

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

**Intelektikos pagrindai**

Individualus darbas

Data: 2021-05-07

**Dėstytojai:**

Agnė Paulaiskaitė-Tarasevičienė

Germanas Budnikas

**Studentai:**

Arvydas Miklovis IFF-8/10

Dainius Čepulis IFF8/10

KAUNAS, 2021

Turinys

[1. Darbo dalys 2](#_Toc73269240)

[2. Įvadas 2](#_Toc73269241)

[3. Duomenų pasiruošimas 2](#_Toc73269242)

[4. Gaussian Mixture 2](#_Toc73269243)

[4.1. Modelio pritaikymas 2](#_Toc73269244)

[4.2. Išvados 5](#_Toc73269245)

[5. K-vidurkių metodas 6](#_Toc73269246)

[5.1. Inercija 6](#_Toc73269247)

[5.2. Silueto koeficiantas 6](#_Toc73269248)

[5.3. Vizualizacija 7](#_Toc73269249)

[5.4. Išvados 9](#_Toc73269250)

[6. Programos kodas 10](#_Toc73269251)

[7. Apibendrintos išvados 10](#_Toc73269252)

# Darbo dalys

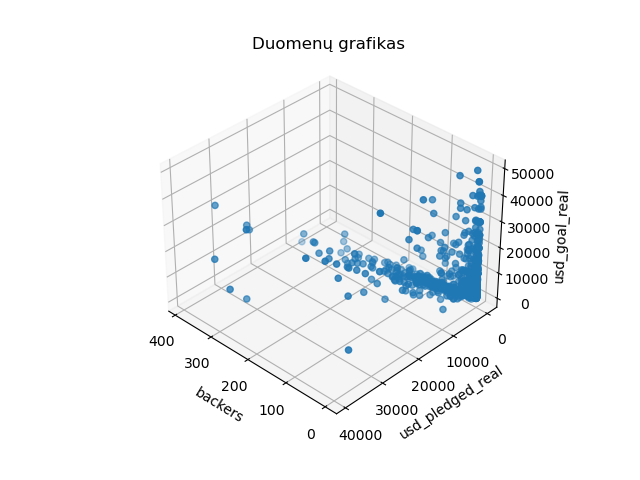
Gaussian Mixture - Dainius Čepulis

K-vidurkiai- Arvydas Miklovis

# Įvadas

Metodams palyginti buvo naudotas duomenų rinkinys iš pirmo laboratorinio darbo. Iš jo naudosime tik 3 atributus, nes buvo pastebėtai, kad jie vizualiai išsiskirto klasterius. Duomenų rinkinys dėl paprastumo buvo sumažintas, tačiau tai menkai, keičia atsakymus.

Taip pat galime pažiūrėti į pasirinktų duomenų grafika.



Iš grafiko, matome kad vizualiai duomenis galime ganėtinai lengvai padalinti į du klasterius, vienas susitelkęs prie „backers“ ir „usd\_pledged\_real“ ašies, o kitas yra sudarytas iš likusiu duomenų, kuris sudaro „šluotos“ formos plokštumą.

pav. 1 pasirinktų duomenų grafikas

# Duomenų pasiruošimas

Kadangi mūsų duomenys turėjo įtin ekstremalių reikšmių, prieš taikydami duomenis modeliuose, jas pašalinome.

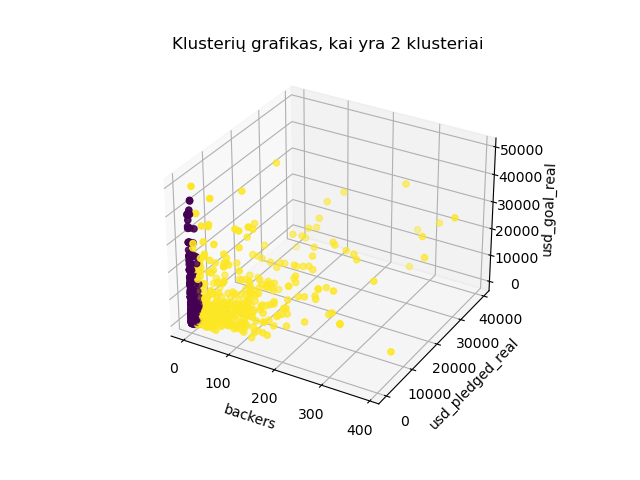
# Gaussian Mixture

Kadangi buvo sunku atrast python bibliotekas, kurios palaike SOM su klasterių limitu, buvo pasirinktas kitas metodas.

Gaussian Mixture yra mašininio mokimosi metodas, kuris atranda klasterius darydamas prielaida, kad šie klasteriai sudaro normalų pasiskirstymą.

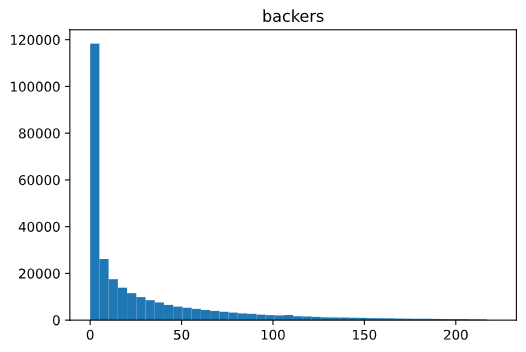
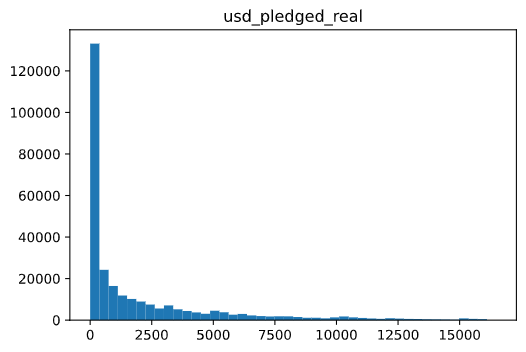
## Modelio pritaikymas

Kad įvertintumėme, kaip gerai šis metodas iškrito klasterius ir kada jis susidaro geriausius klasterius keisime jų skaičių (2,3,5,9). Ir gautus rezultatus parodysime trimatėje erdvėje, kiekvieną duomenų imties elementą nuspalvinę jam priskirto klasterio spalva.



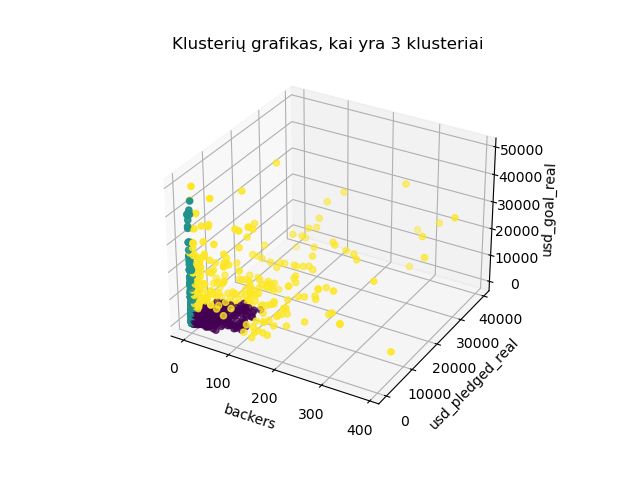
pav. 2. Išskirti du klasteriai

Kai buvo pasirinkti 2 klusteriai matome, duomenis buvo išskirstyti, kurie turėjo mažai „backers“ ir „usd\_pledged\_real“. Ką tai galimai reikštu galime pažiūrėti iš histogramų.

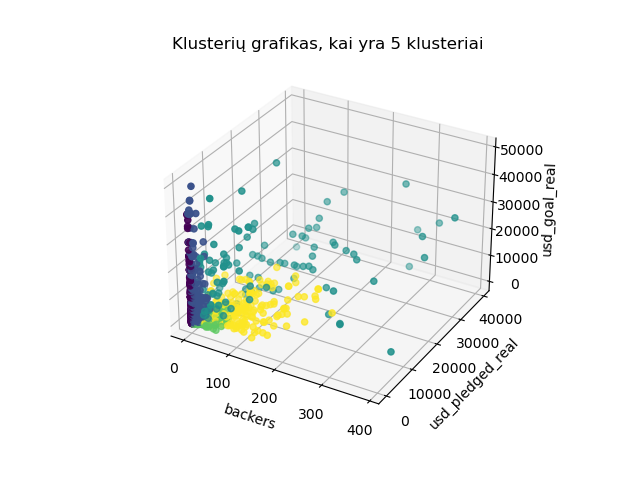
pav. 3. „backers“ ir „usd\_pleadged\_real“ histogramos

Ir šių duomenų histogramų matome, kad tai atsitiko, kad tokiu reikšmių buvo labai daug. Ir tai yra ganėtinai geras pasirinkimas, nes tai yra galimai nepasisekė projektai.



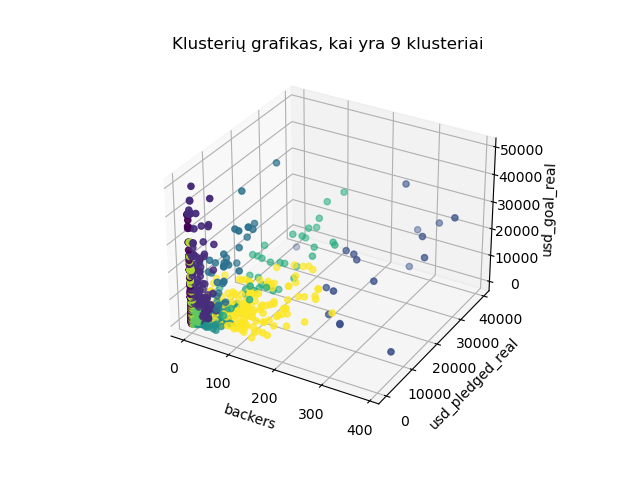
pav. 4. Išskirti trys klasteriai

Kai nustatėme klasterių į 3, mūsų prieš tai aptartas klasteris išliko, o trečias klasteris buvo atskirtas nuo antro. Ir galima spėti, kad buvo atskirti maži, galimai pasisekė projektai.



pav. 5. Išskirti penki klasteriai

Iš 5 klasterių grafiko jau sunku nuspręsti, ką kiekvienas klasteris gali reikšti. Tačiau jie neatrodo pernelyg netikėtini.



pav. 6. Išskirti devini klasteriai

Kai sudarome 9 klasterių grafika, jis pasidaro sunkiai skaitomas ir galime tikėtis, kad pasirinkome per daug klasterių.

## Išvados

Kadangi dažnai raliuose duomenyse yra normaliojo pasiskirstymo tendencijų, šis metodas buvo neblogas pasirinkimas ir išskirti klasteriai, kai pasirenkama tarp 2 ir 5 klasterių, buvo ganėtinai tikėtini, tačiau dėl darbo apimties buvo sunkų tai įvertinti.

# K-vidurkių metodas

**K-vidurkių metodas** – neprižiūrimojo tipo duomenų panašumu grįstas algoritmas, kuris duomenis bando susiskirstyti į 𝐾 nepersidengiančių klasterių. Nustatyti, ar gerai atliktas grupavimas, būtų galima, jeigu turėtume atsakymus; deja, šiuose uždaviniuose tokios informacijos nėra. K-vidurkių metodas labiau skirtas panašumui tarp duomenų nustatyti ir pagal tai pasidaryti tam tikras išvadas. Tam tikrais atvejais duomenys labai aiškiai skiriasi pagal konkrečias savybes, o kartais jie būna per daug panašūs, ir toks sugrupavimas gali būti betikslis

Modelis skaičiuotas 3 skirtingomis atributų variacijomis

Pirmas variantas:  *backers, usd\_pledged\_real, usd\_goal\_real*

Antras variantas:  *project\_time, usd\_pledged\_real, usd\_goal\_real*

Trečias variantas:  *backers, usd\_pledged\_real, project\_time*

## Inercija

|  |  |  |  |
| --- | --- | --- | --- |
| **Klasteriai** | **Inercijos reikšmė NR1** | **Inercijos reikšmė NR2** | **Inercijos reikšmė NR3** |
| 2 | 43228073402 | 8874580746 | 8968415223 |
| 3 | 28279830243 | 4042952354 | 3822256545 |
| 4 | 20201370010 | 2349934956 | 2128245239 |
| 5 | 14439257859 | 1490596688 | 1396440898 |
| 6 | 11200675896 | 896960720 | 975760813 |
| 7 | 9183050286 | 662409575 | 682575219 |
| 8 | 7703207724 | 504637797 | 493187101 |
| 9 | 6886510287 | 370483087 | 359299982 |
| 10 | 6041935061 | 300983535 | 295338352 |

|  |  |  |
| --- | --- | --- |
|  | pav. 7. Inercijos ir klasterių priklausomybė |  |

Pagal gautus rezultatus, matome, jog vadinamoji „alkūnė“- rekomenduojamas klasterių kiekis skiriasi nuo atributų variantų. Pirmame variante tinkami pasirinkimai būtų 5-7 klasteriai, kai tuo tarpu antrame ir trečiame variante alkūnė gauname ties 4-6 klasteriais.

## Silueto koeficiantas

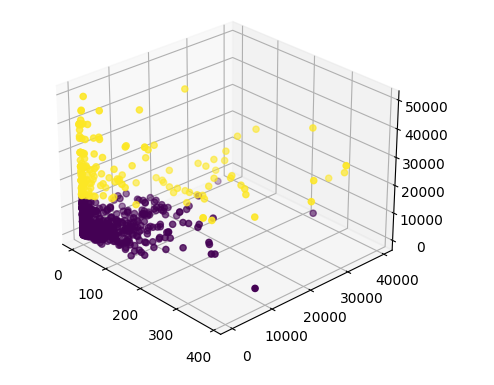
|  |  |  |  |
| --- | --- | --- | --- |
| **Klasteriai** | **Inercijos reikšmė NR1** | **Inercijos reikšmė NR2** | **Inercijos reikšmė NR3** |
| 2 | 0.650 | 0.751 | 0.800 |
| 3 | 0.661 | 0.700 | 0.773 |
| 4 | 0.559 | 0.666 | 0.746 |
| 5 | 0.600 | 0.656 | 0.721 |
| 6 | 0.536 | 0.614 | 0.712 |
| 7 | 0.546 | 0.616 | 0.705 |
| 8 | 0.542 | 0.613 | 0.708 |
| 9 | 0.502 | 0.621 | 0.706 |
| 10 | 0.491 | 0.620 | 0.692 |

|  |  |  |
| --- | --- | --- |
|  | pav. 8. Silueto koeficiantas |  |

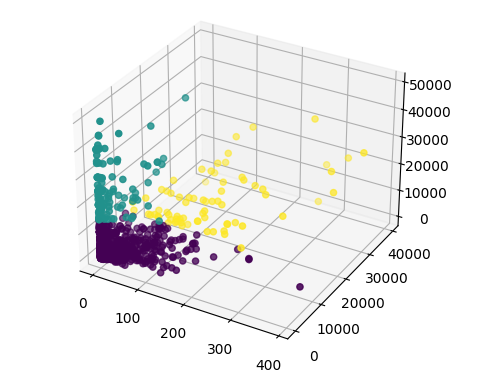
Vėlgi, kaip ir su inercija, matome, kad rezultatai skiriasi pagal atributų variacijas. Pirmuoju variantu gauname aukščiausią įverti ties 3 klasteriais, kai tuo tarpu antrame ir trečiame variantuose, geriausi rezultatai rinkinį skeliant tik į du klasterius. Taip pat verta paminėti, jog labiausiai klasteriai išsiskyrė trečiuoju duomenų rinkinių ir įvertis gaunamas gana aukštas- 0,8.

## Vizualizacija

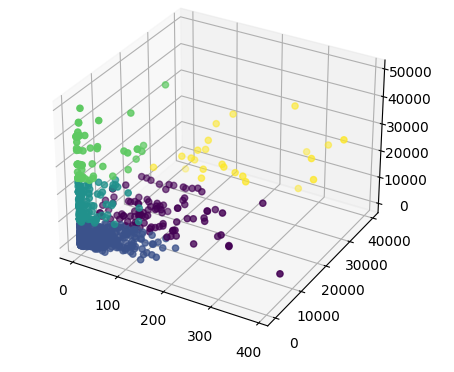
Taip pat pateikiama trečiojo duomenų rinkinio klasterių vizualizacija



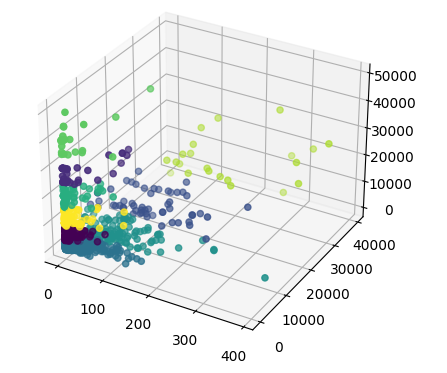
pav. 9. Išskirti du klasteriai



pav. 10. Išskirti trys klasteriai



pav. 11. Išskirti penki klasteriai



pav. 12. Išskirti devyni klasteriai

## Išvados

Priimti vienareikšmį sprendimą šitam moduliui klasterizuoti itin sudėtinga. Inertijos ir silueto metrikos sufleruoja skirtingus atsakymus. Manau sprendžiant tokią problemą taip pat reikėtų diskutuoti ir su užsakovu, galbūt jis įžvelgtu vieno ar kito pasirinkimo pranašumus ir trūkumus, bet klasterizuojant šį duomenų rinkinį rinkčiausi 2-4 klasterius. Silueto metodas sufleruoja, jog rinktis reiktu 2, inertijos alkūnė gaunama ties 4-6 klasteriais, tai 3-4 klasteriai būtų logiškiausias pasirinkimas, juolab, jog rezultatai nesiskiria drastiškai, viskas keičiasi itin palaipsniui.

# Programos kodas

K\_mean.py

1. from matplotlib import pyplot as plt
2. from sklearn.cluster import KMeans
3. import numpy as np
4. from sklearn.metrics import silhouette\_score
6. def prepareData():
7. df = np.genfromtxt("ks-project\_short.csv", delimiter=",", skip\_header=1)
8. rng = np.random.default\_rng(12345)
9. rints = rng.integers(low=0, high=df.shape[0], size=1000)
10. df = df[rints]
11. # ,0category,1main\_category,2state,3backers,4country,5usd\_pledged\_real,6usd\_goal\_real,7project\_time
12. df = np.delete(df, [0, 1, 2, 4, 6], 1)
13. df = df[np.all([df[:, 0] < 400, df[:, 1] < 50000, df[:, 2] < 50000], axis=0)]
14. return df
16. def calculateModel(df, n\_cluster):
17. model = KMeans(n\_clusters=n\_cluster, random\_state=0)
18. cluster = model.fit\_predict(df)
20. # score = model.score(df)
21. silhouette\_avg = silhouette\_score(df, cluster)
22. silhoutes.append(silhouette\_avg)
23. inertias.append(model.inertia\_)
24. print("For n\_clusters =", n\_cluster,
25. "The average silhouette\_score is :", silhouette\_avg)
26. print("For n\_clusters =", n\_cluster,
27. "The inertia is :", model.inertia\_)
28. return cluster
30. def draw():
31. # %% scatter plot
32. fig = plt.figure()
33. ax = plt.axes(projection='3d')
34. ax.scatter(df[:, 0], df[:, 1], df[:, 2], c=cluster)
35. plt.show()
37. def drawSilhoutes(values):
38. fig = plt.figure()
39. ax = plt.axes()
40. ax.scatter(range(2, len(values)+2), values)
41. plt.show()
43. df = prepareData()
44. silhoutes = []
45. inertias = []
46. for i in range(2, 11):
47. cluster = calculateModel(df, i)
48. draw()
50. drawSilhoutes(silhoutes)
51. drawSilhoutes(inertias)

# Apibendrintos išvados

K-vidurkių metodas, dėl savo apribojimų, kai klasteriai yra ne apskritimo formos, buvo netinkamas šiems duomenims išskirstyti. Tačiau sudėtingesnis Gaussian Mixture metodas, karkas geriau tiko mūsų pasirinktiems duomenims ir išskirstė juos į ganėtinai įtikinančius klasterius.