Norges miljø- og biovitenskapelige universitet

Masteroppgave 2018 30 stp
Fakultet for realfag og teknologi

Utvikling av analyseprogram for identifikasjon av pulvermateriale basert på bildetekstur

Development of Software for Identification of Powder Samples based on Image Texture

Anja Katarina Smit
Linn Eirin Sogn
Miljøfysikk og fornybar energi

Forord

Denne masteroppgaven er skrevet ved Fakultet for realfag og teknologi ved Norges miljø- og biovitenskapelige universitet. Oppgaven har et omfang på 30 studiepoeng og markerer avslutningen på vårt studie i Miljøfysikk og fornybar energi.

Først og fremst rettes en stor og varm takk til vår hovedveiler Cecilia Marie Futsæther for eksepsjonell veiledning, gode råd, konstruktive diskusjoner, et godt samarbeid og for motivasjon gjennom hele prosessen. Videre vil vi takke våre biveiledere Knut Kvaal og Oliver Tomic som har hjulpet oss på veien. Takk til Knut for god hjelp med AMT og PCA og for å ha delt generøst av sine erfaringer fra dataanalyse med oss. Takk til Oliver Tomic som har stilt opp med uvurderlig programmeringshjelp og gode råd om maskinlæring.

We also want to thank laboratory technician and scientific researcher Lorenzo Fongaro, European Commission, Joint Research Centre (JRC), Nuclear Safety and Security, Karlsruhe, Germany, for the data consisting of SEM images and for making this project possible.

Vi vil også takke venner og familie for god støtte under arbeidet med oppgaven, og en spesiell takk til vennene våre på Ås som har bidratt til at vi har hatt en flott studietid.

Til slutt vil vi takke hverandre for et godt samarbeid helt fram til mållinja.

Ås, 14. mai 2018

_____ _____

Anja Katarina Smit Linn Eirin Sogn

Sammendrag

Kjernefysisk materiale som kommer på avveie skaper uro og bekymring grunnet assosiasjoner som gjerne kobles til anvendelse av slikt materiale, for eksempel knyttet til bruk i kjernefysiske våpen. Beslag av kjernefysisk materiale gir opphav til en rekke spørsmål som må besvares; hvor kommer materialet fra? Hva var det tiltenkte bruksområdet? Hvem eier materialet? Fenomenet som omfatter ulovlig smugling av kjernefysisk materiale og de dertil tilhørende spørsmål, har gitt opphav til et nytt felt kalt *Nuclear forensics*. *Nuclear forensics* henspiller på etterforskningen av slikt materiale, der hovedformålet er å identifisere materialets herkomst.

Hovedformålet med denne oppgaven var å utvikle et analyseprogram for bruk til klassifisering av bilder av uranprøver tatt med skanningelektronmikroskop (SEM). Datasettet med SEM-bilder er tatt av uranprøver fra seks forskjellige prosesseringssteder, og gir opphav til seks klasser. Prøvene fra de seks klassene har varierende tekstur og kjemisk sammensetning, hvilket vil gjenspeiles i bildene, og hypotesen er at prøvene fra de seks klassene kan skilles fra hverandre ved hjelp av bildeanalyse.

Klassifiseringen av prøvene ble foretatt på bakgrunn av bildeanalyse og maskinlæring. Gjennom bildeanalysen ble det trukket ut egenskaper fra bildene som beskriver fordelingen av intensitetsverdier (førsteordens statistikk) og den romlige fordelingen av intensitetsverdier (tekstur) i bildene. Åtte metoder for kvantifisering av bildetekstur ble vurdert. Disse egenskapene dannet grunnlaget for klassifiseringen.

Programmet leser inn SEM-bilder, tildeler dem en klasse, beregner bildeegenskaper, velger ut de mest relevante egenskapene med tre metoder og bygger opp klassifiseringsmodeller. Åtte klassifiseringsalgoritmer ble implementert og ble validert gjennom nøstet kryssvalidering. Modellenes endelige ytelse ble validert gjennom klassifisering av bilder i et uavhengig testsett.

Bildeegenskaper basert på Gray Level Run-Length Matrix (GLRLM) og Local Binary Patterns (LBP) viste seg å fange opp teksturforskjeller som skilte klassene, i motsetning til egenskaper basert på førsteordens statistikk. Klassifiseringsalgoritmene Random Forest og Support Vector Machines (SVM), som har ikke-lineære beslutningsgrenser, ga høye og stabile nøyaktigheter for klassifiseringen. Derimot ga klassifiseringsalgoritmene AdaBoost og K-nærmeste nabo lavere nøyaktighet. Modellen med seks utvalgte egenskaper fra LBP og med SVM som klassifikator ga en klassifiseringsnøyaktighet lik 0,92 på testsettet. Modellen med ni utvalgte egenskaper fra GLRLM, Gray Level Size Zone Matrix og LBP, og med SVM som klassifikator ga en klassifiseringsnøyaktighet lik 1,00.

Analyseprogrammet som er utviklet kan dermed brukes til å klassifisere og identifisere pulvermateriale i form av uranprøver basert på bildetekstur og maskinlæring.

Abstract

Nuclear material that goes astray creates uncertainty and worry, due to associations often made to the potential uses of such material, for example in nuclear weapons. Seizure of nuclear material prompts a series of questions that need answering: Where does it come from? What was the intended use? Who owns the material? The phenomenon of illegal smuggling of nuclear material and the questions surrounding it has given rise to a new field of research, *Nuclear forensics*. *Nuclear forensics* alludes to the investigation of such material, where the main task is to identify the material's origin.

The primary goal of this thesis was to develop an analysis program for use in classifying images of uranium samples taken with a scanning electron microscope (SEM). The dataset of SEM-images is made up of uranium ore concentrate samples from six different process sites, that form the basis for six classes. The samples from the six classes have different textures and chemical composition, which will reflect in the image. The hypothesis is that the samples from the six classes can be distinguished from one another using image analysis.

The classification of the samples was done based on image analysis and machine learning. Image features were extracted that describe the distribution of intensity values (first order statistics) and the spatial distribution of intensity values (texture) of the images. Eight methods of quantifying image texture were examined. These features formed the basis for the classification.

The program reads SEM-images, assigns them a class, calculates image features, uses three methods to select the most relevant features, and builds classification models. Eight classification algorithms were implemented and validated through nested cross-validation. The model's final performance was validated through classification of images in an independent test set.

Image features based on the Gray Level Run-Length Matrix (GLRLM) and Local Binary Patterns (LBP) captured texture differences that separated the classes, as opposed to properties based on first order statistics. The classification algorithms Random Forest and Support Vector Machines (SVM), that have non-linear decision boundaries, gave high and stable classification accuracies. On the other hand, the classification algorithms AdaBoost and K-nearest neighbour gave lower accuracy. The model with six chosen features from LPB and with SVM as a classifier, resulted in a classification accuracy of 0.92 on the test set. The model with nine chosen features from GLRLM, Gray Level Size Zone Matrix and LBP, with SVM as a classifier gave a classification accuracy of 1.00.

The developed analysis program can therefore be used to classify and identify powder material in the form of uranium samples based on image texture and machine learning.

Innhold

1 Innledning

Beslag av kjernefysisk materiale i de senere år vitner om at slikt materiale har blitt en smuglevare og at det er et marked for slik handel i verden [1]. Kjernefysisk materiale som har kommet på avveie skaper bekymring og frykt grunnet assosiasjoner som gjerne kobles til anvendelse av slikt materiale; for eksempel knyttet til bruk i kjernefysiske våpen. På denne måten kan sikkerheten knyttet til kjernefysisk materiale, eller den *nukleære sikkerheten*, synes å være svekket.

Et beslag av kjernefysisk materiale gir opphav til en rekke spørsmål som må bli besvart: Hvor kommer materialet fra? Hva var det tiltenkte bruksområdet til materialet? Hvem eier materialet? Hvilken smuglerrute har materialet fulgt før det ble beslaglagt?

Som et resultat av fenomenet knyttet til ulovlig smugling av kjernefysisk materiale og de dertil tilhørende spørsmål, har det blomstret opp et nytt felt kalt *"Nuclear forensics"*. *Nuclear forensics* henspeiler altså på etterforskningen av kjernefysisk materiale som har kommet på avveie. Hovedformålet med en slik etterforskning er å identifisere materialer det har blir gjort beslag på for å finne ut hvor de kommer fra [1]. Prosessering av uran foregår på litt forskjellig måte på ulike produksjonssteder i verden. Dessuten fraktes uranet gjerne i store mengder og over lange distanser, hvilket kan medføre variasjoner, men også skader på materialet. Variasjonen i de kjernefysiske stoffene etter forskjellige prosseseringsteknikker gjenspeiles i sammensetningen av forskjellige grunnstoff og isotoper, samt i det mikroskopiske og makroskopiske utseende til materialet [1]. Disse faktorene kan bidra til at en prøve med kjernefysisk materiale får et "nukleært fingeravtrykk" slik at den kan kobles til det stedet den ble prosessert.

Etterforskningen av prøvene kan ha ulike tilnærminger og foregå på bakgrunn av ulike analytiske teknikker. Det kan deles inn i to hovedkategorier, der den første er kjemiske og fysiske analytiske metoder, og den andre er strålingsanalytiske metoder [1]. Varga et al. [2] benytter Fouriertransform-infrarød (FTIR)-spektroskopi for karakterisering og klassifisering av Uranium Ore Concentrates (UOC). Formålet med undersøkelsen var å identifisere kjemiske bestanddeler og urenheter i uranprøver som kunne gi informasjon om produksjonsmetoden. Resultatet av studien var blant annet at FTIR-spektroskopi er et lettvint og godt verktøy som kan brukes til å identifisere bestanddeler av UOC. Plaue et al. [3] undersøker nær-infrarød reflektans (NIR)-spektroskopi som metode for å finne "signaturen" knyttet til prosesseringshistorikken

for uranoksider. Undersøkelsen viste at et NIR-spekter av uranutfellinger kunne brukes som en rask analysemetode i *Nuclear forensics*-sammenheng. NIR-spektrene gjorde det mulig å skille prøvenes opprinnelsessted ved visuell undersøkelse av spektrene. Olsen et al. [4] benyttet skanningelektronmikroskopbilder av Uranium Ore Concentrates (UOC) og beregnet størrelsen og formen på partiklene i bildene. Undersøkelsen gikk ut på å syntetisere uranprøvene med forskjellige kalsineringstemperaturer (henholdsvis på 600, 650, 700, 750 og $800°$C), hvorpå hver prøve ble karakterisert med pulver-røntgen-diffraksjon og skanningelektronmikroskopi. Undersøkelsen viste en statistisk forskjell på form og størrelse på partiklene ved de forskjellige temperaturene og at dette kan brukes til å identifisere historikken til en prosessert uranprøve.

Keegan et al. [5] benytter både kjemiske og fysiske metoder. I studien ble det sett på mikrostrukturen til et ukjent radioaktivt pulver som ble funnet under en politirazzia i 2009 i Australia. Pulveret ble sammenlignet med prøver med kjent opphav, og konklusjonen var at prøven mest sannsynlig kom fra Mary Kathleen og at forskjellene i mikrostrukturen til prøvene var et resultat av forskjellige prosesseringsmetoder. Fongaro et al. [6] benyttet bildetekstуranalyse basert på Angle Measure Technique (AMT) [7, 8] som metode for å identifisere uranprøver. AMT ble i kombinasjon med multivariat analyse brukt til å oppnå en ny ”signatur” på en prøve basert på teksturkarakteristikker, i form av et AMT-spekter. Med overvåket klassifikasjon (PLS-DA) ble det oppnådd resultater som vitner om at bildeanalyse er en god metode for å identifisere uranprøver.

Denne oppgaven har som formål å utvikle en metode som kan brukes til rask identifikasjon av Uranium Ore Concentrates (UOC), eller *”Yellowcakes”*, ved å bruke bildeanalyse og maskinlæring. Arbeidet bygger på metodene presentert av Fongaro et al. [6], og det samme datasettet vil bli benyttet i analysene. I denne oppgaven vil ytterligere metoder for å trekke ut bildeegenskaper bli undersøkt.

Bildeegenskaper henspeiler på fordelingen av intensitetsverdier, samt mønstre i et bilde, og angir et mål på den romlige strukturen i bildet, eller *strukturen* i bildet [9–11]. Teksturelle bildeegenskaper kan altså brukes til å identifisere og klassifisere ulike bildemotiver [6, 9, 12–14]. I denne oppgaven blir bildeanalyse anvendt på elektronmikroskopbilder av uranprøver. Det vil bli benyttet flere klassifiseringsalgoritmer for klassifisering av uranprøvene, og klassifikatorene vil bli anvendt i oppbygningen av en maskinlæringsmodell.

Forhåpentligvis kan metodene som utvikles i denne oppgavn bli et nyttig verktøy som kan brukes for å identifisere ukjente prøver av uran. For eksempel kunne en innretning med programvaren installert bli brukt på en flyplass, et sted der smuglevarer kan oppdages, for rask og effektiv prediksjon av opprinnelsessted for en beslaglagt uranprøve.

Oppgaven består av syv hovedkapitler. Kapittel 2, *Teori*, inneholder teori med relevans for oppgaven. Kapittelet tar blant annet for seg prosessering av uran, skanningelektronmikroskopi,

bildeanalyse og forskjellig metoder for klassifikasjon. I kapittel 3 presenteres datasettet, samt hvilke former for programvare som er benyttet i oppgaven. I kapittel 4 fremlegges metoden for oppbyggingen av analyseprogrammet, beregning av bildeegenskaper og oppbyggingen av klassifiseringsmodellen. I kapittel 5 presenteres resultatene av analysen og klassifiseringen. I kapittel 6 diskuteres oppgavens resultater, samt metodene som er benyttet. I kapittelet legges det også frem forslag til videre arbeid. Til slutt, i kapittel 7, konkluderes oppgaven på bakgrunn av problemstillingen til oppgaven og resultatene.

2 Teori

2.1 Prosessering av uran

Uran er et metallisk og radioaktivt grunnstoff. Det finnes naturlig i nær hele jordskorpen [15], og utgjør naturlig omlag 0,0004 % av den [16]. Uran kan spores opp i de fleste levende systemer i naturen; for eksempel i jord, elver, sedimenter, stein, og til og med i menneskekroppen [15]. Til tross for at uran er relativt hyppig forekommende, for eksempel 1000 ganger mer utbredt enn gull [16] og like utbredt som tinn [17], opptrer uranet ofte spredt i små spormengder [17]. Dersom konsentrasjonen av uran er høy nok, kategoriseres uranet som uranmalm, og det kan utvinnes fra berggrunnen [17]. Urankonsentrasjonen i malmen kan være opp mot 20 % [18]. Der uran forekommer i naturen, forekommer også andre grunnstoffer som stammer fra radioaktiv henfall av uran, der de hyppigst forekommende er radium-226 og radon-222 [19]. Dette medfører at måling av radioaktivitet *kan* kobles til tilstedeværelse av uran.

Gjennom gruvedrift blir uranmalm hentet ut av jordskorpen de stedene i verden der forekomsten av uran er tilstrekkelig stor. Formålet med gruvedriften og det primære bruksområdet til uran er som brensel i kjernekraftverk for strømproduksjon. Uran gjennomgår en rekke prosesseringssteg, definert ved begrepet ”front-end”, som henspeiler på forberedelse av uran for bruk i en reaktor i et kjernekraftverk [18]. Prosessen inkluderer stegene knyttet til gruvedrift, omdannelse, anrikning og fabrikasjon [18].

Det finnes en rekke forskjellig former for gruvedrift for utvinning av uranmalm fra berggrunnen. Hvilken metode som benyttes avhenger i høy grad av lokaliteten til forekomsten [17]. De konvensjonelle formene for gruvedrift kalles ”open pit” og ”underground” [17, 20]. Videre er ”in situ leaching” og ”heap leaching” eksempler på metoder som har blitt utviklet i senere år [18]. Ved tradisjonelle metoder blir uran knust, malt og blandet med vann for å danne en masse bestående av fine uranpartikler oppslemmet i vann. Gruvedrift i form av ”in situ leaching” innebærer en lavere påvirkning på berggrunnen. En kjemisk løsning blir sprøytet inn i grunnen der uranmalmen er, trekker ut uranet, hvorpå uranløsningen blir pumpet opp til overflaten [18]. Felles for metodene er at de resulterer i en masse bestående av uranpartikler oppløst i en væske. Videre blir uranet separert ut, renset og tørket, og tilslutt konsentrert ved ioneutveksling

og/eller løsemiddelkretser [18]. Resultatet er et uranoksid-konsentrat, omtalt som Uranium Ore Concentrates (UOC), ofte i form av forbindelsen U_3O_8. Uranet, nå i pulverform, fraktes så videre for ytterligere prosessering. UOC blir ansett som "low level radioactive material", hvilket betyr at det emitterer kun en liten mengde stråling, og er derfor ikke forbundet med stor helsefare [15]. Fargen på uranpulveret gjenspeiler den kjemiske sammensetningen - det kan være svart, men er oftest gult, slik at UOC ofte går under betegnelsen "yellow cakes".

I videre prosessering av UOC blir gjerne store mengder behandlet og transportert, hvilket fører til at pulveret kan få forskjellig utseende og struktur basert på hvor prosesseringen foregår. Dette gir opphav til at UOC kan kobles til det landet og/eller det stedet de ble prosessert [6]. Dette er kjernen i *Nuclear forensics* og bakgrunnen for analysene i denne oppgaven.

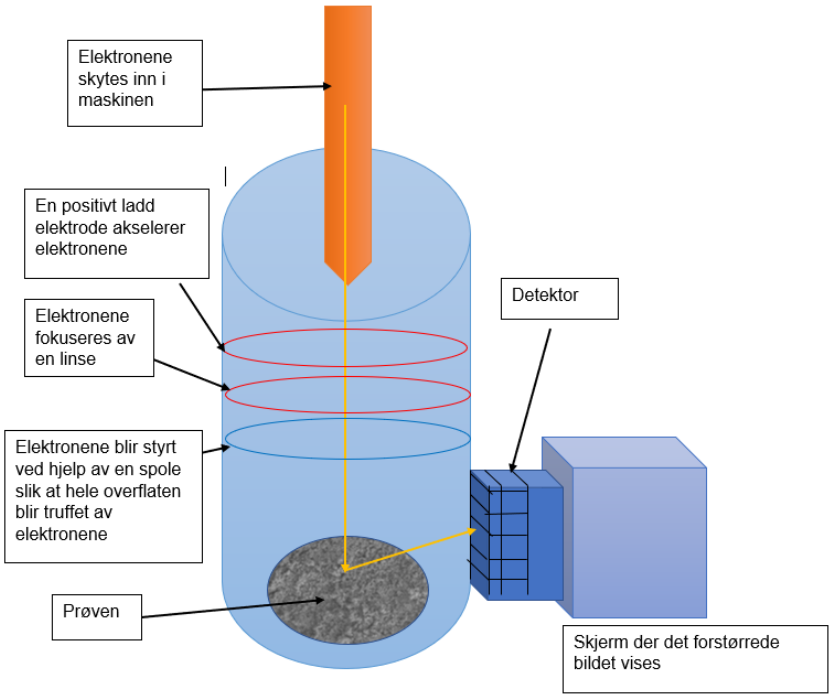
2.2 Skanningelektronmikroskopi

Dersom en prøve skal undersøkes på mikroskopisk nivå, er ordinære mikroskop som benytter lys ikke tilstrekkelig gode. Et bedre alternativ er å benytte et skanningelektronmikroskop (SEM). SEM genererer høyoppløselige bilder av overflaten til en prøve, som gir informasjon om krystallstruktur, kjemisk sammensetning og tekstur på et mikroskopisk nivå [21]. SEM ble første gang demonstrert av Knoll i 1935, men ble kommersialisert ved Universitetet i Cambridge først på 1960-tallet [22].

Det er knyttet enkelte begrensninger til hva som kan avbildes med SEM. Prøven kan ikke være større enn 10x4 cm, må være et fast materiale og de må være stabile i vakuum, da kammeret der prøven ligger holder et trykk på mindre enn $10^4 Pa$ [21, 22]. Prøvene blir ofte dekket med et tynt lag elektrisk ledende materiale, for eksempel gull slik som i Fongaro et al. [6]. Hvilket materiale som blir påført avhenger av hvilke strukturer fra prøven som skal fremheves i bildet [21].

De viktigste komponentene i et skanningelektronmikroskop er en elektronkilde, linser, en skanningspole, et kammer for prøven og en detektor [23]. Figur 2.1 viser en skjematisk fremstilling av komponentene i et skanningelektronmikroskop. Grunnprinsippet i et elektronmikroskop er at elektroner med høy energi samles i en fokusert stråle som skytes mot prøven som skal undersøkes. Elektronene som skytes mot prøven blir spredt og fanget opp av en detektor som digitaliserer signalet, slik at det kan vises som et todimensjonalt bilde [21].

Den første parameteren som defineres i et elektronmikroskop er energien til elektronene i elektronstrålen, stråleenergien E_0. Denne måles i elektronvolt, og varierer gjerne fra $E_0 = 0,1keV$ til $30keV$ [22]. Energien til elektronene kan endre seg flere ganger fra de forlater elektronkilden til de treffer prøven. Ladningsenergien E_l defineres som energien elektronene har idet de treffer prøven. Strålestrømmen I_b defineres som antallet elektroner som treffer overflaten



Figur 2.1: Oversiktsskisse av et skanningelektronmikroskop. Tilpasset fra [24].

av prøven per sekund. En typisk verdi for denne er omlag $1nA$, tilsvarende $6,25\text{x}10^9$ elektroner per sekund [22]. Denne strømmen kan endres ved å justere diameteren på strålen.

Etter at elektronene har forlatt elektronkilden blir de akselerert til høy energi ved at det blir satt opp en elektrostatisk potensialforskjell (målt i volt (V)). De negativt ladde elektronene frastøtes overflater med negativt elektrisk potensial og tiltrekkes overflater med positivt potensial [22].

Etter at strålen er akselerert går den gjennom magnetiske og/eller elektrostatiske linser og elektrostatiske spoler som fokuserer og reduserer diameteren på strålen. Prøven blir skannet i x- og y-retning ved at elektronstrålen flyttes og elektronene treffer overflaten av prøven og trenger ned i en dybde på omlag $1\mu m$, og elektronene produserer et signal som transformeres til et bilde [25]. For å fange opp dette signalet brukes elektrondetektorer. Idet elektronstrålene treffer overflaten av prøven blir elektronene spredt fra forskjellige dybder og noe vil forlate overflaten. Elektronene som blir spredt fra toppen av overflaten til prøven støter sammen *uelastisk*, og omtales som *sekundære elektroner*. Elektronene som trenger dypere ned i prøven, støter sammen *elastisk* og de som blir spredt tilbake omtales som *tilbakespredte elektroner* [25]. De sekundære elektronene har lav energi ($<$50eV), mens elektronene som spres tilbake har høyere energi ($>$50eV). De sekundære elektronene bidrar til å gi informasjon om teksturer i prøven [21].

Forstørrelsen av prøvene som blir avbildet kan beskrives ved formelen:

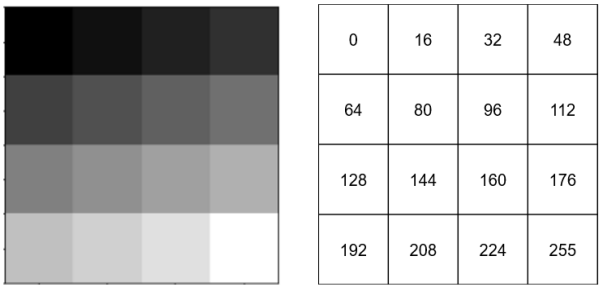
$$M = \frac{L}{l}, \tag{2.1}$$

der M er forstørrelsen, L er størrelsen på skjermen der bildet vises og l er størrelsen på prøven som er skannet med elektronstrålen [25]. For de fleste skanningmikroskopene vil L være en konstant størrelse, slik at det er l som må endres for å endre forstørrelsen. Med et SEM-mikroskop kan prøven forstørres opp til $10,0000\times$ [25].

2.3 Bildeanalyse og tekstur

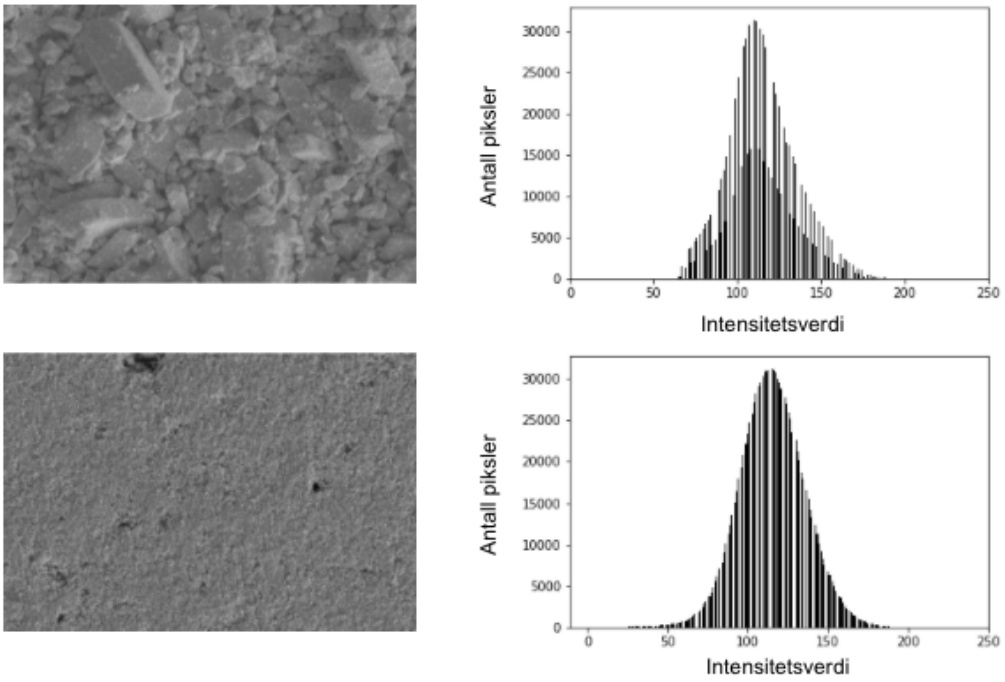
Bildeanalyse henspeiler på analyse av bilder med den hensikt å trekke ut meningsfull informasjon fra bildene [26]. Ved å analysere bilder, kan det identifiseres mønstre eller karakteristiske strukturer som kan beskrive det som er avbildet. Bildeanalyse har mange bruksområder; for eksempel i næringsmiddelindustrien, der bildeanalyse kan brukes til å sortere matvarer basert på kvaliteten på matvaren [27, 28], eller i pantesystemer [27]. Et annet bruksområde for bildeanalyse er i landbruksforskning for å kvantifisere skade på for eksempel planteblader, slik som i Kruse et al. [29].

Når et bilde, bestående av farger og objekter, blir digitalisert, blir det transformert til en matrise, bestående av rader og kolonner, med tallverdier. Hvert element i matrisen representerer en piksel med en tilhørende tallverdi. Tallverdiene i matrisen representerer fargene og intensitetene til bildet. Et RGB-fargebilde består av tre kanaler, én for hver av fargene rød, grønn og blå, slik at hver piksel har tre tallverdier. I et gråtonebilde består hver piksel av kun én verdi. For eksempel med et 8-bits gråtonebilde, vil hver piksel kunne ha 2^8 (256) forskjellige intensitetsverdier. Hver piksel blir tildelt en intensitetsverdi mellom 0 og 255. Verdiene blir lagret som en todimensjonal matrise [26]. Figur 2.2 viser et 4x4 piksler stort eksempelbilde med gråtoner (venstre) og tilhørende tallverdier til pikslene (høyre).



Figur 2.2: Gråtonebilde (venstre) med tilhørende intensitetsverdier for hver piksel i bildet (høyre).

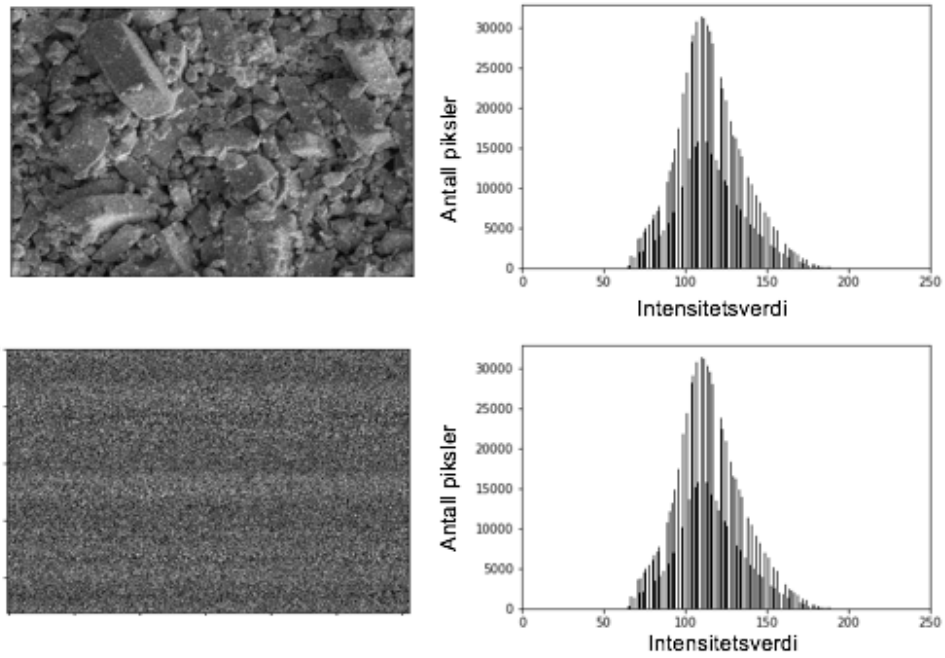
Intensitetsverdiene til et bilde kan også fremstilles i et histogram, der hver søyle i histogrammet representerer intensitetsverdiene som er tilstede i bildet. Høyden på søylene gjenspeiler antall piksler i bildet med hver intensitetsverdi. Figur 2.3 viser to eksempler på gråtonebilder (venstre) og tilhørende histogrammer for intensitetsverdiene (høyre).



Figur 2.3: Figuren viser to eksempler på 8-bits gråtonebilder (venstre) med tilhørende histogrammer for intensitetsverdiene (høyre). Histogrammene viser intensitetsverdi på x-aksen og antall piksler på y-aksen.

Det finnes ulike tilnærminger til uthenting av informasjon, eller *egenskaper*, fra et digitalt gråtonebilde. Den enkleste metoden baserer seg på førsteordens statistikk, som kun tar hensyn til fordelingen av gråtoner i bildet basert på bildets histogram (se Figur 2.3). Målene fra førsteordens statistikk inkluderer gjennomsnittsintensitet, den vanligste intensiteten, og formen på histogrammet. Imidlertid tar ikke metoden hensyn til den romlige fordelingen av pikslene i bildet, og vil i mange tilfeller ikke være tilstrekkelig for å skille bilder fra hverandre. Dette er illustrer i Figur 2.4: til venstre i figuren vises det opprinnelige bildet (øverst) og en randomisert versjon av bildet (nederst), og tilhørende histogrammer (høyre). At histogrammene er like illustrerer at to bilder kan ha samme gråtonefordeling og samtidig se helt forskjellige ut. Dermed finnes det en rekke metoder for å trekke ut *tekstur*-egenskaper fra bilder basert på andreordens statistikk, der den romlige fordelingen av pikslene tas i betraktning og karakteriseres [9]. Med slike egenskaper åpner det seg flere muligheter for å for eksempel skille bilder fra hverandre.

Tekstur kan brukes for å identifisere objekter og områder i et bilde, eller skille bilder fra hverandre, uavhengig av hva slags bilde som analyseres [9]. Tekstur kan gi informasjon om karakteristiske faktorer i bildet, slik som for eksempel *ruhet*, *glatthet* og grad av ordnethet. Haralick et al. viser eksempler på hvordan teksturegenskaper kan brukes til å klassifisere forskjellige typer stein [9]. I Kvaal et al. [30] benyttes multivariat teksturanalyse på bilder av brød og i Haralick et al. blir enkle teksturelle egenskaper beregnet for å klassifisere forskjellige typer sandstein. Eksempler på egenskaper basert på tekstur er Gray Level Co-occurence Matrix



Figur 2.4: Figuren viser et gråtonebilde (øverst, venstre) og en randomisert versjon av bildet der pikslene er stokket om tilfeldig (nederst, venstre) med tilhørende histogrammer (høyre) til bildene. Histogrammene er identiske og illustrerer at to bilder kan ha samme gråtonefordeling og samtidig se helt forskjellige ut.

(GLCM) [9, 31], Gray Level Run-Length Matrix (GLRLM) [32] og Angle Measure Technique (AMT) [7, 8].

Både intensitet og tekstur er alltid tilstede i et bilde, men en av dem kan dominere over den andre [9]. Når det er lite variasjon i gråtonene i et område i et bilde, vil det være intensiteten som dominerer. Er det stor variasjon i gråtonene på et lite område i bilde, vil det være tekstur som dominerer. Hvilken som dominerer avhenger i høy grad av størrelsen på området som betraktes. Ved å betrakte én piksel er det kun intensitet som er beskrivende. Økes størrelsen på området vil tekstur dominere.

2.4 Overvåkede og ikke-overvåkede metoder for klassifisering

Klassifisering innebærer å dele observasjoner i et datasett inn i grupper, det vil si tildele hver observasjon en *klasse*. Klassifisering kan deles inn i to hovedkategorier; overvåket (*supervised*) og ikke-overvåket (*unsupervised*) [33]. I overvåket klassifisering er de forskjellige klassene kjent på forhånd. I ikke-overvåket klassifisering finnes det ingen informasjon om klassene på forhånd,

slik at informasjonen om klassene må oppdages av klassifikatoren. En klassifikator vil i et slikt tilfelle undersøke dataene nøye og søke etter mønstre eller klynger med observasjoner som deler like egenskaper og som kan danne grunnlaget for en klassifisering. Prinsipalkomponentanalyse (PCA) er et eksempel på en ikke-overvåket metode, mens Support Vector Machines (SVM) er et eksempel på en overvåket metode [33]. Innenfor de to hovedkategoriene med klassifikatorer finnes det flere underkategorier; det finnes klassifikatorer med lineære (Logistisk regresjon) og ikke-lineære beslutningsgrenser [33], det finnes beslutningstrær [33, 34], nevrale nett [34] og ensemble-metoder (Random Forest, AdaBoost) [35, 36] hvor en klassifikator består av et ensemble med klassifikatorer. En rekke klassifikatorer, samt PCA [37], blir beskrevet i de påfølgende delkapitlene.

Analyse av ulike slag har ett hovedtrekk felles; det innebærer behandling av *data*. Data kan deles inn i forklaringsvariabler, representert ved datamatrisen X, og responsvariabler, representert ved vektoren y. Datamatrisen X består av n variabler (kolonner) og m prøver (rader), slik at X er en $m \times n$ matrise. Responsen y er en $m \times 1$ vektor, som inneholder responsen for hver prøve. Matrisen X og vektoren y kan da presenteres slik:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

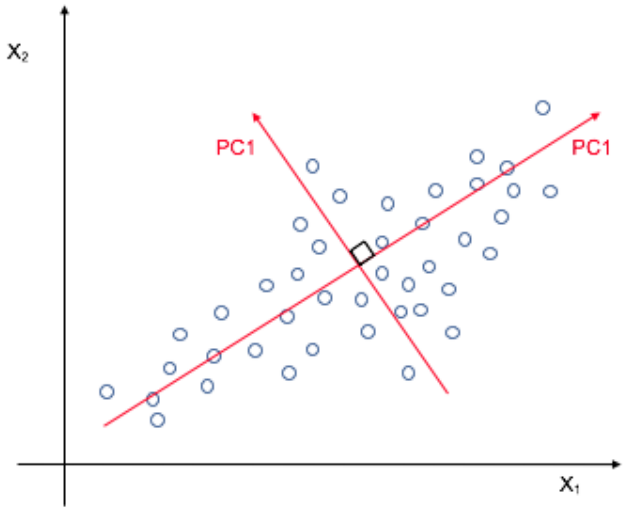
For eksempel for et problem med to klasser, vil y være en vektor bestående av 0 og 1. Hensikten med analysen er å forklare responsvariablene ved hjelp av forklaringsvariablene. I et klassifiseringsproblem basert på bildeteksturanalyse, vil hver variabel i X representere de forskjellige teksturegenskapene til hvert av bildene som analyseres og y bestå av klassen til hvert bilde.

2.4.1 Prinsipalkomponentanalyse

Prinsipalkomponentanalyse (PCA) er en metode som brukes til å komprimere dimensjonen til et datasett, slik at antall variable i datasettet reduseres. I en slik analyse erstattes de opprinnelige variablene med et sett med nye variable kalt prinsipalkomponenter. Disse er lineærkombinasjoner av de opprinnelige variablene og konstrueres slik at de fanger opp variasjonen i datasettet best mulig. På denne måten kan datasettet inneholde færre komponenter som kan gi (nesten) like mye informasjon som de opprinnelige variablene [37].

Prinsipalkomponentene er ortogonale, det vil si at de står vinkelrett på hverandre, og de spenner ut et nytt koordinatsystem sett i forhold til de opprinnelige variablene. Den første komponenten (PC1) velges i den retningen i datasettet som har størst varians, den andre komponenten (PC2)

velges i den retningen i datasettet som har nest størst varians *og* står vinkelrett på PC1, den tredje komponenten velges i den retningen med tredje størst varians *og* som står vinkelrett på PC1 *og* PC2, osv.



Figur 2.5: Figuren viser et eksempel på et todimensjonalt tilfelle hvor de to første prinsipalkomponentene (PC1 og PC2) velges i forhold til de opprinnelige variablene (x_1 og x_2). PC1 velges i retningen med størst varians i datapunktene (blå sirkler) og PC2 velges i retningen med nest mest varians i datapunktene og er ortogonal med PC1.

Figur 2.5 viser hvordan to prinsipalkomponenter (PC1 og PC2) velges i et to-dimensjonalt system med to variable (x_1 og x_2). Figuren viser at PC1 velges i den retningen der dataene har størst spredning og at PC2 velges i retningen med størst spredning *og* er ortogonal med PC1.

PCA er en ikke-overvåket metode som også kan brukes til å lage klynger (”clustering”) av data [38]. I analysen identifiseres eventuelle mønstre i dataene som kan danne grunnlag for klynger, eller det undersøkes om dataene danner klynger av observasjoner som deler bestemte egenskaper.

2.4.2 Logistisk regresjon

En lineær regresjonsmodell kan beskrives slik som i likning 2.2:

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_p x_{ip} + \varepsilon_i, \tag{2.2}$$

der y_i er responsvariabelen på den $i$te prøven i datasettet, b_0 er skjæringspunktet, b_j er den

estimerte koeffisienten for den $j$te forklaringsvariabelen for den $i$te prøven, x_{ij} er verdien for den $j$te forklaringsvariabelen for den $i$te prøven, og ε_i er feilleddet som ikke kan forklares av modellen. Denne modellen vil prøve å predikere responsvariablene y, mens i logistisk regresjon er målet å gi en sannsynlighet for hvilken kategori eller klasse y hører til [33]. Alle sannsynlighetene vil ligge mellom 0 og 1. I et binært tilfelle vil klassene være 0 eller 1, for eksempel representert ved en prediksjon om en pasient er syk eller frisk, slik:

$$Y = \begin{cases} 0 & \text{hvis syk} \\ 1 & \text{hvis frisk} \end{cases} \tag{2.3}$$

I et slikt tilfelle kan en definere alt med en sannsynlighet over 0,5 er frisk og alt under er syk. For å kunne gjøre dette trengs det en modell som gir verdier mellom 0 og 1. I logistisk regresjon brukes den *logistiske funksjonen* [33]:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{2.4}$$

der X er en forklaringsvariabel, og β_0 og β_1 er koeffisienter. Dersom det er flere forklaringsvariabler, kan modellen skrives om slik:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}, \tag{2.5}$$

der p er antall forklaringsvariabler. For å tilpasse modellen gitt i likning 2.4 brukes en metode som kalles *maximum likelihood*, hvor β_0 og β_1 tilpasses slik at den predikerte sannsynligheten $p_i(X)$ for prøve i gitt av likning 2.4, blir så nær som mulig den sanne verdien til prøven [33]. I likning 2.3 er dermed målet å få $p_i(X)$ så nærme 0 som mulig dersom pasienten er syk. Likning 2.4 skrives om som en funksjon av X, og logaritmen tas på begge sider, slik at:

$$log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X \tag{2.6}$$

I denne oppgaven blir logistisk regresjon brukt til å predikere seks klasser, noe som gjør modellen en del mer komplisert. Modellen vil da bestå av et sett med modeller, slik som i likning 2.4, og alle sannsynlighetene blir summert opp til 1 [39].

L1-regularisering i logistisk regresjon

Algoritmen til logistisk regresjon som klassifikator benytter regularisering for å identifisere de mest optimale variablene i en modell. Regularisering er et konsept som benyttes for å håndtere overtilpasning av data og redusere kompleksiteten til en modell [34]. Det er et nyttig verktøy for å redusere kollinearitet, det vil si korrelasjon mellom egenskaper, og identifisere egenskaper som skaper støy, og derigjennom unngå overtilpasning. Ideen bak regularisering er å tilføre supplerende informasjon om kompleksiteten til modellen ved å "straffe" ("penalize") de mest ekstreme variablene (vekter) [34]. En rekke teknikker for regularisering blir presentert av Friedman et al. [40]. En regulariseringsmetode som blir beskrevet nærmere i det følgende er L1-regularisering.

L1-regularisering, også kalt "Lasso", ble introdusert av Tibshirani [41] i 1996. L1-regularisering er en teknikk for egenskapsutvelgelse som egner seg godt for datasett med mange variable, altså et datasett med høy dimensjon [34]. I et datasett med mange egenskaper, særlig i tilfeller med mange flere egenskaper enn prøver, er det stor sannsynlighet for at flere av egenskapene er irrelevante for modellen. L1-regularisering krymper vektingen av mange av de mindre viktige egenskapene til null, slik at de blir fjernet. L1-regularisering egner seg dermed godt som metode for egenskapsutvelgelse i tilfeller med mange egenskaper og få observasjoner.

L1-regularisering baseres på logistisk regresjon med en tilhørende kostnadsfunksjon, som kvantifiserer kostnaden (feilen) som fremkommer ved å klassifisere alle treningsobservasjonene. Kostnadsfunksjonen kan reguleriseres ved å legge til et ledd bestående av L1-regulariseringen vektet med en regulariseringsparameter λ.

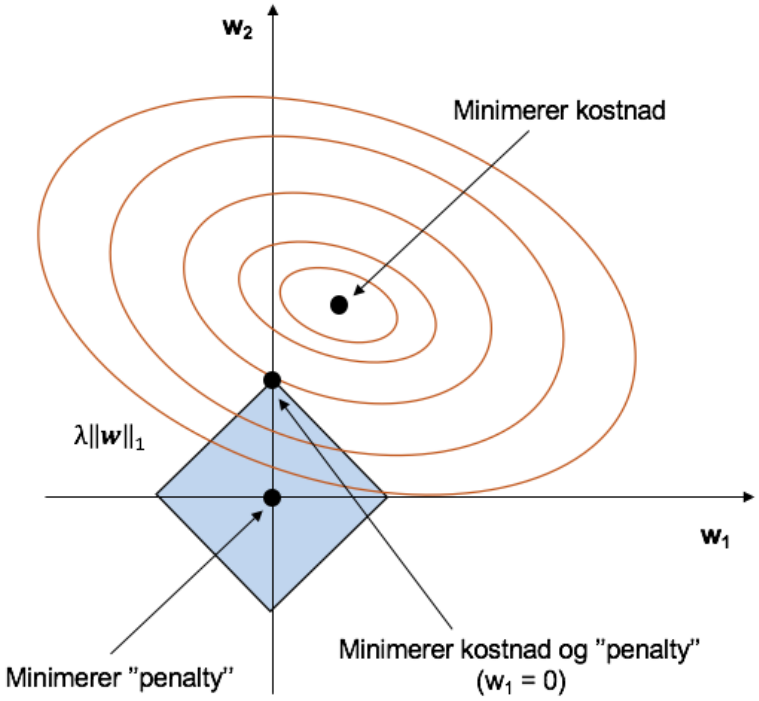
L1-regulariseringen $||\mathbf{w}||$ fremstilles ved summen av absoluttverdien til alle egenskapsvektene:

$$||\mathbf{w}|| = \sum_{j=1}^{m} |w_j| \tag{2.7}$$

Regulariseringsparameteren λ bestemmer hvor streng regulariseringen er og hvor godt modellen kan tilpasses treningsdataene samtidig som vektene holdes lave. Størrelsen på λ angir størrelsen på *begrensningsområdet* for regulariseringen, gitt ved likning 2.7. Det vil for eksempel bety at en lav verdi av λ gir et lite begrensningsområde og en streng regularisering.

Konturene av en konveks kostnadsfunksjon (SSE-kostnadsfunksjon) kan plottes med to vektingskoeffisienter (w_1 og w_2) på aksene, slik som i Figur 2.6. Konturene til kostnadsfunksjonen er angitt ved ellipsene. I opptrening av en klassifiseringsmodell er det ønskelig å identifisere kombinasjonen av verdier w_1 og w_2 som minimerer kostnaden (feilen) til klassifiseringen av

treningsdataene [34]. Dette punktet er angitt ved det sorte punktet i den minste ellipsen. I et slikt tilfelle er det stor risiko for overtilpasning, og en regularisering vil kunne være nødvendig for å minimere modellens avhengighet til treningsdataene. Begrensningsområdet for regulariseringen er angitt ved den blå "diamanten", der sidelengdene er proporsjonale med λ. Diamant-formen er karakteristisk for begrensningsområdet i L1-regularisering i et tilfelle med to vekter [33]. Ved å inkludere regulariseringen "oppfordres" høye vektingsverdier (minimere "penalty"), og i ytterste konsekvens "tvinges" både w_1 og w_2 til null (angitt ved sort punkt i skjæringen mellom aksene). En avveining mellom de to nevnte scenariene, kan medføre et tredje tilfelle, som både minimerer kostnad og "straff" ("penalty"). Et slikt tilfelle er markert i figuren, angitt ved punktet kostnadsfunksjonen skjærer begrensningsområdet i det ene hjørnet på "diamanten", og w_1 er lik null.



Figur 2.6: Grafisk fremstilling som viser betydningen av L1-regularisering i et klassifiseringstilfelle. Konturene av en konveks kostnadsfunksjon er angitt ved ellipsene og begrensningsområdet til regulariseringen er angitt ved den blå "diamanten". De svarte punkene angir hvilke kombinasjoner av egenskapsvektene w_1 og w_2 som minimerer straff ("penalty"), minimerer kostnad og minimerer både straff og kostnad. Tilpasset fra [34].

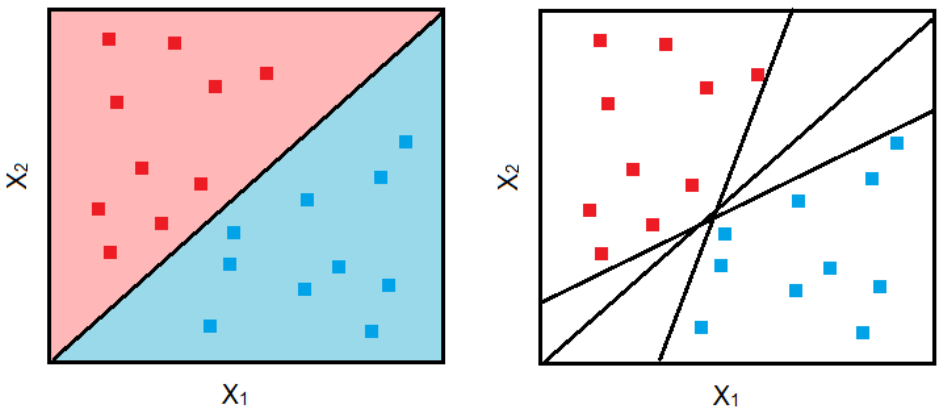
2.4.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) ble utviklet på 1990-tallet og har vist seg å prestere bra på flere forskjellige områder [33]. SVM består av tre forskjellige klassifikatorer; *maximal margin classifier*, *support vector classifier* og *support vector machine*, som alle har litt forskjellige

begrensninger. *Maximal margin classifier* er forholdsvis enkel, men kan ikke benyttes der klassene ikke kan skilles med en lineær grense. *Support vector classifier* er en utvidelse av den forrige, og kan benyttes på flere typer datasett. *Support vector machine* er en videre utvidelse av *support vector classifier* igjen, og kan brukes der grensene mellom klassene er ikke-lineære.

Når to klasser skal skilles fra hverandre, kan de skilles med et såkalt *hyperplan*. Hvis klassene ligger i et n-dimensjonalt rom, hvor n tilsvarer antall variabler, kan de bli skilt med et underrom som har $n - 1$ dimensjoner [33]. For eksempel, dersom det er to klasser med observasjoner som ligger i et to-dimensjonalt plan, kan disse bli skilt med en endimensjonal linje, som vist i Figur 2.7 (venstre). Tilsvarende, dersom observasjonene ligger i et tre-dimensjonalt rom, kan de bli skilt med et to-dimensjonalt plan. Dette gjelder også for høyere dimensjoner.

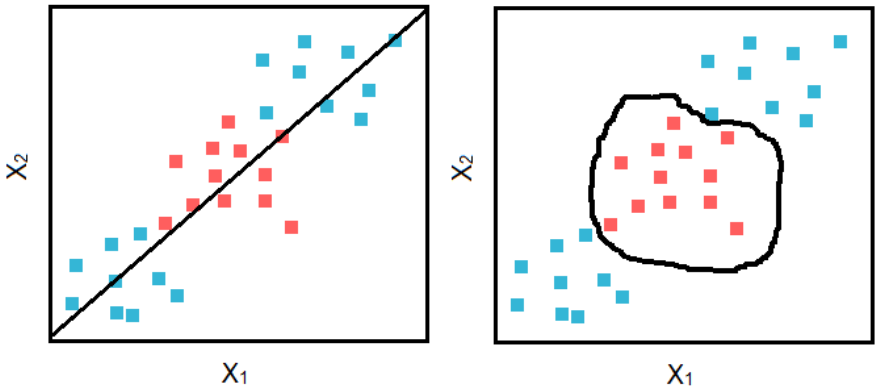
Til venstre i Figur 2.7 vil en ny observasjon havne i den røde eller blå klassen avhengig av hvilket område observasjonen havner i. I den høyre figuren finnes det flere forskjellige hyperplan som kan skille klassene. I prinsippet finnes det et uendelig antall slike hyperplan [33]. *Maximal margin classifier* forsøker å finne det hyperplanet som separerer klassene optimalt [33]. Målet her er å gjøre om en lineær klassifikator til en klassifikator som kan lage ikke-lineære beslutningsgrenser (*decision boundaries*). Support vector machine gjør dette automatisk.



Figur 2.7: I figurene vises det to klasser (markert med røde og blå punkter) med to variabler, X_1 og X_2. I figuren til venstre er klassene skilt med et hyperplan, og nye prøver vil bli tildelt klasse ut i fra om de havner i det røde eller det blå området. I figuren til høyre vises de samme prøvene, men med tre forskjellige hyperplan som alle kan skille klassene. Tipasset fra [33]

Klassene kan ikke alltid skilles med et hyperplan, og *maximal margin classifier* har en tendens til å flytte grensen mellom klassene basert på en enkel observasjon. Dette betyr at den lett kan predikere nye testobservasjoner feil. *Support vector classifier* er en mer robust modell, som kan klassifisere nye treningsobservasjoner bedre [33]. Det fungerer på samme måte som *maximal margin classifier*, og skiller de fleste av treningsobservasjonene, men noen kan havne på feil side av hyperplanet.

Dersom det ikke er mulig å skille klassene med en lineær beslutningsgrense, slik som vist i Figur 2.8, kan *support vector machine* (SVM) benyttes. Dette er en utvidelse av *support vector classifier*, hvor dimensjonen som observasjonene ligger i økes, slik at en ikke-lineær grense mellom klassene kan lages [33].



Figur 2.8: Figurene viser to klasser (markert med røde og blå punkter) som er tydelig adskilt. I figuren til venstre vises det at klassene ikke kan skilles lineært. Til høyre er et eksempel på en ikke-lineær beslutningsgrense som for eksempel kan lages med en SVM.

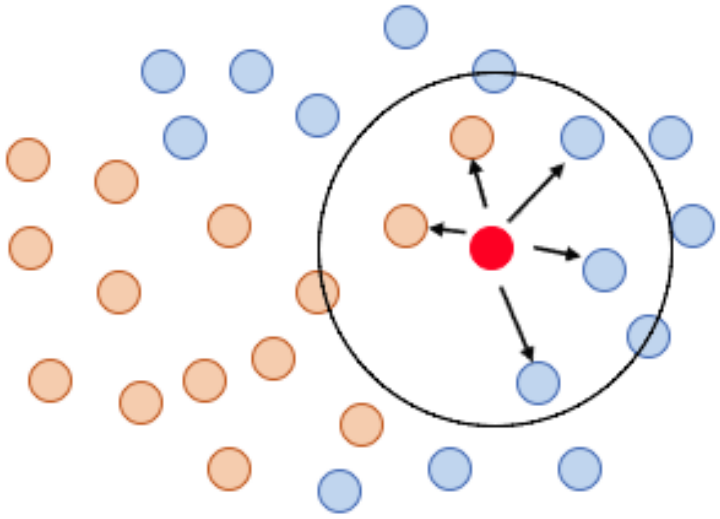
Figurene 2.7 og 2.8 viser eksempler med kun to klasser. Skal SVM benyttes i tilfeller med $K >$ antall klasser, finnes det to forskjellige metoder; ”en-mot-en” eller ”en-mot-alle” [33]. En-mot-en-metoden lager $\binom{K}{2}$ forskjellige klassifiseringsmodeller, en for alle parkombinasjonene. For hver klassifiseringsmodell blir alle testobservasjonene tildelt en klasse. Den endelige klassen en testobservasjon blir tildelt er den klassen testobservasjonen ble tildelt flest ganger i alle de $\binom{K}{2}$ forskjellige klassifiseringsmodellene. En-mot-alle-metoden tilpasser K klassifikatorer og sammenligner én og én mot resten [33].

2.4.4 K-nærmeste nabo

K-nærmeste nabo (K-Nearest Neighbors, knn) er en overvåket og ikke-lineær algoritme som brukes i klassifisering. Algoritmen beregner avstanden fra en ukjent prøve til enhver prøve i treningssettet og finner de k nærmeste naboene i treningssettet. En ukjent prøve blir lagt til i den klassen som har flest representanter blant de k nærmeste. En fordel med denne måten å klassifisere på er at det ikke tas antagelser om formen på gruppene [42]. I et tilfelle med to klasser, er det fordelaktig å benytte verdier av k som er oddetall, da muligheten for at en ukjent prøve har like mange naboer i hver av klassene blir unngått. Med flere klasser må det angis et supplerende kriterium for klassifiseringen. Figur 2.9 viser et eksempel på hvordan en knn-algoritme klassifiserer en ukjent prøve i et tilfelle med to klasser.

Når knn-klassifikatoren benyttes er det viktig å betrakte treningssettet og måten hver prøve i

dette fordeler seg i de ulike klassene på. For eksempel vil klassifiseringen av en ukjent prøve kunne avhenge av hvor mange medlemmer hver av klassene har. En ukjent prøve vil altså kunne ha større sannsynlighet for å bli klassifisert til en klasse som er bedre representert (i antall) enn en annen, hvilket ikke er fordelaktig. En ideell situasjon vil være at antall medlemmer fra treningssettet i hver klasse samsvare med den forutgående sannsynligheten for at en ukjent prøve tilhører klassen [42]. Det optimale valget av k er i høy grad avhengig av dataene som analyseres. Generelt sett vil en større verdi av k føre til reduksjon av støy, men vil gå på bekostning av mindre distinkte beslutningsgrenser [43].



Figur 2.9: Eksempel som viser K-nærmeste nabo-metoden. De oransje og de blå punktene representerer prøver fra to forskjellige klasser i treningssettet. Pilene angir avstanden fra en ukjent prøve fra testsettet (rødt punkt) til de nærmeste naboene. Ved én nærmeste nabo vil den ukjente prøven klassifiseres til den oransje klassen siden et medlem fra den klassen er den aller nærmeste naboen. Ved fem nærmeste nabo klassifiseres den ukjente prøven til den blå klassen. Tilpasset fra [42].

Et annet aspekt ved knn-klassifisering som må belyses, er den potensielt omfattende beregningstiden for algoritmen. Årsaken til tidsbruken er at for enhver ny ukjent prøve som skal klassifiseres, må avstanden til samtlige prøver i treningssettet beregnes. Omfattende beregningstid vil særlig være tilfellet med et stort datasett med mange prøver i treningssettet og mange ukjente prøver som skal klassifiseres [42].

2.4.5 Beslutningstrær og Random Forest

Beslutningstre (Decision tree, DT) og Random forest (RFC) er to ikke-lineære klassifikatorer som baserer seg på beslutningstrær. Beslutningstrær kan anvendes på både regresjon og klassifisering

[33]. I denne oppgaven benyttes beslutningstrær for klassifikasjon.

En beslutningstre-modell har som formål å bryte ned dataene og ta en beslutning basert på en rekke spørsmål som blir ”stilt” underveis i prosessen [33]. Disse ”spørsmålene” skal ende med at hver observasjon som analyseres tildeles en klasse i enden av beslutningstreet. Algoritmen begynner ved ”roten” til treet og splitter dataene på en slik måte som bidrar til størst informasjonsgevinst [34]. Informasjonsgevinst defineres som en funksjon av et splitte-kriterium, eller grad av ”urenhet”, som er et mål på sannsynligheten for å feilklassifisere [34]. For hvert steg i prosessen, eller ved hver *node* i treet, blir klasser avslått. En gren slutter der det oppnås en klasse som aksepteres [14]. Figur 2.10 viser et eksempel på et beslutningstre med kategoriske variable. På figuren markerer de blå boksene spørsmål som blir stilt (indre node) og de røde boksene markerer klassene. Et beslutningstre deler ”egenskapsrommet” inn i rektangler, og konstruerer på den måten komplekse beslutningsområder for klassifikasjonen. Dybden på treet, altså hvor mange spørsmål som blir stilt, avgjør hvor kompleks beslutningsområdene er. Et viktig poeng i denne forbindelse er å ikke velge et for dypt tre, da dette kan føre til overtilpasning av modellen [34]. Dybden på treet bestemmes av brukeren. En metode for å redusere sannsynligheten for overtilpasning er ved å kombinere flere beslutningstrær i det som kalles ”Random Forest”.

Figur 2.10: Figuren viser en enkel illustrasjon på et beslutningstre. Det blir stilt spørsmål i de blå boksene, og ut i fra dette ender en opp i de forskjellige klassene (røde boksene).

Random Forest er en robust klassifikator, som på bakgrunn av god ytelse, brukervennlighet og skalerbarhet, er populær å anvende for maskinlæringsproblemer [34]. Denne klassifikatoren kan anses som en samling av beslutningstrær [34] og omtales derfor som en *ensemble*-metode. Virkemåten til Random Forest går ut på at det blir tatt et gjennomsnitt av flere enkeltstående beslutningstrær som hver innehar stor varians. Random Forest-prosessen kan i så måte omtales som en *de-korrelering* av de enkeltstående trærne. På denne måten fremstår Random Forest som en klassifikator som bygger en *robust* modell. En Random Forest-modell kan dermed forventes å ha god generell ytelse og er mindre utsatt for overtilpasning [34], hvilket er en stor fordel når modellen skal anvendes på et maskinglæringsproblem. Hovedstegene til Random Forest-algoritmen presenteres i det følgende:

1. For i = 1 til k, hvor k er antall trær, gjøres følgende:

   (a) Et undersett bestående av n prøver velges tilfeldig ut fra treningssettet *med* tilbakelegging, altså genereres en såkalt *"bootstrap"* [38].

   (b) Et beslutningstre, eller en *modell*, bygges basert på undersettet. Ved hver indre node (splitt) gjøres følgende:

      i. d egenskaper velges ut *med* tilbakelegging.

      ii. Noden splittes basert på de egenskapene som gir den beste splitten hva angår informasjonsgevinst.

2. Prediksjonen fra hvert beslutningstre brukes til å tildele klasser til prøvene. Dette gjøres ved flertallsvalg ("*majority vote*"), altså blir klassene som blir predikert flest ganger valgt som den endelige klassen [34].

For en Random Forest-modell må antall trær (verdi av k) velges. Generelt sett øker modellens nøyaktighet med antall trær, men det går på bekostning av beregningstid for algoritmen [34].

2.4.6 Boosting

I denne oppgaven blir det tatt for seg to typer boosting-modeller som kan brukes til å klassifisere; AdaBoost og Gradient Boosting Classifier (GBC). AdaBoost og GBC er to metoder tilhørende en gruppe med ensemble-metoder som bruker "boosting", og som i likhet med Random Forest brukes for å forbedre ytelsen til et beslutningstre. I "boosting" benyttes enkle klassifikatorer ("weak learners"), som gjerne kun så vidt har ytelse som overskrider tilfeldig tildeling av klasse [34]. Denne metoden anvendes gjerne på datasett som er vanskelig å klassifisere. De svake klassifikatorenes feilaktige klassifisering brukes til å styrke ensemblet [34]. I "boosting" gror trærne fortløpende, i motsetning til Random Forest, der trærne gror uavhengig av hverandre [33]. I "boosting" brukes altså informasjonen fra tidligere trær når et nytt tre skal dannes.

AdaBoost er den hyppigst brukte ensemble-klassifikatoren som bygger på "boosting". Den praktiske tilnærmingen til algoritmen bak AdaBoost ble undersøkt og eksperimentert med av Freund og Schapire [36] i 1996. Algoritmen genererer en rekke svake klassifikatorer. For hver iterasjon (k iterasjoner) blir den beste klassifikatoren basert på de gjeldende vektede datapunktene [38] identifisert. En prøve som blir feilaktig klassifisert i den $k$te iterasjonen vil bli tillagt høyere vekting i den $(k+1)$te iterasjonen. Prøver som blir klassifisert riktig vil derimot bli tillagt lavere vekting i påfølgende iterasjon [38]. Denne fremgangsmåten vil altså bety at de prøvene som er vanskelig å klassifisere vil få stadig høyere vekting i løpet av de

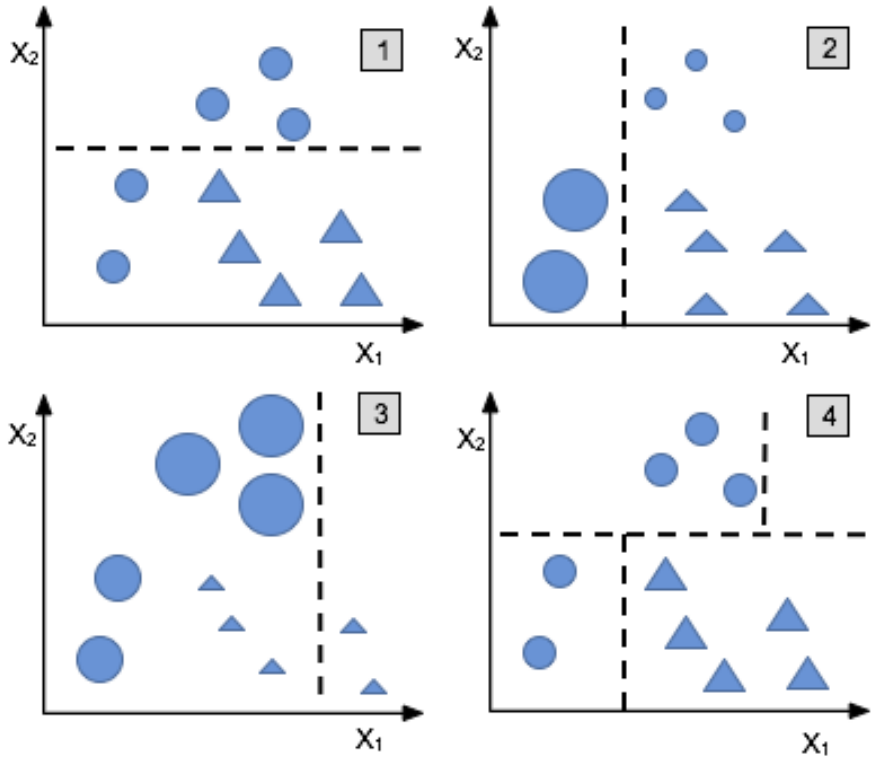
k iterasjonene, inntil algoritmen oppdager en modell som kan klassifisere disse riktig [38]. Algoritmen fokuserer altså på områder i dataene som er vanskelig å klassifisere. AdaBoost anses å være motstandsdyktig mot overtilpasning av en modell [35].

For hver iterasjon $(1, ..., k)$ tilpasses en svak klassifikator basert på de vektede datapunktene og den $k$te modellens feilklassifiseringen err_k. Den trinnvise vektingen for hver k beregnes ved [38]:

$$ln(\frac{1 - err_k}{err_k}) \tag{2.8}$$

der err_k er feilklassifiseringen til iterasjon k.

Vektingen til hvert datapunkt oppdateres så for hver iterasjon. Avslutningsvis i algoritmen kombineres de vektede klassifikatorene til et *ensemble* (”additive” modell) og har muligheten til sammen å kunne klassifisere bedre enn hver klassifikator har kapasitet til individuelt [38].



Figur 2.11: Illustrasjon som viser hvordan AdaBoost-algoritmen klassifiserer i et tilfelle med to klasser (sirkler og trekanter), her eksemplifisert med en algoritme med fire iterasjoner. For hver iterasjon blir det utført en klassifikasjon (markert med stiplet linje) og størrelsen på datapunktene øker i størrelse ved riktig klassifikasjon og minker i størrelse ved feilklassifikasjon. Tilpasset fra [34].

Figur 2.11 viser et eksempel på klassifikasjon med to klasser (markert med henholdsvis sirkler og trekanter) og fire iterasjoner. De stiplede linjene representerer beslutningsgrensen for klassifikatoren. For hver iterasjon kommer det frem hvordan representanter fra de to klassene enten blir større (høyere vekting) eller mindre (lavere vekting) ettersom de blir henholdsvis feilklassifisert eller riktig klassifisert. I det fjerde steget er informasjonen fra de tre foregående stegene kombinert, slik at *ensemblet* klarer å skille de to klassene.

I Friedman et al. [35] blir flere versjoner av AdaBoost presentert og der blir AdaBoost også satt sammen med en ”loss funksjon”, en *additiv modell* og logistisk regresjon. Det blir vist at boosting kan bli tolket som en forover trinnvis modell som kan minimere eksponentielle tap [35, 38]. Det ble så laget et rammeverk som førte til generaliseringer av Real AdaBoost, Gentle AdaBoost og LogitBoost som blir diskutert av Friedman et al. [35]. Dette rammeverket kan tilpasses mye og metodene som det inneholder kaller Friedman for ”gradient boosting machines” i en annen artikkel som ble publisert året etter [44]. Denne metoden omfatter både regresjon og klassifikasjon [38].

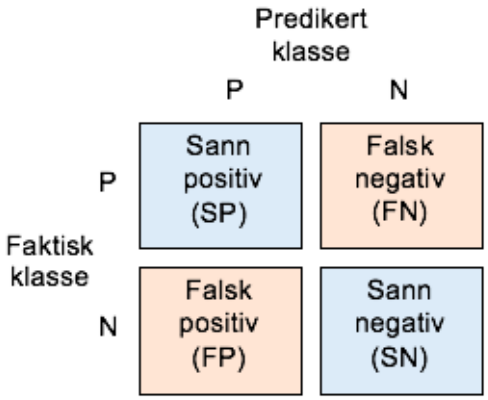
Grunnprinsippet til gradient boosting, i motsetning til AdaBoosting, er at det starter med en ”loss funksjon” som for eksempel minste kvadraters metode (square error) som blir mye brukt i regresjon, og en enkel klassifikator (beslutningstre). Så brukes en algoritme som skal finne en ”additiv modell” som de svake læringsalgoritmene blir lagt til slik at ”loss funksjonen” blir minimert [38]. Gradienten blir regnet ut (som for eksempel residualet) og en modell blir så tilpasset gradienten slik at ”loss funksjonen” blir minimert. Gradienten definesres som den partiellderiverte av ”loss funksjonen” med hensyn på prediksjonsmodellen [39].

Første steg i klassifisering med gradient boosting er å initialisere en prediksjonsmodell. Videre går algoritmen over et gitt antall iterasjoner der gradienten blir regnet ut, datasettet blir splittet tilfeldig opp i et testsett og et treningssett, beslutningstre-modellen trenes opp og modellen som blir initialisert blir oppdatert.

2.5 Evaluering av nøyaktigheten til en modell

Ytelsen til en prediksjonsmodell bestemmes ut ifra hvor god den er til å predikere virkeligheten gjennom klassifiseringen. Det finnes flere måter å uttrykke hvor god ytelsen er, og hvilken måte som brukes til hvilket formål avhenger ofte av hva slags datasett som analyseres. For et ubalansert datasett, der for eksempel én klasse er sterkere representert enn en annen, kan ROC AUC (Area Under Curve) være et godt mål på nøyaktighet. Andre alternativer er presisjon (*”precision”*), *”Recall”* og *”F1-score”* [34]. For et balansert datasett, benyttes gjerne *nøyaktighet* som et mål på ytelsen.

Ytelsen kan beskrives ved en såkalt *forvirringsmatrise* (confusion matrix), se Figur 2.12. De blå boksene i figuren angir de riktige prediksjonene, det vil si de sanne positive (SP) og de sanne negative (SN), ergo de prediksjonene som stemmer overens med virkeligheten. De lyserøde boksene angir de feilaktige prediksjonene, det vil si de falske positive (FP) og de falske negative (FN), ergo de prediksjonene som *ikke* stemmer overens med virkeligheten. Det er viktig å påpeke at nevnte figur gjelder for et binært tilfelle, altså et tilfelle med kun to klasser. I denne oppgaven opereres det med seks klasser, men figuren står frem som et illustrerende eksempel.



Figur 2.12: Forvirringsmatrise for et tilfelle med to klasser. De blå boksene angir de riktige prediksjonene, det vil si de sanne positive (SP) og de sanne negative (SN). De lyserøde boksene angir de feilaktige prediksjonene, det vil si de falske positive (FP) og de falske negative (FN).

Nøyaktigheten (ofte omtalt som "Accuracy" i litteraturen) forstås som antall riktige prediksjoner dividert på antall prediksjoner (merk at denne formelen gjelder for et binært klassifiseringstilfelle):

$$\text{nøyaktighet} = \frac{\text{antall riktige prediksjoner}}{\text{antall prediksjoner}} = \frac{SP + SN}{FP + FN + SP + SN} \tag{2.9}$$

der SP betegner de sanne positive, SN betegner de sanne negative, FP de falske positive og FN de falske negative. SP og SN angir prediksjoner som stemmer overens med virkeligheten, mens FP og FN angir prediksjonen som *ikke* stemmer overens med virkeligheten.

I klassifiserings-systemer med *flere* klasser, benyttes andre beregningsmetoder. For eksempel, via en-mot-alle-klassifikasjon beregnes gjennomsnittet fra alle prediksjonen i systemet, og kan forstås som en utvidelse av forvirringsmatrisen vist i Figur 2.12. Se Raschka og Mirjalili [34] og Scikit-learn [43] for detaljer om beregningsmetoder for klassifisering med flere klasser.

3 Materialer

3.1 Datasettet

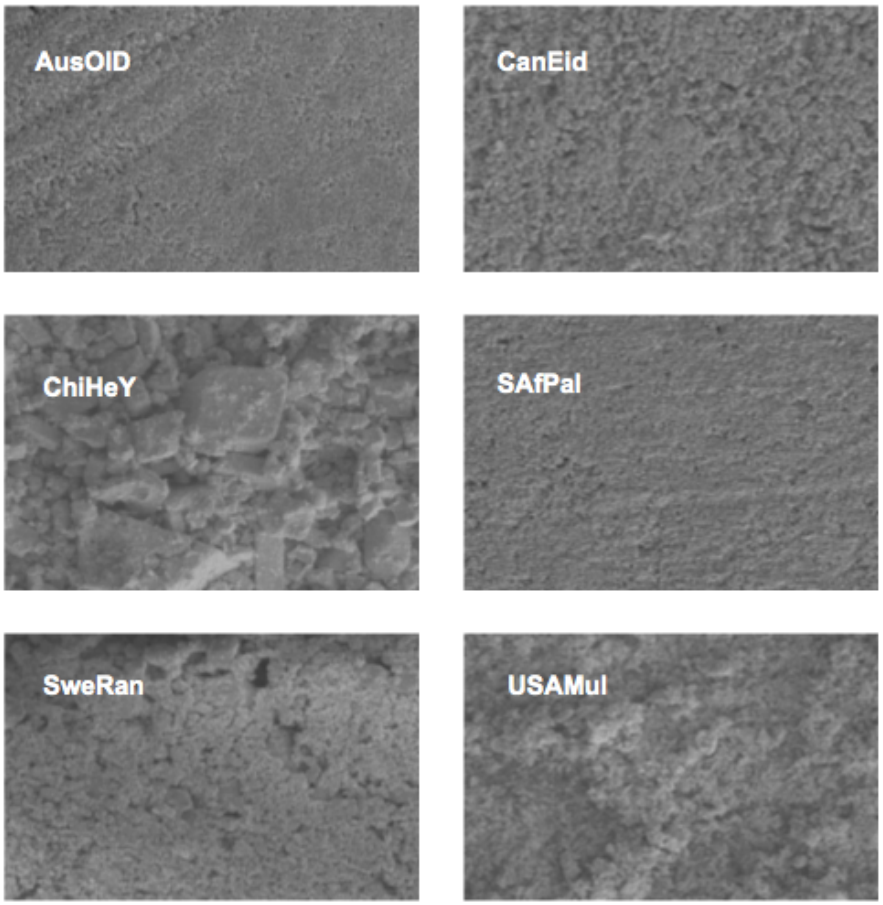
Datasettet stammer fra *European Commission, Joint Research Centre (JRC), Nuclear Safety and Security, Karlsruhe, Tyskland* [45] og består av SEM-bilder av uranprøver (Uranium Ore Concentrate, UOC). Uranprøvene har sitt opphav i 26 forskjellige produksjonssteder fordelt på elleve land. Det originale datasettet består av $2 \times 5 = 130$ SEM-bilder. Fra det originale datasettet er det tatt et tilfeldig utvalg av bilder fra seks produksjonssteder, og det er dermed seks *klasser*. Dette subsettet brukes for å utvikle og teste analyseprogrammet som blir utviklet i denne oppgaven, og det utgjør et datasett på $6 \times 5 = 30$ bilder. Tabell 3.1 viser en oversikt over de uranprøvene som er benyttet i analysene og viser prøvenummer, forkortelse, sted, opprinnelsesland, kjemisk sammensetning, samt prosessen som ligger til grunn for hver prøve [6]. I Figur 3.1 vises ett bilde fra hver av de seks prøvene (klassene). Bildene stammer fra seks ulike prosesseringssteder og det vises at prøvene har forskjellig struktur og kornstørrelse. Prøven AusAll og SAfPal har en glatt struktur og ser svært like ut. Prøvene CanEid og SweRan har også en del likheter, mens ChiHeY skiller seg mer ut fra de andre prøvene.

*Tabell 3.1: Oversikt over uranprøvene som er blitt benyttet i oppgaven. Prøvene merket med * er sorte i fargen, mens resten av prøvene er gule [6]. Tallene i den første kolonnen representerer de forskjellige klassene, og er det som har blitt brukt videre i beregningene. I kolonnen for prosess står ppt for 'precipitation', som betyr fellingsreaksjon.*

Prøve	**Forkortelse**	**Sted**	**Land**	**Kjemisk sammensetning**	**Prosess**
0	AusOlD	Olympic Dam	Australia	U_3O_8 **(U308)**	U ppt med NH_3
1	CanEid	El Dorado	Canada	ADU **(ADU)**	U ppt med $NaOH$
2*	ChiHey	Heng Yang	Kina	UO_2 (+ U_3U_8) **(UO2)**	Ukjent
3*	SAfPal	Palabora	Sør-Afrika	U_3O_8 **(U3O8)**	U ppt med NH_3
4	SweRan	Ranstad	Sverige	$Sodiumdiuranate$ **(SDU)**	U ppt med $NaOH$
5	USAMul	Mulberry	USA	ADU **(ADU)**	Ukjent

Analysen av SEM-bildene baserer seg på 8-bits tif-bilder med en oppløsning på 1850×2450 piksler. Hver uranprøve er avbildet fem ganger, der hvert bilde avbilder ulike deler av prøven. I SEM-opptaket av prøvene ble det benyttet en forstørrelsesfaktor på $250\times$.

Fongaro et al. [6] angir fremgangsmåten for hvordan prøvene ble forberedt for bildetaking gjennom elektromikroskopet: noen hundre milligram uranpulver ble plassert i en petriskål og overført til en ny beholder belagt med dobbeltsidig tape. Prøvene ble så dekket med et 10 nanometer lag med gull for å sikre konduktivitet i SEM-undersøkelsen. Se Fongaro et al. [6] for en mer detaljert beskrivelse.



Figur 3.1: Eksempelbilder fra hver av de seks klassene med SEM-bilder som blir analysert i denne oppgaven. Se Tabell 3.1 for detaljer om klassene.

3.1.1 Forberedelse og preprosessering av bildene

Datasettet bestående av 30 SEM-bilder er delt tilfeldig opp i et treningssett og et testsett i størrelsesforholdet 80 %/20 %, det vil si at treningssettet består av 24 bilder og testsettet består av 6 bilder. Videre er hvert av bildene delt opp i fire ikke-overlappende småbilder med en oppløsning lik 1225x925 piksler. Dette er for å utvide datasettet til å inneholde flere bilder til

analysen. Resultatet er et treningssett bestående av 96 småbilder og et testsett bestående av 24 småbilder. Det er disse bildene som blir analysert i denne oppgaven.

Hvert av småbildene i treningssettet og i testsettet var preprosessert på forhånd. Preprosessering av bildene korrigerer for intensitetsforskjeller i bildene, slik at de kan sammenlignes med hverandre i analysene. Bildene var preprosessert med standardisering i programverktøyet MATLAB.

3.2 Programvare

Datamaskinen HP ProBook 450 G3 med Intel® Core™ i7-6500U CPU prosessor (2,50 GHz, 2,59GHz, 8 GB RAM) og opperativsystem Windows 10 Pro, 64-bit ble brukt. For utvikling av analyseprogrammet i denne oppgaven ble programmeringsspråket Python (versjon 3.6) benyttet. Spyder [46] ble benyttet som integrert utviklingsmiljø (Integrated Development Environment, IDE). Spyder er en type programvare som brukes til å utvikle annen programvare. Ulike Pythonpakker ble installert på lokal datamaskin og benyttet; Sciki-learn (versjon 0.19.1) [47] for maskinlæring, Scikit-image [48] for bildebehandling og Pyradiomics [49] for beregning av bildeegenskaper og tekstur. Disse blir beskrevet i de påfølgende avsnittene. Microsoft Excel, v.16.0 ble benyttet til lagring av egenskaper og resultater. Plottene og figurer fra Spyder er laget med Matplotlib [50]. Varmekartene er laget ved hjelp av Seaborn [51].

Scikit-learn [47] er en ”open source”-programvare for maskinlæring i Python. Programvaren inneholder en stor samling med verktøy til bruk i dataanalyse. I denne oppgaven har flere moduler fra Scikit-learn blitt benyttet. Disse modulene inneholder en rekke klasser og funksjoner som brukes for å utføre spesifikke oppgaver. Modulene fra Scikit-learn er hentet fra nettsiden til Scikit-learn [43] (se Scikit-learn dokumentasjon for detaljer [52]). Modulene som ble benyttet i programutviklingen er presentert i Tabell 3.2. Klassene og funksjonene fra Scikit-learn blir omtalt i metodedelen i delkapitlene 4.3 og 4.6 i denne oppgaven.

Scikit-image [48] er også ”open source”-programvare og blir beskrevet som en samling med algoritmer for bildebehandling implementert for Python. Hensikten er å ha et godt dokumentert og brukervennlig verktøy for de vanligste bildebehandlingsalgoritmene, og hjemmesiden inneholder eksempler og detaljer [53]. Scikit-image er blitt brukt til å lese inn bilder (modulen ”skimage.io”) og til beregning av bildeegenskaper basert på ”Local Binary Patterns” (modulen ”skimage.feature.local_binary_pattern”).

Pyradiomics [49] er nok en ”open source”-programvare utviklet for bruk i Python. Programvaren har innebygde algoritmer som brukes for å beregne og trekke ut intensitet og tekstur fra bilder og består av syv metoder. Hovedbruksområdet til Pyradiomics er kvantifisering av egenskaper

og strukturer i medisinske tredimensjonale bilder, for eksempel CT-bilder av en kreftsvulst. I et slikt bilde defineres et sett med voksler (tredimensjonale bildeelementer) som avgrenser svulsten i bildet, altså området av interesse (”region of interest”, ROI), også kalt en *maske*. I denne oppgaven har Pyradiomics blitt tilpasset til todimensjonale bilder og brukt for å beregne og trekke ut egenskaper fra bilder av uranprøver. Hele bildet av uranprøvene er av interesse, og dermed tilsvarer ROI, og masken, hele bildet. Et todimensjonalt bilde kan betraktes som et tredimensjonalt bilde med kun ett bilde i z-retning (retningen som definerer et volum), slik at Pyradiomics også kan benyttes for bilder som denne oppgaven tar for seg.

En annen metode for å trekke ut teksturegenskaper fra bilder er Angle Measure Technique (AMT) [7]. Algoritmen for AMT er tilpasset for bruk i Python av Professor Emeritus Knut Kvaal ved Fakultet for realfag og teknologi, Norges miljø-og biovitenskapelige universitet.

Tabell 3.2: Oversikt over modulene fra Scikit-learn som ble benyttet i programutviklingen, samt deres bruksområde [43].

Modul	Funksjon og bruksområde
sklearn.preprocessing	Preprosessering av bildeegenskaper, for eksempel standardisering av bildeegenskaper med klassen ”StandardScaler.”
sklearn.feature_selection	Egenskapsutvelgelse, for eksempel med ”VarianceThreshold” som fjerner egenskaper med varians lavere enn en gitt terskelverdi.
sklearn.model_selection	Beregning av kryssvalideringsnøyaktighet for bruk i modellseleksjon, for eksempel funksjonen ”cross_val_score”, som gir nøyaktigheten til klassifikatoren i hver validering.
sklearn.metrics	Beregning av nøyaktigheten i en modell, for eksempel funksjonen ”accuracy_score”, som gir et tall mellom 0 og 1 og angir hvor godt modellen klassifiserer.
sklearn.pipeline	Samle flere estimatorer til én felles estimator for å forenkle modelleringen, for eksempel funksjonen ”make_pipeline”.
sklearn.linear_model	Inneholder lineære modeller, slik som klassifikatoren ”LogisticRegression”.
sklearn.svm	Inneholder modeller som baserer seg på Support Vector Machines, slik som klassifikatoren ”svc” (Support Vector Classifier).
sklearn.neighbors	Inneholder modeller som baserer seg på nærmeste-nabo-modeller, slik som klassifikatoren ”KNeighborsClassifier”.
sklearn.tree	Inneholder modeller som baserer seg på beslutningstrær, slik som ”DecisionTreeClassifier”.
sklearn.ensemble	Inneholder ensemble-baserte modeller, slik som klassifikatorene ”RandomForestClassifier”, ”AdaBoostClassifier” og ”GradientBoostingClassifier”.
sklearn.decomposition	Inneholder dekomposisjonsmodeller, slik som ”PCA” (prinsipalkomponentanalyse).

4 Metode for analyse av SEM-bilder

4.1 Oppbygging av analyseprogrammet

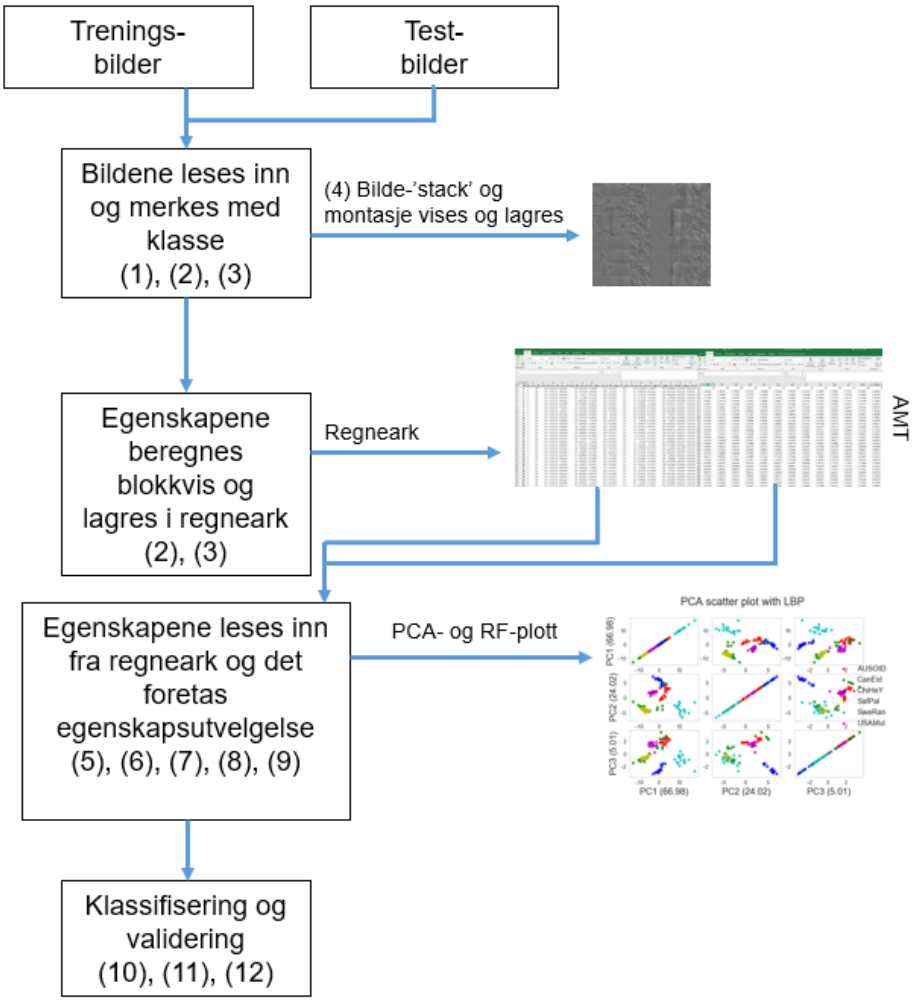
Formålet med analyseprogrammet er å bygge opp en modell som kan brukes til å klassifisere SEM-bilder av uranprøver som har forskjellige opprinnelsesland. Klassifiseringen skal gjøres på bakgrunn av bildenes egenskaper, det vil si egenskaper som fanger tekstur og fordeling av intensiteter i bildene. Programmet skal lese inn bilder i et datasett, bestående av et treningssett og et testsett, tildele disse en klasse, for så å beregne forskjellige bildeegenskaper fra dem. Bildene kan vises og lagres som en bilde-"stack" og en bilde-montasje. Videre skal programmet velge ut relevante egenskaper gjennom en såkalt egenskapsutvelgelse, som skal benyttes som grunnlag for klassifiseringen. Til slutt skal en klassifikasjonsmodell bygges opp basert på treningssettet. Denne modellen fininnstilles og optimaliseres for forskjellige klassifikasjonsalgoritmer før programmet skal klassifisere bildene i det uavhengige testsettet.

Analyseprogrammet består av flere Python-filer bestående av skript med kode som utfører forskjellige oppgaver i programmet. Navnet på disse filene, samt en kort beskrivelse av hva de inneholder er listet opp i Tabell 4.1. De fleste Python-filene består av flere funksjoner, og disse kan importeres og benyttes for å kunne kjøre andre funksjoner. Programmet kan tilpasses slik at brukeren selv definerer hvilke mapper med bilder som leses inn og hva funksjonene skal returnere.

I Figur 4.1 vises en oversiktsskisse over hvordan analyseprogrammet er bygget opp. Tallene i parentes i figuren gjenspeiler nummeret til Python-filene i Tabell 4.1. Hvert av hovedpunktene i analyseprogrammet vil bli beskrevet i detalj i de påfølgende avsnittene, med hovedvekt på beregning av egenskaper, egenskapsutvelgelse og oppbygging av klassifikasjonsmodellen.

Tabell 4.1: Tabellen viser en oversikt over Python-skriptene som er utviklet i analyseprogrammet, samt en kort beskrivelse av deres funksjon.

	Navn på Python-skript	Beskrivelse
(1)	fuel_class.py	En funksjon som tildeler bildene forskjellige klasser ut ifra filnavnene.
(2)	extract_pyrad_features.py	Leser inn bilder, tildeler klasse med (1), forbereder bildene til analyse, lager bildemasker og trekker ut egenskaper fra bildene med Pyradiomics uten og med forskjellige waveletsfiltere. Lagrer egenskapene i regneark (Microsoft Excel, v.16.0).
(3)	extract_lbp_features.py	Leser inn bilder, tildeler klasse med (1). Trekker ut egenskaper fra Local Binary Patterns og lagrer i regneark.
(4)	bilder_pre_mont_stack.py	Preprosseserer bildene, lager bilde-"stack" og bilde-montasje.
(5)	feature_selection.py	Funksjoner for egenskapsutvelgelse.
(6)	amt_plotte_spectrum.py	Plotter spektre for Angle Measure Technique-egenskaper.
(7)	RF_feat_selection.py	Plotter nøyaktigheter for forskjellige antall egenskaper med Random Forest.
(8)	pca_matriser.py	Lager plott med ulike komponenter for PCA.
(9)	get_features.py	Funksjon som leser inn alle egenskapene fra regnearkene, deler de inn blokkvis og benytter funksjonene i (5).
(10)	nestedCV_classifiers.py	Funksjoner med nøstet kryssvalidering, én funksjon for hver klassifikator.
(11)	run_nestedCV.py	Kjører de nøstede kryssvalideringene (10) på treningssettet med egenskapene som deles inn i (9) og lagrer resultatene i et regneark.
(12)	final_model_predict.py	Funksjoner med *grid search* for klassifikatorene, de endelige modellene blir trent opp på hele treningssettet og det blir gjort prediksjoner på testsettet. Resultatene lagres i et regneark.



Figur 4.1: Figuren gir et overblikk på hvordan analyseprogrammet er bygget opp. Tallene i parentes tilsvarer nummeret på skriptene som er presentert og forklart i Tabell 4.1. AMT-egenskapene blir ikke regnet ut, men leses inn fra et regneark.

4.2 Beregning av bildeegenskaper

Egenskaper til bildene ble beregnet og trukket ut. Disse egenskapene fanger variasjoner i intensitet, samt romlige endringer i intensitetsverdier i bildene. Fra programvaren Pyradiomics [54] ble seks klasser for beregning av egenskaper benyttet. Videre ble Scikit-image [48] benyttet for å hente ut egenskaper basert på Local Binary Patterns [55]. I tillegg er det hentet ut egenskaper basert på Angle Measure Technique (AMT) [7, 8], av Professor Emeritus Knut Kvaal. Tabell 4.2 presenterer de ulike metodene som ble benyttet for å beregne bildeegenskaper, hvilke moduler de tilhører, samt antall egenskaper. Egenskapene ble beregnet med skriptene (2) og (3) i Tabell 4.1. Totalt ble 649 forskjellige egenskaper trukket ut fra de autoskalerte bildene. Beskrivelse av hver metode for beregning av egenskaper er gitt i de påfølgende delkapitlene.

Tabell 4.2: Oversikt over metodene for beregning av bildeegenskaper, hvilke moduler de tilhører og antall egenskaper.

Metode	Modul	Antall egenskaper
Førsteordens statistikk	Pyradiomics	19
Gray Level Co-occurence Matrix (GLCM)	Pyradiomics	21
Gray Level Run-Length Matrix (GLRLM)	Pyradiomics	16
Gray Level Size Zone Matrix (GLSZM)	Pyradiomics	15
Neighboring Gray Tone Difference Matrix (NGTDM)	Pyradiomics	4
Gray Level Dependence Matrix (GLDM)	Pyradiomics	14
Local Binary Pattern (LBP)	Scikit-image	60
Angle Measure Technique (AMT)	Annet	500

4.2.1 Pyradiomics - parameterinnstillinger og bildetransformasjoner

Pyradiomics tillater at brukeren kan justere parametrene til egenskapsalgoritmene. For de fleste parametrene ble standardinnstillinger (”default”) benyttet. For eksempel ble det benyttet standardinnstilling for parameteren som angir antall gråtoner i bildet. I utgangspunktet har bildene 256 gråtoner, men dette antallet ble redusert til omlag 10 ved å sette parameteren ”bin width” til å være 25. Dette er standardinnstilling for Pyradiomics, og er den verdien som har vist seg å virke godt på generell basis [54].

For beregning av egenskaper i Pyradiomics, ble det i tillegg generert åtte wavelet-transformerte bilder ved å bruke høypassfiltre og lavpassfiltre på bildene. Dermed ble det generert ni sett med de samme bildeegenskapene som ble beregnet med Pyradiomics, ett for hvert av de åtte transformerte bildene og ett for det opprinnelige bildet. Ved å transformere bildet, kan spesielle egenskaper som ikke blir fanget i det opprinnelige bildet bli fremhevet. Wavelet-transformasjonen som ble benyttet er implementert i Pyradiomics [54], og beskrives senere i dette kapittelet.

De matematiske formlene tilhørende egenskapene som er trukket ut ved bruk av Pyradiomics, samt detaljerte beskrivelser av hver egenskap, finnes i dokumentasjonen til Pyradiomics (versjon 1.3.0) [54].

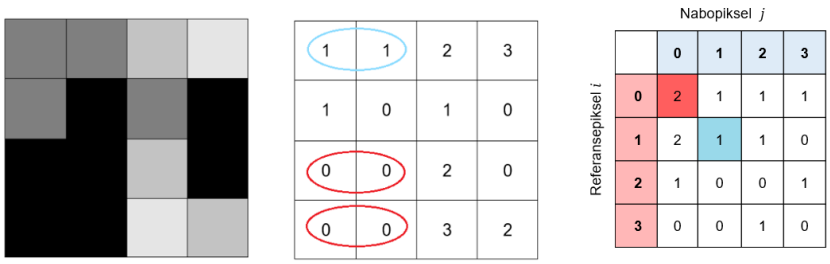
4.2.2 Førsteordens statistikk

Bildeegenskaper beregnet fra førsteordens statistikk baserer seg på fordelingen av intensitetsverdier i et bilde, se delkapittel 2.3. Disse tar ikke hensyn til den romlige fordelingen til intensitetsverdiene i bildet, men fanger intensitetsvariasjoner i bildet. Det betyr at førsteordens statistikk ikke tar hensyn til teksturen slik de andre metodene for beregning av bildeegenskaper gjør.

For et bilde $\mathbf{X}$ betegnes $\mathbf{P}(i)$ som bildets førsteordens histogram med N_g diskrete intensitetsverdier, der N_g representerer antallet søyler i histogrammet for intensitetsverdiene i bildet, se delkapittel 2.3. 19 bildeegenskaper ble trukket ut. Disse beskriver for eksempel den største (*Maximum*) og den minste (*Minimum*) intensitetsverdien i bildet. En oversikt over de resterende egenskapene som ble trukket ut er presentert i Tabell A1 i Vedlegg A.

4.2.3 Gray Level Co-occurence Matrix

Gray Level Co-occurence Matrix (heretter GLCM) brukes til å trekke ut teksturelle egenskaper fra et bilde. En GLCM-matrise er en matrise som beskriver kombinasjonen av intensitetverdiene (gråtonene) til nabopiksler i en gitt retning i bildet. GLCM defineres som $P(i, j|\theta, \alpha)$. $P(i, j|\delta, \theta)$ angir hvor mange ganger en piksel med intensitetsverdi i finnes blant naboene til et piksel med intensitetsverdi j, der i og j er adskilt med en avstand δ og en vinkel θ [49]. For beregning av GLCM-egenskaper i denne oppgaven har det blitt benyttet forskjellig verdier for avstanden δ; 1, 2, 3, 10 og 15 for å fange tekstur med forskjellige lengdeskalaer. Med $\delta = 1$ vil pikslene som betraktes som naboer være ved siden av hverandre, mens med $\delta = 2$ vil det være én piksel i mellom, osv. Pyradiomics regner ut egenskapene separat for alle vinkler (θ) og returnerer gjennomsnittsverdien til egenskapen. I et bilde med N forskjellige gråtoneverdier vil resultatet være en $N \times N$ stor GLCM-matrise.



Figur 4.2: Figuren viser et eksempelbilde (venstre) med fire forskjellige gråtoner, de tilhørende intensitesverdiene (midten) og GLCM-matrisen for eksempelbildet (høyre). I figuren i midten er det merket av to forskjellige naboskap i x-retning med nærmeste nabo. Figuren til høyre viser en 4×4 stor GLCM-matrise ($P(i, j)$) for eksempelbildet. Det er kun pikselen til høyre i x-retningen som blir betraktet.

Figur 4.2 viser et eksempel på et bilde med fire forskjellige gråtoner, tilhørende verdier og GLCM-matrisen. Det er kun sett på nærmeste nabo for én vinkel, $\delta = 1$ og $\theta = 0$.

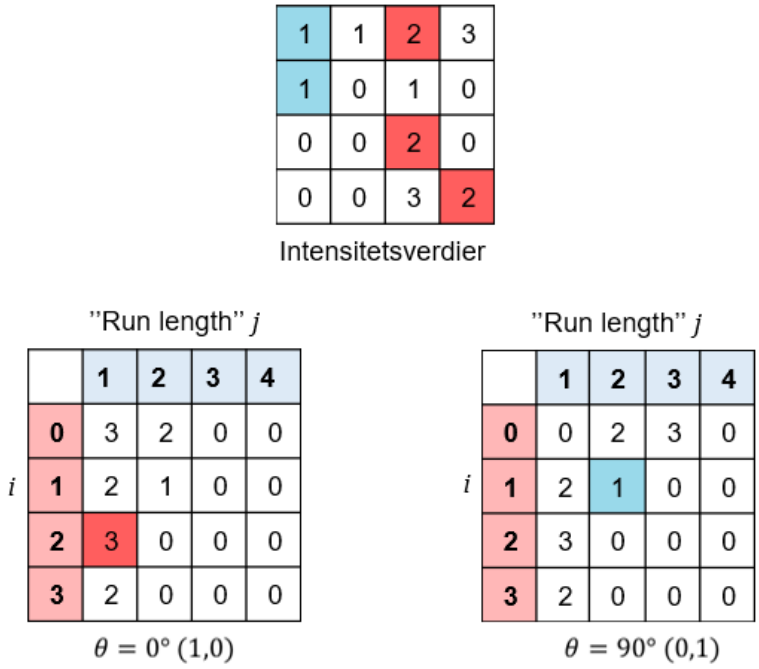
Basert på matrisen lengst til høyre i Figur 4.2 kan det regnes ut en rekke bildeegenskaper. I denne oppgaven ble 21 bildeegenskaper trukket ut. Disse beskriver blant annet korrelasjon, som viser den lineære avhengigheten av gråtoner, og tilstedeværelse av de hyppigst forekommende naboparene. En oversikt over bildeegenskapene basert på GLCM som har blitt trukket ut er presentert i Tabell A2 i Vedlegg A.

4.2.4 Gray Level Run-Length Matrix

Bildeegenskaper basert på Gray Level Run-Length Matrix (heretter GLRLM) tar i likhet med GLCM-egenskaper utgangspunkt i den romlige fordelingen av gråtoner i bildet. Teksturanalyse med bruk av GLRLM ble først introdusert av Galloway [32] i 1975. En GLRLM-matrise kvantifiserer "gray level runs", som er definert ved lengden, angitt i antall piksler, på en rekke med påfølgende piksler i bildet med samme intensitetsverdi [54]. I en GLRLM-matrise, som kan uttrykkes ved $P(i, j|\theta)$, beskriver element nummer (i, j) antall "runs" med intensitetsverdi i og lengde j som forekommer i bildet langs en vinkel θ [54].

Figur 4.3 illustrerer hvordan GLRLM-baserte egenskaper beregnes. Den øverste matrisen i figuren representerer intensitetsverdiene til et eksempelbilde med fire gråtoner (0,1,2,3). Matrisene nederst viser opptelling av "run lengths" for $\theta = 0^\circ$ (til venstre) og $\theta = 90^\circ$ (til høyre). For tilfellet med $\theta = 0^\circ$, tilsvarende ett steg i x-retning og null i y-retning (1,0), blir det talt opp hvor mange ganger hver av intensitetsverdiene (i) opptrer i bildet i en påfølgende sekvens på én, to, tre og fire. For eksempel opptrer verdien 2 ($i = 2$) tre ganger i en sekvens på én, $j = 1$, (markert med mørk rød). For tilfellet med $\theta = 90^\circ$, altså null steg i x-retning og ett i y-retning (0,1), opptrer verdien 1 ($i = 1$) én gang i en sekvens på 2, $j = 2$, (markert med mørk blå).

16 bildeegenskaper ble trukket ut basert på GLRLM. Disse kvantifiserer blant annet variansen i intensitetsverdi for hver "run" og fordelingen av korte "run lengths", hvilket indikerer finkornede strukturer i bildet. En oversikt over alle bildeegenskapene basert på GLRLM som ble trukket ut er presentert i Tabell A3 i Vedlegg A.

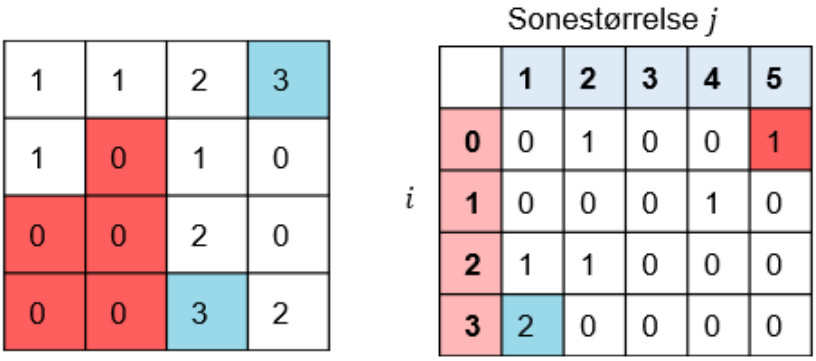


Figur 4.3: Illustrasjon som viser hvordan GLRLM-egenskaper beregnes fra et bilde. Den øverste figuren representerer intensitetsverdier til et bilde med fire verdier (0,1,2,3). De to nederste figurene viser opptelling av "run lengths" for tilfellet med $\theta = 0°$ (venstre) og $\theta = 90°$ (høyre). De mørkerøde pikslene markerer at verdien 2 ($i = 2$) opptrer 3 ganger i en sekvens på én ($j = 1$) i tilfellet med $\theta = 0°$, som vist i matrisen nederst til venstre. De mørkeblå pikslene markerer at verdien 1 ($i = 1$) opptrer én gang i en sekvens på to ($j = 2$) for tilfellet med $\theta = 90°$, som vist i matrisen nederst til høyre.

4.2.5 Gray Level Size Zone Matrix

Gray Level Size Zone Matrix (heretter GLSZM) brukes også til å trekke ut teksturelle bildeegenskaper basert på den romlige fordelingen av gråtonene i bildet. En GLSZM-matrise kvantifiserer såkalte "soner", eller *grupper*, med gråtoner i et bilde. En slik sone kan defineres som antallet sammenhengende bildeelementer som har samme intensitetsverdi. Et bildeelement anses som sammenhengende dersom naboelementet har samme gråtone. I tilfellet med todimensjonale bilder bestemmes naboskapet til bildeelementer ved 8 naboelementer [54].

I en GLSZM-matrise, som kan uttrykkes ved $P(i, j)$, vil element nummer (i, j) tilsvare antall *soner* som har intensitetsverdi lik i og med størrelse på sonen lik j. Et eksempel vises i Figur 4.4. I figuren vises det at intensitetsverdi $i = 0$ er markert i et område som er $j = 5$ piksler stort. Dette forekommer kun én gang i bildet, og får da verdien "1" i matrisen, altså; $P(0, 5) = 1$. Intensitetsverdien $i = 3$ er merket to forskjellige steder i figuren med et område med størrelse $j = 1$, og i matrisen vil verdien da bli $P(3, 1) = 2$.



1	1	2	3
1	0	1	0
0	0	2	0
0	0	3	2

i \ j	1	2	3	4	5
0	0	1	0	0	1
1	0	0	0	1	0
2	1	1	0	0	0
3	2	0	0	0	0

Figur 4.4: Eksempelbildet (til venstre) fra Figur 4.2, markert med noen utvalgte "soner" i rødt og blått. Matrisen til høyre er en GLSZM-matrise tilhørende figuren til venstre, der det vises hvor mange ganger sonestørrelsen j finnes for intensitetsverdi i.

15 bildeegenskaper basert på GLSZM ble trukket ut. Disse beskriver blant annet tilstedeværelsen av små og store soner som angir henholdsvis finkornede og grovkornede strukturer i bildet. En oversikt over bildeegenskapene som ble trukket ut basert på GLSZM er presentert i Tabell A4 i Vedlegg A.

4.2.6 Neighbouring Gray Tone Difference Matrix

Neighbouring Gray Tone Difference Matrix (heretter NGTDM) er nok en metode som brukes for å trekke ut teksturelle egenskaper fra bilder. Metoden ble introdusert av Amadasun og King [56] i 1989. En NGTDM-matrise kvantifiserer *forskjellen* mellom intensitetsverdien i et piksel og den gjennomsnittlige intensitetsverdien til dens nabopiksler i en avstand δ. NGTDM-matrisen inneholder summen av forskjellene (absoluttverdien) for hver intensitetsverdi i [54].

I et bilde med i intensitetsverdier, n_i antall piksler for hver i og N_p piksler i området av interesse (Region Of Interest, ROI), der den gjennomsnittlige intensitetsverdiene til naboene til hvert piksel er definert ved $\bar{A}_i$ defineres "neighborhood average gray level", s_i, [57] ved:

$$s_i = \begin{cases} \sum^{n_i} |i - \bar{A}_i| & n_i \neq 0 \\ 0 & n_i = 0 \end{cases} \tag{4.1}$$

Sannsynligheten for tilstedeværelse av hver gråtone p_i defineres [57] ved:

$$p_i = \frac{n_i}{N_p} \tag{4.2}$$

der n_i er antall piksler for hver i og N_p er antall piksler i bildet, det vil si $N_p = \sum^i n_i$.

NGTDM-matrisen blir seende ut eksempelvis som i Figur 4.5. Figuren til venstre representerer intensitetsverdiene til et bilde, og figuren til høyre viser den tilhørende NGTDM-matrisen for $\delta = 1$. Verdiene for s_i er beregnet ved hjelp av likning 4.1 med i = (0,1,2,3), og p_i er beregnet ved hjelp av likning 4.2 med N_p = 16. For verdier av n_i er antall piksler i referansebildet med de ulike intensitetsverdiene talt opp.

Beregningene blir, for eksempel for intensitetsverdi i = 3, seende slik ut: $n_3 = 2$, da det er to piksler i bildet med intensitetsverdi lik 3. Videre, med likning 4.2, blir det at $p_3 = \frac{2}{16} = 0,1250$. For beregning av s_3 med likning 4.1 blir det slik: $s_3 = \sum |3 - \frac{3}{3}| + |3 - \frac{4}{5}| = 2 + 2,2 = 4,2$. Disse beregnede verdiene vises i nederste rad (markert med blått) i matrisen i Figur 4.5.

1	1	2	3
1	0	1	0
0	0	2	0
0	0	3	2

Intensitetsverdier

i	n_i	p_i	s_i
0	7	0,4375	6,28
1	4	0,250	0,933
2	3	0,1875	2,58
3	2	0,1250	4,20

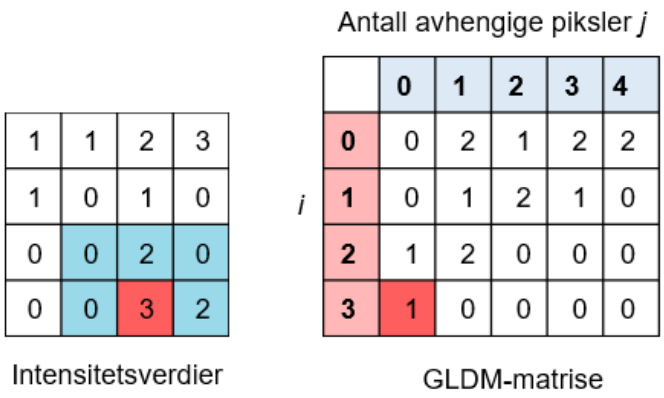
NGTDM-matrise

Figur 4.5: Figuren viser et eksempelbilde med intensitetsverdier (venstre) og en tilhørende NGTDM-matrise (høyre). s_i er beregnet med likning 4.1 med i = (0,1,2,3), og p_i er beregnet med likning 4.2 med N_p = 16. n_i viser antallet av hver intensitetsverdi i eksempelbildet til venstre.

Basert på NGTDM ble fire bildeegenskaper trukket ut. Disse beskriver blant annet kompleksiteten til bildet, som er gitt ved hvor hyppig gråtonene endrer seg, altså i hvilken grad bildet er homogent eller ikke. En oversikt over bildeegenskapene basert på NGTDM som ble trukket ut er presentert i Tabell A5 i Vedlegg A.

4.2.7 Gray Level Dependence Matrix

Denne metoden for beregning av teksturegenskaper til bilder ble introdusert av Sun og Wee [58] i 1983. Gray Level Dependence Matrix (heretter GLDM) kvantifiserer gråtone-avhengighet i et bilde. En slik avhengighet kan defineres som antallet sammenhengende piksler i en avstand δ som er avhengig av senterpikselen [54]. Et nabopiksel med intensitetsverdi lik j er definert som avhengig av senterpikselen med intensitetsverdi lik i dersom $|i - j| \leq \alpha$. I en GLDM-matrise $P(i, j)$ representerer element nummer (i, j) antall ganger en piksel med verdi i og tilhørende antall avhengige piksler j opptrer i bildet [54].



1	1	2	3
1	0	1	0
0	0	2	0
0	0	3	2

	0	1	2	3	4
0	0	2	1	2	2
1	0	1	2	1	0
2	1	2	0	0	0
3	1	0	0	0	0

Figur 4.6: Figuren viser et eksempelbilde med intensitetsverdier (venstre) og en tilhørende GLDM-matrise (høyre). Det er benyttet at $\delta = 1$ og $\alpha = 0$. Den ene pikselen med verdi lik tre (markert ved rødt i venstre matrise) har ingen piskler i en avstand lik δ som oppfyller kriteriet $|i - j| \leq 0$, og dette forekommer én gang i bildet. Dermed får elementet for $i = 3$ og antall avhengige piksler j lik null, verdien "1" (markert med rødt i matrisen til høyre).

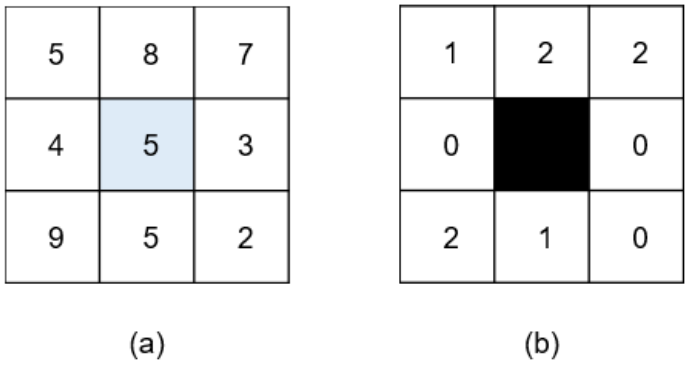
Figur 4.6 viser et eksempelbilde med intensitetsverdier (venstre) og en tilhørende GLDM-matrise (høyre) hvor det er benyttet at $\delta = 1$ og $\alpha = 0$. I matrisen til venstre er pikslene markert med blått i en avstand δ lik 1 i forhold til pikselen med verdi lik 3 (markert med rødt). Ingen av de omkringliggende pikslene er avhengige av senterpikselen, da de ikke oppfyller kriteriet $|i - j| \leq \alpha$, hvor $\alpha = 0$. Dermed får elementet (i, j) for $i = 3$ og antall avhengige piksler j lik null, verdien "1" (markert med rødt i matrisen til høyre).

Basert på GLDM ble 14 bildeegenskaper trukket ut. Disse kvantifiserer blant annet fordelingen av små områder med avhengige piksler, hvilket angir graden av homogenitet til teksturen. I Tabell A6 i Vedlegg A vises en oversikt over GLDM-egenskapene som ble trukket ut av bildene.

4.2.8 Local Binary Patterns

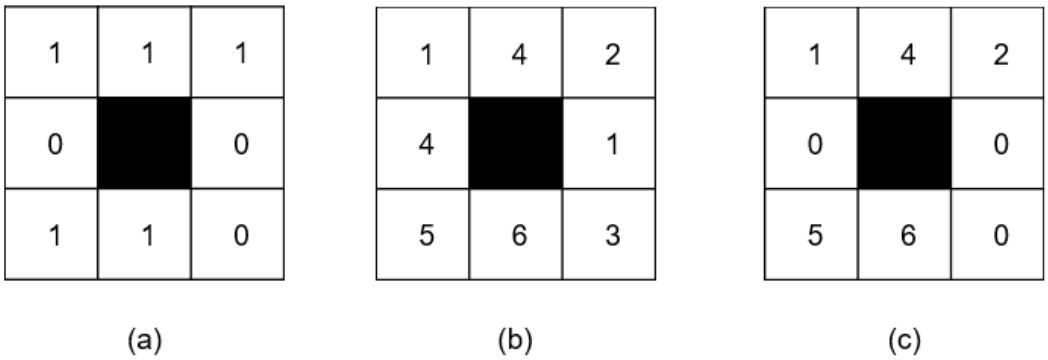
Local Binary Patterns (heretter LBP) er nok en metode som brukes til å trekke ut egenskaper fra et bilde. Den originale LBP-operatoren ble introdusert på 1990-tallet av Ojala et al. [10, 55] og bygger på at tekstur baseres på mønster og mønsterets styrke. Tekstur kan sees på som et to-dimensjonalt fenomen, der mønsteret tar for seg den romlige strukturen og kontrasten er styrken til mønsteret [55]. I Ojala et al. [10] blir LBP presentert som en to-nivå versjon av "teksturenheten" som presenteres av Wang et al. [59]. En teksturenhet representeres ved åtte elementer, og avledes fra et naboskap på 3×3 piksler i originalbildet. Et teksturbilde kan dermed karakteriseres som et strukturspekter, og de åtte elementene i teksturenheten kan ha verdiene 0, 1 eller 2, eller kun verdiene 0 og 1 slik som i to-nivå versjonen til Ojala et al. [10]. Verdiene i teksturenheten blir bestemt basert på om nabopikslene i originalbildet har lavere, lik

eller høyere intensitetsverdi enn senterpikselen. I versjonen til Wang et al. [59] kan dette gi 3^8 = 6561 forskjellige strukturelementer, og et eksempel på dette vises i Figur 4.7, der (a) viser et eksempel på pikselverdier. Pikselen i midten uthevet med lyst blått med verdi 5 (senterpikselen) har åtte nabopiksler rundt, og verdiene i teksturenheten i (b) blir angitt basert på om verdiene til nabopikslene er lavere, lik eller høyere enn referansepikselen. I to-nivå versjonen til Ojala et al. [10] der verdiene er binære (0 eller 1), kan det da bli 2^8 = 256 forskjellige teksturenheter.



Figur 4.7: Figuren viser et eksempelbilde med tilhørende teksturenhet fra Wang et al. [59]. (a) viser et eksempel der verdien 5 uthevet i lyst blått er senterpikselen og har åtte nabopiksler. (b) viser den tilhørende teksturenheten der verdiene rundt senterpikselen blir tildelt verdiene 0, 1 eller 2.

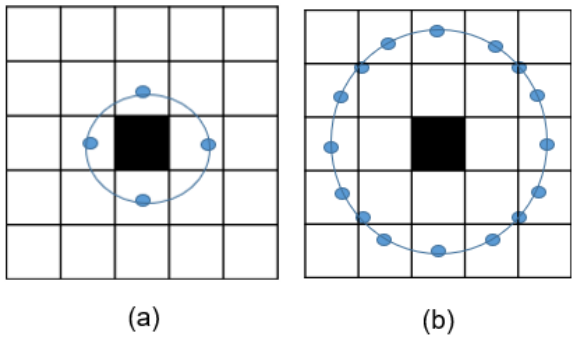
I Figur 4.8 vises et eksempel på LBP der Figur 4.7(a) er benyttet som utgangspunkt. De binære verdiene blir multiplisert med en styrke, det vil si at de får en vekting, og resultatet vises i Figur 4.8 (c).



Figur 4.8: Figuren viser et eksempel på Local Binary Patterns som gjort i Ojala et al. [10] med Figur 4.7 (a) som referanse. (a) viser de binære verdiene og er satt til 0 eller 1 avhengig om nabopikslene er mindre, større eller lik senterpikselverdien. (b) viser vektene til nabopikslene og (c) viser resultatet.

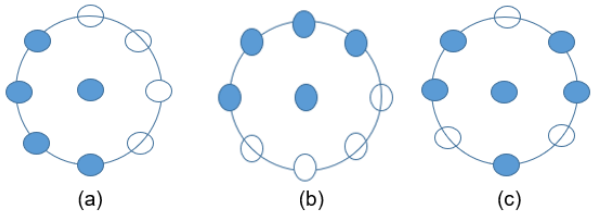
Ojala et al. [11] presenterer videre en mer robust versjon av LBP, der naboskapet ikke lenger er begrenset av en 3×3 stor pikselblokk, og avstanden (radien) og antall naboer kan bestemmes ut

ifra senterpikselen som betraktes [55]. I denne oppgaven er det blitt brukt (4,1), (8,1), (16,2) og (24,3), der første verdi tilsvarer antall nabopiksler og andre verdi tilsvarer radien. I eksempelet i Figur 4.8 har (8,1) blitt brukt. I Figur 4.9 er det vist eksempler på naboskap med henholdsvis (4,1) og (16,2).



Figur 4.9: Figuren viser to eksempler på innstillinger for naboskap som kan gjøres ved bruk av LBP. (a) viser et tilfelle med (4,1), altså 4 naboer og nærmeste nabo (avstand lik 1 piksel) ut fra senterpikselen. (b) viser et tilfelle med (16,2), altså 16 naboer og en avstand på 2 ut fra senterpikselen.

Algoritmen for LBP tar for seg hele bildet piksel for piksel, og de forskjellige mønstrene blir talt opp. Dette kan fremstilles i form av et nytt bilde eller et histogram. Hver søyle i histogrammet tilsvarer en egenskap til bildet.



Figur 4.10: Figuren viser eksempler på forskjellige typer mønstre. (a) og (b) er uniforme og rotert i forhold til hverandre, men blir likevel tolket som det samme mønsteret. (c) er ikke uniform.

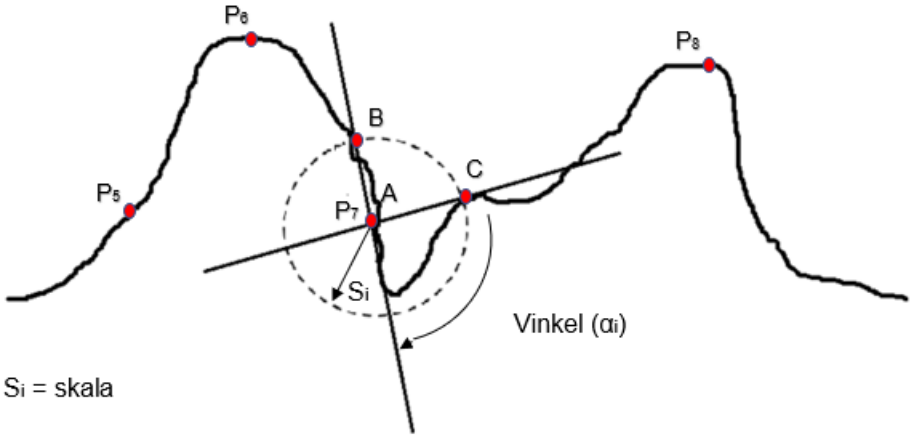
For å trekke ut egenskaper for LBP ble Scikit-image [53] benyttet. I funksjonen fra Scikit-image kan P (antall naboer), R (radius) og metode spesifiseres. P ble satt til å være 4, 8, 16 og 24, mot R som ble satt til å være 1, 1, 2 og 3. Metoden som ble brukt var ”uniform”, hvilket betyr at algoritmen kun inkluderer mønstrene som er uniforme, og mønstrene vil være rotasjonsuavhengige [53, 55]. Mønstrene vil dermed bli tolket som like selv om de roteres. I Figur 4.10 er (a) og (b) uniforme og rotert i forhold til hverandre, men vil tolkes likt. Figur 4.10 (c) viser et eksempel der mønsteret ikke er uniformt. Slike mønstre vil ikke bli talt opp og tatt med i beregningene. Bildene som analyseres i denne oppgaven har samme størrelse, men

dersom dette ikke er tilfellet, må egenskapene fra LBP korrigeres for størrelsesforskjellene på bildene for å kunne sammenlignes.

4.2.9 Angle Measure Technique

Angle Measure Technique (heretter AMT) er en tilnærming til evaluering av tekstur som baserer seg på spektrale egenskaper [6]. AMT-transformen ble introdusert for første gang i 1994 [7] av den amerikanske fysikeren Robert Andrle for å karakterisere kompleksiteten til geomorfiske kystlinjer. Den ble modifisert for uthenting av tekstur fra bilder i kjemometri av Esbensen et al. i 1996 [8]. Kvaal et al. beskriver en programvare-pakke for tekstur-og signalanalyse ved å benytte AMT [60]. Anvendelse av AMT-algoritmen i *"Nuclear forensics"*-sammenheng er ikke undersøkt i stor grad. Fongaro et al. [6] synes å være piloter hva angår anvendelse av AMT-algoritmen på det nukleære fagområdet.

AMT er en teknikk for å måle *kompleksiteten* til en måleserie [12]. AMT kan defineres som en funksjon av en skala-dimensjon S, og på denne måten genereres et helt nytt domene for karakterisering av alle slags måleserier.



Figur 4.11: Illustrasjon som viser vinkelberegning av vinkelen α_i i AMT-algoritmen. Grafen representerer intensitetslinjen for det utfoldete bildet. Punkter P_i på intensitetslinjen er punkter valgt for beregning av vinkelen α_i. En sirkel med radius S_i slås om hvert punkt, for eksempel om punktet A, og der sirkelen skjærer linjen dannes to nye punkter B og C. De tre punktene danner en trekant ABC med en vinkel BAC. Vinkelen α_i er vinkelen BAC trukket fra 180°. Tilpasset fra [6].

Det første steget i AMT er en utfoldingsprosess. Hvert to-dimensjonale bilde, med tilhørende intensitetsverdier, foldes ut til en vektor av lengde n, der n er antall piksler i bildet. Utfoldingsprosessen kan foregå i ulike retninger; for eksempel rad for rad, kolonne for kolonne eller

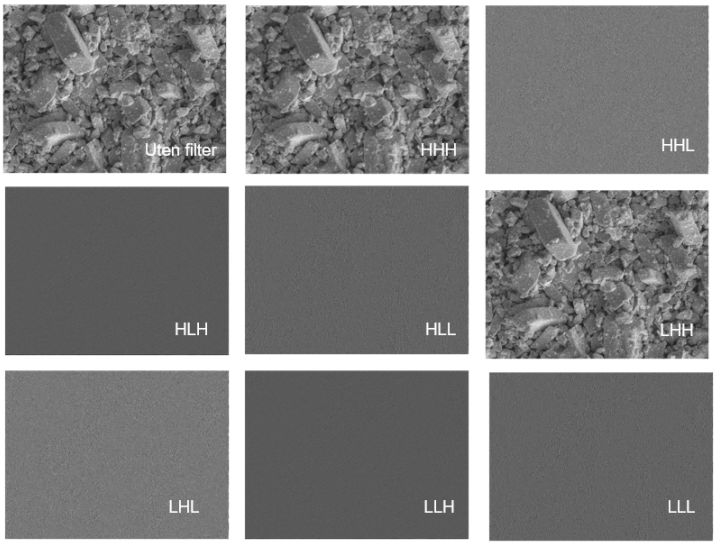
som en spiral [6]. Kucheryavski et al. undersøker også hvordan konturen av objektene som analyseres kan brukes i utfoldingsprosessen [61]. I denne oppgaven ble bildene foldet ut som en spiral. Etter utfoldingen kan vektoren presenteres som en intensitetslinje i et xy-koordinatsystem, der x-aksen og y-aksen viser henholdsvis hvert piksel i bildet og tilhørende intensitetsverdi. Fra intensitetslinjen velges det ut en delmengde bestående av et bestemt antall (m) punkter ($P_1, P_2, \ldots, P_m$), vanligvis omlag 2-5% av det totale antall piksler i bildet [6]. Figur 4.11 viser intensitetslinjen til et bilde, der A representerer ett av punktene i delmengden $P_1, P_2, \ldots, P_m$. Med radius lik S_i, der $S_i = S_1, S_2 \ldots S_n$, blir det slått en sirkel om punktet A. Den minste verdien av S tilsvarer S = 1 piksel. Der sirkelen skjærer intensitetslinjen angis to nye punkter, B og C, og vinkelen α_i beregnes som vinkelen BAC trukket fra 180° (se Figur 4.11). Prosedyren blir gjennomgått for hvert punkt i delmengden, og etter at vinkelen for hvert punkt er identifisert, beregnes *gjennomsnittet* av vinklene for hvert punkt. Gjennomsnittsverdien er definert som "Mean Angle Value". Deretter økes størrelsen på radius S_i, og prosedyren for beregning av "Mean Angle Value" gjentas for hvert punkt i delmengden. Ved å plotte "Mean Angle Value" som funksjon av S, oppstår et såkalt AMT-spekter [6], som er et mål på gjennomsnittlig endring i intensitetsverdier i bildet som funksjon av lengdeskala, og beskriver dermed *kompleksiteten* til bildet. Størrelsen på vinkelen α_i gir et mål på hvordan intensitetsverdiene endrer seg i det utfoldete bildet ved å betrakte lokal naboskapkompleksitet. Store lokale endringer i intensitetsverdi gir en stor α_i og henspeiler på kompleks tekstur, mens små lokale endringer i intensitetsverdi gir en liten α_i. Basert på AMT-spektrene til hvert bilde vil det dermed være mulig å skille bildene, eller klassene med bilder, fra hverandre, da disse beskriver kompleksiteten til teksturen i bildene.

Størrelsen på den maksimale verdien av radius S_i velges på forhånd [60], og i denne oppgaven ble maksimalverdi for S_i valgt til å være 500. Dette genererer 500 bildeegenskaper for hvert bilde. Videre kan antall punkter i delmengden velges. I denne oppgaven ble det brukt både 500 punkter og 2000 punkter, som utgjør henholdsvis 0,04% og 0,2% av totalt antall piksler i bildene. Antall punkter ble valgt som et kompromiss mellom tidsbruk for algoritmen og en forventning om liten gevinst ved å øke antallet punkter ytterligere. Esbensen et al. [8] benytter 500 punkter av totalt 65536 piksler og utgjør 0,8%, mens Fongaro et al. [6] benytter 4000 punkter i analyse av SEM-bilder bestående av 200000 piksler, som utgjør 2%. For sistenevnte ble det benyttet et annet og hurtigere programverktøy enn det som benyttes i denne oppgaven.

For å kunne bruke resultatene fra AMT-algoritmen på en fornuftig måte, er det spesielt viktig å foreta en variabelseleksjon. Dette medfører utvelgelse av de egenskapene, eller verdiene av lengdeskalaen S, som på best måte forklarer variasjonen mellom bildene fra de forskjellige klassene. Videre ble det foretatt en manuell utvelgelse av den delen av AMT-spekteret (med 2000 punkter i delmengden) som syntes å skille klassene best. Her ble det tatt hensyn til at de minste verdiene av S gjerne er forbundet med støy, og dermed inngikk ikke disse i den utvalgte delen av spekteret. Dette utvalgte datasettet omtales som "AMT2000_redusert" i oppgaven.

4.2.10 Wavelettransformasjon

En metode som kan brukes til å generere flere bildeegenskaper, går ut på å transformere bilder med et filter. Bildeegenskaper trekkes så ut fra de transformerte bildene. Ved bruk av wavelettransformasjoner blir bildene representert ved flere oppløsninger, og egenskaper i bildene som ellers ikke ville blitt oppdaget kan fanges opp [62]. Wavelttransformasjoner tar for seg både bildenes romlige og frekvensielle karakteristikk [62]. I denne oppgaven transformeres bildene med en *Coiflet1*-wavelet [63, 64] anvendt med høypassfilter og lavpassfilter i tre dimensjoner. Metodene for wavelettransformasjon er implementert i Pyradiomics [54] og tar utgangspunkt i tredimensjonale bilder. Dermed genereres transformerte bilder basert på alle mulige kombinasjoner av å anvende enten høypassfilter (H) eller lavpassfilter (L) i tre dimensjoner: LLL, LHL, LHH, LLH, HHH, HLH, HLL, HHL. Da bildene som benyttes i denne oppgaven er todimensjonale og dermed kun har én pikselverdi i den tredje dimensjonen, vil det kun være de to siste filtrene i transformasjonen som utgjør det transformerte bildet. Transformasjonene LLL og HLL vil derfor resultere i like bilder. Det samme gjelder for transformasjonsparene LHL-HHL, HHH-LHH og LLH-HLH. Til tross for at det genereres like par med transformerte bilder, ble alle transformasjonene benyttet. Årsaken til dette er begrensninger for tilpasning i Pyradiomics. Fra de transformerte bildene ble metodene for beregning av bildeegenskaper i Pyradiomics



Figur 4.12: Figuren viser et eksempel på et uranbilde (markert "uten filter"), samt åtte wavelettransformerte versjoner av bildet, som er generert med Pyradiomics [54]. Bildeparene HHL-LHL, HLH-LLH, HLL-LLL og HHH-LHH gir like transformerte bilder. Dette kommer av at Pyradiomics tar utgangspunkt i tredimensjonale bilder, mens bildene som er transformert her er todimensjonale bilder.

(Førsteordens statistikk, GLCM, GLRLM, GLSZM, NGTDM, GLDM) gjentatt, slik at ni sett med de samme bildeegenskapene ble beregnet; bildeegenskaper fra åtte transformerte bilder, samt fra det opprinnelige bildet.

Figur 4.12 viser et eksempel på et opprinnelig uranbilde (markert ”uten filter”), samt åtte wavelettransformerte versjoner av bildet. Av figuren kommer det til syne at bildeparene HHL-LHL, HLH-LLH, HLL-LLL og HHH-LHH gir like bilder.

4.2.11 Inndeling av egenskapsblokker

Alle egenskapene som ble beregnet og trukket ut fra bildene ble delt inn i tolv *egenskapsblokker*. Disse blokkene blir omtalt ved bestemte navn videre og er presentert i Tabell 4.3 sammen med antall egenskaper for hver egenskapsblokk.

Tabell 4.3: Oversikt over egenskapsblokkene som har blitt analysert i denne oppgaven. Tabellen viser også antall egenskaper for hver av blokkene.

Egenskapsblokk	Antall egenskaper	Merknad
Førsteorden	171	Inkludert egenskaper beregnet fra wavelet-transformerte bilder.
GLCM	945	Inkludert egenskaper beregnet fra wavelet-transformerte bilder.
GLRLM	144	Inkludert egenskaper beregnet fra wavelet-transformerte bilder.
GLSZM	135	Inkludert egenskaper beregnet fra wavelet-transformerte bilder.
NGTDM	36	Inkludert egenskaper beregnet fra wavelet-transformerte bilder.
GLDM	126	Inkludert egenskaper beregnet fra wavelet-transformerte bilder.
Pyradiomics - alle	1557	Alle egenskaper fra Pyradiomics, inkludert egenskaper beregnet fra wavelet-transformerte bilder.
LBP	60	Bare LBP egenskaper fra originalbildene.
Pyradiomics og LBP	1617	Alle egenskaper fra Pyradiomics, inkludert egenskaper beregnet fra wavelet-transformerte bilder, samt LBP.
AMT500	500	Bare AMT-spekter fra originalbildene.
AMT2000	500	Bare AMT-spekter fra originalbildene.
AMT2000 - redusert	230	Delmengde av AMT2000.

4.2.12 Standardisering av bildeegenskapene

Skalering av bildeegenskaper brukt i en treningsmodell, er et preprosesserings-steg for en rekke maskinlærings-algoritmer, slik som logistisk regresjon, K-nærmeste nabo og SVM [43]. Bildeegenskapene i denne oppgaven ble standardisert ved å benytte en preprosesserings-klasse (*StandardScaler*) som er implementert i Scikit-learn [43]. Standardisering innebærer at egenskapskolonnene sentreres til et gjennomsnitt lik null og et tilhørende standardavvik lik 1. Egenskapene oppnår altså kvalitetene til en standardnormalfordeling [43]. Standardiseringen av en egenskap $x^{(i)}$ kan uttrykkes ved [34]:

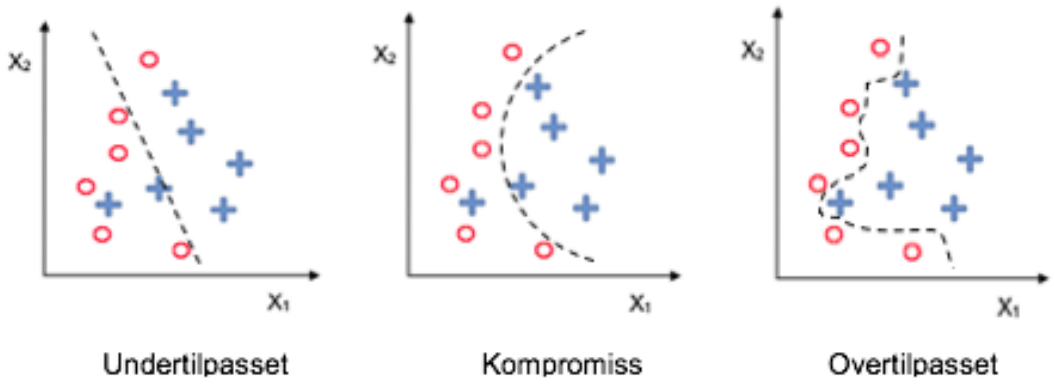
$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x} \tag{4.3}$$

der $x_{std}^{(i)}$ er den standardiserte egenskapen, μ_x er gjennomsnittet for en gitt egenskapskolonne og σ_x er det tilhørende standardavviket.

Standardisering kompenserer for at egenskaper kan ha forskjellige størrelser og enheter, slik at egenskaper med store verdier ikke nødvendigvis dominerer i modellen. Dessuten kan standardisering bidra til informasjon om eventuelle "uteliggere" i datasettet [34].

4.3 Egenskapsutvelgelse og komprimering av datasettet

Basert på egenskapene som ble trukket ut av bildene, ble en treningsmodell bygget opp. Målet er å predikere klassene til et uavhengig testsett basert på treningsmodellen. En utfordring knyttet til oppbygning av treningsmodellen, er at modellen blir såpass godt tilpasset observasjonene i treningsdataene, at den ikke vil kunne predikere godt på et uavhengig testsett [34]. Dersom prediksjonens nøyaktighet spriker mye mellom treningsdataene og testdataene, tyder det på at modellen er *overtilpasset* [34]. Å unngå overtilpasning er svært viktig når en treningsmodell skal bygges opp. Det motsatte tilfellet er når modellen er undertilpasset. En for enkel modell klarer ikke fange opp variasjonene i treningssettet, og vil klassifisere feil på testsettet. Det er derimot ønskelig at modellen har en god generaliseringsytelse og derigjennom en lav *generaliseringsfeil*, slik at modellen kan predikere godt også på et uavhengig testsett [34]. For mange variabler kan også føre til støy slik at modellene ikke klarer å predikere. Det er med andre ord om å gjøre å utvikle en modell som er kompleks nok for å fange informasjonen i egenskapene, men samtidig ikke *for* kompleks. Figur 4.13 illustrerer et eksempel på undertilpasning av en modell, overtilpasning av en modell og et kompromiss mellom disse. I figuren er det brukt et eksempel med kun to klasser og er ment som et illustrerende eksempel.



Figur 4.13: Illustrasjon som viser forskjellen på undertilpasning (til venstre), overtilpasning (til høyre) og et kompromiss mellom de to (midten). Figurene viser et tilfelle med kun to klasser; de røde ringene representerer den ene klassen og de blå korsene representerer den andre klassen. Tilpasset fra [34].

Det finnes flere metoder som kan brukes for å redusere generaliseringsfeilen. En av metodene er å bruke flere prøver for å opparbeide et større treningssett, slik at modellen kan trenes opp på et datasett som er mer representativt for den sanne virkeligheten. Ettersom muligheten for å innhente flere data for analysene i denne oppgaven ikke var tilstede, ble andre metoder benyttet. Disse metodene representerer en *egenskapsutvelgelse*. Egenskapsutvelgelsen ble gjennomført ved å bruke skript nummer (5) i Tabell 4.1.

4.3.1 Fjerne egenskaper med lav varians

En metode for å fjerne de egenskapene som har mindre betydning for modellen, og dermed unngå overtilpasning av modellen, går ut på å fjerne variabler med ingen eller lav varians. I denne oppgaven ble en klasse, *VarianceThreshold* [43], fra Scikit-learn benyttet. Formålet var å fjerne de variablene med *lav* varians, og det ble derfor implementert en terskelverdi for variansen. Bildeegenskaper med lavere varians enn terskelverdien ble fjernet. Denne metoden ble brukt som et innledende steg i egenskapsutvelgelsen før andre metoder ble testet ut og satt opp mot hverandre. Variansen er gitt ved [43]:

$$Var(X) = p(1-p), \tag{4.4}$$

der p er sannsynligheten for at verdien til variablene er like [52]. Terskelverdien ble satt til å være $(0,8) \times (1-0,8))$, hvilket medfører at variabler som har samme verdi i mer enn 80% av tilfellene fjernes.

4.3.2 Viktighet av egenskaper med Random Forest

En metode som ble brukt for å velge ut de mest relevante bildeegenskapene, bygger på Random Forest-klassifikatoren (se avsnitt 2.4.5 for beskrivelse av denne). Med Random Forest kan viktigheten av hver bildeegenskap måles ved den gjennomsnittlige avtakingen av "urenhet" for beslutningstrærne [34]. Algoritmen velger altså ut de egenskapene som i gjennomsnitt gir grunnlag for riktig klassifisering flest ganger [34].

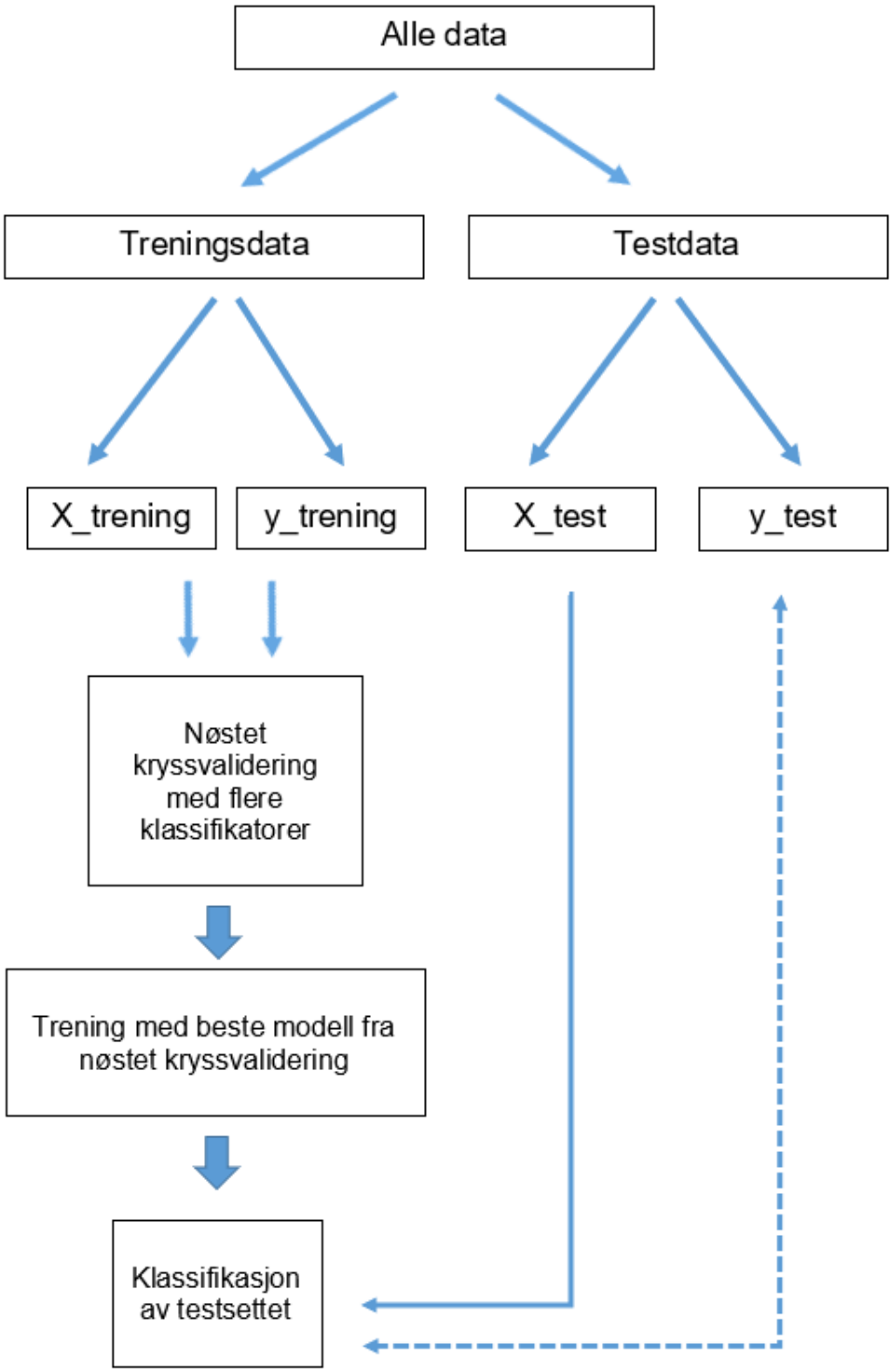
Ved å tilpasse dataene til Random Forest-klassifikatoren i Scikit-learn [43] og bruke attributtet *feature_importance*, velger algoritmen ut de n egenskapene som er mest viktige for klassifiseringsmodellen. I denne oppgaven ble verdien av n optimalisert for hver av egenskapsblokkene. Gjennom en nøstet kryssvalidering (se delkapittel 4.5.2) med Random Forest-algoritmen ble det iterert gjennom en liste med femten forskjellige n-verdier (n = 1,2,3,...,15) for hver av egenskapsblokkene. For hver verdi av n ble en gjennomsnittlig nøyaktighet og et tilhørende standardavvik fra den nøstede kryssvalideringen returnert. De gjennomsnittlige nøyaktighetene ble plottet som funksjon av antall egenskaper for hver egenskapsblokk (skript nummer (7) i Tabell 4.1), og valget for de optimale verdiene av n ble tatt på bakgrunn av disse plottene. Først og fremst ble nøyaktigheten betraktet, men også hvordan nøyaktigheten endret seg med ulike verdier av n, samt standardavviket til nøyaktigheten. De verdiene av n som ble valgt ut hadde høy og *stabil* nøyaktighet og et lite standardavvik. De optimaliserte verdiene av n for hver enkelt egenskapsblokk ble brukt i videreutviklingen av modellen.

4.3.3 Prinsipalkomponentanalyse

Gjennom prinsipalkomponentanalysen ble datasettet komprimert, slik at variablene, eller egenskapene, ble representert ved prinsipalkomponeter i stedet for de originale variablene. Antall prinsipalkomponenter som skulle inkluderes i modellen ble bestemt for hver egenskapsblokk. For å visualisere betydningen av prinsipalkomponentene, ble forklart varians plottet som funksjon av prinsipalkomponent for hver egenskapsblokk. Det ble også laget spredningsplott der to og to prinsipalkomponenter ble plottet mot hverandre, for å visualisere hvordan komponentene skilte klassene for de forskjellige egenskapsblokkene. PCA-plottene ble laget med skript nummer (8) i Tabell 4.1.

4.4 Fremgangsmåte for klassifiseringsmodellen

I Figur 4.14 er det skissert en overordnet fremgangsmåte for prosedyren for utvikling av klassifikasjonsmodellen. Etter at bildene er delt inn i et treningssett og et testsett, blir de forskjellige egenskapene, som beskrevet i kapittel 4.2, trukket ut for hvert bilde og lagret i et regneark (Microsoft Excel, v.16.0). De forskjellige egenskapene organiseres i forskjellige egenskapsblokker etter hvilken metode som er benyttet (skript nummer (9) i Tabell 4.1). Etter at alle egenskapene er lagret, gjøres det utvelgelser av egenskapene for hver egenskapsblokk med den hensikt å redusere mulighetene for overtilpasning av modellen, se kapittel 4.3. X_trening og X_test representerer matriser bestående av egenskaper til bildene for henholdsvis treningssettet og testsettet. y_trening og y_test representerer vektorer bestående av responsvariablene, det vil si klassene til bildene, for henholdsvis treningssettet og testsettet. y-vektorene angir klassene og består av tall mellom 0 og 5 korresponderende med klassene som vist i Tabell 3.1. Det blir så foretatt en nøstet kryssvalidering med forskjellige klassifikator-algoritmer som returnerer klassifiseringsmodellens nøyaktigheter og standardavvik for alle egenskapsblokkene (skript nummer (10) i Tabell 4.1). Den nøstede kryssvalideringen gir en indikasjon på hva som kan forventes når modellen skal klassifisere prøver fra et uavhengig testsett. For å undersøke om forventningen stemmer overens med virkeligheten blir nye modeller trent opp med et nytt *grid search* og en k-fold kryssvalidering, der *hele* treningssettet brukes til opptreningen (skript nummer (11) i Tabell 4.1). Testsettet, representert med X_test og y_test, blir klassifisert med den nye modellen og nøyaktighet for hvor godt den endelige modellen klarer å klassifisere testsettet blir beregnet (skript nummer (12) i Tabell 4.1). Klassen som hvert enkelt bilde ble tildelt blir også lagret (skript nummer (12)).



Figur 4.14: Figuren viser en oversikt over fremgangsmåten for utvikling av klassifikasjonsmodellen. Alle dataene deles inn i et treningssett og et testsett. Egenskaper trekkes ut fra bildene, organiseres i egenskapsblokker, og det blir foretatt forskjellige metoder for egenskapsutvelgelse. X_trening og X_test representerer matriser med egenskaper, og y_trening og y_test representerer vektorer med responsvariable, det vil si klassene til bildene. Egenskapsblokkene blir matet inn i en nøstet kryssvalidering for å finne optimale modellen. Den beste modellen blir trent opp på nytt, hvorpå testsettet blir klassifisert med den nye modellen.

4.5 Valg av modell

Hovedformålet i analysen er å utvikle en klassifiseringsmodell som kan predikere hvor en ukjent prøve kommer fra. Det er med andre ord ønskelig å teste ut forskjellige modeller for å danne et bilde på hva som kan forventes hva angår nøyaktighet i prediksjonen når modellen skal klassifisere en prøve fra et uavhengig testsett. Det foretas altså det som kan kalles en *modellseleksjon*, og dette gjøres ved å justere modellenes hyperparametre gjennom en nøstet kryssvalidering. Dette blir beskrevet i de påfølgende delkapitlene.

4.5.1 Justering av hyperparametrene

Et av stegene i *modellselekteringen* går ut på å justere, eller *fininnstille*, hyperparametrene i modellen. Hyperparametere er parameterne tilhørende en gitt læringsalgoritme [34]. Dette innebærer at hyperparametrene i modellen justeres og endres på og kombineres på forskjellige måter for å oppnå den modellen som gir den høyeste nøyaktigheten. Måten modellselekteringen ble gjort på i denne oppgaven, var å benytte en modul som er implementert i Sckit-learn; *GridSearchCV* [47].

Grid search er et nyttig verktøy for å optimalisere hyperparametrene og derigjennom en modell. I et *grid search* blir alle mulige kombinasjoner av de angitte hyperparameterne gjennomgått. Modellen evaluerer hver kombinasjon gjennom en kryssvalidering, og velger de kombinasjonen som gir størst nøyaktighet, det vil si de som fører til at flest mulig prøver blir klassifisert riktig. Kryssvalidering forklares grundigere i neste delkapittel.

Grid search-algoritmen bruker kryssvalidering og itererer gjennom et *paramgrid* med en tilhørende *paramrange*; to lister som henholdsvis angir hvilke hyperparametre som er tilgjengelige og et spenn for disse. Algoritmen kombinerer de forskjellige hyperparametrene med et bestemt spenn på alle mulige måter og søker etter den optimale kombinasjonen [34].

Listene *paramgrid* og *paramrange* velges på bakgrunn av hvilken klassifikator som benyttes. De to listene er altså klassifikator-spesifikke. Hvilke parametre og spenn som ble benyttet for de forskjellige klassifikatorene beskrives i delkapittel 4.6. I søken etter den beste modellen blir *grid search*-algoritmen kombinert med nok en kryssvalidering i det som kalles en *nøstet* kryssvalidering.

4.5.2 Kryssvalidering og nøstet kryssvalidering

Kryssvalidering er en metode som benyttes for å selektere den mest optimale modell eller algoritme. I en k-veis ("k-fold") kryssvalidering deles treningssettet opp i k mindre sett, eller *folder*. For hver av de k foldene blir en modell trent opp ved å bruke $(k-1)$ folder av delsettene, og modellen blir så validert på den resterende folden med data, og dette gjentas for hver k. I en klassifiseringsmodell vil valideringssettet representere et testsett som brukes for å beregne presisjonen for klassifiseringen.

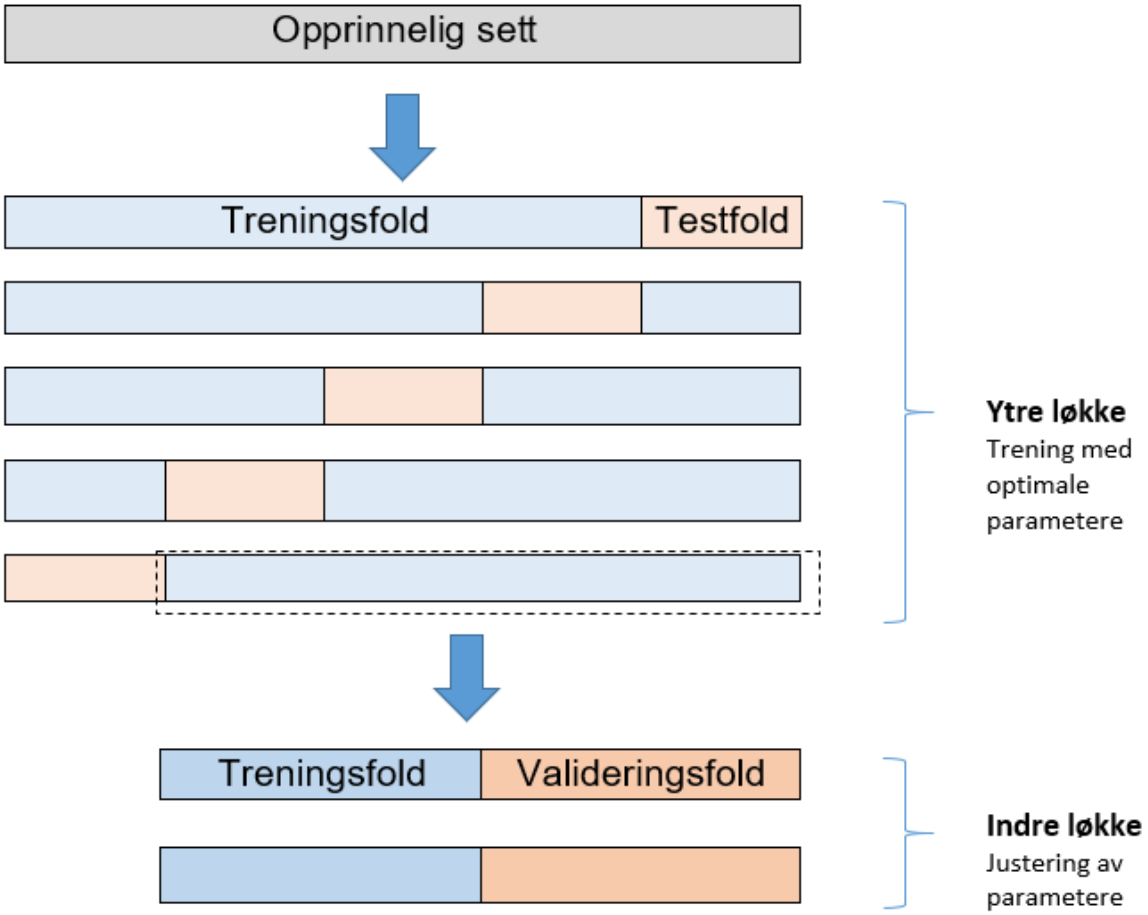
Nøstet kryssvalidering (nested cross validation) er mer robust enn ordinær kryssvalidering og gir et godt estimat på hva som kan forventes ved å benytte modellen på usette data [34]. Varma og Simon [65] konkluderer i sin rapport om *bias* i estimering av feil at den sanne feilen til estimatet er omtrent "unbiased" i forhold til testsettet ved å benytte nøstet kryssvalidering. Med "unbiased" menes det at forskjellen mellom en estimators forventede verdi og den sanne verdien til parameteren som estimeres er minimal. I denne oppgaven ble nøstet kryssvalidering gjennomført for hver av de benyttede klassifikatorene. For hver kryssvalidering ble et *grid search* implementer, se nærmere beskrivelse i delkapittel 4.5.1.

Nøstet kryssvalidering kan enten ta for seg *hele* datasettet eller *kun* treningssettet. I denne oppgaven ble kun treningssettet benyttet, mens testsettet ble holdt tilbake som et uavhengig sett for prediksjon avslutningsvis i klassifiseringsmodellen. Algoritmen deler opp datasettet i en treningsfold og en testfold n ganger i en såkalt *ytre løkke* med k-fold kryssvalidering. I en *indre løkke* deles treningsfolden opp i en ny treningsfold og en valideringsfold m ganger. Her justeres hyperparametrene og det velges en modell basert på den nye treningsfolden og k-fold kryssvalidering [34]. Modellen velges basert på de mest optimale parametrene (se avsnitt 4.5.1). Når modellen er valgt, brukes testfolden til å evaluere ytelsen til modellen [34]. Ytelsen, eller *nøyaktigheten* (accuracy), til modellen er den gjennomsnittlige nøyaktigheten til hver fold i kryssvalideringen med et tilhørende standardavvik.

Den ytre løkken deles, som tidligere nevnt, opp i trenings- og testsett n ganger. Dette gjøres slik at enhver prøve i datasettet er en del av testfolden i løpet av analysen. På denne måten unngås en overtilpasning av modellen til én bestemt del av datasettet. En nøstet kryssvalidering omtales som en $n \times m$-kryssvalidering, der n og m henviser til antall folder i henholdvis den ytre og den indre løkken.

En skjematisk fremstilling av virkemåten for algoritmen bak nøstet kryssvalidering vises i Figur 4.15. I dette eksempelet vises en 5x2-kryssvalidering. I oppgaven ble en 5x5-kryssvalidering benyttet, slik at det var fem kryssvalideringer i både den indre og den ytre løkken. Valget av antall kryssvalideringer er basert på størrelsen på datasettet og tidsbruk. Merk spesielt at det som i figuren omtales som "Opprinnelige sett" er treningssettet med SEM-bilder som benyttes i

oppgaven. Det uavhengige testsettet holdes utenfor og brukes først når de endelige modellen skal klassifisere ukjente prøver.



Figur 4.15: Skjematisk fremstilling av virkemåten for nøstet kryssvalidering med fem og to kryssvalideringer i henholdsvis ytre og indre løkke (5x2-kryssvalidering). Merk at det som omtales som "opprinnelig sett" er treningssettet med bilder. Figuren er tilpasset fra [34].

4.6 Klassifiseringsmetoder

Under oppbygningen av modellen i denne oppgaven ble en rekke klassifikatorer implementert og testet. En klassifikator ble brukt til å klassifisere, eller *predikere*, en ukjent prøve i form av bildet av en prøve. For å finne ut hvilken eller hvilke klassifikatorer kunne predikere flest prøver riktig, ble flere klassifikatorer testet og satt opp mot hverandre. I kapittel 2.4.6 beskrives det teoretiske grunnlaget for klassifikatorene.

4.6.1 Hyperparameterinnstilliger

Samtlige klassifikator-algoritmer er hentet fra Scikit-learn [43]. For hver klassifikator-algoritme er det en rekke innstillinger som kan justeres og endres på for å oppnå den mest optimale modellen. For noen av disse hyperparametrene ble standardinnstillinger fra Scikit-learn benyttet, mens andre ble justert etter eksempler fra Raschka og Mirjalili [34]. Enkelte av parametrene ble optimalisert via et *grid search* med innstillinger for *paramrange* og *paramgrid* tilpasset fra Raschka og Mirjalili [34]. *Paramrange* og *paramgrid* representerer lister med henholdsvis hyperparametre og et spenn for disse, som beskrevet i kapittel 4.5.1. I Tabell 4.4 presenteres innstillinger for hver av klassifikator-algoritme, samt hvilke hyperparametre som ble optimalisert via et *grid search*. Detaljer for parameterinnstillinger og betydningen av disse finnes i dokumentasjonen til Scikit-learn [52]. I de påfølgende delkapitlene beskrives blant annet parametrene som er presentert i Tabell 4.4.

Tabell 4.4: Oversikt over parameterinnstillinger for klassifikator-algoritmene. De to midterste kolonnene representerer innstillinger for optimalisering av hyperparametre via et grid search, mens kolonnen lengst til høyre viser eksempler på andre innstillinger.

	Optimalisert med *Grid Search*		
Klassifiserings-algoritme	Hyperparameter (*paramgrid*)	Spenn til hyperparameteren (*paramrange*)	Andre parameter-innstillinger
Logistisk regresjon	c	[0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]	max_iter = 100 tol = 10^{-4}
Support Vector Machines (SVM)	c, gamma (γ)	[0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]	max_iter = -1 tol = 10^{-3}
K-nærmeste nabo	n_neighbors	[3, 5, 7, 9, 11]	Avstand: euklidisk
Beslutningstre (Decision tree)	max_depth	[1, 2, 3, 4, 5, 6, 7, None]	n = 500 criterion = gini
Random Forest (RFC)	max_depth	[1, 2, 3, 4, 5, 6, 7, None]	n_estimators = 50 n_jobs = -1 criterion = gini
AdaBoost	learning_rate	[1, 2, 3 , 4]	n_estimators = 50
Gradient Boosting Classifier (GBC)	max_depth	[1, 2, 3, 4, 5, 6, 7, None]	n_estimators = 100 learning_rate = 0,1

Logistisk regresjon

I algoritmen for logistisk regresjon ble den inverse regulariseringsstyrken c optimalisert via et *grid search* hvor alle verdiene i spennet til parameteren ble testet (se Tabell 4.4). Parameteren c er direkte koblet til regulariseringsparameteren λ (se delkapittel 2.4.2), som er den inverse til c. Parameteren *max_iter* angir antall iterasjoner i algoritmen før det blir konvergens, og er satt til 100. Parameteren *tol* angir toleransen for stopp-kriterium for algoritmen og er satt til 10^{-4}. Videre ble det benyttet to forskjellige løsere (”solver”) med hver sin regulariseringsparameter (L1 og L2); for L1-regularisering ble løseren ”saga” benyttet og for L2-regularisering ble løseren ”newton-cg” benyttet.

Support Vector Machines

Parameteren *max_iter* angir antall iterasjoner i algoritmen før det blir konvergens, og er satt til -1, hvilket betyr at det er en endeløs grense. Parameteren *tol* angir toleransen for stopp-kriterium for algoritmen og er satt til 10^{-3}. Videre ble det benyttet to forskjellige Kernel-typer for algoritmen; lineær og ikke-lineær (”radial basis function”, rbf). For den lineære kernelen ble straffeparameteren til feilleddet c optimalisert og for den ikke-lineære kernelen ble c og kernelkoeffisienten *gamma* optimalisert, alle via et *grid search* som vist i Tabell 4.4.

Beslutningstre og Random Forest

Både for beslutningstre (Decision Tree) og Random Forest ble den maksimale dybden på trærne i algoritmen, *max_depth*, optimalisert via et *grid search* som vist i Tabell 4.4. Et av elementene i spennet til hyperparameteren er ”None” og henviser til at det ikke settes noen grense for dybden på trærne. Både for beslutningstre og Random Forest ble splitte-kriteriet angitt ved *gini*, også kalt *gini impurity*, som kan forstås som et kriterium for å *minimere* sannsynligheten for feilklassifisering [34]. Antall trær, n, ble satt til 500. Valget av antall trær er tatt på bakgrunn av et eksempel fra Raschka og Mirjalili [34], der 500 trær benyttes. Det er anslått at dette antallet trær gir en passende balanse mellom beregningstid for algoritmen og en forventning om at å øke antall trær ikke vil gi en betraktelig gevinst i presisjon for modellen. For Random Forest ble det angitt at datamaskinen skulle benytte alle tilgjengelige prosessorer (CPUer) parallelt i beregningene ved å sette parameteren n_jobs = -1.

K-nærmeste nabo

For K-nærmeste nabo-algoritmen ble antall naboer, $n_{neighbors}$, optimalisert via et *grid search* som vist i Tabell 4.4. Modellen ble altså testet med 3,5,7,9 og 11 nærmeste naboer for å finne det optimale antallet naboer. Avstandsberegningene i algoritmen er basert på euklidisk avstand.

AdaBoost

I AdaBoost-algoritmen benyttes beslutningstrær som utgangspunkt for ensemblet med klassifikatorer (*base_estimator*). Videre er antall estimatorer i "boostingen", *n_estimators*, valgt til å være 50. Via et *grid search* ble parameteren *learning_rate* optimalisert. Denne parameteren angir raten for hvor mye bidraget fra hver klassifikator blir krympet i algoritmen og fungerer dermed som et verktøy mot overtilpasning av modellen.

Gradient Boosting Classifier

For Gradient Boosting Classifier ble den maksimale dybden på trærne, *max_depth*, optimalisert via et *grid search* som vist i Tabell 4.4. Verdien til *learning_rate* ble sattt il 0,1 og *n_estimators* ble satt til 100 (se avsnittet over for en kort beskrivelse av disse).

4.7 Trening av endelig modell og klassifisering av testdata

Modellene som ble optimalisert og trent opp gjennom den nøstede kryssvalideringen gir et godt estimat på hvilken nøyaktighet som kan forventes ved å klassifisere prøver fra et uavhengig testsett. Resultatene fra den nøstede kryssvalideringen gir dermed en indikasjon på hvilke blokker med egenskaper, hvilke metoder for utvelgelse av egenskaper og hvilke klassifikatorer som synes å gi en modell som har høyest nøyaktighet i klassifiseringen. Siden nøyaktighetene fra den nøstede kryssvalideringen er gjennomsnittverdier, blir også de tilhørende standardavvikene tatt i betraktning. Antall egenskaper blir også tatt i betraktning, da en enklere modell er å foretrekke.

Det neste steget er å undersøke om forventningen stemmer overens med klassifisering av ukjente prøver fra testsettet. For å undersøke dette blir nye modeller trent opp. Det gjennomføres en k-fold kryssvalidering og en ny *grid search*. Denne gangen er ikke kryssvalideringen *nøstet*, det vil si at hele treningssettet brukes i opptreningen av modellen. Treningssettet blir altså ikke delt opp i et internt treningssett og testsett slik det blir gjort i den nøstede kryssvalideringen som vist

i Figur 4.15. De endelige modellene blir returnert med de optimale hyperparametrene.

Det er først nå det utelatte testsettet blir brukt i modellen som er trent på trenignssettet. Bildegenskapene som ble brukt til å trene opp den endelige modellen blir trukket ut fra bildene i testsettet, og modellen bruker disse til å gjøre en prediksjon ved å klassifisere alle bildene i testsettet. Modellen returnerer en nøyaktighet for treningssettet, en nøyaktighet for testsettet og den returnerer modellens prediksjoner hva angår klassetilhørighet av bildene.

5 Resultater

I prosessen for oppbygging av analyseprogrammet for klassifisering av SEM-bilder av uranprøver, ble det foretatt flere steg før de endelige klassifiseringsmodellene kunne trenes opp. Resultatene fra disse stegene vil bli beskrevet i dette kapittelet og utvalgte resultater fra klassifiseringsmodellene blir presentert. I vedleggene vises samtlige resultater fra klassifiseringsmodellene.

5.1 AMT-spekter

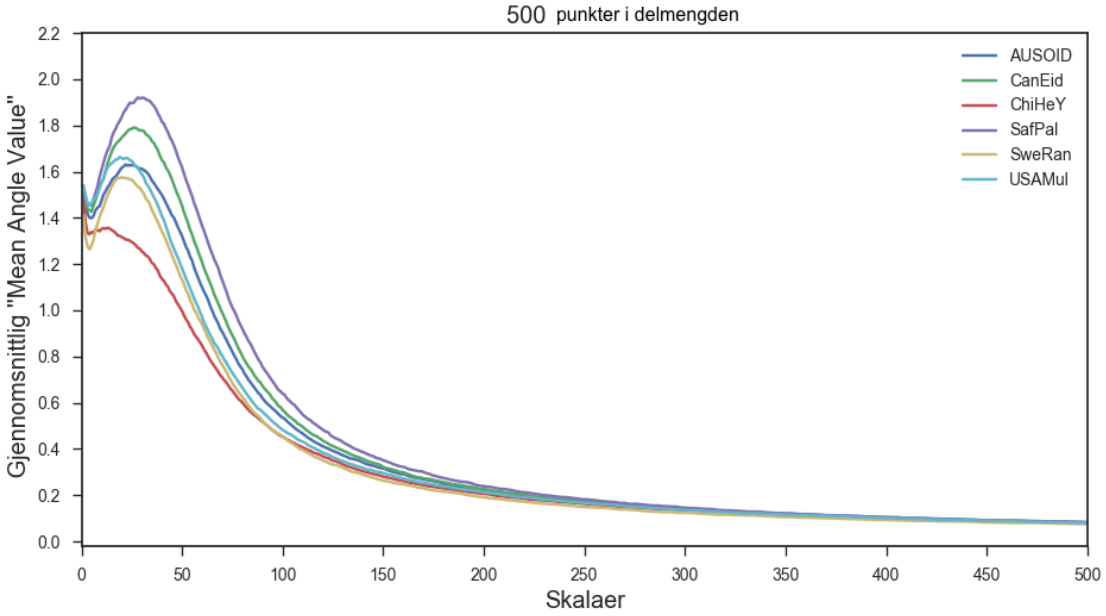
AMT-spektrene ble plottet for å visualisere hvordan ”Mean Angle Value” endrer seg som funksjon av skala-verdien S (radien) for bildene i hver klasse. Spektrene ble plottet med AMT-data for tilfellet med 500 og med 2000 utvalgte punkter i delmengden. I begge tilfellene er det benyttet at maksimal verdi for S er 500.

Figur 5.1 og Figur 5.2 viser plottene for *gjennomsnittlig* ”Mean Angle Value” for hver av de seks klassene. Det er altså tatt et gjennomsnitt av AMT-spektrene for bildene fra hver klasse, og dermed vises seks spektrallinjer her; én for hver av de seks klassene. Figur 5.2 viser at med 2000 punkter i delsettet blir kurvene glattere sammenlignet med tilfellet med 500 punkter i delmengden i Figur 5.1. Dette viser at med flere punkter i delmengden blir klassene mer distinkte. Figurene viser også at ved økt verdi av S blir klassene gradvis mindre adskilt. Dette viser at ved større radier blir klassene vanskeligere å skille. I Figur D1 og Figur D2 i Vedlegg D vises AMT-spektre for *hvert* bilde i datasettet.

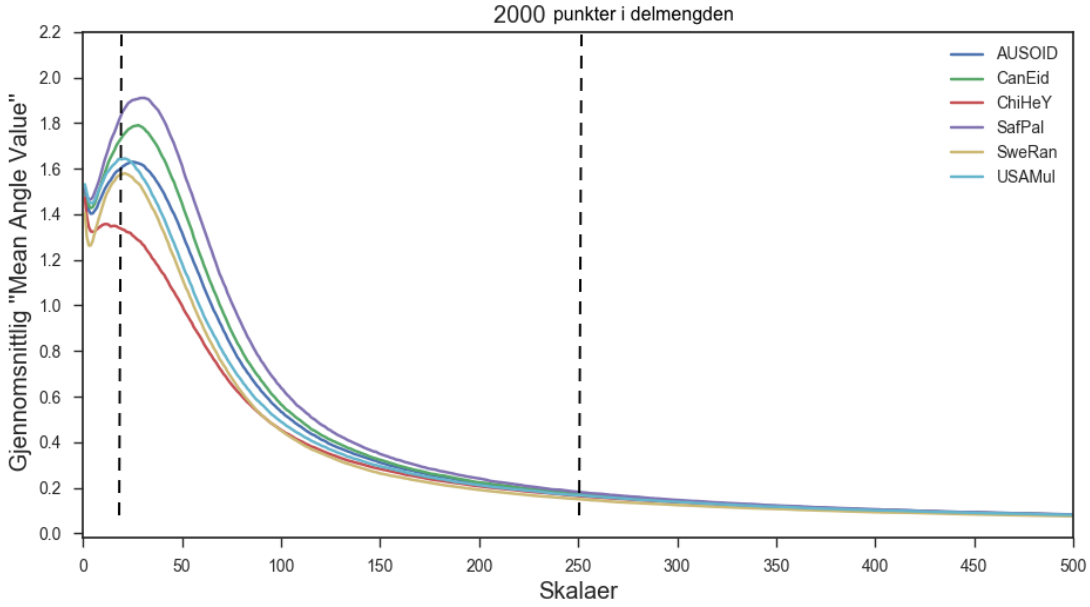
Ved manuell inspeksjon av Figur 5.2 ble det valgt ut et område (omsluttet av stiplede linjer) i spekteret som syntes å ta for seg mesteparten av variasjonen mellom klassene. Dette området ble valgt til å omfatte verdier av S i intervallet 20-250. Det reduserte AMT-datasettet er det som er omtalt som ”AMT2000 - redusert” i oppgaven.

AMT-spektrene beskriver hvordan ”Mean Angle Value” endrer seg som funksjon av skalaen S. Verdien av ”Mean Angle Value” svarer til den lokale variasjonen i bildekompleksitet, som angir teksturelle kvaliteter ved prøvene slik som grad av ruhet, glatthet og ujevnhet. Høye verdier av ”Mean Angle Value” er knyttet til kompleks bildetekstur, mens lave verdier svarer til

mindre kompleks bildetekstur. Forskjellene på spekter-linjene for de seks klassene kan dermed karakterisere forskjeller i teksturegenskaper for uranprøvene i de seks klassene.



Figur 5.1: AMT-spekter med 500 skalaverdier (S) og 500 punkter i delmengden. Her er gjennomsnittet av "Mean Angle Value" for hver klasse med bilder plottet som funksjon av skala-verdien S.



Figur 5.2: AMT-spekter med 500 skala-verdier og 2000 punkter i delmengden. Her er gjennomsnittet av "Mean Angle Value"-verdiene for hver klasse med bilder plottet som funksjon av skala-verdien (S). Området mellom de stiplede linjene representerer det området i plottet som synes å fange den største delen av variasjonen mellom klassene, og som i oppgaven blir omtalt som "AMT2000 - redusert".

Ved å betrakte AMT-spektrene i Figur 5.2 kommer det frem at spekterlinjene har størst spredning i første del av spekteret, og dermed har klassene størst variasjon hva angår bildekompleksitet i dette området.

Figur 5.2 viser videre at enkelte av klassene lar seg skille fra resten av klassene. Det er spesielt to klasser, ChiHeY (Kina) og SAfPal (Sør-Afrika), som skiller seg fra resten av klassene. Den røde linjen tilhører klassen ChiHeY og viser de laveste verdiene av "Mean Angle Value", mens den fiolette linjen tilhører klassen SAfPal og har de høyeste verdiene av "Mean Angle Value". Dette vitner om at prøvene fra ChiHeY har en overflatestruktur med store korn, det vil si større områder i bildet med små endringer i intensitetsverdier, mens prøvene fra SAfPal har en jevnere overflate med tilstedeværelse av små korn. Dette sammenfaller med det som kommer frem av eksempelbildene fra klassene som er vist i Figur 3.1, der spesielt prøven fra ChiHeY skiller seg ut fra resten av prøvene med en overflate med store korn sammenlignet med de andre. Prøvene med middels høye verdier av "Mean Angle Value" skiller seg ikke like mye fra de andre klassene og vitner om prøver med overflater som har middels store korn.

5.2 Egenskapsutvelgelse

Forskjellige metoder for egenskapsutvelgelse ble tatt i bruk. Resultatene for de forskjellige metodene presenteres her. Hvilke egenskaper som ble valgt ut med de forskjellige metodene er presentert i Tabell B1 som finnes i Vedlegg B.

5.2.1 Fjerne egenskaper med lav varians

En funksjon fra Scikit-learn ble benyttet for å fjerne variablene med lav varians slik at antall egenskaper ble redusert. I Tabell 5.1 vises resultatet fra variabelseleksjonen der terskelverdien er satt slik at egenskapene som har lik verdi i mer enn 80% av prøvene blir fjernet. Tabellen viser hvor mange egenskaper tillagt hver egenskapsblokk det var før variabelseleksjonen og etter at egenskaper med lav varians ble fjernet, samt andel av egenskapene angitt i prosent (%) som ble fjernet.

Tabell 5.1 viser at for de ulike egenskapsblokkene ble det fjernet ulike andeler med egenskaper basert på den angitte terskelverdien for varians. For egenskapsblokken "GLCM" ble størst andel av egenskapene fjernet, 89%, mens for alle egenskapsblokkene med AMT-egenskaper og for LBP-egenskaper ble ingen av egenskapene fjernet.

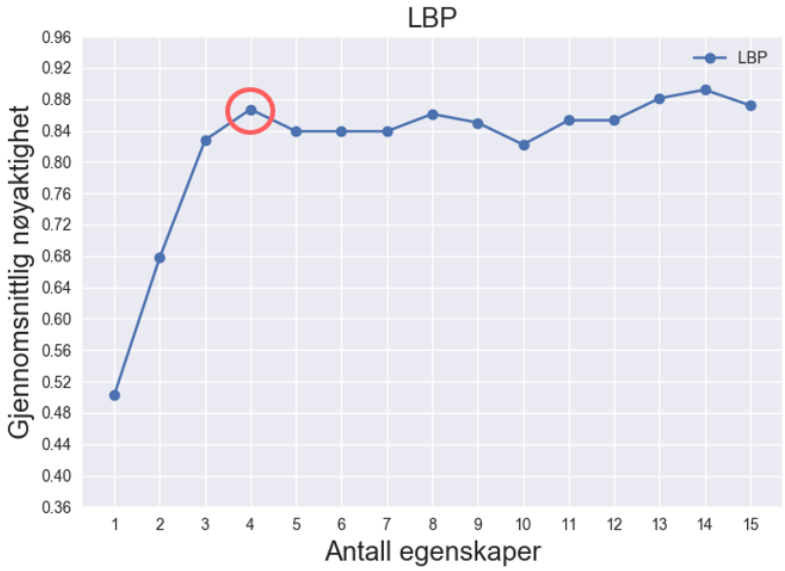
Tabell 5.1: Oversikt over antall bildeegenskaper før og etter at bildeegenskapene med varians lavere enn terskelverdien ble fjernet, samt andelen egenskaper (i %) som ble fjernet.

Egenskapsblokk	# egenskaper før	# egenskaper etter	Andel fjernet [%]
Førsteorden	171	122	29%
GLCM	945	107	89%
GLRLM	144	64	56%
GLSZM	135	72	47%
NGTDM	36	9	75%
GLDM	126	57	55%
Pyradiomics - alle	1557	431	72%
LBP	60	60	0%
Pyradiomics og LBP	1617	491	70%
AMT500	500	500	0%
AMT2000	500	500	0%
AMT2000 - redusert	230	230	0%

5.2.2 Viktighet av egenskaper med Random Forest

Den gjennomsnittlige nøyaktigheten fra den nøstede kryssvalideringen med klassifiseringsalgoritmen Random Forest ble plottet som funksjon av antall egenskaper for hver egenskapsblokk. Basert på disse plottene ble det valgt ut et optimalt antall egenskaper for hver egenskapsblokk som ble brukt videre i opptreningen av modellen.

Figur 5.3 viser et slikt plott med egenskaper fra Local Binary Patterns (LBP) som eksempel, der det ble valgt ut fire egenskaper (markert med rød ring). Denne verdien ble valgt ved å identifisere området på grafen der den gjennomsnittlige nøyaktigheten syntes å være høy og stabil, samt ved å se dette i sammenheng med antallet egenskaper, da en enklere modell er å foretrekke. Plottet viser at etter de fire første egenskapene blir ikke den gjennomsnittlige nøyaktigheten markant høyere før fjorten egenskaper, men siden det var ønskelig med en enklere modell, ble det valgt ut fire egenskaper for dette tilfellet.



Figur 5.3: Figuren viser et nøyaktighetsplott med Random Forest for egenskapsblokken "LBP" med antall egenskaper på x-aksen og gjennomsnittlig nøyaktighet fra nøstet kryssvalidering på y-aksen. Den røde ringen markerer det antallet egenskaper som ble valgt ut.

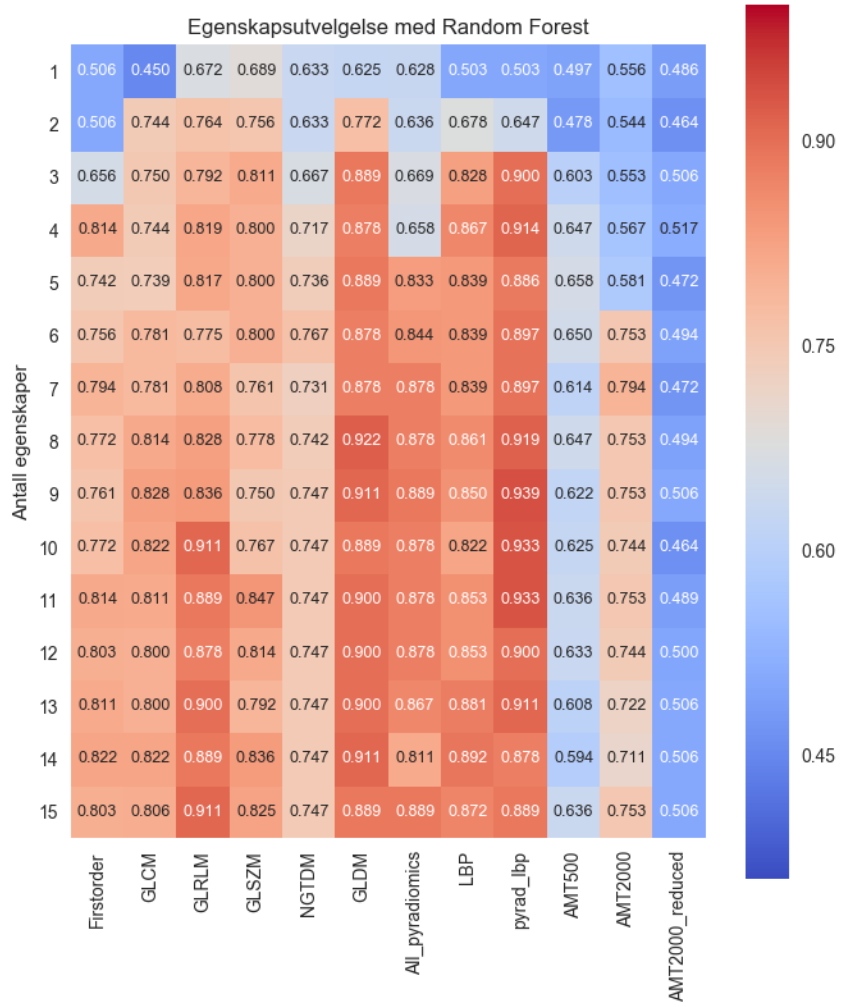
Tabell 5.2: Oversikt over antall egenskaper som ble selektert ut for hver egenskapsblokk basert på Random Forest sin algoritme for utvelgelse av de mest relevante egenskapene ("feature importance") og gjennomsnittlig nøyaktighet fra nøstet kryssvalidering. Alle egenskaper med varians lavere enn en angitt terskelverdi (se delkapittel 5.2.1) var fjernet på forhånd.

Egenskapsblokk	**# egenskaper**
Førsteorden	4
GLCM	9
GLRLM	10
GLSZM	11
NGTDM	6
GLDM	8
Pyradiomics - alle	9
LBP	4
Pyradiomics og LBP	9
AMT500	5
AMT2000	7
AMT2000 - redusert	4

Valg av antall egenskaper ble gjennomført for alle egenskapsblokkene, og plottene for de resterende egenskapsblokkene er presentert i Figur B1 i Vedlegg B. Figur 5.4 viser et såkalt "varmekart" (*heatmap*) over gjennomsnittlig nøyaktighet fra den nøstede kryssvalideringen

som funksjon av antall egenskaper for de forskjellige blokkene. Denne figuren fremstiller de samme resultatene som Figur B1, men på en måte som gjør det enklere å sammenligne de forskjellige blokkene. Der vises det blant annet at egenskapsblokkene for AMT gir de generelt laveste verdiene, mens ”Pyradiomics og LBP”, ”GLDM” og ”GLRLM” gir høyere verdier for nøyaktighetene. Figur 5.4 viser også at få egenskaper generelt gir lavere nøyaktighet, og at nøyaktigheten er høyest ved middels antall egenskaper. For eksempel, for ”GLDM” gir åtte egenskaper høyere nøyaktighet enn femten egenskaper.

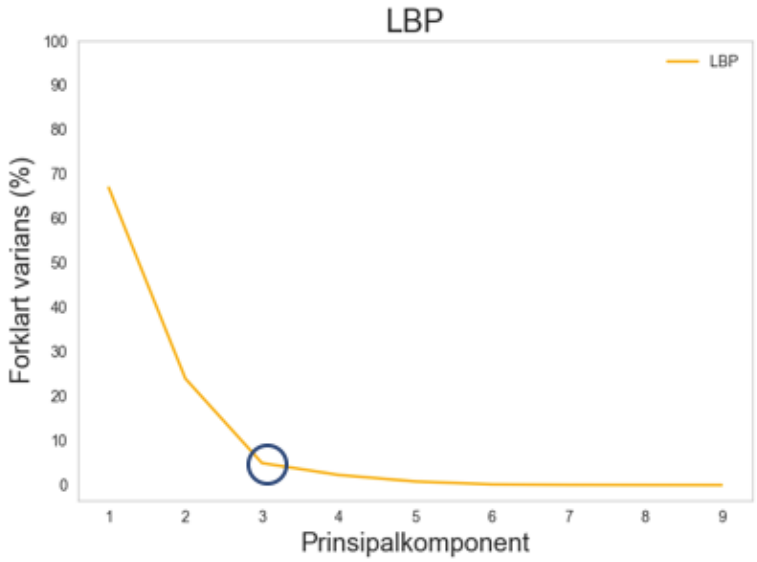
En oversikt over antall egenskaper som ble valgt ut for hver egenskapsblokk er presentert i Tabell 5.2. Hvilke bildeegenskaper som ble valgt ut er presentert i Tabell B1 i Vedlegg B. Fra disse tabellene kommer det frem at for alle egenskapsblokkene ble det valgt ut mellom fire og elleve egenskaper. For AMT ble det valgt ut rundt fem egenskaper, der alle de utvalgte egenskapene er ved små skalaverdier (S). For både ”AMT500” og ”AMT2000” er verdiene av S under 19, mens for ”AMT2000 - redusert” er det skalaverdier mellom 21 og 27. Alle skalaverdiene for ”AMT2000 - redusert” ligger mellom 21 og 250, og dette kan tyde på at det burde vært tatt med flere av de lave skalaverdiene. I Tabell B1 vises det at for egenskapene som er trukket ut ved hjelp av Pyradiomics, er det stort sett kun de egenskapene fra de wavelettransformerte bildene som blir valgt. Et unntak er egenskapen $''SZN\_glszm''$ som er trukket ut fra de opprinnelige bildene. Denne egenskapen blir også valgt ut i egenskapsblokken der alle egenskapene fra Pyradiomics er samlet. Denne egenskapen er et mål på variasjonene i størrelsene på de sammenhengende områdene i bildene [54]. Egenskapene som er valgt ut for ”Pyradiomics - alle” er stort sett egenskaper fra ”GLRLM”, og noen av de samme egenskapene er også valgt ut for ”Pyradiomics og LBP”, men der er de aller fleste egenskapene fra ”LBP”.



Figur 5.4: Varmekart som viser gjennomsnittlig nøyaktighet fra nøstet kryssvalidering for antall egenskaper mot egenskapsblokk. Random Forest sin algoritme for utvelgelse av de mest relevante egenskapene (Feature Importance) er benyttet.

5.2.3 Prinsipalkomponentanalyse

For hver egenskapsblokk ble forklart varians plottet for hver prinsipalkomponent. Ved å betrakte grafene, samt å se på den totale forklarte variansen til komponentene, ble det valgt ut et antall prinsipalkomponenter for hver egenskapsblokk som ble brukt videre i modellene. Figur 5.5 viser et slikt plott med egenskapsblokken ”LBP” som eksempel. Der er det valgt ut tre komponenter. Antall komponenter ble valgt der grafen har et knekkpunkt og neste komponent ikke gir betydelig økning i forklart varians. Den blå ringen i figuren angir det punktet på grafen der den forklarte variansen ikke øker i særlig grad ved økt antall prinsipalkomponenter, slik at for dette tilfellet ble det valgt ut tre prinsipalkomponenter.

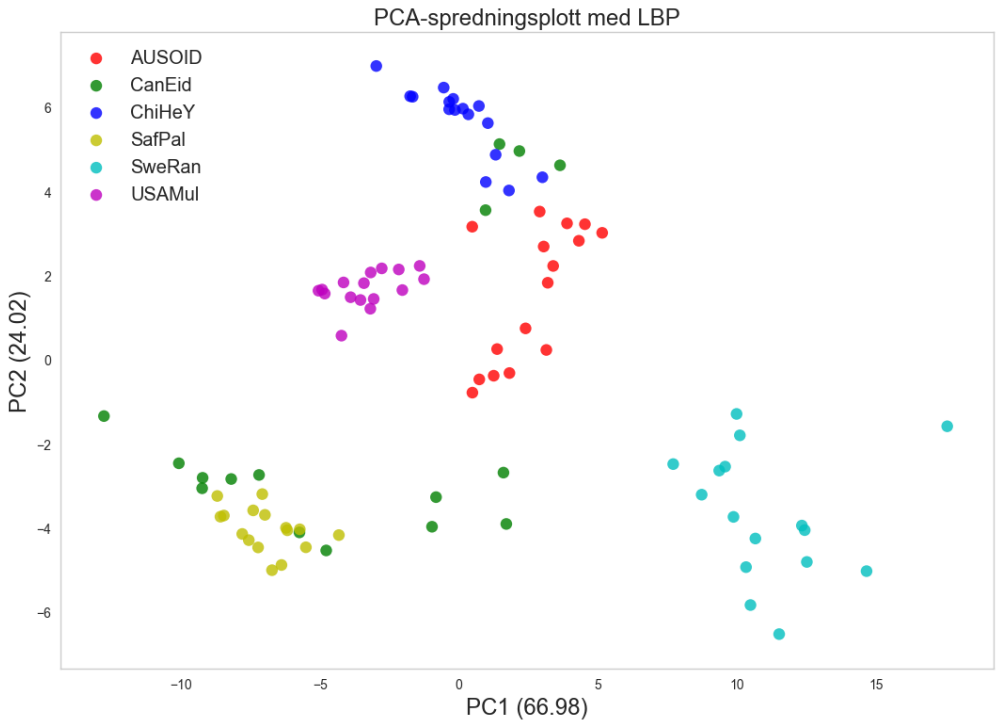


Figur 5.5: Plottet viser forklart varians for hver prinsipalkomponent for egenskapsblokken "LBP". Den blå ringen angir at det ble valgt ut tre prinsipalkomponenter som ble brukt i videreutviklingen av modellene.

Valg av antall prinsipalkomponenter ble gjort for alle egenskapsblokkene og ble brukt i videreutviklingen av modellene. Grafer med forklart varians for de resterende egenskapsblokkene, vises i Figur B2 i Vedlegg B. Antallet prinsipalkomponentene som ble valgt ut for hver egenskapsblokk, samt summen av den forklarte variansen for de utvalgte komponentene for hver egenskapsblokk, er presentert i Tabell 5.3. Der vises det at det ble valgt mellom to og syv komponenter for alle egenskapsblokkene. For alle egenskapsblokkene med AMT, ble det kun valgt ut to komponenter der de til sammen gir svært mye av den forklarte variansen ($\geq 88\%$). For de andre egenskapsblokkene er det valgt ut noen flere komponenter, og den totale forklare variansen for hver av egenskapsblokkene er mellom 80 % og 100 %. Variansen i spesielt egenskapsblokkene "GLRLM", "NGTDM" og "LBP" forklares godt ($\geq 94\%$). Hvor mye forklart varians hver av komponentene beskriver er presentert i Tabell B1 i Vedlegg B. Der vises det at for "AMT2000 - redusert" vil over 90 % av den forklarte variansen ligge i den første prinsipalkomponenten. For egenskapsblokkene fra Pyradiomics, bortsett fra "Førsteorden", forklarer den første komponenten kun omlag 30 % av variansen.

Tabell 5.3: Oversikt over hvor mange prinsipalkomponenter som ble valgt for hver egenskapsblokk, samt den forklarte variansen som de utvalgte prinsipalkomponentene utgjør til sammen.

Egenskapsblokk	# prinsipal-komponenter	Forklart varians [%]
Førsteorden	4	84
GLCM	6	90
GLRLM	6	94
GLSZM	7	89
NGTDM	6	100
GLDM	6	88
Pyradiomics - alle	6	84
LBP	3	96
Pyradiomics og LBP	5	80
AMT500	2	88
AMT2000	2	94
AMT2000 - redusert	2	99



Figur 5.6: PCA-spredningsplott med to prinsipalkomponenter (PC1 og PC2) for bildeegenskapene basert på LBP. Komponentene forklarer henholdsvis 67% og 24% av den totale variansen i dataene.

Figur 5.6 viser et spredningsplott, der to prinsipalkomponenter er plottet mot hverandre for egenskapsblokken ”LBP”. Dette plottet viser at ved å benytte to prinsipalkomponenter, PC1 og PC2, som forklarer henholdsvis 67% og 24% av den totale variansen i dataene, kan alle klassene synes å skille seg ut, bortsett fra CanEid (grønn). I figuren vises det at de forskjellige klassene danner klynger, bortsett fra CanEid (grønn). Det vises at denne klassen ligger spredt i plottet og blander seg både med SweRan (gul) og ChiHeY (mørk blå). Dette tyder på at egenskapene fra LBP egner seg godt for å skille de forskjellige klassene, med unntak av klassen CanEid. Figuren viser også at klassen SweRan (lys blå) ligger i en klynge lenger unna de resterende klassene. Spredningsplott for de resterende egenskapsblokkene er gitt i Figur B3 i Vedlegg B. Der vises det at spredningsplottene for flere av egenskapsblokkene viser de samme mønstrene som i Figur 5.6.

5.3 Nøstet kryssvalidering

Nøstet kryssvalidering ble gjennomført for alle kombinasjoner av klassifiseringsalgoritmer og egenskapsblokker med tilhørende metoder for egenskapsutvelgelse.

Tabell 5.4 presenterer ordforklaringer på forkortelser som blir benyttet i resultatene.

Tabell 5.4: Tabellen viser forklaringer på forkortelser som blir benyttet i resultatene.

Forkortelse	Ordforklaring	Forkortelse	Ordforklaring
RLV	Egenskaper med lav varians er fjernet (Removed Low Variance)	KNN	K-nærmeste nabo
LR_L1	Logistisk regresjon med L1 som regulariseringsparameter	SVM	Support Vector Machines
LR_L2	Logistisk regresjon med L2 som regulariseringsparameter	GBC	Gradient Boosting Classifier
RF	Random Forest egenskapsutvelgelse	DT	Decision Tree
RFC	Random Forest Classifier	k.v.	kryssvalidering

I Tabell 5.5 og Tabell 5.6 vises henholdsvis gjennomsnittlig nøyaktighet og tilhørende standardavvik fra den nøstede kryssvalideringen for fire egenskapsblokker og er henholdsvis de to egenskapsblokkene som gir de høyeste nøyaktighetene, ”LBP” og ”GLRLM”, og de to som gir de laveste nøyaktighetene, ”NGTDM” og ”AMT2000 - redusert”. Standardavvikene viser variasjoner i klassifisering ved nøstet kryssvalidering. Resultatet fra de resterende åtte egenskapsblokkene finnes i Tabell C1 og Tabell C2 i Vedlegg C.

Tabell 5.5 viser at RFC og SVM utpeker seg som de klassifikatorene som gir høyest nøyaktighet,

og tilhørende lave standardavvik som vist i Tabell 5.6. AdaBoost og KNN gir generelt de laveste nøyaktighetene, samt de høyeste standardavvikene, hvilket tyder på større usikkerhet for disse klassifikatorene. Videre kommer det frem av tabellene at PCA generelt gir lavere nøyaktighet enn RF. RLV og RF gir lignende og generelt høye nøyaktigheter. Dette viser at ved å redusere antall egenskaper med RF som selekteringsmetode, blir informasjonen som brukes til å skille klassene bevart.

Resultatene fra den nøstede kryssvalideringen gir en indikasjon på det som kan forventes når den endelige modellen trenes opp med tilsvarende egenskaper og klassifikatorer, og testes på et uavhengig datasett. For eksempel forventes det at nøyaktigheten for klassifikatoren Random Forest, egenskapsblokken "GLRLM" og selekteringsmetoden RF (markert med rødt i Tabell 5.5 og Tabell 5.6) gir en nøyaktighet som ligger innefor intervallet 0,91 $\pm$ 0,06. Denne modellen bruker få antall egenskaper, i motsetning til den tilsvarende modellen der det kun er fjernet egenskaper med lav varians (RLV). Modellene som er markert i rødt for egenskapsblokken "LBP" gir også høye nøyaktigheter og lave standardavvik som er gitt i Tabell 5.6. Der blir det kun benyttet tre prinsipalkomponenter.

Tabell 5.5: Tabellen viser gjennomsnittsnøyaktighetene fra den nøstede kryssvalideringen for alle klassifikatorene som har blitt benyttet for fire utvalgte egenskapsblokker. Tallene i parentes i den første kolonnen angir antall egenskaper/prinsipalkomponenter. Nøyaktigheter markert i rødt angir modeller som forventes å gi høy nøyaktighet for klassifisering på et uavhengig testsett. Disse modellene er trent opp med få egenskaper, har høy nøyaktighet og lave standardavvik (se Tabell 5.6).

Selekteringsmetode	DT	LR_l1	LR_l2	RFC	SVM	AdaBoost	KNN	GBC
GLRLM								
RLV (64)	0,81	0,82	0,86	0,91	0,81	0,81	0,78	0,88
RF (10)	0,86	0,74	0,77	0,91	0,80	0,71	0,59	0,82
PCA (6)	0,69	0,70	0,72	0,79	0,76	0,63	0,78	0,71
LBP								
RLV (60)	0,86	0,83	0,89	0,84	0,97	0,70	0,83	0,84
RF (4)	0,85	0,82	0,83	0,87	0,84	0,76	0,83	0,81
PCA (3)	0,89	0,78	0,79	0,92	0,94	0,78	0,83	0,77
NGTDM								
RLV (9)	0,63	0,41	0,41	0,71	0,50	0,55	0,55	0,70
RF (6)	0,67	0,38	0,36	0,77	0,53	0,59	0,69	0,74
PCA (6)	0,41	0,41	0,41	0,45	0,56	0,24	0,41	0,50
AMT2000 - redusert								
RLV (230)	0,60	0,61	0,73	0,63	0,71	0,36	0,58	0,59
RF (4)	0,50	0,68	0,64	0,52	0,60	0,42	0,55	0,49
PCA (2)	0,52	0,60	0,60	0,58	0,62	0,43	0,51	0,52

Tabell 5.6: Tabellen standardavvikene tilhørende de gjennomsnittlige nøyaktighetene fra den nøstede kryssvalideringen for de samme utvalgte egenskapsblokkene som vises i Tabell 5.5. Verdiene markert i rødt er standardavvikene for de modellene som ga de høyeste nøyaktighetene i den nøstede kryssvalideringen.

Selekteringsmetode	DT	LR_l1	LR_l2	RFC	SVM	AdaBoost	KNN	GBC
GLRLM								
RLV (64)	0,04	0,04	0,09	0,07	0,08	0,08	0,13	0,08
RF (10	0,04	0,05	0,11	0,06	0,08	0,16	0,06	0,07
PCA (6)	0,06	0,04	0,09	0,08	0,08	0,11	0,11	0,06
LBP								
RLV (60)	0,08	0,00	0,04	0,06	0,04	0,09	0,00	0,05
RF (4)	0,07	0,10	0,09	0,04	0,04	0,10	0,07	0,04
PCA (3)	0,06	0,04	0,05	0,07	0,02	0,07	0,00	0,17
NGTDM								
RLV (9)	0,16	0,08	0,08	0,11	0,16	0,09	0,10	0,06
RF (6)	0,12	0,13	0,13	0,10	0,11	0,09	0,19	0,03
PCA (6)	0,08	0,08	0,08	0,11	0,18	0,04	0,10	0,09
AMT2000 - redusert								
RLV (230)	0,07	0,04	0,10	0,13	0,10	0,14	0,11	0,08
RF (4)	0,13	0,08	0,04	0,07	0,05	0,15	0,07	0,10
PCA (2)	0,09	0,09	0,09	0,06	0,09	0,08	0,06	0,09

5.4 Opptrening av endelig modell og prediksjon

Etter den nøstede kryssvalideringen ble det foretatt opptrening av nye modeller med de utvalgte egenskapene på hele treningsettet før det ble foretatt prediksjoner på det usette testsettet. Dermed vil det være mulig å sammenligne resultatene fra den nøstede kryssvalideringen og resultatene fra de endelige modellene.

For å kunne knytte resultatene fra prediksjonen opp mot klassene som undersøkes, blir en oversikt over hvilke prøvenumre som korresponderer med hvilke klasser vist i Tabell 5.7 som en repetisjon.

Tabell 5.7: Tabellen gir en oversikt over hvilke klasser (og land) de forskjellige prøvenumrene tilhører.

Prøve	Forkortelse (Land)	Prøve	Forkortelse (Land)
0	AusOlD (Australia)	3	SAfPal (Sør-Afrika)
1	CanEid (Canada)	4	SweRan (Sverige)
2	ChiHey (Kina)	5	USAMul (USA)

I Tabell 5.8 vises resultatet etter den endelige opptreningen for de *samme* fire utvalgte egenskapsblokkene som ble vist i Tabell 5.5, det vil si for egenskapsblokkene ”GLRLM” og ”LBP” som generelt ga høyest nøyaktighet i den nøstede kryssvalideringen, og ”NGTDM” og ”AMT2000 - redusert” som generelt ga lavest nøyaktighet. Det er valgt ut to klassifikatorer, Random Forest og K-nærmeste nabo, henholdsvis den som generelt ga høyest nøyaktighet og den som generelt ga lavest nøyaktighet. Tabellen viser én nøyaktighet for klassifisering for treningssettet, én nøyaktighet for det uavhengige testsettet, samt hva de 24 prøvene i testsettet ble klassifisert som (listen med tallverdier). Klassene, representert ved tall mellom 0 og 5, korresponderer med de forskjellige prosesseringsstedene, slik som Tabell 5.7 viser. De uthevede tallene markerer feilaktige prediksjoner. De resterende resultatene fra de endelige modellene er presentert i Tabell C3 i Vedlegg C.

Tabell 5.8 viser at forskjellen mellom nøyaktigheten for treningen og nøyaktigheten for testsettet varierer for de ulike modellene. For en godt tilpasset modell vil differansen mellom de to nøyaktighetene være liten. Dette er tilfellet for eksempelvis to av modellene (markert med rødt i tabellen). Modellen med egenskapsblokken ”GLRLM” og Random Forest gir en nøyaktighet på testsettet lik 1, det vil si at alle prøvene klassifiseres korrekt, mens for modellen med egenskapsblokken ”AMT2000 - redusert” og KNN er testnøyaktigheten lik 0,5 og klassifiserer halvparten av prøvene feilaktig. Modellen som gir nøyaktighet lik 1 har ti utvalgte egenskaper, og har samme nøyaktighet som modellen over som har 64 egenskaper. Dette viser at med færre egenskaper fanger modellen fortsatt opp informasjonen som brukes til å skille klassene. Modellen der egenskapene for ”LBP” er redusert til tre prinspialkomponeter og klassifikatoren Random Forest er benyttet gir den samme nøyaktigheten for treningssettet som i den nøstede kryssvalideringen, men gir lavere nøyaktighet på treningssettet, slik som vist i Tabell 5.8. Det samme gjelder der klassifikatoren SVM blir benyttet, og gir en nøyaktighet for treningssettet lik 0,96 og en nøyaktighet for testsettet lik 0,83, gitt i Tabell C3 i Vedlegg C. Denne modellen har også et lavt standardavvik. For noen av modellene er nøyaktigheten for test betraktelig høyere enn for trening, eksempelvis den som er markert med grønt i tabellen. For andre modeller er tilfellet det motsatte, med nøyaktighet for trening høyere enn for test, slik som den som er markert med gult i tabellen. Dette tyder på at modellen er overtilpasset og ikke representativ for testsettet med de fire utvalgte egenskapene.

Tabell 5.8: Tabellen viser nøyaktighet for testsettet (Test) og for treningssettet (Trening), samt prediksjonene for prøvene i testsettet for fire forskjellige egenskapsblokker og to forskjellige klassifisseringsalgoritmer, Random Forest og K-nærmeste nabo. Tallene i parentes i første kolonne angir antall egenskaper/prinsipalkomponenter. Dersom alle prøvene klassifiseres korrekt, vil prediksjonen se slik ut: [0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]. De uthevede tallene markerer feilaktige prediksjoner. Nøyaktighetene markert i rødt viser modeller med liten differanse mellom trening og test, nøyaktighetene markert i grønn viser en modell med høyere nøyaktighet for test enn for trening, og nøyaktighetene markert i gult viser en modell med høyere nøyaktighet for trening enn for test.

GLRLM				
	Selekteringsmetode	**Test**	**Trening**	**Prediksjoner**
RFC	RLV (64)	1,00	0,92	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (10)	1,00	0,92	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,92	0,79	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**2**,5,5]
KNN	RLV (64)	1,00	0,77	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (10)	0,96	0,64	[**2**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,92	0,79	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,5,5]
LBP				
	Selekteringsmetode	**Test**	**Trening**	**Prediksjoner**
RFC	RLV (60)	0,79	0,83	[0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,5,5,5,**0**]
	RF (4)	0,67	0,88	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,**0**,**0**,5,**0**]
	PCA (3)	0,75	0,92	[0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,**1**,3,3,4,4,4,4,5,5,5,**0**]
KNN	RLV (60)	0,83	0,83	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,**1**,3,4,4,4,4,5,5,5,5]
	RF (4)	0,79	0,86	[0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,5,**0**]
	PCA (3)	0,79	0,83	[0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**0**]
NGTDM				
	Selekteringsmetode	**Test**	**Trening**	**Prediksjoner**
RFC	RLV (9)	0,83	0,75	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**2**,4,4,4,5,**0**,5,**0**]
	RF (6)	0,83	0,77	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**2**,4,4,4,5,**0**,5,**0**]
	PCA (6)	0,79	0,48	[0,0,0,0,**3**,**3**,1,1,**4**,2,2,2,3,**4**,3,3,**2**,4,4,4,5,5,5,5]
KNN	RLV (9)	0,71	0,55	[0,0,**2**,0,**3**,**3**,1,1,2,**0**,**0**,**0**,3,3,3,3,**2**,4,4,4,5,5,5,5]
	RF (6)	0,75	0,70	[**2**,0,0,0,1,**3**,1,1,2,**0**,**0**,**0**,3,3,3,3,4,4,4,4,5,5,5,**0**]
	PCA (6)	0,83	0,48	[0,0,0,0,1,1,1,1,2,2,**5**,**0**,3,**4**,3,3,**5**,4,4,4,5,5,5,5]
AMT2000 redusert				
	Selekteringsmetode	**Test**	**Trening**	**Prediksjoner**
RFC	RLV (230)	0,79	0,65	[0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,**2**,**2**,5,5]
	RF (4)	0,63	0,55	[**4**,**5**,0,0,**3**,**3**,1,1,2,2,2,2,3,3,3,3,4,**5**,**5**,4,**1**,**4**,5,**0**]
	PCA (2)	0,67	0,61	[0,**4**,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,**2**,**2**,5,**2**]
KNN	RLV (230)	0,67	0,57	[**4**,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,**5**,**5**,4,4,**2**,**2**,5,5]
	RF (4)	0,50	0,53	[0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,**1**,3,3,**5**,**5**,**5**,**2**,**2**,**2**,5,**0**]
	PCA (2)	0,79	0,56	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**2**,**2**,5,5]

Tabell 5.9 viser nøyaktigheten både fra nøstet kryssvalidering (med tilhørende standardavvik) og fra klassifiseringen i de endelige modellene for de samme fire utvalgte egenskapsblokkene. Ved å sammenlikne resultatene fra de to analysene, eksempelvis for modellen markert med rødt i tabellen, ligger treningsnøyaktigheten lik 0,92 innenfor standardavviket til nøyaktigheten fra den nøstede kryssvalideringen med tilsvarende modell, 0,91 $\pm$ 0,06. Det samme gjelder for samtlige av modellene, og dermed stemmer forventningen fra den nøstede kryssvalideringen overens med resultatene fra den endelige modellen brukt på er uavhengig testsett.

Tabell 5.9: I tabellen vises både de gjennomsnittlige nøyaktighetene med tilhørende standardavvik etter den nøstede kryssvalideringen fra Tabell 5.5 og nøyaktigheten for prediksjonene, både for trening og for test, fra Tabell 5.8. Feltene markert i rødt viser eksempler på modeller der nøyaktigheten for treningen i den endelige modellen ligger innenfor standardavviket til nøyaktigheten fra den nøstede kryssvalideringen.

		RFC			KNN		
	Selekteringsmetode	Nøstet k.v.	Trening	Test	Nøstet k.v.	Trening	Test
GLRLM	RLV (64)	0,91 $\pm$ 0,07	0,92	1,00	0,78 $\pm$ 0,13	0,77	1,00
	RF (10)	0,91 $\pm$0,06	0,92	1,00	0,59 $\pm$ 0,06	0,64	0,96
	PCA (6)	0,79 $\pm$ 0,08	0,79	0,92	0,78 $\pm$ 0,11	0,79	0,92
LBP	RLV (60)	0,84 $\pm$ 0,06	0,83	0,79	0,83 $\pm$ 0,00	0,83	0,83
	RF (4)	0,87 $\pm$ 0,04	0,88	0,67	0,83 $\pm$ 0,07	0,86	0,79
	PCA (3)	0,92 $\pm$ 0,07	0,92	0,75	0,83 $\pm$ 0,00	0,83	0,79
NGTDM	RLV (9)	0,71 $\pm$ 0,11	0,75	0,83	0,55 $\pm$ 0,10	0,55	0,71
	RF (6)	0,77 $\pm$ 0,10	0,77	0,83	0,69 $\pm$ 0,19	0,70	0,75
	PCA (6)	0,45 $\pm$ 0,11	0,48	0,79	0,41 $\pm$ 0,10	0,48	0,83
AMT2000 - redusert	RLV (230)	0,63 $\pm$ 0,13	0,65	0,79	0,58 $\pm$ 0,11	0,57	0,67
	RF (4)	0,52 $\pm$ 0,07	0,55	0,63	0,55 $\pm$ 0,07	0,53	0,50
	PCA (2)	0,58 $\pm$ 0,06	0,61	0,67	0,51 $\pm$ 0,06	0,56	0,79

Resultatene fra de endelige modellene for egenskapsblokken ”LBP”, se Tabell C3 i Vedlegg C, blir trukket frem som eksempel her. For å visualisere hvilke prediksjoner som var korrekte og hvilke som var feilaktige, presenteres en forvirringsmatrise i Figur 5.7, der prediksjonene for alle kombinasjonen av klassifikator og metode for egenskapsutvelgelse for ”LBP” betraktes. Det er åtte klassifikatorer og tre metoder for egenskapsutvelgelse, hvilket gir 24 modeller. Siden hver klasse har fire prøver, blir det dermed 96 prediksjoner for hver klasse. Forvirringsmatrisen viser hvor mange prøver som blir klassifisert korrekt og hvor mange som blir klassifisert feilaktig. Diagonalen markert med rødt viser antall riktige prediksjoner for hver av klassene, mens de andre tallene beskriver feilaktige prediksjoner. Prediksjonene for ”LBP” sett under ett gir en gjennomsnittlig nøyaktighet for testsettet lik 0,78.

Predikert klasse

Faktisk klasse	0	1	2	3	4	5
0	95	1	0	0	0	0
1	0	22	0	74	0	0
2	0	0	96	0	0	0
3	0	22	0	74	0	0
4	0	1	0	0	95	0
5	23	3	0	0	0	70

Figur 5.7: Figuren viser en forvirringsmatrise for prediksjonene fra alle kombinasjoner av klassifikator og metode for egenskapsutvelgelse for egenskapsblokken "LBP". Diagonalen markert med rødt angir antall korrekte prediksjoner for hver av de seks klassene.

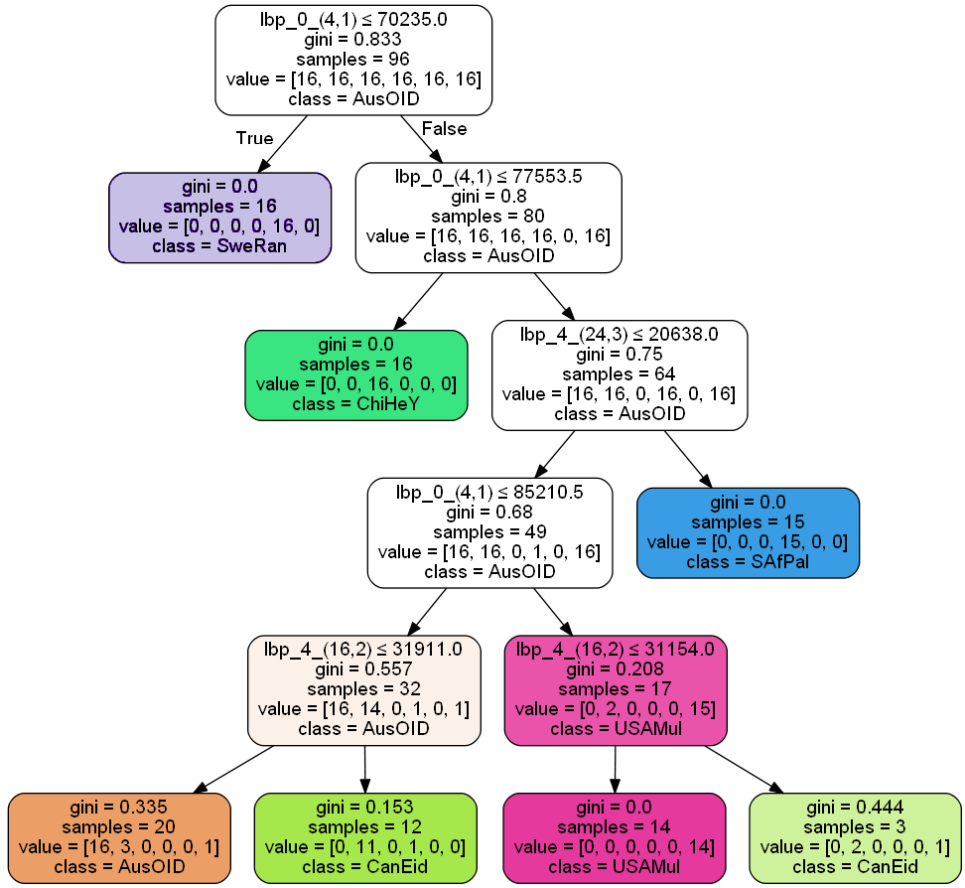
Det kommer frem av figuren at klassene 0 (AusOlD, Australia), 2 (ChiHeY, Kina) og 4 (SweRan, Sverige) blir klassifisert riktig flest ganger, i henholdsvis 95, 96 og 95 av 96 tilfeller. Prøvene fra klasse 1 (CanEid, Canada) blir oftest feilklassifisert, og blir i 74 av 96 tilfeller klassifisert som klasse 3 (SAfPal, Sør-Afrika). Prøvene fra klasse 3 og 5 (USAMul, USA) blir klassifisert riktig i henholdsvis 74 og 70 av 96 tilfeller.

Resultatene fra disse prediksjonene samsvarer med klyngene dannet i spredningsplottet av PCA-analysen i Figur 5.6. Spesielt er det resultatene fra klassen CanEid (1, Canada) som skiller seg ut som den klassen som vanskeligst lar seg klassifisere, både i spredningsplottet og i prediksjonene i forvirringsmatrisen. De andre klassene lar seg skille enklere, og dette kommer også frem i spredningsplottet.

5.4.1 Beslutningstre

Figur 5.8 viser oppbygningen av et beslutningstre for alle egenskapene basert på LBP etter en egenskapsutvelgelse med Random Forest. Etter opptrening på testsettet med et *grid search* ble parameteren for den maksimale dybden optimalisert til å være 5 da dette ga høyest nøyaktighet for klassifiseringen. Det betyr at algoritmen stiller fem "spørsmål" for å klassifisere de 96 prøver fra de seks forskjellige klassene. Det første "spørsmålet" algoritmen stiller, det vil si den øverste boksen i Figur 5.8, er om egenskapen *lbp_0_(4,1)* er mindre eller lik 70235. Alle prøvene som oppfyller dette blir direkte klassifisert som "SweRan", og dermed er denne klassen den første

som blir skilt fra resten av klassene. Dette samsvarer med det som ble vist i spredningsplottet i Figur 5.6 der prøvene for denne klassen lå i en klynge for seg selv.



Figur 5.8: Figuren viser et beslutningstre for egenskapsblokken "LBP", der egenskapsutvelgelse med Random Forest er blitt benyttet. Beslutningstreet starter med 16 prøver i hver klasse. "value" i figuren representerer antall prøver i hver klasse og følger den samme rekkefølgen som blir vist i Tabell 5.7. De forskjellige fargene representerer de forskjellige klassene.

Videre i beslutnigstreet vises det at den samme egenskapen, *lbp_0_(4,1)*, er med på å skille ut klassen ChiHeY. 15 av de 16 prøvene fra klassen SAfPal blir så trukket ut, men med en annen egenskap fra "LBP" som tar for seg et større naboskap (*lbp_4_(24,3)*). Beslutningsteet i Figur 5.8 vil til slutt feilklassifisere noen av prøvene. Nederst i figuren fra venstre vises det at tre prøver fra klassen CanEid klassifiseres som AusOlD. Én av prøvene fra USAMul og én av prøvene fra SAfPal blir til gjengjeld klassifisert som CanEid. Totalt ble kun 6 av 96 prøver feilklassifisert, og i likhet med spredningsplottet i Figur 5.6 ser det ut til at det er klassen CanEid som gir opphav til feilklassifiseringene.

5.5 Analyse uten wavelettransformerte bilder

For å kunne vurdere om bruk av wavelettransformerte bilder ga andre resultater enn å kun analysere de opprinnelige bildene uten filtertransformasjoner, ble det foretatt en ny opptrening av et utvalg av modellene. Sammenlikningen ble gjort på bakgrunn av to utvalgte egenskapsblokker, ”GLRLM” og ”Alle fra Pyradiomics og LBP”. Det ble foretatt egenskapsutvelgelse på samme måte som beskrevet i kapittel 4.3, og resultatet fra analysen vises i Tabell 5.10. De seks egenskapene i egenskapsblokken ”Pyradiomics og LBP” som ar valgt ut med RF er det kun blitt valgt ut egenskaper fra Local Binary Patterns. I tilsvarende egenskapsblokk der wavelettransformasjoner er blitt benyttet, har egenskaper fra Pyradiomics også blitt valgt ut, men da nesten utelukkende egenskaper fra de wavelettransformerte bildene. Dette vises i Tabell B1 i vedlegg B.

Tabell 5.10: Tabellen viser antall egenskaper etter egenskapsutvelgelse der kun de opprinnelige bildene uten wavelttransformasjoner er blitt benyttet. For RF vises også hvilke egenskaper som er trukket ut, og for PCA vises hvor mye varians hver av komponentene forklarer.

Egenskapsblokk	Totalt	RLV	RF (egenskaper)	PCA (forklart varians)
GLRLM	16	8	5 (’SRHGLE’, ’LRHGLE’, ’GLN’, ’LRLGLE’, ’HGLRE’)	3 (0,81, 0,16, 0,02)
Pyradiomics og LBP	233	144	6 (’lbp_0_(4,1)’, ’lbp_4_(24,3)’, ’lbp_1_(4,1)’, ’lbp_20_(24,3)’, ’lbp_4_(16,2)’, ’lbp_13_(16,2)’)	3 (0,53, 0,25, 0,08)

Det ble foretatt en nøstet kryssvalidering, men kun med klassifiseringsalgoritmene *Random Forest* og *Support Vector Machine*, samt en ny opptrening på hele treningssettet slik som beskrevet i delkapittel 4.4. Resultatene fra dette er gitt i Tabell 5.11. For å kunne sammenligne disse resultatene, er resultatene fra de samme klassifiseringsalgoritmene og egenskapsblokkene der wavelettransformasjonene er blitt benyttet hentet ut fra tabellene C1, C2 og C3 og samlet i Tabell 5.12.

Tabell 5.11: Tabellen viser resultater fra nøstet kryssvalidering og opptrening av endelig modell med Random Forest og Support Vector Machines, der wavelettransformasjoner ikke er blitt benyttet. Dette betyr at kun originalbildene er benyttet.

	RFC			SVM		
Selekterings-metode	Nøstet k. v.	Trening	Test	Nøstet k. v.	Trening	Test
GLRLM						
RLV (8)	0,63 ± 0,17	0,67	0,83	0,71 ± 0,19	0,82	0,75
RF (5)	0,67 ± 0,17	0,67	0,83	0,69 ± 0,25	0,76	0,79
PCA (3)	0,66 ± 0,16	0,65	0,71	0,69 ± 0,16	0,72	0,75
Pyradiomics og LBP						
RLV (114)	0,88 ± 0,07	0,88	0,88	0,86 ± 0,04	0,91	0,88
RF (6)	0,87 ± 0,04	0,88	0,71	0,92 ± 0,04	0,92	0,92
PCA (3)	0,84 ± 0,04	0,84	0,92	0,82 ± 0,04	0,88	0,92

Tabell 5.12: Tabellen viser resultater fra nøstet kryssvalidering og opptrening av endelig modell med Random Forest og Support Vector Machines som klassifikatorer, der wavelettransformasjoner er benyttet.

	RFC			SVM		
Selekterings-metode	Nøstet kryss-validering	Trening	Test	Nøstet kryss-validering	Trening	Test
GLRLM						
RLV (64)	0,91 ± 0,07	0,92	1,00	0,81 ± 0,08	0,86	1,00
RF (10)	0,91 ± 0,06	0,92	1,00	0,80 ± 0,08	0,88	1,00
PCA (6)	0,79 ± 0,08	0,79	0,92	0,76 ± 0,08	0,79	0,83
Pyradiomics og LBP						
RLV (491)	0,87 ± 0,07	0,89	1,00	0,84 ± 0,07	0,84	1,00
RF (9)	0,94 ± 0,04	0,94	0,79	0,89 ± 0,04	0,93	0,96
PCA (5)	0,69 ± 0,06	0,74	0,92	0,75 ± 0,08	0,79	0,71

Ved å sammenligne i Tabell 5.11 og Tabell 5.12 vises det at nøyaktigheten blir generelt høyere når wavelettransformasjonene blir benyttet. I begge tabellene er det markert et eksempel med egenskapene fra "GLRLM". Det vises at ved å benytte wavelettransformasjoner kan nøyaktighet for treningssettet øke fra 0,67 til 0,92. I tabellene er det også markert der det er benyttet klassifiseringsalgoritmen "SVM" på egenskapsblokken "Pyradiomics og LBP". Disse nøyaktighetene er svært like, men de fleste av de utvalgte egenskapene er også de samme som vist i Tabell 5.10 og Tabell B1 i vedlegg B, altså egenskaper fra "LBP".

5.6 Valg av modell

Flere av klassifiseringsmodellene som blir trent opp gir svært høye nøyaktigheter, men det er tre som utpeker seg. Disse er fra egenskapsblokkene ”Pyradiomics og LBP” og ”Pyradiomics - all” med henholdsvis klassifiseringsalgoritmene Logistisk regresjon med regulariseringsparameteren L2 og SVM, gg egenskapsblokken ”Pyradiomics og LBP” der det ikke er blitt benyttet wavelettransformasjoner med SVM. For alle modellene blir det benyttet egenskapsutvelgelse med Random Forest. De to første modellene velger ut 9 egenskaper og den tredje velger ut 6. I Tabell 5.13 vises resultatene for disse modellene. Tabellen viser at de tre modellene gir høye nøyaktigheter fra både den nøstede kryssvalideringen og den endelige opptreningen av modellene. De to første modellene klassifiserer alle prøvene i testsettet korrekt.

I Tabell 5.14 vises hvilke egenskaper som har blitt valgt ut for de forskjellige modellene. For den første modellen blir det valgt ut egenskaper fra LBP og GLRLM der egenskapene er trukket ut fra bilder med wavelettransformasjonene LLL og HLH. For den andre modellen er blir det også valgt ut egenskaåer fra GLRLM, men også GLSZM, også her blir egenskaper fra de wavelettransformerte bildene valgt ut. For den tredje modellen er det kun valgt ut egenskaper fra LBP, er blir egenskaper med både nært naboskap og naboskap litt lengre vekk valgt ut.

Tabell 5.13: Tabellen viser nøyaktigheter fra den nøstede kryssvalideringen samt opptreningen av de endelige modellene på hele testsettet og validering på et uavhengig testsett. Disse modellene er de som gir høyest nøyaktigheter.

Modell	Antall egenskaper	Nøstet k. v.	Trening	Test
Pyradiomics og LBP med RF og LR_l2	9	0,94 $\pm$ 0,07	0,96	1,00
Pyradiomics - alle med RF og SVM	9	0,88 $\pm$ 0,08	0,97	1,00
Pyradiomics og LBP uten wavelet med RF og SVM	6	0,92 $\pm$ 0,04	0,92	0,92

Tabell 5.14: Oversikt over de utvalgte egenskapene tilhørende modellene i Tabell 5.13.

Modell	Egenskaper
Pyradiomics og LBP med RF og LR_l2	lbp_4_(24,3)', 'LRE_glrlm_wavelet-LLL', 'LDHGLE_gldm_originals', 'LRE_glrlm_wavelet-HLH', 'lbp_0_(4,1)', 'Variance_first_wavelet-LLL', 'RV_glrlm_wavelet-LLL', 'lbp_20_(24,3)', 'RLN_glrlm_wavelet-HLH'
Pyradiomics - alle med RF og SVM	'LRE_glrlm_wavelet-HLH', 'GLN_glrlm_wavelet-HLH', 'LRE_glrlm_wavelet-LLH', 'LDE_gldm_wavelet-HLH', 'SZN_glszm_originals', 'RLN_glrlm_wavelet-LLH', 'Standard Deviation_first_wavelet-LLL', 'RV_glrlm_wavelet-LLL', 'GLN_glszm_wavelet-LLH'
Pyradiomics og LBP uten wavelet med RF og SVM	'lbp_0_(4,1)', 'lbp_4_(24,3)', 'lbp_1_(4,1)', 'lbp_20_(24,3)', 'lbp_4_(16,2)', 'lbp_13_(16,2)'

6 Diskusjon

6.1 Datasettet

En utfordring ved analysene i denne oppgaven er at datasettet består av få prøver. For hver av de seks klassene er det fem bilder totalt, hvorav fire er brukt til trening og ett er brukt til test. Med få prøver i både treningssettet og testsettet er det vanskelig å bygge opp en solid modell som har evne til å yte god presisjon på nye, uavhengige data. Dessuten er det større sjanse for at det ene bildet som ble brukt til test skiller seg betraktelig fra bildene fra treningssettet tilhørende samme klasse. Siden oppdelingen av trenings-og testsett ble gjort tilfeldig, var det ikke mulig å kontrollere dette. Dersom datasettet hadde hatt flere prøver, kunne modellenes ytelse ha blitt forbedret ytterligere. De fem bildene fra hver klasse er tatt av samme uranprøve, men av forskjellige deler av prøven. De fem bildene kan dermed ikke betraktes som ”biologiske replikater”, men derimot ”tekniske replikater”. Dette kan ha hatt innvirkning på utfallet av resultatene. Med et større datasett kunne også eventuelle ”outliere”, det vil si bilder som skiller seg betydelig ut fra de andre og som dermed bidrar med støy til modellen, ha blitt fanget opp og fjernet fra modellen. Det er ikke blitt tatt hensyn til eventuelle ”outliere” i denne oppgaven.

Det har ikke vært mulig å hente inn flere data, da bildetaking av uranprøver er en omstendelig prosess som må gjøres under helt bestemte forhold hva angår sikkerhet og utstyr. I et forsøk på å utvide datasettet ble hvert av bildene delt i fire ikke-overlappende bilder. Disse ble betraktet som uavhengige i oppbygningen av modellen. Dette er en antagelse som har blitt gjort, og ved tilgang på et større datasett kunne denne antagelsen ha vært unngått.

Datasettet med bilder er fra starten av splittet opp i et treningssett med 96 bilder og et testsett med 24 bilder. Det kunne vært interessant å testet ut om resultatene hadde endret seg ved å benytte en annen oppdeling.

Et annet aspekt ved datasettet som er verdt å kommentere, er at uranprøvene det er tatt bilde av stammer fra 1980-tallet [66], hvilket vil si at de er nesten 40 år gamle. Dette kan ha bidratt til at kvaliteten på prøvene har blitt redusert. Det kan blant annet være tilfellet at prøvene har fått skader i løpet av sin levetid. Dessuten er det ikke sikkert at såpass gamle prøver vil være

representative for det som finnes i dag. Metodene som brukes for å hente ut og prosessere uran kan endre seg, og da vil også strukturen på UOC-prøvene endre seg. Igjen, dette er et aspekt som begrenses gjennom at det er svært vanskelig å oppdrive nye prøver.

6.2 Egenskaper

Egenskapsblokken "GLRLM" (Gray Level Run-Length Matrix) [32] gir generelt høye nøyaktigheter i forhold til for eksempel egenskapene som kun baserer seg på førsteordens statistikk. Dette tyder på at egenskaper som beskriver den romlige fordelingen (tekstur) av intensitetsverdiene i bildene, danner det beste grunnlaget for å skille klassene. Siden "GLRLM" fanger opp teksturelle egenskaper knyttet til sammenhengende lengder i bildet, kan den beskrive homogene områder, som svarer til for eksempel størrelsen på kornene i prøven. Førsteordens statistikk gir generelt lavere nøyaktigheter, hvilket kan tyde på at å kun ta fordelingen av intensitetesverdier i bildet i betraktning ikke er tilstrekkelig for å skille klassene fra hverandre.

Egenskapsblokken "GLSZM" (Gray Level Size Zone Matrix) [57] gir også generelt høy nøyaktighet i klassifiseringen, men gir generelt litt lavere nøyaktighet enn "GLRLM". For beregning av egenskaper basert på "GLSZM", teller algoritmen opp sammenhengende *områder* (soner) i bildet som har lik intensitetsverdi, slik som vist i Figur 4.4, i motsetning til algoritmen for "GLRLM" som teller opp antall sammenhengende *lengder*, slik som vist i Figur 4.3. Algoritmen for GLSZM er mindre restriktiv enn algoritmen for "GLRLM". For eksempel, for bildet av en finkornet prøve, kan strukturer i bildet bli slått sammen i samme sone for GLSZM på en måte som fører til at ikke alle strukturene i bildet fanges like godt opp. Dette kan være en medvirkende forklaring til at "GLRLM" gir høyere nøyaktighet enn "GLSZM" i klassifiseringen.

Egenskapene basert på Local Binary Patterns (LBP) [10] gir også generelt høy nøyaktighet i klassifiseringen. For eksempel egenskapen $''lbp\_0\_(4,1)''$ fra LBP viser seg å være den egenskapen som blir brukt til å kunne skille ut klassen SweRan (4, Sverige), som vist øverst i beslutningstreet i Figur 5.8. Egenskapen tar for seg de fire nærmeste naboene med mønster '0', som vil si at alle nabopikslene er lavere enn senterpikselen. Egenskapen beskriver således lokale mønster ved den minste lengdeskalaen. Bildene fra klassen SweRan (4, Sverige) har de laveste verdiene for denne egenskapen. I beslutnigstreet i Figur 5.8 blir den samme verdien også brukt til å skille ut klassen ChiHeY (2, Kina), men da med en annen terskelverdi, og det samme gjelder for klassen USAMul (5, USA). Verdien av egenskapen $''lbp\_0\_(4,1)''$ tilhørende hver prøve i testsettet er presentert i Figur D3 i Vedlegg D, og ved å studere disse verdiene vises det at klassene SweRan og USAMul har henholdsvis de laveste og de høyeste verdiene for denne egenskapen.

Basert på egenskapsspektre beregnet av AMT [7, 8] vist i Figur 5.2 vises det at det er spesielt to av klassene, ChiHeY og SAfPal, som skiller seg fra resten av klassene, mens de andre klassene

blander seg i høyere grad. Fra prediksjonene basert på AMT, se Tabell 5.8, kommer det frem at spesielt ChiHeY (0, Kina) og også SAfPal (3, Sør-Afrika) blir predikert korrekt flest ganger, mens de andre klassene blander seg. Disse resultatene er altså i tråd med det som kommer frem av AMT-spektrene. Dette betyr at basert på AMT-egenskaper er det disse to klassene som skiller seg mest fra hverandre. I klassifiseringsmodellen gir AMT-egenskapene generelt de laveste nøyaktighetene sammenlignet med de andre egenskapene, se Tabell 5.8, hvilket er i tråd med at fire av klassene ikke lar seg klassifisere enkelt.

Fra egenskapsutvelgelsen med Random Forest viser det seg at det er lave skalaverdier (S) som blir valgt ut som de mest relevante fra AMT, slik som vist i Tabell B1. For egenskapsblokken "AMT2000 - redusert" er det noe høyere skalaverdier som blir inkludert siden alle skalaverdier under 20 er fjernet. Slik det kommer frem av AMT-spektrene er de aller minste skalaverdiene knyttet til støy og egner seg ikke til å skille klassene fra hverandre, men det synes også at verdier mellom 10 og 20 kunne bidratt til å gi mer relevant informasjon om klassene, og dermed kunne et mer optimalt valg for "AMT2000 - redusert" være skalaverdier mellom 10 og 250 i stedet for skalaverdier mellom 20 og 250.

Fongaro et al. [6] oppnår gode prediksjonsresultater for klassifisering av uranprøver basert på AMT-egenskaper. Der blir PCA benyttet for å redusere dimensjonen i datasettet, PLS-DA blir benyttet som klassifiseringsmodell, og stort sett de samme SEM-bildene som denne oppgaven tar for seg blir analysert. Fongaro et al. [6] reduserte antall piksler i SEM-bildene i forkant av analysen, slik at antallet punkter i delmengden for AMT-egenskapene tilsvarte 2% av antall piksler i bildene. I denne oppgaven er ikke antall piksler i bildene redusert, og det er benyttet et antall punkter i delmengden som maksimalt utgjør 0,2% av pikslene i bildene (2000 punkter i delmengden). Ved å øke antall punkter i delmengden eller redusere antall piksler i bildene vil klassene muligens kunne skilles tydeligere, slik som det kommer frem av forskjellen mellom spekterlinjene i Figur 5.2 og Figur 5.1. Dermed kan antallet punkter i delmengden være en forklarende årsak til at Fongaro et al. oppnår mer nøyaktige prediksjoner for klassifisering av uranprøvene.

6.3 Egenskapsutvelgelse

Det er blitt brukt to metoder for egenskapsutvelgelse; PCA [37] og Random Forest [33, 34] sin metode for å velge ut de mest relevante egenskapene, og begge metodene ble anvendt for hver egenskapsblokk separat. Antall egenskaper med Random Forest ble valgt for hver egenskapsblokk, slik som eksemplifisert i Figur 5.3 og i Figur 5.4. Det ble satt en begrensning på maksimalt femten egenskaper fra hver blokk. Det kunne ha vært utforsket ytterligere hvilken effekt flere egenskaper hadde hatt på nøyaktigheten. På den annen side syntes valget av maksimalt

femten egenskaper å være fornuftig med tanke på å unngå overtilpasning av modellen. Dessuten er Random Forest en tidkrevende algoritme, og et lavere antall egenskaper vil være mindre ressurskrevende. Antall prinsipalkomponenter ble valgt ved å betrakte den forklarte variansen til hver prinsipalkomponent, slik som vist i Figur 5.5, og anslå det optimale antallet som det punktet der forklart varians ikke økte betraktelig. PCA er rask, reduserer dimensjonen til datasettet kraftig samtidig som den beskriver variansen i datasettet på en akseptabel måte.

Basert på resultatene fra den endelige modellen, viste det seg at Random Forest generelt ga høyere nøyaktighet enn PCA. Dette kan vitne om at Random Forest er en mer robust metode for egenskapsutvelgelse, men det kan også være et resultat av at valg av antall prinsipalkomponenter ikke var optimalt. Ved å undersøke om prinsipalkomponentene valgte ut de samme egenskapene som Random Forest, kunne metodene som sådan blitt sammenlignet på et mer robust grunnlag. Dette kunne vært gjort med et ladningsplott for prinsipalkompontene som ble valgt ut for å visualisere hvilke egenskaper prinsipalkomponentene består av.

Resultatene fra blokkene der kun egenskaper med lav varians ble fjernet, viste generelt høye nøyaktigheter, men da disse blokkene inneholdt mange egenskaper ble det vurdert at disse åpner for overtilpasning av modellen, og dermed blir tatt i bruk kun som et innledende steg før videre egenskapsutvelgelse med andre metoder.

6.4 Nøstet kryssvalidering

I den nøstede kryssvalideringen [34] ble datasettet for treningen delt inn i flere test- og treningssett, eller ”folder”, slik som det kommer frem av Figur 4.15. Bildeegenskapene til hver prøve var organisert i rekkefølge etter klasse i regnearket som egenskapene ble lest inn fra. Dermed *kan* det ha vært tilfellet at alle egenskapene tilhørende samme klasse ble lagt til testsettet i en eller flere av foldene i den nøstede kryssvalideringen. Dersom dette har vært tilfellet, kan dette ha påvirket ytelsen til treningsmodellen. For å unngå denne problematikken hadde det vært fornuftig å randomisere rekkefølgen på egenskapene. Dette ble ikke gjort i denne oppgaven, men burde undersøkes nøyere og tas i betraktning i videre arbeid.

Resultatene fra den nøstede kryssvalideringen, som vises i Tabell 5.5 og Tabell 5.6, gir en indikasjon på hvor nøyaktige prediksjoner modellen vil gjøre på et uavhengig datasett. I nøstet kryssvalidering trenes flere modeller opp for å vurdere den gjennomsnittlige ytelsen til modellen. Det tilhørende standardavviket beskriver spredningen blant de opptrente modellene i kryssvalideringen. De fleste standardavvikene har verdier mellom 0,04 og 0,1 (se Tabell 5.6 og Tabell C2), hvilket tyder på at det ikke er stor spredning innad i modellene i den nøstede kryssvalideringen.

Fra den nøstede kryssvalideringen viste det seg at egenskapsblokkene ”GLRLM” og ”LBP” ga generelt høyere nøyaktigheter enn de andre, samtidig som at ”NGTDM” og ”AMT2000 - redusert” ga generelt lavere nøyaktigheter (se Tabell 5.5 og Tabell 5.6). Dette var i tråd med resultatene fra de endelige modellene (se Tabell 5.8). I Tabell 5.9 vises resultatet fra den nøstede kryssvalideringen, samt nøyaktighetene for treningen på hele testsettet. Eksempelvis, for egenskapsblokken ”GLRLM”, der Random Forest er benyttet som metode for egenskapsutvelgelse og for klassifisering (markert i rødt i tabellen), vises det at nøyaktigheten fra den nøstede kryssvalideringen er lik 0,91 med et standardavvik lik 0,06. Nøyaktigheten for hele treningssettet er lik 0,92 og er dermed innenfor standardavviket fra den nøstede kryssvalideringen. Dette tyder på at nøstet kryssvalidering gir et godt estimat på hva som kan forventes av en klassifiseringsmodell som trenes opp på hele treningssettet og predikerer prøver fra et uavhengig testsett.

En av utfordringene med et såpass lite datasett er at de utelatte prøvene i testsettet kan variere mye fra prøvene i treningssettet, spesielt når det ikke er tatt hensyn til eventuelle ”outliers”. De endelige klassifiseringsmodellene vil dermed kunne gi en høy nøyaktighet for treningssettet, men lavere for testsettet.

6.5 Klassifiseringsalgoritmer

De ulike klassifiseringsalgoritmene som har blitt implementert i denne oppgaven gir ulik grad av nøyaktighet i den endelige modellen, slik det fremkommer av Tabell 5.8. Resultatene viser at Random Forest [34] og SVM [33] står frem som de klassifikatorene som generelt gir høyest nøyaktighet. Både SVM og Random Forest benytter seg av ikke lineære beslutnigsgrenser. Ved å sammenligne nøyaktighetene for SVM og Random Forest med klassifikatorene for logistisk regresjon i Tabell C1 og Tabell C3 i , som bruker lineære beslutningsgrenser [33, 39], vises det at de ikke-lineære klassifikatorene gir generelt høyere nøyaktigheter. Random Forest er en klassifikator som bygger modeller basert på mange beslutningstrær, og den endelige modellen baserer seg på hvilke klassifiseringer som blir gjort flest ganger av beslutningstrærne i ensemblet. Dermed blir den endelige modellen robust, og dette reflekteres i resultatene fra de endelige modellene [33, 34] i denne oppgaven.

Til tross for at Random Forest gir høye nøyaktigheter både i den nøstede kryssvalideringen og i de endelige modellene, må det tas i betraktning at denne algoritmen er svært tidkrevende i forhold til for eksempel SVM, som også generelt gir høy nøyaktighet i klassifiseringen og som ikke er like tidkrevende.

Sammenligningen av et utvalg av nøyaktigheter fra den nøstede kryssvalideringen og fra de endelige modellene, vist i Tabell 5.9, viser at treningsnøyaktighetene fra de endelige modellene ligger innenfor standardavviket tilhørende nøyaktighetene fra den nøstede kryssvalideringen.

For modellen med egenskaper fra egenskapsblokken ”GLRLM” der Random Forest er blitt benyttet både for egenskapsutvelgelse og klassifikator, gi den nøstede kryssvalideringen en nøyaktighet på 0,91 $\pm$ 0,06 og en nøyaktighet for hele treningssettet på 0,92. Dette tyder på at nøstet kryssvalidering gir et passende estimat på hva som kan forventes ved å trene opp modellen på hele treningssettet.

6.6 Opptrening av endelig modell og prediksjon

Når testsettet blir predikert med den endelige modellen, viser det seg at nøyaktighetene for testsettet avviker i forhold til nøyaktigheten for treningssettet for noen av modellene. Slik som modellen for egenskapsblokken ”GLRLM” der klassifikatoren KNN er benyttet, markert med grønt i Tabell 5.8, med nøyaktigheter henholdsvis lik 0,96 og 0,64 for test og trening, og modellen for egenskapsblokken LBP med Random Forest som klassifikator, markert med gult i den samme tabellen. Denne modellen har en nøyaktighet lik 0,67 for testsettet og 0,88 for trenigssettet. Modellene som gir høyere nøyaktighet for treningssettet enn for testsettet, vitner om overtilpasning av modellen slik at egenskapene som er valgt ut for treningen ikke er representative for testsettet. For de modellene som gir høyere nøyaktighet for testsettet sammenlignet med treningssettet, er det en annen forklaring, og kan kobles til at størrelsen på både treningssettet og treningssettet er lite. Nøyaktigheten for treningssettet gir en indikasjon på hvor godt modellen kan prestere, men med et lite testsett blir indikasjonen svekket. De få prøvene i testsettet kan for eksempel avvike fra de få prøvene i treningssettet, og dermed være vanskelig å klassifisere. Med et større testsett kunne det forventes at nøyaktighetene for treningssettet og for testsettet var mer like.

6.7 Wavelettransformasjoner

Pyradiomics [49] som har blitt benyttet i denne oppgaven er laget for å behandle tredimensjonale bilder, og for å enkelt kunne ta i bruk funksjonene, ble bildene i denne oppgaven konvertert fra todimensjonale til tredimensjonale. Da fikk alle bildene kun ett element (piksel) i den tredje dimensjonen. Alle kombinasjonene med lavpassfilter og høypassfilter for wavelettransformasjonene [54] for tredimensjonale bilder ble brukt da det allerede var ferdig implementert i Pyradiomics. Dette ga åtte forskjellige kombinasjoner og åtte nye transformerte bilder. To og to av de åtte kombinasjonene ga det samme transformerte bildet, og det var i realiteten kun fire forskjellige transformasjoner (bilder). Det ble trukket ut egenskaper fra alle de åtte transformerte bildene i tillegg til de opprinnelige bildene. Halvparten av de transformerte bildene kunne med fordel ha vært fjernet i forkant av analysene, da egenskapene for de like transformerte bildene

også ble de samme. Dette vil gi et mindre og datasett, det unngås overlappende egenskaper og beregningene vil ta kortere tid.

Sammenligningen av de wavelettransformerte bildene og de opprinnelige bildene (Tabell 5.10) viste at nøyaktigheten for klassifiseringen var generelt høyere for egenskapene fra de transformerte bildene, spesielt for "GLRLM". Dette tyder på at å transformere bildene bidrar til å fange opp informasjon fra bildene på en måte som ikke lar seg gjøre med kun de opprinnelige bildene. Det at egenskapsblokken "Pyradiomics - alle" presterer omtrent like godt med og uten de transformerte bildene kan være fordi egenskapene for LBP også er tatt med her. LBP er kun benyttet på de ikke-transformerte bildene, og har egenskaper som viser seg å kunne skille klassene godt. Der egenskapsutvelgelse med Random Forest er blitt benyttet på egenskapene uten wavelttransformasjoner, har det kun blitt valgt ut egenskaper fra LBP. Et forslag til videre arbeid er å trekke ut egenskaper basert på LBP fra wavelettransformerte bilder.

En ulempe med å transformere bildene er at beregningstiden øker og at modellen blir mer omfattende. Dessuten er det intuitivt enklere å tolke teksturelle egenskaper fra bilder som ikke er transformert. Dette burde testes mer, og potensielt sett kan enkelte transformasjoner gi bedre resultater enn andre. Dersom en vil ha en enkel og rask modell, kan egenskaper hentet ut med LBP og med SVM som klassifikator være en metode som kan fungere.

Et punkt som må tas i betrakning er at de samme egenskapene med de tilsvarende wavelettransformasjonene kan bli valgt ut som de mest relevante i egenskapsutvelgelsen. Et eksempel på dette finnes i egenskapsblokken "Pyradiomics - alle", der egenskapen $''LRE\_glrlm''$ er blitt valgt ut med transformasjonene tilhørende både HLH og LLH. Disse to egenskapene er like, slik som forklart i delkapittel 4.2.10, og bidrar dermed ikke til å beskrive mer av teksturen i bildene. Dette kunne vært unngått ved å fjerne korrelerte egenskaper i forkant av analysene.

6.8 Forslag til videre arbeid

6.8.1 Egenskaper

Det foreslås at AMT undersøkes nøyere for SEM-bilder for å finne et optimalt antall punkter i delmengden for dette datasettet. I denne forbindelse foreslås det at en annen programvare enn Python benyttes, da AMT-algoritmen tilpasset bruk i Python er tidkrevende.

Siden det viser seg at det å utføre wavelettransformasjoner på bildene kan fange opp nyttig informasjon og at LBP gir gode resultater, burde det også gjøre et forsøk der dette kombineres, altså å trekke ut egenskaper med LBP fra transformerte bilder. Dette ble ikke gjort, da LBP ikke var implementert i Pyradiomics [49] da oppgaven ble påbegynt. Videre kan andre bildetransfor-

masjoner testes ut for å vurdere om egenskaper trukket ut fra disse kan øke modellytelsen. Flere transformasjonsmetoder er implementert i Pyradiomics [54], for eksempel transformasjoner som beregner kvadratroten og logaritmen til intensitetsverdiene i originalbildet, eller transformasjoner som beregner eksponenten til originalbildet [54]. Disse fremhever forskjellige egenskaper i bildene, slik som høye og lave intensiteter, samt kanter og raske intensitetsendringer.

6.8.2 Egenskapsutvelgelse

PCA kunne vært utforskes ytterligere. Det anbefales at et annet programverktøy enn Python benyttes til dette, for eksempel PLS Toolbox for bruk i MATLAB, som tillater flere muligheter for visualisering av klassifiseringsproblemer med flere klasser.

Valg av antall prinsipalkomponenter og antall egenskaper fra egenskapsutvelgelse med Random Forest kunne vært automatisert, og vært implementert i et *grid search*, men dette kan være svært tidkrevende.

Videre kan andre metoder for egenskapsutvelgelse implementeres, for eksempel Sequential Backward Selection (SBS) [34]. SBS starter med alle egenskapene og fjerner én og én inntil det oppnås et valgt antall egenskaper. Metoden ble testet i denne oppgaven, men det var svært tidkrevende å kjøre koden i Python, og dermed utgikk denne metoden i analyseprogrammet. SBS er ikke implementert i Scikit-learn slik som de andre metodene, men er beskrevet av Raschka og Mirjalili [34, s. 130], der SBS blir brukt sammen med klassifiseringsalgoritmen K-nærmeste nabo, en klassifikator som er beskrevet tidligere i oppgaven.

En annen metode som kunne vært utforsket ytterligere, er L1-regularisering [41], som er en hyppig brukt metode for egenskapsutvelgelse [33, 34]. Denne metoden ble testet grundig via Scikit-learn sin *select_from_model*-algoritme [43] i denne oppgaven, men utgikk ettersom det viste seg at denne ikke var kompatibel med klassifiseringsproblemer som innebærer *flere* klasser. Det foreslås derfor at denne metoden utforskes videre for klassifisering med flere klasser.

6.8.3 Innstilling og optimalisering av parametere

Alle moduler, klasser og funksjoner som er hentet fra allerede eksisterende programvare, først og fremst Scikit-learn [47], Pyradiomics [49] og Skimage [48], åpner for at brukeren av disse kan stille inn en rekke ulike parametre og andre innstillinger. Her er det med andre ord svært mange muligheter for innstillinger, noe som ikke har blitt utforsket i detalj i denne oppgaven. Parametrene kunne med fordel ha blitt optimalisert ytterligere, og noen av disse blir presentert i de påfølgende avsnittene.

I algoritmen for logistisk regresjon kunne for eksempel toleransen for stopp-kriteriet (*tol*) ha blitt optimalisert for å sikre at algoritmen gir identisk resultat for hver gang algoritmen blir benyttet for det samme datasettet.

For Random Forest som klassifikator kunne for eksempel algoritmen ha blitt testet for flere verdier av antall trær. Ved å teste ut enda flere innstillinger og sette disse resultatene opp mot hverandre, kunne modellenes ytelse ha blitt forbedret ytterligere. Det foreslås at dette blir tatt hensyn til i videre arbeid av lignende karakter.

Terskelverdien i funksjonen som fjerner egenskaper med lav varians ble kun testet med én verdi for terskelverdien. Ved å eksperimentere med denne verdien, kunne denne metoden blitt brukt som en enkeltstående metode for egenskapsutvelgelse uten å risikere overtilpasning av modellen.

For beregning av bildeegenskaper i Pyradiomics, ble intensitetsverdiene i bildene slått sammen via standardinnstilling av parameteren "binWidth". Pyradiomics benytter at 25 intensitetsverdier blir slått sammen, og forsvarer dette ved at dette er det antallet som oftest er optimalt hva angår tidsbruk og oppfanging av egenskaper i bilder [54]. Likevel kunne denne parameteren med fordel ha blitt optimalisert for bildene som har blitt benyttet i denne oppgaven.

For alle klassifiseringsalgoritmene er det benyttet kun én verdi av *random state*, en parameter som gir mulighet til å gjenskape resultatene fra kjøringene. Ved å teste ut flere verdier av *random state* og for eksempel beregne et gjennomsnitt fra hver av disse, kunne modellene ha gitt mer stabile resultater.

For klassifiseringsalgoritmen Logistisk Regresjon med L1 som regulariseringsparameter, kommer det opp feilmelding om at maksimalt antall iterasjoner blir nådd og at det ikke blir konvergens. $''max\_iter''$ er en input i klassifiseringsfunksjonen for logistisk regresjon i Scikit-learn [43] og er satt til 100 i denne oppgaven, slik det fremkommer av standardinnstillingen i Scikit-learn. Denne parameteren kan stilles opp til for eksempel 200. Løseren kan også byttes til "liblinear" i stedet for "saga", som har blitt brukt i denne oppgaven [43].

6.8.4 Forbedring av koden

I store deler av koden er det rom for forbedring. Flere av variablene som blir brukt blir definert flere ganger, og opptar da unødvendig plass. Det burde skrives tester for kodene slik at eventuelle feil kan bli fanget opp. Det kan også lages egne feilmeldinger som printes ut hvis brukeren av koden gjør feil.

I en av funksjonene er det implementert en såkalt *"Progressbar"*, som viser brukeren hvor mange prosent av koden som har blitt utført. Dette kunne med fordel blitt implementert flere

steder i koden, spesielt der Random Forest brukes, da algoritmen for Random Forest kan være svært tidkrevende. Ved å innføre dette kan en eventuell bruker se hvor mye som er igjen før koden er ferdigkjørt.

6.8.5 Utvikling av Graphical User Interface (GUI)

Programpakken som er utviklet i denne oppgaven forutsetter at brukeren av programmet er kjent med hvordan programmeringsverktøyet skal håndteres og brukes. Dette kan være en begrensende faktor med tanke på at fremtidige brukere med lite eller ingen erfaring med programmering potensielt sett ikke vil være i stand til å benytte seg av programpakken. For å håndtere dette, kan programpakken tilordnes et mer brukervennlig grensesnitt og på den måten tilpasses brukere med ulik faglig bakgrunn.

Et forslag til videre arbeid er dermed å utvikle en GUI (Graphical User Interface) eller en webapplikasjon. Det kunne også vært aktuelt med en mobilapplikasjon for raskt å kunne klassifisere prøver. En GUI kan for eksempel lages ved hjelp av modulen Qt [67] som bruker Python som programmeringsspråk [68]. Qt muliggjør et grensesnitt der brukeren enkelt kan navigere seg i programmet og utføre operasjoner ved kun å bruke datamus, altså uten å måtte skrive inn kommandoer. Programmet vil visualiseres som et ”datavindu” med menyer og knapper som kan klikkes på for de ulike operasjonene som programpakken tar for seg. En GUI laget i Qt vil i utgangspunktet kreve at brukeren må laste ned programvare. GUI-en kan dog også omstruktureres som en frittstående applikasjon, for eksempel ved å bruke Pyinstaller [69]. På denne måten kan GUI-en distribueres til brukere uten krav om nedlasting av programvare som ligger til grunn for GUI-en.

Med en webapplikasjon kan brukeren gå direkte inn på en nettside og utføre de samme operasjonene som er mulig i en GUI. Brukeren vil også i dette tilfellet ikke behøve å laste ned programvare selv. En ulempe med en webapplikasjon er at den kan være tidkrevende å utvikle, da også javascript og HTML må benyttes i tillegg til kode i Python. En annen ulempe med å bruke webapplikasjon er at det er en server som kjører alle skriptene. Denne serveren må kjøres til enhver tid, da det ikke er mulig å forutsi til hvilket tidspunkt en bruker vil benytte seg av applikasjonen. Dette er gjerne plasskrevende og krever en datamaskin med stor kapasitet, hvilket samsvarer med en mer kostbar datamaskin. En webapplikasjonen kan derfor potensielt sett gi flere utfordringer for utvikleren sammenlignet med en GUI, der programmet lastes ned og kjøres lokalt på hver enkelt brukers datamaskin.

Det foreslås at GUI-en muliggjør følgende: valg av bilder eller allerede beregnede egenskaper fra en fil, beregning og lagring av egenskaper, valg av metode for egenskapsutvelgelse, valg av klassifikator og klassifiseringsalgoritme, visning av resultater og klassifiseringsmodellens

nøyaktighet og prediksjoner, visning av figurer, lagring av resultater og figurer, visning av forventet tidsbruk for modellen mens programmet kjører, samt klassifisering av bilder basert på en ferdig opptrent modell der forutbestemte egenskaper trekkes ut fra bildene.

6.8.6 Andre bruksområder

I Haralick et al. [9] trekkes det ut flere av de samme teksturelle bildeegenskapene som har blitt trukket ut i denne oppgaven for å kunne klassifisere forskjellige typer sandstein. Analyseprogrammet som ble bygget opp i denne oppgaven tar utgangspunkt i bilder av UOC, men de samme metodene kan benyttes på andre typer materialer.

7 Konklusjon

I denne oppgaven har det blitt utviklet et rammeverk for et analyseprogram i Python som kan brukes til å klassifisere SEM-bilder av pulvermateriale i form av uranprøver. I oppgaven blir det vist at prøvene fra de seks forskjellige klassene lar seg skille ved hjelp av analyseprogrammet.

Det ble trukket ut teksturelle egenskaper fra bildene med åtte forskjellige metoder. Egenskapene basert på Local Binary Patterns og Gray Level Run-Length Matrix inneholdt svært nyttig informasjon for klassifiseringen, mens egenskaper basert på førsteordens statistikk ikke inneholdt tilstrekkelig med informasjon om bildene. Dette viser at den relevante informasjonen er knyttet til tekstur, det vil si den romlige fordelingen av intensitetsverdiene i bildene.

Random Forest sin metode for egenskapsutvelgelse ga høye nøyaktigheter i klassifiseringen. Ved å benytte denne metoden viste det seg at med færre egenskaper, vil klassene fortsatt være mulig å skille, samtidig som modellen er mindre utsatt for overtilpasning. Prinsipalkomponentanalyse er en rask metode for å redusere dimensjonen til et datasett, men ga generelt lavere nøyaktighet enn Random Forest.

Det ble benyttet åtte forskjellige klassifikatorer, der henholdsvis Support Vector Machine og Random Forest ga de høyeste nøyaktighetene i klassifiseringen, mens K-nærmeste nabo og AdaBoost i gjennomsnitt ga de laveste nøyaktighetene.

Klassifiseringsmodellene ble bygget opp ved å trene klassifiseringsalgoritmer på treningsdata og validert gjennom nøstet kryssvalidering. Den nøstede kryssvalideringen viste at modellene hadde liten spredning, og dermed at nøstet kryssvalidering gir et godt estimat på hva som kan forventes av de endelige modellene når de trenes opp på hele datasettet og testes på et uavhengig testsett.

Flere av klassifiseringsmodellene presterte svært godt på dette datasettet. Dette gjelder både for nøstet kryssvalidering og for klassifiseringen i den endelige modellen.

Modellen med seks utvalgte egenskaper basert på Local Binary Patterns og med Support Vector Machines som klassifikator ga en klassifiseringsnøyaktighet lik 0,92 på testsettet. Modellen med ni utvalgte egenskaper basert på Gray Level Run-Length Matrix, Gray Level Size Zone

Matrix og Local Binary Patterns, og med Support Vector Machines som klassifikator ga en klassifiseringsnøyaktighet lik 1,00. Dette vil si at den siste modellen klassifiserte alle prøvene i testsettet korrekt.

Analyseprogrammet som er utviklet kan benyttes til å identifisere pulvermateriale i form av uranprøver, men har potensiale til å brukes på andre typer bilder. For å kunne utvikle enda mer robuste klassifiseringsmodeller, er det nødvendig med et større datasett.

Referanser

[1] Mayer, K., Wallenius, M. og Ray, I. (2005). "Nuclear forensics-a methodology providing clues on the origin of illicitly trafficked nuclear materials". *The Analyst* 130, s. 433. DOI: 10.1039/b412922a.

[2] Varga, Z. et al. (2011). "Characterization and classification of uranium ore concentrates (yellow cakes) using infrared spectrometry". *Radiochimica Acta* 99.12, s. 807–813. DOI: 10.1524/ract.2011.1886.

[3] Plaue, J. W. et al. (2013). "Near infrared reflectance spectroscopy as a process signature in uranium oxides". *Journal of Radioanalytical and Nuclear Chemistry* 296.1, s. 551–555. ISSN: 1588-2780. DOI: 10.1007/s10967-012-2027-0.

[4] Olsen, A. M. et al. (2017). "Quantifying morphological features of α-U3O8 with image analysis for nuclear forensics". *Analytical chemistry* 89.5, s. 3177–3183. DOI: 10.1021/acs.analchem.6b05020.

[5] Keegan, E. et al. (2014). "Nuclear forensic analysis of an unknown uranium ore concentrate sample seized in a criminal investigation in Australia". *Forensic Science International* 240, s. 111–121. DOI: 10.1016/j.forsciint.2014.04.004.

[6] Fongaro, L. et al. (2016). "Application of the angle measure technique as image texture analysis method for the identification of uranium ore concentrate samples: New perspective in nuclear forensics". *Talanta* 152, s. 463 –474. ISSN: 0039-9140. DOI: 10.1016/j.talanta.2016.02.027.

[7] Andrle, R (1994). "The angle measure technique: A new method for characterizing the complexity of geomorphic lines". *Mathematical Geology* 26.1, s. 83–97. ISSN: 1573-8868. DOI: 10.1007/BF02065877.

[8] Esbensen, K. H., Hjelmen, K. H. og Kvaal, K. (1996). "The AMT approach in chemometrics—first forays". *Journal of Chemometrics* 10.5-6, s. 569–590. DOI: 10.1002/(SICI)1099-128X(199609)10:5/6<569::AID-CEM466>3.0.CO;2-W.

[9] Haralick, R. M., Shanmugam, K. og Dinstein, I. (1973). "Textural Features for Image Classification". *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3.6, s. 610–621. ISSN: 0018-9472. DOI: 10.1109/TSMC.1973.4309314.

[10] Ojala, T., Pietikäinen, M. og Harwood, D. (1996). "A comparative study of texture measures with classification based on featured distributions". *Pattern Recognition* 29.1, s. 51–59. DOI: 10.1016/0031-3203(95)00067-4.

[11] Ojala, T., Pietikainen, M. og Maenpaa, T. (2002). "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7, s. 971–987. DOI: 10.1109/tpami.2002.1017623.

[12] Huang, J. og Esbensen, K. H. (2000). "Applications of Angle Measure Technique (AMT) in image analysis: Part I. A new methodology for in situ powder characterization". *Chemometrics and Intelligent Laboratory Systems* 54.1, s. 1 –19. ISSN: 0169-7439. DOI: 10.1016/S0169-7439(00)00100-3.

[13] Huang, J. og Esbensen, K. H. (2001). "Applications of AMT (Angle Measure Technique) in image analysis: Part II: Prediction of powder functional properties and mixing components using Multivariate AMT Regression (MAR)". *Chemometrics and Intelligent Laboratory Systems* 57.1, s. 37 –56. ISSN: 0169-7439. DOI: 10.1016/S0169-7439(01)00120-4.

[14] Theodoridis, S. og Koutroumbas, K. (2009). *Pattern recognition*. 4. utg. Elsevier/Academic Press. ISBN: 1-282-54115-3.

[15] World Nuclear Transport Institute. *The Safe Transport of Uranium Ore Concentrates*. Tilgjengelig fra: https://www.wnti.co.uk/media/52017/FS7_EN_MAR13_V2.pdf. (Lest 26.04.2018).

[16] Morrell, J. S., Jackson, M. J. et al. (2013). *Uranium processing and properties*. Springer.

[17] Settle, F. A. (2009). "Uranium to electricity: the chemistry of the nuclear fuel cycle". *Journal of chemical education* 86.3, s. 316.

[18] Krajko, J. (2016). "Isotopic signatures for origin assessment of natural uranium samples". Doktoravhandling. Delft University of Technology. DOI: doi:10.4233/uuid:5577973b-85f3-4c2a-9c2b-353c1d6ef498.

[19] Edwards, C. og Oliver, A. (2000). "Uranium processing: a review of current methods and technology". *JOM* 52.9, s. 12–20.

[20] International Atomic Energy Agency(IAEA) (1993). *Uranium Extraction Technology - Tecnical Reports Series 359*. Tekn. rapp.

[21] Swapp, S, University of Wyomning (2018). *Scanning Electron Microscopy*. Tilgjengelig fra: https://serc.carleton.edu/research_education/geochemsheets/techniques/SEM.html. (Lest 04.04.2018).

[22] Goldstein, J. I. et al. (2018). *Scanning Electron Microscopy and X-Ray Microanalysis*. 4. utg. New York: Springer. DOI: 10.1007/978-1-4939-6676-9.

[23] Nanoscience Instruments (2018). *Scanning Electron Microscopy*. Tilgjengelig fra: https://www.nanoscience.com/technology/sem-technology/. (Lest 04.04.2018).

[24] Woodford, C. (2007/2016). *Electron microscopes*. Tilgjengelig fra: http://www.explainthatstuff.com/electronmicroscopes.html. (Lest 17.04.2018).

[25] Khursheed, A. (2011). *Scanning Electron Microscope Optics And Spectrometers.* World Scientific. ISBN: 9789812836670.

[26] Burger, W. og Burge, M. J. (2016). *Digital Image Processing: An Algorithmic Introduction Using Java*. 2. utg. London, UK: Springer Publishing Company, Incorporated. Kap. 1, s. 1–4. ISBN: 1447166833, 9781447166832.

[27] Tomra. Tilgjengelig fra: https://www.tomra.com/en.

[28] Tomra (2018). *Tomato sorting machine Sentinel II - TOMRA Sorting*. Tilgjengelig fra: https://www.youtube.com/watch?v=j4RWJTs0QCk.

[29] Kruse, O. M. O. et al. (2014). "Pixel classification methods for identifying and quantifying leaf surface injury from digital images". *Computers and Electronics in Agriculture* 108, s. 155 –165. ISSN: 0168-1699. DOI: 10.1016/j.compag.2014.07.010.

[30] Kvaal, K. et al. (1998). "Multivariate feature extraction from textural images of bread". *Chemometrics and intelligent laboratory systems* 42.1-2, s. 141–158. DOI: 10.1016/S0169-7439(98)00017-3.

[31] Hall-Beyer, M. (2017). "GLCM Texture: A Tutorial v. 3.0 March 2017". DOI: 10.13140/rg.2.2.12424.21767.

[32] Galloway, M. M. (1975). "Texture analysis using gray level run lengths". *Computer Graphics and Image Processing* 4.2, s. 172–179. DOI: 10.1016/S0146-664X(75)80008-6.

[33] James, G. et al. (2013). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer. ISBN: 1461471370, 9781461471370.

[34] Raschka, S. og Mirjalili, V. (2017). *Python Machine Learning*. 2. utg. Birmingham, UK: Packt Publishing. ISBN: 978-1787125933.

[35] Friedman, J., Hastie, T. og Tibshirani, R. (2000). "Additive logistic regression: a statistical view of boosting". *The annals of statistics* 28.2, s. 337–407. DOI: 10.1214/aos/1016218223.

[36] Freund, Y. et al. (1996). "Experiments with a new boosting algorithm". *Icml*. Bd. 96. Bari, Italy, s. 148–156.

[37] Johnson, R. A. og Wichern, D. W. (2007). *Applied multivariate statistical analysis*. 6. utg. Upper Saddle River, New Jersey: Pearson Prentice Hall. ISBN: 0131877151.

[38] Kuhn, M. og Johnson, K. (2013). *Applied Predictive Modeling*. New York: Springer. DOI: 10.1007/978-1-4614-6849-3.

[39] Hastie, T., Friedman, J. og Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer New York. DOI: 10.1007/978-0-387-21606-5.

[40] Friedman, J., Hastie, T. og Tibshirani, R. (2010). "Regularization paths for generalized linear models via coordinate descent". *Journal of statistical software* 33.1, s. 1.

[41] Tibshirani, R. (1996). "Regression Shrinkage and Selection via the Lasso". *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, s. 267–288. ISSN: 00359246.

[42] Næs, T. et al. (2002). *A user-friendly guide to multivariate calibration and classification*. NIR Publications. ISBN: 0952866625.

[43] Scikit-learn developers. *Scikit-learn*. Tilgjengelig fra: http://scikit-learn.org/stable/. (Lest 15.01.2018).

[44] Friedman, J. H. (2001). "Greedy function approximation: A gradient boosting machine." *Ann. Statist.* 29.5, s. 1189–1232. DOI: 10.1214/aos/1013203451.

[45] Joint Research Centre (JRC). URL: https://ec.europa.eu/jrc/en.

[46] *Spyder - Documentation*. URL: https://pythonhosted.org/spyder/#.

[47] Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Rasearch* 12, s. 2825–2830.

[48] Walt, S. van der et al. (2014). "scikit-image: image processing in Python". *PeerJ* 2, e453. DOI: 10.7717/peerj.453.

[49] Griethuysen, J. J. van et al. (2017). "Computational Radiomics System to Decode the Radiographic Phenotype". *Cancer Research* 77.21, e104–e107. ISSN: 0008-5472. DOI: 10.1158/0008-5472.CAN-17-0339.

[50] Hunter, J. D. et al. (2007). "Matplotlib: A 2D graphics environment". *Computing In Science & Engineering* 9.3, s. 90–95. DOI: 10.1109/MCSE.2007.55.

[51] Michael, W. et al. (2017). *mwaskom/seaborn: v0.8.1 (September 2017)*. (Lest 20.02.2018). DOI: 10.5281/zenodo.883859.

[52] *Scikit-learn user guide* (2018). Release 0.20.dev0.

[53] Scikit-image development team (2007). *Scikit Image*. Tilgjengelig fra: http://scikit-image.org/. (Lest 20.04.2018).

[54] Pyradiomics developers (2018). *Pyradimoics documentation, v. 1.3.0*. Tilgjengelig fra: http://pyradiomics.readthedocs.io/en/1.3.0/. (Lest 21.04.2018).

[55] Pietikäinen, M. et al. (2011). *Computer Vision Using Local Binary Patterns*. Springer London. DOI: 10.1007/978-0-85729-748-8.

[56] Amadasun, M. og King, R. (1989). "Textural features corresponding to textural properties". *IEEE Transactions on systems, man, and Cybernetics* 19.5, s. 1264–1274. DOI: 10.1109/21.44046.

[57] Zwanenburg, A. et al. (2016). "Image biomarker standardisation initiative-feature definitions". *CoRR* abs/1612.07003.

[58] Sun, C. og Wee, W. G. (1983). "Neighboring gray level dependence matrix for texture classification". *Computer Vision, Graphics, and Image Processing* 23.3, s. 341–352. DOI: 10.1016/0734-189X(83)90032-4.

[59] Wang, L. og He, D.-C. (1990). "Texture classification using texture spectrum". *Pattern Recognition* 23.8, s. 905–910. DOI: 10.1016/0031-3203(90)90135-8.

[60] Kvaal, K. et al. (2008). “eAMTexplorer: a software package for texture and signal characterization using Angle Measure Technique”. *Journal of Chemometrics* 22.11-12, s. 717–721. DOI: 10.1002/cem.1160.

[61] Kucheryavski, S. og Belyaev, I. (2009). “Classification and analysis of non-isotropic images by Angle Measure Technique (AMT) with contour unfolding”. *Analytica Chimica Acta* 642.1. Papers presented at the 11th International Conference on Chemometrics in Analytical Chemistry, s. 135 –141. ISSN: 0003-2670. DOI: 10.1016/j.aca.2008.12.016.

[62] Gonzalez, R. C., Woods, R. E. og Eddins, S. L. (2009). *Digital Image Processing Using MATLAB*. 2. utg. Gatesmark Publishing. ISBN: 0982085400.

[63] Daubechies, I. (1993). “Orthonormal bases of compactly supported wavelets II. Variations on a theme”. *SIAM Journal on Mathematical Analysis* 24.2, s. 499–519. DOI: 10.1137/0524031.

[64] Lee, G et al. (2006-). *PyWavelets - Wavelet Transforms in Python*. Tilgjengelig fra: https://github.com/PyWavelets/pywt. (Lest 20.03.2018).

[65] Varma, S. og Simon, R. (2006). “Bias in Error Estimation When Using Cross-validation for Model Selection”. *BMC Bioinformatics* 7.1, s. 91. DOI: 10.1186/1471-2105-7-91.

[66] Futsæther, C. M. (2018). Personlig kommunikasjon. Fakultet for realfag og teknologi, Norges miljø-og biovitenskapelige universitet, Ås, Norge.

[67] Qt (2018). *Qt Documentation*. Tilgjengelig fra: https://doc.qt.io/qt-5.10/qtgui-index.html. (Lest 12.03.2018).

[68] Vennemo, S. B. og Mørk, H. (2018). Personlig kommunikasjon. Fakultet for realfag og teknologi, Norges miljø-og biovitenskapelige universitet, Ås, Norge.

[69] PyInstaller (2018). *PyInstaller*. Tilgjengelig fra: http://www.pyinstaller.org. (Lest 12.03.2018).

Vedlegg

Vedlegg A

Tabell A1 - Tabell A6 viser oversikt over alle bildeegenskapene som har blitt beregnet og trukket ut av bildene ved bruk av Pyradiomics.

Tabell A1: Oversikt over bildeegenskaper basert på førsteordens statistikk.

Førsteordens statistikk	
Egenskap	**Forkortelse**
10th Percentile	10th perc
90th Perentile	90th perc
Energy	Energy
Entropy	Entropy
Interquartile Range	Interquartile Range
Kurtosis	Kurtosis
Maximum	Maximum
Mean	Mean
Mean Absolute Deviation	MAD
Median	Median
Minimum	Minimum
Range	Range
Robust Mean Absolute Deviation	RMAD
Root Mean Square	RMS
Skewness	Skewness
Standard Deviation	Standard Deviation
Total Energy	Total Energy
Uniformity	Uniformity
Variance	Variance

Tabell A2: Oversikt over bildeegenskaper basert på Gray Level Co-occurence Matrix.

Gray Level Co-occurence Matrix	
Egenskap	**Forkortelse**
Auto Correlation	AutoCorrelation
Cluster Prominence	ClusterProminence
Cluster Shade	ClusterShade
Cluster Tendency	ClusterTendency
Contrast	Contrast
Correlation	Correlation
Difference Average	DifferenceAverage
Difference Entropy	DifferenceEntropy
Difference Variance	DifferenceVariance
Informal Measure of Correlation 1	IMC1
Informal Measure of Correlation 2	IMC2
Inverse Difference	ID
Inverse Difference Moment	IDM
Inverse Difference Moment Normalized	IDMN
Inverse Variance	InverseVariance
Joint Average	JointAverage
Joint Energy	JointEnergy
Joint Entropy	JointEntropy
Maximum Probability	MaxProb
Sum Entropy	SumEntropy
Sum of Squares	SumSquares

Tabell A3: Oversikt over bildeegenskaper basert på Gray Level Run-Length Matrix.

Gray Level Run-Length Matrix	
Egenskap	**Forkortelse**
Gray Level Non-Uniformity	GLN
Gray Level Non-Uniformity Normalized	GLNN
Gray Level Variance	GLV
High Gray Level Run Emphasis	HGLRE
Long Run Emphasis	LRE
Long Run High Gray Level Emphasis	LRHGLE
Long Run Low Gray Level Emphasis	LRLGLE
Low Gray Level Run Emphasis	LGLRE
Run Entropy	RE
Run Length Non-Uniformity	RLN
Run Length Non-Uniformity Normalized	RLNN
Run Percentage	RP
Run Variance	RV
Short Run Emphasis	SRE
Short Run High Gray Level Emphasis	SRHGLE
Short Run Low Gray Level Emphasis	SRLGLE

Tabell A4: Oversikt over bildeeganskper basert på Gray Level Size Zone Matrix.

Gray Level Size Zone Matrix	
Egenskap	**Forkortelse**
Gray Level Non-Uniformity	GLN
Gray Level Non-Uniformity Normalized	GLNN
Gray Level Variance	GLV
High Gray Level Zone Emphasis	HGLZE
Large Area Emphasis	LAE
Large Area High Gray Level Emphasis	LAHGLE
Large Area Low Gray Level Emphasis	LALGLE
Low Gray Level Zone Emphasis	LGLZE
Size-Zone Non-Uniformity	SZN
Size-Zone Non-Uniformity Normilized	SZNN
Small Area Emphasis	SAE
Small Area High Gray Level Emphasis	SAHGLE
Sone Entropy	ZE
Zone Percentage	ZP
Zone Variance	ZV

Tabell A5: Oversikt over bildeegenskaper basert på Neighbouring Gray Tone Difference Matrix.

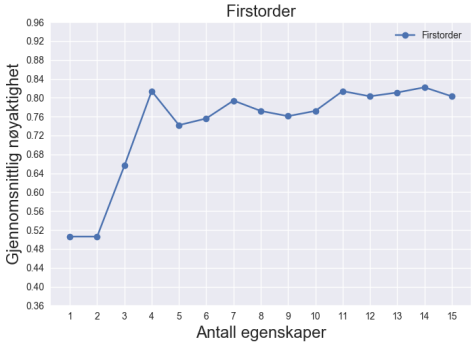
Neigbouring Gray Tone Difference Matrix	
Egenskap	**Forkortelse**
Busyness	Busyness
Coarseness	Courseness
Complexity	Complexity
Contrast	Contrast

Tabell A6: Oversikt over bildeegenskaper basert på Gray Level Dependence Matrix.

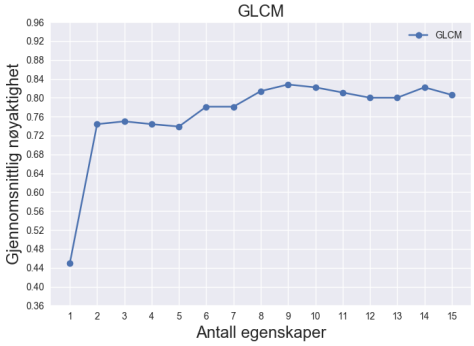
Gray Level Dependence Matrix	
Egenskap	**Forkortelse**
Dependence Entropy	DE
Dependence Non-Uniformity	DN
Dependence Non-Uniformity Normalized	DNN
Dependence Variance	DV
Gray Level Non-Uniformity	GLN
Gray Level Variance	GLV
High Gray Level Emphasis	HGLE
Large Dependence Emphasis	LDE
Large Dependence High Gray Level Emphasis	LDHGLE
Large Dependence Low Gray Level Emphasis	LDLGLE
Low Gray Level Emphasis	LGLE
Small Dependence Emphasis	SDE
Small Dependence High Gray Level Emphasis	SDHGLE
Small Dependence Low Gray Level Emphasis	SDLGLE

Vedlegg B

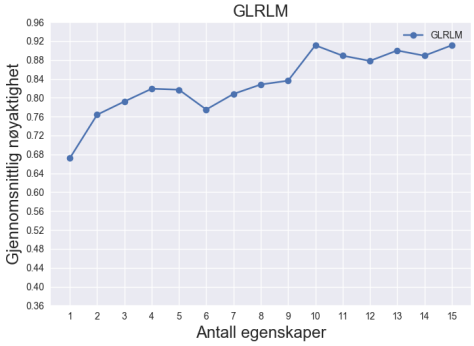
Her vises forskjellige plott som ble brukt for å velge ut antall egenskaper og antall prinsipalkomponenter, en oversikt over hvilke egenskaper/prinsipalkomponenter som ble valgt ut, samt PCA-spredningsplott for alle egenskapsblokkene.



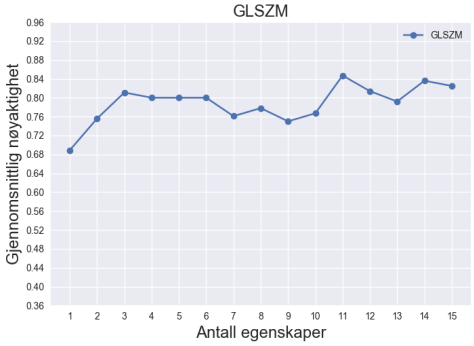
(a)



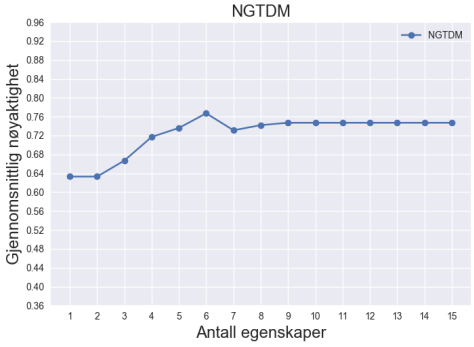
(b)



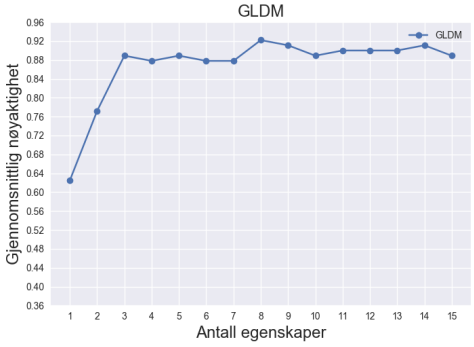
(c)



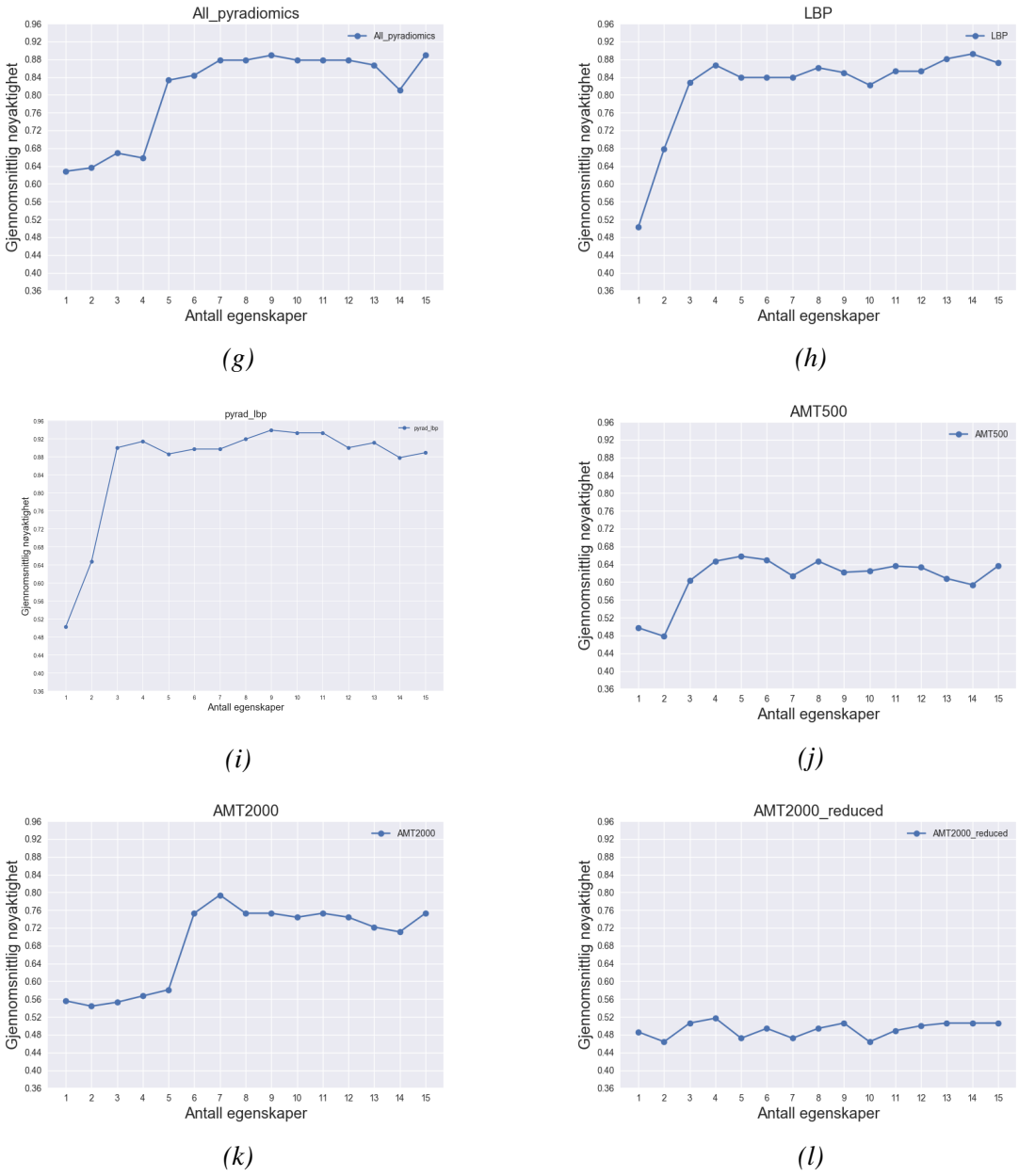
(d)



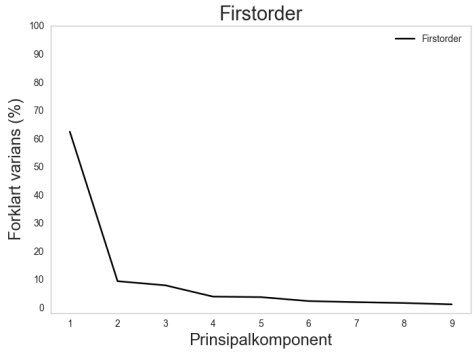
(e)



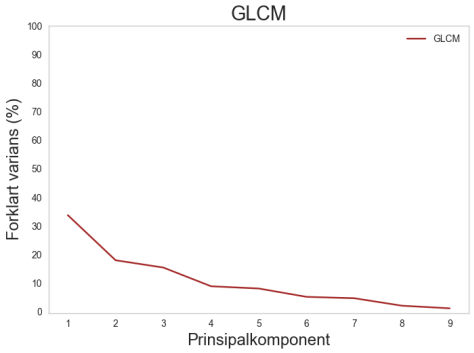
(f)



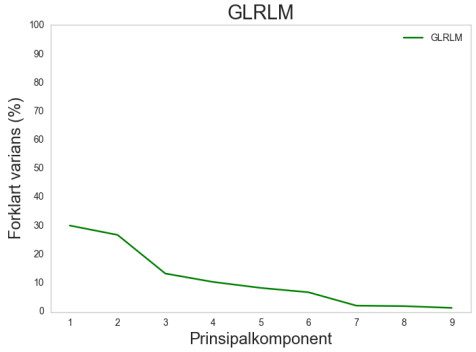
Figur B1: Figuren viser plott fra egenskapsutvelgelse med Random Forest. Plottene viser gjennomsnittlig nøyaktighet fra nøstet kryssvalidering som funksjon av antall egenskaper for hver egenskapsblokk ((a)-(k)).


Firstorder
Firstorder
Forklart varians (%)
Prinsipalkomponent

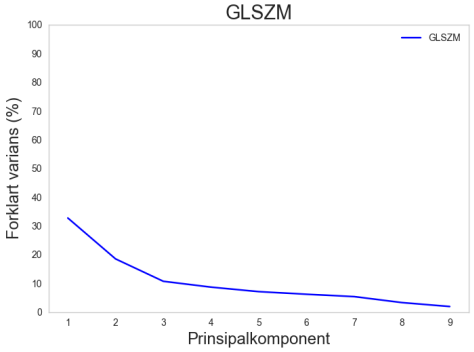
(a)


GLCM
GLCM
Forklart varians (%)
Prinsipalkomponent

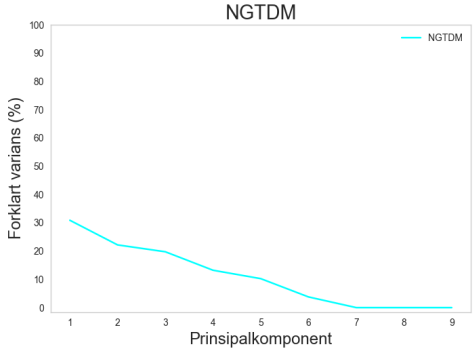
(b)


GLRLM
GLRLM
Forklart varians (%)
Prinsipalkomponent

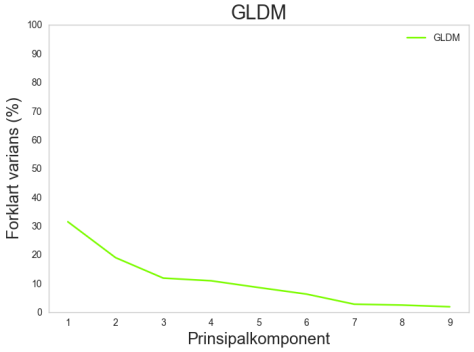
(c)


GLSZM
GLSZM
Forklart varians (%)
Prinsipalkomponent

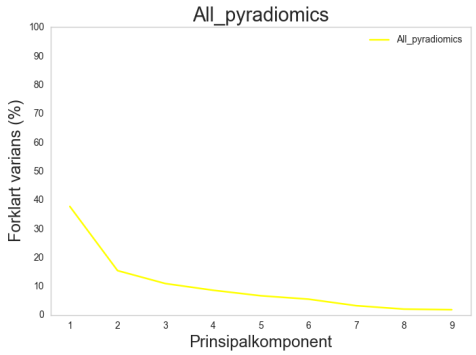
(d)


NGTDM
NGTDM
Forklart varians (%)
Prinsipalkomponent

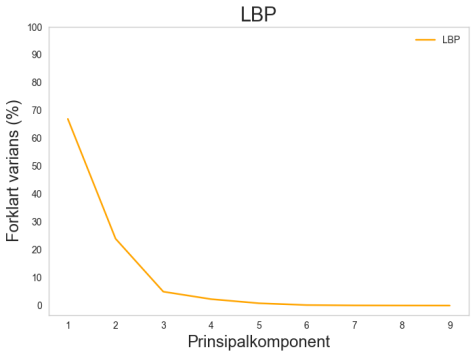
(e)


GLDM
GLDM
Forklart varians (%)
Prinsipalkomponent

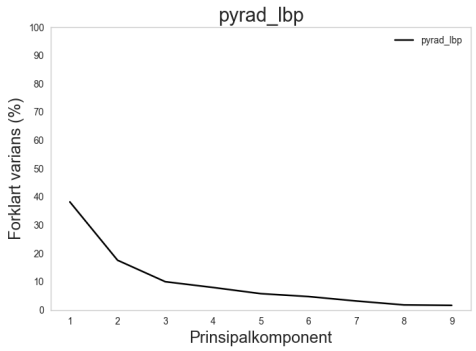
(f)


All_pyradiomics
All_pyradiomics
Forklart varians (%)
Prinsipalkomponent

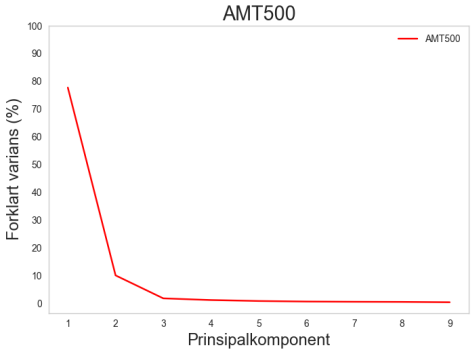
(g)


LBP
LBP
Forklart varians (%)
Prinsipalkomponent

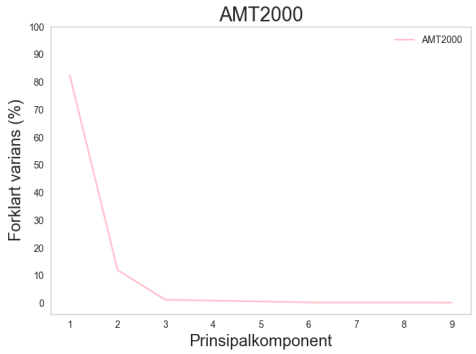
(h)



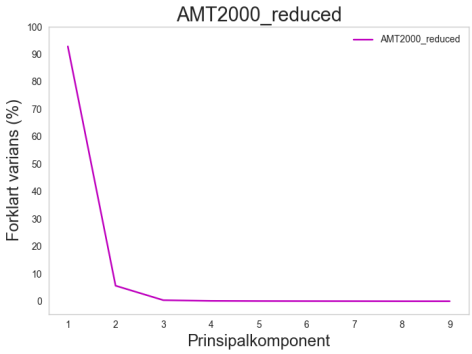
(i)



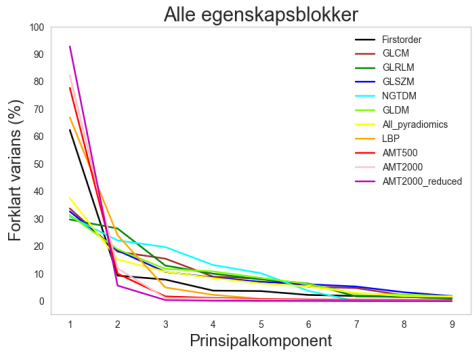
(j)



(k)



(l)



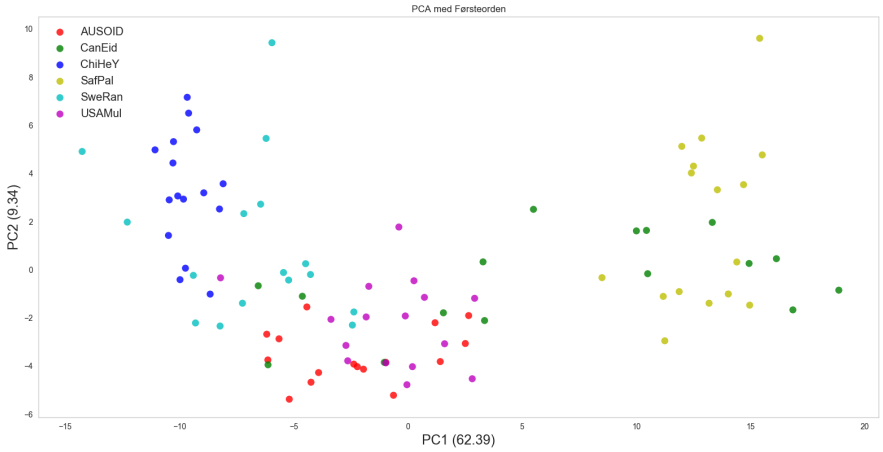
(m)

Figur B2: Plottene viser forklart varians for hver prinsipalkomponent for hver av egenskapsblokkene ((a)-(l)). I den siste figuren (m) er forklart varians for alle egenskapsblokkene plottet sammen. Plottet for egenskapsblokken "pyrad_lbp" (i) er ikke lagt til i (m) da figur (i) ble laget i ettertid.

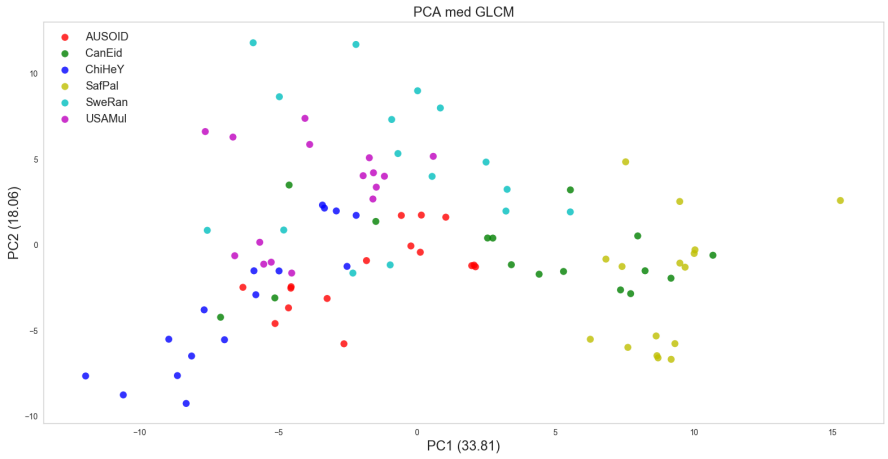
Tabell B1: Tabellen gir en oversikt over hvilke egenskaper som er valgt ut ved hjelp av Random Forest og hvor mange prinsipalkomponenter som er valgt ut med PCA for hver egenskapsblokk. For PCA vises det hvor mye forklart varians hver komponent har, samt summen av den forklarte variansen til de valgte komponentene. Navnene på egenskapene som er hentet ut med Pyradiomics er angitt ved følgende møster: 'egenskap_egenskapsblokk_wavelettransformasjon'. For LBP: 'lbp_mønsternummer_(antall naboer, avstand)'.

Selekteringsmetode	**Egenskaper**
	Førsteorden
RF (4)	'Standard Deviation_first_wavelet-LLL', 'Variance_first_wavelet-LLL', 'Median_first_wavelet-LLL', '10th perc_first_wavelet-HLL'
PCA (4)	0.624 , 0.093, 0.079 , 0.039 (= 0.835)
	GLCM
RF (9)	'ClusterTendency_glcm2_wavelet-HLL', 'ClusterProminence_glcm2_wavelet-LLL', 'ClusterProminence_glcm3_wavelet-LLL', 'ClusterProminence_glcm10_wavelet-LLL', 'ClusterProminence_glcm1_wavelet-LLL', 'AutoCorrelation_glcm15_originals', 'ClusterTendency_glcm3_wavelet-LLL', 'ClusterShade_glcm15_wavelet-LLL', 'ClusterTendency_glcm3_wavelet-HLL'
PCA (6)	0.338, 0.181, 0.155, 0.09, 0.082, 0.053 (= 0.898)
	GLRLM
RF (2)	'LRE_glrlm_wavelet-LLL', 'LRE_glrlm_wavelet-HLH', 'RV_glrlm_wavelet-LLL', 'GLN_glrlm_wavelet-HLH', 'RLN_glrlm_wavelet-LLH', 'LRE_glrlm_wavelet-LLH', 'LRHGLE_glrlm_wavelet-LHH', 'GLN_glrlm_wavelet-LLH', 'LRHGLE_glrlm_wavelet-HHH', 'RV_glrlm_wavelet-HHL'
PCA (6)	0.298, 0.265, 0.13, 0.1, 0.08, 0.064 (= 0.937)
	GLSZM
RF (11)	GLN_glszm_wavelet-LLL', 'SZN_glszm_originals', 'GLN_glszm_wavelet-HLH', 'GLN_glszm_wavelet-LLH', 'ZV_glszm_wavelet-HLH', 'ZV_glszm_wavelet-LLH', 'LAE_glszm_originals', 'LAE_glszm_wavelet-LLH', 'LAE_glszm_wavelet-HLH', 'ZV_glszm_originals', 'ZV_glszm_wavelet-HHL'
PCA (7)	0.327, 0.185, 0.107, 0.087, 0.071, 0.062, 0.053 (= 0.891)
	NGTDM

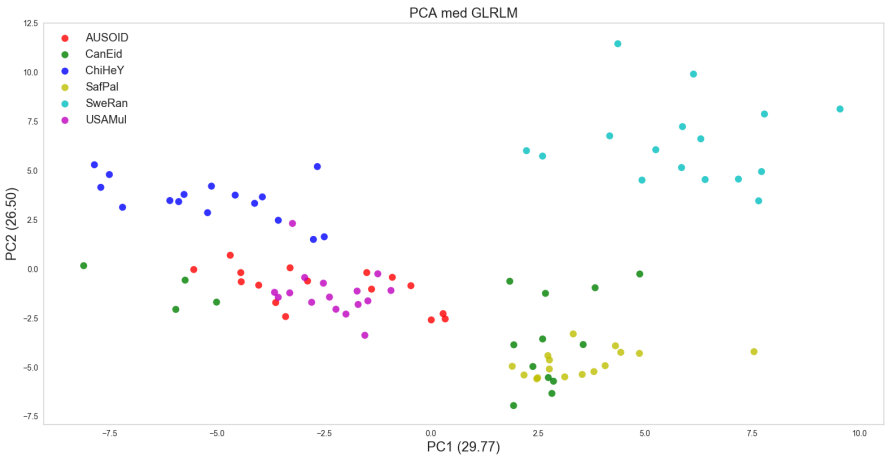
RF (6)	'Busyness_ngtdm_wavelet-LHH', 'Busyness_ngtdm_wavelet-HHH', 'Busyness_ngtdm_originals', 'Busyness_ngtdm_wavelet-HLH', 'Busyness_ngtdm_wavelet-LLH', 'Busyness_ngtdm_wavelet-LLL'
PCA (6)	0.309, 0.222, 0.197, 0.132, 0.102, 0.038 (= 1.0)
GLDM	
RF (8)	'LDE_gldm_wavelet-HLH', 'DN_gldm_wavelet-LLL', 'LDHGLE_gldm_originals', 'DN_gldm_wavelet-LLH', 'SDHGLE_gldm_originals', 'DN_gldm_wavelet-HLH', 'LDE_gldm_wavelet-LLH', 'LDE_gldm_wavelet-LLL'
PCA (6)	0.314 ,0.189, 0.118, 0.109, 0.085, 0.062 (= 0.877)
Pyradiomics - alle	
RF (9)	'LRE_glrlm_wavelet-HLH', 'GLN_glrlm_wavelet-HLH', 'LRE_glrlm_wavelet-LLH', 'LDE_gldm_wavelet-HLH', 'SZN_glszm_originals', 'RLN_glrlm_wavelet-LLH', 'Standard Deviation_first_wavelet-LLL', 'RV_glrlm_wavelet-LLL', 'GLN_glszm_wavelet-LLH'
PCA (6)	0.376, 0.152, 0.108, 0.084, 0.065, 0.053 (= 0.838)
LBP	
RF (4)	'lbp_4_(24,3)', 'lbp_4_(16,2)', 'lbp_0_(4,1)', 'lbp_20_(24,3)'
PCA (3)	0.67, 0.24, 0.05 (= 0.96)
Pyradiomics og LBP	
RF (9)	'lbp_4_(24,3)', 'LRE_glrlm_wavelet-LLL', 'LDHGLE_gldm_originals', 'LRE_glrlm_wavelet-HLH', 'lbp_0_(4,1)', 'Variance_first_wavelet-LLL', 'RV_glrlm_wavelet-LLL', 'lbp_20_(24,3)', 'RLN_glrlm_wavelet-HLH'
PCA (5)	0.382, 0.176, 0.100, 0.079, 0.058 (= 0.795)
AMT500	
RF (5)	17, 19, 2, 18, 16
PCA (2)	0.777 , 0.101 (= 0.878)
AMT2000	
RF (7)	3, 2, 1, 5, 4, 17, 15
PCA (2)	0.823, 0.120 (= 0.943)
AMT2000 - redusert	
RF (4)	21, 22, 23, 27
PCA (2)	0.928, 0.057 (= 0.985)


PCA med Førsteorden
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (9.34)
PC1 (62.39)

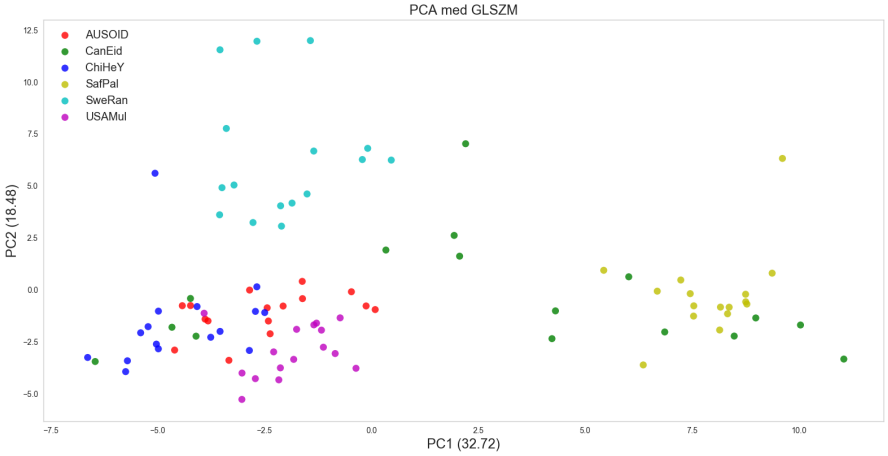
(a)


PCA med GLCM
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (18.06)
PC1 (33.81)

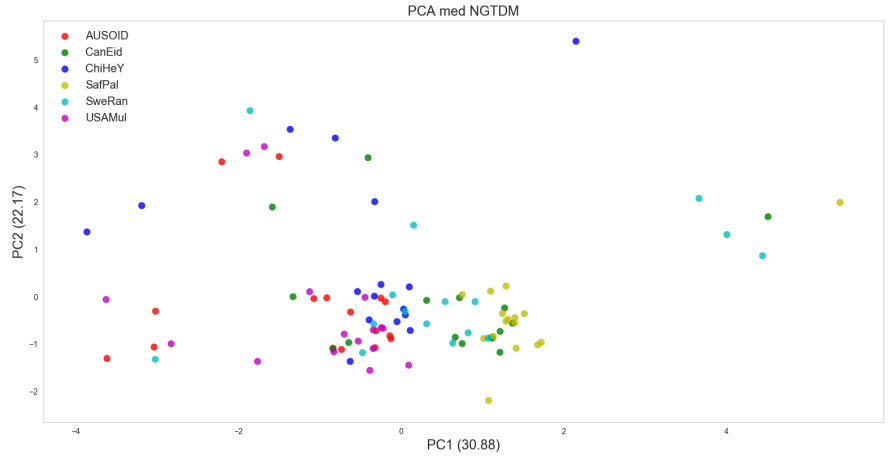
(b)


PCA med GLRLM
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (26.50)
PC1 (29.77)

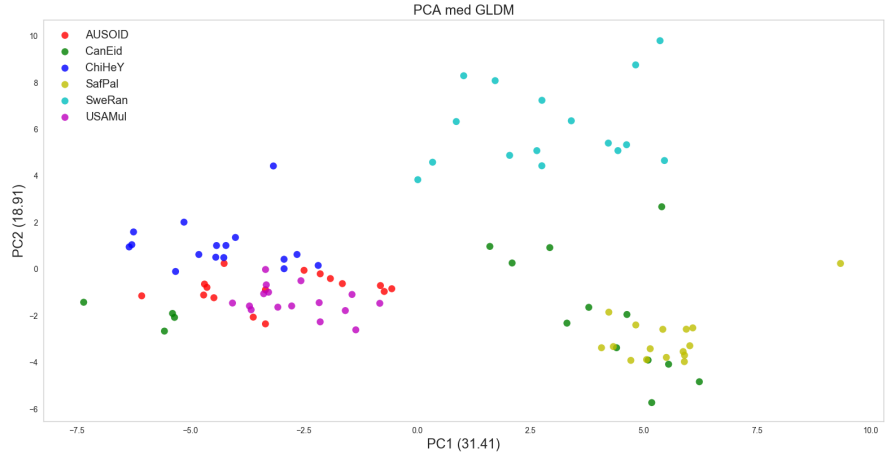
(c)


PCA med GLSZM
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (18.48)
PC1 (32.72)

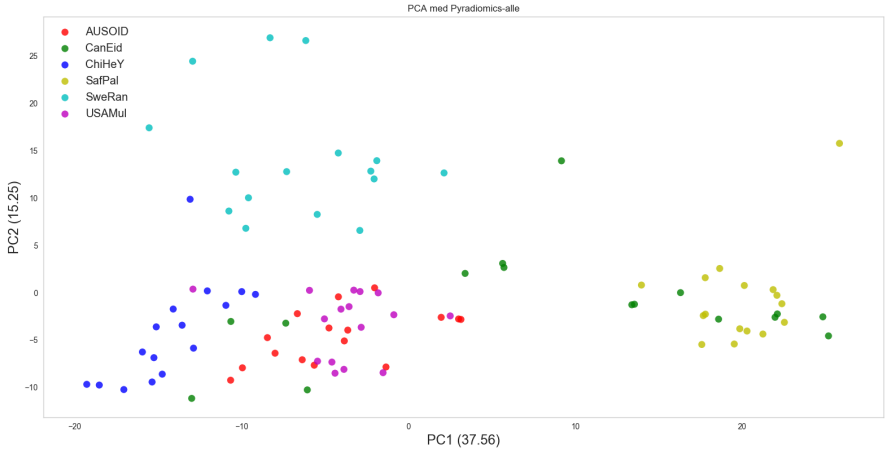
(d)


PCA med NGTDM
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (22.17)
PC1 (30.88)

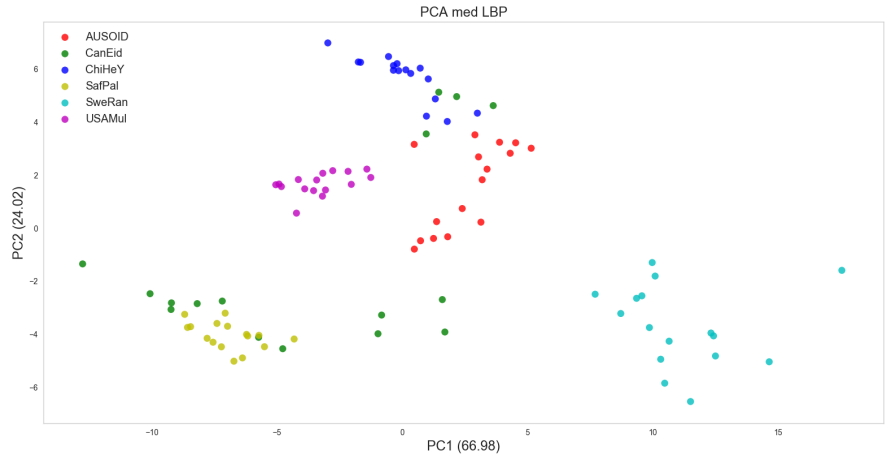
(e)


PCA med GLDM
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (18.91)
PC1 (31.41)

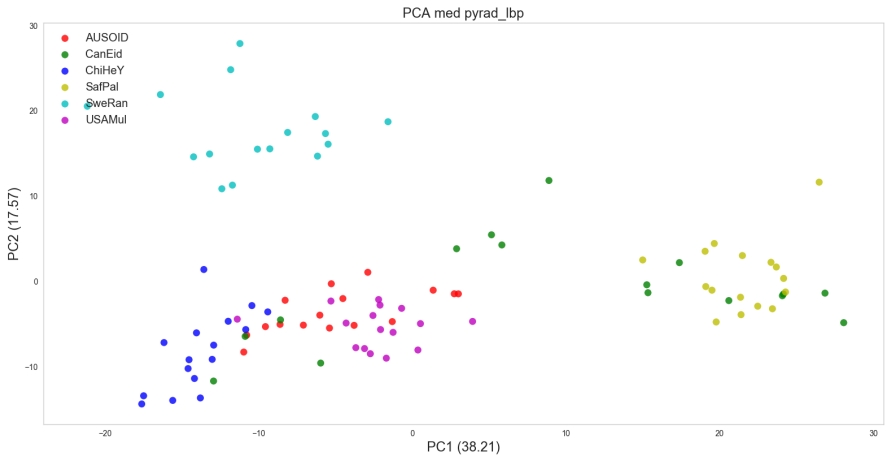
(f)


PCA med Pyradiomics-alle
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (15.25)
PC1 (37.56)

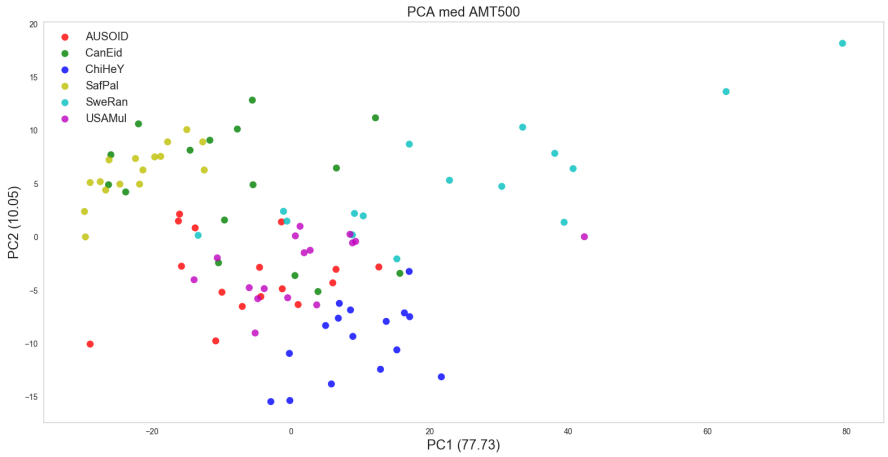
(g)


PCA med LBP
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (24.02)
PC1 (66.98)

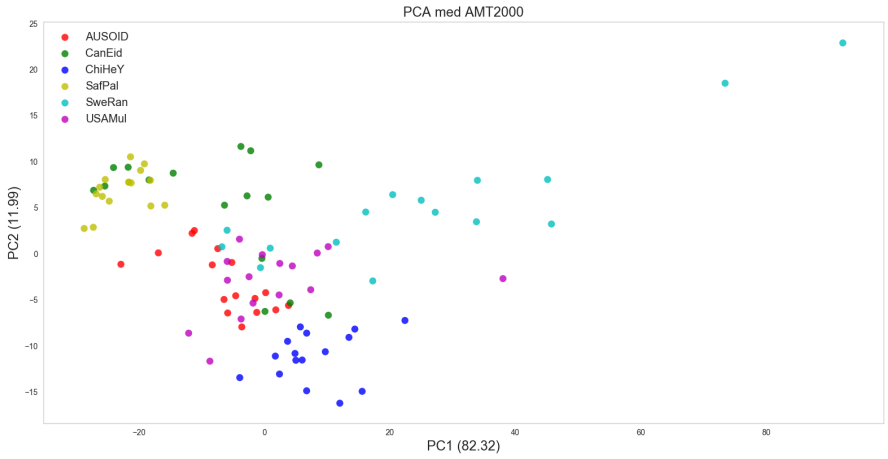
(h)


PCA med pyrad_lbp
AUSOID
CanEid
ChiHeY
SafPal
SweRan
USAMul
PC2 (17.57)
PC1 (38.21)

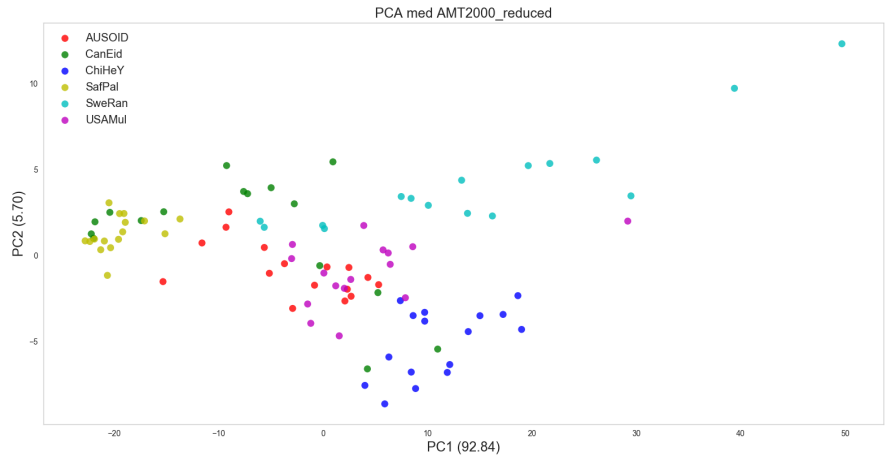
(i)



(j)



(k)



(l)

Figur B3: Figuren viser sprednigsplott med de to første prinsipalkomponentene for hver av egenskapsblokkene ((a)-(l)). På aksetitlene (i parentes) angis hvor mange prosent forklart varians hver av komponentene gir.

Vedlegg C

Her vises resultatene fra den nøstede kryssvalideringen og fra de endelige modellene med tilhørende prediksjoner.

Tabell C1: Tabellen viser gjennomsnittsnøyaktighetene fra den nøstede kryssvalideringen for alle de 12 egenskapsblokkene og for alle klassifikatorene som har blitt benyttet. I den første kolonnen angir tallet i parentes antall egenskaper/antall prinsipalkomponenter som ble valgt ut.

Selekteringsmetode	DT	LR_l1	LR_l2	RFC	SVM	AdaBoost	KNN	GBC
Førsteorden								
RLV (122)	0,70	0,83	0,83	0,82	0,80	0,72	0,53	0,73
RF (4)	0,68	0,68	0,67	0,81	0,74	0,74	0,68	0,70
PCA (4)	0,57	0,60	0,62	0,50	0,56	0,48	0,56	0,54
GLCM								
RLV (107)	0,81	0,81	0,78	0,83	0,79	0,68	0,71	0,79
RF (9)	0,73	0,86	0,87	0,83	0,72	0,71	0,77	0,77
PCA (6)	0,55	0,75	0,77	0,70	0,72	0,54	0,68	0,64
GLRLM								
RLV (64)	0,81	0,82	0,86	0,91	0,81	0,81	0,78	0,88
RF (10	0,86	0,74	0,77	0,91	0,80	0,71	0,59	0,82
PCA (6)	0,69	0,70	0,72	0,79	0,76	0,63	0,78	0,71
GLSZM								
RLV (72)	0,81	0,81	0,82	0,88	0,84	0,76	0,47	0,80
RF (11)	0,72	0,84	0,84	0,85	0,88	0,82	0,73	0,83
PCA (7)	0,55	0,70	0,72	0,69	0,78	0,57	0,61	0,69
NGTDM								
RLV (9)	0,63	0,41	0,41	0,71	0,50	0,55	0,55	0,70
RF (6)	0,67	0,38	0,36	0,77	0,53	0,59	0,69	0,74
PCA (6)	0,41	0,41	0,41	0,45	0,56	0,24	0,41	0,50
GLDM								
RLV (57)	0,83	0,88	0,86	0,90	0,86	0,81	0,80	0,89
RF (8)	0,88	0,86	0,90	0,92	0,88	0,84	0,57	0,88
PCA (6)	0,72	0,60	0,59	0,75	0,74	0,56	0,65	0,64
Pyradiomics - alle								
RLV (431)	0,73	0,89	0,80	0,90	0,83	0,74	0,48	0,84
RF (9)	0,82	0,79	0,86	0,89	0,88	0,78	0,84	0,86
PCA (6)	0,57	0,72	0,76	0,70	0,73	0,56	0,74	0,71
LBP								
RLV (60)	0,86	0,83	0,89	0,84	0,97	0,70	0,83	0,84

RF (4)	0,85	0,82	0,83	0,87	0,84	0,76	0,83	0,81
PCA (3)	0,89	0,78	0,79	0,92	0,94	0,78	0,83	0,77
Alle fra Pyradiomics og LBP								
RLV (491)	0,81	0,86	0,84	0,87	0,84	0,82	0,48	0,87
RF (9)	0,89	0,91	0,94	0,94	0,89	0,88	0,87	0,92
PCA (5)	0,64	0,72	0,71	0,69	0,75	0,56	0,71	0,67
AMT500								
RLV (500)	0,52	0,63	0,69	0,66	0,70	0,54	0,62	0,54
RF (5)	0,56	0,56	0,57	0,66	0,67	0,54	0,54	0,53
PCA (2)	0,52	0,61	0,62	0,55	0,56	0,44	0,53	0,55
AMT2000								
RLV (500)	0,64	0,67	0,76	0,71	0,74	0,60	0,61	0,74
RF (2)	0,64	0,74	0,74	0,79	0,76	0,65	0,72	0,64
PCA (2)	0,60	0,66	0,65	0,59	0,63	0,46	0,66	0,56
AMT2000 - redusert								
RLV (230)	0,60	0,61	0,73	0,63	0,71	0,36	0,58	0,59
RF (4)	0,50	0,68	0,64	0,52	0,60	0,42	0,55	0,49
PCA (2)	0,52	0,60	0,60	0,58	0,62	0,43	0,51	0,52

Tabell C2: Tabellen viser alle standardavvikene tilhørende de gjennomsnittlige nøyaktighetene fra den nøstede kryssvalideringen som er presentert i Tabell C1. I den første kolonnen angir tallet i parentes antall egenskaper/prinsipalkomponenter.

Selekteringsmetode	DT	LR_l1	LR_l2	RFC	SVM	AdaBoost	KNN	GBC
Førsteorden								
RLV (122)	0,09	0,04	0,09	0,11	0,08	0,14	0,08	0,08
RF (4)	0,07	0,11	0,09	0,10	0,06	0,13	0,09	0,13
PCA (4)	0,12	0,12	0,10	0,08	0,13	0,11	0,12	0,09
GLCM								
RLV (107)	0,10	0,14	0,12	0,18	0,10	0,18	0,16	0,12
RF (9)	0,17	0,08	0,07	0,13	0,07	0,17	0,15	0,13
PCA (6)	0,11	0,16	0,17	0,08	0,11	0,09	0,11	0,12
GLRLM								
RLV (64)	0,04	0,04	0,09	0,07	0,08	0,08	0,13	0,08
RF (10	0,04	0,05	0,11	0,06	0,08	0,16	0,06	0,07
PCA (6)	0,06	0,04	0,09	0,08	0,08	0,11	0,11	0,06
GLSZM								
RLV (72)	0,08	0,08	0,04	0,10	0,07	0,05	0,05	0,10
RF (11)	0,05	0,11	0,11	0,04	0,08	0,09	0,05	0,11

PCA (7)	0,08	0,10	0,08	0,06	0,08	0,10	0,11	0,09
NGTDM								
RLV (9)	0,16	0,08	0,08	0,11	0,16	0,09	0,10	0,06
RF (6)	0,12	0,13	0,13	0,10	0,11	0,09	0,19	0,03
PCA (6)	0,08	0,08	0,08	0,11	0,18	0,04	0,10	0,09
GLDM								
RLV (57)	0,06	0,05	0,06	0,11	0,06	0,12	0,08	0,08
RF (8)	0,04	0,06	0,05	0,07	0,08	0,07	0,18	0,07
PCA (6)	0,13	0,08	0,12	0,06	0,13	0,05	0,10	0,06
Pyradiomics - alle								
RLV (431)	0,20	0,05	0,04	0,08	0,07	0,08	0,07	0,11
RF (9)	0,12	0,12	0,09	0,08	0,08	0,11	0,09	0,06
PCA (6)	0,03	0,09	0,08	0,06	0,07	0,14	0,07	0,04
LBP								
RLV (60)	0,08	0,00	0,04	0,06	0,04	0,09	0,00	0,05
RF (4)	0,07	0,10	0,09	0,04	0,04	0,10	0,07	0,04
PCA (3)	0,06	0,04	0,05	0,07	0,02	0,07	0,00	0,17
Pyradiomics og LBP								
RLV (491)	0,11	0,06	0,03	0,07	0,07	0,08	0,07	0,09
RF (9)	0,06	0,06	0,07	0,04	0,04	0,04	0,10	0,06
PCA (5)	0,08	0,11	0,07	0,06	0,08	0,12	0,11	0,06
AMT500								
RLV (500)	0,03	0,06	0,03	0,08	0,09	0,12	0,07	0,08
RF (5)	0,05	0,09	0,09	0,06	0,07	0,06	0,12	0,07
PCA (2)	0,11	0,04	0,03	0,08	0,08	0,11	0,11	0,06
AMT2000								
RLV (500)	0,06	0,06	0,07	0,07	0,07	0,17	0,07	0,08
RF (2)	0,08	0,06	0,06	0,03	0,05	0,09	0,10	0,08
PCA (2)	0,11	0,07	0,06	0,08	0,08	0,11	0,08	0,10
AMT2000 - redusert								
RLV (230)	0,07	0,04	0,10	0,13	0,10	0,14	0,11	0,08
RF (4)	0,13	0,08	0,04	0,07	0,05	0,15	0,07	0,10
PCA (2)	0,09	0,09	0,09	0,06	0,09	0,08	0,06	0,09

Tabell C3: Tabellen viser resultatene fra opptrening av de endelige modellene etter grid search, samt prediksjonene som ble gjort på testsettet. Tabellen er delt opp blokkvis, etter hvilken metode som er brukt til å klassifisere og etter hvilke egenskaper som ble benyttet. Test og Trening angir den nøyaktigheten modellen ga for henholdsvis treningssettet og testsettet. Dersom alle prøvene blir klassifisert riktig, vil prediksjonen bli: [0,0,0,0,1, 1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]. Prediksjonene som er hevet ut med fet skrift markerer gale prediksjoner. I den første kolonnen i tabellen angir tallene i parentes antall egenskaper/prinsipalkomponenter.

Førsteorden				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (122)	0,83	0,72	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**4**,**4**,5,**4**]
	RF (4)	0,58	0,69	[**1**,0,0,0,**3**,**3**,**3**,**3**,**0**,**0**,2,2,3,3,3,3,4,4,4,4,**4**,**4**,5,**4**]
	PCA (4)	0,75	0,57	[0,0,0,0,1,1,**3**,1,2,2,2,2,3,3,3,3,**0**,4,4,**5**,**0**,5,**0**,**1**]
LR_l1	RLV (122)	0,83	0,84	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**4**]
	RF (4)	0,71	0,68	[0,0,0,0,1,1,1,1,**0**,**0**,2,2,3,3,3,3,4,4,**5**,**3**,5,**4**,**4**,**4**]
	PCA (4)	0,75	0,65	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,**0**,4,4,4,5,5,**0**,**4**]
LR_l2	RLV (122)	0,83	0,83	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**4**]
	RF (4)	0,71	0,67	[0,0,0,0,1,1,1,1,**0**,**0**,2,2,3,3,3,3,4,4,**5**,**3**,5,**4**,**4**,**4**]
	PCA (4)	0,75	0,65	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,**0**,4,4,4,5,5,**0**,**4**]
RFC	RLV (122)	0,88	0,81	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**4**,**4**,5,**4**]
	RF (4)	0,79	0,81	[0,0,0,0,1,1,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**4**,**4**,5,**4**]
	PCA (4)	0,63	0,59	[0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,**0**,4,4,**2**,5,**0**,**0**,**4**]
SVM	RLV (122)	0,96	0,81	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,**5**,4,5,5,5,5]
	RF (4)	0,79	0,76	[0,0,0,0,**2**,**3**,1,1,2,2,2,2,3,3,3,3,4,4,**5**,**3**,5,5,5,**4**]
	PCA (4)	0,71	0,59	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,**0**,4,4,4,5,**4**,**0**,**4**]
Ada-Boost	RLV (122)	0,88	0,75	[**5**,**3**,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**4**]
	RF (4)	0,83	0,77	[0,0,0,0,1,1,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,**4**,**4**]
	PCA (4)	0,71	0,55	[0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,3,**0**,4,**5**,**5**,5,5,**4**,**1**]
KNN	RLV (122)	0,83	0,57	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,4,4,**0**,**0**]
	RF (4)	0,63	0,68	[**1**,**1**,0,0,1,1,1,1,**3**,**3**,**3**,**3**,3,3,3,3,4,4,**5**,**5**,5,5,5,**4**]
	PCA (4)	0,63	0,59	[0,**5**,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,**0**,4,4,4,**4**,**4**,**4**,**0**]
GBC	RLV (122)	0,88	0,73	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,**5**,4,5,5,5,**4**]
	RF (4)	0,63	0,73	[0,0,0,0,**2**,**3**,**3**,**3**,**1**,**1**,2,2,3,3,3,3,4,4,4,4,**4**,**4**,5,**4**]
	PCA (4)	0,63	0,55	[0,**5**,0,0,**3**,**3**,**3**,1,2,2,2,2,3,3,3,3,**0**,4,4,4,**4**,**4**,**0**,**1**]
GLCM				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (107)	0,71	0,81	[0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,**5**,4,4,4,5,**4**,5,**4**]
	RF (9)	0,71	0,77	[0,**1**,0,0,1,1,1,1,**4**,**4**,2,2,3,3,3,3,4,4,4,4,**4**,**4**,**4**,**4**]
	PCA (6)	0,67	0,55	[0,0,0,0,**4**,**3**,1,1,2,2,**5**,2,3,3,3,3,**1**,**1**,4,**5**,5,**4**,**4**,**5**]
	RLV (107)	0,96	0,81	[0,0,0,0,**4**,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]

LR_l1	RF (9)	0,83	0,86	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,5,5,5,**4**,**4**]
	PCA (6)	0,92	0,78	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,**4**,5]
LR_l2	RLV (107)	0,96	0,81	[0,0,0,0,**0**,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	0,83	0,88	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,**3**,**5**,5,5,**4**,**4**]
	PCA (6)	0,92	0,78	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,**4**,5]
RFC	RLV (107)	0,92	0,84	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,**4**,5,**4**]
	RF (9)	0,71	0,81	[0,0,0,0,1,1,1,**3**,2,2,2,2,3,3,3,3,**5**,**5**,**5**,**5**,5,5,**4**,**4**]
	PCA (6)	0,92	0,72	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,**2**,5,**4**]
SVM	RLV (107)	0,88	0,83	[0,0,0,0,1,**3**,1,1,2,**1**,2,2,3,3,3,3,4,4,4,**5**,5,5,5,5]
	RF (9)	0,79	0,83	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,**5**,**5**,5,5,**4**,**4**]
	PCA (6)	0,96	0,74	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
Ada-Boost	RLV (107)	0,79	0,78	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,**4**,5,**4**]
	RF (9)	0,71	0,73	[0,0,0,0,1,1,1,1,2,**1**,2,2,**1**,**1**,**1**,**1**,4,4,4,**5**,5,5,5,**4**]
	PCA (6)	0,54	0,52	[0,**1**,0,**1**,1,**3**,1,1,2,**1**,**1**,2,**1**,**1**,**1**,3,**1**,4,4,4,4,5,**1**,**4**,5]
KNN	RLV (107)	0,92	0,76	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,**5**,4,**4**,5,5,5]
	RF (9)	0,46	0,78	[0,0,**2**,**2**,1,**3**,**2**,**2**,**0**,**0**,**0**,**0**,3,3,3,3,4,4,**5**,**5**,5,5,**4**,**4**]
	PCA (6)	0,88	0,70	[0,0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,**1**,4,4,4,4,5,**0**,5,5]
GBC	RLV (107)	0,71	0,84	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**5**,**3**,**3**,**2**,5,**4**,**4**,**4**]
	RF (9)	0,67	0,80	[0,**1**,0,0,1,1,**3**,**3**,2,**1**,2,2,3,3,3,3,4,4,**5**,**5**,5,5,**4**,**4**]
	PCA (6)	0,88	0,67	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,**1**,4,4,**2**,**2**,5,5]
			GLRLM	
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (64)	1,00	0,83	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (10)	1,00	0,90	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,92	0,73	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**2**,**2**,5,5]
LR_l1	RLV (64)	1,00	0,84	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (10)	0,83	0,76	[0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,75	0,72	[0,**1**,0,**1**,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
LR_l2	RLV (64)	0,96	0,85	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,3,**1**,4,4,4,5,5,5,5]
	RF (10)	0,96	0,79	[0,0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,75	0,72	[0,**1**,0,**1**,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
RFC	RLV (64)	1,00	0,92	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (10)	1,00	0,92	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,92	0,79	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**2**,5,5]
SVM	RLV (64)	1,00	0,86	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (10)	1,00	0,88	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,83	0,79	[0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RLV (64)	0,96	0,77	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**0**]

Ada-Boost	RF (10)	0,96	0,76	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**1**]
	PCA (6)	0,71	0,71	[0,0,0,0,1,1,1,1,2,2,2,2,**1**,**1**,**1**,3,4,4,4,4,**2**,**2**,**0**,**0**]
KNN	RLV (64)	1,00	0,77	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (10)	0,96	0,64	[**2**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,92	0,79	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,5,5]
GBC	RLV (64)	0,92	0,90	[0,0,0,0,1,1,1,1,2,2,2,2,3,**1**,3,3,4,4,4,4,5,5,5,**0**]
	RF (10)	0,96	0,83	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**0**]
	PCA (6)	0,75	0,74	[0,0,0,0,1,1,1,**3**,2,2,2,2,3,3,3,3,**1**,4,4,4,**2**,**2**,**0**,**0**]
GLSZM				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (72)	0,88	0,82	[0,0,0,0,**4**,**4**,**4**,**1**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (11)	0,71	0,75	[0,0,0,0,4,4,4,1,2,2,2,2,1,1,1,1,4,4,4,4,5,5,5,5]
	PCA (7)	0,71	0,58	[**5**,0,0,0,**3**,**3**,**3**,1,2,2,**5**,2,3,3,3,3,**1**,4,4,4,5,**2**,5,5]
LR_l1	RLV (72)	0,96	0,81	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (11)	0,96	0,85	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (7)	0,83	0,69	[0,0,0,0,1,1,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**2**,**2**,5,5]
LR_l2	RLV (72)	0,96	0,82	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**1**,4,4,4,5,5,5,5]
	RF (11)	0,96	0,85	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (7)	0,88	0,71	[0,0,0,0,1,1,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,**2**,5,5]
RFC	RLV (72)	0,92	0,88	[**5**,0,0,0,1,1,**3**,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (11)	0,88	0,84	[**5**,0,0,0,1,**3**,1,1,2,2,2,2,3,**1**,3,3,4,4,4,4,5,5,5,5]
	PCA (7)	0,92	0,73	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,**2**,5,5]
SVM	RLV (72)	0,96	0,85	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (11)	0,88	0,88	[**5**,0,0,0,1,1,1,1,2,2,2,**1**,3,3,3,3,4,4,4,**1**,5,5,5,5]
	PCA (7)	0,83	0,79	[0,0,0,0,**1**,3,3,3,2,2,2,2,3,3,3,3,4,4,4,4,5,**2**,5,5]
Ada-Boost	RLV (72)	0,75	0,80	[0,0,**1**,**1**,1,1,**3**,**3**,2,2,2,2,3,**1**,3,3,**1**,4,4,4,5,5,5,5]
	RF (11)	0,88	0,82	[0,0,0,0,1,1,1,1,2,2,2,**0**,3,**1**,3,3,**1**,4,4,4,5,5,5,5]
	PCA (7)	0,50	0,57	[0,0,**1**,0,1,**3**,**3**,**3**,**5**,2,2,2,**1**,3,**1**,3,**2**,4,4,**2**,**0**,**2**,**0**,5]
KNN	RLV (72)	0,79	0,49	[0,0,0,0,1,1,**3**,1,2,2,2,2,**1**,3,3,3,4,4,4,4,**0**,**2**,**1**,5]
	RF (11)	0,88	0,75	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,**1**,3,**1**,4,4,4,4,5,5,5,5]
	PCA (7)	0,75	0,70	[0,0,0,0,1,**3**,**3**,1,2,2,2,**0**,3,**1**,3,3,4,4,4,4,**0**,**2**,5,5]
GBC	RLV (72)	0,88	0,88	[0,0,0,0,1,**3**,**3**,1,2,2,2,2,3,**1**,3,3,4,4,4,4,5,5,5,5]
	RF (11)	0,96	0,85	[0,0,0,0,1,1,1,1,2,2,2,2,3,**5**,3,3,4,4,4,4,5,5,5,5]
	PCA (7)	0,92	0,72	[0,0,0,0,1,1,1,1,2,2,2,2,3,**1**,3,3,**2**,4,4,4,5,5,5,5]
NGTDM				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
	RLV (9)	0,79	0,68	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**2**,4,**3**,4,5,**0**,5,**0**]
	RF (6)	0,75	0,71	[**5**,0,0,0,1,1,1,1,2,2,**0**,2,3,3,3,3,**2**,4,**3**,4,5,**0**,5,**0**]

DT				
	PCA (6)	0,83	0,46	[0,0,0,0,1,1,1,1,**4**,2,2,**0**,3,**4**,3,3,**5**,4,4,4,5,5,5,5]
LR_l1	RLV (9)	0,75	0,47	[0,**5**,0,0,**3**,**3**,1,1,2,2,**0**,2,3,3,3,3,3,**2**,4,4,4,5,5,**2**,5]
	RF (6)	0,71	0,39	[0,**5**,0,**5**,**3**,**3**,1,1,2,2,**5**,2,3,3,3,3,3,3,4,4,4,**2**,5,5,5]
	PCA (6)	0,67	0,46	[0,**5**,0,**5**,**3**,**3**,**3**,1,2,2,**0**,2,3,3,3,3,3,**2**,4,4,4,5,5,**2**,5]
LR_l2	RLV (9)	0,67	0,43	[0,**5**,0,**5**,**3**,**3**,**3**,1,2,2,**0**,2,3,3,3,3,3,**2**,4,4,4,5,5,**2**,5]
	RF (6)	0,58	0,40	[0,**5**,0,**5**,**3**,**3**,**3**,**3**,2,2,**5**,2,3,3,3,3,3,**3**,4,4,4,**2**,0,5,5]
	PCA (6)	0,67	0,46	[0,**5**,0,**5**,**3**,**3**,**3**,1,2,2,**0**,2,3,3,3,3,3,**2**,4,4,4,5,5,**2**,5]
RFC	RLV (9)	0,83	0,75	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,3,**2**,4,4,4,4,5,**0**,5,**0**]
	RF (6)	0,83	0,77	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,3,**2**,4,4,4,4,5,**0**,5,**0**]
	PCA (6)	0,79	0,48	[0,0,0,0,**3**,**3**,1,1,**4**,2,2,2,3,**4**,3,3,**2**,4,4,4,5,5,5,5]
SVM	RLV (9)	0,92	0,50	[0,0,0,0,1,1,1,1,2,2,2,2,3,**4**,3,3,**2**,4,4,4,5,5,5,5]
	RF (6)	0,71	0,57	[**2**,0,0,0,1,1,1,1,2,**0**,**5**,2,3,3,3,3,3,**2**,4,4,**2**,**0**,5,5,**0**]
	PCA (6)	0,54	0,56	[**5**,0,0,0,1,1,**2**,1,2,**5**,**5**,**5**,**4**,**4**,**1**,3,**5**,**5**,**5**,4,5,5,5,5]
Ada-Boost	RLV (9)	0,79	0,63	[**5**,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,**0**,4,4,4,4,5,**2**,5,**2**]
	RF (6)	0,63	0,65	[0,0,0,0,**3,3,3,3**,2,2,2,2,**4**,**4**,3,3,**2**,4,4,4,4,5,**2**,5,**2**]
	PCA (6)	0,46	0,38	[**5**,**4**,0,**4**,1,**3**,1,1,**4**,2,**4**,2,3,**4**,**2**,3,**2**,4,4,4,**4**,**4**,2,4]
KNN	RLV (9)	0,71	0,55	[0,0,**2**,0,**3**,**3**,1,1,2,**0,0,0**,3,3,3,3,**2**,4,4,4,5,5,5,5]
	RF (6)	0,75	0,70	[**2**,0,0,0,1,**3**,1,1,2,**0**,**0**,**0**,3,3,3,3,4,4,4,4,5,5,5,**0**]
	PCA (6)	0,83	0,48	[0,0,0,0,1,1,1,1,2,2,**5**,**0**,3,**4**,3,3,**5**,4,4,4,5,5,5,5]
GBC	RLV (9)	0,92	0,73	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,**0**,5,**0**]
	RF (6)	0,75	0,74	[**2**,0,0,0,1,**3**,1,1,2,**0**,**0**,2,3,3,3,3,4,4,4,4,5,**0**,5,**0**]
	PCA (6)	0,75	0,53	[0,0,0,0,1,1,1,1,**4**,2,**4**,**5**,3,**4**,3,3,**1**,4,4,4,5,**4**,5,5]
GLDM				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (57)	0,96	0,83	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (8)	0,92	0,88	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**3**]
	PCA (6)	0,79	0,71	[0,0,0,0,1,**3**,1,1,**0**,2,**0**,2,3,3,3,3,4,4,4,4,**0**,**2**,5,5]
LR_l1	RLV (57)	1,00	0,88	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (8)	1,00	0,88	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,71	0,63	[**5**,**5**,0,**5**,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
LR_l2	RLV (57)	1,00	0,88	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (8)	1,00	0,91	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,67	0,63	[0,**5**,0,**5**,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,5,5]
RFC	RLV (57)	1,00	0,91	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (8)	1,00	0,90	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,83	0,76	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,**2**,4,4,4,**0**,**2**,5,5]
	RLV (57)	1,00	0,90	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (8)	1,00	0,93	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]

SVM				
	PCA (6)	0,75	0,76	[0,0,0,0,**3**,**3**,1,**3**,2,2,2,2,3,3,3,3,**1**,4,4,4,**0**,**2**,5,5]
Ada-Boost	RLV (57)	0,75	0,84	[**1**,**1**,**1**,**1**,1,1,1,1,2,2,2,2,3,3,3,3,**1**,4,4,4,5,5,5,**1**]
	RF (8)	0,92	0,85	[**1**,**1**,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,50	0,60	[**5**,**5**,**5**,**5**,**0**,**3**,**3**,**0**,**0**,2,**0**,2,3,3,3,3,4,4,4,4,**0**,**1**,5,5]
KNN	RLV (57)	0,88	0,80	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**5**,4,4,4,**4**,**4**,5,5]
	RF (8)	0,75	0,68	[**1**,**1**,0,0,1,1,**0**,**0**,2,2,**3**,**3**,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,79	0,68	[**5**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**1**,4,4,4,**0**,**0**,5,**0**]
GBC	RLV (57)	0,96	0,88	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**0**]
	RF (8)	0,92	0,94	[0,0,0,0,0,1,1,**5**,**5**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,79	0,66	[0,0,0,0,0,1,1,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**1**,**0**,5]
Pyradiomics - alle				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (431)	0,88	0,74	[0,0,0,0,**4**,**4**,**4**,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	0,96	0,82	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,**2**,4,4,4,5,5,5,5]
	PCA (6)	0,75	0,60	[**5**,**5**,**5**,**5**,1,**3**,1,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
LR_l1	RLV (431)	1,00	0,88	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	1,00	0,82	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,83	0,75	[0,0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
LR_l2	RLV (431)	1,00	0,86	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	1,00	0,89	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,92	0,77	[0,0,0,0,0,**3**,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
RFC	RLV (431)	1,00	0,90	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	1,00	0,89	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,83	0,73	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,**0**,**0**]
SVM	RLV (431)	1,00	0,88	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	1,00	0,97	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,75	0,77	[0,0,0,0,0,**3**,1,**3**,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**1**,5,**0**]
Ada-Boost	RLV (431)	0,96	0,81	[**1**,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	0,71	0,82	[0,**1**,**1**,0,1,**3**,**3**,1,2,2,2,2,3,3,3,3,4,4,4,4,0,0,5,0]
	PCA (6)	0,58	0,64	[0,**1**,0,**1**,**4**,1,1,**3**,2,2,2,2,3,**1**,3,**1**,4,4,4,**2**,**2**,5,**0**,**4**]
KNN	RLV (431)	0,88	0,49	[0,0,0,0,0,1,1,**3**,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**2**,5,5]
	RF (9)	0,92	0,83	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,**1**,5,**0**]
	PCA (6)	0,88	0,71	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,5,**0**]
GBC	RLV (431)	0,96	0,84	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**0**]
	RF (9)	1,00	0,85	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (6)	0,79	0,70	[0,0,0,0,0,1,1,1,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,**0**,**0**]
LBP				

	Egenskaper	Test	Trening	Prediksjoner
DT	RLV (60)	0,79	0,85	[0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1**,3,3,4,4,4,4,5,5,5,**0**]
	RF (4)	0,67	0,85	[0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,**0**,**0**,5,**0**]
	PCA (3)	0,71	0,89	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,5,5,5,**0**]
LR_l1	RLV (60)	0,83	0,84	[0,0,0,0,**3,3,3,3,**2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (4)	0,79	0,83	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,3,**1**,3,4,4,4,4,5,5,5,5]
	PCA (3)	0,79	0,82	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,**0**]
LR_l2	RLV (60)	0,88	0,93	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (4)	0,83	0,83	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,3,**1**,3,4,4,4,4,5,5,5,5]
	PCA (3)	0,79	0,82	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,**0**]
RFC	RLV (60)	0,79	0,83	[0,0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,5,5,5,**0**]
	RF (4)	0,67	0,88	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,**0**,**0**,5,**0**]
	PCA (3)	0,75	0,92	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,**1**,3,3,4,4,4,4,5,5,5,**0**]
SVM	RLV (60)	0,88	0,97	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (4)	0,88	0,89	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (3)	0,83	0,96	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
Ada-Boost	RLV (60)	0,75	0,78	[0,0,0,0,0,1,1,1,**3**,2,2,2,2,3,**1**,**1**,3,**1**,4,4,4,5,**0**,5,**0**]
	RF (4)	0,79	0,77	[0,0,0,0,0,1,1,1,**3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,**0**,**0**,5,5]
	PCA (3)	0,71	0,79	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,3,**1**,3,4,4,4,4,5,**1**,5,**1**]
KNN	RLV (60)	0,83	0,83	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,3,**1**,3,4,4,4,4,5,5,5,5]
	RF (4)	0,79	0,86	[0,0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**0,0**,5,**0**]
	PCA (3)	0,79	0,83	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,**0**]
GBC	RLV (60)	0,75	0,88	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1**,3,3,4,4,4,4,5,**1**,5,**0**]
	RF (4)	0,67	0,84	[0,0,0,1,1,**3,3,3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,5,**0**,5,**0**]
	PCA (3)	0,88	0,84	[0,0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,**1**,3,4,4,4,4,5,5,5,5]
Pyradiomics og LBP				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (491)	0,79	0,82	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,5,5,5,5]
	RF (9)	0,79	0,92	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,5,5,5,5]
	PCA (5)	0,96	0,66	[0,0,0,0,0,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,5,**0**,5,5]
LR_l1	RLV (491)	0,96	0,88	[0,0,0,0,0,1,1,1,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	0,96	0,92	[0,0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (5)	0,67	0,74	[0,**1**,0,**1**,**3,3,3,3**,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,5,5]
LR_l2	RLV (491)	1,00	0,88	[0,0,0,0,0,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	1,00	0,96	[0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (5)	0,71	0,73	[0,**1**,0,**1**,1,**3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**0**,**0**,5,5]
	RLV (491)	1,00	0,89	[0,0,0,0,0,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	0,79	0,94	[0,0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1**,**1**,3,4,4,4,4,5,5,5,5]

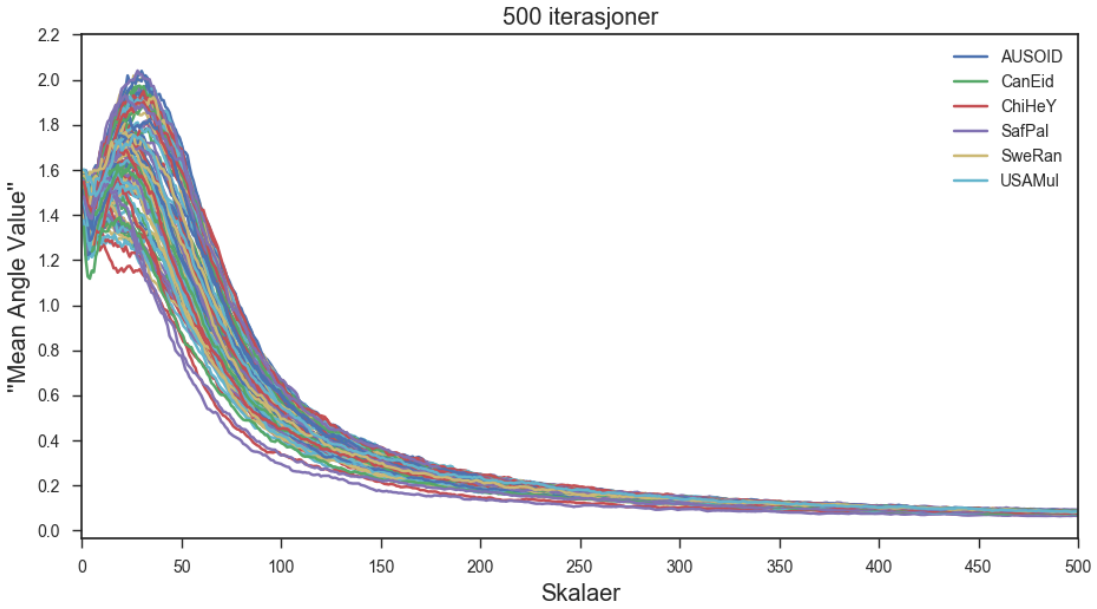
RFC				
	PCA (5)	0,92	0,74	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,**0**,5,**0**]
SVM	RLV (491)	1,00	0,84	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (9)	0,96	0,93	[0,0,0,0,1,**3**,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (5)	0,71	0,79	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**1**,5,**0**]
Ada-Boost	RLV (491)	0,96	0,84	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**0**]
	RF (9)	0,83	0,89	[**1**,**1**,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]
	PCA (5)	0,54	0,55	[0,**1**,0,**1**,1,**3,3,3,1**,2,**1**,2,3,3,3,3,4,4,4,4,**1,1,0,1**]
KNN	RLV (491)	0,88	0,49	[0,0,0,0,1,1,**3**,1,2,2,2,2,3,3,3,3,4,4,4,4,**0,2**,5,5]
	RF (9)	0,83	0,85	[0,0,0,0,1,1,1,1,2,2,2,2,**1**,3,**1**,3,4,4,4,4,**0,0**,5,5]
	PCA (5)	0,88	0,74	[0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,**0**,**0**,5,**0**]
GBC	RLV (491)	0,75	0,86	[0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,**3**,1,1,**3**,4,4,4,4,5,5,5,**0**]
	RF (9)	0,75	0,91	[0,0,0,0,1,**3,3,3**,2,2,2,2,3,**1,1**,3,4,4,4,4,5,5,5,**0**]
	PCA (5)	0,75	0,70	[0,0,0,0,1,1,1,1,2,2,2,2,3,**1**,3,**1**,4,4,4,4,**0,0,0,0**]
			AMT500	
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (500)	0,58	0,58	[**5,5,3,5,3,3,3,3**,2,2,2,2,**5**,3,3,3,4,4,4,4,5,**4**,5,5]
	RF (5)	0,58	0,59	[**5,5,3,5,3,3,3,3**,2,2,2,2,**5**,3,3,3,4,4,4,4,5,**4**,5,5]
	PCA (2)	0,50	0,58	[**5**,**1**,**5**,**5**,**4**,**3**,1,1,**0**,2,2,2,3,3,3,**1**,4,4,4,4,**0,2,0,0**]
LR_l1	RLV (500)	0,63	0,68	[**4,3,3,3,4,3,3,3**,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,**2**]
	RF (5)	0,54	0,61	[**4,5,3,4,3,3,3,3,**2,2,2,2,**5**,3,3,3,4,4,4,4,5,**4**,5,**4**]
	PCA (2)	0,54	0,64	[**5**,**1**,0,0,**4**,**3**,1,**3,5**,2,2,2,3,3,3,**1**,4,4,4,4,**4,2,0,0**]
LR_l2	RLV (500)	0,79	0,71	[**1**,**1**,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,5,5,5,5]
	RF (5)	0,54	0,61	[**4,5,3,4,3,3,3,3**,2,2,2,2,**5**,3,3,3,4,4,4,4,5,**4**,5,**4**]
	PCA (2)	0,54	0,65	[**5,1**,0,0,**4,3**,1,**3,5**,2,2,2,3,3,3,**1**,4,4,4,4,**4**,2,**0**,**0**]
RFC	RLV (500)	0,79	0,67	[0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,**4,2**,5,5]
	RF (5)	0,54	0,65	[**5,5,3,5,3,3,3,3**,2,2,2,2,**5**,3,3,3,4,4,4,4,5,**4**,5,**0**]
	PCA (2)	0,67	0,57	[**5**,**1**,0,0,**4**,**3**,1,**3,5**,2,2,2,3,3,3,3,4,4,4,4,5,**2,0**,5]
SVM	RLV (500)	0,79	0,70	[**1**,**1**,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,5,5,5,5]
	RF (5)	0,58	0,68	[**4,5,3**,0,**3,3,3,3**,2,2,2,2,**5**,3,3,3,4,4,4,4,5,**4**,5,**4**]
	PCA (2)	0,71	0,61	[**1**,**1**,0,0,1,**3**,1,1,**5**,2,2,2,3,3,3,**1**,4,4,4,4,5,**2,0**,5]
Ada-Boost	RLV (500)	0,67	0,56	[**1,1,1**,0,1,**3**,1,**3**,2,2,2,2,3,3,3,3,3,4,**5**,4,4,5,5,**1**,**1**]
	RF (5)	0,63	0,55	[**5,1,1,5**,1,1,1,1,2,2,2,2,**5**,1,1,1,4,4,4,4,5,**4**,5,5]
	PCA (2)	0,63	0,47	[**4,4**,0,**1**,1,**3**,1,1,**1**,2,2,2,**0**,3,3,**1**,4,4,4,4,5,**2**,**0**,5]
KNN	RLV (500)	0,67	0,65	[**4**,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,**1**,**0**,4,**5**,4,4,**1**,**1**,5,5]
	RF (5)	0,50	0,64	[**4,5,3,4,3,3,3,3**,2,2,2,2,**5**,3,3,3,4,4,4,4,**0**,**4**,5,**4**]
	PCA (2)	0,58	0,56	[**5,1,1,1**,1,**3**,1,**3**,2,2,2,2,3,3,3,3,4,4,4,4,**2,2,0,0**]
	RLV (500)	0,71	0,60	[**1**,0,**1**,0,1,**3**,1,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,5,**4**,5,**1**]
	RF (5)	0,54	0,56	[**1,5,1,1,3**,1,**3**,1,2,2,2,2,**5**,3,3,**1**,4,4,4,4,**1,4**,5,**0**]

GBC				
	PCA (2)	0,54	0,55	[**5**,0,**5**,**5**,1,**0**,1,**3**,**0**,2,2,2,3,**1**,**1**,3,4,4,4,4,**4**,**2**,**0**,5]
AMT2000				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (500)	0,50	0,65	[0,0,0,**5**,1,**3**,**3**,**3**,2,**4**,2,2,3,**1**,3,**1**,4,**1**,4,4,**1**,**1**,**1**,**0**]
	RF (2)	0,54	0,68	[0,**5**,0,**5**,**3**,**3**,**3**,**3**,2,2,2,2,2,**1**,3,3,3,4,**0**,4,4,**0**,**0**,5,**0**]
	PCA (2)	0,63	0,60	[**5**,0,0,0,1,**3**,**3**,**3**,**5**,2,2,2,3,3,3,**1**,4,4,4,4,**4**,**2**,5,**2**]
LR_l1	RLV (500)	0,71	0,70	[0,**3**,**3**,**3**,**4**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,5,5]
	RF (2)	0,63	0,74	[0,0,0,**1**,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**0**,**2**,**1**,**0**]
	PCA (2)	0,54	0,66	[0,1,1,0,4,3,3,3,5,2,2,2,3,3,3,1,4,4,4,4,4,2,5,2]
LR_l2	RLV (500)	0,79	0,76	[0,0,**1**,0,1,**3**,**3**,**3**,2,2,2,2,3,**1**,3,3,4,4,4,4,5,5,5,5]
	RF (2)	0,63	0,74	[0,0,0,**1**,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**0**,**2**,**1**,**0**]
	PCA (2)	0,54	0,65	[0,**1**,**1**,0,**4**,**3**,**3**,**3**,**5**,2,2,2,3,3,3,**1**,4,4,4,4,**4**,**2**,5,**0**]
RFC	RLV (500)	0,71	0,73	[0,0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,**1**,**1**,5,**0**]
	RF (2)	0,58	0,80	[0,0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,**1**,3,3,3,4,**0**,4,4,**0**,**0**,**0**,**0**]
	PCA (2)	0,67	0,65	[0,0,0,0,0,1,**3**,**3**,**3**,**5**,2,2,2,3,3,3,**1**,4,4,4,4,**4**,**2**,5,**2**]
SVM	RLV (500)	0,67	0,77	[0,0,0,0,0,1,**3**,**3**,**3**,**1**,2,2,2,3,**1**,3,**1**,4,4,4,4,5,**2**,5,**0**]
	RF (2)	0,63	0,78	[0,0,0,**1**,**3**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**0**,**0**,**0**,**0**]
	PCA (2)	0,63	0,69	[0,**4**,0,0,1,**3**,**3**,**3**,**5**,2,2,2,3,3,3,3,**1**,4,4,4,**4**,**2**,5,**0**]
Ada-Boost	RLV (500)	0,67	0,67	[0,0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,**1**,**0**,**0**,4,4,**1**,**0**,5,**0**]
	RF (2)	0,58	0,65	[0,0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,**1**,3,3,3,4,**0**,4,4,**0**,**0**,**0**,**0**]
	PCA (2)	0,63	0,49	[**5**,0,0,0,**4**,1,1,1,**5**,2,2,2,**1**,**1**,**1**,**1**,4,4,4,4,5,**2**,5,**2**]
KNN	RLV (500)	0,75	0,63	[0,0,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**2**,**4**,5,**0**]
	RF (2)	0,58	0,76	[0,0,0,0,0,**3**,**3**,**3**,**3**,2,2,2,2,**1**,3,**1**,3,4,4,4,4,**0**,**0**,**0**,**0**]
	PCA (2)	0,71	0,66	[**5**,0,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**2**,**2**,5,**0**]
GBC	RLV (500)	0,67	0,74	[0,0,0,0,0,1,**3**,1,**3**,2,2,2,2,2,3,**0**,3,**1**,4,**1**,4,4,**1**,**1**,5,**0**]
	RF (2)	0,58	0,65	[0,0,0,**5**,**3**,**3**,1,**3**,2,2,2,2,**1**,3,3,3,4,**0**,4,4,**0**,**0**,**0**,**0**]
	PCA (2)	0,58	0,59	[**5**,**5**,0,0,1,**3**,**3**,**3**,**5**,2,2,2,3,3,3,**1**,4,4,4,4,**4**,**2**,5,**0**]
AMT2000 - redusert				
	Egenskaper	**Test**	**Trening**	**Prediksjoner**
DT	RLV (230)	0,67	0,61	[0,0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,**1**,3,**1**,4,4,4,4,**4**,**4**,**4**,**1**]
	RF (4)	0,50	0,54	[**4**,**5**,0,0,1,**3**,1,1,2,**4**,2,2,3,**1**,3,**1**,**5**,**5**,4,**0**,**0**,**0**,5,**0**]
	PCA (2)	0,67	0,53	[0,**1**,0,0,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**2**,**2**,**0**,**2**]
LR_l1	RLV (230)	0,67	0,60	[0,**3**,**3**,**3**,**4**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,**4**,5]
	RF (4)	0,63	0,67	[0,0,0,0,0,**3**,**3**,1,1,2,2,2,2,2,3,**1**,3,3,**1**,**5**,4,**5**,**2**,**4**,5,**4**]
	PCA (2)	0,58	0,60	[0,**1**,**1**,**1**,**4**,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**2**,**2**,5,**2**]
	RLV (230)	0,79	0,73	[0,0,0,**1**,1,**3**,**3**,**3**,2,2,2,2,3,3,3,3,3,4,4,4,4,5,5,**0**,5]
	RF (4)	0,63	0,64	[0,0,0,0,**3**,**3**,1,1,2,2,2,2,2,3,**1**,3,3,**0**,**5**,4,**5**,**2**,**4**,5,**4**]

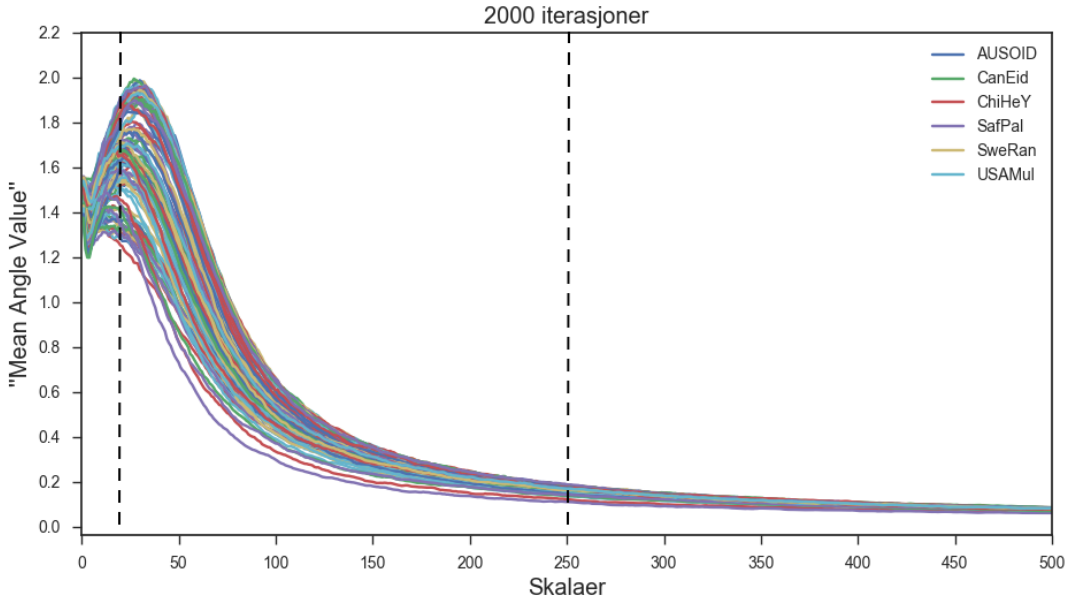
LR_l2				
	PCA (2)	0,58	0,60	[0,**1,1,1,4,3,3,3**,2,2,2,2,3,3,3,3,4,4,4,4,**2,2**,5,**2**]
RFC	RLV (230)	0,79	0,65	[0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,3,3,**1**,4,4,4,4,**2,2**,5,5]
	RF (4)	0,63	0,55	[**4**,**5**,0,0,**3**,**3**,1,1,2,2,2,2,3,3,3,3,3,4,**5**,**5**,4,**1**,**4**,5,**0**]
	PCA (2)	0,67	0,61	[0,**4**,0,0,1,**3,3,3**,2,2,2,2,3,3,3,**1**,4,4,4,4,**2**,**2**,5,**2**]
SVM	RLV (230)	0,79	0,73	[0,0,0,0,**1**,**3,3,3**,2,2,2,2,3,3,3,3,3,4,4,4,4,**2**,5,**0**,5]
	RF (4)	0,71	0,64	[0,0,0,0,**3**,**3**,1,1,2,2,2,2,3,**1**,3,3,4,**5**,4,4,**4**,**4**,5,**0**]
	PCA (2)	0,71	0,63	[0,0,0,0,1,**3,3,3**,2,2,2,2,3,3,3,3,**1**,4,**1**,4,**2**,**2**,5,5]
Ada-Boost	RLV (230)	0,42	0,42	[**5**,0,**1**,0,**4,3**,1,1,**4**,**4**,2,**4**,**1**,1,3,**1**,4,**1**,**5**,4,**4**,**4**,5,5]
	RF (4)	0,42	0,44	[**4,4,4,4,3,3,3,3,4**,2,2,2,3,3,3,3,**5**,**5**,4,4,**4**,**4**,5,**4**]
	PCA (2)	0,50	0,45	[**1,4,1**,0,1,**0,3,3**,2,2,2,2,3,**0**,3,3,**5,5,5,5**,5,5,**1**,5]
KNN	RLV (230)	0,67	0,57	[**4**,0,0,0,1,**3,3,3**,2,2,2,2,3,3,3,3,**5,5**,4,4,**2,2**,5,5]
	RF (4)	0,50	0,53	[0,0,0,0,**3,3,3,3**,2,2,2,2,3,**1**,3,3,**5,5,5,2,2,2**,5,**0**]
	PCA (2)	0,79	0,56	[0,0,0,0,1,**3,3,3**,2,2,2,2,3,3,3,3,4,4,4,4,**2**,**2**,5,5]
GBC	RLV (230)	0,67	0,64	[0,0,0,0,1,**3**,1,**3**,2,2,2,2,3,**1**,3,**1**,**5**,4,4,4,**4**,**4**,5,**1**]
	RF (4)	0,54	0,52	[**4**,0,**4**,0,**3,3,3,3**,2,2,2,2,3,3,3,3,4,**5**,**5**,4,**1**,**4**,5,**0**]
	PCA (2)	0,58	0,55	[**5**,**4**,0,0,1,**3,3,3**,2,2,2,**1**,3,3,3,**1**,4,4,4,4,**2**,**2**,5,**2**]

Vedlegg D

Her vises figurer med AMT-spektre for tilfellet med 500 og 2000 punkter i delmengden, samt en figur som viser verdier av en egenskap fra egenskapsblokken ”LBP”, som bidrar til å skille ut en av klassene.



Figur D1: AMT-spekter med 500 skalaverdier og 500 punkter i delmengden. Plottet viser ”Mean Angle Value” som funksjon av ”Skala” (S) for hver uranprøve i treningssettet.



Figur D2: AMT-spekter med 500 skalaverdier og 2000 punkter i delmengden. Plottet viser ”Mean Angle Value” som funksjon av ”Skala” (S). Området mellom de stiplede linjene i plottet angir det området som synes å ha størst betydning for variasjonen mellom klassene. Dette området er det som i oppgaven blir omtalt som ”AMT2000 - redusert”.

Prøve	Verdi	Prøve	Verdi	Prøve	Verdi
AusOID (0)	83304	ChiHeY (2)	74907	SweRan (4)	61870
AusOID (0)	82420	ChiHeY (2)	75541	SweRan (4)	62078
AusOID (0)	83281	ChiHeY (2)	75767	SweRan (4)	65104
AusOID (0)	83255	ChiHeY (2)	75489	SweRan (4)	63019
AusOID (0)	80343	ChiHeY (2)	77506	SweRan (4)	62878
AusOID (0)	79471	ChiHeY (2)	76965	SweRan (4)	59225
AusOID (0)	77932	ChiHeY (2)	74858	SweRan (4)	63580
AusOID (0)	78145	ChiHeY (2)	75554	SweRan (4)	65600
AusOID (0)	80391	ChiHeY (2)	75157	SweRan (4)	62613
AusOID (0)	83762	ChiHeY (2)	74432	SweRan (4)	66038
AusOID (0)	80727	ChiHeY (2)	75223	SweRan (4)	61323
AusOID (0)	81603	ChiHeY (2)	75373	SweRan (4)	62625
AusOID (0)	81677	ChiHeY (2)	75720	SweRan (4)	61065
AusOID (0)	80617	ChiHeY (2)	74814	SweRan (4)	60547
AusOID (0)	79535	ChiHeY (2)	76395	SweRan (4)	57606
AusOID (0)	79789	ChiHeY (2)	74529	SweRan (4)	53120
CanEid (1)	84311	SAfPal (3)	81726	USAMul (5)	88201
CanEid (1)	88155	SAfPal (3)	82482	USAMul (5)	86364
CanEid (1)	83128	SAfPal (3)	82586	USAMul (5)	88093
CanEid (1)	85078	SAfPal (3)	83846	USAMul (5)	85343
CanEid (1)	79635	SAfPal (3)	84026	USAMul (5)	88157
CanEid (1)	78386	SAfPal (3)	83100	USAMul (5)	87068
CanEid (1)	80433	SAfPal (3)	83833	USAMul (5)	86949
CanEid (1)	77601	SAfPal (3)	82358	USAMul (5)	81062
CanEid (1)	79950	SAfPal (3)	83550	USAMul (5)	86996
CanEid (1)	83177	SAfPal (3)	84892	USAMul (5)	85909
CanEid (1)	80223	SAfPal (3)	83378	USAMul (5)	86506
CanEid (1)	81687	SAfPal (3)	82340	USAMul (5)	85686
CanEid (1)	80742	SAfPal (3)	80839	USAMul (5)	86713
CanEid (1)	82158	SAfPal (3)	82914	USAMul (5)	90370
CanEid (1)	85363	SAfPal (3)	81074	USAMul (5)	89207
CanEid (1)	81080	SAfPal (3)	81721	USAMul (5)	88309

Figur D3: Figuren viser verdiene for egenskapen "$lbp\_0\_(4,1)$" for hver prøve i testsettet. Prøvene fra klassen SweRan (4, Sverige) er farget i rødt og har de laveste verdiene av denne egenskapen, hvilket indikerer at denne egenskapen kan skille ut klassen SweRan.

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

Postboks 5003
NO-1432 Ås
Norway