Norges miljø- og biovitenskapelige universitet

Masteroppgave 2018 30 stp
Fakultet for realfag og teknologi
Professor Cecilia Marie Futsæther

Prediksjon av behandlingsutfall for hode- og halskreft ved bruk av radiomics av PET/CT-bilder

Prediction of treatment outcome of head and throat cancer using radiomics of PET/CT images

Alise Danielle Midtfjord
Miljøfysikk og fornybar energi
Fakultet for realfag og teknologi

Forord

Denne oppgaven er skrevet ved Fakultet for realfag og teknologi ved Norges Miljø- og biovitenskapelige universitet (NMBU) våren 2018. Oppgaven utgjør 30 studiepoeng, og markerer avslutningen på en femårig mastergrad i Miljøfysikk og fornybar energi. Oppgaven har blitt utført i samarbeid med avdelingen for Medisinsk Fysikk ved Oslo universitetssykehus og Fysisk institutt ved Universitet i Oslo.

Jeg ønsker å rette en stor takk til min hovedveileder professor Cecilia Marie Futsæther, for grundige tilbakemeldinger og fantastisk oppfølging hele veien. Jeg vil også takke Oliver Tomic og Aurora Rosvoll Grøndahl, for gode svar på spørsmål og interessante dialoger på våre møter. Takk til professor Eirik Malinen fra UiO for å ha skaffet datasettet, og til Helene Midtfjord og Håkon Wøllo for korrekturlesing og tilbakemeldinger.

Jeg vil også takke familie og venner, som har støttet meg under arbeidet med denne oppgaven. Til slutt ønsker jeg å takke alle mine medstudenter for fem fine år på Ås, og uforglemmelige minner.

Ås, 15.05.2018

Alise Danielle Midtfjord

Sammendrag

Radiomics er konverteringen av digitale bilder til høydimensjonale data, og har oppstått på bakgrunn av konseptet om at medisinske bilder inneholder informasjon om en sykdom eller skade som kommer til syne ved bruk av kvantitative bildeanalyser. Flere studier indikerer at radiomics tilbyr ny informasjon om kreftsvulster, uten bruk av kirurgiske prosedyrer.

Formålet med denne oppgaven var å undersøke om svulstegenskaper beregnet fra intensitetsverdier og den romlige fordelingen av voxler i PET- og CT-bilder kan benyttes for å utvikle modeller for å klassifisere hode- og halskreft etter behandlingsutfall. I denne oppgaven ble det sett på to ulike behandlingsutfall, lokalregionalt tilbakefall eller progresjonsfri overlevelse. I tillegg ble det undersøkt om det er en sammeheng mellom HPV-status og behandlingsutfallet.

Grunnlaget for datasettet som ble analysert i denne oppgaven, består av klinisk informasjon og PET- og CT-bilder av 254 pasienter, som har fått behandling for hode-halskreft ved Oslo universitetssykehus med behandlingsstart i perioden 2007-2013. Radiomics ble brukt for å trekke ut førsteordens-, form- og teksturegenskaper av hode- og halssvulster fra PET- og CT-bildene. I tillegg ble bildene transformert ved å bruke filtre for å fremheve spesifikke egenskaper. Fjorten klassifikasjonsalgoritmer ble testet i kombinasjon med syv egenskapsutvelgere, for å lage klassifikasjonsmodeller. Ytelsen til de endelige modellene ble estimert gjennom nøstet kryssvalidering ved å beregne ROC AUC.

Egenskapene som hadde den høyeste univariate Pearson-korrelasjonen (r_p) med behandlingsutfallet var lengden til den lengste aksen til tumor ($r_p = 0.26$, $p = 0.0003$) og heterogeniteten til tumor (Zone Entropy) ($r_p = 0.26$, $p = 0.0004$) beregnet gjennom en Gray Level Size Zone Matrix (GLSZM). Modellene med den høyeste multivariate prediksjonsytelsen av behandlingsutfallet ($AUC = 0.66 \pm 0.10$) ble bygget på to egenskaper plukket ut av egenskapsutvelgeren ReliefF, og enten PLSR, Logistisk regresjon, LDA eller AdaBoost som klassifiseringsalgoritme. Svulstegenskapene som ReliefF selekterte flest ganger, var egenskaper som beskrev lengden på tumors lengste akse i tillegg til raske verdiendringer i bildet (Busyness). Disse modellene gav høyere AUC enn prediksjon av utfall gitt av modeller laget på kun kliniske faktorer ($AUC = 0.57 \pm 0.12$). Det ble også funnet at av pasientene med positiv HPV-status fikk 76% av pasientene progresjonsfri overlevelse etter behandling, mens dette gjaldt kun 57% av pasientene med negativ HPV-status.

Resultatene viste at egenskapene trukket ut fra PET- og CT-bildene av svulstene kunne gi mer informasjon om behandlingsutfallet enn kliniske faktorer alene. På sikt er det mulig at videre utvikling av fremgangsmåten og prediksjonsmodellene, kan gi forbedring i pasienters behandling ved å kunne tilpasse behandlinger etter prognoser gitt av svulst-egenskaper, beregnet fra medisinske bilder.

Abstract

Radiomics is the conversion of digital images into high dimensional data. It is motivated by the concept that medical images contain information about a disease or injury that can be revealed by quantitative image analyses. Several studies indicate that radiomics can offer new information about cancer tumors, without the need for invasive, medical procedures.

The goal of this work was to investigate if intensity values and the spatial distribution of voxels in PET and CT images can be used to develop models for classification of treatment outcome of head- and throat cancer. Treatment outcome was separated into two classes, localregional relapse and progression free survival. In addition, the relationship between HPV-status and treatment outcome was also investigated.

The dataset, analyzed in this study, consisted of clinical information and tumor features derived from PET and CT images of 254 patients that have received treatment for head- and throat cancer at Oslo University Hospital with treatment beginning in the period 2007-2013. Radiomics was used to extract first order, shape and texture features from the PET and CT images of the tumors. In addition, the images were also transformed using different filters to highlight specific properties of the tumors. Fourteen classifiers, in combination with seven feature selectors, were used to develop different classification models for predicting treatment outcome. The performance of the final models was estimated through nested cross-validation by computing ROC AUC.

The features with the highest univariate Perason correlation (r_p) with treatment outcome were the length of the longest axis of the tumor ($r_p = 0.26$, $p = 0.0003$) and the heterogeneity of the tumor (Zone Entropy) ($r_p = 0.26$, $p = 0.0004$) computed through a Gray Level Size Zone Matrix (GLSZM). The models with the highest multivariate prediction performance ($AUC = 0.66 \pm 0.10$) were obtained using two features selected by the feature selector ReliefF, and by using either PLRS, Logistic regression, LDA or AdaBoost as classifier. These models gave higher AUC than the prediction performance obtained using a model made from only clinical factors ($AUC = 0.57 \pm 0.12$). It was also shown that 76% of the patients with positive HPV-status had progression free survival after treatment, whereas this was only the case for 57% of the patients with negative HPV-status.

The results indicated that tumor features, extracted from PET- and CT-images of the tumors, could provide more information about treatment outcome than clinical factors alone. In the long term, it is possible that futher development of the method and the prediction models, this could provide improved patient treatment, by using customized treatment selected using predictors based on tumor features derived from medical images.

Innhold

Kapittel 1

Introduksjon

Kreft er en betegnelse på 200 forskjellige sykdommer som skyldes ukontrollert celledeling [1]. Kreft er den vanligste dødsårsaken i Norge etter hjerte- og karsykdommer, og hvert år dør over 10 000 nordmenn av kreft [1]. Hode- og halskreft utgjør i dag i overkant av 4% av totalt antall nyoppdaget kreft i Norge, og i 2016 fikk ca. 782 nordmenn kreft i hode- og halsregionen [2]. De største risikofaktorene for å utvikle kreft i hode- og halsområdet er røyking og misbruk av alkohol. I tillegg er HPV-infeksjoner i økende grad en årsak til kreft i dette området [3] [2].

Før behandling av hode- og halskreft er det vanlig å utføre en kombinert PET/CT-skanning, for å få oversikt over omfanget og plassering til tumor [4]. PET/CT brukes før behandling både for å stille diagnosen kreft, skille godartede forandringer fra kreftsvulster, vurdere omfanget av sykdommen, og se etter eventuell spredninger. PET-bilder viser områder med høy metabolsk aktivitet, mens CT-bilder viser absorpsjonen av røntgenstråler i vev, og gir et detaljert anatomisk bilde av området.

Radiomics er konverteringen av digitale bilder til høydimensjonale data, og har oppstått på bakgrunn av konseptet om at medisinske bilder inneholder informasjon om en sykdom eller skade som kommer til syne ved bruk av kvantitative bildeanalyser [5]. Dette er et raskt økende forskningsområde, med et økende antall publikasjoner [6]. Det er vist at radiomics tilbyr ny informasjon om kreftsvulstene, uten bruk av kirurgiske prosedyrer [6] [5].

Flere studier indikerer at egenskaper funnet gjennom radiomics av PET-, CT- og MRI-bilder av kreftsvulster har en bedre evne til å predikere utfallet av kreftbehandlingen, enn kun kliniske faktorer [6]. Disse funnene tyder på at radiomics har et høyt potensiale for å være til hjelp for vurdering av behandlingsmetode for pasienter, og for å tilpasse behandlingen mer individuelt [5]. Dette kan innebære å gi mindre stråledoser til pasienter med gode forutsetninger for å bli frisk, og dermed minske stråleskader på kroppen. Det kan også innebære å gi en kraftigere behandling til pasienter der forutsetningene ikke er like gode.

Maskinlæring er en gren under kunstig intelligens, som involverer selvlærende algo-

ritmer som trekker ut informasjon fra data, for å foreta prediksjoner [7]. Maskinlæring tilbyr en effektiv måte for å finne kunnskap i data ved å gradvis forbedre ytelsen til modeller ved å lære fra datamengden. Klassifikasjon er en gren av maskinlæring, som predikerer klassen til nye forekomster ved bruk av tidligere observasjoner. Innenfor medisin forventes det at maskinlæring vil øke treffsikkerheten til prognoser, forenkle mye av jobben til radiologer og patologer og forbedre diagnostisering [7].

Grunnlaget for datasettet, som ble analysert i denne oppgaven, består av klinisk informasjon og PET- og CT-bilder av pasienter som har fått behandling for hode-halskreft ved Oslo universitetssykehus med behandlingsstart i perioden 2007-2013. Datasettet inneholdt 254 pasienter som var diagnostisert med hode- halskreft, og hadde tatt en PET/CT-skanning fra skuldrene og opp før behandlingsstart. Dette datasettet har blitt brukt til analyse i to tidligere masteroppgaver ved NMBU [8] [9].

Formålet med oppgaven var å undersøke om svulstegenskaper beregnet fra intensitetsverdier og den romlige fordelingen av voxler i PET- og CT-bilder kan benyttes for å utvikle modeller for å klassifisere hode- og halskreft etter behandlingsutfall. I denne oppgaven ble det sett på to ulike behandlingsutfall, lokalregionalt tilbakefall og progresjonsfri overlevelse. Oppgaven har inkludert bruk av radiomics til å trekke ut egenskaper fra bildene, og bruke klassifikasjonsalgoritmer til å lage modeller for å predikere utfallet av kreftbehandlingen. I tillegg ble det undersøkt om det er en sammenheng mellom HPV-status og behandlingsutfallet. Oppgaven er en del av et større prosjekt *Bioradiance*, ledet av Oslo universitetssykehus og Universitetet i Oslo.

Oppgaven starter med å forklare hode- og halskreft og det teoretiske grunnlaget bak PET, CT og maskinlæring til bruk i klassifisering. Dette blir beskrevet i kapittel 2. Kapittel 3 tar for seg metodikken brukt i analysen. Dette innebærer informasjon om datagrunnlaget, bildeanalysen, klassifiseringen og hvordan modellenes ytelse ble vurdert. Resultatene fra analysene blir fremstilt i kapittel 4, og diskutert i kapittel 5. Kapittel 6 og 7 inneholder hhv. konklusjon og vedlegg.

Kapittel 2

Teori

2.1 Kreft

Kreft er en fellesbetegnelse på rundt 200 forskjellige sykdommer/kreftformer som skyldes ukontrollert celledeling [1]. Kroppen bytter hele tiden ut cellene i kroppen. Dette skjer ved at celler dør og nye celler produseres ved å doble sitt arvestoff (DNA) og dele seg i flere celler. Blandt kroppens celler foregår det hele tiden mutasjoner, som er forandringer i cellens arvestoff [10]. Kroppen har et effektivt reparasjonssystem, som kontrollerer og korrigerer skader gjort på arvestoff i celler. Dersom mutasjoner i en celle ikke blir reparert, kan dette føre til at celler deler seg ukontrollert [1]. Etterhvert som cellene fortsetter å dele seg ukontrollert, vil det bli en opphopning av muterte celler som danner en kreftsvulst.

Gjennom blodårer og lymfe kan svulsten spre seg til andre deler av kroppen. Dette kalles metastase, og gjør kreftbehandlingen vanskeligere, ettersom det gir opphav til nye kreftceller og svulster utenfor den opprinnelige svulsten [11].

2.1.1 Hode- og halskreft

Hode- og halskreft utgjør i dag i overkant av 4% av det totalt antall nyoppdagede krefttilfeller i Norge [2]. Dette er en krefttype som oftere rammer menn enn kvinner, og gjennomsnittsalderen er omtrent 64 år. 90% av all kreft i hode- og halsområdet kommer av plateepitelkreft, en av de vanligste formene for hudkreft [12] [13].

Hode- og halskreft begynner som oftest i de ytre cellene i huden som kalles plateepitel, som dekker fuktige områder som i munnen, i nesen eller i svelget [3]. Hode- og halskreft kategoriseres etter området hvor svulsten starter. Hode- og halsområdet kan deles opp i nese/bihule, strupehode (larynx), munnhule (oropharyx) og svelg og spyttkjertler [12]. Svelget deles igjen inn i tre deler, nasopharyx (øverste del), oropharynx (midtre del) og hypopharynx (nederste del).

2.1.2 Risikofaktorer

De to største risikofaktorene for å utvikle kreft i hode- og halsområdet er røyking og misbruk av alkohol [3]. Her har de som både røyker og misbruker alkohol en større risiko enn de som bare bruker én av delene. I tillegg er HPV (human papillomavirus), spesielt type 16, i økende grad en årsak til kreft i enkelte steder av hode- og halsområdet, spesielt da i oropharynx [3][2]. En mulig årsak til dette kan være endringer i folks seksualvaner, som økning i bruk av oralsex.

HPV er den vanligste seksuelt overførbare infeksjonen. Den tilhører en gruppe virus som inkluderer mer enn 100 forskjellige typer [14]. De fleste av disse er ufarlige, men noen kan forårsake kjønnsvorter, celleforandringer i underlivet eller kreft. Det er anslått at over 70% av seksuelt aktive kvinner vil bli smittet av HPV en eller annen gang i løpet av livet, og det antas at HPV er like hyppig hos menn [14].

2.1.3 Behandlinger

Behandlingen av hode- og halskreft kommer an på faktorer som stadium i utviklingen, plassering til tumor og helsen til pasienten [3]. De vanligste behandlingene er operasjon, strålebehandling og bruk av cellegift. Operasjonen vil som oftest bestå av fjerning av svulst og eventuelle lymfeknuter på halsen, dersom det er spredning til disse [2]. Før behandling er en PET/CT-skann vanlig å gjennomføre, for å få oversikt over omfanget og plasseringen til tumor.

2.2 PET/CT

I dag finner vi de fleste PET-skannerne i en kombinert PET/CT-installasjon. Her vil bilder fra PET-skanningen vise opphopninger av celler med høy metabolsk aktivitet, mens CT-bildene gir et bilde av hvor i kroppen disse cellene befinner seg [4]. Etter en skanning i det kombinerte apparatet, kan PET- og CT-bildene vises sammen i et fusjonert bilde.

Både PET- og CT-bilder kan brukes til å lage tredimensjonale bilder av objektet i fokus. I et tredimensjonalt bilde presenteres elementene som *voxler* (volume elements) istedet for pixler. En voxel er et lite volum i et tredimensjonalt bilde, på samme måte som en pixel er lite areal i et todimensjonalt bilde.

2.2.1 Positronsemisjonstomografi

Positronsemisjonstomografi (PET) er en kjernefysisk bildeteknikk, der radionuklider positronemisjon brukes [15]. Radionuklider er atomer med ustabile kjerner, og i

PET-skanninger brukes radionuklider som sender ut beta-stråling i form av positroner [16]. Atomkjernen består av to typer nuklider; protoner og nøytroner. Disse har tilnærmet lik masse, men forskjellige ladninger; protoner er positive mens nøytroner er uten ladning. Antallet protoner som et atom har i kjernen, bestemmer atomnummeret til atomet [15]. Atomer med likt antall protoner, men forskjellig antall nøytroner, kalles isotoper [16]. Noen isotoper er mer ustabile enn andre, og kan være radioaktive. Da vil de sende ut radioaktiv stråling, for å få en mer stabil kjerne. Stabiliteten til atomer avhenger av tiltrekkende og frastøtende krefter mellom nukleonene. Eksempler på radionuklider som kan brukes til PET-skanninger er ^{11}C, ^{13}N og ^{18}F med henhodsvis halveringstider på 20,4 minutter, 9,95 minutter og 110 minutter [15] [16]. Fordelen med denne korte halveringsstiden, er at det radioaktive stoffet ikke blir i kroppen over lang tid. Ulempen er at stoffene ikke kan bli transportert langt unna stedet de blir laget [17].

Positronemisjon

Før PET-skanningen utføres, bindes radionuklider til et molekyl (hovedmolekylet) eller et stoff og danner forbindelser (referert til som radioaktiv markør), som injiseres i kroppen [15]. Hovedmolekylet er et passende, stabilt grunnstoff, som ofte er et annet grunnstoff enn radioisotopen som blir laget til slutt [16]. Som eksempel blir Fluor-18 vanligvis produsert ved å bombardere oksygen-18-beriket vann med protoner.

De radioaktive markørene fordeler seg rundt i vev på bakgrunn av vevets biokjemiske egenskaper. Positronemisjonen skjer ved at et proton i nuklidet konverteres til et nøytron og et positron [17]:

$$ {}^{A}_{Z}X_N \rightarrow {}^{A}_{Z-1}Y_{N+1} + e^+ + v + energi \tag{2.1} $$

der X er det originale nuklidet med Z protoner, N nøytroner og A er det totale antall protoner og nøytroner, Y er det nye nuklidet med et ekstra nøytron og et mindre proton, (e^+) er et positron og (v) er et nøytrino. Et positron er elektronets antipartikkel, altså en partikkel med mange av de samme egenskapene som elektroner, men nøyaktig motsatt ladning [16]. Positronet, sammen med en nøytrino, sendes ut fra nuklidet. Energien som frigjøres i reaksjonen er i form av kinetisk energi til de frigjorte partiklene. Etter at positronet er sendt ut fra nuklidet, kan det bevege seg ut i kroppen. Positroner som frigjøres på denne måten har veldig kort levetid, ettersom de fort støter på elektroner i vevet.

Annihilering

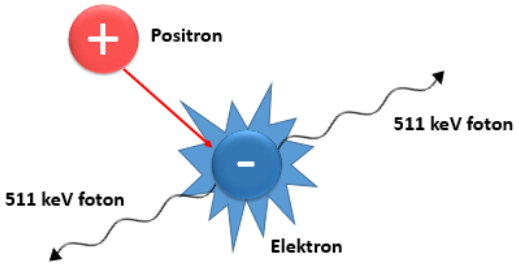
Elektroner og positroner tiltrekker hverandre, på grunn av deres motsatte ladninger, og de vil danne et positronium [16]. Denne tilstanden varer kun rundt 10^{-10} sekunder, før prosessen annihililering (eller tilintentgjøring) begynner [15]. Her blir massen til positronet og elektronet omgjort til elektromagnetisk energi i form av to gammafotoner med energien 0,511 MeV [16]. Dette kan regnes ut ved hjelp av

Einsteins masseenergilov:

$$E = mc^2 = m_e c^2 + m_p c^2 \tag{2.2}$$

der m_e er massen til elektronet, m_p er massen til positronet, c er lysets hastighet og E er den samlede energien til de to fotonene [15]. Fotonene har såpass høy energi, at de har god sjanse til å komme seg ut av kroppen. Det er dermed disse fotonene som blir oppdaget av PET-skanneren, ikke selve positronene.

Ettersom positronet og elektronet er nesten helt i ro før annihilileringen, er den totale bevegelsesmengden nesten null. For å bevare bevegelsesmengden, må to fotoner dannes, og sendes i motsatt retning. Denne prosessen er illustrert i figur 2.1.

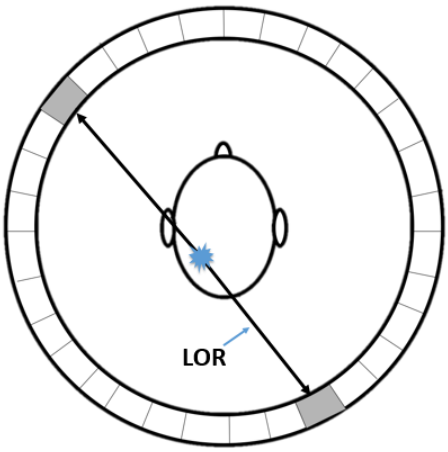


Figur 2.1: *Annihilering. Positroner og elektroner tiltrekker hverandre p.g.a. deres motsatte ladninger. Etter sammenstøtet blir partiklene om til to fotoner som sendes ut i motsatt retning.*

Hvis begge fotonene blir oppdaget av PET-skanneren, vil linjen som kan trekkes mellom de to fotonene (kalt Line of Response (LOR)) gå rett igjennom punktet der annhilieringen skjedde. Punktet der annhilieringen skjedde vil være kun noen millimeter fra der positronemisjonen tok plass, på grunn av positronets korte levetid i vevet [16]. Dette gjør det mulig å se hvor i kroppen det originale radionuklidet befant seg ved emisjonstidspunktet. Disse millimeterene som positronet beveger seg setter en begrensning for oppløsningen til PET-bilder. Andre faktorer som setter en begrensning for oppløsningen til PET-bilder er størrelsene på detektorene og fotonenes avvik fra de 180 ° gradene forskjell mellom to fotoner fra samme annihilering [18]. Derfor har PET-bilder dårligere maksimal oppløsning enn f.eks. CT-bilder og MRI-bilder [16]. Størrelsene på voxlene i PET-bildene i denne oppgaven er ca. $3 \times 3 \times 2\,mm^3$, mens størrelsen på voxlene i CT-bildene er ca. $1 \times 1 \times 2\,mm^3$.

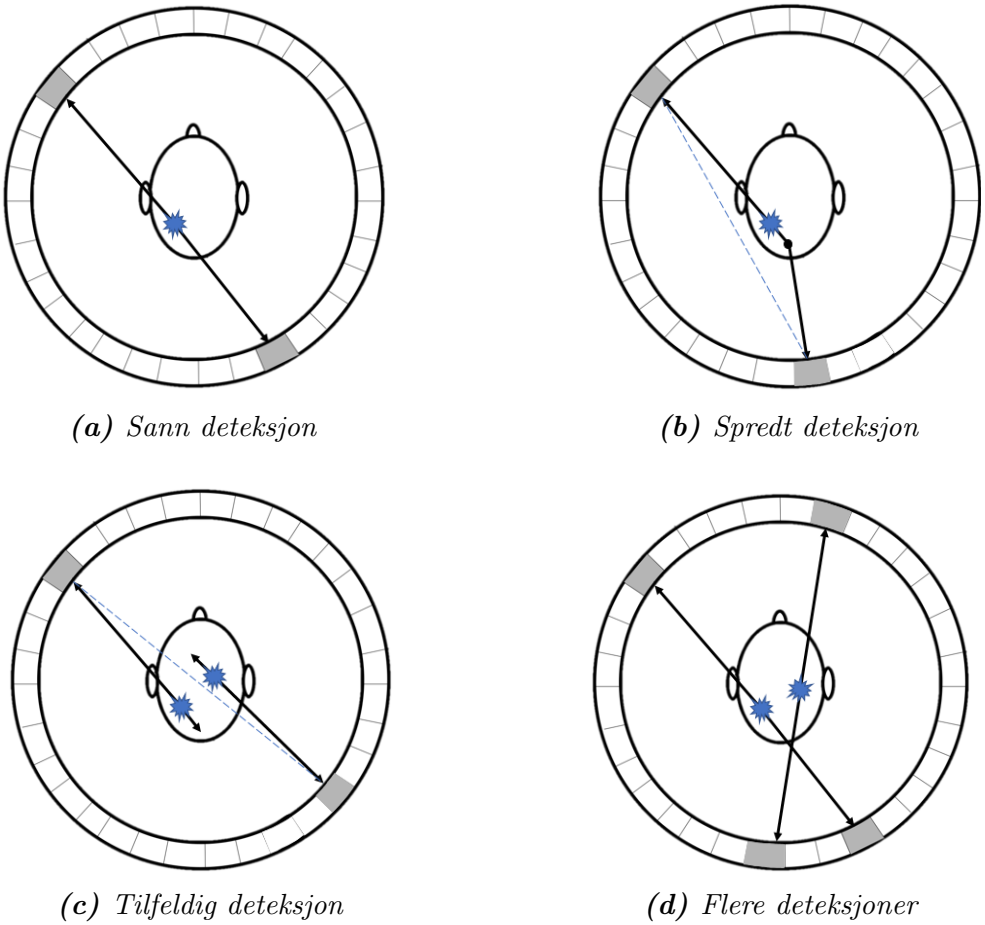
PET-skanneren

Etter at en person har fått injisert væsken som inneholder radionuklidene, må det ventes en times tid, før personen blir plassert i skanneren [4]. Skanneren består av ringer med mange separate detektorer [16]. I figur 2.2 er en slik ring tegnet inn. Når to fotoner blir oppdaget samtidig (eller veldig nært i tid), vil skanneren tolke det som at de kom fra samme annihiliering, og regne ut deres LOR. Perioden som fotonene må oppdages innenfor er vanligvis på et par nanosekunder, og varierer



Figur 2.2: *En av mange ringer som en PET-skanner består av. To detektorer observerer et foton tilnærmet samtidig, og regner ut deres LOR.*

mellom PET-skannere [15]. Antallet LOR som krysser et område vil gi en indikasjon hvor mye positronemisjon som skjer der.



(a) *Sann deteksjon*

(b) *Spredt deteksjon*

(c) *Tilfeldig deteksjon*

(d) *Flere deteksjoner*

Figur 2.3: *Illustrasjon av fire hovedtyper deteksjoner i en PET-skanner. Laget etter inspirasjon fra figur av M. E. Phelps [15].*

Figur 2.3 viser fire forskjellige typer deteksjon:

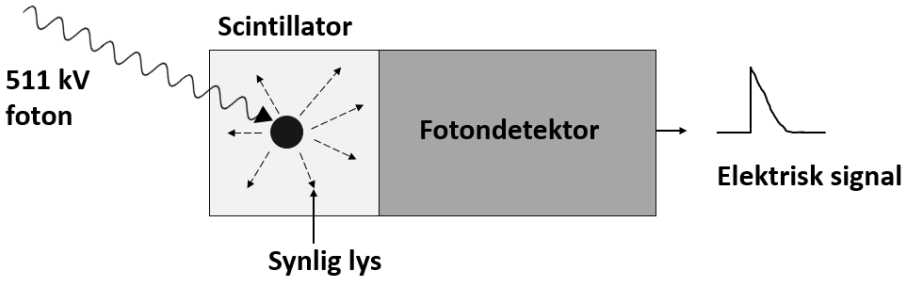
(a) Sann deteksjon: Begge fotonene forlater kroppen og blir oppdaget av detektorene på riktig sted.

(b) Spredt deteksjon: En eller begge fotonene vekselvirker med en partikkel i kroppen, og endrer retning. Dette kalles å bli compton-spredt (compton scattered).

(c) Tilfeldig deteksjon: To fotoner fra to forskjellige annihileringer treffer detektorene samtidig, og skaper en falsk LOR.

(d) Flere deteksjoner: Tre eller flere fotoner blir oppdaget samtidig, og det blir usikkert hvilken annihilering de kommer fra.

Mye brukte detektorer i PET-skannere er krystalliserte scintillatorer, som består av et stoff som emmiterer synlig lys når det blir truffet av høyenergifotoner [15]. Dette lyset blir igjen oppdaget av en detektor som ser synlig lys, og blir konvertert til et elektrisk signal, illustrert i figur 2.4.

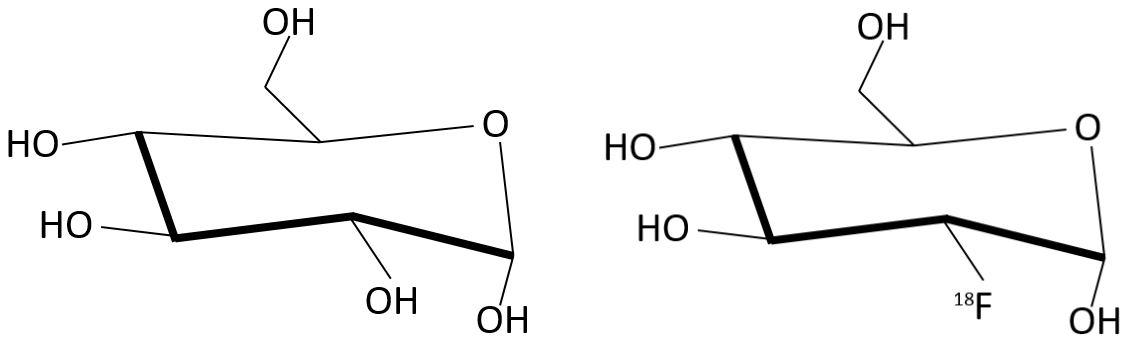


Figur 2.4: *Skjematisk tegning av hvordan et foton fra en annihilering gjøres om til et elektrisk signal. Tegnet etter figur av Phelps [15].*

Fluorodeoksyglukose

Den forbindelsen som er mest brukt som radioaktiv markør i PET-skanninger er fluorodeoksyglukose (^{18}F-FDG). Dette er glukose, der en hydroksyl-gruppe er erstattet med Fluor-18, slik som vist i figur 2.5. Fluor-18 har en halveringstid på 109.8 minutter, og 97% av den radioaktive strålingen skjer gjennom positronemisjon [17]. Opptaket av glukose i et vev viser hvor metabolsk aktiv dette vevet er [16]. ^{18}F-FDG kan ta plassen til glukose i de kjemiske reaksjonene i cellene, og opptaket av ^{18}F-FDG gir dermed en indikasjon på hvor mye glukose vevet tar opp. Kreftceller tar ofte opp mye glukose, ettersom de bruker mer energi enn vanlige celler [19]. Dermed vil disse lyse opp på PET-bilder.

^{18}F-FDG har vist seg være veldig egnet for å måle metabolsk aktivitet for mange forskjellige organer og sykdommer i den menneskelige kroppen. Positronet som sendes ut fra forbindelsen har lavere energi enn mange andre radioaktive markører, som



(a) *Glukose* ***(b)*** *Fluorodeoksyglukose*

Figur 2.5: *Forskjellen mellom glukose og fluorodeoksyglukose. I fluorodeoksyglukose er en hydroksylgruppe erstattet med fluor-18.*

gjør at disse PET-bildene kan ha høyere oppløsning. Rekkevidden for disse positronene i vann er mye kortere enn 2 mm [17]. I tillegg gjør den lengre halveringstiden at mer komplekse/tidskrevende undersøkelsesmetoder kan benyttes.

SUV

En vanlig måte å måle FDG-opptaket til en pasient, er ved bruk av SUV-verdier (Standardized Uptake Value) [20]. Noe av det som varierer mest mellom pasienter og skanninger er mengden injisert væske og vekten til pasienten. Dette tar SUV-verdier hensyn til [20]:

$$SUV = \frac{r}{a/w} \tag{2.3}$$

der r er radioaktivitetskonsentrasjonen [kBq/ml] i området målt av PET-skanneren, a er mengden injisert FDG [kBq] og w er vekten til pasienten [g].

2.2.2 Computertomografi

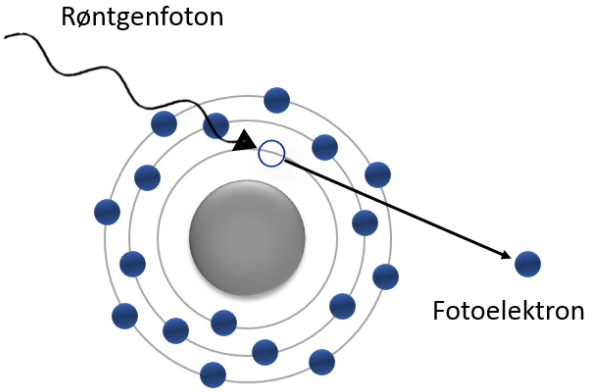
Computertomografi (CT) tar i bruk røntgenbilder tatt fra forskjellige vinkler, for å skape tomografiske bilder (snittbilder) av en kroppsdel [16]. Dette gjør at man får åpnet kroppen og kan se kroppens anatomi i tre dimensjoner.

Røntgenstråler er energirike fotoner, med bølgelengde kortere enn 10 nm. Røntgenbilder blir skapt ved at røntgenstråler blir sendt gjennom kroppen og ned på en detektor. Bildene er skygger som blir skapt når røntgenstrålene blir absorbert av noe i kroppen, som f.eks. skjelettet. Røntgenstrålene som ikke blir absorbert fortsetter gjennom kroppen og ned til detektorene. Resultatet er en projeksjon av objektene som røntgenstrålene sendes på, der intensiteten varierer med absorpsjonsevnen til forskjellige deler av kroppen. Lyse regioner på bilder tilsier at færre stråler treffer detektorene (mye absorbert), mens mørke regionene tilsier at flere røntgenstråler treffer detektorene [16].

Når røntgenfotoner sendes gjennom kroppen, finnes det tre forskjellige måter som fotonene kan vekselvirke med vevet: Fotoelektrisk effekt, compton-spredning eller pardannelse. Dersom ingen av disse hendelsene tar plass, vil fotonene gå gjennom kroppen uten å bli absorbert. Detektorene observerer fotonene som ikke blir absorbert i kroppen, i tillegg til de som har blitt spredt.

Fotoelektrisk effekt

Når en røntgenstråle blir absorbert i kroppen, skjer det vi kaller fotoelektrisk effekt [16]. Når dette skjer, vekselvirker et foton med et elektron i en av de innerste elektronskallene rundt et atom. Disse elektronene er tett bundet til atomet, og krever mer energi å løsrive enn elektroner i de ytre elektronskallene. Det er en sannsynlighet for at elektronet absorberer all energien til fotonet, og får nok energi til å løsrives fra atomet (figur 2.6). Et slikt fritt elektron kalles et fotoelektron. I en foto-



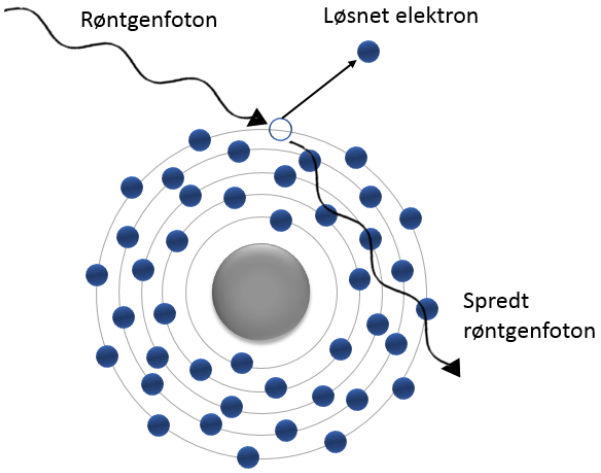
Figur 2.6: *Fotoelektrisk effekt, der en røntgenstråle løsner et elektron fra den innerste orbitalen til et atom.*

elektrisk effekt overføres all energien fra fotonet til elektronet, og det opprinnelige fotonet eksisterer ikke lenger. Den overflødige energien som ikke ble brukt til å rive løs elektronet, blir til den kinetiske energien til elektronet. Elektronhullet som nå er i orbitalen rundt atomet blir fort fyllt opp igjen, og fotoelektronet beveger seg bare en kort strekning gjennom kroppen, før energien blir overført videre [16].

Compton-spredning

Som regel synker sannsynligheten for fotoelektrisk effekt når energien til røntgenstålene øker [16]. Her begynner en annen prosess å bli viktigere, og det er compton-spredning. I denne prosessen vekselvirker røntgenfotonet med et elektron i et av de ytre elektronskallene til atomet, og river det løs fra atomet, se figur 2.7. Denne interaksjonen krever mye mindre energi enn ved fotoelektrisk effekt. Fotonet fortsetter videre etter interaksjonen, med en endret vinkel og bølgelengde. Fotonet kan fortsette å treffe på flere slike elektroner, og gradvis miste energi. Fotonet har blitt compton-spredt,

og har fått en ny retning og bølgelengde i forhold til utgangspunktet. Slik spredning vil være støy i røntgenbildet.



Figur 2.7: *Compton-spredning, der et røntgenfoton vekselvirker med et elektron i et av de ytterste elektronskallene til et atom. Dette fører til løsrivelse av elektronet, og et spredt foton.*

Pardannelse

Pardannelse kan oppstå når fotoner har energi over 1.002 MeV [21]. Når et foton vekselvirker med et elektrisk felt rundt en cellekjerne, kan energien i fotonet transformeres til et elektron og et positron. Dette er den reverserte prosessen av annihilering beskrevet i avsnitt 2.1. Grunnen til at pardannelse krever fotoner med energier over 1,002 MeV, er at massen til elektronet og positronet uttrykt i energi er 0,511 MeV, og dermed kreves det minst $2 \times 0,551 = 1,002\,MeV$.

Absorpsjon

All interaksjon mellom røntgenstråler og materie baserer seg på sannsynlighet [16]. Sannsynligheten for fullstendig energioverføring eller delvis absorpsjon er avhengig av de kjemiske egenskapene til stoffet. Materie med mange elektroner (og dermed mange protoner) har større sannsynlighet for interaksjon med røntgenstråler. Atomnummeret Z til et grunnstoff er lik antall protoner stoffet har i kjernen, og sannsynligheten for fotoelektrisk effekt øker med kuben til atomnummeret Z^3 [16]. Mykere deler av kroppen består ofte av stoffer med lave atomnumre som karbon ($Z = 6$) eller oksygen ($Z = 8$), og har mindre sannsynlighet for å absorpsjon ved fotoelektrisk effekt. Mer solide deler, som f.eks. bein, har stoffer med høyere atomnummer, som kalsium ($Z = 20$). Disse har høyere sannsynlighet for å absorbere røntgenstråler.

Hvor mye av røntgenstrålene som slippes igjennom et stoff avhenger av attenuasjonsevnen til stoffet. Denne er avhengig av attenuasjonskoeffisienten μ til et stoff

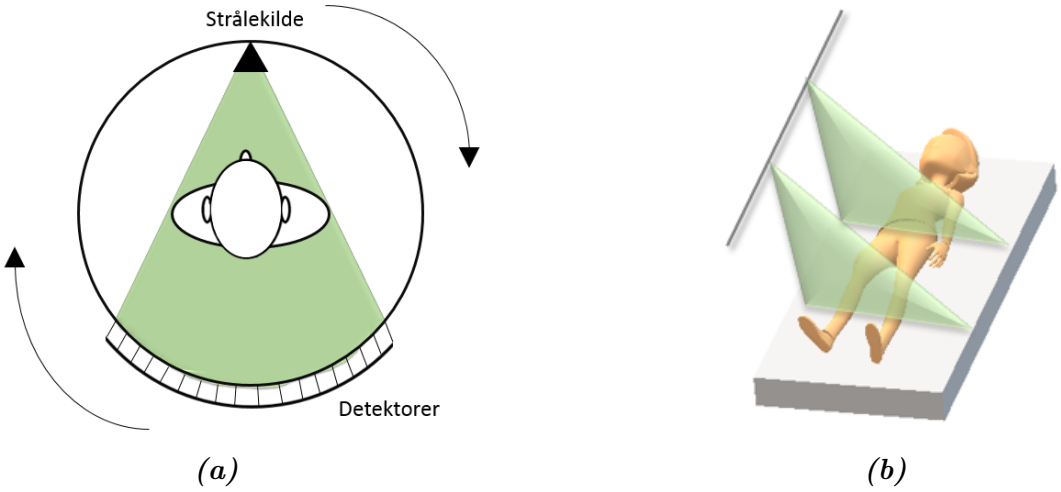
og massetettheten. Intensiteten til en røntgenstråle som transmitteres igjennom et materiale I_{trans} blir bestemt av følgende ligning:

$$I_{trans} = I_0 e^{-\mu x} \tag{2.4}$$

der I_0 er intensiteten til røntgenstrålen før vekselvirkningen og x er tykkelsen til stoffet [16].

Fra røntgen til CT

Som tidligere nevnt, gir røntgenbilder kun en todimensjonal projeksjon av et objekt. Ved bruk av CT, kan leger se på et detaljert bilde som viser et tverrsnitt av gangen, eller se alt som et tredimensjonalt bilde. Et CT-apparat har en åpning som en smultring, der pasienten beveges inn i åpningen [16]. Her blir røntgenbilder tatt som tidligere forklart, men fra mange forskjellige vinkler (figur 2.8). CT-skannere



Figur 2.8: *En mulig måte som en CT-skanner fungerer på. (a) Både stålekilden og rekken med detektorer roterer rundt kroppen, og lager projeksjoner for mange vinkler innenfor et plan. (b) Etter at alle vinklene har blitt målt, flyttes kroppen lenger inn i skanneren, og prosessen starter på nytt for et nytt plan. Laget etter inspirasjon fra figur i S. A. Kane [16].*

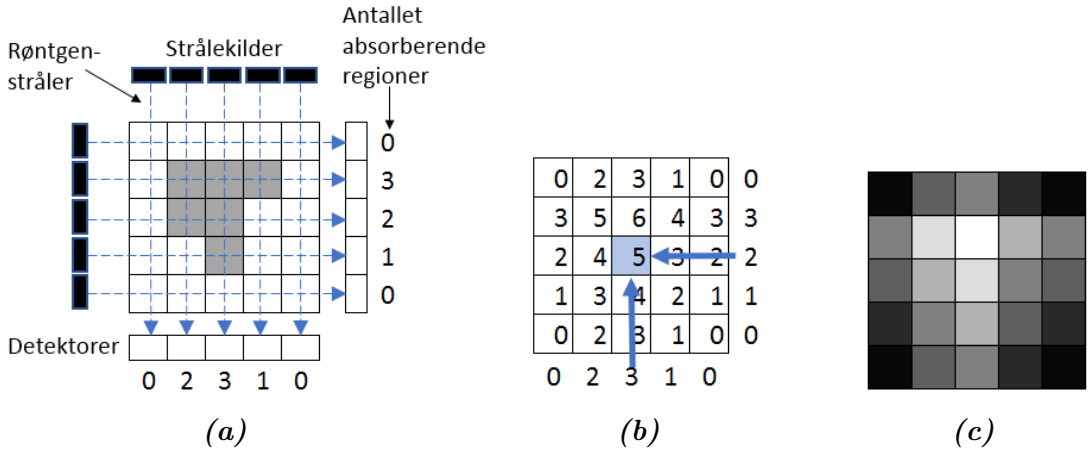
representerer absorpsjonen av røntgenstråler med enheten CT nummer:

$$CT\,nummer = \frac{\mu_{vev} - \mu_{vann}}{\mu_{vann}} \times 1000 \tag{2.5}$$

der μ_{vev} er attenuasjonskoeffisienten til vevet og μ_{vann} er attenuasjonskoeffisienten til vann [16]. Vann har CT-nummer 0, vev med mye fett kan ha negative CT-nummer, men mesteparten av vev har et positivt CT-nummer. De delene av kroppen som har høyest CT-nummer er skjelettet, mens de laveste CT-numrene er luft i f.eks. lunger eller tarm [16].

Projeksjon

Måten CT-skannere gjør projeksjoner om til et bilde er kompleks, men ideen bak er enkel. Hvis man ser et objekt fra to sider, kan man få et bilde av hvordan objektet ser ut i tre dimensjoner. Det er mange forskjellige matematiske teknikker for å gjøre dette, men en mye anvendt metode er *filtered back projection* [16]. Ideen bak denne metoden illustreres i figur 2.9. Under selve skanningen, får vi en projeksjon



Figur 2.9: *Fremgangsmetoden for filtered back projection med (a) forward projection, (b) back projection og (c) er det endelige bildet, der svart er verdien 0 og hvit er verdien 6.*

kalt *forward projection*, der detektorene måler røntgenintensiteten som forklart tidligere [16]. Figur 2.9a illustrerer hvordan projeksjonen vil se ut fra to vinkler, der antallet absorberende regioner er telt opp. For å rekonstruere røntgenabsorbsjonen til kroppen, brukes deretter *back projection*, illustrert i figur 2.9b. Her brukes antall absorberende regioner som hver detektor har telt opp, og projeksjonsmatrisen kan vises som et bilde ved bruk av gråskalaverdier (figur 2.9c).

Før prosessen med *back projection* uføres, anvendes et filter for å fjerne støy i projeksjonen som oppstod under prossessen med *forward projection*. I praksis kan CT-bilder ha oppløsning på mindre enn 1 mm^3, men høyere oppløsninger fører til at sterkere røntgenstråler må bli brukt over lenger tid [16].

Kontrastvæske

Mange forskjellige myke vev i kroppen kan fremstilles ganske likt ved røntgenbilder og CT, ettersom de har ganske like Z-verdier [16]. For å forsterke synligheten av myke vev i røntgen- og CT-bilder, er det vanlig å injisere kontrastvæske. Barium og jod, med henholdsvis Z-verdier på 56 og 53, er grunnstoffer som er vanlige å bruke i kontrastvæsker. Disse absorberer røntgenstråler godt, og vil gjøre at vevet det er injisert i, vil lyse opp på røntgen- og CT-bilder.

2.3 Maskinlæring

I denne oppgaven brukes maskinlæring for å klassifisere pasienter etter behandlingsutfall fra et datasett bestående av klinisk pasientinformasjon og egenskaper trukket ut av kreftsvulster fra PET- og CT-bilder. Maskinelæring er en gren av kunstig intelligens, som involverer selvlærende algoritmer som trekker ut kunnskap fra data for å foreta prediksjoner [7]. Istedenfor at mennesker manuelt skal utvikle og bygge modeller ved å analysere store mengder data, tilbyr maskinlæring en effektiv måte å finne kunnskap i data ved å gradvis forbedre ytelsen til modeller ved å lære fra datamengden. Maskinlæring har i dag gjort at vi kan bruke f.eks. robuste spamfiltre, stemme- og bildegjenkjenning og velutviklede søkemotorer [7].

I maskinlæring er det vanlig å kalle et datasett med variabler for en matrise X. Her er variablene kolonner i matrisen, og radene i matrisen er observasjonene. Figur 2.10 viser en generell X-matrise (øverst) og et eksempel på en X-matrise fra data i denne oppgaven (nederst). Her ser vi at kliniske faktorer og egenskaper fra bildene

X

	Variabel 1	Variabel 2	Variabel 3
Observasjon 1	x_{11}	x_{12}	x_{13}
Observasjon 2	x_{21}	x_{22}	x_{23}
Observasjon 3	x_{31}	x_{32}	x_{33}
Observasjon 4	x_{41}	x_{42}	x_{43}

y

	Respons
Observasjon 1	y_1
Observasjon 2	y_2
Observasjon 3	y_3
Observasjon 4	y_4

	Alder	Uniformitet	Median
Pasient 1	46	0,01	20
Pasient 2	52	0,4	25
Pasient 3	64	3	17
Pasient 4	83	0,07	20

	Behandlingsutfall
Pasient 1	Tilbakefall
Pasient 2	Sykdomsfri
Pasient 3	Sykdomsfri
Pasient 4	Tilbakefall

Figur 2.10: *Illustrasjon av hvordan observasjoner er lagt i en X-matrise og responsen er en vektor y. De to øverste figurene viser det generelle tilfellet, der x_{ij} er observasjon nummer i for variabel nummer j. De to nederste figurene er eksempler på observasjoner og responser i denne oppgaven.*

er variablene, mens de forskjellige pasientene er observasjonene.

Ved veiledet læring (forklart i avsnitt 2.3.1) har dataene i tillegg en respons-vektor y, som har en respons til hver av observasjonene. Denne vises til høyre i figur 2.10, der den øverste er et eksempel på en generell respons til den generelle matrisen X, mens den nederste er et eksempel på responsen i denne oppgaven. Her er observasjonene fortsatt pasientene, men responsen er behandlingsutfallet, som er enten tilbakefall eller progresjonsfri overlevelse (også kalt sykdomsfri). Variabler i denne oppgaven blir også referert til som egenskaper.

Som regel deles datasettet inn i et treningssett X_{train} og y_{train} og et valideringssett X_{test} og y_{test}. Treningssettet brukes for å trene klassifiseringsmodellene, mens

valideringssettet brukes for å teste/predikere ytelsen til modellene. Ved kryssvalidering deles datasettet opp i treningssett og valideringssett manger ganger. Dette blir nærmere forklart i avsnitt 3.6.4.

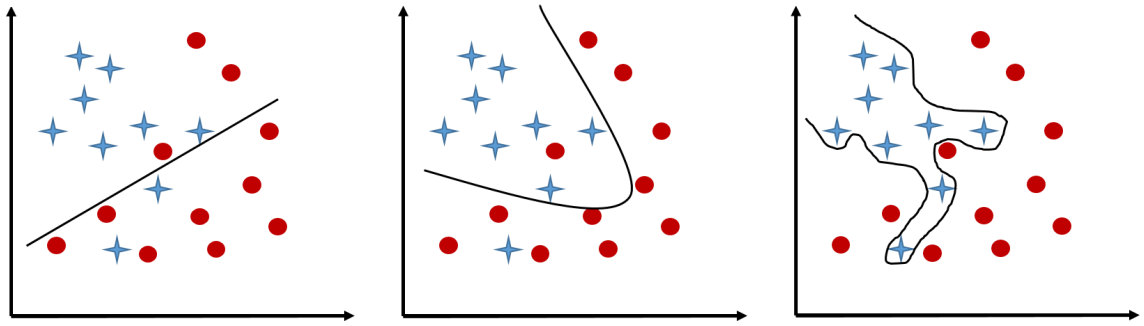
2.3.1 Veiledet og ikke-veiledet læring

Veiledet læring er en hovedgren under maskinlæring, der vi ønsker å modellere data i forhold til en respons [22]. Vi har altså en respons som veileder læringsprosessen mot målet, som er å predikere responsen for nye data [7]. Det er metoden som er brukt for klassifisere observasjonene i denne oppgaven. Veiledet læring kan brukes til både klassifiseringsproblemer og regresjon.

En annen hovedgren av maskinlæring er ikke-veiledet læring. I ikke-veiledet læring ser man bare på egenskaper og observasjoner, og har ingen respons til disse [22]. Målet er derimot å utforske strukturen til dataene, og observere hvordan de er organisert eller gruppert. Dette har blitt brukt i to dimensjonsreduksjonsmetoder i denne oppgaven.

2.3.2 Tilpasning av en modell

En viktig del av maskinlæring er å finne et godt kompromiss mellom varians og bias [7]. Dette er illustrert i figur 2.11. En modell som har høy varians varierer mye fra gjennomsnittsverdien, og er da en ujevn modell, slik som figur 2.11c. En av grunnene er at modellen er for kompleks i forhold til dataene. Dette fører til overtilpasning, der modellen fungerer veldig godt på prediksjon av treningsdataene, men ikke på fremtidig data.



(a) *Undertilpasset (høy bias)*

(b) *Godt kompromiss mellom varians og bias*

(c) *Overtilpasset (høy varians)*

Figur 2.11: *En illustrasjon av (a) en undertilpasset modell med høy bias, (b) en passende modell som er et godt kompromiss mellom varians og bias og (c) en overtilpasset modell med høy varians. Her er blå stjerner observasjoner som tilhører én klasse, og røde rundinger er observasjoner som tilhører den andre klassen. Den sorte linje er beslutningsgrensen til modellen som prøver å skille de to klassene fra hverandre.*

En modell som har høy bias predikerer responsen dårlig, ettersom den ikke er godt nok tilpasset til dataene, slik som figur 2.11a. Modellen er ikke kompleks nok til å fange mønsteret i treningsdataene, og den vil ikke ha en god prediksjon på hverken treningsdataene eller fremtidig data. Et hovedproblem innenfor maskinlæring er å finne et passende kompromiss mellom varians og bias, for å få en modell som predikerer treningsdata så vel som fremtidig data, slik som figur 2.11b.

2.3.3 Klassifiseringsalgoritmer

Det finnes mange forskjellige typer klassifiseringsalgoritmer. Noen eksempler er lineære klassifiseringsalgoritmer, ikke-lineære klassifiseringsalgoritmer, trær og minnebaserte-klassifiseringsalgoritmer [22]. Alle disse har som mål å skille to eller flere klasser fra hverandre. Under følger en kort forklaring på klassifiseringsalgoritmene brukt i denne oppgaven.

Logistisk Regresjon

Logistisk regresjon er en klassifiseringsalgoritme basert på prinsippet om å estimere sannsynligheten til en binær respons basert på variabler [22]. Den er enkel å implementere, og har generelt en god ytelse når klassene er lineært separable [7]. Lineære klassifiseringsalgoritmer er klassifiseringalgoritmer der responsen kan estimeres som en lineær kombinasjon av variablene:

$$\hat{f}(X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{2.6}$$

der $\hat{f}(X)$ er den estimerte responsen (klassen), x_j er variabel nummer j, β_j er koeffisienten til variabel nummer j og n er antall variabler [22]. Logistisk regresjon er en sannsynlighetsmodell, som estimerer koeffisientene ved bruk av maksimal sannsynlighet. Dette betyr at koeffisientene som gir høyest sannsynlighetsfunksjon blir valgt. Log-sannsynlighetsfunksjonen for N observasjoner er gitt som:

$$l(\beta) = \sum_{i=1}^{N} log p_k(x_i; \beta) \tag{2.7}$$

der $p_k(x_i; \beta)$ er sannsynlighten for at observasjon i er i klasse k gitt koeffisienten β [22]. Ved bruk av en binær respons, og ved å legge til en straffeparameter gitt av det siste leddet i (2.8), blir sansynnlighetsfunksjonen som skal maksimeres slik:

$$max_{\beta_0,\beta}\{\sum_{i=1}^{N}[y_i(\beta_0 + \beta^T x_i) - log(1 + e^{\beta_0+\beta^T x_i})] - \lambda \sum_{j=1}^{p} \beta_j^2\} \tag{2.8}$$

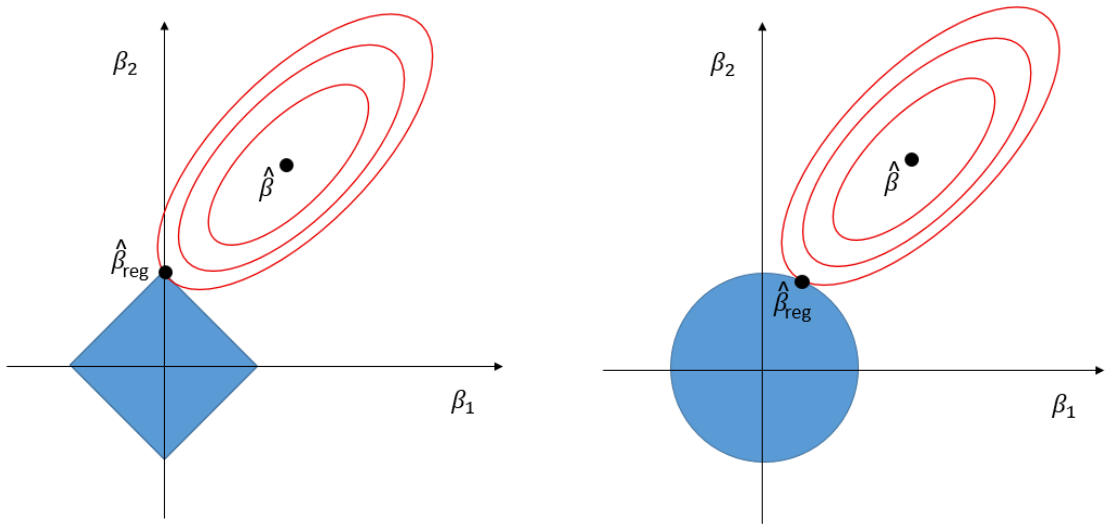
der β_0 er skjæringspunktet med y-aksen, β er koeffisientene til variablene, x_i er observasjon i, y_i er klassen til observasjon i, $\lambda \geq 0$ er kompleksitetsparameteren som

kontrollerer styrken på straffeparameteren, og β_j er koeffisienten til variabel j [22]. Denne sannsynlighetsfunksjonen er en regularisert funksjon, som betyr at koeffisientene blir krympet nærmere null, ved å legge en straff på størrelsen deres. Funksjon (2.8) inneholder en L2-straffeparameter: $\sum_i^p \beta_j^2$, også kalt en ridge-straffparameter eller en L2-regularisering [22]. Her er koeffisientene straffet med koeffisientenes sum av kvadrater. Dette betyr at den totale sannsynligheten maksimeres, men uten å bruke for store koeffisienter på variablene.

Et alternativ er å bruke en L1-straffeparameter, også kalt en lasso-straffparameter: $\sum_i^p |\beta_j|$ [22]. Denne skiller seg ut fra ridge-straffeparameteren ved at koeffisientene kan bli krympet til nøyaktig null ved å øke styrken på straffeparameteren (siste ledd i ligning (2.9)). Da blir sannsynlighetsfunksjonen som skal maksimeres:

$$max_{\beta_0,\beta}\{\sum_{i=1}^{N}[y_i(\beta_0 + \beta^T x_i) - log(1 + e^{\beta_0+\beta^T x_i})] - \lambda \sum_{j=1}^{p} |\beta_j|\} \tag{2.9}$$

der β_0 er skjæringspunktet med y-aksen, β er koeffisientene til variablene, x_i er observasjon i, y_i er klassen til observasjon i, $\lambda \geq 0$ kontrollerer styrken på straffeparameteren, og β_j er koeffisienten til variabel j [22]. Figur 2.12 illustrerer forskjellen på logistisk regresjon med en lasso-straffeparameter 2.12a og en ridge-straffeparameter 2.12b.



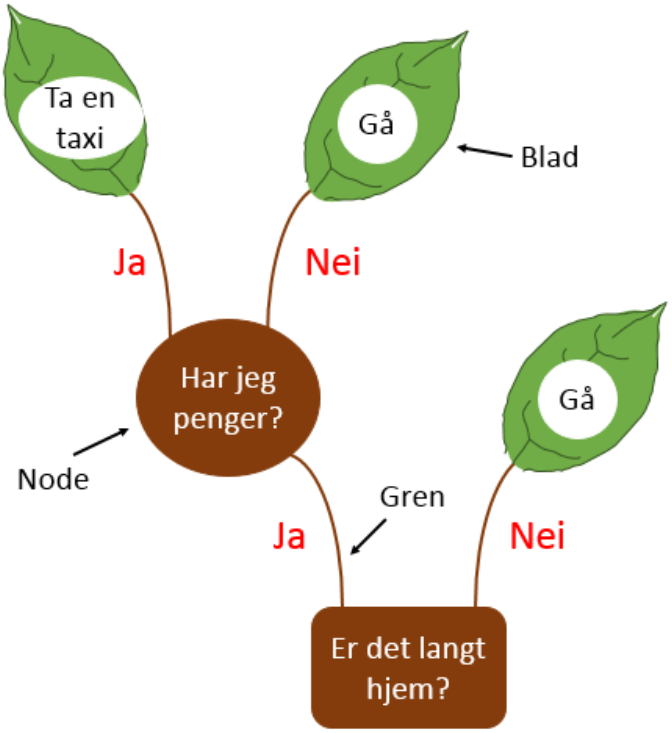
(a) *Lasso)* ***(b)*** *Ridge*

Figur 2.12: *En illustrasjon over forskjellen mellom en lasso-straffeparmeter (a) og en ridge-straffeparameter (b). De røde ellipsene er konturene av residualenes sum av kvadrater for en logistisk regresjon, som er minimert i $\hat{\beta}$. De blå figurene er hhv. lasso og ridge sine begrensninger, og målet er å både minimere de blå figurene og de røde ellipsene. Det endelige estimatet for koeffisientene er der de røde ellipsene og den blå figuren møtes, i $\hat{\beta}_{reg}$. Lasso sin kvadratiske begrensning gjør at det er en mulighet for at noen koeffisienter blir krympet til nøyaktig null. Tegnet etter figur fra Hastie et al. [22].*

I denne oppgaven ble Scikit-Learn sin pakke LogisticRegression brukt [23].

Beslutningstrær

Et beslutningstre (decision tree) er blandt de klassifikasjonsalgoritmene som gir enklere tolkninger av resultat, ved bryte ned et datasett i oppdeling med spørsmål eller tester som må svares på [24]. Et spørsmål kalles for en node, og beslutningen fører ut til to grener. Figur 2.13 er et eksempel på et beslutningstre. På bakgrunn av treningssettet, lærer beslutningstreet hvilke spørsmål det skal stille for å klassifisere responsen riktig.



Figur 2.13: *Et dagligdags eksempel på et enkelt besluningstre. Problemstillingen er hvordan en person skal komme seg hjem. Første spørsmål er "Er det langt hjem?" som fører ut i to grener. Hvis svaret er "Nei", så føres grenen ut i en blad, som blir beslutningen "Gå". Hvis svaret er "Ja" fører grenen til en node, som stiller et nytt spørsmål.*

I denne oppgaven er *Gini Urenhet* (Gini Impurity) brukt som målenheten for "beste splitt", og den er et mål for hvor mye av data i en gren eller et blad som tilhører samme klasse. Gini Urenhet er gitt ved:

$$G(T) = \sum_{i=1}^{n} \sum_{j \neq i} f_i f_j \tag{2.10}$$

der f_i og f_j er andelen av treningsdata for klasse i og j og T er treningsdataene for den gitte noden [25]. Et blad (ytterste del av en gren) er ansett som helt "ren"hvis all data i bladet tilhører samme klasse. Beslutningstreet prøver å splitte dataene ved en node slik at urenheten i begge de to nye grenene minimeres.

Ved bruk av mange egenskaper, kan et beslutningstre bli veldig dypt, og kan enkelt

overtilpasses treningsdataene. Derfor er det vanlig å beskjære treet, slik at et det er en begrensning for dybden på treet.

En algoritme som bruker beslutningstrær, og som har blitt veldig populær, er Random Forests (RF) [7]. RF tar gjennomsnittet av mange beslutningstrær, som individuelt har høy varians, og bygger en mer robust modell som har mindre sjanse for å overtilpasse treningsdataene.

RF bygger på bootstrap-metoden, der hovedideen er å lage nye datasett med erstatninger fra det originale datasettet, og med samme størrelse som det originale datasettet [22]. Dette betyr at noen observasjoner vil bli representert flere ganger i det nye datasettet, og noen observasjoner vil ikke være i det nye datasettet i det hele tatt [26]. Figur 2.14 viser hvordan det lages B antall bootstrap-datasett basert på et opprinnelige datasettet.

Figur 2.14: *Et eksempel på hvordan datasett deles opp i B antall boostrap-datasett. I hver bootstrap-datasett er det samme antall observasjoner som i det originale datasettet, men noen av observasjonene er representert flere ganger. Modellen brukes for prediksjon på observasjoner som ikke var inkludert i treningen av modellen.*

Ved bruk av bootstrap-datasett, følger RF følgende algoritme:

1. For $b = 1$ til B, hvor B er antall beslutningstrær:
   (a) Trekk et tilfeldig bootstrap-datasett b av størrelse N fra treningsdataene, der N er antall observasjoner i det originale datasettet.
   (b) Voks et beslutningstre T_b til bootstrap-dataene, ved å rekursivt repetere de følgende stegene for hver node i treet.
      i. Velg en delmengde med m antall variabler tilfeldig fra alle variablene, der m er et valgfritt tall.
      ii. Velg den beste variablen/splitt-punktet blant de m variablene.
      iii. Split noden til to grener med to nye noder.
2. Resultatet av steg 1 er et ensemble av beslutningstrær $\{T_b\}_1^B$.
3. For å gjøre en prediksjon på en ny observasjon x: La $\hat{C}_b(x)$ være klassen som beslutningstre nummer b predikerer. Da er $\hat{C}_{rf}^B(x) = flertallstemmen\{\hat{C}_b(x)\}_1^B$

I denne oppgaven ble Scikit-Learn sine pakker DecisionTreeClassifier og RandomForestClassifier brukt [23].

K-Nearest Neighbor

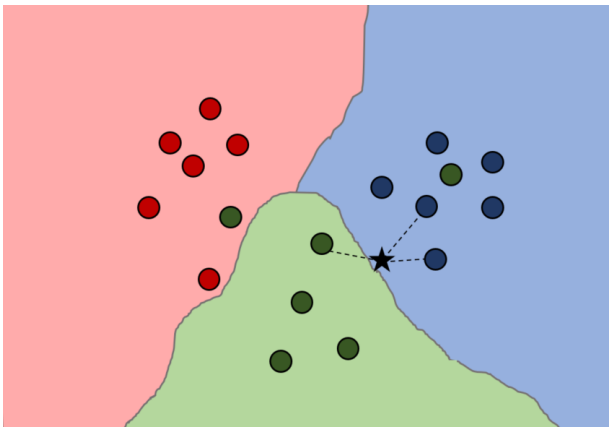
KNN (K-Nearest Neighbor) er en metode som bruker observasjonene i treningssettet som er nærmest observasjonen av interesse til å predikere klassen [22]. Den predikerte klassen defineres slik:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \epsilon N_k(x)} y_i \tag{2.11}$$

der k er antall observasjoner som skal være med i beregningen, $N_k(x)$ er nabolaget til observasjon x definert av de k nærmeste observasjonene x_i i treningssettet og y_i er klassen til x_i [22]. Prinsippet går ut på at et nytt punkt blir tildelt den klassen som har flest representanter blandt de k-nærmeste naboene. Dette er illustrert i figur 2.15. Nærhetberegnes ved bruk av en målenhet for lengde. I denne oppgaven er Minikowskiavstanden av andre orden brukt, som tilsvarer den euklidiske avstanden:

$$d = (\sum_{i=1}^{m} |u_j - v_j|^2)^{\frac{1}{2}} \tag{2.12}$$

hvor $(u_1, u_2, ..., u_m)$ og $(v_1, v_2, ..., v_m)$ er to vektorer i et m-dimensjonalt euklidisk rom [27]. I maskinlæring tilsvarer disse vektorene observasjoner, og u_j og v_j er variablene tilhørende disse observasjonene.



Figur 2.15: *En illustrasjon av en klassifisering med KNN med $k = 3$ naboer. Her er det tre forskjellige klasser (rød, blå og grønn), som er adskilt med beslutningsgrenser bestemt av de tre nærmeste observasjonene. Den sorte stjernen illustrerer en ny observasjon, som skal klassifiseres på bakgrunn av datasettet. To av de tre nærmeste naboene er av klassen blå, dermed blir stjernen klassifisert som blå.*

KNN er en type "lat lærer", som betyr at den ikke lager en modell på bakgrunn av treningsdataen, men lagrer heller treningsdataene slik at nye data kan sammenlignes med disse [7]. Dette gjør at den kan være en treg algoritme, som kan bruke lang tid på store og høydimensjonale datasett.

I denne oppgaven ble Scikit-Learn sin pakke KNeighborsClassifier brukt [24].

Boosting

Boosting er en familie klassifieringsalgoritmer, der hovedideen er å kombinere mange *svake* klassifiseringsalgoritmer til en sterkt *komitee* [22]. En svak klassifieringsalgoritme er en algoritme der prediksjonen ikke er mye bedre enn tilfeldig gjetning. Hensikten med boosting er å sekvensielt anvende den svake klassifiseringsalgoritmen til modifiserte versjoner av dataene, og dermed lage en sekvens av svake klassifiseringsalgoritmer.

En av de mest populære boosting-algoritmene er AdaBoost, som fungerer ved å gi en høyere vekt til de observasjonene som blir feilklassifisert [22]. Den fungerer på følgende måte for en binær respons med klassene -1 og 1:

1. Initialiser observasjonsvektene $\omega_i = \frac{1}{N}, i = 1, 2, ..., N$, hvor N er antall observasjoner.

2. For $m = 1$ til M, hvor M er antall boosting-operasjoner:

   (a) Tren en klassifiseringsalgoritme $G_m(x)$ på treningsdataene ved å bruke vektene ω_i.

   (b) Regn ut feilklassifieringsraten

   $$err_m = \frac{\sum_{i=1}^{N} \omega_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \omega_i}$$

   (c) Regn ut $\alpha_m = log(\frac{1-err_m}{err_m})$

   (d) Sett $\omega_i \leftarrow \omega_i e^{\alpha_m I(y_i \neq G_m(x_i))}, i = 1, 2, ..., N$

3. Klassen $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

I punkt 2a er klassifiseringsalgoritmen introdusert. I punkt 2b regnes feilklassifiseringsraten ut, ved å summere alle observasjonene som var feilklassifisert med deres respektive vekter. I punkt 2c introduseres vekten α_m, som er vekten gitt til denne boosting-operasjonen $G_m(x)$ ved den endelige summeringen. I punkt 2d blir vektene til observasjonene som ble feilklassifisert oppdatert med en faktor e^{α_m}, som gjør at de får sterkere innflytelse på neste klassifiseringsalgoritme ved neste boosting-operasjon $G_{m+1}(x)$. Resultatet (punkt 3) er en sum av alle modellene som ble bygget, og gir et negativt eller et positivt fortegn for observasjon x.

I denne oppgaven ble Scikit-Learn sin pakke AdaBoostClassifier brukt, sammen med en logistisk regresjon som kjernemodell [23].

Discriminant Analysis

Linear Discriminant Analysis (LDA) er en dimensjonsreduksjonsmetode, som har som mål å finne lineære kombinasjoner av de orignale variablene som optimerer separerbarheten mellom klasser [7]. LDA blir både brukt som en klassifiseringsalgoritme, men også som en dimensjonsreduksjonsmetode før en klassifisering. Både LDA, QDA og Gaussian Naives Bayes (som blir forklart senere) er basert på en kombinasjon av Bayes teorem (2.13) og normalfordelinger. Tanken bak disse, er å beregne sannsynligheten for at en observasjon x er i klasse k ($y = k$), gitt observasjon x, og bruker Bayes teorem:

$$P(y = k|x) = \frac{P(x|y = k)P(y = k)}{P(x)} \tag{2.13}$$

der $P(y = k|x)$ er sannsynligheten for at y er klasse k gitt observasjon x, og $P(x|y = k)$ er sannsynligheten for observasjon x gitt at y er klasse k, $P(y = k)$ er den generelle sannsynligheten for at y er klasse k og $P(x)$ er den generelle sannsynligheten for at observasjon x skjedde, som er det samme som $P(x|y = l)P(y = l)$, der l er den andre klassen [26]. LDA estimerer $P(x|y = k)$ ved å bruke den multivariate normalfordelingen:

$$f_k(X) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)} \tag{2.14}$$

der k er klassen, p er antall dimensjoner/variabler, μ_k er forventningsverdien og Σ_k er den felles kovarians-matrisen [22].

Med utgangspunkt i den multivariate normalfordelingen, og antakelsen om at klassene har en felles kovariansmatrise, har LDA som mål å maksimerere LD-funksjoner slik at klassen til observasjon x blir den som maksimerer $\delta_k(x)$:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k \tag{2.15}$$

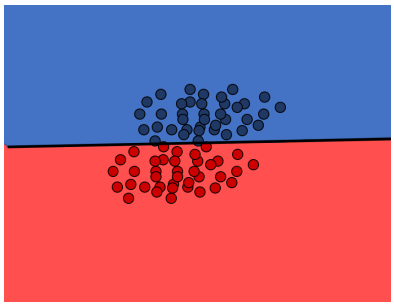
der $\pi_k = (N_k)/(N)$, N er antall observasjoner, N_k er antall observasjoner i klasse k, $\mu_k = (\sum_{g_i=k} x_i)/(N_k)$ og $\Sigma = \sum_{k=1}^{K}\sum_{g_i=k}(x_i - \mu_k)(x_i - \mu_k)^T(N - K)$.

Dersom klassene ikke har en felles kovariansmatrise, oppstår ligningen for Quadratic Discriminant Analysis (QDA) som prøver å maksimere QD-funksjoner:

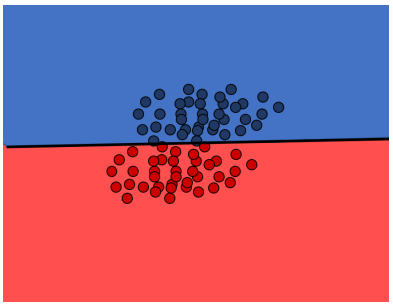
$$\delta_k(x) = -\frac{1}{2}log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T\Sigma_k^{-1}(x - \mu_k) + log\pi_k \tag{2.16}$$

LDA bruker lineære grenser, mens QDA kan bruke kvadratiske grenser og er da mer fleksibel. Figur 2.15 viser hvordan LDA og QDA har forskjellig oppførsel når klassene ikke har en felles kovariansmatrise.

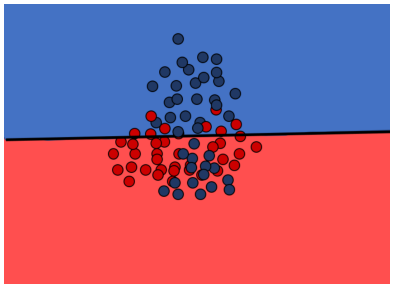
I denne oppgaven ble Scikit-Learn sine pakker LinearDiscriminantAnalysis og QuadraticDiscriminantAnalysis brukt [23].



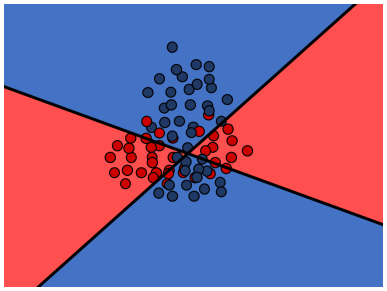
(a) LDA (Lik kovarians)



(b) QDA (Lik kovarians)



(c) LDA (Ulik kovarians)



(d) QDA (Ulik kovarians)

Figur 2.16: *Et plot over data fra to klasser med deres beslutningsgrenser (grensen mellom klassene). De øverste bildene viser tilfellet der begge klassene har lik kovarians, og QDA og LDA gir lik klassifikasjon. De nederste bildene viser tilfellet der klassene har ulik kovarians. Her kan QDA bruke sine kvadratiske beslutningsgrenser for å skille klassene bedre enn det LDA gjør med sin lineære beslutningsgrense. Laget etter inspirasjon fra Scikit-Learn brukermanual [28].*

Gaussian Naives Bayes

På samme måte som LDA og QDA tar også Gaussian Naives Bayes (GNB) utgangspunkt i Naives teorem (2.13) og en normalfordeling, men antar i tillegg at variablene i X er uavhengige av hverandre [22]. Dette gjør at GNB kan bruke en univariat normalfordeling:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}} \tag{2.17}$$

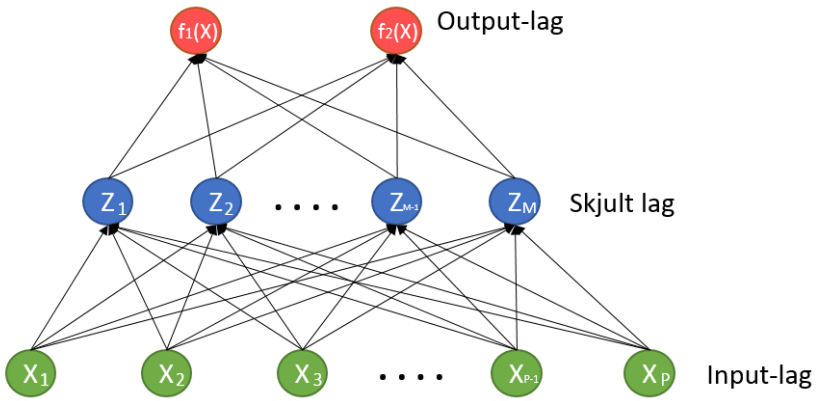
der μ_y er forventningsverdien til responsen $y$og σ_y er standardavviket til responsen y [28]. Dermed kan sannsynligheten for observasjon x_i regnes ut, gitt at den tilhører klasse k. GNB bruker dette, for å klassifisere observasjon x i klassen som er mest sannsynlig [28]:

$$\hat{y} = argmax_y P(y)\Pi_{i=1}^{n} P(x_i|y) \tag{2.18}$$

I denne oppgaven ble Scikit-Learn sin pakke GaussianNB brukt [23].

Nevrale Nettverk

Hovedideen bak Nevrale Nettverk (Neural Networks) er å trekke ut lineære kombinasjoner av dataene, og deretter lage en ikke-lineær modell av disse kombinasjonene [22]. Resultatet er en kraftig læringsmetode, utbredt innenfor mange områder. Mellom dataene som puttes inn (input-laget), og klassifikasjonen som kommer ut (output-laget), har Nevrale Nettverk ett eller flere ikke-lineære lag kalt *skjulte lag* (hidden layers).



Figur 2.17: *En skjematisk tegning over et neuralt nettverk med binær klassifisering og ett skjult lag. Laget etter inspirasjon fra [22].*

Figur 2.17 viser en skjematisk tegning over et Nevralt Nettverk med ett skjult lag og en binær klassifisering. Input-laget består av et sett med nerveceller $X = \{x_1, x_2, ..., x_i\}$ som representerer dataene som puttes inn [28]. Hver nervecelle i det skjulte laget transformerer nervecellene i det forrige laget ved vektede, lineære kombinasjoner $\alpha_{0m} + \alpha_m^T X, m = 1, ..., M$, etterfulgt av en ikke-lineær aktivasjonsfunksjon $\sigma(v)$, slik at

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, ..., M \tag{2.19}$$

der $Z = (Z_1, Z_2, ..., Z_M)$ og M er antall nerver i det skjulte laget [22] [28]. Antall nerver i det skjulte laget trenger ikke være det samme som antallet nerver i input-laget. Aktivasjonsfunksjonen er ofte satt til å være sigmoid-funksjonen $\sigma(v) = 1/(1+e^{-v})$. Deretter lager output-laget lineære kombinasjoner av nervecellene i det skjule laget:

$$T_k = \beta_{0k} + \beta_k^T Z, k = 1, ..., K \tag{2.20}$$

der $T = (T_1, T_2, ..., T_K)$, K er antall klasser. Deretter blir observasjonen tilegnet klassen med høyest sannsynlighet, som oftest gjennom sigmoid-funksjonen eller softmax-funksjonen [22]:

$$f_k(X) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}, k = 1, ..., K} \tag{2.21}$$

I denne oppgaven ble Scikit-Learn sin pakke MLPClassifier brukt [23].

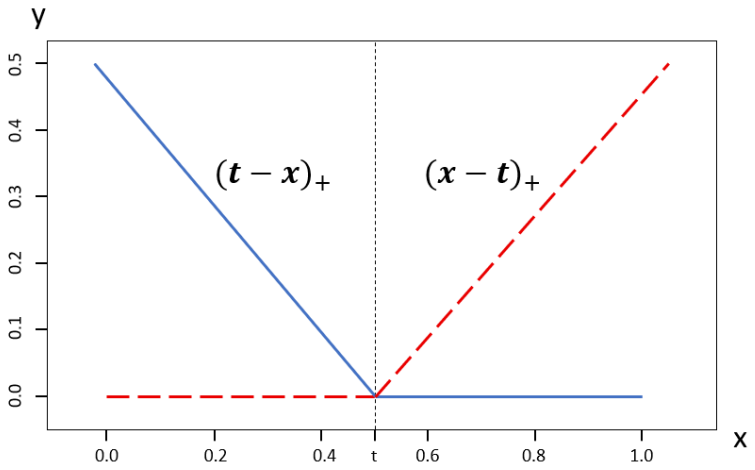
Multivariate Adaptive Regression Splines

Multivariate Adaptive Regression Splines (MARS) er en tilpasningsdyktig algoritme for regresjon, som er godt egnet for høydimensjonale problemer [22]. MARS bruker stykkvise, lineære funksjoner på formen $(x-t)_+$ og $(t-x)_+$. "+betyr den største verdien (eller den positive delen). Funksjonene er gitt som:

$$(x-t)_+ = \left\{ \begin{array}{cc} x-t, & hvis\ x > t \\ 0, & ellers \end{array} \right\} \tag{2.22}$$

$$(x-t)_- = \left\{ \begin{array}{cc} t-x, & hvis\ x < t \\ 0, & ellers \end{array} \right\} \tag{2.23}$$

Figur 2.18 er en illustrasjon av funksjonene $(x-0.5)_+$ og $(0.5-x)_-$ som eksempel. Disse to funksjonene kalles et reflektert par.



Figur 2.18: *Funksjonene $(t-x)_+$ (blå linje) og $(x-t)_+$(rød, stiplet linje). Tegnet etter figur fra Hastie et al. [22].*

Hver funksjon er lineær, men med et knutepunkt ved verdien t. Ideen bak MARS er å lage reflekterte par for alle egenskapene x_j med knutepunkt for hver observasjon x_{ij}. Dette gir samlingen av basisfunksjoner:

$$C = \{(X_j - t)_+, (t - X_j)_+\} \tag{2.24}$$

der $t \epsilon \{x_{1j}, x_{2j}, ..., x_{Nj}\}$ og $j = 1, 2, ..., p$, N er antall observasjoner og p er antall variabler [22]. Selve byggingen av modellen foregår som en fremover stegvis lineær regresjon, men istedet for å bruke det opprinnelige datasettet, brukes funksjoner fra samlingen C og deres produkter. Modellen blir på formen [22]:

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X) \tag{2.25}$$

der β_0 er skjæringspunktet med y-aksen, $h_m(X)$ er en funksjon i C eller et produkt av funksjoner i C, og har fått koeffisienten β_m, og M er antall funksjoner modellen

består av. Algoritmer for regresjon kan også brukes for klassifikasjon, ved å sette en terskel for hvilke verdier som gir klasse 0 og hvilke verdier som gir klasse 1.

I denne oppgaven ble Py-earth sin pakke Earth brukt [29], som er en python-pakke som implementerer Jerome H. Friedmans Multivariate Adaptive Regression Splines [30].

PLSR

PLSR (Partial Least Squares Regression) er enda en metode som konstruerer et sett med lineære kombinasjoner av dataene [22]. I motesetning til PCA (som blir beskrevet senere) bruker PLSR i tillegg responsvektoren y for å danne disse lineære kombinasjonene. Denne prosessen er beskrevet i detalj i følgende algoritme:

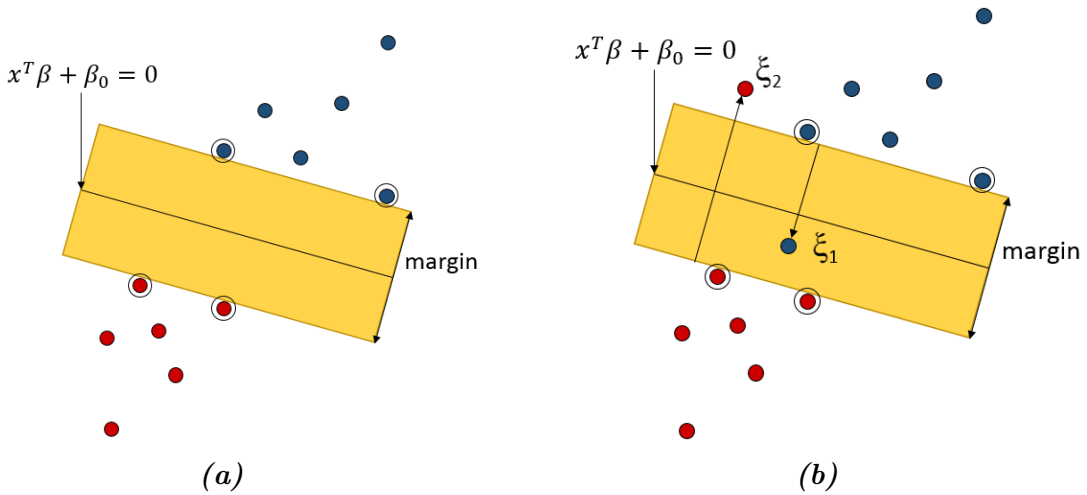
1. Standardiser hver x_j til å ha gjennomsnitt lik null og varians lik en. Sett $\hat{y}^{(0)} = \bar{y}\mathbf{1}$, og $x_j^{(0)} = x_j, j = 1, ..., p$, hvor p er antall PLS-retninger.
2. For $m = 1, 2, ..., p$
   (a) $z_m = \sum_{j=1}^{p} \hat{\varphi}_{mj} x_j^{m-1}$, der $\hat{\varphi}_{mj} = \langle x_j^{m-1}, y \rangle$, der z_m er PLS-retning m.
   (b) $\hat{\theta}_m = \langle z_m, y \rangle / \langle z_m, z_m \rangle$.
   (c) $\hat{y}^{(m)} = \hat{y}^{(m-1)} + \hat{\theta}_m z_m$.
   (d) Ortogonaliser hver $x_j^{(m-1)}$ med respekt på z_m, med $x_j(m) = x_j^{(m-1)} - [\langle z_m, x_j^{(m-1)} \rangle / \langle z_m, z_m \rangle] z_m, j = 1, 2, ..., p$.
3. Resultatet er en tilpasset responsvektor $\hat{y}^{(m)} = X\hat{\beta}^{pls}$ [22].

I punkt 2a konstrueres PLS-retningen z_m ved å summere alle indreproduktene mellom variablene og responsen. I punkt 2b og c blir koeffisienten til denne retningen $\hat{\theta}_m$ regnet ut ved regresjon mellom responsen og retningen, og en ny respons-vektor $\hat{y}^{(m)}$ blir beregnet. I punkt 2d blir egenskapene ortogonalisert med hensyn på PLS-retningen. PLSR søker altså etter retninger som både har høy varians og høy korrelasjon med responsen y.

I denne oppgaven ble Scikit-Learn sin pakke PLSRregression brukt [23].

Support vector classifiers

Support vector classifiers (SVC) er en teknikk for å konstruere et separerende hyperplan mellom to klasser [22]. Målet er å maksimere marginen. Marginen er definert som avstanden mellom hyperplanet, som skiller de to klassene fra hverandre, og treningsobservasjonene som er nærmest hyperplanet. Disse treningspbservasjonene er



Figur 2.19: *En illustrasjon over hvordan SVC fungerer. Figur (a) viser tilfellet der klassene er lineært separable mens figur (b) viser tilfellet der klassene ikke er lineært separable. Observasjoner omringet av en sort sirkel er støttevektorene, som gir posisjonen til marginen som skiller klassene. Tegnet etter inspirasjon fra figur i Hastie et al. [22].*

de såkalte støttevektorene (support vectors) [7]. Tanken bak å ha størst mulig marginer er at de vanligvis har lavere generaliseringfeil, ettersom modeller med mindre marginer raskere kan overtilpasse treningsdataene [7]. Figur 2.19 er en illustrasjon over en SVC der (a) klassene er separable, og (b) et eksempel der de ikke er separable.

Anta at et treningssett består av N par med observasjoner $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, der $x_i \epsilon \mathbb{R}^p$ er observasjon i og $y_i \epsilon \{-1, 1\}$ er responsen til observajon i. Definer et hyperplan som:

$$\{x \ : \ f(x) = X^T\beta + \beta_0 = 0\} \tag{2.26}$$

der β er en enhetsvektor: $||\beta|| = 1$ [22]. Da er den predikerte klassen $G(x_i)$ til observasjon x_i definert som:

$$G(x_i) = sign[x^T\beta + \beta_0] \tag{2.27}$$

For å lage den største marginen M i tilfeller der klassene er separable, får vi følgende optimeringsproblem:

$$max_{\beta,\beta_0}(\quad M) \tag{2.28}$$

med betingelsene:

$$y_i(x_i^T\beta + \beta_0) \geq M, i = 1, ..., N$$
$$||\beta|| = 1$$

som gjør at ingen av punktene skal falle inni marginen (se figur 2.19a).

Dersom klassene derimot ikke er separable (figur 2.19b), går det ikke an å lage et hyperplan som skiller de to klassene. Et alternativ er å fortsette å maksimere marginen, men la et par punkter være på feil side av hyperplanet. Da må begrensningen

til (2.28) modifiseres til:

$$y_i(X_i^T\beta + \beta_0) \geq M(1 - \xi_i)$$

der alle $\xi_i \geq 0$ er ξ_i andelen av observasjoner der prediksjonen $f(x_i) = x_i^T\beta + \beta_0$ kan være på feil side av marginen og $\sum_{i=1}^{N} \xi_i \leq konstant$.

SVC trenger ikke være kun en lineær funksjon. Vi kan få ikke-lineære marginer ved å bytte ut alle indreprodukter med en ikke-lineær kjernefunksjon. I denne oppgaven er en lineær SVC og en SVC med RBF-kjerne (Radial Basis Function) brukt fra Scikit-Learn sine pakker SVC og LinearSVC [23]. RBF-kjernen er gitt ved:

$$k(x_i, x_j) = exp(-\gamma||x_i - x_j||^2) \tag{2.29}$$

der γ er en skalerbar konstant som er større enn 0 [28].

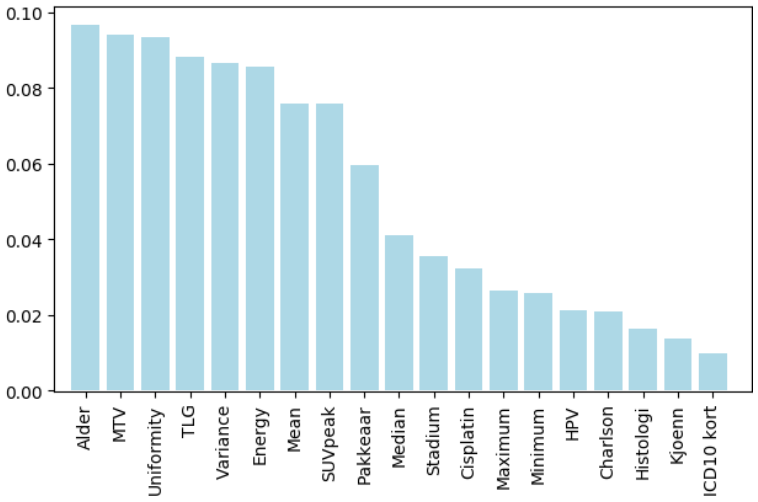
2.3.4 Egenskapsutvelgere og dimensjonsreduksjonsmetoder

En utfordring innen maskinlæring er å plukke ut relevante egenskaper til å bruke i modellene [31]. I de fleste datasett er kun en del av egenskapene/variablene relevante for responsen. De irrelevante egenskapene skaper flere dimensjoner til modellen og kan innføre støy. Ved for mange dimensjoner kan *Dimensjonalitetens forbannelse* (The curse of dimensionality) oppstå [22]. Kort forklart betyr dette at antall egenskaper som trengs for å gi en god prediksjon vokser eksponensielt med antall dimensjoner i modellen. Det er derfor viktig å finne gode egenskapsutvelgere/dimensjonsreduksjonsmetoder som finner de egenskapene i datasettet som er viktig for å predikere responsen.

Hvor godt klassifieringsalgoritmene tåler å ha med egenskaper som kun tilfører støy varierer veldig. Lineær regresjon, PLSR, Nevrale Nett og SVC er klassifikasjonsalgoritmer som oftest håndterer ikke-informative egenskaper dårlig. Derimot håndeterer MARS, beslutningstrær og RF ofte dette bedre [26]. Under følger de egenskapsutvelgerne/dimensjonsreduksjonsmetodene som er blitt brukt i denne oppgaven.

Random Forests

Random Forests kan også brukes som en egenskapsutvelger, ved at viktigheten av de forskjellige egenskapene måles [7]. Viktigheten av en egenskap måles ved å ta gjennomsnittet av reduksjonen av urenhet beregnet på alle beslutningstrærne der en node splittes med hensyn på den egenskapen. Etter at dette er regnet ut for alle egenskapene, velges de egenskapene som gjennomsnittlig gav størst reduksjon i urenhet. Viktigheten til egenskapene kan plottes som vist i figur 2.20.



Figur 2.20: *Et eksempel på viktigheten til noen tilfeldige egenskaper regnet ut ved hjelp av RF. Y-aksen viser viktigheten til hver av egenskapene, og disse summeres til 1.*

ReliefF

Den originale versjonen av Relief er sjeldent i bruk i dag. Derimot er ReliefF (den sjette versjonen av Relief) en mer populær algoritme [31]. ReliefF er en egenskapsutvelger som bygger på prinsippet til KNN [32]. For et tilfelle F_i (observasjon i), søker ReliefF etter de k-nærmeste naboene fra samme klasse kalt *treff* T_j, og de k-nærmeste naboene fra den andre klassen kalt *bom* B_j. Deretter oppdateres vektene $W[A]$ for alle egenskaper A på bakgrunn av alle verdier for F_i, T_j og B_j. Dette er forklart i detalj for et to-klasse-tilfelle i følgende algoritme [33]:

1. Sett alle vektene $W[A] = 0$
2. For $i = 1, 2, ..., m$, hvor m er antall iterasjoner som skal gjennomføres:
   (a) Velg et tilfeldig tilfelle F_i.
   (b) Finn alle k nærmeste treff H_j .
   (c) Finn alle k nærmeste bom B_j.
   (d) For $A = 1, 2, ..., a$, hvor a er antall egenskaper:
      i. $W[A] = W[A] - \sum_{j=1}^{k} diff(A, F_i, T_j)/(m \cdot k) + \sum_{j=1}^{k} diff(A, F_i, B_j)/(m \cdot k)$.

der m er antall ganger iterasjonen skal gjennomføres og a er antall egenskaper. I punkt 3a får de egenskapene som hadde gitt observasjonen riktig klasse høyere vekt, mens de egenskapene som hadde gitt observasjonen feil klasse får laverer vekt. De egenskapene som har fått høyest vekt, er de egenskapene som ifølge ReliefF predikerer responsen best.

I denne oppgaven er ble Scikit-rebate sin pakke ReliefF brukt [34].

Logistisk Regresjon

Som tidligere nevnt kan logistisk regresjon med en L1-straffeparameter (lasso) (2.9) også brukes som en egenskapsutvelger. Ved å bruke en L1-straffeparameter vil noen egenskaper krympes til 0 ved bruk av en stor nok kompleksitetsparameter λ. Ved bruk av logistisk regresjon som egenskapsutvelger beholder man de egenskapene som ikke ble krympet til 0.

I denne oppgaven ble logistisk regresjon i kombinasjon med en RFE (recursive feature elimination) brukt [28]. Dette betyr at den logistiske regresjonen tildeler egenskapene koeffisienter i forhold til hvor viktige de er for modellen, og RFE velger egenskaper med å gjentagende se på mindre og mindre delmengder av disse egenskapene. For hver gjentagende gang ble de egenskapene som var minst viktige for modellen (de som var krympet nærmest 0) eliminert.

I denne oppgaven ble Scikit-Learn sin pakke RFE brukt [23].

Mutual Information

Mutual Information (MI) er et mål på avhengigheten mellom to variabler [28]. MI innebærer metoder basert på entropi-estimasjon (mengden informasjon) fra KNN-avstander. En annen måte å beskrive MI på er at den er et mål på hvor mye informasjon vi får om en tilfeldig variabel gjennom en annen variabel. Den mest utbredte måten å estimere MI på er å kvantifisere en variabel x og responsen y til diskrete verdier og approksimere MI med en endelig sum [35]:

$$I(x,y) \approx I_{binned}(x,y) = \sum_{ij} p(i,j) log \frac{p(i,j)}{p_x(i)p_y(j)} \tag{2.30}$$

der $p_x(i) = \int_i dx\mu_x(x)$, $p_y(j) = \int_j dy\mu_y(y)$, $p(i,j) = \int_i \int_j dxdy\mu(x,y)$ og $\int_i$ betyr integralet over bin i. i står for bin nummer i for verdiene i x, og j står for bin nummer j for verdiene i y. $\mu_x(x)$ er tetthetsfunksjonen til variabel x og $\mu_y(y)$ er tetthetsfunksjonen til responsen y.

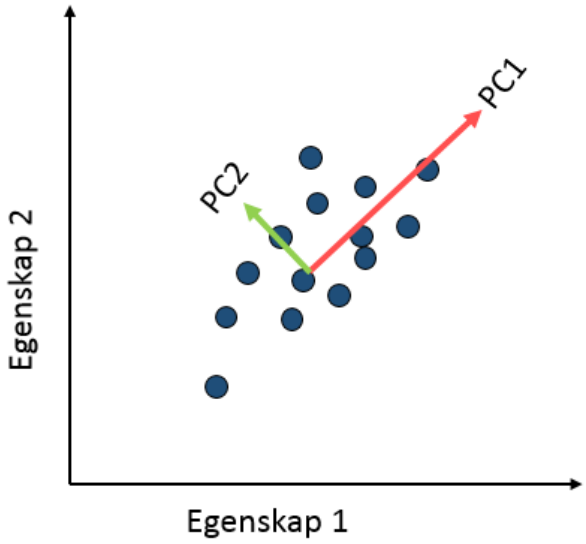
I denne oppgaven ble Scikit-Learn sin pakke mutual_info_classif i kombinasjon med Scikit-Learn sin pakke SelectKBest brukt [23].

Linear Discriminant Analysis

Som tidligere nevnt kan LDA ligning (2.15) brukes som en dimensjonsreduksjonsmetode før en klassifiering [28]. Da finner algoritmen lineære kombinasjoner som optimerer separerbarheten mellom klassene som vanlig. De nye, lineære kombinasjonene brukes da som de nye egenskapene i klassifiseringsalgoritmene. I tilfellet der det er kun to klasser som respons, vil LDA kun gi ut én variabel, siden maks antall kombinenter er antall klasser minus en.

Principal Component Analysis

PCA (Principal Component Analysis) er en dimensjonsreduksjonsmetode, som transformerer et datasett til ortogonale komponenter kalt prinsipalkomponenter [7]. Målet med PCA er å finne retningene med maksimal varians, og projisere de originale dataene ned på det nye underrommet (subspace) som er spent ut av disse retningene/prinsipalkomponentene. Den første prinsipalkomponenten er den med størst varians, og de påfølgende prinsipalkomponentene er de med størst mulig varians, med begrensningen at de skal være ortogonale på de tidligere prinsipalkomponentene. Figur 2.21 viser et eksempel på hvordan de to første prinsipalkomponente ville blitt for et tilfeldig datasett.



Figur 2.21: *En illustrasjon av hvordan to prinsipalkomponenter dannes for å forklare mest mulig av variansen i dataene. Den røde pilen forklarer mest av variansen og blir prinsipalkomponent 1, mens den grønne, som er ortogonal på den røde prinsipalkomponenten, blir prinsipalkomponent 2.*

Der LDA er en såkalt veiledet lærer, som reduserer dimensjoner med hensyn på responsen y, er PCA en ikke-veiledet lærer, som bare ser på variansen i egenskapsmatrisen X, og ikke responsen y.

Den første prinsipalkomponenten til et datasett vil være normaliserte, lineære kombinasjoner av de originale egenskapene på formen

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ...\phi_{p1}x_{ip} \tag{2.31}$$

som har den største variansen, og er begrenset med $\sum_{j=1}^{p} \phi_{j1}^2 = 1$ [36]. Her blir z_{i1} den første prinsipalkomponenten, $\phi_{11}, ..., \phi_{p1}$ er ladningene som danner PCA-ladningsvektoren $\phi_1 = (\phi_{11}\phi_{21}...\phi_{p1})$ og $x_{i1}, ..., x_{ip}$ er de normaliserte, originale egenskapene. Mer konkret må ladningsvektoren til den første prinsipalkomponenten løse følgende optimeringsproblem:

$$max_{\phi_{11},...,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (\sum_{j=1}^{p} \phi_{j1}x_{ij})^2 \right\} \quad der \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1 \tag{2.32}$$

Deretter blir den neste prinsipalkomponenten beregnet på samme måte, men med den ekstra begrensningen at den må være ortogonal på den første prinsipalkomponenten.

I denne oppgaven ble Scikit-Learn sin pakke PCA brukt [23].

Independent Component Analysis

Independent Component Analysis (ICA) er enda en ikke-veiledet dimensjonsreduksjonsmetode. Den separerer multivariate signaler til additive komponenter som er maksimalt uavhengige av hverandre [28]. ICA er vanligvis brukt til å skille signaler som er blandet sammen. Eksempler på dette er å skille ut forskjellige lydkilder fra et lydklipp eller skille forskjellige signaler i hjernen [22].

ICA bruker at egenskapene i observasjonsmatrisen X er lineære kombinasjoner av vektede, uavhengige komponenter $s_k, i = 1, 2, ..., k$ [22]. Dette kan skrives som:

$$x_i = a_{i1}s_1 + a_{i2} + ... + a_{ip}s_p \tag{2.33}$$

der a_{ik} er vekten til observasjon i og den uavhengige komponenten k [22]. Å løse selve ICA-problemet betyr å finne en ortogonal $A = (a_{1,k}, ..., a_{1,m})$ slik at komponentene i variabelen $S = A^T X$ er uavhengige og ikke normalfordelt. Mange av de mest brukte metodene med ICA er basert på entropi og MI (2.30).

I denne oppgaven ble Scikit-Learn sin pakke FastICA brukt [23].

Kapittel 3

Metode

3.1 Programvare

3.1.1 Databehandling og analyse

Python versjon 3.6, i kombinasjon med pakken Scikit-Learn [23] versjon 0.19.1, ble brukt til databehandlig og analyse. Python er et programmeringsspåk med åpen kilde [37]. Dette betyr at kildekoden til dataprogrammet er gjort tilgjengelig for alle. Python er brukt i tusenvis av applikasjoner verden over, som Youtube og Google, og blir benyttet ved flere universiteter [37]. Python regnes som det raskest voksende programmeringsspråket, med bakgrunn i at det er en stor økning i andelen viste spørsmål om Python på internett [38] [39]. En mulig årsak til dette, kan være økningen i bruk av maskinlæring, og at Python er et programmeringsspråk mange velger å bruke innenfor dette feltet [40].

Scikit-Learn er en tilleggspakke for Python med åpen kilde, som inneholder en rekke verktøy innen dataanalyse og maskinlæring [28]. Eksempler på dette er egenskapsutvelgelse, dimensjonsreduksjon, regresjonsalgoritmer og klassifikasjonsalgoritmer. Scikit-Learn tilbyr et bredt spekter av de nyeste og mest moderne maskinslæringsalgoritmene, i tillegg til de mest kjente [24].

3.1.2 Bildeanalyse

Tilleggpakken PyRadiomics er brukt til å trekke ut egenskaper fra PET- og CT-bildene. PyRadiomics er en pythonpakke med åpen kildekode, som brukes til å trekke ut egenskaper (såkalte "features") fra medisinske bilder [23]. PyRadiomics har som mål å etablere standarder for radiomics-analyser og gi en testet og vedlikeholdt platform med åpen kilde for enkle og reproduserbare egenskapsuttrekninger (såkalt "feature extractions"). PyRadiomics har en del innebygde preprosesseringer av bilder

(beskjæring, kvantifisering av gråtonenivåer og en rekke filtre) i tillegg til selve egenskapsuttrekningssalgoritmene. Radiomics blir nærmere beskrevet i kapittel 3.3.

En del Python-program ble utviklet for å utføre analysene. Fire av hovedprogrammene ligger vedlagt i vedlegg B.

- Python-program B.1 endrer bildefilene fra Matlab-filer til nrrd-filer og lager maskene til bildene. I tillegg lages et excel-ark med kolonner som inneholder pasient ID, plasseringen til bildene og plasseringen til maskene.
- Python-program B.2 trekker ut egenskaper fra alle PET- og CT-bildene ved hjelp av PyRadiomics og excel-arket laget i program B.1.
- Python-program B.3 er en "pakkesom inneholder alle klassifiseringsalgoritmene og egeneskapsutvelgerne som funksjoner.
- Python-program B.4 bruker funksjonene i skript B.3 til å klassifiere behandlingsutfallet ved hjelp av bildeegeskapene. Den kan også brukes for finne optimalt antall egenskaper til modellene.

3.2 Datasett

Datasettet, som ble analysert i denne oppgaven, består av klinisk informasjon og PET- og CT-bilder av pasienter som har fått behandling for hode- og halskreft med behandlingsstart i perioden 2007-2013 på Oslo Universitetssykehus. Datasettet inneholdt 254 pasienter som var diagnostisert med hode/hals-kreft og hadde tatt en kombinert PET/CT-skanning fra skuldrene og opp før behandlingsstart [9].

3.2.1 Bildedata

PET- og CT-bildene ble mottatt i form av MATLAB-filer, som Kari Kvandal utviklet fra tekstfiler fra Oslo Universitetssykehus i sin masteroppgave [9]. Disse tekstfilene inneholdt noe kliniske informasjon om pasientene i tillegg til CT-tall og SUV-verdier for voxler tilhørende tumor. De inneholdt også informasjon om oppløsning på bildene og størrelse på voxlene.

PET- og CT-bildene hadde størrelser mellom $341 \times 341 \times 341\,mm^3$ og $682 \times 682 \times 396\,mm^3$. Oppløsningen for PET- og CT-bildene var i utgangspunktet forskjellige (hhv. $3\times3\times2\,mm^3$ og $1\times1\times2\,mm^3$), men ble mottat i oppløsning med voxelstørrelser på $1 \times 1 \times 1\,mm^3$. Disse hadde gjennomgått en glatting og interpolasjon ved Oslo Universitetssykehus før overleverelse [8].

Tumorene i PET- og CT-bildene var tegnet inn av onkologer, og bildene inneholdt bare verdier fra de opprinnelige PET- og CT-bildene innenfor inntegnet område.

Verdiene utenfor dette området var med i bildene, men var satt til verdi 0. Ved analyse av bildene ble en maske brukt, som markerte området onkologen hadde tegnet inn. Dermed ble bare verdiene innenfor inntegnet området med i analysene. Inntegnet området kalles for ROI (Region of Interest).

3.2.2 Klinisk informasjon

I tillegg til bildedataene, ble det brukt et regneark med anonymisert informasjon om hver pasient. Tabell 3.1 gir en oversikt over hvilke faktorer som ble brukt i analysene fra regnearket. De tre nederste faktorene (MTV, TLG og SUVpeak) er PET-parametere, og ble inkludert blandt PET-egenskapene i analysene, og ikke blant de kliniske faktorene.

Tabell 3.1: *Pasientinformasjonen som ble hentet fra det anonymiserte regnearket. Informasjonen i tabellen, som ikke viser til en referanse, er hentet fra Grøndahl [41].*

Faktor	**Forklaring**
Pasient-ID	Et unikt pasientnummer mellom 1-256.
Alder	Alderen på pasienten ved behandlinggstart.
Kjønn	Pasientens kjønn.
ICD10 kort	Diagnose i henhold til systemet ICD-10-klassifikasjonen [42]. Angir tumors plassering.
T-stadium	Representerer omfanget av primærtumor.
N-stadium	Representerer spredning til regionale lymfeknute på halsen.
Stadium	Sykdommens stadium.
Histologi	Histologisk diagnostikk basert på vevsprøve.
HPV-status	Om pasienten har HPV.
ECOG	Eastern Cooperative Oncology Group. Beskriver allmenntilstand ved behandlingsstart [42].
Charlson	Brukes for å anslå overlevelse for pasienter med kormorbiditet.
Pakkeår	1 pakkeår tilsvarer 20 sigaretter hver dag i ett år.
Naxogin dager	Antall dager pasienten har tatt Naxogin. Naxogin er et medikament som forsterker effekten av strålebehandling i enkelte celler [43].
Cispatin	Antall kurer med Cispatin. Cispatin er en cellegift, som brukes for å svekke og drepe kreftceller [44].
MTV	Metabolic Tumor Volume. Tumorvolum basert på autoterskling. Her telles en voxel som del av MTV dersom den er en del av primætrumor og har $SUV \geq SUV_{peak}$.

TLG	Total Lesion Glycolysis. Mål på metabolsk aktivitet i tumor. $TLG = MTV \times SUV_{mean}$.
SUVpeak	Maksimal gjennomsnittlig SUV i et sfærisk volum på 1 cm^3.

Pasientene hadde pasientnummer fra 1 til 256, men pasient nummer 7 og 85 manglet, som gjorde at det var 254 pasienter totalt. Behandlingsutfallet til pasientene var også gitt i regnearket, og vises i tabell 3.2.

Tabell 3.2: *Tabellen viser antallet pasienter som fikk lokalregionalt tilbakefall, metastase eller progresjonsfri overlevelse etter kreftbehandlingen i det originale datasettet og det endelige datasettet. I kolonnen Metastase er pasientene som kun fikk metastase og ikke lokalregionalt tilbakefall telt opp. 17 av pasientene som fikk lokalregionalt tilbakefall hadde også metastase.*

Behandlingsutfall	Lokalt/regionalt tilbakefall	Metastase	Progresjonsfri overlevelse
Totalt	66	14	174
Endelig datasett	47	0	135

I denne oppgaven ble behandlingsutfallet delt i to klasser: Lokalregionalt tilbakefall eller progresjonsfri overlevelse. Det betyr at de 14 pasientene som kun fikk metastase ikke ble tatt med i analysen. 36 av pasientene hadde ikke fått kontrastvæske før CT-skanningen, og disse ble også fjernet fra analysen, ettersom dette påvirker intensitetsverdiene i bildene. Kvandal viste i sin masteroppgave at bildene der pasientene hadde fått kontrastvæske skilte seg ut fra resten i et PCA-plott, og at histogrammene over intensitetsverdiene ser annerledes ut ved bruk av kontastvæske [9]. I tillegg var det en del av pasientene som ikke hadde en hovedtumor, men kun kreftinfilterte lymfeknuter. Disse ble også fjernet fra analysen, så det endelige datasettet inneholdt 182 pasienter.

Pasientene som er med i datasettet hadde tumor i forskjellige deler av hode- og halsregionen. Plasseringene er oppgitt i tabell 3.3 med antallet pasienter som hadde en tumor på den plasseringen.

Tabell 3.3: *Oversikt over plasseringen til tumor for pasientene. Tabellen viser antallet pasienter som hadde hovedtumor på de forskjellige plasseringene i det originale datasettet og det endelige datasettet.*

Plassering	Cavum-Oris	Oropharynx	Hypopharynx	Larynx
Totalt	21	185	20	28
Endelig datasett	16	132	14	20

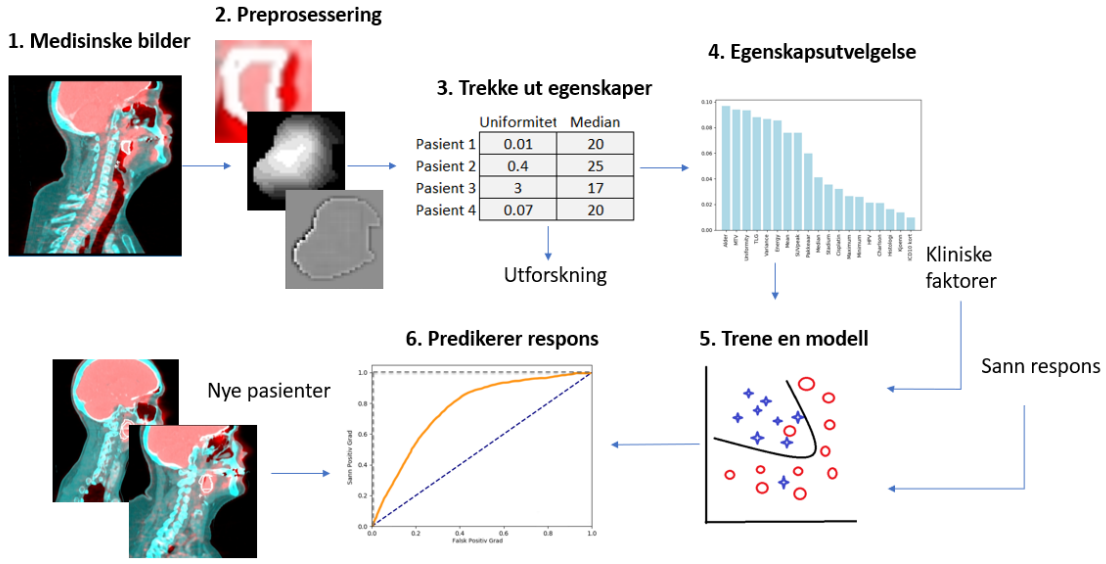
HPV-statusen var ukjent for mange av pasientene i datasettet. Tabell 3.4 viser antall pasienter med ukjent HPV-status, HPV positiv og negativ.

Tabell 3.4: *Oversikt over HPV-statusen til pasientene. Tabellen viser antallet pasienter som hadde ukjent, positiv eller negativ HPV-status i det originale datasettet og det endelige datasettet.*

HPV-status	Ukjent	Positiv	Negativ
Totalt	100	127	27
Endelig datasett	89	79	14

3.3 Radiomics

Radiomics er en metode for å konvertere digitale bilder til høydimensjonale data, som kan brukes for å predikere responser, eller til annen analyse [5]. Radiomics har oppstått på bakgrunn av konseptet om at medisinske bilder inneholder informasjon om en sykdom/skade som kommer til syne ved bruk av kvantitative bildeanalyser [5]. Fremgangsmåten i radiomics innebærer bruk av temaer beskrevet i avsnitt 2.3 og 3.4. Hvordan radiomics har blitt brukt i denne oppgaven er illustrert i figur 3.1.



Figur 3.1: *Flytskjema som viser fremgangsmåten for radiomics brukt i denne oppgaven. Prosessen starter ved å bruke de tilgjengelige, medisinske bildene, preprosessere disse, trekke ut egenskaper fra disse gjennom bildeanalyse, velge ut de beste egenskapene og trene en modell på bakgrunn av disse som kan brukes til prediksjon på nye bilder.*

Radiomics er på ingen måte kun knyttet til medisinske bilder, men kan brukes på en rekke problemstillinger. Likevel er dette foreløpig mest utviklet innen onkologien [5]. Innenfor kreftforskning er det vist at egenskaper trukket ut fra bilder, som informasjon om en tumors størrelse, intensitet, form eller tekstur, gir ekstra informasjon som skiller seg fra det kliniske rapporter og laboratorieforsøk gir [5]. Radiomics ser ut til å tilby en ubegrenset mengde med informasjon som kan være til støtte for å finne kreft, diagnostisere sykdommer, vurdering av prognoser, prediksjon av behandlingsutfall og overvåkning av sykdomsstatus [5].

3.4 Bildeanalyse

Bilde- og teksturanalyse er et felt innenfor bildebehandling som er mye benyttet til karakterisering, tolkning og analyse av digitale bilder [45]. Hensikten er å finne forskjeller som det menneskelige øyet ikke klarer å se. Ved bruk av matematiske algoritmer trekkes det ut egenskaper fra bilder, som kan brukes til videre analyse.

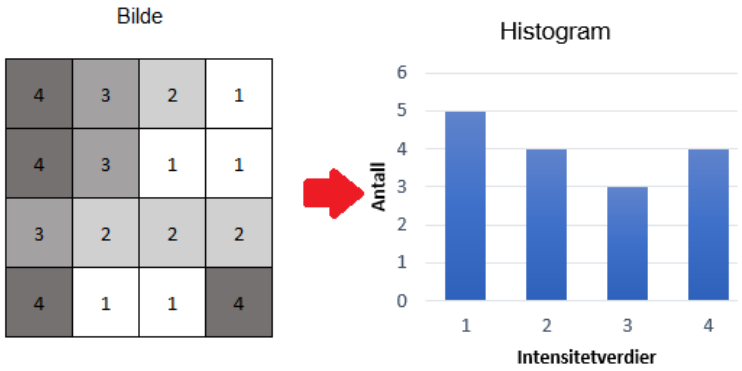
I denne oppgaven er tre forskjellige typer bildeegenskaper trukket ut av PET- og CT-bildene:

1. Førsteordens statistikk, som kun ser på intensitetsverdiene i bildet, og ikke tar for seg den romlige fordelingen av voxlene.
2. Form, som kun ser på 2D/3D-strukturen til ROI, og ikke på intensitetsverdiene.
3. Andreordens statistikk/teksturanalyse, som ser på intensitetsverdiene i bildet i sammenheng med den romlige fordelingen til voxlene.

I de neste avsnittene blir hver av disse metodene beskrevet nærmere. I denne oppgaven er de forskjellige egenskapene som trekkes ut gjennom bildeanalyse referert til med sitt originale, engelske navn, ettersom de fleste egenskapene ikke har norske navn.

3.4.1 Førsteordens egenskaper

Førsteordens egenskaper beskriver fordelingen av voxelintensiteter i bildet gjennom vanlige, statistiske mål. Førsteordens egenskaper baserer seg på bruk av et histogram som teller opp antallet voxler med de forskjellige intensitetsverdiene. Histogramverdien $P(i)$ er definert som antallet voxler med intensitetsverdi i [46] og er illustrert i figur 3.2 på et bilde med fire intensitetsverdier. Fra histogrammet ble det beregnet 15 egenskaper, som gjennomsnittlig intensitetsverdi, persentiler og fasong på histogrammet. En komplett liste med alle førsteordens egenskaper brukt i denne oppgaven er gitt i vedlegg A.1.



Figur 3.2: *Eksempel på hvordan et førsteordens histogram regnes ut fra et bilde med fire intensitetsverdier. Histogrammet viser antall ganger en intensitetsverdi forekommer i bildet.*

3.4.2 Formegenskaper

Formegenskaper beskriver den tre-dimensjonale størrelsen og fasongen på svulsten. Disse egenskapene er uavhengige av intensitetsverdiene til voxlene, og ser bare på den romlige posisjonen til voxlene. Det ble beregnet 13 formegenskaper, som tumors volum, rundhet og størrelse. En komplett liste med alle formegenskapene brukt i denne oppgaven er gitt i vedlegg A.2.

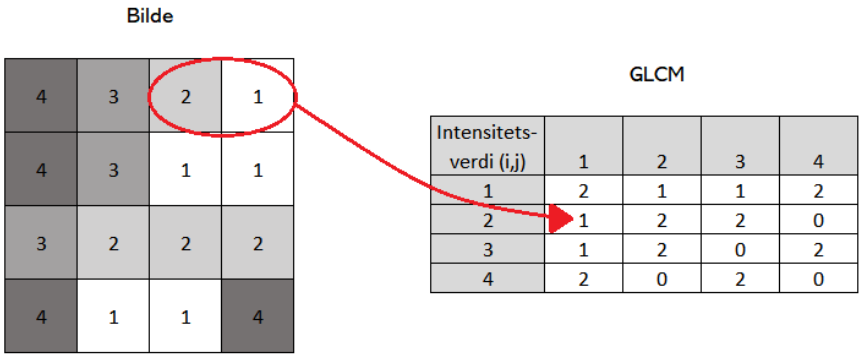
3.4.3 Gray level Co-occurence Matrix

Teksturanalyse ved bruk av en Gray Level Co-occurence Matrix (GLCM) er en statistisk metode som ser på de romlige forholdene mellom voxlene i et bilde [47]. GLC-matrisen karakteriserer teksturen i bildet ved å regne ut hvor ofte et par med voxler med spesifikke intensitetsverdier og en spesifikk avstand δ fra hverandre dukker opp i bildet.

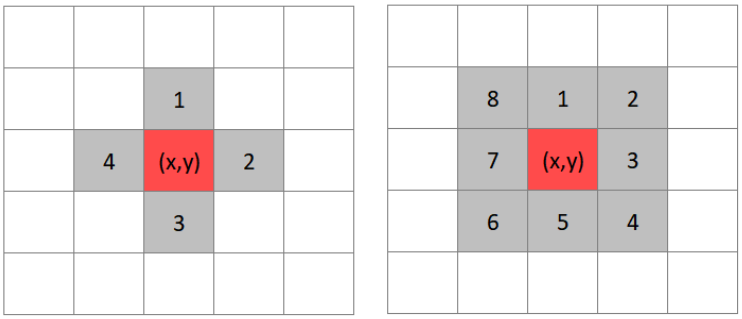
Figur 3.3 viser hvordan GLC-matrisen beregnes for et todimensjonalt bilde med fire intensitetsverdier. Størrelsen på en GLC-matrise fra et bilde med n antall diskrete intensitetsverdier blir $n \times n$. Basert på GLC-matrisen ble 23 teksturegenskaper beregnet, slik som korrelasjonen mellom intensitetsverdi og posisjonen til voxelen og homogenitet beregnet fra nabovoxlenes verdier. Den komplette listen med alle egenskaper er gitt i vedlegg A.3.

Hva som defineres som nabovoxler kan varieres for et bilde. Figur 3.4 viser et eksempel på et 4-naboskap og et 8-naboskap for et to-dimensjonalt bilde. Figur 3.5 viser et 6-naboskap, et 18-naboskap og et 26-naboskap for et tre-dimensjonalt bilde.

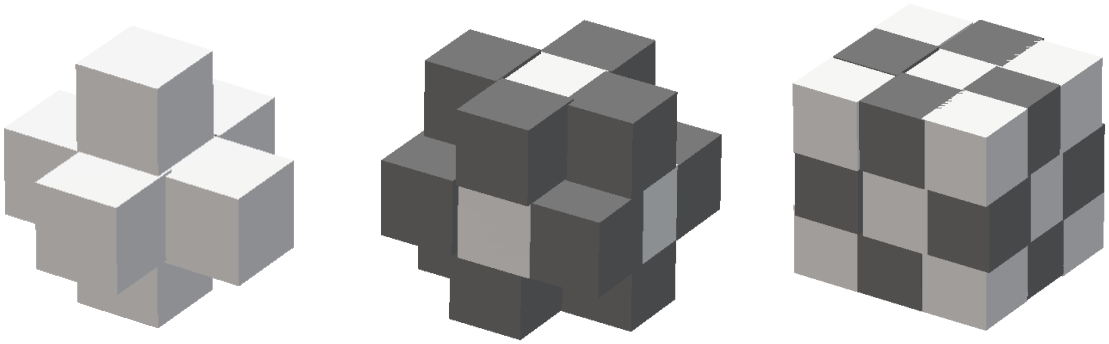
GLC-matrisene i denne oppgaven ble beregnet med avstand 1 mellom voxlene. Dette resulterer i to nabovoxler i 13 vinkler og et 26-naboskap [48]. Figur 3.6 viser hvordan et 8-naboskap for et todimensjonalt-bilde dekkes ved bruk av 4 vinkler, og ved overgangen til et tredimensjonalt bilde vil et 26-naboskap på samme måte bli dekket



Figur 3.3: *Et eksempel på hvordan en GLCM med avstand $\delta = 1$ mellom pixlene regnes ut fra et bilde med fire intensitetsverdier. Element (2,1) i matrisen inneholder verdien 1, ettersom det bare er ett tilfelle der intensitetsverdiene 1 og 2 dukker opp etter hverandre horisontalt i bildet.*



Figur 3.4: *4-naboskap (til venstre) og 8-naboskap (til høyre) i to dimensjoner.*



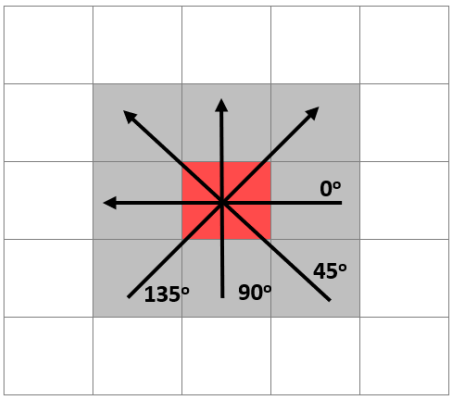
(a) *6-naboskap* ***(b)*** *18-naboskap* ***(c)*** *26-naboskap*

Figur 3.5: *Tre forskjellige type naboskap i et tredimensjonalt bilde.*

av 13 vinkler.

For hver av de 13 vinkelene ble en egen GLC-matrise beregnet. Til slutt ble gjennomsnittet av alle GLC-matrisene brukt for å beregne den endelige GLC-matrisen. Ingen vekting ble brukt, slik at GLC-matrisene for alle vinklene hadde et like stort bidrag.

GLC-matrisene i denne oppgaven beregnet symmetrisk GLCM. Det betyr at tilfeller

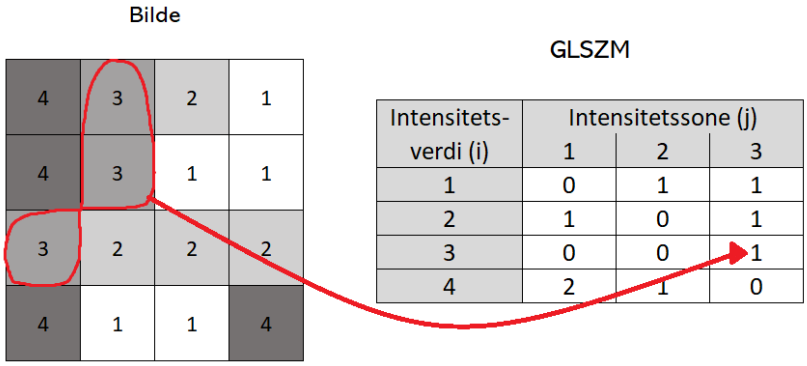


Figur 3.6: *Ved bruk av to piksler i fire retninger dekkes alle åtte naboer rundt en piksel for et todimensjonalt bilde.*

med like par blir beregnet i to retninger per vinkel [49]. Dette førte til en symmetrisk GLCM, der verdien for punkt (i, j) er den samme som verdien for punkt (j, i). Med andre ord: det blir det samme om intensitetsverdiene 1 og 2, eller 2 og 1 kommer etter hverandre. Figur 3.3 viser en symmetrisk GLCM, der verdiene er speilet om diagonalen.

3.4.4 Gray Level Size Zone Matrix

Teksturanalyse ved bruk av en Gray Level Size Zone Matrix (GLSZM) er en statistisk metode som ser på intensitetssoner i et bilde [49]. En intensitetssone er definert som koblede voxler som har samme intensitetsverdi. To voxler definerers som koblede hvis de har avstand 1 fra hverandre (i et 3D-bilde; hver voxel har 26 naboer med avtand 1). I en GLSZ-matrise $P(i, j)$ er verdien i punkt (i, j) lik antaller soner med intensitetsverdi i og størrelse j. GLSZ-matrisen er uavhengig av rotasjon, slik at det blir bare én matrise som er uavhengig av vinkel. Figur 3.7 viser hvordan verdier i en todimensjonal GLSZ-matrise regnes ut.



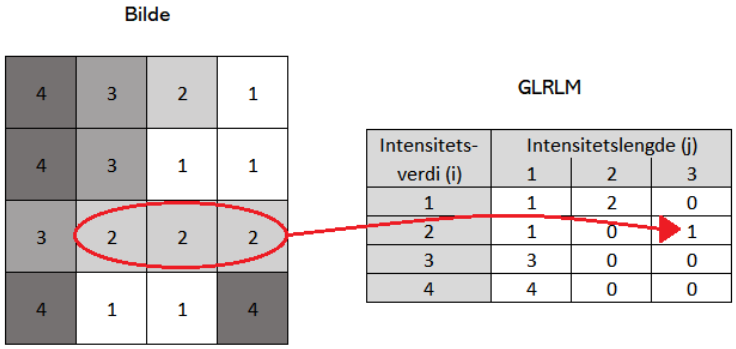
Figur 3.7: *Et eksempel på hvordan en GLSZM regnes ut fra et bilde med fire intensitetsverdier. Element (3,3) i matrisen inneholder verdien 1, ettersom det er én sone i bildet med intensitetsverdi 3 og størrelse 3.*

Basert på GLSZ-matrisen ble 16 teksturegenskaper beregnet. Disse beskriver egen-

skaper som variabiliteten i intensitetsverdier eller heterogenitet i sonestørrelser. Den komplette liste over alle egenskapene fra GLSZ-matrisen er gitt i vedlegg A.3.

3.4.5 Gray Level Run Length Matrix

Teksturanalyse ved bruk av en Gray Level Run Length Matrix (GLRLM) er en statistisk metode som ser på antallet intensitetsrekker i et bildet [50]. En intensitetsrekke defineres som en rekke med voxler som har samme intensitetsverdi. I GLRL-matrisen $P(i, j|\theta)$ er verdien i punkt (i, j) antallet intensitetsrekker med intensitetsverdi i og lengde j langs vinkel θ [49]. Figur 3.8 viser hvordan verdier i en to-dimensjonal-horisontal GLRL-matrise regnes ut. Horisontal betyr at den regnes ut for vinkel $0\,^\circ$ i forhold til figur 3.6.



Figur 3.8: *Et eksempel på hvordan en GLRLM regnes ut fra et bilde med fire intensitetsverdier. Element* $(2, 3)$ *i matrisen inneholder verdien 1, ettersom det bare er én intensitetsrekke med lengde 3 og intensitetsverdi 2 i bildet.*

Basert på GLRL-matrisen ble 16 teksturegenskaper beregnet. Disse beskriver egenskaper som fordelingen av høye intensitetsverdier og fordelingen av lange rekker med like verdier. Den komplette liste over alle egenskapene fra GLSZ-matrisen er gitt i vedlegg A.5.

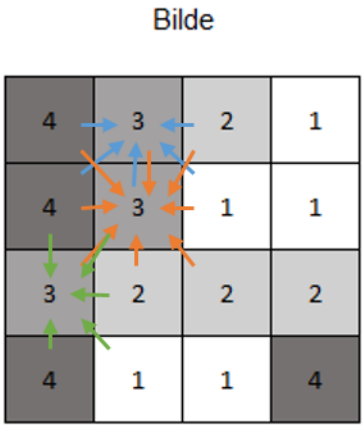
3.4.6 Neigbouring Gray Tone Difference Matrix

Teksturanalyse ved bruk av en Neigbouring Gray Tone Difference Matrix (NGTDM) er en statistisk metode som ser på forskjellen mellom en voxels intensitetsverdi og gjennomsnittsintensitetsverdien til voxelens naboer [49].

La X_{gl} være et sett med voxler og $x_{gl}(j_x, j_y, j_z)\,\epsilon\, X_{gl}$ være intensitetsverdien til voxelen i punkt (j_x, j_y, j_z). Da er gjennomsnittsintensitetsverdien til voxelens nabolag:

$$\bar{A}_i = \bar{A}(j_x, j_y, j_z) = \frac{1}{N} \sum_{k_x=-\delta}^{\delta} \sum_{k_y=-\delta}^{\delta} \sum_{k_z=-\delta}^{\delta} x_{gl}(j_x + k_x, j_y + k_y, j_z + k_z) \tag{3.1}$$

hvor $(k_x, k_y, k_z) \neq (0,0,0)$ og $x_{gl}(j_x + k_x, j_y + k_y, j_z + k_z) \in \mathbf{X}_{gl}$ [49]. N er antall voxler i nabolaget til voxelen og δ er avstanden nabovoxlene skal være innenfor. Figur 3.9 viser hvordan verdier i en to-dimensjonal NGTD-matrise regnes ut.



NGTDM

i	n_i	p_i	s_i
1	5	0,3125	4,53
2	4	0,250	0,90
3	3	0,1875	0,78
4	4	0,250	6,00

Figur 3.9: *Et eksempel på hvordan en NGTDM regnes ut fra et bilde med fire intensitetsverdier. Her er n_i antall voxler med intensitetsverdi i, p_i sannsynligheten for intensitetsverdi i og s_i er summen av absolutte forskjeller mellom intensitetsverdi i og nabolagene. Som eksempel finnes det tre voxler $(n_3 = 3)$ med intensitetsverdi 3 $(i \neq 0)$ av 16 voxler totalt. Det gir en sannsynlighet p_3 på $\frac{3}{16} = 0.1875$. Summen av absolutte forskjeller s_3 blir $|3 - \frac{4+4+3+1+2}{5}| + |3 - \frac{4+4+3+2+2+1+2+3}{8}| + |3 - \frac{4+4+1+2+3}{5}| = 0,78$.*

Sannsynligheten for intensitetsverdi i (p_i) beregnes ved å dele antallet voxler med intensitetsverdi i på antall voxler i bildet: $p_i = \frac{n_i}{N_v}$. Summen av absolutte forskjeller i nabolaget s_i regnes ut etter følgende ligning:

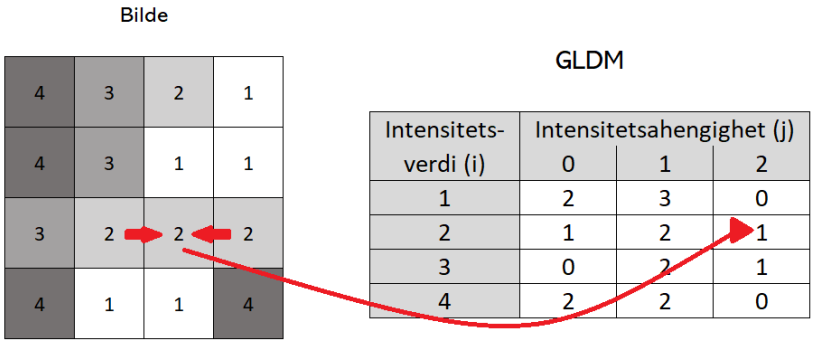
$$s_i = \begin{cases} \sum^{n_i} |i - \bar{A}_i| & \text{for} \quad n_i \neq 0 \\ 0 & \text{for} \quad n_i = 0 \end{cases} \tag{3.2}$$

I denne oppgaven er avstanden $\delta = 1$ mellom voxlene brukt, som gir et 26-naboskap. Basert på NGTD-matrisen ble 5 teksturegenskaper beregnet. Disse beskriver egenskaper som grovhet, kontrast og kompleksitet. Den komplette liste over alle egenskapene fra NGTD-matrisen er gitt i vedlegg A.6.

3.4.7 Gray Level Dependence Matrix

Teksturanalyse ved bruk av en Gray Level Dependence Matrix (GLDM) er en statistisk metode som ser på intensitetsavhengigheter i et bilde [47]. En intentitetsavhengighet blir definert som tilkoblede voxler innenfor avstanden δ som er avhengig av hovedvoxelen. En nabovoxel med intensitetsverdi j er avhengig av hovedvoxelen med intensitetsverdi i hvis $|i - j| \leq \alpha$, der α er en valgfri konstant [49]. I en GLD-matrise $P(i,j)$ beskriver verdien i element (i,j) antall ganger en voxel med intensitetsverdi i med j antall avhengige voxler i nabolaget inntreffer.

Figur 3.10 viser hvordan verdier i en todimensjonal GLD-matrise regnes ut.



Figur 3.10: *Et eksempel på hvordan en GLDM regnes ut fra et bilde med fire intensitetsverdier. Her er forskjellen mellom intensitetsverdiene $\alpha = 0$ og avstanden mellom tilkoblede voxler $\delta = 1$. Element $(2, 2)$ i matrisen inneholder verdien 1, ettersom det er kun én voxel med verdi 2 som har 2 avhengige voxler i bildet.*

I denne oppgaven er avstanden δ satt til 1, og grenseverdien α er satt til 0. Dette betyr at avhengige voxler er voxler med samme intensitetsverdi, innenfor et 26-naboskap. Basert på GLD-matrisen ble 14 teksturegenskaper beregnet. Disse beskriver egenskaper som variasjonen i intensitetsverdier og homogenitet i avhengigheter. Den komplette listen over alle egenskapene fra GLD-matrisen er gitt i vedlegg A.7.

3.5 Standardisering

Alle egenskapene (variablene i datamatrisen X) ble standardisert før de ble brukt i analyse. Mesteparten av egenskapsutvelgere og klassifikasjonsalgoritmer fungerer mye bedre dersom egenskapene er i samme skala [7]. RF og beslutningstrær er unntak fra dette. Standardisering går ut på å sentrere hver egenskap til å ha gjennomsnitt 0 og standardavvik 1. Standardiseringen følger følgende likning:

$$x_{std} = \frac{x - \mu_x}{\sigma_x} \tag{3.3}$$

der x er den originale observasjonen, μ_x er gjennomsnittsverdien til egenskapen og σ_x er standardavviket til egenskapen [7]. Standardiseringen ble tilpasset på treningsdataene, og deretter ble disse parameterne brukt for å standardisere nye datapunkter/valideringsdata.

Et eksempel på hvorfor standardisering er viktig, er ved bruk av klassifikasjonsalgoritmen KNN, der en egenskap har verdier innenfor skalaen 1-10, og en annen har verdier innenfor skalaen 1-1000. Da har den andre egenskapen mye større betydning for avstanden mellom observasjonene enn den første egenskapen.

I denne oppgaven ble Scikit-Learn sin funksjon *StandardScaler* brukt.

3.6 Vurdering av resultater

Et par utvalgte metoder ble brukt for å vurdere ytelsen til modeller eller egenskaper. Korrelasjon ble brukt til univariat analyse av de enkelte egenskapene, men også til å velge optimalt filter. ROC AUC ble brukt til å måle den endelige ytelsen til modellene. Begge disse metodene, og hvordan de ble brukt, blir beskrevet i videre avsnitt.

3.6.1 Pearsons korrelasjonskoeffisient

Anta at det er n par med observasjoner fra to vektorer a og b. Pearsons korrelasjonskoeffisient er definert som:

$$r_p = \frac{S_{ab}}{\sqrt{S_{aa}S_{bb}}} = \frac{n\sum_{i=1}^{n} a_i b_i - (\sum_{i=1}^{n} a_i)(\sum_{i=1}^{n} b_i)}{\sqrt{[n\sum_{i=1}^{n} a_i^2 - (\sum_{i=1}^{n} a_i)^2][n\sum_{i=1}^{n} b_i^2 - (\sum_{i=1}^{n} b_i)^2]}} \tag{3.4}$$

der S_{ab} er kovariansen mellom a og b, S_{aa} er variansen til a, S_{bb} er variansen til b [51]. r_p blir tall mellom -1 og 1, der -1 gir en perfekt negativ korrelasjon, 1 gir en perfekt positiv korrelasjon og 0 gir ingen korrelasjon.

3.6.2 Spearmans rangkorrelasjonskoeffisient

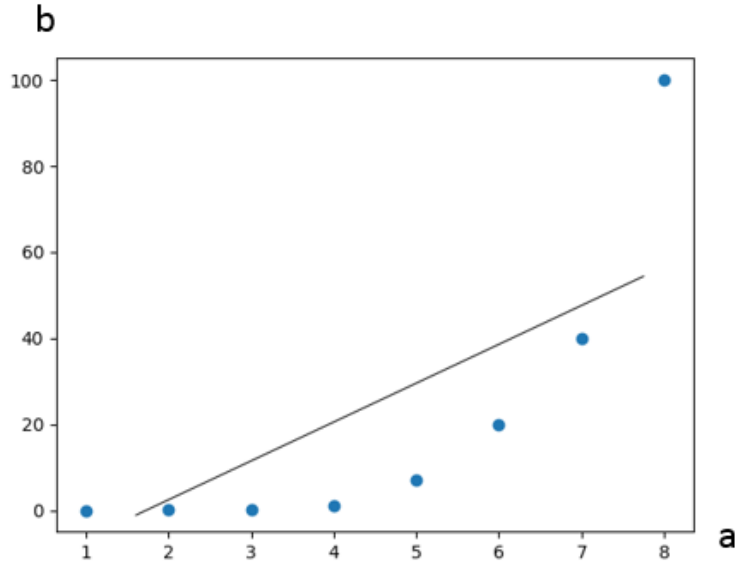
Spearmans rangkorrelasjonskoeffisient (ρ) er definert som Pearsons korrelasjonskoeffisient mellom rangerte variabler [51]. Anta at det er n par med observasjoner fra to variabler a og b. For hver av variablene rangeres observasjonene fra de to variablene fra lavest til høyest, hver for seg. Like verdier får rangen som er gjennomsnittet av posisjonene til de like verdiene. La u_i være rangen til observasjon i i a, og la v_i være rangen til observasjon i i b. Spearmans korrelasjonskoeffisient blir regnet ut ved å bytte ut de opprinnelige variablene i ligningen til Pearson korrelasjonskoeffisients med de rangerte variablene:

$$r_s = \frac{S_{uv}}{\sqrt{S_{uu}S_{vv}}} = \frac{n\sum_{i=1}^{n} u_i v_i - (\sum_{i=1}^{n} u_i)(\sum_{i=1}^{n} v_i)}{\sqrt{[n\sum_{i=1}^{n} u_i^2 - (\sum_{i=1}^{n} u_i)^2][n\sum_{i=1}^{n} v_i^2 - (\sum_{i=1}^{n} v_i)^2]}} \tag{3.5}$$

der u er den nye, rangerte vektoren til a, v er den nye, rangerte vektoren til b, S_{uv} er kovariansen mellom u og v, S_{uu} er standardavviket til u, S_{vv} er standardavviket til v, n er antall par med observasjoner, u_i er rangen til observasjon i i a og v_i er rangen til observasjon i i b. r_s blir er tall mellom -1 og 1, der -1 gir en perfekt negativ korrelasjon, 1 gir en perfekt positiv korrelasjon og 0 gir ingen korrelasjon.

Det er vanlig å bruke Spearmans rangkorrelasjonskoeffisient når det ikke nødvendigvis er en bivariat normalfordeling (likning (2.14) med $p = 2$) [52]. Spearmans rangkorrelasjonskoeffisient danner istedet monotoniske forhold, altså funksjoner uten ekstremalpunkter, men med en stadig økning eller reduksjon. Dette gjør at Spearmans

rangkorrelasjonskoeffisient kan fange opp mer ikke-lineære korrelasjoner enn Pearson korrelasjonskoeffisient kan [52]. Som et eksempel er Pearson korrelasjon og Spearman rangkorrelasjon begge regnet ut fra tilfellet i figur 3.11. Dette er et ikke-lineært forhold, så Pearson korrelasjonskoeffisient ble på 0.81, mens Spearmans rangkorrelasjonskoeffisient viste en perfekt sammenheng på 1.



Figur 3.11: *Et scatter-plott over tilfeldige datapunkter fra to variabler a og b. Den sorte linjen viser den lineære Pearson-korrelasjonen for variablene med* $p_r = 0.81$*. Spearman-korrelasjonen var derimot på* $p_s = 1.00$*.*

3.6.3 ROC-AUC

Feilmatrise

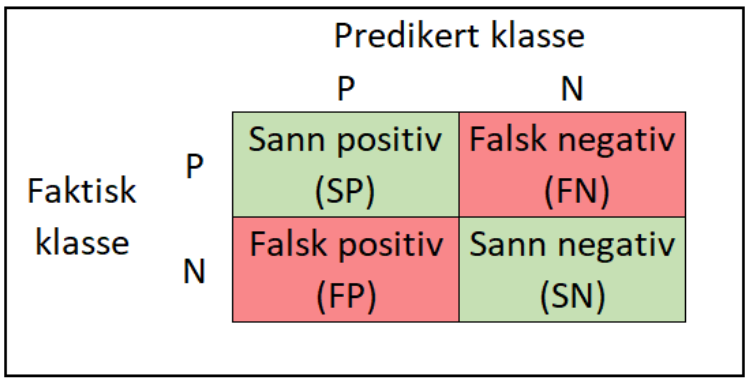
Feilmatrise er en måte å måle prediksjonsytelsen til en modell [7]. Dette er en 2x2 matrise som teller antallet klassifiseringer som var Falsk positive (FP), Falsk negative (FN), Sann negative (SN) og Sann positive (SP), og er vist i figur 3.12.

Fra feilmatrisen kan blant annet nøyaktighet (ACC), sann positiv grad (SPG) og falsk positiv grad (FPG) beregnes:

$$ACC = \frac{SP + SN}{FP + FN + SP + SN} \tag{3.6}$$

$$SPG = \frac{SP}{FN + SP} \tag{3.7}$$

$$FPG = \frac{FP}{FP + SN} \tag{3.8}$$

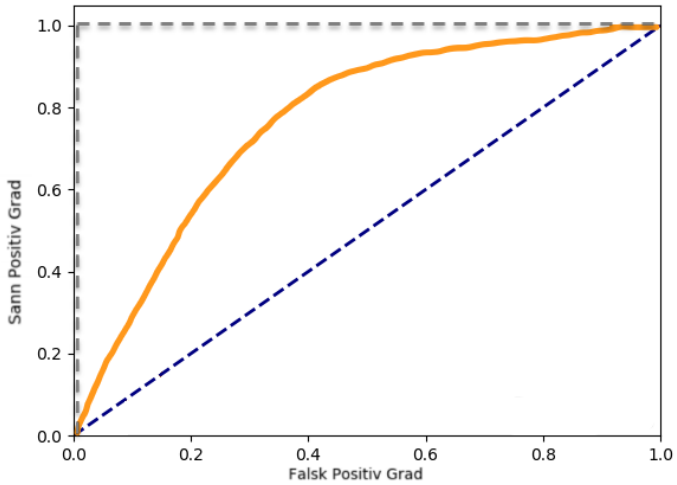


Figur 3.12: *Feilmatrise*

Dersom SPG = 1 er alle faktiske positive observasjoner klassifisert riktig, og dersom FPG = 1 er alle faktiske negative observasjoner klassifisert feil. SPG kalles også for sensitivitet, og spesifisitet er det samme som $1 - FPG$ [53].

Receiver Operating Characteristic

SPG og FPG kan brukes for å se på en models ytelse gjennom Receiver Operating Characteristic (ROC) [7]. En ROC-kurve er et plott av en kurve som viser SPG mot FPG hvor terskelen mellom disse er variert. Figur 3.13 er et eksempel på en ROC-kurve. Basert på ROC-kurven kan man regne ut arealet under kurven, den



Figur 3.13: *Eksempel på en ROC-kurve. Den blå stiplede diagonalen tolkes som en klassifisering like dårlig som tilfeldig gjetning. Den grå, stiplede linjen viser en perfekt model, der SPG er 1 og FPG er 0 i øverste, venstre hjørne. Hvis en ROC-kurve er under den blå diagonalen sier man at modellen fungerer dårligere enn ren gjetning. Den gule ROC-kurven er et eksempel på en typisk ROC-kurve, som er bedre enn kun gjetning, men ikke helt perfekt.*

såkalte ROC AUC (ROC Area Under the Curve). Denne er ofte bare kalt AUC.

Hvis AUC = 1 betyr det at modellen gir en perfekt prediksjon, mens AUC = 0.5 betyr at klassifiseringen ikke er bedre enn tilfeldig gjetning. Hvis AUC < 0.5 er modellen verre enn tilfeldig gjetning. ROC AUC er en metode som er egnet til bruk på ubalanserte datasett, der nøyaktigheten (Accuracy) ikke alltid gir en like god indikasjon [7]. Hvis en klasse tilsvarer 80% av datasettet, vil vi få en nøyaktighet på 80% kun ved å sette all responsen til denne klassen. Ved bruk av ROC AUC slipper man denne effekten, ved at den ser på prosentvis antall Sann positive mot Falsk positive.

3.6.4 Kryssvalidering

Et av de viktige stegene innenfor maskinlæring er å vurdere hvor bra en modell fungerer på data som modellen ikke har sett før [7]. En vanlig måte å gjøre dette på er å dele opp et datasett i et treningssett og et valideringssett. Da skal modellene trenes på treningssettet, og deretter kan modellens ytelse predikeres ved å bruke valideringssettet.

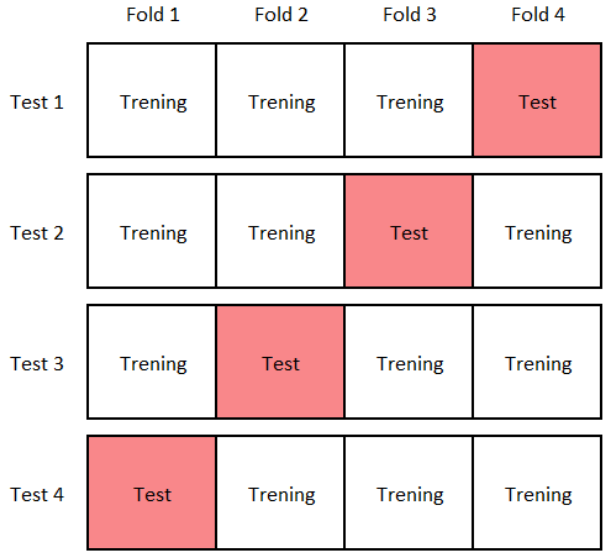
Hvis et datasett ikke har nok data til å deles opp, er kryssvalidering en av de vanligste metodene for å estimere ytelsen til en modell [22]. Dette innebærer å bruke deler av datasettet til trene modellen, og en annen del for å teste modellen. *K-folds-kryssvalidering* går ut på å dele datasettet inn i K deler, og stadig trene modellen på all foldene utenom fold K, og deretter teste modellen på fold K. Hvis $K = 4$ blir situasjonen som i figur 3.14.

Det er fort gjort å gjøre kryssvalidering på feil måte, og ende opp med en for optimistisk prediksjon på ytelsen. I denne oppgaven er kryssvalidering brukt til både å velge ut beste egenskaper, velge beste parametere for modellene, trene modellene og teste ytelsen til modellene. Derfor er kryssvalideringen gjort på følgende måte, for å ikke få en for optimistisk prediksjonsytelse:

1. Del datasettet tilfeldig inn i K folder.
2. For hver fold $k = 1, 2, ..., K$
   (a) Bruk en egenskapsutvelger for å velge ut de N beste egenskapene, ved å bruke alle foldene bortsett fra fold k.
   (b) Del dataene i alle foldene unntatt fold k i nye P antall folder. Bruk kryssvalidering på disse foldene for å finne beste parametere for modellen.
   (c) Tren en modell på all dataene i alle foldene unntatt fold k ved å bruke parameterene funnet i forrige punkt.
   (d) Bruk modellen for å predikere klassen til dataene i fold k, som har blitt holdt utenfor både egenskapsutvelgelse, valg av parametere og trening. Sammenlign prediksjonene med de faktiske klassene, for å predikere ytelsen til modellen.

3. Etter å ha utført dette for alle K foldene er gjennomsnittet av alle testene tatt, som da er estimatet på ytelsen til modellen. Bruk av en indre løkke i en ytre løkke på denne måten kalles for en nestet løkke, eller en Grid Search [22].

Her er K er antall folder, N er antall egenskaper og P er antall folder i kryssvalideringen for å finne beste parametere i den indre løkken i punkt (b).



Figur 3.14: *Illustrasjon av en 4-folds-kryssvalidering. I test 1 trenes modellen på fold 1, 2 og 3, og modellen testes på fold 4. I test 2 trenes modellen på fold 1, 2 og 4, og modellen testes på fold 3. Etter at dette er gjort slik at alle foldene har fått vært valideringssett én gang, tas gjennomsnittet av alle testene.*

De eneste utvelgelsene som ikke tok del i hver av disse kryssvaliderings-løkkene var valg av oppløsning, optimalt filter og antall egenskaper. Disse ble valgt ut på forhånd med egne kryssvalideringer, slik som beskrevet i avsnitt 3.7 og 3.10. Deretter ble resultatet av disse brukt videre i den siste kryssvalideringen beskrevet her, for å vurdere ytelsen til de siste modellene.

I denne oppgaven er 4-folds-kryssvalidering brukt i både den ytre og den indre løkken ($K = 4$ og $P = 4$). Dette ble gjort for alle fjorten klassifiseringsalgoritmene i kombinasjon med alle syv metoder for egenskapsutvelgelse av egenskaper i datasettet.

3.7 Preprosessering av bildene

3.7.1 Kvantifisering av intensitetsverdier

For å finne optimal kvantifisering av bildene til bruk i teksturanalyse, ble seks forskjellige antall gråtonenivåer testet ut, $N_g \epsilon \{8, 16, 32, 64, 128, 256\}$. Testingen for å

finne optimal kvantifisering ble bare utført på de opprinnelige bildene, uten filtertransformering. To forskjellige fremgangsmåter ble benyttet; en basert på Logistisk regresjon og Spearmans rangkorrelasjonskoeffisient, og en der flere forskjellige klassifiseringsalgoritmer og ROC AUC ble brukt.

Fremgangsmåte 1: Logistisk regresjon og Spearmans rangkorrelasjonskoeffisient

Logistisk regresjon ble brukt som egenskapsutvelger for å lage nye delmengder X_{N_f} med forskjellige antall egenskaper. Antall egenskaper fra 1 til 10 ble testet ut, $N_f \, \epsilon \, \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. For hver nye delmengde X_{N_f} ble logistisk regresjon brukt som klassifieringsalgoritme for å beregne sannsynlighetsestimatene på utfallet $\hat{y}$. Deretter ble Spearmans rangkorrelasjonskoeffisient beregnet mellom den reelle utfallsvektoren y og de forskjellige sannsynlighetsestimat-vektorene $\hat{y}$.

De tre beste korrelasjonskoeffisientene ble plukket ut; altså de tre delmengdene X_{N_f} som gav prediksjoner med høyest korrelasjon med utfallvektyoren y. Gjennomsnittet av disse tre ble beregnet. Hele prosessen ble gjentatt tre ganger, med 3-folds kryssvalidering, der gjennomsnittet av disse tre kryssvalideringen ble den endelige korrelasjonskoeffisienten. Kvantifiseringen som gav høyest korrelasjonskoeffisient ble vurder som den optimale kvantifiseringen.

Fremgangsmåte 2: Klassifiseringsalgoritmer og ROC AUC

To forskjellige egenskapsutvelgere ble brukt og kombinert med tre forskjellige klassifiseringsalgoritmer, for å predikere behandlingsutfallet $\hat{y}$ og sammenligne dette med det reelle behandlingsutfallet y. Til egenskapsutvelgelse ble ReliefF og Logistisk regresjon brukt, og det ble plukket ut 10 egenskaper. Som klassifiseringsalgoritmer ble Random Forest, Logistisk regresjon og K-Nearest Neighbour brukt. Testingen ble gjort gjennom en 4-folds kryssvalidering. Etter hver kryssvalidering ble AUC for prediksjonen $\hat{y}$ regnet ut, og gjennomsnittet av alle 4 foldene ble brukt som endelig AUC for den kvantifiseringen. Kvantifiseringen som gav høyest AUC ble vurdert som optimal kvantifisering.

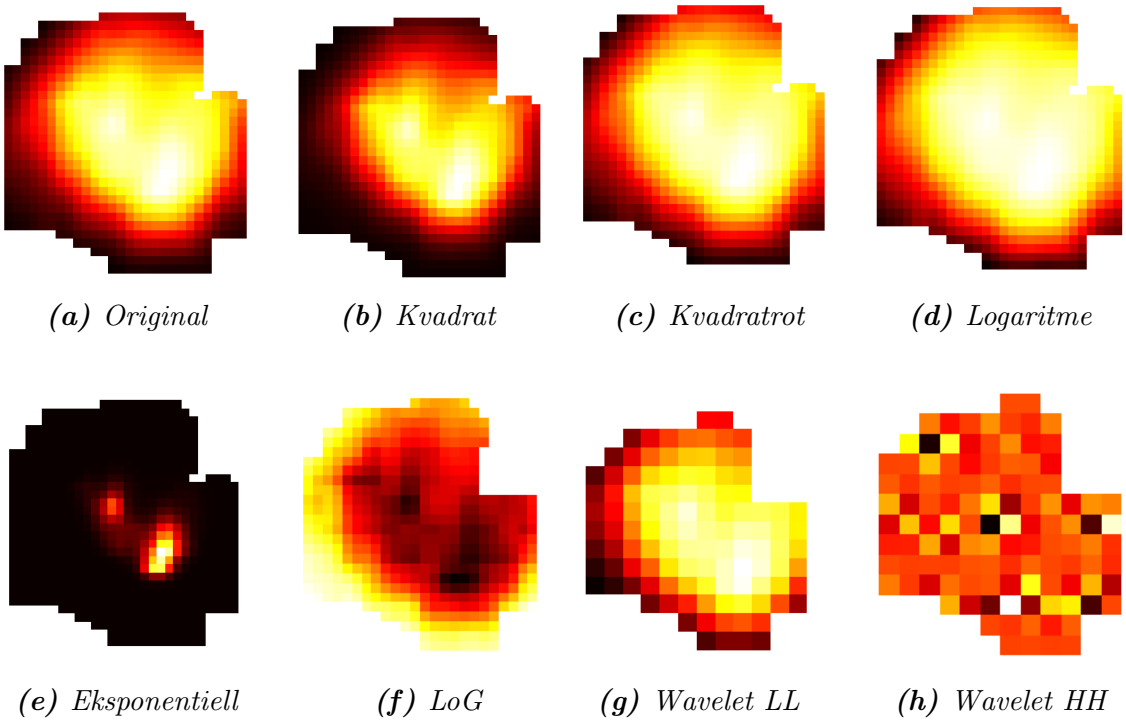
3.7.2 Filtertransformering av bildene

Informasjon om de utvalgte filtrene

Før egenskaper ble trukket ut av PET- og CT-bildene, ble bildene transformert ved å legge på ulike filtre. Å legge på et filter er en bildeteknikk som endrer intensitetsverdiene i voxlene/pixlene i bildene for å fremheve spesifikke egenskaper. Det finnes filtre som bare ser på hver enkelt voxel, f.eks. tar kvadratroten av verdien i voxelen.

Det finnes også filtre som ser på voxlene rundt voxelen vi skal endre verdien på. Eksempler på dette er kantdeteksjonsfiltre, som fremhever voxler der det er store endringer fra en voxel til en annen, og fremhever da kanter og hjørner. Tabell 3.5 viser de ulike filtrene som ble benyttet i denne oppgaven.

Kvadrat-, kvadratrot-, logaritme- og eksponentiell-filteret er alle filter som kun ser på verdien i voxelen i fokus, og endrer den. Disse filtrene endrer intensitetsforholdene i bildet. Figur 3.15 viser et eksempel på et PET-bilde (todimensjonalt) av en tumor, der de forskjellige filtrene er brukt. Figuren viser at kvadrat-filteret og eksponensiell-filteret gjør at andelen mørke verdier i bildet øker, mens kvadratrot-filteret og logaritme-filteret gjør at andelen lyse verdier i bildet øker.



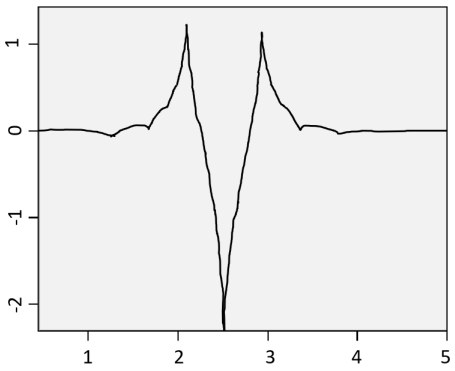
(a) *Original* ***(b)*** *Kvadrat* ***(c)*** *Kvadratrot* ***(d)*** *Logaritme*

(e) *Eksponentiell* ***(f)*** *LoG* ***(g)*** *Wavelet LL* ***(h)*** *Wavelet HH*

Figur 3.15: *Et todimensjonalt utsnitt fra et PET-bilde av tumor til pasient nr. 2. (a) viser det originale bildet, mens de andre bildene viser bildet transformert ved hjelp av forskjellige filtre. Bildene er vist i en varme-fargepalett istedet for gråskala, for å fremheve forskjellene.*

Tabell 3.5: *En liste over filtrene brukt i denne oppgaven.*

Original
Det opprinnelige bildet uten noe filter.
LoG
LoG (Laplacian of Gaussian) er et kantdeteksjonsfilter, som trekker frem områder med raske endringer i intensitetsverdier, også kalt områder med diskontinuitet [54]. Her settes en verdi for sigma σ, der en liten σ trekker frem fine teksturer (endringer over små områder) og en stor σ trekker frem grove teksturer (endringer over større områder). I denne oppgaven ble $\sigma = \{0.5, 1, 2\}$ testet ut.
Kvadrat
Tar kvadratet av intensitetsverdiene og skalerer dem tilbake til det opprinnelige området for verdiene.
Kvadratrot
Tar kvadratroten av absoluttverdien til intensitetsverdiene og skalerer dem tilbake til det opprinnelige området for verdiene.
Logaritme
Tar logaritmen av absoluttverdien til intensitetsverdiene pluss 1 og skalerer dem tilbake til det opprinnelige området for verdiene.
Eksponentiell
Tar eksponensialverdien til intensitetsverdiene, altså er den nye verdien e opphøyd i den gamle verdien. Verdiene er deretter skalert tilbake til det opprinnelige området for verdiene.
Wavelet
Et waveletfilter, som dekomponerer bildet via et høypassfilter H eller et lavpassfilter L ved bruk av en wavelet-funksjon [54]. Det ble tilsammen 8 dekomposisjoner per bilde, som er alle kombinasjoner av høypassfilter og lavpassfilter i alle 3 retninger/dimensjoner. Et lavpassfilter er et filter som bare slipper igjennom de sakte endringene i et bilde, mens et høypassfilter slipper igjennom de raske endringene i bildet. Coiflet1 ble brukt som bølgefunksjon, og er illustrert i figur 3.16.



Figur 3.16: *Illustrasjon av en Coiflet1-bølge. Tegnet etter figur fra Wavelet Browser [55].*

Wavelet-filteret og LoG-filteret er filtre som i tillegg til intensitetsverdien i selve voxelen, også ser på intensitetsverdiene til voxlene rundt. LoG-filteret 3.15f har satt høye verdier der det er raske endringer i bildet, og gjør dette ved hjelp av gradienter. Wavelet LL 3.15g er lavpass-dekomponeringen av det originale bildet i begge retninger. Den blir ofte vurdert som en redusert versjon av det opprinnelige bildet, ettersom den beholder mesteparten av detaljene [56]. Wavelet HH 3.15h er høypass-dekomponeringen av det originale bildet i begge retninger. Denne inneholder kun høyfrekvent informasjon, og kan ofte være ganske støyete [56].

Valg av filter

PET- og CT-bilder, kvantifisert med valgt antall intensitetsverdier, ble transformert med filtrene gitt i tabell 3.5. Intensitets- og teksturegenskaper ble beregnet for de transformerte bildene. Disse egenskapene ble organisert i datamatriser X_{filter}, hvor X_{filter} er gitt som $X_{original}$, $X_{eksponentiell}$, $X_{kvadrat}$, $X_{kvadratrot}$, $X_{logaritme}$, X_{LoG}, $X_{WaveletLLL}$ og $X_{WaveletHHH}$. Innholdet i X_{filter}-matrisene er gitt i tabell 3.6. Alle kliniske faktorer og formegenskaper ble inkludert i disse matrisene, ettersom disse er uavhengige av filter.

Observasjonene i hver av datamatrisene X_{filter} ble klassifisert i henhold til utfall ved bruk av ReliefF som egenskapsutvelger og alle fjorten klassifiseringsalgoritmene. Fire egenskaper ble valgt ut av ReliefF. Modellene laget av hver av klassifiseringsalgoritmene ble validert gjennom en 4-folds-kryssvalidering, gjentatt to ganger. Gjennomsnittlig AUC for hver klassifiseringsalgoritme ble beregnet, og den høyest oppnådde gjennomsnittlige AUC AUC_{en} ble lagret. I tillegg ble gjennomsnittlig AUC for alle klassifiseringsalgoritmene AUC_{alle} beregnet.

Tabell 3.6: *Beskrivelse av hva matrisene, brukt til sammenligning av filter, inneholder. Førsteordens egenskaper og Teksturegeskaper viser til førsteordens egenskaper og teksturegenskaper fra bildene transformert med det respektive filteret. Som eksempel inneholder $X_{eksponentiell}$ førsteordens egenskaper og teksturegeskaper fra PET- og CT-bildene transformert med eksponentiell-filteret.*

X_{filter}	$X_{filter-ny}$
Førsteordens egenskaper fra PET- og CT-bildene transformert med *filter*	Førsteordens egenskaper fra PET-bildene transformert med *filter*
Teksturegenskaper fra PET- og CT-bildene transformert med *filter*	Teksturegenskaper fra PET-bildene transformert med *filter*
Formegenskaper	
Kliniske faktorer	

Utforskning av filtertransformasjoner ekskludert kliniske faktorer og formegenskaper

Etter at matrisene X_{filter} ble undersøkt hver for seg, ble det foretatt en ny undersøkelse, der kun egenskapene som var avhengige av filtertransformering var inkludert i matrisene. Dette ble kun gjort på egenskapene trukket ut fra PET-bildene, og ble gjort for å undersøke effekten filtertransformeringen hadde på prediksjonsytelsen til egenskapene.

Intensitets- og teksturegenskaper, beregnet for de transformerte bildene, ble igjen organisert i datamatriser $X_{filter-ny}$, hvor $X_{filter-ny}$ er gitt som $X_{original-ny}$, $X_{eksponentiell-ny}$, $X_{kvadrat-ny}$, $X_{kvadratrot-ny}$, $X_{logaritme-ny}$, X_{LoG-ny}, $X_{WaveletLLL-ny}$ og $X_{WaveletHHH-ny}$. Innholdet i $X_{filter-ny}$-matrisene er gitt i tabell 3.6. Alle kliniske faktorer og formegenskaper ble ekskludert i disse matrisene.

Observasjonene i hver av datamatrisene $X_{filter-ny}$ ble klassifisert i henhold til utfall ved å bruke ReliefF som egenskapsutvelger og L1-Logistisk regresjon som klassifikasjonsalgoritme. Fem egenskaper ble valgt ut av ReliefF. Modellene ble validert gjennom en 4-folds-kryssvalidering, gjentatt ti ganger. Gjennomsnittlig AUC over alle kryssvalideringene ble beregnet.

3.8 Sammenligning av PET- og CT-bildeegenskaper

3.8.1 Sammenligning av PET- og CT-bildeegenskaper inkludert kliniske faktorer og formegenskaper

Egenskapene trukket ut fra PET- og CT-bildene gjennom valgt filtertransformeringen ble lagt i hver sine matriser X_{PET} og X_{CT}. Innholdet i disse matrisene er gitt i tabell 3.7. Alle kliniske faktorer og formegenskaper var inkludert i disse matrisene, ettersom disse er uavhengige av bildetype.

Tabell 3.7: *Beskrivelse av hva matrisene, brukt til sammenligning av egenskaper fra PET- og CT-bildene, inneholder. Førsteordens egenskaper og Teksturegeskaper viser til førsteordens egenskaper og teksturegenskaper fra bildene transformert med utvalgt filter.*

X_{PET}	X_{CT}	X_{PET-ny}	X_{CT-ny}
Førsteordens egenskaper fra PET-bildene transformert med optimalt filter.	Førsteordens egenskaper fra CT-bildene transformert med optimalt filter.	Førsteordens egenskaper fra fra PET-bildene transformert med optimalt filter.	Førsteordens egenskaper fra CT-bildene transformert med optimalt filter.
Teksturegenskaper fra PET-bildene transformert med optimalt filter.	Teksturegenskaper fra CT-bildene transformert med optimalt filter	Teksturegenskaper fra PET-bildene transformert med optimalt filter	Teksturegenskaper fra CT-bildene transformert med optimalt filter
Formegenskaper	Formegenskaper		
Kliniske faktorer	Kliniske faktorer		

Observasjonene i hver av datamatrisene X_{PET} og X_{CT} ble klassifisert i henhold til behandlingsutfall ved å bruke ReliefF som egenskapsutvelger og L1-Logistisk regresjon som klassifikasjonsalgoritme. Det ble valgt at ReliefF skulle velge ut kun to egenskaper, ettersom det var to egenskaper som gav høyest gjennomsnittlig ytelse for ReliefF (figur 4.4a). Modellene ble validert gjennom en 4-folds-kryssvalidering, gjentatt ti ganger. Gjennomsnittlig AUC over alle kryssvalideringene ble beregnet.

3.8.2 Sammenligning av PET- og CT-bildeegenskaper ekskludert kliniske faktorer og formegenskaper

Etter at matrisene X_{PET} og X_{CT} ble undersøkt hver for seg, ble det foretatt en ny undersøkelse, der kun egenskapene som var avhengige av bildetype var inkludert i matrisene.

Intensitets- og teksturegenskaper, beregnet for bildene med den utvalge bildetransformeringen, ble igjen organisert i datamatriser X_{PET-ny} og X_{CT-ny}. Innholdet i disse datamatrisene er gitt i tabell 3.7. Alle kliniske faktorer og formegenskaper ble ekskludert i disse matrisene. Observasjonene i hver av datamatrisene X_{PET-ny} og X_{CT-ny} ble klassifisert ved bruk av samme metode som beskrevet i avsnitt 3.8.1.

3.9 Sammenligning av egenskapsklasser

Egenskapene trukket ut fra PET- og CT-bildene, gjennom denne valgte filtertransformeringen, ble lagt i datamatriser $X_{egenskapsklasse}$ basert på egenskapsklassene forklart i avsnitt 3.4. $X_{egenskapsklasse}$ er gitt som $X_{førsteordens}$, X_{form}, X_{GLCM}, X_{GLSZM}, X_{GLRLM}, X_{NGTDM} ,X_{GLDM}, X_{alle}. Her inneholdt $X_{egenskapsklasse}$ både PET- og CT-egenskapene beregnet for den egenskapsklassen. X_{alle} inneholder egenskapene fra alle egenskapsklassene. Observasjonene i hver av datamatrisene $X_{egenskapsklasse}$ ble klassifisert ved bruk av samme metode som beskrevet i avsnitt 3.7.2.

3.10 Valg av antall egenskaper i modellene

Ved bruk av for mange egenskaper i en modell, er det større sannsynlighet for å overtilpasse treningsdataene (figur 2.11). Derfor er det viktig å redusere antall dimensjoner eller egenskaper, og bruke en egenskapsutvelger for å finne de mest relevante egenskapene i datasettet.

Følgende fremgangsmåte ble brukt for hver enkelt egenskapsutvelger, for å finne optimalt antall egenskaper for hver egenskapsutvelger:

Den utvalgte egenskapsutvelgeren ble brukt til å lage en ny delmengde X_n, med kun noen av egenskapene i X. Antall egenskaper fra 1 til 20, $n \epsilon \{1 : 20\}$ ble testet. Følgende ble utført for hver nye delmengde X_n med n antall egenskaper:

De ti raskeste klassifiseringsalgoritmene (LOG-1, LOG-2, Decision Tree, Gradient Boosting, GNB, LDA, QDA, PLSR, SVC og linear SVC) ble brukt på delmengden av egenskaper X_n for å predikere behandlingsutfallet $\hat{y}$. Disse ble testet med en 4-folds kryssvalidering. Dette ble gjentatt to ganger, og gjennomsnittlig AUC av alle foldene ble beregnet for hver klassifiseringsalgoritme. Deretter ble gjennomsnittlig AUC

beregnet over alle klassifiseringsalgoritmene, for å beskrive hvordan disse presterer med n antall egenskaper.

3.11 Balansering av datasett

Ubalanserte datasett er et vanlig problem når man jobber med reelle data, og betyr at observasjoner fra en klasse er overrepresentert i datasettet [7]. I det endelige datasettet var 26% av pasientene i klassen *lokalregionalt tilbakefall*, mens 74% av pasientene var i klassen *progresjonsfri overlevelse* (tabell 3.2). En måte å håndtere ubalanserte datasett er å tilegne en større straff til feil klassifisering av den minste klassen. Ved bruk av klassifiseringsalgoritmer fra Scikit-learn kan dette bli gjort ved bruk av kommandoen $class\_weight = balanced$. Dette ble gjort for de klassifiseringsalgoritmene det var mulig å gjøre det for, som var Logistisk regresjon, RF, AdaBoost, Beslutningstre, SVC og lineær SVC.

Kapittel 4

Resultater

4.1 Innledende analyser

For å finne den optimale modellen for å predikere behandlingutfallet, ble en del innledende analyser gjort. Dette inkluderte å finne den optimale kvantifiseringen av intensitetsverdiene til bildene, finne hvilken transformering (filter) av bildene som gav velfungerende egenskaper og finne antall egenskaper som skal være med i modellene. For mesteparten av de innledende analysene, er det alternativet som har gitt høyest AUC som har blitt valgt.

4.1.1 Kvantifisering av intensitetsverdier

Presentasjonen av resultatene starter med å optimalisere antall intensitetsverdier i bildene. Tabell 4.1 viser gjennomsnittlig AUC for en 4-folds-kryssvalidering med de forskjellige antall intensitetsverdiene for PET-bildene. Her var det flest av klassifiseringsalgoritmene (tre stykker) som gav høyest AUC med 16 bins og gjennomsnittlig AUC over alle klassifiseringsalgoritmene var også høyest ved 16 bins. Figur 4.1 viser et utsnitt av et PET-bilde av en tumor med (a) 256 bins og (b) 16 bins. Det er tydelig at bildet med 256 bins viser et mer kontinuerlig bilde, der det er ganske få tilfeller med helt like intensitetsverdier ved siden av hverandre, mens bildet med 16 bins har blitt delt inn i store områder med like verdier.

Tabell 4.2 viser gjennomsnittlig AUC for en 4-folds-kryssvalidering med de forskjellige antall intensitetsverdiene for CT-bildene. Her var det flest av klassifiseringsalgoritmene (tre stykker) som gav høyest AUC med 128 bins og gjennomsnittlig AUC over alle klassifiseringsalgoritmene var også høyest ved 128 bins. Figur 4.2 viser et utsnitt av et CT-bilde av en tumor med 256 bins og 128 bins. Bildene viser at 7-bits bildet (b) har områder med like verdier, mens 8-bits bildet (a) har en mer kontinuerlig fordeling.

Tabell 4.1: *Gjennomsnittlig AUC for klassifisering av PET-bilder med forskjellige antall bins etter 4-folds-kryssvalidering. Random Forests, L1-Logistisk regresjon og KNN ble brukt som klassifiseringsmetoder, og ReliefF og L1-Logistisk regresjon ble brukt til egenskapsutvelgelse. Den høyeste AUC for hver kolonne er markert i grønn.*

Feature selection	**ReliefF**			**LOG**			**Snitt**
Klassifisering	**RF**	**LOG**	**KNN**	**RF**	**LOG**	**KNN**	
8 bins	0.64	0.63	0.64	0.63	0.63	0.64	0.64
16 bins	0.66	0.59	0.65	0.66	0.70	0.63	0.65
32 bins	0.64	0.54	0.57	0.63	0.68	0.59	0.61
64 bins	0.65	053	0.57	0.59	0.67	0.57	0.60
128 bins	0.64	0.57	0.58	0.59	0.65	0.58	0.60
256 bins	0.65	0.54	0.61	0.62	0.72	0.63	0.63

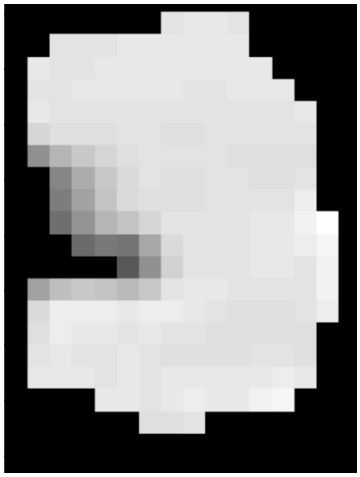
(a) *256 bins* ***(b)*** *16 bins*

Figur 4.1: *Et tverrsnitt av et PET-bilde av tumor for pasient nr. 4. (a) viser et 8-bits bilde (256 bins), mens (b) viser et 4-bits bilde (16 bins), som i gjennomsnitt gav høyest ytelse.*

For å kontrollere resultatet i tabell 4.1 og 4.2, ble det brukt en annen metode for å sjekke optimal kvantifisering av intensitetsverdiene. Tabell 4.3 viser Spearmans rangkorrelasjonskoeffisient ρ mellom utfallsvektoren y og sannsynlighetsestimatene $\hat{y}$ som beskrevet i avsnitt 3.7.1. Her gav også 16 bins kvantifisering høyest ρ for PET-bildene og 128 bins kvantifisering for CT-bildene. Dermed ble 16 bins (PET) og 128 bins (PET) valgt til videre analyse.

Tabell 4.2: *Gjennomsnittlig AUC for klassifisering av CT-bilder med forkjellige antall bins etter 4-folds-kryssvalidering. Random Forests, L1-Logistisk regresjon og KNN ble brukt som klassifiseringsmetoder, og ReliefF og L1-Logistisk regresjon ble brukt til egenskapsutvelgelse. Den høyeste AUC for hver kolonne er markert i grønn.*

Feature selection	**Relief**			**LOG**			**Snitt**
Klassifisering	**RF**	**LOG**	**KNN**	**RF**	**LOG**	**KNN**	
8 bins	0.63	0.47	0.58	0.56	0.56	0.57	0.56
16 bins	0.58	0.50	0.58	0.57	0.54	0.58	0.56
32 bins	0.55	0.50	0.56	0.54	0.65	0.55	0.56
64 bins	0.56	0.50	0.56	0.57	0.59	0.60	0.56
128 bins	0.59	0.50	0.59	0.60	0.56	0.65	0.58
256 bins	0.57	0.50	0.51	0.61	0.61	0.56	0.56



(a) *256 bins*

(b) *128 bins*

Figur 4.2: *Et tverrsnitt av et CT-bilde av tumor for pasient nr. 4. (a) viser et 8-bits bilde (256 bins), mens (b) viser et 7-bits bilde (128 bins), som i gjennomsnitt gav høyest ytelse.*

Tabell 4.3: *Spearmans rangkorrelasjonskoeffisient mellom utfallsvektoren og sannsynlighetsestimatene for forskjellige antall bins. Den høyeste AUC for hver kolonne er markert i grønn.*

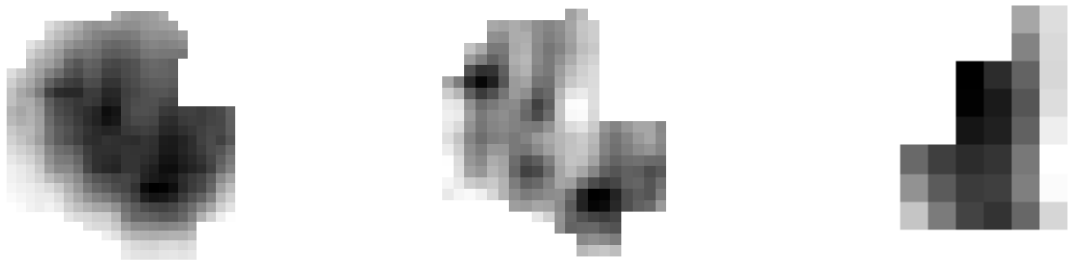
Antall bins	**PET**	**CT**
8	0.25	0.17
16	0.38	0.07
32	0.33	0.25
64	0.16	0.19
128	0.19	0.29
256	0.22	0.28

4.1.2 Valg av filter for bildetransformering

Først ble det undersøkt hvilken innstillingsparameter σ for LoG-filteret som gav høyest AUC for LoG-filterete PET-bilder. ROC for klassifisering ved bruk av egenskaper fra bildene transformert med forskellige σ-verdier er gitt i tabell 4.4. Det var små forskjeller i ytelsen for de tre verdiene av σ. Figur 4.3 viser et eksempel på todimensjonalt utsnitt av et PET-bilde av en tumor transformert med et LoG-filter med de tre σ-verdiene. I dette eksempelet trekker $\sigma = 2$ frem såpass grove teksturer at den ikke lenger ligner på det opprinnelige bildet. Derimot er tverrsnitt for $\sigma = 0.5$ relativt lik det opprinnelig bildet, men kantene er fremhevet. $\sigma = 0.5$ ble valgt til videre analyse.

Tabell 4.4: *Gjennomsnittlig AUC for testing av optimal σ for LoG-filteret. Den høyeste AUC for hver kolonne er markert i grønn. AUC_{en} gir gjennomsnittlig AUC for klassifikasjonsalgoritmen med høyest gjennomsnittlig AUC. AUC_{alle} gir gjennomsnittlig AUC over alle fjorten klassifikasjonsalgoritmene.*

Sigma	**AUC_{en}**	**AUC_{alle}**
0.5	0.58	0.56
1	0.56	0.54
2	0.58	0.55



(a) $\sigma = 0,5$ ***(b)*** $\sigma = 1$ ***(c)*** $\sigma = 2$

Figur 4.3: *Et todimensjonalt utsnitt fra et PET-bilde av tumor til pasient nr. 2. (a) viser bildet med $\sigma = 0.5$, (b) viser bildet med $\sigma = 1$ og (c) viser bildet med $\sigma = 2$.*

Tabell 4.5 og 4.6 viser AUC etter klassifiseringen ved bruk av egenskaper fra henholdsvis PET- og CT-bildene med forskjellig filtertransformeringer. Generelt gav kvadratrot-filterer og ufilterte bilder høyest AUC_{en} og AUC_{alle}, definert i avsnitt 3.7.2. For PET-bildene gav også bilder transformert med eksponentiell-filteret høy AUC. Tilsvarende gav bilder transformert med logaritme-filteret høy AUC for CT-bildene. Merk at kun wavelet-filtrene HHH (høypass i alle retninger) og LLL (lavpass i alle retninger) ble vurdert. Disse gav lav AUC for både PET- og CT-bildene. Merk også at LoG-filteret ikke ble brukt for CT-bildene, på grunn av en feiltakelse. Siden denne gav såpass lav ytelse for PET-bildene ble det ikke utført en ny test med denne for CT-bildene.

Tabell 4.5: *Gjennomsnittlig AUC for testing av filtre på PET-bildene. Alle betyr at egenskapene fra alle filtrene ble testet i samme X-matrise. AUC_{en} gir gjennomsnittlig AUC for klassifikasjonsalgoritmen med høyest gjennomsnittlig AUC. AUC_{alle} gir gjennomsnittlig AUC over alle fjorten klassifikasjonsalgoritmene. Den høyeste AUC for hver kolonne er markert i grønn.*

Filter	$\mathbf{AUC_{en}}$	$\mathbf{AUC_{alle}}$
Original	0.64	0.61
Eksponentiell	0.64	0.61
Logaritme	0.60	0.57
Kvadrat	0.60	0.60
Kvadratrot	0.65	0.62
LoG	0.58	0.56
Wavelet L	0.55	0.53
Wavelet H	0.58	0.52
Alle	0.57	0.54

Tabell 4.6: *Gjennomsnittlig AUC for testing av filtre på CT-bildene. Alle betyr at egenskapene fra alle filtrene ble testet i samme X-matrise. AUC_{en} gir gjennomsnittlig AUC for klassifikasjonsalgoritmen med høyest gjennomsnittlig AUC. AUC_{alle} gir gjennomsnittlig AUC over alle fjorten klassifikasjonsalgoritmene. Den høyeste AUC for hver kolonne er markert i grønn.*

Filter	$\mathbf{AUC_{en}}$	$\mathbf{AUC_{alle}}$
Original	0.65	0.62
Eksponentiell	0.60	0.53
Logaritme	0.64	0.62
Kvadrat	0.62	0.59
Kvadratrot	0.65	0.62
Wavelet L	0.56	0.53
Wavelet H	0.590	0.545
Alle	0.55	0.52

Dersom egenskapene for alle filtertypene ble slått sammen, ble klassifiseringen blandt de med lavest prediksjonsytelse. Det ble valgt å gå videre med svulsgtegenskapene trukket ut fra PET-bildene og CT-bildene gjennom kvadratrotfilteret.

4.1.3 Sammenligning av egenskapsklasser

Målet med sammenligningen av egenskapsklassene, var å få en oversikt over betydningen av egenskapene fra de forskjellige egenskapsklassene, og ble ikke brukt til egenskapsutvelgelse. Tabell 4.7 viser at egenskapene som beskriver tumors form predikerte behandlingsutfallet med høyere AUC enn de andre egenskapsklassene. Egenskapene førsteordens statistikk og teksturmetodene GLCM, GLRLM, NGTDM og GLDM gav tilnærmet lik ytelse, mens GLSZM gav noe lavere.

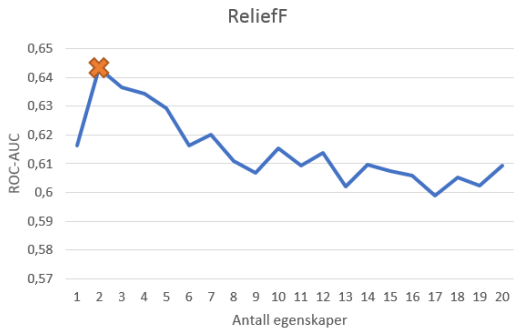
Tabell 4.7: *Gjennomsnittlig AUC for to 4-folds-kryssvalideringer. AUC_{en} viser gjennomsnittlig AUC for klassifiseringsalgoritmen som gav høyest AUC, og AUC_{alle} viser gjennomsnittlig AUC over alle fjorten klassifiseringsalgoritmene. Den høyeste AUC for hver kolonne er markert i grønn.*

Klasse	$\mathbf{AUC_{en}}$	$\mathbf{AUC_{alle}}$
Førsteordens	0.56	0.52
Form	0.65	0.60
GLCM	0.56	0.51
GLSZM	0.49	0.46
GLRLM	0.57	0.53
NGTDM	0.59	0.55
GLDM	0.58	0.54
Alle	0.64	0.61

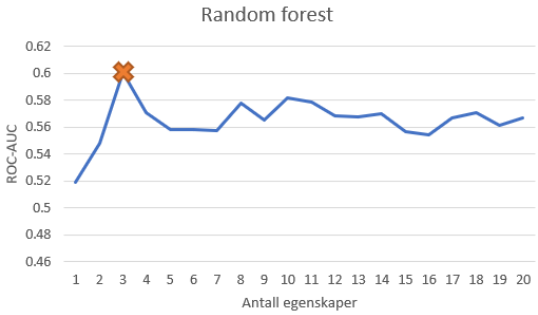
4.1.4 Antall egenskaper

Figur 4.4 viser gjennomsnittlig AUC for klassifikasjonsalgoritmene som funksjon av antall egenskaper, for forskjellige egenskapsutvelgere. Figuren viser at ReliefF og RF hadde høyest ytelse ved få antall egenskaper $n = \{2, 3\}$, mens Logistisk regresjon og MI hadde høyest ytelse ved et høyere antall egenskaper $n = \{19, 14\}$.

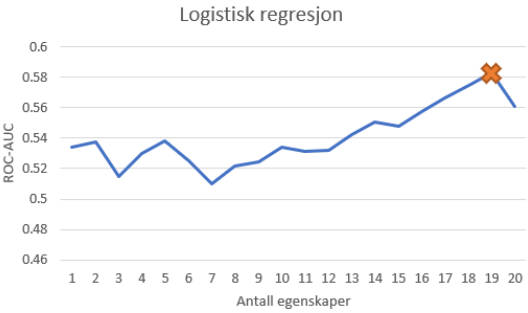
I tillegg til de fire egenskapsutvelgerne, ble tre dimensjonsreduksjonsmetoder undersøkt. Figur 4.5 viser gjennomsnittlig AUC for klassifikasjonsalgoritmene som funksjon av antall komponenter for to av dimensjonsreduksjonsmetodene. Høyest AUC ble funnet for 4 komponenter for PCA og 5 komponenter for ICA. Den siste dimensjonsreduksjonsmetoden, LDA, kan maksimalt ha antall komponenter gitt som antall klasser minus 1 [28]. I dette tilfellet er det kun to klasser, slik at kun én komponent er mulig.



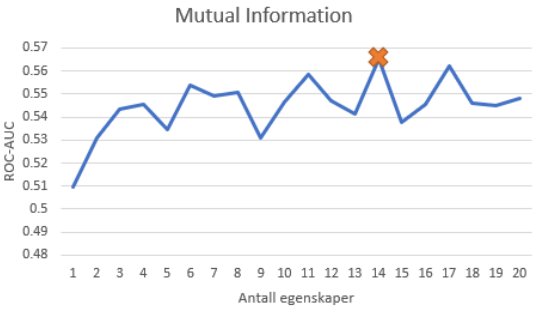
(a) ReliefF



(b) Random forest

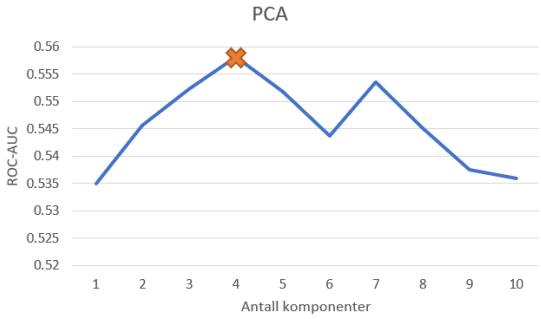


(c) Logistisk Regresjon

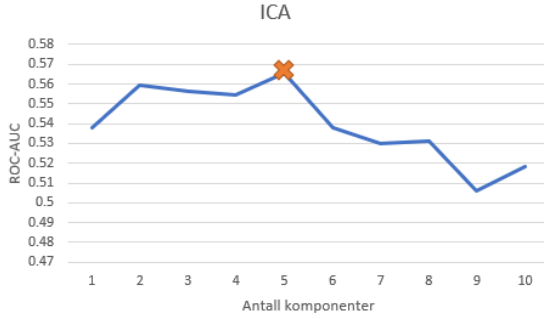


(d) Mutual Information

Figur 4.4: *Gjennomsnittlig AUC beregnet for klassifiseringer gitt av ti klassifiseringsalgoritmer som funksjon av antall egenskaper. Antall egenskaper som ble valgt er markert ned X, og er valgt slik at ytelsen er høyest ved færrest antall egenskaper.*



(a) PCA



(b) ICA

Figur 4.5: *Gjennomsnittlig AUC beregnet for klassifiseringer gitt av ti klassifiseringsalgoritmer som funksjon av antall komponenter. Antall egenskaper som ble valgt er markert med X, og er valgt slik at ytelsen er høyest ved færrest antall komponenter.*

4.2 Univariate resultater

Alle svulstegenskapene ble beregnet fra de originale PET- og CT-bildene kvantifisert med hhv. 16 bins og 128 bins. Egenskapene ble også beregnet fra de kvantifiserte bildene transformert med seks ulike filtre. Dette gav til slutt 2699 egenskaper, som er inkludert de kliniske faktorene i tabell 3.1.

For å få et inntrykk av sammenhengden mellom de forskjellige egenskapene og behandlingsutfallet, ble egenskapenes korrelasjon (Pearson) med behandlingsutfallet beregnet. Tabell 4.8 gir en oversikt over de syv egenskapene med høyest korrelasjon med progresjonsfri overlevelse, og tabell 4.9 gir en kort forklaring på betydningen av disse.

Tabell 4.8: *Egenskapene med høyest korrelasjon med hendelse progresjonsfri overlevelse. Det negative fortegnet betyr negativ korrelasjon med tilbakefall. Høy verdi av disse egenskapene er dermed knyttet til progresjonsfri overlevelse.*

Type	**Filter**	**Klasse**	**Egenskap**	**Korr**	**p-verdi**
CT	Wavelet LLH	GLCM	IMC2	-0.24	0.001
PET	Wavelet HLL	GLCM	Maximum-Probability	-0.23	0.002
PET	Wavelet HLL	GLSZM	GLNN	-0.22	0.002
PET	LoG-1	GLCM	Id	-0.22	0.002
CT	Wavelet LHH	GLRLM	LRLGLE	-0.22	0.003
PET	LoG-1	GLCM	Idm	-0.22	0.003
CT	Wavelet HLH	NGTDM	Strength	-0.22	0.003

Tabell 4.9: *Kort forklaring på betydningen av de syv egenskapene med høyest korrelasjon med hendelse progresjonsfri overlevelse. En mer utfyllende forklaring er gitt i vedlegg A.*

Egenskap	**Kort forklaring**
IMC2	Avhengighet verdi og posisjon
Maximum-Probability	Tilfeller av det dominante parret med naboer
GLNUN	Heterogenitet i intensitetsverdier
Id	Lokal homogenitet
LRLGLE	Lange rekker med små verdier
Idm	Lokal homogenitet
Strength	Små endringer og grov tekstur

Tabell 4.10 gir en oversikt over de syv egenskapene med høyest korrelasjon med lokalregionalt tilbakefall, og tabell 4.11 gir en kort forklaring på betydningen av disse. Generelt var svulstegenskapene som beskriver lokal homogenitet, små intensitetsverdier og grov tekstur knyttet til progresjonsfri overlevelse, og egenskaper som beskriver heterogenitet i svulstene, store intensitetverdier, størrelse på svulsten og fin tekstur knyttet til lokalregionalt tilbakefall.

Tabell 4.10: *Egenskaper som har høyest korrelasjon med hendelse tilbakefall. Høy verdi av disse egenskapene er dermed knyttet til lokalregionalt tilbakefall. Siden flere egenskaper er knyttet til diameteren til svulsten, er kun den med høyest korrelasjon inkludert (MajorAxis). Opprinnelig kom også Maximum2DDiameterColumn, Maximum3DDiameter og Maximum2DDiameterRow ble rangert på 2., 4., og 6. plass, men er også et mål for diameter.*

Type	**Filter**	**Klasse**	**Egenskap**	**Korr**	**p-verdi**
Begge	Ingen	Form	MajorAxis	0.26	0.0003
PET	Wavelet HLL	GLSZM	ZE	0.26	0.0004
CT	Wavelet LHL	Førsteordens	Energy	0.24	0.0009
CT	Logaritme	GLRLM	GLNU	0.24	0.0011
PET	Wavelet HLH	GLSZM	SAE	0.24	0.0012
CT	Wavelet LHL	GLSZM	SZNU	0.24	0.0016
PET	Wavelet HLH	GLRLM	RE	0.23	0.0017

Tabell 4.11: *Kort forklaring på betydningen av de syv egenskapene som hadde høyest korrelasjon med hendelse lokalregionalt tilbakefall. En mer utfyllende forklaring er gitt i vedlegg A.*

Egenskap	**Kort forklaring**
MajorAxis	Lengste akse
ZE	Heterogenitet i teksturen
Energy	Høye intensitetsverdier
GLNU	Heterogenitet i intensitetsverdier
SAE	Fin tekstur
SZNN	Heterogenitet i volum på størrelsessoner
RE	Heterogenitet i teksturen

Til sammenligning viser 4.12 Pearson korrelasjonen mellom kliniske faktorene og behandlingsutfallet, i rekkefølge fra høyest til lavest (absoluttverdi). Tabellen viser at stadium er den kliniske faktoren med høyest korrelasjon med behandlingsutfallet. Korrelasjonen for HPV-status ble kun beregnet for pasientene der HPV-status var kjent.

Tabell 4.12: *Pearson korrelasjon mellom de tretten kliniske faktorene og behandligsutfallet. Positiv korrelasjon viser sammenheng med tilbakefall, mens negative verdier viser sammenheng med progresjonsfri overlevelse.*

Faktor	**Korr**	**p-verdi**
Stadium	0.21	0.004
ECOG	0.17	0.020
T-klassifiering	0.17	0.021
HPV	-0.16	0.023
N-klassifisering	0.13	0.077
Kjønn	-0.10	0.194
Cispatin	0.08	0.264
Alder	0.08	0.307
Pakkeår med røyk	0.08	0.307
ICD10	0.07	0.323
Naxogin dager	0.05	0.497
Charlson	0.04	0.596
Histologi	0.02	0.790

Siden det ble valgt å kun bruke kvadratrotfilteret som bildetransformasjon, ble det på nytt foretatt en univariat analyse av Pearson korrelasjonen mellom behandlingsutfallet og egenskaper trukket ut fra kvadratrottransformerte PET- og CT-bilder. Tabell 4.13 og 4.14 viser egenskaper med hhv. høyest negativ og høyest positiv korrelasjon med hendelse lokalregionalt tilbakefall.

Tabell 4.13: *Oversikt over de fem egenskapene, trukket ut fra kvadratrottransformerte PET- og CT-bilder, med høyest negativ korrelasjon med tilbakefall. Dette betyr at høyere verdi av disse egenskapene har en sammenheng med progresjonsfri overlevelse.*

Type	**Klasse**	**Egenskap**	**Kort forklaring**	**Korr**	**p-verdi**
Begge	Form	Sphericity	Rundhet	-0.19	0.009
CT	NGTDM	Coarseness	Lokalt ensformig tekstur	-0.16	0.027
CT	GLDM	SDLGLE	Små områder med lave verdier	-0.16	0.030
CT	GLSZM	LGLZE	Soner med lave verdier	-0.14	0.047
CT	GLSZM	SALGLE	Små områder med lave verdier	-0.14	0.051

Tabell 4.14: *Oversikt over de fem egenskapene, trukket ut fra kvadratrottransformerte PET- og CT-bilder, med høyest positiv korrelasjon med tilbakefall. Siden flere egenskaper er knyttet til diameteren til svulsten, er kun den med høyest korrelasjon inkludert (MajorAxis). Opprinnelig kom også Maximum2DDiameterColumn, Maximum3DDiameter og Maximum2DDiameterRow ble rangert på 2., 3., og 4. plass, men er også et mål for diameter.*

Type	**Klasse**	**Egenskap**	**Kort forklaring**	**Korr**	**p-verdi**
Begge	Form	MajorAxis	Lengste akse	0.26	0.0003
Begge	Form	SurfaceArea	Overflateareal	0.20	0.003
Ingen	Kliniske	Stadium	Stadiumet til tumor	0.20	0.004
PET	GLRLM	RE	Heterogenitet i teksturen	0.20	0.005
CT	GLSZM	GLNU	Heterogenitet i intensitetsverdier	0.20	0.005

4.3 Multivariate resultater

4.3.1 Sammenligning av klassifikasjonsalgoritmer og egenskapsutvelgere

I dette delkapittelet presenteres hovedresultatene, som er den endelige ytelsen til klassifiseringsmodellene på datasettet, gitt som ROC gjennom kryssvalidering, slik som beskrevet i avsnitt 3.6.4.

Basert på de innledende analysene i avsnitt 4.1, ble PET- og CT-bildene kvantifisert med henholdsvis 16 og 128 intensitetsverdier, og transformert med kvadratrotfilteret. I tillegg til bildeegenskapene fra de transformerte bildene, ble formegenskaper og kliniske faktorer inkludert i datasettet.

Figur 4.6 viser et såkalt *heatmap* over gjennomsnittlig AUC for 40 kryssvalideringer av klassifikasjonsalgoritmene, i kombinasjon med forskjellige egenskapsutvelgere. Disse 40 kryssvalideringene ble utført ved å bruke en 4-folds-kryssvalidering 10 ganger, med forskjellige delinger av datasettet hver gang. 95%-konfidensintervaller for ytelsen til modellene, gitt i AUC, er vist i tabell 4.15 for egenskapsutvelgerne, og tabell 4.16 for dimensjonsreduksjonsmetodene. I gjennomsnitt varierer ytelsen på modellene i underkant av 10%, som tyder på at foldene representerer dataene godt.

	ReliefF	LDA	RF	LOG-1	MI	PCA	ICA
PLSR	0,66	0,60	0,60	0,57	0,57	0,57	0,57
LOG-2	0,66	0,60	0,60	0,57	0,56	0,57	0,56
LDA	0,66	0,60	0,60	0,57	0,56	0,56	0,56
LOG-1	0,66	0,60	0,60	0,57	0,58	0,55	0,55
QDA	0,65	0,62	0,59	0,53	0,56	0,58	0,57
AdaBoost	0,66	0,60	0,59	0,57	0,56	0,54	0,54
Linear SVC	0,63	0,60	0,60	0,57	0,54	0,54	0,54
Neural Net	0,64	0,60	0,58	0,58	0,53	0,54	0,54
SVC	0,65	0,60	0,57	0,57	0,54	0,51	0,51
GNB	0,65	0,62	0,60	0,54	0,54	0,49	0,49
Mars	0,63	0,58	0,55	0,57	0,53	0,51	0,51
KNN	0,61	0,59	0,58	0,55	0,53	0,50	0,50
RF	0,62	0,58	0,56	0,53	0,51	0,52	0,52
Beslutningstre	0,61	0,59	0,54	0,50	0,52	0,51	0,50

Figur 4.6: *Gjennomsnittlig AUC for 40 testinger av klassifikasjonsalgoritmene (radene) i kombinasjon med egenskapsutvelgerne (kolonnene). Datasettet inkluderer egenskaper trukket fra kvadratrottransformerte PET- og CT-bilder, formegenskaper og kliniske faktorer.*

Generelt gav egenskapsutvelgeren ReliefF høyest ytelse, etterfulgt av LDA og RF. PCA og ICA, som er dimensjonsreduksjoner som ikke bruker responsen/behand-

lingsutfallet, gav lavest ytelse. Flere av klassifikasjonsalgoritmene gav modeller med relativt høy ytelse. Kombinasjonen av ReliefF sammen med PLSR, Logistisk regresjon, LDA eller AdaBoost gav modeller med høyest ytelse, som hadde gjennomsnittlig AUC på 0.66 over 40 testinger, og standardavvik på 0.10. Derimot gav KNN, RF og beslutningstrær jevnt over lavere ytelse enn de andre klassifiseringsalgoritmene. Disse klassifiseringsalgoritmene, i kombinasjon med dimensjonsreduksjonsmetodene PCA og ICA, gav ytelse på høyde med tilfeldig gjetning.

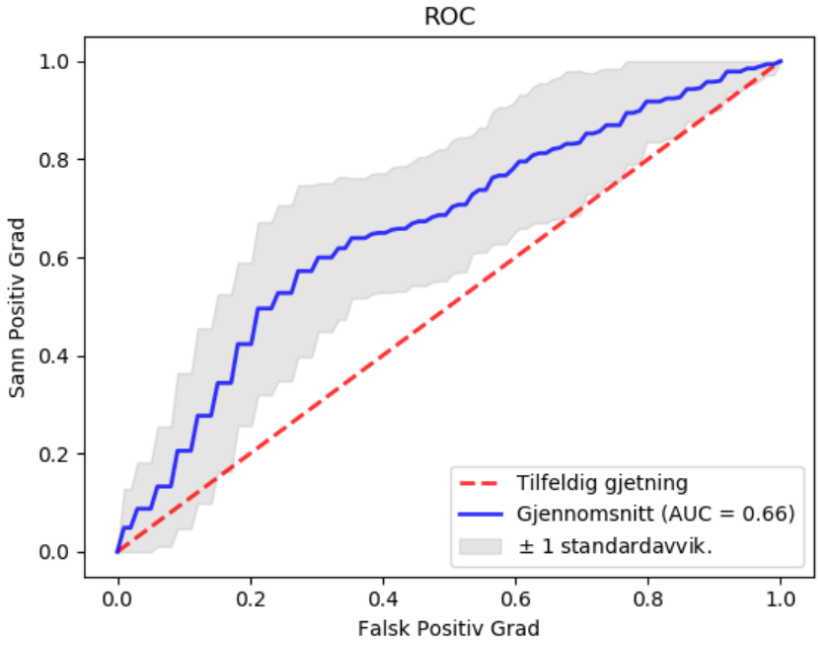
Tabell 4.15: *Konfidensintervaller for ytelsen gitt i AUC med et signifikansnivå* $\alpha = 0.05$ *for egenskapsutvelgerne i kombinasjon med klassifikasjonsalgoritmene, for modellene vist i figur 4.6.*

	ReliefF	**MI**	**RF**	**LOG-1**
PLSR	[0.63,0.68]	[0.54,0.60]	[0.57,0.63]	[0.54,0.60]
LOG-2	[0.63,0.69]	[0.53,0.59]	[0.57,0.63]	[0.54,0.56]
LDA	[0.63,0.69]	[0.53,0.60]	[0.57,0.63]	[0.54,0.60]
LOG-1	[0.63,0.69]	[0.55,0.61]	[0.57,0.63]	[0.54,0.56]
QDA	[0.62,0.68]	[0.53,0.59]	[0.56,0.62]	[0.50,0.57]
AdaBoost	[0.63,0.69]	[0.53,0.59]	[0.56,0.62]	[0.54,0.60]
Lineær SVC	[0.59,0.67]	[0.52,0.57]	[0.57,0.63]	[0.55,0.56]
NN	[0.61,0.67]	[0.50,0.56]	[0.54,0.61]	[0.55,0.61]
SVC	[0.61,0.68]	[0.52,0.57]	[0.54,0.61]	[0.54,0.60]
GNB	[0.62,0.68]	[0.51,0.57]	[0.57,0.63]	[0.51,0.57]
Mars	[0.60,0.66]	[0.51,0.56]	[0.52,0.59]	[0.54,0.60]
KNN	[0.59,0.64]	[0.51,0.55]	[0.55,0.60]	[0.52,0.57]
RF	[0.59,0.66]	[0.48,0.53]	[0.53,0.59]	[0.49,0.56]
Beslutningstre	[0.58,0.65]	[0.50,0.54]	[0.51,0.57]	[0.47,0.53]

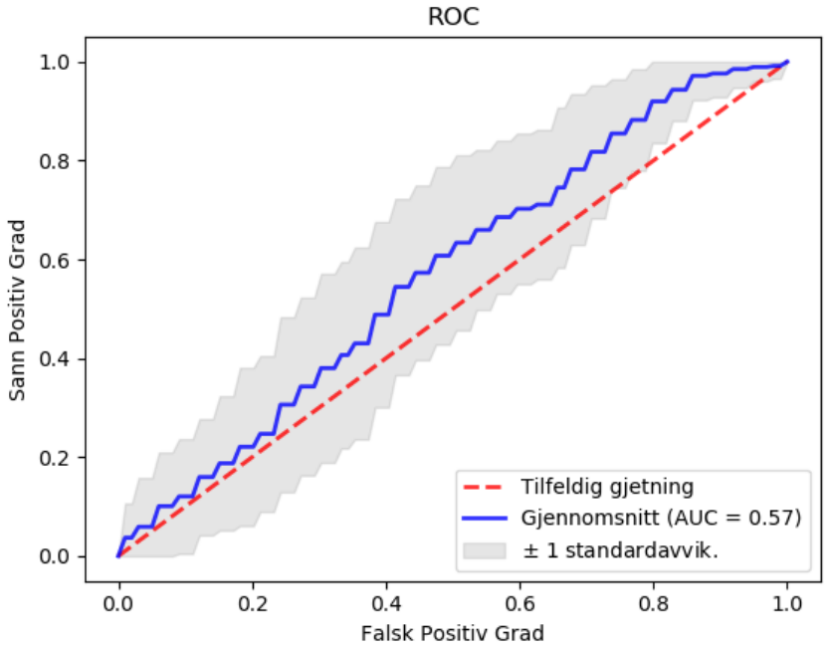
Gjennomsnittlig ROC for de 40 modellene, bygget på kombinasjoner av ReliefF og logistisk regresjon, er vist i figur 4.7, og gav en gjennomsnittlig AUC på 0.66 (standardavvik 0.10). Til sammenligning er ROC for kun kliniske faktorer vist i figur 4.8 og hadde en AUC på 0.57 (standardavvik 0.12). Disse kliniske klassifiseringsmodellene ble beregnet ved bruk av ReliefF som egenskapsutvelger og Logistisk regresjon med L1-tap som klassifikasjonsalgoritme, ettersom denne kombinasjonene generelt gav høy ytelse for bildeegenskapene (figur 4.6). Antall kliniske faktorer som ble tatt med i klassifiseringsmodellen, ble bestemt basert på figur 4.9, som viser AUC for klassifisering basert på 1-13 kliniske faktorer.

Tabell 4.16: *Konfidensintervaller for ytelsen gitt i AUC med et signifikansnivå* $\alpha = 0.05$ *for dimensjonsreduksjonsmetodene i kombinasjon med klassifikasjonsalgoritmene, for modellene vist i figur 4.6.*

	LDA	**PCA**	**ICA**
PLSR	[0.56,0.63]	[0.54,0.60]	[0.53,0.59]
LOG-2	[0.56,0.63]	[0.53,0.60]	[0.53,0.58]
LDA	[0.56,0.63]	[0.53,0.60]	[0.53,0.59]
LOG-1	[0.56,0.63]	[0.52,0.58]	[0.52,0.57]
QDA	[0.59,0.65]	[0.54,0.61]	[0.51,0.57]
AdaBoost	[0.56,0.63]	[0.50,0.58]	[0.53,0.59]]
Lineær SVC	[0.56,0.63]	[0.51,0.58]	[0.52,0.59]
NN	[0.56,0.63]	[0.51,0.57]	[0.52,0.57]
SVC	[0.57,0.63]	[0.48,0.54]	[0.52,0.58]
GNB	[0.59,0.65]	[0.46,0.52]	[0.48,0.53]]
Mars	[0.55,0.61]	[0.485,0.53]	[0.48,0.53]
KNN	[0.56,0.62]	[0.48,0.52]	[0.49,0.53]
RF	[0.56,0.61]	[0.50,0.55]	[0.49,0.54]
Beslutningstre	[0.56,0.61]	[0.49,0.53]	[0.50,0.55]



Figur 4.7: *Gjennomsnittlig AUC for 40 valideringer ved bruk av ReliefF og Logistisk regresjon, for datasettet bestående av bildeegenskaper fra kvadratrottransformerte PET- og CT-bilder, formegenskaper og kliniske faktorer.*



Figur 4.8: *Gjennomsnittlig AUC for 40 valideringer med kliniske faktorer som egenskaper og ved bruk av ReliefF og logistisk regresjon med L1-regularisering.*

Figur 4.9: *AUC for modellen laget av kliniske faktorer som funksjon av antall kliniske faktorer. 11 antall kliniske faktorer ble valgt, og er markert med X.*

4.3.2 Utvalgte egenskaper

Ettersom både egenskapsutvelgelse og klassifikasjon foregår i hver kryssvalidering, ble det også valgt ut egenskaper 40 forskjellige ganger, basert på de forskjellinge delene (foldene) av datasettet brukt i modell-treningen. ReliefF, som ble egenskapsutvelgeren som jevnt over gav høyest ytelse, valgte ut to egenskaper i hver validering/trening.

Tabell 4.17 viser hvilke egenskaper som ble valgt ut av ReliefF gjennom de 40 valideringene, og antall ganger de ble valgt ut. Egenskapen MajorAxis ble valgt ut 35 ganger, altså nesten hver gang. Det tyder på at egenskapsutvelgeren ReliefF valgte dette som viktigst for behandlingsutfallet. I tillegg til størrelse på diameterne

til svulstene, ble også Busyness valgt ut, som kvantifiserer hvor hyppig verdiene i voxlene endrer seg fra en voxel til voxlene rundt; altså raske endringer i bildet. To ganger ble også Dependence Entropy og SUVpeak valgt ut, som beskriver henholdsvis heterogeniteten i teksturen og maksimal gjennomsnittlig SUV.

Tabell 4.17: *Egenskaper valgt ut av egenskapsutvelgeren ReliefF i 40 klassifiseringer på forskjellige deler/folder av datasettet.*

Bildetype	**Klasse**	**Egenskap**	**Antall**
Begge	Form	Major Axis	35
Begge	Form	Maximum3DDiameter	12
Begge	Form	Maximum2DDiameterRow	11
Begge	Form	Maximum2DDiameterColumn	10
CT	NGTDM	Busyness	8
PET	GLDM	Dependence Entropy	2
PET	Ingen	SUVpeak	2

Tabell 4.18 viser hvilke egenskaper som ble tildelt mest vekt når LDA ble brukt som dimensjonsreduksjonsmetode. Vekten ble regnet ut som gjennomsnittet av absoluttverdiene av alle koeffisientene for hver av de 40 LDA-komponentene, brukt for å redusere datasettet til kun én dimensjon. Egenskaper knyttet til det meste dominante paret med intensitetsverdier, rundheten til tumor og heterogenitet i volumet til størrelsesssoner ble tildelt mest vekt.

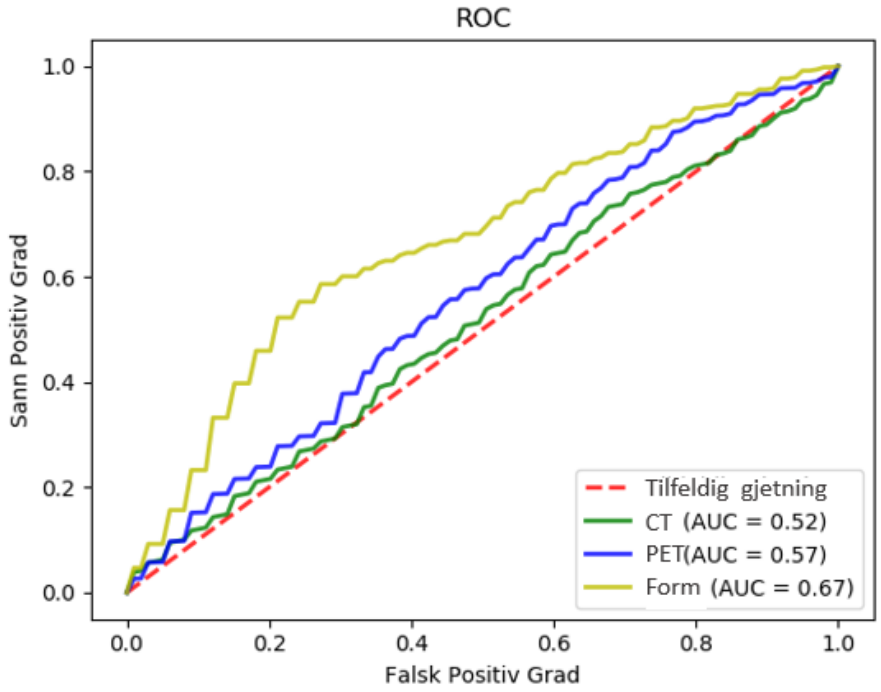
Tabell 4.18: *De ti egenskapene med gjennomsnittlig høyest vekt i 40 LDA-komponenter. Kolonnen til høyre viser gjennomsnittlig (absolutt) vekt.*

Bildetype	**Klasse**	**Egenskap**	**Vekt**
PET	GLCM	MaximumProbability	2.5
Begge	Form	Elongation	2.4
PET	GLSZM	SizeZoneNonUniformityNormalized	2.3
CT	GLRLM	LongRunLowGrayLevelEmphasis	1.9
CT	Førsteordens	Skewness	1.9
PET	GLSZM	LowGrayLevelZoneEmphasis	1.8
PET	GLCM	JointEnergy	1.8
Begge	Form	Maximum3DDiameter	1.7
CT	GLCM	ClusterShade	1.7
PET	GLCM	Correlation	1.6

4.3.3 Sammenligning PET- og CT-bildeegenskaper

Etter at PET-og CT-bildene ble transformert gjennom kvadratrotfilteret, ble egenskapene trukket ut av disse to transformerte bildetypene sammenlignet, slik som beskrevet i avsnitt 3.8.1. Generelt trakk ReliefF kun ut formegenskaper fra X_{CT} og X_{PET}. Fra X_{CT} ble kun formegenskaper valgt ut, mens fra X_{PET} ble et også noen PET-bildeegenskaper valg ut. Dette gav X_{CT} en høyere AUC ($AUC = 0.68$) enn det X_{PET} gjorde ($AUC = 0.66$).

Dette resultatet beskriver ikke intensitets- og teksturegenskapene fra PET- og CT-bildene godt, ettersom de ikke ble valgt ut til å være med i modellene. Dermed ble intensitets- og teksturegenskapene fra PET- og CT-bildene, eksludert formeegenskapene og de kliniske faktorene X_{PET-ny} og X_{CT-ny}, undersøkt. ROC-kurvene for prediksjonen av lokalregionalt tilbakefall ved bruk av disse to matrisene, og fremgangsmåten beskrevet i avsnitt 3.8.2, er gitt i figur 4.10. Her vises også ytelsen til X_{form} (gul linje), som er en datamatrise som bare inneholder formegenskapene til svulstene.

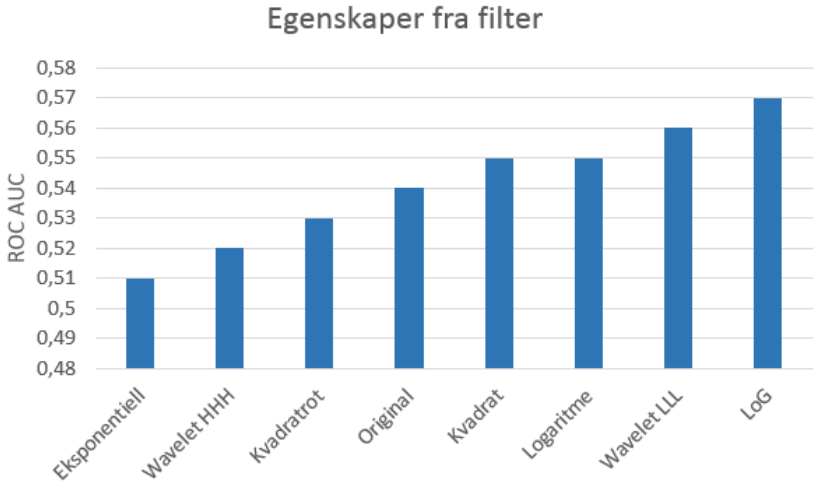


Figur 4.10: *ROC-kurver for klassifikasjon av behandlingsutfall ved bruk av PET-bildeegenskapene X_{PET-ny} (blå), CT-bildeegenskapene X_{CT-ny} (grønn) og kun formegenskaper X_{form} (gul). Før klassifikasjonen ble to egenskaper valgt ut av matrisene, til å lage modellene.*

Prediksjonen ved bruk av egenskapene fra X_{PET-ny} gav en høyere AUC enn X_{CT-ny} med hhv. 0.57 og 0.52, men kun bruk av formegenskapene gav en god del høyere AUC på 0.67.

4.3.4 Sammenligning av filtertransformasjoner ekskludert kliniske faktorer og formegenskaper

Figur 4.11 viser at egenskapene trukket ut fra bildene, transformert med LoG-filteret X_{LoG}, gav den høyeste klassifiseringsytelsen med AUC på 0.57, mens bildene transformert med eksponentiell-filteret $X_{eksponentiell}$ gav den laveste klassifiseringsytelsen med AUC på 0.51. Egenskapene fra det utvalgte filteret, $X_{kvadratrot}$, var blandt de som gav lavere ytelse, med AUC på 0.53.



Figur 4.11: *AUC for klassifiseringsmodeller basert på fem egenskaper trukket ut fra datamatrisene for filter-transformerte PET-bilder $X_{filter-ny}$. Kliniske faktorer og formegenskaper var ikke inkludert i datamatrisene. Disse datamatrisene ble testet ved bruk av 4-folds-kryssvalidering fem ganger. Fra egenskapene trukket fra hver av de filter-transformerte bildene ble fem egenskaper valgt ut. Altså består f.eks. modellen beskrevet som Eksponentiell av fem av alle egenskapene fra $X_{eksponentiell-ny}$.*

4.4 HPV-status og behandlingsutfall

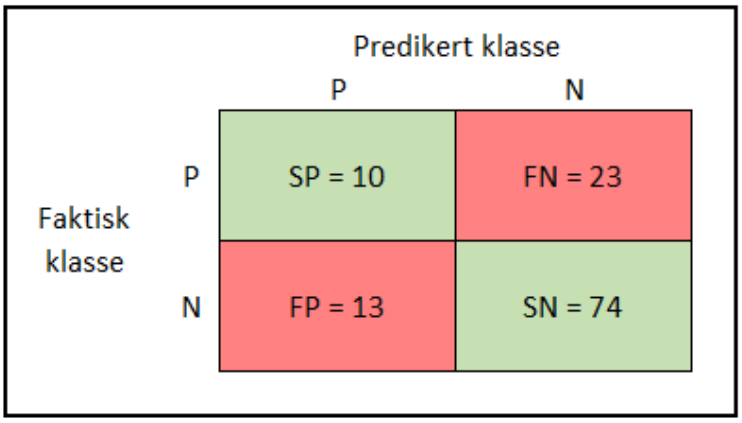
Andre studier [57] [58] har indikert at HPV-status kan ha en sammenheng med behandlingsutfallet. Derfor ble dette også undersøkt i denne oppgaven. I tabell 4.12 ble det vist at HPV-status hadde korrelasjon på -0.16 med lokalregionalt tilbakefall. Denne analysen var basert på 93 pasienter som hadde fått CT-kontrastvæske, og var dermed inkludert i det endelige datasettet. I analysene i dette delkapittelet var alle pasienter med kjent HPV-status og klasser lokalregionalt tilbakefall eller progresjonsfri overlevelse inkludert, uavhengig om pasientene hadde fått kontrastvæske eller ikke (120 pasienter). Blandt disse pasientene var 23 pasienter HPV-negative, og 33 pasienter hadde fått lokalregionalt tilbakefall.

HPV-status gav, med disse 120 pasientene, en Pearson korrelasjon på -0.17 med lokalregionalt tilbakefall. Dermed ble HPV-status en av de kliniske faktorene med størst lineær korrelasjon med behandingsutfallet, sammen med stadium på 0.20 og

ECOG på 0.17. Det negative fortegnet betyr at pasienter med HPV hadde større sannsynlighet for å ikke få lokalregionalt tilbakefall.

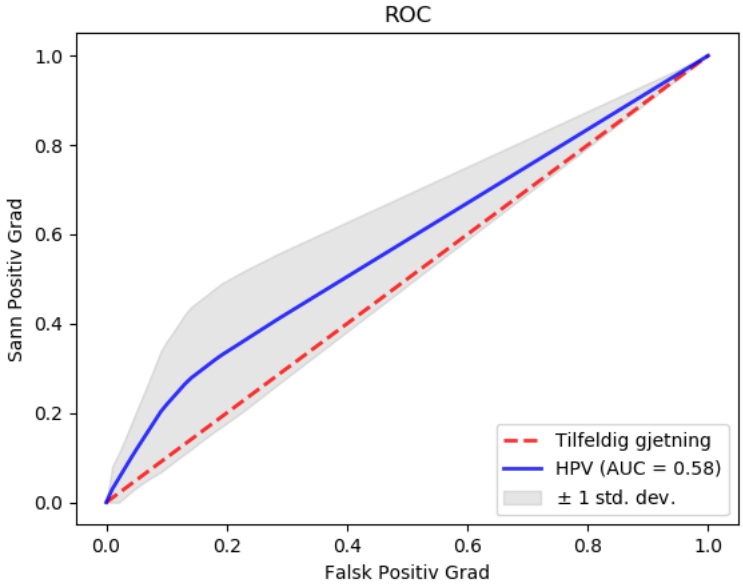
Pearson korrelasjonen mellom HPV-status lokalregionalt tilbakefall for pasienter med kreft i oropharynx-området, ble også undersøkt. Dette gav en lavere korrelasjon på -0.10. Dette delsettet besto av 112 pasienter, der 17 pasienter var HPV-negative og 28 pasienter hadde hatt lokalregionalt tilbakefall.

Hvis kun faktoren HPV-status ble brukt til å predikere behandlingsutfall (Positiv HPV-status = progresjonsfri overlevelse og negativ HPV-status = tilbakefall), ble feilmatrisen gitt som figur 4.12 og gir SPG = 0.30 og FPG = 0.13. Figur 4.12



Figur 4.12: *Feilmatrise for hvordan kun HPV-status predikerte lokalregionalt tilbakefall. Her er den predikerte klassen N hvis HPV-status var positiv, og P hvis HPV-status var negativ.*

gir også at $\frac{74}{74+23} = 76\%$ av pasientene med positiv HPV-status fikk progresjonsfri overlevelse etter behandling, mens dette gjaldt kun $\frac{13}{10+13} = 57\%$ av pasientene med negativ HPV-status (ettersom predikert lokalregionalt tilbakefall er det samme som at pasienten hadde negativ HPV-status). Ved bruk av faktoren HPV-status og Logistisk regresjon som klassifikasjonsalgoritme, ble ROC-kurven figur 4.13



Figur 4.13: *ROC-kurven for prediksjon av lokalregionalt tilbakefall ved bruk av Logistisk regresjon som klassifikasjonsalgoritme og HPV-status som variabel.*

Kapittel 5

Diskusjon

5.1 Datasettet

PET- og CT-bildene i denne analysen, er av pasienter med behandlinger i tidsrommet 2007-2013. Siden tidsrommet er stort, er det flere forhold kan ha endret seg underveis. Det ble f.eks. lagt merke til at størrelsene på bildene endret seg for omtrent 31 pasienter tatt i overgangen 2012-2013, noe som tyder på at bildene har blitt tatt med en annen prosedyre. I tillegg kan svulstene i bildene ha blitt tegnet inn av forskjellige onkologer eller leger, som medfører variasjon i hva som blir tegnet inn som en del av svulsten og ikke. Weiss og Hess viste at interobservervariasjonen (variasjonen i inntegning mellom onklologer) var en av de største faktorene for unøyaktighet i inntegning av kreftsvulster [59].

Noen av faktorene i regnearket om pasientinformasjon var også ufullstendige. HPV-status er kun kjent for 100 pasienter, og histologisk diagnostikk er heller ikke kjent for alle pasientene. Dette gjør at disse er vanskelige å inkludere i analysen, ettersom det er brukt algoritmer som ikke nødvendigvis fungerer på manglende data. I denne oppgaven var ukjent HPV-status eller ukjent histologisk diagnostikk definert som en ny klasse. F.eks. HPV-negativ = 0, HPV-positiv = 1 og ukjent HPV-status = 2, slik at disse faktorene kunne benyttes i analysen.

5.2 Kryssvalidering

Det er viktig at alle typer egenskapsutvelgelse, valg av beste modellparametere, trening og testing av modellene, blir gjort ordentlig gjennom samme løkke av kryssvalidering, for å ikke få et for optimistisk resultat [22]. Hvis egenskapsutvelgelsen skjer i forkant av denne kryssvaliderings-løkken, kan det bli valgt ut egenskaper som helt tilfeldig hadde en stor sammenheng med behandlingsutfallet i dette datasettet, men som ikke vil ha en sammenheng med fremtidig data. Da vil klassifiseringsytelsen

med disse dataene ikke være beskrivende for nye data.

En typisk, feil kryssvalidering er som følger [22]:

1. Bruk en egenskapsutvelger for å velge en delmengde med egenskaper, som har en sammenheng med responsen. Dette gjøres på hele datasetet, og er ikke en del av kryssvalideringsløkken.
2. Bruk delmengden fra punkt 1. til å bygge en multivariat klassfikasjonsmodell.
3. Bruk kryssvalidering til å optimalisere parametere og koeffisienter til modellen på treningsfoldene og predikere ytelsen til modellen på validerings-foldene.

Dermed er klassifiseringsmodellen bygget basert på egenskaper trukket ut fra alle observasjonene, ikke bare de i treningssettet. Modellen har dermed "sett"alle observasjonene, og vil dermed gi en overoptimistisk prediksjon av modellytelsen. Valideringsobservasjonene skal holdes utenfor all trening. Dette inkluderer valg av egenskaper, optimalisering av modellparametere og trening av modellen.

Det er flere publiserte artikler som gjør kryssvalideringen på måten beskrevet ved punkt 1., 2., og 3., og velger ut egenskaper før kryssvalideringen for å predikere ytelsen. Dette gir et resultat som oftest er bedre enn det vil være på nye, usette data. Et eksempel på en artikkel som har utført kryssvalidering på denne måten er Klement et al. [60], som predikerte lokalregionalt tilbakefall i lungekreft med en AUC på 0.73. Her ble den overoptimistiske metoden brukt, til tross for at artikkelen refererte til en metode beskrevet Chen et al. [61], som fokuserte på å gjøre kryssvalideringen på riktig måte. Chen et al. [61] oppnådde en AUC på 0.76 for prediksjon av stråling-indusert pneumonitt i lunger ved bruk av dose-relaterte egenskaper. Et annet eksempel på en artikkel som gjør kryssvalideringen på måten beskrevet ved punkt 1., 2., og 3., er Valliéres et. al [62], som brukte radiomics for å predikere metastase fra lungekreft. Her brukes bootstrapping istedet for kryssvalidering, men det ble fortsatt valgt ut en delmengde med egenskaper før ytelsen ble predikert med bootstrapping. Dette kan forklare den høye ytelsesprediksjonen til den beste modellen, med en AUC på 0.984.

I denne oppgaven var det tre optimeringssteg som ble utført utenfor løkken med kryssvalidering. Dette var kvantifiseringen av intensitetsverdiene til bildene, det å finne det beste filteret og å finne optimalt antall egenskaper. Optimalt burde disse optimeringsstegene også vært utført i samme kryssvaliderings-løkke som prediksjonen av modellytelesen, men det ville blitt for tidskrevende for denne oppgaven. Da måtte klassifiseringer, basert på egenskaper fra seks forskjellige kvantifiseringer av begge bildetypene, med syv forskjellige filtertransformeringer (inkludert originale bilder) og med tyve forskjellige antall egenskaper, blitt utført i samme kryssvalideringsløkke. Det ville blitt et omfattende og tregt Python-program. Det ble vurdert at disse tre stegene ikke ville ha stor innvirkning på overtilpassning av dataene. Det kan likevel ha hatt en liten positiv innvirkning på den endelige prediksjonsytelsen, men mest sannsynlig ikke av stor betydning. Dette er på grunn av at det var såpass små forskjeller mellom alternativene:

1. Kvantifisering av intensitetsverdier: De forskjellige antall bins gav gjennomsnittlig AUC mellom 0.60 og 0.65 for PET-bildene, og mellom 0.56 og 0.58 for CT-bildene.

2. Bildetransformasjon: Det kvadratrottransformerte bildet gav AUC_{en} på 0.65 for både PET-bildene og CT-bildene, mens de originale bildene gav AUC_{en} på 0.64 for PET-bildene og 0.65 for CT-bildene.

3. Antall egenskaper: Ved å bruke antall egenskaper mellom 1-7, valgt ut av ReliefF, varierte prediksjonsytelsen til modellene mellom 0.62 og 0.64. Dermed ville ikke prediksjonsytelsen endret seg mye ved å fjerne eller legge til et par egenskaper.

Et alternativ til kryssvalideringen er å dele datasettet i et treningssett og valideringssett, slik som beskrevet i avsnitt 3.6.4. Å få tak i mer data, eller et uavhengig test-sett, hadde vært et godt alternativ. Det ble observert at datasettet i denne oppgaven gav såpass varierende modeller og resultater ved forskjellige inndelinger. Ved å dele datasettet i to forskjellige deler, kunne to forskjellige inndelinger gi en AUC på alt fra 0.4 til 0.8. Det betyr at dataene varierer mye, og at en tilfeldig inndeling hadde stor innvirkning på prediksjonsytelsen. Derfor ble det valgt å bruke kryssvalidering med 40 testinger, og ta gjennomsnittet av disse, for å få en mer gjennomsnittlig indikasjon på hvordan modellen vil gjøre det på fremtidige, usette data.

5.3 Kvantifisering av intensitetsverdier

Antall intensitetsverdier til bildene ble satt til 16 gråtonenivåer for PET-bildene og 128 gråtonenivåer for CT-bildene. For mange gråtonenivåer i et bilde vil gjøre utregninger krevende, ettersom flere av teksturmatrisene blir av størrelsen $(antall\,bins)^2$. I tillegg er det andre faktorer som spiller inn på hvor mange gråtonenivåer bildene bør ha. Som eksempel ser GLRLM på rekker med like gråtoneverdier. Hvis det er for mange gråtonenivåer, kan det hende det blir ingen tilfeller av like gråtoneverdier ved siden av hverandre, og det blir meningsløst å regne ut en GLRLM. PyRadiomics har som standard å sette bredden på gråtonenivåene til 25. Dette betyr at hvis en opprinnelig har et 8-bits bilde med 256 gråtonenivåer, vil en få $\frac{256}{25} \approx 11$ gråtonenivåer. Dette tyder på at det er vanlig å ha et relativt lavt antall gråtonenivåer i bildene til teksturanalyse. Til sammenligning fant Valliéres i sin masteroppgave [63] at optimal oppløsning for PET-bilder i forhold til prediksjon av metastase for lungekreft var 16 eller 32 gråtonenivåer.

Det kan være flere årsaker til at CT-bildene ble kvantifisert med flere intensitetsverdier enn PET-bildene. En årsak kan være at intensitetsverdier i PET- og CT-bilder er assosiert med forskjellige fenomener. Dette kan gi en helt annen type romlig informasjon i de to bildetypene. Der PET-bildene viser den metabolske aktiviteten i celler, viser CT-bildene absorpsjon av røntgenstråler i vevet [4]. En annen årsak kan være at CT-bildene originalt hadde høyere romlig oppløsning enn PET-bildene, på

hhv. $1 \times 1 \times 2\,mm^3$ og $3 \times 3 \times 2\,mm^3$. Både PET- og CT-bildene var interpolert til en oppløsning på $1 \times 1 \times 1\,mm^3$, som fører til at endringen i intensitetsverdi fra en voxel til en annen var nærmere den opprinnelig endringen for CT-bildene enn for PET-bildene.

5.4 Kantdeteksjons-filter

Både Wavelet-filtrene og LoG-filtrene er filter som ser på nabovoxler når de skal beregne den nye verdien til en voxel [28]. Bildene brukt i denne oppgaven hadde kun verdier i voxlene som var innenfor inntegnet området av onkologer, og verdiene utenfor dette området var satt til 0. Dette betyr at voxlene langs kanten på tumor fikk en feil verdi ved bruk av disse filtrene, ettersom de ser på nabovoxlene, der naboene utenfor inntegnet tumor hadde verdi 0, og ikke den reelle verdien i bildet. Det er mulig at modellytelsen hadde blitt høyere dersom filtrene hadde blitt brukt på de opprinnelige bildene med verdier utenfor den inntegnede ROI.

5.5 Egenskapsutvelgere

Blandt egenskapsutvelgerne, var det de ikke-veiledende dimensjonsredukjsonsmetodene (ICA og PCA) som gav lavest prediksjonsytelse. Dette har mest sannsynlig en sammenheng med at disse kun ser på variasjonen i dataene, og ikke ser på hvordan disse oppfører seg i forhold til behandlingsutfallet. Det ser da ut som variasjonen i dataene ikke forklarer responsen godt. Derimot var LDA blandt egenskapsutvelgerne som gav høyest ytelse, som da er en dimensjonsreduksjonsmetode som bruker responsen.

Egenskapsutvelgelse med ReliefF og RF gav høyest ytelse ved bruk av få egenskaper, mens Logistisk regresjon og MI krevde et høyere antall egenskaper. Dette kan tyde på at ReliefF og RF med en gang fant egenskaper som var viktige for behandlingsutfallet, og deretter tilføyde resterende egenskaper mer støy. Derimot kunne det se ut som Logistisk regresjon og MI trengte å inkludere flere egenskaper, før de hadde fått med alle som var viktige.

Det ble bestemt at et spesifikt antall egenskaper skulle velges for hver egenskapsutvelger, f.eks. at ReliefF skulle trekke ut to egenskaper hver gang. Dette er en forenkling, ettersom det vil variere for hver klassifikasjonsalgoritme hvor mange egenskaper som gir høyest ytelse. Noen klassifikasjonsalgoritmer er mer robuste mot høydimensjonale data, og kunne kanskje fått økt ytelse ved å inkludere flere egenskaper. Eksempler på slike klassifikasjonsalgoritmer er MARS, beslutningsytrær og RF [26]. Ved bedre tid, ville det vært naturlig å bestemme et optimalt antall egenskaper for hver kombinasjon av egenskapsutvelger og klassifikasjonsalgoritme. Det er ikke mye tilleggsarbeid, men krever at alle egenskapsutvelgerne blir testet i kombinasjon med alle klassifikasjonsalgoritmene, noe som ville økt tidsbruken for Python-

programmet. Dette ble likevel gjort for ni av klassifikasjonsalgoritmene (som forklart i neste avsnitt). I tillegg må Python-programmet oppdateres for å kalkulere forskjellige antall egenskaper for de forskjellige klassifiseringsalgoritmene, noe som igjen ville gjort programmet tregere.

En annen forenkling som ble gjort, var å kun bruke de raskeste klassifikasjonsalgoritmene ved bestemmelse av optimalt antall egenskaper. Dette var gjort for å spare tid, ettersom klassifikasjonsalgoritmene ble kjørt gjennom en såkalt *Grid Search* (nøstet kryssvalidering) for hver runde, for å finne optimale modellparametere. I Grid Search deles dataene i trenings-foldene opp i nye trenings-folder og validerings-folder for å undersøke optimale parametere, slik som beskrevet under punkt 2b i avsnitt 3.6.4. Dette gjorde at noen av de tregere algoritmene brukte lenger tid, og det ble valgt å kun se på de ni raskeste algoritmene, slik at analysen kunne utføres i den tilgjengelige tiden. Dermed var ikke AdaBoost, NN, MARS, KNN og RF med å optimere antall egenskaper. Dette kan ha gitt disse algoritmene en ulempe ved den siste sammenligningen av modellene vist i figur 4.6. Det ble likevel tatt noen stikkprøver, ved at disse ble sjekket for noen tilfeldige folder. Det viste seg at antall egenskaper valgt var tilnærmet lik antall valgt ved å bruke forenklingen. Dette kan tyde på at forenklingen ikke hadde stor betydning på modellytelsen.

5.6 Overtilpasning ved egenskapsutvelgelse

Fra resultatene i avsnitt 4.1.2, ble det observert at klassifiseringen ble bedre ved å bruke egenskapene trukket ut fra bildene transformert med kun ett filter, istedet for å bruke alle 2699 egenskapene. Dersom f.eks. et bestemt filter gav de mest relevante bildeegenskapene, måtte egenskapsutvelgeren finne disse egenskapene blandt hundrevis fra andre filtre. Dette var ikke tilfellet, og tyder på at egenskapsutvelgerne ikke fungerer optimalt. Dette kan komme av dimensjonalitetens forbannelse [22], og at det blir vanskelig å finne hvilke egenskaper som faktisk har en sammenheng med responsen blandt så mange egenskaper. Det kan også hende at en egenskap har en tilfeldig sammenheng med treningssettet, og dermed blir valgt ut, uten å passe med valideringssettet. Dette kan tyde på at egenskapsutvelgerne ikke nødvendigvis fungerer optimalt med en for stor mengde egenskaper, og at de meste relevante egenskapene ikke blir funnet.

5.7 Sammenligning PET, CT og ulike filtre

Når behandlingsutfall ble predikert basert på kun bilde- og teksturegenskaper uten formegenskaper, gav PET-bildene egenskaper med høyere AUC enn CT-bildeegenskaper (tabell 4.10). Dette stemmer overens med andre studier [64] [65]. Eksempelvis fant Vallieres et al. [64] at egenskaper trukket ut fra CT-bilder alene gav en AUC i forhold til prediksjon av lokalregionalt tilbakefall for hode- og halskreft på rundt 0.47, mens egenskaper trukket ut fra PET-bilder alene gav en AUC på rundt 0.53. Når

egenskaper fra begge bildetypene ble slått sammen, fikk Vallieres et al. en AUC på 0.64. Bogowics et al. [65] fant at PET- og CT-bildeneegenskaper gav relativt lik ytelse i forhold til prediksjon av lokalt tilbakefall for hode- og halskreft, men anbefalte likevel å bruke PET-bildeegenskaper. Denne anbefalingen kom på bakgrunn av at egenskapene fra CT-bildene overestimerte sannsynligheten for lokalt tilbakefall for høyrisiko-pasienter.

Formegenskapene alene X_{form} gav en høyere ytelse enn både CT-bildeegenskapene X_{CT-ny} og PET-bildeegenskapene X_{PET-ny}, og viser at formegenskapene var overlegne i forhold til å predikere behandlingsutfallet. Når formegenskapene var inkludert i datamatrisene X_{PET} og X_{CT}, gav X_{CT} den høyeste ytelsen. Årsaken til dette er nok at kun formegenskaper ble valgt ut ved bruk av X_{CT}, ettersom CT-teksturegnskapene ikke hadde en stor sammenheng med behandlingsutfallet. Derimot ble noen av PET-teksturegeskapene valgt ut ved bruk av X_{PET}, istedet for formegenskaper. Resultatene tyder igjen på at formegenskapene har størst sammenheng med behandlingsutfallet. Dersom noen av disse blir byttet ut med PET-egenskaper, går ytelsen til modellen ned. Resultatene indikerer videre at PET-teksturegenskapene X_{PET-ny} har større sammenheng med behandlingsutfallet enn CT-teksturegenskapene X_{CT-ny}.

Denne effekten ble også observert i analyser av filtertransformerte PET- og CT-bilder. Når formegenskaper, kliniske faktorer samt egenskaper fra de filtertransformerte bildene var inkludert i datasettene $Xfilter$, gav LoG-filteret og Wavelet LLL-filteret lavest ytelse (tabell 4.5). Derimot, når formegenskapene og de kliniske faktorene var ekskludert fra datasettene $X_{filter-ny}$, som beskrevet i avsnitt 4.1.2, gav egenskaper fra bilder transformert med LoG-filteret og Wavelet LLL-filteres høyest ytelse (figur 4.11). Kvadratrotfilteret $X_{kvadratrot}$, som opprinnelig ble valgt ut som optimalt filter, var blandt de som gav lavest klassifikasjonsytelse når formegenskapene var ekskludert. Dette tyder igjen på at egenskaper trukket ut av filtertransformerte bilder, som gav god ytelse uten formegenskapene, er blant de som gav dårlig ytelse når formegenskapene ble inkludert. Disse egenskapene har såpass stor sammenheng med behandlingsutfallet, at de kan bli valgt ut av egenskapsutvelgeren istedet for formegenskapene, og modellen får lavere ytelse.

5.8 HPV-status og behandlingsutfall

Resultatene, både fra univariate og multivariate analyser, tyder på at det er en sammenheng mellom HPV-status og behandlingsutfallet. Pearson-korrelasjonen på -0.17 var blant de høyeste for de kliniske faktorene. ROC-kurven gav en AUC for prediksjonsmodellen basert på HPV-status, som var høyere enn ved bruk av de andre kliniske faktorene, henholdsvis 0.58 og 0.57. Dette viser at HPV-status er en av de sterkere prediktorene for behandlingsutfall, i forhold til kliniske egenskaper.

Det var ikke uventet å finne en sammenheng mellom HPV-status og behandlingsutfall, ettersom det er flere studier som har funnet akkurat dette, eksempelvis Nichols et al. [57] og Chaturvedi et al. [58]. Sistnevnte fant at median overlevelse (tiden fra

diagnostisering til kun halvparten av pasientene er i live) for pasienter med kreft i oropharynxvar, var 131 måneder for pasienter med HPV, og 20 måneder for pasienter uten HPV. Nichols et al. fant at 82% av pasienter med positiv HPV-status og kreft i oropharynx fikk progresjonsfri overlevelse, mens dette gjaldt kun 53% av de pasientene med negativ HPV-status. Dette er ikke ulikt resultatet fra denne oppgaven, der 76% av pasientene med positiv HPV-status fikk progresjonsfri overlevelse, mens dette gjaldt kun 57% av pasientene med negativ HPV-status.

Det var derimot uventet at sammenhengen mellom behandlingsutfall og HPV-status ble lavere (korrelasjon på -0.10) når datasettet ble komprimert til å kun inneholde pasienter med kreft i oropharynx. Dette kan skyldes at det ble færre pasienter med negativ HPV-status, og færre pasienter som fikk lokalregionalt tilbakefall i oropharynx-datasettet. Selv om kun 7% av pasientene ble fjernet fra hele datasettet, resulterte dette i at 26% av pasientene med negativ HPV-status og 15% av pasientene med lokalregionalt tilbakefall ble fjernet fra oropharynx-datasettet. Det indikerer at en større andel av pasientene med kreft i oropharynx hadde HPV, i forhold til pasienter med kreft i andre hode- og halsområder. Dette stemmer forsåvidt overens med flere studier, som fokuserer på at HPV-status har størst sammenheng med utvikling av kreft i oropharynx [3] [66] [67].

5.9 Forslag til videre arbeid

De fleste metodene for kvantifisering av tekstur (egenskapsklassene) har innstillingsparametere som kan varieres. Som eksempel er avstanden mellom voxler for GLCM og NGTDM i denne oppgaven satt til 1. Dette betyr at det kun ble sett på nabovoxler og dermed teksturer på små lengdeskalaer. Ved å øke avstanden mellom voxlene, er det mulig å kvantifisere teksturer på større lengdeskalaer (grovere teksturer). Dette kan fange opp andre typer mønstre i bildene. For GLDM kunne det vært interessant å bruke en annen terskelverdi enn 0 for avhengighet, f.eks. terskelverdi 1. Da vil voxler med én intensitetsverdi i forskjell sees på som avhengige.

Det finnes flere typer verktøy for å trekke ut bildeteksturegenskaper, enn de som er testet i denne oppgaven. Local Binary Patterns (LBP) transformerer et bilde til en array eller et bilde med tall-etiketter, som beskriver intensitetsmønstre rundt pixler [68]. LBP har vist seg å være et sterkt verktøy for å trekke ut egenskaper av 2D-bilder, men det finnes ikke en god implementasjon for 3D-bilder i Python. Dette ser ut til å være på vei inn i PyRadiomics [49].

I tillegg kan metoden Angle Measure Technique (AMT) brukes for å kvantifisere kompleksiteten av et bilde, men denne metoden er ikke utviklet for 3D-bilder [69] [9]. Andre type filtre kan også testes ut for bildetransformasjoner, slik at andre egenskaper fremheves. For eksempel finnes flere typer Wavelet-filtre enn Coiflets1, som ble brukt i denne oppgaven. I PyRadiomics kan man velge mellom syv forskjellige familier med bølger (wavelets) (haar, discrete meyer, daubechies, coiflets, biorthogonal og reverse biorthogonal), med flere varianter av disse igjen [49].

Som nevnt i avsnitt 5.4 ble ikke verdiene i kantene på tumor transformert riktig av filtre som ser på nabovoxlene. Bildetransformasjon burde dermed gjøres på hele bildene, som inkluderer områder utenfor tumor. Resultatene fra figur 4.11 viser at kantdeteksjonsfiltre bør undersøkes nærmere, da prediksjon basert på disse egenskapene gav relativ god ytelse, når ikke formegenskapene var inkludert.

Det finnes prosedyrer der to typer medisinske bilder kan fusjoneres til ett bilde, før egenskaper blir trukket ut. Ofte har dette vært fusjonering av PET- og MRI-bilder, hvor DWT (Diskrete Wavelet Transform) er en mye brukt metode for denne type fusjonering (eksempelvis Bindu og Prasad [70]). Det har også blitt brukt andre metoder som IHS (Intensity-Hue-Saturation) og RIM (Retina-Inspired Model), som f.eks. Daneshvar og Ghassemian [71]. Det kan hende at en slik type fusjonering kan være aktuelt for PET-bilder og CT-bilder også, for å få ut mer informasjon fra færre egenskaper.

Det ble observert at å bruke færre egenskaper, f.eks. kun fra en type filtertransformasjon, fungerte bedre enn å sette 2699 egenskaper sammen i et stor datasett. Dette tyder på at egenskapsutvelgelsesmetodene ikke nødvendigvis fungerte optimalt. Dette er et svært viktig skritt i utvikling av klassifiseringsmodeller, og bør undersøkes nærmere. Det er mulig at metodene kan optimaliseres, eller at andre metoder bør vurderes. Som eksempel går det an å lage en delmengde av det opprinnelige datasettet, før egenskapsuvelgelse blir gjort på denne delmengden. Dette kan være å ekskludere alle egenskaper med lav variasjon, eller ekskludere alle egenskaper med lav korrelasjon med behandlingsutfallet. Ettersom ReliefF ikke ser på avhengigheter mellom egenskaper, kunne det vært en mulighet å lage en delmengde med egenskaper, slik at ingen av egenskapene i datamengden har høy korrelasjon med hverandre.

En måte å dele opp et datasett, er ved bruk av SO-PLS (Sequential and Orthogonalised PLS Regression) [72]. Dette er en metode der datasettet deles opp i ortogonale blokker istedet for å samle all data i enn stor datamatrise. Blokkene kan legges til én etter én i analysen, for å se forandringen i forklart varians i responsen. Eksempler på hvordan dette datasettet kan deles opp i blokker, kan være kliniske faktorer i én blokk, PET-parametere i én blokk og CT-parametere i én blokk. Et annet alternativ er å legge bildeegenskaper trukket ut fra forskjellige filter-transformerte bilder i hver sine blokker, eller ha egenskaper fra de forskjellige egenskapsklassene i hver sine blokker.

Det er en utfordring å analysere ubalanserte datasett. Med responsen lokalregionalt tilbakefall mot progresjonsfri overlevelse, hadde 26% tilbakefall og 74% progresjonsfri overlevelse. Dette var tilstrekkelig balansert til at klassifikasjonsalgoritmene fungerte. I tillegg ble en balansert vekting ved bruk av $class\_weight = balanced$ brukt for noen algoritmer, slik som beskrevet i avsnitt 3.11. Dersom en annen type respons skal predikeres, som f.eks. metastase mot progresjonsfri overlevelse, må datasettet gjøres mer balansert. Det finnes flere metoder for å gjøre dette, f.eks. å kopiere nye observasjoner fra den minste klassen, redusere observasjoner fra den dominerende klassen, eller SMOTE (Syntethic Minority Over-Sampling Technique) [7].

Bootstrapping [22] er en annen metode for å estimere ytelsen til modeller, og er beskrevet i avsnitt 2.3.3. Dette kan være et alternativ til estimering av ytelse gjennom kryssvalidering. Det ble observert at AUC gitt fra de forskjellige foldene i kryssvalideringen av dette datasettet kunne variere veldig (mellom ca. 0.4 til ca. 0.85). Bootstrapping skal ofte gi lavere varians i estimering av ytelse enn det kryssvalidering gir [73][74], og kan da være et godt alternativ til kryssvalidering for dette datasettet.

Mot slutten av analysen ble blodverdier for pasientene gjort tilgjengelige. Disse kan være aktuelle å ta med blant kliniske faktorer. Dessverre var mesteparten av disse faktorene ufullstendige, der noen pasienter manglet data. Da må de manglende dataene estimeres. Scikit-Learn [24] har en funksjon *Imputer*, som estimerer manglende data ved å bruke gjennomsnittsverdien, medianverdien, eller verdien med flest tilfeller i en rad eller kolonne [28] [7].

Det kunne vært interessant å se på andre responser enn lokalregionalt tilbakefall mot progresjonsfri overlevelse, som f.eks. progresjonsfri overlevelse mot ikke overlevelse, eller metastase mot progresjonsfri overlevelse. Sistnevnte har ofte gitt høyere prediksjonsytelse i flere studier [64] [75].

Kapittel 6

Konklusjon

I denne oppgaven ble et bildeanalyse- og prediksjonsprogram utviklet i Python for å predikere lokalregionalt tilbakefall etter kreftbehandling for hode- og halskreft. Syv metoder for å trekke ut svulstegenskaper, syv metoder for å velge ut de mest relevante egenskapene, og fjorten klassifiseringsalgoritmer ble implementert i programmet. Resultatene i denne oppgaven tyder på at radiomics kan gi nyttig informasjon om behandlingsutfall utover kliniske egenskaper. Følgende resultater ble oppnådd.

Analysene viser at egenskaper trukket ut fra PET- og CT-bilder gav prediksjonsmodeller for behandlingsutfall med høyere ytelse ($AUC = 0.66$) enn modeller basert på kun kliniske faktorer ($AUC = 0.57$). Blandt disse egenskapene var det formegenskaper, som f.eks. størrelse og rundheten til tumor, som generelt gav modeller med høyest prediksjonsytelse ($AUC = 0.67$).

Univariat analyse viste at mange teksturegenskaper hadde en høyere korrelasjon med behandlingsutfallet enn kliniske faktorer. Eksempler på dette er teksturegenskaper som beskriver homogenitet i teksturen, intensitetsendringsrater og grov/fin tekstur i svulstene.

Kombinasjonen av egenskapsutvelgeren ReliefF, og enten PLSR, Logistisk regresjon, LDA eller AdaBoost som klassifikasjonsalgoritme, gav prediksjonsmodeller med høyest AUC. Generelt gav egenskapsutvelgelse ved bruk av ReliefF høyere ytelse i forhold til andre egenskapsutvelgere. Derimot hadde flere klassifikasjonsalgoritmer relativ høy og lik ytelse. Dette viser at ReliefF har et høyt potensiale, selv om den ikke er blandt de mest kjente og brukte egenskapsutvelgerne.

For å finne modellen med høyest prediksjonsytelse ble seks forskjellige intensitetskvantifiseringer på PET- og CT-bildene testet ut, samt syv forskjellige filtertransformasjoner (inkludert de opprinnelige bildene uten transformasjon). Svulstegenskaper trukket ut fra kvadratrotfiltrerte bilder, i kombinasjon med formegenskaper og kliniske faktorer, ble vurdert som beste datasett. Dette betyr ikke nødvendigvis at kvadratrotfilteret var det filteret som gav bildeegenskaper med høyest klassifikasjonsrelevans. Førsteordens egenskaper og teksturegenskaper, trukket ut fra Laplacian

of Gaussian-filtrerte og Wavelet LLL-filtrerte bilder, predikerte behandlingsutfallet med høyest ytelse, når formegenskaper til svulsten og kliniske faktorer var utelatt. Dette tyder på at bruk av filtre som bruker den romlige fordelingen til voxlene, er en fremgangsmåte som bør undersøkes nærmere.

Både univariate analyser og prediksjonsmodeller tydet på at det er en sammenheng mellom HPV-status og behandlingsutfallet for hode- og halskreft. Denne sammenhengen ble ikke observert når kreft i kun oropharynx ble undersøkt.

I denne oppgaven ble det oppnådd modeller for prediksjon av lokalregionalt tilbakefall med AUC på 0.66. Det er flere studier innenfor radiomics som har oppnådd høyere AUC for prediksjon av behandlingsutfall [60] [76] [6]. Dette indikerer at det er et potensiale for videreutvikling av modellene i denne oppgaven. Muligheter for modellforbedring inkluderer andre preprosesseringer eller transformeringer av PET- og CT-bilder, optimalisering av metoder for å trekke ut svulstegenskaper og revurdering av metoder for egenskapsutvelgelse.

Referanser

[1] Kreftforeningen. *Hva er kreft?* (lest 18.04.18). URL: https://kreftforeningen.no/om-kreft/hva-er-kreft/.

[2] Kreftforeningen. *Hode- og halskreft.* (lest 18.04.18). URL: https://kreftforeningen.no/om-kreft/kreftformer/hode-og-halskreft/.

[3] National Cancer Intitute. *Head and Neck Cancers.* (lest 08.05.2018). National Cancer Intitute. Mar. 2017. URL: https://www.cancer.gov/types/head-and-neck/head-neck-fact-sheet#q1.

[4] Kreftforeningen. *PET/CT.* (lest 08.05.2018). https://kreftforeningen.no/om-kreft/undersokelse-ved-kreft/pet-scan/, feb. 2018. URL: https://kreftforeningen.no/om-kreft/undersokelse-ved-kreft/pet-scan/.

[5] Robtert J. Gillies, Paul. E Kinahan og Hedvig Hricak. "Radiomics: Images Are More than Pictures, They Are Data". *Radiology* 278.2 (2016).

[6] E. J. Limkin et al. "Promises and challenges for the implementationof computational medical imaging (radiomics) inoncology". *Annals of Oncology* 28.6 (jun. 2017), s. 1191–1206.

[7] Sebastian Raschka og Vahid Mirajalili. *Python Machine Learning.* 2. utg. Packt: Birmingham, 2017.

[8] Martine Mulstad. "Assessment of a diagnostic program for autodelineation of head and neck cancer based on PET/CT images". Masteroppg. Ås: NMBU, 2017.

[9] Kari Helena Kvandal. "Eksplorativ analye av PET/CT-bilder av hode/halskreft med fokus på prediksjon av behandlingsutfall og HPV-status". Masteroppg. Ås: NMBU, 2017.

[10] Kristin Bøhle. *Mutasjoner.* (lest 18.04.2018). NDLA. 2010. URL: https://ndla.no/nb/node/5517?fag=7.

[11] Olbjørn Klepp. *Metastase.* (lest 18.04.16). Store norske leksikon. URL: https://sml.snl.no/metastase.

[12] Jan Folkvard Evensen. *Kreft i hode-halsregonen.* (lest 18.04.18). URL: http://oncolex.no/Hodehals.

[13] Helsebiblioteket. *Hudkreft (plateepitelkreft/spinocellulær kreft).* (lest 18.04.18). Helsebilbioteket. URL: http://www.helsebiblioteket.no/pasientinformasjon/kreft/hudkreft-plateepitelkreft-spinocellulaer-kreft.

[14] *HPV*. (lest 18.04.18). Kreftregisteret. URL: https://www.kreftregisteret.no/screening/livmorhalsprogrammet/ofte-stilte-sporsmal1/HPV1/.

[15] Michael E. Phelps. *PET - Physics, instrumentation and scanners*. Springer: New York, 2006.

[16] Suzanne Amador Kane. *Physics in modern medicine*. Red. av Suzanne Amador Kane. 2. utg. CRC Press, apr. 2009.

[17] Richard L. Wahl. *Principles and practice of PET and PET/CT*. 2. utg. Wolters Kluwer: Philadelphia, 2009.

[18] William W. Moses. "Fundamental Limits of Spatial Resolution in PET". *Nucl Instrum Methods Phys Res A* 648.Supplement 1 (aug. 2011), s. 236–240.

[19] ASCO. *Positron Emission Tomography and Computed Tomography (PET-CT) Scans*. (lest 08.05.0218). URL: https://www.cancer.net/navigating-cancer-care/diagnosing-cancer/tests-and-procedures/positron-emission-tomography-and-computed-tomography-pet-ct-scans.

[20] James W. Fletcher Paul E. Kinahan. "PET/CT Standardized Uptake Values (SUVs) in Clinical Practice and Assessing Response to Therapy". *Seminars in ultrasound, CT, and MR* 31.6 (des. 2010), s. 496–505.

[21] Stuart Price et al. *Pair production*. (lest 03.05.2018). Radiopedia. URL: https://radiopaedia.org/articles/pair-production.

[22] Trevor Hastie, Tibshirani Robert og Friedman Jerome. *The elements of statistical learning*. 2. utg. Springer: New York, 2001.

[23] J. J. M. van Griethuysen et al. "Computational Radiomics System to Decode the Radiographic Phenotype". *Cancer Research* 77.21 (2017), s. 104–107.

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

[25] Matias Carrasco Kind og Robert J. Brunner. "TPZ: photometric redshift PDFs and ancillary information by using prediction trees and random forests". *Monthly Notices of the Royal Astronomical Society* 432.2 (jun. 2013), s. 1482–1501.

[26] Max Kuhn og Kjell Johnson. *Applied Predictive Modeling*. Springer, 2013.

[27] Binbin Lu et al. "The Minkowski approach for choosing the distance metric in geographically weighted regression". *International Journal of Geographical Information Science* 30.2 (sep. 2015), s. 351–368.

[28] *Scikit-learn User Guide*. (lest 27.04.2018). URL: http://scikit-learn.org/stable/user_guide.html.

[29] Jason Rudy. *Py-earth documentation*. 2013. URL: http://contrib.scikit-learn.org/py-earth/index.html.

[30] Jerome H. Friedman. "Multivariate Adaptive Regression Splines". *The Annals of Statistics* 19.1 (1991), s. 1–67.

[31] Ryan J. Urbanowicz et al. "Relief-Based Feature Selection: Introduction and Review". *Journal of biomedical informatics* (2018).

[32] Marko Robnik-Sikonja og Igor Kononenko. "Theoretical and Empirical Analysis of ReliefF and RReliefF". *Machine Learning, 53, 23–69* 53.1-2 (okt. 2003), s. 23–69.

[33] Lorenzo Boretta og Alessandro Santaiello. "Implementing ReliefF filters to extract meaningful features from genetic lifetime datasets". *Journal of Biomedical Informatics* 44.2 (apr. 2011), s. 361–369.

[34] Randal S. Olson. *ReliefF feature selection algorithms*. (lest 08.05.2018). Mar. 2016. URL: https://pypi.python.org/pypi/ReliefF/0.1.2.

[35] Alexander Kraskov, Harald Stögbauer og Peter Grassberger. "Estimating mutual information". *PHYSICAL REVIEW E* 69.6 (jun. 2004), s. 066138.

[36] Gareth James et al. *An introduction to statistical learning with applications in R*. Red. av G. Casella, S. Fienberg og I. Olkin. Springer, 2013.

[37] The Python community. *Python offisiell nettside*. (lest 28.04.18). URL: https://www.python.org/.

[38] David Robinson. *The Incredible Growth of Python*. (lest 08.05.2018). Stack Overflow blog. Sep. 2017. URL: https://stackoverflow.blog/2017/09/06/incredible-growth-python/.

[39] Nich Heath. *What fuelled Python's rise to become the fastest-growing programming language*. (lest 28.04.2018). TechRepublic. Sep. 2017. URL: https://www.techrepublic.com/article/what-fuelled-pythons-rise-to-become-the-fastest-growing-programming-language/.

[40] David Robinson. *Why is Python Growing So Quickly?* (lest 28.04.2018). Stack Overflow blog. Sep. 2017. URL: https://stackoverflow.blog/2017/09/14/python-growing-quickly/.

[41] Aurora Rossvoll Grøndahl. "Forklaring Pasientinformasjon". Notatark laget etter personlig kommunikasjon (19.06.17 , 14.08.17, 28.08.17 og 08.09.17) med Eirik Malinen et al.

[42] *Elektronisk søkeverktøy for ICD-10, NCMP-NCSP, BUP, ICPC-2 og ICF-CY*. (lest 28.04.2018). Direkoratet for e-helse. 2018. URL: https://finnkode.ehelse.no/.

[43] *Strålebehandling ved kreft i svelg*. (lest 28.04.18). Insitutt for kreftgenestikk og informatikk - Oslo universitetssykehus. URL: http://kreftlex.no/Hodehals-svelgkreft/ProsedyreFolder/BEHANDLING/Stralebehandling/New-ksProcedureChapter.

[44] *Cisplatinkur*. (lest 28.04.18). Insitutt for kreftgenestikk og informatikk - Oslo universitetssykehus. URL: http://kreftlex.no/Bukhinnekreft/ProsedyreFolder/BEHANDLING/Cellegift/Gyn-Cisplatin?lg=ks&CancerType=Bukhinne&containsFaq=False.

[45] Institutt for matematikk UiO. *Medisinsk bildeanalyse*. (lest 08.05.2018). Feb. 2011. URL: http://www.mn.uio.no/ifi/forskning/grupper/dsb/hva/eksempler/medisinsk-bildeanalyse/.

[46] N. Aggarwal og R. K. Agrawal. "First and Second Order Statistics Features for Classification of Magnetic Resonance Brain Images". *Journal of Signal and Information Processing* 3.2 (2012), s. 146–153.

[47] *Texture Analysis Using the Gray-Level Co-Occurrence Matrix (GLCM)*. (lest 08.05.2018). MathWorks, mar. 2018. URL: https://se.mathworks.com/help/images/texture-analysis-using-the-gray-level-co-occurrence-matrix-glcm.html.

[48] Oranit Boonsiri et al. "3D Gray Level Co-Occurence Matrix Based Classificatin of Favor Benign and Borderline Types in Follicular Neoplasm Images". *Jornal of Biosciences and Medicines* 4 (2016), s. 5.

[49] PyRadiomics Community. *PyRadiomics Docmentation*. (lest 24.04.18). URL: http://pyradiomics.readthedocs.io.

[50] Guillaume Thibault et al. "Texture Indexes and Gray Level Size Zone Matrix. Application to Cell Nuclei Classification". *10th International Conference on Pattern Recognition and Information Processing* (nov. 2009), s. 140–145. URL: https://pdfs.semanticscholar.org/fec6/bd9b7f5d6a50410109991857494c8d25f290.pdf.

[51] Stephen Kokoska og Daniel Zwillinger. *CRC Standard Probability and Statistics Tables and Formulae*. Student Edition. Chapman & Hall: New York, 2000.

[52] R. Artusi, P. Verderio og E. Marubini. "Bravais-Pearson and Spearman correlation coefficients: meaning, test of hypothesis and confidence interval". *The International Journal of Biological Markers* 17.2 (2002), s. 148–151.

[53] Rajul Parikh et al. "Understanding and using sensitivity, specificity and predictive values". *Indian Journal of Ophthalmology* 56.1 (jan. 2008), s. 45–50.

[54] Rafael C. Gonzalez, Richard E. Woods og Steven L. Eddins. *Digital image processing using matlab*. Pearson Education: New Jersey, 2004.

[55] Filip Wasilewski. *Wavelet Browser*. (lest 08.05.2018). PyWavelets. URL: http://wavelets.pybytes.com/.

[56] Darshana Mistry. "Discrete wavelet transform using Matlab". *International journal of computer engineering and technology* 4.2 (2013), s. 255–259.

[57] A. C. Nichols et al. "The epidemic of human papillomavirus and oropharyngeal cancer in a Canadian population". *Current Oncology* 20.4 (aug. 2013), s. 212–2019.

[58] Anil K. Chaturvedi et al. "Human Papillomavirus and Rising Oropharyngeal Cancer Incidence in the United States". *Journal of Clinical Oncology* 29.32 (nov. 2011), s. 4294–4301.

[59] E Weiss og C. F. Hess. "The impact of gross tumor volume (GTV) and clinical target volume (CTV) definition on the total accuracy in radiotherapy theoretical aspects and practical experiences". *Strahlentherapie und Onkologie* 179.1 (2003), s. 21–33.

[60] Raier J. Klement et al. "Support Vector Machine-Based Prediction of Local Tumor Control After Stereotactic Body Radiation Therapy for Early-Stage Non-Small Cell Lung Cancer". *International Journal of Radiation Oncology biology physics* 88.3 (2013), s. 732–738.

[61] Shifeng Chen et al. "Investigation of the support vector machine algorithm to predict lung radiation-induced pneumonitis". *Medical Physics* 34.10 (2007), s. 3808–3814.

[62] M. Valliéres et al. "A radiomics model from joint FDG-PET and MRI texture features for the prediction of lung metastases in soft-tissue sarcomas of the extremities". *Physics in Medicine and Biology* 60.14 (jul. 2015), s. 5471–5496.

[63] Martin Carrier-Valliéres. "FDG-PET/MR Imaging for Prediction of Lung Metastases in Soft-Tissue Sarcomas of the Extremities by Texture Analysis and Wavelet Image Fusion". Masteroppg. Montreal: McGill University, 2012.

[64] Martin Vallières et al. "Radiomics strategies for risk assessment of tumour failure in head-and-neck cancer". *Scientific Reports* 7.10117 (aug. 2017).

[65] M. Bogowicz et al. "Comparison of PET and CT radiomics for prediction of local tumor control in head and neck squamous cell carcinoma". *Acta oncoliga* 56.11 (nov. 2017), s. 1531–1536.

[66] S. Elrefaey et al. "HPV in oropharyngeal cancer: the basics to know in clinical practice". *Acta Otorhinolaryngologica Italica* 34.5 (2014), s. 299–309.

[67] T. Dalianis. "Human papillomavirus (HPV) and oropharyngeal squamous cell carcinoma". *Presse medicale* 43 (2014), s. 429–434.

[68] M Pietikäinen et al. *Computer Vision Using Local Binary Patterns*. Springer-Verlag London, 2011.

[69] Jun Huang og Kim H. Esbensen. "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 (des. 2000), s. 1–19.

[70] Hima Bindu og K. Prasad. "MRI-PET Medical Image Fusion Technique by Combining Contourlet and Wavelet Transform". *Das V. (eds) Proceedings of the Third International Conference on Trends in Information, Telecommunication and Computing. Lecture Notes in Electrical Engineering* 150 (2013), s. 145–151.

[71] Sabalan Daneshvar og Hassan Ghassemian. "MRI and PET image fusion by combining IHS and retina-inspired models". *Information Fusion* 11.2 (apr. 2009), s. 114–123.

[72] T. Næs et al. "Path modelling by sequential PLS regression". *Journal of Chemometrics* 25.1 (des. 2010), s. 28–40.

[73] Bradley Efron og Robert Tibshirani. "Improvements on Cross-Validation: The 632+ Bootstrap Method". *Journal of the American Statistical Association* 92.438 (1997), s. 584–560.

[74] Ji-Hyun Kim. "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap". *Computional Statistics and Data Analysis* 53.11 (sep. 2009), s. 3735–3745.

[75] Anastasia Oikonomou et al. "Radiomics analysis at PET/CT contributes to prognosis of recurrence and survival in lung cancer treated with stereotactic body radiotherapy". *Scientific Reports* 5;8.1 (mar. 2018), s. 4003.

[76] Bin Zang et al. "Radiomic machine-learning classifiers for prognostic biomarkers of advanced nasopharyngeal carcinoma". *Cancer Letters* 403 (2017), s. 21–27.

[77] PyRadiomics Community. *Github PyRadiomics*. (lest 10.05.2018). URL: `https://github.com/Radiomics/pyradiomics`.

Vedlegg A

Bildeegenskaper

Tabeller med oversikt over alle bildeegenskapene brukt i denne oppgaven vises her. I alle tabellene og forklaringene refererer "bildettil bildets ROI. Alle definisjonene på egenskapene er tatt fra dokumentasjonen til PyRadiomics [49].

A.1 Førsteordens egenskaper

Følgende størrelser ble brukt i definisjonene til egenskapene i tabell A.1:

- **X** er en matrise med $\mathbf{N_p}$ antall voxler inkludert i bildet.
- **P(i)** er et førsteordens histogram med $\mathbf{N_g}$ antall diskrete intensitetsverdier.
- **p(i)** er det normaliserte førsteordens histogrammet: $\frac{P(i)}{N_p}$.
- $\bar{X}$ er gjennomsnittsverdien til intensitetsverdiene i bildet.

Tabell A.1: *En liste over førsteordens egenskaper brukt i denne oppgaven.*

Egenskap	Ligning
Energy Et mål på størrelsen av verdiene til voxlene et bilde.	$\sum_{i=1}^{N_p}(X(i))^2$
Entropy Spesifiserer usikkerheten/tilfeldigheten i bildet. ϵ er et vilkårlig, lite tall $\approx 2.2 * 10^{-16}$.	$-\sum_{i=1}^{N_p} p(i) log_2(p(i)+\epsilon)$
Minimum Den minste intensiteten i bildet.	$min(X)$

10th percentile 10. persentilen til bildet.	
90th percentile 90. persentilen til bildet.	
Maximum Den største intensiteten i bildet.	$max(X)$
Mean Gjennomsnittsverdien i bildet.	$\bar{X} = \frac{1}{N_p} \sum_{i=1}^{N_p} X(i)$
Median Medianverdien i bildet.	
Interquartile Range Forskjellen mellom 75. persentilen og 25. persentilen.	$P_{75} - P_{25}$
Range Spennet til intensitetsverdiene i bildet.	$max(X) - min(X)$
Mean Absolute Deviation (MAD) Gjennomsnittlig avstand fra $\bar{X}$.	$\frac{1}{N_p} \sum_{i=1}^{N_p} \lvert X(i) - \overline{X} \rvert$
Robust Mean Absolute Deviation (rMAD) Gjennomsnittsavstanden fra $\bar{X}$ regnet ut fra en delmengde av verdiene i bildet, hvor verdiene var mellom 10. og 90. persentilene.	$\frac{1}{N_{10-90}} \sum_{i=1}^{N_{10-90}} \lvert X_{10-90}(i) - \overline{X}_{10-90} \rvert$
Root Mean Squared (RMS) En måte å måle størrelsen på intensitetsverdiene i bildet.	$\sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} (X(i) + c)^2}$
Skewness Et mål på asymmetrien til distribusjonen av intensitetsverdier rundt $\bar{X}$. En positiv verdi tilsier at histogrammet til bildet heller mot venstre, mens en negativ verdi tilsier at histogrammet heller mot høyre.	$\frac{\frac{1}{N_p} \sum_{i=1}^{N_p} (X(i) - \overline{X})^3}{(\sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} (X(i) - \overline{X})^2})^3}$
Kurtosis Et mål på hvor spiss distribusjonen av intensitetsverdier er. Høye kurtosis-verdier tilsier at distribusjonen er konsentrert mer rundt "halenetil et histogram enn rundt $\bar{X}$.	$\frac{\frac{1}{N_p} \sum_{i=1}^{N_p} (X(i) - \overline{X})^4}{(\sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} (X(i) - \overline{X})^2})^2}$

A.2 Formegenskaper

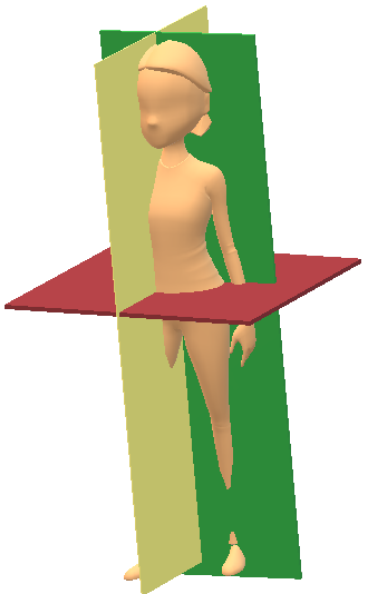
Følgende størrelser ble brukt i definisjonene til egenskapene i tabell A.1:

- V er volumet til ROI i mm^3.
- A er overflatearealet til ROI i mm^2.

Tabell A.2: *En liste over formegenskapene brukt i denne oppgaven.*

Feature	**Ligning**
Volume Volumet til ROI, som i dette tilfellet tilsvarer antall voxler i ROI, ettersom volumet til én voxel er 1 mm^3	$V = \sum_{i=1}^{N_p} V_i$
Surface Area En approksimasjon på overfaltearealet til ROI. N er antall trekanter som danner overflaten til ROI. a_ib_i og a_ic_i er hjørner i trekant nummer i.	$A = \sum_{i=1}^{N} \frac{1}{2}\lvert a_ib_i \times a_ic_i \rvert$
Surface Area to Volume ratio Forholdet mellom A og V. En lavere verdi indikerer en mer kule-lik form.	$\frac{A}{V}$
Sphericity Et tall mellom 0 og 1 som indikerer hvor rund formen til ROI er. En verdi på 1 tilsier en perfekt sfære.	$\frac{\sqrt[3]{3\pi V^2}}{A}$
Maximum 3D diameter Den største (euklidiske) avstanden mellom overflate-voxler i ROI.	
Maximum 2D diameter (Slice) Den største (euklidiske) avstanden mellom overflate-voxler i planet spent ut av radene og kolonnene. Dette er det røde planet i figur A.1	
Maximum 2D diameter (Column) Den største (euklidiske) avstanden mellom overflate-voxler i planet spent ut av radene og tverrsnittet. Dette er det grønne planet i figur A.1	
Maximum 2D diameter (Row)	

Den største (euklidiske) avstanden mellom overflate-voxler i planet spent ut av kolonnene og tverrsnittet. Dette er det gule planet i figur A.1	
Major Axis λ_{major} er lengden til den lengste aksen i ROI.	$4\sqrt{\lambda_{major}}$
Minor Axis λ_{minor} er lengden til den nest største aksen i ROI, som er ortogonal på den største aksen.	$4\sqrt{\lambda_{minor}}$
Least Axis λ_{least} er lengden til den minste aksen u ROI.	$4\sqrt{\lambda_{least}}$
Elongation En verdi mellom 0 og 1, der 1 gir at tverrsnittet gjennom de to største, men ortogonale aksene, er tilnærmet sirkulær. 0 gir at objektet er et punkt eller en linje.	$\sqrt{\frac{\lambda_{minor}}{\lambda_{major}}}$
Flatness En verdi mellom 0 og 1, der 1 gir et sirkulært objekt og 0 gir et flatt objekt.	$\sqrt{\frac{\lambda_{least}}{\lambda_{major}}}$



Figur A.1: *De forskjellige planene som diameterne til ROI er kalkulert på. Det røde planet er planet spent ut av radene og kolonnene, det grønne planet er planet spent ut av radene og tverrsnittet og de gule planet er planet spent ut av kolonnene og tverrsnittet.*

A.3 Gray Level Co-occurence Matrix

Følgende størrelser ble brukt i definisjonene til egenskapene i tabell A.3:

- ϵ er et vilkårlig, lite tall $\approx 2.2 * 10^{-16}$.
- $\mathbf{P(i,j)}$ er GLC-matrisen for en tilfeldig ϵ og θ.
- $\mathbf{p(i,j)}$ er den normaliserte GLC-matrisen lik $\frac{P(i,j}{\sum P(i,j)}$.
- $\mathbf{N_g}$ er antallet intensitetsnivåer i bildet.
- $\mathbf{p_x(i)} = \sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p(i,j)}$ er sannsynligheten for hver rad.
- $\mathbf{p_y(j)} = \sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p(i,j)}$ er sannsynligheten for hver kolonne.
- $\mu_\mathbf{x}$ er gjennomsnitts-intensiteten til $\mathbf{p_x}$ definert som $\sum_{i=1}^{N_g} p_x(i)i$.
- $\mu_\mathbf{y}$ er gjennomsnitts-intensiteten til $\mathbf{p_y}$ definert som $\sum_{j=1}^{N_g} p_y(j)j$.
- $\sigma_\mathbf{x}$ er standardavviket til p_x.
- $\sigma_\mathbf{y}$ er standardavviket til p_y.
- $\mathbf{p_{x+y}(k)} = \sum_{\mathbf{i=1}}^{\mathbf{N_g}} \sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p(i,j)}$ hvor i + j = k, og k = 2,3,...,$2N_g$.
- $\mathbf{p_{x-y}(k)} = \sum_{\mathbf{i=1}}^{\mathbf{N_g}} \sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p(i,j)}$ hvor $|i-j| = k$, og k = 0,1,...,$N_g - 1$.
- $\mathbf{HX} = -\sum_{\mathbf{i=1}}^{\mathbf{N_g}} \mathbf{p_x(i)} \log_\mathbf{2} \left(\mathbf{p_x(i)} + \epsilon\right)$ er entropien til p_x.
- $\mathbf{HY} = -\sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p_y(j)} \log_\mathbf{2} \left(\mathbf{p_y(j)} + \epsilon\right)$ er entropien til p_y.
- $\mathbf{HXY} = -\sum_{\mathbf{i=1}}^{\mathbf{N_g}} \sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p(i,j)} \log_\mathbf{2} \left(\mathbf{p(i,j)} + \epsilon\right)$ er entropien til p(i,j).
- $\mathbf{HXY1} = -\sum_{\mathbf{i=1}}^{\mathbf{N_g}} \sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p(i,j)} \log_\mathbf{2} \left(\mathbf{p_x(i)p_y(j)} + \epsilon\right)$.
- $\mathbf{HXY2} = -\sum_{\mathbf{i=1}}^{\mathbf{N_g}} \sum_{\mathbf{j=1}}^{\mathbf{N_g}} \mathbf{p_x(i)p_y(j)} \log_\mathbf{2} \left(\mathbf{p_x(i)p_y(j)} + \epsilon\right)$.

Tabell A.3: *En liste over egenskaper fra GLCM brukt i oppgaven.*

Egenskap	Ligning
Autocorrelation Et mål på finheten og grovheten i teksturen.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)ij$
Joint average Gjennomsnittsverdien til fordeling i.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)i$
Cluster Prominence Et mål på asymmetrien og skjevheten til GLC-matrisen. En høyere verdi tilsier mer asymmetri om gjennomsnittsverdien, mens en lavere verdi indikerer en topp nær gjennomsnittet og mindre variasjon.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i+j-\mu_x-\mu_y)^4 p(i,j)$
Cluster Shade Et mål på skewness og ensartetheten for GLC-matrisen. En høyerer verdi indikerer større asymmetri rundt gjennomsnittet.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i+j-\mu_x-\mu_y)^3 p(i,j)$
Cluster Tendency Et mål på grupperingen av voxler med lignende intensitetsnivåer.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i+j-\mu_x-\mu_y)^2 p(i,j)$
Contrast Et mål på den lokale intensitetsvariasjonen, som vektlegger verdier som ikke er på diagonalen i GLC-matrisen. En større verdi tilsier en større forskjell i intensitetsverdier for nabovoxler.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i-j)^2 p(i,j)$
Correlation Korrelasjonen mellom intensitetsverdier og deres respektive voxler i GLC-matrisen.	$\frac{\sum_{i=1}^{Ng}\sum_{j=1}^{Ng} p(i,j)-\mu_x\mu_y}{\sigma_x(i)\sigma_y(j)}$
Difference average Et mål på forholdet mellom forekomster av par med like intensitetsverdier og forekomster av par med forskjellige intensitetsverdier.	$\sum_{k=0}^{N_g-1} k p_{x-y}(k)$
Difference Entropy Et mål på tilfeldigheten i intensitetsforskjeller i nabolag.	$\sum_{k=0}^{N_g-1} p_{x-y} log_2(p_{x-y}(k)+\epsilon$
Difference Variance	$\sum_{k=0}^{N_g-1}(k-DA)^2 p_{x-y}$

Et mål på heterogenitet som setter høyere vekter på par med forskjellige intensitetsverdier som avviker fra gjennomsnittet. Her er DA Difference average.	
Joint Energy Et mål på homogene mønstre i bildet. En høyere energi tilsier at det er flere forekomster av intensitetsverdipar som dukker opp med høyere frekvenser.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(p(i,j))^2$
Joint Entropy Et mål på tilfeldigheten i intensitetsverdiene til nabovoxler.	$-\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(p(i,j))log_2(p(i,j)+\epsilon)$
Informal Measure of Correlation (IMC) 1 Et uformelt mål for korrelasjon.	$\frac{HXY-HXY1}{max\{HX,HY\}}$
Informal Measure of Correlation (IMC) 2 Et uformelt mål for korrelasjon	$\sqrt{1-e^{-2(HXY2-HXY)}}$
Inverse Difference Moment (IDM) Et mål på lokal homogenitet i et bilde som vektlegger verdier nærmere diagonalen i GLC-matrisen.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{p(i,j)}{1+\lvert i-j\rvert^2}$
Inverse Difference Moment Normalized (IDMN) En normalisert IDM.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{p(i,j)}{1+\frac{\lvert i-j\rvert^2}{N_g^2}}$
Inverse Difference (ID) Et annet mål på lokal homogenitet i bildet.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{p(i,j)}{1+\lvert i-j\rvert}$
Inverse Difference Normalized (IDN) En normalisert ID	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{p(i,j)}{1+\frac{\lvert i-j\rvert}{N_g}}$
Inverse Variance Et mål på lokal homogenitet, som ikke tar med diagonalen i GLC-matrisen.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{p(i,j)}{\lvert i-j\rvert^2}, i \neq j$
Maximun Probability Antall forekomster av det mest dominerende paret av intensitetsverdier.	$max(p(i,j))$
Sum Average Forholdet mellom forekomster av par med lave intensiteter og forekomster av par med høye intensiteter.	$\sum_{k=2}^{2N_g}p_{x+y}(k)k$
Sum Entropy	$\sum_{k=2}^{2N_g}p_{x+y}log_2(p_{x+y}(k)+\epsilon)$

Sum av intensitetsforskjellene i nabolagene.	
Sum of Squares Et mål på fordelingen av intensitetsverdipar rundt gjennomsnittsintensitetsverdien i GLC-matrisen.	$-\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i-\mu_x)^2 p(i,j)$

A.4 Gray Level Size Zone Matrix

Følgende størrelser ble brukt i definisjonene til egenskapene i tabell A.4:

- $\mathbf{N_g}$ er antallet intensitetsverdier i bildet.
- $\mathbf{N_s}$ er antallet sonestørrelser i bildet.
- $\mathbf{N_p}$ er antallet voxler i bildet.
- $\mathbf{N_z}$ er antall soner i bildet, som er lik $\sum_{i=1}^{N_g}\sum_{j=1}^{N_s} P(i,j)$ og $1 \leq N_z \leq N_p$.
- $\mathbf{P(i,j)}$ er GLSZ-matrisen.
- $\mathbf{p(i,j)}$ er den normaliserte GLSZ-matrisen, definert som $\frac{P(i,j)}{N_z}$.

Tabell A.4: *En liste over egenskaper fra GLSZM brukt i oppgaven.*

Feature	**Ligning**
Small Area Emphasis Et mål på fordelingen av små størrelses-soner, der en større verdi indikerer flere små størrelses-soner og finere teksturer.	$\frac{\sum_{i=1}^{Ng}\sum_{j=1}^{Ns}\frac{P(i,j)}{j^2}}{N_Z}$
Large Area Emphasis (SAE) Et mål på fordelingen av store størrelses-soner, der en større verdi indikerer flere store størrelses-soner og grovere teksturer.	$\frac{\sum_{i=1}^{Ng}\sum_{j=1}^{Ns}P(i,j)j^2}{N_Z}$
Gray Level Non-Uniformity (GLN) Et mål på variabiliteten av intensitetsverdier i bildet, der en lavere verdi indikerer mer homogenitet i intensitetsverdier.	$\frac{\sum_{i=1}^{Ng}\left(\sum_{j=1}^{Ns}P(i,j)\right)^2}{N_z}$
Gray Level Non-Uniformity Normalized (GLNN)	$\frac{\sum_{i=1}^{Ng}\left(\sum_{j=1}^{Ns}P(i,j)\right)^2}{N_z^2}$

En normalisert GLN	
Size-Zone Non-Uniformity (SZN) Et mål på variabiliteten i volumene på størrelses-sonene i bildet, der en lavere verdi indikerer mer homogenitet i volumene.	$\frac{\sum_{j=1}^{N_s}\left(\sum_{i=1}^{N_g} P(i,j)\right)^2}{N_z}$
Size-Zone Non-Uniformity Normalized (SZNN) En normalisert SZN.	$\frac{\sum_{j=1}^{N_s}\left(\sum_{i=1}^{N_g} P(i,j)\right)^2}{N_z^2}$
Zone Percentage (ZP) Et mål på grovheten av teksturen, som tar forholdet mellom antall soner og antall voxler i bildet.	$\frac{N_z}{N_p}$
Gray Level Variance (GLV) Et mål på variansen av intensiteter for sonene.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_s} p(i,j)(i-\mu)^2$
Zone Variance (ZV) Et mål på variansen i sone-størrelser for sonene.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_s} p(i,j)(j-\mu)^2$
Zone Entropy (ZE) Her er ϵ et tilfeldig, lite tall ($\approx 2.2 \times 10^{-16}$). Et mål på tilfeldigheten i fordelingen av størrelses-soner og intensiteter. Et høyere tall indikerer mer heterogenitet i teksturen.	$-\sum_{i=1}^{N_g}\sum_{j=1}^{N_s} p(i,j)\log_2(p(i,j)+\epsilon)$
Low Gray Level Zone Emphasis (LGLZE) Et mål på fordelingen av lavere intensitets-størrelses-soner. En høyere verdi indikerer en høyere andel lavere intensitetsverdier og størrelses-soner i bildet.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_s}\frac{P(i,j)}{i^2}}{N_z}$
High Gray Level Zone Emphasis (HGLZE) Et mål på fordelingen av høyere intensitets-størrelses-soner. En høyere verdi indikerer en høyere andel høyere intensitetsverdier og størrelses-soner i bildet.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_s} P(i,j)i^2}{N_z}$
Small Area Low Gray Level Emphasis (SALGLE) Andelen av den felles fordelingen av mindre størrelses-soner med små intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_s}\frac{P(i,j)}{i^2 j^2}}{N_z}$

Small Area High Gray Level Emphasis (SAHGLE) Andelen av den felles fordelingen av mindre størrelses-soner med høye intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_s}\frac{P(i,j)i^2}{j^2}}{N_z}$
Large Area Low Gray Level Emphasis (LALGLE) Andelen av den felles fordelingen av større størrelses-soner med små intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_s}\frac{P(i,j)j^2}{i^2}}{N_z}$
Large Area High Gray Level Emphasis (LAHGLE) Andelen av den felles fordelingen av større størrelses-soner med høye intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_s}P(i,j)i^2j^2}{N_z}$

A.5 Gray Level Run Length Matrix

Følgende størrelser ble brukt i definisjonene til egenskapene i tabell A.5:

- $\mathbf{N_g}$ er antallet intensitetsverdier i bildet.
- $\mathbf{N_r}$ er antallet intensitetsrekker i bildet.
- $\mathbf{N_p}$ er antallet voxler i bildet.
- $\mathbf{N_z}(\theta)$ er antallet intensitetsrekker i bildet langs aksen θ, lik $\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\mathbf{P}(i,j|\theta)$ og $1 \leq N_z(\theta) \leq N_p$.
- $\mathbf{P}(\mathbf{i},\mathbf{j}|\theta)$ er GLRL-matrisen for en vilkårlig retning θ.
- $\mathbf{p}(\mathbf{i},\mathbf{j}|\theta)$ er den normaliserte GLRL-matrisen, definert som $\frac{P(i,j|\theta).}{N_z(\theta))}$

Tabell A.5: *En liste over egenskaper fra GLRLM brukt i oppgaven.*

Egenskap	Ligning
Short Run Emphasis (SRE) Et mål på fordelingen av korte rekker, der en større verdi indikerer kortere rekker og finere teksturer.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\frac{P(i,j|\theta)}{j^2}}{N_z(\theta)}$
Long Run Emphasis (LRE) Et mål på fordelingen av lange rekker, der en større verdi indikerer lengre rekker og grovere teksturer.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}P(i,j|\theta)j^2}{N_z(\theta)}$

Gray Level Non-Uniformity (GLN) Et mål på likheten av intensitetsverdier i bildet, der en lavere verdi indikerer tilsier større likhet i intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\left(\sum_{j=1}^{N_r} P(i,j\|\theta)\right)^2}{N_z(\theta)}$
Gray Level Non-Uniformity Normalized (GLNN) En normalisert GLN.	$\frac{\sum_{i=1}^{N_g}\left(\sum_{j=1}^{N_r} P(i,j\|\theta)\right)^2}{N_z(\theta)^2}$
Run Length Non-Uniformity (RLN) Et mål på likheten av rekkelengder i bildet, der en lavere verdi indikerer mer homogenitet i rekkelengdene.	$\frac{\sum_{j=1}^{N_r}\left(\sum_{i=1}^{N_g} P(i,j\|\theta)\right)^2}{N_z(\theta)}$
Run Length Non-Uniformity Normalized (RLNN) En normalisert RLN.	$\frac{\sum_{j=1}^{N_r}\left(\sum_{i=1}^{N_g} P(i,j\|\theta)\right)^2}{N_z(\theta)^2}$
Run Percentage (RP) Et mål på grovheten i teksturen, ved å ta forholdet mellom antallet rekker og antallet voxler i bildet. En høyere verdi indikerer at en større andel av bildet består av korte rekker (finere tekstur).	$\frac{N_z(\theta)}{N_p}$
Gray Level Variance (GLV) Variansen i intensitetsverdi for rekkene.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_r} p(i,j\|\theta)(i-\mu)^2$
Run Variance (RV) Variansen i antall rekker for rekkelengdene.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_r} p(i,j\|\theta)(j-\mu)^2$
Run Entropy (RE) Her er ϵ et vilkårlig, lite tall ($\approx 2.2 \times 10^{-16}$). Et mål på tilfeldigheten i fordelingen av rekkelengder og intensitetsverdier. En høyere verdi indikerer mer heterogenitet i teksturen.	$-\sum_{i=1}^{N_g}\sum_{j=1}^{N_r} p(i,j\|\theta)\log_2(p(i,j\|\theta)+\epsilon)$
Low Gray Level Run Emphasis (LGLRE) Et mål på fordelingen av lave intensitetsverdier, der en høyere verdi indikerer en større konsentrasjon av lave intensitetsverdier i bildet.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\frac{P(i,j\|\theta)}{i^2}}{N_z(\theta)}$
High Gray Level Run Emphasis (HGLRE)	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r} P(i,j\|\theta)i^2}{N_z(\theta)}$

Et mål på fordelingen av høye intensitetsverdier, der en høyere verdi indikerer en større konsentrasjon av høye intensitetsverdier i bildet.	
Short Run Low Gray Level Emphasis (SRLGLE) Et mål på den felles distribusjonen av korte rekker med lave intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\frac{P(i,j\|\theta)}{i^2j^2}}{N_z(\theta)}$
Short Run High Gray Level Emphasis (SRHGLE)g Et mål på den felles distribusjonen av korte rekker med høye intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\frac{P(i,j\|\theta)i^2}{j^2}}{N_z(\theta)}$
Long Run Low Gray Level Emphasis (LRLGLE) Et mål på den felles distribusjonen av lange rekker med høye intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\frac{P(i,j\|\theta)j^2}{i^2}}{N_z(\theta)}$
Long Run High Gray Level Emphasis (LRHGLE) Et mål på den felles distribusjonen av lange rekker med lave intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}P(i,j\|\theta)i^2j^2}{N_z(\theta)}$

A.6 Neighbouring Gray Tone Difference Matrix

Følgende størrelser ble brukt i definisjonene til egenskapene i tabell A.6:

- $\mathbf{n_i}$ er antallet voxler i X_{gl} med intensitetsverdi i.
- $\mathbf{N_p}$ er antallet voxler i bildet.
- $\mathbf{p_i}$ er intensitetsverdi-sannsynligheten, $\frac{n_i}{N_v}$.
- $\mathbf{s_i}$ er summen av de absolutte forskjellene for intensitetsverdi i,
  $s_i = \begin{cases} \sum^{n_i} |i - \bar{A}_i| & \text{for} \quad n_i \neq 0 \\ 0 & \text{for} \quad n_i = 0 \end{cases}$
- $\mathbf{N_g}$ er antallet intensitetsverdier.
- $\mathbf{N_{g,p}}$ er antallet intensitetsverdier der $p_i \neq 0$.

Tabell A.6: *En liste over egenskaper fra NGTDM brukt i oppgaven.*

Egenskap	**Ligning**
Coarseness Et mål på gjennomsnittsforskjellen mellom sentervoxelen og dens nabolag, og indikerer den romlige endringsraten.	$\frac{1}{\sum_{i=1}^{Ng} p_i s_i}$
Contrast Et mål på den romlige intensitetsendringen, men er også avhengig av det totale spennet med intensitetsverdier. En høyere verdi indikerer et bilde med et stort utvalg av intensitetsverdier og store endringer mellom voxler og nabolag.	$\left(\frac{1}{N_{g,p}(N_{g,p}-1)} \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p_i p_j (i-j)^2\right)\left(\frac{1}{N_p}\sum_{i=1}^{N_g} s_i\right)$
Busyness Et mål på forandringen fra en voxel til dens nabolag. En høyere verdi indikerer et bilde med raske intensitetsendringer mellom voxel og nabolag.	$\frac{\sum_{i=1}^{Ng} p_i s_i}{\sum_{i=1}^{Ng}\sum_{j=1}^{Ng} \lvert i p_i - j p_j\rvert}$
Complexity Et mål på uensartetheten til et bilde. Et bilde blir vurdert som kompleks hvis det er raske endringer i intensitetsverdiene til voxlene.	$\frac{1}{N_p}\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} \lvert i-j\rvert \frac{p_i s_i + p_j s_j}{p_i + p_j}$
Strength Et mål på sakte endringer i bildet og større, grove forskjeller mellom intensitetsverdiene.	$\frac{\sum_{i=1}^{Ng}\sum_{j=1}^{Ng}(p_i+p_j)(i-j)^2}{\sum_{i=1}^{Ng} s_i}$

A.7 Gray Level Dependence Matrix

Følgende størrelser ble brukt i definisjonene til egenskapene i tabell A.7:

- $\mathbf{N_g}$ er antallet intensitetsverdier i bildet.
- $\mathbf{N_d}$ er antallet avhengigshetsstørrelser i bildet.
- $\mathbf{N_z}$ er antallet avhengigshetssoner i bildet, $\sum_{i=1}^{N_g}\sum_{j=1}^{N_d} \mathbf{P}(i,j)$.
- $\mathbf{P(i,j)}$ er avhengighetsmatrisen.
- $\mathbf{p(i,j)}$ er den normaliserte avhengighetsmatrisen, $\frac{\mathbf{P}(i,j)}{N_z}$.

Tabell A.7: *En liste over egenskaper fra GLDM brukt i oppgaven.*

Egenskap	Ligning
Small Dependence Emphasis (SDE) Et mål på fordelingen av små avhengigheter, der en høyere verdi indikerer mindre homogene teksturer.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{P(i,j)}{i^2}}{N_z}$
Large Dependence Emphasis (LDE) Et mål på fordelingen av store avhengigheter, der en høyere verdi indikerer mer homogene teksturer.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}P(i,j)j^2}{N_z}$
Gray Level Non-Uniformity (GLN) Et mål på likheten av intensitetsverdier i bildet, der en lavere verdi indikerer større likhet i intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\left(\sum_{j=1}^{N_d}P(i,j)\right)^2}{N_z}$
Dependence Non-Uniformity (DN) Et mål på avhengigheten gjennom hele bildet, der en lavere verdi indikerer mer homogenitet i avhengighetene i bildet.	$\frac{\sum_{j=1}^{N_d}\left(\sum_{i=1}^{N_g}P(i,j)\right)^2}{N_z}$
Dependence Non-Uniformity Normalized (DNN) En normalisert DN	$\frac{\sum_{j=1}^{N_d}\left(\sum_{i=1}^{N_g}P(i,j)\right)^2}{N_z^2}$
Gray Level Variance (GLV) Variansen i intensitetsverdier i bildet.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}p(i,j)(i-\mu)^2$
Dependence Variance (DV) Variansen i avhengighetsstørrelser i bildet.	$\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}p(i,j)(j-\mu)^2$
Dependence Entropy (DE) Et mål på tilfeldigheten i avhengighetsstørrelsene. En høyere verdi indikerer mer heterogenitet i teksturen.	$-\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}p(i,j)\log_2(p(i,j)+\epsilon)$
Low Gray Level Emphasis (LGLE) Et mål på fordelingen av lave intensitetsverdier, der en høyere verdi indikerer en større konsentrasjon av lave intensitetsverdier i bildet.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{P(i,j)}{i^2}}{N_z}$
High Gray Level Emphasis (HGLE)	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}P(i,j)i^2}{N_z}$

Et mål på fordelingen av høye intensitetsverdier, der en høyere verdi indikerer en større konsentrasjon av høye intensitetsverdier i bildet.	
Small Dependence Low Gray Level Emphasis (SDLGLE) Et mål på den felles fordelingen av små avhengigheter med lavere intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{P(i,j)}{i^2j^2}}{N_z}$
Small Dependence High Gray Level Emphasis (SDHGLE) Et mål på den felles fordelingen av små avhengigheter med høyere intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{P(i,j)i^2}{j^2}}{N_z}$
Large Dependence Low Gray Level Emphasis (LDLGLE) Et mål på den felles fordelingen av store avhengigheter med lavere intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{P(i,j)j^2}{i^2}}{N_z}$
Large Dependence High Gray Level Emphasis (LDHGLE) Et mål på den felles fordelingen av store avhengigheter med høyere intensitetsverdier.	$\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}P(i,j)i^2j^2}{N_z}$

Vedlegg B

Python-program

B.1 Endre bildene fra Matlab-filer til nrrd-filer. Lage masker og lage input-filen til PyRadiomics

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 23 2018

@author: Alise Midtfjord
"""

#A script to convert the images to nrrd-files, and create masks for the images.
"-----------------------------------------"

import numpy as np
import pandas as pd
import nrrd
import scipy.io as sio
import os
import os.path

"""-----------PET-images-----------"""

#Creating the table and setting the directory
length = np.zeros(shape = (255,3))
tabell = pd.DataFrame(data = length, columns = ['pasientID', 'Image', 'Mask'])
directory = os.getcwd()
j = 0

#Loop over all images in the folder
for i in range(2,255):

    #Need this because the names have two, one or zero zeroes before the number
    if i < 10:

        #Insert the name of the pictures in the folder, from matlab
```

```
        filnavn = 'PETstackP00' + str(i) + '.mat'
        kortnavn = 'PETstackP00' + str(i)
        masknavn = '\PETstackP00' + str(i) + 'mask.nrrd'
    elif i < 100:
        filnavn = 'PETstackP0' + str(i) + '.mat'
        kortnavn = 'PETstackP0' + str(i)
        masknavn = '\PETstackP0' + str(i) + 'mask.nrrd'
    else:
        filnavn = 'PETstackP' + str(i) + '.mat'
        kortnavn = 'PETstackP' + str(i)
        masknavn = '\PETstackP' + str(i) + 'mask.nrrd'
    if os.path.isfile(filnavn):
        matlab = sio.loadmat(filnavn)
        bilde = matlab[kortnavn]

        #Create the image as a nrrd-file
        nrrd.write(kortnavn + '.nrrd', bilde)
        maske = bilde

        #Create the mask
        maske[np.isnan(maske)] = 0
        maske[maske != 0] = 1
        maske = maske.astype(int)
        nrrd.write(kortnavn + 'mask.nrrd', maske)

        #Setting the values of the table
        tabell.iloc[j,0] = i
        tabell.iloc[j,1] = directory +  kortnavn
        j += 1

    print(i)

#Make a excel-file with the paths.
tabell.to_excel('paths.xlsx')

"""-----------CT-images, with the same mask-----------"""

length = np.zeros(shape = (255,3))
tabell = pd.DataFrame(data = length, columns = ['pasientID', 'Image', 'Mask'])
directory = os.getcwd() #
j = 0

for i in range(2,255):
    if i < 10:
        filnavn = 'CTstackP00' + str(i) + '.mat'
        kortnavn = 'CTstackP00' + str(i)
        masknavn = '\PETstackP00' + str(i) + 'mask.nrrd'
    elif i < 100:
        filnavn = 'CTstackP0' + str(i) + '.mat'
        kortnavn = 'CTstackP0' + str(i)
        masknavn = '\PETstackP0' + str(i) + 'mask.nrrd'
    else:
        filnavn = 'CTstackP' + str(i) + '.mat'
        kortnavn = 'CTstackP' + str(i)
        masknavn = '\PETstackP' + str(i) + 'mask.nrrd'
    if os.path.isfile(filnavn):
        matlab = sio.loadmat(filnavn)
        bilde = matlab[kortnavn]
```

```
        nrrd.write(kortnavn + '.nrrd', bilde)

        tabell.iloc[j,0] = i
        tabell.iloc[j,1] = directory +  kortnavn
        tabell.iloc[j,2] = directory +  masknavn
        j += 1

    print(i)

tabell.to_excel('paths_CT.xlsx')
```

B.2 Trekke ut egenskaper med PyRadiomics.

Kjernen til koden er tatt fra et PyRadiomics-eksempel [77].

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 23 2018

@author: Alise Midtfjord
"""

#Extract features from a batch of images using PyRadiomics.
"-----------------------------------------------"

import os
from radiomics import featureextractor
import numpy as np
import pandas
import logging
import radiomics

def pyradiomics_batch(input_file_name, output_file_name):
    '''
    Extract features from a batch of images using radiomics.

    :param str input_file_name: Name of the excel-document that contains patient ID, path
                                the image and path to the mask of the image
    :param str output_file_name: Name of the excel-document that is going to
                                 containing the extracted features
    '''

    #Finding the directory and setting up a logging file
    outPath = os.getcwd()
    inputXLSX = os.path.join(outPath, input_file_name)
    outputFilepath = os.path.join(outPath, output_file_name)
    progress_filename = os.path.join(outPath, 'pyrad_log.txt')
    rLogger = logging.getLogger('radiomics')
    handler = logging.FileHandler(filename=progress_filename, mode='w')
    handler.setFormatter(logging.Formatter('%(levelname)s:%(name)s: %(message)s'))
    rLogger.addHandler(handler)
    logger = rLogger.getChild('batch')
    radiomics.setVerbosity(logging.INFO)
```

```
39
40      try:
41      # Use pandas to read and transpose ('.T') the input data
42      # The transposition is needed so that each column represents one test case. This
            is easier for iteration over the input cases
43          flists = pandas.read_excel(inputXLSX).T
44      except Exception:
45          logger.error('Excel READ FAILED', exc_info=True)
46          exit(-1)
47
48      logger.info('Loading Done')
49      logger.info('Patients: %d', len(flists.columns))
50
51      #Change the settings
52      settings = {}
53      settings['binWidth'] = 2 #To get the right amount of bins
54      settings['sigma'] = [1]
55      extractor = featureextractor.RadiomicsFeaturesExtractor(**settings)
56      extractor.enableAllImageTypes()
57
58      logger.info('Enabled input images types: %s', extractor._enabledImagetypes)
59      logger.info('Enabled features: %s', extractor._enabledFeatures)
60      logger.info('Current settings: %s', extractor.settings)
61
62      results = pandas.DataFrame()
63
64      for entry in flists:  # Loop over all columns (i.e. the test cases)
65          logger.info("(%d/%d) Processing Patient: %s",
66                      entry + 1,
67                      len(flists.columns),
68                      flists[entry]['pasientID'])
69
70          imageFilepath = flists[entry]['Image']
71          maskFilepath = flists[entry]['Mask']
72          label = flists[entry].get('Label', None)
73
74          if str(label).isdigit():
75            label = int(label)
76          else:
77              label = None
78
79          if (imageFilepath is not None) and (maskFilepath is not None):
80            featureVector = flists[entry]  # This is a pandas Series
81            featureVector['Image'] = os.path.basename(imageFilepath)
82            featureVector['Mask'] = os.path.basename(maskFilepath)
83
84            try:
85              # PyRadiomics returns the result as an ordered dictionary, which can be
                    easily converted to a pandas Series
86              # The keys in the dictionary will be used as the index (labels for the
                    rows), with the values of the features
87              # as the values in the rows.
88              result = pandas.Series(extractor.execute(imageFilepath, maskFilepath,
                    label))
89              featureVector = featureVector.append(result)
90            except Exception:
91                logger.error('FEATURE EXTRACTION FAILED:', exc_info=True)
92
```

```
        featureVector.name = entry
          # By specifying an 'outer' join, all calculated features are added to the
              data frame, including those not
          # calculated for previous cases. This also ensures we don't end up with an
              empty frame, as for the first patient
          # it is 'joined' with the empty data frame.
        results = results.join(featureVector, how='outer')

    logger.info('Extraction complete, writing Excel')
      # .T transposes the data frame, so that each line will represent one patient,
          with the extracted features as columns

    results.T.to_excel(outputFilepath, index=False, na_rep='NaN')
    logger.info('Excel writing complete')

if __name__ == "__main__":
    name_input = 'paths_CT.xlsx'
    name_output = 'radiomics_features_CT_128_masse.xlsx'
    pyradiomics_batch(name_input,name_output)
```

B.3 Definering av klassifikasjonsalgoritmer og egenskapsutvelgere

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 12 2018

@author: Alise Midtfjord
"""

#Defining fourteen different classifiers and seven different feature selectors.
"------------------Classification------------------"

import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from pyearth import Earth
from sklearn.cross_decomposition import PLSRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,
    QuadraticDiscriminantAnalysis
from sklearn.neural_network import MLPClassifier
from skrebate import ReliefF
from sklearn.feature_selection import RFE, SelectKBest, mutual_info_classif
from sklearn.decomposition import PCA, FastICA
from sklearn.model_selection import GridSearchCV
import warnings

def logrel1(X_std_train, y_train, state):
    '''
```

```
    Classification using Logistic regression with L1-regularization.

    :param str X_std_train: Training data
    :param str y_train: Response to the training data
    :param int state: Random state
    :return: Classification model made from the training data
    '''

    param_range = [0.0001, 0.001,0.005, 0.01,0.05, 0.1, 1.0, 10.0, 100.0, 1000.0]
    param_grid = {'C' : param_range}
    gs = GridSearchCV(estimator=LogisticRegression(fit_intercept = True, random_stat
        = state,
                                                   solver = 'liblinear',
                                                   penalty = 'l1',class_weight = '
                                                      balanced'),
                      param_grid=param_grid,
                      scoring='roc_auc',
                      cv=4)

    model = gs.fit(X_std_train,y_train)
    print('Training score Logistic regression:', gs.best_score_)
    print(gs.best_params_)
    return(model)


def logrel2(X_std_train, y_train, state):
    '''
    Classification using Logistic regression with L2-regularization.

    :param str X_std_train: Training data
    :param str y_train: Response to the training data
    :param int state: Random state
    :return: Classification model made from the training data
    '''
    param_range = [0.0001, 0.001,0.005, 0.01,0.05, 0.1, 1.0, 10.0, 100.0, 1000.0]
    param_grid = {'C' : param_range}
    gs = GridSearchCV(estimator=LogisticRegression(fit_intercept = True, random_stat
        = state,
                                                   solver = 'liblinear', class_weigh
                                                       = 'balanced',
                                                   penalty = 'l2'),
                      param_grid=param_grid,
                      scoring='roc_auc',
                      cv=4)

    model = gs.fit(X_std_train,y_train)
    print('Training score Logistic regression:', gs.best_score_)
    print(gs.best_params_)
    return(model)

def rf(X_train,y_train, X_test, y_test, state):
    '''
    Classification using Random Forests.

    :param str X_std_train: Training data
    :param str y_train: Response to the training data
    :param int state: Random state
    :return: Classification model made from the training data
```

```
    '''
    param_range = [1, 2, 3, 4, 5, 6, 7, None]
    param_grid = {'max_depth': param_range}
    gs = GridSearchCV(estimator=RandomForestClassifier(
                                                n_estimators=100,
                                                n_jobs=-1, random_state = state,
                                                    class_weight = 'balanced'),
                  param_grid=param_grid,
                  scoring='roc_auc',
                  cv=4)
    model = gs.fit(X_train,y_train)
    print('Training score Random forest:', model.best_score_)
    print(model.best_params_)
    return(model)


def knn(X_std_train, y_train):
    '''
    Classification using K-nearest-neighbors

    :param str X_std_train: Training data
    :param str y_train: Response to the training data
    :return: Classification model made from the training data
    '''
    param_range = [2,5,8, 10, 15,20]
    feature_range = [10,30,50]
    param_grid = {'n_neighbors': param_range, 'leaf_size': feature_range}
    gs = GridSearchCV(estimator = KNeighborsClassifier(n_jobs = -1),
                      param_grid = param_grid,
                      scoring = 'roc_auc',
                      cv = 4)
    model = gs.fit(X_std_train,y_train)
    print('Training score KNN:', gs.best_score_)
    print(gs.best_params_)
    return(model)

def adaboostlog(X_std_train, y_train, state):
    '''
    Classification using Adaboost with Logistic Regression as base.

    :param str X_std_train: Training data
    :param str y_train: Response to the training data
    :param int state: Random state
    :return: Classification model made from the training data
    '''
    base = LogisticRegression()
    param_range = [0.5,1, 2, 3]
    param_grid = {'learning_rate': param_range}
    gs = GridSearchCV(estimator = AdaBoostClassifier(base, n_estimators = 1000,
                                                     random_state = state),
                      param_grid = param_grid,
                      scoring = 'roc_auc',
                      cv = 4)

    model = gs.fit(X_std_train,y_train)
    print('Training score AdaBoost:', gs.best_score_)
    print(gs.best_params_)
    return(model)
```

```
142
143
144 def decisiontree(X_train, y_train, state):
145     '''
146     Classification using Decision Trees.
147
148     :param str X_std_train: Training data
149     :param str y_train: Response to the training data
150     :param int state: Random state
151     :return: Classification model made from the training data
152     '''
153     param_range = [1, 2, 3, 4, 5, 6, 7, None]
154     param_grid = {'max_depth': param_range}
155
156
157     gs = GridSearchCV(estimator = DecisionTreeClassifier(random_state = state,
        class_weight = 'balanced'),
158                       param_grid = param_grid,
159                       scoring = 'roc_auc',
160                       cv = 4)
161
162     model = gs.fit(X_train,y_train)
163     print('Training score Decision Tree:', gs.best_score_)
164     print(gs.best_params_)
165     return(model)
166
167
168
169 def gnb(X_std_train, y_train, state):
170     '''
171     Classification using Gaussian Naives Bayes.
172
173     :param str X_std_train: Training data
174     :param str y_train: Response to the training data
175     :param int state: Random state
176     :return: Classification model made from the training data
177     '''
178     param_grid = {}
179     gs = GridSearchCV(estimator=GaussianNB(),
180                       param_grid=param_grid,
181                       scoring='roc_auc',
182                       cv=4)
183
184     model = gs.fit(X_std_train,y_train)
185     print('Training score Linear GNB:', gs.best_score_)
186     print(gs.best_params_)
187     return(model)
188
189 def lda(X_std_train, y_train):
190     '''
191     Classification using LDA.
192
193     :param str X_std_train: Training data
194     :param str y_train: Response to the training data
195     :return: Classification model made from the training data
196     '''
197     param_range = ['auto',0.1,0.5,0.8,0]
198     feature_range = [5,7,9,10]
```

```
199     tol_range = [0.0001, 0.00001, 0.001, 0.01]
200     param_grid = [{'solver' : ['lsqr'], 'shrinkage' : param_range, 'n_components' :
            feature_range,
201                    'tol' : tol_range}, {'solver' : ['svd'], 'n_components' :
                       feature_range,
202                    'tol' : tol_range}]
203     gs = GridSearchCV(estimator=LinearDiscriminantAnalysis(priors = [0.776,0.224]),
204                       param_grid=param_grid,
205                       scoring='roc_auc',
206                       cv=4)
207 
208     warnings.filterwarnings('ignore')
209     model = gs.fit(X_std_train,y_train)
210     print('Training score Linear LDA:', gs.best_score_)
211     print(gs.best_params_)
212     warnings.filterwarnings('default')
213     return(model)
214 
215 def qda(X_std_train, y_train):
216     '''
217     Classification using QDA.
218 
219     :param str X_std_train: Training data
220     :param str y_train: Response to the training data
221     :return: Classification model made from the training data
222     '''
223     tol_range = [0.0001, 0.00001, 0.001, 0.01]
224     reg_range = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
225     param_grid = {'tol' : tol_range,'reg_param' : reg_range}
226     gs = GridSearchCV(estimator=QuadraticDiscriminantAnalysis(priors = [0.776,0.224])
            ,
227                       param_grid=param_grid,
228                       scoring='roc_auc',
229                       cv=4)
230     warnings.filterwarnings('ignore')
231     model = gs.fit(X_std_train,y_train)
232     print('Training score QDA:', gs.best_score_)
233     print(gs.best_params_)
234     warnings.filterwarnings('default')
235     return(model)
236 
237 def nnet(X_std_train, y_train, state):
238     '''
239     Classification using Neural Network.
240 
241     :param str X_std_train: Training data
242     :param str y_train: Response to the training data
243     :param int state: Random state
244     :return: Classification model made from the training data
245     '''
246     param_range = [5,10,50, 100, 150, 200]
247     feature_range = [0.00001, 0.0001, 0.001, 0.01, 0.1]
248     tol_range = [0.0001, 0.00001, 0.001, 0.01]
249     param_grid = {'hidden_layer_sizes': param_range, 'alpha' : feature_range, 'tol' :
            tol_range}
250 
251     gs = GridSearchCV(estimator=MLPClassifier(random_state=state),
252                       param_grid=param_grid,
```

```
                      scoring='roc_auc',
                      cv=4)
    warnings.filterwarnings('ignore')
    model = gs.fit(X_std_train,y_train)
    print('Training score Neural networkC:', gs.best_score_)
    print(gs.best_params_)
    warnings.filterwarnings('default')
    return(model)

def mars(X_std_train, y_train):
    '''
    Classification using Multivariate Adaptive Regression Splines.

    :param str X_std_train: Training data
    :param str y_train: Response to the training data
    :return: Classification model made from the training data
    '''
    param_range = [1,3,5]
    feature_range = [0.01,0.05,0.1,0.2]
    param_grid = {'penalty' : param_range, 'minspan_alpha' : feature_range}
    gs = GridSearchCV(estimator=Earth(),
                      param_grid=param_grid,
                      scoring='roc_auc',
                      cv=4)

    warnings.filterwarnings('ignore')
    model = gs.fit(X_std_train,y_train)
    print('Training score MARS:', gs.best_score_)
    print(gs.best_params_)
    warnings.filterwarnings('default')
    return(model)


def plsr(X_std_train, y_train):
    '''
    Classification using Partial Least Squares.

    :param str X_std_train: Training data
    :param str y_train: Response to the training data
    :return: Classification model made from the training data
    '''
    param_range =  np.arange(1,X_std_train.shape[1]+1)
    feature_range = [0.0000001, 0.000001, 0.00001]
    param_grid = {'n_components' : param_range, 'tol' : feature_range}
    gs = GridSearchCV(estimator=PLSRegression(scale = False),
                      param_grid=param_grid,
                      scoring='roc_auc',
                      cv=4)

    warnings.filterwarnings('ignore')
    model = gs.fit(X_std_train,y_train)
    print('Training score PLSR:', gs.best_score_)
    print(gs.best_params_)
    warnings.filterwarnings('default')
    return(model)

def svc(X_std_train, y_train, state):
    '''
```

```
311     Classification using Support Vector Classifer.
312 
313     :param str X_std_train: Training data
314     :param str y_train: Response to the training data
315     :param int state: Random state
316     :return: Classification model made from the training data
317     '''
318     param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
319     feature_range = [0.001,0.05,0.1,0.2]
320     cache_range = [50,100,200,300]
321     param_grid = {'C': param_range, 'gamma' : feature_range, 'cache_size' :
            cache_range}
322     gs = GridSearchCV(estimator=SVC(random_state = state, class_weight = 'balanced'),
323                       param_grid=param_grid,
324                       scoring='roc_auc',
325                       cv=4)
326 
327     model = gs.fit(X_std_train,y_train)
328     print('Training score SVC:', gs.best_score_)
329     print(gs.best_params_)
330     return(model)
331 
332 def linearsvc(X_std_train, y_train, state):
333     '''
334     Classification using Linear Support Vector Classifier.
335 
336     :param str X_std_train: Training data
337     :param str y_train: Response to the training data
338     :param int state: Random state
339     :return: Classification model made from the training data
340     '''
341     param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
342     feature_range = [0.00001,0.0001,0.001,0.1,1]
343     param_grid = {'C': param_range, 'tol' : feature_range}
344     gs = GridSearchCV(estimator=LinearSVC(fit_intercept = True, random_state = state,
            class_weight = 'balanced'),
345                       param_grid=param_grid,
346                       scoring='roc_auc',
347                       cv=4)
348 
349     model = gs.fit(X_std_train,y_train)
350     print('Training score Linear SVC:', gs.best_score_)
351     print(gs.best_params_)
352     return(model)
353 
354 "------------------Feature selection------------------"
355 
356 def relieff(X_std_train, X_std_test, y_train,n_features, colNames, features):
357     '''
358     Feature selection using ReliefF.
359 
360     :param str X_std_train: Training data
361     :param str X_std_test: Validation data
362     :param str y_train: Response to the training data
363     :param int n_features: Number of features to be selected
364     :param colNames: List with the names of the columns/features
365     :features: List that the selected features will be added to
366     :return: The training data and validation data with only the selected features
```

```
367                 and the list with the features
368     '''
369     relieff = ReliefF(n_features_to_select=n_features, n_neighbors=20)
370     relieff.fit(X_std_train,y_train)
371     importances = relieff.feature_importances_
372     indices = np.argsort(importances)[::-1]
373     feature_names = []
374 
375     for f in range(X_std_train.shape[1]):
376         feature_names.append(colNames[indices[f]])
377     print(feature_names[0:n_features])
378     X_std_train = X_std_train[:,indices[0:n_features]]
379     X_std_test = X_std_test[:,indices[0:n_features]]
380     features.append(feature_names[0:n_features])
381     return (X_std_train, X_std_test, features)
382 
383 def rf_selection(X_train, X_test, y_train, colNames, state, n_features, features):
384     '''
385     Feature selection using Random Forests.
386 
387     :param str X_std_train: Training data
388     :param str X_std_test: Validation data
389     :param str y_train: Response to the training data
390     :param colNames: List with the names of the columns/features
391     :param state: Random state
392     :param int n_features: Number of features to be selected
393     :features: List that the selected features will be added to
394     :return: The training data and validation data with only the selected features
395              and the list with the features
396     '''
397     forest = RandomForestClassifier(n_estimators=1000,
398                                 n_jobs=-1, class_weight = 'balanced', random_state=
                                       state)
399     forest.fit(X_train, y_train)
400     importances = forest.feature_importances_
401 
402     indices = np.argsort(importances)[::-1]
403     feature_names = []
404 
405     for f in range(X_train.shape[1]):
406         feature_names.append(colNames[indices[f]])
407     print(feature_names[0:n_features])
408     features.append(feature_names[0:n_features])
409 
410     X_train = X_train[:,indices[0:n_features]]
411     X_test = X_test[:,indices[0:n_features]]
412     return (X_train, X_test, features)
413 
414 def logrel1_selection(X_std_train, X_std_test, y_train,n_features, state, colNames):
415     '''
416     Feature selection using L1-logistic regression.
417 
418     :param str X_std_train: Training data
419     :param str X_std_test: Validation data
420     :param str y_train: Response to the training data
421     :param int n_features: Number of features to be selected
422     :param state: Random state
423     :param colNames: List with the names of the columns/features
```

```
424     :return: The training data and validation data with only the selected features
425     '''
426     leng = X_std_train.shape[1]
427     logistic = LogisticRegression(penalty = 'l1',class_weight = 'balanced',
            random_state = state)
428     logre = RFE(logistic, n_features,step = 10)
429     X_std_train = logre.fit_transform(X_std_train, y_train)
430     X_std_test = logre.transform(X_std_test)
431     for i in range(leng):
432         if logre.get_support()[i] == True:
433             print (colNames[i])
434     return(X_std_train, X_std_test)
435 
436 def lda_selection(X_std_train, X_std_test, y_train):
437     '''
438     Dimension reduction using LDA.
439 
440     :param str X_std_train: Training data
441     :param str X_std_test: Validation data
442     :param str y_train: Response to the training data
443     :return: The training data and validation data with only the new component
444     '''
445     mod = LinearDiscriminantAnalysis(tol = 0.1)
446     X_std_train = mod.fit_transform(X_std_train,y_train)
447     X_std_test = mod.transform(X_std_test)
448 
449     return(X_std_train, X_std_test, mod)
450 
451 def pca(X_std_train, X_std_test, n_features, state):
452     '''
453     Dimension reduction using PCA.
454 
455     :param str X_std_train: Training data
456     :param str X_std_test: Validation data
457     :param int n_features: Number of components
458     :return: The training data and validation data with only the new components
459     '''
460     mod = PCA(n_components = n_features, random_state = state)
461     X_std_train = mod.fit_transform(X_std_train)
462     X_std_test = mod.transform(X_std_test)
463     return(X_std_train, X_std_test)
464 
465 def mutual(X_std_train, X_std_test, y_train, n_features):
466     '''
467     Feature selection using Mutual Information.
468 
469     :param str X_std_train: Training data
470     :param str X_std_test: Validation data
471     :param str y_train: Response to the training data
472     :param int n_features: Number of features to be selected
473     :return: The training data and validation data with only the selected features
474     '''
475     leng = X_std_train.shape[1]
476     mod = SelectKBest(score_func = mutual_info_classif, k = n_features)
477     X_std_train = mod.fit_transform(X_std_train, y_train)
478     X_std_test = mod.transform(X_std_test)
479     for i in range(leng):
480         if mod.get_support()[i] == True:
```

```
            print (colNames[i])
    return(X_std_train, X_std_test)

def ICA(X_std_train, X_std_test, n_feautures, state):
    '''
    Dimension reduction using ICA.

    :param str X_std_train: Training data
    :param str X_std_test: Validation data
    :param int n_features: Number of components
    :param state: Random state
    :return: The training data and validation data with only the new components
    '''
    ica = FastICA(n_components = n_features, random_state = state, fun = 'exp', tol =
        0.001)
    X_std_train = ica.fit_transform(X_std_train)
    X_std_test = ica.transform(X_std_test)
    return(X_std_train, X_std_test)
```

B.4 Klassifisering og testing av antall egenskaper

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 12 2018

@author: Alise Midtfjord
"""


#Classification using the thirteen classifiers givens in the script "functions".
"------------------Classification------------------"

def klassifisering(input_excel, ark, y_navn, n_features):
    '''
    Classification using thirteen classifiers defined in the script "functions".
    Uses 4-folds-CV ten times with different splits each times. Uses given number of
    features and chosen feature selector.

    :param str input_excel: The name of the excel-file with the dataset
    :param str ark: The name of the sheet with the dataset
    :param str y_navn: The name of the column with the response
    :param int n_features: Number of features to use in the models
    :return: Matrix with the AUC of all classificatons and matrix with the selected
             features
    '''
    # Reads the excel-file
    xls = pd.ExcelFile(input_excel)
    data_raw_df = pd.read_excel(xls, sheetname=ark, index_col = 0)

    # Creates the result-matrix
    results = [[],[],[],[],[],[],[],[],[],[]]
    for i in range(0,10):
        results[i] = np.zeros((13,4))

    # Splits the respons y and the variables X and sets the random states
```

```
    y_name = y_navn
    y = data_raw_df[y_name].values
    X= data_raw_df.drop(y_name,1)
    colNames = list(X.columns)
    states =  [108, 355, 44, 129, 111, 362, 988, 266, 82,581]
    features = []
    stdsc = StandardScaler()

    # Splits the dataset into the 10*4 folders and selects features and uses the
       classifers
    for k in range(0,10):
        i = 0
        state = states[k]
        cv = StratifiedKFold(n_splits=4, random_state = state, shuffle = True)
        for train, test in cv.split(X, y):
            print(k,i)
            X_train = X.iloc[train]
            X_test = X.iloc[test]
            y_train = y[train]
            y_test = y[test]
            X_std_train = stdsc.fit_transform(X_train)
            X_std_test = stdsc.transform(X_test)
            X_std_train, X_std_test, features = relieff(X_std_train, X_std_test,
               y_train,n_features, colNames, features)

            model = logrel1(X_std_train, X_std_test, state)
            print('Test score L1-Logistic regression:', model.score(X_std_test,y_test
               ))
            results[k][0,i] = model.score(X_std_test,y_test)
            model = logrel2(X_std_train, X_std_test, state)
            print('Test score L2-Logistic regression:', model.score(X_std_test,y_test
               ))
            results[k][1,i] = model.score(X_std_test,y_test)
            model = rf(X_train,y_train, state)
            print('Test score Random forest:', model.score(X_test,y_test))
            results[k][2,i] = model.score(X_test,y_test)
            model = knn(X_std_train, y_train)
            print('Test score KNN:', model.score(X_std_test,y_test))
            results[k][3,i] = model.score(X_std_test,y_test)
            model = adaboostlog(X_std_train,  y_train,  state = state)
            print('Test score AdaBoost:', model.score(X_std_test,y_test))
            results[k][4,i] = model.score(X_std_test,y_test)
            model = decisiontree(X_std_train,y_train, state)
            print('Test score Decision Tree:', model.score(X_test,y_test))
            results[k][5,i] = model.score(X_std_test,y_test)
            model = gnb(X_std_train, y_train,  state = state)
            print('Test score GNB:', model.score(X_std_test,y_test))
            results[k][6,i] = model.score(X_std_test,y_test)
            model = lda(X_std_train, y_train)
            print('Test score Linear LDA:', model.score(X_std_test,y_test))
            results[k][7,i] = model.score(X_std_test,y_test)
            model = qda(X_std_train, y_train)
            print('Test score QDA:', model.score(X_std_test,y_test))
            results[k][8,i] = model.score(X_std_test,y_test)
            model = nnet(X_std_train, y_train,  state = state)
            print('Test score Neural network:', model.score(X_std_test,y_test))
            results[k][9,i] = model.score(X_std_test,y_test)
            model =  mars(X_std_train, y_train)
```

```
            print('Test score MARS:', model.score(X_std_test,y_test))
            results[k][10,i] = model.score(X_std_test,y_test)
            model =  plsr(X_std_train, y_train)
            print('Test score PLSR:', model.score(X_std_test,y_test))
            results[k][11,i] = model.score(X_std_test,y_test)
            model = svc(X_std_train, X_std_test, y_train, y_test, state)
            print('Test score SVC:', model.score(X_std_test,y_test))
            results[k][12,i] = model.score(X_std_test,y_test)
            model = linearsvc(X_std_train, y_train, state)
            print('Test score Linear SVC:', model.score(X_std_test,y_test))
            results[k][13,i] = model.score(X_std_test,y_test)
            i += 1

    return (results, features)


"------------------Number of features------------------"


def n_features(input_excel, ark, y_navn):
    '''
    Classification using thirteen classifiers defined in the script "funksjoner".
       Uses 4-folds-CV ten times with different splits each times. Tries different
       number of features from 1-20.

    :param str input_excel: The name of the excel-file with the dataset
    :param str ark: The name of the sheet with the dataset
    :param str y_navn: The name of the column with the response
    :return: Matrix with the AUC of all classificatons for different number of
       features and matrix with the selected features
    '''

    # Reads the excel-file
    xls = pd.ExcelFile(input_excel)
    data_raw_df = pd.read_excel(xls, sheetname=ark, index_col = 0)

    # Creates the result-matrix
    results = [[[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]],
               [[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]]]
    for j in range(0,2):
        for i in range(0,20):
            results[j][i] = np.zeros((13,4))

    # Splits the respons y and the variables X and sets the random states
    y_name = y_navn
    y = data_raw_df[y_name].values
    X= data_raw_df.drop(y_name,1)
    stdsc = StandardScaler()
    colNames = list(X.columns)
    states = [209, 979]
    features = []
    stdsc = StandardScaler()

    # Splits the dataset into the 2*4 folders and selects features and uses the
       classifers
    # for 1-20 number of features-
    for l in range(0,2):
        state = states[l]
```

```
        cv = StratifiedKFold(n_splits=4, random_state = state, shuffle = True)
        for k in range(0,20):
            i = 0
            n_features = k + 1
            for train, test in cv.split(X, y):
                print(k,i)
                X_train = X.iloc[train]
                X_test = X.iloc[test]
                y_train = y[train]
                y_test = y[test]
                X_std_train = stdsc.fit_transform(X_train)
                X_std_test = stdsc.transform(X_test)
                X_std_train, X_std_test = relieff(X_std_train, X_std_test, y_train,
                    n_features, colNames, features)

                model = logrel1(X_std_train, X_std_test, y_train, y_test, state)
                results[l][k][0,i] = model.score(X_std_test,y_test)
                model = logrel2(X_std_train, X_std_test, y_train, y_test, state)
                results[l][k][1,i] = model.score(X_std_test,y_test)
                model = rf(X_train,y_train, X_test, y_test, state)
                results[l][k][2,i] = model.score(X_test,y_test)
                model = knn(X_std_train, X_std_test, y_train, y_test)
                results[l][k][3,i] = model.score(X_std_test,y_test)
                model = adaboostlog(X_std_train, X_std_test, y_train, y_test, state =
                    state)
                results[l][k][4,i] = model.score(X_std_test,y_test)
                model = decisiontree(X_std_train,y_train, X_std_test, y_test, state)
                results[l][k][5,i] = model.score(X_std_test,y_test)
                model = gnb(X_std_train, X_std_test, y_train, y_test, state = state)
                results[l][k][6,i] = model.score(X_std_test,y_test)
                model = lda(X_std_train, X_std_test, y_train, y_test)
                results[l][k][7,i] = model.score(X_std_test,y_test)
                model = qda(X_std_train, X_std_test, y_train, y_test)
                results[l][k][8,i] = model.score(X_std_test,y_test)
                model = nnet(X_std_train, X_std_test, y_train, y_test, state = state)
                results[l][k][9,i] = model.score(X_std_test,y_test)
                model =  mars(X_std_train, X_std_test, y_train, y_test)
                results[l][k][10,i] = model.score(X_std_test,y_test)
                model =  plsr(X_std_train, X_std_test, y_train, y_test)
                results[l][k][11,i] = model.score(X_std_test,y_test)
                model = svc(X_std_train, X_std_test, y_train, y_test, state)
                results[l][k][12,i] = model.score(X_std_test,y_test)
                model = linearsvc(X_std_train, X_std_test, y_train, y_test, state)
                results[l][k][13,i] = model.score(X_std_test,y_test)
                i += 1

    return (results)

"------------------Main------------------"


if __name__ == "__main__":
    import pandas as pd
    import numpy as np

    # import modules and functions from the file functions.py
    from sklearn.model_selection import (StratifiedKFold, GridSearchCV)
    from sklearn.preprocessing import StandardScaler
```

```
199     from functions import (logrel1, logrel2, rf, knn, adaboostlog, decisiontree, gnb
            lda,
200                            qda, nnet, mars, plsr, svc, linearsvc, relieff,
                                rf_selection,
201                            logrel1_selection, lda_selection, pca, mutual, ICA)
202
203
204     # Insert the right names
205     input_excel = "X_endelig_squareroot.xlsx"
206     ark = 'tilbakefall'
207     y_navn = 'Toklasser'
208
209     # Choose method. Method = 0 for classification with a spesific number of features
            , or method = 1 for trying 1-20 different number of features.
210     metode = 1
211     if metode == 0:
212         n_features = 2
213         results, features = klassifisering(input_excel, ark, y_navn, n_features)
214     elif metode == 1:
215         results = n_features(input_excel, ark, y_navn)
```

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

Postboks 5003
NO-1432 Ås
Norway