        # Append the crop to list:
        img_area = img[ymin:ymax, xmin:xmax, :]
        for i in range(img_area.shape[0]):
            for j in range(img_area.shape[1]):
                area_pixels.append(img_area[i, j, :])
        areas_.append(area_pixels)

    # Return the list after all areas have been cropped:
    return areas_


def preprocess_image(dict, snv_corr=True):
    """
    Performs preprocessing on the image, erg. white reference correction,
    leaf cropping and leaf-background segmentation using k-means grouping.

    ----- input -----
    dict: dictionary
        Contains path to image file, area of white ref, area of leaf-crop and
        possibly area of rust.
    ----- returns -----
    leaf_pixels: list
        List containing all the pixels from the leaf
    mask: matrix
        matrix contaning the mask for leaf-background segmentation
    c: array(?)
        centers found with k-means
    leaf: spectral image
        The white reference corrected and segmented leaf image.
    rust_pixels: list
        List containing all the rust pixels from the leaf (option)
    """
    path_img = dict['Path image']
    path_hdr = dict['Path hdr']
    ref_area = dict['Area w.r.']
    leaf_area = dict['Area leaf']
    if 'Areas rust' in dict.keys():  # If we have a rust area
        rust_areas = dict['Areas rust']
    if 'Areas healthy' in dict.keys():  # If we have a healthy area
        healthy_areas = dict['Areas healthy']
    if 'Areas senescence' in dict.keys():  # If we have a senescence area
        senescence_areas = dict['Areas senescence']
```

```
    # Read and WR. corr. the image:
    img, hdr_file = read_image_file(path_img, path_hdr)
    img = white_reference_correction(img, ref_area)
    # Crop out leaf area:
    leaf_img = crop_to_area(img, leaf_area)
    # Group the leaf pixels:
    leaf_pixels, mask, c = group_with_kmeans(leaf_img)

    # If areas, return with the cropped out pixels:
    if 'Areas rust' in dict.keys():
        area_pixels = crop_out_area_pixels(img, rust_areas)
    elif 'Areas healthy' in dict.keys():
        area_pixels = crop_out_area_pixels(img, healthy_areas)
    elif 'Areas senescence' in dict.keys():
        area_pixels = crop_out_area_pixels(img, senescence_areas)
    else:
        area_pixels = []

    # If snv is wanted:
    if snv_corr is True:
        leaf_pixels = snv(leaf_pixels)
        area_pixels = snv_on_list(area_pixels)

    return leaf_pixels, mask, leaf_img, area_pixels


def preprocess_images_in_dict(dict_of_dicts, snv_corr=True):
    """
    Functions which calls function preprocess_image on all images in given dict
    """
    # Preprocess all images and append them to lists -------------------------
    all_leaf_pixels = []
    all_leaf_images = []
    all_area_pixels = []
    i = 0
    for dict_ in dict_of_dicts:
        i += 1
        print(f'Processing image:  {i}')
        leaf_pixels, mask, leaf_img, area_pixels = preprocess_image(dict_,
                                                                    snv_corr=snv_corr)
        all_leaf_pixels.append(leaf_pixels)
        all_leaf_images.append(leaf_img)
        all_area_pixels.append(area_pixels)
    return all_leaf_pixels, all_leaf_images, all_area_pixels


def quickplot_image(path_img, path_hdr, area_wr=[0,0,0,0], area_leaf=[0,0,0,0],
                    bands=[0, 288],
                    clim=(0, 0.02)):
    """Simple image plot to check areas"""
    img, hdr = read_image_file(path_img, path_hdr)
    # plt.rc('font', size=8)
    dpi = 100
    plt.figure(figsize=(img.shape[1]/dpi, img.shape[0]/dpi), dpi=dpi)
    # Testing better visualization:
    plt.imshow(np.mean(img[:, :, bands[0]:bands[1]], axis=2),
               cmap='jet', clim=clim)
    plt.colorbar()

    # Drawing the white reference and leaf area
    ymin_wr = area_wr[0]; ymax_wr = area_wr[1]
    xmin_wr = area_wr[2]; xmax_wr = area_wr[3]
    ymin_l = area_leaf[0]; ymax_l = area_leaf[1]
    xmin_l = area_leaf[2]; xmax_l = area_leaf[3]
```

```
    # # White reference area:
    plt.gca().add_patch(Rectangle((xmin_wr, ymin_wr),
                                  (xmax_wr - xmin_wr), (ymax_wr - ymin_wr),
                                  linewidth=1, color='r', fill=False))
    # # Leaf area:
    plt.gca().add_patch(Rectangle((xmin_l, ymin_l),
                                  (xmax_l - xmin_l), (ymax_l - ymin_l),
                                  linewidth=1, color='r', fill=False))
    plt.show()
    return img

def snv(spectrum_list):
    """
    Performs standard normal variate correction on spectrums in a list.

    https://nirpyresearch.com/two-scatter-correction-techniques-nir-spectroscopy-python/
    """
    output_spectrum = np.zeros_like(spectrum_list)
    spectrum_list = np.array(spectrum_list)
    for i in range(spectrum_list.shape[0]):
        output_spectrum[i, :] = (spectrum_list[i,:] -
                                 np.mean(spectrum_list[i,:])) \
                                / np.std(spectrum_list[i,:])
    return output_spectrum


def snv_on_list(list_of_pixels):
    """
    Perform standard normal variate correction on a list of spectrums

    The only difference of this and the above function is the shape of input,
    more specifically the depth of the list erg. [[], []] vs [[ [], [] ]]
    """
    list_of_pixels_corr = []
    # for first_bracket in list_of_pixels:
    for pixel_list in list_of_pixels:
        list_of_pixels_corr.append(snv(pixel_list))
    return list_of_pixels_corr

def snv_single_spectrum(spectrum):
    """Perform standard normal variate correction on a single spectrum"""
    return (spectrum - np.mean(spectrum)) / np.std(spectrum)

def snv_on_image(img):
    """
    Performs standard normal variate correction correction on an entire image

    """
    img_corr = np.zeros(shape=(img.shape[0], img.shape[1], img.shape[2]))
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            img_corr[i, j, :] = snv_single_spectrum(img[i, j, :])
    return np.array(img_corr)


def _absorbance_value(value):
    """Function to convert input value (reflectance) to absorbance."""
    assert value > 0  # if value is less than zero, log10 not possible
    return log10(1/value)

def absorbance_spectrum(spectrum):
    """Function to convert input spectrum (reflectance) to absorbance."""
    assert spectrum.shape == (1, 288) or spectrum.shape == (288,), \
        f'Incorrect shape of input spectrum: {spectrum.shape}'
    new_spectrum = np.zeros_like(spectrum)
    for i, value in enumerate(spectrum):
```

```
        new_spectrum[i] = _absorbance_value(value)
    return new_spectrum


def absorbance_matrix(datamatrix):
    """
    Functino to convert matrix of spectra (reflectance) to matrix of
    absorbance spectra.

    """
    new_matrix = np.zeros_like(datamatrix)
    for i, spectrum in enumerate(datamatrix):
        new_matrix[i, :] = absorbance_spectrum(spectrum)
    return new_matrix


def _aquagram_single_value(value, mean, std):
    """Applies formula for aquagram value on a single value"""
    return (value - mean)/std


def aquagram_values(matrix1, matrix2, matrix3, matrix4):
    """Function to turn all values in given bands into aquagram values"""
    m1_range = range(0, matrix1.shape[0])
    m2_range = range(m1_range[-1] + 1, m1_range[-1] + matrix2.shape[0] + 1)
    m3_range = range(m2_range[-1] + 1, m2_range[-1] + matrix3.shape[0] + 1)
    m4_range = range(m3_range[-1] + 1, m3_range[-1] + matrix4.shape[0] + 1)

    X = np.concatenate([matrix1, matrix2, matrix3, matrix4])
    X_new = np.zeros_like(X)
    for column in range(X.shape[1]):
        mean = np.mean(X[:, column])
        std = np.std(X[:, column])
        for row in range(X.shape[0]):
            X_new[row, column] = _aquagram_single_value(X[row, column],
                                                        mean, std)
    return X_new[m1_range, :], X_new[m2_range, :], X_new[m3_range, :], \
           X_new[m4_range, :]


def pca(image):
    """Performs PCA on an entire image using the spectral package"""
    pc = principal_components(image)
    pc_0999 = pc.reduce(fraction=0.999)
    loading = pc_0999.eigenvectors
    score = pc_0999.transform(image)
    return loading, score


def _first_derivative_of_list(list_with_spectrums, savgol=True,
                              window_length=7, polyorder=5):
    """
    Returns the first derivative of all spectrums in passed list. Assumes dx=1.
    """
    first_derivative_spectrums = []
    for spectrum in list_with_spectrums:
        first_der = np.diff(spectrum)
        if savgol:
            first_der = savgol_filter(first_der, window_length, polyorder)
        first_derivative_spectrums.append(first_der)
    return np.array(first_derivative_spectrums)


def _second_derivative_of_list(list_with_spectrums, savgol=True,
                               window_length=7, polyorder=5):
    """
```

```
    Returns the seoncd derivative of all spectrums in passed list. Assumes
    dx=1.
    """
    second_derivative_spectrums = []
    for spectrum in list_with_spectrums:
        first_der = np.diff(spectrum)
        second_der = np.diff(first_der)
        if savgol:
            second_der = savgol_filter(second_der, window_length, polyorder)
        second_derivative_spectrums.append(second_der)
    return np.array(second_derivative_spectrums)


def make_first_and_second_der(pixel_list):
    """
    Takes a list of spectrums as input and returns them as first and second
    derivatives in two lists.
    """
    first_der = _first_derivative_of_list(pixel_list, savgol=True,
                                          window_length=7, polyorder=5)
    # Arrays with second derivative of spectrums
    second_der = _second_derivative_of_list(pixel_list, savgol=True,
                                            window_length=11, polyorder=3)
    return first_der, second_der


def make_x_y(list_of_lists):
    """
    Makes X and y matrices of list of lists for use in machine learning.
    Each list in list of lists correspond to a class.
    """
    y = []
    X = []

    i = 0
    for single_list in list_of_lists:
        y.extend(i for _ in range(len(single_list)))
        i += 1
        X.extend(single_list)
    return np.array(X), np.array(y)

def give_deviation_graphs(list_of_spectrums):
    """Returns mean and std of list of spectrums"""
    std = np.std(list_of_spectrums, axis=0)
    mean_spectrum = np.mean(list_of_spectrums, axis=0)
    return mean_spectrum + std, mean_spectrum - std

class PLSDA_crossval():
    """Dummyclass used to replicate the behaviour of a real classifier."""
    def __init__(self, n_components, n_classes=2):
        self.n_components = n_components
        self.plsr = PLSRegression(n_components=n_components)

        self.y_true = None
        self.y_pred = None
        assert n_classes in [2, 4] # Function can only handle 2 or 4 classes
        self.n_classes = n_classes

    def f1(self):
        return f1_score(y_true=self.y_true, y_pred=self.y_pred,
                        average='weighted')

    def precision(self):
        return precision_score(y_true=self.y_true, y_pred=self.y_pred,
                               average='weighted')
```

```
    def recall(self):
        return recall_score(y_true=self.y_true, y_pred=self.y_pred,
                            average='weighted')

    def confm(self):
        return confusion_matrix(y_true=self.y_true, y_pred=self.y_pred)

    def statistics(self, y_true=None, y_pred=None):
        """Returns various statistics"""
        self.y_true = y_true
        self.y_pred = y_pred
        return self.f1(), self.precision(), self.recall(), self.confm()

    def cross_val(self, cv=5, X=None, y_true=None):
        """Performs crossvalidation on input data and returns statistics"""
        self.plsr = PLSRegression(n_components=self.n_components)
        strat_kfold = StratifiedKFold(n_splits=cv)
        outer_acc = []
        outer_f1 = []
        outer_rec = []
        outer_pre = []
        if self.n_classes == 2:
            outer_confm = np.zeros(shape=(2, 2))
        elif self.n_classes == 4:
            outer_confm = np.zeros(shape=(4, 4))

        for train_i, test_i in strat_kfold.split(X, y_true):
            self.plsr.fit(X[train_i], y_true[train_i])
            y_pred = self.plsr.predict(X[test_i])
            if self.n_classes == 2:
                y_pred = (y_pred > 0.5).astype('uint8')  # Making the pred 1/0
            elif self.n_classes == 4:
                y_pred = np.around(y_pred).astype(int)
                y_pred = np.clip(y_pred, 0, 3)

            outer_acc.append(accuracy_score(y_true[test_i], y_pred))

            f1, prec, reca, confm = self.statistics(y_true=y_true[test_i],
                                                    y_pred=y_pred)

            outer_f1.append(f1)
            outer_pre.append(prec)
            outer_rec.append(reca)
            outer_confm = outer_confm + confm
            print(confm)

        return np.mean(outer_acc), np.std(outer_acc), np.mean(outer_f1), \
               np.mean(outer_pre), np.mean(outer_rec), outer_confm


def classifier_img_pred(X, y, img, mask, pca=None, classifier=None,
                        sensitive_bands=None, n_classes=2, emsc_corr=False,
                        emsc_degree=0, emsc_Xref=None):
    """
    Function to classify pixels on passed hyperspectral image with passed
    classifier trained on passed data.

    Parameters
    ----------
    X: array
        Matrix containing spectrums
    y: array
        Array containing classes of spectrums in X
    img: array
        3D matrix (hypercube) containing spectrums to be classified
    mask: array
```

```
        2D binary matrix masking leaf on img
    pca: class
        PCA from sklearn to use if PCA transformation needed
    classifier: class
        Sklearn class to classify spectrums from img
    sensitive_bands: array
        Bands to crop img if necessary
    n_classes: int
        2 or 4 classes (method 1 or 2)
    emsc_corr: bool
        If emsc is necessary
    emsc_degree: int
        Degree of emsc
    emsc_Xref: array
        Reference spectrum to use in emsc

    Returns
    -------
    classified_img: array
        2D matrix of img classified

    """
    classifier.fit(X, y)
    # Using this classifyer to classify pixels on image:
    img_to_classify = img
    img_classified = np.zeros(shape=(img_to_classify.shape[0],
                                     img_to_classify.shape[1]))
    for i in range(img_to_classify.shape[0]):
        for j in range(img_to_classify.shape[1]):
            if mask[i,j] == 0:  # If this pixel is background
                img_classified[i,j] = -1
            else:  # If this pixel is leaf
                spectrum = img_to_classify[i, j, :].reshape(-1, 1).T

                if pca is not None:
                    spectrum = pca.transform(spectrum)
                if emsc_corr is True:
                    spectrum = emsc(X=spectrum, ref_spec=emsc_Xref,
                                    d=emsc_degree)
                if sensitive_bands is not None:
                    spectrum = spectrum[:, sensitive_bands]

                pred = classifier.predict(spectrum)
                if n_classes == 2:
                    if pred < 0.5:
                        img_classified[i, j] = 0
                    elif 0.5 < pred:
                        img_classified[i, j] = 1
                elif n_classes == 4:
                    if -0.5 < pred < 0.5:
                        img_classified[i, j] = 0
                    elif 0.5 < pred < 1.5:
                        img_classified[i, j] = 1
                    elif 1.5 < pred < 2.5:
                        img_classified[i, j] = 2
                    elif 2.5 < pred < 3.5:
                        img_classified[i, j] = 3
    return img_classified


def plot_hypercube(cube, mask, bands, Xref_emsc=False, emsc_degree=2,
                   clim=(0,1)):
    """Visualizing the hypercube, applying EMSC if necessary"""
    for i in range(cube.shape[0]):
        for j in range(cube.shape[1]):
            if mask[i,j] == 0:  # If this pixel is background
```

```
                cube[i,j] = 0
            elif Xref_emsc is not None:
                cube[i, j, :], = emsc(X=cube[i, j, :].reshape(-1, 1).T,
                                      ref_spec=Xref_emsc, d=emsc_degree)
    cube = cube[:, :, bands]
    plt.figure(figsize=(5, 12))
    plt.imshow(cube, clim=clim, interpolation='None')
    plt.colorbar()
    plt.show()


def reach_area_pixels(df_, class_, is_healthy=False):
    """Returns wanted areas from dataframe, used for abstraction"""
    if is_healthy:
        area_pixels_ = df_.loc[df_['class'] == class_, 'leaf pixels']
        area_pixels = []
        for area in area_pixels_:
            area_pixels.extend(area)
    else:
        area_pixels_ = df_.loc[df_['class'] == class_, 'area pixels']
        area_pixels = []
        for area in area_pixels_:
            area_pixels.extend(area)
    return np.array(area_pixels)


def undersample_list(majority_list, new_size=None):
    """
    Performs random removal of elements in a list, until it reaches wanted
    size
    """
    indices = np.random.choice(majority_list.shape[0], new_size)
    return majority_list[indices]


def get_wavelength(keep_floats=False):
    """
    Returns the wavelengths used for one hdr-file, common wavelengths for
    all images
    """
    hdr = r'C:\Users\svein\Desktop\Rust\leaves160720\rust_01.hdr'
    hdr_file = open(hdr, 'r').read()  # open the hdr file
    wavelength = hdr_file[148:2989 + 7].split(',')
    if keep_floats:
        return np.array([float(wavelength[i])
                         for i in range(len(wavelength))])
    else:
        return np.array([int(float(wavelength[i]))
                         for i in range(len(wavelength))])


# ----------------------------------------------------------------------------
def get_areas_from_img(img, areas):
    """
    Function to crop out the chosen areas.

    ----- input -----
    img: spectral image
        Image containing the rust pixels to crop out
    area: string
        Areas in image to crop out, format [ymin, ymax, xmin, xmax]
    ----- returns -----
    areas_: list
        Contains list of all areas' pixels from the image.
    """
    areas_ = []
```

```
    for area in areas:
        area_pixels = []
        ymin = area[0]
        ymax = area[1]
        xmin = area[2]
        xmax = area[3]

        # Append the crop to list:
        img_area = img[ymin:ymax, xmin:xmax, :]
        for i in range(img_area.shape[0]):
            for j in range(img_area.shape[1]):
                area_pixels.append(img_area[i, j, :])
        areas_.extend(area_pixels)

    # Return the list after all areas have been cropped:
    return np.array(areas_)


def preprocess_leaf_img(dict, snv_corr=True):

    """
    Performs preprocessing on the image, erg. white reference correction,
    leaf cropping and leaf-background segmentation using k-means grouping and
    and optional SNV correction.

    ----- input -----
    dict: dictionary
        Contains path to image file, area of white ref, area of leaf-crop and
        possibly area of choosing.
    ----- returns -----
    leaf_pixels: list
        List containing all the pixels from the leaf
    mask: matrix
        matrix contaning the mask for leaf-background segmentation
    leaf_img: spectral image
        The white reference corrected and segmented leaf image.
    area_pixels: list
        List containing all the area pixels from the leaf (option)
    """
    path_img = dict['Path image']
    path_hdr = dict['Path hdr']
    ref_area = dict['Area w.r.']
    leaf_area = dict['Area leaf']
    areas_of_interest = dict['AOI']

    # Read and WR. corr. the image:
    img, hdr_file = read_image_file(path_img, path_hdr)
    img = white_reference_correction(img, ref_area)
    # Crop out leaf area:
    leaf_img = crop_to_area(img, leaf_area)
    # Group the leaf pixels:
    print('Starting Kmeans')
    leaf_pixels, mask, c = group_with_kmeans(leaf_img)
    print('Kmeans ended')

    # If areas, return with the cropped out pixels:
    print('Cropping out AOI')
    area_pixels = get_areas_from_img(img, areas_of_interest)
    print(f'AOI cropped: shape{np.shape(area_pixels)}')

    # If snv is wanted:
    if snv_corr is True:
        print('SNV correction applying')
        leaf_pixels = snv(leaf_pixels)
        area_pixels = snv(area_pixels)
        leaf_img = snv_on_image(leaf_img)
```

```
        print('SNV correction applied')

    return leaf_pixels, mask, leaf_img, area_pixels


def read_data_to_df(dicts, snv_corr=True):
    """Reads all data from preprocessing into a dataframe"""
    df = pd.DataFrame()
    for dict_ in dicts:
        leaf_pixels, mask, img, area_pixels = \
            preprocess_leaf_img(dict_, snv_corr=snv_corr)
        df_leaf = pd.DataFrame(columns=['class', 'leaf pixels', 'area pixels',
                                        'leaf number', 'image', 'mask'])
        df_leaf['class'] = [dict_['Class']]
        df_leaf['leaf pixels'] = [leaf_pixels]
        df_leaf['area pixels'] = [area_pixels]
        df_leaf['leaf number'] = [dict_['Path image'].split('\\')[-1][:-4]]
        df_leaf['image'] = [img]
        df_leaf['mask'] = [mask]
        df = df.append(df_leaf)
    return df


if __name__ == '__main__':
    path_folder160720 = r'C:\Users\svein\Desktop\Rust\leaves160720'
    path_folder230720 = r'C:\Users\svein\Desktop\Rust\leaves230720'
    leaves_160720 = [
        {
            'Path image': path_folder160720 + r'\rust_02.img',
            'Path hdr': path_folder160720 + r'\rust_02.hdr',
            'Area w.r.': '50:380, 220:350',
            'Area leaf': '120:560, 50:130',
            'Class': 'rust',
            'AOI': [[448,450,88,90], [452,458,89,90]]},
        {
            'Path image': path_folder160720 + r'\rust_03.img',
            'Path hdr': path_folder160720 + r'\rust_03.hdr',
            'Area w.r.': '50:380, 220:350',
            'Area leaf': '110:500, 50:130',
            'Class': 'rust',
            'AOI': [[274, 280, 101, 102], [302, 304, 101, 102], [312, 314, 107,
                                                                 108]]},
    ]

    other_leaves_160720_rust = [
        {
            'Path image': path_folder160720 + r'\rust_01.img',
            'Path hdr': path_folder160720 + r'\rust_01.hdr',
            'Area w.r.': '50:380, 220:350',
            'Area leaf': '100:550, 50:130',
            'Class': 'rust',
            'AOI': [],},
        {
            'Path image': path_folder160720 + r'\rust_04.img',
            'Path hdr': path_folder160720 + r'\rust_04.hdr',
            'Area w.r.': '50:380, 220:350',
            'Area leaf': '130:670, 50:130',
            'Class': 'rust',
            'AOI': [],},
    ]

    df = read_data_to_df(other_leaves_160720_rust, snv_corr=False)
    df.to_pickle(r'C:\Users\svein\Desktop\leaf_dataframe_other_leaves.csv')


#============================================================================
```

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 13 2021 13:26

@author: Svein Jakob Kristoffersen (svein.jakob.kristoffersen@nmbu.no)

Code to handle method 1 data containing 2 classes. This script will
attempt classifying based on different datatypes, highlighting appropriate data
for problem and further work.
"""

import os
from array import array

from leaf_processing_1_new import *
from emsc import emsc
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from matplotlib.patches import Rectangle
import numpy as np
from sklearn.model_selection import StratifiedKFold, cross_val_score, \
    cross_val_predict, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, \
    recall_score, precision_score
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
import random
random.seed(14)
np.random.seed(13)
plt.rc('xtick', labelsize=7)
plt.rc('ytick', labelsize=7)
plt.rc('font', size=15)

# read data of leaves used for image prediction:
df_image_pred = pd.read_pickle(r'C:\Users\svein\Desktop\leaf_dataframe_other_leaves.csv')
# read data
df = pd.read_pickle(r'C:\Users\svein\Desktop\leaf_dataframe_noSNV.csv')
df = df[df['class'].isin(['rust', 'healthy'])]

colors = ['black', 'r', 'b']  # Colors for use in plotting
wavelength = get_wavelength()

# Setting up data:
pixels_rust = reach_area_pixels(df, 'rust')
pixels_healthy = reach_area_pixels(df, 'healthy', is_healthy=True)
Xref_emsc = np.mean(pixels_healthy, axis=0)
pixels_healthy = undersample_list(pixels_healthy, new_size=100)
pixels_rust_1der, pixels_rust_2der = make_first_and_second_der(pixels_rust)
pixels_healthy_1der, pixels_healthy_2der = make_first_and_second_der(pixels_healthy)
pixels_rust_snv = snv(pixels_rust)
pixels_healthy_snv = snv(pixels_healthy)

pixels_rust_emsc = emsc(X=pixels_rust, ref_spec=Xref_emsc, d=2)
pixels_healthy_emsc = emsc(X=pixels_healthy, ref_spec=Xref_emsc, d=2)

# #%% Visualize data ------------------------------------------------------
# # All spectrums white corrected ------
std_pos_rust, std_neg_rust = give_deviation_graphs(pixels_rust)
std_pos_healthy, std_neg_healthy = give_deviation_graphs(pixels_healthy)
plt.figure(figsize=(8, 6))
plt.plot(np.mean(pixels_healthy.T, axis=1), 'b')
```

```
plt.plot(np.mean(pixels_rust.T, axis=1), 'r')
plt.title('Klassespektrum gjennomsnitt og standardavvik \n'
          'HR-korrigert')
plt.legend(['Sunt blad', 'Gulrust'])
plt.xticks(ticks=range(0, 288, 50), labels=wavelength[0:-1:50])
plt.ylabel('Reflektanse')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.ylim(0, 0.6)
plt.show()
# # -------------------------------------
# # 1. der spectrums:
std_pos_rust, std_neg_rust = give_deviation_graphs(pixels_rust_1der)
std_pos_healthy, std_neg_healthy = give_deviation_graphs(pixels_healthy_1der)
plt.figure(figsize=(8, 6))
plt.plot(np.mean(pixels_healthy_1der.T, axis=1), 'b')
plt.plot(np.mean(pixels_rust_1der.T, axis=1), 'r')
plt.fill_between(x=range(287), y1=std_pos_rust, y2=std_neg_rust,
                 alpha=0.2, color='r')
plt.fill_between(x=range(287), y1=std_pos_healthy, y2=std_neg_healthy,
                 alpha=0.2, color='b')
plt.title('Klassespektrum gjennomsnitt og standardavvik \n'
          '1. derivat')
plt.legend(['Sunt blad', 'Gulrust'], loc='lower right')
plt.xticks(ticks=range(0, 288, 50), labels=wavelength[0:-1:50])
plt.ylabel('Reflektanse')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.ylim(-0.04, 0.013)
plt.show()
# # -------------------------------------
# # 2. der spectrums:
std_pos_rust, std_neg_rust = give_deviation_graphs(pixels_rust_2der)
std_pos_healthy, std_neg_healthy = give_deviation_graphs(pixels_healthy_2der)
plt.figure(figsize=(8, 6))
plt.plot(np.mean(pixels_healthy_2der.T, axis=1), 'b')
plt.plot(np.mean(pixels_rust_2der.T, axis=1), 'r')
plt.fill_between(x=range(286), y1=std_pos_rust, y2=std_neg_rust,
                 alpha=0.2, color='r')
plt.fill_between(x=range(286), y1=std_pos_healthy, y2=std_neg_healthy,
                 alpha=0.2, color='b')

plt.title('Klassespektrum gjennomsnitt og standardavvik \n'
          '2. derivat')
plt.legend(['Sunt blad', 'Gulrust'], loc='lower right')
plt.xticks(ticks=range(0, 288, 50), labels=wavelength[0:-1:50])
plt.ylabel('Reflektanse')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.ylim(-0.007, 0.007)
plt.show()
# # -------------------------------------
# # SNV corrected spectrums:
std_pos_rust, std_neg_rust = give_deviation_graphs(pixels_rust_snv)
std_pos_healthy, std_neg_healthy = give_deviation_graphs(pixels_healthy_snv)
plt.figure(figsize=(8, 6))
plt.plot(np.mean(pixels_healthy_snv.T, axis=1), 'b')
plt.plot(np.mean(pixels_rust_snv.T, axis=1), 'r')
plt.fill_between(x=range(288), y1=std_pos_rust, y2=std_neg_rust,
                 alpha=0.2, color='r')
plt.fill_between(x=range(288), y1=std_pos_healthy, y2=std_neg_healthy,
                 alpha=0.2, color='b')

plt.title('Klassespektrum gjennomsnitt og standardavvik \n'
          'SNV')
plt.legend(['Sunt blad', 'Gulrust'])
```

```
plt.xticks(ticks=range(0, 288, 50), labels=wavelength[0:-1:50])
plt.ylabel('Reflektanse')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.show()
# -----------------------------------
# # EMSC corrected spectrums:
std_pos_rust, std_neg_rust = give_deviation_graphs(pixels_rust_emsc)
std_pos_healthy, std_neg_healthy = give_deviation_graphs(pixels_healthy_emsc)
plt.figure(figsize=(8, 6))
plt.plot(np.mean(pixels_healthy_emsc.T, axis=1), 'b')
plt.plot(np.mean(pixels_rust_emsc.T, axis=1), 'r')
plt.fill_between(x=range(288), y1=std_pos_rust, y2=std_neg_rust, alpha=0.2,
                 color='r')
plt.fill_between(x=range(288), y1=std_pos_healthy, y2=std_neg_healthy,
                 alpha=0.2, color='b')

plt.title('Klassespektrum gjennomsnitt og standardavvik \n'
          'EMSC')
plt.legend(['Sunt blad', 'Gulrust'])
plt.xticks(ticks=range(0, 288, 50), labels=wavelength[0:-1:50])
plt.ylabel('Reflektanse')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.ylim(0, 0.5)
plt.show()

#%% X, y data ------------------------------------------------------------------
X, y = make_x_y([pixels_rust, pixels_healthy])
X_1der, y = make_x_y([pixels_rust_1der, pixels_healthy_1der])
X_2der, y = make_x_y([pixels_rust_2der, pixels_healthy_2der])
X_snv, y = make_x_y([pixels_rust_snv, pixels_healthy_snv])
X_emsc, y = make_x_y([pixels_rust_emsc, pixels_healthy_emsc])

#%% PCA ------------------------------------------------------------------------
pca = PCA(n_components=5)
pca.fit(X_emsc)
X_t_pca = pca.transform(X)
print(sum(pca.explained_variance_ratio_))

variance = pca.explained_variance_ratio_
components = [1, 2, 3, 4, 5]
for comp_x in components:
    for comp_y in components[comp_x:]:
        fig, ax = plt.subplots()

        ax.scatter(X_t_pca[82:, comp_x-1], X_t_pca[82:, comp_y-1], c='b',
                   alpha=0.5)
        ax.scatter(X_t_pca[:82, comp_x-1], X_t_pca[:82, comp_y-1], c='r',
                   alpha=0.5)

        plt.xlabel(f'komponent {comp_x} ({variance[comp_x-1]:.5f})')
        plt.ylabel(f'komponent {comp_y} ({variance[comp_y-1]:.5f})')
        ax.legend(['Sunt blad', 'Gulrust'])
        plt.axvline(x=0, c='black', linestyle='dotted')
        plt.axhline(y=0, c='black', linestyle='dotted')
        plt.title('PCA: data i komponentrom\n'
                  '2. derivat')
        # for i in range(X_t.shape[0]):
        #     ax.annotate(i, (X_t[i, comp_x-1], X_t[i, comp_y-1]), alpha=0.5,
        #     fontsize=8)
        # fig.savefig(r'C:\Users\svein\Desktop\temp_images\Method1_PCA_c{0}_v_c{1}_2der.PNG'.form
        plt.close()

loadings = pca.components_.T
loadings = savgol_filter(loadings, 11, 3, axis=0)
```

```
plt.figure(figsize=(20, 10))
plt.plot(loadings[:, :])
plt.legend(['PC 1', 'PC 2', 'PC 3', 'PC 4', 'PC 5'])
plt.title('PCA loadings from five PCs\n \n'
          'WR corrected\n'
          'SG-filter(windowlength=11, polyorder=3)')
plt.ylim(-0.2, 0.2)
plt.xticks(ticks=range(0, 288, 30), labels=wavelength[0:-1:30])
plt.show()


#%% LDA ---------------------------------------------------------------------
lda = LinearDiscriminantAnalysis()
lda.fit(X, np.ravel(y))
X_t_lda = lda.transform(X)

# Components -------------------------
plt.plot(X_t_lda[0], 'o', c='r')
plt.plot(X_t_lda[82], 'o', c='b')
plt.plot(X_t_lda[1:82], 'o', c='r', label='_nolegend_')
plt.plot(X_t_lda[83:], 'o', c='b', label='_nolegend_')
plt.title('LDA on area pixels \n '
          '2. derivative')
plt.xlabel(f'Spectrum number in class')
plt.ylabel(f'LDA component')
plt.legend(['Rust', 'Healthy'])
plt.show()

# Loading ------------------------------
plt.figure(figsize=(20, 5))
plt.plot(lda.coef_[0].T)
# plt.ylim(-500, 500)
plt.title('LDA weight vectors \n 2. derivative')
# plt.legend(['Rust', 'Healthy'])
# plt.axvspan(140, 148, alpha=0.5)
plt.hlines(y=0, xmin=0, xmax=288)
plt.ylabel(f'LDA component')
plt.xlabel(f'Wavelength')
plt.xticks(ticks=range(0, 288, 30), labels=wavelength[0:-1:30])
plt.show()



#%% SVM classifier ----------------------------------------------------------
# # ------ Cross validation ------
cvs = cross_val_score(estimator=SVC(kernel='rbf', degree=2, C=8), X=X_t_pca,
                      y=y.ravel(), cv=5)
pred = cross_val_predict(estimator=SVC(kernel='rbf', degree=2, C=8), X=X_t_pca,
                         y=y.ravel(), cv=5)
confm = confusion_matrix(y_true=y, y_pred=pred)
f1 = f1_score(y_true=y, y_pred=pred, average='weighted')
pscore = precision_score(y_true=y, y_pred=pred, average='weighted')
recall = recall_score(y_true=y, y_pred=pred, average='weighted')
print(f' ------ SVC(kernel=rbf, degree=4, C=6) ------: \n'
      f'5-fold CV\n'
      f'Datatype: LDA transformed data (1 component)\n'
      f'Accuracy: {cvs.mean():.4f}, std {cvs.std():.4f} \n'
      f'Recall: {recall:.4f}, Precision: {pscore:.4f}, F1-score: {f1:.4f}\n')
print('Confusion matrix from 5-fold CV:')
print(confm)
# #
param_grid = {'C': [1, 2, 3, 5, 6,8,9, 10, 11,  12, 15, 18, 20, 30, ],
              'kernel': ['poly', 'rbf', 'linear'],
              'degree': [2, 4, 6, 7, 8, 10],
              # 'coef0': [0, 1, 4]
              }
```

```
gs = GridSearchCV(estimator=SVC(), param_grid=param_grid, scoring='accuracy')
gs.fit(X_t_pca, y)
print(gs.best_params_)


# ------- Prediction on images ----------
svc = SVC(kernel='poly', degree=4, C=6)
img = df_image_pred['image'].iloc[9]
mask = df_image_pred['mask'].iloc[9]
print(df_image_pred['leaf number'].iloc[9])
img_c = classifier_img_pred(X_emsc, y.ravel(), img, mask=mask, classifier=svc,
                            emsc_corr=True, emsc_degree=2, emsc_Xref=Xref_emsc)

fig, ax = plt.subplots(figsize=(6,16))
cax = ax.imshow(img_c, cmap=ListedColormap(colors), interpolation='None')
ax.set_title('Bildeklassifisering \n SVC trent på EMSC \n Blotch blad nr. 2 '
             '(230720)')
cbar = fig.colorbar(cax, ticks=[-1, 0, 1])
cbar.ax.set_yticklabels(['Bakgrunn', 'Gulrust', 'Sunt blad'], fontsize=30)  # horizontal colorbar
plt.axis('Off')
plt.show()

#%% PLSDA classifier ------------------------------------------------------------
# # ------ Cross validation ------
plsda = PLSDA_crossval(n_components=5)
acc, std, f1, precision, recall, confm = plsda.cross_val(X=X_t_pca, y_true=y,
                                                         cv=5)
print(f' ------ PLSDA(n_components=5) ------: \n'
        f'5-fold CV\n'
        f'Datatype: EMSC data\n'
      f'Accuracy: {acc:.4f}, std {std:.4f} \n'
      f'Recall: {recall:.4f}, Precision: {precision:.4f}, F1-score: {f1:.4f}\n')
print('Confusion matrix from 5-fold CV:')
print(confm)
#
plsda = PLSRegression(n_components=5)
img = df['image'].iloc[1]
mask = df['mask'].iloc[1]
print(df['leaf number'].iloc[1])
img_c = classifier_img_pred(X, y.ravel(), img, mask=mask, classifier=plsda)
plt.figure(figsize=(5,16))
plt.imshow(img_c, interpolation='None')
plt.title('PLSDA trained on area pixels \n no SNV \n pred on rust  #3')
plt.colorbar()
plt.show()


# Random Forest Classifier  -----------------------------------------------
# Cross validation ----
cvs = cross_val_score(estimator=RandomForestClassifier(n_estimators=300),
                      X=X_t_pca, y=y.ravel(), cv=5)
pred = cross_val_predict(estimator=RandomForestClassifier(n_estimators=300),
                         X=X_t_pca, y=y.ravel(), cv=5)
confm = confusion_matrix(y_true=y, y_pred=pred)
f1 = f1_score(y_true=y, y_pred=pred, average='weighted')
pscore = precision_score(y_true=y, y_pred=pred, average='weighted')
recall = recall_score(y_true=y, y_pred=pred, average='weighted')
print(f' ------ RandomForestClassifier() ------: \n'
      f'5-fold CV\n'
      f'Datatype: sensitive areas from emsc pca comp. 2, 4\n'
      f'0-3, 50-60, 79-81, 100-110, 160-173, 235-247 \n\n'

      f'Accuracy: {cvs.mean():.4f} and std {cvs.std():.4f} \n'
      f'Recall: {recall:.4f}, Precision: {pscore:.4f}, F1-score: {f1:.4f}\n')
print('Confusion matrix from 5-fold CV:')
print(confm)
```

```
param_grid = {'n_estimators': [10, 20, 30, 35, 40, 45, 50, 100, 300, 500, 700],
              # 'max_depth': [None],
              # 'min_samples_split': [2],
              # 'coef0': [0, 1, 4]
              }

gs = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid,
                  scoring='accuracy')
gs.fit(X_t_pca, y)
print(gs.best_params_)

# ------- Prediction on images ----------
rfc = RandomForestClassifier(n_estimators=50)
img = df_image_pred['image'].iloc[9]
mask = df_image_pred['mask'].iloc[9]
print(df_image_pred['leaf number'].iloc[9])
img_c = classifier_img_pred(X_t_pca, y.ravel(), img, mask=mask, classifier=rfc,
                            emsc_corr=False, emsc_degree=2,
                            emsc_Xref=Xref_emsc, pca=pca)
print('done')
#
fig, ax = plt.subplots(figsize=(7,16))
cax = ax.imshow(img_c, cmap=ListedColormap(colors), interpolation='None')
ax.set_title('Bildeklassifisering \n RFC trent på PCA trans HR-korrigert \n '
             'Blotch blad nr. 2 (230720)')
cbar = fig.colorbar(cax, ticks=[-1, 0, 1])
cbar.ax.set_yticklabels(['Bakgrunn', 'Gulrust', 'Sunt blad'], fontsize=30)  # horizontal colorbar
plt.axis('Off')
plt.show()


#==============================================================================
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 13 2021 13:26

@author: Svein Jakob Kristoffersen (svein.jakob.kristoffersen@nmbu.no)

Code to handle method 2 data containing 4 classes. Sensitive wavelengths will
be chosen and models trained and validated.
"""

import os
from leaf_processing_1_new import *
from emsc import emsc
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import StratifiedKFold, cross_val_score, \
    cross_val_predict, GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, \
    recall_score, precision_score
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SequentialFeatureSelector, RFE
import pandas as pd
from matplotlib.colors import ListedColormap
import random
random.seed(13)
np.random.seed(13)

plt.rc('xtick', labelsize=9)
plt.rc('ytick', labelsize=9)
plt.rc('font', size=15)
```

```
#%% Datasetup ------------------------------------------------------------------
# read data of leaves used for image prediction:
df_image_pred = pd.read_pickle(r'C:\Users\svein\Desktop\leaf_dataframe_other_leaves.csv')
# read data
df = pd.read_pickle(r'C:\Users\svein\Desktop\leaf_dataframe_noSNV.csv')
# df = df[df['class'].isin(['rust', 'healthy', 'blotch n', 'blotch'])]
colors = ['black', 'r', 'g', 'c', 'b']  # Colors for use in plotting

wavelength = get_wavelength(keep_floats=True)
print(wavelength)

# Setting up data in lists after class:
pixels_rust = reach_area_pixels(df, 'rust')
pixels_blotch_n = reach_area_pixels(df, 'blotch necrosis')
pixels_blotch_c = reach_area_pixels(df, 'blotch chlorosis')
pixels_blotch_c = undersample_list(pixels_blotch_c, new_size=100)
pixels_healthy = reach_area_pixels(df, 'healthy', is_healthy=True)
Xref_emsc = np.mean(pixels_healthy, axis=0)  # Reference spectrum for EMSC
pixels_healthy = undersample_list(pixels_healthy, new_size=100)

pixels_rust_emsc = emsc(X=pixels_rust, ref_spec=Xref_emsc, d=2)
pixels_healthy_emsc = emsc(X=pixels_healthy, ref_spec=Xref_emsc, d=2)
pixels_blotch_n_emsc = emsc(X=pixels_blotch_n, ref_spec=Xref_emsc, d=2)
pixels_blotch_c_emsc = emsc(X=pixels_blotch_c, ref_spec=Xref_emsc, d=2)

# Standard deviations used for plotting graphs.
std_p_rust, std_n_rust = give_deviation_graphs(pixels_rust)
std_p_healthy, std_n_healthy = give_deviation_graphs(pixels_healthy)
std_p_blotch_n, std_n_blotch_n = give_deviation_graphs(pixels_blotch_n)
std_p_blotch_c, std_n_blotch_c = give_deviation_graphs(pixels_blotch_c)

std_p_rust_emsc, std_n_rust_emsc = give_deviation_graphs(pixels_rust_emsc)
std_p_healthy_emsc, std_n_healthy_emsc = give_deviation_graphs(pixels_healthy_emsc)
std_p_blotch_n_emsc, std_n_blotch_n_emsc = give_deviation_graphs(pixels_blotch_n_emsc)
std_p_blotch_c_emsc, std_n_blotch_c_emsc = give_deviation_graphs(pixels_blotch_c_emsc)

# Merging classes into X and y matrices for use in classification:
X, y = make_x_y([pixels_rust, pixels_blotch_n,
                 pixels_blotch_c, pixels_healthy])
X_emsc, y = make_x_y([pixels_rust_emsc, pixels_blotch_n_emsc,
                      pixels_blotch_c_emsc, pixels_healthy_emsc])
pd.to_pickle(y, r'C:\Users\svein\Desktop\y.csv')

#%% Plotting spectrums ----------------------------------------------------------
# EMSC spectrums
plt.figure(figsize=(8, 6))
plt.plot(np.mean(pixels_healthy_emsc.T, axis=1), 'b')
plt.plot(np.mean(pixels_rust_emsc.T, axis=1), 'r')
plt.plot(np.mean(pixels_blotch_c_emsc.T, axis=1), 'c')
plt.plot(np.mean(pixels_blotch_n_emsc.T, axis=1), 'g')
plt.fill_between(x=range(288), y1=std_p_rust_emsc, y2=std_n_rust_emsc,
                 alpha=0.2, color='r')
plt.fill_between(x=range(288), y1=std_p_healthy_emsc, y2=std_n_healthy_emsc,
                 alpha=0.2, color='b')
plt.fill_between(x=range(288), y1=std_p_blotch_c_emsc, y2=std_n_blotch_c_emsc,
                 alpha=0.2, color='c')
plt.fill_between(x=range(288), y1=std_p_blotch_n_emsc, y2=std_n_blotch_n_emsc,
                 alpha=0.2, color='g')
plt.title('Klassespektrum gjennomsnitt og standardavvik \n'
          'EMSC prosessert')
plt.legend(['Gulrust', 'Sunt blad', 'Blotch Chlorosis', 'Blotch Necrosis'])
plt.xticks(ticks=range(0, 288, 50), labels=wavelength[0:-1:50])
plt.ylabel('Reflektanse')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
```

```
# plt.axvspan(xmin=75, xmax=110, alpha=0.2)
# plt.axvspan(xmin=160, xmax=195, alpha=0.2)
# plt.axvspan(xmin=260, xmax=288, alpha=0.2)
plt.show()

#%% PCA ------------------------------------------------------------------------
pca = PCA(n_components=200)
pca.fit(X_emsc)
exp_var_cumul = np.cumsum(pca.explained_variance_ratio_)
plt.plot(exp_var_cumul)
plt.title('PCA explined variance vs # components')
plt.xlabel('# components')
plt.ylabel('Explained variance')
plt.show()


pca = PCA(n_components=5)
pca.fit(X)

X_t_pca = pca.transform(X)
loadings = pca.components_.T
loadings = savgol_filter(loadings, 11, 3, axis=0)
#
variance = pca.explained_variance_ratio_
components = [1, 2, 3, 4, 5]
for comp_x in components:
    for comp_y in components[comp_x:]:
        fig, ax = plt.subplots()

        ax.scatter(X_t_pca[286:, comp_x-1], X_t_pca[286:, comp_y-1],
                   c='b', alpha=0.5)
        ax.scatter(X_t_pca[:82, comp_x-1], X_t_pca[:82, comp_y-1],
                   c='r', alpha=0.5)
        ax.scatter(X_t_pca[82:186, comp_x-1], X_t_pca[82:186, comp_y-1],
                   c='g', alpha=0.5)
        ax.scatter(X_t_pca[186:286, comp_x-1], X_t_pca[186:286, comp_y-1],
                   c='c', alpha=0.5)

        plt.xlabel(f'komponent {comp_x} ({variance[comp_x-1]:.5f})')
        plt.ylabel(f'komponent {comp_y} ({variance[comp_y-1]:.5f})')
        ax.legend(['Sunt blad', 'Gulrust', 'Blotch Necrosis',
                   'Blotch Chlorosis'], fontsize=11)
        plt.title('PCA: data i komponentrom \n '
                  'EMSC prosessert')
        plt.axvline(x=0, c='black', linestyle='dotted')
        plt.axhline(y=0, c='black', linestyle='dotted')
        # for i in range(X_t.shape[0]):
        #     ax.annotate(i, (X_t[i, comp_x-1], X_t[i, comp_y-1]), alpha=0.5,
        #     fontsize=8)
        # fig.savefig(r'C:\Users\svein\Desktop\temp_images\PCA_area pixels_c{0}_v_c{1}_emsc.PNG'.
        plt.close()

plt.figure(figsize=(15, 7))
plt.plot(loadings[:, :3])
plt.legend(['Komponent 1', 'Komponent 2', 'Komponent 3', 'Komponent 4',
            'Komponent 5'])
plt.title('PCA: komponenters vektlegging av bølgelengder\n'
          'EMSC\n'
          # 'Utvalgte bånd urelatert til vannabsorbsjon'
          )

# plt.vlines(74.5, ymax=0.2, ymin=-0.2, colors='black', linestyles='solid')
# plt.vlines(124.5, ymax=0.2, ymin=-0.2, colors='black', linestyles='solid')
plt.hlines(y=0,xmin=-20, xmax=300, colors='black', linestyles='dashed',
           alpha=0.4)
plt.axvspan(xmin=0, xmax=71, alpha=0.3, color='purple')
```

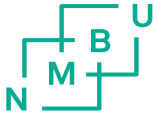
```
plt.axvspan(xmin=71, xmax=121, alpha=0.5, color='purple')
plt.axvspan(xmin=121, xmax=186, alpha=0.7, color='purple')
plt.ylim(-0.22, 0.18)
plt.xlim(0, 186)
plt.ylabel('Vektlegging')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.vlines(x=143, ymin=-0.22, ymax=0.18)
plt.show()

#%% LDA ------------------------------------------------------------------------
lda = LinearDiscriminantAnalysis()
lda.fit(X_emsc, np.ravel(y))
X_t_lda = lda.transform(X_emsc)

# Components -------------------------
variance = lda.explained_variance_ratio_
components = [1, 2, 3]
for comp_x in components:
    for comp_y in components[comp_x:]:
        fig, ax = plt.subplots()
        ax.scatter(X_t_lda[286:, comp_x-1], X_t_lda[286:, comp_y-1],
                   c='b', alpha=0.5)
        ax.scatter(X_t_lda[:82, comp_x-1], X_t_lda[:82, comp_y-1],
                   c='r', alpha=0.5)
        ax.scatter(X_t_lda[82:186, comp_x-1], X_t_lda[82:186, comp_y-1],
                   c='g', alpha=0.5)
        ax.scatter(X_t_lda[186:286, comp_x-1], X_t_lda[186:286, comp_y-1],
                   c='c', alpha=0.5)
        plt.xlabel(f'komponent {comp_x} ({variance[comp_x-1]:.5f})')
        plt.ylabel(f'komponent {comp_y} ({variance[comp_y-1]:.5f})')
        ax.legend(['Sunt blad', 'Gulrust', 'Blotch Necrosis',
                   'Blotch Chlorosis'], fontsize=11)
        plt.title('LDA: data i komponentrom \n '
                  'EMSC prosessert')
        plt.axvline(x=0, c='black', linestyle='dotted')
        plt.axhline(y=0, c='black', linestyle='dotted')
        # for i in range(X_t.shape[0]):
        #     ax.annotate(i, (X_t[i, comp_x-1], X_t[i, comp_y-1]), alpha=0.5,
        #     fontsize=8)
        # fig.savefig(r'C:\Users\svein\Desktop\temp_images\LDA_area_pixels_c{0}_v_c{1}_emsc.PNG'.
        plt.close()

# Loading ------------------------------
lda = LinearDiscriminantAnalysis()
lda.fit(X_emsc, np.ravel(y))
loadings = lda.scalings_
loadings = savgol_filter(loadings, 21, 2, axis=0)
plt.figure(figsize=(15, 7))
plt.plot(loadings)
plt.title('LDA vektleggingsvektorer (.scalings_) \n EMSC data, S-G filter')
plt.legend(['Komponent 1', 'Komponent 2', 'Komponent 3'])
plt.hlines(y=0, xmin=0, xmax=288)
plt.xticks(ticks=range(0, 288, 10), labels=wavelength[0:-1:10])
plt.ylabel('Vektlegging')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
# plt.ylim(-1500, 1500)
plt.xlim(0, 288)
plt.show()

#%% Choosing sensitive bands ---------------------------------------------------
# PCA
plt.figure(figsize=(20, 10))
plt.plot(loadings)
# plt.legend(['PC 2', 'PC 4'])
```

```
plt.title('PCA loadings from five PCs\n \n'
          'EMSC\n'
          'SG-filter(windowlength=11, polyorder=3)')
plt.ylim(-0.2, 0.2)
# plt.xticks(ticks=range(0, 288, 30), labels=wavelength[0:-1:30])
plt.grid('On')
# plt.vlines(x=0, ymax=0.2, ymin=-0.2, linestyles='dashed')
plt.axvspan(xmin=75, xmax=83, alpha=0.2)
plt.axvspan(xmin=100, xmax=120, alpha=0.2)
plt.axvspan(xmin=75, xmax=83, alpha=0.2)
plt.axvspan(xmin=162, xmax=173, alpha=0.2)
plt.show()

# RFC
rfc = RandomForestClassifier(n_estimators=300)
rfc.fit(X_emsc, y)
plt.figure(figsize=(12, 7))
plt.plot(np.multiply(np.mean(pixels_healthy_emsc.T, axis=1), 10**-1), 'b',
         linestyle='dotted')
plt.plot(rfc.feature_importances_, 'red')
plt.title('Random Forest Classifier: '
          'Vektlegging av egenskaper (Feature Importance)\n'
          'EMSC-prosessert\n'
          )
# plt.xticks(ticks=range(0, 288, 30), labels=wavelength[0:-1:30])
plt.legend(['Sunne piksler snittspektrum (skalert)', 'RFC vektlegging'])
plt.ylabel('Vektlegging')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.axvspan(xmin=0, xmax=71, alpha=0.3, color='purple')
plt.axvspan(xmin=110, xmax=160, alpha=0.5, color='purple')
plt.axvspan(xmin=195, xmax=260, alpha=0.7, color='purple')
plt.show()

sensitive_bands_emsc_not_water = np.concatenate([[i for i in range(0, 71)],
                                                 [i for i in range(110, 160)],
                                                 [i for i in range(195, 260)],
                                                 ])
X_sens_emsc_not_water = X_emsc[:, sensitive_bands_emsc_not_water]
#
rfc = RandomForestClassifier(n_estimators=300)
rfc.fit(X_sens_emsc_not_water, y)

plt.figure(figsize=(12, 7))
plt.plot(rfc.feature_importances_, 'red')
plt.title('Random Forest Classifier: Vektlegging av egenskaper '
          '(Feature Importance)\n'
          'EMSC-prosessert\n'
          'Utvalgte bånd urelatert til vannabsorbsjon'
          )
# plt.xticks(ticks=range(0, 186, 10), labels=wavelength[0:-1:10])
plt.legend(['RFC vektlegging'])
plt.ylabel('Vektlegging')
plt.xlabel('Bølgelengde [nm]')
plt.grid('On')
plt.axvspan(xmin=0, xmax=71, alpha=0.3, color='purple')
plt.axvspan(xmin=71, xmax=121, alpha=0.5, color='purple')
plt.axvspan(xmin=121, xmax=186, alpha=0.7, color='purple')
plt.vlines(x=104, ymin=0, ymax=0.04)
plt.xticks(ticks=range(0, 186, 10),
           labels=list(set(sensitive_bands_emsc_not_water))[::10])
plt.show()

sens_bands_pca = [0,60,88,106,170,179,193,245]
sens_bands_pca_nowater = [0,61,66,134,140,198,216,233,259]
sens_bands_rfc = [9,68,72,83,86,160,165,168,179,186]
```

```
sens_bands_rfc_nowater = [9,56,69,123,136,140,142,159]
water_abs_limits = [71, 110, 160, 195, 260]

# From bands to wavelengths:
print(wavelength[sens_bands_pca])
print(wavelength[sens_bands_pca_nowater])
print(wavelength[sens_bands_rfc])
print(wavelength[sens_bands_rfc_nowater])
print(wavelength[water_abs_limits])

#%% Classification ------------------------------------------------------------
# SVM classifier
#                         ------ Cross validation ------
cvs = cross_val_score(estimator=SVC(kernel='poly', degree=6, C=15), X=X_emsc,
                      y=y.ravel(), cv=5)
pred = cross_val_predict(estimator=SVC(kernel='poly', degree=6, C=15),
                         X=X_emsc, y=y.ravel(), cv=5)
confm = confusion_matrix(y_true=y, y_pred=pred)
f1 = f1_score(y_true=y, y_pred=pred, average='weighted')
pscore = precision_score(y_true=y, y_pred=pred, average='weighted')
recall = recall_score(y_true=y, y_pred=pred, average='weighted')
print(f' ------ SVC(kernel=poly, degree=2, C=12) ------: \n'
      f'5-fold CV\n'
      f'Datatype: EMSC, Waterbands excluded\n'
      f'0-75, 110-165, 195-260 \n\n'
      f'Accuracy: {cvs.mean():.4f}, std {cvs.std():.4f} \n'
      f'Recall: {recall:.4f}, Precision: {pscore:.4f}, F1-score: {f1:.4f}\n')
print('Confusion matrix from 5-fold CV:')
print(confm)
# # #
param_grid = {'C': [1,2,3,5,6,8,9,10,11,12,15,18],
              'kernel': ['poly'],
              'degree': [2, 4, 6],
              # 'coef0': [0, 1, 4]
              }
gs = GridSearchCV(estimator=SVC(), param_grid=param_grid, scoring='accuracy')
gs.fit(X_emsc, y)
print(gs.best_params_)

# ------- Prediction on images ----------
svc = SVC(kernel='poly', degree=8, C=1)
img = df_image_pred['image'].iloc[6]
mask = df_image_pred['mask'].iloc[6]
print(df_image_pred['leaf number'].iloc[6])
img_c = classifier_img_pred(X_emsc, y.ravel(), img, mask=mask, classifier=svc,
                            n_classes=4,
                            emsc_corr=True, emsc_degree=2, emsc_Xref=Xref_emsc)

fig, ax = plt.subplots(figsize=(7,16))
cax = ax.imshow(img_c, cmap=ListedColormap(colors), interpolation='None')
ax.set_title('Bildeklassifisering \n SVC trent på EMSC-prosessert data \n '
             'Gulrust blad nr. 2 (230720)')
cbar = fig.colorbar(cax, ticks=[-1, 0, 1, 2, 3])
cbar.ax.set_yticklabels(['Bakgrunn', 'Gulrust', 'Blotch\nnecrosis',
                         'Blotch\nchlorosis', 'Sunt blad'],
                        fontsize=30)  # horizontal colorbar
plt.axis('Off')
plt.show()

plot_hypercube(cube=img, mask=mask, bands=[160], Xref_emsc=Xref_emsc,
               emsc_degree=2, clim=(0.30, 0.32))



#%% PLSDA classifier
# # # ------ Cross validation
```

```
plsda = PLSDA_crossval(n_components=4, n_classes=4)
acc, std, f1, precision, recall, confm = plsda.cross_val(X=X_emsc, y_true=y,
                                                         cv=5)
print(f' ------ PLSDA(n_components=10) ------: \n'
      f'5-fold CV\n'
      f'Datatype: W.R. corr data\n'
      f'Accuracy: {acc:.4f}, std {std:.4f} \n'
      f'Recall: {recall:.4f}, Precision: {precision:.4f},'
      f' F1-score: {f1:.4f}\n')
print('Confusion matrix from 5-fold CV:')
print(confm)

plsda = PLSRegression(n_components=5)
img = df['image'].iloc[1]
mask = df['mask'].iloc[1]
print(df['leaf number'].iloc[1])
img_c = classifier_img_pred(X, y.ravel(), img, mask=mask, classifier=plsda)
plt.figure(figsize=(5,16))
plt.imshow(img_c, interpolation='None')
plt.title('PLSDA trained on area pixels \n no SNV \n pred on rust  #3')
plt.colorbar()
plt.show()

# Testing RFC ------------------------------------------------------------------
# Cross validation ----
cvs = cross_val_score(estimator=RandomForestClassifier(n_estimators=500),
                      X=X_emsc[:, sens_bands_rfc_nowater], y=y.ravel(), cv=5)
pred = cross_val_predict(estimator=RandomForestClassifier(n_estimators=500),
                         X=X_emsc[:, sens_bands_rfc_nowater], y=y.ravel(),
                         cv=5)
confm = confusion_matrix(y_true=y, y_pred=pred)
f1 = f1_score(y_true=y, y_pred=pred, average='weighted')
pscore = precision_score(y_true=y, y_pred=pred, average='weighted')
recall = recall_score(y_true=y, y_pred=pred, average='weighted')
print(f' ------ RandomForestClassifier() ------: \n'
      f'5-fold CV\n'
      f'Datatype: sensitive areas from emsc pca comp. 2, 4\n'
      f'0-3, 50-60, 79-81, 100-110, 160-173, 235-247 \n\n'

      f'Accuracy: {cvs.mean():.4f} and std {cvs.std():.4f} \n'
      f'Recall: {recall:.4f}, Precision: {pscore:.4f}, F1-score: {f1:.4f}\n')
print('Confusion matrix from 5-fold CV:')
print(confm)

param_grid = {'n_estimators': [10, 20, 30, 35, 40, 45, 50, 300],
              'max_depth': [None],
              'min_samples_split': [2],
              # 'coef0': [0, 1, 4]
              }

gs = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid,
                  scoring='accuracy')
gs.fit(X_emsc[:, sens_bands_rfc], y)
print(gs.best_params_)

# ------- Prediction on images ----------
rfc = RandomForestClassifier(n_estimators=50)
img = df_image_pred['image'].iloc[9]
mask = df_image_pred['mask'].iloc[9]
print(df_image_pred['leaf number'].iloc[9])
img_c = classifier_img_pred(X_emsc[:, sens_bands_rfc], y.ravel(), img,
                            mask=mask, classifier=rfc, n_classes=4,
                            emsc_corr=True, emsc_degree=2, emsc_Xref=Xref_emsc,
                            sensitive_bands=sens_bands_rfc)

fig, ax = plt.subplots(figsize=(7,16))
```

```
cax = ax.imshow(img_c, cmap=ListedColormap(colors), interpolation='None')
ax.set_title('Bildeklassifisering \n RFC trent på sensitive bølgelengder \n '
             'Gulrust blad nr. 2 (230720)')
cbar = fig.colorbar(cax, ticks=[-1, 0, 1, 2, 3])
cbar.ax.set_yticklabels(['Bakgrunn', 'Gulrust', 'Blotch necrosis',
                         'Blotch chlorosis', 'Sunt blad'],
                        fontsize=25)  # horizontal colorbar
plt.axis('Off')
plt.show()
```



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

Postboks 5003
NO-1432 Ås
Norway