KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Mindaugas Venckus

MIŠRIŲJŲ DUOMENŲ KLASTERIZAVIMAS TAIKANT INFORMACIJOS ENTROPIJĄ

Baigiamasis magistro projektas

Vadovė
Doc. dr. Kristina Šutienė
Konsultantas
Dr. Mindaugas Kavaliauskas

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

MIŠRIŲJŲ DUOMENŲ KLASTERIZAVIMAS TAIKANT INFORMACIJOS ENTROPIJĄ

Baigiamasis magistro projektas
M4026M21 Taikomoji matematika (621G10003)

Vadovė
Doc. dr. Kristina Šutienė
2016-05-30

Recenzentas
Doc. dr. Tomas Ruzgas
2016-05-30

Projektą atliko
Mindaugas Venckus
2016-05-30

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Matematikos ir gamtos mokslų fakultetas
(Fakultetas)

Mindaugas Venckus
(Studento vardas, pavardė)

Taikomoji matematika, 621G10003
(Studijų programos pavadinimas, kodas)

„Mišriųjų duomenų klasterizavimas taikant informacijos entropiją“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2016 m. 05 30 d.
Kaunas

Patvirtinu, kad mano, **Mindaugo Venckaus**, baigiamasis projektas tema „Mišriųjų duomenų klasterizavimas taikant informacijos entropiją“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

TURINYS

ILIUSTRACIJŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

Venckus, Mindaugas. MIŠRIŲJŲ DUOMENŲ KLASTERIZAVIMAS TAIKANT INFORMACIJOS ENTROPIJĄ. Magistro baigiamasis darbas vadovas, doc. dr. Kristina Šutienė, konsultantas dr. Mindaugas Kavaliauskas; Kauno technologijos universitetas, matematikos ir gamtos mokslų fakultetas.

Mokslo kryptis ir sritis: Fiziniai mokslai, matematika.

Reikšminiai žodžiai: Klasterizavimas, mišrieji duomenys, informacijos entropija, klasterizavimo validavimas.

Kaunas, 2016. p. 45

SANTRAUKA

Magistriniame darbe ištirta mišriųjų duomenų klasterizavimo problematika bei pasiūlyti jos sprendimai. Mišriųjų duomenų klasterizavimo algoritmas realizuotas remiantis straipsniu *J. Liang, et al., Determining the number of clusters using information entropy for mixed data, Pattern Recognition (2012), doi:10.1016/j.patcog.2011.12.017*. Atliktos modifikacijos: multidimensinis masteliavimas, neatsitiktinis pradinių centrų parinkimas, pritaikyti Davies-Bouldin ir Dunn indeksai mišriųjų duomenų klasterizavimo rezultatų validavimui. Modifikuotas algoritmas buvo testuotas su sintetinėmis ir realiomis duomenų aibėmis. Realūs duomenys parinkti tokie, kad būtų iš anksto žinomos grupės, tačiau prieš klasterizavimą jos pašalinamos. Tirtoms duomenų imtims tikrasis klasterių skaičius buvo nustatytas teisingai.

Venckus, Mindaugas. APPLICATION OF INFORMATION ENTROPY FOR MIXED DATA CLUSTERING: Master thesis supervised by assoc. prof. Kristina Šutienė, dr. Mindaugas Kavaliauskas. The Faculty of Mathematics And Natural Science, Kaunas University Of Technology.

Research area and field: Physical sciences, mathematics.

Key words: Clustering, mixed data, information entropy, cluster validation

Kaunas, 2016. p. 45

SUMMARY

The main topic of this paper is mixed data clustering analysis and implementation. Clustering algorithm was implemented based on *J. Liang, et al., Determining the number of clusters using information entropy for mixed data, Pattern Recognition (2012), doi:10.1016/j.patcog.2011.12.017* paper. Main contributions are: mixed datatype visualization using multidimensional scaling method, non random initial centers selection, validation of clustering results using Dunn and Davies-Bouldin indices. Modified algorithm was tested with synthetical and real data sets. Originally real data sets were with known groups, but before clustering labels from real data sets were removed. In all cases the number of true clusters were detected correctly.

1 ĮŽANGA

Panašių objektų grupavimas į skirtingas grupes žmonėms buvo suvokiamas nuo seno, tačiau pirmieji matematiniai algoritmai pradėti aprašyti tik XX a. viduryje. Klasterizavimo pagrindinė idėja yra surasti klasterius, kuriuose atstumai tarp taškų yra minimalūs, bet atstumai tarp klasterių yra maksimalūs. Dvimačiu atveju, kai duomenys yra tolydieji, ši problema žmogui dažniausiai yra intuityviai išsprendžiama. Klasterizavimas, kaip tyrimų įrankis, pastarąjį dešimtmetį labai išpopuliarėjo dėl taikymo galimybių. Taikymo pavyzdžių galime rasti biologijoje, botanikoje, medicinoje, psichologijoje, astronomijoje, prekyboje ir kitur. Prekybos analitikai klasterizuoja pirkėjus į grupes pagal pirkinių krepšelį - tokiu būdu jie gali efektyviau reklamuoti bei parduoti prekes. Astronomai grupuoja žvaigždžių spiečius ir tokiu būdu atranda nepastebėtų struktūrų. Bioinformatikai klasterizuoja DNR grupes. Didėjant duomenų kiekiui bei jų įvairovei, klasikiniai klasterizavimo metodai, tinkami klasterizuoti tik tolydžiuosius arba diskrečiuosius duomenis atskirai, nėra tinkami, todėl tyrėjai pastaruoju metu koncentruojasi į mišriųjų duomenų klasterizavimo tobulinimą.

Magistrinio **darbo tikslas** yra sukurti mišriųjų duomenų klasterizavimo algoritmą nekonvertuojant duomenų į vieną tipą bei pasiūlyti validavimo indeksą, kuris būtų taikomas mišriųjų duomenų klasterizavimo kokybei nustatyti.

Darbo uždaviniai:

1. Atlikti tolydžiųjų, diskrečiųjų, mišriųjų duomenų klasterizavimo analitinę apžvalgą.
2. Atlikus literatūros analizę pasiūlyti algoritmą, kuris yra paremtas informacijos entropija.
3. Pasiūlyti mišriųjų duomenų klasterizavimo validavimo indeksą, kuris leistų parinkti rekomenduotiną klasterių skaičių.
4. Algoritmą pritaikyti klasterizuojant sintetines ir realias duomenų aibes.

Atlikti darbai:

1. Realizuotas mišriųjų duomenų klasterizavimo algoritmas, remiantis *J. Liang, et al., Determining the number of clusters using information entropy for mixed data, Pattern Recognition (2012), doi:10.1016/j.patcog.2011.12.017* metodu (žr. 3.1, 3.2 skyriai).
2. Algoritmas patobulintas pritaikius klasterizavimo rezultatų validavimo indeksus Davies-Bouldin ir Dunn (žr. 3.4 skyrius). Naudojantis jais nustatomas rekomenduotinas klasterių skaičius to reikėjo, nes autorių pateiktas metodas veikė netenkinamai.
3. Algoritmas patobulintas realizavus neatsitiktinį pradinių centrų pasirinkimą (žr. 3.5 skyrius). Naudojantis juo išvengiama mažai nutolusių pradinių centrų parinkimo bei pagreitinamas konvergavimas.
4. Algoritmas patobulintas realizavus multidimensinio masteliavimo metodą (žr. 3.6 skyrius). Naudojantis juo galima analizuoti klasterizavimo rezultatą, kuris yra suprojektuotas dvimatėje Euklido erdvėje.

5. Algoritmas ištestuotas su sintetinėmis ir realiomis duomenų aibėmis (žr. 4 skyrius).
6. Sudalyvauta konferencijoje „Matematika ir matematikos dėstymas – 2016“, kuri vyko 2016 Balandžio 8 dieną.

2 LITERATŪROS APŽVALGA

2.1 KLASTERIZAVIMO PROBLEMATIKA

Nors klasterizavimas yra plačiai naudojamas praktikoje, tačiau iki dabar uždaviniai susiję su klasterizavimu yra komplikuoti. Problemos kyla dėl fundamentalių klasterizavimo sąvokų nebuvimo bei atstumų funkcijų maišaties. Žemiau pateiktos problemos, kurios iki dabar yra aktualios klasterizavimo uždaviniuose [1][p. 11].

1. Kas yra klasteris?
2. Kuriuos atributus pasirinkti klasterizavimui?
3. Ar reikia normalizuoti duomenis?
4. Ar duomenys turi išskirčių?
5. Kokią panašumo funkciją pasirinkti?
6. Kokį klasterių skaičių pasirinkti?
7. Kurį klasterizavimo algoritmą pasirinkti?
8. Ar galima duomenyse įžvelgti klasterizavimo tendencija?

Didžioji dalis šių klausimų neturi aiškaus atsakymo, todėl klasterizavimas yra opi problema tyrimuose. Darbe bus mėginama atsakyti į duomenų normalizavimo, klasterių kiekio, klasterizavimo algoritmo pasirinkimo klausimus.

2.2 PAGRINDINIAI POŽIŪRIAI Į KLASTERIZAVIMĄ

Vartotojui viena sunkiausių užduočių yra parinkti tinkamą algoritmą klasterizavimui, nes egzistuoja daug variantų ir gauti rezultatai dažnai skiriasi. Trumpai apžvelgsime pagrindinius požiūrius į klasterizavimą. Klasteriais gali būti vadinamos tankios zonos atskirtos retomis. Egzistuoja nevienas algoritmas, kuris ieško kaip sujungti tankias zonas. **Jarvis-Patrick** algoritme [2] panašumas tarp dviejų taškų yra apibrėžiamas kaip bendrų taškų skaičius pasirinktame regione. **DBSCAN** algoritmas yra panašus į **Jarvis-Patrick** - ieškoma tankių zonų, kur tankumas yra įvertinamas naudojant (angl. *Parzen windom method*). Pagrindiniai faktoriai įtakojantys **Jarvis-Patrick** ir **DBSCAN** [3] klasterizavimo rezultatą yra spindulys, kuriame ieškoma bendrų taškų ir minimalus skaičius kaimyninių taškų. Tokio tipo algoritmų pagrindinis privalumas yra, jog dažnai atrandamos ne iškilios formos, o trūkumai atsiranda jeigu bandoma klasterizuoti didelės dimensijos duomenis, tada nerandama aiškių, tankių zonų.

Jeigu duomenys yra didelės dimensijos dažnai naudojamas **CLIQUE** [4] algoritmas. Jis yra greitas ir pagrindinis algoritmo principas yra tyrinėti duomenyse poaibius atrandant tankias zonas ir jas vėliau apjungti.

Grafais paremtas požiūris į klasterizavimą remiasi tuo, kad taškai laikomi viršūnėmis grafe su svoriais. Viršūnės sujungtos briaunomis, kur svoris nustatomas pagal viršūnių tarpusavio panašumą. Pagrindinė idėja yra, jog viršūnės atskiriamas į dvi grupės A ir B taip, kad svorių sumą tarp viršūnių yra minimizuojama naudojant minimalaus kirtimo metodą.

Nuo 2000 metų tarp tyrėjų išpopuliarėjo klasterizavimo algoritmai, kurie yra paremti informacijos entropijos elementais. Pavyzdžiui Roberts straipsnyje [5] aprašoma idėja, kur minimizuojama bendra sistemos entropija. Pirmiausia traktuojama, kad duomenys yra sugeneruoti naudojant mišinių modelį (angl. *mixture model*) ir kiekvienas klasteris sumodeliuotas pagal pusiau parametrinį tikimybinį tankį (angl. *semi-parametric probability density*). Parametrai randami maksimizuojant Kullback–Leibler divergenciją tarp nesąlyginių ir sąlyginių tankių.

2.3 KLASTERIZAVIMO ALGORITMŲ KLASIFIKAVIMAS

Klasterizavimo algoritmai gali būti klasifikuojami, pagal šaltinį [6][p. 110], į:

1. *Atstumo funkcija paremti algoritmai*. Šiai grupei priklauso populiarus k-vidurkių algoritmas. Tokio tipo algoritmai duomenų aibę išskaido į poaibius pagal atstumo funkciją. Šios kategorijos algoritmai yra populiariausi, nes yra greiti, leidžia klasterizuoti tiek tolydžiuosius, tiek diskrečiuosius duomenis bei dažniausiai yra intuityviai suvokiami vartotojui.

2. *Hierarchinis klasterizavimas*. Tokio tipo algoritmais paeiliui apjungiami maži klasteriai į didesnius arba antraip. Rezultate gaunama dendrograma pagal kurią matoma bendra duomenų struktūra bei leidžia parinkti potencialų klasterių skaičių. Hierarchinio tipo algoritmai yra neefektyvūs didelėms duomenų imtims, nes kompiuterio atmintyje saugoma daug tarpinių skaičiavimų.

3. *Klasterizavimas paremtas tankio įverčiu*. Populiariausias šios grupės algoritmas DBSCAN, kuriuo naudojantis duomenys klasterizuojami remiantis ϵ reikšme - nagrinėjamas kiekvienas taškas ir žiūrima, kurie taškai priklauso jam ϵ aplinkoje. Tokio tipo algoritmai gali suformuoti netikėtas grupes bei išvengti triukšmo.

4. *Blokų skaidymo klasterizavimas*. Tokio tipo algoritmai įprastai naudojami klasterizuojant multidimensinius duomenis. Erdvė suskaidoma į blokus ir juose atliekamas klasterizavimas, po to rezultatai apjungiami.

Magistriniame darbe mišriųjų duomenų klasterizavimo algoritmas bus paremtas atstumo funkcija. Pagrindinis privalumas yra greitis, nes tokio tipo algoritmai įprastai yra $O(n^2)$ sudėtingumo.

2.4 TOLYDŽIŲJŲ DUOMENŲ KLASTERIZAVIMAS

Daugiausiai ištirta homogeninių grupių klasterizavimas. Praktikoje klasterizuojant tolydžiuosius duomenis tyrėjai dažniausiai naudoja **k-vidurkių** [7] algoritmą, kur pagrindinės problemos - lokalus konvergavimas, pradinių centrų parinkimas, klasterių skaičiaus nustatymas. Patobulintas algoritmas buvo pristatytas 2007 metais, kuris vadinasi **k-vidurkių++** [8]. Algoritmo pagrindinis

privalumas yra neatsitiktinis pradinių centrų parinkimas, tačiau padidėja skaičiavimo sąnaudos. Pirmiausia parenkamas atsitiktinis taškas, vėliau apskaičiuojami atstumai iki to centro nuo kiekvieno taško, tada kitas centras parenkamas remiantis su tikimybe proporcinga atstumo kvadratui. Taip pat dažnai naudojamas hierarchinio tipo algoritmas **BIRCH** [9], kuris tinkamas didelių duomenų klasterizavimui, tačiau problemos atsiranda jeigu duomenyse natūralūs klasteriai sudaro ne sferines arba ne iškilas formas (angl. *non convex*).

2.5 DISKREČIŲJŲ DUOMENŲ KLASTERIZAVIMAS

Praktikoje dažnai iškyla problema, jog duomenys yra netolydūs. Šiame skyriuje pateiksiu tokio tipo duomenų klasterizavimo populiariausius algoritmus. Vienas populiariausių algoritmų naudojamų diskrečių kintamųjų klasterizavime yra **k-modų** [10], kuris yra **k-vidurkių** metodo modifikacija. Čia vietoj Euklido atstumo dažniausiai naudojamas Hamingo, tačiau šis algoritmas taip pat paveldi **k-vidurkių** algoritmo trūkumus. Sudipto Guha, Rajeev Rastogi, Kyuseok Shim straipsnyje [13] parodoma, kad tradiciniai algoritmai, kurie naudoja atstumo funkciją yra nekorektiški kategoriniams kintamiesiems. Jų pristatytas algoritmas **ROCK** remiasi hierarchinio klasterizavimo idėjomis, kur sudaromi ryšiai tarp kategorinių duomenų. Literatūros šaltiniuose taip pat dažnai minimas **COOLCAT** 2002 m. [11] algoritmas, kuris yra paremtas informacijos entropijos sąvoka.

2.6 MIŠRIŲJŲ DUOMENŲ KLASTERIZAVIMAS

Pirmiausia pateikiamas mišriųjų duomenų pavyzdys, kad skaitytojas pamatytų kokio tipo duomenis klasterizuosime. 1 lentelėje pateiktas mišriųjų duomenų pavyzdys, kur atributai metai ir darbo valandos žymi tolydžiųjų duomenų poaibį, o išsilavinimas, rasė, lytis ir uždarbis - diskrečiųjų. Mano tikslas - tokio tipo duomenyse rasti natūralius klasterius.

1 lentelė: Mišrių duomenų pavyzdys

metai	išsilavinimas	rasė	lytis	darbo valandos	uždarbis
39	Bachelors	White	Male	40	<=50K
50	Bachelors	White	Male	13	<=50K
38	HS-grad	White	Male	40	<=50K
53	11th	Black	Male	40	<=50K
28	Bachelors	Black	Female	40	<=50K
37	Masters	White	Female	40	<=50K
49	9th	Black	Female	16	<=50K
52	HS-grad	White	Male	45	>50K
31	Masters	White	Female	50	>50K
42	Bachelors	White	Male	40	>50K

Mišrių duomenų klasterizavimo problematika nėra detaliai išnagrinėta, tačiau dažniausiai bandoma heterogeninę duomenų aibę paversti į homogeninę, naudoti universalias atstumo funkcijas, atlikti klasterizavimą atskirai tolydžiųjų ir diskrečiųjų duomenų poaibiams.

2.6.1 DISKREČIŲJŲ KINTAMŲJŲ VERTIMAS Į TOLYDŽIUOSIUS

Diskrečiųjų duomenų vertimas į tolydžiuosius realizuotas Ming-Yi Shih straipsnyje[12]. Algoritmo pagrindinis tikslas yra rasti ryšius tarp diskrečiųjų kintamųjų - ieškoma pasikartojančių kategorijų. Po to taikomas hierarchinio tipo algoritmas HAC, kuris skirtas suklasterizuoti homogeninę duomenų aibę. Toliau parodoma kaip randami ryšiai tarp diskrečiųjų kintamųjų.

2 lentelė: Diskrečiųjų duomenų testinė imtis

a1	a2
A	C
A	C
A	D
B	D
B	C
B	E
A	D

Sukonstruosime ryšių matricą M tokiu būdu: matrica bus kvadratinė ir turės 5 eilutes, nes egzistuoja 5 skirtingi kintamieji. Reikėtų įsivaizduoti, kad eilutėse ir stulpeliuose indeksai $1, n$ atitinka imties skirtingus elementus A, B, C, D, E. Elementas $m_{11} = 4$, nes reikšmė A pasirodo keturis kartus, o elementas $m_{1,3} = 2$, nes reikšmių pora AC pasirodo du kartus.

$$M = \begin{pmatrix} 4 & 0 & 2 & 2 & 0 \\ 0 & 3 & 1 & 1 & 1 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Pagrindinės diskrečiųjų kintamųjų vertimo į tolydžiuosius kintamuosius problemos yra informacijos iškreipimas ir netrivialus realaus skaičiaus priskyrimas kategoriniam kintamajam, pavyzdžiui, žalia, raudona [16] p.5.

2.6.2 TOLYDŽIŲJŲ KINTAMŲJŲ VERTIMAS Į DISKREČIUOSIUS

SpectralCAT[18] algoritmas remiasi idėja, jog sukuriama homogeninių duomenų aibė paverčiant tolydžiųjų duomenų poaibį į diskrečiųjų. Tai yra padaroma surandant optimalią transformaciją pagal *Calinski-Harabasz* indeksą apskaičiuotą kiekvienam įrašui. Čia taip pat iškyla informacijos iškreipimo ar praradimo problema [16] p.5.

2.6.3 UNIVERSALIOS ATSTUMO FUNKCIJOS

Paprasčiausia mišriųjų duomenų atstumo funkcija yra **k-prototipų**, kur sujungtos idėjos iš **k-vidurkių** ir **k-modų** algoritmų. Tegul $a_t = (t_1, ..., t_o)$ - tolydžios komponentės, o $a_d = (d_1, ..., d_p)$

- diskrečios komponentės, $a_m = a_t \cup a_d$, kur mišriųjų duomenų komponentę sudaro $s = o + p$ elementų. Tada galime apibrėžti **k - prototipų** atstumo metriką mišriųjų duomenų įrašams **x,y**

$$d(x_m, y_m) = H(x_d, y_d) + E(x_t, y_t).$$

Goverio panašumo indeksas (angl. *Gower's similarity coefficient*) taip pat dažnai naudojamas mišriųjų duomenų klasterizavime, kuris buvo pristatytas straipsnyje [19] 1971 m. Tarkime, kad norime apskaičiuoti panašumą tarp **x** ir **y** mišriųjų duomenų įrašų naudojant Goverio panašumo indeksą.

$$s_{gov}(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{s} w(x_k, y_k) s(x_k, y_k)}{\sum_{i=1}^{s} w(x_k, y_k)},$$

čia $s(x_k, y_k)$ yra panašumas priklausomas nuo k-tojo atributo, o $w(x_k, y_k)$ įgyja 0 arba 1 reikšmes. Toliau pateikiamos sąlygos

1. Jeigu k-tasis atributas žymi tolydžiuosius duomenis, tada $s(x_k, y_k) = 1 - \frac{|x_k - y_k|}{R_k}$, čia R_k - atstumas tarp didžiausios ir mažiausios reikšmių, $w(x_k, y_k) = 0$, jei **x** ir **y** įrašai k-tojo atributo vietoje turi tuščių elementų, kitu atveju $w(x_k, y_k) = 1$.

2. k-tasis atributas žymi diskrečiuosius duomenis, tada $s(x_k, y_k) = 1$, $w(x_k, y_k) = 1$, jeigu $x_k = y_k$ kitu atveju $s(x_k, y_k) = 0$, $w(x_k, y_k) = 0$.

2.6.4 MIŠRIŲ DUOMENŲ POAIBIŲ KLASTERIZAVIMAS

Dileep Kumar Murala straipsnyje [17] pateikta paprasta idėja, kurios dėka klasterizuojami mišrūs duomenys. Toliau pateikiamas jame aprašytas algoritmas.

- Mišrių duomenų aibė *D* išskaidoma į kategorinių kintamųjų aibę *CD* ir tolydžiųjų kintamųjų aibę *ND*.
- *CD* aibė klasterizuojama naudojant populiarų kategorinių kintamųjų klasterizavimo algoritmą CACTUS.
- *ND* aibė klasterizuojama naudojant populiarų kategorinių kintamųjų klasterizavimo algoritmą CHAMELEON.
- Kiekvienam aibės *D* įrašui priskiriami klasterių numeriai, gauti 2 ir 3 žingsnių metu.
- Numeriai suklasterizuojami naudojant CACTUS algoritmą ir priskiriame gautus klasterius aibei *D*.

Pagrindinis algoritmo trūkumas yra tas, kad remiantis dviem atributais (klasterių numeriais) yra nusprendžiama kuriam klasteriui priskiriamas mišrus įrašas.

2.7 KLASTERIZAVIMO REZULTATŲ VALIDAVIMAS

Egzistuoja du klasterizavimo rezultatų validavimo požiūriai:

1. Vidinis kriterijus (angl. *internal criteria*). Validavimo indeksas yra paremtas tik turimais duomenimis.

2. Išorinis kriterijus (angl. *external criteria*). Validavimo indeksas yra paremtas iš anksto žinoma informacija, tačiau praktikoje šitas indeksas dažniausiai yra netaikomas.

Toliau aptariami pagrindiniai vidiniai validavimo indeksai, kurie yra naudojami klasterizavime pagal [21] straipsnį. Dažniausiai naudojamas validavimo indeksas yra **Silueto** (angl. *Silhouette index*), kurio formulė pateikta 1 išraiškoje

$$k^* = \arg\max_k \left(SI(k) = \frac{1}{n} \sum_{n}^{i=1} \frac{(b_i - a_i)}{max(b_i, a_i)} \right), \tag{1}$$

čia a_i žymi i-tojo įrašo vidutinį atstumą iki kitų įrašų esančių klasteryje, kuriam i-tasis įrašas priklauso. Koeficientas b_i žymi i-tojo įrašo minimalų vidutinį atstumą iki taškų esančių kituose klasteriuose. Algoritme realizuoti **Duno** (angl. *Dunn index*) ir **Davies-Bouldin(DB)** indeksai, kurie yra aptarti 3.4 skyriuje.

2.8 INFORMACIJOS ENTROPIJA

Magistriniame darbe bus taikomi informacijos entropijos elementai nustatant blogiausią klasterį, todėl šiame skyriuje pristatomos pagrindinės sąvokos bei entropijos taikymo pavyzdžiai. Entropija yra matas, įvertinantis atsitiktinio dydžio X neapibrėžtumą. Jeigu $S(X)$ yra atsitiktinio dydžio įgyjamos reikšmės, o $p(x)$ jo tikimybė, tai entropija $E(X)$ yra

$$E(X) = \sum_{x \in S(x)} p(x) log(p(x)).$$

Kalbant paprastai, entropija yra netvarkos matas, kuris, pagal apibrėžimą, tinka klasterizavime. Šis matas naudojamas nustatant kategorinių duomenų klasterių skaičių Tao Li straipsnyje [24]. Taip pat įrodoma, kad sujungus du panašius klasterius į vieną, vidutinė entropija pastebimai nepakinta. Jau minėtame **COOLCAT** algoritme entropijos sąvoka vaidina pagrindinį vaidmenį kategorinių kintamųjų klasterizavime. Tarkime, kad kategorinių duomenų aibė yra išskaidyta į k klasterių $C_1, ..., C_k$, tada entropijos įvertis yra užrašomas kaip

$$E = \sum_k \left(\frac{|P(C_k)|}{|D|} (E(P(C_k))) \right).$$

Čia $E(P(C_k)$ žymi k-tojo klasterio entropiją, $|D|$ - duomenų aibės dydis. **COOLCAT** algoritmo pagrindiniai žingsniai yra

1. Randami tinkami klasteriai iš duomenų poaibio.

2. Likę taškai priskiriam pradiniam klasteriams paeiliui.

Entropijos taikymo pavyzdžių galima rasti ir rezultatų validavime vienas iš metodų yra aprašytas straipsnyje [25].

2.9 ANALITINĖS ANALIZĖS APIBENDRINIMAS

Atlikus literatūros analizę galima daryti išvadas, kad kokybiškas mišrių duomenų klasterizavimas gali būti pasiektas šiais būdais:

1. Parenkant klasterių skaičių automatiškai. Viena didžiausių problemų yra pradinių klasterių skaičiaus pasirinkimas. Netinkamas vartotojo parinktas klasterių skaičius lemia nekorektiškus rezultatus.

2. Atliekant klasterizavimo rezultatų validavimą. Tai leidžia suprasti ar klasterizavimo procesas yra korektiškas.

3. Klasterizavimas turi būti paremtas atstumo funkcijomis. Atstumo funkcijomis paremti algoritmai tokie, kaip k-vidurkių, k-modų yra $O(N^2)$ kompleksiškumo.

4. Neatliekant duomenų tipo transformacijos. Tokiu būdu duomenys bus nepakeisti ir nebus prarandama pradinė informacija.

5. Parenkant pradinius klasterių centrus neatsitiktinai. Tokiu būdu išvengiama problemos, kad pradiniai centrai bus arti vienas bei algoritmo konvergavimas bus greitesnis.

3 TYRIMŲ METODAI

Pagal 2.9 skyriuje esančias literatūros analizės išvadas, dauguma straipsnyje *Determining the number of clusters using information entropy for mixed data* [23] aprašytų algoritmo idėjų tinka mūsų iškeltiems tikslams. Šiame skyriuje pristatomos minėto algoritmo naudojamos atskiros dalys bei aprašomos atliktos modifikacijos, kurios išdėstytos skyriuose 3.4 Klasterių skaičiaus nustatymas, 3.5 Pradinių centrų parinkimas, 3.6 Multidimensinis masteliavimas.

3.1 BLOGIAUSIO KLASTERIO NUSTATYMAS

Tegul A^r - atributai žymintys tolydžiųjų duomenų poaibį, A^c - atributai žymintys tolydžiųjų duomenų poaibį, o A - atributai žymintys mišriųjų duomenų poaibį. Klasterių aibę žymėsime $C^k = \{C_1, C_2, ..., C_k\}$, čia k yra klasterių skaičius.

3.1.1 TOLYDŽIŲJŲ DUOMENŲ POAIBIO BLOGIAUSIO KLASTERIO NUSTATYMAS

Tolydžiųjų duomenų poaibiui apibrėšime Renyi entropiją, kuri buvo pristatyta 1960 m. A. Renyi straipsnyje [26]. Ši entropija naudojama klasterizavimo rezultatų validavimui. Pagal apibrėžimą informacijos entropija naudojama įvertinant diskrečiuosius duomenis, tačiau Renyi entropijos privalumas tas, kad ją galime pritaikyti tolydžiųjų duomenų poaibiui. Bendru atveju Renyi entropija apibrėžiama formule 2

$$H_R(x) = \frac{1}{1-\alpha} log \int f^{\alpha}(x) dx, \alpha > 0, \alpha \neq 1. \tag{2}$$

Parinkus $\alpha = 2$ formulė 2 gali būti perrašyta kaip 3

$$H_R(x) = -log \int f^2(x) dx. \tag{3}$$

Norint apskaičiuoti 3 formulę, reikia įvertinti tikimybinę tankio funkciją. Tegul $\{x_1, x_2, ..., x_N\}$ yra tolydūs, atsitiktiniai, nepriklausomi taškai, pasiskirstę pagal vienodą skirstinio funkciją, čia N - taškų kiekis. Įvertinsime taškų $\{x_1, x_2, ..., x_N\}$ tankio funkciją. Jos apskaičiavimui naudojamas branduolio Gauso įvertis 4 formulė, čia σ^2 - branduolio plotis

$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} W_{\sigma^2}(x, x_i), \tag{4}$$

$$W_{\sigma^2}(x, x_i) = \frac{1}{(2\pi)^{d/2}\sigma^d} exp\left(-\frac{(x-x_i)^T(x-x_i)}{2\sigma^2}\right). \tag{5}$$

Kadangi formulėje 3 tankio funkcija yra pakelta kvadratu tai:

$$(\hat{f}(x))^2 = \int \frac{1}{N} \sum_{i=1}^{N} W_{\sigma^2}(x, x_j) \cdot \frac{1}{N} \sum_{i=1}^{N} W_{\sigma^2}(x, x_i).$$

Pagal sąsūkos teoremą (angl. *convolution theorem*) gauname

$$\int W_{\sigma^2}(x, x_i) W_{\sigma^2}(x, x_j) = W_{2\sigma^2}(x_i, x_j)$$

Remiantis Renyi entropija, apibrėžiame entropiją klasterio viduje WE_N, bei entropiją tarp klasterių BE_N.

Apibrėžimas 1 *Tegul NDT - tolydžių duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$ bet kuriam klasteriui $C_{k'} \in C^k$ entropija jo viduje gali būti apibrėžta*

$$WE\_N(C_{k'}) = -log \frac{1}{N_{k'}^2} \sum_{x \in C_{k'}} \sum_{y \in C_{k'}} W_{2\sigma^2}(x, y). \tag{6}$$

Apibrėžimas 2 *Tegul NDT - tolydžių duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$ su bet kuriais $C_i, C_j \in C^k, (i \neq j)$ koeficientas BE_N užrašomas*

$$BE\_N(C_i, C_j) = -log \frac{1}{N_i N_j} \sum_{x \in C_i} \sum_{y \in C_j} W_{\sigma^2}(x, x_i). \tag{7}$$

$N_i = |C_i|$ ir $N_j = |C_j|$ įrašų skaičius klasteriuose. Indeksas BE_N yra didelis, jeigu C_i ir C_j klasteriai yra atsiskyrę. Toliau apibrėšime matą, kuriuo naudojantis nustatysime kiekvieno klasterio įtaką bendram rezultatui.

Apibrėžimas 3 *Tegul NDT - tolydžių duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$, kur $k > 2$. Bet kuriam klasteriui galime apskaičiuoti koeficientą $SBAE\_N$*

$$SBEA\_N(C_{k'}) = \sum_{C_i \in C^k, i \neq k'} \sum_{C_j \in C^k, j \neq k', j \neq i} BE\_N(C_i, C_j). \tag{8}$$

Pagal 3 apibrėžimą sudaromos i, j koeficientų kombinacijos tokios kaip:
$k = 3$ ir $k' = 1$, tai gauname $[[2,3]]$,
$k = 4$ ir $k' = 1$, tai gauname $[[2,3],[3,4],[2,4]]$.

Koeficientas $SBAE\_N$ parinkus klasterį k' yra didelis, jei minėtas klasteris yra blogai atsiskyręs nuo likusių, tai reiškia k' klasteris yra blogiausias tarp likusių.

3.1.2 DISKREČIŲJŲ DUOMENŲ POAIBIO BLOGIAUSIO KLASTERIO NUSTATYMAS

Toliau apibrėžiamas koeficientas, leidžiantis įvertinti blogiausią klasterį diskrečių duomenų poaibyje, naudojant kitokio tipo entropiją. Pirmiausia aptariami papildomi matematiniai įrankiai, reikalingi tolimesniam dėstymui.

Turime duomenų lentelę $I = (U, A)$, kur U - įrašų aibė lentelėje, A - atributų aibė. Su kiekvienu $a \in A$ teisingas vaizdavimas $a : U \longrightarrow V_a$, kuris aprašo atributo a įgyjamas V_a reikšmes. Parinkus $x \in U$, atributo a įgyjama reikšmė $a(x) \in V_a$.

Apibrėžimas 4 *Tegul CDT - diskrečiųjų duomenų poaibis, o $P \subseteq A^c$ atributų rinkinys. Ekvivalentumo sąryšis (angl.* equivalence relation*) $IND(P)$ apibrėžiamas*

$$IND(P) = \left[(x, y) \in U^2 | \forall \in P, a(x) = a(y)\right] \tag{9}$$

Naudodami ekvivalentumo sąryšį aibę U galime suskirstyti į ekvivalentumo klasę $U/IND(P)$ (angl. *equivalence class*) pagal atributų rinkinį P pagal šaltinį[27].

3 lentelė: Ekvivalentumo klasių pavyzdys

	a1	a2	a3
x1	a	a	c
x2	a	r	t
x3	b	b	b
x4	q	k	c
x5	a	c	t

Pagal lentelėje 3 pateiktus duomenis $A = \{a1, a2, a3\}$, $U = \{x_1, x_2.x_3, x_4, x_5\}$. Atributo $a1$ ekvivalentumo klasė $U/IND(a_1)$ gali būti užrašyta kaip $U/IND(a_1) = \{(x_1, x_2, x_5), (x_3), (x_4)\}$. Atributų $a1, a2$ ekvivalentumo klasė $U/IND(a_1, a2)$ gali būti užrašyta kaip

$$U/IND(a_1, a2) = \{(x_1, x_3), (x_2), (x_4), (x_5)\}.$$

Apibrėžimas 5 *Tegul CDT - diskrečiųjų duomenų poaibis, o $P \subseteq A^c$ atributų rinkinys ir $U/IND(P) = \{X_1, X_2, ..., X_m\}$ ekvivalentumo klasė. Papildinio entropija (angl.* complement entropy*) yra*

$$E(P) = \sum_{i=1}^{m} \frac{|X_i|}{|U|}\left(1 - \frac{|X_i|}{|U|}\right). \tag{10}$$

Remiantis papildinio entropija, apibrėžiame entropiją klasterio viduje WE_C bei entropiją tarp klasterių BE_C.

Apibrėžimas 6 *Tegul CDT - diskrečiųjų duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$ bet kuriam klasteriui $C_{k'} \in C^k$ entropija jo viduje gali būti apibrėžta*

$$WE\_C(C_{k'}) = \sum_{a \in A^c} \sum_{X \in C_{k'}/IND(a)} \frac{|X|}{|C_{k'}|}\left(1 - \frac{|X|}{|C_{k'}|}\right). \tag{11}$$

Toliau pateikiama kaip 11 formulė gali būti suprastinta. Pažymime $Y_a = C_{k'}/IND(a)$ ir perrašome 11 formulę

$$WE\_C(C_{k'}) = \sum_{a \in A^c} \sum_{X \in Y_a} \frac{|X|}{|C_{k'}|}\left(1 - \frac{|X|}{|C_{k'}|}\right) =$$

$$= \frac{1}{|C_{k'}|^2} \sum_{x \in C_{k'}} \sum_{y \in C_{k'}} d_{A^c}(x, y).$$

$d_{A^c}(x, y)$ yra Hamingo atstumas

$$d_{A^c}(x, y) = \sum_{a \in A^c} d_a(x, y), \tag{12}$$

čia

$$d_a(x, y) = \begin{cases} 0, & \text{a(x) = a(y)} \\ 1, & \text{a(x) != a(y)} \end{cases}$$

Remiantis papildinio entropija, apibrėžiame entropiją klasterio viduje WE_N bei entropiją tarp klasterių BE_N.

Apibrėžimas 7 *Tegul CDT - diskrečiųjų duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$ su bet kuriais $C_i, C_j \in C^k, (i \neq j)$ koeficientas BE_C užrašomas*

$$BE\_C(C_i, C_j) = \frac{1}{N_i N_j} \sum_{x \in C_{k'}} \sum_{y \in C_{k'}} d_{A^c}(x, y). \tag{13}$$

$N_i = |C_i|$ ir $N_j = |C_j|$ įrašų skaičius klasteriuose. Indeksas BE_C yra didelis, jeigu C_i ir C_j klasteriai yra atsiskyrę. Toliau apibrėšime matą, kuriuo nustatysime kiekvieno klasterio įtaką bendram rezultatui.

Apibrėžimas 8 *Tegul CDT - diskrečiųjų duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$, kur $k > 2$. Bet kuriam klasteriui galime apskaičiuoti koeficientą $SBAE\_C$*

$$SBEA\_C(C_{k'}) = \sum_{C_i \in C^k, i \neq k'} \sum_{C_j \in C^k, j \neq k', j \neq i} BE\_C(C_i, C_j). \tag{14}$$

3.1.3 MIŠRIŲJŲ DUOMENŲ POAIBIO BLOGIAUSIO KLASTERIO NUSTATYMAS

Remiantis skyriais 3.1.1 Tolydžiųjų duomenų poaibio blogiausio klasterio nustatymas ir 3.1.2 Diskrečiųjų duomenų poaibio blogiausio klasterio nustatymas apibrėšime $SBAE\_M$ koeficientą, kuriuo naudojantis galima nustatyti blogiausią mišriųjų duomenų rinkinio klasterį. Šis koeficientas yra kombinacija $SBAE\_N$ ir $SBAE\_C$ metrikų.

Apibrėžimas 9 *Tegul MDT - mišriųjų duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$ bet kuriam $C_{k'} \in C^k$ koeficientas SBE_M yra apibrėžtas kaip*

$$SBAE\_M(C_{k'}) = \frac{|A^r|}{|A|} \frac{SBAE\_N(C_{k'})}{\sum_{i=1}^{k} SBAE\_N(C_i)} + \frac{|A^c|}{|A|} \frac{SBAE\_C(C_{k'})}{\sum_{i=1}^{k} SBAE\_C(C_i)}. \tag{15}$$

Klasteris $C_{k'}$ yra blogiausias, jeigu įgyjama $SBAE\_M$ reikšmė yra didžiausia.

3.1.4 BLOGIAUSIO KLASTERIO NUSTATYMAS ILIUSTRACINIS PAVYZDYS

Tarkime turime mišriųjų duomenų 4 lentelę, kur $U = \{1, 2, 3, ..., 9\}$, $A^r = \{a3, a4\}$, $A^c = \{a1, a2\}$, $A = \{a1, a2, a3, a4\}$. Taip pat tariame, kad duomenų lentelė yra suskaidyta į 3 klasterius $C^3 = \{c1, c2, c3\}$, čia $c1 = \{1, 2, 3\}$, $c2 = \{4, 5, 6\}$, $c3 = \{7, 8, 9\}$.

4 lentelė: Mišrių duomenų pavyzdys naudotas iliustraciniams skaičiavimams

įrašas	a1	a2	a3	a4	klasteris
1	a	f	0.5	0.6	c1
2	b	f	0.45	0.48	c1
3	c	e	0.55	0.49	c1
4	b	e	0.30	0.35	c2
5	b	f	0.27	0.47	c2
6	c	e	0.35	0.48	c2
7	a	f	0.52	0.32	c3
8	a	d	0.43	0.20	c3
9	c	d	0.55	0.24	c3

Toliau pateikiamas detalus koeficiento $SBAE\_M$ apskaičiavimas kiekvienam klasteriui. Šitame pavyzdyje pasirinkta $\sigma = 0.6$ testavimo būdu. Toliau pateikiamas detalus pirmo klasterio įvertinimas.

$$SBAE\_M(c1) = \frac{2}{4}\frac{SBAE\_N(c1)}{SBAE\_N(c1)+SBAE\_N(c2)+SBAE\_N(c3)}+$$

$$+\frac{2}{4}\frac{SBAE\_C(c1)}{SBAE\_C(c1)+SBAE\_C(c2)+SBAE\_C(c3)}.$$

Pagal 8 ir 14 gaunama, kad

$$SBAE\_N(c1) \approx 0.926.$$

$$SBAE\_N(c2) \approx 0.929.$$

$$SBAE\_N(c3) \approx 0.891.$$

$$SBAE\_C(c1) = \frac{16}{9}.$$

$$SBAE\_C(c2) = \frac{13}{9}.$$

$$SBAE\_C(c3) = \frac{11}{9}.$$

Žemiau pateikiamas galutinis koeficiento $SBAE\_M(c1)$ įvertinimas

$$SBAE\_M(c_1) = \frac{1}{2}\frac{0.926}{0.926+0.929+0.891}+\frac{1}{2}\frac{16/9}{16/9+13/9+11/9} \approx 0.369.$$

Taip pat $SBAE\_M$ įverčio reikšmės be detalių skaičiavimų pateiktos kitiems klasteriams.

$$SBAE\_M(c2) = 0.332.$$

$$SBAE\_M(c3) = 0.299.$$

Pagal gautus rezultatus sužinome, kad $SBAE\_M(c1) > SBAE\_M(c2) > SBAE\_M(c3)$ - didžiausia indekso $SBAE\_M$ reikšmė įgyjama su pirmuoju klasteriu. Blogiausias klasteris yra pirmasis ir jo taškus reikia priskirti likusiems klasteriams pagal atstumo funkciją.

3.2 K-PROTOTIPŲ ALGORITMAS

Šiame paragrafe pateikta naudojama atstumo funkcija ir k-prototipų algoritmas. Paprasčiausia atstumo funkcija galima traktuoti k-prototipų, kuri susideda iš Euklido ir Hamingo atstumų dėmenų.

$$D(x, y) = D_{A^r}(x, y) + \gamma D_{A^c}(x, y). \tag{16}$$

Čia duomenų įrašai $x, y \in U$, o D_{A^r} - Euklido atstumas, D_{A^c} - Hamingo atstumas. Šie atstumai apskaičiuojami su atitinkamais poaibiais. Pagrindinė k-prototipų atstumo problema yra γ svorinio koeficiento įvertis. Straipsnyje [23] įvedamos modifikuotos k-prototipų atstumų funkcijos

$$D(x, y) = \frac{|A^r|}{|A|} D_{A^r}(x, y) + \frac{|A^c|}{|A|} D_{A^c}(x, y). \tag{17}$$

Pagal 17 formulę, svoriai priklauso nuo atributų kiekio mišriųjų duomenų lentelėje. Žemiau pateikta dar viena alternatyvi formulė, kurioje įtraukiama žinoma informacija apie klasterių centrus.

Apibrėžimas 10 *Tegul MDT - mišriųjų duomenų poaibis suskaidytas į $k > 2$ klasterių $C^k = \{C_1, C_2, ..., C_k\}$, centrai (prototipai) žymimi $Z^k = \{z_1, ..., z_k\}$. Atstumas tarp įrašo $x \in U$ ir centro $z \in Z^k$ apskaičiuojamas pagal 18 formulę.*

$$D(x, z) = \frac{|A^r|}{|A|} \frac{D_{A^r}(x, z)}{\sum_{i=1}^{k} D_{A^r}(x, z_i)} + \frac{|A^c|}{|A|} \frac{D_{A^c}(x, z)}{\sum_{i=1}^{k} D_{A^c}(x, z_i)}. \tag{18}$$

Žemiau pateikti k-prototipų algoritmo pagrindiniai žingsniai.

Algorithm 1 k-prototipų algoritmas

procedure K-PROTOTIPŲ ALGORITMAS
  Parenkama K pradinių centrų
  while Centrų koordinatės nesikeičia **do**
    Suformuojami K klasterių priskiriant taškus artimiausiems centrams
    Perskaičiuojami centrai
  end while
end procedure

Toliau pateikiamas pavyzdys, kuriame naudojama atstumo funkcija 18 formulė. Pasinaudojant 4 lent. duomenimis pateikiamos gautos atstumo reikšmės tarp kiekvieno įrašo ir centrų, kur centrai yra $Z^k = 1, 4, 7$ įrašai.

5 lentelė: Kiekvieno įrašo atstumai iki centrų

	1	2	3	4	5	6	7	8	9
z_1	0.000	0.264	0.258	0.588	0.372	0.367	0.308	0.511	0.475
z_4	0.783	0.394	0.425	0.000	0.212	0.188	0.692	0.329	0.341
z_7	0.217	0.342	0.318	0.412	0.417	0.446	0.000	0.160	0.184

Naudojantis atstumų 5 lent. surandame artimiausius taškus parinktiems taškams. Matome, kad antrasis taškas priklauso centrui z_1, o šeštasis z_4. Atitinkamai užrašome susidariusius klasterius $c_1 = \{1,2,3\}$, $c_2 = \{4,5,6\}$, $c_3 = \{7,8,9\}$. Toliau galima perskaičiuoti naujus centrus, kurie yra $z_1 = \{a, f, 0.5, 0.523\}$, $z_2 = \{b, e, 0.3067, 0.433\}$, $z_3 = \{a, d, 0, 5, 0.2533\}$. Toliau procesas tęsiamas tol, kol nusistovi centrai.

3.3 KLASTERIŲ CENTRŲ ATNAUJINIMAS

Suradus blogiausią klasterį c_x pagal 15 formulę mes jį panaikiname ir klasterio c_x taškus priskiriame likusiems klasteriams, naudojantis sudaryta atstumų matrica. Pratęsime pavyzdį pateiktą skyriuje 3.2, kuriuo naudojantis perskaičiuosime klasterių centrus. Tarkime, klasteris $c_1 = \{1,2,3\}$ yra blogiausias, tada taškai $1,2,3$ turi būti paskirstomi klasteriams c_2, c_3 naudojantis 5 lent. Gaunume naujus klasterius $c_2 = \{4,5,6,\}$, $c_3 = \{7,8,9,1,2,3\}$. Toliau galima perskaičiuoti naujus centrus, kurie yra $z_2 = \{b, e, 0.3067, 0.433\}$, $z_3 = \{a, f, 0, 5, 0.388\}$.

3.4 KLASTERIŲ SKAIČIAUS NUSTATYMAS

Nagrinėjamame straipsnyje [23] buvo pateiktas metodas rezultatų validavimui, kuriuo nustatomas klasterių skaičius. Nustatant klasterių skaičių buvo konstruojama kategorijų naudingumo funkcija mišriems duomenims CUM

$$CUM(C^k) = \frac{|A^r|}{|A|} CUN(C^k) + \frac{|A^c|}{|A|} CUC(C^k),$$

kuri susideda iš kategorijų naudingumo funkcijos diskretiems duomenims CUC

$$CUC(C^k) = \frac{1}{k} \sum_{a \in A^c} \sum_{X \in U/IND(|a|)} \sum_{i=1}^{k} \left(\frac{|X \cap C_i|^2}{|C_i|^2} - \frac{|X|^2}{|U|^2} \right)$$

ir kategorijų naudingumo funkcijos tolydiems duomenims CUN

$$CUN(C^k) = \frac{1}{k} \sum_{l=1}^{|A^r|} \left(\sum_{x \in U} (x_l - m_l)/|U| - \sum_{j=1}^{k} \frac{|C_j|}{|U|} \sum_{x \in C_j} (x_l - m_{jl})^2/|C_j| \right).$$

Atlikus CUM koeficientą kompiuterinę realizaciją pastebima, kad dažnai tikrasis klasterių skaičius yra nustatomas neteisingai. Taip pat kilo klausimų dėl metodo korektiškumo, nes dvi dedamosios CUN ir CUC yra neapibrėžtos intervalo ribose. Jeigu skaičiuojant CUN koeficientą būtų padidinamos visos tolydaus poaibio vertės, tada CUN koeficientas irgi padidėtų. To pasekoje, skaičiuojant CUM koeficientą, CUN vertė turėtų didesnį svorį prieš CUC, nors klasterių skaičius nepasikeistų. Dėl šios priežasties buvo nuspręsta ieškoti alternatyvių būdų nustatyti klasterių skaičiui.

Pagal anksčiau atliktą literatūros analizę nuspręsta taikyti du populiarius klasterizavimo rezultatų validavimo indeksus - **Duno indeksas** (angl. *Dunn index*) ir **Davies-Bouldin(DB)**. Šie indeksai naudojami tolydžių duomenų klasterizavime, tačiau galima rasti atitikmenį ir mišriųjų duomenų klasterizavime, nes atstumų matrica, klasterių centrai, atstumo funkcija yra žinomi.

Daugiausiai citatų turintis validavimo indeksas buvo pristatytas 1974 metais J.C. Dunno. Žemiau pateikta Duno indekso formulė:

$$k^* = \arg\max_k \left(DU(k) = \min_{i=1,\dots,k} \left(\min_{j=i+1,\dots,k} \left(\frac{D(C_i, C_j)}{\max_{m=1,\dots,k} diam(C_m)} \right) \right) \right). \quad (19)$$

Pagal 19 formulę ieškome klasterių skaičiaus k su kuriuo k^* reikšmė būtų didžiausia. $D(C_i)$ yra atstumas tarp dviejų klasterių. Jis yra apskaičiuojamas įvertinant atstumus tarp kiekvieno taško iš skirtingų klasterių ir imant mažiausią atstumą. Vardiklyje esantis narys $diam(C_m)$ yra m-tojo klasterio diametras. Jis yra apskaičiuojamas apskaičiuojant atstumus tarp taškų m-tajame klasteryje ir imant didžiausią atstumą. Apibendrinant, mėginama maksimizuoti minimalų atstumą tarp klasterių. Šio indekso pagrindinė problema - jautrumas išskirtims.

Toliau aprašomas dar vienas dažnai naudojamas validavimo indeksas Davies-Bouldin, kuris buvo pristatytas 1979 m.

$$k^* = \arg\min_k \left(DB(k) = \frac{1}{k} \sum_{k}^{i=1} \max_{j=1,\dots,k, i \neq j} \left(\frac{diam(C_i) + diam(C_j)}{d(z_i, z_j)} \right) \right), \quad (20)$$

čia $diam(C_i)$ apibrėžiamas kaip

$$diam(C_i) = \sqrt{\frac{1}{n} \sum_{p \in C_i} d(o, z_i)^2}.$$

Taip pat reikėtų paminėti, kad atstumo funkcija yra parenkama priklausomai nuo algoritmo, o z_i žymi klasterio centrą.

3.5 PRADINIŲ CENTRŲ PARINKIMAS

Atsitiktinis pradinių centrų parinkimas lemia galutinį klasterizavimo rezultatą. Blogai parinkti centrai gali prailginti algoritmo konvergavimą arba sudaryti klasterius, kurie neatitinka tikros struktūros. Problemos iškyla tada, kai pradiniai centrai yra parenkami arti vienas kito. Magistrinio darbo konsultantas Dr. Mindaugas Kavaliauskas siūlo idėją, leidžiančią išvengti minėtos problemos. Detaliai ji yra aprašyta disertacijoje „Daugiamačių Gauso skirstinių statistinė analizė, taikant duomenų projektavimą“ [20], o žemiau pateikti pagrindiniai žingsniai.

1. Apskaičiuojama atstumų matricą D naudojantis 17 formule.
2. Sumuojami atstumai pagal matricos eilutes ir randamas didžiausio elemento indeksas.
3. Randama $t = [\frac{U}{k}]$ artimiausių kaimynų didžiausiam elementui, čia skliausteliai žymi sveikąją dalį, U - imties dydis, k - klasterių skaičius.
4. Iš atstumų matricos panaikinamos eilutės bei stulpeliais, kuriuose yra didžiausias elementas ir jo kaimynai.

Remiantis šia idėja galima paskirstyti taškus multidimensinėje erdvėje nutolusius vienus nuo kito.

3.6 MULTIDIMENSINIS MASTELIAVIMAS

Multidimensinio masteliavimo tikslas yra sumažinti duomenų dimensiją. Kitais žodžiais tariant, turėdami atstumo funkciją galime apskaičiuoti atstumų matricą, kuria naudojantis mišrūs duomenys gali būti atvaizduoti į dvimatę erdvę kaip įmanoma mažiau iškreipiant atstumus multidimensinėje erdvėje [22]. Šis metodas reikalingas dėl to, kad kitu atveju neturėsime galimybės vizualiai įvertinti mišriųjų duomenų klasterizavimo rezultatų. Toliau pristatomi pagrindiniai multidimensinio masteliavimo žingsniai.

1. Apskaičiuojama atstumų matricą D, kur tolydžiųjų duomenų atveju dažniausiai naudojamas Euklido atstumas.

2. Pakeliamas kiekvienas atstumų matricos D elementas kvadratu ir gaunama nauja matrica D^2.

3. Apskaičiuojama centravimo matrica (angl. *centering matrix*) $J = I - \frac{1}{n}\Theta$, čia I - vienetinė matrica, n - atstumų matricos dimensija, Θ - matrica sudaryta tik iš vienetų.

4. Apskaičiuojama matrica $B = -\frac{1}{2}JD^2J$.

5. Apskaičiuojamos matricos B tikrinės reikšmės $\lambda_1, ...\lambda_m$ ir tikriniai vektoriai $e_1, ..., e_m$ ir išrikiuojamos tikrinės reikšmės mažėjimo tvarka.

6. Parenkamos tikrinės reikšmės, kurios yra teigiamos bei joms priklausantys tikriniai vektoriai. Tarkime, kad turime $\lambda_1, ..., \lambda_k$ tikrines reikšmes.

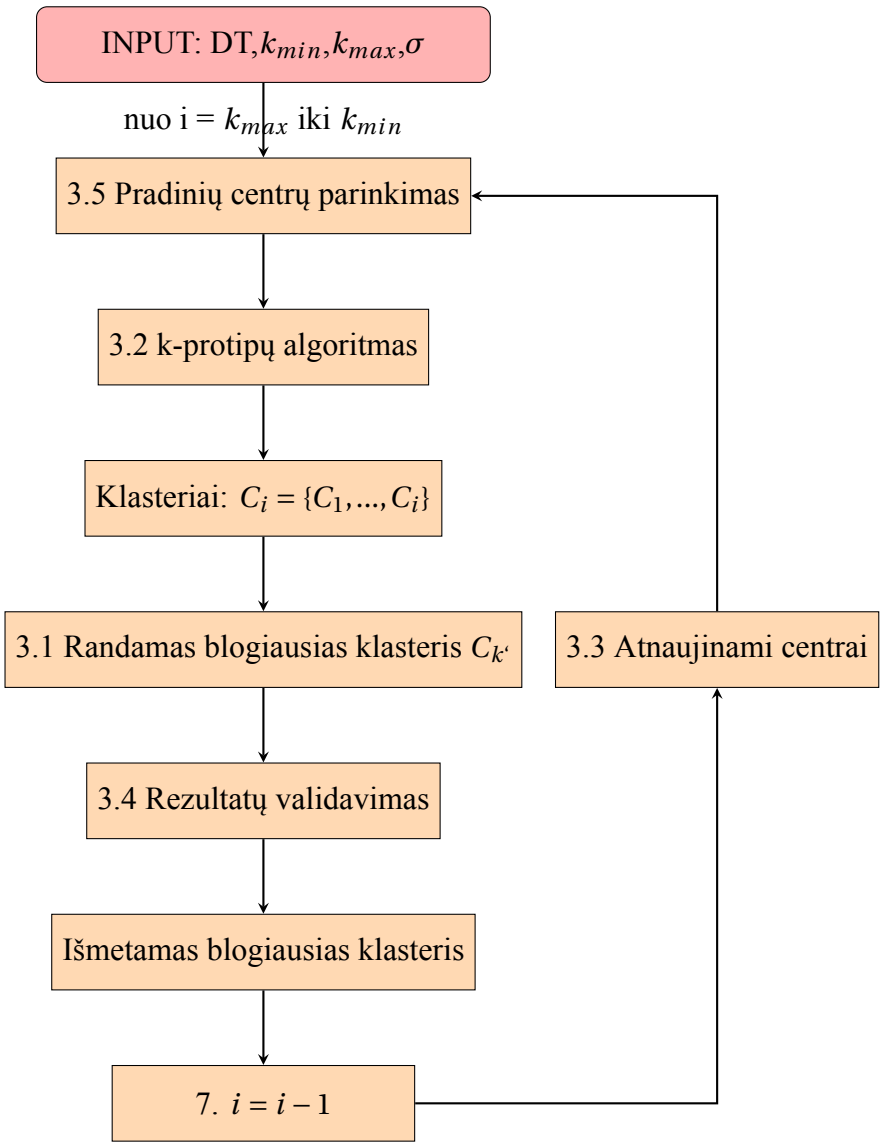
7. Apskaičiuojama matrica $\Lambda_k^{1/2} = \sqrt{diag(\lambda_1, ..., \lambda_k)}$, čia šaknis traukiama paelemenčiui.

8. Randama matrica $X_k = E_k\Lambda_k^{1/2}$, čia matrica E_k sudaryta iš tikrinių vektorių atitinkančių tikrines reikmes $\lambda_1, ..., \lambda_k$.

Paėmus matricos X_k pirmus du stulpelius gauname dvimačius vektorius, kurie yra multidimensinių vektorių atvaizdai. Žinant kuriems klasteriams priklauso dvimačiai vektoriai galime atvaizduoti klasterizavimo rezultatą dvimatėje erdvėje.

3.7 ALGORITMAS

Toliau apjungiamos anksčiau aptartos dalys į bendrą visumą. Taip pat aptariamos problemos, kurios kilo realizavimo metu, bei pateikiami praktiniai patarimai. Algoritmas pateiktas kaip grafinis paveikslas 1 pav., kuriame vizualiai pažymėti žingsniai. Tokio vaizdavimo tikslas padėti skaitytojui įsisavinti anksčiau pateiktą medžiagą ir susidaryti aiškesnį bendrą vaizdą apie pristatytą teorinę medžiagą. Žymėsime DT - duomenų lentelė, kuri gali būti sudaryta iš tolydžiųjų, diskrečiųjų arba mišriųjų duomenų.



1 pav.: Algoritmo vizuali iliustracija

Pirmiausia, prieš pradedant klasterizuoti duomenis reikia pasirinkti tinkamą branduolio plotį σ pagal 4 formulę, nes jis naudojamas tolydžių duomenų blogiausio klasterio įvertinime. Straipsnyje [23] branduolio plotis parenkamas vartotojo testavimo būdu. Vienas iš automatinio parinkimo būdų yra tolydaus duomenų poaibio normavimas, tokiu atveju traktuotume, kad $\sigma = 0.5$, nes normuoti duomenys grupuojasi nedideliuose intervalo rėžiuose. Atliekant praktinius skaičiavimus normavimas atliktas pagal formulę pateiktą žemiau

$$x = (x - \hat{x}) / sd(x). \tag{21}$$

Formulėje 21 x yra tolydžiųjų duomenų vektorius, $\hat{x}$ - vektoriaus x vidurkis, $sd(x)$ - vektoriaus x standartinis nuokrypis. Kitas žingsnis yra minimalaus ir maksimalaus klasterių skaičiaus pasirinkimas. Dažniausiai $k_{min} = 2$, nes norima atlikti iteracijas iki galo, o maksimalus klasterių skaičius priklauso nuo vartotojo. Reikėtų atsižvelgti į tai, jog kuo didesnis k_{max}, tuo skaičiavimai trunka ilgiau. Taip pat reikėtų atkreipti dėmesį į duomenų lentelės dimensiją ir įrašų kiekį. Kad algoritmas funkcionuotų su bet kokio tipo duomenimis daugiau nereikia jokios informacijos.

Algoritmas yra iteracinis - klasterizavimas pradedamas nuo k_{max} klasterių skaičiaus ir mažėjimo eiga tęsiasi iki k_{min}. Pačioje pradžioje parenkamas k_{max} pradiniai centrai, pagal idėją pristatytą 3.5 skyriuje. Žinant pradinius centrus atliekamas k-prototipų algoritmas aprašytas 3.2 skyriuje. Po šio žingsnio žinome nusistovėjusius centrus bei klasterius. Dabar galima pavaizduoti rezultatus naudojantis multidimensinio masteliavimo idėja 3.6 bei atlikti rezultatų validavimą 3.4. Remiantis 3.1 skyriumi identifikuojamas blogiausias klasteris. Šis klasteris yra panaikinamas, o jo taškai priskiriami kitiems klasteriams pagal atstumo funkciją bei perskaičiuojami nauji centrai 3.3. Sumažiname klasterių skaičių ir kartojame procesą su perskaičiuotais klasterių centrais kol pasiekiame k_{min}. Algoritmui pasibaigus stebimi klasterizavimo rezultatų validavimo grafikai ir naudojantis jais nustatoma koks yra rekomenduotinas klasterių skaičius. Žinant rekomenduotiną klasterių skaičių gaunami nusistovėję klasteriai, jų centrai ir analizuojamas multidimensinio masteliavimio grafikas, kuriame atvaizduoti gauti klasteriai.

3.8 KLASTERIZAVIMO REZULTATŲ PALYGINIMAS SU TIKROSIOMIS GRUPĖMIS

Aprašytas mišriųjų klasterizavimo algoritmas 3.7 tolimesniuose skyriuose bus testuojamas su duomenų rinkiniais, kur yra žinomos įrašų grupės. Tokiais atvejais bus atliekamas rastų grupių palyginimas su iš anksto žinomomis. Tarkime pradiniuose duomenyse yra žinoma n grupių $g_1, g_2, ..., g_n$, o atlikus klasterizavimą surandamos kitos grupės $h_1, ..., h_n$. Norint atlikti palyginimą reikia rasti atitikmenis tarp grupių $g_1, g_2, ..., g_n$ ir $h_1, ..., h_n$. Parenkame grupę g_1 ir atrenkame įrašus $i_1, ..., i_m$, čia m yra įrašų skaičius, kuriems ši grupė yra priskirta. Ieškome, kuri grupė $h_x \in h_1, ..., h_n$ iš grupių yra dažniausiai priskirta atrinktiems įrašams $i_1, ..., i_m$. Radus h_x apibrėžiamas sąryšis $g_1 <-> h_x$. Tokiu būdu surandami visi sąryšiai. Po to kiekvienas duomenų įrašas $i_1, .., i_s$, kur s yra duomenų rinkinio eilučių skaičius, turi priskirtas grupes $g_1, g_2, ..., g_n$ ir $h_1, ..., h_n$, kurias sieja žinomas sąryšis $g_x <-> h_y$. Apskaičiuojame kiek yra įrašų, kuriuose yra teisingas sąryšis $g_x <-> h_y$, ir tuo pačiu kiek yra įrašų kuriuose sąryšis yra neteisingas. Tarkime, sutapimų kiekis yra T, o nesutapimų kiekis yra N. Galime apskaičiuoti procentines išraiškas T/s - sutapimų ir N/s - nesutapimų.

4 TYRIMŲ REZULTATAI IR JŲ APTARIMAS

Toliau testuojamas aprašytas klasterizavimo algoritmas 3.7 su sintetiniais ir realiais duomenimis. Tokio testavimo tikslas yra nustatyti, ar algoritmas veikia teisingai ir ar galima jį taikyti realiems duomenims.

4.1 SINTETINIŲ TOLYDŽIŲJŲ IR DISKREČIŲJŲ DUOMENŲ KLASTERIZAVIMAS

Paprasčiausias būdas patikrinti algoritmo veikimą yra klasterizuoti dvimačius tolydžiuosius duomenis, iš kurių galima intuityviai išskirti natūralius klasterius. Toliau bus generuojami Gauso „debesėliai“ ir atliekamas jų klasterizavimas. Tikslas yra identifikuoti norimą klasterių skaičių bei stebėti algoritmo konvergavimą į iš anksto žinomą klasterių kiekį.

6 lentelė: Dvimatės sintetinės duomenų aibės parametrai

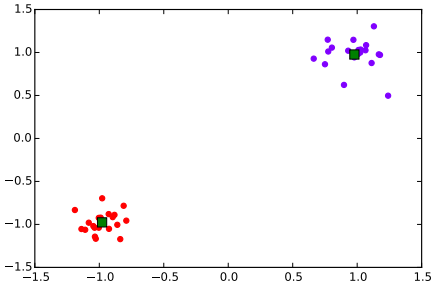
	1	2	3	4
μ	$[0,0]$	$[5,4.5]$	$[5,-2]$	$[-2,-5]$
σ^2	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$

Pagal 6 lent. duomenis sugeneruojami du Gauso „debesėliai“ po 20 atsitiktinių taškų, kurių parametrai yra pateikti pirmame ir antrame stulpeliuose bei atliekamas normalizavimas. Tolydžių duomenų klasterizavimas realizuotas pagal 3.7 skyriuje pristatytą algoritmą. Čia pradiniai parametrai $\sigma = 1$, $k_{min} = 2$, $k_{max} = 7$. Žemiau pateiktas algoritmo iteracinis procesas 2 pav. Pagal jį galime matyti kokie klasteriai gaunami (žymi skirtingos spalvos) iteracijų metu, o žali kvadratai žymi nusistovėjusius klasterių centrus. Pradžioje buvo parinkti 7 neatsitiktiniai taškai žr. 7 lent. iš duomenų aibės, kurie laikomi pradiniais centrais.

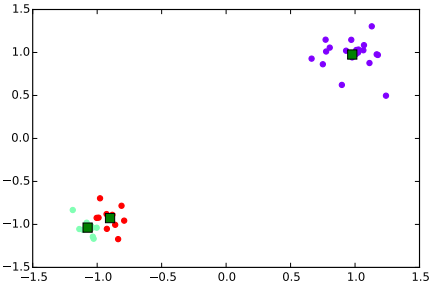
7 lentelė: Pradiniai sintetinės tolydžių duomenų aibės centrai

1.1297263936561046	1.3043071281216432
1.1770464220826744	0.9714043395121035
1.0098910363682077	1.029139874615436
0.7716111982553879	1.1489229633376354
-0.9777508538428562	-0.6962180242822449
-1.189255543122671	-0.8324199483583934
-0.7906008491991113	-0.9561569590952633

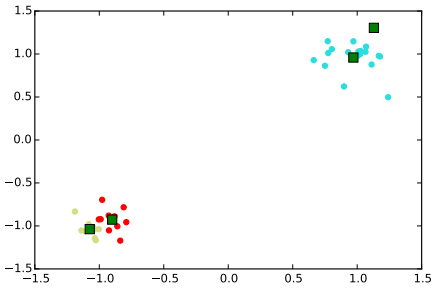
Po to vykdomas anksčiau aprašytas k-prototipų algoritmas ir gaunami nusistovėję klasteriai, kurie yra pateikti 2f pav. Procesas tęsiamas, kol algoritmas pasiekia k_{min} reikšmę taip pat kiekvienos iteracijos metu brėžiami nusistovėjusių klasterių grafikai.



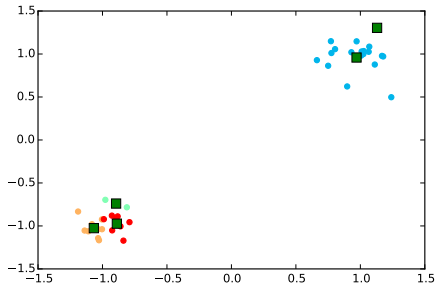
(a) 2 klasteriai



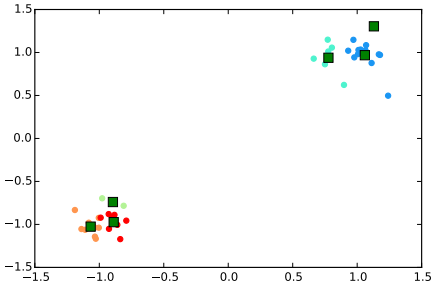
(b) 3 klasteriai



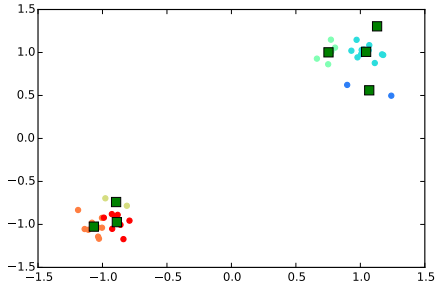
(c) 4 klasteriai



(d) 5 klasteriai



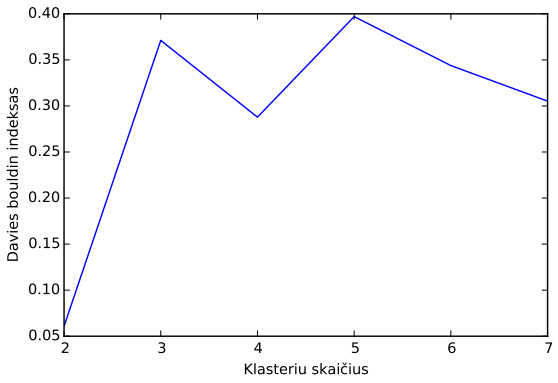
(e) 6 klasteriai



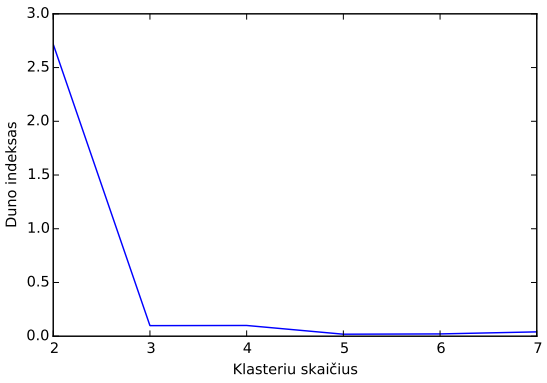
(f) 7 klasteriai

2 pav.: Sintetinės tolydžiųjų duomenų aibės klasterizavimo evoliucija

Pagal žemiau pateiktus klasterizavimo rezultatų validavimo grafikus 3 pav. galime nustatyti, koks yra rekomenduotinas klasterių skaičius. Akivaizdžiai matome, kad DB indekso minimali reikšmė įgyjama ties dviem klasteriais pagal 3a pav., taip pat Duno indekso maksimali reikšmė įgyjama ties dviem klasteriais pagal 3b pav.



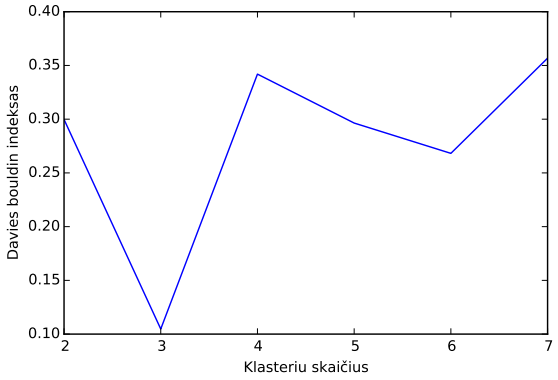
(a) DB validavimo indeksas



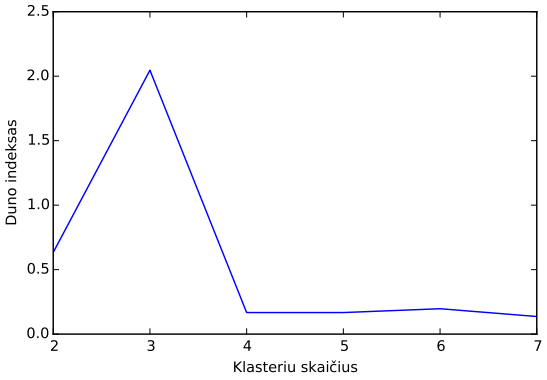
(b) Dunn validavimo indeksas

3 pav.: Testinės tolydžiųjų duomenų aibės (dviejų Gauso „debesėlių“) klasterizavimo rezultatų validavimas

Bandymai pakartoti su trimis sugeneruotomis sintetinėmis duomenų aibėmis remiantis parametrais iš 6 lent. Toliau pateikiami tik validavimo indeksų grafikai 4 pav. bei rasti nusistovėję klasteriai 5 pav.

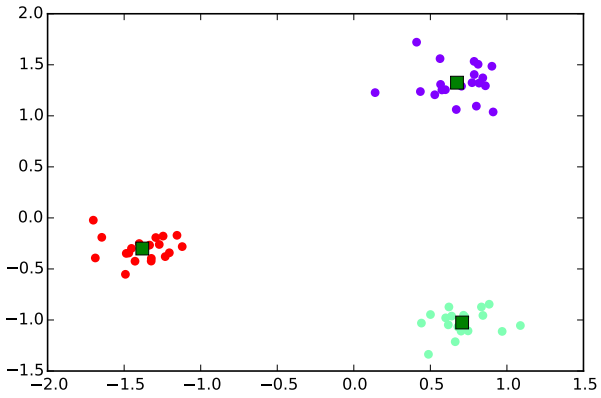


(a) DB validavimo indeksas



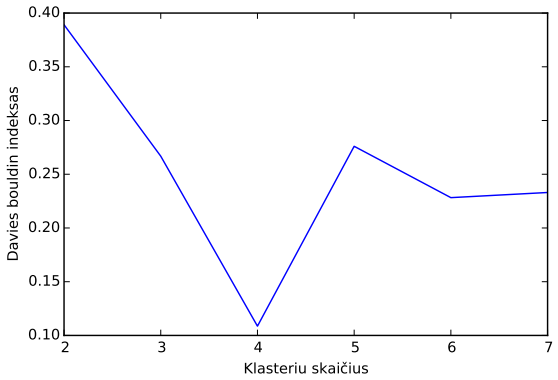
(b) Dunn validavimo indeksas

4 pav.: Testinės tolydžiųjų duomenų aibės (trijų Gauso „debesėlių“) klasterizavimo rezultatų validavimas

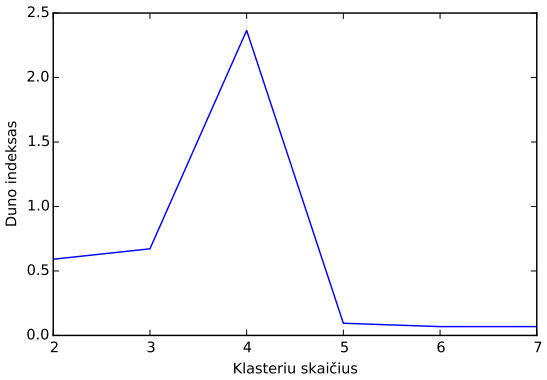


5 pav.: Nusistovėję trys klasteriai, kurie identifikuoja rastus Gauso „debesėlius“

Procesas kartojamas su keturiomis sugeneruotomis sintetinėmis duomenų aibėmis remiantis parametrais iš 6 lent. Žemiau pateikti tik validavimo indeksų grafikai 6 pav. bei rasti nusistovėję klasteriai 7 pav.

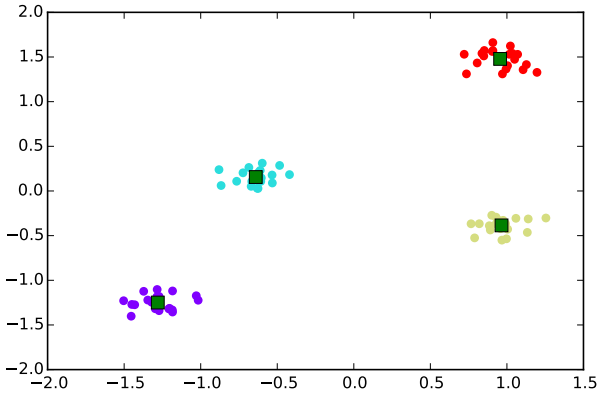


(a) DB validavimo indeksas



(b) Dunn validavimo indeksas

6 pav.: Testinės tolydžiųjų duomenų aibės (keturių Gauso „debesėlių“) klasterizavimo rezultatų validavimas



7 pav.: Nusistovėję keturi klasteriai, kurie identifikuoja rastus Gauso „debesėlius“

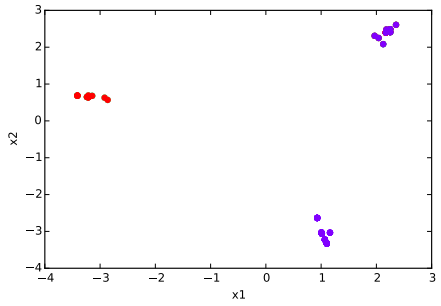
Ištestavus algoritmą su sintetiniais tolydžiais duomenimis gavome, kad visais atvejais tikrasis klasterių skaičius buvo nustatytas teisingai ir pradiniai centrai buvo atitolę vienas nuo kito.

Sintetinė diskrečiųjų duomenų aibė sugeneruota tokiu būdu. Parinkta 3 rinkiniai sudaryti iš 6 elementų $[a,a,a,b,b,b]$, $[c,c,c,d,d,d]$,$[u,u,u,i,i,i,]$, jie pakartojami po 20 kartų ir atsitiktiniuose įrašuose įvedama triukšmo. Įprasti elementai a,b,c,d,u,i pakeičiami kitais, tokiais kaip $[m,r,rr,t,p,q,f,w,o]$. Galutinė duomenų aibė buvo sudaryta iš 60 įrašų ir 6 atributų, kur egzistuoja akivaizdžios 3 grupės. Tikslas - pritaikyti mišriųjų duomenų klasterizavimo algoritmą ir rasti žinomas grupes. Pradiniai algoritmo parametrai yra $k_{min} = 2$ ir $k_{max} = 7$. Lentelėje 8 pateikti pradiniai centrai, kurie buvo parinkti neatsitiktinai.

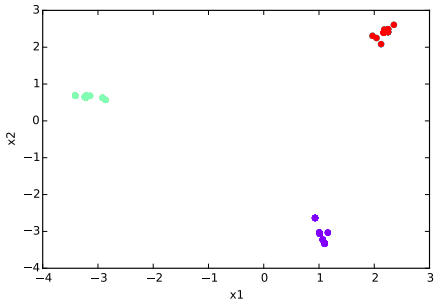
8 lentelė: Pradiniai sintetinės diskrečiųjų duomenų aibės centrai

1	u	e	u	i	i	i
2	o	c	c	d	d	d
3	u	u	u	c	i	i
4	rr	u	u	i	i	i
5	u	u	q	i	i	i
6	c	c	c	d	d	d
7	c	c	w	d	d	d

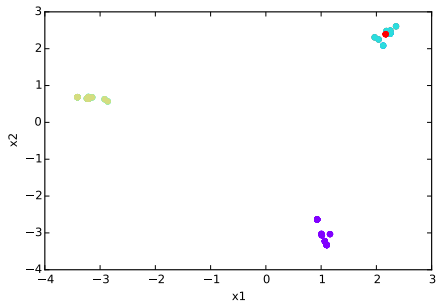
Toliau pateikta visa klasterizavimo evoliucija 8 pav. ir rezultatai pavaizduoti panaudojant multidimensinio masteliavimo įrankį, kur 8f pav. žymi pirmąją iteraciją su nusistovėjusiai klasteriais, o 8a pav. paskutinę iteraciją.



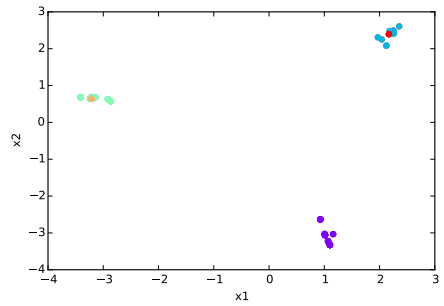
(a) 2 klasteriai



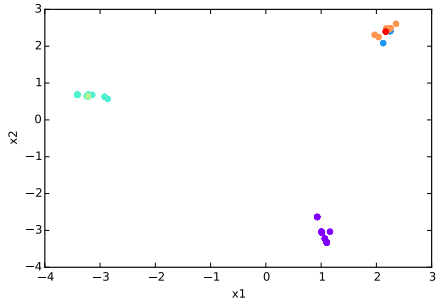
(b) 3 klasteriai



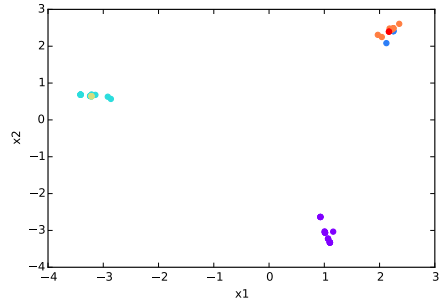
(c) 4 klasteriai



(d) 5 klasteriai

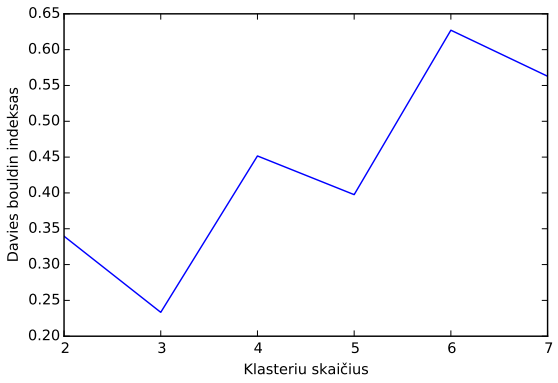


(e) 6 klasteriai

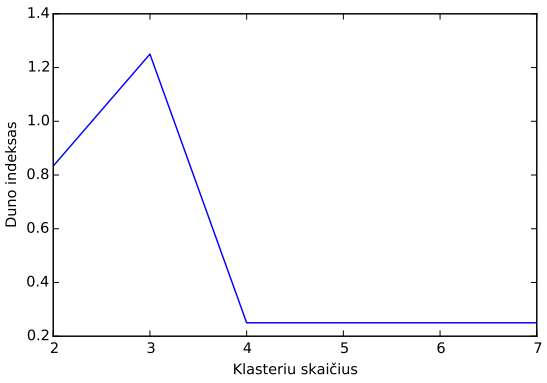


(f) 7 klasteriai

8 pav.: Sintetinės diskrečiųjų duomenų aibės klasterizavimo evoliucija



(a) DB validavimo indeksas



(b) Dunn validavimo indeksas

9 pav.: Sintetinės diskrečiųjų duomenų aibės klasterizavimo rezultatų validavimas

Pagal klasterizavimo rezultatų validavimo grafikus 9 pav. matome, kad rekomenduotinas klasterių skaičius 3 buvo nustatytas teisingai abiejų indeksų. Taip pat nusistovėję klasteriai buvo rasti tokie pat kaip pradiniai rinkiniai pagal 9 lent.

9 lentelė: Sintetinės diskrečiųjų duomenų aibės rastų rekomenduotinų klasterių centrai

1	a	a	a	b	b	b
2	u	u	u	i	i	i
3	c	c	c	d	d	d

Ištestavus algoritmą su sintetiniais diskrečiais duomenimis gavome, kad visais atvejais tikrasis klasterių skaičius buvo nustatytas teisingai.

4.2 REALIŲ DISKREČIŲJŲ DUOMENŲ RINKINIŲ KLASTERIZAVIMAS

Kongreso balsavimo įrašų duomenų aibė (angl. *Congressional Voting Records Data Set*) [28] yra sudaryta iš 16 diskrečiųjų atributų bei 435 stebinių. Duomenyse pateikti kongreso rūmų narių balsavo rezultatai, kur galimos reikšmės yra už, prieš, nežinau. Taip pat žinoma, kad balsavusieji yra respublikonai arba demokratai. Traktuojant, jog informacija apie balsavusių politinę pakraipą yra nežinoma, atliekame mišriųjų duomenų klasterizavimą.

10 lentelė: Kongreso balsavimo duomenų aibės atributai

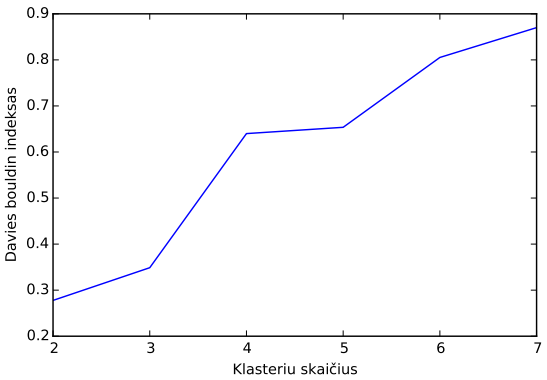
	atributo pavadinimas	atributo tipas	atributo įgyjamos reikšmės
1	handicapped-infants	kategorinis	y,n,?
2	water-project-cost-sharing	kategorinis	y,n,?
3	adoption-of-the-budget-resolution	kategorinis	y,n,?
4	physician-fee-freeze	kategorinis	y,n,?
5	el-salvador-aid	kategorinis	y,n,?
6	religious-groups-in-schools	kategorinis	y,n,?
7	anti-satellite-test-ban	kategorinis	y,n,?
8	aid-to-nicaraguan-contras	kategorinis	y,n,?
9	mx-missile	kategorinis	y,n,?
10	immigration	kategorinis	y,n,?
11	synfuels-corporation-cutback	kategorinis	y,n,?
12	education-spending	kategorinis	y,n,?
13	superfund-right-to-sue	kategorinis	y,n,?
14	crime	kategorinis	y,n,?
15	duty-free-exports	kategorinis	y,n,?
16	export-administration-act-south-africa	kategorinis	y,n,?

Mišrių duomenų algoritmas testuotas su pradiniais parametrais $k_{min} = 2$, $k_{max} = 7$. Šiuo atveju branduolio pločio σ nereikia parinkti, nes visi atributai yra kategoriniai. Pradiniai centrai yra pateikti 11 lent., kurie buvo parinkti neatsitiktinai, o naudojant pradinių centrų parinkimo metodą, kuris buvo aprašytas 3.5 skyriuje.

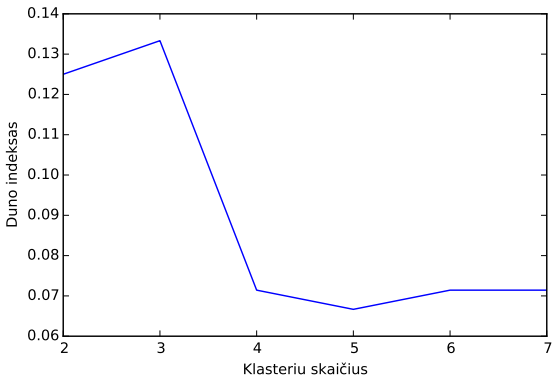
11 lentelė: Kongreso balsavimo duomenų aibės pradiniai klasterių centrai

1	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
2	y	y	y	?	n	n	n	y	n	n	y	?	n	n	y	y
3	y	n	y	n	n	n	y	y	y	?	y	n	n	n	y	?
4	y	n	y	n	?	n	y	y	y	y	n	y	n	?	y	y
5	n	y	y	n	n	?	y	y	y	y	y	n	?	y	y	y
6	y	n	y	y	y	y	y	y	n	n	n	n	n	y	n	?
7	n	n	n	y	y	n	n	n	n	n	n	y	n	y	?	y

Žemiau pateikti klasterizavimo rezultatų validavimo grafikai 10 pav. Šiuo atveju matome, kad pagal DB indeksą geriausias klasterizavimo rezultatas yra su $k = 2$, o pagal Duno $k = 3$. Iš anksto žinant, kad balsavusieji yra dviejų tipų, galima teigti, jog naudojantis DB indeksu rekomenduotinas klasterių skaičius buvo nustatytas teisingai.



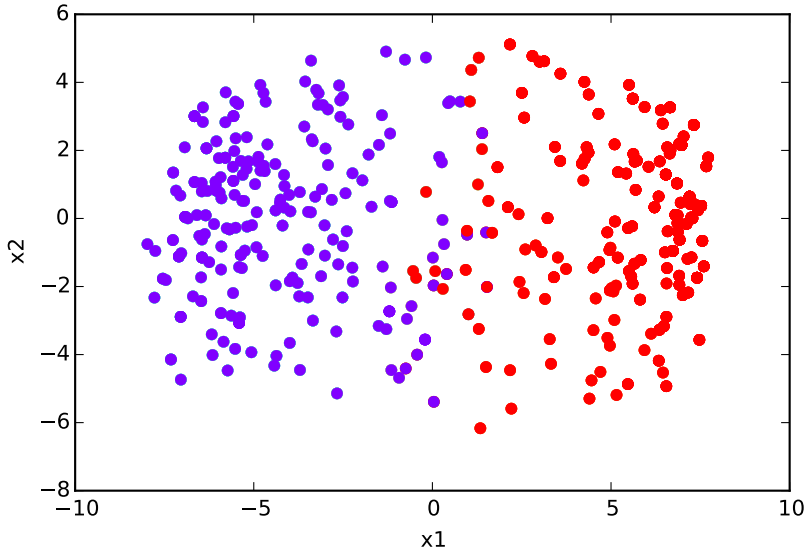
(a) DB validavimo indeksas



(b) Dunn validavimo indeksas

10 pav.: Kongreso balsavimo duomenų aibės klasterizavimo rezultatų validavimas

Žemiau pateiktame 11 pav. matome, pavaizduotus du atsiskyrusius klasterius, kurie turi persidengenčią zoną. Vaizdavimas atliktas naudojant multidimensinio masteliavimo metodą.



11 pav.: Kongreso balsavimo duomenų aibės multidimensinis masteliavimas

Nusistovėjusių klasterių centrai pateikti 12 lentelėje.

12 lentelė: Kongreso balsavimo duomenų aibėje rasti nusistovėję klasterių centrai

1	y	n	y	n	n	n	y	y	y	n	n	n	n	n	y	y
2	n	y	n	y	y	y	n	n	n	y	n	y	y	y	n	y

Kadangi algoritmas buvo testuotas ant duomenų, kur iš anksto yra žinomos įrašų kategorijos, galima atlikti lyginamąją analizę su surastais klasteriais. Realybėje to padaryti būtų neįmanoma, nes klasterizuojami duomenys yra be iš anksto žinomų grupių.

Palyginus iš anksto žinomas grupes su rastomis grupėmis atlikus klasterizavimą gauta, kad 376(86%) tikrųjų grupių sutampa su rastomis, 59(14%) atvejai buvo nustatyti neteisingai pagal metodiką pateiktą 3.8 skyriuje.

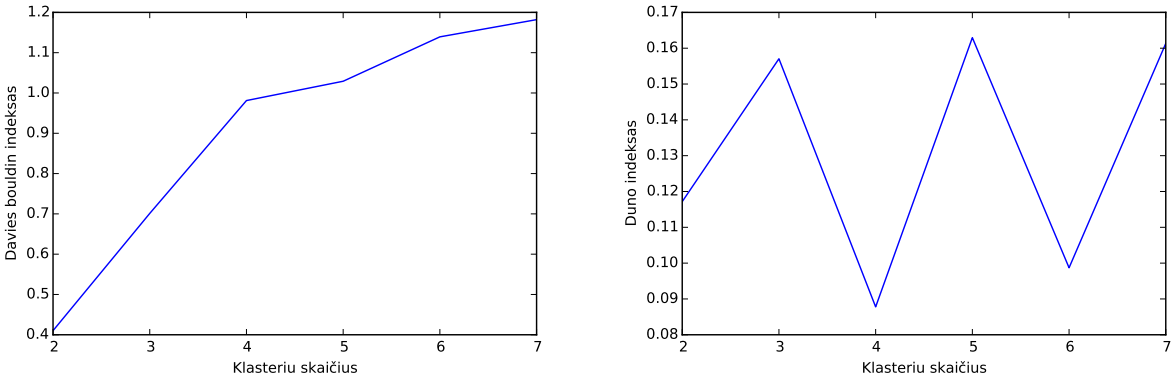
4.3 REALIŲ TOLYDŽIŲJŲ DUOMENŲ RINKINIŲ KLASTERIZAVIMAS

Krūties vėžio duomenų aibė (angl. *Breast Cancer Wisconsin Data Set*) [29] yra sudaryta iš 9 tolydžiųjų atributų bei 699 stebinių. 16 įrašų turi praleistų reikšmių, todėl jos pašalinamos. Duomenyse pateikta Dr. Wolberg ataskaita apie klienčių krūties būklę. Čia taip pat žinoma įrašų kategorija - šiuo atveju kategorijos yra dvi: auglys piktybinis arba nepiktybinis. Traktuosime, kad kategorijos nežinome ir naudodami mišriųjų duomenų algoritmą, rasime klasterius bei nustatysime klasterių skaičių.

13 lentelė: Krūties vėžio duomenų atributai

	atributo pavadinimas	atributo tipas	atributo įgyjamos reikšmės
1	Clump Thickness	tolydus	[1;10]
2	Uniformity of Cell Size	tolydus	[1;10]
3	Uniformity of Cell Shape	tolydus	[1;10]
4	Marginal Adhesion	tolydus	[1;10]
5	Single Epithelial Cell Size	tolydus	[1;10]
6	Bare Nuclei	tolydus	[1;10]
7	Bland Chromatin	tolydus	[1;10]
8	Normal Nucleoli	tolydus	[1;10]
9	Mitoses	tolydus	[1;10]

Pradiniai parametrai $k_{min} = 2$, $k_{max} = 7$. Prieš tai atliktas normalizavimas ir parinkta $\sigma = 1$ reikšmė. Žemiau pateiktas klasterizavimo rezultatų validavimo grafikai 12 pav. Šiuo atveju matome, kad pagal DB indeksą geriausias klasterizavimo rezultatas pasiektas su $k = 2$, o pagal Duno indekso priklausomybę nuo klasterių skaičiaus nematome aiškios tendencijos. Žinodami iš anksto, kad balsavusieji yra dviejų tipų, galime teigti, jog naudojantis DB indeksu rekomenduotinas klasterių skaičius buvo nustatytas teisingai.

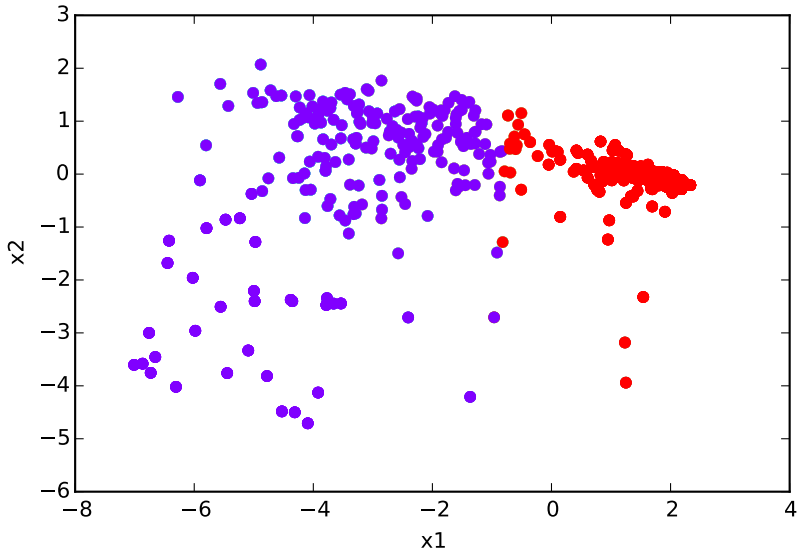


(a) DB validavimo indeksas **(b)** Dunn validavimo indeksas

12 pav.: Krūties vėžio duomenų aibės klasterizavimo rezultatų validavimas

Taip pat 13 pav. pateikti nusistovėjęs rezultatas su dviem klasteriais naudojant multidimensinio klasterizavimo metodą. Pagal grafiką matome du atsiskyrusius klasterius, kur raudona spalva

pažymėta grupė yra mažesnė už kitą.



13 pav.: Krūties vėžio duomenų aibės multidimensinis masteliavimas

Klasterių nusistovėję centrai pateikti lentelėje 14.

14 lentelė: Krūties vėžio duomenų aibėje rasti nusistovėję klasterių centrai

1	0.992	1.210	1.195	1.031	1.019	1.154	1.076	1.059	0.613
2	-0.497	-0.606	-0.5992	-0.5169	-0.5107	-0.5783	-0.539	-0.531	-0.307

Palyginus iš anksto žinomas grupes su rastomis grupėmis atlikus klasterizavimą gauta, kad 652(95%) tikrųjų grupių sutampa su rastomis,31(5%) atvejai buvo nustatyti neteisingai pagal metodiką pateiktą 3.8 skyriuje.

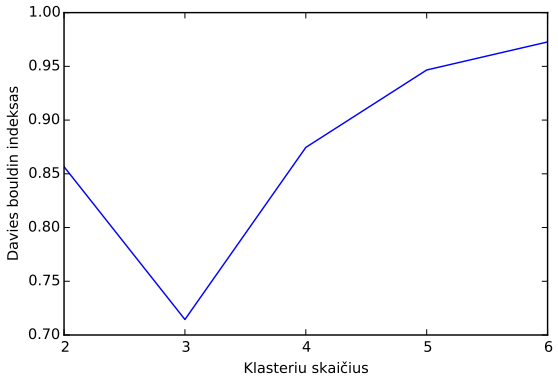
4.4 REALIŲ MIŠRIŲJŲ DUOMENŲ RINKINIŲ KLASTERIZAVIMAS

Mokytojų asistentų vertinimo duomenų aibė (angl. *Teaching Assistant Evaluation Data Set*) [30] yra sudaryta iš 4 kategorinių ir 1 tolydaus atributo, įrašų skaičius yra 151. Duomenyse pateikti mokytojų asistentų įvertinimai Viskonsino-Medisono universitete. Čia taip pat žinoma įrašų kategorija, šiuo atveju kategorijos yra trys: asistentas yra blogas, vidutinis, geras. Traktuosime, kad kategorijos nežinome ir naudodami mišriųjų duomenų algoritmą, rasime klasterius bei nustatysime klasterių skaičių.

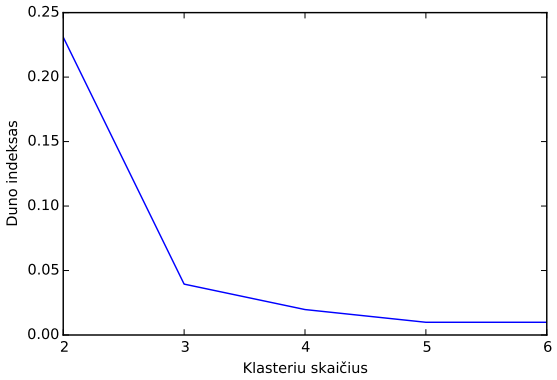
15 lentelė: Mokytojų asistentų vertinimas

	atributo pavadinimas	atributo tipas	atributo įgyjamos reikšmės
1	Is TA English speaker	binarinis	1 arba 0
2	Course instructor	kategorinis	25 categories
3	Course	kategorinis	26 categories
4	Summer or regular semester	binarinis	1 arba 2
5	Class size	tolydus	[3;58]

Pradiniai parametrai $k_{min} = 2$, $k_{max} = 7$. Tolydžių duomenų poaibio normalizavimas neatliktas, o vartotojo parinkta $\sigma = 4$ reikšmė. Žemiau pateikti klasterizavimo rezultatų validavimo grafikai 14 pav. Šiuo atveju matome, kad pagal DB indeksą geriausias klasterizavimo rezultatas pasiektas su $k = 3$, o pagal Duno indeksas rodo, kad geriausias rezultatas yra su $k = 2$. Žinodami iš anksto, kad mokytojų asistentai buvo suskirstyti į tris kategorijas, galime teigti, jog naudojantis DB indeksu rekomenduotinas klasterių skaičius buvo nustatytas teisingai.



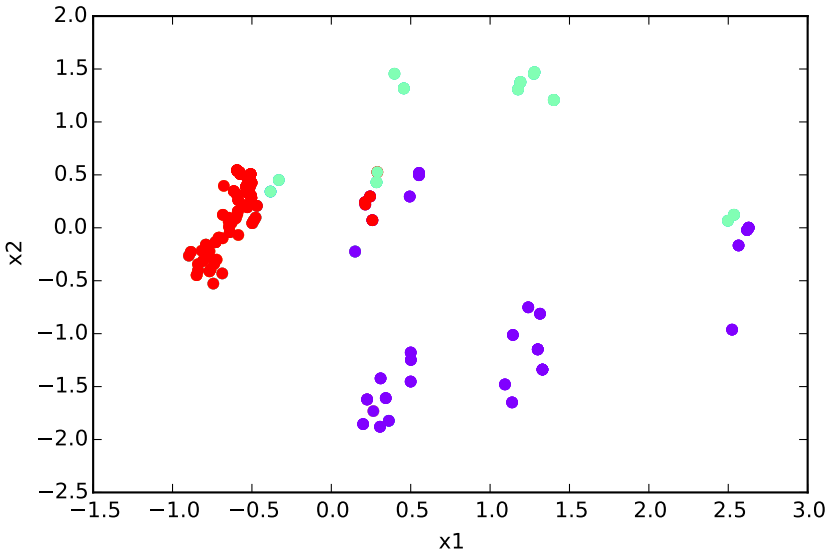
(a) DB validavimo indeksas



(b) Dunn validavimo indeksas

14 pav.: Mokytojų asistentų vertinimo duomenų klasterizavimo rezultatų validavimas

Taip pat 15 pav. pateikta nusistovėjęs rezultatas su trimis klasteriais naudojant multidimensinio klasterizavimo idėją. Pagal grafiką matome, kad taškai erdvėje yra plačiai pasklidę tik raudona spalva pažymėta grupę yra tankiai susikoncentravusi.



15 pav.: Mokytojų asistentų vertinimo duomenų aibės multidimensinis masteliavimas

Nusistovėjusių trijų klasterių centrai yra pateikti lentelėje 16

16 lentelė: Mokytojų asistentų duomenų aibėje rasti nusistovėję klasterių centrai

1	1.0	23.0	3.0	2.0	0.335
2	2.0	13.0	3.0	1.0	-0.825
3	2.0	7.0	2.0	2.0	0.0766

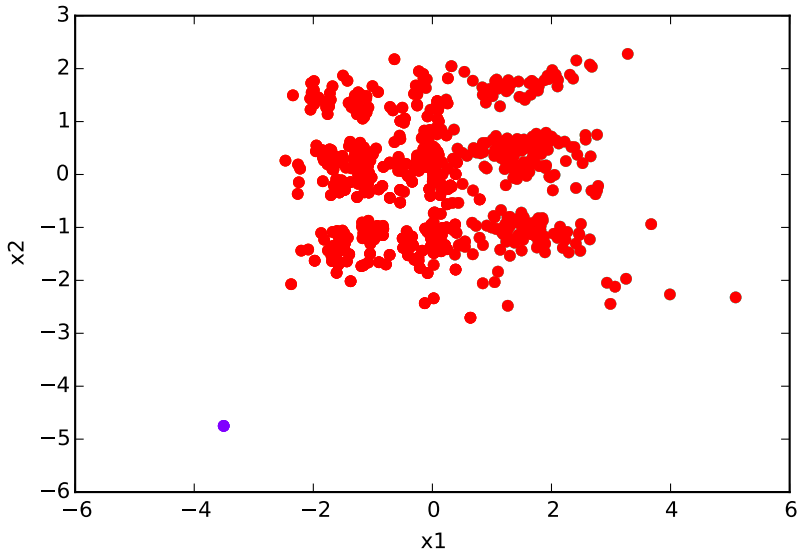
Palyginus iš anksto žinomas grupes su rastomis grupėmis atlikus klasterizavimą gauta, kad 94(62%) tikrųjų grupių sutampa su rastomis,57(38%) atvejai buvo nustatyti neteisingai pagal metodiką pateiktą 3.8 skyriuje.

Toliau pateikiamas antras mišriųjų duomenų klasterizavimo pavyzdys. Turime mišriųjų duomenų rinkinį apie Australijos žmones, kurie teikė prašymą dėl kredito paėmimo (angl. *Australian Credit Approval Data Set*) [32]. Šie duomenys sudaryti iš 8 kategorinių ir 6 tolydaus tipo atributų, įrašų skaičius yra 690. Duomenys yra konfidencialūs, todėl informacija apie kategorinius kintamuosius yra užkoduota skaitinėmis reikšmėmis bei nežinome, kokie yra tikri atributų vardai. Šiuose duomenyse yra žinoma, kuriems žmonėms kreditas buvo duotas, o kuriems ne. Traktuosime, kad informacijos apie kredito davimą nežinome ir naudodami mišriųjų duomenų klasterizavimo algoritmą, rasime klasterius bei nustatysime klasterių skaičių. Toliau pateiksiu informaciją, kokie duomenys yra žinomi. Pradiniai parametrai $k_{min} = 2$, $k_{max} = 5$. Tolydžių duomenų poaibio normalizavimas atliktas, o vartotojo parinkta $\sigma = 2.5$ reikšmė. Žemiau pateikti neatsitiktiniai pradiniai centrai 17 lent.

17 lentelė: Duomenų apie kreditus aibės pradiniai centrai

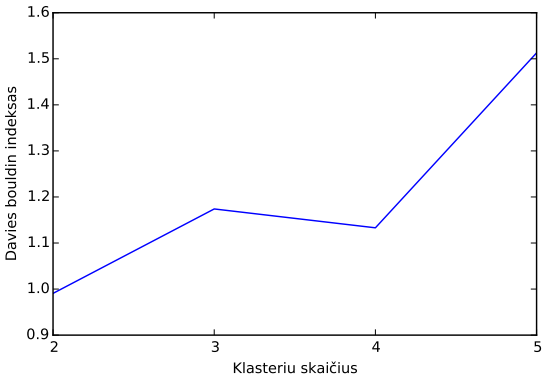
1	1.0	3.0	1.0	7.0	0.0	0.0	1.0	3.0	-1.186	-0.342	-0.286	-0.493	1.544	18.998
2	1.0	2.0	13.0	4.0	1.0	1.0	1.0	2.0	-0.497	-0.345	-0.285	13.284	-0.255	-0.145
3	1.0	2.0	1.0	1.0	1.0	1.0	1.0	2.0	1.941	2.659	9.733	1.768	-0.894	-0.137
4	0.0	1.0	8.0	4.0	0.0	1.0	0.0	2.0	-1.011	-0.136	-0.285	-0.287	10.548	-0.194
5	0.0	1.0	5.0	3.0	1.0	1.0	1.0	2.0	-0.835	-0.345	-0.286	1.768	-1.068	-0.195

Atlikus klasterizavimą pastebėta, kad algoritmas suveikė neteisingai dėl nekorektiškų duomenų. Pateikiama paskutinės iteracijos nusistovėję du klasteriai 16 pav.

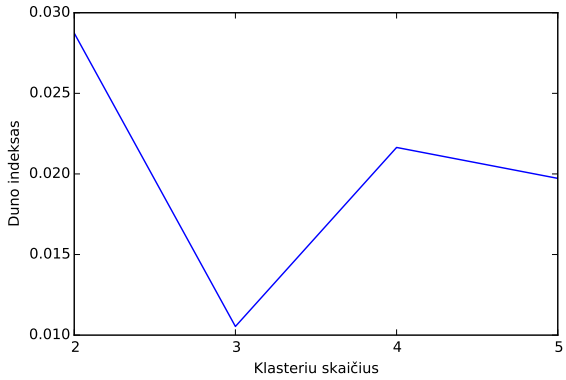


16 pav.: Duomenų apie kreditus aibės multidimensinis masteliavimas

Matome, kad duomenyse yra išskirčių, kurias reikia pašalinti siekiant atlikti korektišką klasterizavimo procesą. Išskirčių pašalinimas atliekamas naudojantis multidimensinio masteliavimo proceso metu sudaryta matrica. Pašalinus išskirtis, su tais pačiais parametrais klasterizavimas atliekamas iš naujo. Pagal klasterizavimo rezultatų validavimo grafikus 17 pav. matome, kad šiuo atveju abu indeksai nustatė tikrąjį klasterių skaičių (2) nustatė teisingai.



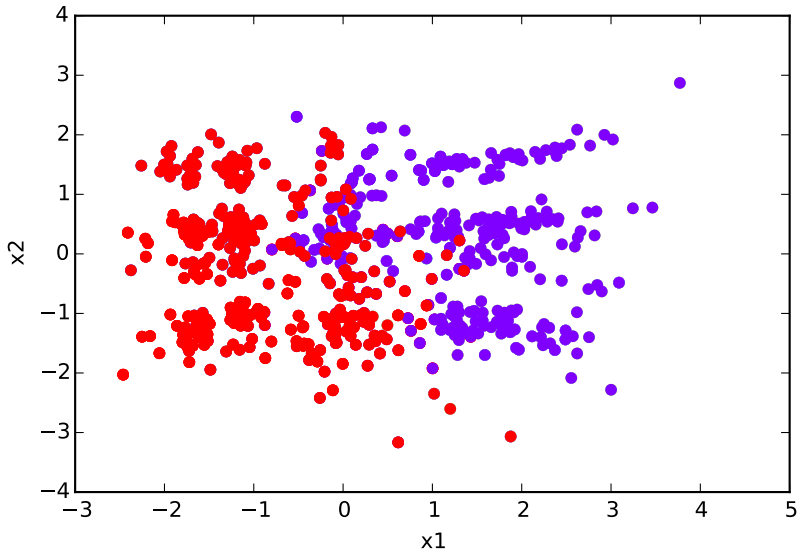
(a) DB validavimo indeksas



(b) Dunn validavimo indeksas

17 pav.: Duomenų apie kreditus aibės klasterizavimo rezultatų validavimas

Pateikiamas multidimensinio masteliavimo rezultatas 18 pav., kur pavaizduotas gautų dviejų klasterių išsidėstymas. Matome, kad, šiuo atveju, pagrindinės išskirtys yra pašalintos, nors, žinoma, būtų galima jas pašalinti atidžiau.



18 pav.: Duomenų apie kreditus aibės multidimensinis masteliavimas

Toliau pateikiama detalesnė klasterizavimo rezultatų analizė 18 lent., nes rezultatai palyginami su KTU mokslininkų Egidijaus Merkevičiaus, Gintauto Garšvos, Rimvydo Simučio algoritmu, kuris buvo pateiktas straipsnyje [33].

18 lentelė: Duomenų apie kreditus aibės klasterizavimo rezultato lyginimas su tikrosiomis grupėmis

	rasta grupė 0	rasta grupė 1	
tikroji grupė 0	320	58	378
tikroji grupė 1	81	223	304
	401	281	

Kolegos sprendė klasifikavimo uždavinį naudodami neuroninių tinklų modelį paremtą savaime evoliucionuojančiais žemėlapiais (angl. *self-organizing map*). Uždavinys sprendžiamas apmokant algoritmą su skirtingais duomenų poaibiais bei jį testuojant su likusiais duomenimis. Parenkamas geriausias algoritmas, su kuriuo klasifikavimo klaida mažiausia. Pagal pateiktą 18 lentelę matome, kad pirmoji grupė klasterizavimo algoritmo pagalba buvo surasta 85 % tikslumu, o antroji 73 %. Tuo tarpu autoriai gavo, kad pirmoji grupė buvo surasta 80.85 % tikslumu, o antroji 81.79 %. Matome, kad rezultatai yra panašūs, tačiau reikėtų atkreipti dėmesį, kad mūsų rezultatas buvo gautas iš pirmo karto, o straipsnio autoriai modelį testavo su keliomis testinėmis imtimis. Taip pat mūsų algoritmas yra neapmokomasis (angl. *unsupervised*), o straipsnio autorių apmokomasis (angl. *supervised*).

5 IŠVADOS

Atlikus pasiūlyto mišriųjų duomenų algoritmo testavimą galime padaryti išvadas:

1. Sintetinių tolydžiųjų ir diskrečiųjų duomenų atvejais gavome, kad Davies-Bouldin ir Dunn indeksai identifikavo tikrąjį klasterių skaičių.

2. Realių diskrečiųjų duomenų (kongreso balsavimo įrašų duomenų aibė) atveju Davies-Bouldin indeksas tiksliai identifikavo rekomenduotiną klasterių skaičių 2 bei 86% tikrųjų grupių sutampa su rastomis klasterizavimo algoritmu.

3. Realių tolydžiųjų duomenų (krūties vėžio duomenų aibė) atveju Davies-Bouldin indeksas tiksliai identifikavo rekomenduotiną klasterių skaičių 2 bei 95% tikrųjų grupių sutampa su rastomis klasterizavimo algoritmu.

4. Realių mišriųjų duomenų (mokytojų asistentų vertinimo duomenų aibė) atveju Davies-Bouldin indeksas tiksliai identifikavo rekomenduotiną klasterių skaičių 3 bei 62% tikrųjų grupių sutampa su rastomis klasterizavimo algoritmu.

5. Realių mišriųjų duomenų (australijos kreditų duomenų aibė) atveju Davies-Bouldin ir Dunn indeksai tiksliai identifikavo rekomenduotiną klasterių skaičių 2 bei 79% tikrųjų grupių sutampa su rastomis klasterizavimo algoritmu.

6. Vartotojas, naudodamasis multidimensinio masteliavimo metodu, galėjo paprasčiau suvokti nagrinėjamų daugiamačių duomenų struktūrą, atvaizduotą dvimatėje Euklido erdvėje. Taip pat naudojantis metodu buvo galima nustatyti išskirtis daugiamačiuose duomenyse.

7. Neatsitiktinis pradinių centrų pasirinkimas pagreitino algoritmo konvergavimą ir užtikrino, kad pradiniai centrai nebūtų arti vienas kito.

Matome, kad Davies-Bouldin validavimo indeksas parinktoms imtims teisingai nustatė tikrąjį klasterių skaičių, o Dunn indeksu surastas klasterių skaičius skirdavosi ± 1 nuo Davies-Bouldin indeksu rasto klasterių skaičiaus. Mišriųjų duomenų algoritmas vidutiniškai 80.5% tikslumu nustatydavo tikrąją realių duomenų grupę.

LITERATŪRA

[1] Anil K. Jain *Data Clustering: 50 Years Beyond K-Means*, Department of Computer Science & Engineering Michigan State University East Lansing, Michigan 48824 USA

[2] Frank, Ildiko E, Todeschini, Roberto *Data analysis handbook*. Elsevier Science Inc. 1994, 227–228

[3] Ester Martin, Peter Kriegel Hans *A density-based algorithm for discovering clusters in large spatial databases with noise*, 1996, 226–231

[4] Agrawal Rakesh, Gehrke Johannes, Gunopulos Dimitrios, Raghavan Prahakar *Automatic subspace clustering of high dimensional data for data mining applications*, 1998, 94–105

[5] Roberts Stephen J, Holmes Christopher, Denison Dave *Minimum-entropy data clustering using reversible jump markov chain monte carlo*, 2001, 103–110

[6] Maria Halkidi, Yannis Batistakis, Michalis Vazirgtannis *On Clustering Validation Techniques*, 2001

[7] J.MacQueen *Some methods for classification and analysis of multivariate observations*, Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability,vol.1,University of California Press, Berkeley,CA 1967, 281–297

[8] S.V.D. Arthur *k-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, Louisiana, 2007, 1027–1035

[9] R.R.T. Zhang, M.Livny, *Birch: an efficient data clustering method for very large databases*,SIGMOD Record 25, 1996, 103–114

[10] Z.Huang *Extensions to the k-means algorithm for clustering large datasets with categorical values*, Data Mining and Knowledge Discovery, 1998, 283–304

[11] J.C.D. Barbara, Y.Li, *Coolcat:an entropy-based algorithm for categorical clustering*, in: Proceedings of the 11th ACM Conference on Information and Knowledge Management(CIKM02), ACMPress, McLean, Virginia, USA, 2002, 582–589

[12] Ming-Yi Shih,Jar-Wen Jheng,Lien-Fu Lai *A Two-Step Method for Clustering Mixed Categroical and Numeric Data*, 2010

[13] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim *ROCK: A Robust Clustering Algorithm for Categorical Attributes*, Information Systems, Volume 25, Issue 5, July 2000, 345–366

[14] Charu C. Aggarwal, Chandan K. Reddy *Data clustering Algorithms and Applications*,978-1-4665-5822-9, 2014

[15] Ming-Yi Shih,Jar-Wen Jheng and Lien-Fu Lai *A Two-Step Method for Clustering Mixed Categroical and Numeric Data*,Tamkang Journal of Science and Engineering, Vol. 13, No. 1, pp. 11-19 (2010)

[16] Zengyou He, Xiaofe i Xu,Schengchun Deng *Clustering mixed numeric and catgorical data: A cluster ensemble approach*

[17] Dileep Kumar Murala *Divide and Conquer Method for Clustering Mixed Numerical and Categorical Data*, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (1), 2013, 103 - 106

[18] Gil David, Amir Averbuch *SpectralCAT: Categorical spectral clustering of numerical and nominal data*, 2012

[19] J.C. Gower, *A general coefficient of similarity and some of its properties*, Biometrics 27 (1971) 857–871

[20] Mindaugas Kavaliauskas *Daugiamčių Gauso skirstinių statistinė analizė, taikant duomenų projektavimą*, Vilnius, 2005

[21] Erendira Rendon, Itzel Abundez, Alejandra Arizmendi, Elvia M. Quiroz *Internal versus External cluster validation indexes* Ineternational Journal Of Computers And Communications, Volume 5, 2011

[22] https://en.wikipedia.org/wiki/Multidimensional_scaling

[23] JJ. Liang, et al., *Determining the number of clusters using information entropy for mixed data*, Pattern Recognition (2012), doi:10.1016/j.patcog.2011.12.17

[24] Tao Li,Sheng Ma,Mitsunori Ogihara *Entropy-Based Criterion in Categorical Clustering*

[25] A. Renyi *On measures of entropy and information*, in: Proceeding of the 4th Berkeley Symposium on Mathematics of Statistics and Probability, 1961, 547–561

[26] A. Renyi *On measures of entropy and information*, in: Proceeding of the 4th Berkeley Symposium on Mathematics of Statistics and Probability, 1961, 547–561

[27] https://en.wikipedia.org/wiki/Rough_set

[28] https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

[29] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

[30] https://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation

[31] https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

[32] https://archive.ics.uci.edu/ml/datasets/Statlog+(Australian+Credit+Approval)

[33] Egidijus Merkevičius, Gintautas Garšva, Rimvydas Simutis *Forecasting of credit classes with the self-organizing maps*, ISSN 1392 – 124X informacinės technologijos ir valdymas, 2004, Nr. 4 (33)

A PRIEDAS. PROGRAMŲ TEKSTAI

```
'''
-need to have numerical attribute names NumAtr, and categorical CatAtr
-need to have first columns categorical and others numerical due to
 Kprototypes function SN,SD calculation
'''
import pandas as pd
import numpy as np
import random
import csv
import math
import kPrototypes as kp
import SBAE_C
import SBAE_N
import CUM
import SBAE_M
import generateGaussian
import plotClusters
import plotCum
import multidimensionalScaling
import Silhouette
import dunnIndex
import chooseInitialCentroids
import DaviesBouldin
import matplotlib.pyplot as plt
from scipy.stats import mode


def Step2(distance_matrix):
    ShortestDistances = np.argmin(distance_matrix,axis = 0)
    return(ShortestDistances.tolist())



def newCenters(dataset,CatAtr,NumAtr,cluster_indices,clust_numb):
    clusters = {j: [] for j in range(clust_numb)}
    Ar = len(NumAtr)
    Ac = len(CatAtr)
    for i,clust_ind in enumerate(cluster_indices):
        clusters[clust_ind].append(i)
    new_centroid = {i: [] for i in range(len(clusters))}
    subsetCat = dataset[CatAtr].values
    subsetNum = dataset[NumAtr].values

    if Ar and Ac:
        for key,cluster in clusters.items():
            new_centroid[key] = list(np.concatenate((mode(subsetCat[cluster]\
            ,axis = 0)[0][0],subsetNum[cluster].mean(axis = 0))))
```

```
            return [new_centroid,clusters]
        elif not Ar:
            for key,cluster in clusters.items():
                if cluster:
                    new_centroid[key] = list(mode(subsetCat[cluster],axis = 0)[0][0])
                else:
                    print("cluster {} is empty" . format(key))
            return [new_centroid,clusters]
        elif not Ac:
            for key,cluster in clusters.items():
                new_centroid[key] = list(subsetNum[cluster].mean(axis = 0))
            return [new_centroid,clusters]

directory = "../datasets/australian_credit_approval_without_label_orig.csv"
data_set = pd.read_csv(directory)
k_min = 1
k_max = 5
sigma = 2.5



def main(data_set,k_min,k_max,sigma):
    #numerical attributes names
    NumAtr = ['a9','a10','a11','a12','a13','a14']
    #categorial attributes names
    CatAtr = ['a1','a2','a3','a4','a5','a6','a7','a8']
    #numerical subset normalization
    data_set[NumAtr] = (data_set[NumAtr] - data_set[NumAtr].mean())\
    /(data_set[NumAtr].std())

    dunn_index = []
    davies = []
    U = data_set.shape[0]
    #Z = random.sample(range(0,U),k_max)
    Z = chooseInitialCentroids.chooseInitialCentroids(data_set,CatAtr,NumAtr,k_max)
    prototypes = {i: data_set[j:j+1].values.tolist()[0] for i,j in enumerate(Z)}
    #prototypes = {0: ['a', 'b'], 1: ['t', 'o'], 2: ['y', 'i'], 3: ['b', 'a']}
    outfile = open('../output/output.csv', 'w')
    writer = csv.writer(outfile, delimiter=',', quotechar='"')
    outfile.write("Clustering started \n")
    outfile.write("Initial centers: \n")
    writer.writerows([prototypes.values()])
    clusters_numb = list(range(k_max,k_min,-1))

    for clust_numb in clusters_numb:
        print("{} clusters".format(clust_numb))
        outfile.write("{} clusters \n" . format(clust_numb))
        # Generate random initial prototopes
        distance_matrix = np.zeros(shape = (clust_numb,U))
```

```
    cluster_indices = [0] * U
    clusters_different = True
    #k-prototypes algorithm
    while clusters_different:
        for i in range(clust_numb):
            for u in range(U):
                distance_matrix[i][u] = kp.Kprototypes(data_set,CatAtr,\
                NumAtr,prototypes,prototypes[i],u)
        new_indices = Step2(distance_matrix)
        if cluster_indices == new_indices:
            outfile.write("clustering stopped \n")
            clusters_different = False
        else:
            outfile.write("indices \n")
            writer.writerows([new_indices])
            cluster_indices = list(new_indices)
            [prototypes,clusters] = newCenters(data_set,CatAtr,NumAtr,\
            cluster_indices,clust_numb)
            if [] in clusters.values():
                print('Empty cluster')
            outfile.write("prototypes \n")
            writer.writerows([prototypes.values()])

    multidimensionalScaling.cmdscale(data_set,CatAtr,NumAtr,clusters,\
    'categorical',clust_numb,"save")

    dunn_index.append(dunnIndex.dunnIndex(data_set,CatAtr,NumAtr,clusters))
    davies.append(DaviesBouldin.DaviesBouldin(data_set,CatAtr,NumAtr,\
    clusters,prototypes))
    # Worst cluster identification
    if clust_numb > 2:
        SBAE_M_values = []
        for cluster_k in clusters.keys():
            SBAE_M_values.append(SBAE_M.SBAE_M(data_set,cluster_k,clusters,\
            CatAtr,NumAtr,sigma))

        outfile.write("SBAE_M values \n")
        writer.writerows([SBAE_M_values])
        worst_clust_numbb = np.argmax(SBAE_M_values)
        mod_distance_matrix = np.delete(distance_matrix,worst_clust_numbb,\
        axis = 0)
        mod_indices = Step2(mod_distance_matrix)
        [prototypes,clusters] = newCenters(data_set,CatAtr,NumAtr,\
        mod_indices,clust_numb-1)
    else:
        continue

#Plotting results
outfile.close()
```

```
    plotCum.plotCum(clusters_numb,dunn_index,"Klasteriu skaiius",\
    "Duno indeksas",'dunn_plot')
    plotCum.plotCum(clusters_numb,davies,"Klasteriu skaiius",\
    "Davies bouldin indeksas",'davies_plot')


a = main(data_set,k_min,k_max,sigma)
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 10 14:02:23 2016

@author: Mindaugas
"""
import math
import itertools
import numpy as np

def gauss_kernel(sigma,d,x,y):
    return (1/(((2*math.pi)**(d/2))*(sigma**d)))*math.exp(-np.dot((x-y),\
    (x-y))/(2*sigma**2))

def BE_N(data_set,cluster1,cluster2,clusters,NumAtr,sigma):
    '''
    Two clusters validation index for numerical attrib
    p. 4, eq. 9
    '''
    subset1 = data_set[NumAtr].values[clusters[cluster1]]
    subset2 = data_set[NumAtr].values[clusters[cluster2]]
    N1 = subset1.shape[0]
    N2 = subset2.shape[0]
    s = 0
    for i in range(N1):
        for j in range(N2):
            s += gauss_kernel(sigma,2,subset1[i],subset2[j])
    return -math.log(s/(N1*N2))

def SBAE_N(data_set,cluster1,clusters,NumAtr,sigma):
    rez = 0
    C_k = list(clusters.keys())
    C_k.remove(cluster1)
    permutations = list(itertools.combinations(C_k,2))
    for permutation in permutations:
        rez += BE_N(data_set,permutation[0],permutation[1],clusters,NumAtr,sigma)
    return rez
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 10 14:05:44 2016
```

```
@author: Mindaugas
"""
import itertools
import distances as ds

def BE_C(data_set,cluster1,cluster2,clusters,CatAtr):
    '''
    Two clusters validation index for categorical attrib
    p. 5, eq. 16
    '''
    subset1 = data_set[CatAtr].values[clusters[cluster1]]
    subset2 = data_set[CatAtr].values[clusters[cluster2]]
    N1 = subset1.shape[0]
    N2 = subset2.shape[0]
    s = 0
    for i in range(N1):
        for j in range(N2):
            s += ds.hamdist(subset1[i],subset2[j])
    return s/(N1*N2)

def SBAE_C(data_set,cluster1,clusters,CatAtr):
    rez = 0
    C_k = list(clusters.keys())
    C_k.remove(cluster1)
    permutations = list(itertools.combinations(C_k,2))
    for permutation in permutations:
        rez += BE_C(data_set,permutation[0],permutation[1],clusters,CatAtr)
    return rez
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 10 14:14:21 2016

@author: Mindaugas
"""
import SBAE_N
import SBAE_C
def SBAE_M(data_set,cluster,clusters,CatAtr,NumAtr,sigma):
    Ar = len(NumAtr)
    Ac = len(CatAtr)
    A = Ac + Ar
    C_k = list(clusters.keys())
    SUM_SBAE_N = 0
    SUM_SBAE_C = 0

    if Ar and Ac:
        KTH_SBAE_N = SBAE_N.SBAE_N(data_set,cluster,clusters,NumAtr,sigma)
```

```
        KTH_SBAE_C = SBAE_C.SBAE_C(data_set,cluster,clusters,CatAtr)

        for k in C_k:
            SUM_SBAE_N += SBAE_N.SBAE_N(data_set,k,clusters,NumAtr,sigma)
            SUM_SBAE_C += SBAE_C.SBAE_C(data_set,k,clusters,CatAtr)
        return (Ar/A) * (KTH_SBAE_N/SUM_SBAE_N) + (Ar/A) * (KTH_SBAE_C/SUM_SBAE_C)
    elif not Ar:
        KTH_SBAE_C = SBAE_C.SBAE_C(data_set,cluster,clusters,CatAtr)
        for k in C_k:
            SUM_SBAE_C += SBAE_C.SBAE_C(data_set,k,clusters,CatAtr)
        return (Ac/A) * (KTH_SBAE_C/SUM_SBAE_C)
    elif not Ac:
        KTH_SBAE_N = SBAE_N.SBAE_N(data_set,cluster,clusters,NumAtr,sigma)
        for k in C_k:

            SUM_SBAE_N += SBAE_N.SBAE_N(data_set,k,clusters,NumAtr,sigma)
        return (Ar/A) * (KTH_SBAE_N/SUM_SBAE_N)
```

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 12 21:48:46 2016

@author: Mindaugas
"""
import distances as ds
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
def cmdscale(data_set,CatAtr,NumAtr,clusters,title,number,status):
    # http://www.nervouscomputer.com/hfs/cmdscale-in-python/
    """
    Classical multidimensional scaling (MDS)

    Parameters
    ----------
    D : (n, n) array
        Symmetric distance matrix.

    Returns
    -------
    Y : (n, p) array
        Configuration matrix. Each column represents a dimension. Only the
        p dimensions corresponding to positive eigenvalues of B are returned.
        Note that each dimension is only determined up to an overall sign,
        corresponding to a reflection.

    e : (n,) array
        Eigenvalues of B.
```

```
"""
subsetCat = data_set[CatAtr].as_matrix()
subsetNum = data_set[NumAtr].as_matrix()
U = data_set.shape[0]
D = np.zeros(shape = (U,U))

clust_numb = len(clusters)
colors = iter(plt.cm.rainbow(np.linspace(0, 1, clust_numb)))
Ar = len(NumAtr)
Ac = len(CatAtr)
A = Ar+Ac


if Ar and Ac:
    for i in range(U):
        for j in range(U):
            D[i][j] =  (Ar/A)*ds.Eucdist(subsetNum[i],subsetNum[j]) +\
            (Ac/A)*ds.hamdist(subsetCat[i],subsetCat[j])
elif not Ar:
    for i in range(U):
        for j in range(U):
            D[i][j] = ds.hamdist(subsetCat[i],subsetCat[j])
elif not Ac:
    for i in range(U):
        for j in range(U):
            D[i][j] =  ds.Eucdist(subsetNum[i],subsetNum[j])



# Number of points
n = len(D)
# Centering matrix
H = np.eye(n) - np.ones((n, n))/n
# YY^T
B = -H.dot(D**2).dot(H)/2
# Diagonalize
evals, evecs = np.linalg.eigh(B)
# Sort by eigenvalue in descending order
idx   = np.argsort(evals)[::-1]
evals = evals[idx]
evecs = evecs[:,idx]
# Compute the coordinates using positive-eigenvalued components only
w, = np.where(evals > 0)
L  = np.diag(np.sqrt(evals[w]))
V  = evecs[:,w]
Y  = V.dot(L)


if status == "save":
```

```
        pp = PdfPages('../output/{}_{}_multidimensional_scaling.pdf'.\
        format(title,number))
        for key,cluster in clusters.items():
            plt.scatter(x = Y[cluster][:,0],y = Y[cluster][:,1],color =\
            next(colors))
            plt.xlabel("x1")
            plt.ylabel("x2")
        pp.savefig()
        pp.close()
    else:
        for key,cluster in clusters.items():
            plt.scatter(x = Y[cluster][:,0],y = Y[cluster][:,1],color =\
            next(colors))
            plt.xlabel("x1")
            plt.ylabel("x2")
        plt.show()
    return Y
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 24 22:51:33 2016

@author: Mindaugas
"""
import numpy as np
import distances as ds



def diameter(subsetNum,subsetCat,cluster_indexes,CatAtr,NumAtr,prototype,clust_numb):
    Ar = len(NumAtr)
    Ac = len(CatAtr)
    A = Ar + Ac
    ss = 0
    if Ar and Ac:
        for x in cluster_indexes:
            ss += (Ar/A)*ds.Eucdist(subsetNum[x],prototype[Ac:A]) +\
            (Ac/A)*ds.hamdist(subsetCat[x],prototype[0:Ac])
        return ss/len(cluster_indexes)

    elif not Ar:
        for x in cluster_indexes:
            ss += ds.hamdist(subsetCat[x],prototype[0:Ac])
        return ss/len(cluster_indexes)
    elif not Ac:
        for x in cluster_indexes:
            ss += ds.Eucdist(subsetNum[x],prototype[Ac:A])
        return ss/len(cluster_indexes)
```

```
def DaviesBouldin(data_set,CatAtr,NumAtr,clusters,prototypes):
    clusters_numb = len(clusters)
    subsetCat = data_set[CatAtr].as_matrix()
    subsetNum = data_set[NumAtr].as_matrix()
    Ar = len(NumAtr)
    Ac = len(CatAtr)
    A = Ar+Ac

    if Ar and Ac:
        s = []
        for i,cluster_i in enumerate(clusters.values()):
            prototype_i = prototypes[i]
            d_i = diameter(subsetNum,subsetCat,cluster_i,CatAtr,NumAtr,\
            prototype_i,clusters_numb)
            t = 0
            for j,cluster_j in enumerate(clusters.values()):
                if i != j:
                    prototype_j = prototypes[j]
                    d_i = diameter(subsetNum,subsetCat,cluster_i,CatAtr,NumAtr,\
                    prototype_i,clusters_numb)
                    d_j = diameter(subsetNum,subsetCat,cluster_j,CatAtr,NumAtr,\
                    prototype_j,clusters_numb)
                    d_ij = (Ar/A)*ds.Eucdist(prototype_i[Ac:A],prototype_j[Ac:A])\
                    + (Ac/A)*ds.hamdist(prototype_i[0:Ac],prototype_j[0:Ac])
                    t += (d_i+d_j)/d_ij
            s.append(t/len(clusters))
        return max(s)

    elif not Ar:
        s = []
        for i,cluster_i in enumerate(clusters.values()):
            prototype_i = prototypes[i]
            d_i = diameter(subsetNum,subsetCat,cluster_i,CatAtr,NumAtr,\
            prototype_i,clusters_numb)
            t = 0
            for j,cluster_j in enumerate(clusters.values()):
                if i != j:
                    prototype_j = prototypes[j]
                    d_j = diameter(subsetNum,subsetCat,cluster_j,CatAtr,NumAtr,\
                    prototype_j,clusters_numb)
                    d_ij = ds.hamdist(prototype_i[0:Ac],prototype_j[0:Ac])
                    t += (d_i+d_j)/d_ij
            s.append(t/len(clusters))
        return max(s)
    elif not Ac:
        s = []
        for i,cluster_i in enumerate(clusters.values()):
```

```
            prototype_i = prototypes[i]
            d_i = diameter(subsetNum,subsetCat,cluster_i,CatAtr,NumAtr,\
            prototype_i,clusters_numb)
            t = 0
            for j,cluster_j in enumerate(clusters.values()):
                if i != j:
                    prototype_j = prototypes[j]
                    d_j = diameter(subsetNum,subsetCat,cluster_j,CatAtr,\
                    NumAtr,prototype_j,clusters_numb)
                    d_ij = (ds.Eucdist(prototype_i[Ac:A],prototype_j[Ac:A]))
                    t += (d_i+d_j)/d_ij
            s.append(t/len(clusters))
        return max(s)
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 17 20:33:11 2016

@author: Mindaugas
"""
import numpy as np
import distances as ds

def dunnIndex(data_set,CatAtr,NumAtr,clusters):
    clusters_numb = len(clusters)
    subsetCat = data_set[CatAtr].as_matrix()
    subsetNum = data_set[NumAtr].as_matrix()
    U = data_set.shape[0]
    D = np.zeros(shape = (U,U))
    Ar = len(NumAtr)
    Ac = len(CatAtr)
    A = Ar+Ac

    if Ar and Ac:
        for i in range(U):
            for j in range(U):
                D[i][j] =  (Ar/A)*ds.Eucdist(subsetNum[i],subsetNum[j]) +\
                (Ac/A)*ds.hamdist(subsetCat[i],subsetCat[j])
    elif not Ar:
        for i in range(U):
            for j in range(U):
                D[i][j] = ds.hamdist(subsetCat[i],subsetCat[j])
    elif not Ac:
        for i in range(U):
            for j in range(U):
                D[i][j] =  ds.Eucdist(subsetNum[i],subsetNum[j])
```

```
    nc = len(clusters)
    interClust = np.zeros(shape = (nc,nc))
    intraClust = np.zeros(shape = (1,nc))

    interClust = np.empty((nc,nc,))
    interClust[:] = np.NAN

    for i in range(nc):
        c1 = clusters[i]
        for j in range(nc):
            if j == i:
                D_sub = D[c1,:]
                D_sub = D_sub[:,c1]
                intraClust[0][i] = np.max(np.max(D_sub,axis = 0))
            if j > i:
                c2 = clusters[j]
                D_sub_j = D[c1,:]
                D_sub_j = D_sub_j[:,c2]
                interClust[i][j] = np.min(np.min(D_sub_j,axis = 0))
    return np.nanmin(interClust)/np.max(intraClust)
```

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 23 21:11:15 2016

@author: Mindaugas
"""
import numpy as np
import distances as ds

def chooseInitialCentroids(data_set,CatAtr,NumAtr,clust_numb):
    subsetCat = data_set[CatAtr].as_matrix()
    subsetNum = data_set[NumAtr].as_matrix()
    U = data_set.shape[0]
    D = np.zeros(shape = (U,U))

    Ar = len(NumAtr)
    Ac = len(CatAtr)
    A = Ar+Ac

    if Ar and Ac:
        for i in range(U):
            for j in range(U):
                D[i][j] =  (Ar/A)*ds.Eucdist(subsetNum[i],subsetNum[j]) + \
                (Ac/A)*ds.hamdist(subsetCat[i],subsetCat[j])
    elif not Ar:
        for i in range(U):
            for j in range(U):
                D[i][j] = ds.hamdist(subsetCat[i],subsetCat[j])
```

```
    elif not Ac:
        for i in range(U):
            for j in range(U):
                D[i][j] =  ds.Eucdist(subsetNum[i],subsetNum[j])
    centers = []
    remembered_indexes = [i for i in range(U)]
    step = int(U/clust_numb)
    for i in range(clust_numb):
        sums_axis = []
        sums_axis = list(np.sum(D,axis = 0))
        max_ind =  sums_axis.index(max(sums_axis))
        max_ind_real = remembered_indexes[max_ind]
        centers.append(max_ind_real)
        max_ind_row = list(D[max_ind,:])
        for j in range(step):
            min_ind_inner =  max_ind_row.index(min(max_ind_row))
            D = np.delete(D, (min_ind_inner), axis=0)
            D = np.delete(D, (min_ind_inner), axis=1)
            remembered_indexes.pop(min_ind_inner)
            max_ind_row.pop(min_ind_inner)
    return centers
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 10 14:06:47 2016

@author: Mindaugas
"""
import numpy as np
from math import sqrt

def hamdist(str1, str2):
    '''
    Hammington distance
    '''
    diffs = 0
    for ch1, ch2 in zip(str1, str2):
            if ch1 != ch2:
                    diffs += 1
    return diffs

def Eucdist(X2,X1):
    '''
    Euclidean distance
    '''
    return sqrt(sum(np.subtract(X1,X2)**2))
```

```
# -*- coding: utf-8 -*-
"""
```

```
Created on Sun Apr 10 14:20:40 2016

@author: Mindaugas
"""
import distances as ds

def Kprototypes(dataset,CatAtr,NumAtr,Z,z,x):
    subsetCat = dataset[CatAtr].as_matrix()
    subsetNum = dataset[NumAtr].as_matrix()

    Ar = len(NumAtr)
    Ac = len(CatAtr)
    A = Ar+Ac
    SN = 0
    SD = 0
    if Ar and Ac:
        for key,prototype in Z.items():
            SN += ds.Eucdist(subsetNum[x],prototype[Ac:A])
            SD += ds.hamdist(subsetCat[x],prototype[0:Ac])
        return((Ar/(A*SN))*ds.Eucdist(subsetNum[x],z[Ac:A])+\
        (Ac/(A*SD))*ds.hamdist(subsetCat[x],z[0:Ac]))
    elif not Ar:
        for key,prototype in Z.items():
            SD += ds.hamdist(subsetCat[x],prototype[0:Ac])
        return((Ac/(A*SD))*ds.hamdist(subsetCat[x],z[0:Ac]))
    elif not Ac:
        for key,prototype in Z.items():
            SN += ds.Eucdist(subsetNum[x],prototype[0:Ar])
        return((Ar/(A*SN))*ds.Eucdist(subsetNum[x],z[0:Ar]))
```