Modeli klasteriranja u alatu Scikit-Learn Python

Pauletić, Iva

Sveučilište u Rijeci – Odjel za informatiku

Preddiplomski jednopredmetni studij infromatike

Iva Pauletić

Modeli klasteriranja u alatu Scikit-Learn Python

Završni rad

Mentor: dr. sc. Lucia Načinović Prskalo

Rijeka, rujan 2018.

Zadatak za završni rad

Rijeka, 15.3.2018.

Zadatak za završni rad

Pristupnik: Iva Pauletić

Naziv završnog rada: Modeli klasteriranja u alatu Scikit-Learn Python

Naziv završnog rada na eng. jeziku: Clustering models in Scikit-Learn Python

Sadržaj zadatka: Zadatak završnog rada je pojasniti postupak klasteriranja, pojasniti osnovne pojmove koji se pojavljuju prilikom primjene postupka klasteriranja, zatim opisati i isprobati neke od najznačajnijih algoritama za klasteriranje u alatu Scikit-Python kao što su primjerice K-Means, Affinity propagation, Spectral clustering i slično te pokazati njihov rad na primjerima.

Mentor

Dr. sc. Lucia Načinović Prskalo

Voditelj za završne radove

Dr. sc. Miran Pobar

Zadatak preuzet: 22.3.2018.

(potpis pristupnika)

SADRŽAJ

SAŽETAK

Ovaj završni rad prikazuje modele klasteriranja u alatu Scikit-Learn Python. U uvodnom se dijelu navode definicije osnovnih pojmova, zatim se nabrajaju vrste klasteriranja te mogući ulazni podaci koji se prihvaćaju.

U radu su također za svaki algoritam opisani način na koji se vrši postupak klastreriranja, osnovne karakteristike te vizualizacija. Objašnjeno je i samo ocjenjivanje učinkovitosti klasteriranja gdje se spominju prilagođeni Rand indeks, rezultati bazirani na zajedničkim informacijama, homogenost, potpunost i drugi. Tako se za svaki od ovih pojmova pojašnjava njegovo značenje, koje su mu prednosti i nedostaci te matematička formulacija gdje se ta mjera prikazuje formulama.

Nadalje, u radu je opisana i primjena K-Means algoritma za klasteriranje dokumenata popraćeno kodom samog algoritma uz prikazivanje konačnog rezultata i razrade postupaka koji se primjenjuju prilikom klasteriranja dokumenata.

Ključne riječi: klasteriranje, Scikit-Learn, algoritam, uzorak, klaster, oznake, prilagođeni Rand indeks, zajedničke informacije, homogenost, potpunost, V-mjera, koeficijent siluete, tokeni

1. Uvod

Klasteriranje ili grupiranje [1] je tehnika klasterske analize nad statističkim podacima. Koristi se kod strojnog učenja, rudarenja podataka, raspoznavanja uzoraka, analiza slika i slično. Slične objekte klasificira u različite grupe, tj. dijeli skup podataka u podskupove koji dijele zajedničko obilježje.

Neke od vrsta klasteriranja su K-Means, Affinity propagation, Mean-shift, Spectral clustering, Ward hierarchical clustering, Agglomerative clustering, DBSCAN, Birch koji mogu biti vizualno prikazani slikama.

Postoje dvije vrste algoritama klasteriranja koji se dijele na hijerarhijske i nehijerarhijske. Hijerarhijski se baziraju na ideji da se objekti više vežu za one objekte koji su im bliži nego za one koji su im dalje. Tako se formiraju klasteri pomoću mjere sličnosti između novih elemenata i ostalih prije određenih klastera. Kod nehijerarhijskih se svaki objekt smješta u točno jedan od disjunktnih klastera. Broj klastera se unaprijed određuje. Najpoznatiji takav algoritam je K-Means.

Klasteriranje neoznačenih podataka može se izvesti pomoću modula *sklearn.cluster*.

Svaki algoritam klasteriranja dolazi u dvije varijante: klasa, koja implementira *fit* metodu (metodu prilagodbe) kako bi učila klastere na testnim podacima, i funkcija, koja, danim testnim podacima, vraća niz cjelobrojnih oznaka odgovarajući na različite klastere. Za klasu, oznake testnih podataka mogu biti pronađene u *labels_* atributu.

Važno je napomenuti da algoritmi implementirani u ovom modulu mogu unositi različite vrste matrica (ulazne podatke). Sve metode prihvaćaju standardne matrice podataka oblika *[n_uzorci, n_značajke]*. To se može dobiti iz klasa u modulu *sklearn.feature_extraction*. Za AffinityPropagation, SpectralClustering i DBSCAN također se mogu unijeti slične matrice oblika *[n_uzorci, n_značajke]*. To se može dobiti od funkcija u modulu *sklearn.metrics.pairwise*.

Završni rad organiziran je kako slijedi. U sljedećem poglavlju pojašnjeni su algoritmi za klasteriranje sa svim njihovim karakteristikama i vizualnim primjerima, dok je ocjenjivanje učinkovitosti klasteriranja sa svim načinima ocjenjivanja opisan u trećem poglavlju. Četvrto poglavlje sadrži primjenu K-Means algoritma za klasteriranje dokumenata što uključuje postupke klasteriranja sa pripadnim djelom koda. Naposljetku rad završava poglavljem u kojem su izvučeni glavni zaključci.

2. Algoritmi klasteriranja

U ovom su poglavlju opisani odabrani algoritmi klasteriranje i njihove glavne karakteristike.

2.1. K-Means

K-Means algoritam [1] klasterira podatke pokušavajući rastaviti uzorke u n grupa jednakih varijanci, umanjujući kriterij poznat kao inercija ili zbroj kvadrata unutar klastera. Ovaj algoritam zahtjeva određen broj klastera. Dobro se slaže sa velikim brojem uzoraka i koristi se u širokom rasponu različitih aplikacijskih područja.

K-Means algoritam dijeli skup N uzoraka X u K odvojenih klastera C, od kojih je svaki opisan sredinom μ_j uzoraka u klasteru. Sredine se obično nazivanju „centroidnim“ klasterima; uzimajući u obzir da nisu, uglavnom, točke od X, iako žive u istom prostoru. K-Means algoritam ima za cilj odabrati centroide koji umanjuju inerciju, ili kriterij zbroja kvadrata unutar klastera:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_j - \mu_i||^2).$$

Inercija, ili kriterij zbroja kvadrata unutar klastera, može se prepoznati kao mjera koherentnosti klastera. Taj kriterij, međutim, ima raznih nedostataka: inercija pretpostavlja da su klasteri konveksni i izotropni, što nije uvijek slučaj. Slabo reagira na izdužene klastere, ili mnogostruke sa nepravilnim oblicima. Inercija nije normalizirana mjera zato što znamo da su niže vrijednosti bolje i da je nula optimalna. No, u vrlo visokim dimenzionalnim prostorima, Euklidske su udaljenosti obično napuhane (primjer tzv. „prokletstva dimenzionalnosti“). Pokretanje algoritma smanjenja dimenzionalnosti kao što je PCA prije K-Means klasteriranja može ublažiti taj problem i ubrzati računanja.

K-Means se često naziva Lloydovim algoritmom. U osnovi, algoritam ima tri koraka. Prvi korak odabire početne centroide, sa najobičnijom metodom izabire k uzorke iz skupa podataka X. Nakon inicijalizacije, K-Means se sastoji od petlje između druga dva koraka. Prvi korak svaki uzorak dodjeljuje najbližem centroidu. Drugi korak stvara nove centroide uzimajući srednju vrijednost svih uzoraka koji su dodijeljeni svakom prethodnom centroidu. Računa se razlika između starih i novih centroida, a algoritam ponavlja posljednja dva koraka sve dok ta vrijednost nije manja od početne. Drugim riječima, ponavlja se dok se centroidi značajno ne pomaknu.

K-Means je ekvivalentan algoritmu očekivanja-maksimizacije sa malom, ujednačenom, dijagonalnom matricom kovarijancije.

Algoritam se također može shvatiti kroz koncept Voronoi diagrama [1]. Prvo, Voronoi dijagram točaka se računa pomoću trenutnih centroida. Svaki segment u Voronoi dijagramu postaje zasebni klaster. Drugo, centroidi se ažuriraju na sredinu svakog segmenta. Tada ga aloritam ponavlja dok se ne ispuni kriterij zaustavljanja. Obično se algoritam zaustavi kada je relativno

smanjenje u objektivnoj funkciji između iteracija manje od zadane tolerancije. U ovom slučaju iteracija staje kada se centroidi kreću manje od tolerancije.

S obzirom na dovoljno vremena, K-Means će uvijek konvergirati, međutim to može biti prema lokalnom minimumu. To jako ovisi o inicijalizaciji centroida. Kao rezultat, izračun se često provodi nekoliko puta, s različitim inicijalizacijama centroida. Metoda za rješavanje ovog problema je inicijalizacijska shema K-Means++, koja je implementirana u scikit-learnu. To inicijalizira centroide da budu udaljeni jedni od drugih, što dovodi do boljih rezultata od slučajne inicijalizacije, kao što je prikazano u referenci [2].

Može se dati parametar koji omogućuje da se K-Means pokreće paralelno, zvan *n_jobs*. Davajući taj parametar pozitivnu vrijednost koristimo za mnoge procesore. Vrijednost -1 koristi sve dostupne procesore, dok -2 koristi jedan manje, i tako dalje. Paralelizacija općenito ubrzava računanje po cijeni memorije (u tom slučaju, potrebno je pohranjeniti više primjeraka centroida, jedan za svaki posao).

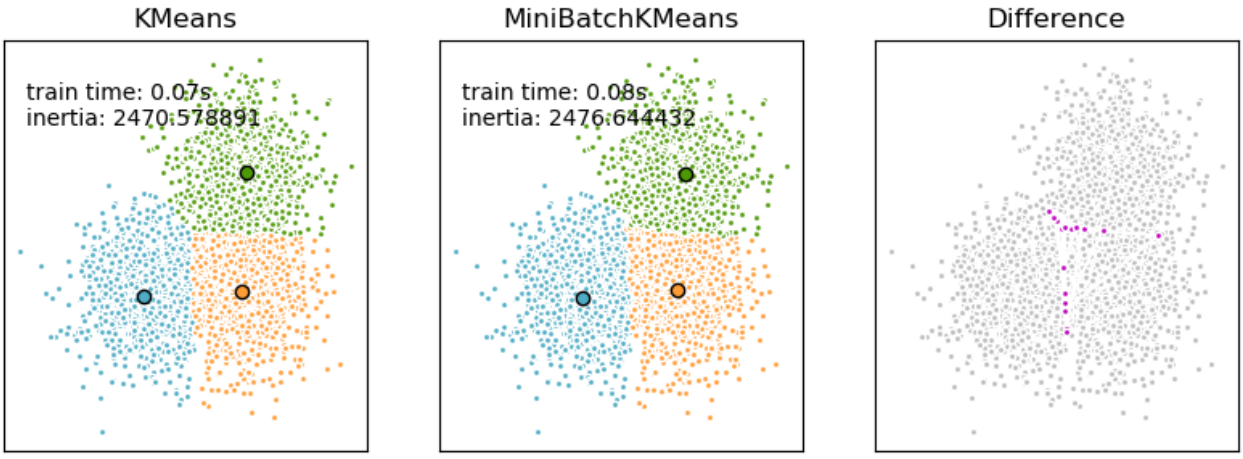
K-Means se može koristiti za kvantizaciju vektora. To se postiže primjenom metode transformacije osposobljenog modela K-Means.

2.2. Mini Batch K-Means

Mini Batch K-Means [1] je varijanta K-Means algoritma koji koristi mini-serije kako bi smanjio vrijeme računanja, dok još pokušava optimizirati istu objektivnu funkciju. Mini-serije su podskupovi ulaznih podataka, nasumično uzorkovanih u svakoj probnoj iteraciji. Te mini-serije drastično smanjuju količinu računanja potrebnu za konvergiranje u lokalno rješenje. Nasuprot ostalim algoritmima koji smanjuju vrijeme konvergencije K-Meansa, Mini Batch K-Means daje rezultate koji su uglavnom malo gore od standardnog algoritma.

Algoritam iteratira između dva glavna koraka, slično K-Means. U prvom koraku, b uzorci su nasumično izabrani iz skupa podataka, kako bi se formirale mini-serije. Oni se zatim dodjeljuju najbližem centroidu. U drugom koraku, centroidi se ažuriraju. U odnosu na K-Means, to se vrši po uzorku. Za svaki uzorak u mini-seriji, dodijeljeni centroid se ažurira uzimajući tekući prosjek uzorka i svih prethodnih uzoraka dodijeljenih tom centroidu. Posljedica toga je smanjenje brzine kretanja centroida tijekom vremena. Ovi se koraci izvode sve dok se ne postigne konvergencija ili unaprijed određeni broj iteracija.

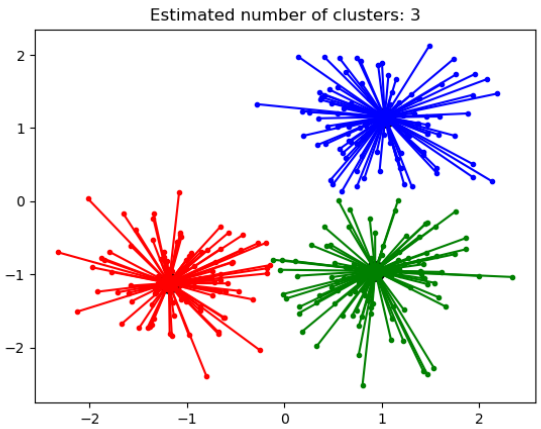
Mini Batch K-Means konvergira brže od K-Means, ali je kvaliteta rezultata manja. U praksi ova razlika u kvaliteti može biti vrlo mala, kao što je prikazano u primjeru.



Slika 1: Razlika između K-Means i Mini Batch K-Means[1]

2.3. Affinity Propagation

Affinity Propagatiom [1] stvara klastere slanjem poruka između parova uzoraka sve do konvergencije. Skup podataka je tada opisan korištenjem malog broja primjeraka, koji su identificirani kao najreprezentativniji od drugih uzoraka. Poruke poslane između parova predstavljaju pogodnost da jedan uzorak bude primjer ostalima, koji je ažuriran kao odgovor na vrijednosti iz drugih parova. To ažuriranje se događa iterativno sve do konvergencije, izabiru se konačni primjeri, i tako je dano konačno klasteriranje.



Slika 2: Affinity Propagation[2]

Affinity Propagation može biti interesantan jer odabire broj klastera na temelju dostavljenih podataka. U tu svrhu, dva važna parametra su *prednost*, koja kontrolira koliko primjeraka se koristi, i *faktor prigušenja* koji odbacuje poruke odgovornosti i dostupnosti kako bi izbjegao brojne

[1] Izvor: http://scikit-learn.org/stable/_images/sphx_glr_plot_mini_batch_kmeans_0011.png
[2] Izvor: http://scikit-learn.org/stable/_images/sphx_glr_plot_affinity_propagation_0011.png

oscilacije prilikom ažuriranja tih poruka.

Glavni nedostatak Affinity Propagationa je kompleksnost. Algoritam ima vrijeme kompleksnosti reda $O(N^2T)$, gdje je N broj uzoraka i T broj iteracija do konvergencije. Nadalje, memorija kompleksnosti je reda $O(N^2)$ ako se koristi gusta matrica sličnosti, ali se smanjuje ako se koristi rijetka matrica sličnosti. To čini Affinity Propagation više prikladnijim za male do srednje veličine skupova podataka.

Opis algoritma: Poruke poslane između točaka pripadaju jednoj od dvije kategorije. Prva je odgovornost $r(i,k)$, koja je akumulirani dokaz da uzorak k treba biti primjer uzorku i. Druga je dostupnost $a(i,k)$ koja je akumulirani dokaz da uzorak i treba izabrati uzorak k da bude njegov primjer, te razmatra vrijednost svih ostalih uzoraka kojima k treba biti primjer. Na taj način, primjeri su odabrani uzorcima ako su dovoljno slični mnogim uzorcima i ako su odabrani od strane mnogih uzoraka kako bi predstavljali sebe.

Formalnije, odgovornost uzorka k da bude primjer uzorku i je dana:

$$r(i,k) \leftarrow s(i,k) - \max[a(i,k') + s(i,k')\ \forall k' \neq k\ ]$$

gdje je $s(i,k)$ je sličnost između uzoraka i i k. Dostupnost uzorka k da bude primjer uzorku i je dana:

$$a(i,k) \leftarrow \min[0, r(k,k) + \sum_{i'\ s.t.\ i' \notin \{i,k\}} r(i',k)].$$

Za početak, sve vrijednosti za r i a su postavljene na nulu, i izračun svake iteracije sve do konvergencije. Kao što je gore opisano, kako bi se izbjegle brojne oscilacije tijekom ažuriranja poruka, faktor prigušenja λ se uvodi u proces iteracije:

$$r_{t+1}(i,k) = \lambda \cdot r_t(i,k) + (1 - \lambda) \cdot r_{t+1}(i,k)$$
$$a_{t+1}(i,k) = \lambda \cdot a_t(i,k) + (1 - \lambda) \cdot a_{t+1}(i,k)$$

gdje t prikazuje vrijeme iteracije.

2.4. Mean Shift

Klasteriranje Mean Shift [1] ima cilj otkriti „blobove“ u gustoći uzoraka. To je algoritam koji se temelji na centroidu, a funkcionira tako da ažurira kandidate za centroide kako bi bili sredina točke unutar određene regije. Ti kandidati se zatim filtriraju u postprocesnoj fazi kako bi se eliminirali duplikati za stvaranje konačnog skupa centroida.

S obzirom na kandidatski centroid x_i za iteraciju t, kandidat se ažurira prema sljedećoj jednadžbi:

$$x_i^{t+1} = x_i^t + m(x_i^t)$$

gdje je $N(x_i)$ susjedstvo uzoraka unutar određene udaljenosti x_i i m je *vektor pomaka* koji se izračunava za svaki centroid koji pokazuje prema regiji maksimalnog povećanja u gustoći točaka.

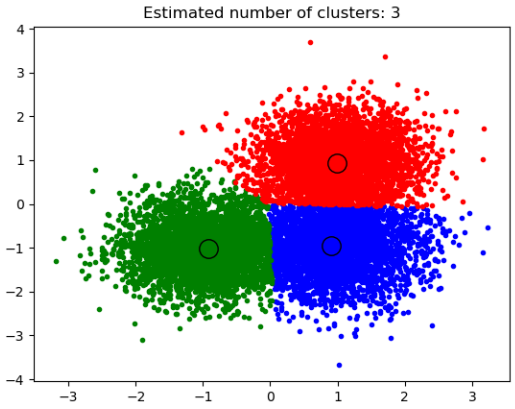
Ovo se izračunava pomoću sljedeće jednadžbe, koja predstavlja učinkovito ažuriranje centroida kako bi bio sredina uzoraka unutar njegovog susjedstva:

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}.$$

Algoritam automatski postavlja broj klastera, umjesto da se oslanja na parametar *bandwidth* (propusnost), koji diktira veličinu regije za pretraživanje. Taj parametar može se postaviti ručno, ali se može procijeniti pomoću predviđene *estimate_bandwidth* funkcije, koja je pozvana ako bandwidth nije postavljen.

Algoritam nije visoko skalabilan, jer zahtijeva pretraživanje više najbližih susjeda tijekom izvršavanja algoritma. Algoritam garantira konvergenciju, no zaustaviti će iteraciju kada je promjena centroida mala.

Označavanje novog uzorka se izvodi traženjem najbližeg centroida za određeni uzorak.



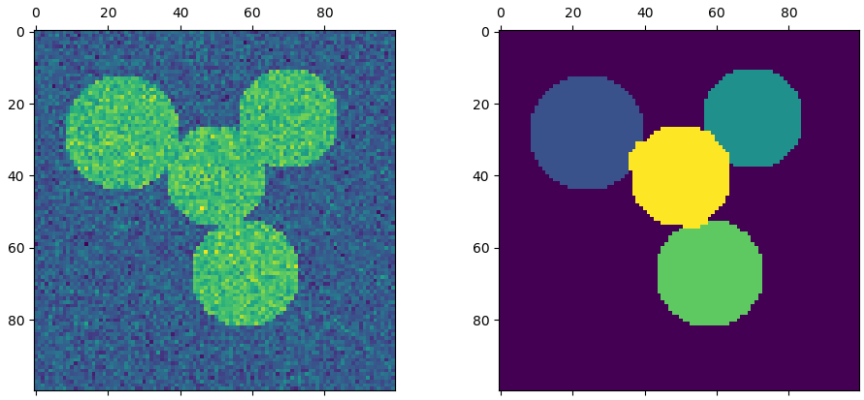
Slika 3: Mean Shift[3]

2.5. Spektralno klasteriranje (Spectral clustering)

Spektralno klasteriranje [1] ugrađuje malu dimenziju matrice afiniteta između uzoraka, nakon čega slijedi K-Means u malom dimenzionalnom prostoru. Posebno je učinkovito ako je matrica afiniteta rijetka i instaliran modul pymag. Spektralno klasteriranje zahtijeva određeni broj klastera. Dobro radi za mali broj klastera ali se ne preporučuje pri korištenju više njih.

Za dva klastera, rješava konveksnu relaksaciju normalizirano smanjujući problem na grafikonu sličnosti: rezanje grafa na dva tako da je težina rezanja rubova manja u odnosu na težine rubova unutar svakog klastera. Ovaj kriterij je posebno zanimljiv kada se radi na slikama: vrhovi grafa su pikseli, a rubovi sličnog grafa su funkcije gradijenta slike.

[3] Izvor: http://scikit-learn.org/stable/_images/sphx_glr_plot_mean_shift_0011.png



Slika 4: Spektralno klasteriranje[4]

2.5.1. Različite strategije dodjeljivanja oznaka

Mogu se koristiti različite strategije dodjeljivanja oznaka, koje odgovaraju parametru *assign_labels* za Spektralno klasteriranje. Strategija „k-means“ se može podudarati sa sitnim detaljima podataka, ali katkad može biti nestabilnija. Potrebno je kontrolirati *random_state*, koji se slučajno inicijalizira, jer se ova strategija možda neće moći reproducirati iz pokretanja. S druge strane, strategija „discretize“ je sto posto izvodljiva, ali teži stvaranju parcela prilično jednakog i geometrijskog oblika.

2.6. Hijerarhijsko klasteriranje (Hierarchical clustering)

Hijerarhijsko klasteriranje [1] je općenito obitelj algoritama klasteriranja koja gradi ugniježđene klastere spajajući ili rastavljajući ih sukcesivno. Ova hijerarhija klastera se predstavlja kao stablo. Korijen stabla je jedinstveni klaster koji sakuplja sve uzorke, a listovi su klasteri sa samo jednim uzorkom.

Objekt *Agglomerative Clustering* obavlja hijerarhijsko klasteriranje koristeći pristup od dna prema vrhu: svako promatranje počinje u svom klasteru, i klasteri se sukcesivno spajaju. Kriterij povezivanja određuju mjeru koja se koristi za strategiju spajanja. Tako *Ward* smanjuje sumu kvadratnih razlika unutar svih klastera. To je pristup minimiziranja varijance i u tom smislu je sličan funkciji k-means, ali se rješava sa aglomerativnim hijerarhijskim pristupom. Druga mjera je maksimalna ili potpuna povezanost koja smanjuje maksimalnu udaljenost između promatranih parova klastera, te posljednja mjera prosječne povezanosti koja smanjuje prosjek udaljenosti između svih promatranih parova klastera.

Agglomerative Clustering može se također mjeriti i na velikom broju uzoraka kada se

[4] Izvor: http://scikit-learn.org/stable/_images/sphx_glr_plot_segmentation_toy_0011.png
http://scikit-learn.org/stable/_images/sphx_glr_plot_segmentation_toy_0021.png

upotrebljava zajedno s matricom povezanosti, ali je računalno skupo kada između uzoraka nisu dodana ograničenja povezanosti: uzimaju se u obzir sva moguća spajanja na svakom koraku.

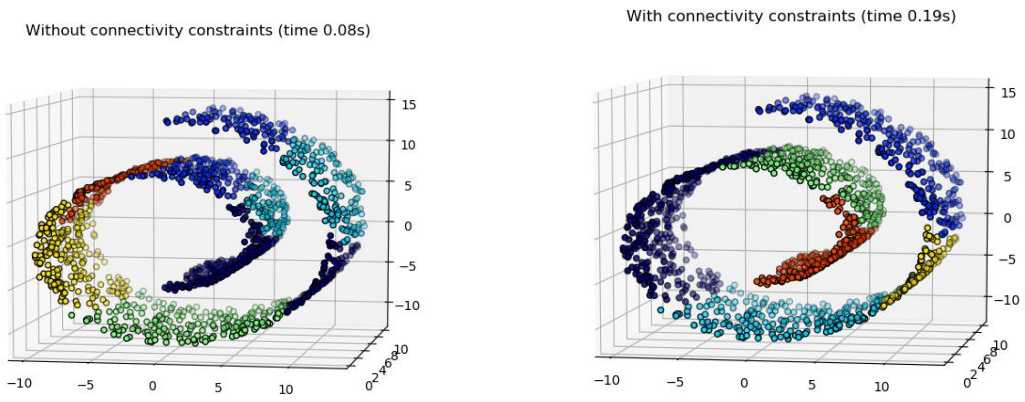
Feature Agglomeration je alat za smanjenje dimenzionalnosti, koji koristi aglomerativno klasteriranje za grupiranje značajki koje izgledaju jako slično i na taj način, smanjuje broj značajki.

2.6.1. Različiti tipovi veza: Ward, potpuna i prosječna povezanost

Agglomerative Clustering podržava *Ward*, prosječne i potpune strategije povezivanja. Aglomerativni klaster ima ponašanje „bogati se bogate“ koji vodi do nejednakih veličina. U tom pogledu, potpuna povezanost je najgora strategija, a *Ward* daje najregularnije veličine. No, sklonost (ili udaljenost koja se koristi u klasteriranju) ne može se mijenjati s Wardom, zbog čega je prosječna povezanost dobra alternativa za neeuklidska mjerenja.

2.6.2. Dodavanje ograničenja povezivanja

Zanimljiv aspekt *Agglomerative Clustering* je to da mu se mogu dodati ograničenja povezivanja (samo susjedni klasteri se mogu spojiti), putem matrice povezanosti koja određuje susjedne uzorke za svaki uzorak slijedeći danu strukturu podataka. Na primjer, u primjeru švicarskog svitka prikazanog ispod, ograničenja povezanosti zabranjuju spajanje točaka koje nisu susjedne na švicarskom svitku, i time izbjegava formiranje klastera koji se protežu preko preklapajućih nabora svitka.



Slika 5: Primjer *Agglomerative Clustering*-a prije i nakon dodavanja ograničenja povezivanja (*Ward*)[5]

Ta ograničenja su korisna za nametanje određene lokalne strukture, ali također čine algoritam bržim, posebno kada je broj uzoraka visok.

[5] Izvor: http://scikit-learn.org/stable/_images/sphx_glr_plot_ward_structured_vs_unstructured_0011.png
http://scikit-learn.org/stable/_images/sphx_glr_plot_ward_structured_vs_unstructured_0021.png

Ograničenja povezivanja su nametnuta putem matrice povezanosti: rijetka scipy matrica koja ima elemente samo na sjecištu reda i stupca sa indeksima skupa podataka koji bi trebali biti povezani. Ta matrica može biti konstruirana od teorijskih informacija: na primjer, grupiranje web stranice samo spajanjem stranica vezom od jedne do druge. Može se također naučiti iz podataka, na primjer koristimo *sklearn.neighbors.kneighbors_graph* da bismo ograničili spajanje s najbližim susjedima, ili koristimo *sklearn.feature_extraction.image.grid_to_graph* da bismo omogućili samo spajanje susjednih piksela na slici.

2.6.3. Mijenjanje mjernog podatka

Prosječna i potpuna povezanost se može koristiti s različitim udaljenostima (sklonostima), posebno Euklidska udaljenost, Manhattanova udaljenost (ili Cityblockova), kosinusna udaljenost ili bilo koja unaprijed izračunata matrica afiniteta.

Manhattanova udaljenost je često dobra za rijetke značajke, ili rijetke šumove: tj. mnoge značajke su nula, kao pojava rijetkih riječi u rudarenju teksta. Kosinusna udaljenost je zanimljiva jer je nepromjenjiva globalnim skaliranjima signala.

Smjernice za biranje mjernog podatka je korištenje jednog koji maksimizira udaljenost između uzoraka u različitim klasama, i minimizira to unutar svake klase.

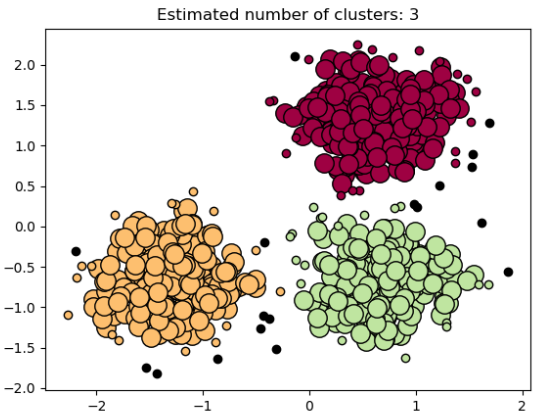
2.7. DBSCAN

Algoritam DBSCAN [1] pregledava klastere kao područja visoke gustoće razdvojene područjima niske gustoće. Zbog toga prilično općenitog pogleda, klasteri koje je pronašao DBSCAN mogu biti bilo kakovog oblika, za razliku od k-meansa koji pretpostavlja da su klasteri konveksno oblikovani. Središnja komponenta DBSCAN-a je koncept jezgrenih uzoraka, čiji se uzorci nalaze u područjima visoke gustoće. Klaster je stoga skup jezgrenih uzoraka, svaki međusobno blizu (mjeri se pomoću mjerenja udaljenosti) i skup nejezgrenih uzoraka koji su blizu jezgrenog uzorka (ali sami nisu jezgreni uzorci). U algoritmu postoje dva parametra, *min_samples* i *eps*, koji formalno definiraju što mislimo kada kažemo gusto. Viši *min_samples* ili niski *eps* ukazuju na veću gustoću potrebnu za formiranje klastera.

Formalnije, definiramo jezgreni uzorak kao uzorak u skupu podataka tako da postoji *min_samples* ostalih uzoraka unutar udaljenosti od *eps*, koji su definirani kao susjedi jezgrenog uzorka. To nam govori da je jezgreni uzorak u gustom području vektorskog prostora. Klaster je skup jezgrenih uzoraka koji može biti izgrađen rekurzivno uzimajući jezgreni uzorak, pronalazeći sve susjede koji su jezgreni uzorci, zatim pronalazeći sve njihove susjede koji su jezgreni uzorci, i tako dalje. Klaster također ima skup nejezgrenih uzoraka, koji su susjedi jezgrenog uzorka u klasteru ali sami nisu jezgreni uzorci. Intuitivno, ti uzorci su na rubovima klastera.

Po definiciji, bilo koji jezgreni uzorak je dio klastera. Bilo koji uzorak koji nije jezgreni uzorak, i koji je barem *eps* udaljen od bilo kojeg jezgrenog uzorka, algoritam smatra netipičnim.

Na slici 5, boja označava članstvo u klasteru, s velikim krugovima prikazuju se jezgreni uzorci koje je osnovao algoritam. Manji krugovi su nejezgreni uzorci koji su i dalje dio klastera. Štoviše, netipične vrijednosti (outlieri) su prikazani crnim točkama.



Slika 6: DBSCAN[6]

2.7.1. Izvršenje

Algoritam DBSCAN je deterministički, uvijek generira iste klastere kada se daju isti podaci u istom redoslijedu. Međutim, rezultati se mogu razlikovati kada se podaci daju različitim redoslijedom. Prvo, iako će jezgreni uzorci uvijek biti dodijeljeni istim klasterima, oznake tih klastera će ovisiti o redoslijedu u kojem se ti uzorci pojavljuju u podacima. Drugo i još važnije, klasteri kojima su nejezgreni uzorci dodijeljeni mogu se razlikovati ovisno o redoslijedu podataka. To će se dogoditi kada nejezgreni uzorak ima manju udaljenost od *eps*-a do dva jezgrena uzoraka u različitim klasterima. Po nejednakosti trokuta, ta dva jezgrena uzorka moraju biti više udaljeni jedan od drugoga nego *eps*, ili će biti u istom klasteru. Nejezgreni uzorak je dodijeljen onom klasteru koji je prvi generiran u prolazu kroz podatke, i tako će rezultati ovisiti o redoslijedu podataka.

Trenutna implementacija koristi *ball stabla* i *kd-stabla* kako bi odredila susjedstvo točaka, što izbjegava izračunavanje matrice pune udaljenosti (kao što je učinjeno u scikit-learn verzijama prije 0.14). Zadržana je mogućnost upotrebe prilagođenih mjernih podataka.

2.7.2. Potrošnja memorije za velike veličine uzoraka

Ova implementacija sama po sebi nije učinkovita za memoriju zato što konstruira punu matricu sličnosti parova u slučaju da se *dk-stabla* ili *ball-stabla* ne mogu koristiti (npr. sa rijetkim matricama). Nekoliko mehanizama kojim se to može ispraviti su graf s rijetkim radijusom

[6] Izvor: http://scikit-learn.org/stable/_images/sphx_glr_plot_dbscan_0011.png

susjedstva (gdje se pretpostavlja da su nestali unosi izvan *eps*-a) može biti unaprijed izračunat na memorijski učinkovit način a DBSCAN se može pokretati sa *metric='precomputed'* te se skup podataka može komprimirati, uklanjanjem pravih duplikata ako se to dogodi u podacima, ili koristeći metodu *Birch*. Tada imamo samo relativno mali broj predstavnika za veliki broj točaka. Onda možemo ponuditi *sample_weight* prilikom pripreme DBSCAN-a.

2.8. Birch

Birch [1] gradi stablo zvano *Characteristic Feature Tree* (CFT) za dane podatke. Podaci se u suštini gube komprimiranjem na skup *Characteristic Feature nodes* (CF čvorovi). CF čvorovi sadrže broj podklastera zvanih *Characteristic Feature subclusters* (CF podklasteri) i ti CF podklasteri locirani u neterminalnim CF čvorovima mogu imati CF čvorove kao djecu.

CF podklasteri sadrže potrebne informacije za klasteriranje čime se sprječava potreba za sadržavanjem svih ulaznih podataka u memoriji. Ta informacija uključuje broj uzoraka u podklasteru, linearni zbroj (n-dimenzionalni vektor koji sadrži zbroj svih uzoraka), zbroj kvadrata (zbroj kvadrata L2 norme svih uzoraka), centroide (kako bi se izbjeglo ponovno računanje linearnog zbroja/n-uzoraka) i kvadratnu normu centroida.

Algoritam *Birch* ima dva parametra, *prag* i *fakor grananja*. Faktor grananja ograničava broj podklastera u čvoru, a prag ograničava udaljenost između ulaznog uzorka i postojećih podklastera.

Ovaj algoritam se može promatrati kao instanca ili metoda smanjenja podataka, jer smanjuje ulazne podatke na skup podklastera koji se dobivaju izravno iz listova CFT-a. Smanjeni podaci se mogu dalje procesirati tako da se pohrane u globalni klasterer. Taj globalni klasterer može biti skup *n_klastera*. Ako su *n_klasteri* postavljeni na ništa, podklasteri iz listova se izravno čitaju, inače korak globalnog klasteriranja pretvara te podklastere u globalne klastere, a uzorci se prenose na globalnu oznaku najbližeg podklastera.

2.8.1. Opis algoritma:

Novi uzorak je umetnut u korijen CF Tree-a koji je CF Node. Tada se spaja s podklasterom korijena, koji ima najmanji radijus poslije spajanja, ograničen je uvjetima praga i faktora granjanja. Ako podklaster ima bilo koji čvor djeteta, tada se postupak ponavlja sve dok se ne dosegne list. Nakon pronalaženja najbližeg podklastera u listu, svojstva tog podklastera i njegovih roditelja se rekurzivno ažuriraju.

Ako je radijus podklastera dobiven spajanjem novog uzorka i najbližeg podklastera veći od kvadrata praga i ako je broj podklastera veći od faktora grananja, tada je prostor privremeno dodijeljen tom novom uzorku. Uzeta su dva najdalja podklastera i podklasteri su podijeljeni u dvije grupe na osnovi udaljenosti između tih podklastera.

Ako taj razdvojeni čvor ima roditelja podklastera i ako ima mjesta za novi podklaster, tada se roditelj razdvaja na dva. A ako nema mjesta, tada se taj čvor ponovno razdvaja na dva, a proces se rekurzivno nastavlja, sve dok se ne dosegne korijen.

3. Ocjenjivanje učinkovitosti klasteriranja

Ocjenjivanje učinkovitosti algoritma klasteriranja nije tako jednostavno kao brojanje pogrešaka ili računanje preciznosti i odziva nadziranog algoritma klasifikacije. U pravilu, bilo koja metoda ocjenjivanja ne bi trebala uzeti u obzir apsolutne vrijednosti oznaka klastera ali uzima ako ovo klasteriranje definira razdvajanje podatka sličnih nekom skupu točno klasificiranih klasa tako da članovi iste klase su sličniji članovima različitih klasa po nekim mjerilima sličnosti.

3.1. Prilagođeni Rand indeks

S obzirom na poznavanje zadataka točno klasificirane klase *labels_true* i zadataka algoritma klasteriranja istih uzoraka *labels_pred*, prilagođeni Rand indeks (ARI) je funkcija koja mjeri sličnost dvaju zadatka, ignorirajući permutacije i slučajnu normalizaciju.

Nadalje, *adjusted_rand_score* je simetričan: zamjena argumenta ne mijenja rezultat. Stoga se može koristiti kao konsenzusna mjera. Savršeno označavanje je rezultat 1.0. Loše (npr. nezavisno označavanje) ima negativne ili rezultate približne 0.0.

3.1.1. Prednosti

Dodjele slučajne (ravnomjerne) oznake imaju *ARI* rezultat blizu 0.0 za bilo koju vrijednost *n_klastera* i *n_uzoraka* (što nije slučaj za sirov Rand indeks ili V-mjeru na primjer).

Ograničeni raspon [-1,1]: negativne vrijednosti su loše (nezavisno označavanje), slična klasteriranja imaju pozitivan *ARI*, 1.0 je savršen rezultat.

Nema pretpostavke o strukturi klastera: može se koristiti za usporedbu algoritama klasteriranja kao k-means koji pretpostavlja izotropne oblike blobova s rezultatima algoritama spektralnih klasteriranja koji mogu pronaći klaster sa „presavijenim“ oblicima.

3.1.2. Nedostaci

Suprotno inerciji, *ARI* zahtijeva poznavanje točno klasificiranih klasa, a gotovo nikada nije dostupan u praksi ili zahtijeva ručno dodjeljivanje ljudskim oznakama (kao u okruženju nadziranog učenja).

Međutim *ARI* se može također koristiti u čisto nenadziranom postavljanju kao građevinski blok za konsenzusni indeks koji se može koristiti za klasteriranje odabira modela.

3.1.3. Matematička formulacija

Ako je C zadatak točno klasificirane klase i K klasteriranje, tada je a broj parova elemenata koji su u istom skupu u C, i u istom skupu u K te b je broj parova elemenata koji su u različitim

skupovimu u C i u različitim skupovima u K.

Sirovi (neprilagođeni) Rand indeks zadan je formulom:

$$RI = \frac{a+b}{C_2^{n_{uzoraka}}}$$

gdje je $C_2^{n_{uzoraka}}$ ukupan broj mogućih parova u skupu podataka.

Međutim rezultat RI ne garantira da će dodjeljivanje nasumičnih oznaka dobiti vrijednost blizu nule (pogotovo ako je broj klastera u istom redoslijedu veličine kao i broj uzoraka).

Za suzbijanje ovog učinka možemo odbiti očekivani RI $E[RI]$ slučajnih označavanja određivanjem prilagođenog Rand indeksa kako slijedi:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}.$$

3.2. Rezultati bazirani na zajedničkim informacijama

S obzirom na poznavanje zadataka točno klasificirane klase *labels_true* i zadataka algoritma klasteriranja istih uzoraka *labels_pred*, zajednička informacija je funkcija koja mjeri sporazum dvaju zadataka, ignorirajući permutacije. Dostupne su dvije različite normalizirane verzije te mjere, *Normalized Mutual Information* (NMI) i *Adjusted Mutual Information* (AMI). NMI se često koristi u literaturi dok je AMI nedavno predložen i normaliziran.

Svi, *mutual_info_score*, *adjusted_mutual_info_score* i *normalized_mutual_info_score* su simetrični: zamjenom argumenta ne mijenja se rezultat. Stoga se mogu koristiti kao konsenzusna mjera. Kada koristimo *adjusted_mutual_infro*_score savršeno označavanje je rezultat 1.0. Ovo ne vrijedi za *mutual_info_score*, što je stoga teže suditi. Loše (npr. nezavisno označavanje) ima negativne rezultate.

3.2.1. Prednosti

Dodjele slučajne (ravnomjerne) oznake imaju *AMI* rezultat blizu 0.0 za bilo koju vrijednost n_klastera i n_uzoraka.

Ograničeni raspon [0,1]: vrijednosti blizu nule naznačuju dvije dodjele oznake koje su u velikoj mjeri samostalne, dok vrijednosti blizu jedan pokazuju značajan dogovor. Nadalje, vrijednosti točno 0 označavaju dodjele čisto nezavisne oznake i *AMI* od točno 1 ukazuje to da dvije dodjele oznake su iste (sa ili bez permutacija).

Nema pretpostavke o strukturi klastera: može se koristiti za usporedbu algoritama klasteriranja kao k-means koji pretpostavlja izotropne oblike blobova s rezultatima algoritama spektralnih klasteriranja koji mogu pronaći klaster sa „presavijenim“ oblicima.

3.2.2. Nedostaci

Suprotno inerciji, mjere utemeljene na MI zahtijevaju poznavanje točno klasificiranih klasa, a gotovo nikada nisu dostupne u praksi ili zahtijevaju ručno dodjeljivanje ljudskim oznakama (kao u okruženju nadziranog učenja).

Međutim mjere utemeljene na MI se mogu također koristiti u čisto nenadziranom postavljanju kao građevinski blok za konsenzusni indeks koji se može koristiti za klasteriranje odabira modela.

NMI i MI nisu prilagođene slučajnosti.

3.2.3. Matematička formulacija

Pretpostavimo dvije dodjele oznake (istih N objekata), U i V. Njihova entropija je količina nesigurnosti za skup particija, definirana pomoću:

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i))$$

gdje $P(i) = |U_i|/N$ je vjerojatnost da slučajno odabrani objekt iz U spada u klasu U_i. Također za V:

$$H(V) = -\sum_{j=1}^{|V|} P'(j)\log(P'(j)).$$

sa $P'(j) = |V_j|/N$. Zajednička informacija (MI) između U i V izračunata je pomoću:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

gdje $P(i,j) = \left|U_i \cap V_j\right|/N$ je vjerojatnost da slučajno odabrani objekt spada u obe klase U_i i V_j.

Također se može izraziti u skupu formulacije kardinalnosti:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N}\log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right)$$

Normalizirana zajednička informacija je definirana kao:

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{H(U)H(V)}}.$$

Ova vrijednost zajedničke informacije i normalizirana varijanta također nije prilagođena za slučajnost te će se povećavati kako se broj različitih oznaka (klastera) povećava, bez obzira na stvarnu veličinu „zajedničke informacije“ između dodjeljivanja oznaka.

Očekivana vrijednost za zajedničku informaciju može se izračunati pomoću sljedeće jednadžbe, iz Vinh, Epps, i Baley [3]. U toj jednadžbi, $a_i = |U_i|$ (broj elemenata u Ui) i $b_j = |V_j|$ (broj elemenata u Vj).

$$E[MI(U,V)] = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|}\sum_{n_{ij}=max(a_i+b_j-N,0)}^{\min(a_i,b_j)} \frac{n_{ij}}{N}\log\left(\frac{N.n_{ij}}{a_i b_j}\right)\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}.$$

Koristeći očekivanu vrijednost, prilagođena zajednička informacija se može tada izračunati pomoću obrasca sličnog onom prilagođenog Rand indeksa:

$$AMI = \frac{MI - E[MI]}{\max\left(H(U), H(V)\right) - E[MI]}.$$

3.3. Homogenost, potpunost i V-mjera

S obzirom na poznavanje zadataka točno klasificirane klase uzoraka, moguće je definirati neka intuitivna mjerenja pomoću uvjetnih entropijskih analiza.

Posebno Rosenberg i Hirschberg [4] definiraju sljedeća dva poželjna cilja za bilo koji zadatak klastera. To su homogenost gdje svaki klaster sadrži samo članove jedne klase te potpunost gdje svi članovi određene klase su dodijeljeni istom klasteru

Te koncepte možemo pretvoriti u rezultate *homogeneity_score* i *completeness_score*. Oba su ograničena između 0.0 i 1.0 (veći je bolji). Njihova harmonijska sredina pod nazivom V-mjera se izračunava kao *v_measure_score*.

Homogenost, potpunost i V-mjera mogu se izračunati odjednom pomoću *homogeneity_completeness_v_measure*.

V_measure_score je simetričan: može se koristiti za ocjenjivanje suglasnosti dvaju nezavisnih zadataka na istom skupu podataka. To nije slučaj za *completeness_score* i *homogeneity_score* koja su oba vezana odnosom: *homogeneity_score(a, b) == completeness_score(b, a).*

3.3.1. Prednosti

Ograničeni rezultati: 0.0 je najlošiji, 1.0 je savršen rezultat.

Intuitivno tumačenje: klasteriranje sa lošom V-mjerom se može kvalitativno analizirati u smislu homogenosti i potpunosti kako bi se bolje vidjelo kakve pogreške su učinjene zadatkom.

Nema pretpostavke o strukturi klastera: može se koristiti za usporedbu algoritama klasteriranja kao što je k-means koji pretpostavlja izotropne oblike blobova s rezultatima algoritama spektralnog klasteriranja koji mogu pronaći klaster s „presavijenim“ oblicima.

3.3.2. Nedostaci

Prethodno uvedena mjerenja nisu normalizirana s obzirom na nasumično označavanje: to znači da, ovisno o broju uzoraka, klastera i točno klasificiranih klasa, potpuno slučajno označavanje neće uvijek dati iste vrijednosti za homogenost, potpunost i V-mjeru. Konkretno, slučajno označavanje neće dati nulte rezultate pogotovo kada je broj klastera velik.

Ovaj problem može se sigurno zanemariti kada je broj uzoraka veći od tisuću i broj klastera manji od 10. Za manje veličine uzoraka ili veći broj klastera, sigurnije je koristiti prilagođeni indeks kao što je prilagođeni Rand Indeks (ARI).

Ta mjerenja zahtjevaju poznvanje točno klasificiranih klasa, a gotovo nikada nije dostupan u praksi ili zahtjeva ručno dodjeljivanje ljudskim oznakama (kao u okruženju nadziranog učenja).

3.3.3. Matematička formulacija

Rezultati homogenosti i potpunosti formalno daju:

$$h = 1 - \frac{H(C|K)}{H(C)}$$

$$c = 1 - \frac{H(K|C)}{H(K)}$$

gdje $H(C|K)$ je uvjetna entropija klasa s obzirom na zadatke klastera i dana je:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot\log\left(\frac{n_{c,k}}{n_k}\right)$$

i $H(C)$ je entropija klasa koja je dana:

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\cdot\log\left(\frac{n_c}{n}\right)$$

s n ukupnim brojem uzoraka, n_c i n_k brojem uzoraka koji pripadaju klasi c i klasteru k, i napokon $n_{c,k}$ brojem uzoraka iz klase c dodijeljen klasteru k.

Uvjetna entropija klastera s obzirom na klasu $H(K|C)$ i entropija klastera $H(K)$ su definirane na simetričan način.

Rosenberg i Hirschberg [4] dalje definiraju V-mjeru kao harmonijsku sredinu homogenosti i cjelovitosti:

$$v = 2\cdot\frac{h\cdot c}{h+c}.$$

3.4. Koeficijent siluete

Ako točno klasificirane oznake nisu poznate, procjena mora biti izvedena pomoću samog modela. Koeficijent siluete (*sklearn.metrics.silhouette_score*) je primjer takve procjene, gdje se veći rezultat koeficijenta siluete odnosi na model s bolje definiranim klasterima. Koeficijent siluete je definiran za svaki uzorak i sastoji se od dva rezultata. Prvi rezultat *a* je srednja udaljenost između uzorka i svih ostalih točaka u istoj klasi, a drugi *b* je srednja udaljenost između uzorka i svih ostalih točaka u sljedećem najbližem klasteru.

Koeficijent siluete *s* za pojedinačni uzorak daje se kao:

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

Koeficijent siluete za skup uzoraka dan je kao sredina koeficijenta siluete za svaki uzorak. U normalnoj upotrebi, koeficijent siluete se primjenjuje na rezultatima klasterske analize.

3.4.1. Prednosti

Rezultat je ograničen između -1 za pogrešno klasteriranje i +1 za vrlo gusto klasteriranje. Rezultati oko nule ukazuju na preklapajuće klastere.

Rezultat je veći kada su klasteri gusti i dobro odvojeni, što se odnosi na standarni koncept klastera.

3.4.2. Nedostaci

Koeficijent siluete je općenito veći za konveksne klastere od ostalih koncepata klastera, kao što su klasteri na temelju gustoće poput onih dobivenih putem DBSCAN-a.

4. Primjena K-Means algoritma za klasteriranje dokumenata

Prvi korak klasteriranja je tokenizacija. Tokenizacija je proces raščlanjivanja tekstualnih podataka u manje jedinice (tokene) kao što su riječi i fraze. Svaku riječ ispisuje jednu ispod druge te tu spadaju i interpunkcijski znakovi i svi ostali znakovi koji su u tekstu. U nastavku je prikazan dio koda implementiran u Scikit-Learn modulu u Pythonu koji se odnosi na postupak tokenizacije prikazan na primjeru.

```
sents = [sent for sent in nltk.sent_tokenize("Buddhism [1][2] is the world\'s
fourth-largest religion[3][4] with over 520 million followers, or over 7% of the
global population, known as Buddhists.")]
print(sents)
print()
words = [word for word in nltk.word_tokenize(sents[0])]
print(words)
print()
```

```
["Buddhism [1][2] is the world's fourth-largest religion[3][4] with over 520
million followers, or over 7% of the global population, known as Buddhists."]
['Buddhism', '[', '1', ']', '[', '2', ']', 'is', 'the', 'world', "'s", 'fourth-
largest', 'religion', '[', '3', ']', '[', '4', ']', 'with', 'over', '520',
'million', 'followers', ',', 'or', 'over', '7', '%', 'of', 'the', 'global',
'population', ',', 'known', 'as', 'Buddhists', '.']
```

Uobičajeno je izbaciti sve interpunkcijeske i ostale nevažne znakove koji nisu bitni za klasteriranje. neke riječi kao što su primjerice „je“, „na“, „koji“ i ostale slične riječi neće biti od velike koristi za otkrivanje bitnih značajki teksta. Takve se riječi nazivaju zaustavne riječi odnosno „stopwords“. Tako da ih je potrebno ukloniti prije daljnje analize. Taj je postupak prikazan u dijelu koda koji slijedi.

```
def tokenize_only(text):
    tokens = [word.lower() for sent in nltk.sent_tokenize(text) for word in
nltk.word_tokenize(sent)]
    filtered_tokens = []
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)
    return filtered_tokens
words_only = tokenize_only("Buddhism [1][2] is the world\'s fourth-largest
religion[3][4] with over 520 million followers, or over 7% of the global
population, known as Buddhists.")
print(words_only)
print()
```

```
['buddhism', 'is', 'the', 'world', "'s", 'fourth-largest', 'religion', 'with',
'over', 'million', 'followers', 'or', 'over', 'of', 'the', 'global',
'population', 'known', 'as', 'buddhists']
```

Treći korak je korjenovanje (stemming) i lematizacija gdje različiti tokeni mogu provoditi slične informacije (npr. tokenizacija i tokeniranje). Možemo izbjeći višekratno izračunavanje sličnih informacija smanjivanjem svih tokena u osnovni oblik koristeći različite rječnike za korjenovanje i lematizaciju. Dio koda u kojem je implementiran postupak korjenovanja u Scikit-Learn modulu na primjeru prikazan je u nastavku.

```
def tokenize_and_stem(text):
    tokens = [word for sent in nltk.sent_tokenize(text) for word in
nltk.word_tokenize(sent)]
    filtered_tokens = []
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)
    stems = [stemmer.stem(t) for t in filtered_tokens]
    return stems
words_stemmed = tokenize_and_stem("Buddhism [1][2] is the world\'s fourth-
largest religion[3][4] with over 520 million followers, or over 7% of the global
population, known as Buddhists.")
print(words_stemmed)
print()

['buddhism', 'is', 'the', 'world', "'s", 'fourth-largest', 'religion', 'with',
'over', 'million', 'follow', 'or', 'over', 'of', 'the', 'global', 'popul',
'known', 'as', 'buddhist']
```

Nakon predprocesiranja tekstualnih podataka, možemo nastaviti generirati značajke. Za grupiranje dokumenata, jedan od najčešćih načina generiranja značajki dokumenta je izračunavanje pojma frekvencije svih njegovih tokena (tf-idf). Iako nisu savršene, ove frekvencije obično mogu dati neke naznake o temi dokumenta. Ponekad je korisno izraziti pojam frekvencije inverznim frekvencijama dokumenta koji nastoji prikazati koliko je riječ važna dokumentu u korpusu. Slijedi kod za taj dio postupka u Scikit-Learn modulu.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=200000,
                                   min_df=0.2, stop_words='english',
                                   use_idf=True,tokenizer=tokenize_and_stem,
                                   ngram_range=(1,3))

%time tfidf_matrix = tfidf_vectorizer.fit_transform(synopses)
print(tfidf_matrix.shape)
terms = tfidf_vectorizer.get_feature_names()
print(len(terms))

Wall time: 157 ms
(9, 392)
392
```

Zatim možemo grupirati različite dokumente na temelju značajki koje smo generirali. Pri tome nam pomaže tehnika klasterske analize čiji je kod prikazan niže.

```
from sklearn.cluster import KMeans
num_clusters = 3
km = KMeans(n_clusters=num_clusters)
%time km.fit(tfidf_matrix)
clusters = km.labels_.tolist()

from sklearn.externals import joblib
joblib.dump(km,  'doc_cluster.pkl')

km = joblib.load('doc_cluster.pkl')
clusters = km.labels_.tolist()
print(clusters)
print()

text = { 'title': titles, 'rank': ranks, 'synopsis': synopses, 'cluster':
clusters, 'genre': genres }
frame = pd.DataFrame(text, index = [clusters] , columns = ['rank', 'title',
'cluster', 'genre'])
print(frame)
print()
frame['cluster'].value_counts()

from __future__ import print_function
print("Top terms per cluster:")
print()
order_centroids = km.cluster_centers_.argsort()[:, ::-1]

for i in range(num_clusters):
    print("Cluster %d words:" % i, end='')

    for ind in order_centroids[i, :6]:
        print(' %s' % vocab_frame.ix[terms[ind].split('
')].values.tolist()[0][0].encode('utf-8', 'ignore'), end=',')
    print()
    print()

    print("Cluster %d titles:" % i, end='')
    for title in frame.ix[i]['title'].values.tolist():
        print(' %s,' % title, end='')
    print()
    print()

print()
print()

Wall time: 40 ms
[2, 2, 2, 0, 0, 0, 1, 1, 1]

   rank         title  cluster     genre
2     1        Techno        2     Music
```

```
2     2         Dance        2     Music
2     3          Rock        2     Music
0     4     Left-wing        0  Politics
0     5    Right-wing        0  Politics
0     6  Barack Obama        0  Politics
1     7  Christianity        1  Religion
1     8      Buddhism        1  Religion
1     9         Islam        1  Religion

Top terms per cluster:

Cluster 0 words: b'right', b'left', b'supports', b'act', b'term', b'social',

Cluster 0 titles: Left-wing, Right-wing, Barack Obama,

Cluster 1 words: b'islamic', b'teaching', b'practices', b'asia', b'religion',
b'world',

Cluster 1 titles: Christianity, Buddhism, Islam,

Cluster 2 words: b'music', b'rock', b'dance', b'dance', b'styles', b'producer',

Cluster 2 titles: Techno, Dance, Rock,
```

Zadnji korak je evaluacija i vizualizacija. Modeli klasteriranja se mogu procijeniti različitim mjernim podacima, a ponekad je korisno vizualizirati rezultate tako što prebacuju klastere u niski (dvo) dimenzionalni prostor.

5. Zaključak

Klasteriranje se može koristiti u različitim znanstvenim područjima, primjerice u biologiji u grani genetike, zatim kod različitih istraživanja kao što su podjela populacije u segmente potrošača ili potraživača kako bi se njihovi odnosi bolje razumjeli. Naravno, može se primjeniti i u ostalim područjima te nam tako olakšati posao koji je zahtjevan kod većih skupova podataka.

Klasteriranje se dijeli na više algoritama koji funkcioniraju svaki na svoj način. Ovi algoritmi su grupirani u dvije glavne grupe, i to u hijerarhijske i nehijerarhijske algoritme.

Hijerarhijski algoritmi su prikladniji za pregledavanje ali imaju problem učinkovitosti. *Agglomerative Clustering* je dobar kada nemamo nužno kružne klastere i ne znamo broj klastera unaprijed.

Nehijerarhijski algoritmi su učinkovitiji i pružaju dovoljno informacija za mnoge svrhe. K-Means je dobar za korištenje jer je jednostavan i brzina mu dopušta izvođenje na velikim skupovima podataka. Nikad ne daje isti rezultat sa svakim novim radom algoritma. *Birch* ne skalira baš najbolje velike dimenzijske podatke. Kao pravilo, ako je *n_značajki* veći od dvadeset, obično je bolje koristiti MiniBatch K-Means. Ako broj instanci podataka treba biti smanjen, ili ako je potreban veliki broj podklastera bilo kao predprocesni korak ili na neki drugi način, *Birch* je mnogo korisniji nego MiniBatch K-Means. DBSCAN se bavi različitim oblicima klastera tako što su grupe točaka blisko povezane i onda se klasteri proširuju u bilo kojem smjeru gdje se nalaze obližnje točke. Glavni nedostatak *Affinity Propagation*-a je kompleksnost, ali je prikladan za klasteriranje manjih skupova podataka. Kod Mean Shift-a je pozitivno to što ima samo jedan parametar ali je računalno skup te se ne slaže dobro sa dimenzijom značajnog prostora. Spektralno klasteriranje je elegantan i matematički dobro utemeljen ali skupovi podataka sa šumovima stvaraju problem zato što se izvođenje može naglo promijeniti iz dobrog na strašno.

Učinkovitost tih algoritama može biti ocjenjena raznim mjerama. Zapravo se ocjenjuje postavljanje oznake algoritma, tako mjenjanjem nekih oznaka rezultat ostaje isti, rezultat 1.0 je savršen, a 0.0 najgori, i tako svaka mjera ocjenjivanja uz to ocjenjivanje ima prednosti i nedostatke.

Za klasteriranje se većinom uzima veći skup podataka pa možemo zaključiti da je K-Means jedan od boljih algoritama klasteriranja. U ovom je radu analizirana primjena K-Means algoritma za klasteriranje dokumenata popraćeno kodom i pojašnjenjima samog algoritma.

6. Popis slika

7. Popis literature

[1] 2.3. Clustering — scikit-learn 0.19.2 documentation, scikit-learn developers, http://scikit-learn.org/stable/modules/clustering.html#clustering, 10.6.2018.

[2] k-means++: The Advantages of Careful Seeding, D. Arthur, S. Vassisvitskii, http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf, 12.9.2018.

[3] Vinh N.X., Epps J., Bailey J. (2009.): Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance

[4] Rosenberg A., Hirschberg J. (2007.): V-Measure: A conditional entroy-based external cluster evaluation measure

[5] Document clustering, Wikipedia Foundation, Inc., https://en.wikipedia.org/wiki/Document_clustering, 9.9.2018.

[6] Document Clustering with Python, Brandon Rose, http://brandonrose.org/clustering, 9.9.2018.

[7] Klasterska analiza, Diplomski rad, Tea Ungaro, http://digre.pmf.unizg.hr/5298/1/Diplomski%20rad%20Tea%20Ungaro.pdf, 10.9.2018.