VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Duomenų tyryba. Daugiamačių duomenų dimensijų mažinimas.

Savarankiško darbo ataskaita

Atliko: Justinas Rimavičius, Edvardas Ražanskas

VU el. p.: edvardas.razanskas@mif.stud.vu.lt, justinas.rimavicius@mif.stud.vu.lt

Vertino: dr. Jolita Bernatavičienė

Vilnius
2024

TURINYS

1. ĮVADAS

1.1. Darbo tikslas

Šio darbo tikslas – pritaikant dimensijų mažinimo metodus daugiamačių duomenų vizualizavimui, atlikti SDSS (Sloan‘o skaitmeninio dangaus tyrimo DR17) duomenų rinkinio analizę, pateikti vizualizavimo rezultatus ir jų interpretaciją. Siekiama ištirti dimensijų mažinimo metodų galimybes bei atlikti lyginamąją analizę, kad nustatyti, ar žvaigždžių, galaktikų ir kvazarų požymių rinkiniai skiriasi reikšmingai ir turi galimybę būti naudojami šių kosminių objektų klasifikavimui.

1.2. Darbo uždaviniai

1. Trumpai aprašyti tiriamą duomenų aibę, jos požymius, pagrindines savybes.
2. Paruošti duomenų aibę jos analizei, ją sunormuoti.
3. Pritaikyti dimensijų mažinimo metodus su skirtingais argumentais/parametrais.
4. Vizualizuoti rezultatus.
5. Apibendrinti rezultatus ir parašyti jų išvadas.
6. Aprašyti kiekvieno metodo privalumus, trūkumus, juos palyginti.

1.3. Darbo įrankiai

Duomenų apdorojimas, transformacija, analizė ir dimensijų mažinimo metodai buvo pritaikyti naudojant „Python 3.12.0” programavimo kalbą ir jos bibliotekas (daugiau žiūrėti skyrių 6. Kodas).

2. DUOMENŲ ANALIZĖ

2.1. Tiriamos duomenų aibės ir jos požymių aprašymas

Pateiktoje žvaigždžių klasifikacijos duomenų aibėje („Stellar Classification Dataset“) yra 100000 eilučių, 18 požymių stulpelių. Jutiklių matavimai yra „float“ tipo (t.y. priklauso realiųjų skaičių aibei) , „class“ požymis yra „object“ tipo (t.y. simboliai), likę požymiai yra „int“ tipo (t.y. priklauso sveikųjų skaičių aibei).

```
Data columns (total 18 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   obj_ID       100000 non-null  float64
 1   alpha        100000 non-null  float64
 2   delta        100000 non-null  float64
 3   u            100000 non-null  float64
 4   g            100000 non-null  float64
 5   r            100000 non-null  float64
 6   i            100000 non-null  float64
 7   z            100000 non-null  float64
 8   run_ID       100000 non-null  int64
 9   rerun_ID     100000 non-null  int64
 10  cam_col      100000 non-null  int64
 11  field_ID     100000 non-null  int64
 12  spec_obj_ID  100000 non-null  float64
 13  class        100000 non-null  object
 14  redshift     100000 non-null  float64
 15  plate        100000 non-null  int64
 16  MJD          100000 non-null  int64
 17  fiber_ID     100000 non-null  int64
dtypes: float64(10), int64(7), object(1)
```

2.1 pav. pradinė duomenų aibė

Duomenų aibės požymių aprašymai:

- obj_ID = objekto identifikatorius, unikali dangaus kūno vertė, identifikuojanti objektą CAS naudojamame vaizdų kataloge.
- alpha = dešiniojo pakilimo kampas (pagal J2000 epochą)
- delta = deklinacijos kampas (pagal J2000 epochą)
- u = ultravioletinis astrofotometrinės sistemos filtras

- g = žaliasis astrofotometrinės sistemos filtras
- r = raudonasis astrofotometrinės sistemos filtras
- i = artimųjų infraraudonųjų spindulių filtras astrofotometrinė sistemoje
- z = infraraudonųjų spindulių filtras astrofotometrinė sistemoje
- run_ID = serijos numeris, naudojamas konkrečiam nuskaitymui identifikuoti
- rereun_ID = pakartotinio paleidimo numeris, nurodantis, kaip vaizdas buvo apdorotas
- cam_col = kameros stulpelis, skirtas skenavimo linijai nustatyti
- field_ID = lauko numeris kiekvienam laukui identifikuoti
- spec_obj_ID = unikalus optinių spektroskopinių objektų ID (tai reiškia, kad 2 skirtingi stebėjimai su tuo pačiu spec_obj_ID turi turėti bendrą išvesties klasę)
- class = objekto klasė (galaktika, žvaigždė arba kvazaras)
- redshift (raudonasis poslinkis) = raudonojo poslinkio vertė, pagrįsta bangos ilgio padidėjimu
- plate = plokštės ID, identifikuojantis kiekvieną SDSS plokštę
- MJD = modifikuota Julijaus data, naudojama nurodyti, kada buvo paimta tam tikra SDSS duomenų dalis
- fiber_ID = pluošto ID, identifikuojantis pluoštą, kuris nukreipė šviesą į židinio plokštumą kiekvieno stebėjimo metu

2.2. Požymių ir objektų apdorojimas

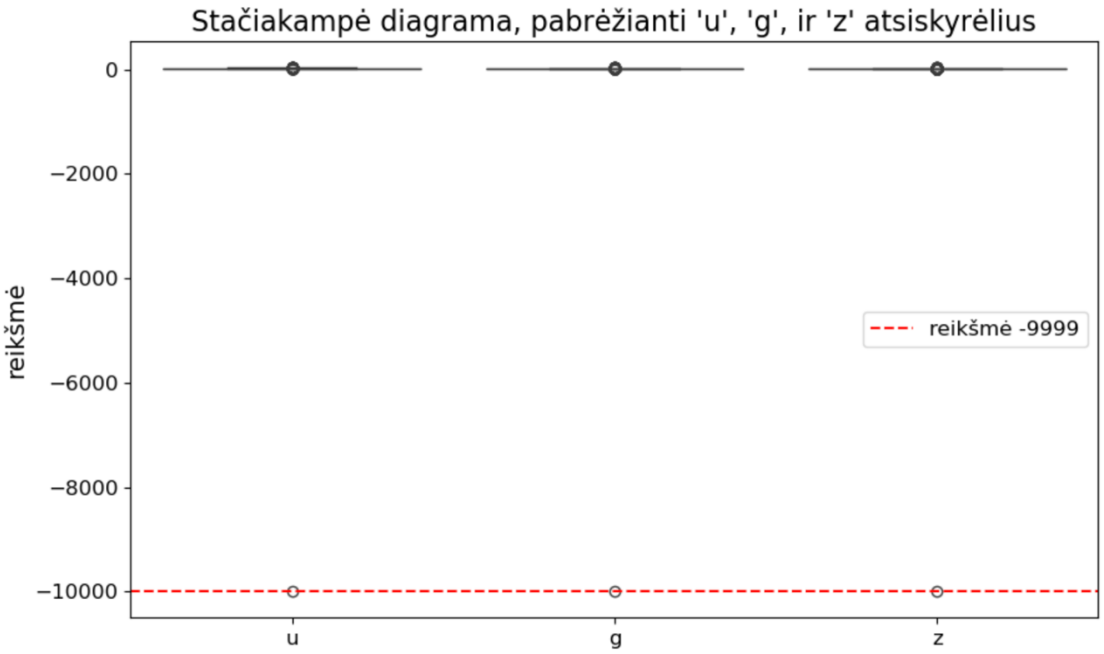
Pašalinti šie požymiai, nedarantys įtakos kosminio kūno klasifikavimui:

- „obj_ID“ požymis, nes tai identifikacinis numeris nedarantis įtakos duomenims;
- „alpha“ ir „delta“ požymiai nusako kosminio objekto poziciją, o jos nėra susijusios su skirtingų objektų (galaktikų, žvaigždžių, kvazarų) fizinėmis savybėmis;
- „spec_obj_ID“ požymis, nes 2 skirtingi stebėjimai su tuo pačiu spec_obj_ID turi turėti bendrą išvesties klasę, o visos šio požymio reikšmės yra skirtingos;
- „rerun_ID“ požymis, nes yra tik viena unikali reikšmė;
- „MJD“ požymis, nes ji simbolizuoja datą, kada užfiksuotas stebėjimas

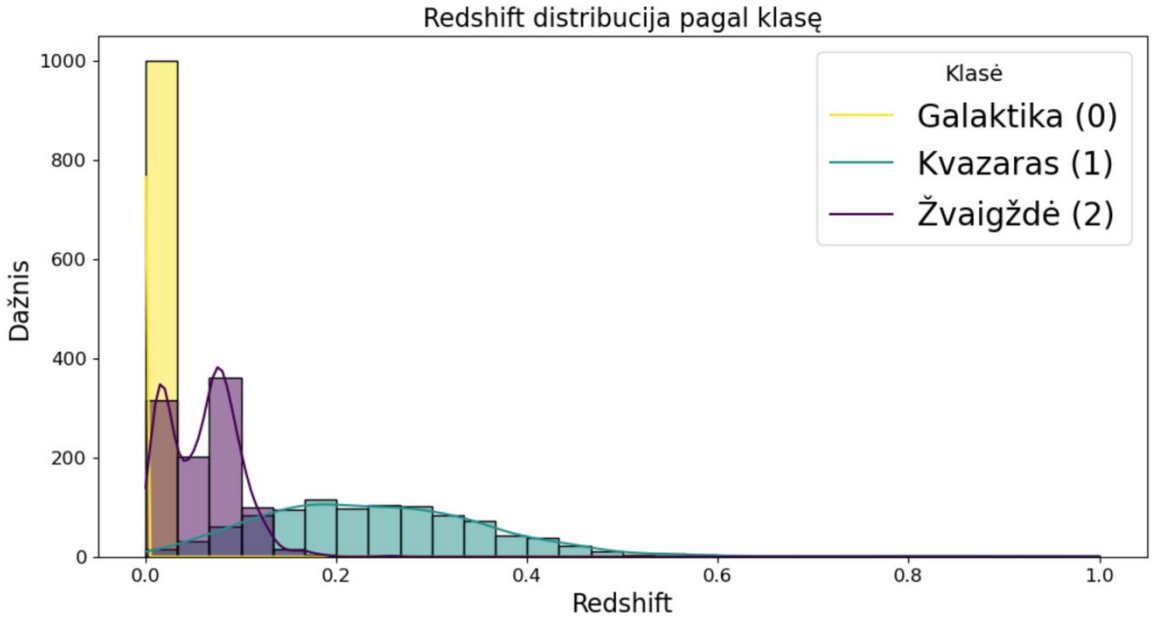
Duomenų aibė neturėjo praleistų reikšmių. Tolimesniems uždaviniams pasirinkome „redshift“, „u“, „g“, „r“, „i“ ir „z“ požymius.

Duomenų aibė turėjo vieną eilutę, kurioje „u“, „g“ ir „z“ reikšmės buvo -9999, tad šią triukšmo eilutę panaikinome (2.2 pav.).

„Class“ požymis yra kategorinis požymis, kuris turi tris unikalias reikšmes duomenų aibėje: GALAXY – galaktika, QSO – kvazaras(ypač šviesus objektas galaktikos centre), STAR – žvaigždė. Kiekviena šių reikšmių buvo pakeista atitinkamai į skaičius 0, 1, 2.



2.2 pav. Stačiakampė „u“, „g“, „z“ reikšmių diagrama.



2.3 pav. „redshift“ reikšmės pasiskirstymas pagal klases

Kitų reikšmių atsiskyrėlių nešalinome, nes jos neturėjo jokių didelių išskirčių - daugumoje jų matoma Gauso distribucija (3.2.2. poskyris), išskyrus „redshift“ reikšmėje (2.3 pav.), tačiau ši reikšmė rodo šviesos bangų ilgėjimą, todėl jos aukštų reikšmių naikinimas pakenktų analizės tikslumui. Taip pat, galima atkreipti dėmesį, kad objektų, priklausančių pirmai klasei (kvazarai), „redshift“ požymio reikšmės sąlyginai yra labai aukštos.

2.3. Objektų atrinkimas

Tolimesniam duomenų analizavimui, apdorojimui ir vizualizavimui buvo atsitiktinai atrinkti po 1000 objektų iš kiekvienos klasės (2.4 pav.):

```
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   u          3000 non-null   float64
 1   g          3000 non-null   float64
 2   r          3000 non-null   float64
 3   i          3000 non-null   float64
 4   z          3000 non-null   float64
 5   run_ID     3000 non-null   int64
 6   cam_col    3000 non-null   int64
 7   field_ID   3000 non-null   int64
 8   class      3000 non-null   int64
 9   redshift   3000 non-null   float64
 10  plate      3000 non-null   int64
 11  fiber_ID   3000 non-null   int64
dtypes: float64(6), int64(6)
```

2.4 pav. Duomenų aibė su pasirinktais objektais.

2.4. Duomenų aibės normavimas

Duomenų normavimui buvo parinkti šie požymiai: „redshift“, „u“, „g“, „r“, „i“ ir „z“. Kiti požymiai buvo nenormuoti, nes jie yra arba identifikaciniai („run_ID“, „field_ID“, „cam_col“, „plate“, ir „fiber_ID“) arba kategoriniai („class“). Duomenys buvo normuoti naudojant du metodus:

1. Vidurkio ir dispersijos normavimas (2.7 pav.);
2. Min-max normavimas (2.6 pav.).

	u	g	r	i	z	run_ID	cam_col	field_ID	class	redshift	plate	fiber_ID
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	21.764612	20.453767	19.696286	19.255546	18.979423	4463.033667	3.531000	183.694667	1.000000	0.715718	5324.207333	451.430000
std	2.148211	1.906341	1.808770	1.766683	1.780294	1973.851258	1.592655	144.563996	0.816633	0.927248	3010.616165	274.568411
min	14.646170	13.133170	12.760100	12.678570	12.625930	109.000000	1.000000	12.000000	0.000000	-0.003146	266.000000	1.000000
25%	20.211562	19.001137	18.355740	18.055810	17.765642	3051.250000	2.000000	82.000000	0.000000	0.000065	2682.750000	221.750000
50%	21.707205	20.853485	20.151955	19.574855	19.203925	4072.000000	4.000000	146.000000	1.000000	0.416217	5174.500000	438.000000
75%	23.193045	21.834385	21.060747	20.622673	20.312820	5415.000000	5.000000	239.250000	2.000000	1.136566	7696.250000	665.000000
max	28.723970	26.817360	27.334760	24.473360	23.204610	8162.000000	6.000000	980.000000	2.000000	7.010295	12547.000000	1000.000000

2.5 pav. Nenormuotos duomenų aibės statistika.

	u	g	r	i	z	run_ID	cam_col	field_ID	class	redshift	plate	fiber_ID
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	0.505650	0.534968	0.475907	0.557617	0.600594	4463.033667	3.531000	183.694667	1.000000	0.102498	5324.207333	451.430000
std	0.152596	0.139310	0.124104	0.149785	0.168291	1973.851258	1.592655	144.563996	0.816633	0.132210	3010.616165	274.568411
min	0.000000	0.000000	0.000000	0.000000	0.000000	109.000000	1.000000	12.000000	0.000000	0.000000	266.000000	1.000000
25%	0.395331	0.428814	0.383929	0.455900	0.485856	3051.250000	2.000000	82.000000	0.000000	0.000458	2682.750000	221.750000
50%	0.501572	0.564178	0.507172	0.584689	0.621816	4072.000000	4.000000	146.000000	1.000000	0.059794	5174.500000	438.000000
75%	0.607117	0.635859	0.569526	0.673526	0.726640	5415.000000	5.000000	239.250000	2.000000	0.162504	7696.250000	665.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	8162.000000	6.000000	980.000000	2.000000	1.000000	12547.000000	1000.000000

2.6 pav. Min-max metodu normuota duomenų aibės statistika.

	u	g	r	i	z	run_ID	cam_col	field_ID	class	redshift	plate	fiber_ID
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	0.000000	0.000000	0.000000	0.000000	-0.000000	4463.033667	3.531000	183.694667	1.000000	-0.000000	5324.207333	451.430000
std	1.000000	1.000000	1.000000	1.000000	1.000000	1973.851258	1.592655	144.563996	0.816633	1.000000	3010.616165	274.568411
min	-3.313661	-3.840129	-3.834752	-3.722782	-3.568789	109.000000	1.000000	12.000000	0.000000	-0.775267	266.000000	1.000000
25%	-0.722950	-0.761999	-0.741137	-0.679090	-0.681787	3051.250000	2.000000	82.000000	0.000000	-0.771804	2682.750000	221.750000
50%	-0.026723	0.209678	0.251922	0.180739	0.126104	4072.000000	4.000000	146.000000	1.000000	-0.323001	5174.500000	438.000000
75%	0.664941	0.724224	0.754359	0.773838	0.748976	5415.000000	5.000000	239.250000	2.000000	0.453867	7696.250000	665.000000
max	3.239607	3.338118	4.223021	2.953452	2.373309	8162.000000	6.000000	980.000000	2.000000	6.788451	12547.000000	1000.000000

2.7 pav. Vidurkio ir dispersijos metodu normuota duomenų aibės statistika.

3. DIMENSIJŲ MAŽINIMO METODAI IR VIZUALIZACIJA

3.1. Principinių komponenčių analizė (PCA)

3.1.1. Aprašymas

PCA („Principal Component Analysis“) yra tiesinis dimensijų mažinimo metodas, plačiai naudojamas duomenų analizėje ir vizualizacijoje. Skirtingai nuo netiesinių metodų, tokių kaip t-SNE ir UMAP, PCA siekia išlaikyti globalią duomenų struktūrą, sumažindama dimensijų skaičių taip, kad išsaugotų kuo daugiau duomenų pasiskirstymo.

Kadangi PCA yra tiesinis metodas, sumažintų dimensijų ašys turi aiškią interpretaciją pagal pradines savybes. Tai leidžia analizuoti, kurie pradinių požymių deriniai labiausiai prisideda prie duomenų variacijos ir kaip jie susiję su naujais komponentais.

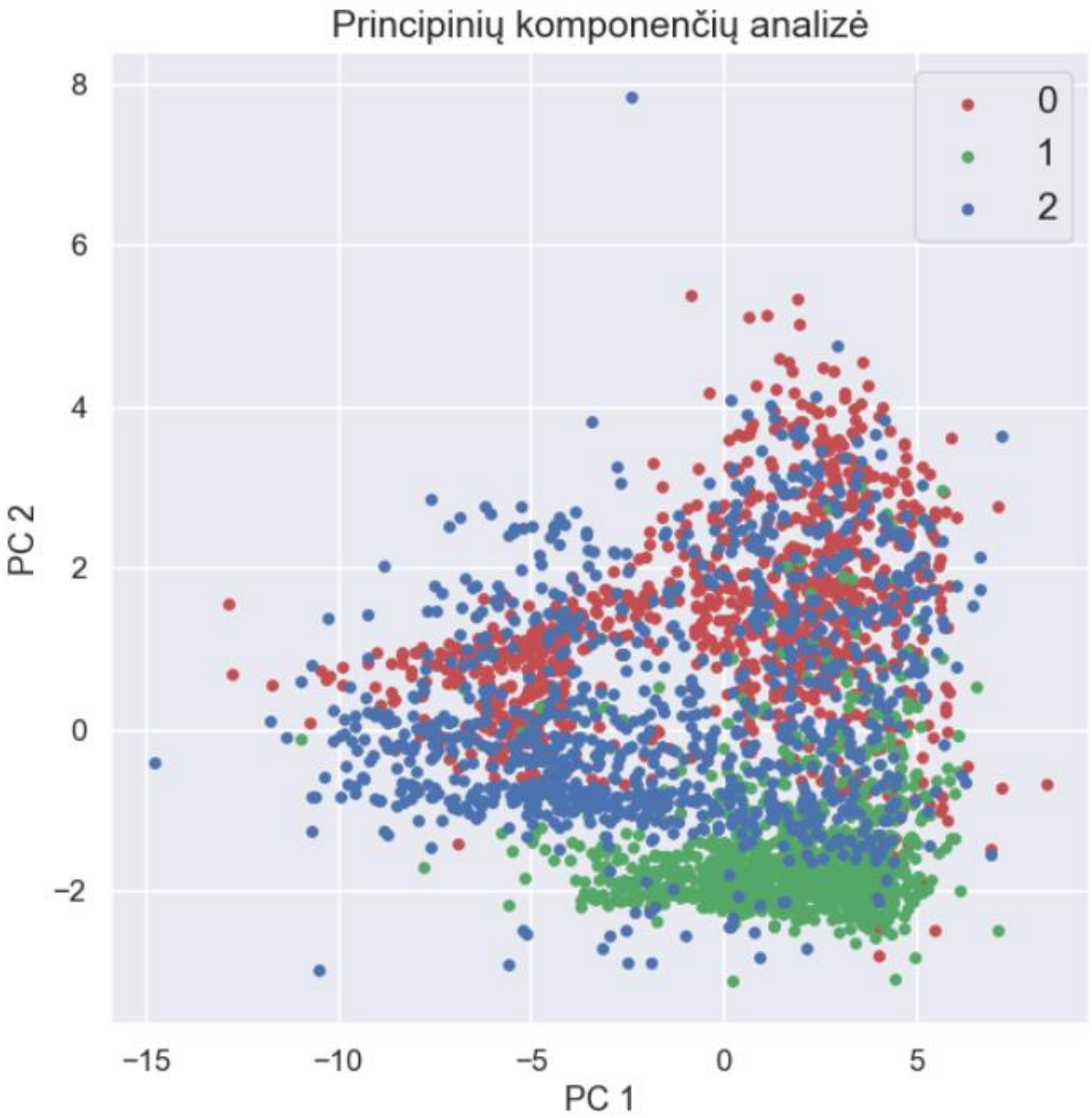
PCA metodui pasirinkome „u“, „g“, „r“, „i“, „z“ ir „redshift“ požymius. Toliau pateiktuose grafikuose matysime, kaip pirmosios dvi pagrindinės komponentės (PC1 ir PC2) atspindi duomenų struktūrą ir kokią variacijos dalį jos paaiškina.

3.1.2. Analizė

Kiekvienas taškas grafike atitinka vieną duomenų aibės objektą, o spalvos (raudona, žalia ir mėlyna) reprezentuoja skirtingas objektų klases: atitinkamai galaktikas, kvazarus ir žvaigždes. Grafike (3.2 pav.) matoma, kad klasės iš dalies persidengia, tačiau galima pastabėti objektų reprezentuojančių kvazarų klasę išsiskyrimą.

	u	g	r	i	z	redshift
PC1	0.449163	0.474751	0.460201	0.429496	0.409167	0.095887
PC2	0.743025	0.196483	-0.148082	-0.349596	-0.452531	-0.245721

3.1 pav. “u”, “g”, “r”, “I”, “z”, “redshift” reikšmių įtaka PCA rezultatų ašims.



3.2 pav. dimensijų mažinimas PCA metodu.

Kvazarų klasė (3.2 pav. legendoje žalia spalva) daugiausiai išsiskiria Y ašyje dėl „redshift“ reikšmės. Nors ji turi gan nedidelę neigiamą įtaką (-0.245721), objektų priklausančių kvazarų klasei „redshift“ požymio reikšmės yra didelės, lyginant su kitomis klasėmis (2.3 pav.) - tai matoma PC2 ašyje.

PC1 turi didžiausią apkrovą iš g (0.474751), r (0.460201), u (0.449163) ir i (0.429496) požymių (3.1 pav.), o „redshift“ (raudonasis poslinkis) turi mažiausią įtaką (0.095887) šiam komponentui. Tai reiškia, kad šie požymiai yra stipriausiai susiję su pagrindine duomenų aibės variacija, kurią aprašo PC1.

PC2 rodo didžiausią teigiamą apkrovą „u“ požymiui (0.743025), tačiau „z“ (-0.452531) ir „i“ (-0.349596) požymiai turi didelę neigiamą apkrovą.

Pirmoji pagrindinė komponentė (PC1) paaiškina 79.5% visos duomenų aibės variacijos, o antroji komponentė (PC2) – papildomus 14.8%. Tai reiškia, kad šie du komponentai bendrai paaiškina apie 94.3% visos variacijos, kas yra pakankamai daug, kad būtų galima sumažintą duomenų aibės erdvę naudoti vizualizacijai ir tolimesnei analizei.

3.1.3. PCA išvados

Skirtingos duomenų klasės—galaktikos, kvazarai ir žvaigždės—iš dalies atsiskiria PCA vizualizacijoje. Pirmieji pagrindiniai komponentai, sudaryti iš požymių „z“, „i“, „r“, „g“ ir „u“, paaiškina 94,3% duomenų variacijos. Vizualizacijoje pastebėjome objektų grupių persidengimą, priklausančių 0 ir 2 klasėms, tačiau objektų, priklausančių 1 klasei, grupė atsiskiria pakankamai aiškiai.

Tačiau, atsižvelgiant į tai, kad „redshift“ požymio pasiskirstymas tarp klasių yra labai skirtingas, o PCA metodas jam nesuteikė didelės reikšmės, negalime visiškai pasitikėti vien PCA metodo rodomais rezultatais. Tai rodo, kad PCA gali praleisti svarbią informaciją, susijusią su „redshift“ požymiu, ir gali būti nepakankamas šių duomenų analizės metodas.

3.2. t-SNE metodas

3.2.1. Aprašymas

t-SNE („t-Distributed Stochastic Neighbor Embedding“) yra vizualizacijos technika, naudojama dimensijų mažinimui. Skirtingai nuo tiesinių metodų, tokių kaip PCA, kurie bando išlaikyti globalią struktūrą, t-SNE daugiausia dėmesio skiria lokalių struktūrų išsaugojimui – tai reiškia, kad jis stengiasi išlaikyti artimiausius taškus kartu net ir sumažintoje dimensijoje, todėl ši technika puikiai tinka grupių atskleidimui duomenyse.

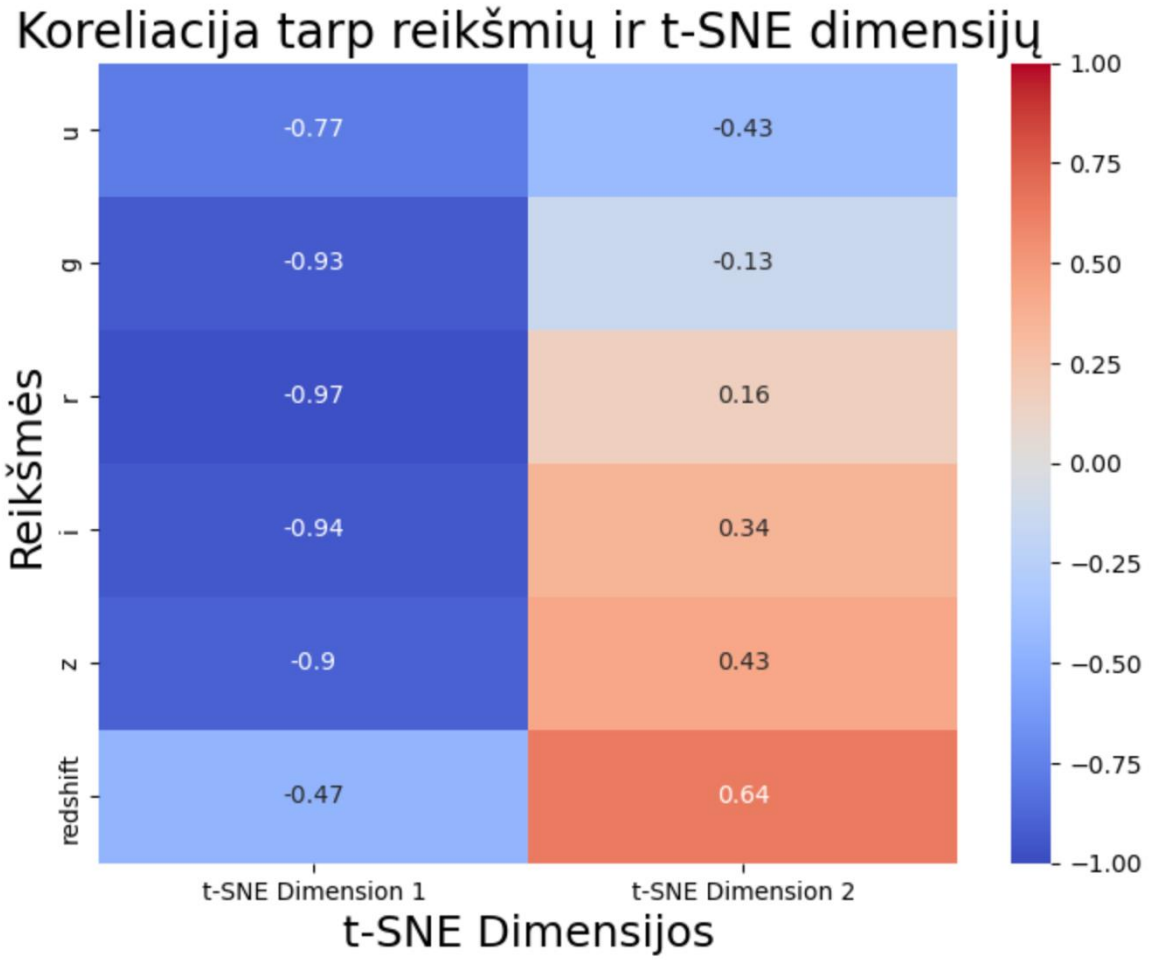
Kadangi t-SNE yra netiesinis metodas, sumažintų dimensijų ašys neturi aiškios interpretacijos pagal pradines savybes. Norint geriau suprasti, kurie pradinių duomenų požymiai gali daryti įtaką t-SNE rezultatams, galima atlikti koreliacijos analizę tarp pradinių objektų savybių ir t-SNE dimensijų. t-SNE metodui buvo parinkti „u“, „g“, „r“, „i“, „z“, „redshift“ požymiai ir jis taikytas normuotiem pagal „min-max“ metodą duomenim (nebent nurodyta kitaip).

Kuriant t-SNE modelius, yra svarbūs du pagrindiniai parametrai: „perplexity“ ir „n_iter“. Toliau pateiktuose grafikuose bus matoma, kokią įtaką grafikams daro skirtingos šių parametrų reikšmės.

- „perplexity“ parametras nurodo kiekvieno taško kaimynų skaičių.
- „n_iter“ parametras nustato iteracijų skaičių, per kurias algoritmas optimizuoja taškų padėtis sumažintoje erdvėje.

3.2.2. Branduolinių tankių ir koreliacijų grafikai

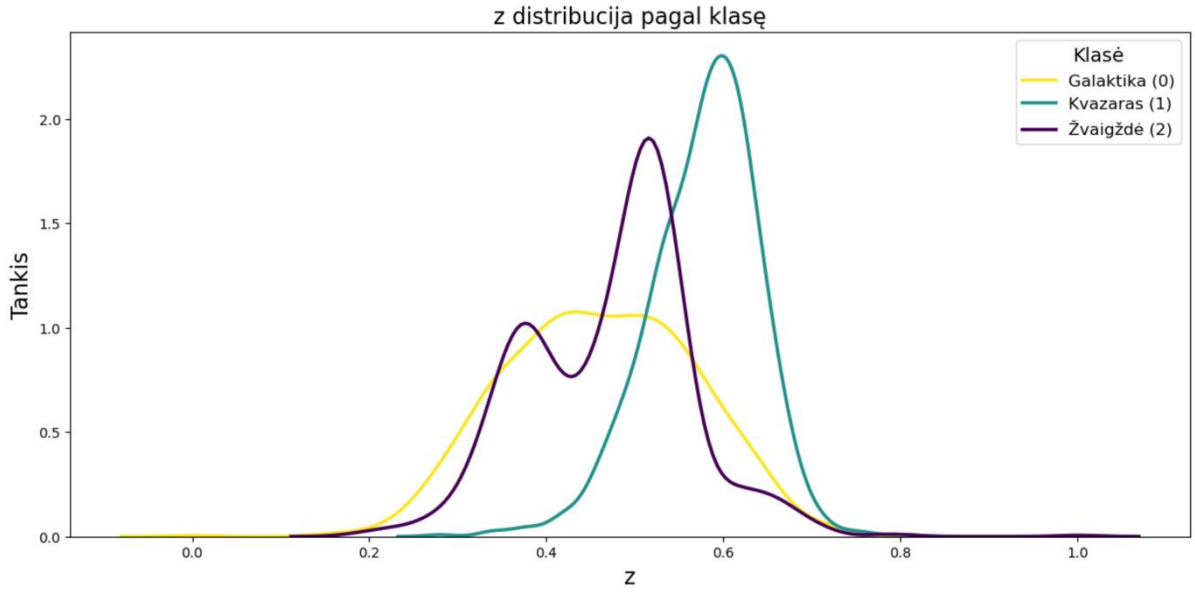
Žemiau pateiktas koreliacijų grafikas ir „u“, „g“, „r“, „i“ bei „z“ reikšmių branduolinio tankio grafikai padės suprasti 3.2.3 punkte aprašomą dimensijų mažinimą t-SNE metodu.



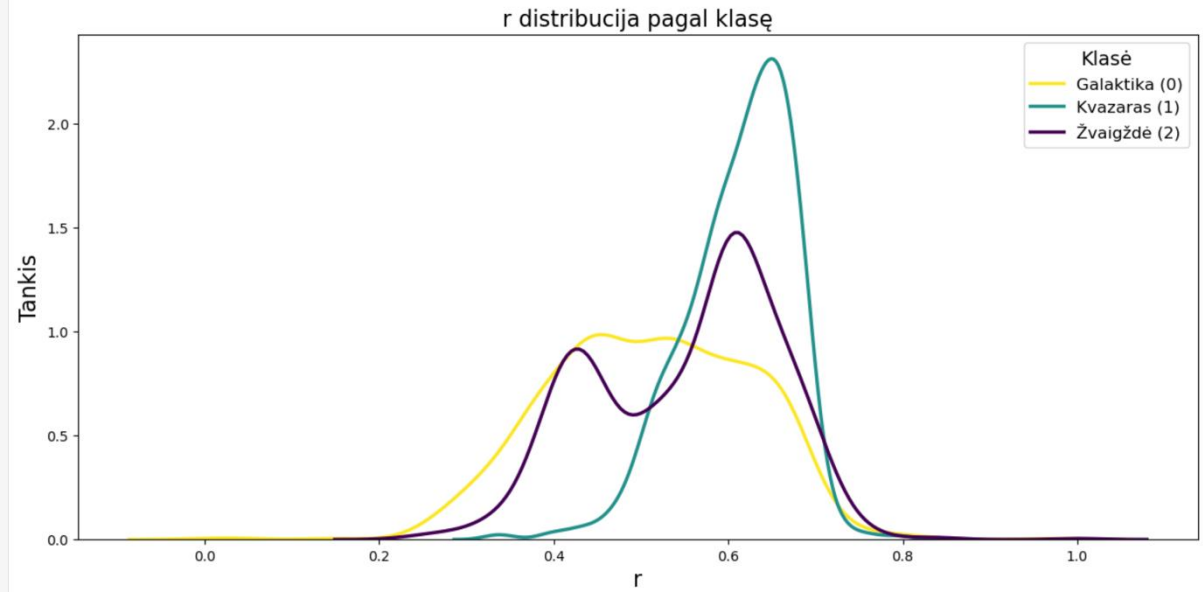
3.3 pav. Koreliacija tarp „redshift“, „u“, „g“, „r“, „z“ reikšmių ir t-SNE dimensijų.

Koreliacijų grafikas (3.3 pav.) atvaizduoja t-SNE metodu sumažintų dimensijų grafiko ašių „t-SNE Dimension 1” ir „t-SNE Dimension 2“ koreliacijas su „redshift“, „u“, „g“, „r“, ‚i“ ir „z“ vertėmis.

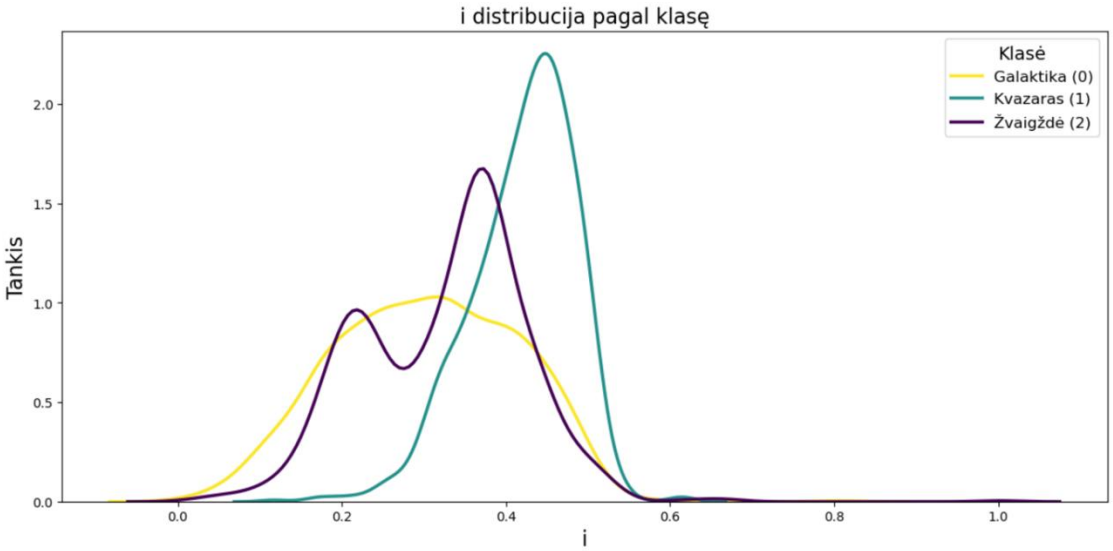
Sekantys 5 grafikai atvaizduoja normuotų šviesos spektro duomenų histogramas (3.4 pav. – 3.8 pav.). Toliau analizuojant t-SNE dimensijų mažinimo metodą remsimės šiais grafikais.



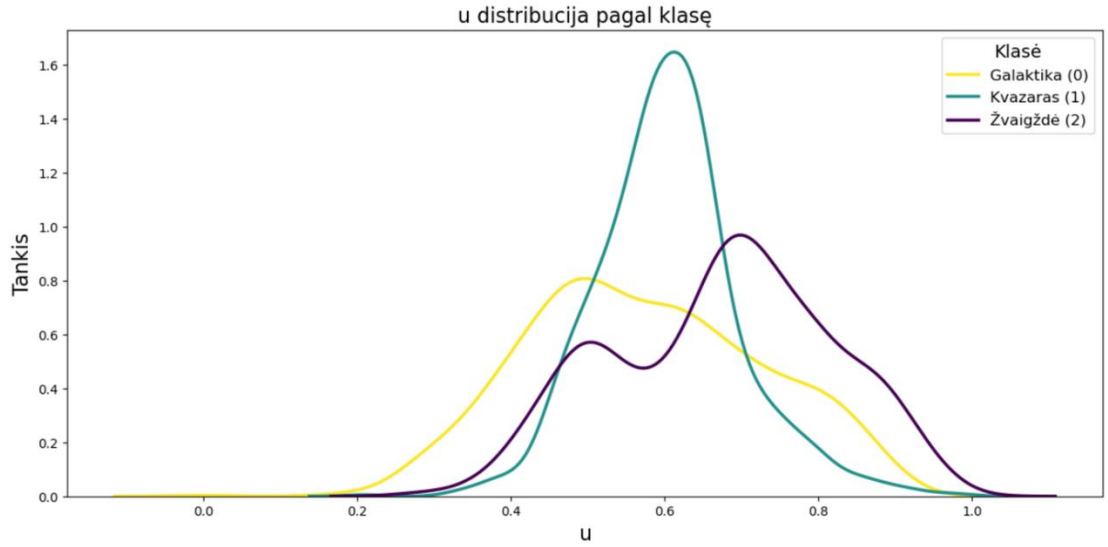
3.4 pav. „z“ reikšmės distribucija pagal klasę.



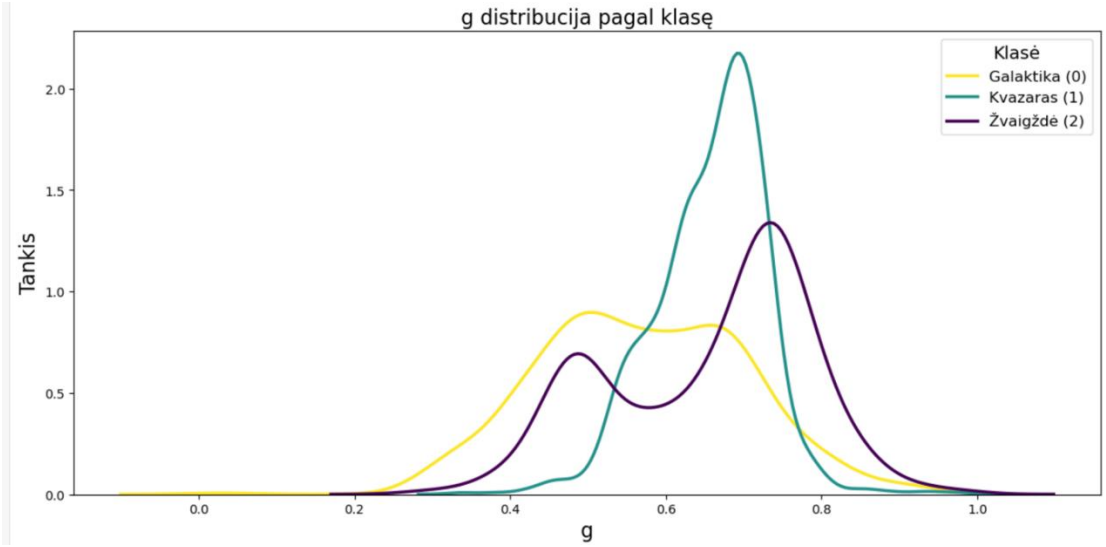
3.5 pav. „r“ reikšmės distribucija pagal klasę.



3.6 pav. „i“ reikšmės distribucija pagal klasę.

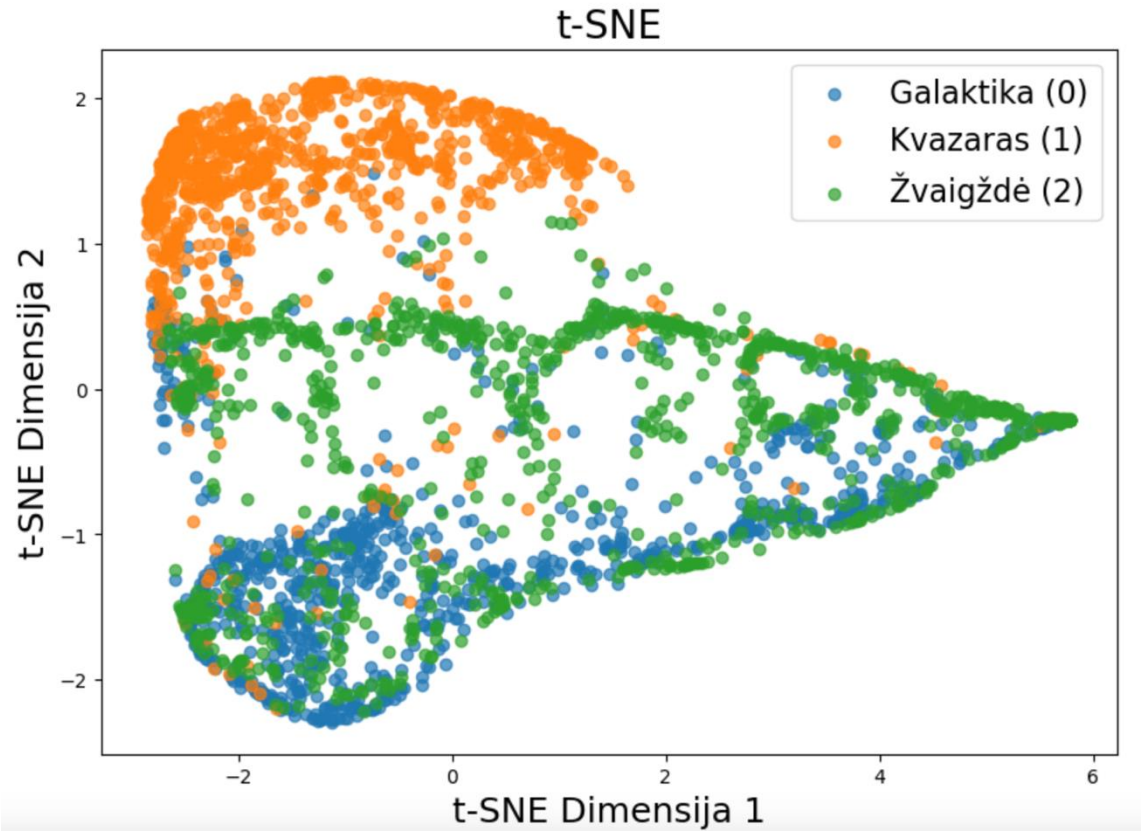


3.7 pav. „u“ reikšmės distribucija pagal klasę.



3.8 pav. „g“ reikšmės distribucija pagal klasę.

3.2.3. t-SNE nenormuotų duomenų grafikas



3.9 pav. t-SNE metodu 5 dimensijų, sumažintų į 2, grafikas.

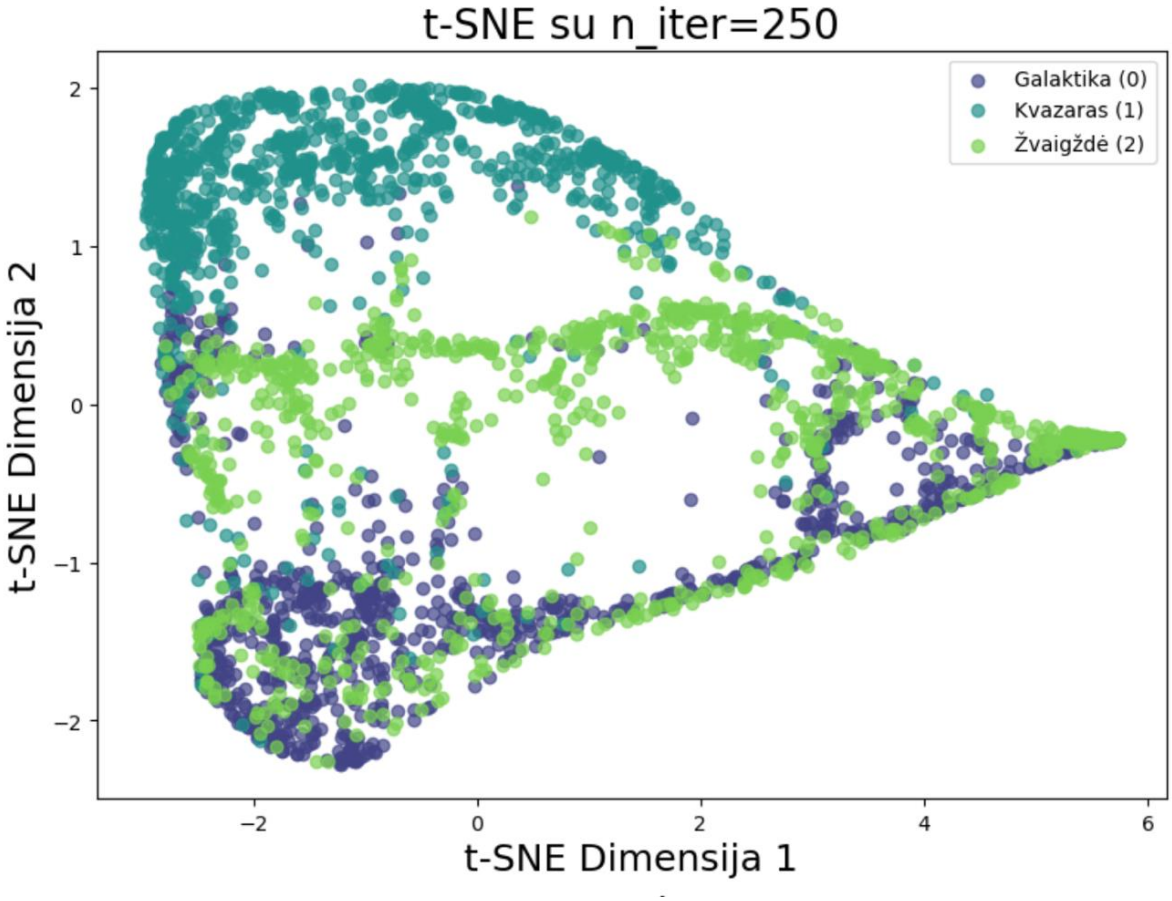
Pradiniame t-SNE grafike, kuriame požymiai nebuvo normuoti (3.9 pav.), yra matomos dalinai išsiskiriančios struktūros. Labiausiai išsiskiria kvazarai, pagrinde turintys aukštas 2 dimensijos ir žemas 1 dimensijos reikšmes.

Galima pastebėti, kad šiai objektų grupei įtaką daro „redshift“ reikšmė. Tai matosi koreliacijų grafike (3.3 pav.) – 1 dimensija su „redshift“ reikšme koreliuoja neigiamai (-0.47), o 2 dimensija koreliuoja teigiamai (0.64). Šie skaičiai stipriai paveikia objektų grupės, priklausančių kvazarų klasei, poziciją, nes „redshift“ reikšmė šioje kategorijoje yra išsidėsčiusi plačiame intervale – didžioji dalis reikšmių yra nuo 0 iki 0.4, o mažytė dalis reikšmių siekia 0.7 (2.3 pav.).

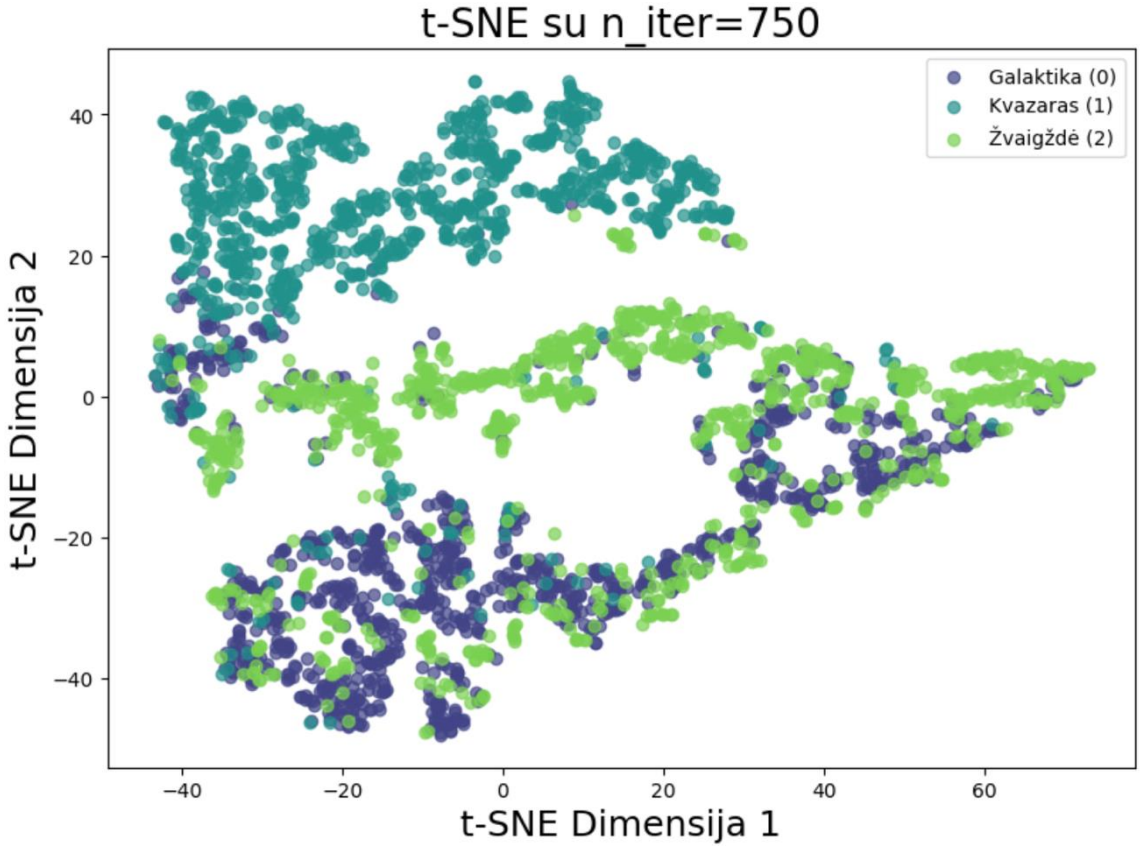
Taip pat galima pastebėti platų objektų, priklausančių žvaigždžių klasei (2 kategorija legendoje), išsidėstymą abiejose dimensijose. Žvelgiant į distribucijų pagal klasę grafikus (3.4 pav. – 3.8 pav.) matosi, jog žvaigždžių „u“, „g“, „r“, „i“ ir „z“ spektrai yra išsidėstę plačiausiai – [0.3; 1], [0.3, 1], [0.3; 0.8], [0.1; 0.5] ir [0.2; 0.7] intervaluose atitinkamai, tai gali paaiškinti tokį platų grupių formavimasi 1 dimensijoje.

3.2.4. „n_iter“ ir „perplexity“. Normuotų duomenų rezultatai

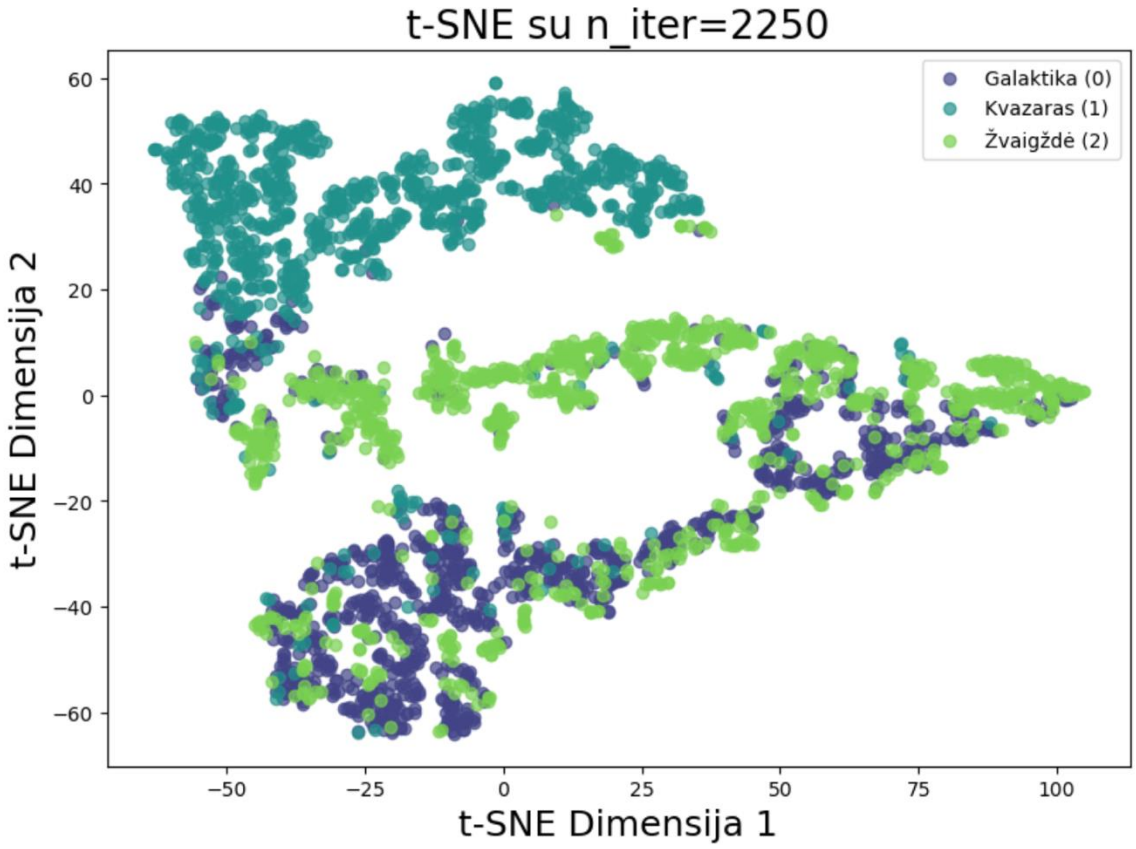
Žemiau pateikti grafikai rodo t-SNE algoritmo vizualizacijas su skirtingomis parametro „n_iter“ reikšmėmis. Šis parametras rodo atliekamų iteracijų kiekį. Galima pastebėti, kad didesnis iteracijų kiekis labiau atskiria objektų grupes. Tai matoma žemiau pateiktuose grafikuose – kai „n_iter“ reikšmė lygi 250, grafike (3.10 pav.) matomi didesni atstumai tarp objektų ir didesnis objektų grupių persidengimas, kai „n_iter“ reikšmė lygi 750 (3.11 pav.), objektų klasės yra labiau grupuotos. Atitinkamai dar didesnė „n_iter“ reikšmė lygi 2250 (3.12 pav.) objektų klases grupuoja dar labiau, nors skirtumas nebėra toks didelis. Taigi, didesnės „n_iter“ parametro reikšmės labiau atskirs objektus į skirtingas grupes.



3.10 pav. Dimensijų mažinimas t-SNE metodu. „n_iter“ reikšmė 250.



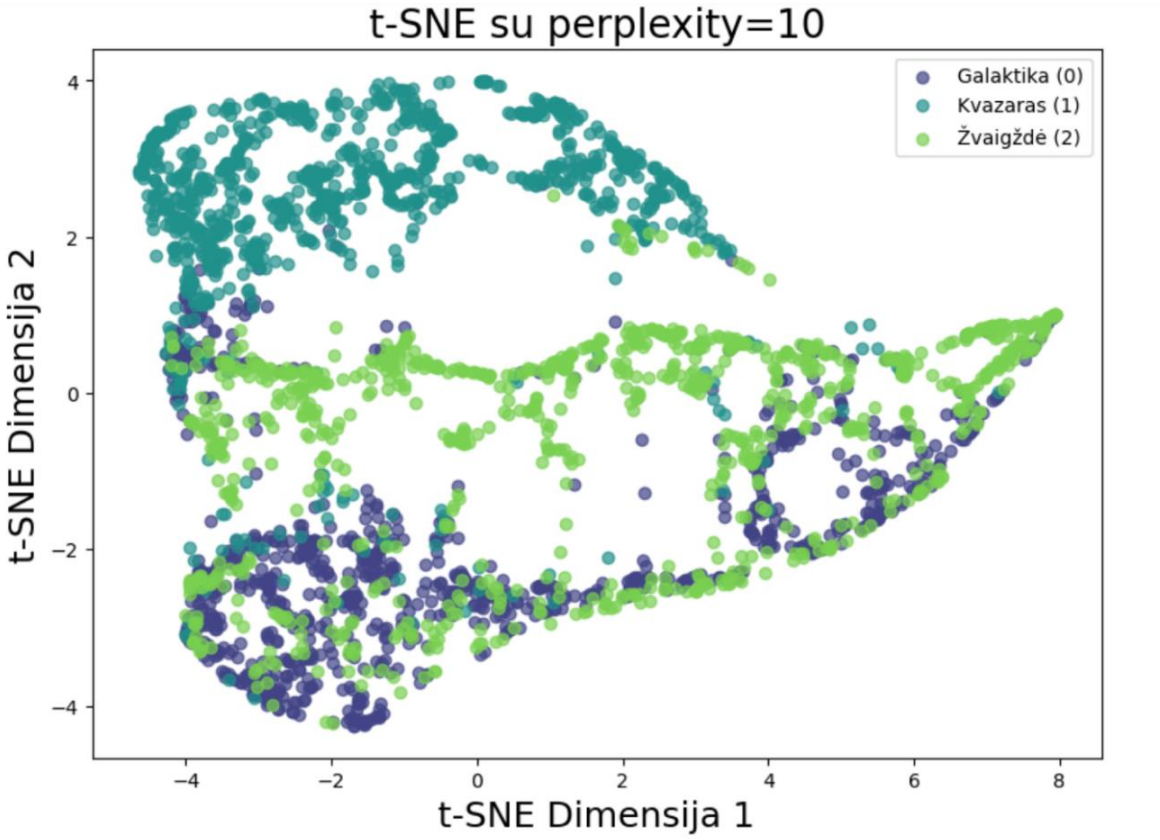
3.11 pav. Dimensijų mažinimas t-SNE metodu. „n_iter“ reikšmė 750.



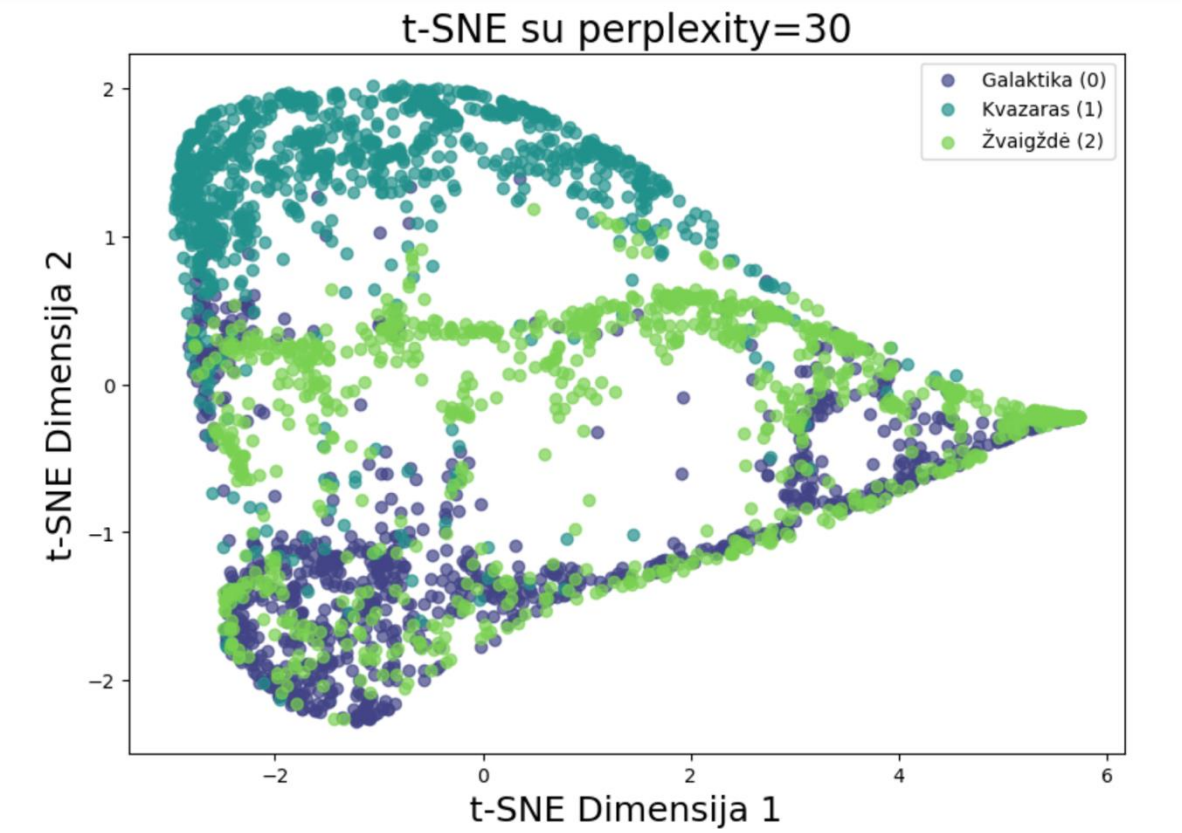
3.12 pav. Dimensijų mažinimas t-SNE metodu. „n_iter“ reikšmė 2250.

Žemiau pateikti grafikai rodo t-SNE algoritmo vizualizacijas su skirtingomis parametro „perplexity“ reikšmėmis. Šis parametras yra itin svarbus t-SNE, nes jis nurodo objektų artimiausių kaimynų kiekį, į kurį algoritmas turėtų atsižvelgti, siekiant nustatyti objektų panašumus. Mažesnė „perplexity“ reikšmė orientuojasi į lokalią duomenų struktūrą, todėl labiau išryškina smulkesnes, vietines objektų grupes (3.13 pav.). Tuo tarpu didesnės „perplexity“ reikšmės, lygios 30 (3.14 pav.) ir 90 (3.15 pav.), leidžia algoritmui atsižvelgti į platesnį kontekstą, įtraukiant ir tolimesnius taškus. Tokiu būdu gaunama vizualizacija su labiau apjungtomis objektų grupėmis, kurios atspindi bendresnį duomenų modelį.

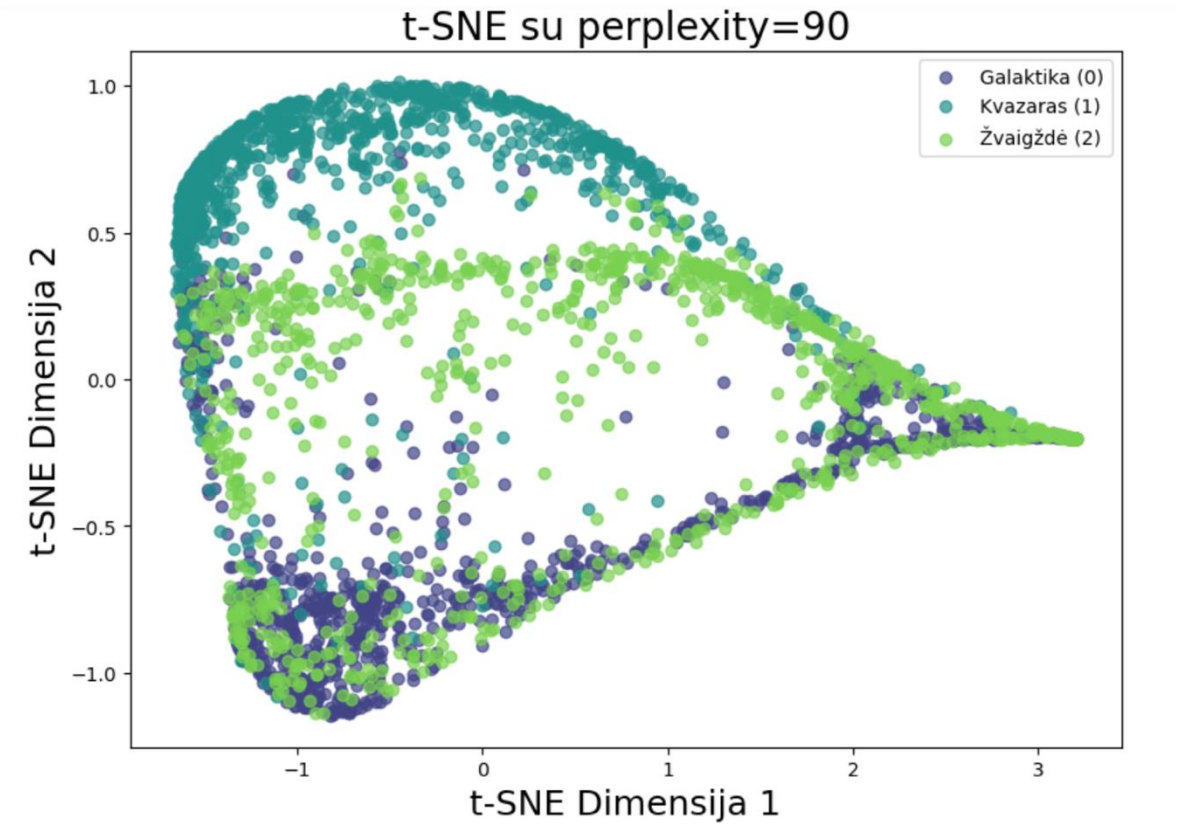
Taigi, „perplexity“ parametras turi didelę įtaką galutinei vizualizacijai – mažesnės reikšmės atsižvelgia į vietines struktūras, o didesnės apjungia atsižvelgia į šiek tiek platesnes struktūras.



3.13 pav. Dimensijų mažinimas t-SNE metodu. „perplexity“ reikšmė 10.



3.14 pav. Dimensijų mažinimas t-SNE metodu. „perplexity“ reikšmė 30.



3.15 pav. Dimensijų mažinimas t-SNE metodu. „perplexity“ reikšmė 90.

3.2.5. t-SNE Išvados

Skirtingos duomenų klasės (galaktikos, kvazarai, žvaigždės) atsiskiria t-SNE vizualizacijoje. Nors objektų, priklausančių galaktikų ir žvaigždžių klasėms, grupės vietomis persidengia, galime gana aiškiai atskirti kvazarų klasės objektų grupę.

t-SNE parametrų „n_iter“ ir „perplexity“ keitimas daro didelę įtaką vizualizacijos rezultatams. Grafikuose matoma, kad didesnė „n_iter“ reikšmė suteikia daugiau iteracijų, leidžiančių algoritmui stabiliau optimizuoti taškų pozicijas, kas lemia aiškesnį grupių išsidėstymą. „Perplexity“ keitimas leidžia reguliuoti, kiek algoritmas atsižvelgia į vietines struktūras (mažesnis „perplexity“) arba šiek tiek platesnes struktūras (didesnis „perplexity“), tačiau t-SNE metodas iš esmės yra labiau orientuotas į vietinių struktūrų išryškinimą.

Geriausiai objektus sugrupuojančio t-SNE metodo parametrai yra, kai „perplexity“ lygus 90, o kiti parametrai nekeičiami. Tačiau dalis antros ir pirmos klasės grupių persidengia, tad nerekomenduojama naudotis vien šiuo metodu.

3.3. UMAP metodas

3.3.1. Aprašymas

UMAP („Uniform Manifold Approximation and Projection“) yra vizualizacijos technika, naudojama dimensijų mažinimui, ypač efektyvi grupių atskleidimui duomenyse. Panašiai kaip t-SNE, UMAP yra netiesinis metodas, kuris daugiausia dėmesio skiria lokalių struktūrų išsaugojimui. UMAP taip pat atsižvelgia į globalią struktūrą, todėl jis gali išsaugoti daugiau informacijos apie bendrą duomenų pasiskirstymą lyginant su t-SNE metodu, tuo pačiu efektyviai vizualizacijoje rodant grupes.

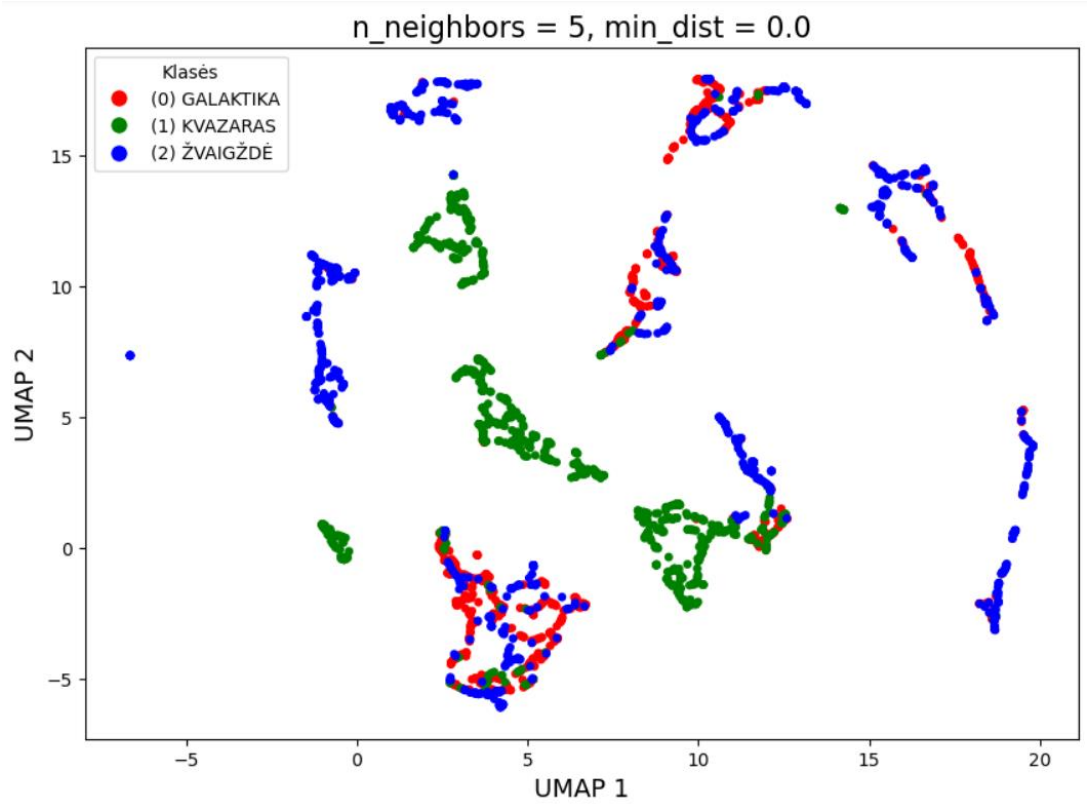
Kadangi UMAP yra netiesinis metodas, norint geriau suprasti, kurie pradinių duomenų bruožai gali daryti įtaką UMAP rezultatams, galima atlikti koreliacijos analizę tarp pradinių objektų savybių ir UMAP dimensijų.

Kuriant UMAP modelius išskiriami trys pagrindiniai parametrai: „n_neighbors“, „min_dist“ ir „metric“. Toliau pateiktuose grafikuose matysime kokią įtaką grafikui daro skirtingos „n_neighbors“, „min_dist“ ir „metric“ reikšmės.

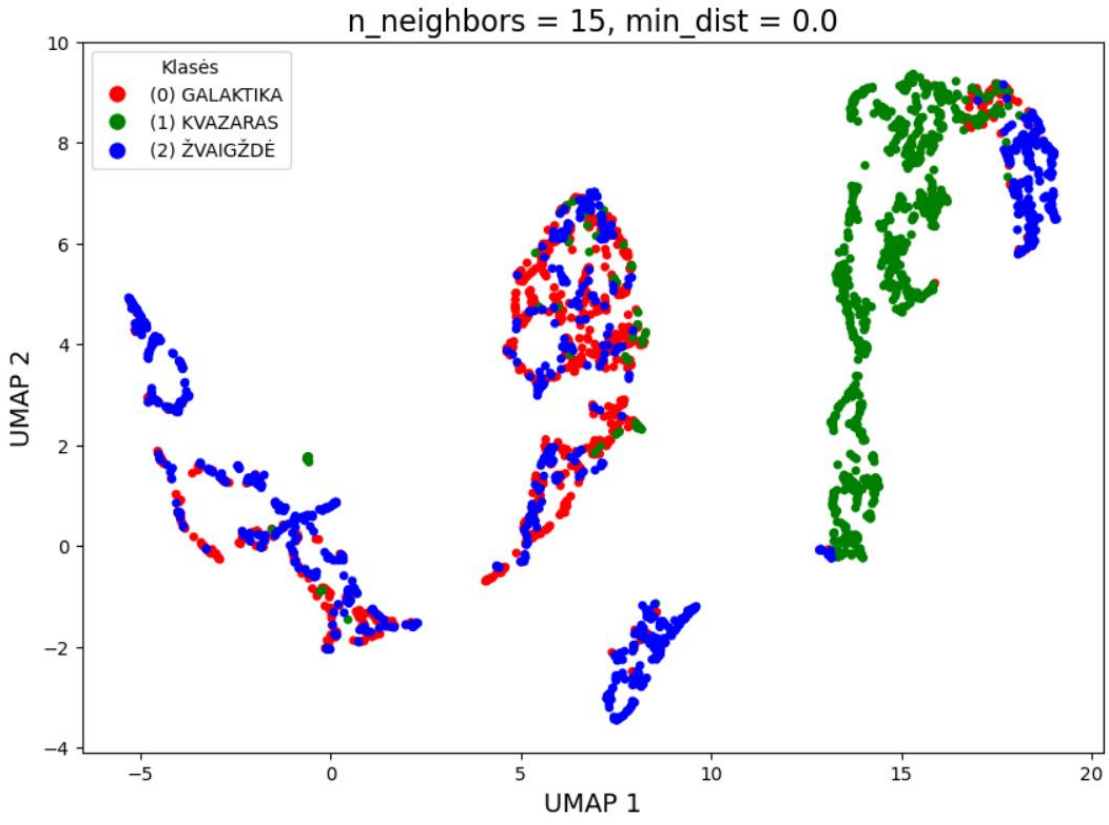
- „n_neighbors“ parametras nurodo, kiek artimiausių kaimynų yra laikoma kiekvieno taško vietinės struktūros apibrėžimui aukštesnėje dimensijoje.
- „min_dist“ parametras nustato minimalią leistiną atstumą tarp taškų sumažintoje dimensijoje. „min_dist“ kontroliuoja, kaip glaudžiai ar laisvai išdėstyti taškai.
- „metric“ parametras nurodo funkciją, kurią UMAP metodas naudoja skaičiuodamas atstumus tarp taškų aukštos dimensijos erdvėje. Tai turi įtakos tam, kaip algoritmas supranta taškų panašumus ir formuoja vietinę bei globalią struktūrą.

UMAP metodas buvo taikytas normuotiem pagal „Z-score“ duomenims ir pasirinkti „u“, „g“, „r“, „i“, „z“, „redshift“ požymiai. Grafikai sugrupuoti po tris į tris dalis: kai „min_dist“ parametras lygus 0 (3.3.2.), kai „min_dist“ parametras lygus 0,5 (3.3.3.) ir kai „min_dist“ parametras lygus 1 (3.3.1.). Kiekvienas iš grafikų su skirtingu „min_dist“ parametru, turi „n_neighbors“ parametrą: 5, 15 ir 50.

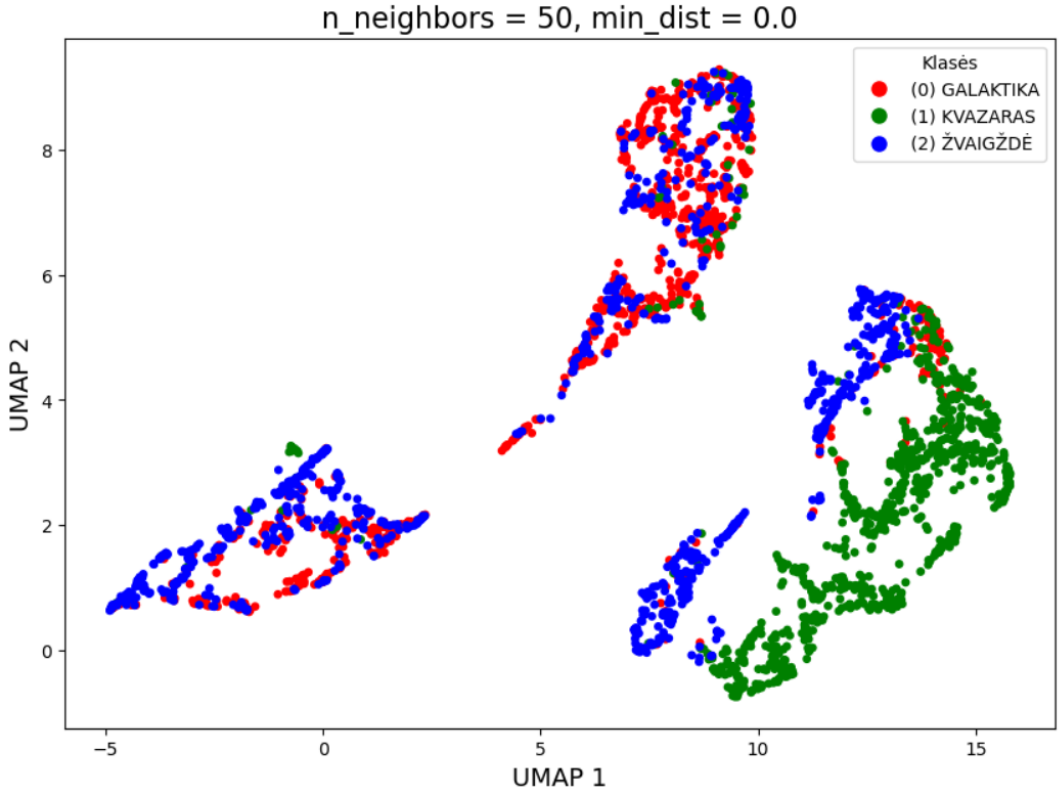
3.3.2. Grafikai, kai „min_dist“ vertė lygi 0



3.16 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 0, n_neighbors vertės keitimas.

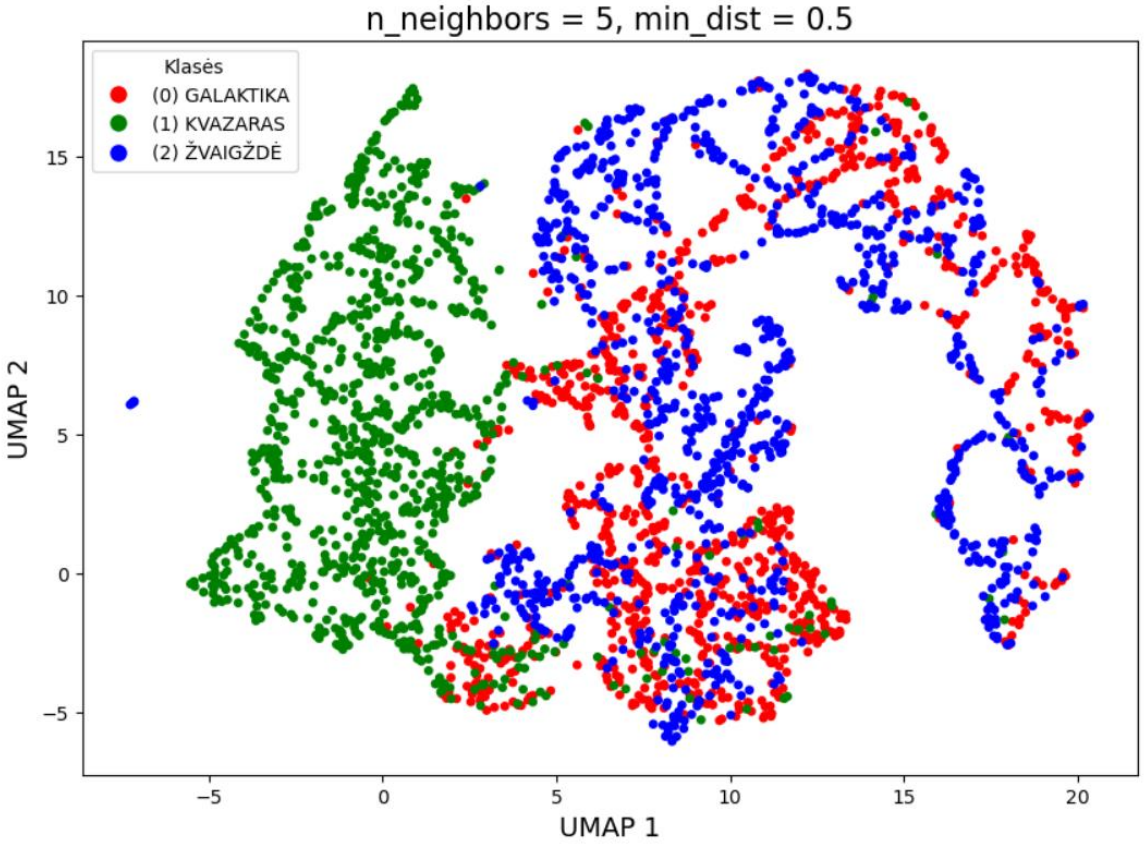


3.17 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 0, n_neighbors vertės keitimas.

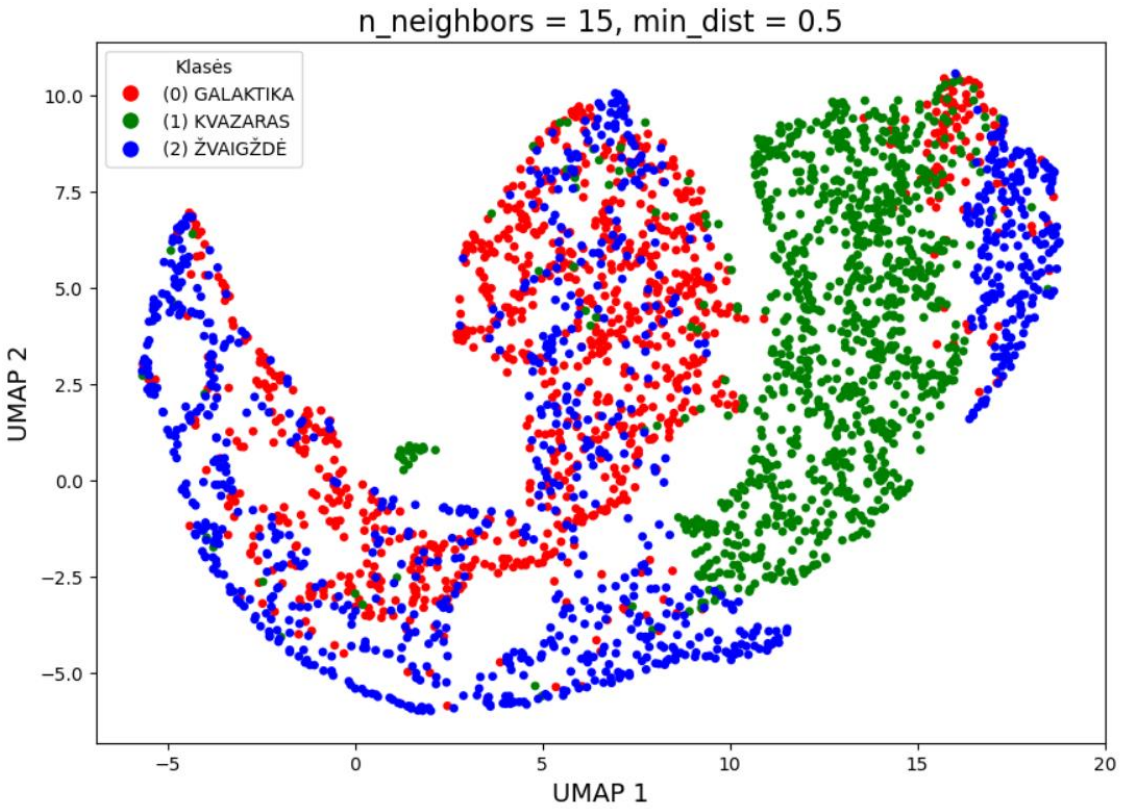


3.18 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 0, n_neighbors vertės keitimas.

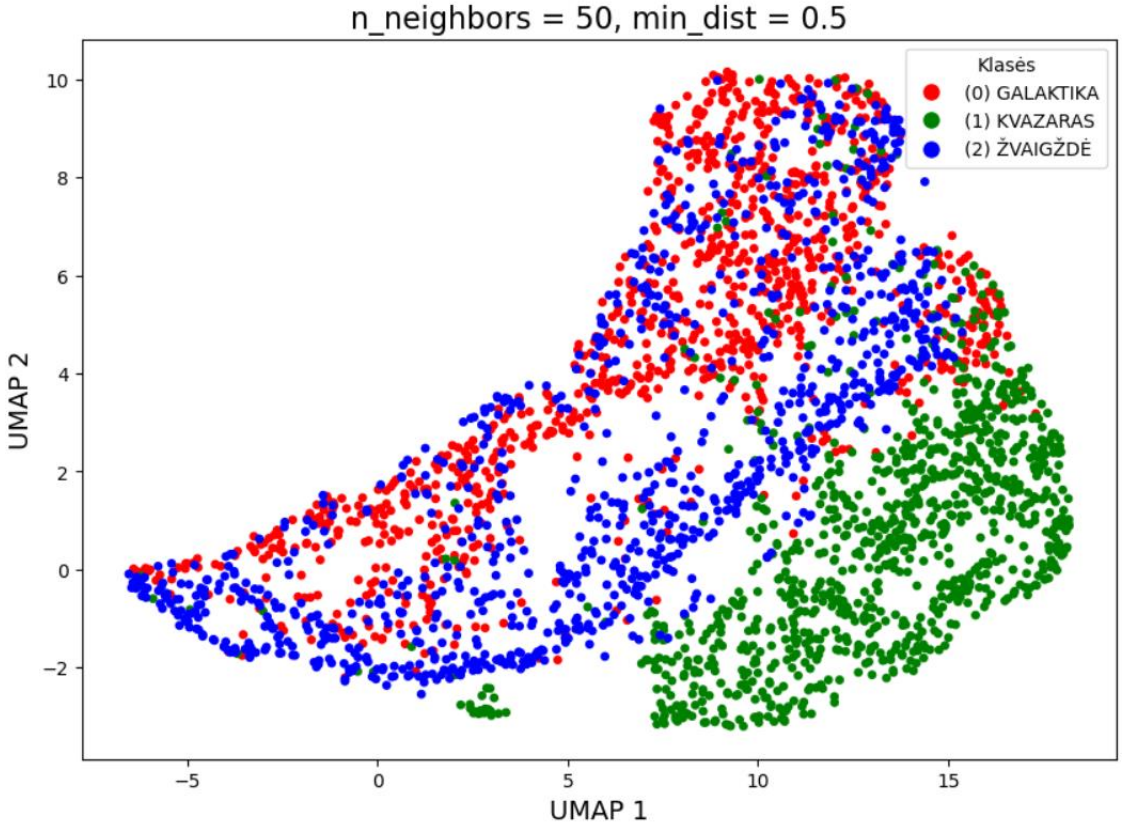
3.3.3. Grafikai, kai „min_dist“ vertė lygi 0.5



3.19 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 0.5, „n_neighbors“ vertės keitimas.

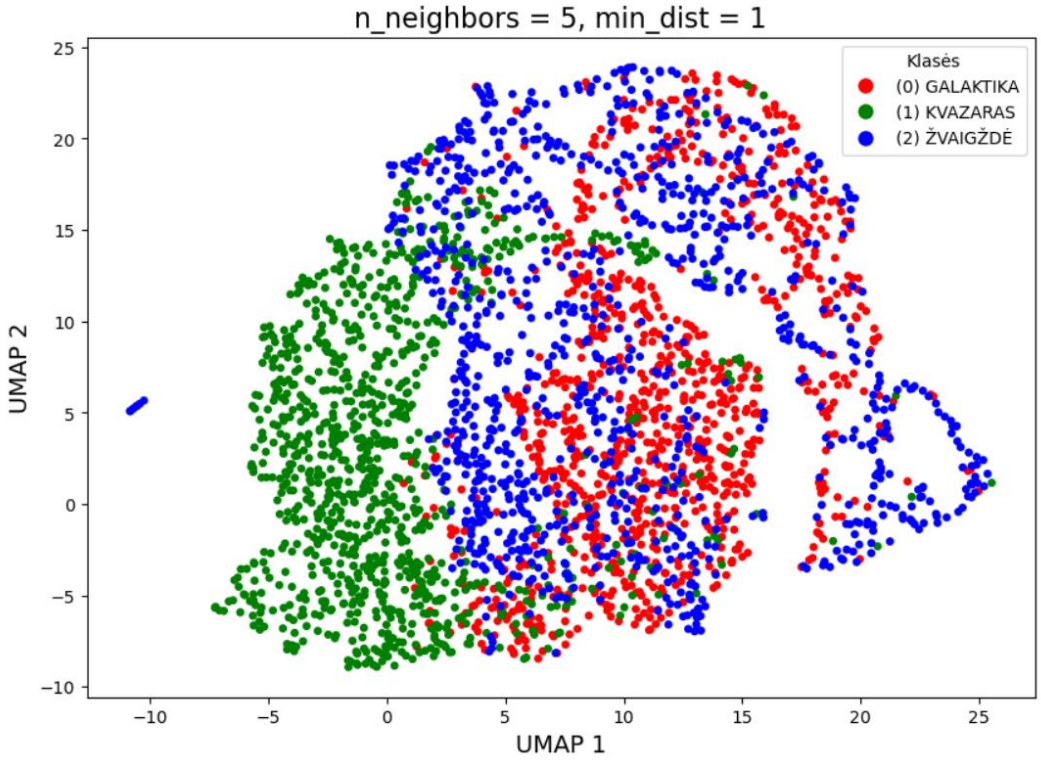


3.20 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 0.5, „n_neighbors“ vertės keitimas.

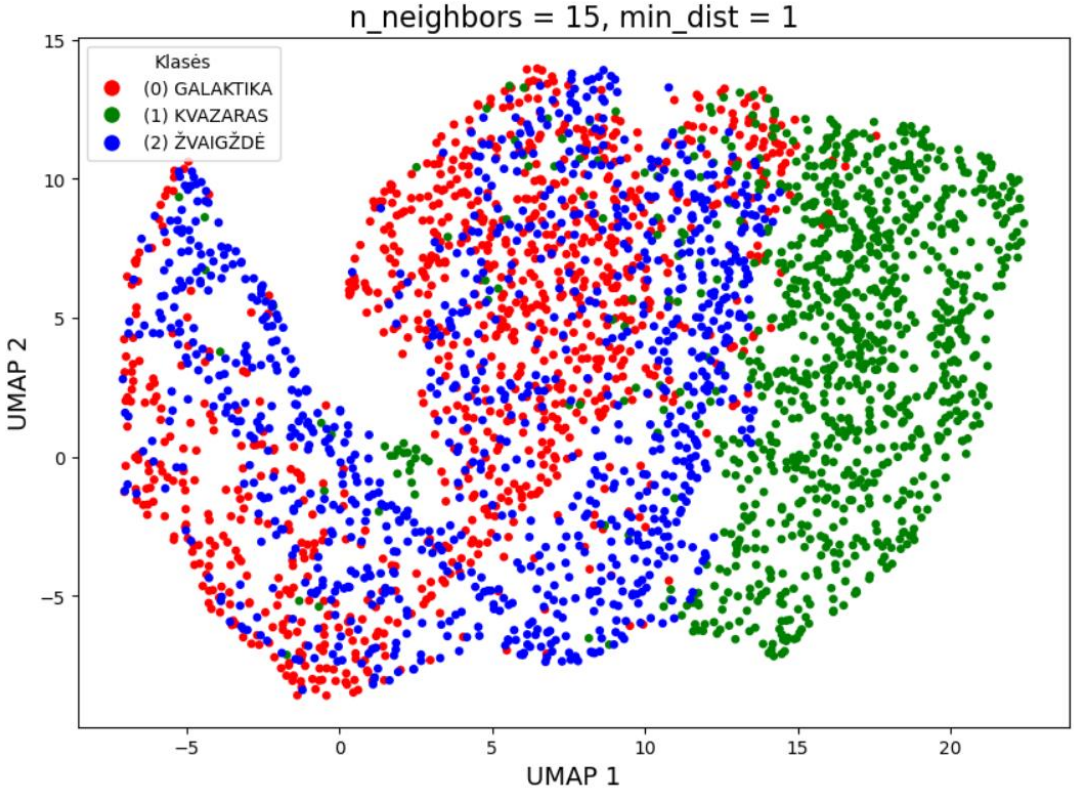


3.21 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 0.5, „n_neighbors“ vertės keitimas.

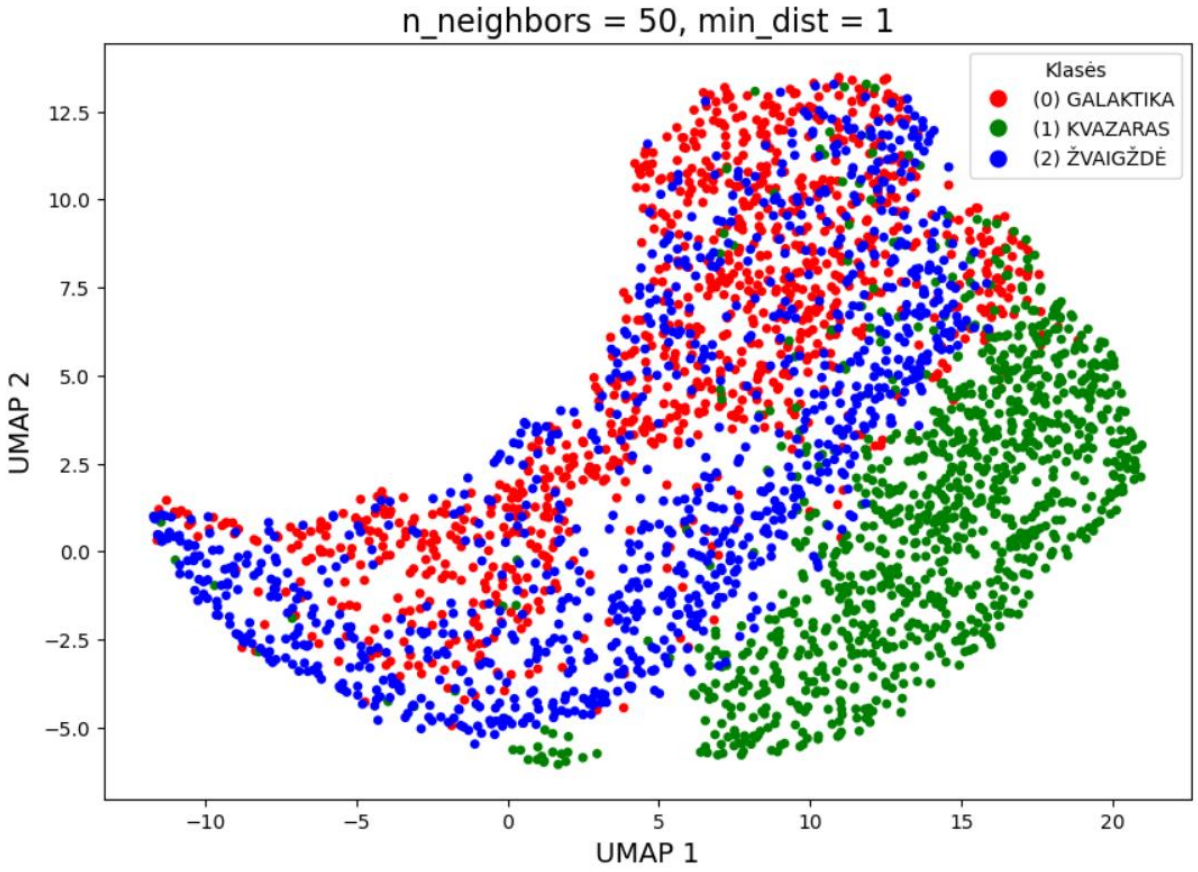
3.3.4. Grafikai, kai „min_dist“ vertė lygi 1



3.22 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 1, n_neighbors vertės keitimas.



3.23 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 1, n_neighbors vertės keitimas.



3.24 pav. Dimensijų mažinimas UMAP metodu. „min_dist“ vertė 1, n_neighbors vertės keitimas.

Šiuose grafikuose aiškiai matosi grafikų pokytis keičiantis „n_neighbors“ ir „min_dist“ parametrams.

Ryškiausi „n_neighbors“ parametro pokyčio rezultatai matosi, kai „min_dist“ parametras lygus 0 (3.3.2. dalis). Mažesnės „n_neighbors“ vertės (3.16 pav.) orientuojasi į lokalią struktūrą ir visus objektus sutraukia į mažesnes grupes. Didesnės „n_neighbors“ vertės (3.17 pav. ir 3.18 pav.) apima daugiau objektų, daugiau dėmesio skiriama globaliam duomenų struktūros vaizdui, išryškinamos platesnės grupės ir globali struktūra.

Parametro „min_dist“ verčių skirtumus galime matyti visuose poskyriuose (3.3.2, 3.3.3 ir 3.3.4 poskyriai), atitinkamuose „n_neighbors“ pokyčių grafikuose. Šis parametras reguliuoja, kaip glaudžiai yra išsidėstę objektai. Šio parametro įtaką iškarto pastebime tarp 3.16 pav. ir 3.19 pav. – abiejuose grafikuose parametras n_neighbors išlieka tas pats (lygus nuliui), tačiau min_dist vertė pasikeičia iš 0 į 0.5, ir atstumas tarp visų objektų tampa akivaizdžiai didesnis. Mažesnė „min_dist“ vertė (3.16 pav.) leidžia taškams sumažintoje dimensijoje būti arti vienas kito, todėl susidaro kompaktiškesnės grupės ir atvaizduojama lokali struktūra. Didesnė min_dist vertė (3.19 pav.) leidžia taškams būti toliau vienas nuo kito sumažintoje dimensijoje, todėl vizualizacijoje matomi didesni tarpai tarp taškų. Tai vėlgi atvaizduoja globalesnę struktūrą.

3.3.5. Parametras „metric“

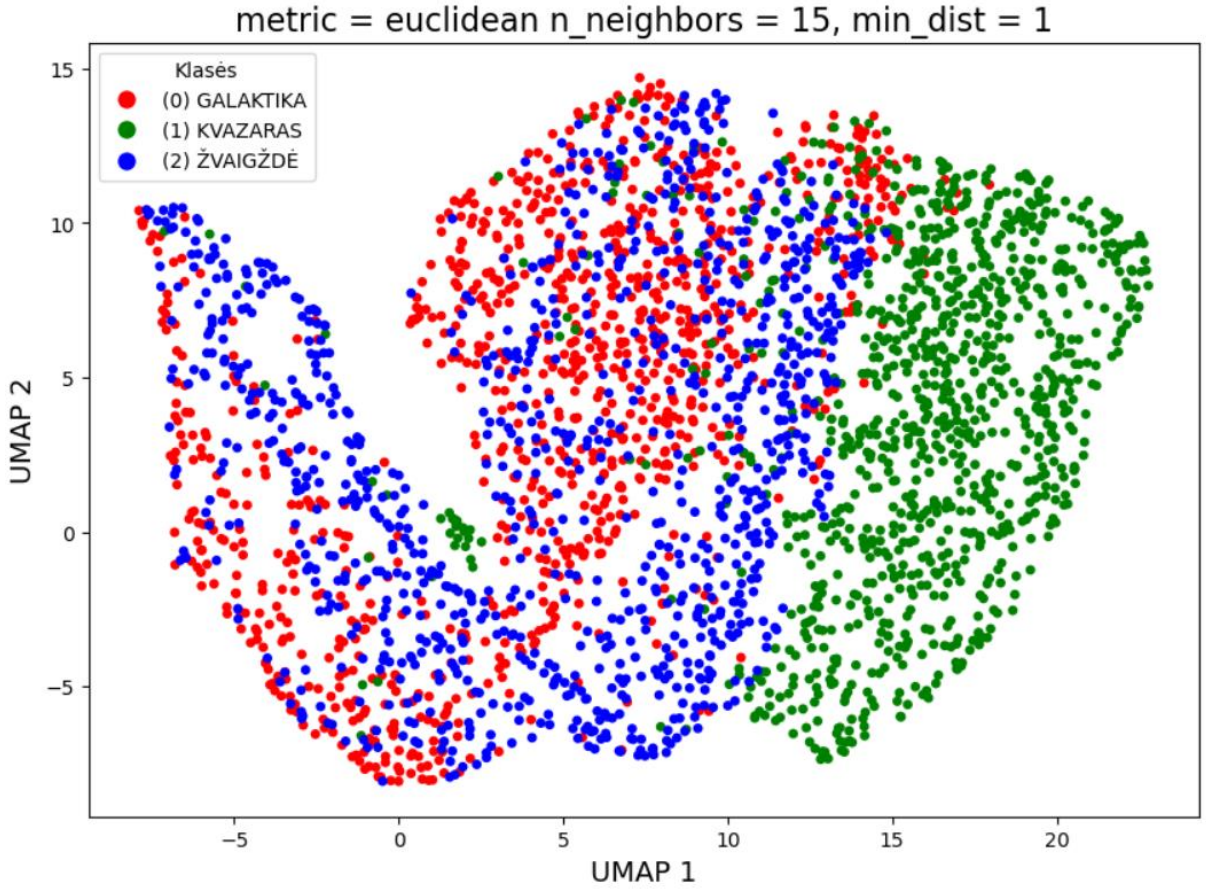
Grafikas, kuriame yra sąlyginai mažiausiai persidengimų tarp skirtingų klasių, turi parametrus – „n_neighbors“ reikšmė lygi 15 ir „min_dist“ reikšmė lygi 1 (3.23 pav.). Toliau bus bandoma dar labiau sumažinti grupių pagal klases persidengimą, naudojant metric parametrą.

Parametro „metric“ reikšmė yra atstumo funkcija. Čia buvo naudojamos „euclidean“, „manhattan“, „cosine“ ir „correlation“ metrikų reikšmės.

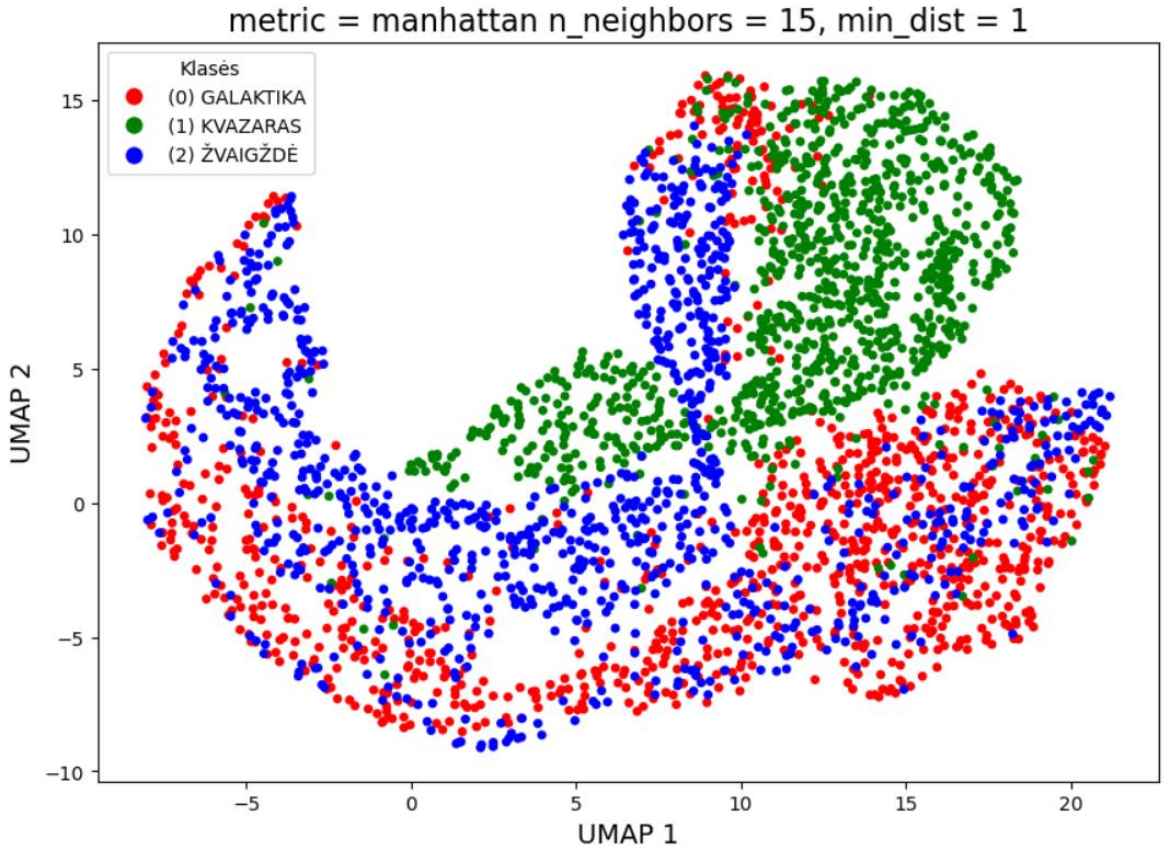
- „euclidean“ metrika skaičiuoja tiesioginį geometrinį atstumą tarp taškų;
- „manhattan“ metrika skaičiuoja atstumą pagal „miesto kvartalo“ principą, kur atstumai matuojami tik horizontaliai ir vertikaliai;

„cosine“ metrika vertina kampinį panašumą tarp taškų, orientuojantis į jų vektorių kryptį, o ne atstumą tarp jų. Nors ši metrika dažniausiai taikoma analizuojant tekstinius dokumentus, tačiau ji taip pat pasitarnavo ir mūsų analizei, padėdama geriau atskirti grupes;

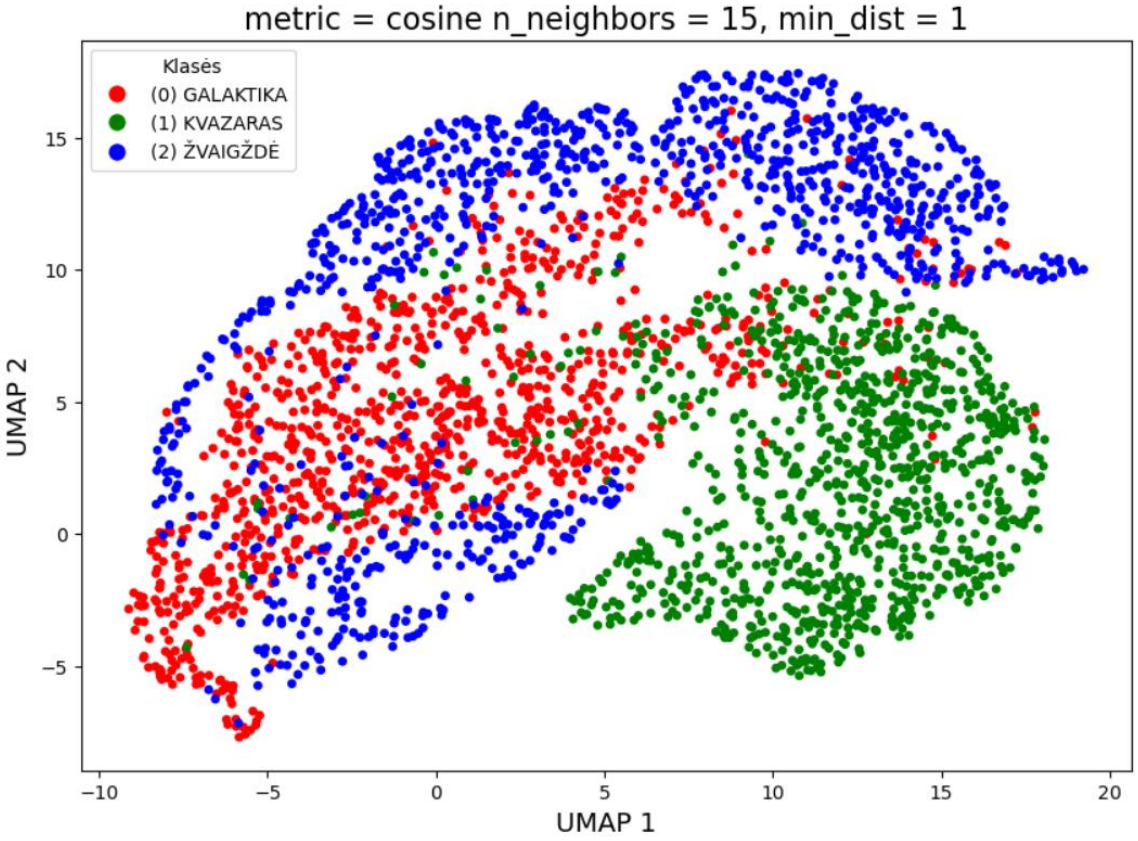
- „correlation“ metrika matuoja, kiek dvi savybės yra tarpusavyje priklausomos, t.y. koreliuoja, o ne kiek jos yra nutolusios viena nuo kitos. Jei savybės didėja ar mažėja panašiai, „correlation“ metrika parodys stipresnį panašumą tarp tų taškų, net jei jų tikrosios vertės yra skirtingos;



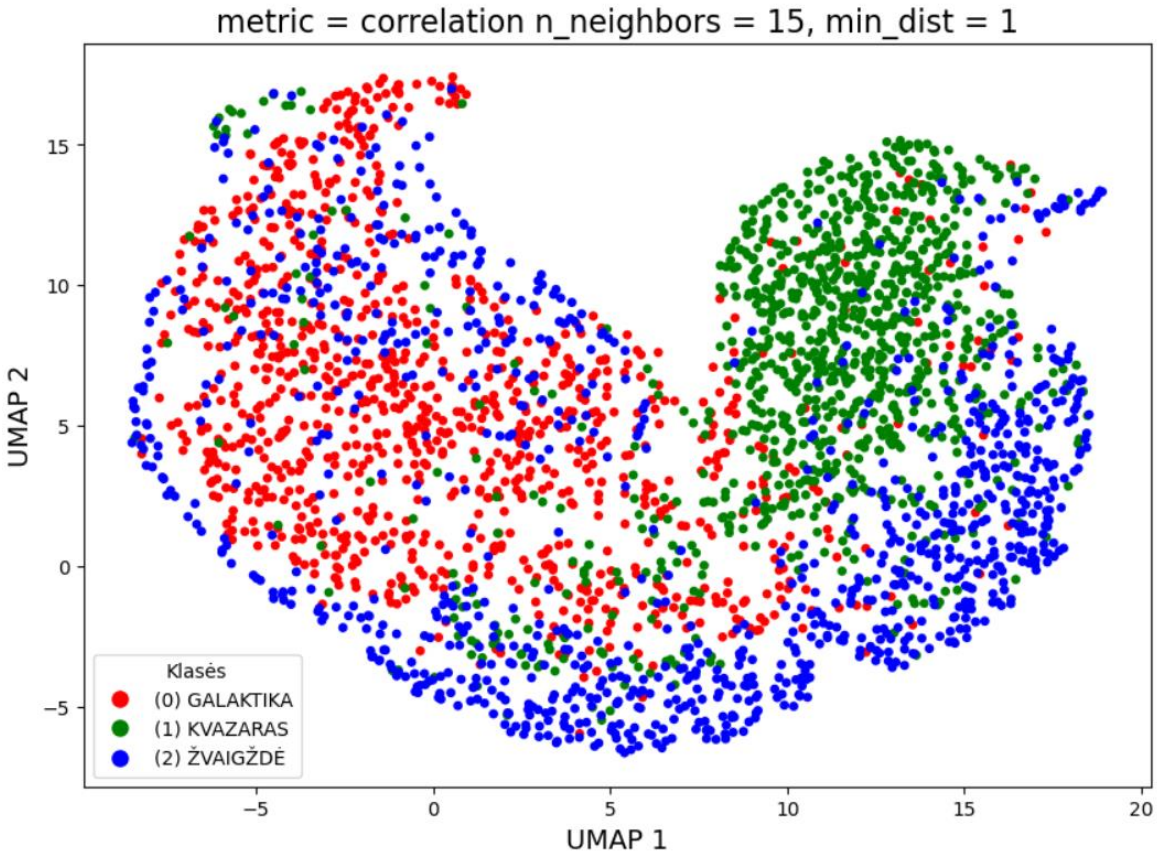
3.25 pav. Dimensijų mažinimas UMAP metodu. „metric“ parametras lygus „euclidean“.



3.26 pav. Dimensijų mažinimas UMAP metodu. „metric“ parametras lygus „manhattan“.



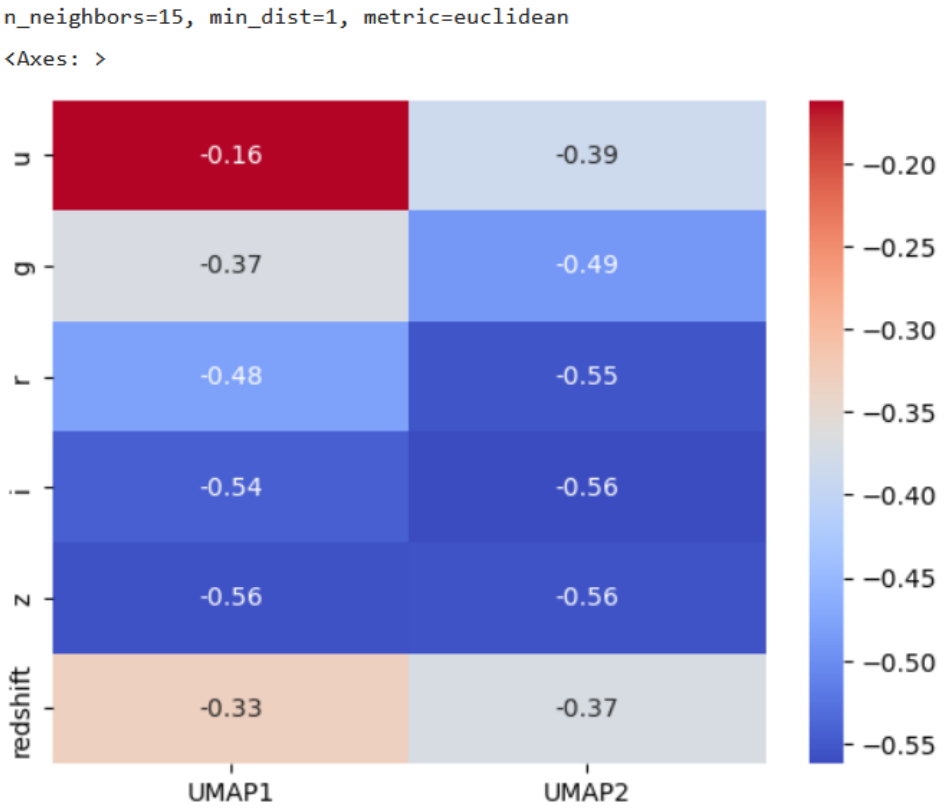
3.27 pav. Dimensijų mažinimas UMAP metodu. „metric“ parametras lygus „cosine“.



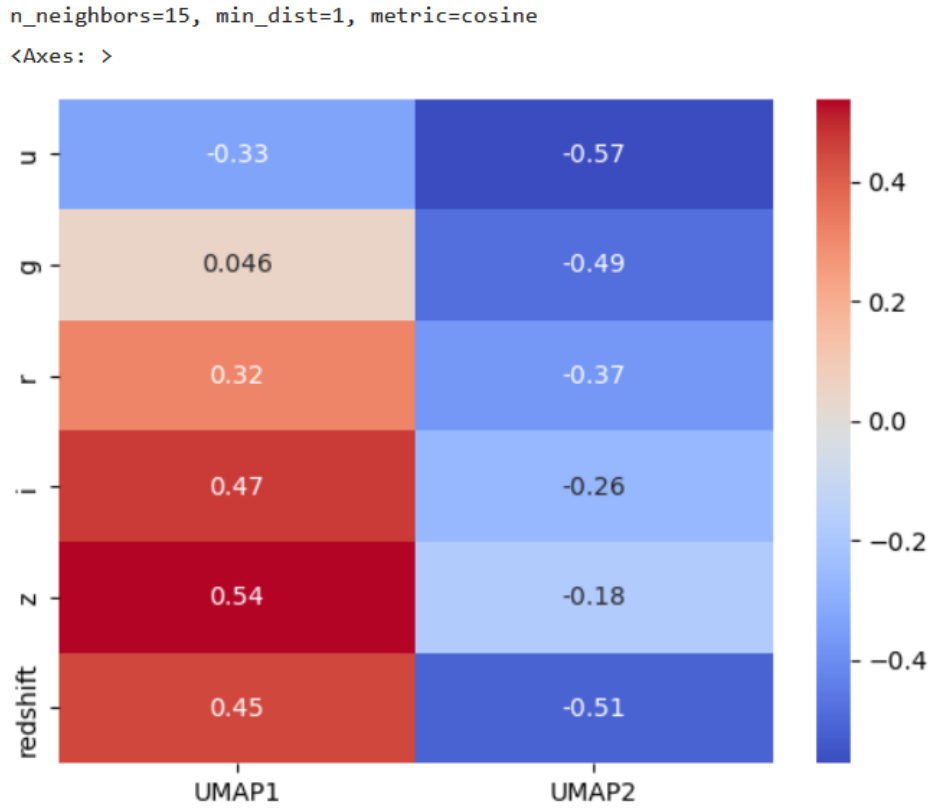
3.28 pav. Dimensijų mažinimas UMAP metodu. „metric“ parametras lygus „correlation“.

Galime aiškiai pastebėti, kad grafikuose, kai atstumai skaičiuojami naudojant „correlation“ ir „cosine“ metrikas, objektų grupės tsiskiria daug labiau negu naudojant numatytąją „euclidean“ metriką. Požymių įtaką galime daugmaž įvertinti atlikus koreliacijos analizę tarp pradinių objektų savybių ir UMAP dimensijų. Verta atkreipti dėmesį, kad kiekvieną kartą skaičiuojant koreliacijas gausime šiek tiek skirtingus rezultatus, tačiau galime pastebėti bendras tendencijas.

Pirmame grafike (3.29 pav.) matome UMAP modelio koreliacijas, kai parametro „n_neighbor“ reikšmė lygi 15, „min_dist“ reikšmė lygi 1 ir „metric“ reikšmė lygi „euclidean“, o antrame grafike (3.30 pav.) koreliacijas, kai „n_neighbor“ ir „min_dist“ reikšmės tokios pačios, tik „metric“ skiriasi – ji lygi „cosine“. Matoma, kad aktualiausi parametrai naudojant „euclidean“ metriką yra „r“, „i“ ir „z“ (vertės tiek UMAP 1, tiek UMAP 2 ašyje koreliuoja neigiamai, apie -0.5). O naudojant „cosine“ metriką aktualiausi parametrai yra „i“, „z“ ir „redshift“, kuria teigiamai koreliuoja UMAP 1 ašyje apie 0.5, ir neigiamai koreliuojantys UMAP 2 ašyje (apie -0.5) “u”, „g“ ir „redshift“.



3.29 pav. UMAP gauto grafiko „euclidean“ metrika ašių ir „redshift“, „u“, „g“, „r“, „i“, „z“ reikšmių koreliacijos.



3.30 pav. UMAP gauto grafiko „cosine“ metrika ašių ir „redshift“, „u“, „g“, „r“, „i“, „z“ reikšmių koreliacijos.

3.3.6. UMAP Išvados

Skirtingos duomenų klasės – galaktikos, kvazarai ir žvaigždės – aiškiausiai atsiskiria UMAP vizualizacijoje kai yra šios parametrų reikšmės: „n_neighbor“ reikšmė lygi 15, „min_dist“ reikšmė lygi 1 ir „metric“ reikšmė lygi „cosine“ arba „correlation“.

UMAP metodas yra veiksmingas dimensijų mažinimo metodas, kurį galime pritaikyti keičiant „n_neighbors“, „min_dist“ ir „metric“ parametrus. UMAP metodo vizualizacijos rezultatuose matoma, kad didesnės parametrų „n_neighbors“ ir „min_dist“ reikšmės lemia grafiką su didesniais atstumais tarp taškų – mažiau atskirtos objektų grupės, o mažesnės reikšmės išryškina vietines struktūras. Be to, „metric“ parametro pasirinkimas nusako metriką, kurią UMAP naudoja skaičiuojant atstumus tarp objektų, o tai daro įtaką objektų grupavimuisi. Metrika „cosine“ yra naudingesnė analizuojant duomenis, kuriuose svarbu išlaikyti savybių proporcinius santykius ir priklausomumus, o ne absoliučius dydžius. Tai buvo pastebėta ir tankio grafikuose (3.4 pav. – 3.8 pav.), kur „z“, „i“, „r“, „g“, „u“ požymių tankio pasiskirstymas tarp klasių buvo panašus, t. y. jų proporciniai santykiai buvo panašūs.

4. IŠVADOS

Atliekant duomenų analizę naudojant PCA, t-SNE ir UMAP metodus, pastebėjome, kad kiekvienas iš jų turi savo stipriąsias ir silpnąsias puses. PCA metodas efektyviai sumažina dimensijų skaičių ir paaiškina didelę duomenų dispersiją, tačiau gali praleisti svarbius požymius, tokius kaip „redshift“, kurie nėra gerai atspindėti pagrindinėse komponentėse.

t-SNE ir UMAP metodai geriau išryškina duomenų struktūrą ir grupes, leidžiančias aiškiau atskirti skirtingas klases – galaktikas, kvazarus ir žvaigždes. Tačiau šių UMAP ir t-SNE dimensijų reikšmės neturi konkrečios prasmės, todėl sunku interpretuoti kaip konkretūs požymiai įtakoja galutinę struktūrą. Jų rezultatai stipriai priklauso nuo metodų parametrų, tokių kaip „perplexity“ ir „n_iter“ (t-SNE), bei „n_neighbors“, „min_dist“ ir „metric“ (UMAP). „t-SNE“ metodas prastai išlaiko globalią struktūrą, todėl jį naudingiau taikyti ieškant lokalių struktūrų. Mūsų atveju, informatyviausias metodas, kuriame aiškiausiai atskiriamos grupės yra UMAP su parametrais „n_neighbor“ reikšmė lygi 15, „min_dist“ reikšmė lygi 1 ir „metric“ reikšmė lygi „cosine“ arba „correlation“. PCA metodo naudoti nerekomenduojama su šia duomenų aibe.

Apibendrinant, norint efektyviai analizuoti aukštos dimensijos duomenis ir aptikti bei atvaizduoti atsiskiriančias grupes, verta naudoti kelis dimensijų mažinimo metodus. Tai leidžia gauti išsamesnį supratimą apie duomenų struktūrą ir užtikrina, kad svarbi informacija nebūtų prarasta dėl konkretaus metodo apribojimų. Be to, parametrų ir metrikų pasirinkimas turi būti kruopščiai apsvarstytas, siekiant išlaikyti svarbias savybių proporcijas ir priklausomybes duomenyse.

5. ŠALTINIAI

- https://umap-learn.readthedocs.io/en/latest/parameters.html
- https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/
- https://pandas.pydata.org/docs/
- https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17

6. KODAS

```
import pandas as pd
import numpy as np

# libraries for easier visualisation
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import seaborn as sns

# libraries for dimensionality reduction
import scipy.stats as stats
from sklearn import manifold
from sklearn.decomposition import PCA
from sklearn.decomposition import TruncatedSVD
import umap

# setting options
pd.set_option('display.max_columns', None)
pd.set_option('float_format', '{:f}'.format)

df_orig = pd.read_csv("star_classification.csv", delimiter=',')

df_orig.info()

df_orig.describe()

df_orig.isna().sum().sum()

df_orig[df_orig['plate'].duplicated()]

df_orig['run_ID'].nunique()

df_orig['rerun_ID'].nunique()

# ## Data Cleaning

# ### Removing not needed columns

df_orig.drop(columns=['obj_ID', 'alpha', 'delta', 'spec_obj_ID', 'rerun_ID', 'MJD'],
inplace=True)

# ### Outlier Removing

# IQR nustatyti atsiskyrėlių ribas
def detect_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (data < lower_bound) | (data > upper_bound)
    return outliers, lower_bound, upper_bound

iqr_outliers, lower_bound, upper_bound = detect_outliers_iqr(df_orig[['u', 'g', 'r',
'i', 'z']])

for column in df_orig[['u', 'g', 'r', 'i', 'z']].columns:
    print(f"IQR lower bound for '{column}': {lower_bound[column]:.2f}")
    print(f"IQR upper bound for '{column}': {upper_bound[column]:.2f}")

    print("\n")

# panaikinti ekstremalių atsiskyrėlių vieną eilutę, kurioje reikšmės yra -9999
df_orig = df_orig[(df_orig[['u', 'g', 'r', 'i', 'z']] != -9999).all(axis=1)]
```

```
df_orig.describe()

# ### Encoding labels

df_orig.replace(['GALAXY', 'QSO', 'STAR'], [0, 1, 2], inplace=True)

# ### Choosing random 1000 objects from each class

df = df_orig.groupby('class', group_keys=False).apply(lambda x:
x.sample(n=1000)).reset_index(drop=True)

# ### Normalization

# kiti masyvai nebuvo normalizuoti, nes jie buvo pavadinimai, kampo laipsniai,
kategorijos ar ID
normalization_cols = ['redshift', 'u', 'g', 'r', 'i', 'z']

dfn = df.copy()
dfminmax = df.copy()
for col in normalization_cols:
    dfn[col] = (dfn[col] - dfn[col].mean()) / dfn[col].std()
    #min-max normalization
    dfminmax[col] = (dfminmax[col] - dfminmax[col].min()) / (dfminmax[col].max() -
dfminmax[col].min())


df.describe()

dfn.describe()

dfminmax.describe()

# ## Visualisation

# ### PCA

feature_cols = ['u', 'g', 'r', 'i', 'z', 'redshift']
pca = PCA(n_components=2)
pca_result = pca.fit_transform(df[feature_cols].values)
pca_df = pd.DataFrame(data = pca_result, columns = ['principal component 1',
'principal component 2'])

# Explained variability per principal component
print(pca.explained_variance_ratio_)

plt.figure()
plt.figure(figsize=(7,7))
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xlabel('PC 1', fontsize=13)
plt.ylabel('PC 2', fontsize=13)
plt.title("Principinių komponenčių analizė", fontsize=15)
targets = [0, 1, 2]
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = df['class'] == target
    plt.scatter(pca_df.loc[indicesToKeep, 'principal component 1']
               , pca_df.loc[indicesToKeep, 'principal component 2'], c = color, s =
15)

plt.legend(targets,prop={'size': 15}) #

# Fit PCA
pca.fit(df[feature_cols])
# create df for showing loadings
```

```
loadings_df = pd.DataFrame(pca.components_, columns=feature_cols, index=['PC1',
'PC2'])
loadings_df


# ### TSNE


import matplotlib.pyplot as plt
from sklearn import manifold
import pandas as pd

n_iter = 250

fig, ax = plt.subplots(1, 1, figsize=(8, 6), constrained_layout=True)

class_labels = ["Galaktika (0)", "Kvazaras (1)", "Žvaigždė (2)"]

tsne = manifold.TSNE(n_components=2, n_iter=n_iter, random_state=42)
tsne_data = tsne.fit_transform(df[feature_cols].values)

for class_value, label in zip([0, 1, 2], class_labels):
    class_mask = pd.factorize(df['class'])[0] == class_value
    ax.scatter(tsne_data[class_mask, 0], tsne_data[class_mask, 1], label=label,
alpha=0.7)

ax.set_title("t-SNE", fontsize=20)
ax.set_xlabel('t-SNE Dimensija 1', fontsize=18)
ax.set_ylabel('t-SNE Dimensija 2', fontsize=18)
ax.legend(loc="upper right", fontsize=16)

# Show the plot
plt.show()


df['t-SNE Dimension 1'] = tsne_data[:, 0]
df['t-SNE Dimension 2'] = tsne_data[:, 1]

correlation_matrix = df[['u', 'g', 'r', 'i', 'z', 'redshift', 't-SNE Dimension 1', 't-
SNE Dimension 2']].corr()

correlation_with_tsne = correlation_matrix[['t-SNE Dimension 1', 't-SNE Dimension
2']].iloc[:-2]

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_with_tsne, annot=True, cmap="coolwarm", vmin=-1, vmax=1)
plt.title("Koreliacija tarp reikšmių ir t-SNE dimensijų", fontsize=20)
plt.xlabel("t-SNE Dimensijos", fontsize=18)
plt.ylabel("Reikšmės", fontsize=18)
plt.show()

# ##### n_iter increase


initial_n_iter = 250
doublings = 2

fig, axes = plt.subplots(doublings + 1, 1, figsize=(8, 6 * (doublings + 1)),
constrained_layout=True)

class_labels = ["Galaktika (0)", "Kvazaras (1)", "Žvaigždė (2)"]
colors = plt.cm.viridis([0.2, 0.5, 0.8])

for i in range(doublings + 1):
    n_iter = initial_n_iter * (3 ** i)
    tsne = manifold.TSNE(n_components=2, n_iter=n_iter, random_state=42)
```

```
    tsne_data = tsne.fit_transform(df[feature_cols].values)
    ax = axes[i] if doublings > 0 else axes

    for class_value, color, label in zip([0, 1, 2], colors, class_labels):
        class_mask = pd.factorize(df['class'])[0] == class_value
        ax.scatter(tsne_data[class_mask, 0], tsne_data[class_mask, 1], color=color,
label=label, alpha=0.7)

    ax.set_title(f"t-SNE su n_iter={n_iter}", fontsize=20)
    ax.set_xlabel('t-SNE Dimensija 1', fontsize=18)
    ax.set_ylabel('t-SNE Dimensija 2', fontsize=18)
    ax.legend(loc="upper right")

plt.show()

# ##### perplexity increase

initial_perp = 10
doublings = 2

fig, axes = plt.subplots(doublings + 1, 1, figsize=(8, 6 * (doublings + 1)),
constrained_layout=True)

class_labels = ["Galaktika (0)", "Kvazaras (1)", "Žvaigždė (2)"]
colors = plt.cm.viridis([0.2, 0.5, 0.8])

for i in range(doublings + 1):
    perp = initial_perp * (3 ** i)
    tsne = manifold.TSNE(n_components=2, n_iter=250, perplexity=perp, random_state=42)
    tsne_data = tsne.fit_transform(df[feature_cols].values)
    ax = axes[i] if doublings > 0 else axes

    for class_value, color, label in zip([0, 1, 2], colors, class_labels):
        class_mask = pd.factorize(df['class'])[0] == class_value
        ax.scatter(tsne_data[class_mask, 0], tsne_data[class_mask, 1], color=color,
label=label, alpha=0.7)

    ax.set_title(f"t-SNE su perplexity={perp}", fontsize=20)
    ax.set_xlabel('t-SNE Dimensija 1', fontsize=18)
    ax.set_ylabel('t-SNE Dimensija 2', fontsize=18)
    ax.legend(loc="upper right")

plt.show()

# ### UMAP

feature_cols = ['u', 'g', 'r', 'i', 'z', 'redshift']
def draw_umap(n_neighbors=15, min_dist=0.1, n_components=2, metric='euclidean',
title='', data=df):
    fit = umap.UMAP(
        n_neighbors=n_neighbors,
        min_dist=min_dist,
        n_components=n_components,
        metric=metric
    )
    u = fit.fit_transform(df[feature_cols])

    colors = ['r', 'g', 'b']
    class_labels = ['GALAKTIKA', 'KVAZARAS', 'ŽVAIGŽDĖ']
    cmap = mcolors.ListedColormap(colors)

    fig, ax = plt.subplots(figsize=(10, 7))
    scatter = ax.scatter(u[:, 0], u[:, 1], c=data['class'], cmap=cmap, s=15)
    plt.title(title, fontsize=16)

    plt.xlabel('UMAP 1', fontsize=14)
```

```
    plt.ylabel('UMAP 2', fontsize=14)

    legend_handles = [plt.Line2D([0], [0], marker='o', color='w',
markerfacecolor=colors[i], markersize=10)
                      for i in range(len(class_labels))]
    plt.legend(legend_handles, [f"({i}) {label}" for i, label in
enumerate(class_labels)], title="Klasės")

    plt.show()

for x in (df, dfminmax, dfn):
    for n in (15, 50, 200):
        draw_umap(n_neighbors=n, metric='manhattan', title='n_neighbors =
{}'.format(n), data=x)

# for each of min distances 0, 0.5 and 1 take n_neighbors 5, 15 and 50 and draw umap
graphs
for d in (0.0, 0.5, 1):
    for n in (5, 15, 50):
        draw_umap(n_neighbors=n, min_dist=d, title='n_neighbors = {}, min_dist =
{}'.format(n, d), data=dfn)

for metric_s in ('manhattan', 'euclidean', 'cosine', 'correlation'):
    draw_umap(n_neighbors=15, min_dist=1, metric=metric_s, title='metric = {}
n_neighbors = 15, min_dist = 1'.format(metric_s), data=dfn)

# Define UMAP model and get the embeddings
umap_model = umap.UMAP(n_neighbors=15, min_dist=1, n_components=2, metric='cosine')
umap_embedding = umap_model.fit_transform(df[feature_cols])

# DataFrame for the UMAP results and feature columns
umap_df = pd.DataFrame(umap_embedding, columns=['UMAP1', 'UMAP2'])
features_df = df[feature_cols]

# Correlations between each feature and the UMAP axes
correlations = features_df.corrwith(umap_df['UMAP1']).to_frame(name='UMAP1')
correlations['UMAP2'] = features_df.corrwith(umap_df['UMAP2'])

print("n_neighbors=15, min_dist=1, metric=cosine")
sns.heatmap(correlations, annot=True, cmap="coolwarm")

# Define UMAP model and get the embeddings
umap_model = umap.UMAP(n_neighbors=5, min_dist=1, n_components=2, metric='euclidean')
umap_embedding = umap_model.fit_transform(df[feature_cols])

# DataFrame for the UMAP results and feature columns
umap_df = pd.DataFrame(umap_embedding, columns=['UMAP1', 'UMAP2'])
features_df = df[feature_cols]

# Correlations between each feature and the UMAP axes
correlations = features_df.corrwith(umap_df['UMAP1']).to_frame(name='UMAP1')
correlations['UMAP2'] = features_df.corrwith(umap_df['UMAP2'])

print("n_neighbors=15, min_dist=1, metric=euclidean")
sns.heatmap(correlations, annot=True, cmap="coolwarm")

# ### histograms, box plots

plt.figure(figsize=(12, 6))
sns.histplot(data=dfminmax, x='redshift', hue='class', bins=30, kde=True,
palette='viridis')
plt.title("Redshift distribucija pagal klasę", fontsize=16)
plt.xlabel("Redshift", fontsize=16)
plt.ylabel("Dažnis", fontsize=16)
plt.legend(title="Klasė", labels=["Galaktika (0)", "Kvazaras (1)", "Žvaigždė (2)"],
fontsize=20)
```

```
plt.show()

features = ['u', 'g', 'r', 'i', 'z']

n_rows = len(features)
fig, axes = plt.subplots(n_rows, 1, figsize=(12, 6 * n_rows), constrained_layout=True)

for i, feature in enumerate(features):
    sns.kdeplot(data=dfminmax, x=feature, hue='class', palette='viridis', ax=axes[i],
fill=False, linewidth=2.5)
    axes[i].set_title(f"{feature} distribucija pagal klasę", fontsize=16)
    axes[i].set_xlabel(feature, fontsize=16)
    axes[i].set_ylabel("Tankis", fontsize=16)
    axes[i].legend(title="Klasė", labels=["Galaktika (0)", "Kvazaras (1)", "Žvaigždė
(2)"], fontsize=12, title_fontsize=14, loc="upper right")

plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(data=df_orig[['u', 'g', 'z', 'r', 'i', 'redshift']], color='skyblue')
plt.title("Stačiakampė diagrama, pabrėžianti 'u', 'g', ir 'z' atsiskyrėlius")
plt.ylabel("Reikšmė")
plt.show()

# Create a separate figure for histograms
fig, axs = plt.subplots(2, 3, figsize=(15, 10))

# List of columns to plot histograms for
columns = ['u', 'g', 'z', 'r', 'i', 'redshift']

# Generate histograms for each column
for i, col in enumerate(columns):
    sns.histplot(df_orig[col], bins=30, ax=axs[i//3, i%3], kde=True, color='skyblue')
    axs[i//3, i%3].set_title(f"'{col}' histograma", fontsize=20)
    axs[i//3, i%3].set_xlabel("Reikšmė", fontsize=18)
    axs[i//3, i%3].set_ylabel("Dažnis", fontsize=18)

plt.tight_layout()
plt.show()

max_redshift_per_class = df_orig.groupby('class')['redshift'].max()
print("Maximum redshift value per class:\n", max_redshift_per_class)

# Plot a histogram or boxplot to visualize the redshift distribution for each class
sns.boxplot(data=df_orig, x='class', y='redshift')
plt.title("'redshift' distribucija pagal klasę")
plt.xlabel("Klasė (0 = galaktika, 1 = kvazaras, 2 = žvaigždė)")
plt.ylabel("'Redshift'")
plt.yscale("linear")
plt.show()
```