Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za automatiku i elektroniku

Seminarski rad

Predmet: Data Mining

Studenti:
Mirza Alispahić (1282/16978),
Emina Hasanovic (1297/16907),
Mehmed Kadrić (1292/16979)

Akademska godina: 2018/2019.

Kratak sadržaj: U ovom seminarskom radu obrađene su dvije teme: klasifikacija i klastering. U zadatku 1 obrađena je tema klasifikacije, dok je u zadatku 2 obrađena tema klasteringa.

Sarajevo, maj 2019.

Sadržaj

Poglavlje 1

Uvod

Mašinsko učenje predstavlja podoblast umjetne inteligencije čiji je cilj konstruisanje algoritama i računarskih sistema koji su sposobni da se adaptiraju na nove situacije i uče na bazi iskustva.

Ideja je da postoji neki algoritam koji smišlja sopstvenu logiku na osnovu podataka.

Osnovna podjela mašinskog učenja je na:

- nadgledano (supervised)
- nenadgledano (unsupervised)

Generalno, nadgledano (supervised) učenje uključuje kreiranje statističkog modela za predikciju ili estimaciju (procjenu) izlaza na osnovu jednog ili više ulaza. Problemi ovog tipa se javljaju u različitim oblastima biznisa, medicine, astrofizike i javne politike.

Kod nenadgledanog učenja (unsupervised) postoje ulazi, ali nemamo nadzor nad izlazom, tj. ne znamo kakav je izlaz. Bez obzira na to, cilj je pronaći odgovarajući šablon u ulaznim podacima i njihovoj strukturi.

Poglavlje 2

Zadatak 1 - Klasifikacija

1 Postavka zadatka

a) Potrebno je da specificitrate hipotezu koju želite riješiti tehnikom klasifikacije, a koja je zasnovana na data setu po Vašem izboru, ili sa nekog od repozitorija (npr. UCI Machine Learning Repository ili Kaggle Datasets) ili iz realnog sektora. Na osnovu specificirane hipoteze, odaberite dva adekvatna algoritma za rješavanje iste, od kojih barem jedan niste radili na predavanjima ili vježbama. Prilagodite Vaš skup podataka odabranim algoritmima primjenom metoda vizualizacije, deskriptivne statistike i tehnika za čišćenje i transformaciju podataka. (3 boda)
b) U programskom jeziku R, implementirajte odabrane algoritme na Vaš data set. Implementirajte dodatnu funkciju pomoću koje ćete izmjeriti tačnost, stepen greške, osjetljivost, specifičnost i preciznost Vaših algoritama, a koristeći k-fold cross1 validaciju. Obrazložite dobijene rezultate. Ukoliko postoji razlika u performansama dva algoritma, objasnite zašto se to dešava. (5 bodova)

2 Postavka problema

Za zadatak klasifikacije odabran je data set sa UCI Machine Learning Repository repozitorija koji se može pronaći ovdje.

Dostupni su podaci iz proizvodnje poluprovodnika (semiconductors). To je vrlo složen proces, te je dostupan ogroman broj različitih mjerenja. Također, dostupna je informacija da li je proizvedeni poluprovodnik uspješno prošao testiranje ili ne. Potrebno je na osnovu dostupnih podataka kreirati model koji će, na osnovu dostupnih mjerenja, klasificirati proizvedeni poluprovnodnik u kategoriju *ispravnih* ili *neispravnih* poluprovodnika. Pod pojmom *ispravan poluprovodnik* misli se na poluprovodnik koji je zadovoljio određene standarde.

3 Opis data seta

Za ovaj zadatak koriste se dva data seta *secom.data* i *Secom_labels.data*. *secom.data* se sastoji od 1567 obzervacija i 590 varijabli (featura). *secom_labels.data* sadrže varijablu koja pokazuje da li je poluprovodnik prošao test ili nije, dakle ova varijabla ima vrijednosti -1 (*pass*) i 1 (*fail*), te sadrži datum i vrijeme kada je poluprovodnik testiran (time stemp).Dakle ovaj data set sadrži 15767 instanci i 2 atributa. *Secom.data* se sastoji od skupa featura gdje svaki podatak predstavlja jednu proizvedenu jedinicu sa odgovarajućim mjerenim featurima. Data set *secom.data* sadrži i neke nedostajuće vrijednosti i zašumljene podatke s obzirom da su navedene atributi dobiveni sa odgovarajućih senzora koji su mjerili određene vrijednosti pri proizvodnji. Pri realizaciji ovog zadatka prvo će se pristupiti ćišćenju i pripremi 9 podataka tako što će se popuniti nedostajuće vrijednosti, otkloniti nepotrebne kolone koje ne doprinose klasifikaciji, itd. Izgled oba data seta je prikazan na narednim slikama, slika 3.1 i slika 3.2.

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
3030.93	2564.00	2187.733	1411.1265	1.3602	100	97.6133	0.1242	1.5005	0.0162	-0.0034	0.9455	202.4396	0	7
3095.78	2465.14	2230.422	1463.6606	0.8294	100	102.3433	0.1247	1.4966	-0.0005	-0.0148	0.9627	200.5470	0	10
2932.61	2559.94	2186.411	1698.0172	1.5102	100	95.4878	0.1241	1.4436	0.0041	0.0013	0.9615	202.0179	0	9
2988.72	2479.90	2199.033	909.7926	1.3204	100	104.2367	0.1217	1.4882	-0.0124	-0.0033	0.9629	201.8482	0	9
3032.24	2502.87	2233.367	1326.5200	1.5334	100	100.3967	0.1235	1.5031	-0.0031	-0.0072	0.9569	201.9424	0	10
2946.25	2432.84	2233.367	1326.5200	1.5334	100	100.3967	0.1235	1.5287	0.0167	0.0055	0.9699	200.4720	0	8
3030.27	2430.12	2230.422	1463.6606	0.8294	100	102.3433	0.1247	1.5816	-0.0270	0.0105	0.9591	202.0901	0	9
3058.88	2690.15	2248.900	1004.4692	0.7884	100	106.2400	0.1185	1.5153	0.0157	0.0007	0.9481	202.4170	0	13
2967.68	2600.47	2248.900	1004.4692	0.7884	100	106.2400	0.1185	1.5358	0.0111	-0.0066	0.9494	202.4544	0	12
3016.11	2428.37	2248.900	1004.4692	0.7884	100	106.2400	0.1185	1.5381	0.0159	0.0049	0.9440	202.5999	0	12
2994.05	2548.21	2195.122	1046.1468	1.3204	100	103.3400	0.1223	1.5144	-0.0190	0.0013	0.9433	201.7125	0	11
2928.84	2479.40	2196.211	1605.7578	0.9959	100	97.9156	0.1257	1.4690	0.0170	-0.0154	0.9445	202.1264	0	9
2920.07	2507.40	2195.122	1046.1468	1.3204	100	103.3400	0.1223	1.5310	-0.0259	0.0216	0.9595	202.1269	0	8
3051.44	2529.27	2184.433	877.6266	1.4668	100	107.8711	0.1240	1.5236	-0.0209	-0.0031	0.9441	226.0086	0	9
2963.97	2629.48	2224.622	947.7739	1.2924	100	104.8489	0.1197	1.4474	0.0144	-0.0119	0.9582	195.3787	0	9
2988.31	2546.26	2224.622	947.7739	1.2924	100	104.8489	0.1197	1.5465	0.0250	-0.0024	0.9616	192.9787	0	12

Slika 3.1 – Izgled data seta *secom.data*

V1	V2
-1	19/07/2008 11:55:00
-1	19/07/2008 12:32:00
1	19/07/2008 13:17:00
-1	19/07/2008 14:43:00
-1	19/07/2008 15:22:00
-1	19/07/2008 17:53:00
-1	19/07/2008 19:44:00
-1	19/07/2008 19:45:00
-1	19/07/2008 20:24:00
-1	19/07/2008 21:35:00
1	19/07/2008 21:57:00
1	19/07/2008 22:52:00
-1	20/07/2008 03:35:00
-1	21/07/2008 08:21:00
1	21/07/2008 11:53:00
-1	22/07/2008 00:03:00
-1	22/07/2008 02:59:00

Slika 3.2 – Izgled data seta *secom_labels.data*

4 Realizacija zadatka u programskom jeziku R

Pretprocesiranje podataka

Izvršeno je pretprocesiranje *secom.data* data seta. Prvo su otklonjene/popunjene kolone u kojima postoje nedostajuće vrijednosti, a zatim je izvršeno balasiranje klasa:

- Prvo su otklonjene kolone gdje je procenat vrijednosti koje nedostaju veći od 50%, pa nakon toga imamo 562 kolone.
- Nakon toga su otklonjene kolone koje imaju konstantne vrijednosti s obzirom da ne utiču na klasifikaciju i kasniju usporedbu korištenih algoritama.Nakon toga ostalo je 446 kolona.
- Kolone koje imaju manje od 50% vrijednosti koje nedostaju.
- Data set koji koristimo ima veoma rijedak problem, a radi se o tome da je većinska klasa *Pass* mnogo brojnija 93.4%, dok je *Fail* klasa zastupljena 6.6% u obrzervacijama, pa dolazi do nebalansa pri predikciji (svi modeli indiciraju 0% osjetljivost).
- Korištena je Synthetic Data Generation i kreirani su balansirani podaci koristeći SMOTE.

Selekcija feature-a i ocjena modela

Ovi koraci su izvršeni na sljedeći način:

- Naime, nisu svi atributi tj. signali koji su u data setu korisni jer postoji mnogo šuma u podacima s obzirom da su dobijeni od odgovarajućih senzora.

- PCA metoda (Principal Commponent Analysis) je korištena kako bi se kreirao potpuno "čist" data set.
- LASSO (least absolute shrinkage and selection operator) je korišten za selekciju feature-a na cijelom predikcijskom prostoru i na osnovu toga je odabrana PCA metoda jer ona obično ima najbolji performans.
- Izvršeno je kodiranje klasa gdje -1 vrijednost označava klasu *Pass*, a vrijednost 1 klasu *Fail*.
- Ovako kodirane varijable su dodane prethodnom "očišćenom" data setu.
- Nakon toga je novi data set podijeljen u dva data seta: training data set i test data set. Pri čemu je odabranu da training data set sadrži 80% obzervacija,a test data set 20% obzervacija.
- Odabrani su i algoritmi za klasifikaciju, te se izvršila validacija i usporedba tih algoritama koristeći mjere performanse, osjetljivost, specifičnost, tačnost, stepen greške koristeći k-fold cross validaciju.

Izgled finalnog data seta nakon pretprocesiranja i selekcije odgovarajućih feature-a (kolona) prikazan je na slici 4.1

:90	PC91	PC92	PC93	PC94	PC95	PC96	PC97	PC98	PC99	PC100	Response	yield
1.44466266	-0.81144435	0.34239663	0.18038388	-0.611953677	-1.26151552	1.957161887	1.398784431	-0.176644212	-1.297229090	1.16139347	-1	Pass
0.47497208	-1.53557547	0.15937739	-1.35312932	1.125939193	0.79660090	0.466181049	0.412906882	0.823042384	0.373486879	0.68424372	-1	Pass
-0.28604363	0.35205264	0.20373238	-0.67832824	1.626327853	2.33277082	-0.497353836	-0.756723022	0.692634272	-0.819930479	-2.08890905	-1	Pass
-0.87127733	1.05903351	0.20293569	0.82887482	-0.877634995	1.26983974	0.280946352	1.629463248	1.173893174	-0.974335737	-0.60848404	-1	Pass
-0.17633880	-1.54722511	0.08049044	0.87003488	-1.125491357	-1.69216513	0.031803353	-1.054301616	1.230888166	2.503772118	-0.63230487	-1	Pass
0.12964540	-0.68609114	-1.22636897	0.24978012	0.035225482	1.15528437	-0.768156620	-0.515195557	-1.020815637	1.636128641	0.24248569	-1	Pass
1.75661812	-0.86872233	0.64317124	-0.27958594	-0.390042873	0.35438437	0.219888764	0.575695278	0.128439659	0.242360773	-0.87175610	-1	Pass
1.63630384	-1.17957676	-0.72827021	-3.57046533	0.809258166	2.86406231	-0.096254874	0.428069955	1.267912928	0.724559403	-4.06166359	-1	Pass
1.95004391	0.54094440	0.22021519	1.70256538	-1.745220402	0.30446461	-0.521218729	0.190279724	0.135926645	0.605787871	-1.17632984	-1	Pass
-0.07604106	-1.84524958	2.05527625	2.16808710	0.829526377	-0.06190985	1.931715588	0.346154132	1.669238706	-0.047060939	-0.28551388	-1	Pass
-1.30407350	-1.65345325	-0.02865042	1.20930763	-0.534978407	-0.14744121	-0.219300872	-1.484776762	-0.168644767	-0.062577367	1.33489949	-1	Pass
0.10856585	1.56788547	0.69452315	0.96086197	-0.683498893	0.72820955	1.153463026	0.878153283	-0.203700270	0.572665207	0.64802049	-1	Pass
-0.71205312	-0.04821298	1.22488575	0.75482001	0.665433612	-0.56088377	0.647324685	-1.325571680	-0.099621289	-1.360103987	-1.64384540	-1	Pass
2.11849197	-0.97308461	0.36521053	-1.04274678	-0.009195305	1.97403261	0.760008893	1.361990285	0.367929560	0.629350739	1.36676755	-1	Pass
0.40898441	1.43626359	-0.21381304	-1.37237346	1.047538840	0.23670164	0.536311503	1.688119119	0.536399086	-0.007715793	-1.99441406	-1	Pass
-0.49514260	0.43607806	-0.44983377	0.44269958	-0.202281090	0.13528466	0.031600568	0.104129427	0.448411348	-0.182762743	1.21744340	-1	Pass

Slika 4.1 – Izgled finalnog data seta *secom_labels.data*

4.1 Stablo odlučivanja

Kako je ovaj algoritam klasificirao poluprovodnike koji su prošli odnosno pali test prikazano je na narednoj tabeli (slika 4.2)

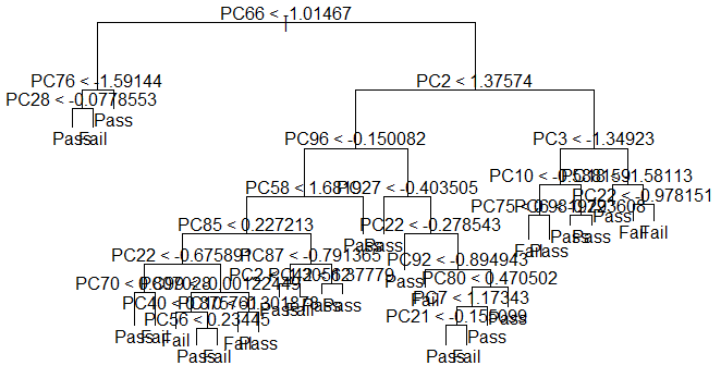
K-fold cross validacija je prikazana na slici 4.3 Nakon proračuna ovih vrijednosti za sve algoritme bit će kreirana konačna ocjena svih na kraju ovog poglavlja, te ćemo uočiti koji algoritam je najbolje uradio klasifikaciju. Izgled stabla odlučivanja je prikazan na slici 4.4. Nakon toga je pronađena optimalna veličina stabla

	Yield.test	
Tree.pred	Fail	Pass
Fail	3	68
Pass	18	225

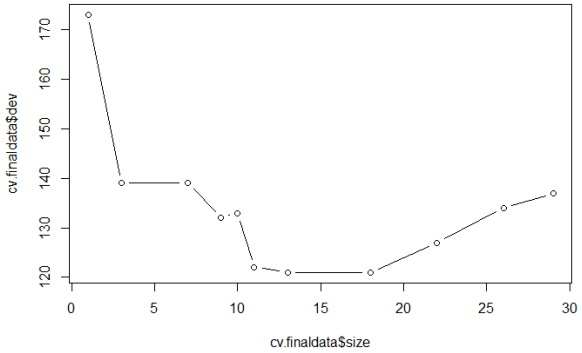
Slika 4.2 – Broj ispravno i neispravno klasificiranih poluprovodnika

```
> confusionMatrix(tree.pred, Yield.test)
Confusion Matrix and Statistics

          Reference
Prediction Fail Pass
      Fail    3   68
      Pass   18  225

               Accuracy : 0.7261
                 95% CI : (0.6732, 0.7747)
    No Information Rate : 0.9331
    P-Value [Acc > NIR] : 1

                  Kappa : -0.0424

 Mcnemar's Test P-Value : 1.265e-07

            Sensitivity : 0.142857
            Specificity : 0.767918
         Pos Pred Value : 0.042254
         Neg Pred Value : 0.925926
             Prevalence : 0.066879
         Detection Rate : 0.009554
   Detection Prevalence : 0.226115
      Balanced Accuracy : 0.455388

       'Positive' Class : Fail
```

Slika 4.3 – K-fold cross validacija stabla odlučivanja



Slika 4.4 – Izgled stabla odlučivanja



Slika 4.5 – Određivanje najbolje veličine stabla odlučivanja

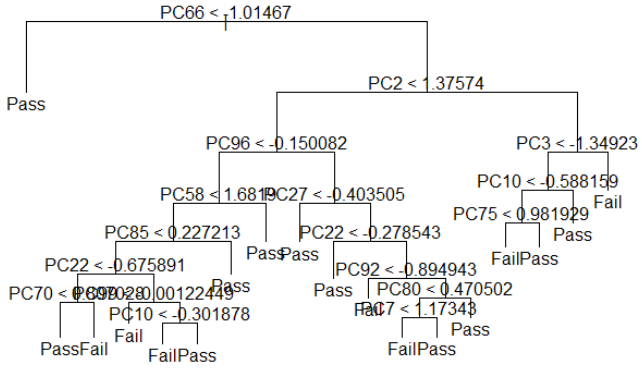
odlučivanja. Na slici 4.5 vidimo da je optimalna/najbolja veličina stabla 15-20.

K-fold cross validacija novog stabla odlučivanja je prikazana na slici 4.6 i njegov izgled na slici 4.7

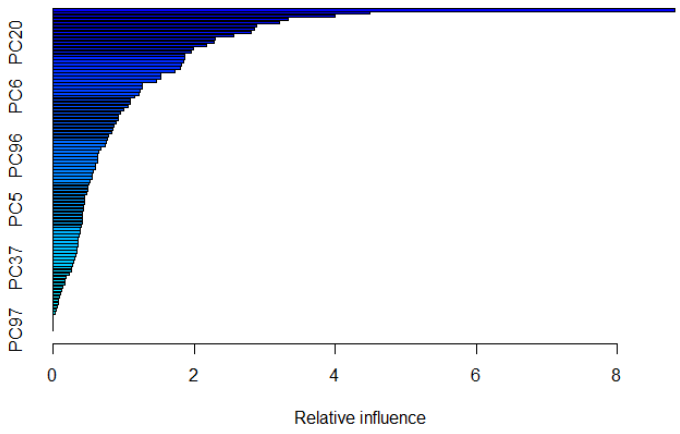
```
> confusionMatrix(prunetree.pred, Yield.test)
Confusion Matrix and Statistics

          Reference
Prediction Fail Pass
      Fail    5   65
      Pass   16  228

               Accuracy : 0.742
                 95% CI : (0.6899, 0.7895)
    No Information Rate : 0.9331
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0078

 Mcnemar's Test P-Value : 9.643e-08

            Sensitivity : 0.23810
            Specificity : 0.77816
         Pos Pred Value : 0.07143
         Neg Pred Value : 0.93443
             Prevalence : 0.06688
         Detection Rate : 0.01592
   Detection Prevalence : 0.22293
      Balanced Accuracy : 0.50813

       'Positive' Class : Fail
```

Slika 4.6 – K-fold cross validacija novog stabla odlučivanja

Određeno je i koja varijabla tj. feature najviše utiče na to da li će poluprovodnik pasti ili proći test, to je prikazano na slici 4.8 Vidimo da komponenta PC20 tj. feature 20 najviše utiče na to da li će poluprovodnik proći ili pasti test. Ostale komponente koje na to utiču su: 6, 96, 5, 37, 97, itd.



Slika 4.7 – Izgled novog stabla odlučivanja



Slika 4.8 – Uticaj feature-a na klasifikaciju

4.2 One-class SVM algoritam

Pretprocesiranje

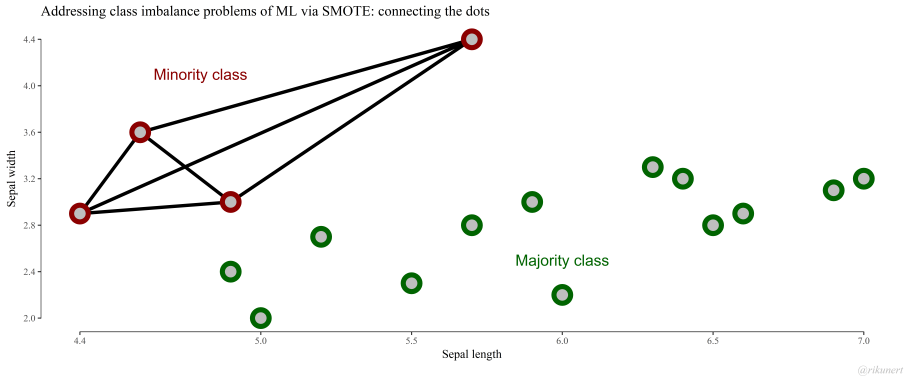
Dati dataset ima 1,567 rekorda i 591 atribut. Odnos izmedju dvije klase je neravnomjerno rasporedjen i iznosi **1:14**. Sve varijable su numeričke. Prvo, sve kolone u datasetu čija je varijansa nula (tj. čije su vrijednosti neka konstanta) su eliminisane i na taj način je smanjena dimenzionalnost problema (otpisano je preko 100 atributa). Drugo, sve "NaN" vrijednosti su zamijenjena median vrijednošću odgovarajuće kolone.

Problem nebalansiranog dataseta

Problem nebalansiranog dataseta se može riješiti na više načina (Under/Down-sampling, oversampling - korišten prilikom realizacije stabla odlučivanja, itd). Generalno, postoje 4 načina adresiranja ovog problema:

- Sintetiziranje instanci minorne klase
- Oversampling minorne klase
- Undersampling većinske klase
- Podešavanje funkcije troška tako da se debalans uzme u obzir

Prvi način predstavlja vještačko ubacivanje novih instanci u dataset. SMOTE sintetiziranje je prikazano na slici 4.9.



Slika 4.9 – SMOTE sintetiziranje

Na slici 4.9 vidimo da su crvene tačkice minorna klasa. SMOTE algoritam poveže instance minorne klase i na tim "putevima", ovisno o parametrima, rendomistično kreira nove instance te minorne klase. Na taj način stvara se balans u datasetu.

Kakogod, ovaj problem ćemo pokušati izbjeći implementacijom *One-class SVM* algoritma.

Objašnjenje algoritma

Tradicionalno, mnogi problemi klasifikacije pokušavaju riješiti problem klasifikacije u slučaju dvije ili više klasa. Cilj aplikacije mašinskog učenja je razlikovati testne podatke između nekoliko klasa, koristeći trening podatke. Ali šta ako imate samo podatke jedne klase i cilj je testirati nove podatke i saznati da li su podaci slični ili ne kao trening podaci? Metoda za ovaj zadatak, koja je stekla veliku popularnost u posljednje dvije decenije, je One-Class Support Vector Machine. Ovaj algoritam je zapravo nadogradnja poznatog algoritma SVM (Support Vector Machine) koji je obrađen na laboratorijskoj vježbi, tako da osnovni koncept SVM-a vrijedi i ovdje.

Ovaj algoritam se često koristi za *novelty detection*. Jedan od prvih radova na ovu temu je i ***Support Vector Method for Novelty Detection***, autora *Bernhard Scholkopf, Robert Williamsonx, Alex Smolax, John Shawe-Taylory, John Plat*. Takodjer, SVM je objašnjen i u radu **An overview of classification algorithms for imbalanced**, autora *Vaishali Ganganwar*.

Analiza rezultata

Za One-class SVM klasifikator je najbitnija mjera Recall i Accuracy, koji su u ovom slučaju isti. U kodu je implementirana funkcija za računanje tačnosti (accuracy), a recall je implementiran na osnovu *confusion matrix* tabele.

$$Recall = Accuracy = 98.97611\%$$

Svakako, performansa algoritma se može dodatno poboljšati ovisno o pretprocesiranju podataka. Na primjer, normalizacija (Z-score ili min-max) podataka bi znatno ubrzala algoritam, zatim dodatno smanjenje dimenzionalnosti (na primjer, na osnovu male varijanse), itd.

4.3 Usporedba korištenih algoritama

Kako imamo neravnomjernu raspodjelu klasa onda je kao ocjena performanse algoritma bolja osjetljivost i specifičnost nego greška (error rate). Ovo proizilazi iz činjenice da tačnost u ovakvim slučajevima obično bude jako visoka, a desi se da osjetljivost bude 0%. Model koji ima dobar kompromis između osjetljivosti i specifičnosti (F1 score) je odabran kao konačni model jer želimo da tačno predvidimo sve poluprovodnike koji su pali test minimizirajući mogućnost klasificiranja onih koji su prošli test u klasu *Fail*. Vidimo da je One-Class SVM algoritam koji ima znatno bolju performansu prilikom rješavanja ovog problema. To možemo zaključiti na osnovu mjere performanse *Recall* koja nam govori koliko je ispravno klasificiranih rekorda. Kod One-class SVM-a, recall je veći od 98%, dok je kod stabla odlučivanja 0%.

5 Drugi pristup rješavanju datog problema - detekcija anomalija

Postoji mnogo tehnika detekcije anomalije, kao što su: klasifikacijski bazirani, pristup baziran na neuronskim mrežama, klastering bazirani, itd. Prvo, odabrao sam klastering bazirani algoritam - DBSCAN. Ovo je veoma moćan algoritam baziran na gustoći podataka u hiperprostoru. Ideja je bila da se zapravo svi ispravni poluprovodnici nalaze blizu jedan-drugog u nekom hiperprostoru. Ipak, rezultati su bili jako loši. Puno bolje rezultate je opet dao One-Class SVM algoritam. U sljedećim tabelama (5.1 i 5.2) prikazana je peroformansa DBSCAN algoritma, kao i One-Class SVM algoritma (nakon dodatnog pretprocesiranja podataka).

	Precision	Recall	F1-score	Support
Outlier	0.07	0.51	0.13	104
Not outlier	0.94	0.54	0.69	1463
Avg/total	0.88	0.54	0.65	1567

Slika 5.1 – Izvještaj klasifikacije korištenjem DBSCAN algoritma

	Precision	Recall	F1-score	Support
Outlier	0.58	1.00	0.74	104
Not outlier	1.00	0.95	0.97	1463
Avg/total	0.97	0.95	0.96	1567

Slika 5.2 – Izvještaj klasifikacije korištenjem One-class SVM algoritma

Poglavlje 3

Zadatak 2 - Klastering

1 Postavka zadatka

a) Potrebno je da specificitrate hipotezu koju želite riješiti tehnikom klasteringa, a koja je zasnovana na data setu po Vašem izboru, ili sa nekog od repozitorija (npr. UCI Machine Learning Repository ili Kaggle Datasets) ili iz realnog sektora. Dokažite da je Vaš odabrani skup podataka pogodan za primjenu klastering tehnika (procjenom klastering tendencije). Na osnovu specificirane hipoteze, odaberite dva adekvatna algoritma za rješavanje iste, od kojih barem jedan niste radili na predavanjima ili vježbama. Prilagodite Vaš skup podataka odabranim algoritmima primjenom metoda vizualizacije, deskriptivne statistike i tehnika za čišćenje i transformaciju podataka. (3 boda)
b) U programskom jeziku R, implementirajte odabrane algoritme na Vaš data set. Implementirajte dodatnu funkciju pomoću koje ćete evaluirati Vaše klastering algoritme pomoću mjera za kvalitet klastera, npr. SSE, Shioulette itd. Obrazložite dobijene rezultate. Ukoliko postoji razlika u performansama dva algoritma, objasnite zašto se to dešava. (4 bodova)

2 Postavka problema

Za zadatak klasterizacije odabran je data set sa UCI Machine Learning Repository repozitorija koji se može pronaći ovdje. Ovaj data set *(iris.csv)* sadrži atribute koji opisuju tri različite vrste cvijeća. Potrebno je na osnovu raznih kriterija npr. dužinu i širinu latice ili dužinu i širinu čašičnih listova napraviti klastere koje razvrstavaju svaku instancu u odgovarajući klaster koji označava kojoj vrsti cvijet pripada.

3 Opis data seta

Data set se sastoji od 150 rekorda koji su opisani sa 5 atributa - *sepal_length*, *sepal_width*, *petal_length*, *petal_width*, *species*. Posljedni atribut, *species*, predstavlja vrstu cvijeta i postoje 3 vrijednosti: *Iris-setosa*, *Iris-versicolor*, *Iris-virginica*.

Svaka od 3 vrste cvijeta sadrži po 50 rekorda. Data set ne sadrži vrijednosti koje nedostaju. Atributi data seta su sljedeći:

1. **petal_length**: predstavlja dužinu latice u cm.
2. **petal_width**: predstavlja širinu latice u cm.
3. **sepal_length**: predstavlja dužinu čašičnih listova u cm.
4. **sepal_width**: predstavlja širinu čašičnih listova u cm.

Zanimljivo je spomenuti da je ovaj data set poznati set podataka koji se zove **Fisher-ov iris** (irisov cvijet). Ovaj data set podataka je prikupljen od strane britanskog statističara i biologa Ronalda Fishera-a 1936. godine. Fisher-ov iris se korisiti upravo za probleme klasifikacija i klasteringa tj. predstavlja tipični testni set za mnoge klasifikacijske/klastering tehnike u mašinskom učenju. Tri vrste irisovog cvijeta su prikazane na sljedećim slikama 3.1, 3.2 i 3.3

Slika 3.1 – Iris setosa

Slika 3.2 – Iris versicolor

Slika 3.3 – Iris virginica

4 Relizacija zadatka u programskom jeziku R

4.1 Hijerarhijska klasterizacija

Opis algoritma

Hijerarhijska klasterizacija predstavlja algoritam u kojem je glavna akcija uzastopno spajanje dva najbliža klastera u jedan veći klaster. Ovaj proces se sastoji od sljedećih koraka:

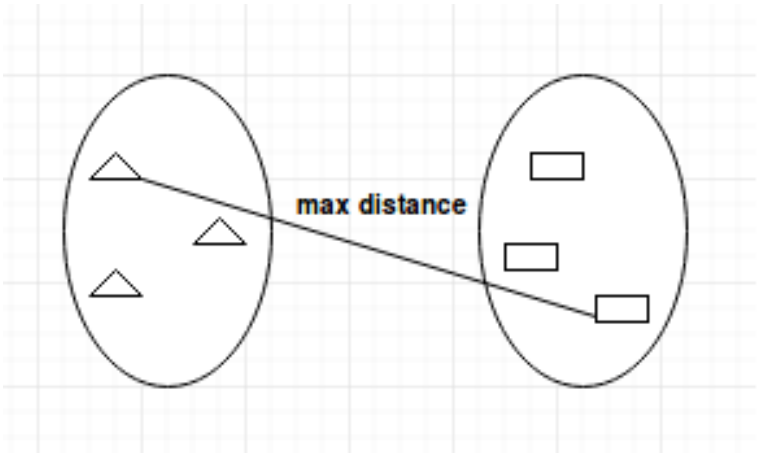
1. Proces započinje sa računanjem udaljenosti između svakog para obzervacijskih tačaka i smještanje tih vrijednosti u matricu distanci.
2. Svaka tačka se postavlja u zaseban klaster.
3. Započinje se proces spajanja najbližih parova tačaka na osnovu udaljenosti od matrice distanci i kao rezultat raste broj klastera za 1.
4. Nakon toga se ponovo izračunava udaljenost od novog klastera do starih klastera i smješta se u novu matricu distanci.
5. Ponavljaju se koraci 2 i 3 dok se svi klasteri ne spoje u jedan klaster.

Postoji više načina na koje se može izračunavati udaljenost između dva klastera. U ovom seminarskom radu bit će korištene dvije metode: *complete* i *average* metoda.

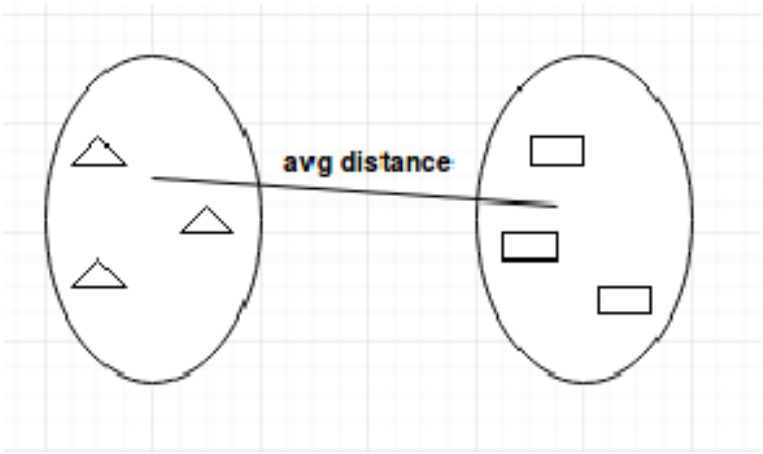
Complete metoda izračunava maksimalnu udaljenost između klastera prije spajanja, što se može vidjeti na slici 4.1.

Average metoda izračunava srednju udaljenost između dva klastera, što je prikazano na slici 4.2.

U hijerarhijskoj klasterizaciji koriste se posebni dijagrami nazvani dendrogrami, jer se vrši kategorizacija objekata u hijerarhiju sličnu grafiku stabla. Na slici 4.3 prikazan je primjer jednog dendrograma, gdje je udaljenosti dijeljenja ili spajanja (nazvana visina) prikazana na y-osi.



Slika 4.1 – Complete metoda računanja udaljenosti između klastera

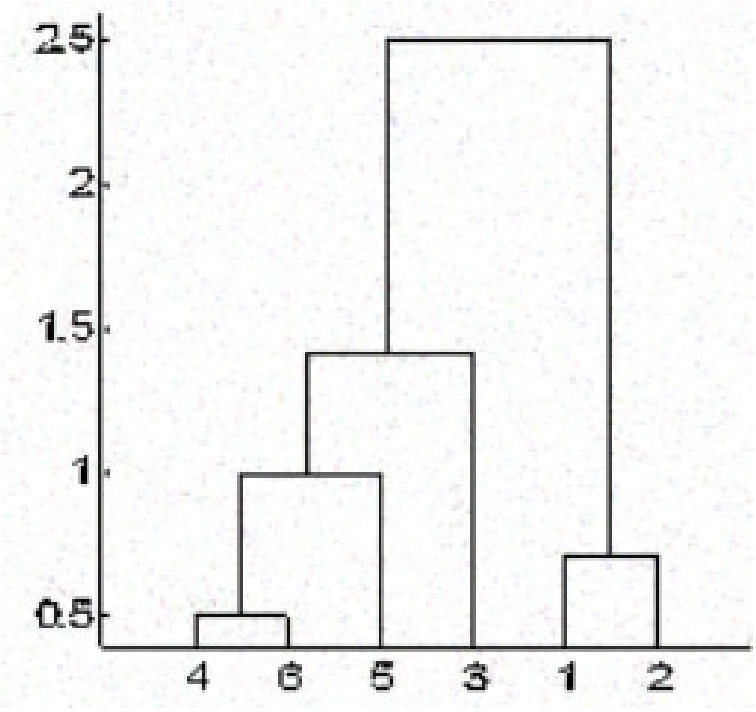


Slika 4.2 – Average metoda računanja udaljenosti između klastera

Na prethodnoj slici, prvo se kombinuju 4 i 6 u jedan klaster, nazvat ćemo ga klaster 1, pošto su bili najbliži po udaljenosti a zatim slijede tačke 1 i 2, koje spadaju u klaster 2. Nakon toga se 5 spaja u klaster 1 te 3. Na kraju se ova dva klastera spajaju u jedan klaster i ovdje se proces klasterizacije završava.

Završetak procesa hijerarhijske klasterizacije zavisi od znanja koje imamo u setu podataka. Naprimjer, ukoliko vršimo klasterizaciju fudbalskih igrača na osnovu njihove pozicije na terenu, što će predstavljati njihove koordinate za računanje udaljenosti, onda već na početku znamo da mogu biti samo dva tima igrača koji igraju fudbalski meč. Za određivanje broj klastera mogu se koristiti i dodatne informacije koje možemo dobiti kao što je Silhouette dijagram, numeričke mjere kao što je Dunn-ov index, Hubertova gama itd. Prethodna analiza nam daje informaciju o varijaciji greške u zavisnosti od broja klastera i možemo odabrati broj klastera k tako da greška bude minimalna.

Recimo da imamo set podataka koji predstavlja tri vrste pšenice na osnovu određenih atributa, te da želimo klasterizirati taj data set. Na slici 4.4 možemo vidjeti ilustrativan prikaz rezultata klasterizacije gdje imamo 3 klastera koja predstavljaju 3 vrste pšenice, prikazanih različitim bojama.



Slika 4.3 – Primjer dendrograma

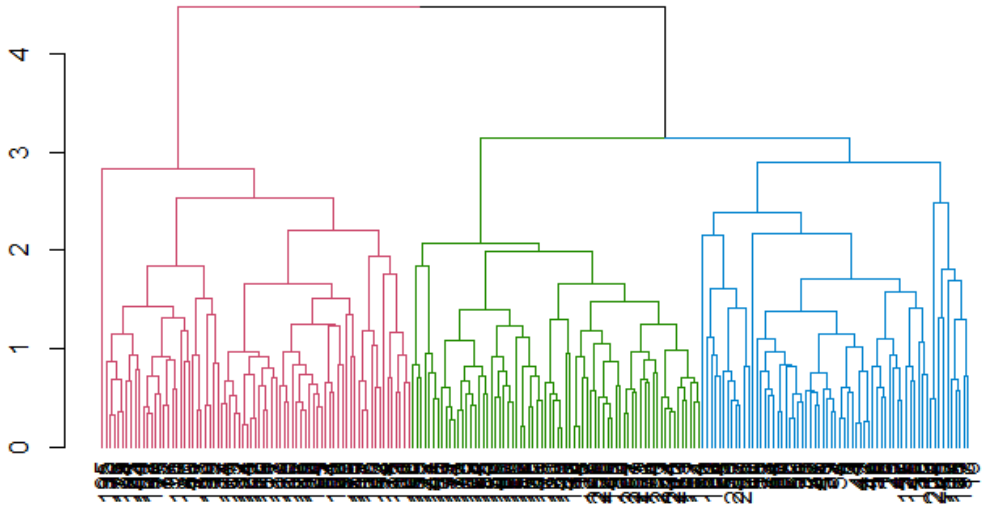
Glavna prednost hijerarhijske klasterizacije je što na osnovu dendrograma možemo zaključiti kako se podklasteri međusobno povezani i koliko su međusobno udaljene tačke.

Stručni/naučni radovi koji koriste hijerarhijsku klasterizaciju:

- Hierarchical Clustering Algorithms for Document Datasets - Rad koji analizira različite postavke hijerarhijske klasterizacije za pružanje intuitivne navigacije i mehanizama pretraživanja tako što se veliki broj informacija organizira u manji broj značajnih klastera. Ovaj način klasterizacije omogućava kreiranje značajnih hijerarhija od velikih kolekcija dokumenata za interaktivnu vizualizaciju koja je konzistentna, prediktivna i sa različitim nivoima detalja.

- Hierarchical clustering of self-organizing maps for cloud classification Ovaj rad prezentira novu metodu za segmentaciju multispektralnih satelitskih slika. Hijerarhijska klasterizacija se koristi kako bi se svaki piksel preuzeo labelu od najbližeg vektora, odnosno da bi se izvršilo povezivanje grupe piksela u klastere.

Na osnovu prethodnog, možemo zaključiti da je glavna prednost i upotreba hijerarhijske klasterizacije u aplikacijama gdje želimo imati informaciju o podklasterima te kako su ti podklasteri povezani u veće klastere.



Slika 4.4 – Primjer hijerarhijske klasterizacije na dummy setu podataka

Realizacija algoritma

Na osnovu prva 4 atributa ćemo vršiti klasterizaciju svakog od rekorda u jedan od 3 klastera koji predstavljaju vrstu cvijeta. Dakle, za realizaciju klasterizaicije, nećemo uzimati u obzir posljednju tabelu, pošto je to vrsta cvijeća koju trebamo odrediti procesom klasterizacije.

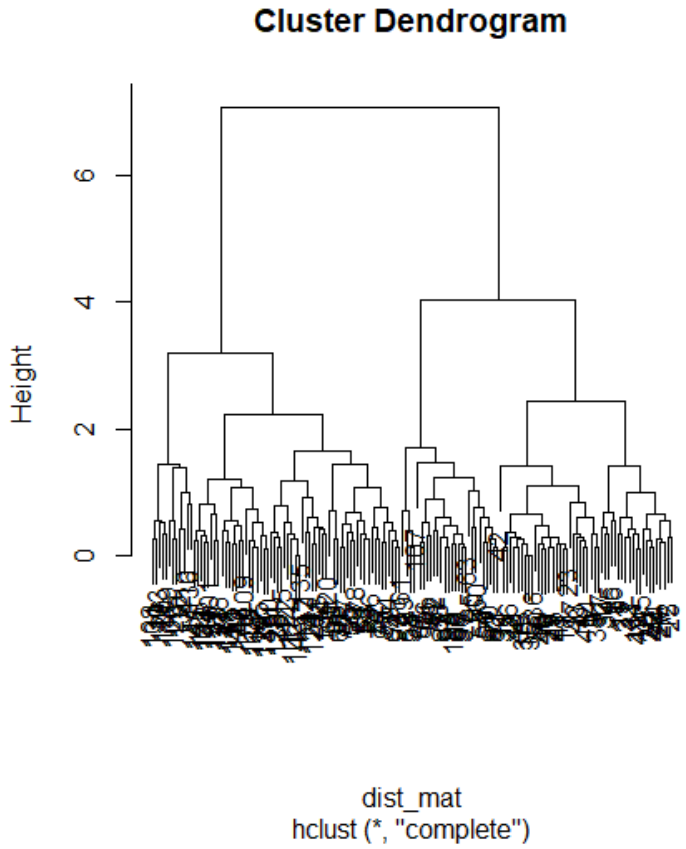
Kao metodu za povezivanje hijerarhijske klasterizacije iskoristit ćemo više funkcija, te ćemo analizirati poklapanje sa stvarnim klasterima. Nije potrebno vršiti predprocesiranje dataseta kao ni normalizaciju, jer su vrijednosti 4 atributa unutar istog opsega, te ne postoje nedefinisane vrijednosti.

Prva metoda koja je iskorištena je *complete* metoda. Na osnovu hijerarhijske klusterizacije prilikom podjele dataseta u 3 klastera, dobili smo dijagram prikazan na slici 4.5 Ukoliko na dijagramu odaberemo visinu 3, možemo primjetiti da imamo tri klastera koja se u nižim visinama dalje dijele.

Na slici 4.6 je usporedba ovako dobivene klasterizacije sa vrstama cvijeća iz početnog dataseta. Primjetimo da u drugom klasteru imamo skoro sve vrijednosti Iris-virginica ali imamo veliki broj vrijednosti sa vrstom Iris-versicolor. Dakle, dobivena klasterizacija ne daje zadovoljavajuće rezultate pa ćemo iskoristiti metodi *average*.

Na slici 4.7 prikazan je dendrogram dobiven pomoću metode *average*, gdje je ponovljen isti postupak kao i za metodu *complete*.

Ukoliko ponovo usporedimo dobivene tačke u klasterima sa vrstama cvijeća iz početnog dataseta, što je prikazano na slici 4.8, primjetimo da se sada u klasteru dva nalazi svih 50 rekorda vrste Iris-versicolor i određen broj iz vrste Iris-virginica. Ovaj broj je manji nego što je to slučaj sa klasterizacijom dobivenom metodom *complete*.



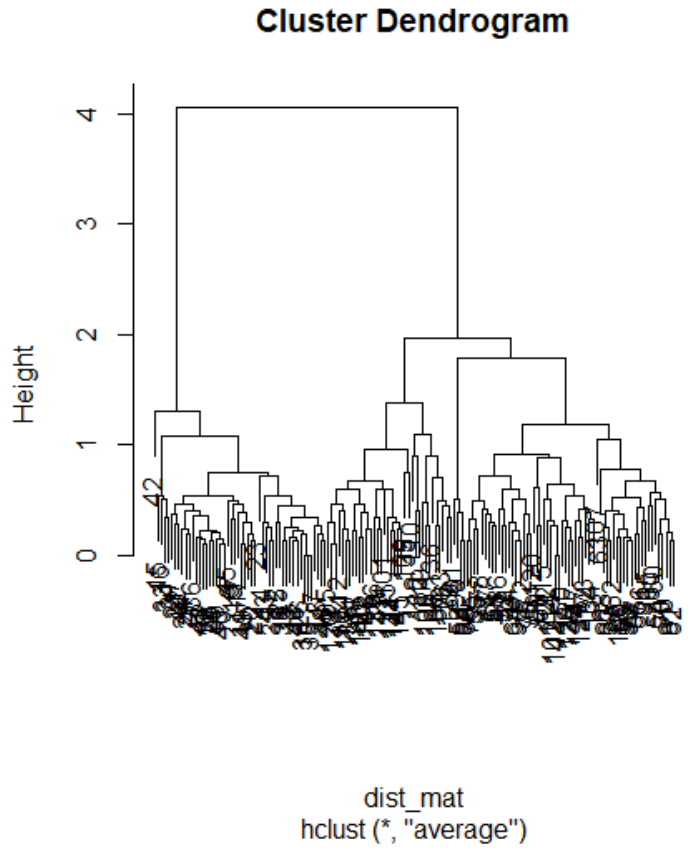
Slika 4.5 – Dendrogram za metodu complete

```
> table(clusterCut, iris$species)

clusterCut Iris-setosa Iris-versicolor Iris-virginica
         1          50               0              0
         2           0              23             49
         3           0              27              1
```

Slika 4.6 – Usporedba klasterizacije sa metodom complete i početnog dataseta

U 2D ravni nacrtat ćemo tačke koje predstavljaju atribute cvijeta, tako da odaberemo parove karakteristika (*sepal_length*, *sepal_width*) i (*petal_length*, *petal_width*), što je prikazano na slici 4.9 gdje je na lijevoj strani prikazan (*petal_length*, *petal_width*), a na desnoj par (*sepal_length*, *sepal_width*). Na slikama je bojama označena svaka vrsta cvijeta dobivena iz početnog dataseta.

Sa prethodne slike možemo zaključiti da je položaj tačaka para vrijednosti (*sepal_length* , *sepal_width*) znatno lakši za klasterizaciju jer u određenom području nema miješanja velikog broja vrijednosti iz različitih klastera, kao što je to slučaj na lijevoj slici. Upravo zbog toga se kod prethodne klasterizacije koju smo izvršili, javio problem relativno velikog broja vrijednosti jedne vrste (klastera) unutar drugog klastera (14 vrijednosti Iris-virginica unutar klastera za Iris-versicolor). Analizirat ćemo klasterizaciju ukoliko uzmemo u obzir samo



Slika 4.7 – Dendrogram za metodu average

```
> table(clusterCut, iris$species)

clusterCut Iris-setosa Iris-versicolor Iris-virginica
         1          50               0              0
         2           0              50             14
         3           0               0             36
```

Slika 4.8 – Usporedba klasterizacije sa metodom average i početnog dataseta

atribute *sepal_length* i *sepal_width*, pošto nam druga dva atributa narušavaju tačnost klasterizacije. Možemo očekivati znatno bolju klasterizaciju, jer je miješanje vrijednosti na jednom području znatno manje bez ovog para vrijednosti.

Dakle odabrali smo samo 3 i 4 kolonu što odgovara vrijednostima *sepal_length* i *sepal_width*. Rezultati ove klasterizacije prikazani su na slici 4.18 gdje vidimo da je broj tačaka u pogrešnom klasteru znatno manji nego što je to bio slučaj kod prethodnih klasterizacija.

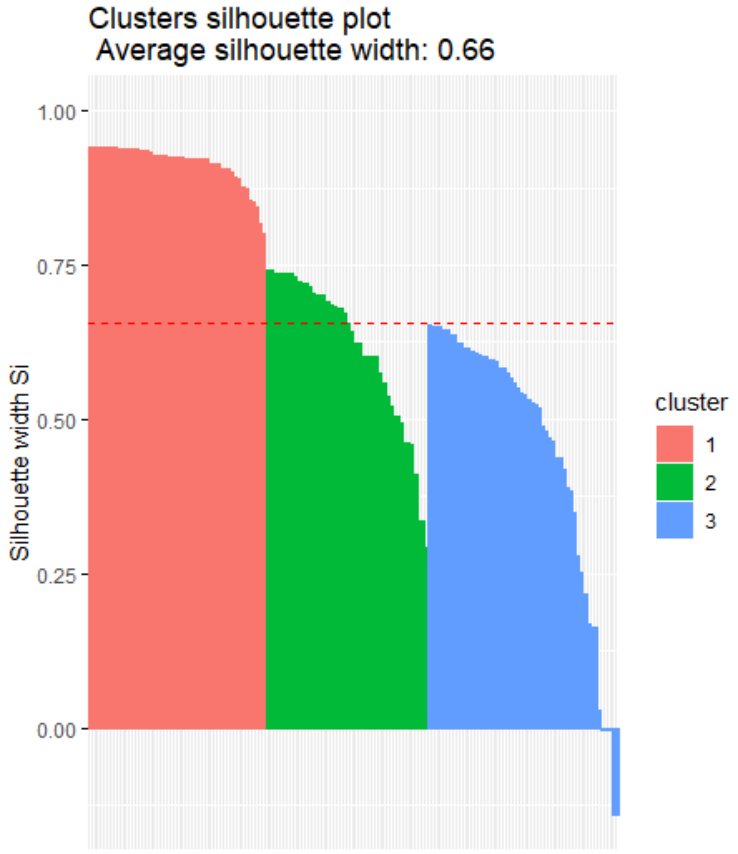
Na slici 4.11 prikazan je Silhouette dijagram koji nam daje informaciju za validaciju rezultata klasteringa. Na osnovu ovog dijagrama možemo zaključiti koliko su elementi jednog klastera međusobno slični. Kao što možemo vidjeti elementi u klasteru 1 su poprilično slični jedni drugima, dok se određena razlika javlja u klasterima 2 i 3, što smo mogli i očekivati s obzirom na postojanje određenog

Slika 4.9 – Parovi atributa sa označenim vrstama cvijeta

```
> table(clusterCut, iris$species)

clusterCut Iris-setosa Iris-versicolor Iris-virginica
         1          50               0              0
         2           0              45              1
         3           0               5             49
```

Slika 4.10 – Usporedba klasterizacije na osnovu dvije kolone data seta

broja elemenata iz klastera 3 u klasteru 2. Također primjetimo da određen broj elemenata ima negativnu vrijednost, što nam govori da su ti elementi pogrešno klasterovani. Informacija o elementima koji su pogrešno klasterovani prikazana je na slici 4.12.



Slika 4.11 – Silhouette dijagram

```
> fviz_silhouette(res)
  cluster size ave.sil.width
1       1   50          0.92
2       2   46          0.62
3       3   54          0.45
>
```

Slika 4.12 – Elementi koji su pogrešno klasterovani

4.2 K-means

Prvo je primjenom klastering tendencije ustanovljeno da li se dati data set može koristiti za klastering koristeći Hopkinsovu statistiku i vizualni pristup. Ukoliko je vrijednost Hopkinsove statistike blizu nule (mnogo manje od 0.5), onda možemo zaključiti da je data set značajno klasterabilan. Za ovaj data set ova vrijednost je 0.1712254 pa zaključujemo da je data set itekako klasterabilan. U nastavku su objašnjeni koraci kako je implementiran K-means u R-u.

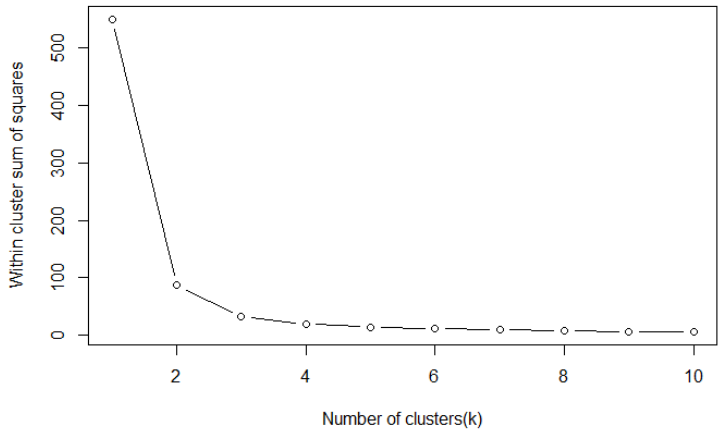
- Prvo je urađeno pretprocesiranje data seta. Kako je klastering nenadzirano učenje, nije nam potrebna labela klase tj. *species* tokom izvršavanja algo-

ritma, pa je ona otklonjena. Nije urađen neki tip normalizacije niti popunjavanje i uređivanje podataka kako podaci ne sadrže nedostajuće vrijednosti, a i sve vrijednosti su predstavljene u centimetrima.

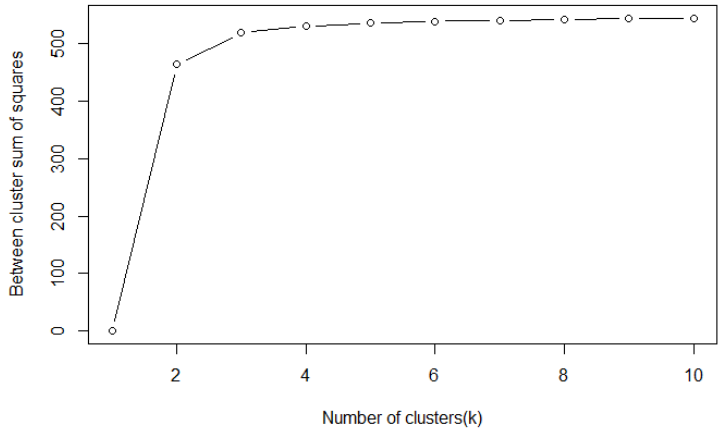
- Određen je optimalan broj klastera koristeći koheziju i separaciju. Kohezija određuje koliko su slični objekti unutar jednog klastera, dok separacija određuje koliko je jedan klaster dobro odvojen od drugih klastera tj. koliko je drukčiji. Kohezija je **wss** (eng. cluster sum of squares) within vrijednošću. Traži se onaj broj klastera gdje je vrijednost wss najmanja (optimalna). (slika 4.13) Separacija je mjerena **bss** (between cluster sum of squares) vrijednošću. Traži se onaj broj klastera gdje je bss vrijednost najveća (slika 4.14). Dobije se da je broj klastera 3! Petal length i petal width pokazuju tri klastera pa ćemo raspodjelu cvjetova u tri klastera raditi na osnovu ova dva atributa.

- Da bismo validirali klastering korišten je Silhouette koeficjent. Ovaj koeficjent predstavlja mjeru kombinacije i kohezije i separacije. Obično ima vrijednost između 0 i 1. Što je bliže jedinici to bolje, za vrijednost 1 algoritam je svaku instancu stavio u odgovarajući klaster. Visoka vrijednost koeficjenta indicira da je objekt vrlo sličan svom klasteru, a vrlo se razlikuje od susjednih klastera. Na slici 4.15 je prikazan Silhouette dijagram za K-means algoritam. Prosječna vrijednost je 0.69, pa je algoritam sa skoro 70% tačnosti smjestio svaki objekat u odgovarajući klaster. Možemo primjetiti da postoji i negativne vrijednosti na Silohouette dijagramu što znači da su ti elementi definitivno pogrešno klasterizovani. Informacija o tome koji su elementi pogrešno klasterovani (na osnovu dvije kolone data seta: petal length i petal width) prikazana je na slici 4.16 i 4.18. Vidimo da je 6 objekata pogrešno klasterovano i to za cvijet oblika versicolor i virginica, dok su objekti koji pripadaju cvijetu setosa dobro svi klasterovani.

- Na kraju je dat finalni k-means model (slika 4.17)

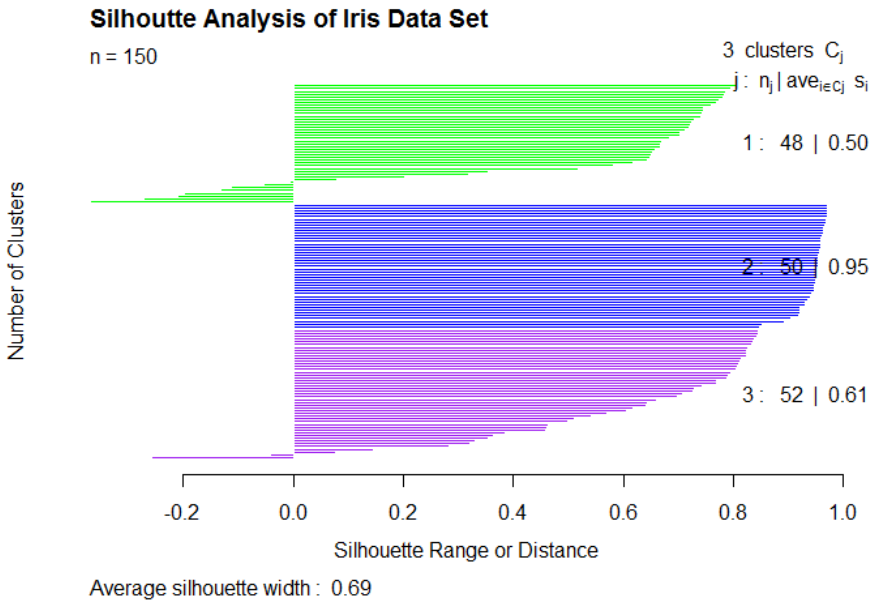


Slika 4.13 – Wss vs. broj klastera



Slika 4.14 – Bss vs. broj klastera

Raspodjela orginalnog data seta, Petal length i Petal width po klasama, species, slika 4.19 Raspodjela cvjetova preko čašičnih listova (Petal length i Petal width) u tri klastera prikazana je na slici 4.20.



Slika 4.15 – Silhouette dijagram

	Iris-setosa	Iris-versicolor	Iris-virginica
1	0	2	46
2	50	0	0
3	0	48	4

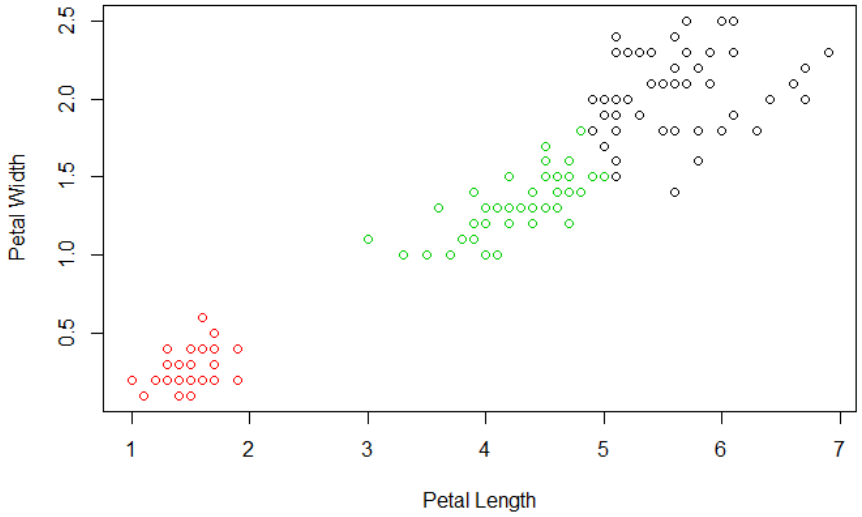
Slika 4.16 – Usporedba klasterizacije na osnovu dvije kolone data seta

```
K-means clustering with 3 clusters of sizes 48, 50, 52

Cluster means:
  petal_length petal_width
1     5.595833    2.037500
2     1.464000    0.244000
3     4.269231    1.342308

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
 [57] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1
[113] 1 1 1 1 1 1 1 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 16.29167  2.03840 13.05769
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

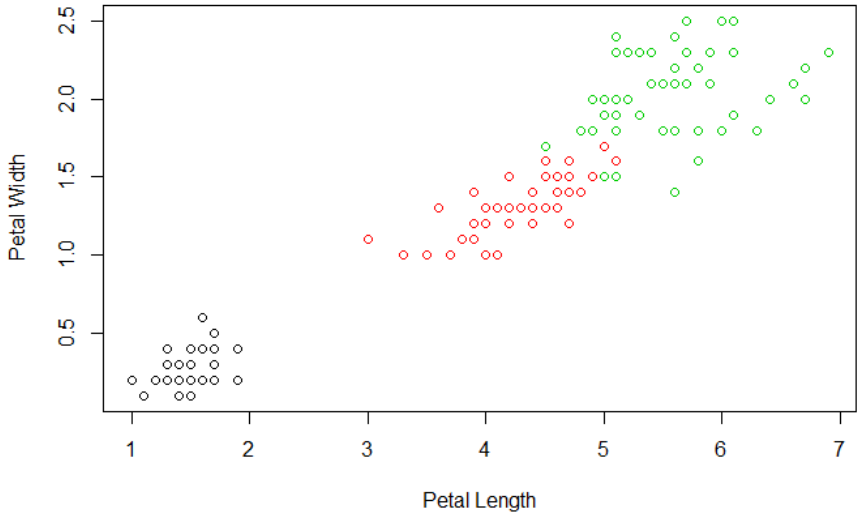
Slika 4.17 – Finalni K-means model

```
> fviz_silhouette(sil)
  cluster size ave.sil.width
1       1   48          0.50
2       2   50          0.95
3       3   52          0.61
```

Slika 4.18 – Informacija o pogrešno klasterovanim elementima



Slika 4.19 – Raspodjela Petal length i Petal width u orginalnom data setu po klasama/species



Slika 4.20 – Raspodjela Petal length i Petal width u tri klastera metodom k-means

4.3 Usporedba korištenih algoritama

Oba korištena algoritma za klastering nisu dali idealne rezultate ako posmatramo Silhouette koeficjent koji smo koristili pri validaciji. Ipak k-means algoritam daje bolje rezultate, tj. prosječna Silhouette širina je za k-means 0.69., dok je prosječna Silhouette širina kod algoritma hijerarhijske klasterizacije je 0.66. Međutim,

također se može uočiti da su oba algoritma pogrešno klasificirali 6 instanci koje pripadaju klasama versicolor i virginica. Bitno je spomenuti neke i općenite razlike između ova dva algoritma. Tako naprimjer, k-means algoritam može da operira nad velikim podacima dok hierarhijski ne može. Ovo slijedi iz činjenice da je vremenska kompleksnost k-means algoritma linearna dok je kod hijerarhijskog algoritma kvadratna. K-means klastering, s obzirom da počinjemo sa random brojem klastera, ukoliko pokrenemo algoritam više puta, rezultati se mogu razlikovati, dok su rezultati kod hijerarhijskog algoritma reproducibilni. K-means se koristi uglavnom kada je oblik klastera sferičan (krug u 2D ili sfera u 3D), kao što je slučaj u primjeru sa iris cvjetovima. K-means zahtijeva unaprijed poznavanje broja klastera, dok denogram kod hijerarhijskog algoritma služi da uočimo taj broj.

Poglavlje 4

Zaključak

Može se zaključiti da su algoritmi mašinskog učenja primjenjivi u svakodnevnom životu (kako u privatnom životu tako i u poslovnom svijetu), te predstavljaju temelj inteligentnih sistema. Inteligentnost sistema koje napravi čovjek ovise u mnogo čemu od kreativnosti koju ima čovjek - projektant tog sistema. Također, ne postoji generalni pristup rješavanja ML problema, već odabir algoritma striktno ovisi o problemu koji se rješava što smo pokazali i u ovom radu.