Istraživanje podataka 1 - vežbe 9, 2020.

1 Kvalitet klasterovanja

1.1 Silueta koeficijent, eng. Silhouette coefficient

Silueta koeficijent je mera koliko su instance grupisane sa instancama koje su slične njima samima. Prvo se silueta koeficijent računa za svaku instancu po formuli

$$s = \frac{b - a}{\max(a, b)}$$

gde je

- a - prosečno rastojanje između instance i ostalih instanci u istom klasteru
- b - prosečno rastojanje između instance i svih instanci iz najbližeg susednog klastera

Silueta koeficijent za ceo skup je prosečna vrednost koeficijenata za pojedinačne instance. Vrednost silueta koeficijenta je između [-1,1] pri čemu je

- -1 za neispravno grupisanje
- +1 za gusto grupisanje

Vrednost koeficijenta je veća kada su klasteri gusti i dobro razdvojeni.

1.2 Silueta koeficijent u biblioteci scikit-learn programskog jezika Python

Za računanje silueta koeficijenta koristi se funkcija **sklearn.metrics.silhouette_score** koja ima parametre:

- X - instance nad kojima je izvršeno klasterovanje
- *labels* - oznake klastera kojima pripadaju instance
- *metric* - metrika za računanje rastojanja između dve instance; default='euclidean'

2 Hijerarhijsko klasterovanje

Hijerarhijsko klasterovanje je jedna od najstarijih i široko korišćenih metoda. Postoje dva pristupa u hijerarhijskom klasterovanju:

- *sakupljajuće*: inicijalno je svaka instanca zaseban klaster. U svakom koraku se spajaju dva najbliža/najsličnija klastera sve dok sve instance ne pripadaju istom (jednom) klasteru. Kod sakupljajućeg hijerarhijskog klasterovanja potrebno je definisati kako se računa bliskosti dva klastera.
- *razdvajajuće*: inicijalno sve instance pripadaju jednom klasteru. U svakom koraku se jedan klaster deli na dva dela sve dok ne ostanu klasteri sa po jednom instancom. Kod razdvajajućeg hijerarhijskog klasterovanja potrebno je definisati kako se bira klaster nad kojim će se izvršiti deljenje i kako podeliti klastera na dva dela.

2.1 Algoritam hijerarhijsko sakupljajuće klasterovanje

Koraci hijerarhijskog sakupljajućeg klasterovanja su opisani u algoritmu 1.

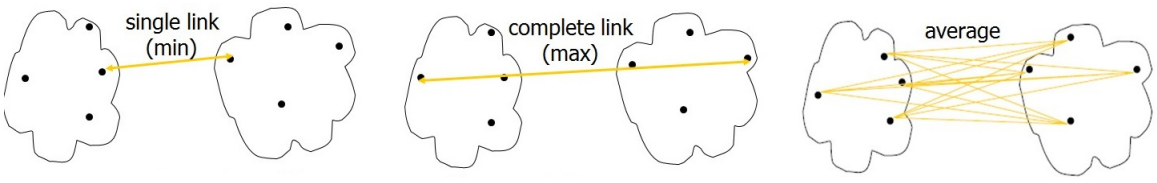
Algoritam 1 Hijerarhijsko sakupljajuće klasterovanje

1: Svaka instanca je zaseban klaster. Računa se matrica bliskosti klastera, tj. matrica bliskosti instanci.
2: Spajaju se dva **najbliža** klastera.
3: Ažurira se matrica bliskosti klastera.
4: Ponavljaju se koraci 2 i 3 dok ne ostane jedan klaster.

Pri primeni hijerarhijskog sakupljajućeg klasterovanja, potrebno je izabrati meru za računanje bliskosti dve instance (npr. euklidsko rastojanje, kosinusna sličnost ...), kao i kako se računa bliskost dva klastera. Za određivanje bliskosti klastera mogu se koristiti veze:

- najbolja (min, single) veza - bliskost dva klastera je jednaka bliskosti najbližeg para instanci iz različitih klastera
- najgora (max, complete) veza - bliskost dva klastera je jednaka bliskosti najudaljenijeg para instanci iz različitih klastera
- prosečna (avg) veza - bliskost dva klastera je jednaka prosečnoj bliskosti parova instanci iz različitih klastera.

U tabeli 1 date su prednosti i mane navedenih veza za određivanje bliskosti dva klastera kod hijerarhijskog sakupljajućeg klasterovanja, a na slici 1 njihov grafički prikaz.



Slika 1: Prikaz veza koje se mogu birati kao kriterijum za određivanje bliskosti dva klastera

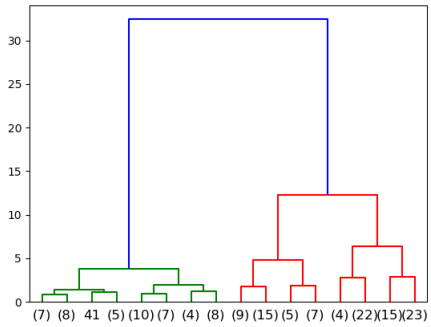
veza	prednosti	mane
single	pogodna za ne-eliptičke klastere	osetljiva na šum i elemente van granica
complete	otporna na šum i elemente van granica	sklonost ka globularnim klasterima i razbijanju velikih klastera
average	otporna na šum i elemente van granica	sklonost ka globularnim klasterima

Tabela 1: Pregled prednosti i mana veza za određivanje bliskosti dva klastera kod hijerarhijskog sakupljajućeg klasterovanja

Potrebno je izabrati meru bliskosti koja maksimizira udaljenost između instanci u različitim klasterima, a minimizuje za instance unutar istog klastera. Rezultat hijerarhijskog klasterovanja se obično prikazuje pomoću dendograma ili dijagrama sa ugnežđenim klasterima.

Za odabir broja klastera moguće je

- izvršiti algoritam do kraja (kada sve instance pripadaju jednom klasteru), a zatim na osnovu podataka o bliskosti klastera koji su spajani u svakom koraku odlučiti koji je željeni broj klastera. Za ovaj pristup je koristan prikaz klasterovanja pomoću dendograma, jer se može jednostavno uočiti u kom koraku počinje spajanje suviše različitih klastera (slika 2).
- zadati kriterijum zaustavljanja algoritma (kada se dođe do željenog broj klastera ili se dostigne unapred zadati prag bliskosti klastera prilikom spajanja).



Slika 2: Primer prikaza rezultata hijerarhijskog sakupljajućeg klasterovanja preko dendograma. Na x-osi su prikazane oznake instanci, a na y-osi udaljensot klastera koji se spajaju. Može se primetiti da postoje dva dobro razdvojena klastera (označeni su zelenom i crvenom bojom), te ne bi trebalo izvršiti poslednji korak punog hijerarhijskog klasterovanja.

2.1.1 Zadatak

Data je matrica sličnosti skupa podataka. Izvršiti hijerarhijsko klasterovanje korišćenjem min veze. Rezulatat prikazati dendogramom.

	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

Rešenje

Na početku svaka instanca predstavlja poseban klaster. U svakom koraku se spajaju dva najbliža, tj. najsličnija klastera. Pošto se primenjuje min veza, sličnost između dva klastera se određuje na osnovu dve najsličnije (najbliže) instance u različitim klasterima. Postupak se nastavlja dok sve instance ne budu u jednom klasteru. Spajanje izvršeno u svakom koraku je označeno crvernom bojom u pridruženom dendogramu.

- I korak - spajaju se klasteri sa instancama p2 i p5, pošto je ovaj par instanci najsličniji. Pre sledećeg spajanja, potrebno je izračunati sličnost (s) između novog klastera {p2,p5} i ostalih klastera primenom min veze. Kako se kao mera bliskosti koristi sličnost, najbliži par iz dva klastera (min veza, videti sliku 1) će imati najveću sličnost.

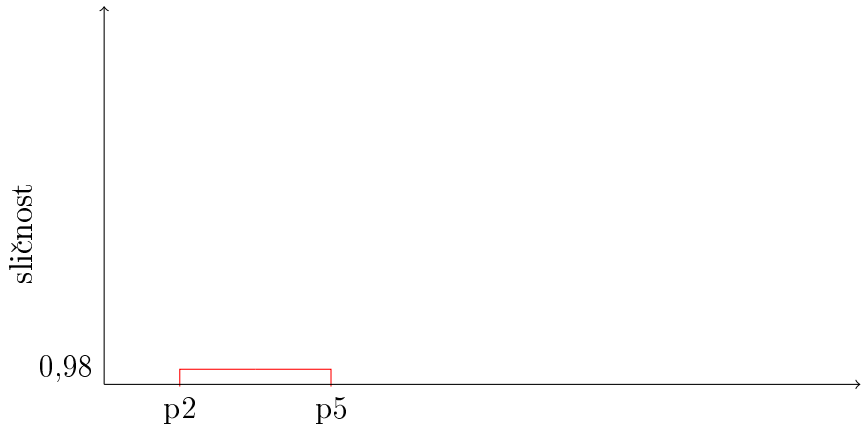
$$s(\{p2, p5\}, p1) = max(s(p2, p1), s(p5, p1)) = max(0,1, 0,35) = 0,35$$
$$s(\{p2, p5\}, p3) = max(s(p2, p3), s(p5, p3)) = max(0,64, 0,85) = 0,85$$
$$s(\{p2, p5\}, p4) = max(s(p2, p4), s(p5, p4)) = max(0,47, 0,76) = 0,76$$

  Napomena: da je data matrica različitosti (umesto matrice sličnosti) i da se koristi min veza za određivanje bliskosti dva klastera, za najbliži par iz dva klastera bi važilo da imaju *najmanju* udaljenost.

  Nakon spajanja, matrica sličnosti klastera izgleda:

	p1	{p2,p5}	p3	p4
p1	1	0,35	0,41	0,55
{p2,p5}	0,35	1	0,85	0,76
p3	0,41	0,85	1	0,44
p4	0,55	0,76	0,44	1

  a dendogram:



- II korak - najsličniji su klasteri {p3} i {p2, p5}, te se oni spajaju. Sličnost novog klastera sa ostalim klasterima je:

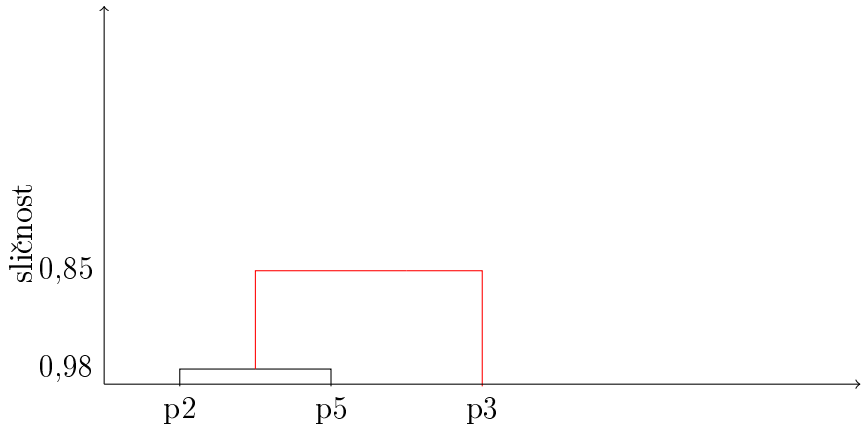
$$s(\{p2, p3, p5\}, p1) = max(s(\{p2, p5\}, p1), s(p3, p1)) = max(0,35, 0,41) = 0,41$$

	p1	{p2,p3,p5}	p4
p1	1	0,41	0,55
{p2,p3, p5}	0,41	1	0,76
p4	0,55	0,76	1

$$s(\{p2, p3, p5\}, p4) = max(s(\{p2, p5\}, p4), s(p3, p4)) = max(0, 76, 0, 44) = 0, 76$$

Nakon spajanja, matrica sličnosti klastera izgleda:

a dendogram:



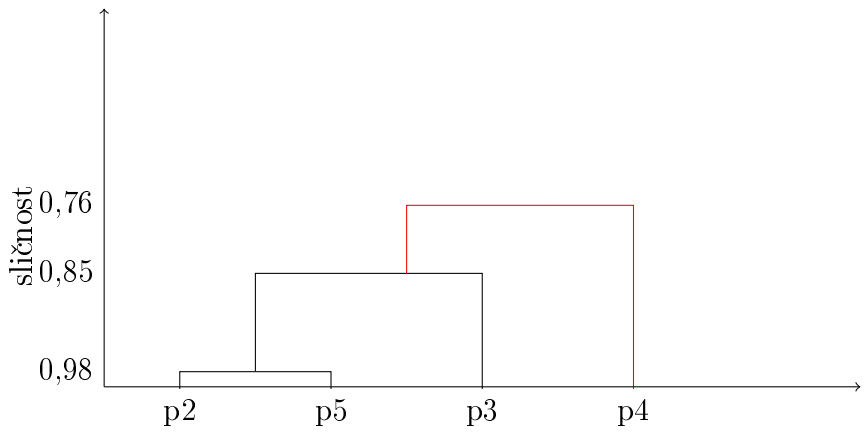
- III korak - najsličniji su klasteri {p4} i {p2, p3, p5}, te se spajaju u jedan. Sličnost poslednja dva klastera {p2, p3, p4 ,p5} i {p1} je

$$s(\{p2, p3, p4, p5\}, p1) = max(s(\{p2, p3, p5\}, p1), s(p4, p1)) = max(0, 41, 0, 55) = 0, 55$$

Nakon spajanja, matrica sličnosti klastera izgleda:

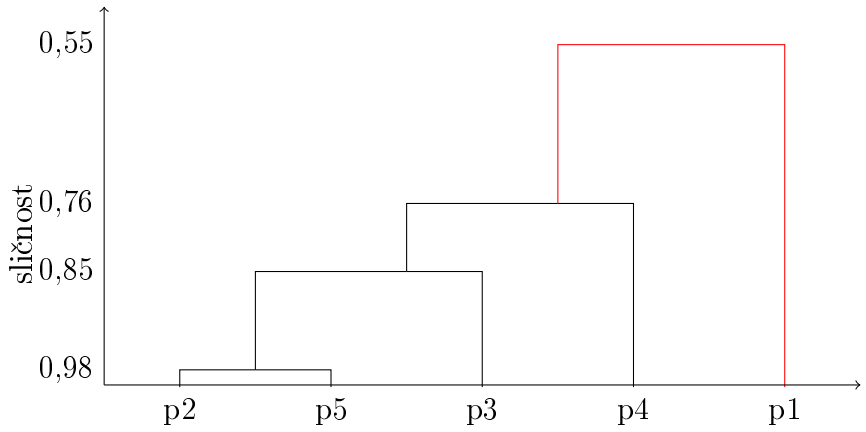
	p1	{p2,p3,p4,p5}
p1	1	0,55
{p2,p3, p4, p5}	0,55	1

a dendogram:



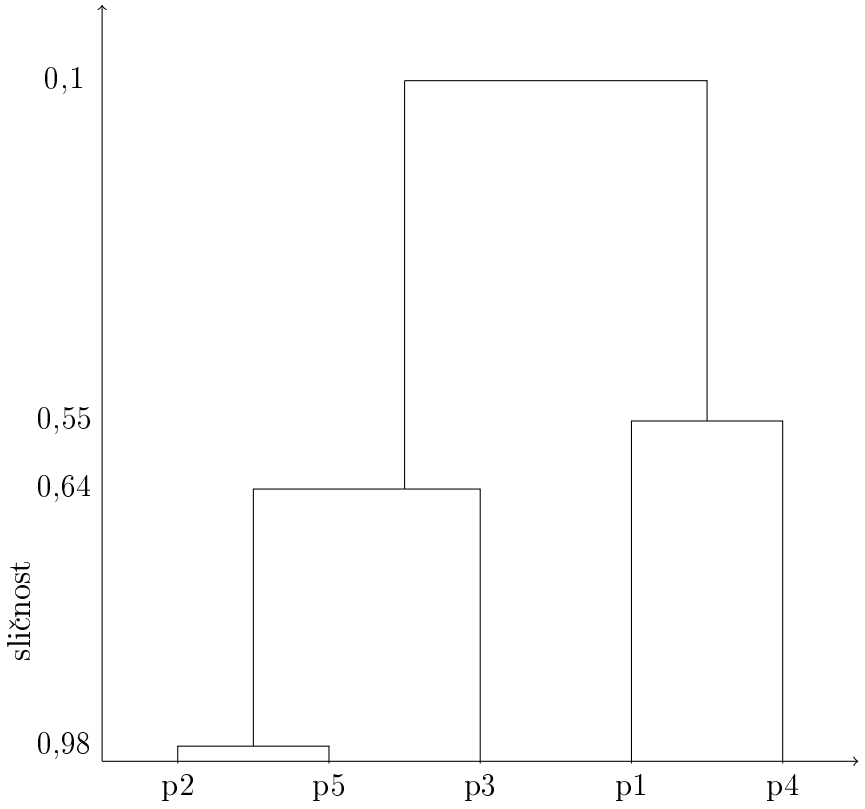
- IV korak - spajaju se poslednja dva klastera: klaster sa instancom p1 i klaster sa instancama {p2, p3, p4, p5}.

  Dendogram:



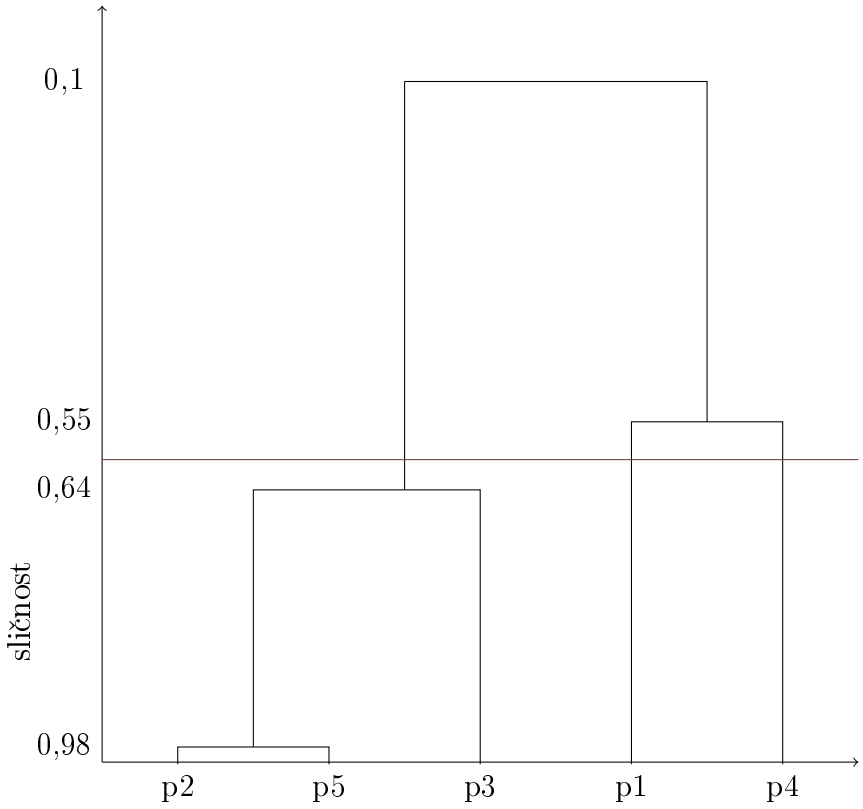
Ukoliko je potrebno izdvojiti dva klastera, poništava se poslednje spajanje i izdvajaju se klasteri {p1} i {p2, p3, p4, p5}. Ukoliko je potrebno izdvojiti tri klastera, poništavaju se poslednja dva spajanja i izdvajaju se klasteri {p1}, {p4} i {p2, p3, p5}.

Ako se umesto min veze koristi max veze pri računanju sličnosti dva klastera, traži se par instanci tih klastera sa najmanjom sličnošću i njihova sličnost je sličnost tih klastera. Dendogram klasterovanja sa max vezom izgleda:



Da je pre primene hijerarhijskog klasterovanja, kao kriterijum za zaustavljanje klasterovanja zadat prag za sličnost klastera 0,6, poslednja dva spajanja se ne bi izvršila i izdvojeni bi bili

klasteri: {p4}, {p1} i {p2, p3, p5}. Na sledećem dendogramu je crvenom linijom prikazan zadati prag. Sva spajanja klastera čija je sličnost manja od praga se ne izvršavaju.



2.2 Hijerarhijsko sakupljajuće klasterovanje u biblioteci scikit-learn programskog jezika Python

Algoritam hijerarhijskog sakupljajućeg klasterovanja je implementiran klasom **sklearn.cluster.AgglomerativeClustering** koja ima

- parametre konstruktora
  - *n_ clusters* - broj klastera, default=8
  - *affinity* - mera za računanje bliskosti između instanci (može biti: *euclidean*, *l1*, *l2*, *manhattan*, *cosine*), default: *euclidean*
  - *linkage* - kriterijum za određivanje bliskosti dva klastera (*complete*, *average*, *single*, *ward*), default:*ward*
- atribute
  - *labels_* - oznake klastera kojima su instance dodeljene; oznake klastera su u intervalu [0, n_clusters-1]
  - *children_* - matrica koja čuva podatak o deci unutrašnjih čvorova u drvetu koje predstavlja klasterovanje. Instance skupa su listovi u drvetu klasterovanja, a unutrašnji čvorovi označavaju spajanja. Drvo omogućava da se prati koji su klasteri spojeni u kom koraku. Instancama u skupu se dodeljuju oznake od 0 do broj_instanci-1. U i-toj iteraciji, *deca[i][0]* i *deca[i][1]* se spajaju da bi formirali čvor broj_instanci + i.

- metode
  - *fit* - izvršavanje klasterovanja
  - *fit_predict* - izvršavanje klasterovanja i dodela oznake klastera svakoj instanci

Primer 1: Dat je skup *dogs* koji ima atribute:

- *breed* - rasa psa
- *height* - visina psa
- *weight* - težina psa

Primeniti hijerarhijsko sakupljajuće klasterovanje na osnovu visine i težine pasa korišćenjem biblioteke *scikit-learn* programskog jezika Python.

Rešenje: **hijerarhijsko_scikitlearn.py**

2.3 Hijerarhijsko sakupljajuće klasterovanje u biblioteci scipy programskog jezika Python

Modul **scipy.cluster.hierarchy** sadrži funkcije čijom primenom se može izvršiti sakupljajuće hijerarhijsko klasterovanje. Funkcije su:

- **scipy.cluster.hierarchy.linkage** - izvršava klasterovanje, a argumenti koje prima su :
  - *y* - skup podataka ili matrica rastojanja
  - *method* - kriterijum za određivanje blizine klastera (*complete*, *average*, *single*, *ward*, *centroid*), default:*single*
  - *metric* - mera različitosti dve instance (*euclidean*, *cityblock*, *cosine*), ...) default: *euclidean*

  Vraća matricu spajanja Z (u *i.* iteraciji dobija se *n+i.* klaster spajanjem klastera Z[i,0] i Z[i,1] čije je rastojanje Z[i,2], a nakom spajanja sadrži Z[i,3] instanci).
- **scipy.cluster.hierarchy.dendrogram** - predstavlja rezultat hijerarhijskog klasterovanja pomoću dendograma, a argumenti koje prima su:
  - *Z* - matrica spajanja
  - *color_threshold* - sva spajanja koja imaju rastojanje iznad zadatog praga se boje plavom bojom. (default: 0.7*max(Z[:,2]))
  - *labels* - oznake instanci
  - *leaf_font_size* - veličina slova za ispis oznaka
- **scipy.cluster.hierarchy.fcluster** - dodeljuje id klastera svakoj instanci, a argumenti koje prima su:
  - *Z* - matrica spajanja
  - *t* - prag za spajanje klastera

– *criterion* - kriterijum za određivanje klastera. Koristimo samo ***distance*** - klasteri čije je rastojanje iznad zadatog praga t neće biti spojeni.

Id klastera su u intervalu [1, k], gde je k broj klastera.

Primer 2: Dat je skup *dogs* koji ima atribute:

- *breed* - rasa psa
- *height* - visina psa
- *weight* - težina psa

Primeniti hijerarhijsko sakupljajuće klasterovanje na osnovu visine i težine pasa korišćenjem biblioteke *scipy* programskog jezika Python.

Rešenje: **hijerarhijsko_scipy.py**

3 Algoritam DBSCAN (Density-based spatial clustering of applications with noise)

Algoritam DBSCAN klastere pronalazi na osnovu gustine instanci. U algoritmu DBSCAN se ne zadaje željeni broj klastera. Prednost algoritma je što može da pronađe klastere proizvoljnog oblika. Pri primeni algoritma DBSCAN moraju se zadati parametri:

- *Eps* - prag za rastojanje suseda. Dve instance su susedne ako im je rastojanje manje ili jednako *Eps*
- $MinPts$ - prag za broj suseda instanci

Korišćenjem ova dva parametra, za svaku instancu skupa se određuje tip:

- Instance u jezgru klastera - instanca je u jezgru klastera ako je broj suseda na rastojanju *Eps* bar $MinPts$.
- Instance na granici klastera - instanca nije u jezgru, ali je na rastojanju do *Eps* od neke instance koja je u jezgru klastera.
- Šum - instanca koja nije ni u jezgru ni na granici klastera. Ove instance neće biti dodeljene nijednom klasteru.

Koraci za DBSCAN su opisani u algoritmu 2.

Algoritam 2 DBSCAN

1: Za svaku instancu odrediti tip: *u jezgru*, *na granici* ili *šum*.
2: Eliminisati instance koje su *šum*.
3: Povezati sve instance *u jezgru* koje su na međusobnom rastojanju do *Eps*.
4: Napraviti poseban klaster za svaku grupu instanci *u jezgru* koje su povezane.
5: Svaku instancu *na granici* dodeliti klasteru kojem pripada instanca *u jezgru* u čijem je susedstvu ta instanca *na granici*.

3.1 DBSCAN u biblioteci scikit-learn programskog jezika Python

Algoritam DBSCAN je implementiran klasom **sklearn.cluster.DBSCAN** koja ima

- parametre konstruktora
  - *eps* - maksimalna udaljenost između dve instance da bi se smatralo da su u susedstvu, default:0,5
  - *min_samples* - neophodan broj instanci u susedstvu da bi se neka instanca imala status *u jezgru*. Ovaj broj ukuljučuje i samu instancu.
  - *metric* - mera rastojanja, (*euclidean*, *l1*, *l2*, *manhattan*, *cosine*)
- atribute
  - *core_sample_indices_* - indeksi instanci u jezgru
  - *labels_* - oznake klastera kojima su instance dodeljene. Instance označene sa *šum* imaju oznaku -1, a oznake klastera su u intervalu [0, k]
- metode
  - *fit* - izvršavanje klasterovanja
  - *fit_predict* - izvršavanje klasterovanja i dodela oznake klastera svakoj instanci

Primer 3: Dat je skup *dogs* koji ima atribute:

- *breed* - rasa psa
- *height* - visina psa
- *weight* - težina psa

Primeniti klasterovanje algoritmom DBSCAN na osnovu visine i težine pasa korišćenjem programskog jezika Python.

Rešenje: **dbscan.py**