TEK9020 - Prosjektoppgave 3

Kristoffer Langstad
kristoffer.langstad@its.uio.no

1 Introduksjon

I denne oppgaven er målet å se hvordan forskjellige metoder for klyngeanalyse fordeler samplene i et datasett til ulike klasser utifra egenskapene til samplene, for så å evaluere klyngene med noen standard klassifikatorer. Klyngeanalyse brukes ofte i ikke-ledet læring sammenhenger for å fordele samplene i datasett inn i klasser for å gi en bedre oversikt og kunnskap om datasettet. I dette tilfellet brukes Fisher's Iris blomster datasett, som inneholder kjente klasser. De kjente klassene brukes for evaluering sin skyld, der det fokuseres på det ikke-ledet læring tilfellet med klyngeanalyse. Det skal ses på to forskjellige klyngemetoder, KMeans og agglomerativ metode, for å dele samplene i Iris datasettet inn i klynger. Så vil disse klyngemetodene evalueres opp mot de sanne klassene, og ses på forskjeller mellom metodene.

Først presenteres litt teori om klyngeanalyse og de to klyngemetodene, så presenteres datasettet, etterfulgt av metodikken med eksperimenter og hvilke klassifikatorer som skal brukes. Resultater og diskusjoner av resultatene blir så presentert til slutt med en liten konklusjon av oppgaven. Programmeringen er utført i Python (A).

2 Teori - klyngeanalyse

Det sentrale i klyngeanalyse er å dele inn samplene i datasett inn i klynger utifra likheter i egenskapene til samplene. Samplene med mest mulig like egenskaper fordeles i klynger, samtidig mens det er størst mulig ulikhet til andre sampler. For slike tilfeller er det som oftest ikke kjent klassetilhørighet, ikke-ledet læring, slik at klyngeinndelingene kun baseres på egenskapvektorene til samplene, og ikke a priori kunnskap. På denne måten få kunnskap om strukturer i data som ellers ofte er ukjent. Disse lagde klyngene kan så brukes til videre analyse, der det kan antas at hver klynge tilsvarer hver sin klasse. I tilfeller der datasettet kan merkes utifra klyngene, så kan det brukes til ledet læring ved å trene klassifikatorer.

For å dele samplene inn i klynger med like egenskaper, brukes en avstandsmåler. Denne avstanden spiller en viktig rolle i hvordan datasettet blir fordelt. En av de mest vanlige metrikkene er Euklidsk avstand:

$$d(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{x}_1 - \mathbf{x}_2|| \tag{1}$$

Utifra avstanden mellom to sampler, kan det da settes kriterier for hvilke sampler som skal i samme klynge. Det enkleste er oftest å fordele sampler i samme klynge dersom den Euklidske avstand er mindre enn en satt verdi. F.eks. at sample $\mathbf{x}_1$ og sample $\mathbf{x}_2$ er i samme klynge dersom $d(\mathbf{x}_1, \mathbf{x}_2) < d_0$. Det vil da være klynger der innbyrdes avstand er liten mellom sampler, sammenlignet med sampler i andre klynger.

Det er flere slike klyngemetoder, men det skal fokuseres på KMeans clustering og en hierarkisk metode i dette prosjektet.

2.1 KMeans clustering

Den første metoden i bruk er KMeans clustering. Dette er en slags topp-ned metode som er basert på å lage klyngesenter og iterativt fordele samplene til nærmeste klynge med hjelp av en distanse metode, samtidig som klyngesentrene oppdateres underveis. Distanse metoden måler avstanden mellom de ulike samplene og klyngene for å iterativt tilegne samplene til k gitte klynger. Iterasjonsprosessen er lignende som Algoritme 1.

Ulemper: En mulig ulempe med KMeans metoden er at den krever en viss a prior kunnskap om datasettet, nemlig antall klasser/klynger datasettet skal deles inn i. Dersom dette er ukjent informasjon, så finnes det metoder for å få mer informasjon som kan tilnærme seg hva som er riktig antall klynger. En slik metode er Albue metoden (Eng: Elbow method), der det regnes ut og plottes "within-cluster sum of squares" (WCSS)

Algorithm 1 KMeans iterasjonsprosess.

*Initialisér $\hat{\mu}_1$, ..., $\hat{\mu}_c$
*Gjenta inntil ferdig:
-Klassifiser x_k, k=1, ..., n til nærmeste senteroide ved hjelp av Euclidsk avstand
-Oppdater $\hat{\mu}_1$, ..., $\hat{\mu}_c$
-Hvis ingen endring → ferdig

Algorithm 2 Agglomerativ iterasjonsprosess.

*Start med k klynger, dvs. χ_i={x_i}, i=1, ..., c; $\hat{c} = k$
*Finn nærmeste par av klynger, f.eks. χ_i, χ_j
*Slå sammen disse klyngene $\hat{c} \rightarrow \hat{c} - 1$
*Gjenta de to forrige trinnene til antall ønsket klynger $\hat{c} = c$ er funnet.

mot antall klynger. WCSS beregner den kvadratiske avstanden mellom hvert sample og senteroiden dens. Albue metoden produserer en kurve lik som Figur 1, som er lagd til dette prosjektet på Iris datasettet for å vise den optimale mengden klynger, som her er 3 både utifra kurven og siden dette er kjent for oss med 3 klasser. Det optimale antall klynger finnes i knekken på kurven der kurven ikke lenger endrer seg betraktelig, som passer for 3 klynger.

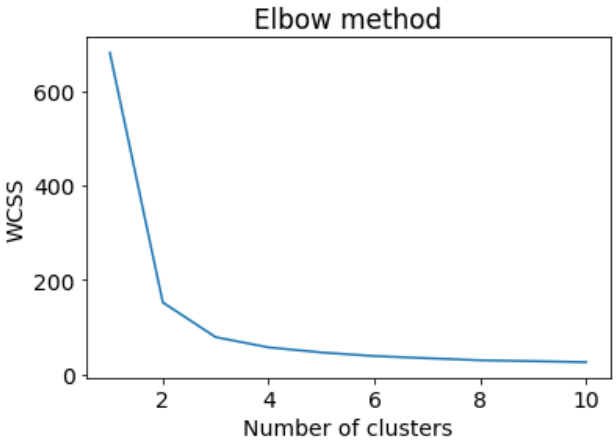


Figure 1: Albue metoden for å finne optimalt antall klynger for et gitt datasett. Her brukt på Iris datasettet.

2.2 Hierakiske metoder

Det er flere hierarkiske metoder brukt innen klyngeanalyse, men det fokuseres på den agglomerative metoden. Den agglomerative metoden fungerer som en bunnen-opp metode der hvert sample er hver sin klynge til å begynne med. Deretter brukes en avstands måler, ofte Euklidsk avstand, for å slå sammen de to nærmeste klyngene hvert steg. Dette fortsetter til alle klyngene har blitt slått sammen til en stor klynge. Den agglomerative metoden kan ses i Algoritme 2.

Så lages et Dendrogram som i Figur 2, denne figuren kan minne om et tre med røtter, der kan det ses utifra figuren hvilket antall klynger det kan være i datasettet. Antall klynger kan bestemmes ut ifra høyden mellom stegene/avstanden. Dendrogrammet i Figur 2 er lagd på Iris datasettet, og viser stor avstand når den har kommet til 3 klynger, som stemmer bra med det som er kjent om datasettet med 3 klasser.

3 Metode

Her ses det kort på hva Iris datasettet inneholder, før selve eksperimentene forklares for å produsere resultatene.

3.1 Datasett

Datasettet som brukes i dette prosjektet er et mye brukt datasett innen mønstergjenkjenning som heter Iris. Fisher's Iris datasett inneholder tre typer varianter av iris blomster (Setosa, Versicolor, Virginica) med 50 sampler hver, i tillegg til at hver sample har fire egenskaper relatert til lengde og bredde på begerbladene og kronbladene:

- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm

Alle blomstene er variasjoner av Iris blomsten, der to av typene er fra samme område og tidspunkt, som gjør at to av blomstene har likere egenskaper men fortsatt ulike hverandre.

Datasettet inneholder også de sanne klassene til blomstene. For å bruke dette datasettet til ikke-ledet læring og klyngeanalyse så utelates de sanne klassene fra datasettet for KMeans og agglomerativ klyngemetoder.

3.2 Eksperimenter

Dette prosjektet kan deles inn i tre eksperimenter: Først se hvordan det originale Iris datasettet ser ut

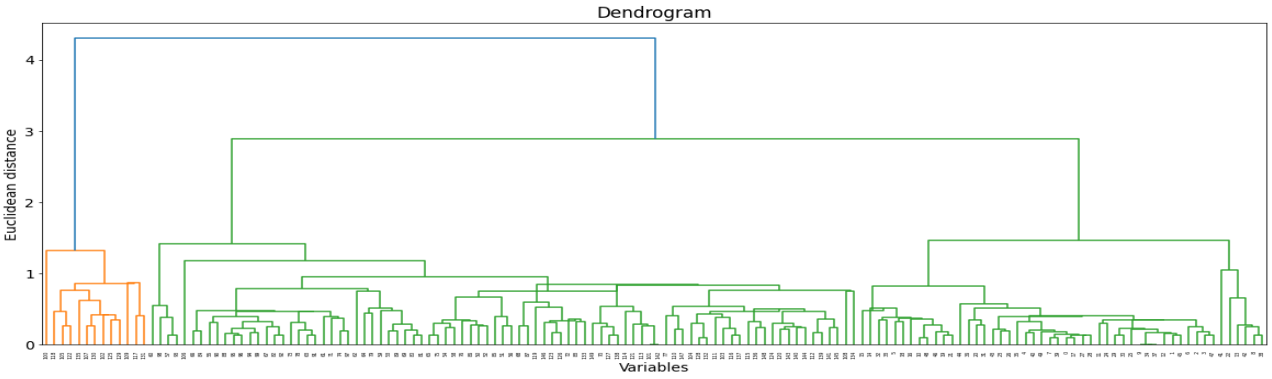


Figure 2: Dendrogam lagd på Iris datasettet for å finne optimalt antall klynger.

i klynger og hvordan noen standard klassifikatorer klarer å trene og klassifisere datasettet. Nummer to er å bruke KMeans metoden for å tildele datasettet, uten klasser, inn i nye klynger, og se hvordan klassifikatorene gjør det på disse klyngene. Nummer tre er å bruke agglomerativ metode for å lage klynger og gjøre samme analyse som for KMeans metoden.

Med det originale Iris datasettet med klassene tilgjengelige, visualiseres enkelt de tre blomster klynge egenskapkombinasjonene i en samlet figur, i et såkalt par-plot. For KMeans og agglomerativ metode, så brukes albue metoden og dendrogram for å vise at antall klasser i datasettet er tre. Så må de først dele samplene inn i klynger, som er uten kjent klassetilhørighet. Dette gjør at de produserte klyngene i utgangspunktet ikke er markerte med klasse, men for visualisering og sammenlignbarhet, så brukes de kjente klassetilhørighetene for å vise med navn hva de forskjellige klyngene er. Å legge til klassenavnene gjøres manuelt for de visualiseres.

Etter at klyngene har blitt produsert med de to klyngemetodene, så brukes de kjente klassene for å evaluere hvor nøyaktige de to metodene er sammenlignet med de originale klyngene. Deretter brukes klassifikatorene til å trene og predikere sampler inn i riktige klynger, som kan sammenlignes mellom de tre klyngene som er lagd og visualisert.

Siden klyngemetodene er statistiske metoder, hjelper det ofte å involvere kryssvalidering og ta gjennomsnitt av prediksjonene. Dette er for å være sikker på at ikke resultatene bare er avhengige av kun en prediksjon, der noe av dataen kanskje utelates fra analysen. Dette er med tanke på å dele datasettet inn i trening og test sett, f.eks. 80% trening og 20% testing. Med kryssvalidering sørges det for at for hver kryssvalidering så endres hvilke sampler som blir en del av test settet. Derfor utføres kryssvalidering 5 ganger for hver klassifikator for å gi endelig evaluering.

3.2.1 Klassifikatorer

For å bruke klassifikatorene til å evaluere klyngemetodene, så brukes det at klassene er kjente for å bruke ledet-læring klassifisering. Klassifikatorene som brukes i dette prosjektet er:

- K-Nærmeste Nabo (Eng: K-Nearest Neighbors, KNN): Fordeler sampler inn i klasser basert på likheter i egenskapene til samplene.
- Logistisk regresjon (Eng: Logistic regression, LR): Den bruker sannsynlighet for å bestemme hvilken klasse et sample tilhører.
- Støttevektormaskiner (Eng: Support Vector Machine, SVM): Prøver å finne det hyperplanet som best mulig separerer samplene i klasser.
- Beslutningstrær: Bruker trestrukturer for å tildele et sample til en klasse utifra flere spørsmål eller kriterier. Dette kan være et enkelt tre (Eng: Decision Tree, DT), eller det kan brukes sammensetninger av flere trær (Eng: Random Forest, RF).
- Nevrale nettverk (Eng: Neural network, NN): Bruker opptil flere lag med nevroner som samarbeider om å lære mønstre i data, som så brukes til å predikere klasser. Bruker her en enkel Multilayer perceptron model (MLP).

For å evaluere klassifikatorene, brukes hovedsaklig hvor nøyaktige prediksjonene er sammenlignet med de sanne klassene (accuracy score), og forvirringsmatriser (confusion matrix) som viser antall riktige og ikke riktige gjetninger per klasse.

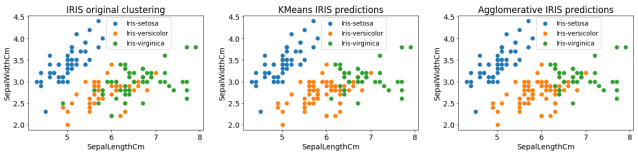
4 Resultater

Presenterer her noen av resultatene som er produsert som brukes for å sammenligne klyngene og metodene. Andre resultater som er produsert, i tillegg til de som vises her, kan ses ved å følge GitHub-linken i A.

4.1 Klynge-inndelinger

I Figur 3 vises for en kombinasjon med to av egenskapene, fra venstre; de originale Iris klyngene (3a), klyngene som er lagd med KMeans metoden (3b), og klyngene som er lagd med agglomerativ metode (3c). Dette er en av egenskapkombinasjonene for å vise klyngene for de tre tilfellene. I Figur 5 vises alle egenskapkombinasjonene for å vise hvordan kombinasjoner av egenskapene til Iris datasettet kan brukes for å vise samplene. I visse tilfeller er det enklere å skille to av klassene, mens for andre kombinasjoner er det mer overlapp mellom samplene. Egenskapkombinasjonen i Figur 3, er en av de med mest overlapp for Iris-versicolor og Iris-virginica klassene.

I Figur 4 vises forvirringsmatrisene lagd av KMeans og agglomerativ metode på Iris datasettet, uten kjente klasser, sammenlignet med de sanne klassene i Iris datasettet for å se hvilke klasser de to klyngemetodene predikerer bra.



(a) Original (b) KMeans (c) Agglomerativ

Figure 3: Fra venstre: Klyngeinndelinger for det originale Iris datasettet (3a), klynger lagd med KMeans metode (3b), og klynger lagd med agglomerativ metode (3c).

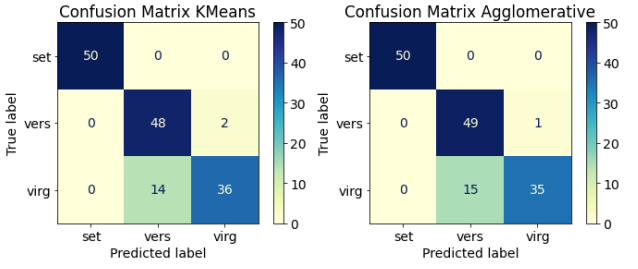


Figure 4: Forvirringsmatriser for KMeans og agglomerativ metode.

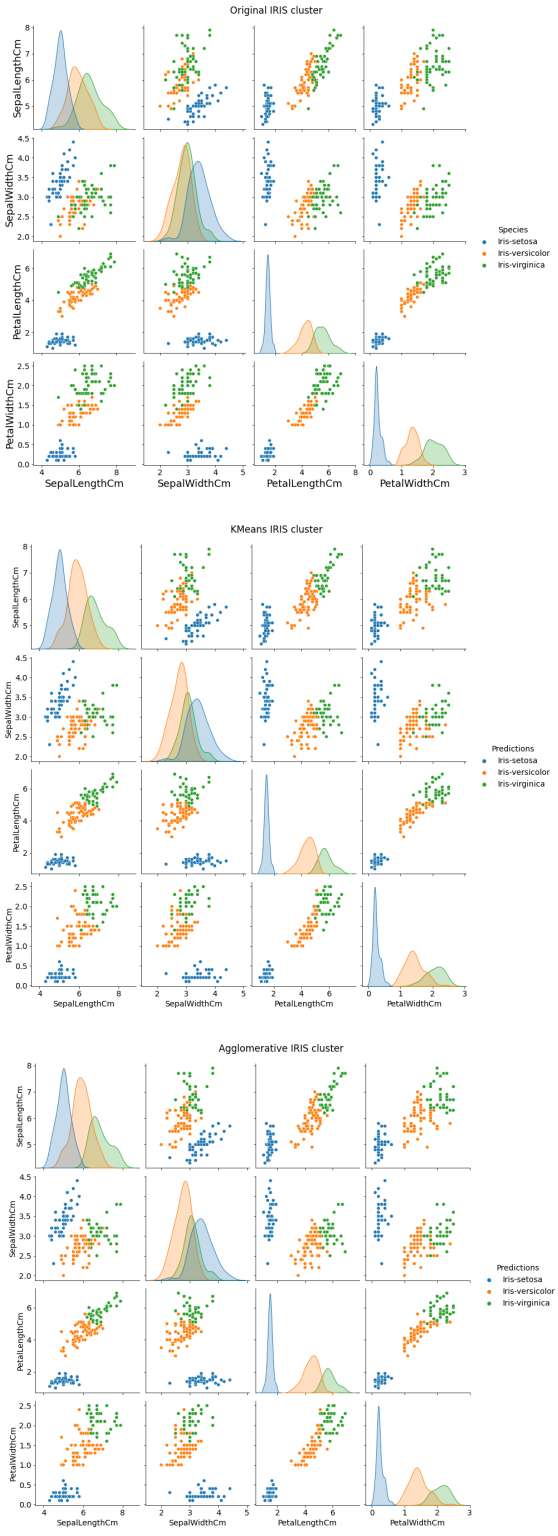


Figure 5: Alle parvis egenskapkombinasjoner for de tre klynge tilfellene.

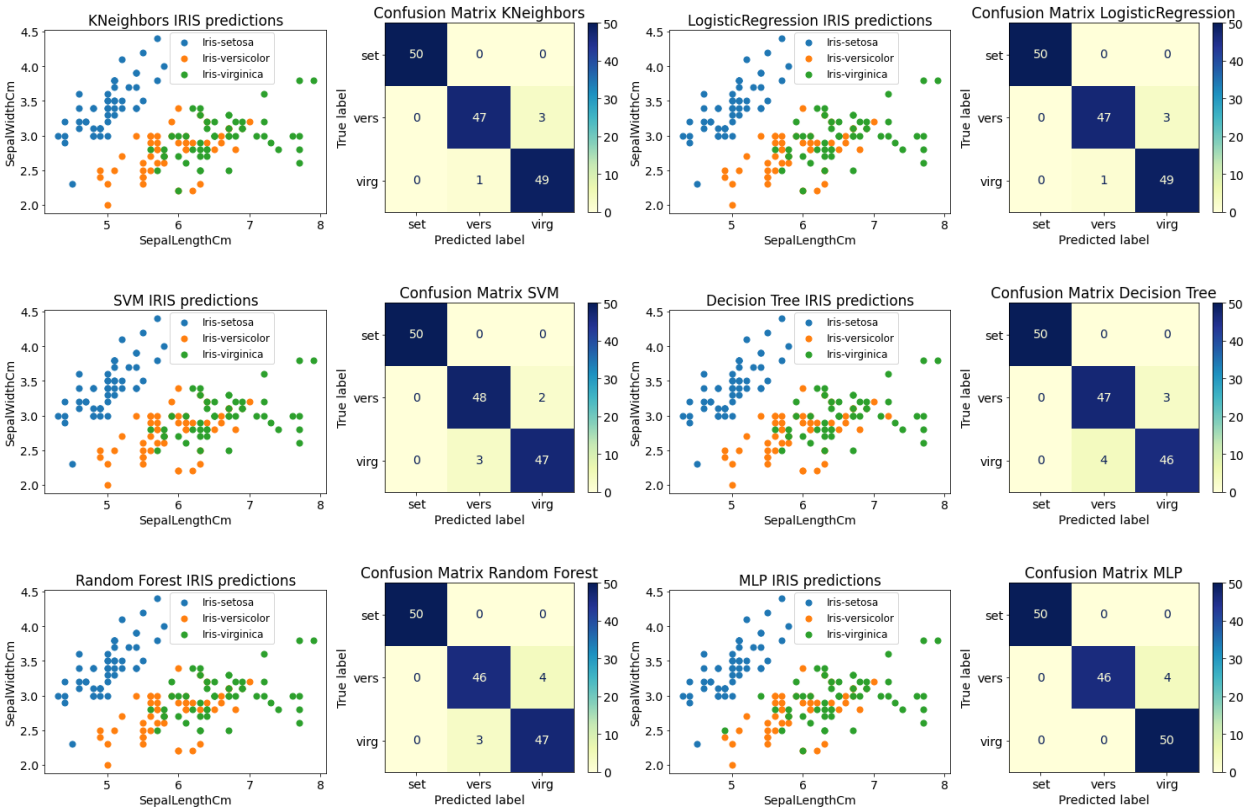


Figure 6: Predikerte klynger og forvirringsmatriser for de forskjellige klassifikatorene på de originale Iris klyngene.

4.2 Klassifikator resultater

I Tabell 1 er accuracy (score) for de ulike klassifikatorene på de tre forskjellige klyngene, der 1 tilsvarer 100% riktig gjetning sammenlignet med de sanne klassene. Predikert accuracy viser hvor likt KMeans (Fig. 3b) og agglomerativ (Fig. 3c) metode lager klynger sammenlignet med de originale klassene (Fig. 3a), m.a.o. samme som accuracy for forvirringsmatrisene i Figur 4. Gjennomsnitt accuracy er over bare de seks klassifikatorene brukt.

Klynge	Original	Kmeans	Agglomerativ
Predikert	-	0.893	0.893
LR	0.973	0.893	0.893
KNN	0.973	0.913	0.907
SVM	0.967	0.900	0.900
DT	0.953	0.893	0.907
RF	0.953	0.913	0.907
MLP	0.973	0.953	0.893
Gjennomsnitt	0.965	0.911	0.901

Table 1: Accuracy score for de ulike klassifikatorene på de tre forskjellige klyngene. Predikert accuracy viser hvor likt KMeans og agglomerativ metode lager klynger sammenlignet med de originale klassene.

4.3 Klassifikator klyngeinndelinger

Figur 6-8 viser de predikerte resultatene med de ulike klassifikatorene på hver av de tre klyngene; original klynge (Fig. 6), KMeans (Fig. 7) og agglomerativ (Fig. 8). For hver klassifikator er det vist de predikerte klyngene og tilhørende forvirringsmatrisene ved siden av hverandre. Egenskapkombinasjonen som er valgt for å vise de predikerte klyngene, er den samme som tidligere der noen av samplene til Iris-versicolor og Iris-virginica overlapper hverandre. Klassifikatorene er for hver av de tre klynge tilfellene trent på sin respektive klynge, men evaluert opp mot de sanne klassene i det originale Iris datasettet som har 50 sampler av hver klasse.

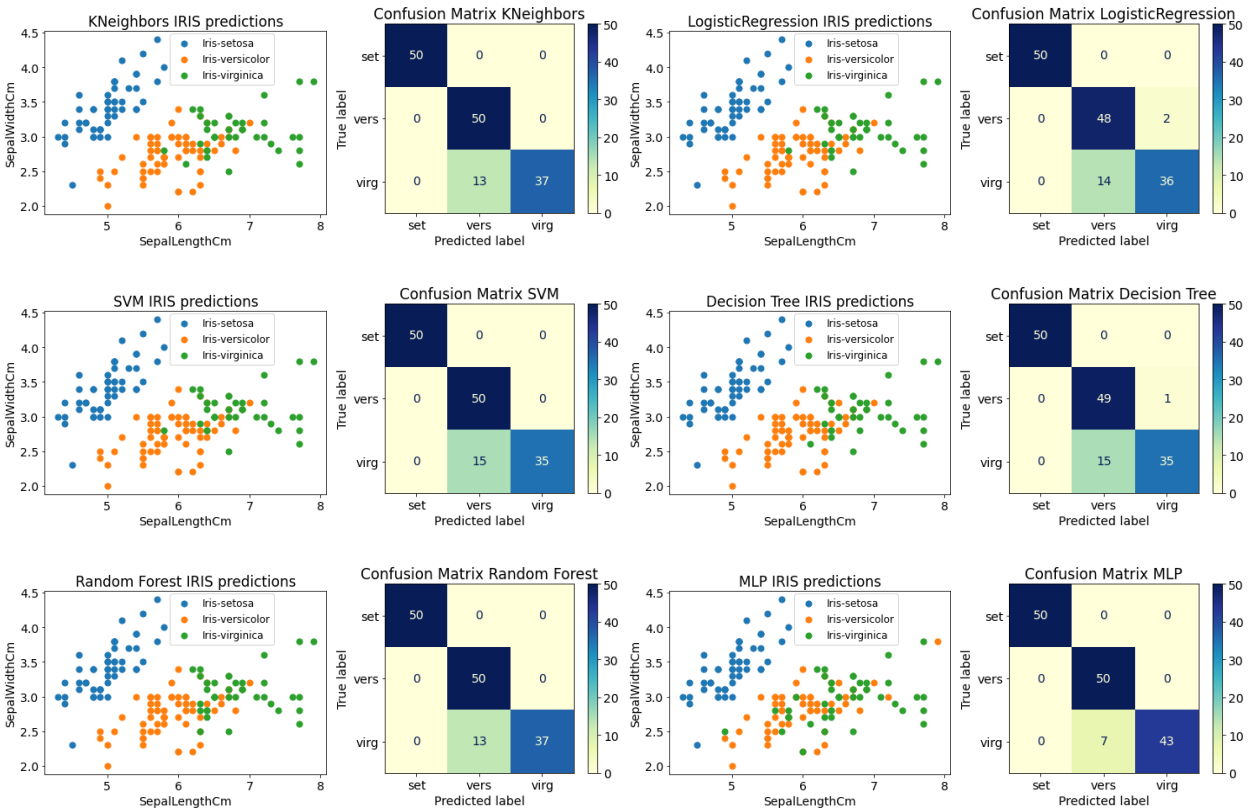


Figure 7: Predikerte klynger og forvirringsmatriser for de forskjellige klassifikatorene trent på de predikerte KMeans klyngene.

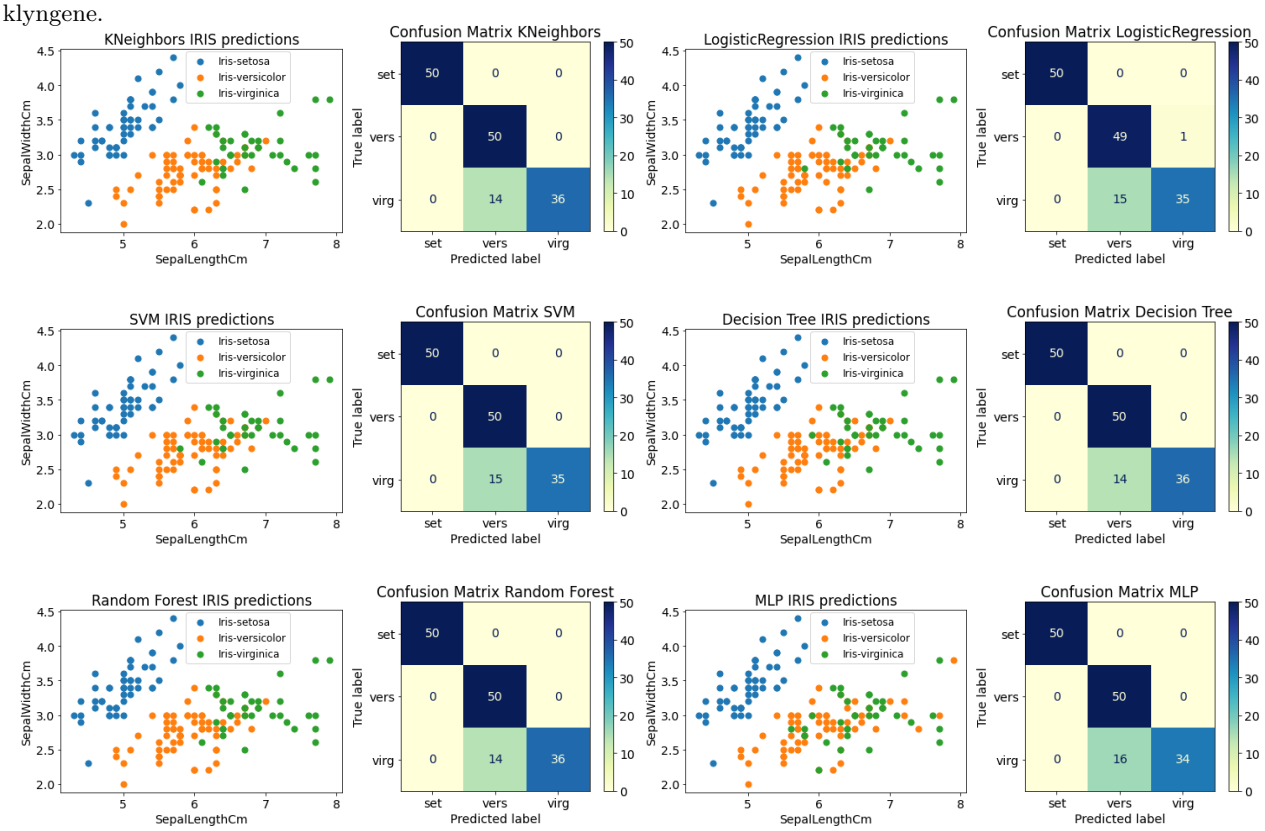


Figure 8: Predikerte klynger og forvirringsmatriser for de forskjellige klassifikatorene trent på de predikerte agglomerative klyngene.

5 Diskusjon

I denne seksjonene blir resultatene og forskjeller på KMeans og agglomerativ metode diskutert.

5.1 Forskjeller på KMeans og Agglomerativ metode

Vanligvis pleier KMeans å være bra på store datasett fordi den er rask og enkel å implementere, og er ofte god til å balansere effektivitet og nøyaktighet. Den egner seg godt dersom det fins noe a prior kunnskap om datasettet, slik at det er en forståelse om hvor mange klynger datasettet skal deles inn i. For å bruke KMeans metoden så er nettopp en av input parameterne antall klynger datasettet skal deles i. I slike tilfeller er albue metoden god til å hjelpe å estimere hva antallet klynger kan være. KMeans metoden har en tendens til å favorisere sirkulære klynger, som ofte egner seg på datasett med global klynge form, og prøver å lage tette klynger.

Sammenlignet med KMeans, er ikke den agglomerative metoden like godt egnet for store datasett på grunn av kompleksiteten i hvordan den lager klynger. Siden den lager et tre av mange klynger med utgangspunkt i antall klynger som antall sampler i datasettet, som reduseres til et gitt antall klynger eller til en stor felles klynge. Den trenger i tillegg å lage gode klynger i starten siden den kan ikke omgjøre valgene av å produsere tidligere klynger underveis, som kan lede til feil eller dårligere valg senere. Tilfellene der agglomerativ metode egner seg godt er for mindre datasett der det lønner seg å utforske forskjellig aspekter av data for å få finne strukturer som kan komme fram gjennom hierarkiske klynger. Klyngene behøver ikke ha like størrelser eller klynge former, og kan håndtere flere typer data.

5.2 Analyse av resultatene

Ved å sammenligne klyngene som produseres fra det originale Iris datasettet, KMeans og agglomerativ metode i Figur 3 og 5, så er det tydelig at det er mindre overlapp mellom Iris-versicolor og Iris-virginica samplene i KMeans og agglomerativ i forhold til det originale datasettet. Så det er tydelig at de to klyngemetodene skiller mer mellom versicolor og virginica samplene, som er naturlig siden de i utgangspunktet ikke har tilgang til de sanne klasse når de lager de nye klyngene. Måten de lager de nye klyngene på er å samle likere sampler sammen. Så når det er overlapp mellom noen av sample-egenskapene i forskjellige klasser, så vil det også være noen prediksjoner som blir forskjellige siden de setter samplene til den nærmeste klynge senteret.

Rent visuelt så er det ikke stor forskjell mellom KMeans og agglomerativ klyngene, det er noen versicolor og virginica sampler som er litt forskjellige i overlapp områdene mellom de to klyngene i Figur 3 og 5. For å se nærmere på prediksjonene av samplene, så vises det hva KMeans og agglomerativ predikerer riktig og feil i forvirringsmatrisen i Figur 4. Der kommer det mer tydelig fram hva de to metodene predikerer, og i utgangspunktet så er det 50 sampler av hver klasse i det originale datasettet. Begge metodene predikerer flere versicolor sampler enn virginica sampler. Istedenfor 50 av hver så er det i de nye klyngene 62 og 64 versicolor sampler. Iris-setosa samplene har egenskaper som er så ulike de to andre klassene, slik at de alltid er mye lettere å gruppere og predikere riktig. Men totalt så har de begge en accuracy score på 0.893 (Se Tabell 1), så de er totalt sett like gode.

Av klassifikatorene som lærer og predikerer på det originale datasettet, så er det LR, KNN og MLP klassifikatorene som er best med accuracy på 0.973. Alle klassifikatorene har accuracy over 0.95, som er ganske bra på dette datasettet. Ingen av klassifikatorene har blitt optimalisert med hensyn på modell parametre, men kun brukt for å vise forskjeller med enkle, ofte brukte klassifikatorer. Det samme gjelder på KMeans og agglomerativ klyngene.

For å evaluere KMeans og agglomerativ klyngene, så er disse klyngene ikke like det originale datasettet, siden de har predikert nye klasser slik at det ikke lenger er 50 sampler av alle tre klassene, som vist i Figur 4. Dette vil da ha noe å si for hvordan klassifikatorene trener på de to predikerte klynge-settene. Dette synes ved å se på accuracy score i Tabell 1, der klassifikatorene presterer bedre på det originale datasettet enn på de to predikerte klynge-metodene. Dette kommer fra de tilfellene der egenskapene til noen av versicolor og virgincia samplene overlapper hverandre, slik at KMeans og agglomerativ predikerer feil klasse på noen sampler. Klassifikatorene klarer da ikke å predikere disse riktig i forhold til de sanne klassene, som senker accuracy noe.

Mellom KMeans og agglomerativ, så presterer klassifikatorene gjennomsnittlig litt bedre når de er trent på KMeans, 0.911 mot 0.901 med agglomerativ. Det er ikke mye forskjell, og hovedforskjellen mellom dem er hvordan MLP klassifikatorene presterer. For de andre klassifikatorene er accuracy mye likere, mens med MLP er det 0.953 mot 0.893. Denne forskjellen kan

være på grunn av tilfeldigheter, siden prestasjonene vil avhenge av tilfeldigheter i både trening og i klassifikatorene som kan endre seg hver gang programmet i Python kjøres. Men kan være andre grunner.

Setosa samplene blir alltid predikert riktig, uansett hvilken klassifikator som er brukt. Dette kommer av at disse samplene har ulike nok egenskaper i forhold til de to andre klassene, at det er lett å skille denne klassen fra de to andre. For forvirringsmatrisene i Figur 6, så er det ganske lik fordeling av predikerte feil mellom versicolor og virginica klassene. For KMeans og agglomerativ så er det hovedsaklig overtall av predikerte feil der noen virginica sampler er klassifisert som versiocolor. Dette kommer mest sannsynlig av at KMeans og agglomerativ begge lager klynger der virginica sampler blir satt i klynger med versicolor sampler utifra noen like egenskaper, som igjen klassifikatorene lærer seg og predikerer feil mot de sanne klassene. Dette blir og mer tydelig vist i Figur 5, ved å sammenligne den øverste figuren med klynger for de originale egenskapkombinasjonene mot de to andre under, for KMeans og agglomerativ. For de to predikerte så er det ofte flere oransje versicolor prikker i forhold til grønne virginica prikker enn det er for de originale kombinasjonene. De predikerte klyngene har tydeligere mindre overlapp mellom versicolor og virginica sampler.

Totalt sett så går prestasjonene litt ned for klassifikatorene når de trenes med klyngemetodene, men ikke med mye. Alle gjennomsnittene for de tre klyngene er over 0.9, som ofte regnes som bra resultater i klassifisering. Dette er i tillegg med at de sanne klassene er kjente, som ofte ikke er tilgjengelige i tilfeller det brukes klyngemetoder. Så klyngemetoder kan ofte være veldig gode på å gruppere sampler sammen i klynger som kan tilnærmes som klasser for opptrening av klassifikatorer og videre analyse.

6 Konklusjon

Klyngeanalyse brukes ofte i ikke-ledet læring tilfeller der vi har sampler fra flere ukjente klasser i et datasett. Forskjellige klyngemetoder, som KMeans og agglomerativ metode, kan brukes for å tilegne samplene inn i like klynger etter egenskapene til samplene. Disse inndelte klyngene kan brukes til å trene opp f.eks. klassifikatorer utifra klasseinndelingen/klyngene til klyngemetodene.

Det originale datasettet er best å bruke, og gir høyere predikert accuracy for de brukte klassifikatorene. Men ofte så er det ikke sikkert at de sanne klassene er tilgjengelige i datasett. Så klyngemetoder, som har blitt brukt her, viser at på Iris datasettet så er det en liten nedgang i hvor nøyaktige klassifikatorene blir, men klyngemetodene viser at det ikke er stor forskjell slik at de fint kan brukes for å tildele ukjente sampler inn i klynger/klasser som er i nærheten av de sanne klassene. Det kan ofte of være ekstra informasjon å få ved å gjøre klyngeanalyse.

A Python kode

Koden og resultater som er brukt i dette prosjektet kan bli funnet i GitHub på: https://github.com/krilangs/TEK9020/tree/main/Project3