



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

MATEMATINĖS STATISTIKOS KATEDRA

Rūta Dubovikienė

**BIBLIOGRAFINIŲ DUOMENŲ STATISTINĖ
ANALIZĖ
STATISTICAL ANALYSIS OF BIBLIOGRAPHICAL DATA**

Baigiamasis magistro darbas

Duomenų mokslo ir statistikos studijų programa, valstybinis kodas 6211AX009

Statistikos studijų kryptis

Vilnius, 2022

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

MATEMATINĖS STATISTIKOS KATEDRA

TVIRTINU
Katedros vedėjas

(Parašas)

(Vardas, pavardė)

(Data)

Rūta Dubovikienė

**BIBLIOGRAFINIŲ DUOMENŲ STATISTINĖ
ANALIZĖ
STATISTICAL ANALYSIS OF BIBLIOGRAPHICAL DATA**

Baigiamasis magistro darbas

Duomenų mokslo ir statistikos studijų programa, valstybinis kodas 6211AX009

Statistikos studijų kryptis

Vadovas	doc. dr. Rūta Simanavičienė	_____	_____
Konsultantas	_____	_____	_____
Konsultantas	_____	_____	_____
Lietuvių kalbos konsultantas	dr. Vaida Buivydienė	_____	_____

Vilnius, 2022

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
MATEMATINĖS STATISTIKOS KATEDRA

Studijų kryptis: Statistika

Studijų programa: Duomenų mokslas ir statistika , valstybinis kodas 6211AX009

Specializacija: Duomenų mokslas ir statistika

TVIRTINU

Katedros vedėjas

Kazimieras Padvelskis

2022-05-27

MAGISTRO BAIGIAMOJO DARBO UŽDUOTIS

Nr. DMfm-20-273

Vilnius

Studentas (-ė): Rūta Dubovikienė

Baigiamojo darbo tema: Bibliografinių duomenų statistinė analizė

Baigiamojo darbo užbaigimo terminas pagal numatytą studijų kalendorinį grafiką.

BAIGIAMOJO DARBO UŽDUOTIS:

Planuojamas rezultatas: VILNIUS TECH mokslininkų publikacijų registruotų VILNIUS TECH bibliotekoje statistinė analizė.

Kita: Nagrinėjant mokslinius straipsnius susijusius su bibliografinių duomenų analize pastebima, jog dažniausia bibliografinių duomenų statistinė analizė apsiriboja tik aprašomąja statistika, bei bibliometrinių tinklų diagramomis, išsamesnė statistinė analizė nėra vykdoma. Šio darbo užduotis, nagrinėjant Vilniaus Gedimino technikos universiteto mokslininkų mokslinių straipsnių, publikuotų 2010-2020 metais bibliografinius duomenis parengti metodiką, kurios pagalba būtų galima atlikti platesnę statistinę bibliografinių duomenų analizę, panaudojant tekstinių duomenų analizės, grafų teorijos, bei klasterinės analizės metodus.

Uždaviniai:

1. Atlikti literatūros analizę, susijusią su bibliografinių duomenų moksliniais tyrimais;
2. Išnagrinėti tekstinės duomenų analizės, grafų teorijos ir klasterinės analizės metodus;
3. Pasiūlyti bibliometrinės analizės metodiką, kuri praplėstų bibliografinių duomenų analizę;
4. Pasiūlytą metodiką pritaikyti VILNIUS TECH mokslininkų publikacijų registruotų VILNIUS TECH bibliotekoje statistinę analizę.

Vadovas docentas Rūta Simanavičienė

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Matematinės statistikos katedra

ISBN ISSN
Egz. sk. 1
2022-05-30

Statistikos studijų programos baigiamasis magistro darbas
Bibliografinių duomenų statistinė analizė
Autorius Rūta Dubovikienė
Vadovas doc. dr. Rūta Simanavičienė

Kalba
☒ lietuvių
☐ užsienio

Anotacija

Bibliometrija yra daugiadalykė mokslo sritis, tirianti mokslo ir dokumentų bibliografines charakteristikas statistiniais ir matematiniais metodais. Baigiamojo magistro darbo tikslas – praplėsti bibliometrinės analizės tyrimų sritį, pasiūlant metodiką, kuri apima bibliografinių duomenų paruošimą, tekstinę duomenų analizę, grafų teoriją bei klasterizavimo modelio kūrimą.

Teorinėje darbo dalyje apžvelgiama ir pristatoma nagrinėta literatūra, susijusi su bibliografinė analize. Šioje dalyje taip pat nagrinėjamos pagrindinės grafų teorijos sąvokos, tinklų analizė ir jų vizualizavimas, bendruomenių radimas taikant Girvan – Newman algoritimą, k -vidurkių klasterizavimo modelio kūrimo procesas ir tekstinė duomenų analizė. Praktinėje darbo dalyje pristatoma siūloma bibliometrinės analizės metodika ir pritaikoma tyrimo objektui: VILNIUS TECH bibliotekoje registruotiems mokslinių publikacijų bibliografiniams duomenims. Joje aprašomas bibliografinių duomenų paruošimo ir valymo procesas bei pritaikomas teksto palyginimo algoritmas. Sukuriami autorių bendradarbiavimo tinklai, analizuojami grafų teorijos rodikliai, išskiriamos bendruomenės ir pateikiamas klastrinės analizės rezultatų vertinimas.

Darbą sudaro 7 dalys: įvadas, susijusių darbų analizė, bibliografinių duomenų analizės metodų apžvalga, tiriamoji dalis, išvados, literatūros sąrašas, priedai. Darbo apimtis: 76 p. teksto be priedų, 26 iliustracijos, 21 lentelė, 58 šaltiniai.

Prasminiai žodžiai: bibliografiniai duomenys, grafų teorija, bendruomenės, klasterinė analizė, k -vidurkių metodas, tekstinė duomenų analizė

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Mathematical Statistics

ISBN ISSN
No of copies: 1
2022-05-30

Master's Thesis of the Study Program in Statistics
Statistical Analysis of Bibliographical Data
Author Rūta Dubovikienė
Thesis supervisor doc. dr. Rūta Simanavičienė

Language
☐ Lithuanian
☒ Foreign

Summary

Bibliometrics is a multidisciplinary field of science that researches the bibliographic characteristics of documents using statistical and mathematical methods. The aim of the final Master's thesis is to expand the field of bibliometric analysis research by offering a methodology that includes the preparation of bibliographic data, text data analysis, graph theory and the development of a clustering model.

The theoretical part of the work is dedicated to the review and presentation of literature related to bibliographic data analysis. The basic concepts of graph theory, network analysis and their visualization, finding communities using the Girvan - Newman algorithm, the process of developing a k -means clustering model, and text data analysis are also presented in this section. In the practical part of the work, the proposed methodology of bibliometric analysis is presented and applied to the research object: bibliographic data of scientific publications registered in VILNIUS TECH library. The process of preparing and cleaning bibliographic data and application of text comparison algorithm are described as well as creation of author collaboration networks, analysis of indicators of graph theory, singled out communities and cluster analysis results are presented.

The final Master's thesis consists of 7 parts: introduction, analysis of related works, overview of bibliographic data analysis methods, research part, conclusions, references and appendices. Scope of work: 76 pp. of text without appendices, 26 illustrations, 21 tables, 58 references.

Key words: bibliographic data, graph theory, communities, cluster analysis, k -means method, text data analysis

TURINYS

1. Įvadas	6
1.1. Tyrimo objektas.....	7
1.2. Darbo tikslas ir uždaviniai.....	7
1.3. Temos aktualumas	7
1.4. Mokslinė darbo vertė.....	8
1.5. Darbo struktūra.....	8
2. Susijusių darbų analizė	9
2.2. Bibliografijos ir bibliometrijos sąvokos.....	9
2.3. Bibliografinių duomenų analizė	11
2.4. Žurnalų cituojamumo rodikliai.....	13
2.4.1. „Scopus“ citavimo rodikliai	13
2.4.2. WoS citavimo rodikliai	15
2.5. Socialinių tinklų analizė	16
2.6. Bibliografiniai tinklai	17
2.7. Klasterinės analizės taikymas bibliometrinėje analizėje	19
3. Metodai bibliografinių duomenų analizei	20
3.2. Grafų teorija ir jos taikymas tinklų analizėje	20
3.2.1. Pagrindinės sąvokos.....	20
3.2.2. Grafų teorijos taikymas tinklų analizėje	21
3.3. Bendruomenės sąvoka.....	25
3.4. Tinklo grafinis vaizdavimas	26
3.5. k -vidurkių klasterizavimo metodas	29
3.6. Duomenų normalizavimas.....	30
3.7. Optimalaus klasterių skaičiaus nustatymas	31
3.8. Tekstinė duomenų analizė	33

3.8.1.	Taisyklėmis pagrįsti algoritmai.....	33
3.8.2.	Neapibrėžti algoritmai	34
4.	Tiriamoji dalis.....	37
4.2.	Siūloma bibliometrinės analizės metodika.....	37
4.3.	Duomenys.....	38
4.3.1.	Duomenų rinkinio apžvalga.....	38
4.3.2.	Duomenų rinkinio valymas ir duomenų paruošimas	39
4.3.3.	Žurnalų indeksavimas	44
4.4.	Bibliometrinė analizė	46
4.5.	Autorių bendradarbiavimo tinklas.....	50
4.5.1.	Gephi duomenų rinkinių sudarymas ir tinklo vizualizavimas	50
4.5.2.	Autorių bendraautorystės tinklo analizė	53
4.5.3.	Autorių bendraautorystės tinklas pašalinus išskirtis	60
4.6.	Žurnalų klasterizavimas	67
4.6.1.	K-vidurkių modelio kūrimas.....	67
4.6.2.	Optimalaus klasterių skaičiaus parinkimas.....	68
4.6.3.	Rezultatų analizė.....	72
	Tyrimo rezultatai ir išvados	76
	Literatūros ir kitų šaltinių sąrašas	77
	Priedai	83
1.	Python kodas, duomenų rinkinio valymui ir tekstinei duomenų analizei atlikti bei Gephi duomenų rinkiniams sukurti	83
2.	Python kodas, naudotas bibliometrinei analizei atlikti.....	100
3.	Python kodas, skirtas grafų teorijos taikumui	106
4.	Python kodas, skirtas klasterinei analizei atlikti	113

LENTELIŲ SĄRAŠAS

1 lentelė. Bibliometrinės analizės kryptys literatūros šaltiniuose	11
2 lentelė. Pagrindinės neapdorotų duomenų kokybės problemos	41
3 lentelė. Duomenų rinkinys.....	43
4 lentelė. Bendraautorų skaičius.....	49
5 lentelė. Produktiviausi autoriai	49
6 lentelė. Duomenų rinkiniai grafo sudarymui.....	51
7 lentelė. Laipsnių statistiniai rodikliai	54
8 lentelė. Grafo statistiniai rodikliai	56
9 lentelė. Grafo centriškumo matai	57
10 lentelė. Viršūnių pasiskirstymas bendruomenėse.....	59
11 lentelė. Bendruomenės iš 11 narių analizė	59
12 lentelė. Bendruomenės iš 3 narių analizė	60
13 lentelė. Laipsnių statistiniai rodikliai	62
14 lentelė. Grafo statistiniai rodikliai	63
15 lentelė. Grafo centriškumo matai	63
16 lentelė. Viršūnių pasiskirstymas bendruomenėse.....	65
17 lentelė. Viršūnių pasiskirstymas naujai susiformavusioje bendruomenėje	65
18 lentelė. Duomenų rinkinys klasterizavimo modeliui sudaryti.....	67
19 lentelė. Silueto koeficientas.....	70
20 lentelė. Vidutinės rodiklių reikšmės klasteriuose	73
21 lentelė. 10 dažniausiai pasikartojusių žurnalų	75

ILIUSTRACIJŲ SĄRAŠAS

1 pav. Minimalistinis straipsnių tinklo pavyzdys	17
2 pav. Optimalaus klasterių skaičiaus radimas „alkūnės“ metodu	32
3 pav. Siūloma bibliometrinės analizės metodika	37
4 Pav. Publikacijų skaičiaus pokytis 2010 – 2019 metais	46
5 Pav. Publikacijų pasiskirstymas pagal mokslo sritis	47
6 Pav. Technologijos mokslų sritys	47
7 Pav. Technologijos mokslų sričių publikacijų skaičiaus pokytis 2010 – 2019 metais	48
8 Pav. Publikacijų skaičiaus pokytis 2010 – 2019 metais, pagal leidinius, kuriuose buvo spausdinama ...	48
9 Pav. Duomenų bazėse taplinamų publikacijų skaičiaus pokytis 2010 – 2019 metais	50
10 Pav. Tinklo parametrai	52
11 Pav. Visų autorių bendraautorystės tinklas	53
12 Pav. Viršūnių laipsniai	54
13 Pav. Antisimetriškumo ir eksceso rodikliai	55
14 Pav. Vidutinis laipsnių jungumas	56
15 Pav. Vidutinis kaimyninių viršūnių laipsnis	56
16 Pav. Bendruomenės autorių tinkle	58
17 Pav. Autorių bendraautorystės tinklas, pašalinus išskirtis	61
18 Pav. Viršūnių laipsniai	62
19 Pav. Vidutinis laipsnių jungumas ir vidutiniai kaimyninių viršūnių laipsniai	63
20 Pav. Bendruomenės autorių tinkle be išskirčių	64
21 Pav. Bendruomenės grafinis vaizdavimas	66
22 Pav. Optimalaus klasterių skaičiaus radimas „alkūnės“ metodu	69
23 Pav. Sklaidos diagrama	71
24 Pav. Žurnalų klasterių pasiskirstymas	72
25 Pav. Vidutinės rodiklių reikšmės klasteriuose	73
26 Pav. Vidutinės rodiklių reikšmės klasteriuose ir klasterių dydžiai	74

SANTRUMPOS

APA – Amerikos psichologijos asociacijos (*American Psychological Association* – APA) citavimo stilius

AIF – Aggregate Impact Factor rodiklis skaičiuojamas, atsižvelgiant į visų kategorijos žurnalų citatų skaičių ir straipsnių iš visų kategorijos žurnalų skaičių.

API – Aplikacijų programavimo sąsają

DB – Duomenų bazė

DOI – Skaitmeninis objekto identifikatorius

IF – Impact Factor yra žurnalo cituojamumo rodiklis

JCR – Journal Citation Reports, žurnalų cituojamumo ataskaitos

JIF – Journal Impact Factor žurnalo poveikio rodiklis

SCIE – Science Citation Index Expanded, išplėstas mokslo šaltinių indeksas

SJR – SCImago Journal Rank rodikliu yra apskaičiuojamas žurnalo prestižas pagal žurnalų tarpusavio citavimų tinklus

SNIP – Source Normalized Impact per Paper rodiklis yra realaus ir tikėtino (potencialaus) vidutinio citavimų skaičiaus, tenkančio vienai žurnalo publikacijai atstovaujamoje mokslo srityje, santykis

SSCI – Social Sciences Citation Index, socialinių mokslų šaltinių citavimo indeksas

VILNIUS TECH – Vilniaus Gedimino technikos universitetas

WoS – „Web of Science“ svetainė, kuri prenumeratos pagrindu suteikia prieigą prie įvairių duomenų bazių, kuriose pateikiami išsamūs daugelio skirtingų akademinių disciplinų citavimo duomenys.

1. Įvadas

Bibliometrinė analizė literatūroje apibūdinama kaip statistinių metodų rinkinys, kurio pagalba bibliografiniai duomenys analizuojami įvairiais pjūviais – vertinamas citavimo dažnumas, bendra autorių skaičius, geografinis leidinių pasiskirstymas ir jų skaičius tam tikroje srityje bei kiekio pasikeitimas laikui bėgant. Atlikus bibliometrinę analizę, galima nustatyti tam tikros mokslo šakos padėtį institucijos ar kito tiriamo objekto lygmeniu ir vertinti mokslinių tyrimų perspektyvas.

Bibliometrinė analizė turi daug panašumų su aprašomąja statistika dėl duomenų sisteminimo ir grafinio vaizdavimo metodų taikymo. Taikant bibliometrinę analizę, panašiai kaip ir taikant aprašomąją statistiką, siekiama koncentruotai užrašyti informaciją, esančią dideliuose duomenų masyvuose, ją vizualizuoti ir daryti pagrįstas išvadas apie visos populiacijos nagrinėjamas savybes.

Literatūros analizė atskleidė, jog autoriai beveik nenagrinėja bibliografinių duomenų ypatumų. Šie duomenų rinkiniai pasižymi keletu savybių, kurios gali apsunkinti tyrimą: duomenų rinkiniuose gausu nereikšmingų, neaiškių, trūkstančių ir netvarkingų įrašų. Bibliografiniai duomenys yra sudaryti iš tekstinių įrašų, kurie yra apibūdinami kaip nestruktūrizuoti duomenys. Dėl šių savybių būtinas kruopštus išankstinis bibliografinių duomenų apdorojimas, taikant tekstinių duomenų analizės principus. Neatlikus šio analizės etapo, rizikuojama gauti klaidingas arba netikslas išvadas.

Nagrinėtuose literatūros šaltiniuose daug dėmesio skiriama duomenų vizualizavimui ir bibliometrinių tinklų sudarymui. Šiam tikslui pasiekti naudojami įvairūs įrankiai, kurių pagalba yra nesudėtinga sukurti tokius tinklus. Tačiau dažnu atveju, tinklų analizė tuo ir apsiriboja – nagrinėti autoriai netiria tinklų sandaros, nebando atrasti įvairių dėsnų tarp skirtingų tinklų ar tinklus sudarančių objektų, nenagrinėja atskirų objektų svarbos ir jų tarpusavio priklausomybės.

Klasterizavimas taip pat yra vienas iš dažniausiai nagrinėtoje literatūroje pasikartojančių bibliometrinės analizės metodų. Bibliografiniai duomenys skirstomi į klasterius pagal pagrindinius kriterijus: mokslo šakas, geografinę priklausomybę, autorius, raktinius žodžius. Klasterizavimo metodas dažnai derinamas su bibliometriniais tinklais ir rezultatai vaizduojami viename paveiksle.

1.1. Tyrimo objektas

VILNIUS TECH bibliotekoje registruotų mokslinių publikacijų bibliografiniai duomenys.

1.2. Darbo tikslas ir uždaviniai

Darbo tikslas: Praplėsti bibliometrinės analizės tyrimų sritį, pasiūlant metodiką, kuri apima bibliografinių duomenų paruošimą, tekstinę duomenų analizę, grafų teoriją bei klasterizavimo modelio kūrimą.

Darbo uždaviniai:

1. Atlikti literatūros analizę, susijusią su bibliografinių duomenų moksliniais tyrimais;
2. Išnagrinėti tekstinės duomenų analizės, grafų teorijos ir klasterinės analizės metodus;
3. Pasiūlyti bibliometrinės analizės metodiką, kuri praplėstų bibliografinių duomenų analizę;
4. Pasiūlytą metodiką pritaikyti VILNIUS TECH mokslinių publikacijų, registruotų VILNIUS TECH bibliotekoje, statistinei analizei.

1.3. Temos aktualumas

Dėl duomenų skaitmeninimo, bibliografiniai duomenys tampa vis labiau prieinami ir šis tyrimų laukas plečiasi. Mokslinių darbų, kuriuose taikoma bibliometrinė analizė, yra daug ir jie sparčiai keičiasi, nes atrandami vis nauji analizės metodai, tobulėja naudojami analizės ir vizualizavimo įrankiai. Bibliometrinės analizės sritis yra labai plati, tačiau nepaisant to, daugelis autorių dažniausiai taiko pakankamai paprastus analizės metodus, kaip aprašomoji statistika ir duomenų vizualizavimas. Nėra nagrinėjami grafų teorijos rodikliai, klasterinė analizė atliekama remiantis vienu ar tik keliais rodikliais.

Bibliometrinės analizės galimybės yra labai didelės, sukuriami ir pritaikomi vis daugiau įvairių įrankių ir programavimo kalbų, kurios tinka tokiems tyrimams vykdyti. Dideliems bibliometriniais tinklams kurti ir analizuoti užtenka kasdienėje veikloje naudojamų kompiuterių, tyrėjams yra prieinamos atvirojo kodo sistemos. Dėl to, naujų galimybių ieškojimas ir įvairių statistinių metodų taikymas, bibliometrinės analizės galimybes gali labai smarkiai išplėsti.

Šiame darbe siekiama praplėsti bibliometrinės analizės ribas ir tokiu būdu vertinti grafų teorijos rodiklius ir sudaryti klasterizavimo modelį, naudojant daug rodiklių. Kuriant metodiką, kuri apima

išvardintus analizės būdus, siekiama išplėsti bibliometrinės analizės tyrimų lauką ir pateikti platesnį požiūrį į šią mokslinių tyrimų sritį.

1.4. Mokslinė darbo vertė

Siūloma nauja bibliometrinės analizės metodika, kuri apima bibliografinių duomenų tyrimą nuo jų paruošimo, sisteminimo, įvairių statistinių metodų taikymo iki galutinio analizės rezultato, kuriuo gaunamos įvairiapusės išvados apie nagrinėjamą duomenų rinkinį.

1.5. Darbo struktūra

1. Antrajame skyriuje „Susijusių darbų analizė“ aprašomos sąvokos, susijusios su tyrimo objektu, aptiriamos bibliografinių duomenų savybės ir ypatumai, nagrinėjami bibliografinių rodiklių apibrėžimai. Aptiriamos socialinių ir bibliografinių tinklų sąvokos ir klasterinės analizės taikymas bibliometrinėje analizėje.
2. Skyriuje „Metodai bibliografinių duomenų analizei“ pateikiamos pagrindinės grafų teorijos sąvokos, pagrindiniai klasterizavimo modelio kūrimo principai ir nagrinėjamos tekstinių duomenų savybės bei aptariami pagrindiniai jų analizės metodai. Galiausiai pristatoma siūloma metodika, kuri remiasi ankstesniuose skyriuose nagrinėtomis temomis.
3. Trečiasis skyrius „Tiriamoji dalis“ skirtas naudojamų duomenų pristatymui ir jų tvarkymo procesui aprašyti. Jame taip pat aprašomas autorių bendradarbiavimo tinklas, jo sudarymo procesas bei analizuojamas žurnalų klasterizavimo modelis.
4. „Išvados“ – paskutinis darbo skyrius, kuriame pateikiamos gautos tyrimo išvados.

2. Susijusių darbų analizė

2.2. Bibliografijos ir bibliometrijos sąvokos

Terminai *bibliografija* ir *bibliometrija* yra panašūs, tačiau prasmė skiriasi. Bibliografija – tai mokslinė disciplina, kuri gali būti apibrėžta dvejopai: kaip organizuotas knygų sąrašas (surašoma bibliografija) arba sistemingas knygų kaip objektų aprašymas (aprašomoji bibliografija). Bibliografiniai įrašai dažniausiai yra bibliotekos katalogo arba kitos duomenų bazės dalis, kurie identifikuoja konkretų leidinį. Juose saugoma svarbi informacija apie šį leidinį: autorius, pavadinimas, raktiniai žodžiai, leidimo metai bei vieta ir pan.

Statistinės bibliografijos apibrėžimas ir tikslas yra taip pat formuluojami keliais variantais. Vienas iš jų yra kaip rašytinės komunikacijos procesų, disciplinos pobūdžio bei eigos apžvalga, skaičiuojant ir analizuojant įvairius rašytinės komunikacijos aspektus. Statistinė bibliografija taip pat gali būti apibrėžta kaip statistinių duomenų apie knygas ir periodinius leidinius rinkimas ir aiškinimas, siekiant parodyti istorinius pokyčius, nustatyti knygų ar žurnalų naudojimą nacionaliniu ar visuotiniu mastu ir daugelyje vietinių situacijų išsiaiškinti, koks yra bendras knygų ir žurnalų naudojimas. Šis terminas praktikoje gali būti lengvai supainiojamas su statistikos terminu. Dėl šios priežasties, įvairiuose literatūros šaltiniuose jis yra keičiamas terminu bibliometrija. Bibliometrijos sąvoką suformulavo A. Pritchard 1969 metais kaip matematinių ir statistinių metodų visuma dokumentų srautams ir jų bibliografinėms charakteristikoms tirti (Pritchard, 1969).

Bibliometrija gali būti apibrėžiama ne tik kaip mokslinis metodas, skirtas produkcijos kiekybinei ir statistinei analizei, bet ir kaip daugiadalykė mokslo sritis, tirianti mokslo ir dokumentų bibliografinės charakteristikas statistiniais ir matematiniais metodais (Petrauskienė ir Aleksandrovė, 2015). Šis terminas išsiskiria tuo, kad nors ir yra naujadaras, turi labai glaudžius ryšius su priimtomis ir analogiškėmis sąvokomis, kaip biometrija, ekonometrija ir scientometrija (Kalantari, Kamsin, Kamaruddin, Ebrahim, Gani, Ebrahimi, ir Shamshirband, 2017).

Bibliometrija praktikoje yra taikoma mokslinių tyrimų veikloje ir vertinant mokslinių tyrimų kokybę, atliekant analizę ir vertinimą ne tik viename ar keliuose universitetuose, bet ir mokslinių tyrimų grupių lygmeniu ar vertinant individualių mokslininkų mokslinę veiklą ar mokslo sritį. Pasaulyje tokie tyrimai bei jų rezultatai naudojami mokslinių tyrimų politikai bei strategijai generuoti bei mokslo vertinimo kriterijams nustatyti, mokslinių tyrimų programoms vertinti, atskiroms mokslo sritims (kryptims)

analizuoti, bendradarbiavimo bei partnerystės kryptims analizuoti bei nustatyti ir kt. (Petrauskienė ir Aleksandrovė, 2015).

Atliekant bibliometrinę analizę, surinktus duomenis apie įvairius leidinius galima nagrinėti remiantis įvairiomis jų specifikomis. Vienos iš pagrindinių analizės krypčių yra šios:

- Leidinio tipas ir kalba. Nagrinėjama, kokia kalba publikuojama daugiausia leidinių tam tikroje tiriamoje mokslo srityje; vertinamas leidinių tipas.
- Leidimo periodiškumas ir geografinis pasiskirstymas. Tiriama, ar publikacijos leidžiamos vienoje šalyje ar keliose.
- Leidinių kategorijų analizė. Gali būti atliekama lyginant skirtingus leidinių tipus vienoje tiriamoje mokslo srityje, lyginant publikacijų skaičių keliose mokslo srityse, autorių publikacijų skaičių kategorijose ir pan.
- Autorių analizė. Vertinamas publikuotų leidinių skaičius tam tikru periodu, tiriama, ar publikacijos buvo parašytos vieno autoriaus, ar bendradarbiaujant keliems, naudotų nuorodų į kitus šaltinius skaičius publikacijose.
- Nagrinėjamų mokslo sričių analizė. Kokios temos yra dažniausiai minimos leidiniuose, kaip kinta publikacijų skaičius laike, kokios yra mokslo srities aktualijos tiriamajame periode.
- Raktinių žodžių analizė. Tiriama jų gausa, svarba konkrečioje mokslo srityje, raktinių žodžių kaita skirtingais laiko periodais.
- Citatų analizė. Cituojamumo rodiklis, analizė pagrįsta kelių veiksnių, tokių kaip autorių skaičius, puslapių ir nuorodų skaičius leidinyje palyginimu.
- Publikacijų vertinimas pagal jų pavadinimus. Publikacijų ir jas publikavusių leidinių pasiskirstymas (Kalantari et al., 2017).

Lentelėje žemiau pateikiami nagrinėti literatūros šaltiniai ir juose naudojamos bibliometrinės analizės kryptys:

1 lentelė. Bibliometrinės analizės kryptys literatūros šaltiniuose

Mokslinio straipsnio autoriai ir publikavimo metai	Geografinis pasiskirstymas	Raktiniai žodžiai	Citavimo rodikliai	Leidinių analizė	Skaičiaus pokytis laike	Mokslinės sritys	Produktyviausi autoriai
Veloutsou, C. ir Mafe, C.R., 2020	✓	✓	✓	✓	✓		✓
Merediz-Sola, I. ir Bariviera, A. F., 2019	✓	✓	✓	✓		✓	✓
Shamszadeh, S., Asgary, S. ir Nosrat, A., 2019	✓	✓	✓	✓	✓		✓
Ming-Huang, W., Yuh-Shan, H. ir Hui-Zhen, F., 2019	✓	✓			✓	✓	
Eyun-Jung, K., Pasadeos, Y. ir Ertem-Eray, T., 2019			✓				
Guthrie, S., Cochrane, G., Deshpande, A., Macaluso, B. ir Larivière, V., 2019	✓					✓	
Vilutienė, T., Kalibatiene, D., Hosseini, M. R., Pellicer, E. ir Zavadskas, E. K., 2019	✓	✓	✓		✓		
Sugimoto, C. R., Ahn, Y., Smith, E., Macaluso, B. ir Larivière, V., 2019						✓	
Liao, H., Tang, M., Luo, L., Li, C., Chiclana, F. ir Zeng, X.-J., 2018	✓	✓	✓	✓	✓		
Mishra, D., Gunasekaran, A., Papadopoulos, T. ir Hazen, B., 2017	✓	✓	✓	✓	✓	✓	
Mishra, D., Gunasekaran, A., Papadopoulos, T. ir Childe, S.J., 2018	✓	✓	✓	✓			

Atsižvelgiant į bibliometrinės analizės kryptis pastebima, jog dažniausiai bibliometrinės analizės rezultatai atskleidžia vieno ar dviejų rodiklių pasikartojančių reikšmių dažnį.

2.3. Bibliografinių duomenų analizė

Statistinės bibliografijos apibrėžimas ir tikslas yra formuluojami keliais variantais. Vienas iš jų yra kaip rašytinės komunikacijos procesų, disciplinos pobūdžio bei eigos apžvalga, skaičiuojant ir analizuojant įvairius rašytinės komunikacijos aspektus. Statistinė bibliografija taip pat gali būti apibrėžta kaip statistinių duomenų apie knygas ir periodinius leidinius rinkimas ir aiškinimas, siekiant parodyti istorinius pokyčius, nustatyti knygų ar žurnalų naudojimą nacionaliniu ar visuotiniu mastu ir daugelyje vietinių situacijų išsiaiškinti, koks yra bendras knygų ir žurnalų naudojimas (Pritchard, 1969).

Nagrinėjant mokslinius straipsnius, kuriuose taikoma bibliometrinė analizė, galima išskirti keletą pagrindinių problemų, į kurias bando atsakyti autoriai. Merediz-Sola ir Bariviera (2019) akcentuoja poreikį

analizuoti literatūros šaltinius, susijusius su gana nauja sritimi – kriptovaliuta. Autoriai pažymi, kad pirmoji publikacija šia tematika pasirodė tik 2012 metais, tačiau per septynerius metus publikacijų skaičius viršijo tūkstantį. Jie kelia tikslą nustatyti, kokia tematika vyrauja šiose publikacijose, kokie autoriai yra lyderiai ir kaip šią mokslinę literatūrą galima grupuoti. Veloutsou ir Mafe (2020) formuluoja poreikį bibliometrinę analizę taikyti mokslinės srities vystymuisi vertinti: kaip kinta publikacijų skaičius, geografinis autorių pasiskirstymas ir aktualumas nagrinėjamu periodu. Liao, Tang, Luo, Li, Chiclana ir Zeng (2018) savo darbe pabrėžia, kad juntamas grafinės bibliografinių duomenų analizės trūkumas.

Analizuojant mokslinius straipsnius, galima pastebėti pagrindinius bibliometrinės analizės tikslus ir kryptis. Merediz-Sola ir Bariviera (2019) nurodo, kad atliekant citatų analizę galima išskirti autorius, kurių darbai dažniausiai cituojami, taip pabrėžiant jų indelį institucijos veikloje arba nagrinėjamoje mokslo srityje. Bibliometrinė analizė taip pat gali padėti suprasti konkrečios tyrimų srities mastą, tendencijas ir kitimą laike. Mishra, Gunasekaran, Papadopoulos ir Childe (2018) pabrėžia bibliometrijos svarbą naujose mokslo srityse, kaip kad didieji duomenys. Jų nuomone, didžiųjų duomenų literatūros analizės apžvalga apie autorius, šalis bei nagrinėjamas temas yra svarbi joje dirbantiems praktikams ir akademikams, skatinant bendradarbiavimą ir šios srities plėtrą.

Bibliografinių duomenų šaltinių yra pakankamai daug, dažniausiai tai specializuotos duomenų bazės. Vienos didžiausių iš jų yra „Web of Science“, „Scopus“, „PubMed“, tačiau bibliografiniai duomenys gali būti surenkami iš leidinių elektroninių puslapių, bibliotekų ar kitų duomenų bazių. Bibliografiniai įrašai atrenkami pagal raktinius žodžius, mokslo kryptis, publikacijų pavadinimus, leidinių specifikacijas ir pan. Taip sudaromi tyrimui naudojami duomenų rinkiniai.

Valdez, Dehmer ir Holzinger (2016) atkreipia dėmesį, kad bibliografinių duomenų rinkiniai pasižymi keletu savybių, kurios gali apsunkinti tyrimą:

- Heterogeniniai duomenų šaltiniai: duomenų integravimo ir duomenų suliejimo poreikis;
- Duomenų sudėtingumas;
- Dideli duomenų rinkiniai;
- Nereikšmingi, neaiškūs, trūkstami, netvarkingi duomenys: būtinas kruopštus išankstinis duomenų apdorojimas.

Vienas iš pagrindinių sunkumų, su kuriais susiduria mokslininkai, analizuodami bibliografinius duomenis, yra pačių duomenų paruošimas analizei.

2.4. Žurnalų cituojamumo rodikliai

Žurnalų cituojamumo rodikliai pradėti naudoti 1975 metais, kuomet Eugene Garfield ir Mokslinės informacijos institutas (angl. *Institute for Scientific Information*) į savo informacinius produktus įtraukė naują komponentą: „Journal Citation Reports“ (JCR). Tai buvo pirmoji išsami informacija žurnalo lygmeniu. Šis naujas informacijos šaltinis pateikė išsamų žurnalų tarpusavio cituojamumo sąsają tinklą, kuris buvo kitų žurnalų cituojamumo ir vertinimo rodiklių atspirties taškas (Larivière ir Sugimoto, 2019).

Dabartiniais laikais yra sukurtas ir naudojamas ne vienas žurnalų cituojamumo rodiklis. Jų kiekis susidarė priklausimai nuo bibliografinių duomenų bazių atsiradimo. Remiantis VILNIUS TECH bibliotekos puslapyje „Žurnalo pasirinkimas publikavimui“ pateikiama informacija, pagrindinėse bibliografinėse duomenų bazėse (WoS ir „Scopus“) pateikiami įvairūs žurnalų cituojamumo rodikliai, tačiau pagrindiniai ir plačiausiai naudojami vertinimo tikslais yra: WoS – *Journal Impact Factor (JIF)*; *Scopus* – *CiteScore*, *Source Normalized Impact per Paper (SNIP)* ir *Scimago Journal Rank (SJR)* rodikliai. Vertinant atskirų žurnalų kokybę, svarbu atkreipti dėmesį į rodiklių skaičiavimo metus. Tai ypatingai svarbu, kuomet remiantis žurnalų cituojamumo ir/arba pozicijos rodikliais yra vertinamos atskiros juose paskelbtos publikacijos, nes šiuo tikslu taikomos publikacijos paskelbimo metus atitinkančios rodiklių vertės. Pagrindinėse bibliografinėse duomenų bazėse (WoS ir „Scopus“) žurnalai skirstomi į kategorijų kvartiles pagal pagrindinių toje bazėje taikomų žurnalų cituojamumo rodiklių vertes (WoS – pagal JIF, „Scopus“ – pagal *CiteScore*) (Pranckutė, 2021).

2.4.1. „Scopus“ citavimo rodikliai

„Scopus“ yra „Elsevier“ (Nyderlanduose įsikūrusi akademinė leidybos įmonė, kurios specializacija yra mokslinis, techninis ir medicininis turinys) santraukų ir citatų duomenų bazė, įkurta 2004 m. „Scopus“ apima beveik 36 377 įrašus (22 794 aktyvius ir 13 583 neaktyvius) iš maždaug 11 678 leidėjų, iš kurių 34 346 yra recenzuojami žurnalai įvairiose mokslo srityse: gyvybės mokslai, socialiniai mokslai, fiziniai mokslai ir sveikatos mokslai. Ji apima trijų tipų šaltinius: knygų serijas, žurnalus ir prekybos žurnalus. Visi žurnalai, įtraukti į „Scopus“ duomenų bazę, kiekvienais metais peržiūrimi, kad jų kokybė būtų pakankamai aukšta, atsižvelgiant į keturių tipų skaitmeninius kiekvieno pavadinimo kokybės matavimus; *h-Index*, *CiteScore*, *SJR (SCImago Journal Rank)* ir *SNIP (Source Normalized Impact per Paper)*:

1. **H-indeksas.** Šis indeksas nurodo *h* publikacijų, pacituotų *h* ir daugiau kartų, skaičių. Bibliometrijoje *h*-indeksas, pagrįstas publikacijų ir citatų skaičiumi šiose publikacijose, laikomas vienu iš svarbių matų, naudojamų individualaus darbo kokybei, poveikiui, įtakai ir svarbai įvertinti. Atlikti tyrimai

parodė h-indekso svarbą skaičiuojant autorių reitingą, universitetų bei žurnalų poveikį (Khurana ir Sharma, 2021). Remiantis VILNIUS TECH bibliotekos duomenimis, h-indekso privalumai yra galimybė kartu įvertinti produktyvumą ir publikacijų kokybę; paprastas apskaičiavimas, kuomet indeksas gali būti apskaičiuojamas ne tik individualiems autoriams, bet ir autorių grupėms, institucijoms, žurnalams ar kitu būdu atrinktiems publikacijų rinkiniams. Tačiau šis rodiklis turi ir trūkumų: orientuotas tik į daugiausiai pacituotas publikacijas, nepalankus jauniems mokslininkams ir naujiems žurnalams, nėra normalizuotas pagal disciplinas.

2. **CiteScore.** Šis žurnalų įtakos rodiklis buvo pristatytas 2016 metais. Rodiklis gaunamas apskaičiuojant konkretaus žurnalo citatų skaičių per vienerius metus pagal straipsnius, paskelbtus per pastaruosius trejus metus. Kitaip tariant, tai yra visų tipų dokumentų citatų vidurkis. *CiteScore* konkrečiam žurnalui metais y yra paskaičiuojama pagal formulę:

$$CS_y = \frac{Citasos_y + Citatos_{y-1} + Citatos_{y-2} + Citatos_{y-3}}{Publikacijos_{y-1} + Publikacijos_{y-2} + Publikacijos_{y-3}} \quad (1)$$

Taigi kiekvieno vertinamo žurnalo *CiteScore* rodiklis keičiasi kiekvienais metais. Remiantis Khosravi, ir Menon (2019), *CiteScore* rodiklis gali būti skirstomas į kvartilius, kurie apibūdina žurnalo rangą. Tačiau plačiau priimta naudoti kvartilius, kurie apskaičiuojami iš kito rodiklio – SJR.

3. **SJR (SCImago Journal Rank).** SJR rodikliu yra apskaičiuojamas žurnalo prestižas pagal žurnalų tarpusavio citavimų tinklus (remiantis cituojamais dokumentais). Dabartinė rodiklio versija, kaip ir kiti „Scopus“ rodikliai, naudoja trejų metų laikotarpį. Šis rodiklis pasižymi šiomis savybėmis: citavimai geresnės kokybės žurnaluose turi didesnę vertę, atsižvelgiama į savicitavimą, žurnalo dydį ir tematinę ryšį. Visi žurnalai, kurie turi SJR rodiklį, yra skirstomi pagal mokslo sritis į 4 kvartilius: Q1, Q2, Q3, Q4. Pirmąjį ketvirtį užima 25% geriausių sąrašo žurnalų; antrąjį ketvirtį užima žurnalai, priklausantys nuo 25% iki 50%; trečiąjį ketvirtį užima žurnalai, priklausantys nuo 50% iki 75%, o ketvirtąjį ketvirtį užima žurnalai nuo 75% iki 100 %.
4. **SNIP (Source Normalized Impact per Paper).** Šis rodiklis pradėtas naudoti 2009 metais ir yra apibūdinamas kaip realaus ir tikėtino (potencialaus) vidutinio citavimų skaičiaus, tenkančio vienai žurnalo publikacijai atstovaujamoje mokslo srityje, santykis. Disciplinos nėra susietos su duomenų bazėje taikoma mokslo sričių klasifikacija, bet nustatomos kiekvienam žurnalui individualiai, remiantis publikacijose cituojamų šaltinių sąrašais. Šis rodiklis pasižymi šiomis savybėmis: atsižvelgiama į citavimų dažnių skirtumus tarp disciplinų, skaičiuojamas 3 ankstesnių metų moksliniams ir apžvalginiams straipsniams bei konferencijų pranešimams, 1 yra vidutinio žurnalo rodiklis. SNIP rodiklis apskaičiuojamas pagal formulę:

$$SNIP = \frac{RIP}{\frac{R}{M}}, \quad (2)$$

čia RIP – neapdorota vienos publikacijos įtaka, R – citavimo potencialas, M – vidutinis duomenų bazės citavimo potencialas.

2.4.2. WoS citavimo rodikliai

„Web of Science“ (WoS) yra svetainė, kuri prenumeratos pagrindu suteikia prieigą prie įvairių duomenų bazių, kuriose pateikiami išsamūs daugelio skirtingų akademinių disciplinų citavimo duomenys. Ši svetainė priklauso Clarivate, kuri yra Jungtinėse Amerikos valstijose įsikūrusi įmonė, kurios veiklos sritys apima bibliometrijos ir scientometrijos sričių analizę, verslo / rinkos tyrimus, farmacijos ir biotechnologijų sričių profiliavimą ir t.t. Mokslo bendruomenėje Clarivate yra žinoma kaip įmonė, kuri apskaičiuoja poveikio faktorių, naudodama duomenis iš savo „Web of Science“ produktų grupės.

„Web of Science“ pagrindinę kolekciją sudaro šešios internetinės indeksavimo duomenų bazės: *Emerging Sources Citation Index*, *Science Citation Index*, *Social Sciences Citation Index*, *Arts & Humanities Citation Index*, *Book Citation Index* ir *Conference Proceedings Citation Index*, kurios kartu apima 321 000 žurnalų, knygų bei konferencijos pranešimų medžiagos. Visi WoS rodikliai skaičiuojami tik žurnalams, įtrauktiems į *Journal Citation Reports (JCR)* (indeksuojamiems *Science Citation Index Expanded (SCIE)* ir *Social Sciences Citation Index (SSCI)*), žurnalai, indeksuojami *Arts & Humanities Citation Index (A&HCI)* ir *Emerging Sources Citation Index (ESCI)* WoS JIF rodiklių neturi. Cituojamumo rodikliai taip pat nėra skaičiuojami kitų tipų šaltiniams (knygoms, konferencijų pranešimams ir kt.). dažniausiai naudojami WoS cituojamumo rodikliai:

1. **IF (Impact Factor).** Žurnalo cituojamumo rodiklis IF yra vidutinis žurnalo straipsnių citavimo tam tikrais metais rodiklis. Jis naudojamas žurnalo svarbai ar reitingui įvertinti, skaičiuojant jo straipsnių citavimo dažnį. Skaičiavimas pagrįstas dvejų metų laikotarpiu: straipsnių citavimo kartų skaičius padalinamas iš straipsnių, kuriuos galima cituoti, skaičiaus:

$$IF_y = \frac{Citas_{y-1} + Citatos_{y-2}}{Publikacijos_{y-1} + Publikacijos_{y-2}} \quad (3)$$

Panašiai kaip ir „Scopus“ duomenų bazėje naudojamas rodiklis SNIP, IF rodiklis taip pat skirstomas į kvartilius. Nustatyti žurnalo reitingą, jo reitingas kategorijoje padalinamas iš viso žurnalų skaičiaus toje kategorijoje. Tokiu būdu žurnalai išreitinguojami į 4 kvartilius: Q1, Q2, Q3, Q4. Pirmąjį ketvirtį užima 25% geriausių sąrašo žurnalų; antrąjį ketvirtį užima žurnalai, priklausantys nuo 25% iki 50%;

trečiąjį ketvirtį užima žurnalai, priklausantys nuo 50% iki 75%, o ketvirtąjį ketvirtį užima žurnalai nuo 75% iki 100 %. Taigi šis rodiklis turi pakankamai panašumų su SNIP rodikliu.

2. **AIF (Aggregate Impact Factor).** Skaičiuojant šį rodiklį, atsižvelgiama į visų kategorijos žurnalų citatų skaičių ir straipsnių iš visų kategorijos žurnalų skaičių. Remiantis Dorta González (2014), AIF rodiklis priklauso nuo dviejų reikšmių: citatų skaičiaus einamaisiais metais, palyginti su bet kuriuo iš straipsnių, paskelbtų tos srities žurnaluose per pastaruosius dvejus metus ir per tuos dvejus metus paskelbtų visų cituojamų elementų skaičiaus. Į rodiklio skaičiavimą įtraukiami cituojami straipsniai, apžvalgos, konferencijų pranešimai ir korespondencija, taip pat pataisymai, redakciniai tekstai, naujienos ir kita medžiaga. AIF mokslo sričiai F ir metams t apskaičiuojamas pagal formulę:

$$AIF_t^F = a_t^F \cdot r_t^F \cdot p_t^F \cdot w_t^F \cdot b_t^F, \quad (4)$$

kur a_t^F yra mokslo srities augimo rodiklis, r_t^F – vidutinis citavimo rodiklis, p_t^F – Journal Citation Reports (JCR) citavimo proporcingumas, w_t^F – Journal Citation Reports (JCR) citavimo proporcingumas nagrinėjamu periodu, b_t^F – citatų proporcingumo koeficientas. Šio rodiklio 1,0 reikšmė parodo, kad vidutiniškai temos kategorijos straipsniai, paskelbti prieš vienerius ar dvejus metus, buvo cituojami vieną kartą.

2.5. Socialinių tinklų analizė

Bibliometrinė analizė yra glaudžiai susijusi su tinklų sudarymu ir analize. Atitinkamai, tinklų sudarymas yra paremtas grafų teorija ir jos taikymu. Tinklų analizė yra mokslas, kuris nagrinėja įvairius tinklus. Tinklai gali būti įvairiausi: telekomunikacijos, kompiuterių, biologiniai, socialiniai. Jie gali būti visai paprasti arba netgi labai sudėtingi, priklausomai nuo to, kiek objektų juos sudaro, kaip tie objektai yra pasiskirstę, kaip jie yra tarpusavyje susiję. Šis mokslas atlieka tyrimus ir testus, bei skaičiavimus, remiantis tam tikromis formulėmis ir įrodymais, taip analizuojant tinklų sandaras, bandant atrasti įvairius dėsningumus tarp skirtingų tinklų ar tinklus sudarančių objektų, atskirų objektų svarbą ir jų tarpusavio priklausomybę.

Socialiniai tinklai nagrinėti jau iki modernių technologijų atsiradimo, tačiau tuo metu buvo sudėtinga atvaizduoti didelius tinklus. Svarbų indėlį į socialinių tinklų analizę atnešė Jokūbas Moreno, kuris 1930-aisiais metais pristatė sociogramas (Van Steen, 2010). Remiantis Visuotine Lietuvių enciklopedija, sociograma tai socialinės grupės narių tarpasmeninių santykių schemos grafinis pavaizdavimas. Sociograma

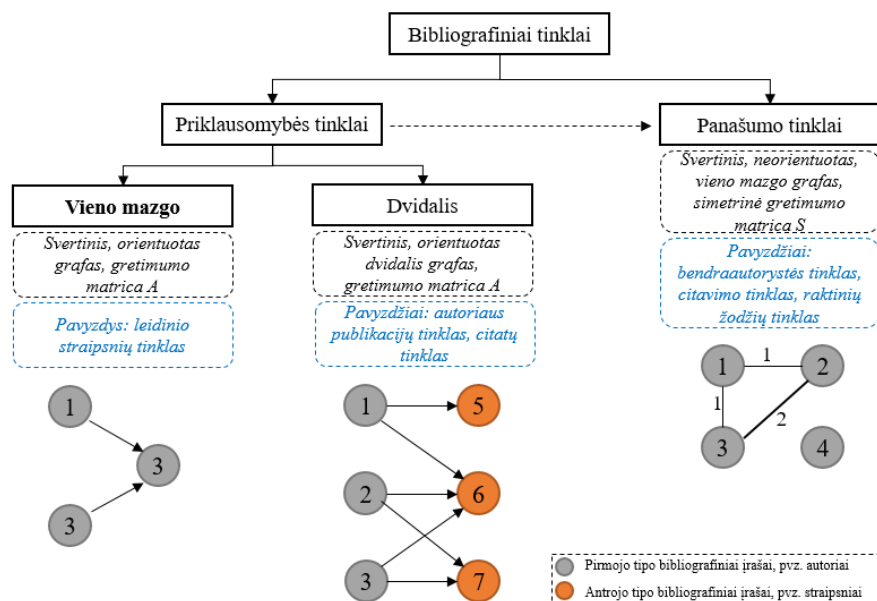
gali atspindėti socialinius santykius, socialinius ryšius, poveikio kanalus, bendravimo ir kitas tam tikros grupės, šeimos, organizacijos, visuomenės ypatybes.

Goldenberg (2021) pastebi, kad socialiniai tinklai skiriasi nuo atsitiktinių matematinių tinklų šiais aspektais:

- Mažo pasaulio fenomenas teigia, kad tikrieji tinklai dažnai turi labai trumpus kelius tarp bet kurių viršūnių. Tai taikoma tikriems ir virtualiems socialiniams tinklams (šešių rankų paspaudimų teorija) ir fiziniams tinklams, tokiems kaip oro uostai.
- Mažo mastelio tinklai sudaryti iš daugumos viršūnių, turinčių mažą laipsnį, bei iš kelių viršūnių su dideliu laipsniu. Dauguma viršūnių turi vieną ar dvi briaunas, tačiau kelios viršūnės turi daug kaimynų. Tokios viršūnės atspindi visuomenės veikėjus, nuomonės formuotojus.
- Homofilija yra asmenų polinkis bendrauti ir užmegzti ryšius su panašiais į save asmenimis, dėl to kaimynai socialiniame tinkle turi panašias savybes.

2.6. Bibliografiniai tinklai

Bibliometrinė analizė susijusi su bibliografinių objektų tinklais. Šie tinklai apibūdina bibliografinių objektų sąsajas ar panašumus tarp leidinių, autorių, žurnalų, institucijų, šalių ar raktinių žodžių. Jie yra naudojami rasti panašių subjektų grupes ir ryšius su kitais subjektais ar grupėmis (Kammerer, Göster, Reichert ir Pryss, 2021). Minimalistinis straipsnių tinklo pavyzdys:



1 pav. Minimalistinis straipsnių tinklo pavyzdys

Šaltinis: (Kammerer et al., 2021)

Priklausomybės tinklai vaizduojami svertiniu, orientuotu grafiku, kurio viršūnės vaizduoja bet kokio tipo bibliografinius subjektus. Dažniausiai yra dvidaliai priklausomybės tinklai, kurie vaizduojami dvidaliu grafu, t. y. grafe yra tik dviejų tipų viršūnės ir to paties tipo viršūnės tiesiogiai nėra sujungti. Dvidalis grafas gali būti vaizduojamas $m \times n$ dydžio gretimumo matrica A , kurioje m reiškia pirmojo tipo viršūnių skaičių, o n – antrojo tipo viršūnių skaičių. Dvidalių priklausomybės tinklų pavyzdys gali būti autoriaus – leidinių tinklas, kuriame viršūnės vaizduoja autorių arba publikaciją, o braunos atspindi autoriaus ryšį su leidiniu arba atvirkščiai.

Priešingai nuo priklausomybės tinklų, panašumo tinklai yra svertiniai, neorientuoti grafai. Kiekviena viršūnė reiškia to paties tipo objektą, pvz., visos viršūnės žymi autorius. Viršūnės yra sujungtos svertinėmis, neorientuotomis briaunomis, vaizduojančiomis jungiamųjų viršūnių panašumą. Šie panašumai apskaičiuojami naudojant panašumo matą, išreiškiantį dviejų bibliografinių objektų panašumas. Panašumo tinklą atspindi jo simetrinė gretimumo matrica S , kuri yra $m \times m$ dydžio. Kiekvienas S įrašas s_{ij} apibūdina viršūnės i ir viršūnės j panašumą. Priklausomai nuo bibliografinio objekto tipo, gali būti naudojami skirtingi panašumo kriterijai. Panašumo kriterijai autoriams gali būti bendraautorystė. Bendraautorystė tarp dviejų autorių apibrėžiama pagal jų kartu atliktų publikacijų skaičių (Kammerer et al., 2021).

Kuriant bibliografinius tinkus, juos galima vaizduoti ne tik kaip autorių bendraautorystės tinklą arba citavimo ryšius. Furstenau (2021) ir kolegos analizuoja su COVID-19 virusu susijusius mokslinius straipsnius ir nagrinėja, kaip keitėsi analizuojamos temos per tam tikrą laiką. Analizuojamos temos atskiriamos pagal raktinius žodžius, suskirstomos nagrinėjamais periodais ir joms priskiriami svoriai pagal tai, kaip dažnai jos kartojasi nagrinėjamuose straipsniuose. Tokiu būdu gaunamas bibliometrinis tinklas, kuris pasiskirstęs pagal periodą ir parodo temų tarpusavio ryšius bei temų pasikartojimo dažnumą. Vizualiai gana lengva matyti tendencijas, išskirti pagrindines temas ir įvertinti sąsajų stiprumą.

Moosavi, Naenil, Fathollahi-Fard ir Fiore (2021) naginėja straipsnius, susijusius su kriptovaliuta. Bibliografiniai duomenys nagrinėjami sudarant įvairius bibliometrinius tinklus: bendraautorystės, raktinių žodžių, geografinio pasiskirstymo ir citavimo. Įdomus pastebėjimas yra tas, kad autorių tinklas yra sudarytas iš atskirų, tarpusavyje nesijungiančių grafų, nes dalis autorių neturi bendraautorystės ryšių. Tačiau kuomet nagrinėjami raktinių žodžių, geografinio pasiskirstymo ir citavimo tinklai – sukuriamas bendras tinklas, nes šie rodikliai yra susiję tarpusavyje. Tokiu būdu, bibliometrinis tinklas sudaro galimybę matyti, kaip vystosi nauja mokslo sritis ir kaip kuriasi autorių bendradarbiavimo tinklas.

2.7. Klasterinės analizės taikymas bibliometrinėje analizėje

Klasterinė analizė, remiantis Stabingiene (2014), yra daugiamačis statistinis metodas, identifikuojantis egzistuojančias grupes objektų aibėje, kurioje kiekvienas objektas yra nusakomas požymių vektoriumi. Visuotinėje Lietuvių enciklopedijoje klasteris yra apibūdinamas kaip autonominė tarpusavyje susijusių duomenų, kompiuterio disko sektorių arba įrenginių grupė. Klasterio viduje skirtumai tarp objektų turi būti kuo mažesni, o tarp klasterių – kuo didesni. Klasteriai gali būti apibūdinami keliomis pagrindinėmis savybėmis:

1. Jie yra nebūtinai vienodo didumo;
2. Objektai į klasterį patenka pagal tam tikrus panašumus;
3. Klasterio forma priklauso nuo to, pagal kokius požymius klasterizuojama;
4. Klasteryje objektai yra panašesni vieni į kitus, o tarp klasterių – labiau skirtingi (Stabingienė, 2014).

Klasterizavimas yra sutinkamas ir nagrinėjant mokslinius straipsnius bibliometrinės analizės tema. Chien, Wang, Kan ir Su (2019) nagrinėja publikacijas, kurių antraštėse naudojami medicininiai terminai, ir taiko duomenų klasterizavimą bei sudaro bibliometrinį tinklą. Kiekvienas iš nagrinėjamų straipsnių buvo priskirtas konkrečiam straipsniui naudojant didžiausio tikėtino įvertinimo metodą. Vizualiai pateikus 10 didžiausių klasterių, galima matyti antraščių raktinių žodžių ir jų temų tinklą. Bazm, Kalantar ir Mirzaei (2016) naudoja panašią metodiką ir klasterizuoja raktinius žodžius, tačiau grafiniame tinklo vaizdavime, kiekviena viršūnė priskiriama ne tik prie konkretaus klasterio, tačiau ir vaizduojama konkretaus dydžio – taip pažymimas raktinių žodžių naudojimo dažnumas. Raktinių žodžių naudojimo dažnumas taip pat vaizduojamas kaip tankio grafikas: nebe išskiriami viršūnių dydžiai, tačiau jos priskiriamos naudojimo dažniui pagal spalvas.

Bibliometrinio tinklo klasterizavimas gali būti naudojama kaip alternatyva tinklo analizei. Waltman, Van Eck ir Noyons (2010) analizuoja informacijos mokslo tema atliktas publikacijas ir pateikia jų tinklą. Sudarytas tinklas suteikia vizualią informaciją ir aiškų supratimą, kaip jis suskirstytas į klasterius. Autoriai taip pat pateikia ir kiekvieno iš klasterių analizę su pagrindiniais autoriais ir temomis bei publikacijų skaičiumi. Tokia kombinuota klasterinė analizė papildo bibliometrinę analizę. Klasterizavimas bibliometrinėje analizėje taip pat gali būti taikomas ne tik tinklams klasterizuoti: Moschini, Fenialdi, Daraio, Ruocco ir Molinari (2020) pritaiko *k*-vidurkių (angl. *K-Means*) algoritmą vertinant Italijos autorių tyrimų rezultatų tarpdiscipliniškumą. Apskaičiavus įvairius bibliografinius rodiklius, autoriai pagal juos yra paskirstomi į klasterius. Toks metodas padeda įžvelgti skirtingus autorių profilius ir bibliografinių rodiklių pasiskirstymą.

3. Metodai bibliografinių duomenų analizei

3.2. Grafų teorija ir jos taikymas tinklų analizėje

3.2.1. Pagrindinės sąvokos

Nagrinėjant bibliografinius duomenis, dažnu atveju yra kuriami ir analizuojami tinklai. Nagrinėjant literatūros šaltinius, susijusius su šio darbo tematika, pastebėta, kad bibliometriniai tinklai yra sudaromi labai dažnai, tačiau jų rodikliai nėra detalios nagrinėjami. Siekiant atlikti bibliometrinę analizę išsamiai, buvo nuspręsta, kad detali tinklų analizė yra svarbi kryptis. Sudarytiems tinklams vertinti, yra labai svarbu suprasti grafų teoriją, kuria ir yra paremta tinklų analizė. Žemiau pateikiamos pagrindinės grafų teorijos sąvokos:

Tegul turime multigrafo viršūnių aibę $V = \{v_1, v_2, \dots, v_n\}$ (angl. *vertex*). Tokio multigrafo lankus galima apibrėžti taip: $l = (v_i, v_j, z_k) \in V \times V \times N$. Šioje išraiškoje v_i yra lanko pradžia, v_j – lanko galas, z_k – lanko numeris, N – natūraliųjų skaičių aibė. Lankas, kurio pradžia ir galas sutampa, vadinamas kilpa. Multigrafas be kilpų ir lygiagrečiųjų lankų vadinamas paprastuoju orientuotuoju grafu. Paprastuoju grafo lankų numeruoti nėra būtina ir jie sudaro aibės $V \times V$ poaibį. Sąryšis $L \subset V \times V$ vadinamas antirefleksyviuoju, kai $(v, v) \notin L \forall v \in V$. Paprastas orientuotasis grafas G apibrėžiamas kaip aibių pora $G = (V, L)$, kai $L \subset V \times V$ yra antirefleksyvusis sąryšis. Tarkime, kad paminėtas paprastas orientuotasis grafas G yra simetrinis. Tada dviejų lankų pora $\{(v_i, v_j), (v_j, v_i)\}$ pakeičiama viršūnių v_i ir v_j pora $\{v_i, v_j\}$ ir vadinama grafo briauna (angl. *edge*). Tokiu atveju visų briaunų aibė yra:

$$E = \{\{v_i, v_j\}, v_i, v_j \in V\} \subset V^{(2)}, \quad (5)$$

$$V^{(2)} = \{\{v_1, v_2\}, \{v_1, v_3\}, \dots, \{v_{n-1}, v_n\}\}. \quad (6)$$

Grafas $G = (V, E)$ vadinamas paprastuoju neorientuotuoju grafu (Krylovas, 2004). Paprasčiausias grafo pavyzdys yra šalies kelių žemėlapis: miestai ir gyvenvietės sudaro viršūnių aibę, o keliai – briaunų aibę.

briaunų aibę $E = \{e_1, e_2, \dots, e_k\}$ (angl. *edge*). Grafu G vadinama viršūnių ir briaunų aibių pora $e_j = (v_{1j}, v_{2j})$ ir žymima $G = (V, E)$ (Krylovas, 2004). Paprasčiausias grafo pavyzdys yra šalies kelių žemėlapis: miestai ir gyvenvietės sudaro viršūnių aibę, o keliai – briaunų aibę.

Grafo $G = (V, E)$ viršūnės $v \in V$ aplinka yra vadinama visa jai gretimų viršūnių aibė:

$$\Gamma(v) = \{\omega \in V: \{v, \omega\} \in E\}. \quad (7)$$

Paprastojo neorientuoto grafo $G = (V, E)$ gretimumo matrica yra simetrinė ir turi nulinę pagrindinę įstrižainę. Gretimumo matrica $S = \|a_{ij}\|_{n \times n}$ yra apibrėžiama:

$$a_{ij} = \begin{cases} 1, & \{v_i, v_j\} \in E, \\ 0, & \{v_i, v_j\} \notin E. \end{cases} \quad (8)$$

Briaunų skaičius $m = |E|$ yra lygus (Krylovas, 2004):

$$m = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n a_{ij}. \quad (9)$$

Matricos S i -osios eilutės nenuliniai elementai apibrėžia viršūnes v_j , į kurias galima patekti iš v_i viršūnės. Atitinkamai, j -ojo stulpelio nenuliniai elementai apibrėžia viršūnes v_i , iš kurių galima patekti į v_j .

Grafo briaunoms gali būti priskirti realūs skaičiai, įvertinantys atstumą, laiką, svorį ir panašius požymius. Toks grafas yra vadinamas svertiniu. Briaunos $e_j \in E$ įvertis (svoris) žymimas $w(e_j)$ (angl. *weight*) (*Algoritmai keičiantys pasaulį*, 2014).

Grafų teorija plačiai taikoma šiuolaikiniame pasaulyje sprendžiant įvairias problemas:

- Grafų spalvinimo uždaviniai (taikomi kartografijoje);
- Trumpiausio kelio paieška;
- Socialinių ryšių nustatymui;
- Įvairių tinklų analizei;
- Tvarkaraščių sudarymas tarp kelių oro uostų ar kitų bendradarbiaujančių įstaigų;
- Neuroninių tinklų uždaviniuose.

3.2.2. *Grafų teorijos taikymas tinklų analizėje*

Nagrinėjant socialinius tinklus, galima išskirti keletą svarbių rodiklių, kurie leidžia geriau suprasti tinklo sandarą ir jo išskirtinumus: viršūnių laipsniai, grafo skersmuo, atstumas tarp grafo viršūnių, centriškumo matai. Tai yra grafų teorijos taikymas tinklų analizėje ir pritaikoma šiame darbe nagrinėjant bibliografinius duomenis. Toliau pateikiami tinklo sandaros rodikliai pagal Beineke, Oellermann ir Pippert (2002):

1. **Viršūnių laipsniai.** Iš grafo viršūnės v išeinančių (įeinančių) briaunų skaičius vadinamas viršūnės valentingumu arba laipsniu. Skaičius $p(v) = |\Gamma(v)|$ yra vadinamas viršūnės v laipsniu (Krylovas,

2004). Nagrinėjant viršūnių laipsnius galima išskirti svarbiausius socialinio tinklo narius. Taip pat tai padeda nagrinėti tinklo struktūrą: jei daugumos viršūnių laipsniai yra panašūs, galima daryti išvadą, kad tokiame socialiniame tinkle nariai užima panašias roles. Tačiau jei yra išsiskiriančių viršūnių, remiantis jomis galima atlikti tinklo klasterizavimą arba skaidymą į mažesnius.

1.1. Viršūnių laipsnių pasiskirstymas. Tai yra viršūnių skaičius, kurios turi tą patį laipsnį. Viršūnių laipsnių pasiskirstymas dažnai vaizduojamas histograma arba viršūnių laipsnių dalimi visų viršūnių aibėje.

1.2. Laipsnių koreliacija. Tai yra laipsnių koreliacija tarp atitinkamų dviejų gretimų viršūnių laipsnių. Šis rodiklis parodo, viršūnių laipsnių tarpusavio koreliaciją. Laipsnio koreliacija dažnai paskaičiuojama naudojant koreliacijos koeficientą. Tegul x ir y yra du stochastiniai kintamieji, kuriems turima eilė stebėjimo porų $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$. Koreliacijos koeficientas $r(x, y)$ tarp x ir y apibrėžiamas taip:

$$r(x, y) = \frac{\sum_{i=1}^n ((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}, \quad (10)$$

kur $\bar{x}$ yra kintamojo x reikšmių vidurkis, o $\bar{y}$ – kintamojo y reikšmių vidurkis.

2. Grafo metrinės charakteristikos. Tai yra kelio ilgis, atstumas tarp grafo viršūnių, ir grafo skersmuo.

2.1. Kelio ilgis. Baigtinė seka, sudaryta iš grafo $G = (V, E)$ viršūnių ir briaunų $v_{i_0}, e_{i_1}, v_{i_1}, e_{i_2}, \dots, v_{i_k}, e_{i_k}, v_{i_k}$, taip, kad kiekviena briauna $e_{i_j} = \{v_{i_{j-1}}, v_{i_j}\}$, vadinama maršrutu. Maršrutas, kurio visos briaunos yra skirtingos, vadinamas grandine. Atviroji grandinė yra vadinama keliu. Kelio ilgiu vadinamas į jį įeinančių briaunų skaičius Krylovas (2004).

2.2. Atstumas tarp grafo viršūnių. Tai yra trumpiausias šias viršūnes jungiantis kelio ilgis. Tarkime, kad $u, w \in V$ – dvi kurios nors grafo $G = (V, E)$ viršūnės. Sunumeruojami visi jas jungiantys keliai: $P_k = (u, v_{i_1}, v_{i_2}, \dots, v_{i_{l_k}}, w)$. Atstumu tarp grafo viršūnių vadinamas trumpiausio jas jungiančio kelio ilgis Krylovas (2004):

$$p(u, w) = \min_k |P_k(u, \dots, w)|. \quad (11)$$

2.3. Grafo skersmuo – tai maksimalus atstumas tarp grafo viršūnių. Krylovas (2004) pateikia grafo skersmens formulę:

$$d(G) = \max_{u, w \in V} p(u, w). \quad (12)$$

3. Centriškumas. Centriškumo skaičiavimas gali būti suprantamas kaip svarbiausiųjų objektų tinkle ieškojimas. Šis terminas atsirado būtent socialinių tinklų analizėje. Centriškumas gali padėti rasti daug

naudingos informacijos: labiausiai įtakingus žmones socialiniame tinkle, populiariausius internetinius puslapius iš tam tikros jų grupės, kaip plinta virusai (surandant pagrindinius šaltinius). Surasti svarbiausius objektus tinkle galima įvairiais metodais, pagal įvairius kriterijus. Tam yra skirti centriškumo matai. Viršūnės centriškumas apskaičiuojamas pagal formulę:

$$C_C = \sum_{k=0}^{\infty} \frac{(S\beta)^k}{k!}, \quad (13)$$

kur k – kelio ilgis, S – gretimumo matrica, β – nuolaidos parametras, užtikrinantis sumos konvergenciją. Centriškumo matai yra skirti rasti labiausiai centre esančius objektus pagal tam tikrus kriterijus.

3.1. Laipsnio centriškumo matas – naudojamas, kai norima socialiniame tinkle atrasti žmones, kurie yra svarbiausieji esantys tinkle. Šis centriškumo matas yra vienas paprasčiausių, nes jis atitinka viršūnės v laipsnį:

$$C_D(v) = \deg(v). \quad (14)$$

Skaičiuojant laipsnių centriškumo matą, galimas verčių normalizavimas, jas padalinus iš didžiausio galimo viršūnės laipsnio, esančio grafe G .

3.2. Artumo centriškumo matas – remiasi tuo, jog svarbiausieji objektai tinkle yra tie, kurie arčiausiai kitų objektų. Svarbiausi ne tie, kurie turi daug kaimynų, bet reikia atsižvelgti ir į kaimynų kaimynus ir dar tolimesnius ryšius. Siekiama rasti trumpiausius kelius nuo objekto iki visų kitų objektų su kuriuo jis yra kaimynas ir lyginant trumpiausius rastus kelius, rasti kurie objektai tame tinkle turi daugiau artimiausių kaimynų už kitus taip sužinant, kuris objektas tinkle, remiantis šiuo matu, yra svarbiausias. Artumo centriškumas, viršūnei u , apskaičiuojamas pagal formulę:

$$C_A(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v, u)}, \quad (15)$$

kur $d(v, u)$ – trumpiausias atstumas tarp viršūnių v ir u , n – viršūnių skaičius, kurias galima pasiekti iš viršūnės u .

3.3. Tarpusavio centriškumo matas. Svarbiausiais tinkle laiko tuos objektus, kurie turi daugiausiai trumpiausių ryšių su kitais objektais. Naudojantis šiuo metu, tinkle svarbiausi objektai yra tie, kurie yra lyg tiltai, jungia daug aplinkui esančių objektų ir jų grupes (kaimynai irgi yra svarbūs). Tarpusavio centriškumas, viršūnei v , apskaičiuojamas pagal formulę (Mazeliauskas, 2016):

$$C_T(v) = \sum_{s,t \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}, \quad (16)$$

kur V – grafo viršūnių aibė, $\sigma(s, t)$ – nesikartojančių trumpiausių kelių nuo viršūnės s iki viršūnės t skaičius, $\sigma(s, t|v)$ – trumpiausių kelių nuo viršūnės s iki viršūnės t , einančių per viršūnę v , skaičius.

4. **Grafo tankumas.** Grafo tankis parodo santykį tarp grafe esančių briaunų ir didžiausio įmanomo grafo briaunų skaičiaus. Konceptualiai tai apibrėžia, koks tankus yra grafas, vetinant briaunų ryšį. Ieškant grafo tankumo, briaunos, priklausančios grafui $G = (V, E)$ yra padalinamos iš didžiausio įmanomo briaunų skaičiaus:

$$\text{Neorientuoto grafo tankumas: } Den_{Neor.} = \frac{2|E|}{|V|*(|V|-1)} \quad (17)$$

$$\text{Orientuoto grafo tankumas: } Den_{Or.} = \frac{|E|}{|V|*(|V|-1)} \quad (18)$$

5. **Grafo vidutiniai jungumo matai.** Geriausiai žinomas ir naudingiausias grafo matas, norint įvertinti kaip gerai grafo viršūnės yra sujungtos, yra jungumas. Jis apibrėžiamas kaip minimalus viršūnių skaičius, kurį ištrynus grafikas tampa nebejungus. Yra keletas rodiklių, kurie gali apibūdinti grafo jungumą:

- 5.1. **Vidutinis viršūnių jungumas.** Dvi grafo G viršūnės u ir v laikomos sujungtos k dydžiu, jei egzistuoja k arba daugiau tiesiogiai nesusijusių kelių tarp šių viršūnių. Ši viršūnių u ir v juntis žymima $\kappa_G(u, v)$ ir reiškia didžiausią galimą k reikšmę, kuria u ir v yra sujungtos. Grafo G , kurio eilė yra n (n atitinka viršūnių skaičių), vidutinis viršūnių jungumas užrašomas taip (Beineke et al., 2002):

$$\bar{\kappa}(G) = \frac{\sum_{u,v} \kappa_G(u, v)}{\binom{n}{2}}. \quad (19)$$

- 5.2. **Vidutinis laipsnių jungumas.** Tai yra viršūnių, turinčių k laipsnį, atimiamųjų kaimynų vidutinis laipsnis.

- 5.3. **Vidutinis kaimyninių viršūnių laipsnis.** Neorientuoto grafo atveju, viršūnės kaimynais laikomos tos viršūnės, kurios yra tiesiogiai sujungtos briauna. Orientuoto grafu atveju yra vertinama jungianšios briaunos kryptis. Vidutinis viršūnės v , priklausančios grafui G , kaimyninių viršūnių laipsnis apskaičiuojamas:

$$k_{nn,v} = \frac{1}{|N(v)|} \sum_{j \in N(v)} k_j. \quad (20)$$

3.3. Bendruomenės sąvoka

Nagrinėjant įvairius tinklus, galima pastebėti, kad vertinant bendrąjį jų vaizdą, jie gali atrodyti reti, tačiau tam tikros tinklo dalys yra susitelkusios tankiau. Tokios tankiau susitelkusių viršūnių grupės yra vadinamos bendruomenėmis (angl. *communities*). Vienas iš pagrindinių bendruomenių bruožų yra tai, kad joms priklausančios viršūnės yra stipriai susijusios vienos su kitomis, tačiau turi nedaug jungčių su likusiomis tinklo viršūnėmis. Tokia struktūra suteikia daug informacijos apie visą tinklą: medžiagų apykaitos tinkle bendruomenės atitinka biologines ląstelės funkcijas, interneto tinkle bendruomenės atitinka dominančias temas (Pons ir Latapy, 2005).

Šių autorių teigimu, bendruomenės sąvoka, nagrinėjant socialinius tinklus, buvo plačiai nagrinėjama ir buvo pateikta nemažai įvairių jos apibrėžimų. Autoriai savo darbe teigia, kad tinkama bendruomenės struktūra pasiekama, kuomet grafo $G = (V, E)$ ($\forall i, C_i \subseteq V$) viršūnių atskyrimo $P = \{C_1, \dots, C_k\}$ proporcija C_1 viršūnių grupės viduje yra didesnė, palyginus su viršūnių atskyrimo proporcija tarp šių grupių (Pons ir Latapy, 2005).

Atliekant tyrimą su didelės apimties duomenų kiekiu, jam atvaizduoti reikalingas didelis viršūnių ir briaunų skaičius. Šiai problemai spręsti naudojami grupavimo arba klasterizavimo metodai, kurie leidžia pagal įvairius parametrus išskirti panašias arba susijusias nagrinėjamų duomenų grupes ir atvaizduoti jas spalvomis (Jasiukevičius, 2016). Analizuojant literatūrą bei programavimo kalbas, randama nemažai metodų, kurie taikomi bendruomenėms tinkluose rasti:

1. Kernighan – Lin algoritmas, kuris padalina tinklą į dvi dalis;
2. k -klikų metodas, kuris randa k dydžio klikas grafe;
3. Medžio skaidymo algoritmai;
4. Liuveno bendruomenės aptikimo algoritmas;
5. Atsitiktinio vaikščiojimo (angl. *Random Walk*) metodas;
6. Girvan – Newman algoritmas, kuris paremtas tinklo centrinėmis metrikomis.

Šių metodų yra ir daugiau, tačiau toliau plačiau nagrinėjamas Girvan – Newman algoritmas. Jasiukevičiaus (2016) suformuotas metodo tikslas yra identifikuoti briaunas siejančias atskiras organizacines grupes. Šioms jungiančiosioms briaunoms rasti yra naudojamos centrinės metrikos, kurios išreiškia mazgo svarbumą trumpiausiam kelyje. Naudojant šį metodą, grafo struktūra vertinama modulinio principu. Grafas $G = (V, E)$ yra padalinamas į m grupių (bendruomenių) ir tokio tinklo struktūros modalumą galime išreikšti formule:

$$Q = \sum_{s=1}^m \left[\frac{l_s}{E} - \left(\frac{d_s}{2|E|} \right)^2 \right], \quad (21)$$

kur l_s yra briaunų skaičius tarp dviejų viršūnių, priklausančių s grupei, E – grafo G visų briaunų aibė, d_s – viršūnių, priklausančių s grupei, laipsnių suma, m – maksimalus grupių skaičius visoje tinklo struktūroje.

Aukšta Q reikšmė atitinka didelę l_s reikšmę kiekvienai bendruomenei, kurios viršūnės tarpusavyje yra sujungtos stipriau, nei su viršūnėmis už bendruomenės ribų. Girvan – Newman klasterizavimo algoritmo veikimo principas yra:

1. Surandami vis grafo kritiniai keliai;
2. Apskaičiuojamas tarpusavio centriškumo matas (angl. *betweenness centrality*);
3. Pašalinamos briaunos, kurių centriškumo mato reikšmė didžiausia;
4. Perskaičiuojamas tarpusavio centriškumo matas;
5. Patikrinama ar grafą dar galima skaidyti toliau. Jei taip, kartojami 2 – 4 žingsniai, jei ne – gaunamas galutinis rezultatas (Jasiukevičius, 2016).

3.4. Tinklo grafinis vaizdavimas

Grafinis tinklo arba grafo vaizdavimas yra neatsiejama analizės dalis. Didelės apimties tinklai geriausiai suvokiami, kuomet yra pateikiami vizualiai. Tokiu būdu galima matyti susidariusias bendruomenes, išskirtines viršūnes, dominuojančius ryšius ir t.t. Noack (2018) teigimu, dirbant su socialiniais tinklais, juose svarbu išskirti bendruomenes, kurių viršūnių tarpusavio ryšiai yra stiprūs, tačiau silpnėsni, lyginant su kitomis tinklo viršūnėmis. Autoriaus teigimu, siekiant tinkle įžvelgti bendruomenes, galima naudoti išdėstymą, kuris atitinka energijos modelį arba sąveikaujančių jėgų sistemą. Sąveikaujančių jėgų sistema vaizduoja porines traukos ir atostūmio jėgas tarp viršūnių.

Tinklo grafiniam vaizdavimui pasirinkta Gephi sistema, kuri yra pritaikyta vizualizuoti didelės apimties tinklus. Kuriant tinklą, į sistemą yra įkeliami paruošti duomenų rinkiniai ir sugeneruojamas grafas. Jam atvaizduoti galima pasirinkti daug išdėstymo variantų ir juose keisti įvairius nustatymus. Kadangi tinklą sudaro daug viršūnių ir jas jungiančių briaunų, buvo ieškoma tokio išdėstymo, kuris tiksliausiai atspindėtų gautą autorių bendradarbiavimo tinklo struktūrą.

Tinkamiausias rezultatas buvo pasiektas naudojant „ForceAtlas2“ jėgos veikimo principu paremtą išdėstymą: tinklo išdėstymui jis imituoja fizinę sistemą. Viršūnės stumia vienos kitas kaip įkrautos dalelės, o briaunos traukia viršūnes, kaip spyruoklės. Šios jėgos sukuria judėjimą, kuris susilieja į subalansuotą

būseną, kurios galutinė konfigūracija padeda interpretuoti duomenis. Kiekviena viršūnė tinkle vaizduojama priklausomai nuo to, kokią ryšį ji turi su kitomis viršūnėmis. Kitaip sakant, šis procesas priklauso tik nuo jungčių tarp viršūnių. Šios technikos pranašumas yra galimybė vizualiai interpretuoti struktūrą. Pati jos esmė yra paversti struktūrinius artumus į vizualinį artumą, palengvinant analizę ir ypač socialinių tinklų analizę.

Kiekvienas jėgos veikimo principu paremtas algoritmas remiasi tam tikra traukos ir tam tikra atostūmio jėgos formule. „ForceAtlas2“ išdėstymas yra realaus gyvenimo įkvėptas modeliavimas. Jacomy, Venturini, Heymann ir Bastian (2014) savo darbe teigia, kad šiame išdėstyme naudojama įelektrintų dalelių atostūmio formulė ir traukos formulė, apimanti geometrinį atstumą tarp dviejų viršūnių. Traukos jėga F_a (angl. *attraction force*) tarp dviejų sujungtų viršūnių n_1 ir n_2 tiesiškai priklauso nuo atstumo $d(n_1, n_2)$:

$$F_a(n_1, n_2) = d(n_1, n_2). \quad (22)$$

Siekiant geresnio tinklo vizualinio atvaizdavimo, norima priartinti silpniau sujungtas viršūnes prie stiprius ryšius turinčių viršūnių. Dėl to yra pakoreguojama atostūmio jėga taip, kad ji būtų silpnesnė tarp stiprius ryšius turinčios viršūnės ir silpniau sujungtos viršūnės. Dėl to jos atsiduria arčiau viena kitos pasiekus subalansuotą būseną. Atostūmio jėga F_r (angl. *repulsion force*) yra proporcinga dviejų viršūnių laipsniams plius vienas rezultatui ($\deg + 1$):

$$F_r(n_1, n_2) = k_r \frac{(\deg(n_1) + 1)(\deg(n_2) + 1)}{d(n_1, n_2)} \quad (23)$$

„ForceAtlas2“ išdėstymas turi nustatymus, kuriuos tinklo kūrėjas gali keisti, priklausomai nuo siekiamo rezultato:

- **LinLog mode (LinLog režimas).** Šio nustatymo aktyvavimas suteikia viršūnėms vietas, kurios yra įskaitomiausios, nes jis viršūnėms parenka tokias vietas, kurios atitinka Newmano moduliškumą, plačiai naudojamą bendruomenės struktūros matą. LinLog režimas naudoja logaritminę traukos jėgą:

$$F_r(n_1, n_2) = \log(1 + d(n_1, n_2)) \quad (24)$$

- **Gravity (gravitacija).** Gravitacija yra dažnai sutinkamas rodiklis, naudojant jėgos veikimo principu paremtus išdėstymus. Ši jėga $F_g(n)$ neleidžia nesujungtoms viršūnėms nutolti. Ji pritraukia viršūnes prie tinklo centro. Jos pagrindinis tikslas yra kompensuoti viršūnių, esančių toli nuo centro, atostūmį:

$$F_g(n) = k_g(\deg(n) + 1). \quad (25)$$

k_g koeficientas pasirenkamas tinklo kūrėjo. Šios jėgos trūkumas yra tas, kad ji yra tokia stipri, kad kartais yra stipresnė už kitas jėgas. Dėl to viršūnės gali būti išdėstytos šališkai. Tačiau jos pranašumas yra labai kompaktiškas išdėstymas, kuris gali būti naudingas tam tikrais atvejais.

- **Scaling (mastelis).** Jėgos veikimo principu paremtis išdėstymai turi keletą konstantų, kurios atlieka priešingą vaidmenį tinklo vaizdavime. Siekiant išvengti pasirinkimo, kuomet pasirenkami du priešingi koeficientai, buvo įvestas mastelio parametras. Kuo didesnė reikšmė parenkama, tuo didesnis tinklas bus atvaizduotas.
- **Edge weight (briaunos svoris).** Jei kuriant tinklą yra nurodomi briaunų svoriai, jie bus naudojami tinklo vaizdavime. Tai gali turėti nepaneigiamos įtakos tinklo analizėje. Jei δ parametro reikšmė lygi 0 – briaunų svoriai ignoruojami, jei 1 – briaunų svoriai vaizduojami pagal tikrąsias jų vertes, jei > 1 – briaunų svoriai yra atitinkamai sustiprinami. Šis parametras naudojamas modifikuoti traukos jėgą pagal briaunos svorį:

$$F_a = w(e)^\delta d(n_1, n_2). \quad (26)$$

- **Dissuade Hubs (centrų atskyrimas).** Aktyvavus šį parametą, jis keičia tinklo formą dalindamas kiekvienos viršūnės traukos jėgą iš tos viršūnės laipsnio +1:

$$F_a(n_1, n_2) = \frac{d(n_1, n_2)}{\deg(n_1) + 1} \quad (27)$$

Šis parametras skirtas suteikti viršūnėms, turinčioms didelį įeinančių briaunų laipsnį centrinę padėtį, lyginant su viršūnėmis, turinčioms didelį įeinančių briaunų laipsnį.

- **Prevent Overlapping (persidengimo išvengimas).** Aktyvavus šį parametą, atostūmio jėga modifikuojama taip, kad viršūnės nepersidengtų. Šio parametro tikslas – sukurti įskaitomesnį ir estetiškesnį tinklo vaizdą. Idėja yra atsižvelgti į viršūnės dydį $size(n)$, apskaičiuojant atstumą $d(n_1, n_2)$ tiek traukos, tiek atostūmio jėgos atžvilgiu. Atstumas tarp dvejų viršūnių joms nepersidengiant yra:

$$d'(n_1, n_2) = d(n_1, n_2) - size(n_1) - size(n_2). \quad (28)$$

Jei $d'(n_1, n_2) > 0$, persidengimo nėra ir skaičiuojant traukos F_a bei atostūmio F_r jėgas, naudojamas d' vietoje d ;

$$F_a(n_1, n_2) = d'(n_1, n_2) \quad (29)$$

$$F_r(n_1, n_2) = k_r \frac{(\deg(n_1) + 1)(\deg(n_2) + 1)}{d'(n_1, n_2)}. \quad (30)$$

Jei $d'(n_1, n_2) < 0$, viršūnės persidengia ir traukos jėga F_a prilyginama 0, o atostūmio jėga F_r sustiprinama:

$$F_a(n_1, n_2) = 0 \quad (31)$$

$$F_r(n_1, n_2) = k'_r (\deg(n_1) + 1)(\deg(n_2) + 1). \quad (32)$$

Jei $d'(n_1, n_2) = 0$, tuomet tarp viršūnių nėra nei traukos F_a , nei atostūmio F_r jėgų.

- **Approximate repulsion (apytikslis atostūmis).** Šis parametras skirtas pagerinti erdvinį tinklų atvaizdavimą dideliuose grafuose (Jacomy, et al., 2014).

Kiekvienas sudaromas tinklas yra unikalus savo struktūra. Dėl to konkretūs nustatymai turi būti pasirenkami atsižvelgiant į šią struktūrą. Taip pat, tas pats tinklas gali būti vaizduojamas keliais skirtingais išdėstymo variantais, taip išryškinant konkrečias nagrinėjamas jo dalis.

3.5. k -vidurkių klasterizavimo metodas

Klasterių sudarymo metodų yra daug. Jie skiriami pagal tai, kokia strategija naudojama skirstyti į klasterius, vertinami panašumo matai, pagal kuriuos bus nustatomi objektų tarpusavio panašumai arba skirtumai, atstumų tarp klasterių nustatymo kriterijai. Remiantis Stabingienė (2014), skiriamos dvi pagrindinės klasterinės analizės metodų klasės:

1. Hierarchiniai. Objektai yra laikomi vienu dideliu klasteriu, kurį sudaro mažesni klasteriai, mažesnius dar mažesni ir t.t. Taikant šiuos metodus yra nustatoma klasterių tarpusavio hierarchija.
2. Nehierarchiniai. Metodai taikomi, kai žinomas klasterių skaičius arba šis skaičius yra pasirenkamas. Labai dažnai naudojamas nehierarchinis metodas – k -vidurkių.

k -vidurkių metodas taip pat yra priskiriamas susitelkimu apie centrą (angl. *centroid based*) paremtų klasterizavimo metodų grupei. Jo metu suformuojama k kiekis sferos formos klasterių, o grupėje esančių narių panašumas užtikrinamas iteratyviai mažinant kvadratinį klaidos matą E :

$$E = \sum_{i=1}^k \sum_{j=1}^n |d(x_j - c_i)|^2, \quad (33)$$

kur k yra ieškomų klasterių kiekis, n – analizuojamas objekto požymių skaičius, x_j – analizuojamas objektas, c_i – klasterio centras (Erman, Arlitt ir Mahanti, 2006). Sakykime, kad $X = \{x_1, x_2, \dots, x_n\}$ yra

analizuojamų objektų aibė, o $V = \{v_1, v_2, \dots, v_c\}$ – klasterių centrų aibė. Tuomet k -vidurkių algoritmo žingsniai yra:

- Atsitiktinai parenkama c klasterių centrų;
- Suskaičiuojami atstumai tarp kiekvieno objekto ir klasterio centro;
- Objektas priskiriamas klasteriui, nuo kurio centro yra mažiausiu atstumu;
- Apskaičiuojami nauji klasterių centrai, kur c_i atitinka objektų skaičių i – jame klasteryje:

$$v_1 = \frac{1}{c_i} \sum_{j=1}^{c_i} x_i; \quad (34)$$

- Perskaičiuojami atstumai tarp duomenų objektų ir naujai rastų klasterių centrų;
- Jei atstumai nepasikeitė, tuomet klasteriai stabilizavosi ir algoritmas sustabdomas. Kitu atveju – skaičiavimai kartojami nuo objektų priskyrimo klasteriams žingsnio (Verma, Srivastava, Chack, Diswar ir Gupta, 2012).

3.6. Duomenų normalizavimas

Norint duomenų rinkiniui pritaikyti k -vidurkių algoritmą (arba bet kokią kitą klasterizavimo algoritmą), rekomenduojama duomenis standartizuoti. Borkin, Némethová, Michal'conok ir Maiorov (2019) savo darbe apibūdina šio proceso svarbą: neapdorotų duomenų verčių diapazonas paprastai turi skirtingas skales. Balčiūnas (2019) pabrėžia, kad analizuojant duomenis, svarbi ne tik konkreti stebėjimo reikšmė, bet svarbu ją palyginti su kitomis atliekamo stebėjimo reikšmėmis. Tam, kad būtų galima palyginti tokius duomenis, reikia suvienodinti jų matavimo skales, o tai reiškia – tuos duomenis standartizuoti.

Yra daug duomenų normalizavimo metodų, tačiau vienas populiariausių ir dažniausiai naudojamų praktikoje yra $min - max$ normalizavimas. Borkin (2019) jį apibūdina kaip metodą, pagrįstą duomenų parametrų diapazono keitimu iš originalių verčių į $[0, 1]$ arba $[-1, 1]$ intervalus. Šie intervalai pasirenkami priklausomai nuo duomenų struktūros. $Min - max$ normalizavimas atliekamas pagal formulę:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}, \quad (35)$$

kur x yra originali parametro reikšmė, o x' – reikšmė gauta po normalizavimo.

3.7. Optimalaus klasterių skaičiaus nustatymas

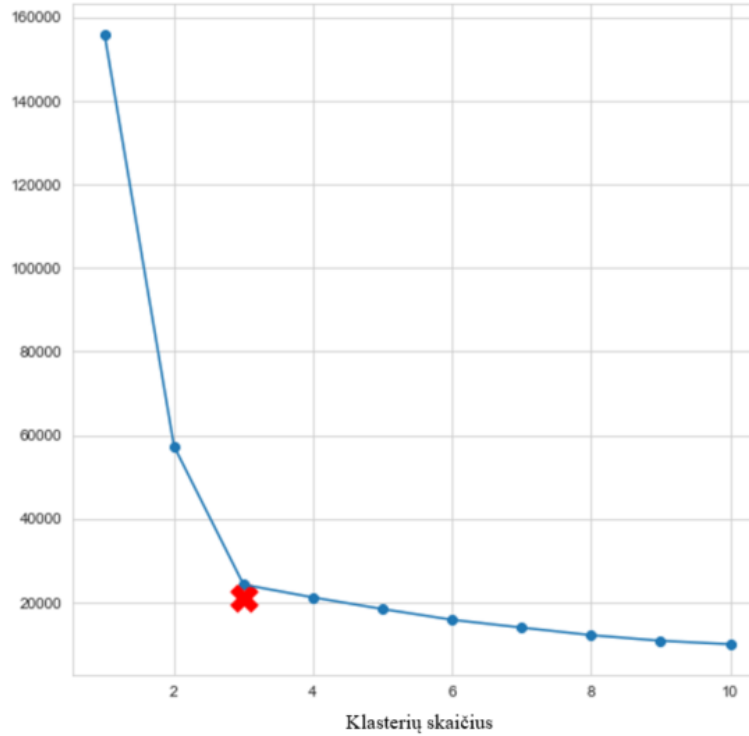
Normalizavus duomenis ir jiems pritaikius k -vidurkių algoritimą, darbas dar nėra baigtas. Įvairiose programavimo kalbose, klasterizavimo modeliai yra jau sukurti ir juos belieka pritaikyti turimam duomenų rinkiniui. Tačiau klasterių skaičius juose ne visada yra parenkamas geriausiai iš pirmojo karto. Dėl to yra rekomenduojama atlikti papildomus testus ir jų pagalba nustatyti optimalų klasterių skaičių. Vienas dažniausiai naudojamų yra vadinamas „alkūnės“ metodas (angl. *Elbow Method*). Putria ir Dwidasmara (2021) apibūdina šį metodą kaip galintį padėti nustatyti klasterių skaičių remiantis klasterių kvadratinės paklaidos sumos mažėjimo ir klasterių tarpusavio ryšio grafiku. Kylant klasterių K skaičiui, grafike vaizduojama kreivė įgaus „alkūnės“ formą, kurios lūžio taškas atitiks susistabilizavusių klasterių skaičių. Remiantis Skrebutėnaite (2021), „alkūnės“ metodas yra paremtas dispersijomis klasteriuose. Parenkamas toks klasterių skaičius, kad dispersijos juose būtų mažiausios:

$$\min \left(\sum_{k=1}^k W(C_k) \right), \quad (36)$$

kur C_k yra k -jį klasterį sudarančių objektų skaičius, $W(C_k)$ – jų dispersija. Autorės teigimu, siekiant pritaikyti „alkūnės“ metodą, turi būti atliekami šie žingsniai:

1. Sudaromas klasterizavimo modelis, naudojant skirtingas k klasterių reikšmes;
2. Kiekvienam pasirinktam k apskaičiuojamos dispersijos;
3. Gauti rezultatai pavaizduojami grafiškai;
4. Vertinama, kurioje grafiko vietoje yra lenkimas, vadinamoji „alkūnė“. Ši vieta laikoma tinkamu klasterių skaičiumi.

Tokio grafiko pavyzdys pateikiamas paveiksle 2:



2 pav. Optimalaus klasterių skaičiaus radimas „alkūnės“ metodu

Šaltinis: (Brus, 2021)

Kitas plačiai taikomas optimalaus klasterių skaičiaus nustatymo metodas yra silueto (angl. *Silhouette*) koeficiento skaičiavimas. Silueto reikšmė yra matas, nurodantis, kiek objektas yra panašus į savo klasterį (sanglauda), palyginti su kitais klasteriais (atskyrimas). Silueto koeficientas taikomas klasterizavimo algoritmų kokybės nustatyme. Šis koeficientas nurodo, kaip gerai kiekvienas taškas buvo priskirtas klasteriui (Paulauskienė ir Kurasova, 2014). Remiantis Skrebutėnaite (2021), vidutiniu silueto metodu randamas vidutinis stebėjimų silueto koeficientas skirtingoms k reikšmėms. Optimalus k skaičius klasteriams parenkamas tas, kurio vidutinis silueto koeficientas yra didžiausias. Vidutinio silueto koeficiento $s(i)$ reikšmė objektui i randama pagal formulę:

$$s(i) = \begin{cases} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} & jei |C_i| > 1, \\ 0 & jei |C_i| = 1. \end{cases} \quad (37)$$

Čia $a(i)$ vidutinis atstumas nuo klasterio C_i elementų iki objekto i , kuris apskaičiuojamas:

$$a(i) = \frac{1}{(|C_i| - 1)} \sum_{j \in C_i, i \neq j} d(i, j), \quad (38)$$

kur $d(i, j)$ yra atstumo matas. $b(i)$ yra vidutinis objekto i atstumas iki artimiausio kaimyninio klasterio C_k , kuris apskaičiuojamas:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j). \quad (39)$$

Silueto koeficiento reikšmės kinta intervale $[-1, 1]$. Kuo didesnė koeficiento $s(i)$ reikšmė, tuo geresnė sanglauda ir atskyrimas (Paulauskienė, Kurasova, 2014), o tai reiškia, tinkamesnį klasterių k skaičių. Silueto koeficientas dažnai yra ne tik apskaičiuojamas, bet ir vizualizuojamas. Tam naudojamos įvairios programavimo kalbos. Pritaikius klasterizacijos modelį, galima gauti ne tik vidutinį Silueto koeficientą, tačiau ir kiekvieno klasterio tankumą bei dydį. Šis vizualizavimo metodas ypač naudingas įvertinti, ar klasteriai pasiskirstę vienodai, ar jiems taškai buvo priskirti teisingai, taip pat tai patogus metodas skirtingam klasterių skaičiui lyginti.

3.8. Tekstinė duomenų analizė

Bibliografinis įrašas yra įrašas bibliografinėje rodyklėje (arba bibliotekos kataloge), kuris atstovauja ir apibūdina konkretų šaltinį. Bibliografiniame įrašė yra duomenų elementai, kurie yra būtini, kad naudotojai galėtų identifikuoti apibūdinamąjį šaltinį. Tokiame įrašė taip pat yra papildoma pagalbinė informacija, pateikta bibliografinio įrašo formatu: ji gali palaikyti tam tikras duomenų bazės funkcijas, tokias kaip paieška arba naršymas (pvz., pagal raktinius žodžius), arba gali pateikti išsamesnį turinio elemento pristatymą (pvz., straipsnio santrauka). Taigi bibliografinis įrašas yra glaudžiai susijęs su tekstiniais duomenimis, nes jo sudedamosios dalys (autorių pavardės, publikacijos pavadinimas, leidinio pavadinimas ir t.t.) yra pateikiamos kaip tekstas. Todėl atliekant bibliometrinę analizę, tekstinė duomenų analizė yra neatsiejama darbo dalis.

3.8.1. Taisyklėmis pagrįsti algoritmai

Remiantis Makhija ir Ahuja (2018), pradinė informacija, kuri naudojama analizėje, gali būti gerai apibrėžta, įrašai joje gali būti susiję semantiškai arba pagal kontekstą, tačiau labai dažnai sutinkamas ir priešingas variantas. Dėl to informacijos išgavimas (angl. *information extraction*) yra labai svarbus tekstinių duomenų analizės komponentas, nes jo pagalba galima išgauti aiškius faktus pagal iš anksto nurodytus kriterijus, esybes arba santykius, sukurti prasmingesnius ir daugiau informacijos savyje talpinančius įrašus. Apibendrinant, informacijos išgavimas yra atliekamas, siekiant apdoroti pradinį tekstinių duomenų rinkinį taip, kad jame pateikiama informacija būtų lengviau prieinama. Autoriai savo darbe nagrinėja taisyklėmis

pagrįsto algoritmo sudarymą. Jie teigia, kad norint sukurti tokio tipo algoritmą, reikia apibrėžti taikomą sąlygą ir siekiamą gauti rezultatą. Taisyklėse aprašomi veiksmai, kurių reikia imtis kai įvykdoma sąlyga, o rezultatai atitinka konkrečius vartotojo apibrėžtus kriterijus. Taisyklėmis pagrįsti algoritmai susideda iš šių pagrindinių dalių:

1. Taisyklių aprašymas: sąlyginiai teiginiai, siejantys nurodytas sąlygas su rezultatais;
2. Interpretatorius: ieško informacijos ir sąsajų, pagal pateiktas sąlygas ir prieš tai gautus rezultatus. Interpretatorius, remdamasis faktais ir taisyklėmis, suranda teisingą informaciją ir atitinkamai ją surenka.
3. Aiškinamoji sistema: leidžia vartotojams suprasti, kaip algoritmas pasiekė tam tikrą rezultatą;
4. Naudojamas įrankis: sistema, kurioje kuriamas ir taikomas algoritmas;
5. Vartotojo sąsaja: leidžia vartotojui išgauti atitinkamus rezultatus.

Gera apibrėžtas taisyklėmis pagrįstas algoritmas susideda iš pilno, neperteklinio ir nuoseklaus taisyklių rinkinio. Algoritmo patikrinimas atliekamas po to, kai sistema jau yra sukurta, tačiau aptiktų klaidų taisymas yra galimas ir tai leidžia valdyti taisyklėmis pagrįstos sistemos kokybės gerinimą. Galutinis sukurto algoritmo variantas yra toks, kad tik maža dalis įrašų paliekama rankiniam tikrinimui ir koregavimui.

Makhija ir Ahuja (2018) tyrimo tikslas – sukurti sąvokų duomenų bazę, išvalyti gautus duomenis ir automatizuoti informacijos gavimo procesą iš bibliografinės duomenų bazės. Autoriai pabrėžia, kad vykdant šį procesą ir dirbant su bibliografiniais duomenimis, galima susidurti su duomenų nenuoseklumo problema. Autorių pavardės įvairiomis kalbomis užrašomos skirtingai: pateikiama keletas pavardžių arba inicialų raidžių, šie elementai tarpusavyje atskiriami nevienodai. Dėl to, kuriant taisyklėmis pagrįstą algoritmą, svarbu aprašyti visas šias išimtis, kad galutiniame rezultate pavyktų išgauti kuo daugiau teisingų duomenų. Autoriai taip pat susiduria su problema, kuomet skirtingų dokumentų tipų (straipsnių, knygų, konferencijų ir t.t.) bibliografiniai įrašai turi skirtingus privalomuosius komponentus. Tai taip pat reikalauja atskirų taisyklių kūrimo, kad visi įrašų komponentai būtų identifikuoti teisingai ir informacija nebūtų prarasta.

3.8.2. Neapibrėžti algoritmai

Atliekant tekstinių duomenų analizę, taip pat yra taikomi išplėstiniai algoritmai, naudojančys natūralios kalbos apdorojimą (angl. *natural language processing*, *NLP*), mašininių mokymąsi, statistinius metodus arba jų derinius. Teksto palyginimo (angl. *string-matching*) algoritmai taip pat yra dažnai sutinkami tekstinių duomenų analizės uždaviniuose. Loo (2014) teigia, kad teksto palyginimas yra svarbi daugelio duomenų apdorojimo programų dalis, įskaitant statistinį suderinimą, teksto paiešką, teksto

klasifikavimą, rašybos tikrinimą ir genomiką. Teksto palyginimo proceso esmė yra galimybė kiekybiškai įvertinti dviejų eilučių panašumą pagal eilučių metrikas.

Bosker (2021) savo darbe nagrinėja keletą neapibrėžtų (angl. *fuzzy*) algoritmų, skirtų teksto palyginimui, kitaip žinamam kaip apytikslis eilutės atitikmuo. Šie metodai skirti rasti eilutes, atitinkančias lyginamą eilutę apytiksliai, o ne tiksliai. Metodas yra taikomas įrašų palyginime, klaidų taisyme, nepageidaujamų laiškų atrinkime, kalbos atpažinime ir akustinio modelio apmokyme. Nagrinėjant literatūrą, kurioje yra aptariami neapibrėžti algoritmai, galima sutikti įvairių teksto palyginimo metodų ir skaičiavimo būdų. Loo (2014) teigia, kad teksto metrikos gali būti skirstomos į 3 sritis: redaguojami atstumai (angl. *edit-based*), *q-gram* atstumai, euristiniai (angl. *heuristic*) atstumai. Autoriaus teigimu, redaguojami atstumai gali būti aprašomi, kaip reikiamas veiksmų skaičius, kad vienas įrašas būtų sulygintas su kitu. Veiksmai apima simbolio pakeitimą, ištrynimą, įterpimą arba simbolių perkėlimą. *q-gram* atstumai gaunami lyginant q simbolių sekų atsiradimą tarp eilučių. Euristinis atstumų skaičiavimas neturi tvirto matematinio pagrindo, bet buvo sukurtas kaip praktinė priemonė šiam tikslui pasiekti.

Remiantis kitais autoriais, Kang (2015) ir Loo (2014), keli tekstiniai įrašai gali būti lyginami skaičiuojant Hamingo, Levenšteino, Damero – Levenšteino arba Jaro – Vinkler atstumus. Hamingo atstumas naudojamas, kuomet lyginami vienodo ilgio įrašai ir rezultatas atitinka pozicijų skaičių, kuriose atitinkami simboliai skiriasi. Loo (2014) pateikia Hamingo atstumo formulę:

$$d_h(s, t) = \sum_{i=1}^{|s|} (1 - \delta(s_i, t_i)), \text{ jei } |s| = |t|, \infty \text{ kitu atveju,} \quad (40)$$

čia $\delta(s_i, t_i) = 1$, jei $s_i = t_i$ ir 0 kitu atveju. Kai s ir t įrašai yra vienodo ilgio, didžiausia galima reikšmė yra $|s|$.

Levenšteino atstumas yra skaičius, nurodantis, kuo skiriasi du įrašai. Kuo didesnis skaičius, tuo labiau šie įrašai skiriasi (Loo (2014)):

$$dist_l(s, t) = \begin{cases} 0, & s = t = \varepsilon, \\ \min \begin{cases} dist_l(s, t_{1:|t|-1}) + \omega_1 \\ dist_l(s_{1:|s|-1}, t) + \omega_2 \\ dist_{s,t}(s_{1:|s|-1} - 1, t_{1:|t|-1}) + [1 - \delta(s_{|s|}, t_{|t|})]\omega_3, \end{cases} & \text{kitu atveju.} \end{cases} \quad (41)$$

čia ε reiškia tuščią eilutę, $\omega_1, \omega_2, \omega_3$ – neneigiamos nuobaudos už ištrynimą, įterpimą ir pakeitimą, kai s yra keičiama į t .

Damero – Levenšteino atstumas yra Levenšteino atstumo skaičiavimo papildymas, kuomet paskutinė formulės eilutė papildoma (Loo (2014)):

$$\begin{aligned}
 & dist_{dl}(s, t) \\
 & = \begin{cases} 0, & s = t = \varepsilon, \\ \min \begin{cases} d_{dl}(s, t_{1:|t|-1}) + \omega_1 \\ d_{dl}(s_{1:|s|-1}, t) + \omega_2 \\ d_{dl}(s_{1:|s|-1}, t_{1:|t|-1}) + [1 - \delta(s_{|s|}, t_{|t|})]\omega_3 \\ \min_{(i,j) \in \Lambda} d_{dl}(s_{1:i-1}, t_{1:j-1}) + [(|s| - i) + (|t| - j) - 1]\omega_4 \end{cases} & kitu atveju, \end{cases} \quad (42) \\
 & \Lambda = \{(i, j) \in \{1, \dots, |s|\} \times \{1, \dots, |t|\} : s_{|s|} = t_i, s_i = t_{|t|}\}.
 \end{aligned}$$

Jaro – Vinkler atstumas nurodo sutampančių simbolių skaičių tarp lyginamų eilučių, kuomet jų pozicija nėra labai nutolusi ir prideda nuobaudas už simbolių ištrynimą, įterpimą ir pakeitimą. Šis metodas suteikia galimybę ignoruoti rašybos klaidas. Jis apskaičiuojamas pagal formulę (Loo (2014)):

$$dist_j(s, t) = \begin{cases} 0, & s = t = \varepsilon, \\ 1, & m = 0 \text{ ir } |s| + |t| > 0, \\ 1 - \frac{1}{3} \left(\omega_1 \frac{m}{|s|} + \omega_2 \frac{m}{|t|} + \omega_3 \frac{m - T}{m} \right), & \text{kitu atveju.} \end{cases} \quad (43)$$

Čia m yra sutampančių simbolių skaičius, T – pusė nuobaudų už simbolių ištrynimą, įterpimą ir pakeitimą skaičiaus. Jei gautas Jaro – Vinkler atstumas lygus 0, tuomet įrašai neturi jokių panašumų, jei 1 – įrašai yra identiški.

Bosker (2021) taip pat nagrinėja dar vieną neapibrėžtą algoritmą, vadinamą prieigos raktų rūšiavimo santykiu (angl. *Token Sort Ratio*). Šis metodas buvo sukurtas ilgesnių tekstinių įrašų analizei, pavyzdžiui sakinių. Metodas atlieka šiuos pagrindinius veikslus su lyginamais įrašais: jie yra tokenizuojami (duomenų elementas pakeičiamas atitikmeniu, vadinamu prieigos raktu (tokenu), kuris neturi jokios išorinės ar panaudojamos reikšmės), visos raidės pakeičiamos į mažąsias, pašalinami skyrybos ženklai, įrašų dalys išrikiuojamos abėcėlės tvarka ir sujungiamos. Atlikus šiuos veiksmus, apskaičiuojamas Levenšteino atstumas ir grąžinamas panašumo indeksas. Huynh (2020) ir Dutta (2021) savo darbuose praktiškai taiko Token Sort Ratio metodą, naudodami *Python* programą, ir lygina jį su panačiu metodu – prieigos raktų rinkinio santykiu (angl. *Token Set Ratio*). Token Set Ratio metodas atlieka tuos pačius veiksmus, kaip ir Token Sort Ratio, tik jame yra lyginami sutampantys prieigos raktai (tokenai) vietoje bendro įrašo tokenizavimo. Papildomi ar pasikartojantys žodžiai šiam metodui neturi reikšmės.

4. Tiriamoji dalis

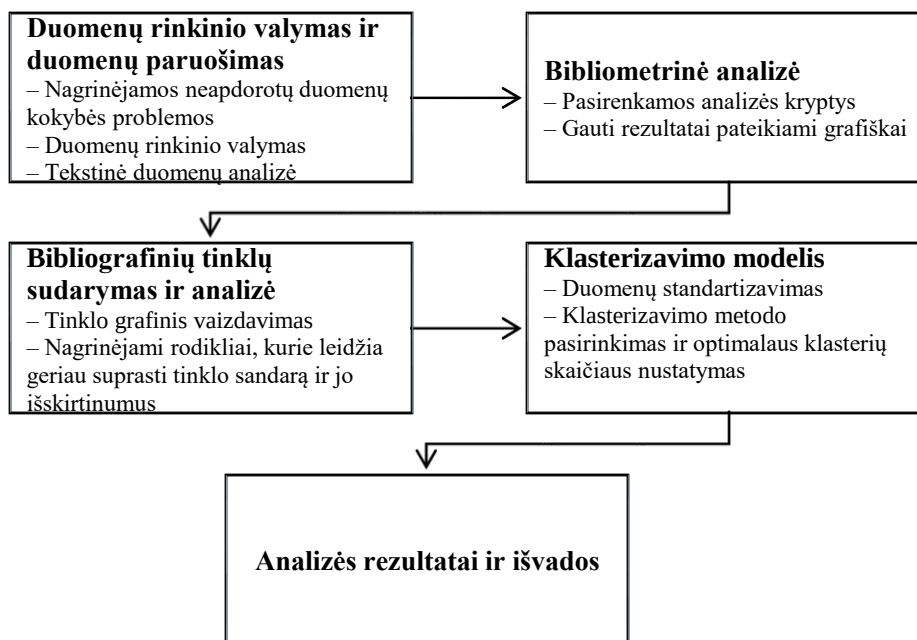
4.2. Siūloma bibliometrinės analizės metodika

Mokslinio tyrimo metodai apsprendžia tai, kaip tyrėjas gali surinkti įvairius jo nagrinėjimo objekto ar reiškinių aspektus apibūdinančią nestruktūrizuotą informaciją ir tai, kaip naujų žinių elementai bus sisteminami, siejami su jau turima informacija, analizuojami ir kaip bus pateikiama gauta mokslinė informacija. Tyrėjai visuomenei ir mokslo bendruomenei pateikia jau apibendrintas, visapusiškai patikrintas išvadas apie tiriamuosius reiškinius.

Mokslinio tyrimo kaip informacinės sistemos struktūrą sudaro:

1. Įeiga, per kurią informacija patenka į sistemą;
2. Transformavimo mechanizmas, kuris padeda perdirbti gautą informaciją;
3. Išeiga, per kurią pertvarkyta (mokslinė) informacija perduodama į išorę (Bitinas, 2013).

Remiantis šia struktūra, sudaroma ir pasiūloma bibliometrinės analizės metodika, kuri pateikiama scheme žemiau:



3 pav. Siūloma bibliometrinės analizės metodika

Siūloma metodika išsiskiria iš kitų bibliometrinės analizės būdų, nagrinėtų literatūros šaltiniuose, ir juos papildo šiais aspektais:

1. Duomenų paruošimas išskiriamas kaip būtinas bibliometrinės analizės etapas.
2. Tekstinė duomenų analizė taip pat yra reikšminga metodikos dalis.
3. Siūloma ne tik sudaryti ir nagrinėti bibliometrinius tinklus, tačiau taip pat analizuoti grafų teorijos rodiklius ir tinklo sandarą.
4. Į klasterizavimo modelį siūloma įtraukti daugiau rodiklių, pritaikant duomenų normalizavimo metodą. Gautiems rezultatams vizualizuoti siūloma taikyti pagrindinių komponentų analizę.
5. Pritaikius siūlomą metodiką, galutiniai bibliometrinės analizės rezultatai yra įvairiapusiški ir pateikia daugiau žinių apie pradinį duomenų rinkinį.

Siūloma metodika pritaikoma tyrimo objektui – VILNIUS TECH bibliotekoje registruotų mokslinių publikacijų bibliografiniams duomenims. Tolimesniuose skyriuose atliekamas empirinis tyrimas ir pateikiamos išvados.

4.3. Duomenys

4.3.1. Duomenų rinkinio apžvalga

Formuojant šio magistrinio darbo tikslą ir uždavinius, buvo nuspręsta analizuoti ne konkrečios srities bibliografinius duomenis, bet rinktis VILNIUS TECH universiteto mokslo darbuotojų publikacijas. Toks sprendimas atsirado perskaičius Petrauskienės ir Aleksandrovės (2015) tyrimą „Bibliografiniai tyrimai Vilniaus universiteto bibliotekos veikloje“, kurio metu buvo analizuojami Vilniaus Universiteto bibliotekos bibliografiniai duomenys. Šis tyrimas tapo pavyzdžiu, kad bibliometrinė analizė gali būti naudojama tiriant universiteto mokslo darbuotojų veiklą. Tai yra puiki priemonė vertinti publikacijų skaičiaus ir tematikos kitimą nagrinėjamame periode, analizuoti autorių bendradarbiavimą, citavimo rodiklius bei leidinius, kuriuose mokslo darbai yra publikuojami.

Suformavus šį tikslą, buvo kreiptasi į VILNIUS TECH bibliotekos darbuotojus su prašymu pasidalinti bibliografiniais duomenimis. Apibendrinti kriterijai bibliografinių įrašų sąrašui buvo tokie:

- Bent vieno iš autorių atstovaujama institucija – VILNIUS TECH;
- Publikavimo metų intervalas – 2010-01-01 – 2020-01-01;
- Dokumentų tipas – moksliniai straipsniai (Lietuvos ir užsienio recenzuojamuose leidiniuose);
- Nurodytas indeksavimas duomenų bazėse (DB);
- Jei yra, būtų nurodyti citavimo rodikliai („Web of Science“ ir/arba „Scopus“ DB);
- Nurodyta mokslų kryptis.

Gavus bibliografinius įrašus buvo atlikta duomenų rinkinio apžvalga. Žemiau pateikiami duomenų rinkinio ypatumai:

- Duomenų rinkinį sudaro 7 513 įrašų;
- Duomenų pateikimo formatas - .csv (comma delimited);
- Bibliografinių įrašų pateikimo stilius – APA;
- Pateikiama informacija apie duomenų bazes, kuriose konkreči publikacija yra patalpinta;
- Nurodyti citavimo rodikliai;
- Nurodyti leidinių vertinimo rodikliai;
- Publikacijos yra įvairių kalbų (Lietuvos ir užsienio leidiniuose).

4.3.2. Duomenų rinkinio valymas ir duomenų paruošimas

Duomenų valymas yra duomenų kokybės gerinimo procesas, taisant netikslius įrašus, esančius duomenų rinkinyje. Šis terminas konkrečiai reiškia nepilnų, neteisingų, netinkamai suformatuotų, pasikartojančių ar nesusijusių duomenų, kitaip vadinamų „nešvariais duomenimis“, aptikimą ir modifikavimą, pakeitimą arba ištrynimą. Duomenų valymas taip pat apima pasikartojančių duomenų pašalinimą iš duomenų rinkinio. Neatlikus duomenų valymo procedūrų tinkamai, analizės rezultatai gali būti netikslūs arba net klaidingi. Dėl to šis etapas yra labai svarbus.

Atlikus mokslinių straipsnių, kuriuose nagrinėjamas duomenų valymo procesas, analizę, galima išskirti 4 pagrindinius proceso etapus:

1. Tikrinimas;
2. Įvertinimas;
3. Valymas;
4. Dokumentavimas.

Brownlee (2020) teigia, kad duomenų paruošimas gali būti svarbiausia projekto dalis. Tai yra daugiausiai laiko atimanti dalis, nors atrodo, kad tai mažiausiai aptarinėjama tema. Duomenų rengimo filosofija yra atrasti, kaip būtų galima geriausiai atskleisti nežinomą problemos struktūrą. Tam dažnai reikalingas pasikartojantis eksperimentavimas, naudojant įvairių duomenų paruošimo metodų rinkinį, siekiant atrasti kas veikia gerai ar geriausiai.

Tikrinimo etapas apima sistematinį duomenų rinkinio apžvelgimą, ieškant savybių, kurios yra išskirtinės arba pasikartojančios. Pradedama nuo pagrindinių duomenų rinkinio savybių.

Duomenų rinkinio formatas – .csv (angl. *comma delimited*), dėl ko duomenis galima paprastai suskirstyti stulpeliais. Tačiau stulpeliai turi ne konkrečią informaciją, o keletą mokslinio straipsnio identifikatorių. Vienas pagrindinių stulpelių, pavadintų *projection_apa* talpina šiuos rodiklius:

- Autoriaus (-ių) pavardė(s) ir inicialai;
- Išleidimo metai;
- Antraštė;
- Leidinio, kuriame išspausdintas straipsnis, antraštė;
- Leidinio tomas ir numeris;
- Straipsnio puslapiai leidinyje;
- DOI (skaitmeninis objekto identifikatorius).

Pagrindinė problema, turint visus šiuos rodiklius viename įrašė – rasti būdą, kaip juos atskirti ir perkelti į skirtingus duomenų rinkinio stulpelius. Kiekvienas įrašas turi įvairius skirtukus, dalis jų kartojasi, dalis unikalūs konkretiems įrašams. Kadangi bibliografiniai įrašai sudaryti iš autorių pavardžių ir inicialų, negalima naudoti taško arba kablelio simbolių kaip skirtukų. Šie simboliai taip pat dažnai pasikartoja antraštėse.

Kitas iššūkis susijęs su autorių pavardėmis ir inicialais: nors bibliografiniai įrašai turi griežtus reikalavimus, pavardės gali būti rašomos keliais būdais. Nagrinėjant duomenų rinkinį, išskirti šie autorių pavardžių ir inicialų užrašymo būdai:

1. Pavardė, V.
2. Pavardė, V.V.
3. Pavardė, V.V.V.
4. Pavardė, V.V.V.V.
5. Pavardė, V.V.V.V.V.
6. Pavardė Pavardė, V.
7. Pavardė-Pavardė, V.
8. Pavardė Pavardė Pavardė, V.

Kiti duomenų rinkinio stulpeliai yra labiau standartizuoti. Jie taip pat talpiną ne vieną reikšmę, tačiau jos yra atskirtos vieningais skirtukais. Stulpelių reikšmės:

- *docs_id*: unikalus dokumento numeris;
- *doc_code*: mokslo publikacijų rūšys;
- *res_fields*: mokslo ir meno sričių ir kryptų klasifikacija;

- *dbases*: duomenų bazės, kuriose talpinamos publikacijos;
- *if_info*: duomenų bazėje „Web of Science“ indeksuojamo žurnalo cituojamumo rodikliai;
- *scopus_info*: „Scopus“ publikacijų duomenų bazė, kurioje jos apibūdinamos skaitiniais kokybės matais;
- *pub_year*: publikacijos metai.

Toliau atliekamas duomenų įvertinimo etapas. Stögera, Schneebergerb, Kiesebergc ir Holzinger (2021) išskiria pagrindines tipiškas neapdorotų duomenų kokybės problemas. Didžioji dalis jų buvo rasta ir nagrinėjamame duomenų rinkinyje:

2 lentelė. Pagrindinės neapdorotų duomenų kokybės problemos

Problema	Problemų aprašymas	Duomenų rinkinio ypatybės
Duomenų trūkumas	Laukai nėra baigti dėl techninių klaidų arba laukai nėra privalomi ir paliekami tušti.	Procentinė originalaus duomenų rinkinio stulpelių dalis, turinti praleistas reikšmes: docs_id 0.00% doc_code 0.00% projection_apa 0.00% res_fields 0.00% dbases 21.00% if_info 63.00% scopus_info 53.00% pub_year 0.00%
Netikros reikšmės	Kai duomenų rinkinyje yra praleistų reikšmių, į laukus dažnai įterpiamos netikros reikšmės.	Netikrų reikšmių duomenų rinkinyje nėra.
Triukšmas	Tai yra beprasmiškie duomenys.	Duomenų rinkinyje yra pateikiama internetinė prieiga prie publikacijos arba doi nuoroda. Ši informacija yra perteklinė ir tolimesnei analizei nereikalinga.
Klaidingi duomenys	Klaidingi duomenys gali atsirasti dėl neteisingo jų įvedimo į duomenų bazę, žmogiškosios arba sistemos klaidos.	Kadangi duomenų rinkinyje yra daug tekstinių duomenų, matoma, kad yra rašybos klaidų. Ne visada naudojamos lietuviškos raidės.
Nenuoseklūs duomenys	Duomenų įrašai gali būti nenuoseklūs, nes juose yra laukai, kuriuose yra tarpusavyje susijusios informacijos, kuriai turi būti pritaikytos taisyklės.	Viename iš duomenų laukų (projection_apa), informacija yra atskirta ne nuosekliais skirtukais. Šiame lauke pateikiamas ne toks pat kiekis informacijos ir, jei jos trūksta, tai nėra pažymima. Tų pačių autorių pavardės nesuvienodintos (ne visur naudojamos lietuviškos raidės, rašomi ne visi inicialai, pasikeitus pavardei, paliekama ir senoji: atsiranda du autoriai, kuomet iš tiesų yra tas pats ir pan.)
Užkoduoti duomenys	Kartais neįmanoma iššifruoti duomenų įrašo dėl sistemos registravimo klaidų, neužbaigtų operacijų duomenų bazėse ir pan.	Užkoduotų reikšmių duomenų rinkinyje nėra.

Pasikartojantys pirminiai raktai	Pirminiai raktai naudojami aiškiai identifikuoti kiekvieną duomenų eilutę.	Pirminiai raktai nesikartoja. Kiekviena eilutė turi unikalų identifikatorių (dokumento numeris docs_id stulpelyje).
Daugiafunkciniai laukai	Jie gali būti naudojami įvairiais būdais: kaupti skirtingą informaciją, priklausyti nuo kitų laukų, turėti laisvo teksto duomenis ir pan.	Daugiafunkcinių laukų duomenų rinkinyje nėra.
Sutartų taisyklių pažeidimas	Kartais duomenys turi atitikti tam tikras griežtai apibrėžtas taisykles.	Bibliografiniai įrašai yra labai griežtai apibrėžti taisyklėmis. Tačiau duomenų rinkinyje yra išimčių, kuomet šioms taisyklėms yra nusižengiama.

Apibendrinant galima teigti, kad praleistos reikšmės yra daugiau susijusios su leidiniais (*if_info*, *scopus_info*) ir 21% publikacijų neturi pažymėtų duomenų bazių. Šios trūkstamos reikšmės negali būti užpildytos dažniausiai pasikartojančiomis reikšmėmis arba reikšmių vidurkiu, nes tai pakenktų analizės rezultatams. Kita problema yra susijusi su duomenų nuoseklumu ir sutartų taisyklių pažeidimu. Dėl to duomenys tampa sunkiau valomi ir standartizuojami.

Pirmasis duomenų valymo žingsnis buvo išskirti autorių pavardes iš stulpelio *projection_apa*. Ankstesniame skyriuje paminėta, kad buvo išskirtos 8 pavardžių ir inicialų kombinacijos. Joms rasti duomenų rinkinyje ir atskirti nuo bendro duomenų stulpelio, buvo sukurtas taisyklėmis pagrįstas tekstinių duomenų išskyrimo algoritmas. Tai buvo atlikta naudojantis *Python* biblioteka *re*. Šis modulis atlieka reguliariųjų išraiškų atitikimo operacijas, panašias į tas, kurios naudojamos Perl („Perl Compatible Regular Expressions“ yra C kalba parašyta biblioteka, kuri įgyvendina reguliariųjų išraiškų operacijas, įkvėptas „Perl“ programavimo kalbos galimybių). Kiekvienai pavardžių ir inicialų kombinacijai buvo sukurta atskira taisyklė, kuri aprašė ieškomo tekstinio įrašo formatą. Žemiau pateikiamas *Python* programos kodo pavyzdys, kuriuo buvo randami, atskiriami ir iš bendro duomenų rinkinio pašalinami įrašai, kurie atitiko formatą *Pavardė, V.V.*:

```

autoriai_parser5 = re.compile("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.])")
autoriai5 = pd.DataFrame(data.projection_apa.map(lambda x:
                                                re.findall(autoriai_parser5, x)))

```

Toliau buvo pašalinti pertekliniai *projection_apa* stulpelyje esantys internetiniai adresai. Jie atskirti naudojant vieną iš skirtukų: „doi“, „Retrieved“ arba „Žiūrėta“. Internetiniai adresai, sekantys skirtukus „Retrieved“ arba „Žiūrėta“, duomenų analizei nėra reikalingi, dėl to pašalinami. Internetinis adresas, sekantis skirtuką „doi“ taip pat nėra reikalingas, tačiau įvedamas naujas kategorinis kintamasis: 0, jei doi adreso nėra, 1 – jei jis yra.

Kitas etapas – rasti ir atskirti publikacijų puslapius leidiniuose. Šie įrašai suteiks informacijos, kokios apimties yra publikacijos. Analizuojant įrašus buvo pastebėta, kad leidinių puslapiai gali būti labai įvairių kombinacijų, kaip ir autorių pavardės. Dėl to jie buvo atskiriami etapais: pašalinant išskirtis, tuomet puslapius, kurie yra keturženkliai (pvz. 1532-1542), triženkliai (pvz. 100-104) ir dviženkliai (pvz. 10-15). Taip pat sukuriama naujas duomenų stulpelis, kuriame apskaičiuojama publikacijos apimtis puslapiais. Pašalinami publikavimo metai, nes jie yra pateikti atskirame stulpelyje (*pub_year*). Šioms užduotims atlikti taip pat naudojama *Python* biblioteka *re*. Žemiau pateikiamas *Python* kodo pavyzdys, kuriuo buvo randami, atskiriami ir iš bendro duomenų rinkinio pašalinami įrašai, kuriuose buvo triženkliai puslapių skaičiai:

```
puslapiai_parser4 = re.compile("([0-9]+[-]+[0-9][0-9][0-9])")
puslapiai4 = pd.DataFrame(data1.new_6.map(lambda x:
                                         re.findall(puslapiai_parser4, x)))
```

Pašalinus dalį duomenų, pavyko pritaikyti bendrą skirtuką likusiems įrašams ir taip atskirti publikacijos pavadinimą bei leidinio numerį. Leidinio numerio atskyrimas pareikalavo kelių etapų, nes informacija pateikiama labai skirtingai ir negalima pritaikyti vieno metodo visiems numeriams identifikuoti. Likusi informacija *projection_apa* stulpelyje yra leidinio pavadinimas. Leidinio pavadinimas prie dalies publikacijų buvo pateiktas keliomis kalbomis. Dėl to sukurti 3 papildomi stulpeliai, kuriuose talpinami leidinio pavadinimai kitomis kalbomis.

scopus_info, *if_ifo* ir *res_fields* stulpeliai pateikti standartizuotai, todėl buvo nesudėtinga juos išskaidyti į sudedamąsias dalis. *dbases* stulpelis taip pat pateiktas standartizuotai, tačiau kiekvienas įrašas turi skirtingą skaičių išvardintų duomenų bazių. Išskaidžius šį stulpelį buvo gautos 340 skirtingų duomenų bazių, kuriose saugomos publikacijos. Dalis iš jų pasikartoja tik kelis kartus. Dėl to buvo priimtas sprendimas atskirai identifikuoti pirmąsias 10, o likusias priskirti kategorijai „other_dbases“.

Atlikus duomenų valymą, iš pirminių 8 stulpelių buvo gauti 37 nauji stulpeliai:

3 lentelė. Duomenų rinkinys

Pirminis duomenų rinkinio stulpelis	Sudaryti nauji duomenų stulpeliai
docs_id	docs_id
doc_code	doc_code
projection_apa	publication journal_name journal_name_in_foreign_language_1 journal_name_in_foreign_language_2 journal_name_in_foreign_language_3 journal_number pages volume_in_pages

	doi
res_fields	res_fields_1 res_fields_2 res_fields_3
dbases	dbases_count Scopus SCIE ICONDA IndexCopernicus INSPEC Academic_Search_Complete DOAJ SSCI VINITI Business_Source_Complete other_dbases
scopus_info	scopus_sources SNIP CiteScore SJR quartile_in_caterory
if_info	IF AIF quartile_in_caterory_2 if_info_years InCites_JCR
pub_year	pub_year

Pakeitimų dokumentavimas reiškia aptiktų klaidų, pakeitimų, papildymų ir klaidų tikrinimo aprašymą, kad prireikus būtų galima grąžinti pradinę duomenų vertę. Duomenų valymas atliktas naudojant *Python* programą ir visi žingsniai aprašyti kuriant kodą. Taip pat, prieš pradėjus duomenų valymą, buvo padaryta duomenų rinkinio kopija, kad būtų išsaugoti pradiniai duomenys. Proceso dokumentavimas padeda bet kuriuo proceso metu grįžti į jau atliktus žingsnius ir patikrinti, kas buvo daroma.

4.3.3. Žurnalų indeksavimas

Atlikus duomenų valymą, buvo pastebėta, kad dalis žurnalų pavadinimų kartojasi, kadangi kelios publikacijos buvo paskelbtos tuose pačiuose leidiniuose. Tačiau dalis besikartojančių žurnalų turėjo ne identiškus pavadinimus dėl duomenų nenuoseklumo. Vienas iš leidinių pavadinimų nevienodumo pavyzdžių gali būti žurnalas *Acta Physica Polonica A*:

1. *Acta physica Polonica A*;
2. *Acta physica Polonica. A*;
3. *Acta physica polonica. A*.

Nagrinėjant duomenis vizualiai, akivaizdu, jog šie įrašai atitinka tą patį leidinį. Siekiant suvienodinti pavadinimus ir toliau atlikti leidinių bibliometrinę analizę, buvo pasirinktas būdas leidinius priskirti grupėms, kur viena grupė apjungtų tą patį leidinį užrašytą skirtingais būdais. Šiai užduočiai atlikti buvo sukurtas neapibrėžtas (angl. *fuzzy*) teksto palyginimo algoritmas, naudojama *Python* biblioteka *FuzzyWuzzy*.

Ieškant panašių leidinių pavadinimų, buvo nuspręsta pritaikyti du metodus, esančius *FuzzyWuzzy* bibliotekoje: *Token Sort Ratio* ir *Token Set Ratio* ir patikrinti, kuris iš jų grąžins geresnius rezultatus. Sudarius modelius, gautos duomenų lentelės, kuriose leidiniai yra suporuojami ir pateikiamas poros panašumo indeksas. Jei įrašai visiškai sutampa, grąžinama reikšmė 100. Siekiant kuo didesnio tikslumo, buvo pasirinktas panašumo indeksas lygus 95 kaip žemiausia panašumo riba. Žemiau pateikiamas *Token Set Ratio* modelio sudarymo kodas, kurio rezultatai perkeliama į duomenų lentelę:

```
unikalus_pavadinimai = sorted(journals.astype(str).unique().tolist())

score_sort = [(x,) + i
               for x in unikalus_pavadinimai
               for i in process.extract(x, unikalus_pavadinimai,
                                       scorer=fuzz.token_set_ratio)]

similarity_sort = pd.DataFrame(score_sort,
                              columns=['journals_set', 'match_set', 'score_set'])
```

Apskaičiavus abiejų modelių rezultatus, gautos išvados: *Token Sort Ratio* metodu, kurio panašumo indeksas lygus 95, gaunamas 202 leidinių poros; *Token Set Ratio* metodu – 505 poros. Dėl to nuspręsta toliau dirbti su *Token Set Ratio* metodu.

Kitu žingsniu visiems leidiniams suteikiamas indeksas. Buvo pastebėta, kad leidiniai rezultatų lentelėje kartojasi: vienu atveju jie yra kaip pirmasis įrašas, su kuriuo lyginami visi likę, kitu atveju – jis yra priskiriamas bendrai grupei. Dėl to vienas leidinys gali turėti kelis skirtingus indeksus. Šiam neatitikimui pašalinti, lyginami vieno leidinio indeksai ir jam priskiriamas tik pirmasis, kitus panaikinant. Atlikus šiuos veiksmus, unikalūs indeksai priskiriami ir tiems leidinių pavadinimams, kurie nebuvo identifikuoti, kaip turintys panašumų su kitais įrašais.

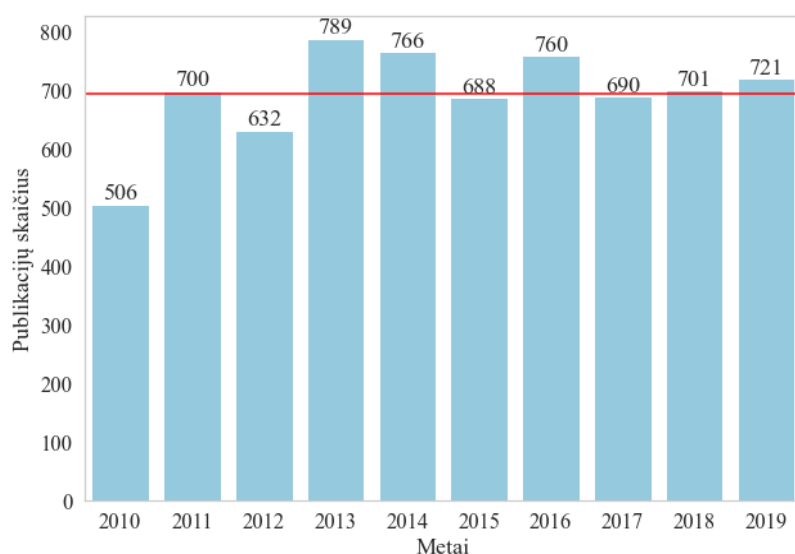
Apibendrinant atliktus veiksmus, gauti tokie rezultatai:

- Prieš pradėdant leidinių grupavimą, duomenų rinkinyje buvo 1 713 unikalūs leidinių pavadinimai;
- Atlikus jų grupavimą ir priskyrus grupės indeksą, liko 1 254 unikalios reikšmės;
- Žurnalų pavadinimų grupavimas pagerino leidinių pavadinimų kokybę 27 %.

4.4. Bibliometrinė analizė

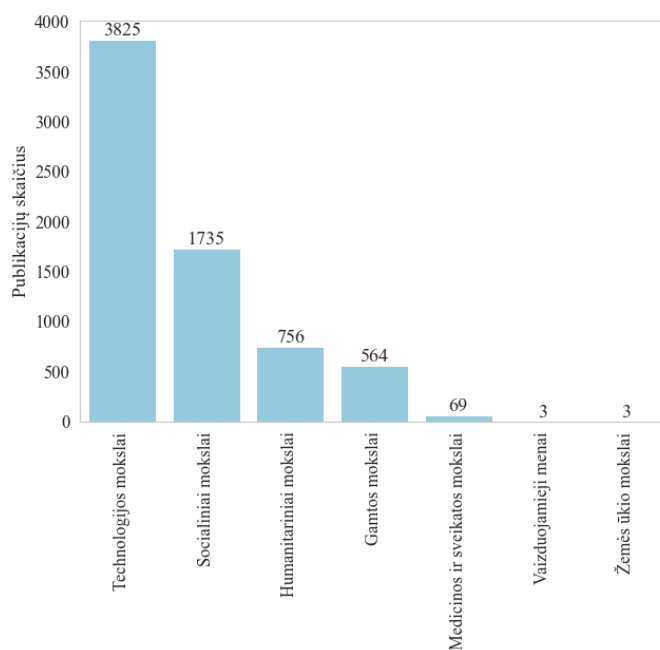
Sudarius galutinį ir išvalytą duomenų rinkinį, atliekama bibliometrinė analizė. Jos tikslas – išanalizuoti turimą duomenų rinkinį, nustatyti statistinius rodiklius, aptarti pasikartojimus ir išskirtis. Atliekant bibliometrinę analizę, vertinamas publikacijų skaičiaus pokytis 2010 – 2019 metais, publikacijų pasiskirstymas pagal mokslo sritis, technologijos mokslų sritis tiriama detaliau, ją skaidant į smulkesnes dalis, ir analizuojant šių smulkesnių sričių publikacijų skaičiaus pokytį 2010 – 2019 metais. Taip pat analizuojamas publikacijų skaičiaus, pagal mokslinių straipsnių sritis, pokytis nagrinėjamu periodu, vertinamas bendra autorių skaičius ir išskiriami produktyviausi autoriai. Galiausiai, pateikiamas duomenų bazėse taplinamų publikacijų skaičiaus pokytis 2010 – 2019 metais.

Bibliometrinė analizė pradedama nuo bendro publikacijų skaičiaus pokyčio per nagrinėjamą periodą (2010 – 2019 metus), kuris vaizduojamas grafike žemiau:



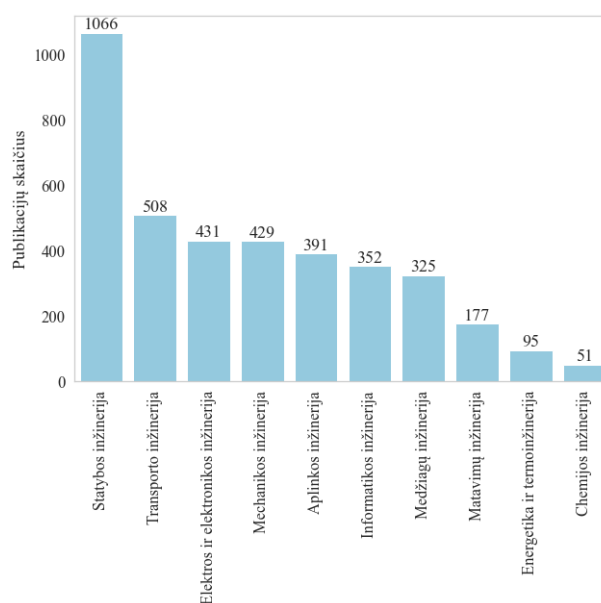
4 Pav. Publikacijų skaičiaus pokytis 2010 – 2019 metais

Vidutinis VILNIUS TECH mokslo darbuotojų publikacijų skaičius per nagrinėjamus metus buvo 695 publikacijos (grafike vaizduojama raudona linija). Iš grafiko matyti, kad metinis publikacijų skaičius nagrinėjamu periodu keičiasi nežymiai. VILNIUS TECH mokslo darbuotojų parengtos publikacijos skirstomos į 7 mokslo sritis, kurios vaizduojamos grafike žemiau:



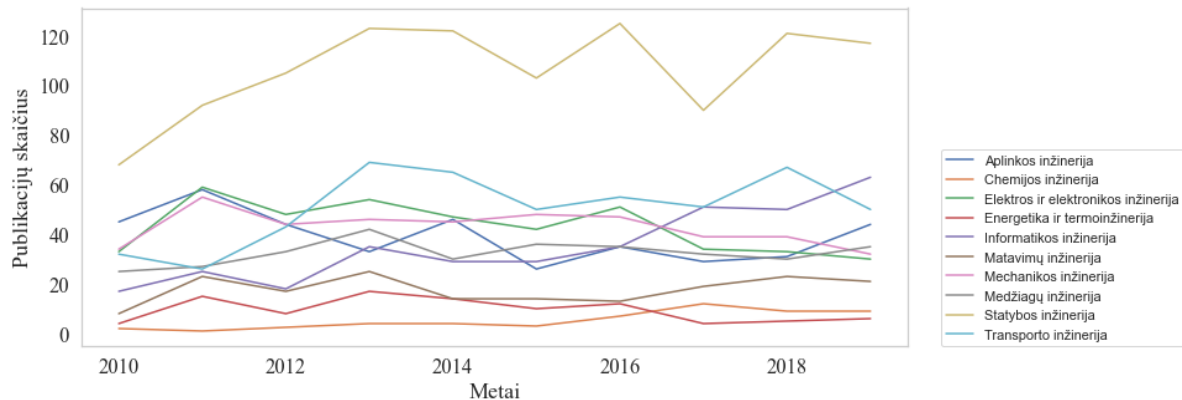
5 Pav. Publikacijų pasiskirstymas pagal mokslo sritis

Gauti rezultatai yra labai tikėti, nes nagrinėjami technologinių mokslų universiteto duomenys. Technologijos mokslai yra skaidomi į dar 10 smulkesnių dalių. 28% iš visų technologijos mokslų publikacijų priskiriama statybos inžinerijai, 13% – transporto inžinerijai, 11% – elektros ir elektronikos inžinerijai. Technologijos mokslams priskiriamų publikacijų pasiskirstymas pagal nagrinėjamas temas pateikiamas grafike žemiau:



6 Pav. Technologijos mokslų sritys

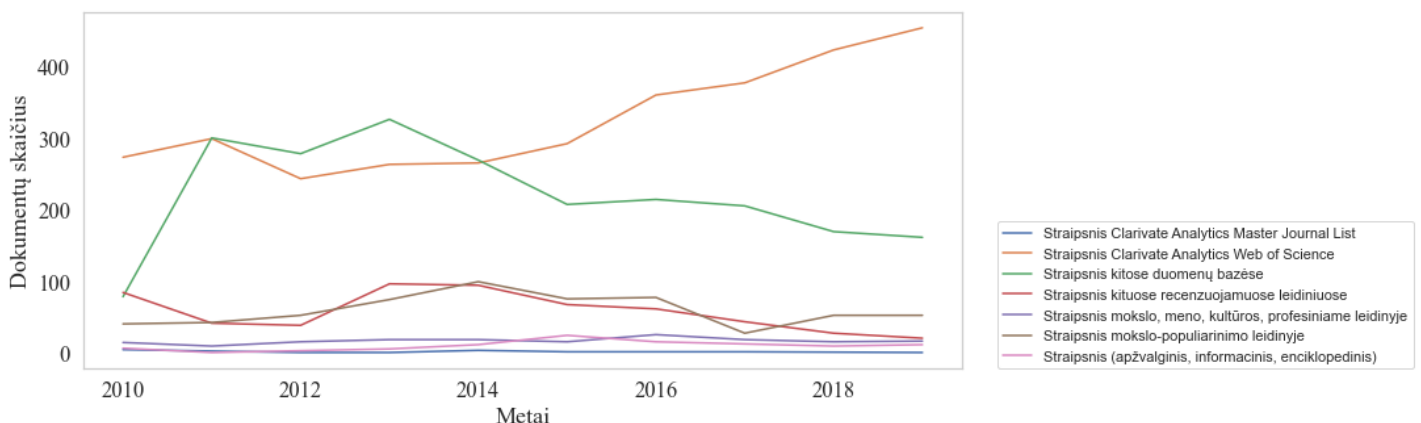
Šios 10 technologijų mokslo sričių taip pat pateikiamos kitu grafiniu pavidalu: vertinamas publikacijų skaičiaus pokytis šiose srityse nagrinėjamu periodu. Iš grafiko matyti, jog dauguma šių sričių laike nekinta labai ženkliai ir publikacijų skaičius daugumoje iš jų išlieka stabilus. Vienintelės statybos inžinerijos srities mokslinių straipsnių skaičius psikeitė ženkliai: pakilo 72%: nuo 68 publikacijų 2010 metais iki 117 publikacijų 2019 metais.



7 Pav. Technologijos mokslų sričių publikacijų skaičiaus pokytis 2010 – 2019 metais

Pagal žurnalą, kuriame moksliniai straipsniai buvo spausdinti, jie yra skirstomi į 7 grupes: straipsnis „Clarivate Analytics Web of Science“, straipsnis „Clarivate Analytics Master Journal List“, straipsnis kitose duomenų bazėse, straipsnis kituose recenzuojamuose leidiniuose, straipsnis mokslo-populiarinimo leidinyje, straipsnis mokslo, meno, kultūros, profesiniame leidinyje, straipsnis (apžvalginis, informacinis, enciklopedinis).

Publikacijų skaičiaus pokytis šiose srityse pateikiamas grafike žemiau. Iš jo matyti, kad didžiausias straipsnių skaičius publikuojamas „Clarivate Analytics Web of Science“ duomenų bazėje ir kitose duomenų bazėse. Publikacijų skaičius „Clarivate Analytics Web of Science“ duomenų bazėje per lyginamą periodą pakilo 66%: nuo 274 publikacijų 2010 metais iki 455 – 2019 metais.



8 Pav. Publikacijų skaičiaus pokytis 2010 – 2019 metais, pagal leidinius, kuriuose buvo spausdinama

Kita labai svarbi bibliometrinės analizės dalis – autorių analizė. Šiame duomenų rinkinyje iš viso yra paminėti 5 556 unikalūs autoriai. Vidutiniškai, 3 autoriai dalinasi vienos publikacijos bendraautoryste. Lentelėje žemiau pateikiamas bendraautorių skaičius ir šią bendraautorystę atitinkantis publikacijų skaičius:

4 lentelė. Bendraautorių skaičius

Bendraautorių skaičius	Publikacijų skaičius	Bendraautorių skaičius	Publikacijų skaičius
2	2 152	10	25
1	1 920	11	20
3	1 410	13	9
4	926	12	5
5	540	14	4
6	268	15	2
7	123	120	2
8	63	20	2
9	36	18, 19, 24, 26, 30, 36	1

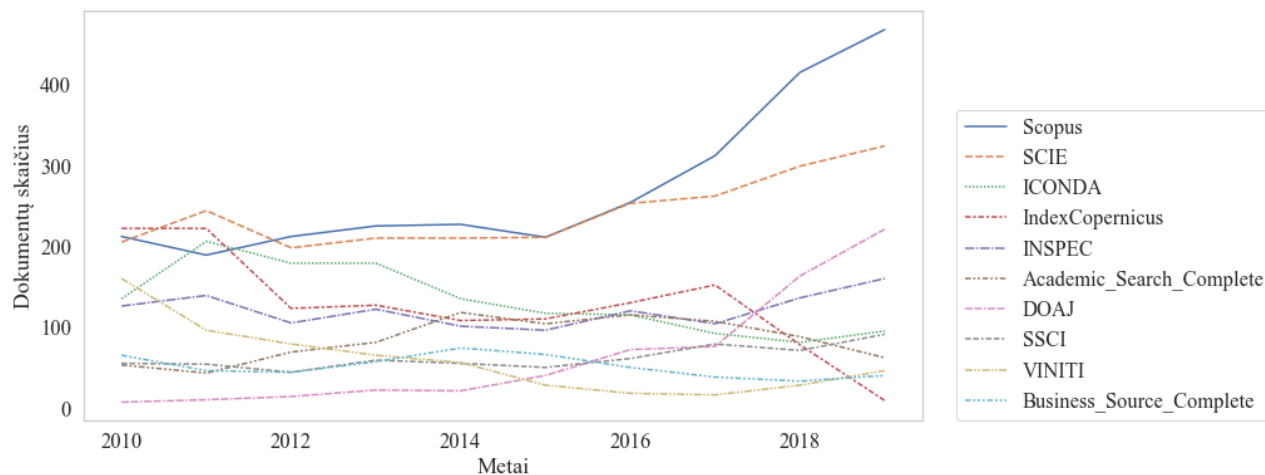
Iš lentelės rezultatų matoma, kad duomenų rinkinyje yra šios išskirtys: 2 publikacijos, kurių bendraautoryste dalinasi 120 autorių. Šios publikacijos yra parašytos temomis: „Strategies to manage hepatitis C virus infection disease burden“ ir „Historical epidemiology of hepatitis C virus (HCV) in select countries – volume 3“. Toliau nagrinėjami produktyviausi autoriai. Lentelėje žemiau pateikiami 10 autorių, kurie yra didžiausio publikacijų skaičiaus autoriai arba bendraautoriai:

5 lentelė. Produktyviausi autoriai

Autorius	Publikacijų skaičius
Zavadskas, E. K.	386
Turskis, Z.	114
Jučiučė, E.	110
Peleckis, K.	108
Antuchevičienė, J.	108
Baltrėnas, P.	99
Matijošius, J.	92
Samalavičius, A. L.	91
Tvaronavičienė, M.	90
Melnikas, B.	89

VILNIUS TECH mokslo darbuotojų publikacijos yra talpinamos įvairiose duomenų bazėse, tačiau detaliau nagrinėjamos 10 iš jų, kuriose publikacijos talpinamos daugiausiai. Jos yra: „Scopus“, „Science Citation Index Expanded (SCIE)“, „ICONDA“, „IndexCopernicus“, „INSPEC“, „Academic_Search_Complete“, „DOAJ“, „Social Sciences Citation Index (SSCI)“, „VINITI“, „Business_Source_Complete“. Grafike žemiau vaizduojama, kaip duomenų bazėse talpinamų publikacijų

skaičius kito nagrinėjamu periodu. Labiausiai išsiskiria „Scopus“ duomenų bazė, kurios publikacijų skaičius išaugo 120% nuo 2010 iki 2019 metų. „Science Citation Index Expanded (SCIE)“ duomenų bazėje talpinamų įrašų skaičius per šį periodą išaugo 56%, tuo tarpu „IndexCopernicus“ – sumažėjo 96%.



9 Pav. Duomenų bazėse talpinamų publikacijų skaičiaus pokytis 2010 – 2019 metais

4.5. Autorių bendradarbiavimo tinklas

4.5.1. Gephi duomenų rinkinių sudarymas ir tinklo vizualizavimas

Autorių sąryšių tinklui sudaryti buvo naudojamas vizualizavimo įrankis *Gephi*. Šis įrankis skirtas vizualizuoti ir nagrinėti didelės apimties grafams ir tinklams bei jų komponentų tarpusavio ryšiams. Prieš importuojant bet kokius duomenis į sistemą, juos reikia apdoroti kitoje, duomenų analizei skirtoje programoje. Nepatariama duomenų koreguoti tiesiogiai *Gephi* sistemoje. Duomenys importuojami naudojant kelis failus: viename nurodomi pagrindiniai ryšiai ir svoriai, kitame – atributai (angl. *edges and nodes*). Abu failai turi turėti bendrą ID stulpelį. Atributai suteikia galimybę duomenis filtruoti, grupuoti ir vizualiai išskirti grafuose (Van Eck ir Waltman, 2014).

Duomenų valymo etape buvo gautos visos autorių pavardės ir inicialai. Pašalinus dublikatus, išskirtas 5 556 unikalūs autoriai, kuriam priskirtas indeksas. Kitu žingsniu buvo sudaromos unikalios autorių bendradarbiavimo poros: jei publikacijos bendraautorystėje dalijasi 4 autoriai, turi būti gautos 6 autorių poros. Taigi kiekvienai duomenų rinkinio eilutei, kurioje yra išvardinti konkrečios publikacijos autoriai, turi būti apskaičiuoti visi galimi deriniai:

$$c(n, r) = \binom{n}{r} = \frac{n!}{r!(n-r)!}, 1 \leq r \leq n, \quad (44)$$

čia: c – baigtinė objektų aibė, turinti n elementų, junginiai iš r ($1 \leq r \leq n$) elementų. Šiam tikslui pasiekti ir visoms autorių poroms sudaryti buvo panaudota *Python* biblioteka *itertools*. Joje esanti funkcija *combinations* grąžina visas galimas įvesties poras. Gavus visas autorių poras, pašalinami dublikatai, o vietoje jų, pasikartojančioms autorių poroms suteikiamas svoris. Svoris apskaičiuojamas pagal tai, kiek kartų bendraautorystė pasikartojo duomenų rinkinyje. Kuo dažniau tie patys autoriai bendradarbiavo – tuo didesnis jų bendraautorystės svoris. Atlikus skaičiavimus gautos 27 633 bendraautorystės poros.

Siekiant rasti ir grafiškai pavaizduoti bendruomenes, esančias tinkle, jos buvo apskaičiuojamos naudojant *Python* biblioteką *NetworkX*, kuri yra skirta sudėtingų tinklų struktūrai, dinamikai ir funkcijoms kurti, manipuliuoti ir tirti. Šioje bibliotekoje esanti *girvan_newman* funkcija grąžina tinklo viršūnių sąrašus, suskirstytus pagal tinkle esančias bendruomenes.

Kad bibliografinių duomenų tinklas būtų sėkmingai atvaizduotas *Gephi* sistemoje, sukuriama 2 nauji duomenų rinkiniai:

6 lentelė. Duomenų rinkiniai grafo sudarymui

Viršūnės		Briaunos	
Stulpelio pavadinimas	Duomenys	Stulpelio pavadinimas	Duomenys
ID	Autoriaus unikalūs ID	Source	Pirmojo autoriaus ID
Label	Autoriaus pavardė ir inicialai	Target	Antrojo autoriaus ID
Weight	Autoriaus straipsnių skaičius	Type	Ryšio tipas (Undirected)
Group	Bendruomenės identifikatorius	Weight	Bendraautorystės pasikartojimų skaičius

Autoriaus straipsnių skaičius nurodo, kiek kartų konkretus autorius pasikartojo duomenų rinkinyje. Bendraautorystės ryšys pasirinktas kaip nekryptinis (angl. *Undirected*), nes autorių bendradarbiavimo ryšys laikomas vienodas abejomis kryptimis. Bendraautorystės pasikartojimų skaičius nurodo autorių tarpusavio bendradarbiavimo pasikartojimų skaičių duomenų rinkinyje. Sudarant bibliometrinį tinklą *Gephi* sistemoje, autoriaus straipsnių skaičius lems viršūnės dydį (kuo didesnė viršūnė, tuo daugiau autoriaus darbų nagrinėta), o bendraautorystės pasikartojimų skaičius – dviejų autorių bendradarbiavimo stiprumą arba dvi viršūnes jungiančios briaunos storį (kuo storesnė briauna, tuo dažniau pasikartojantis bendradarbiavimas).

Autorių tinklui sudaryti buvo pasirinktos tokios *Gephi* sistemos parametrų reikšmės, galimos „ForceAtlas2“ išdėstymo nustatymuose, ir sugeneruotas tinklo vaizdas:

ForceAtlas 2

Run

Threads

Threads number	7
----------------	---

Performance

Tolerance (speed)	1.0
Approximate Repulsion	☒
Approximation	1.2

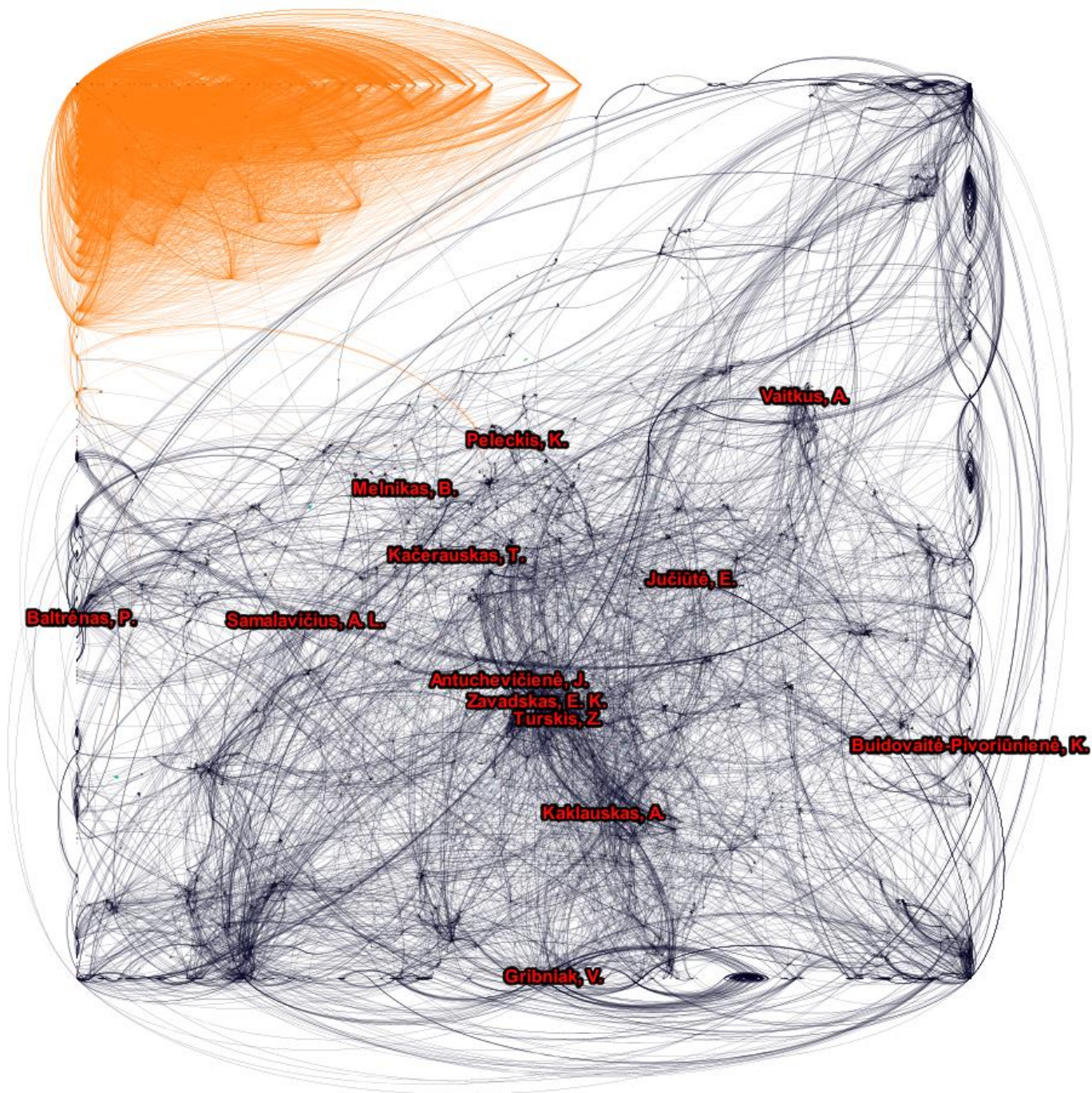
Tuning

Scaling	3.0
Stronger Gravity	☐
Gravity	2.0

Behavior Alternatives

Dissuade Hubs	☒
LinLog mode	☒
Prevent Overlap	☐
Edge Weight Influence	1.0
Normalize edge weights	☐

10 Pav. Tinklo parametrai

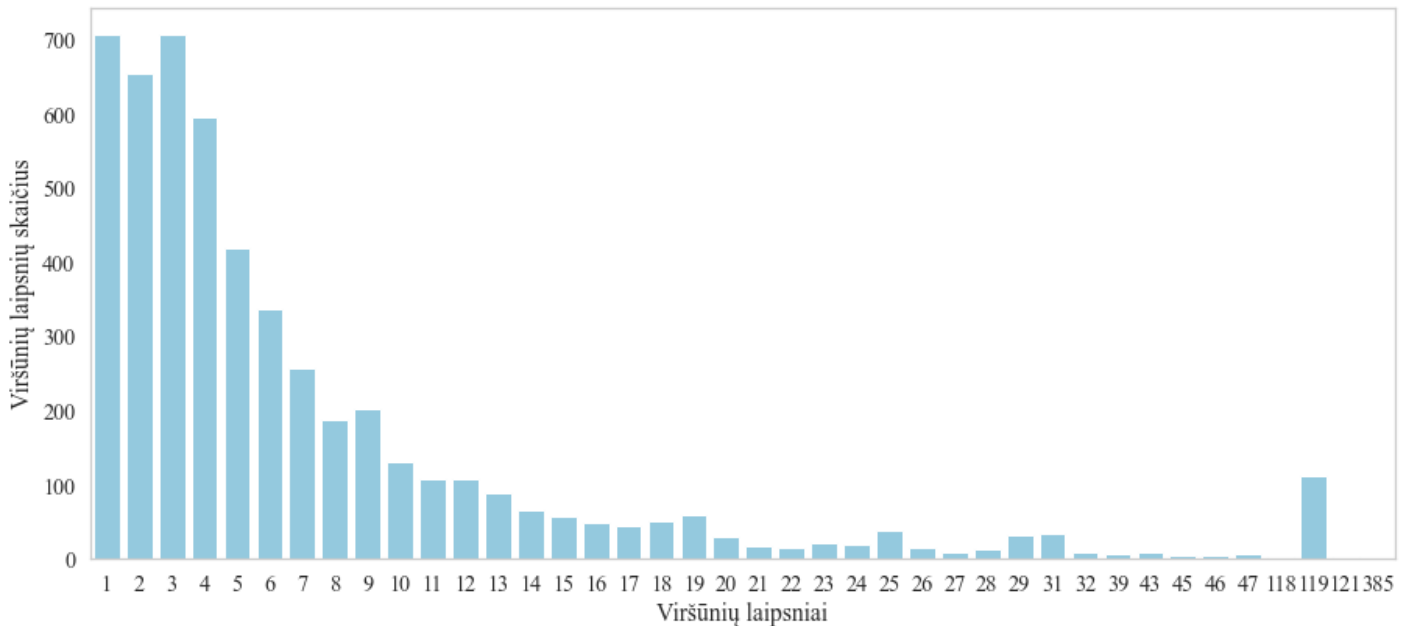


11 Pav. Visų autorių bendraautorystės tinklas

4.5.2. Autorių bendraautorystės tinklo analizė

Dalis sudaryto autorių bendraautorystės tinklo parametrų buvo vertinami tiesiogiai *Gephi* sistemoje, kita dalis apskaičiuoti naudojant *Python* biblioteką *NetworkX*. Tinklo savybės tirtos atsižvelgiant į rodiklius, aprašytus tinklų analizės skyriuje.

Analizė pradėta nuo tinklo viršūnių: naudojantis *NetworkX* biblioteka, rasti visų tinkle esančių viršūnių laipsniai ir jų pasikartojimo dažnis atvaizduotas histogramoje. Dažniausiai pasikartojantys laipsniai yra: 1 (708 viršūnės), 3 (707 viršūnės), 2 (654 viršūnės), 4 (596 viršūnės), 5 (419 viršūnės):



12 Pav. Viršūnių laipsniai

Naudojantis *Python* biblioteka *NetworkX*, rasti gautų laipsnių statistiniai rodikliai:

7 lentelė. Laipsnių statistiniai rodikliai

Viršūnių skaičius	Min laipsnis	Max laipsnis	Laipsnių vidurkis	Dispersija	Antisimetriš- kumas	Ekscesas (Kurtosis)
5 556	1	385	10,3	403,6	5,3	41,4

Plačiau nagrinėjami antisimetriškumo ir eksceso rodikliai. Antisimetriškumo rodiklis apibūdina duomenų (šiuo atveju visų viršūnių laipsnių) simetriją arba jos nebuvimą. Jei šio rodiklio reikšmė yra arti nulio – nagrinėjami duomenys yra iš normaliojo skirstinio. Jei rodiklis įgyja teigiamas arba neigiamas reikšmes – duomenys yra pasiskirstę asimetriškai. Antisimetriškumo rodiklis apskaičiuojamas pagal formulę:

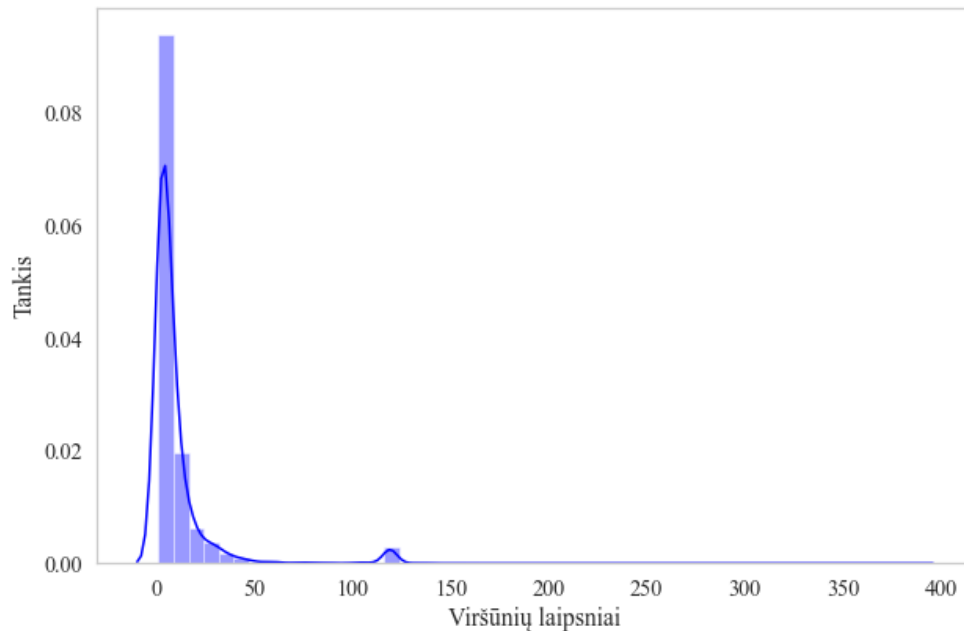
$$g_1 = \frac{\sum_{i=1}^N (Y_i - \bar{Y})^3 / N}{s^3}, \quad (45)$$

čia Y_i – kintamasis iš imties N , $\bar{Y}$ – vidurkis, s – standartinis nuokrypis. Eksceso rodiklis taip pat naudojamas įvertinti duomenų pasiskirstymui. Jei nagrinėjami duomenys yra iš normaliojo skirstinio, eksceso rodiklio reikšmė yra lygi nuliui. Jei gaunama teigiama reikšmė, ji nurodo, kad pasiskirstymas yra su didele liekana,

jei neigiama – pasiskirstymas yra su maža liekana. Eksceso rodiklis apskaičiuojamas pagal panašią formulę, kaip ir antisimetriškumas:

$$g_2 = \frac{\sum_{i=1}^N (Y_i - \bar{Y})^4 / N}{s^4}. \quad (46)$$

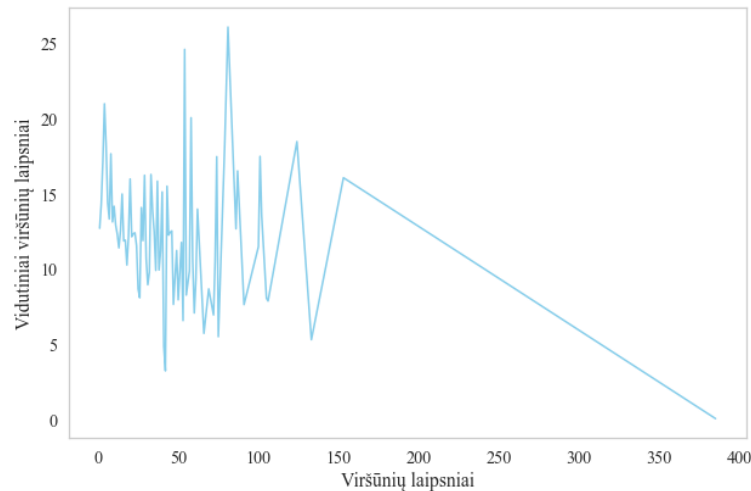
Šie du rodikliai dažniausiai yra vizualizuojami duomenų pasiskirstymui iliustruoti:



13 Pav. Antisimetriškumo ir eksceso rodikliai

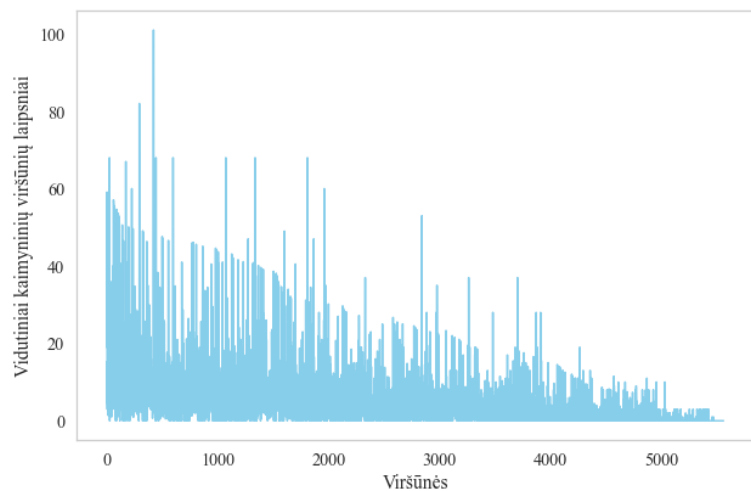
Iš gautų statistinių rodiklių ir grafiko matoma, kad viršūnių laipsnių pasiskirstymas nėra simetriškas ir tai reiškia, kad jis nėra iš normaliojo skirstinio. Didelė eksceso rodiklio reikšmė (41,4) reiškia, kad duomenyse yra išskirčių.

Grafo vidutiniai jungumo matai apskaičiuoti naudojantis *Python* biblioteka *NetworkX*. Vidutinis laipsnių jungumas nurodo vidutinį viršūnių laipsnį kiekvienam tinkle esančiam viršūnių laipsniui. Iš grafiko matyti, kad viršūnės su mažesniu laipsniu vidutiniškai jungiasi su kitomis viršūnėmis, turinčiomis artimą vidutinį viršūnių laipsnių skaičių. Tačiau viršūnės, turinčios didesnį laipsnį – jungiasi su mažesnių vidutinį laipsnį turinčiomis viršūnėmis:



14 Pav. Vidutinis laipsnių jungumas

Vidutinis kaimyninių viršūnių laipsnis apskaičiuojamas su *Python* biblioteka *NetworkX* kiekvienai viršūnei ir pavaizduojamas grafike. Iš jo matoma, kad didėjant viršūnių laipsniams, kaimyninių viršūnių vidutiniai laipsniai mažėja:



15 Pav. Vidutinis kaimyninių viršūnių laipsnis

Kitu žingsniu apskaičiuojami keli rodikliai: grafo tankumas, laipsnių koreliacijos koeficientas, grafo skersmuo, vidutinis atstumas tarp grafo viršūnių ir klasterizacijos koeficientas. Pirmieji du rodikliai randami naudojantis *Python NetworkX* biblioteka, likę – naudojantis *Gephi* sistema:

8 lentelė. Grafo statistiniai rodikliai

Grafo tankumas	Laipsnių koreliacijos koeficientas	Klasterizacijos koeficientas	Grafo skersmuo	Vidutinis atstumas tarp grafo viršūnių
0,002	0,3	0,79	14	5

Grafo tankumas parodo santykį tarp grafe esančių briaunų ir didžiausio įmanomo grafo briaunų skaičiaus. Grafo tankumas gali įgyti reikšmės intervale $[0, 1]$, kur 1 apibrėžia pilnąjį grafą, kurio kiekviena viršūnė yra tiesiogiai sujungta su kiekviena kita viršūne. Autorių bendradarbiavimo tinklo tankumas yra artimas 0, vadinasi jo viršūnės tiesiogiai tarpusavyje jungiasi mažai, bet jis išlieka jungus grafas.

Laipsnių koreliacijos koeficientas gali įgyti reikšmės intervale $[-1, 1]$. Jei koeficiento reikšmė yra mažesnė už 0, toks tinklas apibūdinamas kaip turintis panašių laipsnių viršūnės greta vienos kitų; jei reikšmė yra artima 0 – tinklas laikomas neutraliu, o jei reikšmė didesnė už 0 – toks tinklas apibūdinamas kaip neturintis panašių laipsnių viršūnių greta vienos kitų. Socialiniams tinklams būdingas didesnis laipsnių koreliacijos koeficientas, nes platų socialinį ratą turintys individai dažnai turi sąsają su kitais individais, turinčiais platų socialinį ratą ir atvirkščiai: nestipriai susieti individai lieka mažiau sujungti su visu tinklu.

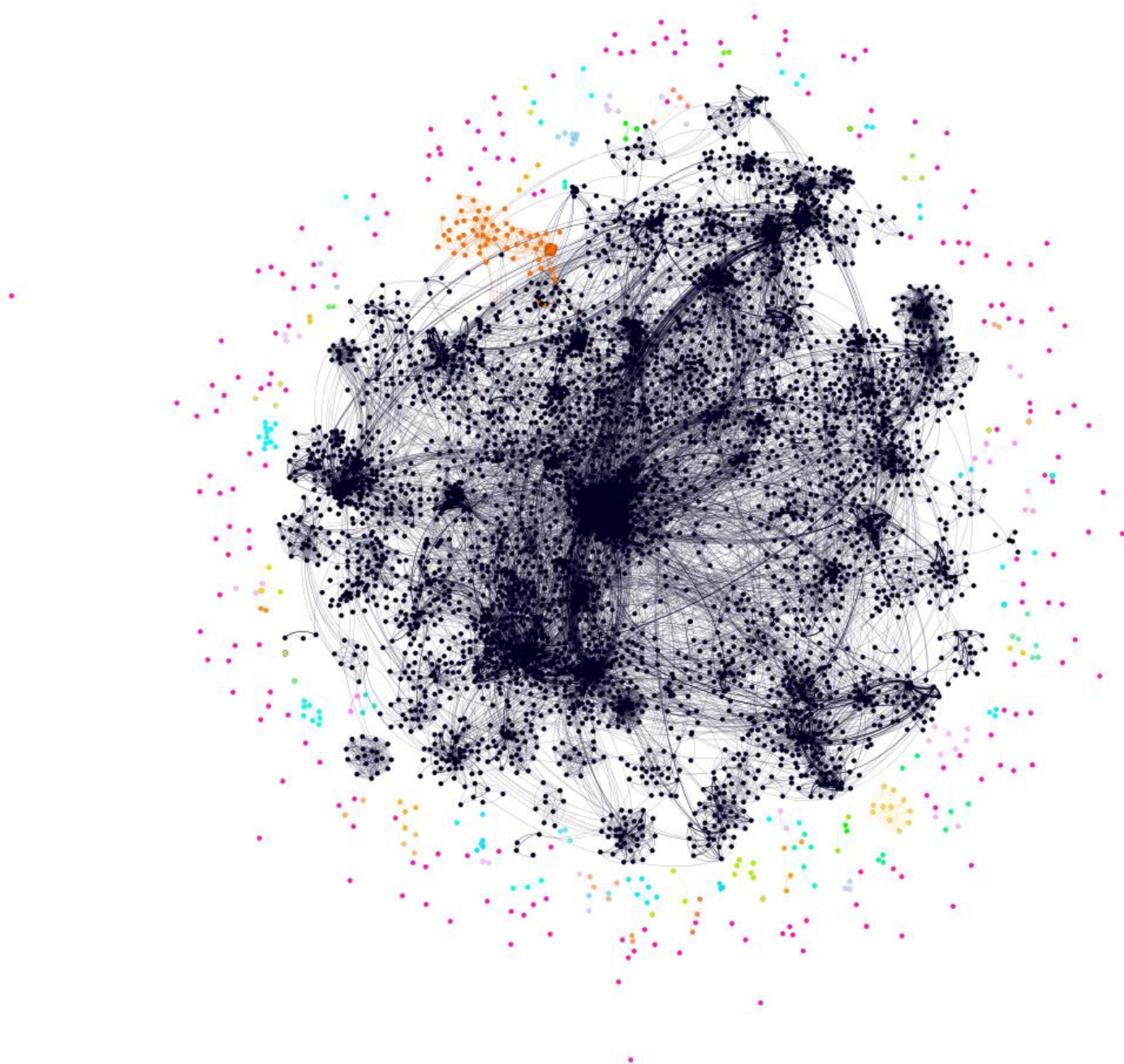
Klasterizacijos koeficientas parodo, kiek viršūnės v kaimynų yra ir vienas kito kaimynai ir kaip stipriai kaimynai yra sujungti vieni su kitais. Šis koeficientas gali įgyti reikšmės intervale $[0, 1]$. Jei reikšmė yra artima 1 – kaimynai yra pilnai sujungti vieni su kitais, jei reikšmė yra arti 0 – kaimyninės viršūnės yra beveik nesujungtos. Autorių bendradarbiavimo tinklo klasterizacijos koeficientas yra artėjantis prie 1, todėl galima teigti, kad kaimyninės viršūnės yra pakankamai stipriai sujungtos.

Naudojantis *Python* biblioteka *NetworkX*, apskaičiuojami centriškumo matai: laipsnių centriškumo matas, artumo centriškumo matas ir tarpusavio centriškumo matas. Jų rezultatai: koeficientai, priskirti kiekvienai viršūnei. Buvo nuspręsta pateikti 10 viršūnių su didžiausiomis koeficientų reikšmėmis iš kiekvieno skaičiavimo. Gauti rezultatai parodo, kad dalis viršūnių, kurios yra identifikuotos kaip svarbiausios tinkle, nebūtinai turi daugiausia trumpiausių ryšių su kitais nariais:

9 lentelė. Grafo centriškumo matai

Laipsnių centriškumas Svarbiausi tinklo nariai		Artumo centriškumas Nariai, esantys arčiausiai kitų narių		Tarpusavio centriškumas Nariai, turintys daugiausiai trumpiausių ryšių su kitais nariais	
Zavadskas, E. K.	0,072	Zavadskas, E. K.	0,188	Jučiuotė, E.	0,004
Kaklauskas, A.	0,029	Žemeckytė, L.	0,143	Kačianauskas, R.	0,004
Liakina, V.	0,025	Zolfani, S.	0,14	Kaklauskas, A.	0,003
Turskis, Z.	0,025	Zhou, W.	0,138	Kilikevičius, A.	0,002
Antuchevičienė, J.	0,023	Turskis, Z.	0,102	Turskis, Z.	0,001
Razavi, S. H.	0,023	Vilutienė, T.	0,099	Prentkovskis, O.	0,001
Abaalkhail, F.	0,022	Vinogradova, I.	0,097	Kaklauskas, G.	0,001
Hunsi, R.	0,022	Zakarevičius, A.	0,093	Gribniak, V.	0,001
Paršeliūnas, E. K.	0,02	Yazdani-Chamzini, A.	0,092	Laurinavičius, A.	0,001
Gribniak, V.	0,02	Ustinovičius, L.	0,091	Matijošius, J.	0,001

Galiausiai, naudojantis *Python* bibliotekoje *NetworkX* esančia *girvan_newman* funkcija, identifiкуotos autorių bendraautorystės tinkle esančios bendruomenės. Jų iš viso rasta 81, tačiau jų dydžiai, kaip galima pastebėti 11 paveiksle, labai skiriasi. Kadangi mažesnės bendruomenės yra beveik nematomos visoje tinklo struktūroje, sukuriamas kitas tinklo vaizdas, kuriame dėmesys skiriamas bendruomenių išryškinimui:



16 Pav. Bendruomenės autorių tinkle

Lentelėje žemiau pateikiama, kiek viršūnių sudaro kiekvieną bendruomenę:

10 lentelė. Viršūnių pasiskirstymas bendruomenėse

Bendruomenės numeris	Viršūnių skaičius bendruomenėje
1	4 892
2	174
3 – 4	11
5 – 6	8
7	7
8 – 9	6
10 – 20	4
21 – 41	3
42 – 80	2
81	195

Iš rezultatų lentelėje bei grafo vaizdo (16 paveikslas) matoma, kad dauguma viršūnių priklauso vienai didelei bendruomenei. Tai nėra netikėtas rezultatas, nes nagrinėjami vienos institucijos mokslo darbuotojų duomenys. Rezultatai atskleidžia, kad mokslo darbuotojai yra glaudžiai susiję bendraautorystės ryšiais. Tačiau šiame tinkle (tai labai ryškiai matoma 11 paveiksle) galima išvelgti vieną atskirą bendruomenę. Ji susiformavo, nes kelios publikacijos yra parašytos labai didelio bendraautorių skaičiaus, kurie beveik neturi bendraautorystės ryšių su kitais autoriais (jos buvo aptartos 4.4 skiriuje). Galiausiai, lentelės apačioje esanti bendruomenė, kuri turi pakankamai daug narių, buvo suformuota papildomai. Į ją priskirtos pavienės viršūnės, kurios nebuvo priskirtos nei vienai bendruomenei sistemos automatiškai. Taip nutiko, nes jos vaizduoja autorius, kurie publikacijas atliko vieni, taigi jie neturi jokio ryšio su likusiu tinklu.

Siekiant geriau suprasti, kaip atskiros bendruomenės susiformuoja bibliografiniame tinkle, detaliau nagrinėjamos kelios iš jų. Vieną iš bendruomenių sudaro 11 narių, jie surandami duomenų faile ir grąžinamos autorių pavardės. Rezultatai patikiami lentelėje žemiau:

11 lentelė. Bendruomenės iš 11 narių analizė

Viršūnės ID	Autoriaus pavardė	Viršūnės svoris
1508	Gurevičius, K.	1
1946	Kalesnykas, G.	1
2246	Kontkanen, O.	1
2550	Leinonen, H.	1
2635	Lipponen, A.	1
3330	Nymark, S.	1
3737	Puolivali, J.	1
3742	Puranen, J.	1
3796	Ragauskas, S.	1
3964	Ronkko, S.	1
4822	Tanila, H.	1

Visų viršūnių, priklausančių šiai bendruomenei svoriai yra lygūs 1, o tai reiškia, kad duomenų rinkinyje šie autoriai pasikartojo tik po vieną kartą. Suradus visus autorius duomenų rinkinyje matoma, kad jie visi yra vienos publikacijos bendraautoriai: „Early retinal function deficit without prominent morphological changes in the R6/2 mouse model of Huntington's disease“.

Kitą nagrinėjamą bendruomenę sudaro 3 nariai, informacija apie juos pateikiama lentelėje žemiau:

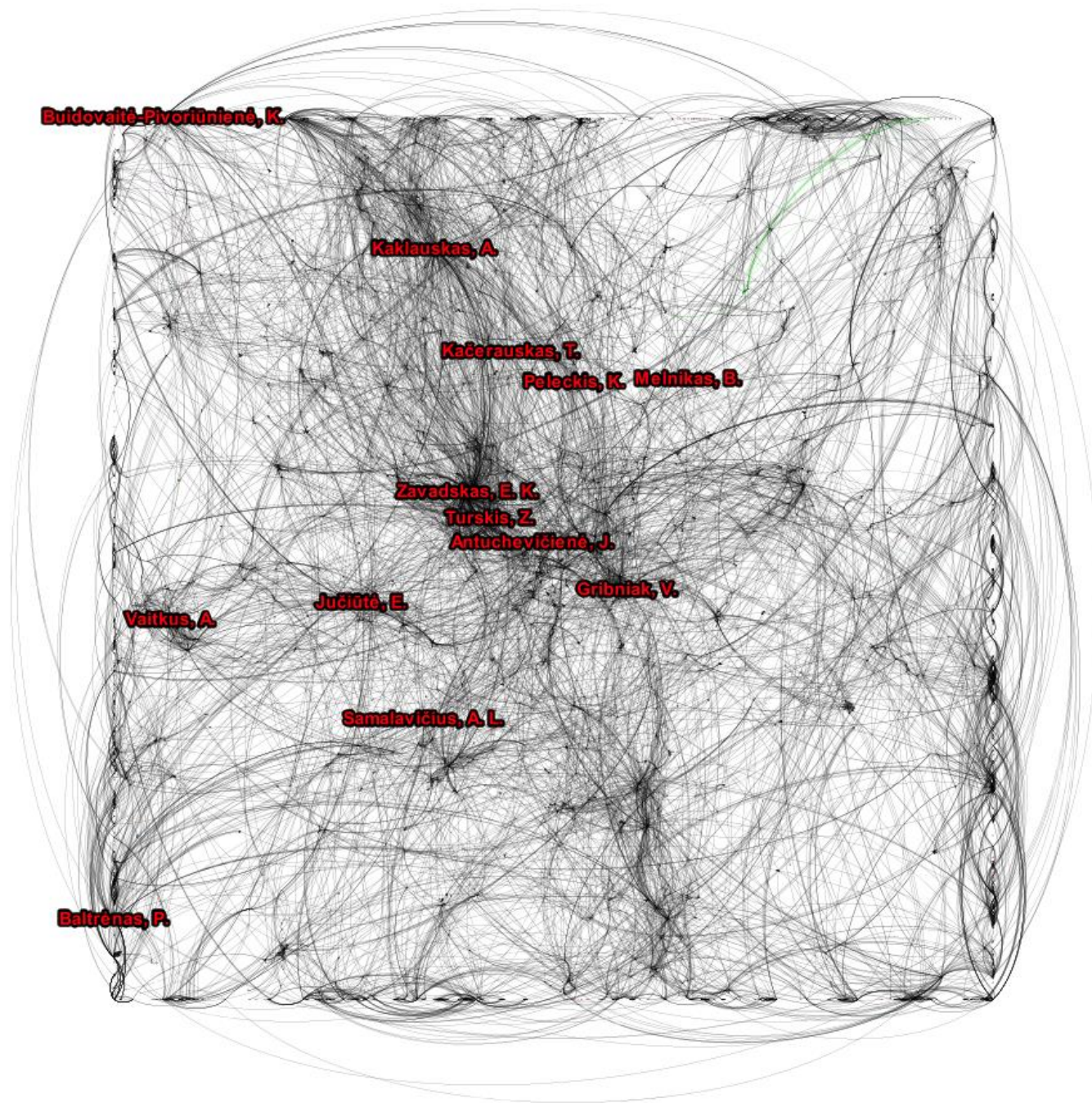
12 lentelė. Bendruomenės iš 3 narių analizė

Viršūnės ID	Autoriaus pavardė	Viršūnės svoris
4016	Rusko, T.	11
3767	Račienė, E.	6
3550	Petrauskaitė, A.	1

Šie rezultatai skiriasi nuo prieš tai analizuotos bendruomenės. T. Rusko autorius duomenų rinkinyje pasikartojo 11 kartų, jis yra vienintelis 9 publikacijų autorius, likusių dviejų publikacijų bendraautorystėje jis dalijasi su E. Račiene ir A. Petrauskaite. E. Račienė yra 6 publikacijų autorė ir vienos iš jų bendraautorystėje dalijasi su T. Rusko. A. Petrauskaitė duomenų rinkinyje pasikartojo tik vieną kartą ir publikacijos bendraautorystėje dalijasi su T. Rusko. Taigi iš šių rezultatų matosi, kad minėti trys autoriai yra susiję tik tarpusavyje ir neturi kitų ryšių su likusiais tinklo nariais.

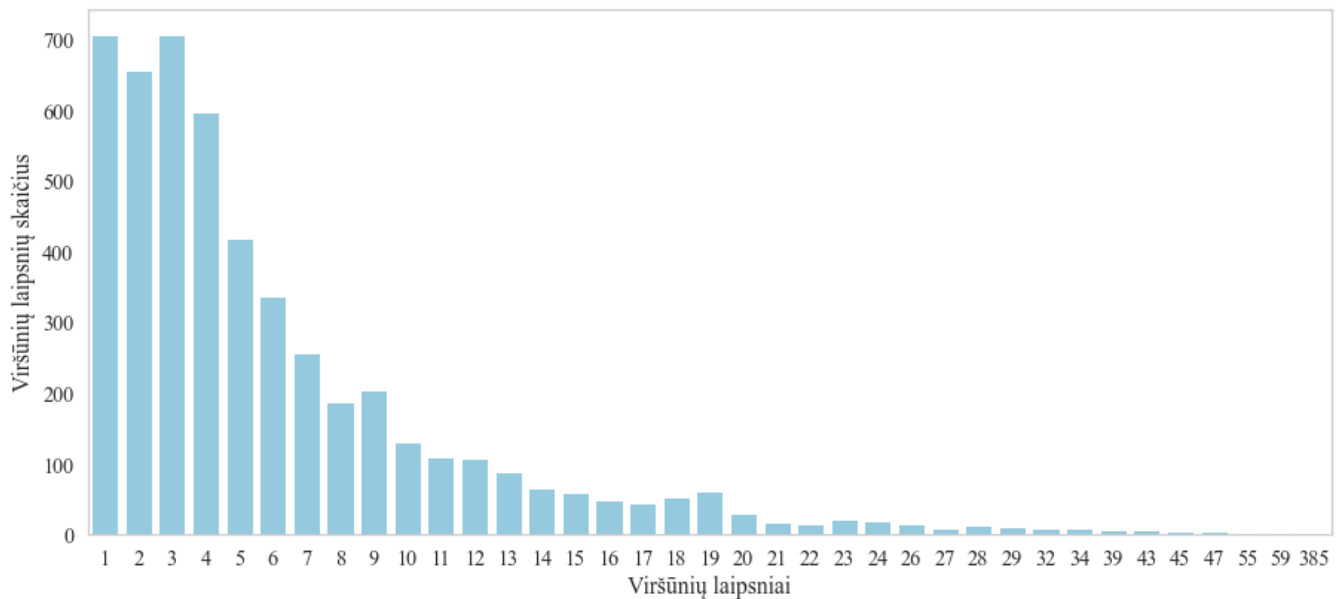
4.5.3. *Autorių bendraautorystės tinklas pašalinus išskirtis*

Iš bendraautorių tinklo, vaizduojamo paveiksle 11 matoma, kad kelios publikacijos, kurios turi itin didelį bendraautorystės ryšių skaičių, daro didelę įtaką visam tinklui. Šių publikacijų įtaka visam tinklui yra tokia didelė, nes šių publikacijų bendraautorių skaičius yra >20 (kelių iš jų siekia net 120), tačiau bendraautorystės ryšių beveik nėra su likusiais tinklo nariais. Įvertinus šią įtaką, buvo nuspręsta pašalinti 6 publikacijas, kurių bendraautorių skaičius yra didesnis už 20. Tai atlikus, sukurtas naujas tinklo vaizdas bei perskaičiuoti prieš tai aptarti tinklo rodikliai.



17 Pav. Autorių bendraautorystės tinklas, pašalinus išskirtis

Naujai sudarytame bendraautorystės tinkle rasti visų viršūnių laipsniai ir jų pasikartojimo dažnis atvaizduotas histogramoje. Pagrindinių laipsnių skaičiai beveik nepasikeitė, tačiau iš grafiko matoma, kad nebeliko viršūnių, turinčių ypač didelius laipsnius:



18 Pav. Viršūnių laipsniai

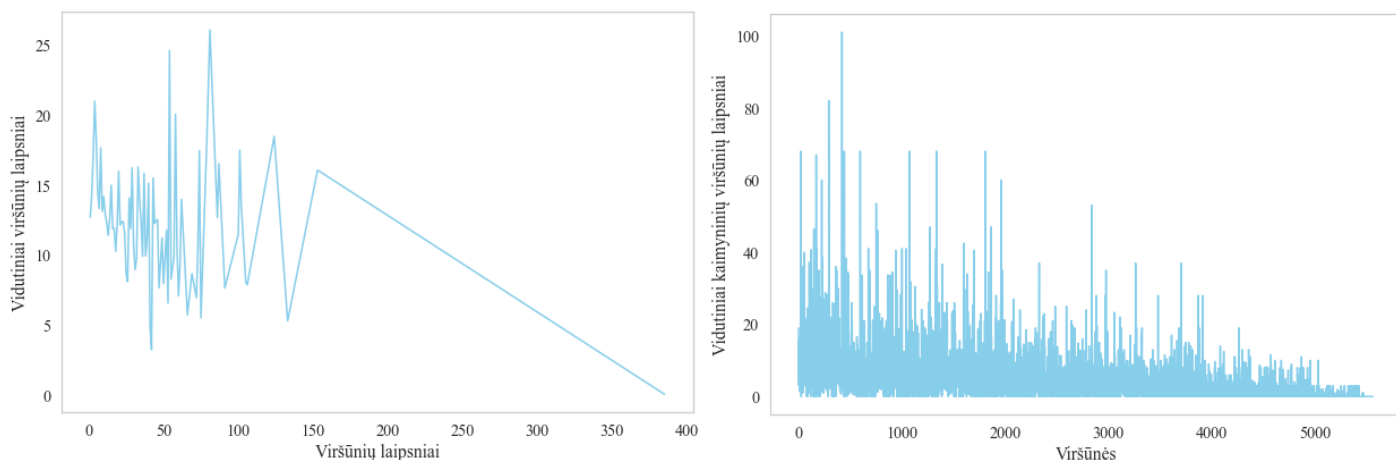
Rasti gautų laipsnių statistiniai rodikliai:

13 lentelė. Laipsnių statistiniai rodikliai

Viršūnių skaičius	Min laipsnis	Max laipsnis	Laipsnių vidurkis	Dispersija	Antisimetriš- kumas	Ekscesas (Kurtosis)
5 550	1	385	7,4	124,7	10,8	275,2

Iš gautų rezultatų matoma, kad pasikeitė vidutinė laipsnių vertė, padidėjo dispersija bei antisimetriškumo ir eksceso rodikliai. Šie rodikliai leidžia daryti išvadą, kad duomenys yra pasiskirstę asimetriškai ir šis pasiskirstymas yra su didele liekana.

Kitu žingsniu perskaičiuojami ir grafiškai pavaizduojami vidutinis laipsnių jungumas ir vidutiniai kaimyninių viršūnių laipsniai. Iš vidutinio laipsnių jungumo grafiko matyti, kad viršūnių jungimasis su vidutinius laipsnius turinčiomis viršūnėmis beveik nepakito, tuo tarpu vidutinis kaimyninių viršūnių laipsnių grafikas pasikeitė. Nebėra tokios griežtos mažėjimo tendencijos, kuomet didėjant viršūnių laipsniams, kaimyninių viršūnių laipsniai tolygiai mažėja.



19 Pav. Vidutinis laipsnių jungumas ir vidutiniai kaimyninių viršūnių laipsniai

Perskaičiuojami naujai sudaryto grafo rodikliai: grafo tankumas, laipsnių koreliacijos koeficientas, grafo skersmuo, vidutinis atstumas tarp grafo viršūnių ir klasterizacijos koeficientas:

14 lentelė. Grafo statistiniai rodikliai

Grafo tankumas	Laipsnių koreliacijos koeficientas	Klasterizacijos koeficientas	Grafo skersmuo	Vidutinis atstumas tarp grafo viršūnių
0,001	-0,07	0,78	12	5

Naujai sudaryto grafo tankumas sumažėjo, taip pat sumažėjo ir laipsnių koreliacijos koeficientas. Klasterizacijos koeficientas liko beveik nepakitęs, o tai reiškia, kad kaimyninių viršūnių tarpusavio sujungimas beveik nepakito. Vidutinis atstumas tarp grafo viršūnių liko toks pat, o grafo skersmuo sumažėjo nuo 14 iki 12.

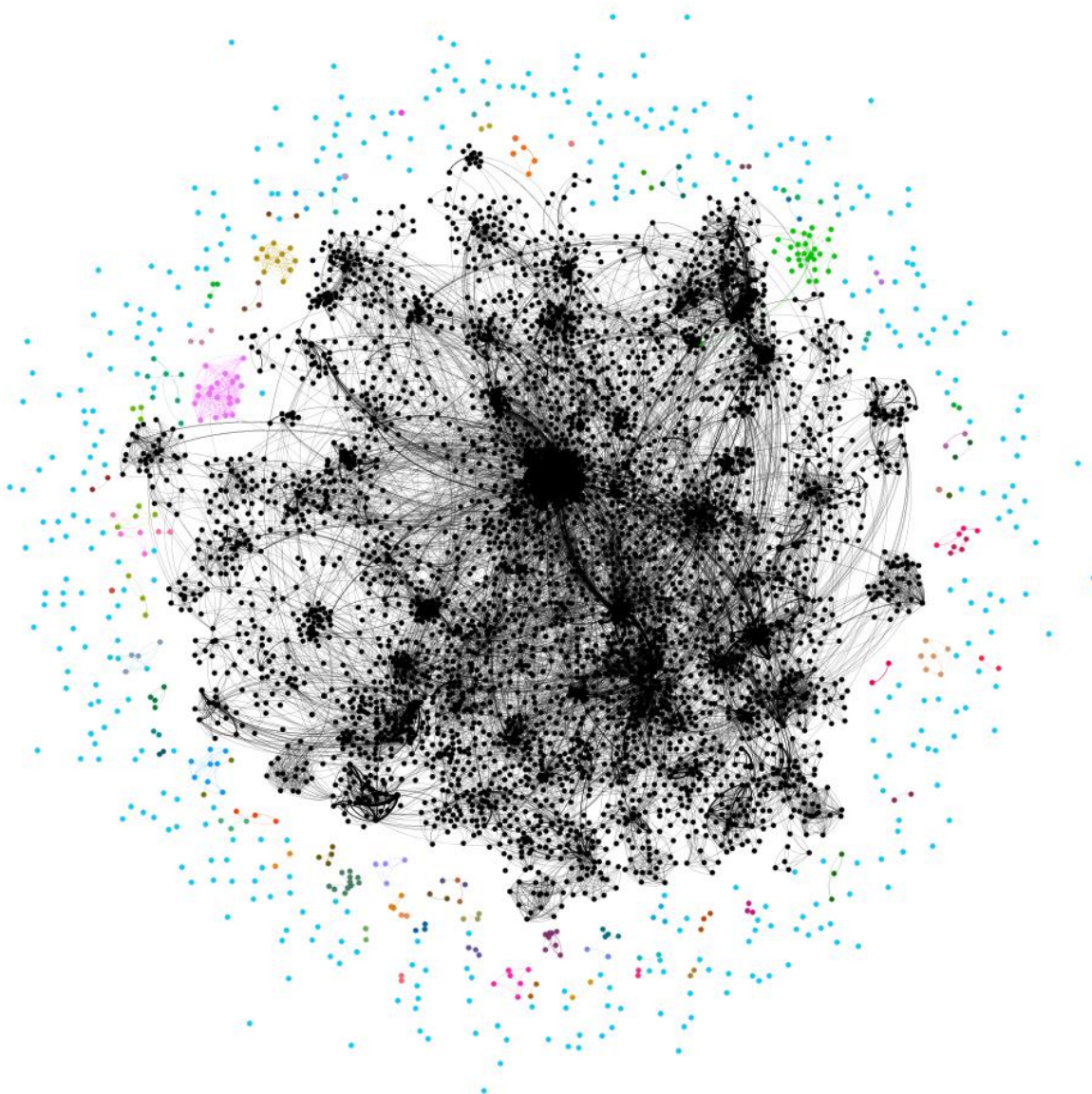
Toliau apskaičiuojami centriškumo matai ir pateikiami lentelėje žemiau.

15 lentelė. Grafo centriškumo matai

Laipsnių centriškumas Svarbiausi tinklo nariai		Artumo centriškumas Nariai, esantys arčiausiai kitų narių		Tarpusavio centriškumas Nariai, turintys daugiausiai trumpiausių ryšių su kitais nariais	
Zavadskas, E. K.	0,074	Zavadskas, E. K.	0,183	Jučiūtė, E.	0,004
Kaklauskas, A.	0,03	Žemeckytė, L.	0,14	Kačianauskas, R.	0,004
Turskis, Z.	0,026	Zolfani, S.	0,137	Kaklauskas, A.	0,003
Antuchevičienė, J.	0,024	Zhou, W.	0,136	Kilikevičius, A.	0,002
Prentkovskis, O.	0,02	Turskis, Z.	0,103	Turskis, Z.	0,002
Kačianauskas, R.	0,02	Vilutienė, T.	0,1	Prentkovskis, O.	0,001
Kilikevičius, A.	0,02	Vinogradova, I.	0,099	Laurinavičius, A.	0,001
Jučiūtė, E.	0,019	Zakarevičius, A.	0,094	Kaklauskas, G.	0,001
Matijošius, J.	0,019	Yazdani-Chamzini, A.	0,093	Matijošius, J.	0,001
Rimkus, A.	0,018	Ustinovičius, L.	0,093	Kliukas, R.	0,001

Lyginant šiuos rezultatus su prieš tai gautais matais, matoma, kad daugiausia pasikeitimų yra susiję su laipsnių centriškumu. Tai reiškia, kad pasikeitė svarbiausi tinklo nariai. Artumo matai pasikeitė labai nežymiai ir prieš tai arčiausiai kitų narių išvardinti autoriai liko tokie patys. Tarpusavio centriškumo lentelėje taip pat išliko tie patys nariai, tik pasikeitė dalies iš jų pozicija.

Paskutiniu žingsniu, nagrinėjami bendruomenių pasikeitimai. Tinkle, kuris vaizduojamas 17 paveiksle, matoma, kad pašalinus išskirtis, autoriai pasiskirstė tolygiau bei išryškėjo kelios kitos bendruomenės. Iš viso jų išskirta – 82, viena daugiau, nei ankstesniu atveju. Tačiau kaip ir pirmuoju atveju, jas nagrinėti tokiam išdėstyme yra sudėtinga, dėl to sukuriamas naujas grafo išdėstymas, kurio akcentas yra bendruomenės:



20 Pav. Bendruomenės autorių tinkle be išskirčių

Lentelėje žemiau pateikiama, kiek viršūnių sudaro kiekvieną bendruomenę:

16 lentelė. Viršūnių pasiskirstymas bendruomenėse

Bendruomenės numeris	Viršūnių skaičius bendruomenėje
1	4 892
2	29
3	20
4 – 5	11
6 – 7	8
8	7
9 – 10	6
11 – 20	4
21 – 42	3
43 – 81	2
82	379

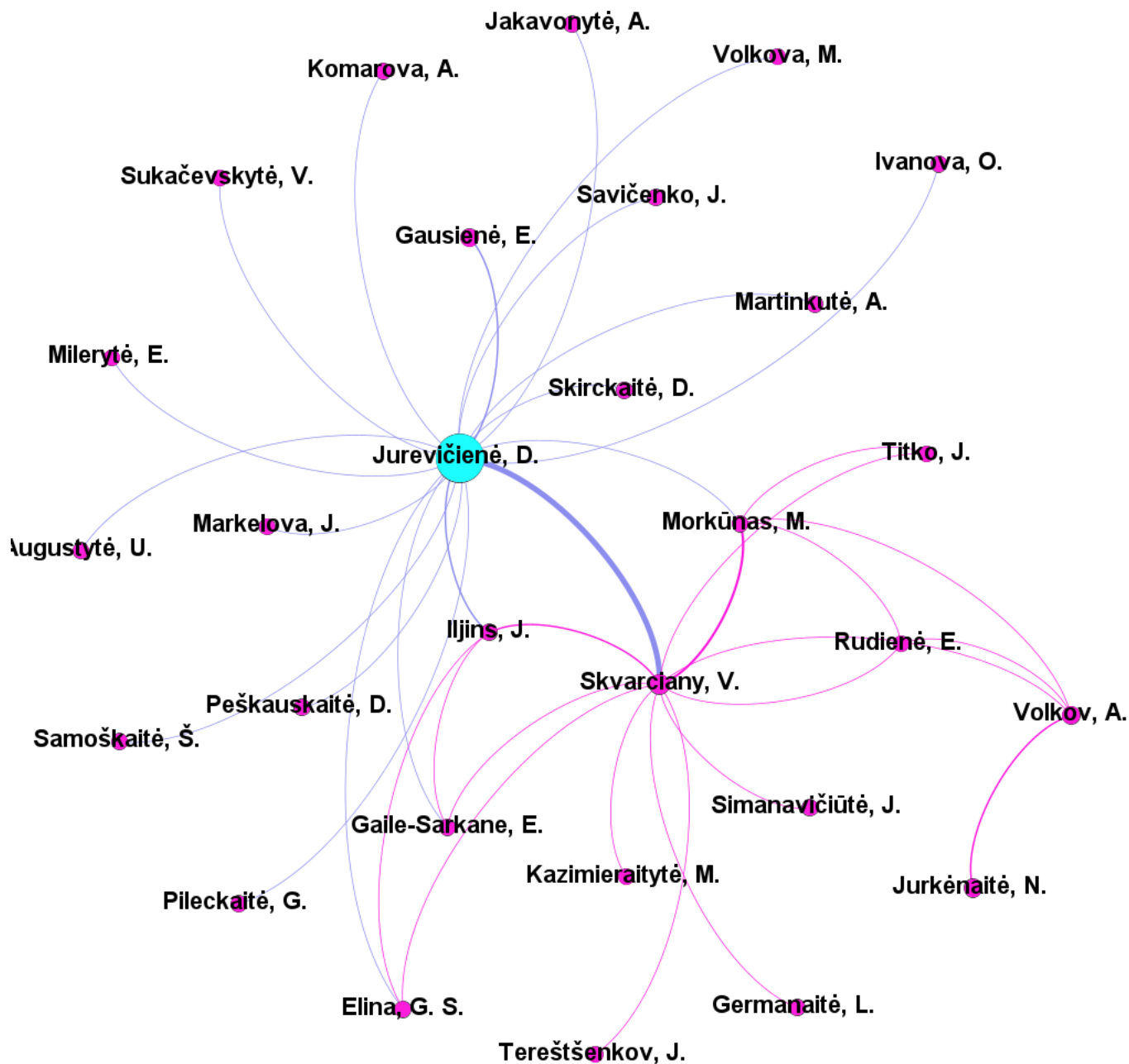
Gautas rezultatas, kad dauguma viršūnių priklauso vienai didelei bendruomenei, nepasikeitė. Pašalinus išskirtis, buvo pašalinta ir antroji pagal dydį bendruomenė. Šis pasikeitimas lėmė, kad atsirado kitos dvi bendruomenės, kurių nebuvo anksčiau. Likusios bendruomenės nepakito. Paskutinėje lentelės eilutėje, kaip ir pirmuoju atveju, pateikiamos pavienės viršūnės, kurios nebuvo priskirtos nei vienai bendruomenei sistemos automatiškai. Pašalinus išskirtis, jų skaičius išaugo.

Detaliau analizuojama viena iš naujai susidariusių bendruomenių, kurią sudaro 29 nariai. Suradus šiuos narius duomenų rinkinyje matoma, kad dauguma iš jų pasikartoja tik vieną kartą, tačiau keli nariai pasikartoja dažniau. Rezultatai pateikiami lentelėje žemiau:

17 lentelė. Viršūnių pasiskirstymas naujai susiformavusioje bendruomenėje

Autoriaus pavardė	Viršūnės svoris
Jurevičienė, D.	18
Skvarciany, V.	3
Gausienė, E., Jurkėnaitė, N.	2
Augustytė, U., Elina, G. S., Gaile-Sarkane, E., Germanaitė, L., Iljins, J., Ivanova, O., Jakavonytė, A., Kazimieraitytė, M., Komarova, A., Markelova, J., Martinkutė, A., Milerytė, E., Morkūnas, M., Peškauskaitė, D., Pileckaitė, G., Rudienė, E., Samoškaitė, Š., Savičenko, J., Simanavičiūtė, J., Skirckaitė, D., Sukačevskytė, V., Tereštšenkova, J., Titko, J., Volkova, M.	1

Rezultatai, pateikiami lentelėje, yra gana sunkiai suprantami, nes nėra aišku, kaip šie autoriai susiję tarpusavyje. Dėl to sukuriamas atskiras bibliografinis tinklas tik šiai bendruomenei vaizduoti:



21 Pav. Bendruomenės grafinis vaizdavimas

Iš gauto grafo iš karto galima daryti išvadas apie šią bendruomenę: jos centras yra viena autorė, Jurevičienė, D., kuri bendraautorystę dalinasi su daugeliu iš bendruomenės narių. Matoma, kad dalis narių yra susiję tarpusavyje, tačiau taip pat yra ir viršūnių, kurios su bendruomenės nariais sujungtos tik vienu bendraautorystės ryšiu. Galiausiai, kaip ir prieš tai darytose bendruomenių išvadose, matoma, kad jos yra uždaros ir nesijungia su kitais tinklo nariais.

4.6. Žurnalų klasterizavimas

4.6.1. K-vidurkių modelio kūrimas

Išanalizavus literatūros šaltinius, kuriuose taikomi įvairūs klasterizavimo metodai bibliometrinėje analizėje, buvo nuspręsta vieną iš jų taikyti ir šiame darbe. Buvo pasirinkta žurnalus suskirstyti į klasterius pagal jų bibliografinius rodiklius. Sudarytas modelis leis įvertinti, kokius leidinius, pagal jų rodiklius, VILNIUS TECH darbuotojai renkasi dažniausiai, į kiek klasterių galima suskirstyti visus leidinius ir kaip šie klasteriai susiję tarpusavyje. Šiai užduočiai atlikti buvo nuspręsta naudoti *k*-vidurkių algoritmą. Šis algoritmas yra vienas populiariausių klasterizavimo metodų, tačiau analizuojant literatūros šaltinius, kuriuose atliekama bibliometrinė analizė, buvo sutinkamas retai.

Klasterizavimo metodas nuspręstas taikyti žurnalams, kurie buvo pateikti duomenų rinkinyje. Šis sprendimas priimtas, nes publikuojant mokslinį straipsnį, svarbu pasirinkti, kokiam leidinyje tai bus padaryta. Kaip jau buvo minėta žurnalų cituojamumo rodiklių skyriuje, leidiniai yra vertinami, jiems yra skaičiuojami prestižiškumo rodikliai ir jie pagal tai yra reitinguojami. Atlikus klasterizavimą, bus siekiama įvertinti, į kiek klasterių pasiskirsto nagrinėjami žurnalai, kokios yra šių klasterių savybės ir galiausiai, kuriuos iš jų dažniausiai renkasi VILNIUS TECH mokslo darbuotojai. Klasterizavimo modeliui sukurti buvo naudojami šie duomenų rinkinio rodikliai:

18 lentelė. Duomenų rinkinys klasterizavimo modeliui sudaryti

Duomenų rinkinio stulpelis	Duomenų aprašymas
Žurnalo indeksas	Indeksas identifikuoja konkretų žurnalą.
scopus_sources	Žurnalų, įtrauktų į „Scopus“ duomenų bazę, rodiklių vertinimo metai.
SNIP	<i>Source Normalized Impact per Paper (SNIP)</i> rodiklis yra realaus ir tikėtino (potencialaus) vidutinio citavimų skaičiaus, tenkančio vienai žurnalo publikacijai atstovaujamoje mokslo srityje, santykis.
CiteScore	<i>CiteScore</i> rodiklis yra visų tipų dokumentų citatų vidurkis.
SJR	<i>SCImago Journal Rank (SJR)</i> rodikliu yra apskaičiuojamas žurnalo prestižas pagal žurnalų tarpusavio citavimų tinklus.
quartile_in_caterory	Visi žurnalai, kurie turi SJR rodiklį, yra skirstomi pagal mokslo sritis į 4 kvartilius: Q1, Q2, Q3, Q4, kurie apibūdina žurnalo rangą.
if_info_years	Žurnalų, įtrauktų į „Web of Science“ duomenų bazę, rodiklių vertinimo metai.
IF	<i>Impact Factor (IF)</i> yra žurnalo cituojamumo rodiklis.
AIF	<i>Aggregate Impact Factor (AIF)</i> rodiklis skaičiuojamas, atsižvelgiant į visų kategorijos žurnalų citatų skaičių ir straipsnių iš visų kategorijos žurnalų skaičių.
quartile_in_caterory_2	Visi žurnalai, kurie turi IF rodiklį, yra skirstomi pagal mokslo sritis į 4 kvartilius: Q1, Q2, Q3, Q4, kurie apibūdina žurnalo rangą.
InCites_JCR	WoS rodikliai skaičiuojami tik žurnalams, įtrauktiems į <i>Journal Citation Reports (JCR)</i> : indeksuojamiems <i>Science Citation Index Expanded (SCIE)</i> ir <i>Social Sciences Citation Index (SSCI)</i>

Duomenų rinkinyje iš viso yra paminėtas 1 301 žurnalas ir 50,3 % iš jų (655 žurnalai) neturi priskirtų nagrinėjamų bibliografinių rodiklių. Tokių žurnalų įtraukti į klasterizacijos modeliui perduodamą duomenų rinkinį nėra prasmės, nes jie būtų priskirti vienam klasteriui, kuris neturėtų nieko bendro su likusiais. Dėl to šie įrašai pašalinami iš duomenų rinkinio. Likę 646 žurnalai taip pat turi ne visus išvardintus rodiklius arba jie priskirti leidiniams ne visais nagrinėjamais metais (duomenų rinkinyje „Scopus“ ir WoS rodikliai pateikti 2009 – 2018 metų periode), todėl trūkstamos reikšmės pakeičiamos į 0.

Kitas svarbus duomenų paruošimo žingsnis prieš taikant klasterizavimo modelį, yra duomenų normalizavimas. Tai yra labai svarbu, nes klasterizavimui bus naudojami rodikliai iš skirtingų šaltinių, o tai reiškia, kad jų matavimų skalės nėra vienodos. Taip pat, *Journal Citation Reports (JCR)* stulpelyje pateikiamos mokslo sritys, kurioms priskiriami žurnalai ir jos yra dvi: tikslųjų mokslų (SCIE) ir socialinių mokslų (SSCI). Dalis žurnalų neturi priskirtos mokslo srities, dėl to jiems priskiriama reikšmė „NA“. Skirtingas mokslo sritis atstovaujančių žurnalų palyginimas negali būti daromas tiesiogiai, nes tie patys rodikliai įgauna skirtingas reikšmes priklausomai nuo atitinkamos mokslo srities, todėl duomenų normalizavimas buvo pritaikytas kiekvienam mažesniui duomenų rinkiniui atskirai.

Normalizavimo užduočiai atlikti nuspręsta taikyti *min – max* metodą, kuris pagrįstas duomenų parametrų diapazono keitimu iš originalių verčių į $[0, 1]$ intervalą. Šiai užduočiai atlikti naudota *Python* biblioteka *sklearn.preprocessing*, kuri apima mastelio keitimo, centravimo, normalizavimo ir normalizavimo metodus. *MinMaxScaler* metodas, priklausantis minėtai bibliotekai, atlieka duomenų normalizavimą. Taikant šį metodą visų rodiklių reikšmės standartizuotos atsižvelgiant į mokslo sritį ir tokiu būdu sukurta galimybė juos palyginti tarpusavyje. Kaip ir minėta anksčiau, praleistos reikšmės buvo pakeistos į 0, tokia pati reikšmė joms liko ir po duomenų normalizavimo.

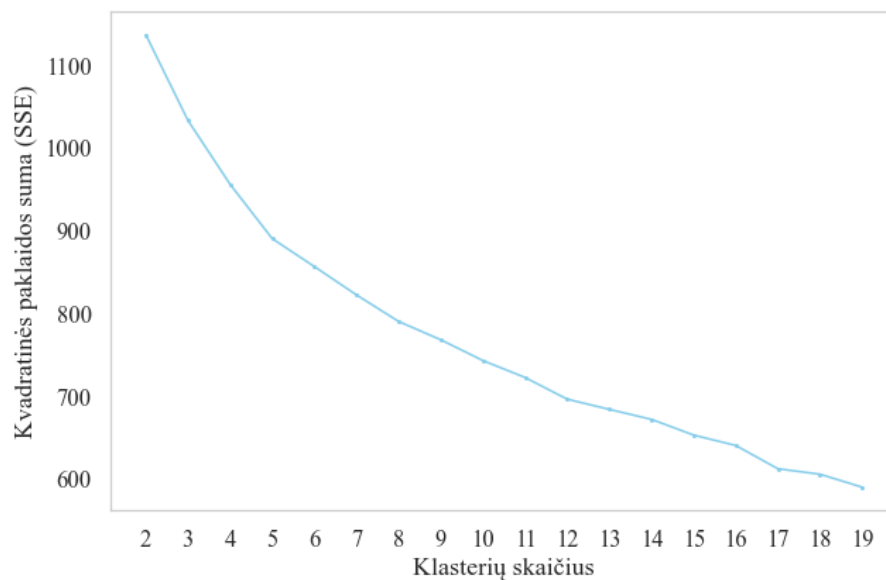
Paruošus duomenų rinkinį klasterizavimui, kitu žingsniu kuriamas *k*-vidurkių modelis, kuriam bus perduoti šie duomenys. *k*-vidurkių modeliui sukurti naudojama *Python* biblioteka *sklearn.cluster*, kurioje galima rasti populiariausius neprižiūrimo mokymosi klasterizacijos algoritmus. *k*-vidurkių modeliui sudaryti pasirenkamas *Kmeans* modelis ir jam perduodami standartizuoti duomenys. Pagrindinis modelio rezultatas yra klasteriai, kuriems priskirti visi nagrinėjami žurnalai. Gautas klasterių skaičius, kuris buvo parinktas modelio automatiškai, yra 8.

4.6.2. Optimalaus klasterių skaičiaus parinkimas

Gauto modelio rezultatai nebūtinai yra geriausi iš pirmojo karto ir svarbu įsitikinti ar klasterių skaičius parinktas teisingai. Šiai užduočiai atlikti buvo naudojami 2 metodai:

1. Taikant „alkūnės“ metodą;
2. Apskaičiuojant silueto koeficientą.

Taikant pirmąjį metodą, sukurtam k -vidurkių modeliui nurodoma, kokį klasterių skaičių taikyti, ir šis procesas pritaikomas klasterių skaičiui nuo 2 iki 20. Tokiu būdu gaunama kvadratinės paklaidos suma kiekvieno klasterių skaičiaus atveju. Gautos reikšmės vizualizuojamos grafike ir tikimasi, kad optimaliausias klasterių skaičius bus matomas kaip „alkūnės“ forma, kurios lūžio taškas atitiks susistabilizavęs klasterių skaičių. Nubrėžus šį grafiką turimam duomenų rinkiniui, „alkūnės“ forma nėra pakankamai aiški, dėl to optimalių klasterių skaičių rasti nėra paprasta. Rezultatai vaizduojami paveiksle žemiau:



22 Pav. Optimalaus klasterių skaičiaus radimas „alkūnės“ metodu

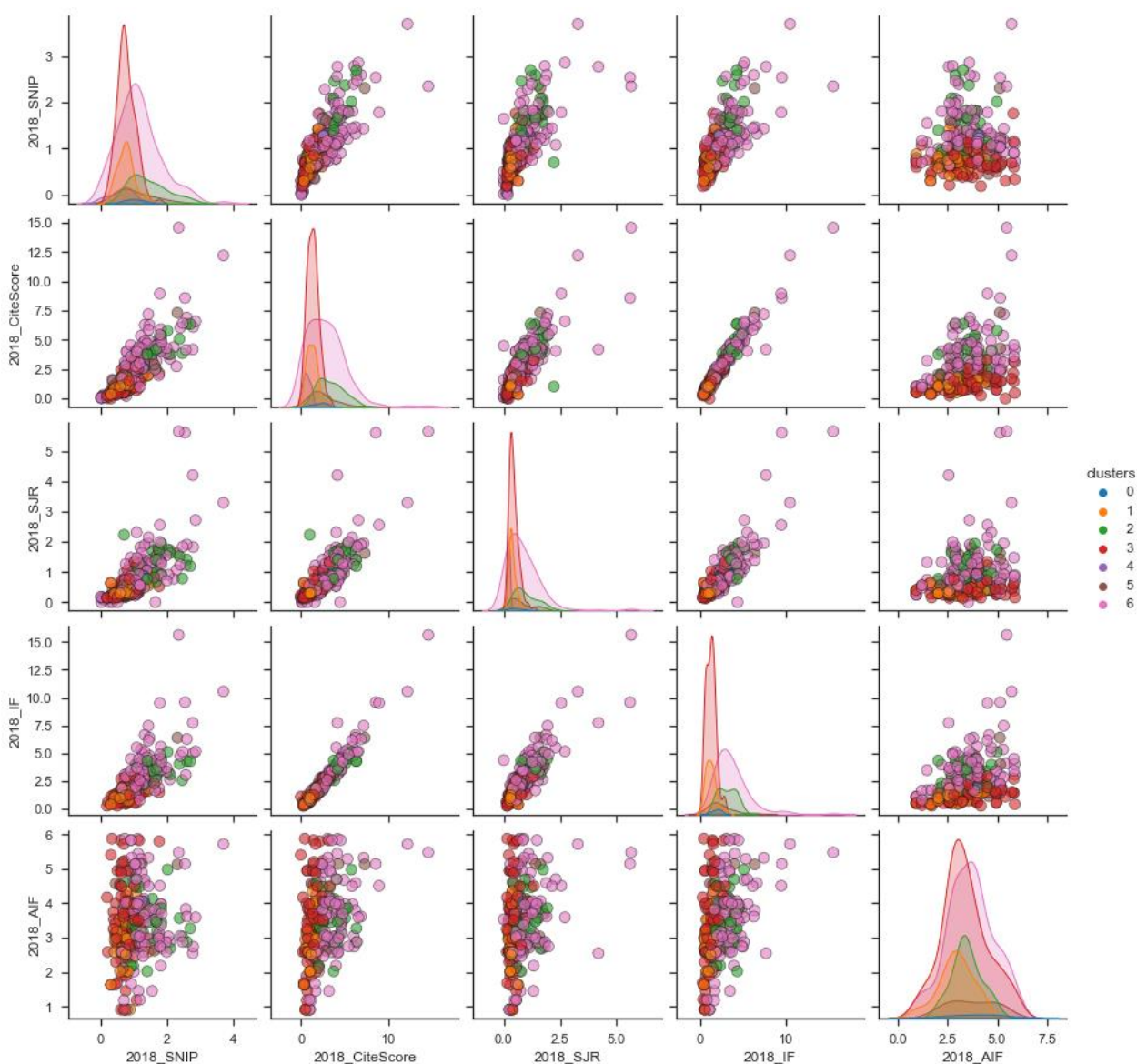
Taikant antrąjį metodą, buvo naudota *Yellowbrick Python* biblioteka. *Yellowbrick* biblioteka išplečia *Scikit-Learn* mašininio mokymosi aplikacijų programavimo sąsają (API), parenkant geriausią modelį ir jo hiperparametrus. Šioje bibliotekoje yra sukurtas silueto koeficiento apskaičiavimo modelis (*silhouette_score*), kuriam perduodamas paruoštas k -vidurkių modelis ir gaunamos silueto koeficiento reikšmės klasteriams nuo 2 iki 20. Iš gautų rezultatų matoma, kad didžiausia silueto koeficiento reikšmė gaunama, kuomet klasterių skaičius lygus 7. Rezultatai pateikiami lentelėje:

19 lentelė. Silueto koeficientas

Klasterių skaičius	Silueto koeficiento reikšmė	Klasterių skaičius	Silueto koeficiento reikšmė
2	0,386	12	0,14
3	0,124	13	0,124
4	0,128	14	0,155
5	0,144	15	0,168
6	0,108	16	0,169
7	0,152	17	0,152
8	0,124	18	0,174
9	0,136	19	0,18
10	0,141	20	0,174
11	0,152		

Koeficiento reikšmė yra didžiausia, kuomet klasterių skaičius yra 2. Atsižvelgiant į prieš tai taikytą metodą, daroma išvada, kad klasterių skaičius 2 nėra tinkamiausias (grafike lūžio taškas nesutampa su klasterių skaičiumi 2). Koeficiento reikšmė taip pat pradeda didėti, kuomet klasterių skaičius pasiekia 11 ir tai yra tikėtina, nes sukuriama daugiau klasterių centrų. Tačiau vertinant turimą duomenų rinkinį, būtų netikslinga rinktis labai didelį klasterių skaičių, nes tuomet jie prarastų savo išskirtinumą ir unikalumą. Todėl galutinis sprendimas, įvertinus abiejų metodų rezultatus, priimamas, kad optimalus klasterių skaičius žurnalų klasterizavimo modeliui yra 7. Nustačius optimalų klasterių skaičių, jis nurodomas k -vidurkių modelyje ir atnaujintas modelis pakartotinai pritaikomas duomenų rinkiniui. Modelio rezultatas – klasterių numeriai, kurie prijungiami prie pradinio duomenų rinkinio (pirmajam klasteriui suteikiamas skaitinis įvertis 0, nes *Python* programavimo kalboje skaičių skalė pradedama nuo šio skaičiaus).

Iš pradinio duomenų rinkinio pasirenkamos paskutiniųjų metų bibliografinių rodiklių reikšmės ir gautų klasterių įverčiai. Šie duomenys pavaizduojami sklaidos diagramoje, kuri yra patogus instrumentas ryšiams tarp dviejų skaitinių kintamųjų tirti:

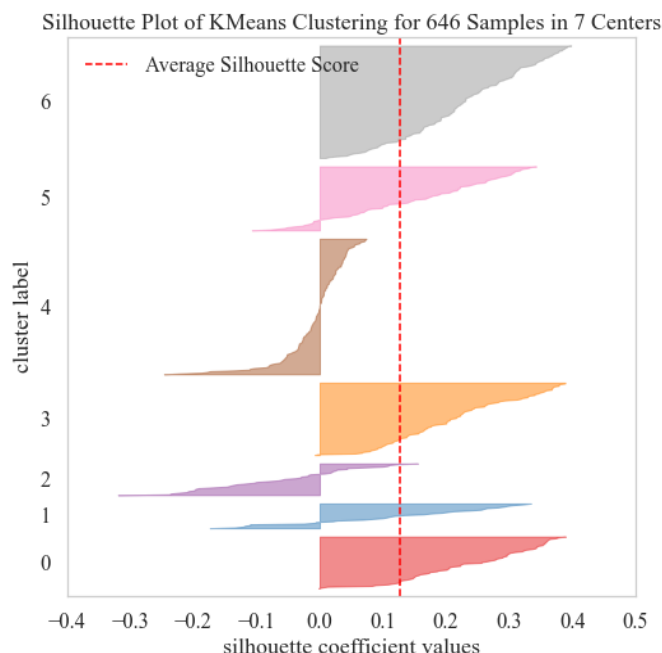


23 Pav. Sklaidos diagrama

Iš jos matoma, kad lyginant rodiklius tarpusavyje, nemažai klasterių yra persidengę ir susitelkę prie kairiojo apatinio kampo. Tai paaiškinama tuo, kad dalyje įrašų yra trūkstančių reikšmių. Sklaidos diagrama taip pat atskleidžia, kad keli rodikliai turi stiprų tiesinį ryšį: *CiteScore* ir IF bei *CiteScore* ir SJR. Taip pat matoma, kad rodiklis AIF neturi ryšio su kitais rodikliais, nes sklaidos diagramos yra įgavusios debesies formą arba gautos tiesės yra lygiagrečios y ašiai. Įstrižainėje esantys grafikai parodo kiekvieno rodiklio reikšmių pasiskirstimą pagal analizuojamus klasterius. Sklaidos diagrama taip pat atskleidžia, kad analizuojamame duomenų rinkinyje yra išskirčių – žurnalų su labai didelėmis bibliografinių rodiklių reikšmėmis. Šios išskirtys priklauso klasteriui 6.

4.6.3. Rezultatų analizė

Klasterių analizė pradedama nuo grafinio klasterių pasiskirstymo pavaizdavimo ir modelio analizės. Tam naudojama jau minėta *Yellowbrick Python* biblioteka ir joje esantis modelis *silhouette_visualizer*. Pritaikius klasterizacijos modelio rezultatus šiai funkcijai, gaunamas grafikas, kuriame vaizduojamas vidutinis silueto koeficientas (0,152) ir klasterių pasiskirstymas:



24 Pav. Žurnalų klasterių pasiskirstymas

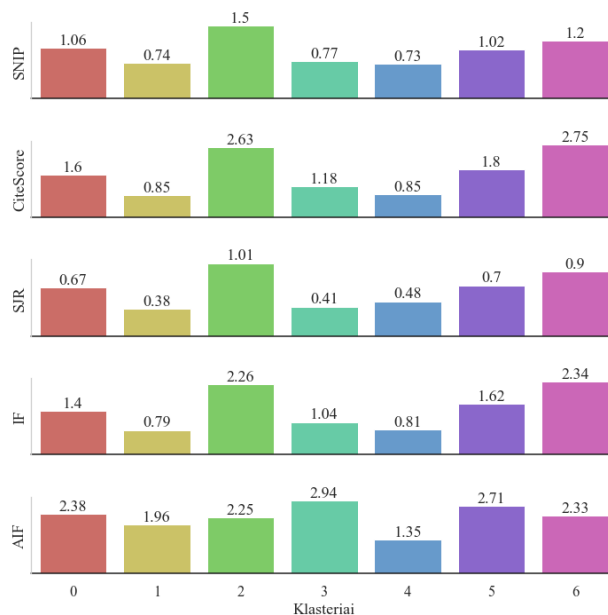
Iš šio grafiko matoma, kad yra 4 didesni klasteriai (0, 3, 5 ir 6), kurių reikšmės yra didesnės už vidutinį silueto koeficientą. Dėl to daroma išvada, kad reikšmės, pakliūnančios į šiuos klasterius, buvo priskirtos teisingai. Klasteris numeris 1 taip pat turi didžiąją dalį reikšmių, kurios yra didesnės už vidutinį silueto koeficientą, tačiau jame yra ir reikšmių, kurios įgyja neigiamą reikšmę. Tai galima interpretuoti kaip reikšmes, kurios galimai buvo priskirtos ne tam klasteriui, arba galėtų būti priskirtos keliems iš jų. Tokios išvados galioja ir likusiems 2 ir 4 klasteriams. Jei palygintume žurnalų klasterių grafiką (pav. 19), matytume, kad klasteris 4 persiengia su klasteriu 0, dėl to žurnalų klasterių pasiskirstymo grafike jis vaizduojamas reikšmėmis, kurios yra mažesnės už vidutinį silueto koeficientą ir neigiamos. Klasterizuojant realius duomenis, tokie rezultatai yra tikėtini, nes jie ne visada yra tolygiai bei lengvai atskiriamai pasiskirstę į natūralius klasterius.

Gauti klasterių numeriai prijungiami prie pradinio duomenų rinkinio ir apskaičiuojamos vidutinės bibliografinių rodiklių reikšmės kiekvienam klasteriui:

20 lentelė. Vidutinės rodiklių reikšmės klasteriuose

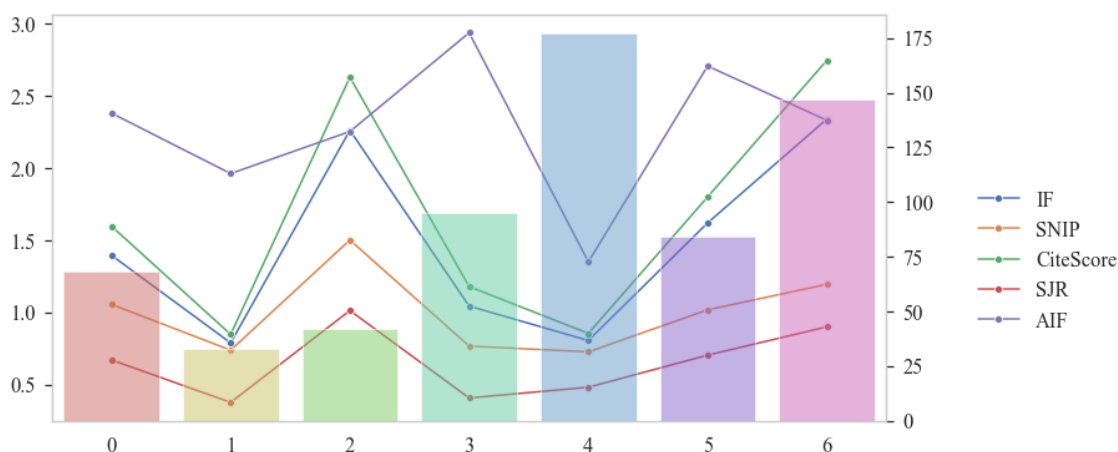
Klasteriai	SNIP	CiteScore	SJR	IF	AIF	Žurnalų skaičius
0	1,06	1,6	0,67	1,4	2,38	68
1	0,74	0,85	0,38	0,79	1,96	33
2	1,5	2,63	1,01	2,26	2,25	42
3	0,77	1,18	0,41	1,04	2,94	95
4	0,73	0,85	0,48	0,81	1,35	177
5	1,02	1,8	0,7	1,62	2,71	84
6	1,2	2,75	0,9	2,34	2,33	147

Lentelėje žalia spalva paryškintos didžiausios vidutinės rodiklių reikšmės, o raudona – mažiausios. Iš jų galima daryti išvadą, kad klasteris numeris 1 išsiskiria, kaip turintis daugiausiai rodiklių, kurių vidutinė reikšmė yra mažiausia. Atitinkamai klasterių 2 ir 6 vidutinės rodiklių reikšmės yra didžiausios. Šiai informacijai įvertinti lengviau, sukuriamas vidutinių rodiklių reikšmių grafikas pagal klasterių numerius:



25 Pav. Vidutinės rodiklių reikšmės klasteriuose

Galutinėms išvadoms priimti, sukuriamas dar vienas grafikas, kuriame vaizduojamos ne tik vidutinės rodiklių reikšmės (linijos), bet ir klasterių dydžiai (stulpeliai):



26 Pav. Vidutinės rodiklių reikšmės klasteriuose ir klasterių dydžiai

Remianis grafikais bei duomenų lentele, galima daryti šias išvadas apie žurnalų klasterius:

1. Į klasterį numeris 1 pakliūna žurnalai, kurių vidutinės rodiklių reikšmės yra mažiausios. Kadangi IF ir SJR vidutinės rodiklių reikšmės yra mažiausios iš visų klasterių, galima teigti, kad į jį patenka mažiausiai prestižiniai žurnalai.
2. Klasteris numeris 2 pasižymi didžiausia SJR rodiklio reikšme. Visų kitų rodiklių vidutinės reikšmės taip pat yra vienos didžiausių, dėl to daroma išvada, kad į šį klasterį pakliūna prestižiškiausi žurnalai.
3. Klasteris numeris 3 išsiskiria vidutine rodiklio AIF reikšme, kuri yra didžiausia iš visų klasterių. Tai reiškia, kad žurnalai, patenkantys į šį klasterį, išsiskiria citavimų skaičiumi, tačiau negali būti priskiriami prie prestižinių leidinių, nes kitų rodiklių vidutinės reikšmės nėra didelės.
4. Daugiausia žurnalų iš tiriamo duomenų rinkinio, priskirta klasteriui numeris 4 (177), tačiau vidutinės žurnalų rodiklių reikšmės yra vienos mažiausių. Šiame klasteryje yra žurnalai, kurių vidutinės SNIP, CiteScore ir AIF rodiklių reikšmės yra mažiausios, o tai reiškia, kad į jį patenka mažiausiai cituojami žurnalai.
5. Klasteris 6 pagal žurnalų skaičių yra antras pagal dydį (147 žurnalai). Jam priskirtų žurnalų WoS rodikliai IF ir AIF turi vienas didžiausių vidutinių reikšmių. Galima teigti, kad šiame klasteryje yra labiausiai cituojami žurnalai pagal WoS duomenų bazės rodiklius.
6. Klasteriai 0 ir 5 yra pakankamai panašūs: visų rodiklių vidutinės reikšmės yra gana panašios ir neišsiskiria iš kitų klasterių. Galima daryti išvadą, kad žurnalai, esantys šiuose klasteriuose, yra vidutinės svarbos.

Atlikus žurnalų klasterių analizę, atrenkami 10 žurnalų, kurie duomenų rinkinyje pasikartojo daugiausiai. Jie, kartu su paskutinių metų (2018 m.) rodiklių vertėmis ir klasterių numeriais pateikiam lentelėje žemiau:

21 lentelė. 10 dažniausiai pasikartojusių žurnalų

Žurnalo pavadinimas	Skaičius duomenų rinkinyje	SNIP	CiteScore	SJR	IF	AIF	Mokslo sritis	Klasteris
Elektronika ir elektrotechnika	170	0,49	0,81	0,18	0,68	3,2	SCIE	1
Journal of civil engineering and management	162	0,9	2,05	0,59	N/A	N/A	N/A	2
The Baltic journal of road and bridge engineering	126	0,69	0,85	0,3	0,77	2,7	SCIE	1
Transport	121	1,22	1,78	0,48	N/A	N/A	N/A	1
Verslas: teorija ir praktika	120	0,93	1,21	0,26	N/A	N/A	N/A	4
Sustainability	104	1,17	3,01	0,55	N/A	N/A	N/A	2
Technological and economic development of economy	104	1,4	3,76	0,78	N/A	N/A	N/A	2
Filosofija. Sociologija	101	0,3	0,36	0,6	0,4	1,7	SSCI	1
Logos	98	0,03	0,03	0,15	N/A	N/A	N/A	4
Journal of business economics and management	92	0,6	1,75	0,4	1,86	2,59	SSCI	1

Tyrimo rezultatai ir išvados

1. Literatūra, kuri yra susijusi su bibliografinių duomenų moksliniais tyrimais, buvo nagrinėjama visą studijų laiką. Literatūros analizė atskleidė pagrindines bibliometrinės analizės kryptis, taikomus metodus, autorių siekiamus tikslus; taip pat padėjo įvertinti, kokie metodai galėtų praplėsti bibliometrinės analizės sritį ir tokiu būdu ją patobulinti.
2. Susijusių mokslo darbų analizė atskleidė, kad grafų teorija yra labai svarbi bibliometrinėje analizėje. Ji taikoma tinklų analizėje, egzistuojančių ryšių tyrime ir bendruomenių nustatyme. Klasterinė analizė taip pat buvo išskirta kaip viena iš pagrindinių bibliometrinės analizės krypčių.
3. Nustatyta, kad moksliniuose tyrimuose, susijusiuose su bibliografinių duomenų analize, tekstinė duomenų analizė taikoma retai, nėra akcentuojamas bibliografinių ir tekstinių duomenų ryšys.
4. Atlikus literatūros apžvalgą bei taikomų metodų analizę, pasiūlyta bibliometrinės analizės metodika, kurios pagalba galima atskleisti daugiau bibliografinių duomenų savybių ir išgauti daugiau žinių apie duomenų rinkinį.
5. Atliktas empirinis VILNIUS TECH bibliotekoje registruotų mokslinių publikacijų bibliografinių duomenų tyrimas, naudojant pasiūlytą metodologiją. Jo rezultatai:
 - 5.1. Bibliografinių duomenų tvarkymas buvo daugiausiai laiko pareikalavusi darbo dalis. Jai atlikti naudota *Python* programa. Iš pradinių 8 duomenų rinkinio stulpelių gauti 37 nauji stulpeliai; žurnalų pavadinimų grupavimas pagerino leidinių pavadinimų kokybę 27 %.
 - 5.2. Bibliometrinė analizė leido įvertintini VILNIUS TECH universiteto mokslinių publikacijų skaičiaus kitimą laike, nustatyti pagrindines mokslo sritis, išskirti autorių bendradarbiavimo tendencijas ir įtakingiausius autorius.
 - 5.3. Sudaryti 2 autorių bendradarbiavimo tinklai, nagrinėta išskirčių įtaka. Įvertintas tinklo tankumas, viršūnių laipsnių savybės, nustatyti svarbiausi ir įtakingiausi tinklo nariai.
 - 5.4. Išskirtos tinkluose esančios bendruomenės ir ištirta jų sandara ir sąlygos, kurioms esant jos susiformuoja.
 - 5.5. Sukurtas klasterizavimo modelis, naudojant didelį kiekį rodiklių, ir žurnalai suskirstyti į 7 klasterius. Klasterinės analizės rezultatas – nustatytos mokslo darbuotojų žurnalų pasirinkimo tendencijos, įvardinti populiariausi žurnalai, analizuoti jų bibliografiniai rodikliai.

Literatūros ir kitų šaltinių sąrašas

- Algoritmai keičiantys pasaulį. (2014). Technomatematika VGTU FMF MMK. Prieiga per internetą: http://www.techmat.vgtu.lt/projektas/views/dalykai/Algoritmai_keicianrys_pasauli/Paskaita10new0415.pdf
- Balčiūnas, A. (2019). Aprašomoji statistika. 3 paskaita. Prieiga per internetą: <http://web.vu.lt/mif/a.balciunas/files/2019/12/Taikomoji-statistika3.pdf>
- Bazm, S., Kalantar, S. M. ir Mirzaei, M. (2016). Bibliometric mapping and clustering analysis of Iranian papers on reproductive medicine in Scopus database (2010-2014). *Int J Reprod Biomed*, 14(6): 371–382. Prieiga per internetą: <https://pubmed.ncbi.nlm.nih.gov/27525320/>
- Beineke, L. W., Oellermann, O. R. ir Pippert, R.E. (2002). The average connectivity of a graph. *Discrete Mathematics*. 252: 31–45. doi: [https://doi.org/10.1016/S0012-365X\(01\)00180-7](https://doi.org/10.1016/S0012-365X(01)00180-7)
- Bitinas, B. (2013). Rinktiniai edukologiniai raštai: Edukologinis tyrimas: sistema ir procesas. Ugdymo tyrimų idėjos ir problemos. II tomas. Vilnius: Lietuvos edukologijos universiteto leidykla.
- Borkin, D., Némethová, A., Michal'conok, G., ir Maiorov, K. (2019). Impact of data normalization on classification model accuracy. *Research Papers Faculty of Materials Science and Technology Slovak University of Technology*, 27(45), 79-84. doi: 10.2478/rput-2019-0029
- Bosker, H. R. (2021). Using fuzzy string matching for automated assessment of listener transcripts in speech intelligibility studies. *Behavior Research Methods*, 53(5), 1945-1953. doi: <https://doi.org/10.3758/s13428-021-01542-4>
- Brownlee, J. (2020). *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*.
- Brus, P. (2021). Clustering: How to Find Hyperparameters using Inertia. Prieiga per internetą: <https://towardsdatascience.com/clustering-how-to-find-hyperparameters-using-inertia-b0343c6fe819>
- Chien, T., Wang, H., Kan, W. ir Su, S. (2019). Whether article types of a scholarly journal are different in cited metrics using cluster analysis of MeSH terms to display. *Medicine*, 98. doi: 10.1097/MD.00000000000017631
- Dorta González, M.I. ir Dorta González, P. (2014). Aggregate impact factor according to scientific fields. *Investigación bibliotecológica*, 28(62). Prieiga per internetą:

http://www.scielo.org.mx/scielo.php?pid=S0187358X2014000100002&script=sci_arttext&tlng=en

Dutta, M. (2021). Fuzzy String Matching – A Hands-on Guide. Prieiga per internetą: <https://www.analyticsvidhya.com/blog/2021/07/fuzzy-string-matching-a-hands-on-guide/>

Erman, J., Arlitt, M. ir Mahanti, A. (2006). Traffic classification using clustering algorithms. In Proceedings of the 2006 SIGCOMM workshop on Mining network data (MineNet '06). Association for Computing Machinery, 281–286. doi: <https://doi.org/10.1145/1162678.1162679>

Eyun-Jung, K., Pasadeos, Y. ir Ertem-Eray, T. (2019). Growth of public relations research networks: a bibliometric analysis. *Journal of Public Relations Research*, 31:1-2, 5-31. doi: 10.1080/1062726X.2019.1577739

Furstenau, L., Rabaioli, B., Kremer Sott, M., Cossul, D., Sott Bender, M., Júdice De Mattos Farina, E. M., Barcellos Filho, F. N., Severo, P. P., Dohan, M. ir Bragazzi, N. L. (2021). A Bibliometric Network Analysis of Coronavirus during the First Eight Months of COVID-19 in 2020. *International Journal of Environmental Research and Public Health*, 18, 952. doi: <https://doi.org/10.3390/ijerph18030952>

Goldenberg, D. (2021). Social Network Analysis: From Graph Theory to Applications with Python. Prieiga per internetą: <https://towardsdatascience.com/social-network-analysis-from-theory-to-applications-with-python-d12e9a34c2c7>

Guthrie, S., Cochrane, G., Deshpande, A., Macaluso, B. ir Larivière, V. (2019). Understanding the contribution of UK public health research to clinical guidelines: a bibliometric analysis. doi: <https://doi.org/10.12688/f1000research.18757.1>

Huynh, T. (2020). FuzzyWuzzy: Find Similar Strings within one column in Python. Prieiga per internetą: <https://towardsdatascience.com/fuzzywuzzy-find-similar-strings-within-one-column-in-a-pandas-data-frame-99f6c2a0c212>

Jacomy, M., Venturini, T., Heymann, S. ir Bastian, M. (2014). ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLOS ONE*. doi: <https://doi.org/10.1371/journal.pone.0098679>

Jasiukevičius, R. (2016). Ad-hoc bendruomenių kontaktų modelio sudarymas ir tyrimas (Baigiamasis magistro projektas). Prieiga per internetą: <https://epubl.ktu.edu/object/elaba:15980975/>

- Kalantari, A., Kamsin, A., Kamaruddin, H.S., Ebrahim, N.A., Gani, A., Ebrahimi, A., ir Shamshirband, S. (2017). A bibliometric approach to tracking big data research trends. *J Big Data*, 4, 30. doi: <https://doi.org/10.1186/s40537-017-0088-1>
- Kammerer, K., Göster, M., Reichert, M. ir Pryss, R. (2021). Ambalytics: A Scalable and Distributed System Architecture Concept for Bibliometric Network Analyses. *Future Internet* 2021, 13(8), 203. doi: <https://doi.org/10.3390/fi13080203>
- Kang, S. S. (2015). Word similarity calculation by using the edit distance metrics with consonant normalization. *Journal Of Information Processing Systems*, 11(4), 573-582. doi: 10.3745/JIPS.04.0018.
- Khosravi, M., Menon, V. (2019). CiteScore-Based Quartiles for Scientometric Analysis. doi: 10.36227/techrxiv.11385939
- Khurana, P. ir Sharma, K. (2021). Impact of h-index on authors ranking: A comparative analysis of Scopus and WoS. 18th International Conference on Scientometrics and Informetrics. doi: <https://doi.org/10.48550/arXiv.2102.06964>
- Krylovas, A. (2004). Diskrečioji matematika. Vilnius: Technika.
- Larivière, V. ir Sugimoto, C.R. (2019). The Journal Impact Factor: A Brief History, Critique, and Discussion of Adverse Effects. Glänzel, W., Moed, H.F., Schmoch, U., Thelwall, M. (eds) *Springer Handbook of Science and Technology Indicators*. Springer Handbooks. Springer, Cham, 3–24. doi: https://doi.org/10.1007/978-3-030-02511-3_1
- Liao, H., Tang, M., Luo, L., Li, C., Chiclana, F. ir Zeng, X.-J. (2018). A Bibliometric Analysis and Visualization of Medical Big Data Research. *Sustainability*, 10, 166. doi: <https://doi.org/10.3390/su10010166>
- Loo, M.P.J. (2014). The stringdist Package for Approximate String Matching. *The R Journal* 6(1), 111-122. Prieiga per internetą: <https://journal.r-project.org/archive/2014-1/loo.pdf>
- Makhija, V., ir Ahuja, S. (2018). Rule based Text Extraction from a Bibliographic Database. *DESIDOC Journal of Library & Information Technology*, 38(1), 5-10. doi: 10.14429/djlit.38.1.12307
- Mazeliauskas, A. (2016). Svarbos matai tinklų analizei. (Baigiamasis bakalauro darbas). Prieiga per internetą: <https://epublications.vu.lt/object/elaba:20203614/MAIN>
- Merediz-Sola, I. ir Bariviera, A. F. (2019). A bibliometric analysis of Bitcoin scientific production. *Research in International Business and Finance*, 50, 294-305. doi: <https://doi.org/10.1016/j.ribaf.2019.06.008>

- Ming-Huang, W., Yuh-Shan, H. ir Hui-Zhen, F. (2019). Global performance and development on sustainable city based on natural science and social science research: A bibliometric analysis. *Science of The Total Environment*, 666, 1245-1254. doi: <https://doi.org/10.1016/j.scitotenv.2019.02.139>
- Mishra, D., Gunasekaran, A., Papadopoulos, T. ir Childe, S.J. (2018). Big Data and supply chain management: a review and bibliometric analysis. *Ann Oper Res*, 270, 313–336. doi: <https://doi.org/10.1007/s10479-016-2236-y>
- Mishra, D., Gunasekaran, A., Papadopoulos, T. ir Hazen, B. (2017). Green supply chain performance measures: A review and bibliometric analysis. *Sustainable Production and Consumption*, 10, 85-99. doi: <https://doi.org/10.1016/j.spc.2017.01.003>
- Moosavi, J., Naenil, L., Fathollahi-Fard, A. ir Fiore, U. (2021). Blockchain in supply chain management: a review, bibliometric, and network analysis. *Environ Sci Pollut Res*. doi: <https://doi.org/10.1007/s11356-021-13094-3>
- Moschini, U., Fenialdi, E., Daraio, C. Ruocco, G. ir Molinari, E. (2020). A comparison of three multidisciplinary indices based on the diversity of Scopus subject areas of authors' documents, their bibliography and their citing papers. *Scientometrics*, 125, 1145–1158. doi: <https://doi.org/10.1007/s11192-020-03481-x>
- Noack, A. (2018). Modularity clustering is force-directed layout. *Phys Rev E Stat Nonlin Soft Matter Phys*, 79. doi: 10.1103/PhysRevE.79.026102
- Paulauskienė, K. ir Kurasova, O. (2014). Dimensijų mažinimo metodais gautų projekcijų įvertinimas. *Lietuvos matematikos rinkinys*, 55(B), 137-142. doi: 10.15388/LMR.B.2014.26.
- Petrauskienė, Ž.; Aleksandrovė, R. (2015). Bibliometriniai tyrimai Vilniaus universiteto bibliotekos veikloje. *Vilniaus universiteto bibliotekos metraštis*, 93-107. doi: 10.15388/VUBmetrastis.2015.1/VUBm
- Pons, P. ir Latapy, M. (2005). Computing Communities in Large Networks Using Random Walks. *Journal of Graph Algorithms and Applications*, 10(2), 191–218. Prieiga per internetą: <https://jgaa.info/accepted/2006/PonsLatapy2006.10.2.pdf>
- Pranckutė, R. (2021). Žurnalų kokybės rodikliai. Prieiga per internetą: https://vilniustech.lt/files/4148/207/10/0_0/urnal%C5%B3_kokyb%C4%97s_rodikliai.pdf
- Pritchard, A. (1969). Statistical Bibliography or Bibliometrics? *Journal of Documentation*, 25(4), 348–349.

- Putria, I. G. A. W. ir Dwidasmarab, I. B. G. (2021). Application of the K-Means Algorithm to Segmentation of Consumer Interest in Silver Craft" Kreasi Slaka Bali. *Jurnal Elektronik Ilmu Komputer Udayana*, 9(4), 541-548. doi: <https://doi.org/10.24843/JLK.2021.v09.i04.p12>
- Shamszadeh, S., Asgary, S. ir Nosrat, A. (2019). Regenerative Endodontics: A Scientometric and Bibliometric Analysis. *Regenerative Endodontics*, 45, 272-280. doi: <https://doi.org/10.1016/j.joen.2018.11.010>
- Skrebutėnaitė, V. (2021). COVID-19 Europoje: klasterinė duomenų analizė. (Baigiamasis bakalauro darbas). Prieiga per internetą: <https://talpykla.elaba.lt/elaba-fedora/objects/elaba:107117141/datastreams/MAIN/content>
- Stabingienė, L. (2014). Ekonometrika. Prieiga per internetą: <http://www.ilab.lt/stabingiene/index.html>
- Stögera, K., Schneeberger, D., Kieseberg, P. ir Holzinger, A. (2021). Legal aspects of data cleansing in medical AI. *Computer Law & Security Review*, 42. doi: 10.1016/j.clsr.2021.105587
- Sugimoto, C. R., Ahn, Y., Smith, E., Macaluso, B. ir Larivi`ere, V. (2019). Factors affecting sex-related reporting in medical research: a cross-disciplinary bibliometric analysis. *The Lancet*, 393, 550-559. doi: [https://doi.org/10.1016/S0140-6736\(18\)32995-7](https://doi.org/10.1016/S0140-6736(18)32995-7)
- Valdez, A. C., Dehmer, M. ir Holzinger, A. (2016). Application of Graph Entropy for Knowledge Discovery and Data Mining in Bibliometric Data, In *Mathematical Foundations and Applications of Graph Entropy* (p. 259 – 273). doi: [10.1002/9783527693245.ch9](https://doi.org/10.1002/9783527693245.ch9)
- Van Eck, N.J., ir Waltman, L. (2014). Visualizing bibliometric networks. In Y. Ding, R. Rousseau, & D. Wolfram (Eds.), *Measuring scholarly impact: Methods and practice*, 285–320. doi: http://dx.doi.org/10.1007/978-3-319-10377-8_13
- Van Steen, M. (2010). Graph theory and complex networks. *An introduction*, 144. Prieiga per internetą: <http://pages.di.unipi.it/ricci/book-watermarked.pdf>
- Veloutsou, C. ir Mafe, C.R. (2020). Brands as relationship builders in the virtual world: A bibliometric analysis. *Electronic Commerce Research and Applications*, 39. doi: <https://doi.org/10.1016/j.elerap.2019.100901>
- Verma, M., Srivastava, M., Chack, N., Diswar, A. K. ir Gupta, N. (2012). A Comparative Study of Various Clustering Algorithms in Data Mining. *International Journal of Engineering Research and Applications*, 2(3): 1379–1384. Prieiga per internetą:

<https://www.ijcsmc.com/docs/papers/November2014/V3I11201499a4.pdf>

VILNIUS TECH biblioteka. (n.d.). Žurnalo pasirinkimas publikavimui. Prieiga per internetą: <https://vilniustech.lt/biblioteka/mokslinė-komunikacija/zurnalo-pasirinkimas-publikavimui/287453#330224>

Visuotinė lietuvių enciklopedija. Prieiga per internetą: <https://www.vle.lt/>

Vilutienė, T., Kalibatiene, D., Hosseini, M. R., Pellicer, E. ir Zavadskas, E. K. (2019). Building Information Modeling (BIM) for Structural Engineering: A Bibliometric Analysis of the Literature. Advances in Civil Engineering, 2019. doi: <https://doi.org/10.1155/2019/5290690>

Waltman, L., Van Eck, N. J. ir Noyons, E. C. M. (2010). A unified approach to mapping and clustering of bibliometric networks. Journal of Informetrics, 4(4), 629-635. doi: <https://doi.org/10.1016/j.joi.2010.07.002>

Priedai

1. Python kodas, duomenų rinkinio valymui ir tekstinei duomenų analizei atlikti bei Gephi duomenų rinkiniams sukurti

```
## Naudojamos bibliotekos

import pandas as pd
import numpy as np
import re
from functools import reduce
import itertools
from fuzzywuzzy import process, fuzz

## Nuskaitomi duomenys

data = pd.read_csv('data.csv')

## Sukuriamos autorių pavardžių ir inicialų kombinacijos ir išsaugomos Excel failuose

#### Gautos autorių pavardės išsaugomos Excel failuose, kad būtų galima jas peržiūrėti

##### autoriai1: Pavarde, V.
##### autoriai2: Pavarde Pavarde, V. - pasalinti su zodeliu ir
##### autoriai3: Pavarde-Pavarde, V.
##### autoriai4: Pavarde Pavarde Pavarde, V. - pasalinti dauguma reiksmiu
##### autoriai5: Pavarde, V.V.
##### autoriai6: Pavarde, V.V.V.
##### autoriai7: Pavarde, V.V.V.V.
##### autoriai8: Pavarde, V.V.V.V.V.

# autoriai1: Pavarde, V.
autoriai_parser = re.compile("([A-Ž][a-ž]+, [A-Ž].)")
autoriai1 = pd.DataFrame(data.projection_apa.map(lambda x:
re.findall(autoriai_parser, x)))
#autoriai1.to_excel('autoriai1.xlsx')

# autoriai2: Pavarde Pavarde, V.
autoriai_parser2 = re.compile("([A-Ž][a-ž]+ [A-Ž][a-ž]+, [A-Ž].)")
autoriai2 = pd.DataFrame(data.projection_apa.map(lambda x:
re.findall(autoriai_parser2, x)))
#autoriai2.to_excel('autoriai2.xlsx')

# autoriai3: Pavarde-Pavarde, V.
autoriai_parser3 = re.compile("([A-Ž][a-ž]+[-][A-Ž][a-ž]+, [A-Ž].)")
autoriai3 = pd.DataFrame(data.projection_apa.map(lambda x:
re.findall(autoriai_parser3, x)))
#autoriai3.to_excel('autoriai3.xlsx')

# autoriai4: Pavarde Pavarde Pavarde, V.
autoriai_parser4 = re.compile("([A-Ž][a-ž]+ [A-Ž][a-ž]+ [A-Ž][a-ž]+, [A-Ž].)")
autoriai4 = pd.DataFrame(data.projection_apa.map(lambda x:
re.findall(autoriai_parser4, x)))
```

```
#autoriai4.to_excel('autoriai4.xlsx')

# autoriai5: Pavarde, V.V.
autoriai_parser5 = re.compile("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.])")
autoriai5 = pd.DataFrame(data.projection_apa.map(lambda x:
                                                re.findall(autoriai_parser5, x)))
#autoriai5.to_excel('autoriai5.xlsx')

# autoriai6: Pavarde, V.V.V.
autoriai_parser6 = re.compile("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.])")
autoriai6 = pd.DataFrame(data.projection_apa.map(lambda x:
                                                re.findall(autoriai_parser6, x)))
#autoriai6.to_excel('autoriai6.xlsx')

# autoriai7: Pavarde, V.V.V.V.
autoriai_parser7 = re.compile("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.])")
autoriai7 = pd.DataFrame(data.projection_apa.map(lambda x:
                                                re.findall(autoriai_parser7, x)))
#autoriai7.to_excel('autoriai7.xlsx')

# autoriai8: Pavarde, V.V.V.V.V.
autoriai_parser8 = re.compile("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.])")
autoriai8 = pd.DataFrame(data.projection_apa.map(lambda x:
                                                re.findall(autoriai_parser8, x)))
# autoriai8.to_excel('autoriai8.xlsx')

## Pašalinami publikacijų metai iš publikacijos aprašymo (metai yra atskiras
stulpelis)

data1 = data

data1['new']=""

for index, row in enumerate(data1.projection_apa):
    autoriai_parsed = re.sub("([0-9][0-9][0-9][0-9])", "", row)
    data1.new[index] = autoriai_parsed

## doi adreso pašalinimas iš publikacijos aprašymo

doi = data1['new'].str.split('doi:', expand=True)

data1['doi_address'] = doi[1]

data1['doi'] = pd.notnull(data1['doi_address']).astype(int)

data1 = data1.drop(['doi_address'], axis=1)

data1['new_1'] = doi[0]

## "Retrieved" adreso pašalinimas iš publikacijos aprašymo

Retrieved = data1['new_1'].str.split('Retrieved ', expand=True)

data1['new_2'] = Retrieved[0]

## "Žiūrėta" adreso pašalinimas iš publikacijos aprašymo
```

```

ziureta = data1['new_2'].str.split(' Žiūrėta ', expand=True)

data1['new_3'] = ziureta[0]

data1 = data1.drop(['new', 'new_1', 'new_2'], axis=1)

## Išskiriami publikacijų puslapiai ir pašalinami iš publikacijos aprašymo

#### Pirmiausia randamos ir pašalinamos skaičių kombinacijos, kurios yra
nerealistinės (klaidingi įrašai arba išskirtys). Jie nebus naudojami puslapių
skaičiui aprašyti.

data1['new_4']="

for index, row in enumerate(data1.new_3):
    autoriai_parsed = re.sub("([0-9]+[-]+[0-9][0-9][0-9][0-9][0-9][0-9])", "", row)
    data1.new_4[index] = autoriai_parsed

#### Tuomet randami ir pašalinami publikacijos puslapiai, kurie yra ypač didelių
verčių. Dalis jų yra nerealistiniai, dėl to tokios vertės bus šalinamos

puslapiai_parser2 = re.compile("([0-9]+[-]+[0-9][0-9][0-9][0-9][0-9])")
puslapiai2 = pd.DataFrame(data1.new_4.map(lambda x: re.findall(puslapiai_parser2,
x)))

puslapiai2.rename(columns = {'new_4' : 'Puslapiai_5_zenkilai'}, inplace = True)

data1['Puslapiai_5_zenkilai'] = puslapiai2['Puslapiai_5_zenkilai']

data1['new_5']="

for index, row in enumerate(data1.new_4):
    autoriai_parsed = re.sub("([0-9]+[-]+[0-9][0-9][0-9][0-9][0-9])", "", row)
    data1.new_5[index] = autoriai_parsed

data1 = data1.drop(['new_3', 'new_4'], axis=1)

#### Atskiriami ir pašalinami publikacijos puslapiai, kurie yra keturženkliai

puslapiai_parser3 = re.compile("([0-9]+[-]+[0-9][0-9][0-9][0-9])")
puslapiai3 = pd.DataFrame(data1.new_5.map(lambda x: re.findall(puslapiai_parser3,
x)))

puslapiai3.rename(columns = {'new_5' : 'Puslapiai_4_zenkilai'}, inplace = True)

data1['Puslapiai_4_zenkilai'] = puslapiai3['Puslapiai_4_zenkilai']

data1['new_6']="

for index, row in enumerate(data1.new_5):
    autoriai_parsed = re.sub("([0-9]+[-]+[0-9][0-9][0-9][0-9])", "", row)
    data1.new_6[index] = autoriai_parsed

#### Atskiriami ir pašalinami publikacijos puslapiai, kurie yra triženkliai

puslapiai_parser4 = re.compile("([0-9]+[-]+[0-9][0-9][0-9])")
puslapiai4 = pd.DataFrame(data1.new_6.map(lambda x:

```

```

re.findall(puslapiai_parser4, x)))

puslapiai4.rename(columns = {'new_6' : 'Puslapiai_3_zenklai'}, inplace = True)

data1['Puslapiai_3_zenklai'] = puslapiai4['Puslapiai_3_zenklai']

data1['new_7']="

for index, row in enumerate(data1.new_6):
    autoriai_parsed = re.sub("([0-9]+[-]+[0-9][0-9][0-9])", "", row)
    data1.new_7[index] = autoriai_parsed

data1 = data1.drop(['new_5', 'new_6'], axis=1)

#### Atskiriami ir pašalinami publikacijos puslapiai, kurie yra dviženkliai

puslapiai_parser5 = re.compile("([0-9]+[-]+[0-9][0-9])")
puslapiai5 = pd.DataFrame(data1.new_7.map(lambda x: re.findall(puslapiai_parser5,
x)))

puslapiai5.rename(columns = {'new_7' : 'Puslapiai_2_zenklai'}, inplace = True)

data1['Puslapiai_2_zenklai'] = puslapiai5['Puslapiai_2_zenklai']

data1['new_8']="

for index, row in enumerate(data1.new_7):
    autoriai_parsed = re.sub("([0-9]+[-]+[0-9][0-9])", "", row)
    data1.new_8[index] = autoriai_parsed

data1 = data1.drop(['new_7'], axis=1)

## Autorių ištrynimasis. Autorius padedama trinti nuo didžiausias išskirtis turinčių
kombinacijų, kad būtų pašalintos visos pavardės ir inicialai

#### Ištrinami autoriai8: Pavarde, V.V.V.V.V.

data1['new_9']="

for index, row in enumerate(data1.new_8):
    autoriai_parsed = re.sub("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.]+
[A-Ž][.])", "", row)
    data1.new_9[index] = autoriai_parsed

#### Ištrinami autoriai7: Pavarde, V.V.V.V.

data1['new_10']="

for index, row in enumerate(data1.new_9):
    autoriai_parsed = re.sub("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.])",
"", row)
    data1.new_10[index] = autoriai_parsed

#### Ištrinami autoriai6: Pavarde, V.V.V.

data1['new_11']="

for index, row in enumerate(data1.new_10):

```

```

    autoriai_parsed = re.sub("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.]+ [A-Ž][.])", "", row)
    data1.new_11[index] = autoriai_parsed

#### Ištrinami autoriai5: Pavarde, V.V.

data1['new_12']="

for index, row in enumerate(data1.new_11):
    autoriai_parsed = re.sub("([A-Ž][a-ž]+, [A-Ž][.]+ [A-Ž][.])", "", row)
    data1.new_12[index] = autoriai_parsed

#### Ištrinami autoriai4: Pavarde Pavarde Pavarde, V.

data1['new_13']="

for index, row in enumerate(data1.new_12):
    autoriai_parsed = re.sub("([A-Ž][a-ž]+ [A-Ž][a-ž]+ [A-Ž][a-ž]+, [A-Ž].)", "",
row)
    data1.new_13[index] = autoriai_parsed

#### Ištrinami autoriai3: Pavarde-Pavarde, V.

data1['new_14']="

for index, row in enumerate(data1.new_13):
    autoriai_parsed = re.sub("([A-Ž][a-ž]+[-][A-Ž][a-ž]+, [A-Ž].)", "", row)
    data1.new_14[index] = autoriai_parsed

#### Ištrinami autoriai2: Pavarde Pavarde, V.

data1['new_15']="

for index, row in enumerate(data1.new_14):
    autoriai_parsed = re.sub("([A-Ž][a-ž]+ [A-Ž][a-ž]+, [A-Ž].)", "", row)
    data1.new_15[index] = autoriai_parsed

#### Ištrinami autoriai1: Pavarde, V.

data1['new_16']="

for index, row in enumerate(data1.new_15):
    autoriai_parsed = re.sub("([A-Ž][a-ž]+, [A-Ž].)", "", row)
    data1.new_16[index] = autoriai_parsed

data1 = data1.drop(['new_8', 'new_9', 'new_10', 'new_11', 'new_12', 'new_13',
'new_14', 'new_15'], axis=1)

## Publikacijos pavadinimo atskyrimas

Publikacija = data1['new_16'].str.split(' <i><i>', expand=True)

data1['publication'] = Publikacija[0]

data1['new_17'] = Publikacija[1]

data1 = data1.drop(['new_16'], axis=1)

## Leidinio numerio atskyrimas

```

```

#### Atskiriami numeriai, kurie turi 2 identifikatorius
Leidinio_numeriai_2 = data1['new_17'].str.split('</i></i>, <i>', expand=True)
data1['Leidinio_numeriai_2'] = Leidinio_numeriai_2[1]
data1['new_17'] = Leidinio_numeriai_2[0]

##### Atskiriamas leidinio numeris nuo mėnesinio numerio
Leidinio_numeris = data1['Leidinio_numeriai_2'].str.split('</i>', expand=True)
data1['journal_number'] = Leidinio_numeris[0]

##### Atskiriami leidinio puslapiai, kurie buvo užrašyti kitaip, nei anksčiau
atskirtieji
##### Mėnesiniai leidinio numeriai nesaugomi, nes formatas per daug nevienodas ir šių
duomenų informacija yra neaktuali

puslapiai = Leidinio_numeris[1].str.split(', ', expand=True)
puslapiai_1 = puslapiai[1].str.split('.', expand=True)
data1['Puslapiai'] = puslapiai_1[0]

#### Atskiriami numeriai, kurie turi 1 identifikatorių
Leidinio_numeriai_1 = data1['new_17'].str.split('</i></i>, ', expand=True)
data1['Leidinio_numeriai_1'] = Leidinio_numeriai_1[1]
data1['new_17'] = Leidinio_numeriai_1[0]

##### Atskiriami leidinio puslapiai, kurie buvo užrašyti kitaip, nei anksčiau
atskirtieji

puslapiai_2 = data1['Leidinio_numeriai_1'].str.split(', ', expand=True)
Leidinio_numeris_1 = puslapiai_2[0].str.split('.', expand=True)
puslapiai_2_2 = puslapiai_2[1].str.split('.', expand=True)
data1['Leidinio_numeris_1'] = Leidinio_numeris_1[0]
data1['Puslapiai_1'] = puslapiai_2_2[0]
data1 = data1.drop(['Leidinio_numeriai_1', 'Leidinio_numeriai_2'], axis=1)

## Leidinio pavadinimo išvalymas

#### [elektroninis išteklius] prierašo pašalinimas
elektroninis_isteklius = data1['new_17'].str.split(' \[Elektroninis išteklius\]',
expand=True)
data1['new_17'] = elektroninis_isteklius[0]

```

```

elektroninis_isteklius_1 = data1['new_17'].str.split(' \[elektroninis išteklis\]',
expand=True)

data1['new_17'] = elektroninis_isteklius_1[0]

#### \</i>\</i> priedašo pašalinimas

prierasas = data1['new_17'].str.split('</i></i>', expand=True)

data1['new_17'] = prierasas[0]

## Atskiriami leidinių pavadinimai kitomis kalbomis

Leidinio_pavadinimas_uzsienio_kalba = data1['new_17'].str.split(' = ', expand=True)

data1['journal_name'] = Leidinio_pavadinimas_uzsienio_kalba[0]

data1['journal_name_in_foreign_language_1'] = Leidinio_pavadinimas_uzsienio_kalba[1]

data1['journal_name_in_foreign_language_2'] = Leidinio_pavadinimas_uzsienio_kalba[2]

data1['journal_name_in_foreign_language_3'] = Leidinio_pavadinimas_uzsienio_kalba[3]

data1 = data1.drop(['new_17'], axis=1)

## Puslapių sutvarkymas

### Tušti sąrašai pakeičiami į NaN reikšmes

sarasas = ['Puslapiai_5_zenklai', 'Puslapiai_4_zenklai', 'Puslapiai_3_zenklai',
'Puslapiai_2_zenklai']

for column in sarasas:
    data1[column] = data1[column].apply(lambda x: np.nan if len(x)==0 else x)

### Sukuriamas vienas puslapių stulpelis

data1["Puslapiai_5_zenklai"] =
data1["Puslapiai_5_zenklai"].fillna(data1["Puslapiai_4_zenklai"])

data1["Puslapiai_5_zenklai"] =
data1["Puslapiai_5_zenklai"].fillna(data1["Puslapiai_3_zenklai"])

data1["Puslapiai_5_zenklai"] =
data1["Puslapiai_5_zenklai"].fillna(data1["Puslapiai_2_zenklai"])

data1["Puslapiai_5_zenklai"] =
data1["Puslapiai_5_zenklai"].fillna(data1["Puslapiai"])

data1["Puslapiai_5_zenklai"] =
data1["Puslapiai_5_zenklai"].fillna(data1["Puslapiai_1"])

data1 = data1.drop(['Puslapiai_4_zenklai', 'Puslapiai_3_zenklai',
'Puslapiai_2_zenklai', "Puslapiai", "Puslapiai_1"], axis=1)

data1.rename(columns = {'Puslapiai_5_zenklai' : 'pages'}, inplace = True)

```

```

### Puslapiai iš listo keičiami į stringą. Sudaromi 2 stulpeliai: puslapiai ir
publikacijos apimtis

#### Puslapių stulpelyje yra NaN, str ir list tipo reikšmių. Pirma užpildomos Nan
reikšmės su 0, tuomet parašoma funkcija, kuri tikrina, ar puslapio formatas yra
tinkamas (turi brūkšnį, kas reiškia, kad yra pradžios ir pabaigos puslapiai). Jei
reikšmė tinkama, tikrinama, ar ji yra list, ar str. Atitinkamai vertė paverčiama į
int. Apskaičiuojamas puslapių pradžios ir pabaigos skirtumas ir gaunama publikacijos
apimtis puslapiais

data1['pages'] = data1['pages'].fillna('0')

def parse_string(x):
    if '-' in x:
        pradzia, pabaiga = x.split('-')
        try:
            return int(pabaiga) - int(pradzia)
        except ValueError:
            return np.nan
    else:
        return np.nan

def puslapiai(x):
    # listas
    if isinstance(x, list):
        stringas = x[0]
        return parse_string(stringas)
    if isinstance(x, str):
        return parse_string(x)
    else:
        return np.nan

data1['pages'] = data1['pages'].apply(lambda x: ''.join(map(str, x)))

data1['pages'] = data1['pages'].replace(['p'], '0')

apimtis = data1['pages'].str.split('-', expand=True)

apimtis[0] = apimtis[0].fillna('0')

apimtis[0] = apimtis[0].replace([''], '0')

apimtis[1] = apimtis[1].fillna('0')

publikacijos_apimtis = apimtis[1].astype(int) - apimtis[0].astype(int)

data1['volume_in_pages'] = publikacijos_apimtis

### Sukuriamas vienas leidinio numerio stulpelis

#### Leidinio numeri sudaro labai skirtingi irasai, nežinau, kaip juos paversti į int

data1['journal_number'] = data1['journal_number'].fillna('0')

data1['journal_number'] = data1['journal_number'].replace([''], '0')

data1['journal_number'] =
data1['journal_number'].astype('str').str.extractall('(\d+)').unstack().fillna('').sum(
axis=1).astype(str).apply(lambda x: x.replace('.0', ''))

```

```

data1['Leidinio_numeris_1'] = data1['Leidinio_numeris_1'].fillna('0')

data1['Leidinio_numeris_1'] = data1['Leidinio_numeris_1'].replace([''], '0')

data1['Leidinio_numeris_1'] =
data1['Leidinio_numeris_1'].astype('str').str.extractall('(\d+)').unstack().fillna('')
).sum(axis=1).astype(str).apply(lambda x: x.replace('.0',''))

data1["journal_number"] = data1["journal_number"].fillna(data1["Leidinio_numeris_1"])

data1['journal_number'] = data1['journal_number'].replace(['0'], np.nan)

data1 = data1.drop(['Leidinio_numeris_1'], axis=1)

## scopus_info

### Stulpelis išskaidomas į dalis

#### sukuriamas scopus sources stulpelis su metais

scopus_info = data1['scopus_info'].str.split('(', expand=True)

scopus_sources = scopus_info[1].str.split(')', expand=True)

scopus_sources = scopus_sources[0].str.split(',', expand=True)

data1['scopus_sources'] = scopus_sources[0].fillna(0).astype(int)

#### scopus_info skaitinės vertės paverčiamos į float objektus

temp = scopus_info[0].str.split(' ', expand=True)

sarasas = [1, 3, 5]

for column in sarasas:
    temp[column] = temp[column].str[:-1].fillna(0)
    temp[column] = pd.to_numeric([str(x).replace(',','.') for x in temp[column]])

#### Sukuriami atskiri rodilių, ketvirčių ir metų stulpeliai ir perkeliami į
pagrindinę DataFrame

temp['SNIP'] = np.nan
temp['SNIP_2'] = np.nan
temp['CiteScore'] = np.nan
temp['SJR'] = np.nan
temp['SJR_2'] = np.nan

temp.loc[temp[0] == 'SNIP:', 'SNIP'] = temp[1]
temp.loc[temp[2] == 'SNIP:', 'SNIP_2'] = temp[3]
temp.loc[temp[0] == 'CiteScore:', 'CiteScore'] = temp[1]
temp.loc[temp[2] == 'SJR:', 'SJR'] = temp[3]
temp.loc[temp[4] == 'SJR:', 'SJR'] = temp[5]

temp['quartile_in_caterory'] = temp[6]
temp['scopus_sources'] = scopus_sources[0]

temp["SNIP"] = temp["SNIP"].fillna(temp["SNIP_2"])

```

```

temp["SJR"] = temp["SJR"].fillna(temp["SJR_2"])
temp["quartile_in_caterory"] = temp["quartile_in_caterory"].fillna(np.nan)

data1['SNIP'] = temp['SNIP']
data1['CiteScore'] = temp['CiteScore']
data1['SJR'] = temp['SJR']
data1['quartile_in_caterory'] = temp['quartile_in_caterory']
data1['scopus_sources'] = temp['scopus_sources']

data1 = data1.drop(['scopus_info'], axis=1)

## if_info

### Stulpelis išskaidomas į dalis

#### sukuriamas if_info stulpelis su metais

if_info = data1['if_info'].str.split('(', expand=True)

#### Dalis eiluciu turi dvigubus irasus. Patikrinama, koks eiluciu skaičius yra tokio
tipo:

print(((if_info[1].str.contains('; IF: ').value_counts().get(key = True)) /
(if_info[1].count()))).round(2)*100, '%')

#### Kadangi tokių eilučių yra 4%, papildomi įrašai nenagarinėjami

if_info_years = if_info[1].str.split(', ', expand=True)

InCites = if_info_years[1].str.split(')', expand=True)

InCites = InCites[0].str.split(' ', expand=True)

#### if_info skaitinės vertės paverčiamos į float objektus

temp_1 = if_info[0].str.split(' ', expand=True)

sarasas = [1, 3]

for column in sarasas:
    temp_1[column] = temp_1[column].str[:-1].fillna(0)
    temp_1[column] = pd.to_numeric([str(x).replace(',','.') for x in temp_1[column]])

#### Sukuriami atskiri rodilių, ketvirčių ir metų stulpeliai ir perkeliama į
pagrindinę DataFrame

temp_1['IF'] = np.nan
temp_1['AIF'] = np.nan

temp_1.loc[temp_1[0] == 'IF:', 'IF'] = temp_1[1]
temp_1.loc[temp_1[2] == 'AIF:', 'AIF'] = temp_1[3]

temp_1['quartile_in_caterory_2'] = temp_1[4]
temp_1['if_info_years'] = if_info_years[0]
temp_1['InCites_JCR'] = InCites[2]

data1['IF'] = temp_1['IF']
data1['AIF'] = temp_1['AIF']

```

```

data1['quartile_in_caterory_2'] = temp_1['quartile_in_caterory_2']
data1['if_info_years'] = temp_1['if_info_years']
data1['InCites_JCR'] = temp_1['InCites_JCR']

data1 = data1.drop(['if_info'], axis=1)

## dbases

### Stulpelis išskaidomas į dalis

dbases = data1['dbases'].str.split('; ', expand=True)

data1['dbases_count'] = dbases.count(1)

dbases = pd.concat([dbases[col] for col in dbases.columns]).value_counts().to_frame()

dbases.head(10)

#### Išskaidžius data bases stulpelį buvo gautos 340 skirtingų duomenų bazių, kuriose
saugomos publikacijos. Dalis iš jų pasikartoja tik kelis kartus. Dėl to buvo priimtas
sprendimas atskirai identifikuoti pirmąsias 10, o likusias priskirti kategorijai
"other_dbases"

data1['Scopus'] = data1['dbases'].str.contains('Scopus').fillna(0).astype(int)
data1['SCIE'] = data1['dbases'].str.contains('Science Citation Index
Expanded').fillna(0).astype(int)
data1['ICONDA'] = data1['dbases'].str.contains('ICONDA').fillna(0).astype(int)
data1['IndexCopernicus'] =
data1['dbases'].str.contains('IndexCopernicus').fillna(0).astype(int)
data1['INSPEC'] = data1['dbases'].str.contains('INSPEC').fillna(0).astype(int)
data1['Academic_Search_Complete'] = data1['dbases'].str.contains('Academic Search
Complete').fillna(0).astype(int)
data1['DOAJ'] = data1['dbases'].str.contains('DOAJ').fillna(0).astype(int)
data1['SSCI'] = data1['dbases'].str.contains('Social Sciences Citation
Index').fillna(0).astype(int)
data1['VINITI'] = data1['dbases'].str.contains('VINITI').fillna(0).astype(int)
data1['Business_Source_Complete'] = data1['dbases'].str.contains('Business Source
Complete').fillna(0).astype(int)

data1['other_dbases'] = 0

data1['other_dbases'] = data1['dbases_count']-data1['Scopus']-data1['SCIE']-
data1['ICONDA']-data1['IndexCopernicus']-data1['INSPEC']-
data1['Academic_Search_Complete']-data1['DOAJ']-data1['SSCI']-data1['VINITI']-
data1['Business_Source_Complete']

data1 = data1.drop(['dbases'], axis=1)

## res_fields / mokslo ir meno sritys

### Stulpelis išskaidomas į dalis

res_fields = data1['res_fields'].str.split('; ', expand=True)

res_fields.fillna(np.nan)

res_fields[0] = res_fields[0].replace('0', np.nan)
res_fields[0] = res_fields[0].replace('00', np.nan)

```

```

#### res_fields išskaidomas į dalis ir matoma, kad publikacijų, kurios būtų
priskirtos 4 arba 5 mokslo sritims yra labai mažai: 0.3 % ir 0.01 % atitinkamai. Dėl
to priimamas sprendimas nagrinėti tik pirmasias 3 mokslo sritis

print(((res_fields[4].count()) / (res_fields[0].count()))*100, '%')

print(((res_fields[3].count()) / (res_fields[0].count()))*100, '%')

print(((res_fields[2].count()) / (res_fields[0].count()))*100, '%')

data1['res_fields_1'] = res_fields[0]
data1['res_fields_2'] = res_fields[1]
data1['res_fields_3'] = res_fields[2]

data1 = data1.drop(['res_fields'], axis=1)

data1 = data1.drop(['projection_apu'], axis=1)

## Leidiniu pavadinimu sutvarkymas

journals = data1['journal_name']

#### Patikrinama, kiek yra unikalių žurnalų pavadinimų

unikalus_pavadinimai_skaicius = journals.str.strip().nunique()

unikalus_pavadinimai_skaicius

unikalus_pavadinimai = sorted(journals.astype(str).unique().tolist())

#### https://towardsdatascience.com/fuzzywuzzy-find-similar-strings-within-one-column-in-a-pandas-data-frame-99f6c2a0c212

#### Patikrinami 2 pavadinimų panašumo radimo būdai: Token Sort Ratio ir Token Set
Ratio

process.extract('Chemical and Process Engineering', unikalus_pavadinimai,
scorer=fuzz.token_sort_ratio)

process.extract('Chemical and Process Engineering', unikalus_pavadinimai,
scorer=fuzz.token_set_ratio)

##### Palyginus keletą atsitiktinių pavadinimų pastebėta, kad Token Set Ratio
pateikia tikslesnius rezultatus. Tačiau nepaisant to, tolimesnis modelis bus kuriamas
pritaikant abu būdus, kad būtų galima palyginti galutinius rezultatus

#### Pirma sukuriamas Token Sort Ratio modelis

#Create tuples of journals names, matched journals names, and the score
score_sort = [(x,) + i
               for x in unikalus_pavadinimai
               for i in process.extract(x, unikalus_pavadinimai,
                                       scorer=fuzz.token_sort_ratio)]

#Create a dataframe from the tuples
similarity_sort = pd.DataFrame(score_sort,
                              columns=['journals_sort', 'match_sort', 'score_sort'])

```

```

len(similarity_sort['journals_sort'])

#### Kadangi pavadinimų panašumai ieškomi viename stulpelyje, gaunami dubliuoti
rezultatai: Indian Journal of Physics - Indian journal of physics ir Indian journal
of physics - Indian Journal of Physics. Kad juos būtų galima pašalinti vėliau,
sukuriamas papildomas stulpelis

similarity_sort['sorted_journals_sort'] =
np.minimum(similarity_sort['journals_sort'], similarity_sort['match_sort'])
similarity_sort.head()

len(similarity_sort['journals_sort'])

#### Išvaloma DataFrame: pasirenkamas score_sort >= 95, panaikinamos eilutės, kuriose
atitikmuo yra tas pats pavadinimas ir panaikinami dublikatai

high_score_sort = similarity_sort[(similarity_sort['score_sort'] >= 95) &
(similarity_sort['journals_sort'] != similarity_sort['match_sort'])
&
(similarity_sort['sorted_journals_sort'] !=
similarity_sort['match_sort'])]
high_score_sort = high_score_sort.drop('sorted_journals_sort',axis=1).copy()

#### Rezultatai pateikiami DataFrame formatu

sugrupuoti_pavadinimai_sort =
high_score_sort.groupby(['journals_sort','score_sort']).agg(
{'match_sort': ', '.join}).sort_values(
['score_sort'], ascending=False)

sugrupuoti_pavadinimai_sort

len(sugrupuoti_pavadinimai_sort)

#### Gauta 220 pavadinimų porų, kurių atitikmuo yra >= 95

#### Visa žingsnių eiga pakartojama Token Set Ratio modeliui sukurti

#Create tuples of brand names, matched brand names, and the score
score_set = [(x,) + i
for x in unikalus_pavadinimai
for i in process.extract(x, unikalus_pavadinimai,
scorer=fuzz.token_set_ratio)]
#Create a dataframe from the tuples
similarity_set = pd.DataFrame(score_set,
columns=['journals_set','match_set','score_set'])

len(similarity_set['journals_set'])

similarity_set['sorted_journals_set'] = np.minimum(similarity_set['journals_set'],
similarity_set['match_set'])
similarity_set.head()

len(similarity_set['journals_set'])

high_score_set = similarity_set[(similarity_set['score_set'] >= 95) &
(similarity_set['journals_set'] != similarity_set['match_set']) &

```

```

        (similarity_set['sorted_journals_set'] !=
similarity_set['match_set'])]
high_score_set = high_score_set.drop('sorted_journals_set',axis=1).copy()

sugrupuoti_pavadinimai_set =
high_score_set.groupby(['journals_set','score_set']).agg(
    {'match_set': ', '.join}).sort_values(
        ['score_set'], ascending=False)

len(sugrupuoti_pavadinimai_set)

#### Gauta 505 pavadinimų porų, kurių atitikmuo yra >= 95

#### Kadangi Token Set Ratio atveju gauta ženkliai daugiau atitikmenų, šis modelis
pasirenkamas kaip toliau naudotinas darbe

#### Sukuriama nauja DataFrame, kurioje kiekvienam leidiniui suteikiamas indeksas

journals_set = pd.DataFrame(high_score_set['journals_set'].unique())

journals_set['index'] = np.arange(journals_set.shape[0])
journals_set = journals_set.rename(columns = {0:'journal'})

#### Šis indeksas pridedamas prie sugrupuotų leidinių duomenų lentelės

high_score_set_index = pd.merge(high_score_set, journals_set, how='left',
, left_on='journals_set', right_on='journal')

high_score_set_index = high_score_set_index.drop(['journal'], axis=1)

#### Buvo pastebėta, kad leidiniai lentelėje kartojasi: vienu atveju jie yra kaip
pirmas leidinys, kuriam priskirti kiti, kitu atveju - jis yra priskirtas kitiems
leidiniams. Deja, bet šis priskyrimas nesutampa 100%, todėl rezultate vienas
leidinys gali turėti kelis indeksus. Tam panaikinti, atskiriama antroji lentelės
dalis su indeksais.

index_check = high_score_set_index.drop(['journals_set', 'score_set'], axis=1)

index_check = index_check.rename(columns = {'match_set':'match_journal',
'index':'second_index'})

index_check = index_check.drop_duplicates(subset ="match_journal")

#### Toliau abi lentelės sujungiamos ir gaunamos 2 indeksų vertės: viena pirmoji,
kita priskirta vėliau. Tikslas jas palyginti ir suvienodinti

high_score_set_index_2 = pd.merge(high_score_set_index, index_check, how='left',
left_on='journals_set', right_on='match_journal')

#### Sukuriamas papildomas stulpelis su mažesne indekso verte

high_score_set_index_2['final_index'] =
high_score_set_index_2[['index','second_index']].min(axis=1)

#### Sukuriama atrskira DataFrame, kurioje būtų tik leidiniai ir jų galutiniai
indeksai

```

```

leidiniu_indeksai = high_score_set_index_2.drop(['score_set', 'index',
'match_journal', 'second_index'], axis=1)

leidiniu_indeksai_1 = leidiniu_indeksai.drop(['match_set'], axis=1)

leidiniu_indeksai_2 = leidiniu_indeksai.drop(['journals_set'], axis=1)

leidiniu_indeksai_1.columns = leidiniu_indeksai_2.columns
leidiniu_indeksai_visi = pd.concat([leidiniu_indeksai_1, leidiniu_indeksai_2],
axis=0, ignore_index=True)

leidiniu_indeksai_visi = leidiniu_indeksai_visi.drop_duplicates()

leidiniu_indeksai_visi = leidiniu_indeksai_visi.sort_values(by=['match_set',
'final_index'])

leidiniu_indeksai_visi = leidiniu_indeksai_visi.drop_duplicates(subset ="match_set")

#### Leidinių indeksai perkeliami į pagrindinę duomenų lentelę

data1 = pd.merge(data1, leidiniu_indeksai_visi, how='left' ,left_on='journal_name',
right_on='match_set')

#### Iš pagrindinės duomenų lentelės išfiltruojami leidiniai, kurie nebuvo priskirti
prie tų, kurie yra su panašiomis vertėmis. Jiems suteikiami atskiri indeksai

missing_index = data1[data1['final_index'].isnull()]

missing_index = missing_index[['journal_name', 'final_index']].drop_duplicates(subset
="journal_name").sort_values(by=['journal_name'])

missing_index = missing_index.drop_duplicates(subset ="journal_name")

missing_index['index'] = np.arange(missing_index.shape[0])

missing_index = missing_index.drop(['final_index'], axis=1)

missing_index['final_index_2'] =
missing_index['index']+max(leidiniu_indeksai_visi['final_index'])+1

missing_index = missing_index.drop(['index'], axis=1)

#### Pavienių leidinių indeksai perkeliami į pagrindinę duomenų lentelę

data1 = pd.merge(data1, missing_index, how='left' ,left_on='journal_name',
right_on='journal_name')

data1["final_index"] = data1["final_index"].fillna(data1["final_index_2"])

data1 = data1.drop(['final_index_2', 'match_set'], axis=1)

#### Originaliai buvusių skirtingų leidinių pavadinimų ir sugrupuotų santykis

data1['journal_name'].unique().size

data1['final_index'].unique().size

(data1['final_index'].unique().size / data1['journal_name'].unique().size)

```

```

## Leidinių ir publikacijų duomenų rinkinių sukūrimas

### Sukurtos DataFrame perkeliamos į csv. failus ir išsaugomos prie duomenų, kad būtų
naudojamos tolimesniuose procesuose

publikacijos = data1.drop(['journal_number',
                           'journal_name_in_foreign_language_1',
                           'journal_name_in_foreign_language_2',
                           'journal_name_in_foreign_language_3',
                           'scopus_sources',
                           'SNIP',
                           'CiteScore',
                           'SJR',
                           'quartile_in_caterory',
                           'IF',
                           'AIF',
                           'quartile_in_caterory_2',
                           'if_info_years',
                           'InCites_JCR'], axis=1)

leidiniai = data1.drop(['docs_id',
                        'doc_code',
                        'pub_year',
                        'doi',
                        'pages',
                        'publication',
                        'volume_in_pages',
                        'dbases_count',
                        'Scopus',
                        'SCIE',
                        'ICONDA',
                        'IndexCopernicus',
                        'INSPEC',
                        'Academic_Search_Complete',
                        'DOAJ',
                        'SSCI',
                        'VINITI',
                        'Business_Source_Complete',
                        'other_dbases',
                        'res_fields_1',
                        'res_fields_2',
                        'res_fields_3'], axis=1)

#publikacijos.to_csv('publikacijos.csv')

#leidiniai.to_csv('leidiniai.csv')

## Autoriai

#### Įkeliamas autorių failas, kuriame autorių vardai pakeisti į indeksus. Šis
procesas buvo atliktas Excel programoje:

- Autorių pavardės peržiūrėtos ir suvienodintos \
- Pašalinti dublikatai \
- Sukurti indeksai

```

```

authors = pd.read_csv('authors.csv')

#### Apskaičiuojama, kiek kartų kiekvienas autorius pasikartojo duomenų rinkinyje

authors_count = pd.concat([authors[col] for col in
authors.columns]).value_counts().to_frame()

### Autorių bendraautorystės sąrašo sudarymas

#### Nan reikšmės pakeičiamos į 0

autoriu_sarasas = authors.fillna(0).values.astype(int).tolist()

#### Pašalinami visi 0 iš autorių sąrašų

bendraautoriai_visi = [[ele for ele in sub if ele != 0] for sub in autoriu_sarasas]

#### Pašalinami visi tušti sąrašai, kurie reiškia, kad publikaciją parašė vienas
autorius (nėra bendraautorystės)

bendraautoriai_poros = [[ele for ele in sub if len(sub) != 1] for sub in
bendraautoriai_visi]

bendraautoriai = list(filter(lambda x: len(x) != 0, bendraautoriai_poros))

#### Sąrašai išrikiuojami didėjimo tvarka

bendraautoriai_1 = [sorted(i) for i in bendraautoriai]

#### Gaunamos visos unikalios autorių poros

poros = [list(itertools.combinations(i,2)) for i in bendraautoriai_1]

#### Poros + svoriai

autoriai_dataframe = pd.DataFrame([i for j in poros for i in j])

edges =
autoriai_dataframe.groupby(autoriai_dataframe.columns.tolist(),as_index=False).size()

edges = edges.drop(edges[edges[0] == edges[1]].index)

edges

#### Apskaičiuojama, kiek yra unikalių bendraautorystės porų

edges.groupby([0, 1]).ngroups

#### Gauta DataFrame išsaugoma kaip atskiras failas, kuris, naudojamas grafo kūrimui.
Pridedamas papildomas stulpelis, kuriame nurodoma, kad grafas nėra orientuotas
(Undirected), nes bendraautorystė yra lygi abiejomis kryptimis

## Autoriai IKI 20 BENDRADARBIAVIMO

#### Kuriamas analogiškas autorių bendradarbiavimo sąrašas, tačiau jame pašalintos
išskirtys: 6 publikacijos, kurių autorių skaičius >= 20

#### Įkeliamas autorių failas, kuriame autorių vardai pakeisti į indeksus

```

```

authors_adj = pd.read_csv('authors_adjusted.csv')

#### Apskaičiuojama, kiek kartų kiekvienas autorius pasikartojo duomenų rinkinyje

authors_count_adj = pd.concat([authors_adj[col] for col in
authors_adj.columns]).value_counts().to_frame()

#### Nan reikšmės pakeičiamos į 0

autoriu_sarasas_adj = authors_adj.fillna(0).values.astype(int).tolist()

#### Pašalinami visi 0 iš autorių sąrašų

bendraautoraii_visi_adj = [[ele for ele in sub if ele != 0] for sub in
autoriu_sarasas_adj]

#### Pašalinami visi tušti sąrašai, kurie reiškia, kad publikaciją parašė vienas
autorius (nėra bendraautorystės)

bendraautoraii_poros_adj = [[ele for ele in sub if len(sub) != 1] for sub in
bendraautoraii_visi_adj]

bendraautoraii_adj = list(filter(lambda x: len(x) != 0, bendraautoraii_poros_adj))

#### Sąrašai išrikiuojami didėjimo tvarka

bendraautoraii_1_adj = [sorted(i) for i in bendraautoraii_adj]

#### Gaunamos visos unikalios autorių poros

poros_adj = [list(itertools.combinations(i,2)) for i in bendraautoraii_1_adj]

#### Poros + svoriai

autoriai_dataframe_adj = pd.DataFrame([i for j in poros_adj for i in j])

edges_adj =
autoriai_dataframe_adj.groupby(autoriai_dataframe_adj.columns.tolist(),as_index=False
).size()

edges_adj = edges_adj.drop(edges_adj[edges_adj[0] == edges_adj[1]].index)

#### Apskaičiuojama, kiek yra unikalių bendraautorystės porų

edges_adj.groupby([0, 1]).ngroups

```

2. Python kodas, naudotas bibliometrinei analizei atlikti

```

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

### Duomenų rinkinių nuskaitymas

```

```

publikacijos = pd.read_csv('publikacijos.csv')
publikacijos = publikacijos.iloc[:, 1:]
publikacijos['pub_year'] = publikacijos['pub_year'].replace([3010], 2010)
mokslo_sritis = publikacijos['res_fields_1'].str.split(' ', expand=True)
publikacijos['science_field'] = mokslo_sritis[0]
bendros_sritys = pd.read_excel('science_field_name.xlsx')
visos_sritys = pd.read_excel('res_field_name.xlsx')
dokumentai = pd.read_excel('doc_code.xlsx')
autoriu_skaicius = pd.read_excel('author_number.xlsx')
publikacijos = pd.merge(publikacijos,
                        bendros_sritys,
                        on = 'science_field',
                        how = 'inner')
publikacijos = pd.merge(publikacijos,
                        visos_sritys,
                        on = 'res_fields_1',
                        how = 'inner')
publikacijos = pd.merge(publikacijos,
                        dokumentai,
                        on = 'doc_code',
                        how = 'inner')
publikacijos = pd.merge(publikacijos,
                        autoriu_skaicius,
                        on = 'docs_id',
                        how = 'inner')

publikacijos.tail(5)

### Missing values apskaičiavimas

data = pd.read_excel('data1.xlsx')
data_raw = pd.read_csv('data.csv')
row_count = data_raw.shape[0]
missing_values = round((data_raw.isnull().sum() / row_count), 2).to_frame()
missing_values = missing_values.to_string(formatters={
    0: '{:, .2%}'.format
})
print(missing_values)

### Bibliometrinė analizė

```

```

#### Publikacijų skaičius 2010 – 2019 periode

metai = pd.DataFrame(publikacijos.groupby(['pub_year'])['pub_year'].count())

metai = metai.drop(metai.tail(1).index,
                  inplace=False).drop(metai.head(1).index,
                  inplace=False).rename(columns = {'pub_year' :
'Count'},
                                     inplace =
False).reset_index()

#### Vidutinis publikacijų skaičius

metai['Count'].mean()

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
rs = np.random.RandomState(8)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10,7))
ax.grid(False)

x = metai['pub_year']
y = metai['Count']
pub_sk = sns.barplot(x=x, y=y, ax=ax, color='skyblue')
pub_sk.axhline(metai['Count'].mean(), color='red')
plt.ylabel('Publikacijų skaičius')
plt.xlabel('Metai')
for i in ax.containers:
    ax.bar_label(i)
plt.show()

#### Mokslo ir meno sričių klasifikatoriai

sritys =
pd.DataFrame(publikacijos.groupby(['science_field_name'])['science_field_name'].count
()).rename(columns = {'science_field_name' : 'Count'},
                                     inplace =
False).reset_index().sort_values(by = 'Count', ascending=False)

sritys

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
rs = np.random.RandomState(8)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10,7))
ax.grid(False)

x = sritys['science_field_name']
y = sritys['Count']
sns.barplot(x=x, y=y, ax=ax, color='skyblue')
plt.xticks(rotation=90)
plt.ylabel('Publikacijų skaičius')
plt.xlabel("")
for i in ax.containers:
    ax.bar_label(i)

```

```

#### Tik T (Technologijos mokslai) mokslo sritis

T = publikacijos[publikacijos['science_field'] == 'T']

T_sritys =
pd.DataFrame(T.groupby(['res_field_name'])['res_field_name'].count()).rename(columns
= {'res_field_name' : 'Count'},
                                                    inplace =
False).reset_index().sort_values(by = 'Count', ascending=False)

T_sritys['%'] = ((T_sritys['Count'] /
T_sritys['Count'].sum()*100).round(2).astype(str) + '%'

T_sritys

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
rs = np.random.RandomState(8)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10,7))
ax.grid(False)

x = T_sritys['res_field_name']
y = T_sritys['Count']
sns.barplot(x=x, y=y, ax=ax, color='skyblue')
plt.xticks(rotation=90)
plt.ylabel('Publikacijų skaičius')
plt.xlabel("")
for i in ax.containers:
    ax.bar_label(i)

T_sritys_2 = T[['pub_year', 'res_field_name']]

T_sritys_2

T_sritys_2['count'] = 1

T_sritys_3 = T_sritys_2.groupby(['pub_year', 'res_field_name'],
as_index=False).count()

#### Statybos inžinerijos srities pokytis nagrinėjamu periodu

pirminis = T_sritys_3[(T_sritys_3['pub_year'] == 2010) &
(T_sritys_3['res_field_name'] == 'Statybos inžinerija')]

galutinis = T_sritys_3[(T_sritys_3['pub_year'] == 2019) &
(T_sritys_3['res_field_name'] == 'Statybos inžinerija')]

pokytis = (((galutinis['count'].values[0].astype(int) -
pirminis['count'].values[0].astype(int)) /
pirminis['count'].values[0].astype(int))*100).round(2).astype(str) + '%'

pokytis

#### Technologijos mokslų sričių pokytis per 2010 – 2019 metus

f, ax = plt.subplots(figsize=(12, 5))
ax.grid(False)

```

```

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
sns.set_theme(style="white")
sns.lineplot(x="pub_year", y="count",
             hue="res_field_name",
             data=T_sritys_3)
plt.ylabel('Publikacijų skaičius')
plt.xlabel('Metai')
plt.legend(loc=(1.04,0));

#### Dokumentų tipai

dokumentu_tipai = publikacijos[['pub_year', 'doc_code_name']]

dokumentu_tipai['count'] = 1

dokumentu_tipai = dokumentu_tipai.groupby(['pub_year', 'doc_code_name'],
as_index=False).count()

dokumentu_tipai = dokumentu_tipai[(dokumentu_tipai['pub_year'] > 2009) &
(dokumentu_tipai['pub_year'] < 2020)]

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
f, ax = plt.subplots(figsize=(12, 5))
ax.grid(False)
sns.lineplot(x="pub_year", y="count",
             hue="doc_code_name",
             data=dokumentu_tipai)
plt.ylabel('Dokumentų skaičius')
plt.xlabel('Metai')
plt.legend(loc=(1.04,0));

#### Straipsnis Clarivate Analytics Web of Science pokytis

pirminis = dokumentu_tipai[(dokumentu_tipai['pub_year'] == 2010) &
(dokumentu_tipai['doc_code_name'] == 'Straipsnis Clarivate Analytics Web of Science')]

galutinis = dokumentu_tipai[(dokumentu_tipai['pub_year'] == 2019) &
(dokumentu_tipai['doc_code_name'] == 'Straipsnis Clarivate Analytics Web of Science')]

pokytis = (((galutinis['count'].values[0].astype(int) -
pirminis['count'].values[0].astype(int)) /
pirminis['count'].values[0].astype(int))*100).round(2).astype(str) + '%'

pokytis

#### Autorių analizė

authors = pd.read_csv('authors.csv')

pavardes = pd.read_excel('autoriu_pavardes.xlsx')

#### Autorių indeksai perkeliama į vieną stulpelį ir apskaičiuojama, kiek kartų
kiekvienas iš jų pasikartoja duomenų rinkinyje

authors_count = pd.concat([authors[col] for col in
authors.columns]).to_frame().rename(columns = {0 : 'index'},

```

```

inplace = False)

visi = authors_count.groupby('index').size().to_frame().sort_values(by=0,
ascending=False).rename(columns = {0 : 'Count'},
inplace = False)

visi.shape[0]

#### Išrenkami 10 produktyviausių autorių

visi_vardai = pd.merge(visi, pavardes, how='left' ,left_on='index', right_on='index')

visi_vardai.head(10)

#### Bendraautorystės analizė

authors['bendraautoriai'] = authors.count(axis = 1)

##### Vidutinis bendraautorijų skaičius

authors['bendraautoriai'].mean().round(2)

bendraautoriai = authors['bendraautoriai'].value_counts().to_frame()

bendraautoriai

#### Duomenų bazių analizė

dbases = data[['pub_year', 'dbases_count',
                'Scopus', 'SCIE',
                'ICONDA', 'IndexCopernicus',
                'INSPEC', 'Academic_Search_Complete',
                'DOAJ', 'SSCI',
                'VINITI', 'Business_Source_Complete',
                'other_dbases']]

dbases_2 = dbases.groupby(['pub_year'], as_index=False).sum()

dbases_2 =
dbases_2.drop(dbases_2.head(1).index,inplace=False).drop(dbases_2.tail(2).index,inpla
ce=False).set_index('pub_year')

dbases_3 = dbases_2.drop(['dbases_count', 'other_dbases'], axis=1)

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")

fig, ax = plt.subplots(figsize=(12,6))
ax.grid(False)
sns.lineplot(data=dbases_3)

plt.ylabel('Dokumentų skaičius')
plt.xlabel('Metai')
plt.legend(loc=(1.04,0));

#### Pagrindinių duomenų bazių publikacijų skaičiaus pokytis

```

```
Scopus = (((dbases_3['Scopus'].iloc[-1] - dbases_3['Scopus'].iloc[0]) /
dbases_3['Scopus'].iloc[0])*100).round(2).astype(str) + '%'
```

Scopus

```
SCIE = (((dbases_3['SCIE'].iloc[-1] - dbases_3['SCIE'].iloc[0]) /
dbases_3['SCIE'].iloc[0])*100).round(2).astype(str) + '%'
```

SCIE

```
IndexCopernicus = (((dbases_3['IndexCopernicus'].iloc[-1] -
dbases_3['IndexCopernicus'].iloc[0]) /
dbases_3['IndexCopernicus'].iloc[0])*100).round(2).astype(str) + '%'
```

IndexCopernicus

3. Python kodas, skirtas grafų teorijos taikumui

```
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import scipy

## Pilnas grafas

### Sukuriamas grafas iš visų viršūnių ir briaunų

df = pd.read_csv('edges_pilnas.csv')
Graphtype = nx.MultiDiGraph()
graph = nx.from_pandas_edgelist(df, source='source', target='target',
edge_attr='weight', create_using=Graphtype)

### Grafas nagrinėjamas apskaičiuojant įvairius rodiklius

#### Viršūnių laipsniai, viršūnių laipsnių pasiskirstymas

degrees = [val for (node, val) in graph.degree()]

degrees_count = {i:degrees.count(i) for i in degrees}

sorted_values_degrees = sorted(degrees_count.values(), reverse=True) # Sort the
values
sorted_dict_degrees = {}

for i in sorted_values_degrees:
    for k in degrees_count.keys():
        if degrees_count[k] == i:
            sorted_dict_degrees[k] = degrees_count[k]
            break

degrees_count_10 = pd.DataFrame(list(sorted_dict_degrees.items())[:10])

degrees_count_10
```

```

degrees_all = pd.DataFrame(list(sorted_dict_degrees.items()))

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(15, 6.5))
ax.grid(False)
ax = sns.barplot(x=0, y=1, data=degrees_all, color='skyblue')
plt.ylabel('Viršūnių laipsnių skaičius')
plt.xlabel('Viršūnių laipsniai');

virsuniu_statistika = scipy.stats.describe(degrees)

virsunes = pd.DataFrame([virsuniu_statistika], columns=virsuniu_statistika._fields)

virsunes

#### https://www.geeksforgeeks.org/scipy-stats-skew-python/

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(10, 6.5))
ax.grid(False)
sns.distplot(degrees, color='blue')
print("Skewness: %f" % scipy.stats.skew(degrees))
print("Kurtosis: %f" % scipy.stats.kurtosis(degrees))
plt.ylabel('Tankis')
plt.xlabel('Viršūnių laipsniai');

#### Grafo jungumo matai

### https://stackoverflow.com/questions/56624273/understanding-average-degree-connectivity-in-networkx

average_degree_connectivity = nx.average_degree_connectivity(graph)

df_degree = pd.DataFrame(list(average_degree_connectivity.items()))

df_degree

#connectivity = nx.average_node_connectivity(graph)

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(10, 6.5))
ax.grid(False)
plt.plot(*zip(*sorted(average_degree_connectivity.items())), color='skyblue')
plt.ylabel('Vidutiniai viršūnių laipsniai')
plt.xlabel('Viršūnių laipsniai')

average_neighbor_degree = nx.average_neighbor_degree(graph)

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(10, 6.5))
ax.grid(False)
plt.plot(*zip(*sorted(average_neighbor_degree.items())), color='skyblue')
plt.ylabel('Vidutiniai kaimyninių viršūnių laipsniai')
plt.xlabel('Viršūnės')

#### Laipsnių koreliacija

```

```

#### https://ona-book.org/similarity.html

nx.degree_assortativity_coefficient(graph).round(3)

nx.degree_pearson_correlation_coefficient(graph).round(3)

#### Grafo skersmuo

#### This finds the maximum distance of a list containing the shortest paths between
any two nodes in G (computed with Dijkstra's algorithm), regardless of what component
they may belong to. Technically, diameter is infinite for disconnected graphs which
is why NetworkX's built-in method does not work. The method above will find the
largest diameter amongst all components within G, but is not the diameter of G
itself.

diameter = max([max(j.values()) for (i,j) in nx.shortest_path_length(graph)])

diameter

#### Grafo tankumas

nx.density(graph)

### Centriškumo matai

#### Laipsnio centriškumo matas - svarbiausi tinklo nariai

pavardes = pd.read_excel('autoriu_pavardes.xlsx')

centriskumas = nx.degree_centrality(graph)

sorted_values_c = sorted(centriskumas.values(), reverse=True) # Sort the values
sorted_dict_c = {}

for i in sorted_values_c:
    for k in centriskumas.keys():
        if centriskumas[k] == i:
            sorted_dict_c[k] = centriskumas[k]
            break

centriskumas_10 = pd.DataFrame(list(sorted_dict_c.items())[:10])

centriskumas_10 = pd.merge(centriskumas_10, pavardes, how='left', left_on=0,
right_on='index')

centriskumas_10 = centriskumas_10[['authors', 1]].rename(columns = {'authors' :
'Autorius', 1 : 'Laipsnio centriškumo matas'}),

inplace =
False).round(3)

centriskumas_10

#### Artumo centriškumo matas - nariai, esantys arčiausiai kitų narių

artumas = nx.closeness_centrality(graph)

sorted_values_a = sorted(artumas.values(), reverse=True) # Sort the values
sorted_dict_a = {}

```

```

for i in sorted_values_a:
    for k in artumas.keys():
        if artumas[k] == i:
            sorted_dict_a[k] = artumas[k]
            break

artumas_10 = pd.DataFrame(list(sorted_dict_a.items())[:10])

artumas_10 = pd.merge(artumas_10, pavardes, how='left' ,left_on=0, right_on='index')

artumas_10 = artumas_10[['authors', 1]].rename(columns = {'authors' : 'Autorius', 1 :
'Laipsnio centriškumo matas'}),
                                                    inplace =
False).round(3)

artumas_10

#### Tarpusavio centriškumo matas - nariai, turintys daugiausiai trumpiausių ryšių su
kitais nariais

tarpusavio = nx.betweenness_centrality(graph)

sorted_values_t = sorted(tarpusavio.values(), reverse=True) # Sort the values
sorted_dict_t = {}

for i in sorted_values_t:
    for k in tarpusavio.keys():
        if tarpusavio[k] == i:
            sorted_dict_t[k] = tarpusavio[k]
            break

tarpusavio_10 = pd.DataFrame(list(sorted_dict_t.items())[:10])

tarpusavio_10 = pd.merge(tarpusavio_10, pavardes, how='left' ,left_on=0,
right_on='index')

tarpusavio_10 = tarpusavio_10[['authors', 1]].rename(columns = {'authors' :
'Autorius', 1 : 'Laipsnio centriškumo matas'}),
                                                    inplace =
False).round(3)

tarpusavio_10

from networkx.algorithms.community.centrality import girvan_newman
a = girvan_newman(graph)

bendruomenes = tuple(sorted(c) for c in next(a))

len(bendruomenes)

result = pd.DataFrame(data=bendruomenes).fillna('0').astype(int).T

result = pd.DataFrame(result)

result['index'] = result.index

result

```

```

df1 = result.set_index('index').stack().reset_index()
df1.columns = ['ID', 'bendruomene', 'virsune']

df1 = df1.drop(['ID'], axis=1)

df1 = df1.drop(df1.index[df1['virsune'] == 0], inplace=False).sort_values(by =
'bendruomene')

df1['bendruomene'] = df1['bendruomene'].apply(lambda x: x + 1)

df1

df1.to_excel('df2.xlsx')

## Grafas, iki 20 bendraaautorių

df = pd.read_csv('edges_adj.csv')
Graphtype = nx.MultiDiGraph()
graph = nx.from_pandas_edgelist(df, source='source', target='target',
edge_attr='weight', create_using=Graphtype)

#### Viršūnių laipsniai, viršūnių laipsnių pasiskirstymas

degrees = [val for (node, val) in graph.degree()]

degrees_count = {i:degrees.count(i) for i in degrees}

sorted_values_degrees = sorted(degrees_count.values(), reverse=True) # Sort the
values
sorted_dict_degrees = {}

for i in sorted_values_degrees:
    for k in degrees_count.keys():
        if degrees_count[k] == i:
            sorted_dict_degrees[k] = degrees_count[k]
            break

degrees_count_10 = pd.DataFrame(list(sorted_dict_degrees.items())[:10])

degrees_count_10

degrees_all = pd.DataFrame(list(sorted_dict_degrees.items()))

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(15, 6.5))
ax.grid(False)
ax = sns.barplot(x=0, y=1, data=degrees_all, color='skyblue')
plt.ylabel('Viršūnių laipsnių skaičius')
plt.xlabel('Viršūnių laipsniai');

virsuniu_statistika = scipy.stats.describe(degrees)

virsunes = pd.DataFrame([virsuniu_statistika], columns=virsuniu_statistika._fields)

virsunes

#### https://www.geeksforgeeks.org/scipy-stats-skew-python/

```

```

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(10, 6.5))
ax.grid(False)
sns.distplot(degrees, color='blue')
print("Skewness: %f" % scipy.stats.skew(degrees))
print("Kurtosis: %f" % scipy.stats.kurtosis(degrees))
plt.ylabel('Tankis')
plt.xlabel('Viršūnių laipsniai');

#### Grafo jungumo matai

average_degree_connectivity = nx.average_degree_connectivity(graph)

df_degree = pd.DataFrame(list(average_degree_connectivity.items()))

df_degree

#connectivity = nx.average_node_connectivity(graph)

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(10, 6.5))
ax.grid(False)
plt.plot(*zip(*sorted(average_degree_connectivity.items())), color='skyblue')
plt.ylabel('Vidutiniai viršūnių laipsniai')
plt.xlabel('Viršūnių laipsniai')

average_neighbor_degree = nx.average_neighbor_degree(graph)

sns.set_theme(style="whitegrid", font_scale=1.3, font="Times New Roman")
f, ax = plt.subplots(figsize=(10, 6.5))
ax.grid(False)
plt.plot(*zip(*sorted(average_neighbor_degree.items())), color='skyblue')
plt.ylabel('Vidutiniai kaimyninių viršūnių laipsniai')
plt.xlabel('Viršūnės')

#### Laipsnių koreliacija

nx.degree_assortativity_coefficient(graph).round(3)

nx.degree_pearson_correlation_coefficient(graph).round(3)

#### Grafo skersmuo

diameter = max([max(j.values()) for (i,j) in nx.shortest_path_length(graph)])

diameter

#### Grafo tankumas

nx.density(graph)

### Centriškumo matai

#### Laipsnio centriškumo matas - svarbiausi tinklo nariai

pavardes = pd.read_excel('autoriu_pavardes.xlsx')

```

```

centriskumas = nx.degree_centrality(graph)

sorted_values_c = sorted(centriskumas.values(), reverse=True) # Sort the values
sorted_dict_c = {}

for i in sorted_values_c:
    for k in centriskumas.keys():
        if centriskumas[k] == i:
            sorted_dict_c[k] = centriskumas[k]
            break

centriskumas_10 = pd.DataFrame(list(sorted_dict_c.items())[:10])

centriskumas_10 = pd.merge(centriskumas_10, pavardes, how='left' ,left_on=0,
right_on='index')

centriskumas_10 = centriskumas_10[['authors', 1]].rename(columns = {'authors' :
'Autorius', 1 : 'Laipsnio centriškumo matas'}),
                                                    inplace =
False).round(3)

centriskumas_10

#### Artumo centriškumo matas - nariai, esantys arčiausiai kitų narių

artumas = nx.closeness_centrality(graph)

sorted_values_a = sorted(artumas.values(), reverse=True) # Sort the values
sorted_dict_a = {}

for i in sorted_values_a:
    for k in artumas.keys():
        if artumas[k] == i:
            sorted_dict_a[k] = artumas[k]
            break

artumas_10 = pd.DataFrame(list(sorted_dict_a.items())[:10])

artumas_10 = pd.merge(artumas_10, pavardes, how='left' ,left_on=0, right_on='index')

artumas_10 = artumas_10[['authors', 1]].rename(columns = {'authors' : 'Autorius', 1 :
'Laipsnio centriškumo matas'}),
                                                    inplace =
False).round(3)

artumas_10

#### Tarpusavio centriškumo matas - nariai, turintys daugiausiai trumpiausių ryšių su
kitais nariais

tarpusavio = nx.betweenness_centrality(graph)

sorted_values_t = sorted(tarpusavio.values(), reverse=True) # Sort the values
sorted_dict_t = {}

for i in sorted_values_t:
    for k in tarpusavio.keys():
        if tarpusavio[k] == i:

```

```

        sorted_dict_t[k] = tarpusavio[k]
        break

tarpusavio_10 = pd.DataFrame(list(sorted_dict_t.items())[:10])

tarpusavio_10 = pd.merge(tarpusavio_10, pavardes, how='left' ,left_on=0,
right_on='index')

tarpusavio_10 = tarpusavio_10[['authors', 1]].rename(columns = {'authors' :
'Autorius', 1 : 'Laipsnio centriškumo matas'},
                                                    inplace =
False).round(3)

tarpusavio_10

from networkx import edge_betweenness centrality as betweenness

from networkx.algorithms.community.centralities import girvan_newman
a = girvan_newman(graph)

bendruomenes = tuple(sorted(c) for c in next(a))

len(bendruomenes)

result = pd.DataFrame(data=bendruomenes).fillna('0').astype(int).T

result = pd.DataFrame(result)

result['index'] = result.index

result

df1 = result.set_index('index').stack().reset_index()
df1.columns = ['ID', 'bendruomene', 'virsune']

df1 = df1.drop(['ID'], axis=1)

df1 = df1.drop(df1.index[df1['virsune'] == 0], inplace=False).sort_values(by =
'bendruomene')

df1['bendruomene'] = df1['bendruomene'].apply(lambda x: x + 1)

df1.to_excel('df1.xlsx')

```

4. Python kodas, skirtas klasterinei analizei atlikti

```

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from yellowbrick.cluster import silhouette_visualizer
import pickle

```

```

### Leidinių duomenų rinkinio sutvarkymas

leidiniai = pd.read_csv('leidiniai.csv')

leidiniai

#### Pirmu žingsniu pašalinamos eilutės, kuriose nėra leidinių rodiklių

leidiniai_2 = leidiniai.dropna(subset=['scopus_sources', 'IF'], how='all')

#### Sukuriama atskira DataFrame su leidiniais, kurie neturi rodiklių

leidiniai_be_rodiklių = leidiniai.drop(leidiniai_2.index)

#### Palyginama, kiek leidinių buvo iš viso ir kiek jų buvo galima klasterizuoti, ir
kiek - ne

pasalinti = np.count_nonzero(leidiniai_be_rodiklių['final_index'].unique())

tinkami = np.count_nonzero(leidiniai_2['final_index'].unique())

visi = pasalinti + tinkami

print("Žurnalai, kurių nebuvo galima klasterizuoti ", round((pasalinti /
visi)*100,2), '%')

print("Žurnalai, kurie buvo klasterizuojami", round((tinkami / visi)*100,2), '%')

### Sutvarkomas gautas duomenų rinkinys

#### 1. Pašalinami neinformatyvūs stulpeliai

leidiniai_2 = leidiniai_2.drop(['Unnamed: 0', 'journal_number',
'journal_name_in_foreign_language_1',
'journal_name_in_foreign_language_2',
'journal_name_in_foreign_language_3'], axis=1)

#### 2. NaN reikšmės pakeičiamos į 0, str reikšmės paverčiamos į int ir pašalinami
dublikatai

leidiniai_2 = leidiniai_2.fillna(0)

leidiniai_2[['scopus_sources', 'if_info_years', 'final_index']] =
leidiniai_2[['scopus_sources', 'if_info_years', 'final_index']].astype('int')

leidiniai_2 = leidiniai_2.drop_duplicates()

#### 3. Sukuriama nauja DataFrame tik Scopus rodikliams. Naudojantis pivot table
funkcija, duomenys pergrupuojami, kad metai taptų stulpeliais, o rodikliai -
eilutėmis. Tai daroma kiekvienam rodikliui atskirai ir galiausiai sukuriami viena
DataFrame

leidiniai_3 = leidiniai_2[['final_index', 'scopus_sources', 'SNIP', 'CiteScore',
'SJR', 'quartile_in_caterory']]

quartile_in_caterory = leidiniai_3['quartile_in_caterory'].str.split('Q',
expand=True)

```

```

leidiniaai_3 = leidiniaai_3.drop(['quartile_in_caterory'], axis=1)

leidiniaai_3['quartile_in_caterory'] = quartile_in_caterory[1].fillna(0).astype(int)

##### SNIP

leidiniaai_3_SNIP = leidiniaai_3.pivot_table(index="final_index",
columns="scopus_sources", values='SNIP')

leidiniaai_3_SNIP = leidiniaai_3_SNIP.add_suffix('_SNIP')

leidiniaai_3_SNIP=leidiniaai_3_SNIP.reset_index()

leidiniaai_3_SNIP = leidiniaai_3_SNIP.drop(['0_SNIP'], axis=1)

##### CiteScore

leidiniaai_3_CiteScore = leidiniaai_3.pivot_table(index="final_index",
columns="scopus_sources", values='CiteScore')

leidiniaai_3_CiteScore = leidiniaai_3_CiteScore.add_suffix('_CiteScore')

leidiniaai_3_CiteScore=leidiniaai_3_CiteScore.reset_index()

leidiniaai_3_CiteScore = leidiniaai_3_CiteScore.drop(['0_CiteScore'], axis=1)

##### SJR

leidiniaai_3_SJR = leidiniaai_3.pivot_table(index="final_index",
columns="scopus_sources", values='SJR')

leidiniaai_3_SJR = leidiniaai_3_SJR.add_suffix('_SJR')

leidiniaai_3_SJR=leidiniaai_3_SJR.reset_index()

leidiniaai_3_SJR = leidiniaai_3_SJR.drop(['0_SJR'], axis=1)

##### quartile_in_caterory

leidiniaai_3_quartile_in_caterory = leidiniaai_3.pivot_table(index="final_index",
columns="scopus_sources", values='quartile_in_caterory')

leidiniaai_3_quartile_in_caterory = leidiniaai_3_quartile_in_caterory.round(0)

leidiniaai_3_quartile_in_caterory =
leidiniaai_3_quartile_in_caterory.add_suffix('_scopus_quartile')

leidiniaai_3_quartile_in_caterory=leidiniaai_3_quartile_in_caterory.reset_index()

leidiniaai_3_quartile_in_caterory =
leidiniaai_3_quartile_in_caterory.drop(['0_scopus_quartile'], axis=1)

##### Scopus DataFrame

leidiniaai_scopus =
pd.merge(pd.merge(pd.merge(leidiniaai_3_SNIP,leidiniaai_3_CiteScore,on='final_index'),

```

```

leidiniai_3_SJR,on='final_index'),leidiniai_3_quartile_in_caterory,on='final_index')

#### 4. Žingsniai atkartojami IF rodikliams

leidiniai_4 = leidiniai_2[['final_index', 'IF', 'AIF', 'quartile_in_caterory_2',
'if_info_years']]

quartile_in_caterory_2 = leidiniai_4['quartile_in_caterory_2'].str.split('Q',
expand=True)

leidiniai_4 = leidiniai_4.drop(['quartile_in_caterory_2'], axis=1)

leidiniai_4['quartile_in_caterory_2'] =
quartile_in_caterory_2[1].fillna(0).astype(int)

##### IF

leidiniai_4_IF = leidiniai_4.pivot_table(index="final_index",
columns="if_info_years", values='IF')

leidiniai_4_IF = leidiniai_4_IF.add_suffix('_IF')

leidiniai_4_IF=leidiniai_4_IF.reset_index()

leidiniai_4_IF = leidiniai_4_IF.drop(['0_IF'], axis=1)

##### AIF

leidiniai_4_AIF = leidiniai_4.pivot_table(index="final_index",
columns="if_info_years", values='AIF')

leidiniai_4_AIF = leidiniai_4_AIF.add_suffix('_AIF')

leidiniai_4_AIF=leidiniai_4_AIF.reset_index()

leidiniai_4_AIF = leidiniai_4_AIF.drop(['0_AIF'], axis=1)

##### quartile_in_caterory

leidiniai_4_quartile_in_caterory = leidiniai_4.pivot_table(index="final_index",
columns="if_info_years", values='quartile_in_caterory_2')

leidiniai_4_quartile_in_caterory = leidiniai_4_quartile_in_caterory.round(0)

leidiniai_4_quartile_in_caterory =
leidiniai_4_quartile_in_caterory.add_suffix('_IF_quartile')

leidiniai_4_quartile_in_caterory=leidiniai_4_quartile_in_caterory.reset_index()

leidiniai_4_quartile_in_caterory =
leidiniai_4_quartile_in_caterory.drop(['0_IF_quartile'], axis=1)

##### IF DataBase

leidiniai_if = pd.merge(pd.merge(leidiniai_4_IF,leidiniai_4_AIF,on='final_index'),
                        leidiniai_4_quartile_in_caterory,on='final_index')

```

```

#### 5. Sujungiamos abi rodiklių lentelės į vieną ir pridedami likę rodikliai
leidiniai_rodikliai = pd.merge(leidiniai_scopus,leidiniai_if,on='final_index')
kiti_rodikliai = leidiniai_2[['final_index', 'InCites_JCR']]
kiti_rodikliai = kiti_rodikliai.drop_duplicates(subset=['final_index'])
leidiniai_final = pd.merge(leidiniai_rodikliai,kiti_rodikliai,on='final_index')
leidiniai_final['InCites_JCR'] = leidiniai_final['InCites_JCR'].replace([0], 'NA')
leidiniai_final = leidiniai_final.sort_values(by=['InCites_JCR'])
leidiniai_final

#### 6. Sudaromi 3 duomenų rinkiniai pagal mokslo šakas: SCIE, SSCI ir NA
NA = leidiniai_final[leidiniai_final['InCites_JCR'] == 'NA']
NA = NA.drop(['InCites_JCR', 'final_index'], axis=1)
SCIE = leidiniai_final[leidiniai_final['InCites_JCR'] == 'SCIE']
SCIE = SCIE.drop(['InCites_JCR', 'final_index'], axis=1)
SSCI = leidiniai_final[leidiniai_final['InCites_JCR'] == 'SSCI']
SSCI = SSCI.drop(['InCites_JCR', 'final_index'], axis=1)

### Klasterizavimas

#### Duomenų normalizavimas

scaler = MinMaxScaler()
scaled_data_NA = scaler.fit_transform(NA)
scaled_data_NA = pd.DataFrame(scaled_data_NA)
scaled_data_SCIE = scaler.fit_transform(SCIE)
scaled_data_SCIE = pd.DataFrame(scaled_data_SCIE)
scaled_data_SSCI = scaler.fit_transform(SSCI)
scaled_data_SSCI = pd.DataFrame(scaled_data_SSCI)

scaled_data = pd.concat([scaled_data_NA, scaled_data_SCIE, scaled_data_SSCI], axis=0,
ignore_index=True)

scaled_data = scaled_data.fillna(0)

test_data = scaled_data.iloc[[0]]

test_data

#### Sukuriamas Kmeans modelis

```

```

km = KMeans()

km.fit(scaled_data)

#### Gauto modelio rezultatai nebūtinai yra geriausi iš pirmojo karto ir svarbu
įsitikinti ar klasterių skaičius parinktas teisingai. Šiai užduočiai atlikti buvo
naudojami 2 metodai:
1. Sudarant klasterių kvadratinės paklaidos sumos mažėjimo ir klasterių tarpusavio
ryšio grafiką;
2. Apskaičiuojant silueto koeficientą.

#### Klasterių kvadratinės paklaidos sumos mažėjimo ir klasterių tarpusavio ryšio
grafikas

k_values = list(range(2,20))

inertias = []

for k in k_values:
    km = KMeans(n_clusters=k)
    km.fit(scaled_data)
    inertias.append(km.inertia_)

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
f, ax = plt.subplots(figsize=(10, 6.5))
ax.grid(False)
plt.plot(k_values, inertias, marker='.',color='skyblue')
plt.xticks(k_values)
plt.xlabel('Klasterių skaičius')
plt.ylabel('Kvadratinės paklaidos suma (SSE)');

#### Silueto koeficientas

klasteriai = list(range(2,21))

score = []

for i in klasteriai:
    km = KMeans(n_clusters=i, random_state=42)
    km.fit(scaled_data)
    s = round(silhouette_score(scaled_data, km.labels_, metric='euclidean'),3)
    score.append(s)

klasteriu_vertinimas = pd.DataFrame(
    {'Klasteriu_skaicius': klasteriai,
     'silhouette_score': score
    })

klasteriu_vertinimas

#### Iš gautų rezultatų, priimamas sprendimas, kad optimalus klasterių skaičius yra 7

#### Optimaliausias klasterių skaičius pritaikomas modeliui

km = KMeans(n_clusters=7, random_state=43)

```

```

km.fit(scaled_data)

#### Klasterių reikšmės pridedamos prie duomenų lentelės

leidiniai_final['clusters'] = km.labels_

leidiniai_final

list(leidiniai_final.columns)

data_viz = leidiniai_final[['2018_SNIP', '2018_CiteScore', '2018_SJR',
                             '2018_IF', '2018_AIF', 'clusters']]

data_viz['clusters'].unique()

sns.set_theme(style="ticks")

sns.pairplot(data_viz, hue="clusters", palette="tab10",
              plot_kws = {'alpha': 0.6, 's': 80, 'edgecolor': 'k'})

### Klasterių analizė

#### Klasteriai vizualizuojami naudojantis silhouette_visualizer

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
f, ax = plt.subplots(figsize=(8, 8))
ax.grid(False)
silhouette_visualizer(KMeans(7, random_state=43), scaled_data, show=True);

#### Gauti klasteriai pridedami prie pradinio duomenų rinkinio ir apibūdinami

klasteriu_dydis =
pd.DataFrame(leidiniai_final.clusters.value_counts()).sort_index().rename(columns =
{'clusters': 'count'})

klasteriai = leidiniai_final[['final_index', 'clusters']]

klasteriu_analize = pd.merge(leidiniai_2, klasteriai, on='final_index')

#### Analizuojami gautų klasterių rodikliai

vidurkiai_1 = klasteriu_analize.groupby('clusters').mean()

vidurkiai_1 = vidurkiai_1.drop(['scopus_sources', 'if_info_years', 'final_index'],
axis=1).reset_index()

vidurkiai = vidurkiai_1.join(klasteriu_dydis)

vidurkiai_lentele = vidurkiai.round(2).rename(columns = {'clusters': 'Klasteriai',
'count': 'Žurnalų skaičius'})

vidurkiai_lentele

#groupedvalues=vidurkiai.groupby('clusters').mean().reset_index()

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")
rs = np.random.RandomState(8)

```

```

# Set up the matplotlib figure
f, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(5, 1, figsize=(10,10), sharex=True)

# SNIP
x = vidurkiai['clusters']
y1 = vidurkiai['SNIP']
sns.barplot(x=x, y=y1, palette="hls", ax=ax1)
ax1.axhline(0, color="k", clip_on=False)
ax1.set_ylabel("SNIP")
ax1.get_xaxis().set_visible(False)
labels1 = vidurkiai_lentele['SNIP'].tolist()
for i in ax1.containers:
    ax1.bar_label(i, labels1)

# CiteScore
y2 = vidurkiai['CiteScore']
sns.barplot(x=x, y=y2, palette="hls", ax=ax2)
ax2.axhline(0, color="k", clip_on=False)
ax2.set_ylabel("CiteScore")
ax2.get_xaxis().set_visible(False)
labels2 = vidurkiai_lentele['CiteScore'].tolist()
for i in ax2.containers:
    ax2.bar_label(i, labels2)

# SJR
y3 = vidurkiai['SJR']
sns.barplot(x=x, y=y3, palette="hls", ax=ax3)
ax3.axhline(0, color="k", clip_on=False)
ax3.set_ylabel("SJR")
ax3.get_xaxis().set_visible(False)
labels3 = vidurkiai_lentele['SJR'].tolist()
for i in ax3.containers:
    ax3.bar_label(i, labels3)

# IF
y4 = vidurkiai['IF']
sns.barplot(x=x, y=y4, palette="hls", ax=ax4)
ax4.axhline(0, color="k", clip_on=False)
ax4.set_ylabel("IF")
ax4.get_xaxis().set_visible(False)
labels4 = vidurkiai_lentele['IF'].tolist()
for i in ax4.containers:
    ax4.bar_label(i, labels4)

# AIF
y5 = vidurkiai['AIF']
sns.barplot(x=x, y=y5, palette="hls", ax=ax5)
ax5.axhline(0, color="k", clip_on=False)
ax5.set_ylabel("AIF")
ax5.set_xlabel("Klasteriai")
labels5 = vidurkiai_lentele['AIF'].tolist()
for i in ax5.containers:
    ax5.bar_label(i, labels5)

# Finalize the plot
sns.despine(bottom=True)
plt.setp(f.axes, yticks=[])
plt.tight_layout(h_pad=2)

```

```

sns.set_theme(style="whitegrid", font_scale=1.5, font="Times New Roman")

fig, ax1 = plt.subplots(figsize=(12,6))
ax1.grid(False)
IF = sns.lineplot(data = vidurkliai['IF'], marker='o', sort = False, ax=ax1)
ax2 = ax1.twinx()
ax2.grid(False)
SNIP = sns.lineplot(data = vidurkliai['SNIP'], marker='o', sort = False, ax=ax1)

CiteScore = sns.lineplot(data = vidurkliai['CiteScore'], marker='o', sort = False,
ax=ax1)

SJR = sns.lineplot(data = vidurkliai['SJR'], marker='o', sort = False, ax=ax1)
AIF = sns.lineplot(data = vidurkliai['AIF'], marker='o', sort = False, ax=ax1)

sns.barplot(data = vidurkliai, x='clusters', y='count', alpha=0.5, ax=ax2,
palette="hls");

ax1.legend(['IF', 'SNIP', 'CiteScore', 'SJR', 'AIF'], loc=(1.1,0.2));

#### Atrenkami top 10 leidinių, kurie buvo klasterizuojami ir patikrinama, kokiam
klasteriui jie buvo priskirti

(Imama 17, o ne 10 leidinių, nes dalis iš daugiausiai minimų žurnalų nebuvo
klasterizuojami, nes neturėjo bibliografinių rodiklių)

top_17 = pd.DataFrame(leidiniai['final_index'].value_counts()).head(17)

top_17 = top_17.reset_index().rename(columns = {'final_index':'value_count',
'index':'final_index'})

top10 =
pd.merge(top_17,klasteriu_analize,on='final_index').drop_duplicates(subset='final_ind
ex', keep="last")

top10

```

SUTIKIMAS DĖL ASMENS DUOMENŲ NAUDOJIMO

2022-05-29

Šiuo sutikimu aš, Rūta Dubovikienė (toliau – Duomenų subjektas) sutinku, kad Vilniaus Gedimino technikos universitetas, juridinio asmens kodas 111950243, adresas Saulėtekio al. 11, LT-10223 Vilnius (toliau – Duomenų valdytojas), tvarkytų mano asmens duomenis kitų studentų mokymosi tikslu. T. y. tvarkytų (*pažymėkite tinkamą*):

- ☐ vardą, pavardę, bakalauro baigiamąjį darbą;
- ☐ bakalauro baigiamąjį darbą, nenurodant vardo, pavardės;
- ☒ vardą, pavardę, magistro baigiamąjį darbą;
- ☐ magistro baigiamąjį darbą nenurodant vardo, pavardės.

Šiuo tikslu tvarkomų asmens duomenų Duomenų valdytojas neperduos jokiems tretiesiems asmenims, studentams su baigiamaisiais darbais bus leidžiama susipažinti vidinėje informacinėje sistemoje. Duomenų subjekto asmens duomenys šiuo tikslu bus naudojami ne ilgiau nei 5 metai.

Šiuo sutikimu Duomenų subjektas patvirtina, kad yra supažindintas su šiomis teisėmis:

- Susipažinti su savo duomenimis ir kaip jie yra tvarkomi (teisė susipažinti);
- Reikalauti ištaisyti arba, atsižvelgiant į asmens duomenų tvarkymo tikslus papildyti asmens neišsamius asmens duomenis (teisė ištaisyti);
- Savo duomenis sunaikinti arba sustabdyti savo duomenų tvarkymo veiksmus (išskyrus saugojimą) (teisė sunaikinti ir teisė „būti pamirštam“);
- Reikalauti, kad asmens duomenų valdytojas apribotų asmens duomenų tvarkymą (teisė apriboti);
- Teisė į duomenų perkėlimą (teisė perkelti);
- Nesutikti, kad būtų tvarkomi asmens duomenys, kai šie duomenys tvarkomi ar ketinami tvarkyti kitais tikslais;
- Pateikti skundą Valstybinei duomenų apsaugos inspekcijai;

Duomenų subjektas turi teisę bet kuriuo metu atšaukti savo sutikimą.

*Rūta Dubovikienė
[Duomenų subjekto vardas, pavardė, parašas]*

Duomenų valdytojo rekvizitai:
Vilniaus Gedimino technikos universitetas
Juridinio asmens kodas: 111950243
Adresas: Saulėtekio al. 11, LT-10223 Vilnius
Tel. (8 5) 274 5030
Faks. (8 5) 270 0112
El. paštas: vgtu@vgtu.lt
PVM mokėtojo kodas: LT119502413
Duomenų apsaugos pareigūno tel. (8 5) 251 2191, el. paštas: dap@vgtu.lt