VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

EKONOMETRIJOS BAKALAURO STUDIJŲ PROGRAMA

Socialinio tinklo „Facebook" puslapių populiarumo vertinimo tyrimas, paremtas mašininio mokymosi metodais

Popularity assessment study of „Facebook“ pages based on machine learning techniques

Baigiamasis bakalauro darbas

Autorius: Dominykas Eirošius
VU el. p.: dominykas.eirosius@mif.stud.vu.lt

Darbo vadovė: Dr. Jolita Bernatavičienė

Vilnius
2022

Santrauka

Šiame darbe yra pristatomas socialinio tinklo „Facebook" puslapių populiarumo vertinimo tyrimas. Puslapių populiarumas buvo vertinamas analizuojant puslapių keliamo turinio strategijas. Turinio strategijos buvo analizuojamos pagal publikuojamų vaizdo įrašų trukmę, kuriamo turinio tematiką, paskelbimo savaitės dieną bei laiką. Analizė buvo atlikta tiriant tris „Facebook“ puslapius, kuriuose buvo patalpinta virš 8 tūkst. vaizdo įrašų. Tyrime buvo atliekamas temų modeliavimas, statistinių hipotezių tikrinimas bei vaizdo įrašų klasterizavimas. Temų modeliavimas atliktas pasinaudojant latentiniu Dirichlė paskirstymo algoritmu, iškeltoms hipotezėms patikrinti buvo naudojami Kruskal-Wallis bei Dunn testai, o vaizdo įrašų klasterizavimas atliekamas K-vidurkių ir hierarchiniu klasterizavimo metodu. Tyrimo metu buvo sudarytos 5 temos, kurios leido detaliau pažvelgti į keliamą turinį. Išanalizavus duomenis buvo ištirtos keliamo turinio strategijos, patikrintos statistinės hipotezės ir nustatyti bruožai, būdingi puslapių keliamo turinio strategijoms.

Raktiniai žodžiai: socialiniai tinklai; „Facebook“ puslapis; temų modeliavimas; latentinis Dirichlė paskirstymas; keliamo turinio strategija; Kruskal-Wallis ir Dunn testai; klasterizavimas K-vidurkių metodu; hierarchinis klasterizavimas.

Summary

This paper presents a study of the popularity of Facebook pages. The popularity was assessed by analyzing the content strategies of the pages. Content strategies were analyzed based on the duration of the published videos, the subject matter of the content being created, publication time and day of the week. The analysis was conducted by investigating three Facebook pages that published over 8,000 videos. Topic modeling, statistical hypothesis testing and video clustering were performed in this study. Topic modeling was performed using Latent Dirichlet Allocation, the Kruskal-Wallis and Dunn tests were used to test the hypotheses and videos clustering was performed using K-means and hierarchical clustering methods. Carrying out topic modeling, 5 topics were formed, which allowed to better understand published content. After analyzing the data, the strategies of the published content were examined, the statistical hypotheses were tested and the strategies features of the published content of the pages were identified.

Keywords: Facebook page; topic modeling; Latent Dirichlet Allocation; content strategy; Kruskal-Wallis and Dunn tests; K-means clustering; hierarchical clustering.

Turinys

1 Įvadas

Žmonės vis dažniau žvelgia į socialinius tinklus kaip į neatsiejamą savo kasdienio gyvenimo dalį ir vis dažniau perkelia savo bendravimą su artimaisiais ar draugais į virtualias socialinių tinklų platformas, tokias kaip „Facebook“, „Instagram“ ar „Twitter“. Toks žmonių elgesio pasikeitimas indikuoja socialinių tinklų didėjančią svarbą, kuri dar labiau išaugo per pastaruosius kelerius metus, kai pasaulį sukrėtė pandemija. Šiuo laikotarpiu socialiniai tinklai buvo viena iš pagrindinių priemonių gyventojams palaikyti ryšį su draugais ir šeima bei kovoti su izoliacijos sukeltu nuoboduliu ir vienatve, kuri ignoruojama iššaukdavo žmonėms ir kitas gilesnias psichologines problemas [1]. Socialinių tinklų sąvoką galime įsivaizduoti kaip kompiuterinę technologiją, kuri suteikia žmonėms galimybes lengvai komunikuoti tarpuysavyje, dalintis mintimis ir kita norima informacija. Nors pirminis šios technologijos atsiradimo tikslas buvo palengvinti bendravimą su draugais ir šeima, vis dėlto vėliau tai tapo puiki priemonė vystyti naujas įmonėms ar plėtoti jau esamus verslus. Šiuo metu gyventojai gali rinktis iš labai plataus socialinių tinklų sąrašo, kurie tarpusavyje skiriasi savo paskirtimi ir funkcionalumu.

Vienas populiariausių socialinių tinklų visame pasaulyje yra „Facebook“. Tai yra socialinis tinklas, kurio pagrindinis tikslas sujungti žmones ir leisti jiems komunikuoti bei dalintis savo mintimis. Remiantis naujausiais duomenimis, šiuo metu „Facebook“ visame pasaulyje naudojasi beveik trys milijardai aktyvių vartotojų. Nors šios technologijos pagrindinis tikslas leisti žmonėms lengvai susisiekti su kitais, ši platfoma suteikia ir kitokias galimybes: publikuoti ir dalintis vaizdo įrašais, nuotraukomis, susirasti naujų pažinčių, taip pat kurti „Facebook“ puslapius. „Facebook“ puslapis yra apibūdinamas kaip profilis, kurį gali susikurti įmonės, organizacijos ar individualūs asmenys. Šiuose puslapiuose gali būti skelbiama įvairi informacija, kuri yra prieinama ir matoma visiems socialinės platformos vartotojams [2]. „Facebook“ platformoje egzistuoja milijonai įvairių puslapių, kurie varžosi tarpusavyje dėl didesnio vartotojų dėmesio ir populiarumo. Apsilankančios auditorijos ir populiarumo didinimui yra pasitelkiamos įvairios priemonės, viena iš jų - keliamo turinio strategija. Keliamo turinio strategijomis siekiama optimizuoti publikuojamo turinio žiūrimumą ir socialinio tinklo vartotojų įsitraukimą. Todėl šiame darbe yra norima atlikti tryjų „Facebook“ puslapių populiarumo vertinimus analizuojant puslapių keliamo turinio strategijas. Šiam tikslui įgyvendinti buvo išsikelti uždaviniai, kurie aprašyti kitame skyrelyje.

Atlikto darbo struktūra yra sudaryta iš dviejų pagrindinių dalių: teorinės ir praktinės. Teorinėje dalyje yra pateikiamas įvadas, susipažįstama su literatūros apžvalga bei pristatomi tyrime naudojami duomenys. Literatūros apžvalgoje yra aptariama socialinių tinklų svarba, samprata bei jos rūšys, plačiau aprašoma tiriamoji aplinka, keliamo turinio strategijos sąvoka, taip pat pristatomi darbe taikyti temų modeliavimo, klasterizavimo bei statistinių hipotezių tikrinimo metodai. Praktinėje dalyje atliekama duomenų statistinė analizė, kurioje pateikiamos pradinės įžvalgos apie nagrinėjamus duomenis. Toliau pereinama prie temų modeliavimo, kurio metu taikomas latentinio Dirichlė paskirstymo algoritmas ir sudaromos 5 vaizdo įrašų temos. Tuomet išanalizuojamos tiriamų puslapių kekiamo turinio strategijos bei patikrinamos statistinės hipotezės. Galiausiai, kiekvieno kanalo vaizdo įrašai yra klasterizuojami pagal požymių rinkinius. Klasterizavimui atlikti naudojamas hierarchinis bei K-vidurkių klasterizavimo metodas. Darbo pabaigoje pateiktiamos išvados, cituojama literatūra, priedai bei naudotas programinis kodas.

1.1 Darbo tikslas ir uždaviniai

Pagrindinis darbo tikslas yra atlikti socialinio tinklo „Facebook“ puslapių populiarumo vertinimą analizuojant keliamo turinio strategijas.

Norint pasiekti šį tikslą, buvo išsikelti šie uždaviniai:

1. Susipažinti su problema ir tiriamąją aplinka;
2. Susipažinti su tyrimo srityje naudojamais metodais, kurie padės pasiekti iškeltą tikslą ;
3. Susikomplektuoti duomenų imtį tyrimui;
4. Atlikti pirminę duomenų analizę ir aprašomąsias statistikas;
5. Atlikti temų modeliavimą;
6. Išanalizuoti „Facebook“ puslapiuose keliamo turinio strategijas;
7. Sudaryti kiekvieno nagrinėjamo puslapio vaizdo įrašų klasterizaciją ir įvertinti klasteriams būdingus bruožus.

Darbo tematika paarašytas straipsnis ir perskaitytas konferencijoje:

- Eirošius, D. (2022). Keliamo turinio strategijos analizė socialinio tinklo „Facebook “puslapiuose. Vilnius University Open Series, 15-24.

2 Literatūros apžvalga

2.1 Socialinių tinklų svarba

Tobulėjant šiuolaikiniam pasauliui, socialiniai tinklai vaidina vis svarbesnį vaidmenį mūsų gyvenimuose. Socialiniai tinklai yra interneto paremta technologija, leidžianti jų vartotojams dalintis įvairiausiu turiniu bei bendrauti tarpusavyje. Nuo šios technologijos atsiradimo ženkliai pasikeitė daugumos žmonių kasdienybė - kaip yra bendraujama su kitais, kaip sužinoma, kokie svarbūs įvykiai įvyko ar vyksta visame pasaulyje. Yra daug būdų, kaip socialinė žiniasklaida gali būti naudinga visuomenei, pavyzdžiui, greitai surasti reikiamos informacijos, susipažinti su naujais žmonėmis, pasidalinti naujienomis, ir daug daugiau. Atliktas „New York Times Consumer Insight Group“ tyrimas atskleidė pagrindinius motyvus, kuriais tyrimo dalyviai rėmėsi dalindamiesi informacija socialinių tinklų erdvėje [3]. Šie motyvai apima norą dalintis vertingu ir įdomiu turinu su kitais, išreikšti save, ieškoti naujų pažinčių, dalintis maloniomis ir skaudžiomis patirtimis bei diskutuoti apie patinkančius dalykus, prekės ženklus. Dėl didelio susidomėjimo ir galimybių, socialiniai tinklai sparčiai populiarėja ir jų naudojimasis tampa neatsiejama žmonių gyvenimo dalis. Be išvardintų internetinės technologijos naudojimosi priežasčių, taip pat socialiniai tinklai gali būti pasitelkti formuojant politinius judėjimus, pasaulio kultūrą ir švietimą, kuriant verslus bei inovacijas. Plačiau aptarsime kiekvieną iš šių sričių:

- **Socialinių tinklų įtaka politikos srityje**. Socialinių tinklų atsiradimas įvairiais būdais paskatino informacijos sklaidą visomis temomis, įskaitant ir politines naujienas. Lyginant su kitomis žiniasklaidos priemonėmis, per pastarąjį dešimtmetį socialinių tinklų įtaka politinėms kampanijoms nepaprastai išaugo. „Pew Research Center“ atliktas tyrimas parodė, jog vidutiniškai vienas iš penkių, vyresnių nei 18 m. amžiaus JAV gyvenančių žmonių, apie politines naujienas pirmiausia sužino socialiniuose tinkluose [4]. Tai tik pabrėžia, jog ši technologija tampa viena populiariausių priemonių, kaip politikai gali pasiekti savo rinkėjus bei sustiprinti savo kandidatūros sėkmę. Vienas iš tokių pavyzdžių yra 2017 m. Donald Trump sėkmė JAV prezidento rinkimuose. „The New York Times“ teigimu, Donald Trump laimėjimas yra bene ryškiausias pavyzdys, kokią didelę reikšmę socialiniai tinklai gali turėti formoujant ir telkiant savo rinkėjus bei taip užtikrinant savo sėkmę [5]. Šią mintį paantrino ir minėtas buvęs JAV prezidentas, teigdamas, jog nebūtų pavykę laimėti rinkimų, jei nebutų naudojęsis socialiniais tinklais [6].

- **Socialinių tinklų įtaka visuomenei**. Daugiau nei ketvirtadalis pasaulio gyventojų naudojasi ar turi savo „Facebook“ paskyrą [7]. Tuo tarpu JAV, beveik 80 proc. visų interneto vartotojų naudojasi šia platforma. Kadangi socialiniai tinklai skatina žmonių komunikaciją ir sąveikavimą tarpusavyje, vartotojų kiekis kiekvienais metais vis auga, kas dar labiau skatina jų populiarumą ir naudojimąsi. Socialiniai tinklai sukuria galimybę burtis žmonėms į grupes ir dalintis įvairiausia informaciją – kurti juokingą ir kūrybišką turinį, dalintis istorijomis bei patirtimis, skleisti idėjas ir diskutuoti apie socialines, etines, aplinkos ir kitas problemas su šimtais tūkstančių ar milijonų vartotojų. Kaip jau buvo paminėta, socialiniai tinklai taip pat suteikia galimybes asmenims palaikyti ryšį su šeima ir draugais, su kuriais kitaip negalėtų susisiekti.

- **Socialinių tinklų įtaka komercijai**. Šiais laikais neįprasta sutikti organizacijos, kuri nebandytų pasiekti potencialių ar esamų savo klientų naudojantis viena ar kita socialinių tinklų platforma.

Šios technologijos populiarumas suteikė naujas galimybes komercijos tobulinimui. Įmonės pamatė, kaip svarbu naudoti socialinę žiniasklaidą, norint išplėsti klientų ratą ir didinti gaunamas pajamas. Taip pat suprato, kad ši priemonė padeda gauti naujų įžvalgų, padedančių dar labiau tobulinti savo produktus [8]. Ir šios technologijos naudojimas yra svarbus ne tik elektroninę prekybą vystančioms įmonėms, bet ir tradiciniams verslams. Pasinaudodami socialinėmis platformomis įmonės gali kurti, reklamuodami ir didinti savo prekės ženklo žinomumą. Kuo daugiau klientų sužino ir kalba apie įmonę, tuo vertingesnis tampa prekės ženklo autoritetas. O tai lemia padidėjusius kuriamų produktų pardavimo rezultatus bei aukštesnį reitingą internetinės paieškos sistemose.

- **Socialinių tinklų įtaka darbo rinkai**. Internetinė technologija ženkliai palengvino potencialių darbuotojų ar naujų darbo vietų paieškas. Socialiniai tinklai, tokie kaip „LinkedIn“, yra svarbi priemonė visiems, norintiems išsiskirti savo profesijoje ar surasti naują darbo vietą. Šioje platformoje vartotojai gali susikurti savo paskyras, kuriose apibūdina jų kompetencijas ir žinias, aprašo sukauptą darbo patirtį ir taip reklamuoja save potencialiems darbdaviams. Remiantis apklausos rezultatais, net 90 proc. darbdavių, ieškodami ir vertindami kandidatus į skelbiamas darbo pozicijas, vertina socialinius tinklus kaip svarbų veiksnį [9]. Taip pat 2018 m. „CareerBuilder“ atliktas tyrimas atskleidė, jog net 70 proc. darbdavių naudojasi socialinių tinklų svetainėmis ieškant kandidatų į laisvas darbo pocizijas [10].

2.2 Socialinių tinklų samprata

Socialinių tinklų apibrėžimų literatūroje yra suformuluota labai daug. Vieni autoriai ar šaltiniai ją apibūdina gana abstrakčiai, kiti – išsamiau, išskirdami ir tam tikrus šių tinklų požymius. Visuotinėje lietuvių enciklopedijoje, socialiniai tinklai apibrėžiami kaip „iš pavienių individų, jų grupių, organizacijų sudarytos socialinių ryšių sistemos“ [11]. Šaltinio teigimu, socialinių tinklų vartotojai tarpusavyje yra artimai susiję draugystės, giminystės ar kitais tarpasmeniniais santykiais bei ekonominiais, socialiniais ar politiniais ryšiais. Labai panašiai socialinius tinklus apibrėžia ir „Merriam-Webster“ žodynas, įvardindamas juos kaip internetinę svetainę, kurioje žmonės gali palaikyti bei kurti tarpasmeninius santykius [12]. Tuo tarpu, „Cambridge Dictionary“ socialinius tinklus įvardina kaip internetinių paslaugų ir svetainių naudojimąsi, bendraujant su kitais asmenimis bei susirandant naujų draugų [13].

Kompiuterinių technologijų pagalba socialiniai tinklai apima asmeninių, taip pat ir verslo santykių kūrimą bei jų puoselėjimą. Tai yra įgyvendinama naudojant socialinių tinklų svetaines, dar kitaip vadinamas socialinių tinklų platformomis. Šios platformos leidžia vartotojams susisiekti vieniems su kitais, užmegzti ryšius bei dalintis informacija, idėjomis ir žinutėmis.

2.3 Socialinių tinklų rūšys

Socialinių tinklų klasifikavimas padeda asmenims pasirinkti tinklus, kuriems jie norėtų priklausyti ir tapti jų naudotojais. Nepaisant to, kad vartotojai turi įvairių pomėgių ir poreikių, vis dėlto kūrėjai stengiasi optimizuoti socialinių tinklų funkcijas ir teikiamas galimybes. Tokiu būdu kūrėjai tikisi pritraukti dar daugiau tikslingos auditorijos, kuri prisijungdama prie kitų vartotojų galėtų gauti didžiausią naudą. Interneto paieškose galima rasti įvairių socialinių tinklų klasifikacijų, tačiau pagal funkcionalumą, socialiniai tinklai neretai yra skirstomi į šias rūšis:

- **Socialinius tinklus, kurie orientuoti į vartotojų tarpusavio komunikaciją ir kitokį sąveikavimą**. Tokių socialinių tinklų pagrindinis tikslas - sujungti žmones ir leisti jiems komunikuoti bei dalintis savo mintimis. Jos dažniausiai naudojamos palaikyti ryšį su artimaisiais, draugais, kolegomis ar klientais. Vienos populiariausių tokio tipo platformų: „Facebook“, „Twitter“ ir „LinkedIn“. Dauguma tokio tipo tinklų leidžia žmonėms lengvai susiekti su kitais asmeninėmis žinutėmis bei vaizdo pokalbiais. Šios funkcijos yra ypač patrauklios asmenims, kurie yra dideliu atstumu atskirti nuo artimųjų. Jos leidžia bet kada susisiekti su šeimos nariais ir bendrauti tarpusavyje juos matant. Be paminėtų funkcijų, šios platfomos suteikia ir kitokias galimybes: publikuoti ir dalintis vaizdo įrašais, nuotraukomis, išreikšti save pasidalinant savo mintis viešose erdvėse. Taip pat vartotojai gali susirasti naujų pažinčių, kurie turi sutampančių interesų. Asmenys tokiose platformose susipažįsta vieni su kitais per įvairias viešas grupes bei diskusijas.

- **Socialinius tinklus, kurie orientuoti į nuotraukų publikavimą**. Vienos didžiausių ir populiariausių platformų, kurios specializuojasi dalijimosi nuotraukomis srityje, yra „Instagram“ ir „Pinterest“. Tokio tipo platformos dažnai turi aiškų ir lengvai suprantamą naudojimosi sistemą, todėl nauji vartotojai greitai pritampa. Pavyzdžiui, „Instagram“ naudotojai gali be didesnių sunkumų greitai perprasti šios technologijos veikimą. Šioje platformoje didzioji dalis publikuojamo turinio yra nuotraukos, tačiau galima kelti ir trumpus vaizdo įrašūs. Vartotojai taip pat gali skelbti tiesioginius vaizdo įrašus arba kurti vadinamas „Instagram“ istorijas, kurios platformoje rodomos laikinai ir yra panaikinamos po vienos paros nuo paskelbimo. Kaip ir įprasta socialinių tinklų platformoms, asmenys gali bendrauti tarpusavyje palikdami komentarus po publikuotos nuotraukos arba susirašinėdami asmeninėmis žinutėmis. Nors „Instagram“ turi platų spektrą galimybių ir funckijų, vis dėlto šios platformos pranašumas yra publikuojamuose nuotraukose. Technologijos sukurtas veikimas leidžia žmonėms įkelti paveiksliukus per kelias minutes, taip pat yra įdiegtos ir plačios nuotraukų redagavimo galimybės.

- **Socialinius tinklus, kurie orientuoti į vaizdo įrašų kūrimą ir publikavimą**. „YouTube“ – bene didžiausia vaizdo įrašų dalijimosi svetainė, sukurta 2005 m. Šioje svetainėje žmonės per dieną peržiūri daugiau nei 2 milijardus vaizdo įrašų visame pasaulyje, o platus platformoje esantis temų spektras pavertė dalijimąsi vaizdo įrašais viena svarbiausių interneto kultūros bruožų. Tokių platformų pranašumas - lengvai surandami vaizdo įrašai norima tema, paprastas vaizdo įrašų publikavimas bei auditorijų pasiekiamumas iš viso pasaulio. Taip pat per porą minučių galima surasti turinio bet kokiomis temomis, nesvarbu, ar tai būtų susiję su laisvo laiko praleidimu, ar norėjimu išmokti bei įgyti naujų žinių.

- **Socialinius tinklus, kurie yra interaktyvūs**. Tokie socialiniai tinklai kaip „Snapchat“ ir „TikTok“ leidžia vartotojams dalintis nuotraukomis ir vaizdo įrašais, jos taip pat turi daugybę unikalių interaktyvių ir naujoviškų funkcijų. Tokios platformos dažniausiai leidžia vartotojams kurti turinį pasinaudojant papildytos realybės (angl. *Augmented reality (AR)*) bei virtualios realybės (angl. *Virtual reality (AR)*) sukurtais efektais, uždėti muzikinį foną bei naudoti interaktyvius žaidimus. Papildyta realybė - tai pasaulio atvaizdavimo technologija, kuri papildo žmogaus matomą vaizdą virtualiais kompiuteriu sugeneruojamais elementais ar vaizdais. Kitaip tariant, šios technologijos pagalba realaus fizinio pasaulio vaizdas yra modifikuojamas [14]. Tuo tarpu virtuali realybė

– programinės įrangos pagalba sukuriama aplinka, kuri yra dirbtinė ir pateikiama vartotojui kaip realybė [15]. Taip pat tokiose platformose publikuojamas turinys dažniausiai yra trumpos trukmės.

- **Socialinius tinklus, kurie orientuoti į informacijos sklaidą ir diskusijas**. Vienas populiariausių tokio tipo socialinių platformų - „Reddit“. Nors tokio tipo platformos yra priskiriamos socialiniams tinklams, tačiau jos išsiskiria iš kitų socialinių erdvių. Dažniausiai jų veikimas ir naudojimąsis yra pagrįstas bendruomenių arba forumų kūrimais ir anonimiškumu. Kaip pavyzdį paimkime jau minėtą „Reddit“ platformą. Čia žmonės jungiasi į bendruomenes, kurios vadinamos subreditais (angl. *subreddits*). Kitaip tariant, subreditai pažymi tam tikrą temą, kuria naudotojai susirenka diskutuoti. Vartotojai, susikūrę anonimines anketas, gali laisvai išsakyti savo nuomones ir dalintis patirtimis bei įžvalgomis. Taigi, dėl savo ypatybės leisti vartotojams išlikti anonimiškiems, tai yra forumas su socialinių tinklų elementais.

2.4 Socialinis tinklas „Facebook“

Šiame tyrime bus nagrinėjami socialinio tinklo „Facebook“ puslapiai. Kaip jau įvardinome praeitame skyriuje, „Facebook“ yra socialinis tinklas, orientuotas į vartotojų tarpusavio komunikaciją. Remiantis vienais naujausių duomenų, 2022 m. pirmąjį ketvirtį aktyvių „Facebook“ vartotojų skaičius visame pasaulyje siekė beveik tris milijardus [16]. 2004 m. šis socialinis tinklas buvo sukurtas Mark Zuckerberg kartu su Edward Saverin [17]. Nors technologijos gyvavimo pradžioje ji buvo naudojama kaip Harvardo universiteto socialinis tinklas, 2006 m. sparčiai išpopuliarėjo ir dabar yra žinomas kaip populiariausias ir daugiausiai aktyvių vartotojų turintis socialinis tinklas [18].

„Facebook“ yra nemokamas ir lengvai prieinamas visiems gyventojams, turintiems internetą ir išmanųjį įrenginį. Net ir mažai technologinių sugebėjimų turintys asmenys gali greitai perprasti platformos veikimą ir visapusiškai įsitraukti į jos naudojimąsi. Nors iš pradžių šio socialinio tinklo tikslas buvo padėti žmonėms palaikyti santykius su artimaisiais ar draugais, tačiau jis greitai tapo pamėgtas verslo, kuris galėjo sparčiau vystyti savo veiklą sukuriant didesnį prekės ženklo žinomumą ir klientų ratą.

Vienos pagrindinių funkcijų, leidusių socialiniui tinklui pasiekti tokį populiarumą, yra:

- galimybė vartotojams kurti savo draugų ratą bei pasirinkti turinį, kokį labiausiai norėtų matyti apsilankius platformoje.
- galimybė publikuoti nuotraukas, vaizdo įrašus, kuriuos gali pamatyti kiti „Facebook“ vartotojai.
- suteikiama naudotojams galimybė komentuoti kitų vartotojų keliamą turinį, bendrauti su kitais bei dalintis įvairia informacija.
- yra suteikiamos sąlygos verslams ar individams kurti viešus puslapius, taip suteikiant galimybę naudoti socialinį tinklą kaip rinkodaros priemonę ar .
- suteikiamos galimybės vartotojams, verslam bei įvairiems kūrėjams gauti pajamas už publikuojamą orginalų turinį.

- suteikiama galimybė vartotojams transliuoti tiesioginius vaizdo įrašus ir bendrauti su kitais naudotojais realiu laiku.

2.5 Keliamo turinio strategija

Socialinių tinklų kontekste, turinys apima ne tik skelbiamus rašytinius pranešimus, bet ir naudojamas nuotraukas bei vaizdo įrašus. Paskelbto pranešimo, nuotraukos ar įrašo populiarumą nulemia sulaukiamas peržiūrų skaičius bei varotojų sąveikavimas (angl. *Engagement*) su paskelbtu turiniu. Vartotojų sąveikavimas turi ypač didžiulę įtaką turinio populiarumui. Pavyzdžiui, jei skelbiamas turinys sulaukia daug vartotojų pasidalinimų, tai jis pasieks platesnį „Facebook“ vartotojų ratą, kuris taip pat galės peržiūrėti paskelbtą turinį. Panašią įtaką populiarumui turi ir komentarai bei mėgti mygtukų paspaudimai. Kai socialinio tinklo vartotojai intensyviai reaguoja bei komentuoja po paskelbtu turniu, socialinės platformos veikimo sistemos yra informuojamos, jog šis turinys yra įdomus, todėl „Facebook“ algoritmas pats pradeda skatinti paskelbto turinio viešinimą [19]. Turinio viešinimas nulemia tai, jog daugiau vartotojų pamato turinį ir taip jis tampa vis populiaresnis. Tačiau norint pasiekti daugiau peržiūrų ir varotojų sąveikavimų, kūrėjai turi išspręsti uždavinį, koks keliamas turinys yra geriausias, norint sulaukti didesnio žiūrovų įsitraukimo ir populiarumo. Dėl šios priežasties kūrėjai pasitelkia įvairias keliamo turinio strategijas, kurios padeda pasiekti didesnį populiarumą.

Literatūroje turinio strategijos apibrėžimų galima surasti įvairių. Tačiau dažnai turinio strategija apibūdinama kaip turinio publikavimo planavimas, kūrimas pasirinktomis tematikomis bei kitoks turinio valdymas [20]. Dar turinio strategiją galime apibrėžti kaip informatyvaus, tinkamo bei naudingo turinio sukūrimo, publikavimo ir valdymo planavimo praktiką [21].

Kiekvieno kūrėjo turinio strategija gali būti skirtinga, ją taip pat gali sudaryti skirtingas kiekis kintamųjų. Tačiau šiame darbe turinio strategija bus analizuojama pagal publikuojamų vaizdo įrašų trukmę, kuriamo turinio tematiką, paskelbimo savaitės dieną bei laiką.

2.6 Temų modeliavimas

Temų modeliavimas yra vienas iš galingiausių teksto apdorojimo metodų, skirtų latentiniams duomenims aptikti bei tekstinių dokumentų ir kitų duomenų ryšiams rasti. Latentinis kintamasis įvardinamas kaip kintamasis, kurio negalime tiesiogiai stebėti ar kitaip išmatuoti. Tačiau tokių kintamųjų egzistavimas aptinkamas iš jų turimo poveikio stebimiems kintamiesiems. Mokslininkai yra paskelbę daugybę įvairių temų modeliavimo straipsnių srityse, tokiose kaip programinės įrangos inžinerija, politikos mokslai, kalbotyra bei socialiniai tinklai. Temų modeliavimo metodų yra įvairių, tačiau latentinis Dirichlė paskirstymas yra vienas populiariausių naudojamų algoritmų, leidžiančių išskirti iš žodžių rinkinių pagrindines temas be papildomos žmogaus intervencijos. Šio metodo taikymas yra platus, tačiau populiariausios sritys yra natūralios kalbos apdorojimas ir rekomendacijų sistemos. 2003 m. latentinis Dirichlė paskirstymą (LDA) išvystė ir sukūrė David Blei su kitais mokslininkais [22]. Jie didžiausią dėmesį sutelkė į tai, kad dokumentai galėtų turėti ir būti sudaryti iš įvairių temų. Nuo tada LDA sparčiai išpopuliarėjo ir tapo standartiniu temų modeliavimo įrankiu, kuriuo naudojosi daugybė tyrinėtojų bei kitų mokslininkų. Vėliau LDA buvo ištobulintas ir išplėtotas įvairioms sritims, ypač socialinių tinklų ir socialinės žiniasklaidos srityse. 2008 m. David Blei ir Jon Mcauliffe pasiūlė patobulintą algoritmą,

kaip prižiūrimą LDA (angl. *supervised LDA*), kuris buvo statistinis dokumentų modelis [23]. Šiame modelyje buvo pritaikytas ir didžiausios tikimybės parametrų įvertinimas. Dar vieną įdomų šios srities tyrimą atliko Li [24]. Jis sujungė LDA ir „Girvan-Newman“ aptikimo algoritmą ir pateikė naują modelį (TTR-LDA). Šis algoritmas buvo universalus ir pritaikytas skirtingiems duomenų rinkiniams. 2012 m. Daud savo tyrime pristatė temų modeliavimo algoritmą, pavadintą „Temoral-Author-Topic (TAT)“ [25]. Šis modelis galėjo simuliuoti modelio tekstą, tyrėjus ir tų tyrimų atlikimo laiką. Šis algoritmas naudojo semantiniu pagrindu esančius žodžius tarp tyrimo autorių ir jų darbų. Modelis galėjo rasti su tom pačiom temom susijusius tyrėjus skirtingu laikotarpiu, taip pat galėjo parodyti autorių interesų kaitą laikui bėgant.

Socialiniai tinklai yra puikus žinių atradimo ir vartotojų elgesio analizės šaltinis. Pavyzdžiui, „Twitter“ yra vienas populiariausių socialinių tinklų, kurio įvertinimas ir analizė gali būti labai efektyvi analizuojant vartotojų elgesį. Dėl šios priežasties per neilgą laikotarį mokslininkai pasiūlė daugybę LDA metodų, kaip analizuoti vartotojų skelbiamus įrašus „Twitter“ platformoje. 2013 m. Lim pristatė „Twitter“ skirtą temų modeliavimo algoritmą [26]. Šis modelis naudojo hierarchinius Puasono-Dirichlė procesus (PDP) teksto modeliavimui ir Gauso proceso atsitiktinių funkcijų modelį socialinių tinklų modeliavimui. 2015 m., viename iš naujesnių šios srities tyrimų, Ostrowski sukūrė naują modelį, kurio veikimas rėmėsi klasifikavimu ir temų identifikavimu, kai jis taikomas surūšiuotam socialinio tinklo „Twitter" žinučių rinkiniui [27]. Tuo tarpu Weng su kitais mokslininkai sutelkė dėmesį į įtakingų „Twitter" vartotojų atpažinimą ir pasiūlė vartotojų įvertinimo metodą, pagrįstą PageRank algoritmu [28]. Šis metodas buvo paremtas LDA modeliu ir buvo skirtas rasti latentinės temos informaciją iš didžiulių dokumentų rinkinių.

LDA yra įvardinamas kaip neprižiūrėtas mašininio mokymosi (angl. *Machine Learning*) algoritmas. Mašininis mokymasis – tai kompiuterių programavimas, kurio metu naudojant duomenis ir ankstesnę partirį siekiama optimizuoti našumo kriterijus [29]. Mašininio mokymosi algoritmai dažniausiai yra skirstomi į keturias pagrindines kategorijas: prižiūrėtas mokymasis, neprižiūrėtas mokymasis, pusiau prižiūrėtas mokymasis ir skatinamasis mokymasis [30]. Taigi, šis neprižiūrėtas mašininio mokymosi algoritmas automatiškai suformuoja temas, kurioms priklauso tam tikri žodžiai. Kitaip tariant, tai yra temų, kurios sudarytos iš žodžių su tam tikromis tikimybėmis, rinkinys. Taigi, įvardinto metodo tikslas – iš žodžių ir dokumentų sukurti tiesiogiai nestebimą (latentinę) temą. Iš struktūrizuoto tekstų rinkinio (angl. *corpus*) LDA pirmiausia atkuria esančius dokumentus. To veikimas yra paremtas iš žodžių ir dokumentų sudarytų temų svarba [22].

Prieš nurodydami LDA specifikacijas, apibrėžkime sąvokas:

- Pagrindinis diskrečių duomenų vienetas yra žodis $\{1, .., V\}$, kuris įvardinamas kaip svarbiausias reikšminis kalbos vienetas. Žodžiai yra vaizduojami kaip vientiniai vektoriai, o v-asis žodis yra vaizduojamas vektoriumi w, kur $w^v = 1$ ir $w^u = 0$, kai $v \neq u$.
- Dokumentas apibrėžiamas kaip N žodžių seka $\mathbf{w} = (w_1, w_2, ..., w_N)$, kur w_n yra n-tasis sekos žodis.
- Struktūrizuotas teksto rinkinys (angl. *corpus*) yra M dokumentų kolekcija, žymima $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$.

Dirichlė skirstinys. LDA modelio kontekste, Dirichlė skirstinys yra labai svarbus šio modelio elementas [31]. Taip pat šis skirstnžinys yra priklausantis eksponentinų skirstinių šeimai. Tegul k-

dimensinis Dirichlė skirstinio atsitiktinis dydis θ yra iš $(k-1)$ simplekso. Tada vektorius θ yra iš $(k-1)$ simplekso, jeigu dydis išpildo šiuos reikalavimus:

- $\theta_i \geq 1$, kur $i \in \{1, ..., k\}$,
- $\sum_{i=1}^{k} \theta_i = 1$

Taigi, k-dimensiniam simpleksui skirstinio tankis yra apibrėžiamas kaip:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} ... \theta_k^{\alpha_k - 1},$$

kur α pažymi k ilgio teigiamų reikšmių vektorių, o $\Gamma(x)$ - Gamą funkciją, kuri išreiškiama:

$$\Gamma(x) = \int_0^{\infty} e^{-t} tkx - 1 \; dt$$

Multinominis skirstinys. Šis skirstinys literatūroje yra įvardinamas kaip binominio skirstinio apibendrinimas [32]. Atliekant n nepriklausomų bandymų, kiekvieno bandymo baigties rezultatas gali reikšti kokio nors k klasės įvykio pasirodymą. Čia visos klasės turi atskirą fiksuotą pasirodymo tikimybę, o bendra visų klasių tikimybių suma yra lygi vienetui. Multinominis skirstinys apibūdina tam tikrų klasių pasirodymų kombinacijos tikimybę:

$$p(x_1, .., x_k; p_1, ..., p_k) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i + 1)} \prod_{i=1}^{k} p_i^{x_i},$$

kur $i \in \{1, ..., k\}$ reiškia nagrinėjamos klasės indeksą, x_i atspindi klasės pasirodymo skaičių, n parodo atliktų bandymų kiekį, p_i pažymi klasės pasirodymo tikimybę, k nurodo bendrą klasių skaičių.

Modelis. LDA generavimo proceso specifikacijas kiekvienam dokumentui **w**, esančiame struktūrizuotame teksto rinkinyje D, galime užrašyti taip:

1. $N \sim Poisson(\xi)$
2. $\theta \sim Dir(\alpha)$ - dokumento temos skirstinys.
3. Kiekvienam žodžiui w_n iš N:
   - Pasirenkama tema $z_n \sim Multinomial(\theta)$.
   - Pasirenkamas žodis w_n iš $p(w_n|z_n, \beta)$ - tikimybės, sąlygotos temos z_n.

Taip pat modelyje yra daromos ir kelios prielaidos. Dirichlė skirstinio dimensija k ir temos kintamojo dimensija z yra fiksuoti ir žinomi. Žodžių tikimybės apibrėžiamos kaip $k \times V$ matrica β, kur $\beta_{ij} = p(w^j = 1|z^i = 1)$ yra laikomu fiksuotu kiekiu. Tuo tarpu N nepriklauso nuo visų duomenis generuojančių kintamųjų, tokių kaip θ ir z.

θ yra k-matis Dirichlė atsitiktinis kintamasis, kuris įgyja reikšmes $(k-1)$-simplekse (k - vektorius θ įgyja reikšmes $(k-1)$-simplekse, jei $\theta_i \geq 0, \sum_{i=1}^{k} \theta_i = 1$), ir turi tikimybių tankį, lygų:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} ... \theta_k^{\alpha_k - 1},$$

kur parametras α yra k-vektorius, kurio komponentai $\alpha_i > 0$, o $\Gamma(x)$ yra gama funkcija. Turėdami α ir β parametrus, bendras θ, temų rinkinio $\mathbf{z}$ ir $\mathbf{w}$ paskirstymas išreiškiamas:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta),$$

kur $p(z_n|\theta)$ yra θ_i unikaliam i, kad $z_n^i = 1$. Integruodami θ ir sumuodami z, gauname ribinį dokumento skirstinį:

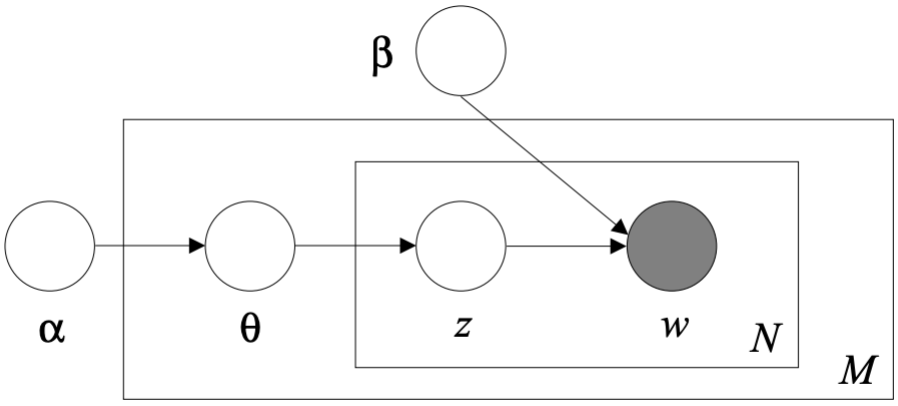
$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) (\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta)) d\theta.$$

Galiausiai, paėmę atskirų dokumentų ribinių tikimybių sandaugą, gauname struktūrizuoto teksto rinkinio tikimybę:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) (\prod_{d=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta) p(w_{dn}|z_{dn}, \beta)) d\theta_d.$$

Žemiau yra pateikiama grafinė tikimybinio modelio reprezentacija (žr., 1 pav.). Galime pastebėti, jog LDA yra sudarytas iš tryjų lygių. Taip pat matome, jog α ir β parametrai yra struktūrizuoto teksto rinkinio lygio parametrai, kurie paimami vieną kartą kuriant teksto rinkinį. Tuo tarpu θ_d kintamieji yra dokumento lygio, kurie taip pat paimami vieną kartą kiekvienam dokumentui. Galiausiai, z_{dn} ir w_{dn} yra žodžio lygmens kintamieji ir yra paimami kiekvienam žodžiui kiekviename dokumente.

Svarbu paminėti, jog reikia atskirti LDA nuo paprastojo daugianario Dirichlė klasterizacijos modelio. Klasikinis klasterizacijos algoritmą sudaro dviejų lygių modelis, kuris struktūrizuoto teksto rinkinį analizuoja vieną kartą. Taip pat daugianario grupavimo kintamąjasis yra pasirenkamas vieną kartą kiekvienam struktūrizuoto teksto rinkinio dokumentui. Taigi, kaip ir daugelyje grupavimo modelių atveju, toks modelis priskiria dokumentą tik vienai temai ir apriboja dokumento susiejimą su keliomis



1 pav.: LDA algoritmo grafinė reprezentacija. Paveikslėlio šaltinis [22]

tomomis. Tuo tarpu LDA yra trijų lygių modelis, todėl dokumento atrinkimas temai yra atliekamas pakartotinai. Todėl šis modelis dokumentą gali susieti su keliomis temomis.

Be LDA, temų modeliavimo srityje yra naudojami ir kiti modeliai, tokie kaip latentinė semantinė analizė (LSA), dar žinoma kaip latentinis semantinis indeksavimas (LSI) ar tikimybinė latentinė semantinė analizė (pLSA). Šių modelių plačiai nenagrinėsime, tačiau glaustai įvardinsime jų trūkumus vertinant LDA kontekste. Lyginant su LDA, LSA nėra toks efektyvus metodas, nes norint gauti tikslius rezultatus, reikia didesnio kiekio žodžių ir dokumentų rinkinio. Taip pat lyginant su pLSA, LDA paprastai veikia geriau, nes gali lengvai apdoroti naujus dokumentus bei pLSA algoritmui labiau būdingas persimokymas (angl. *overfitting*). Temų modeliavimo srityje yra daugybė kitų sukurtų algoritmų. Vis dėlto, daugumai jų yra skirta mažai akademinio dėmesio, dėl šios priežasties jie turi nemažai apribojimų ir kitų trūkumų, kurie lieka nenagrinėjami ir neištaisyti.

Taigi, apibendrinant galime teigti, kad LDA yra populiarus ir įvairiose srityse naudojamas teksto apdorojimo metodas. LDA modelis turi aiškią vidinę struktūrą ir gali skaičiavimams naudoti efektyvius, tikimybėmis paremtus išvadų priėmimo algoritmus [33]. LDA modelio parametrų dydis ir mokymo dokumentų skaičius yra nepriklausomi, dėl to jis labiau tinka didelio masto duomenims apdoroti. Be to, LDA yra hierarchinis modelis, kuris yra stabilesnis, ir mažiau tikėtina, kad jam bus būdingas persimokymas (angl. *overfitting*). Todėl atsižvelgiant į paminėtų mokslinių atliktus tyrimus bei LDA pranašumus, šiame tyrime taip pat bus pasinaudota LDA algoritmu.

2.7 Klasterizavimas

Tyrimuose išanalizuoti ir suprasti didžiulius kiekius duomenų yra labai sudėtinga. Tam, kad šiuos duomenis galėtume lengviau paversti naudinga ir aiškiai suprantama informacija, naudojami įvairūs duomenų apdorojimo ir tvarkymo metodai. Vienas iš tokių metodų – klasterizavimas (angl. *clustering*), dar įvardinama kaip klasterių analizė (angl. *cluster analysis*). Klasterizavimas taip pat yra priskiriamas prie neprižiūrėto mašininio mokymosi metodų. Taikant šį metodą duomenys yra sugrupuojami į tarpusavyje panašių stebėjimų grupes, kurios vadinamos klasteriais. Kitaip tariant, klasterį galime įsivaizduoti kaip tam tikrų objektų rinkinį, kurie tarpusavyje pasižymi panašiomis sąvybėmis ir

yra nepanašūs į objektus, priklausančius kitiems klasteriams [34]. Klasterizavimas yra ypač naudingas ir svarbus apdorojant bei analizuojant naujus duomenis, kai jie yra sugrupuojami į iš anksto neapibrėžtas grupes. Taip gali būti aptikta svarbi informacija, kuri neapdorotuose (angl. *raw*) duomenyse greičiausiai būtų nepastebėta [35].

2.7.1 K-vidurkių metodas

Tarp praktikoje naudojamų įvairių klasterizavimo algoritmų dažniausiai taikomas K-vidurkių klasterizavimo metodas (angl. *K-means*). K-vidurkių klasterizavimo algoritmas minimizuoja kvadratinę paklaidą, todėl jį galima laikyti ir kvadratinės paklaidos algoritmu (angl. *squared error clustering algorithm*) [36]. Pritaikant duomenims algoritmą yra suformuojamas K kiekis klasterių, o kiekviename klasteryje priskirtų narių panašumas pasiekiamas iteratyviai mažinant kvadratinę paklaidą E:

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

kur K - klasterių skaičius, n – klasterizuojamo objekto požymių kiekis, $d(x_j, c_i)$ - atstumas tarp objekto x ir klasterio centro c [37].

Įvertinti klasteryje esančių elementų tarpusavio panašumams gali būti naudojamos įvairios metodikos, tačiau populiariausias ir dažniausiai pasirenkamas metodas yra Euklido atstumas. Kai Euklido atstumas yra mažas, duomenys klasteryje yra vienas su kitu panašūs. Priešingu atveju, didelis atstumas indikuoja, jog tarp klasterio elementų būdingas mažas panašumas. Kai nagrinėjami duomenys yra klasterizuojami pagal n požymių, Euklido atstumas tarp elementų p ir q apskaičiuojamas taip:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Nagrinėjant klasterizavimo rezultatų gerumo dydį E galima įvertinti ir surasti optimalų klasterių kiekį. Vienas iš būdų, padedančių parinkti geriausią klasterių kiekį, yra alkūnės metodas (angl. *elbow method*). Taikant šį metodą klasterizavimas atliekamas vis su skirtingu klasterių kiekiu, kiekvieną kartą apskaičiuojant kvadratinės paklaidos sumą. Tada grafike atvaizduojama kvadratinės paklaidos sumos priklausomybė nuo klasterių kiekio. Galiausiai grafike yra randamas taškas, arba kitaip klasterių skaičius, nuo kurio tęsiant klasterių kiekio didinimą kvadratinės paklaidos sumos reikšmė reikšmingai nebemažėja.

Dar vienas metodas, kuris gali padėti surasti optimalų klasterių kiekį, yra vadinamas silueto koeficientu (angl. *silhouette coefficient*). Šis metodas taip pat gali atskleisti, kaip tinkamai elementai priskirti sudarytiems klasteriams. Silueto koeficiento įverčio skaičiavimas yra atliekamas ieškant vidutinio atstumo $a(i)$ tarp klasteryje esančio elemento i ir kitų tame klasteryje esančių elementų [38]. Tada surandami vidutiniai atstumų skirtumai $d(i, C)$ atstumai tarp elemento i ir kituose klasteriuose

C esančių elementų. Apskaičiavus atstumus yra parenkamas mažiausias $d(i, C)$, kurį įvardinsime kaip $b(i)$. Suradus visus atstumus galiausiai skaičiuojamas silueto koeficiento įvertis:

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

Stebėjimai, kurių koeficiento įverčio $s(i)$ vertė arti 1, yra laikomi labai gerai sugrupuotais, tuo tarpu maža įverčio vertė, kuri arti 0, reiškia, kad stebėjimas yra tarp dviejų grupių. Stebėjimai su neigiama įverčio reikšme parodo, jog tikriausiai stebėjimas yra įtrauktas į neteisingą klasterį.

2.7.2 Hierarchiniis metodas

Dar vienas klasterizacijos metodas, kuris buvo taikytas darbe, yra hierarchinis klasterizavimas. Hierarchinio klasterizavimo metodai yra tokie grupavimo metodai, kuriais pirma yra išsiaiškinama bendra visų klasterių tarpusavio priklausomybių struktūra. Tik nustačius struktūra yra sprendžiama, koks optimaliausias klasterių kiekis turėtų būti pasirinktas. Šio tipo klasterizavimo metodai yra skirstomi į jungimo ir skaidymo metodus. Naudojantis jungimo metodais, pirmiausiai visi stebėjimai yra išskiriami į atskirus klasterius. Kitaip tariant, kiek naudajama stebėjimų, tiek sudaroma klasterių. Tada imami du stebėjimai bei jungiami į bendrą klasterį. Toliau kiekvienas naujas stebėjimas yra sujungiamas su jau sudarytu klasteriu arba sujungiami jau sudaryti du klasteriai. Formuojami klasteriai nėra skaldomi, jie yra tik jungiami vienas prie kito. Šis veiksmas atliekamas, kol galiausiai gaunamas vienintelis klasteris, sudarytas iš visų stebėjimų. Matydamas klasterių tarpusavio struktūrą, tyrėjas pats nusprendžia, koks klasterių kiekis yra optimaliausias. Tuo tarpu skaidymo metodų atveju yra taikomas priešingas procesas - sudarytas bendras klasteris yra nuosekliai skaidomas į mažesnias grupes.

Hierarchinis klasterizavimo atveju dažniausiai klasterių tarpusavio priklausomybių struktūra yra pateikiama grafiškai, nubrėžiant dendrogramą. Tai yra tam tikro tipo diagrama, kuri vaizdžiai parodo tyrėjui klasterių tarpusavio sąryšius ir taip leidžia nuspręsti, koks klasterių kiekis yra tinkamiausias.

Hierarchiniai klasterizavimo metodų yra įvairių, jie tarpusavyje skiriasi taikomu skaičiavimo algoritmu. Šiame darbe taikant hierarchinį klasterizavimą buvo pasinaudota „Ward“ metodu. Skirtingai nei daugumai kitų naudojamų metodų atvejų, „Ward“ metodui nereikia sudaryti atstumo matricos. Atstumų matrica – tai tam tikra dvimatė matrica, atvaizduojanti atstumus tarp dviejų skirtingų stebėjimo porų. Taigi, taikant šį metodą atstumas tarp sudaromų klasterių yra apskaičiuojamas pagal:

$$\mathrm{D}_{KL} = \frac{||\bar{x_K} - \bar{x_L}||^2}{(\frac{1}{n_K} + (\frac{1}{n_L})},$$

kur $\bar{x}$ parodo klasterio C vidurkį, o n - pažymi stebėjimų skaičių.

2.8 Statistiniai testai

Išanalizavus „Facebook“ puslapių keliamo turinio strategijas, buvo pastebėti bruožai, kurie yra būdingi šioms strategijoms. Tam kad pastebėti bruožai būtų statistiškai patvirtinti, buvo pasitelkti statistiniai testai ir tikrinamos išsikeltos hipotezės. Hipotezių tikrinimui bus naudojami Kruskal-Wallis testas bei Dunn testas.

Kruskal-Wallis testas, dar žinomas kaip „Kruskal-Wallis H testas“ arba „Kruskal-Wallis ANOVA“, yra alternatyvus testas vieno faktoriaus dispersinei analizei („One-Way ANOVA“). Šis testas naudojamas patikrinti, ar egzistuoja statistiškai reikšmingas skirtumas tarp trijų ar daugiau nepriklausomų grupių medianų. Testas yra praplėsta „Mann Whitney U testo“ versija, kuri naudojama tik dviejų grupių lyginimui. Kruskal-Wallis testas yra naudojamas, kai yra pažeistos ANOVA prielaidos. Taip pat nuo ANOVA šis testas skiriasi tuo, jog nėra lyginami vidurkiai, kurie yra jautresni išskirtims. Šis testas patogus tuo, jog netaiko jokios prielaidos apie duomenų skirstinį, jį galima naudoti su skirtingo dydžio grupėmis, t.y. duomenys gali būti nesubalansuoti.

Kruskal-Wallis testo statistika H yra apskaičiuojama taip:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1),$$

kur N - bendras imties dydis, k - lyginamų grupių skaičius, R_i - grupės rangų sumą, n_i - grupės imties dydis. Kruskal-Wallis teste naudojamos tokia nulinė ir alternatyvi hipotezė:

$$\begin{cases} H_0 : \text{Visų grupių mediana yra vienoda.} \\ H_1 : \text{Visų grupių mediana nėra vienoda.} \end{cases}$$

Jei atlikus Kruskal-Wallis testą išsiaiškinama, jog rezultatai yra statistiškai reikšmingi, tada tikslinga atlikti Dunn testą, kurio pagalba galima išsiaiškinti, kurios tiriamos grupės tarpusavyje skiriasi. Šio testo metu yra atliekamas tyriamų grupių tarpusavio palyginimas, o gauti rezultatai parodo, kurios grupės skiriasi statistiškai reikšmingai. Dviejų tiriamų grupių skirtumo z-testo statistiką yra apskaičiuojama taip:

$$z_i = \frac{y_i}{\sigma_i},$$

kur i pažymi vieną iš $1, ..., m$ palyginimų, $y_i = W_A - W_B$. Čia W_A reiškia i-osios grupės rangų sumos vidurkis. σ apskaičiuojama pagal:

$$\sigma_i = \sqrt{\left(\frac{N(N+1)}{12} - \frac{\sum(T_s^3 - T_s)}{12(N-1)}\right)\left(\frac{1}{n_A} + \frac{1}{n_B}\right)},$$

kur N - bendras visų grupių stebėjimų kiekis, n_A ir n_B - tarpusavyje lyginamų grupių dydžiai, T - Kruskal-Wallis testo tų pačių stebėjimo verčių korekcijos s- ąjai vertei. Kai tuo pačiu metu yra atliekami keli palyginimai, išauga pirmo tipo klaidos tikimybė. Šiai problemai spręsti pasitelkiamos p-reikšmės koregacijos. Yra keli skirtingi koregacijų būdai, tačiau populiariausia yra „Bonferroni“ koregacija, kur p-reikšmė lygi $p * m$. Čia p reiškia orginalią gautą p-reikšmę, o m - bendras atliekamų palyginimų skaičius. Testuose naudojamas reikšmingumo lygmuo α lygus 0,05.

3 Duomenų aprašas

3.1 Nagrinėjami „Facebook" puslapiai

Tyrime yra nagrinėjami trys „Facebook“ puslapiai: „puslapis1“, „puslapis2“, „puslapis3“, kurių autorinės teisės priklauso „Įmonė1“. Iš išvardintų puslapių populiariausias socialinėje platformoje yra „puslapis1“, turintis per 25 mln. sekėjų. Kiti moderuojami puslapiai savo populiarumu atsilieka nuo pirmojo - jie socialinėje erdvėje turi daugiau kaip 1 mln. sekėjų.

3.2 Kintamieji

„Facebook" puslapių populiarumo vertinimo analizei buvo pasirinka nagrinėti duomenis, ištrauktus iš „Tubular" platformos [39]. Ši platforma yra skirta vaizdo įrašų parametrų sekimui ir jų analizėms. Darbe naudojama duomenų imtis yra subalansuota, o imtį sudaro 188 kintamųjų ir 8264 skirtingų vaizdo įrašų, kurie buvo publikuoti socialinėje platformoje 2021 m. laikotarpyje. Tam, kad būtų lengviau apibūdinti visus duomenų imties kintamuosius, juos suskirstisime į grupes pagal rodiklius, apibūdinančius pagrindinius vaizdo įrašų parametrus, valstybių gyventojų peržiūrų rodiklius ir kintamuosius, atskleidžiančius įrašų žiūrimumą pagal amžių ir lytį.

- Pirmiausiai bus apibūdinami pagrindiniai vaizdo įrašų parametrai, kurie dažniausiai siejami ir su įrašų našumu. Pagrindinius įrašų parametrus sudaro tokie kintamieji, kaip vaizdo įrašo trukmė, išreikšta sekundėmis, įrašo parodymo kiekis „Facebook" vartotojų ekranuose (angl. *impressions*), 3 sekundžių ir 60 sekundžių vaizdo įrašo peržiūros, pasidali- nimų, komentarų bei mėgti (angl. *like*) mygtukų paspaudimo kiekis, bendras vartotojų interakcijų kiekis, vidutinis žiūrėjimo laikas sekundėmis bei paspaudimų ant vaizdo įrašo skaičius. Visi šie išvardinti kintamieji yra kiekybiniai. Taip pat pagrindiniams vaizdo įrašų parametrams yra priskiriami ir šie kokybiniai kintamieji: vaizdo įrašo publikavimo data bei laikas, vaizdo įrašo pavadinimas, filmuko aprašymas ir „Facebook“ puslapio pavadinimas, kuriame buvo patalpintas įrašas. Vartotojų ekranuose parodomų įrašų skaičiaus rodiklis atskleižda, kiek kartų vaizdo įrašas buvo parodytas „Facebook" vartotojų ekranuose nepaisant to, ar jie peržiūrėjo įrašą. Tam pačiam vartotojui gali būti parodytas vaizdo įrašas daugiau kaip vieną kartą, tačiau tam, kad parodymų rodiklis fiksuotų skirtingus parodymus, „Facebook" platformos internetinis puslapis turi būti perkrautas kiekvieną kartą. 3 sekundžių vaizdo įrašo peržiūros parodo, kiek kartų vaizdo įrašas buvo peržiūrėtas mažiausiai 3 sekundes. Vaizdo įrašo peržiūros skaičiuojamos atskirai nuo kiekvieno vaizdo įrašo parodymo ir neįtraukiamas laikas, praleistas paleidžiant vaizdo įrašą. Taip pat šis parametras „Facebook“ platformoje yra naudojamas ir atvaizduojant bendrą vaizdo įrašo peržiūrų skaičių, todėl toliau darbe šį parametrą

vadinsime kaip vaizdo įrašo peržiūros. Tuo tarpu 60 sekundžių vaizdo įrašo peržiūros parodo, kiek kartų vaizdo įrašas buvo žiūrimas mažiausiai 60 sekundžių. Atitinkamai pasidalinimų, komentarų ir mėgti mygtukų paspaudimų rodikliai parodo, kiek vartotojų pasidalino matytu vaizdo įrašu, kiek žmonių pakomentavo įrašą ir kiek žmonių paspaudė mėgti mygtuką. Svarbu paminėti, jog 2016 m. „Facebook“ įvykdė pakeitimus savo platformoje ir išleido naują ypatybę reakcijas (angl. *Reactions*), kuri tiesiogiai pakeitė vartotojų elgseną, kaip jie gali reaguoti į matomą vaizdo įrašą [40]. Ši funkcija yra esamos funkcijos paspausti mėgti ant vaizdo įrašo papildymas: vartotojai dabar gali pasirinkti vieną iš penkių papildomų animuotų jaustukų, kuriais gali išreikšti savo požiūrį į pamatytą vaizdo įrašą. Kiekviena ekrane matoma emocinga piktograma pavadinta pagal reakciją, kurią ji perteikia. Taigi, dabar egzisuoja šešios tokios reakcijos: „Patinka“, kurį jau yra plačiai žinoma, taip pat „Myliu“, „Haha“, „Vau“, „Liūdna“ ir „Pikta“ emocijos. Duomenyse visos šios šešios emocijos yra sujungtos į vieną kintamąjį - mėgti mygtuko paspaudimo kiekį, todėl mėgti mygtuko paspaudimo ir reakcijos sąvokas toliau darbe vartosime kaip sinonimus. Vartotojo interakcija socialinės platformos kontekste yra bet kokia vartotojų sąveika su vaizdo įrašu. Taigi, bendras vartotojų interakcijų kiekis parodo bendrą pasidalinimų, komentarų ir paspaudimo mėgti skaičių. Vidutinis žiūrėjimo laikas sekundėmis atspindi, kiek vidutiniškai vartotojams buvo įdomu žiūrėti vaizdo įrašą, o paspaudimų skaičius ant vaizdo įrašo atskleidžia, kiek kartų vartotojai, pamatę vaizdo įrašą, pasirinko jį žiūrėti.

- Duomenų imtyje taip pat nagrinėjami kintamieji, atspindintys valstybių gyventojų peržiūrų rodiklius. Duomenyse yra pateikiami 149 valstybių parametrai, parodantys, kiek kiekvienos šalies gyventojų peržiūrėjo kiekvieną nagrinėjamą įrašą. Šį valstybių sąrašą sudaro įvairaus ekonominio išsivystymo šalys iš Europos, Azijos, Afrikos, Australijos, Šiaurės ir Pietų Amerikos žemynų. Tokia plati valstybių įvairovė atskleidžia didžiulį nagrinėjamų „Facebook“ puslapių populiarumą ir žiūrimumą visame pasaulyje.

- Galiausiai pristatomi duomenys, atskleidžiantys žiūrinčios auditorijos amžių bei lytį. Imtyje pateikiami du kintamieji, kurie atitinkamai parodo, kiek peržiūrų buvo surinka vyriškos ir kiek moteriškos auditorijos. Peržiūrų skaičių pagal amžių apibūdina 7 kintamieji, atspindintys, kiek kartų įrašas buvo peržiūrėtas asmenų, priklausančių tam tikrai amžiaus kategorijai. Amžiaus kategorijos suskirstytos į 13-17 m., 18-24 m., 25-34 m., 35-44 m., 45-54 m., 55-64 m., 65 m. ir vyresnio amžiaus žmones. Taip pat duomenų imtyje yra 14 kintamųjų, atitinkamai parodančių įrašų peržiūras pagal lyties kiekvieną amžiaus grupę.

3.3 Papildomų kintamųjų sukūrimas

Pradėjus analizuoti duomenis buvo pastebėtas papildomų išvestinių kintamųjų poreikis, be kurių būtų sudėtinga atlikti puslapių populiarumo vertinimą. Dėl šios priežasties buvo sukurti papildomi kintamieji: vaizdo įrašo ilgis minutėmis, metų ketvirtis, mėnesis, savaitės diena bei valanda, kurios metu buvo publikuotas vaizdo įrašas, vaizdo įrašų ilgio kategorijos kintamasis ir vaizdo įrašų parodymų ir paspaudimų ant jo skaičių santykis (angl. *Clickthrough rate (CTR)*). Vaizdo įrašo ilgis minutėmis yra kintamasis, parodantis vidutinę įrašo trukmę, išreikštą ir suapvalintą minučių tikslumu. Vaizdo įrašų ilgio kategorijos kintamasis buvo sukurtas kaip kintamasis, suskirstantis įrašus į 3 kategorijas: iki

5 min. trukmės (trumpi įrašai), 5-20 min. trukmės (vidutinės trukmės įrašai) ir 20 min. ir ilgesnės trukmės vaizdo įrašus (ilgi įrašai). Savaitės dienos, valandos, mėnesio ir metų ketvirčio kintamieji buvo išvesti iš vaizdo įrašo publikavimo laiko. Jie atitinkamai parodo savaitės dieną, valandą, mėnesį ir metų ketvirtį, kurio metu vaizdo įrašas buvo publikuotas socialiniame tinkle. Parodymų ir paspaudimų santykis atspindi, kokia dalis vartotojų, pamačiusių vaizdo įrašą, pasirinko jį pradėti žiūrėti.

3.4 Trūkstamų reikšmių panaikinimas

Peržvelgus duomenų imtį buvo patikrinta, kiek duomenyse yra trūkstamų reikšmių. Nustatyta, jog daugiausia reikšmių trūko kintamajam, apibūdinančiam paspaudimų skaičių ant vaizdo įrašo. Iš viso bendras trūkstamų reikšmių kiekis duomenų imtyje apėmė 263 vaizdo įrašus arba 3,18 proc. visų turinų filmukų. Kadangi vaizdo įrašų populiarumas nėra pastovus ir kiekvieno įrašo parametrai tarpusavyje gali skirtis labai smarkiai, trūkstamas reikšmes užpildyti gretimų įrašų dydžių vidurkiais ar bendru imties vidurkiu būtų netikslinga. Dėl šios priežasties, taip pat ir dėl to, kad trūkstamos reikšmės sudaro sąlyginai nedidelę dalį imties, buvo nuspręsta, jog įrašai su trūkstamais duomenim bus panaikinti.

3.5 Pirminė duomenų analizė

Siekiant išsamiau suprasti duomenis, kiekvienam puslapiui buvo sudaryta bendrą pagrindinių rodiklių statistinės informacijos lentelė, kuri atspindi kintamųjų padėties bei sklaidos charakteristikas. Prieduose pateiktos lentelės (žr., 6, 7 ir 8 pav.) atitinkamai vaizduoja „puslapis1“, „puslapis2“ bei „puslapis3“ rodiklių statistikas.

Statistinės informacijos lentelėse pateikti duomenys patvirtina jau paminėtą teiginį, jog „puslapis1“ yra populiariausias iš visų nagrinėjamų puslapių. Lyginant su kitais kanalais, yra aiškiai pastebimas vaizdo įrašų žiūrimumo ir vartotojų interakcijų skirtumas. Kita vertus, nors „puslapis1“ yra populiaresnis, tačiau vidutinis įrašų žiūrėjimo laikas kiekviename puslapyje yra panašus ir skiriasi tik keliomis sekundėmis.

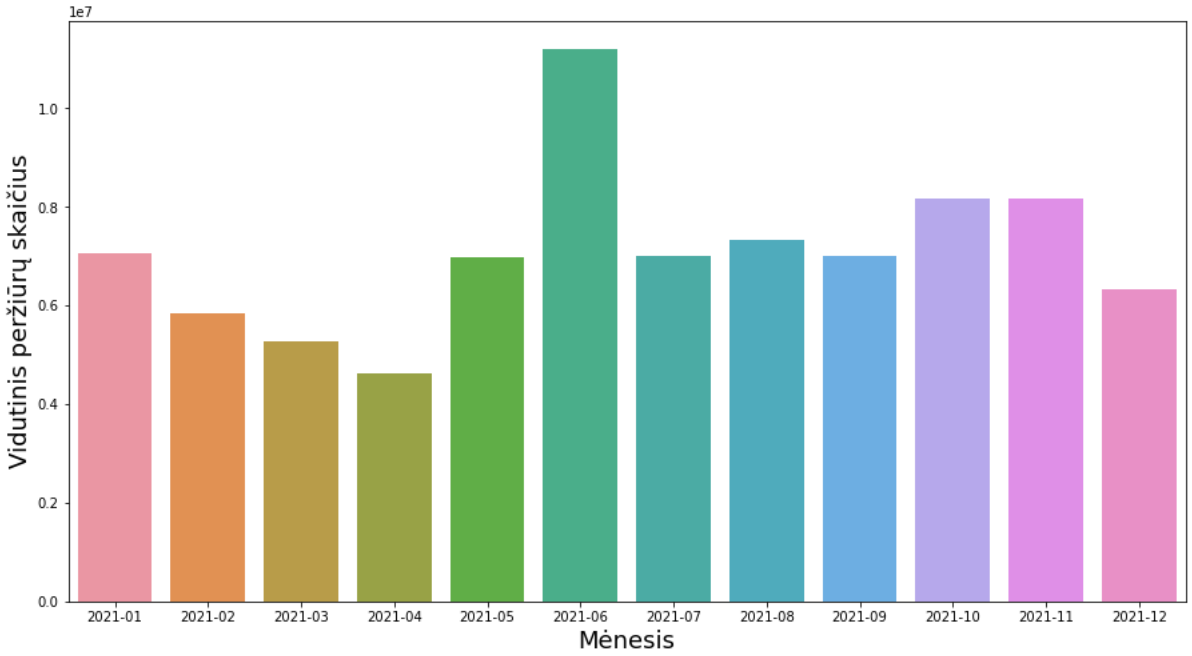
Žvelgiant į vidutines vaizdo įrašų trukmes galima pastebėti, jog „puslapis1“ ir „puslapis2“ puslapiuose dažniau yra talpinami trumpesnės trukmės vaizdo įrašai. Tuo tarpu „puslapis3“ vidutiniškai publikuoja vidutinio ilgio įrašus. Taip pat galime atkreipti dėmesį, jog kiekviename kanale yra patalpintų labai didelės trukmės įrašų. Tai gali indikuoti, jog šie įrašai buvo publikuojami pasitelkiant „Facebook“ tiesioginės transliacijos funkciją, nes įprasta leidžiamų vaizdo įrašų trukmė socialiniame tinkle negali viršyti 240 min. ribos [41]. Ši funkcija leidžia socialinės platformos naudotojams ir kūrėjams tiesiogiai transliuoti įvairius įvykius, pasirodymus ar vaizdo įrašus. Dažniausiai tiesioginės transliacijos yra ilgos, trunkančios daugiau kaip kelias valandas. Remiantis oficialia „Facebook“ informacija, tiesioginės transliacijos platformoje gali trukti iki 8 val., o ši riba sutampa su kiekvieno puslapio ilgiausių įrašų trukme [42].

Taip pat galime pastebėti, kad visuose kanaluose vidutiniškai atsispindi žemas įrašų parodymų ir paspaudimų ant jo skaičių santykis. Tokie rezultatai parodo įrašų patrauklumo trūkumą. Vartotojai sąlyginai retai linkę paspausti ant vaizdo įrašo, kai jį pamato socialinėje platformoje. To priežastis galėtų būti ta, kad auditorijai dažnai yra nepatraukli pirminė nuotrauka ar miniatūra (angl. *thumbnail*), kurią pamato dar prieš paspaudžiant ant vaizdo įrašo. „Facebook“ kontekste, miniatiūra (angl. *thumbnail*)

yra pirmasis elementas, kurį vartotojas pamatys naršydamas ir bandydamas nuspręsti, kokį vaizdo įrašą pasirinkti žiūrėti.

Galiausiai pažvelgę į paspaudimų ant įrašo skaičių bei į vaizdo įrašų peržiūrų kiekį galime pastebėti dar vieną veiksnį - nemaža dalis peržiūrų yra surenkama dėl „Facebook“ automatinio paleidimo funkcijos (angl. *autoplay*) [43]. Vaizdo įrašai, įkelti iš asmeninių „Facebook“ paskyrų ar puslapių, yra automatiškai paleidžiami, kai vartotojai naršo socialinėje platformoje. Taip pat, kai peržiūrimas vienas vaizdo įrašas, kitas įrašas yra iš karto užkraunamas ir paleidžiamas vartotojui. Dėl šios funkcijos aiškiai matomas skirtumas tarp paspaudimų ant įrašo ir vaizdo įrašo peržiūrų kiekio.

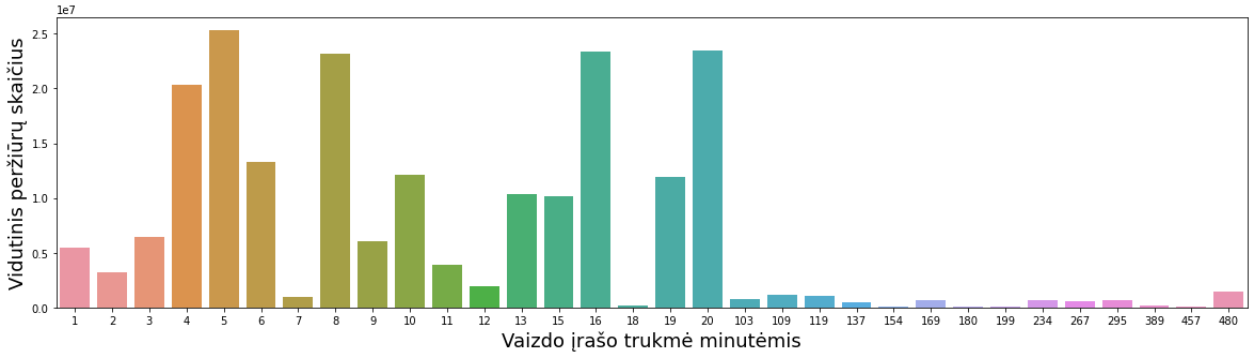
Pradėjus detaliau nagrinėti duomenis pastebėta, jog vidutinis vaizdo įrašų peržiūrų skaičius „puslapis1“ puslapyje neturėjo daug pasikeitimų. Tai iliustruoja ir žemiau pateiktas grafikas (žr., 2 pav.). Pirmame ketvirtyje peržiūrų skaičius mažėjo, tačiau metų viduryje įvyko ženklus peržiūrų šuolis. Lyginant su sausio mėnesiu, vasaros pradžioje peržiūros buvo išaugusios beveik du kartus. Keista tai, kad peržiūros išaugo tik birželio mėnesį, o vėliau rodiklis stabilizavosi. Tokio trumpio trumpo išaugusio rodiklio priežastį sunku įvardinti, tai galėjo sukelti tuo metu populiariomis temomis kuriami vaizdo įrašai, taip pat sėkmingos marketinginės kampanijos, skatinančius vartotojus peržiūrėti įrašus. Tuo tarpu kitų dviejų puslapių rezultatai buvo panašūs. Įdomu tai, jog rugpjūčio ir visais rudens mėnesiais „puslapis2“ puslapyje buvo išaugęs peržiūrų skaičius net iki penkių kartų, o „puslapis3“ puslapyje rugsėjo ir spalio mėnesiais - iki šešių kartų. Tokio teigiamo rezultato priežastys gali būti sėkmingos marketinginės kampanijos, labai kokybiškų ir sudominančių vaizdo įrašų sukūrimas. „puslapis2“ ir „puslapis3“ kanalų grafikai, vaizduojantys vidutines vaizdo įrašų peržiūras, pateikti priedų skiltyje.



2 pav.: Vidutinis vaizdo įrašų peržiūrų skaičius „puslapis1“ puslapyje.

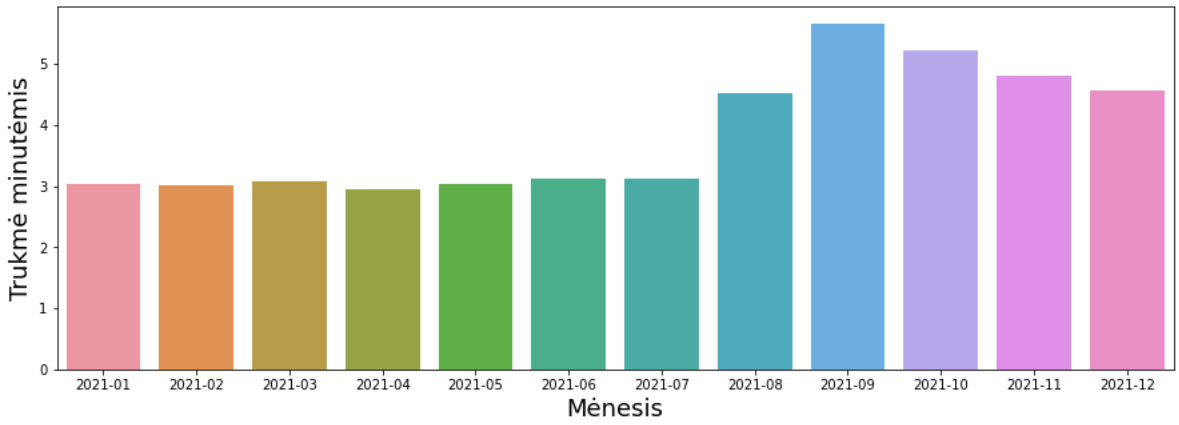
Analizuojant vidutinį vaizdo įrašų peržiūrų skaičių pagal vaizdo įrašų trukmę pastebėta, jog „puslapis1“ kanale didžiausią susidomėjimą sulaukia 5 min., 8 min., 16 min., bei 20 min. ilgio įrašai. Ši tendencija yra pavaizduota 3 grafike. Tai parodo, jog šio kanalo auditorija labiausiai mėgsta žiūrėti vi-

dutinio ilgio vaizdo įrašus. Kituose dviejuose kanaluose labiausiai žiūrimi yra 5 min., 6min. ir 10 min. ilgio įrašai. Puslapių lankytojams labiau patinka stebėti trumpesnius vaizdo įrašūs ir ilgos trukmės įrašai auditorijos dėmesio sulaukia mažai. Svarbu paminėti ir tai, jog žvelgiant į publikuojamų vaizdo įrašų kiekį pagal trukmę, visuose kanaluose labiausiai dominuoja labai trumpi vaizdo įrašai, kurių trukmė neviršija 5 min. Vis dėlto galima daryti prielaidą, kad kanalų moderatoriai atsižvelgė į žiūrovų susidomėjimą vidutinio ilgio įrašais, nes vėlesniais mėnesiais kanaluose buvo pradedami publikuoti ir ilgesnės trukmės filmukai.



3 pav.: Vidutinis peržiūrų skaičius pagal vaizdo įrašų trukmę „puslapis1“ puslapyje.

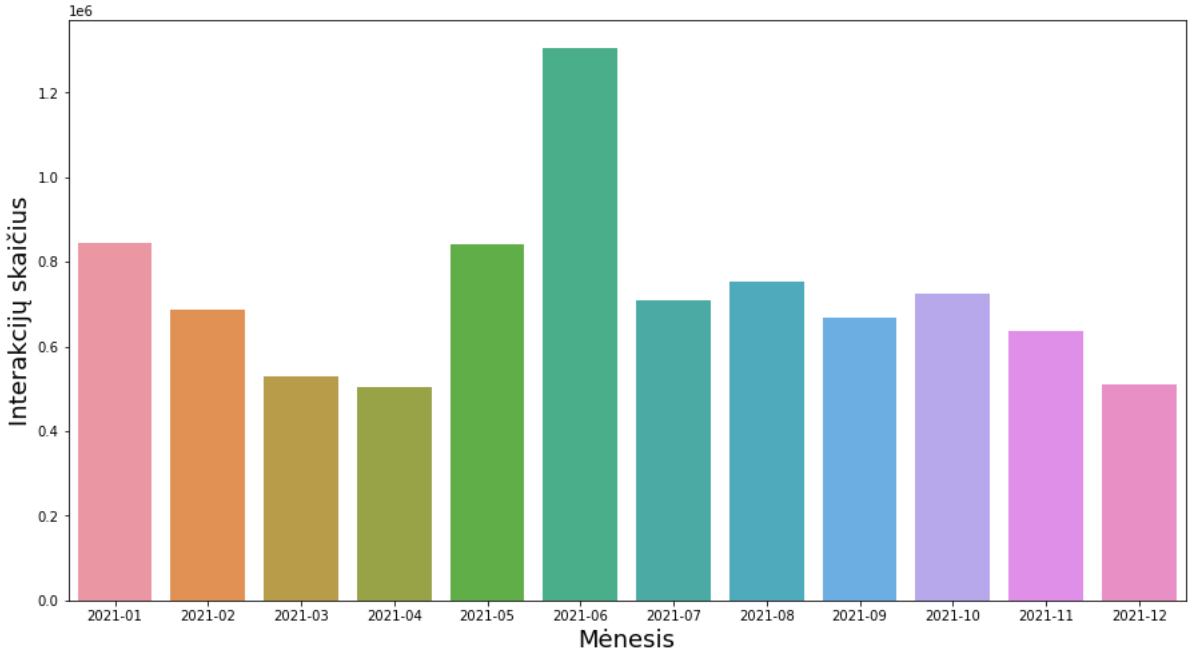
Toliau gilinantis į keliamų vaizdo įrašų trukmę ir peržvelgus vidutinę publikuojamų vaizdo įrašų ilgį kiekvieną mėnesį, labai aiškiai pasimatė, jog „puslapis1“ puslapio moderatoriai ilgą laiką stengėsi kurti 3 min. trukmės vaizdo įrašus. Žemiau pateiktas grafikas tai atspindi (žr., 4 pav.). Remiantis oficialia „Facebook“ pateikiama informacija, skelbiami filmukai, netrumpesni nei 3 min., gali būti monetizuojami socialinėje platformoje, t.y. už keliamus vaizdo įrašus publikuotojai gali gauti pajamas. Galima daryti prielaidą, jog dėl šios priežasties puslapyje stengtasi vidutiniškai publikuoti 3 min. ilgio įrašus. Taip pat pastebima, jog vėlesniais mėnesiais filmukų vidutinė trukmė išauga. Galima manyti, jog to priežastis yra ta, jog didžiausio žiūrovų susidomėjimo sulaukia trumpi ir vidutinės trukmės vaizdo įrašai. Labai panašios tendencijos atsispindi ir kituose nagrinėjamuose puslapiuose.



4 pav.: Vidutinė keliamų vaizdo įrašų trukmė „puslapis1“ puslapyje.

Žvelgiant į auditorijos sąveikavimą su vaizdo įrašais iš karto pastebėta stipri koreliacija tarp peržiūrų kiekio ir auditorijos interakcijų su filmuku. 5 grafike yra vaizduojamas vidutinis vartotojų interakcijų

skaičius „puslapis1“ kanale. Pažvelgus į grafiką galime pastebėti kanalo įrašuose sulaukiamų interakcijų kiekio mažėjimą. Nors interakcijų kiekis mažėja, šis smukimas nėra didelis ryškus. Tačiau tai gali indikuoti žiūrovų nepasitenkinimą turinio kokybe ar pačiu turiniu, taip pat auditorijos tapimą labiau pasyvia, kai vaizdo įrašas yra peržiūrimas, tačiau niekaip su juo nėra sąveikaujama. Tuo tarpu kitų dviejų puslapių pirmais trim ketvirčiais, išskyrus rugpjūčio mėnesį, sąveikų su vaizdo įrašais buvo sulaukiama labai mažai. Tačiau tokio rezultato galima buvo tikėtis, nes šiais mėnesiais šių puslapių vaizdo įrašai atitinkamai sulaukdavo ir mažai vaizdo peržiūrų.



5 pav.: Vidutinis vartotojų interakcijų skaičius „puslapis1“ puslapyje

Kiekvieno puslapio labiausiai žiūrinčių auditorijų pagal valstybes sąrašas yra pateikiamas 1 lentelėje. Galime pastebėti, jog tarp labiausiai žiūrinčių šalių nepatenka beveik jokia Europos žemyno valstybė. Nagrinėjamų kanalų vaizdo įrašus labiausiai mėgsta stebėti Azijos, Arfikos, Pietų Amerikos ir Šiaurės Amerikos valstybės. Taip pat pastebima, jog „puslapis2“ ir „puslapis3“ labiausiai žiūrinčios auditorijos persipina - tarp penkių labiausiai žiūrinčių valstybių, trys iš jų sutampa. Tuo tarpu „puslapis1“ turi pakankamai skirtingas žiūrinčias auditorijas, šio puslapio vaizdo įrašus labiausiai mėgsta žiūrėti Afrikos valstybės.

	„puslapis1“	„puslapis2“	„puslapis3“
1.	Indija(IN)	Meksikas(MX)	Indonezija(ID)
2.	Dramblio Kaulo Krantas(CI)	Egiptas (EG))	Meksikas(MX)
3.	Jungtinės Amerikos Valstijos(US)	Kambodža(KH)	Jungtinės Amerikos Valstijos(US)
4.	Burkina Fasas(BF)	Indonezija(ID)	Azerbaidžanas(AZ)
5.	Senegalas(SN)	Taivanas(TW)	Taivanas(TW)

1 lentelė: Kiekvieno kanalo labiausiai žiūrinčių valstybių gyventojų sąrašas.

2 lentelėje galime pamatyti kiekvieno kanalo amžiaus grupių išsidėstymą pagal žiūrimumą. Aiškiai

matyti, jog visų kanalų įrašus labiausiai mėgsta žiūrėti 25-34 m. asmenys. Tuo tarpu pilnametysės neturintys naudotojai yra vieni mažiausiai žiūrinčių vaizdo įrašų šiuose puslapiuose. Mažiausiai visuoe puslapiuose vaizdo įrašų stebi 65 m. ir vyresni žmonės. Viena vertus, tokio rezultato galima tikėtis. Vyresnio amžiaus žmonės naudojasi „Facebook“ tam, kad labiau socializuotųsi ir galėtų lengviau bendrauti su šeimos nariais bei draugais [44]. Tačiau yra klaidinga galvoti, jog vyresni žmonės sudaro mažą dalį socialinės platformos naudotojų. Remiantis 2019 m. duomenimis, net 41 proc. „Facebook“ vartotojų yra 65 metų ir vyresni asmenys [45].

	„puslapis1“	„puslapis2“	„puslapis3“
1.	25-34 m. amžiaus grupė	25-34 m. amžiaus grupė	25-34 m. amžiaus grupė
2.	18-24 m. amžiaus grupė	18-24 m. amžiaus grupė	35-44 m. amžiaus grupė
3.	35-44 m. amžiaus grupė	35-44 m. amžiaus grupė	18-24 m. amžiaus grupė
4.	45-54 m. amžiaus grupė	45-54 m. amžiaus grupė	45-54 m. amžiaus grupė
5.	55-64 m. amžiaus grupė	13-17 m. amžiaus grupė	55-64 m. amžiaus grupė
6.	13-17 m. amžiaus grupė	55-64 m. amžiaus grupė	13-17 m. amžiaus grupė
7.	65+ m. amžiaus grupė	65+ m. amžiaus grupė	65+ m. amžiaus grupė

2 lentelė: Amžiaus grupių išsidėstymas pagal žiūrimumą.

Gilinantis labiau į vaizdo įrašus stebinčias auditorijas buvo pastebėta, jog „puslapis1“ ir „puslapis3“ įrašus labiausiai mėgsta žiūrėti vyrai, o „puslapis2“ - moterys. Taip pat prieduose pateiktoje 9 lentelėje galime pamatyti labiausiai žiūrinčias puslapių auditorijos pagal amžiaus grupę ir lytį. Susidaro aiškus vaizdas, jog „puslapis2“ dominuojanti auditorija yra 18-54 m. moterys, o „puslapis1“ ir „puslapis3“ - 18-54 m. vyrai. Iš lentelių pateiktos informacijos galima manyti, jog „puslapis1“ ir „puslapis3“ kuriamas turinys orientuotas į vyriškąją auditoriją, o „puslapis2“ - į moteriškąją.

4 Temų modeliavimas

Norint geriau suprasti duomenis ir atlikti gilesnį populiarumo vertinimą, labai svarbu suprasti, kokiomis tematikomis ir apie ką yra publikuojami vaizdo įrašai. Dėl šios priežasties bus atliktas temų modeliavimas. Temų modeliavimas padės apibendrinti duomenų imtyje turimą tekstinę informaciją. Taigi, pasitelkdami šį metodą galėsime ištirti ir išsiaiškinti, koks turinys yra vaizduojamas papublikuotuose vaizdo įrašuose.

4.1 Tekstinio kintamojo paruošimas temų modeliavimui

Raktinių žodžių generavimui ir temų sukūrimui buvo pasitelkta vaizdo įrašų apibūdinimo kintamasis. Prieš atliekant temų modeliavimą, pirma reikėjo sutvarkyti ir paruošti tekstinę informaciją. Peržvelgus vaizdo įrašų apibūdinimus pastebėta, jog juose yra elektronių paštų nuorodos, pavienių kabučių, naujos eilutės simbolių bei jaustukų (angl. *emojis*). Jaustukas yra mažas logotipas, piktograma ar šypsenėlė, kuri yra įterpiama į tekstą bei naudojama atvaizduoti tam tikras emocijas, simbolius ar objektus. Jaustukai dažniausiai yra naudojami komunikuojant su kitais asmenimis socialiniuose tinkluose bei kitose komunikacijų sistemose. Taigi, pirma iš tekstų buvo pašalinta nereikalinga ir trukdanti atlikti temų modeliavimą informacija.

Rankiniu būdu pašalinus nereikalingus simbolius ir tekstą, toliau buvo atliktas tokenizavimas. Tokenizavimas yra tekstinės informacijos apdorojimas ir jo suskaidymas į mažesnes dalis, dar vadinamomis žetonais (angl. *Tokens*) [46]. Žetonus gali sudaryti atskiri žodžiai ar frazės. Šio proceso tikslas sutvarkyti netruktūruotus tekstinius duomenis ir paruošti juos temų modeliavimui. Taigi, tokenizavimo metu tekstas buvo suskaidytas į žoždius ar frazes, taip pat visiškai pašalinti skyrybos ženklai bei kiti nereikalingi simboliai. Tam buvo pasitelkta `Python` biblioteką `Gensim` ir jos funkcija `gensim.utils.simple_preprocess` [47]. Naudojantis šia funkcija buvo pasinaudota papildomu parametru `(deacc=True)` ir pašalinti visi skyrybos ženklai.

Kitu žingsiu tekstui buvo atliekamas lemavimas - žodžio pavertimas jo bendrine forma (angl. *lemmatisation*), ir pašalinamos žodžių galūnės (angl. *stemming*) [48]. Lemavimo procesu paprastai norima supaprastinti žodžio struktūrą ir padaryti jį bendrine formą, kuri yra žinoma kaip lema. Šių veiksmų tikslas yra sumažinti bendrą žodžių skaičių žodyne ir palikti tik unikalius žodžius. Dėl to duomenų rinkinio kokybė turėtų būti pagerinta, gauti patikimesni temų modeliavimo rezultatai bei sukurtos tikslesnės temos.

Paskutiniame teksto apdorojimo ir paruošimo etape buvo sukurta žodžių dokumento matricą (angl. *Document-Word matrix*). Šis etapas reikalingas taikant latentinio Dirichlė paskirstymo modelio algoritmą, nes algoritmas reikalauja žodžių dokumento matricos kaip pagrindinės įvesties. Šiam procesui įgyvendinti buvo pasitelkta funcija `CountVectorizer` iš `scikit-learn` bibliotekos [49]. Parametruose buvo nurodyta, jog funkcija atsižvelgtų tik į žodžius, pasikartojančius bent 15 kartų, pašalintų angliškus nereikšmingus žodžius (angl. *stopwords*) bei konvertuotų visas teskte esančias raides į mažąsias. Nereikšmingi žodžiai - tai savarankiškos prasmės neturintys žodžiai, tokie kaip prielinksniai, artikeliai, jungtukai ar įvardžiai. Šie žodžiai dažnai yra aptinkami tekstuose, todėl svarbu juos pašalinti.

4.2 Latentinio Dirichlė paskirstymo modelis

Apdorojus ir paruošus tekstinę informaciją bei sukūrus žodžių dokumento matricą buvo pasinaudota latentinio Dirichlė paskirstymo modeliu, kurio matematinė išraiška aprašyta temų modeliavimo skyrelyje. Tyrime latentinio Dirichlė paskirstymo modelis buvo sudarytas pritaikant `LatentDirichletAllocation` funkciją iš `scikit-learn` bibliotekos, kuri naudoja adaptyvaus laiko variancinį Bajeso metodą [50]. Taikant modelį buvo pasirinkta surasti ir sudaryti 15 temų.

Modelio tinkamumą tikrinsime pagal log-tikėtinumo funckijos reikšmę. Įprastai log-tikėtinumo funckijos reikšmės gali svyruoti nuo neigiamos iki teigiamos begalybės. Mūsų atveju gauta reiškmė $Log\mathcal{L} = -82409,544$. Pati modelio log-tikėtinumo funckijos reikšmė dažniausiai nesuteikia daug prasmingos informacijos, tačiau ji yra labai naudinga lyginant tarpusavyje du ar daugiau modelių. Dėl šios priežasties sudarysime daugiau modelių ir lygindami log-tikėtinumo funckijos reikšmes surasime tinkamiausią.

4.3 Geriausio modelio parinkimas

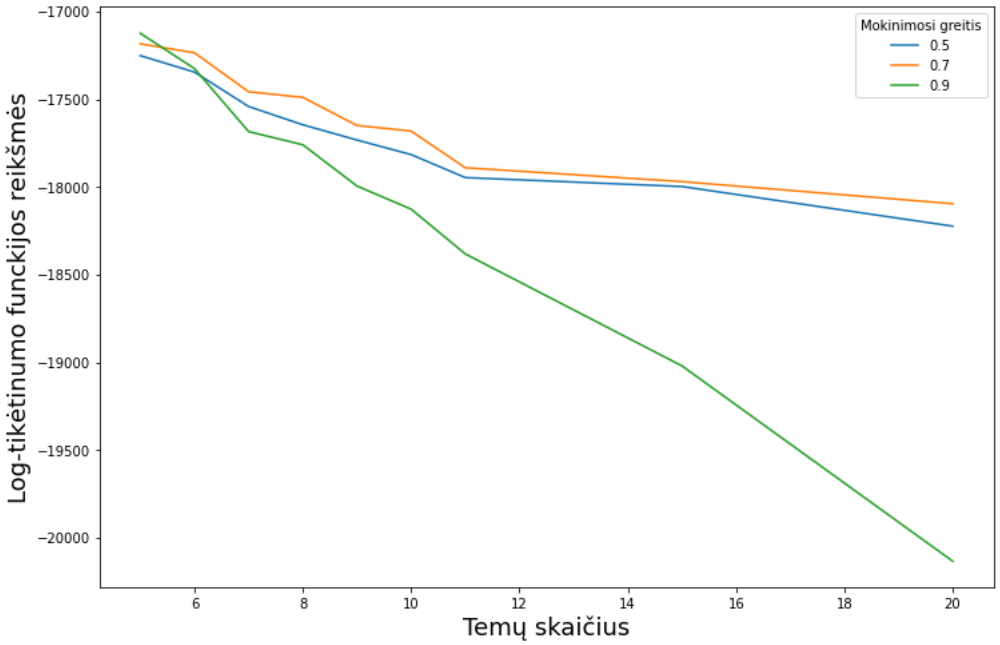
Geriausio modelio parinkimo etapas buvo atliktas naudojantis tinkleline paieška (angl. *grid search*). Šiam procesui atlikti buvo naudojama `GridSearchCV` funkcija iš `scikit-learn` bibliotekos. Šios funkcijos pagalba galime patikrinti skirtingas hiperparametrų kombinacijas ir taip surasti, ko-

kia hiperparametrų kombinacija nulemia didžiausią modelio tikslumą. Kitaip tariant, yra atliekamas hiperparametrų optimizavimas. Mašininio mokymosi modeliuose hiperparametrai yra algoritmo požymiai arba parinktys, kurias nustato algoritmo naudotojas ir kurių reikšmės yra keičiamos, taip siekiant pagerinti algoritmo veikimą. Įprastai iš karto nuspręsti, kokie hiperparametrai tinkamiausi, yra labai sudėtinga, todėl tinklelinė paieška padeda išspręsti šią problemą.

Tinklelinės paieškos proceso metu modelių vertinimas yra atliekamas taikant kryžminę validaciją (angl. *cross validation*). Kryžminės validacijos populiariausias ir dažniausiai naudojamas metodas yra `k-fold` validacija. Šio metodo veikimas yra pagrįstas tuo, jog nagrinėjama duomenų imtis yra suskirstoma į k dalių, tada modelio treniravimas yra atliekamas su $k-1$ dalimis, paliekant k-tąją dalį testavimui kaip validavimo rinkinį. Šis veiksmas yra kartojamas ir atliekamas k kartų, kiekvieną kartą validacijai naudojant vis skirtingą duomenų rinkinį.

Atliekant tinklelinę paiešką buvo nagrinėjami du hiperparametrai: temų skaičius ir mokymosi greitį (angl. *learning rate*). Pasirinkti mokymosi greičio hiperparametrą yra sudėtinga, tačiau labai svarbu. Mažas pasirinktas greitis gali ženkliai prailgtinti modelio treniravimo procesą, dėl kurio procesas gali ir užstrigti. Tačiau pasirinkus per didelį treniravimosi greitį galime gauti nepakankamai tikslius rezultatus.

Ieškant geriausios hiperparametrų kombinacijos buvo pasirinkta nagrinėti 5,6,7,8,9 ir 10 temų skaičių. Tuo tarpu mokinimosi greičio reikšmė gali būti nustatoma intervale $[0, 1]$, todėl pasirinktos reikšmės buvo 0,5, 0,7 ir 0,9.



6 pav.: Log-tikėtinumo funkcijos reikšmės pagal temų skaičių ir mokinimosi greitį.

6 paveiksle pateiktos log-tikėtinumo funkcijos reikšmės pagal skirtingą temų skaičių ir mokinimosi greitį. Iš grafiko matyti, jog geriausia kombinacija yra, kai pasirenkamos 5 temos ir mokinimosi greičio hiperparametro reikšmė lygi 0,9. Tokias pačias hiperparametrų reikšmes pasiūlė ir pritaikyta `GridSearchCV` funkcija. Taigi, temos bus modeliuojamos pasirenkant šiuos hiperparametrų rodiklius.

	Tema1	Tema2	Tema3	Tema4	Tema5
1.	idėja	nulaužimas	padaryti	pastatyti	priemonė
2.	amatas	patarimas	pasukti	padaryti	atstatyti
3.	naudojimas	daryti	šviestuvas	mediena	išvaizda
4.	namai	gyvenimas	klijai	restauracija	plaukai
5.	vamzdelis	žinoti	peilis	medžio apdirbimas	makiažas
6.	cementas	perdarymas	stalas	namas	nagai
7.	pasidaryk pats	atkurti	transformuoti	vaza	oda
8.	patarimas	gudrybė	panaudoti	darbas	technika
9.	pramoga	įrankis	automobilis	paversti	menas
10.	sukurti	išmokti	procesas	plyta	šukuosena
11.	žvakė	namai	metalas	indas	ideja
12.	kava	transformacija	rūšiuoti	apdaila	tenkinti
13.	grožis	pataisyti	įrankis	lenkimas	pasikeitimas
14.	dekoras	statuti	cementas	kiemas	laidas
15.	mada	bandyti	mediena	rąstas	dizainas

3 lentelė: Žodžių matrica

4.4 Sudarytos temos

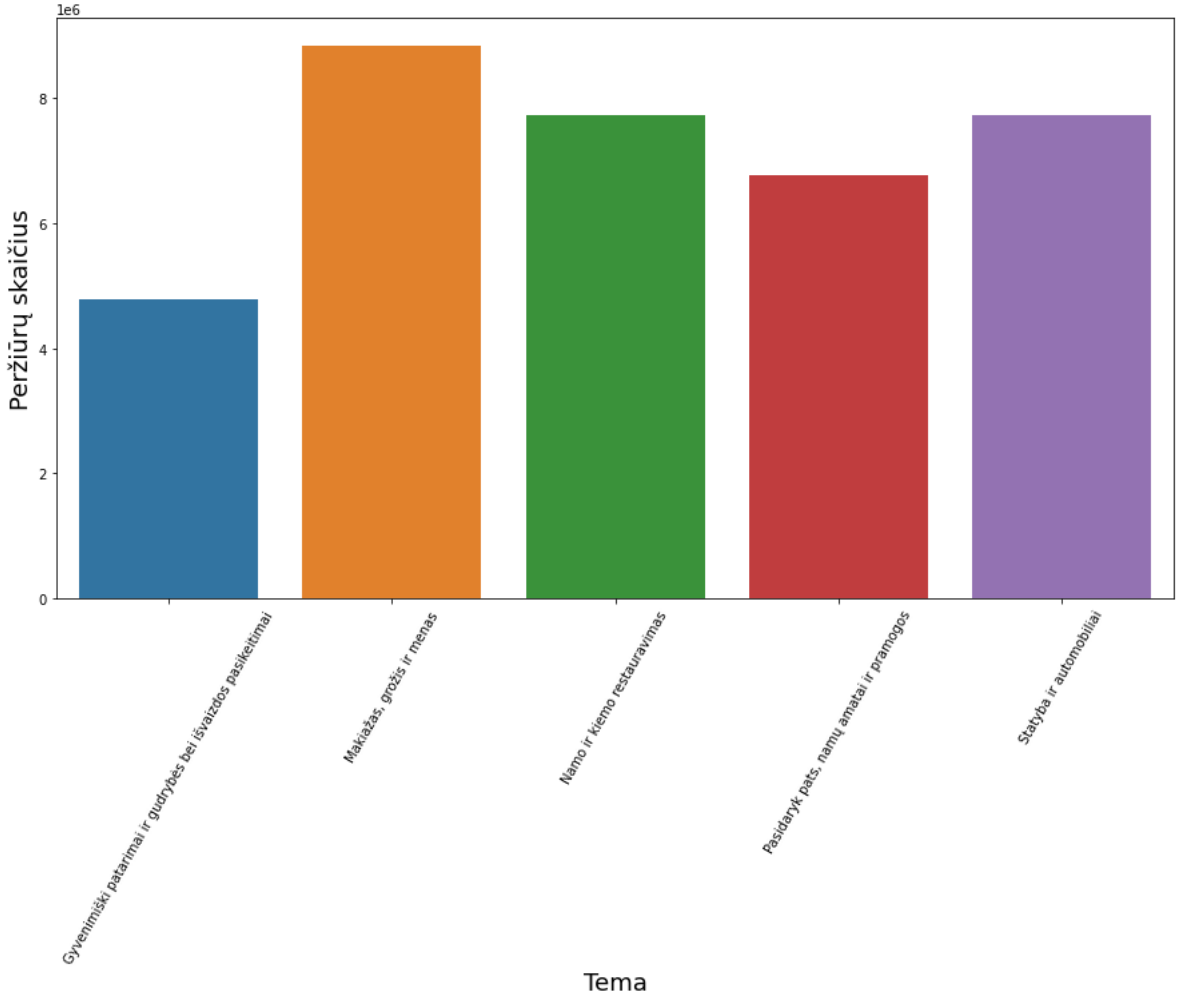
Suradus tinkamiausią hiperparamtrų kombinaciją ir pritaikius latentinio Dirichlė paskirstymo algoritmo modelį buvo gauta temų žodžių matrica. Gauti žodžiai yra išversti į lietuvių kalbą ir pateikti 3 lentelėje. Verčiant žodžius iš anglų kalbos, kai kurie angliški žodžiai negalėjo būti išversti panaudojant tik vieną lietuvišką žodį, todėl pasitaikė atvejų, kai lentelėje pateiktas vertimas buvo sudarytas iš daugiau nei vieno lietuviško žodžio. Iš kiekvienai temai priskiriamų raktinių žodžių buvo sudarytos galutinės apibendrintos temos: „Pasidaryk pats, namų amatai ir pramogos“, „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“, „Statyba ir automobiliai“, „Namo ir kiemo restauravimas“ bei „Makiažas, grožis ir menas“. Dėl patogumo, darbe šios temos taip pat bus įvardinamos atitinkamai kaip tema 1, tema 2, tema 3, tema 4, tema 5.

5 Keliamo turinio strategijos analizė

Šiame skyriuje bus analizuojamos keliamo turinio strategijos nagrinėjamuose kanaluose. Pirmiausia kiekvienas kanalas buvo išanalizuojamas pagal įkeliamus vaizdo įrašus sudarytomis temomis. Prieduose pateikti paveikslai iliustruoja šios analizės rezultatus (žr., 24, 25, 26 pav.). Pastebėta, jog „puslapis1“ pirmaisiais 2021 m. mėnesiais daugiausią vaizdo įrašų publikavo temomis: „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ bei „Statyba ir automobiliai“. Tokie rezultatai nestebina, nes kaip jau buvo išsiaiškina, šio kanalo didžiąją dalį auditorijos sudaro vyrai. Vis dėlto, kitais mėnesiais „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ tema publikuojamų vaizdo įrašų kiekis pradėjo mažėti ir padaugėjo daugiau keliamų vaizdo įrašų tema „Namo ir kiemo restauravimas“. Tuo tarpu „puslapis2“ labiausiai publikavo vaizdo įrašus „Makiažas, grožis ir menas“ tema. Tokio rezultato taip pat buvo tikimasi, nes didžiąją dalį visos žiūrinčios auditorijos sudaro moterys. „puslapis3“ puslapio pirmame 2021 m. ketvirtyje labiausiai dominavo „Gyvenimiški patarimai ir gudrybės bei išvaizdos

pasikeitimai“ tema, vėliau pradėta daugiau publikuoti vaizdo įrašų „Statyba ir automobiliai“ bei „Namo ir kiemo restauravimas“ temomis.

Toliau kanalai buvo analizuojami pagal kiekvienos temos vidutinį filmukų peržiūrų skaičių. Gauti „puslapis1“ ir „puslapis2“ kanalų analizės rezultatai buvo netikėti. Nors „puslapis1“ puslapyje „Statyba ir automobiliai“ bei „Namo ir kiemo restauravimas“ temomis publikuojami filmukai buvo populiarūs ir žiūrimi vartotojų, vis dėlto „Makiažas, grožis ir menas“ kuriami įrašai buvo žiūrimiausi. 7 paveiksle pateikti „puslapis1“ kanalo rezultatai. Taip pat netikėti rezultatai gauti ir išanalizavus „puslapis2“ puslapį. Čia populiariausi ir žiūrimiausi įrašai buvo „Namo ir kiemo restauravimas“ tema bei mažiausiai populiarumo sulaukė vaizdo įrašai, publikuoti „Makiažas, grožis ir menas“ tematika. Tuo tarpu, „puslapis3“ vaizdo įrašų populiarumas pagal temas ženkliai nesiskyrė, tačiau žiūrimiausi šio puslapio įrašai buvo „Pasidaryk pats, namų amatai ir pramogos“ tema. Šių dviejų kanalų grafikai pateikti prieduose (žr., 27, 28 pav.) Nors „puslapis1“ ir „puslapis2“ kanaluose atitinkamomis temomis yra publikuojama mažai vaizdo įrašų, vi dėlto analizės rezultatai indikuoja, jog kanalų moderatoriams vertėtų atsižvelgti į auditorijos susidomėjimą šiomis temomis.



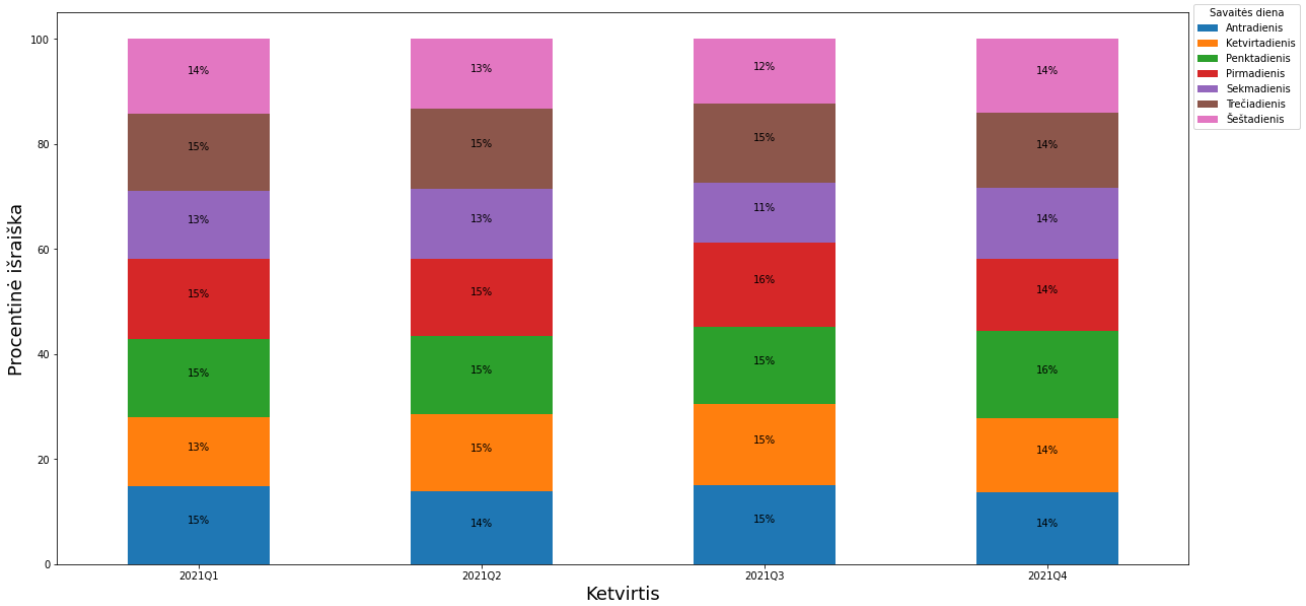
7 pav.: Vidutinis surenkamų peržiūrų skaičius pagal temas „puslapis1“ kanale.

Peržvelgus įkeliamų vaizdo įrašų pasiskirstymą sudarytomis temomis bei kiekvienos temos vidutinį filmukų peržiūrų skaičių, visi kanalai buvo analizuojami pagal tai, kaip žiūrinti auditorija mėgo sąveikauti su matomais vaizdo įrašais. Išsiaiškinta, jog „puslapis1“ kanale vartotojai ilgiausiai žiūrėdavo bei labiausiai mėgo reaguoti, dalintis ir komentuoti po „Namo ir kiemo restauravimas“ tematikos vaizdo įrašais. Tuo tarpu mažiausiai auditorijos įsitraukimo sulaukdavo „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ vaizdo įrašai. Šios tematikos filmukai išsiskyrė auditorijos pasyvumu. Tačiau svarbu paminėti, jog puslapio moderatoriai greičiausiai pastebėjo šią tendenciją ir į tai atsižvelgė, nes nagrinėjamu laikotarpiu vienos temos filmumų pradėta skelbti daugiau, o kitos - mažiau. „puslapis2“ kanale, kaip ir „puslapis1“, pasimatė tokios pačios tendencijos. Vartotojai labiausiai mėgo reaguoti, dalintis ir komentuoti po „Namo ir kiemo restauravimas“ tematikos vaizdo įrašais, o mažiausiai sąveikaudavo su „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ vaizdo įrašais. „puslapis3“ kanalo auditorija taip pat nevienodai sąveikaudavo su vaizdo įrašais: labiausiai mėgo komentuoti ir dalintis „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ vaizdo įrašais, tačiau daugiausiai reakcijų sulaukdavo „Namo ir kiemo restauravimas“ filmukai. Mažiausiai auditorijos įsitraukimo sulauktdavo „Makiažas, grožis ir menas“ skelbiami įrašai.

Taip pat buvo gilinamasi, kokios trukmės filmukai buvo kuriami kiekviena tematika. Pastebėta, jog „puslapis1“ kanale „Pasidaryk pats, namų amatai ir pramogos“, „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ tematikos filmukai buvo kuriami ilgesnės trukmės nei kitomis tematikomis. Vidutiniškai šie įrašai būdavo vidutinės trukmės. Nagrinėjamame „puslapis2“ kanale ilgiausi kuriami įrašai tema „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“. Šie filmukai vidutiniškai buvo kuriami vidutinio ilgio, tuo tarpu kitomis temomis kuriami filmukams buvo mažesnės trukmės. Išanalizavus „puslapis3“ rezultatus buvo pastebėta, kad „Pasidaryk pats, namų amatai ir pramogos“ bei „Makiažas, grožis ir menas“ įrašai publikuojami vidutinės trukmės, „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ filmukai kuriami ilgos trukmės, o likusių temų vaizdo įrašai - trumpos trukmės.

Galiausiai bandyta išsiaiškinti, kokiomis savaitės dienomis bei valandomis dažniausiai buvo publikuojamas turinys. Buvo nustatyta, jog turinys puslapiuose buvo keliamas įvairiomis valandomis, net ir naktį, todėl tam, kad galėtume lengviau nustatyti tendencijas, paros laikas buvo padalintas į 4 lygias dalis: naktį, rytą, dieną bei vakarą. Nustatyta, kad visuose puslapiuose turinys buvo publikuojamas kiekvieną dieną. 8 paveiksle pateikti „puslapis1“ kanalo rezultatai. Nors iš pateikto grafikų nėra aiškiai matyti, jog penktadieniais buvo vidutiniškai daugiausia skelbiama vaizdo įrašų, peržiūrėjus realias skaitines reikšmes buvo pastebėtas nedidelis skirtumas. Išsiaiškinta, jog penktadieniais puslapis vidutiniškai daugiausiai publikavo filmukų. Tuo tarpu mažiausiai buvo skelbiama savaitgaliais. Taip pat pastebėta, jog filmukai publikuojami skirtingomis valandomis. „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ vaizdo įrašai buvo skelbiami dienos metu, kol kitų tematikų įrašai buvo publikuojami ryte. „puslapis2“ kanale vidutiniškai daugiausiai įrašų buvo skelbiama ketvirtadieniais ir mažiausiai savaitgaliais bei pirmadieniais. Šiame kanale įrašai taip pat buvo skelbiami ne vienodomis valandomis. „Pasidaryk pats, namų amatai ir pramogos“, „Statyba ir automobiliai“ bei „Namo ir kiemo restauravimas“ tematikos filmukai buvo skelbiami rytais, „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“ - ryte ir per pietūs, o „Makiažas, grožis ir menas“ - rytais ir vakarais. Peržvelgus „puslapis3“ puslapio skelbiamų įrašų laiką pastebėta, kad visi vaizdo įrašai buvo publikuojami kiekvie-

nos dienos rytą, tačiau daugiausiai publikuojama įrašų ketvirtadieniais ir penktadieniais, o mažiausiai - savaitgalį.



8 pav.: Savaitės dienomis publikuojamų įrašų pasiskirstymas „puslapis1“ kanale.

Yra matomas aiškus vaizdas, jog pasirinktas publikavimo laikas bei pasirinkimas publikuoti mažiau vaizdo įrašų savaitgaliais yra neatsitiktinis. Vienas iš paaiškinimų, kodėl didžioji dauguma vaizdo įrašų publikuojami rytai, yra tai, jog žmonės dažnai mėgsta prieš pradedant darbus ar valgant pusryčius naršyti socialiniuose tinkluose. Dėl šios priežasties rytais susidaro didesnis srautas vartotojų, kurie gali pamatyti publikuojamą turinį [51]. Taip pat savaitgaliais žmonės dažniau praleidžia laiką užsiimdami norima veikla ar išvykdami iš namų, todėl šiomis dienomis vartotojų srautas gali būti mažesnis.

Apibendrinant galima teigti, kad dominuojančių temų kiekviename puslapyje rezultatai nenustebino ir buvo to tikimasi. Tačiau įrašų žiūrimumas kiekviena tematika buvo netikėtas ir atskleidė, jog kanaluose dominuojančios temos nebūtinai yra žiūrimiausios ir mėgstamiausios puslapiuose apsilankančių vartotojų. Taip pat atskleista, kad kiekviename kanale filmukai buvo skelbiami kiekvieną diena, tačiau skelbiamas įrašų kiekis nevienodas ir nevisada skelbiama vienodomis valandomis. Galiausiai išsiaiškinta, jog kiekviename kanale auditorijos labiausiai sąveikaudavo su „Namo ir kiemo restauravimas“ tematikos vaizdo įrašais. Šis pastebėjimas atskleidžia, jog šios tematikos vaizdo įrašai yra labiausiai įtraukiantys ir sudominantys žiūrovus.

5.1 Hipotezių tikrinimas

Atlikus puslapių keliamo turinio strategijos analizę buvo gauti pastebėjimai bei tam tikri bruožai, atsispindintys strategijose. Dėl šios priežasties norima patikrinti, ar gautos įžvalgos yra reikšmingos. Tam pasitelksime statistinių hipotezių tikrinimą. Šiame skyrelyje patikrinsime, ar kanaluose yra reikšmingas peržiūrų skirtumas tarp temų, ar skirtingų temų vaizdo įrašai sulaukia nevienodą kiekį komentarų, pasidalinimų ir mėgti mygtukų paspaudimų. Taip pat sieksime išsiaiškinti, ar skirtingu paros metu skelbiami įrašai sulaukia skirtingą kiekį peržiūrų. Galiausiai išsiaiškinsime, ar savaitės dienomis papub-

likuoti vaizdo įrašai sulaukia skirtingą kiekį peržiūrų bei interakcijų iš vartotojų. Hipotezių tikrinimui bus naudojami Kruskal-Wallis testas bei Dunn testas.

Pirmiausia buvo patikrinta, ar kiekviename kanale reikšmingai skiriasi temų žiūrimumas. „puslapis1“ atveju gauta p-reikšmė buvo lygi 8.69415e-13, „puslapis2“ - 0.00067, „puslapis3“ - 1.41395e-06. Kadangi visais kanalų atvejais gautos p-reikšmės buvo mažesnės už 0.05, nulinė hipotezė visais atvejais buvo atmesta. Dėl šios priežasties toliau buvo taikytas Dunn testas, kurio metu siekta išsiaiškinti, kurios temos yra labiausiai žiūrimos kiekviename kanale. Priedų skiltyje pateiktose 10, 11, 12 lentelese yra matomi Dunn testo rezultatai kiekvieno kanalo atveju. Išanalizavus rezultatus pastebėta, jog „puslapis1“ kanale temos pasiskirstė į dvi grupes: 1 ir 2 temų žiūrimumas statistiškai nesiskyrė, ta pati tendencija pastebėta ir tarp 3, 4 bei 5 temų. Analizuojant šio puslapio vaizdo įrašų žiūrimumą pagal temas buvo pastebėta, jog 3, 4 bei 5 temų vidutinis peržiūrų kiekis buvo aukštesnis už likusias temas, todėl galima daryti išvadą, jog šiame kanale labiausiai žiūrimi yra 3, 4 ir 5 temų vaizdo įrašai. „puslapis2“ atveju buvo pastebėta, jog lyginant su 3 ir 4 temomis, 5 temos vaizdo įrašai sulaukia statistiškai reikšmingai skirtingo kiekio peržiūrų. Pasitelkiant anksčiau gautais pastebėjimais, jog 5 temos įrašai vidutiniškai mažiausiai sulaukia peržiūrų galime teigti, jog šios tematikos įrašai sulaukia mažiausiai peržiūrų. Galiausiai peržvelgus „puslapis3“ rezultatus nustatyta, kad lyginant su 1, 2 ir 5 temomis, 3 temos įrašai surenka reikšmingai skirtingą kiekį įrašų. Pasitelkus testo rezultatus ir ankstesnius pasteb4jimus galima manyti, jog šios temos įrašai yra mažiausiai žiūrimi šiame kanale.

Toliau buvo tikrinama, ar visų tematikų vaizdo įrašai sulaukia tokio pačio „Facebook“ vartotojų interakcijų kiekio. Kadangi praeitame skyriuje buvo pastebėta, jog beveik visuose kanaluose „Namo ir kiemo restauravimas“ tematikos fillmukai surenka daugiausiai komentarų, pasidalinimų bei mėgti mygtukų paspaudimų, šios hipotezės tikrinimas buvo atliekamas bendrai visiems kanalams. Kad rezultatus būtų paprasčiau palyginti, vaizdo įrašų komentarai, pasidalinimai bei mėgti mygtukų paspaudimai buvo sudėti į bendrą interakcijų kiekį. Gauta Kruskal-Wallis testo p-reikšmė buvo mažesnė už reikšmingumo lygmenį ir lygi 6.61141e-53, todėl nulinė hipotezė buvo atmesta ir taikytas Dunn testas. Šio testo rezultatai pateikti 4 lentelėje. Rezultatai atskleidė, jog „Namo ir kiemo restauravimas“ tematikos įrašų sulaukiamas vartotojų interakcijų kiekis statistiškai reikšmingai skiriasi nuo kitų tematikų vaizdo įrašų. Todėl remiantis šio testo rezultatais ir ankstesniais pastebėjimais galima teigti, jog šios tematikos įrašai sulaukia daugiausiai žiūrovų interakcijų.

Temos nr.	1	2	3	4	5
1.	1.000000e+00	1.000000e+00	1.528835e-11	4.432059e-20	7.087965e-03
2.	1.000000e+00	1.000000e+00	6.172530e-17	9.037141e-27	2.046853e-01
3.	1.528835e-11	6.172530e-17	1.000000e+00	1.869447e-02	1.989788e-24
4.	4.432059e-20	9.037141e-27	1.869447e-02	1.000000e+00	8.507291e-35
5.	7.087965e-03	2.046853e-01	1.989788e-24	8.507291e-35	1.000000e+00

4 lentelė: Dunn testo rezultatai. Lentelėje pateikiami p-reikšmės įverčiai.

Patikrinus vaizdo įrašų tematikų žiūrimumą ir vartotojų interakcijų kiekius, buvo tiriama, ar filmukų peržiūrų kiekis skiriasi, kai jie yra publikuojami skiringu paros metu. Tam, kad rezultatus būtų lengviau interpretuoti, paros laikas buvo padalintas į 4 lygias kategorijas: naktį, rytą, dieną bei vakarą. Atlikus Kruskal-Wallis testą gauta, jog p-reikšmė yra 2.56648e-14, kuri yra mažesnė už reikšmingumo lygmenį,

todėl nulinę hipotezę galima buvo atmesti. Pasinaudojus Dunn testu buvo gauti rezultatai, kurie pateikti lentelėje (žr., 13 lent.). Rezultatai atskleidė, jog peržiūros statistiškai reikšmingai nesiskiria tarp vaizdo įrašų, kurie buvo paskelbti rytais ir vakarais, taip pat dienos metu ir vakare, tačiau reikšmingai skiriasi tarp publikuotų filmukų rytais ir dienos metu. Remiantis šiais rezultatais ir ankstesniai pastebėjimais, galime manyti, jog įrašai daugiausia peržiūrų sulaukia, kai yra publikuojami rytais ir vakarais.

Galiausiai buvo tikrinama, ar vaizdo įrašai sulaukia vienodo vartotojų žiūrėjimo masto bei interakcijų skaičiaus, kai yra publikuojami skirtingomis savaitės dienomis. Vartotojų peržiūroms atlikto Kruskal-Wallis testo gauta p-reikšmė buvo lygi 0.57922, o atlikus testą vartotojų interakcijoms - 0.55233. Abiem atvejais p-reikmšė buvo aukstesnė už reikšmingumo lygmenį, todėl negalima buvo atmesti nulinės hipotezės. Testas parodė, jog vaizdo įrašai sulaukia vienodo vartotojų peržiūrų skaičiaus, taip pat interakcijų skaičiaus, kai yra publikuojami skirtingomis savaitės dienomis.

Šiame skyrelyje buvo atliktas hipotezių tikrinimas. Gauti rezultatai parodė, jog „puslapis1“ kanale labiausiai žiūrimi yra 3, 4 ir 5 temų vaizdo įrašai, „puslapis2“ puslapyje mažiausiai peržiūrų sulaukia 5 temos vaizdo įrašai, o „puslapis3“ kanale mažiausiai sulaukia 3 tematikos filmukai. Taip pat įsitikinta, kad 4 tematikos įrašai kanaluose sulaukia daugiausiai vartotojų interakcijų, o daugiausiai įrašai peržiūrų surenka tada, kai yra publikuojami ryte arba vakare. Galiausiai išsiaiškinta, jog įrašų publikavimas skirtingomis savaitės dienomis neturi įtakos vaizdo įrašų peržiūroms ir sulaukiamam vartotojų interakcijų kiekiui.

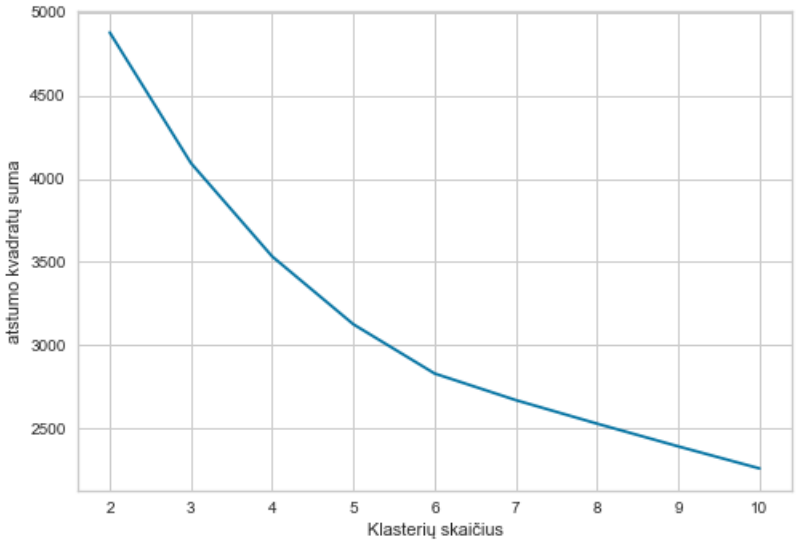
6 Kiekvieno kanalo vaizdo įrašų klasterizavimas pagal požymių rinkinius

Šiame skyriuje pateikiami vaizdo įrašų klasterizavimo (angl. *clustering*) rezultatai. Visų kanalų vaizdo įrašai buvo klasterizuojami naudojantis K-vidurkių bei hierarchiniu klasterizavimo metodu. Kiekvienas klasterizavimo metodas buvo taikomas taip: pirma suklasterizuojami visi kanalų įrašai sujungus du požymių rinkinius, vėliau klasterizavimas taikomas dviems požymių rinkiniams atskirai. Galiausiai buvo klasterizuojami kiekvieno kanalo vaizdo įrašai su kiekvienu požymių rinkiniu. Pirmame požymių rinkinyje buvo įtraukti populiarumo matai (vaizdo įrašų parodymų ir paspaudimų ant jo skaičių santykis, įrašo peržiūrų skaičius, vidutinis žiūrėjimo laikas sekundėmis, surinktų komentarų, mėgti mygtukų bei pasidalinimų skaičius) ir sudarytos vaizdo įrašų tematikos. Antrąjį požymių rinkinį sudarė vaizdo įrašų kokybės matai (filmuko trukmė minutėmis, įrašo publikavimo savaitės diena bei laikas ir vaizdo įrašo tematika) ir peržiūrų skaičius. Taikant klasterizavimo metodus skirtingiems požymių rinkiniams buvo siekta pasižiūrėti, ar klasterizavimo tendencijos skiriasi, kai visi filmukai yra klasterizuojami pagal sujungtų požymių rinkinį ir atskirus požymių rinkinius. Taip pat pritaikant metodus pirmam požymių rinkiniui buvo siekta patikrinti, kaip kiekviename kanale tematikos grupuojasi pagal populiarumą. Tuo tarpu naudojant klasterizavimo metodus antram požymių rinkiniui siekta patikrinti, ar populiariems vaizdo įrašams būdingi kokie nors nagrinėjami įrašo kokybės matai.

6.1 Vaizdo įrašų grupavimas K-vidurkių metodu

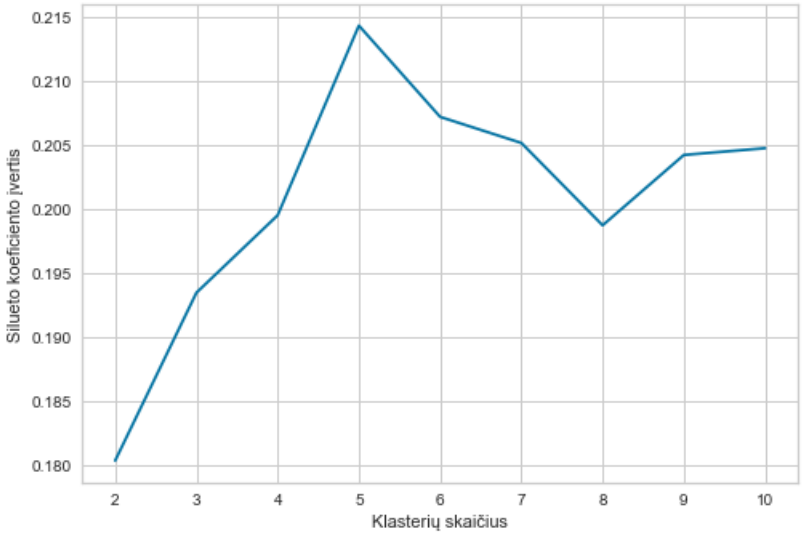
Pradiniame etape buvo klasterizuojami visi vaizdo įrašai. Pirmiausia įrašai buvo klasterizuojami pagal sujungtą požymių rinkinį, vėliau pagal kiekvieną rinkinį atskirai. Prieš atliekant grupavimą, kiekvieno požymių rinkinį reikėjo paruošti metodo taikymui. Kategoriniams kintamiesiems buvo priskirtos skaitinės reikšmės, taip pat požymių rinkiniams buvo atliktas normavimas. Atliekant duomenų klasterizavimą, normavimas dažnai taikomas kaip išankstinio duomenų apdorojimo veiksmas. Dažnai analizėse naudojamų kintamųjų reikšmių skalės skiriasi, dėl to klasterių analizės rezultatai gali būti iškreipti ir netikslūs. Šią problemą padeda išspręsti duomenų normavimas. Atliekant normavimą, visų požymių reikšmių skalės yra suvienodinamos, padarant požymių skirstinius panašius į Gauso skirstinį, su vidurkiu 0 ir dispersija 1. Normavimas buvo atliekamas pritaikant `StandardScaler` funkciją iš `scikit-learn` bibliotekos [52]. Paruošus duomenis, buvo ieškomas optimalus klasterių skaičius. Klasteriai buvo randami naudojant `KMeans` funkciją iš `scikit-learn` bibliotekos. Optimalus klasterių skaičius apskaičiuotas naudojant alkūnės bei silueto metodus, kurie aprašyti klasterizavimo skyrelije. Silueto koeficientai buvo skaičiuojami panaudojant `silhouette_score` funkciją iš `scikit-learn` bibliotekos [53].

Suklasterizavus įrašus pagal jungtinį požymių rinkinį ir atskirus požymių rinkinius, buvo surasta, jog jungtinio požymių rinkinio bei pirmojo požymių rinkinio atvejais optimalus klasterių skaičius yra 5, o antrojo požymių rinkinio atveju - 8 klasteriai. 9 ir 10 paveiksluose pateikti alkūnės bei silueto metodų gauti rezultatai, kai klasterizavimas buvo taikomas jungtiniam požymių rinkiniui. Atskirų požymių rinkinių klasterizavimo rezultatai pateikti priedų skyriuje.



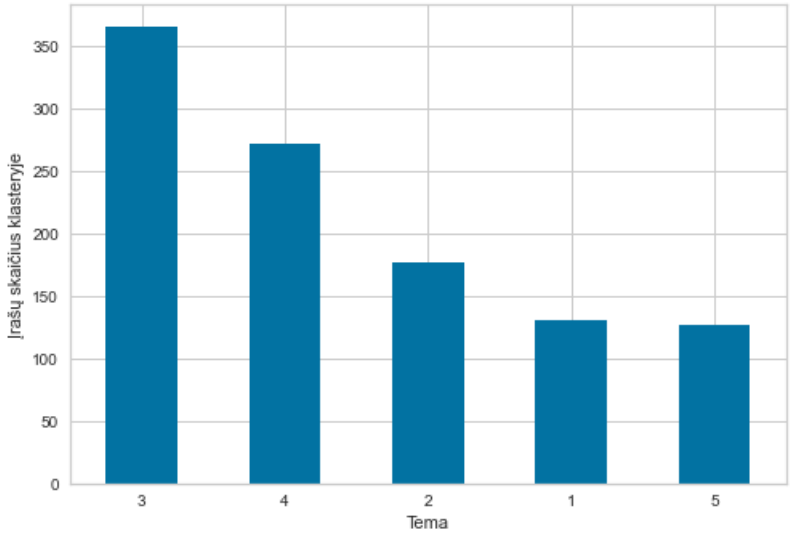
9 pav.: Ieškomas optimalus klasterių skaičius naudojant alkūnės metodą. Klasterizuojami visų kanalų filmukai pagal jungtinius požymių rinkinius.

Išsiaiškinta, jog klasterizavimo tendencijos nesiskyrė klasterizuojant bendrą požymių rinkinį bei požymių rinkinius atskirai. Visais atvejais pastebėta, jog tarp sudarytų klasterių yra klasteris, kuriame surenkami visi populiariausi vaizdo įrašai. Pradėjus analizuoti šį klasterį išsiaiškinta, jog jame labiausiai dominuoja 3 ir 4 temų vaizdo įrašai. 11 paveikslas parodo, kiek kokios tematikos vaizdo įrašų buvo surinktų į klasterį, kuriame buvo surinkti populiariausi vaizdo įrašai. Čia klasterizavimas buvo atlieka-



10 pav.: Ieškomas optimalus klasterių skaičius naudojant silueto koeficientą. Klasterizuojami visų kanalų filmukai pagal jungtinius požymių rinkinius.

mas bendram požymių rinkiniui. Taip pat prieduose pateikiama lentelė (žr., 14 lent.), kuri apibūdiną šio klasterio požymių statistinę informaciją.



11 pav.: Klasteryje surinktų vaizdo įrašų išsidėstymas pagal temas. Pažymėjimai: 1 - „Pasidaryk pats, namų amatai ir pramogos“, 2 - „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“, 3 - „Statyba ir automobiliai“, 4 - „Namo ir kiemo restauravimas“, 5 - „Makiažas, grožis ir menas“.

Toliau kiekvieno kanalo filmukai buvo klasterizuojami atskirai pagal du aukščiau paminėtus požymių rinkinius. Dėl paprastesnio rezultatų palyginimo, visais atvejais buvo pasirinkta sudaryti 5 klasterius. Pirma, kiekvieno kanalo filmukai buvo klasterizuojami pagal pirmąjį požymių rinkinį. Visų kanalų atvejais didžiausias dėmesys buvo skirtas klasteriui, kuriame buvo sugrupuoti populiariausi vaizdo įrašai. Buvo pastebėti šie įdomūs rezultatai:

1. „puslapis1“ - Lyginant su ankstesniais gautais klasterizavimo rezultatais, nebuvo pastebėta jokių

tendencijų pasikeitimų. Klasteryje, kuriame buvo surinkti populiariausi kanalo vaizdo įrašai, didžiausią surinktų filmukų dalį sudarė 3 ir 4 temų vaizdo įrašai.

2. „puslapis2“ - Analizuojant šio kanalo klasterį pastebėta populiarumo tendencijos pasikeitimų. Klasteryje, kuriame buvo surinkti populiariausi kanalo filmukai, didžiausią įrašų dalį sudarė 2 ir 5 temų vaizdo įrašai.

3. „puslapis3“ - analizuojant populiariausių filmukų klasterį pastebėtas dalinis tendencijos pasikeitimas. Tarp populiariausių surinktų filmukų didžiausią dalį sudarė tik 3 temos vaizdo įrašai.

Išanalizavus populiariausių vaizdo įrašų klasterius pagal pirmąjį požymių rinkinį, buvo atlikta filmukų klasterizaciją pagal kokybės matus ir surinktas peržiūras. Išanalizavus kiekvieno kanalo populiariausių įrašų klasterius, pastebėti šie rezultatai:

1. „puslapis1“ - Suklasterizavus ir peržvelgus populiariausių vaizdo įrašų klasterį nebuvo pastebėta jokių pokyčių, lyginant su ankstesniais gautais rezultatais. Nagrinėjamame klasteryje labiausiai dominavo 3 ir 4 temų vaizdo įrašai.

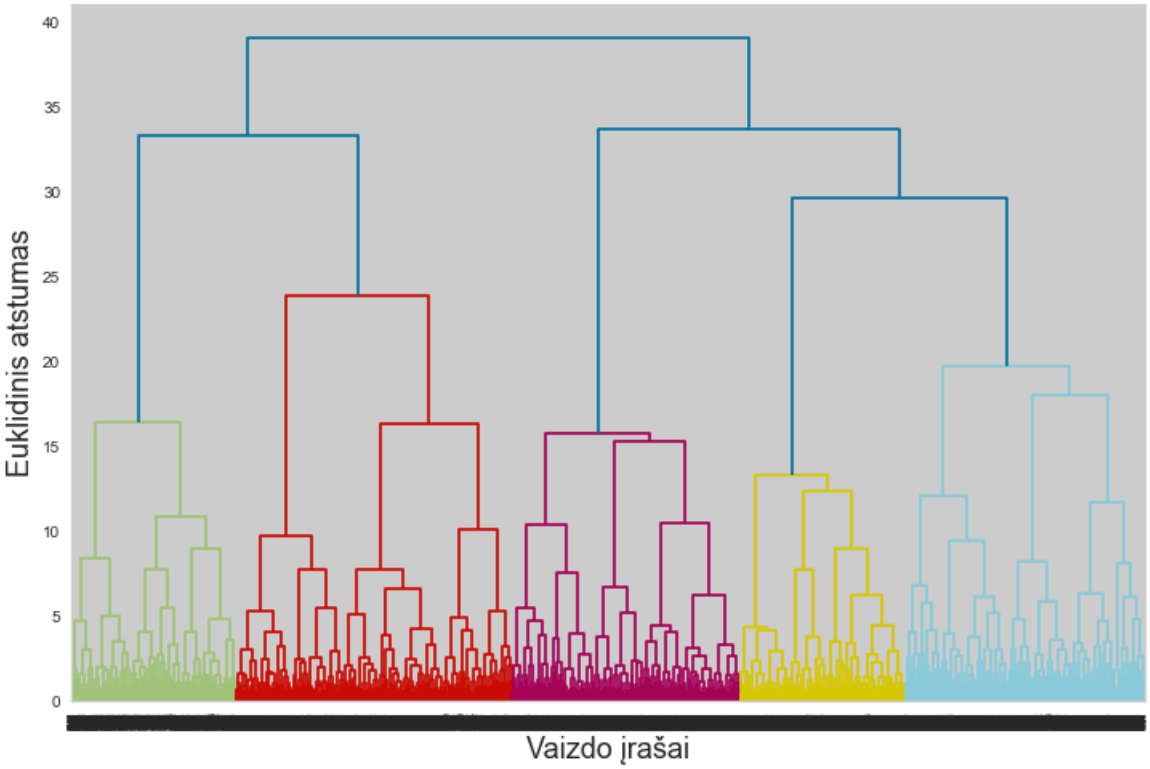
2. „puslapis2“ - pastebėta, jog dominuojančios temos skiriasi nuo ankstesnių rezultatų. Klasteryje, kuriame buvo pateikti populiariausi kanalo vaizdo įrašai, didžiausią surinktų filmukų dalį sudarė 2 ir 3 temų vaizdo įrašai.

3. „puslapis3“ - Tarp populiariausių sugrupuotų įrašų didžiausią klasterio dalį sudarė 3 ir 4 temos vaizdo irasai.

Atlikus klasterizaciją K-vidurkių metodu aiškiai matyti, jog tarp „puslapis1“ kanalo populiariausių filmukų labiausiai dominavo 3 ir 4 temų vaizdo įrašai. „puslapis3“ kanalo populiariausiuose filmukuose labiausiai dominavo 3 tematikos įrašai. „puslapis2“ kanale pamatyta tendencija, jog daugiausiai populiariausių filmukų buvo 2 tematikos, tačiau taip pat nemaža dalis populiariausių įrašų buvo 5 bei 3 tematikos. Atlikus klasterizacija nebuvo aiškiai pamatyta konkreti savaitės diena, laikas ar filmuko trukmė, kuri būtų būdinga populiariems vaizdo įrašams.

6.2 Vaizdo įrašų grupavimas hierarchiniu klasterizavimo metodu

Prieš atliekant klasterizavimą hierarchiniu klasterizavimo metodu, kiekvieną požymių rinkinį taip pat reikėjo paruošti metodo taikymui. Paruošimo veiksmai buvo atlikti tokie patys, kaip ir klasterizuojant duomenis K-vidurikių būdu. Kategoriniams kintamiesiems buvo priskirtos skaitinės reikšmės, taip pat požymių rinkiniams buvo atliktas normavimas. Normavimui atlikti buvo pritaikyta `StandardScaler` funkciją iš `scikit-learn` bibliotekos. Paruošus duomenis buvo bandoma surasti optimalų kiekį klasterių. Pirmiausia buvo nubrėžiamos dendrogramos jungtiniam požymių rinkiniui, po to kiekvienam požymių rinkiniui atskirai. Šiuo etapu siekta surasti ir grafiškai pavaizduoti klasterių tarpusavio priklausomybių struktūra. Dendrogramos buvo nubrėžiamos naudojant funkciją `dendrogram` iš `SciPy` bibliotekos [54]. 12 paveiksle pateikta klasterių tarpusavio priklausomybių struktūra, kai klasterizuojami visi kanalų įrašai pagal sujungtą požymių rinkinį. Atskirų požymių rinkinių dendogramos pateiktos priedų skyriuje.



12 pav.: Dendrograma, sukurta pagal jungtinių požymių rinkinį.

Vizualizavus ir peržvelgus kiekieno atvejo klasterių priklausomybes ir išsidėstymus, galiausiai buvo nustatytas optimalus kiekis klasterių jungtiniam požymių rinkiniui ir atskiriems požymių rinkiniams. Nustatyta, jog jungtinių požymių atveju optimalus klasterių skaičius yra 5, pirmojo požymio atveju - 4 klasteriai, o antrojo - 3 klasteriai. Tam, kad jungtinių požymių ir atskirų požymių rezultatus būtų lengviau palyginti, atskiriems požymiams buvo pakartotos klasterizacijos su 5 klasteriais. Taip pat buvo nustatyta, jog taikant 5 klasterius atskiriems požymiams, sudarytuose klasteriuose esančios tendencijos nepasikeičia ir išlieka tokios pačios, todėl lyginimo rezultatai, kai visais atvejais buvo taikomi 5 klasteriai, smarkiai nesuprastėjo. Nustačius optimalius klasterių kiekius buvo panaudota `AgglomerativeClustering` funkcija iš `scikit-learn` bibliotekos ir suklasterizuoti vaizdo įrašai [55]. Kaip ir K-vidurkių klasterizavimo atveju, didžiausias dėmesys buvo skiriamas klasteriams, kuriuose buvo sugrupuojami populiariausi vaizdo įrašai.

Suklasterizavus visus vaizdo įrašus pagal jungtinį požymių rinkinį ir išanalizavus kiekvieną kintamąjį pastebėta, jog populiariausių vaizdo įrašų klasteryje labiausiai dominuoja 3 ir 4 temų vaizdo įrašai. Dar pastebėta, jog šiame klasteryje daugiausiai priskiriami filmukai, kurie yra 3 min. ir virš 20 min. trukmės. Populiariausių vaizdo įrašų klasterje surinkti įrašai mažiausiai buvo publikuojami savaitgaliais, taip pat daugiausiai įrašų paskelbtų 7-9 val. ryto. 5 lentelėje pateikiama statistinė informacija, apibūdinanti jungtinių požymių rinkinio klasterį, kuriame surinkti populiariausi filmukai.

Toliau buvo atliktas klasterizavimas visiems vaizdo įrašams pagal pirmąjį požymių rinkinį. Lyginant su jungtinio požymio rinkinio rezultatais, populiariausių filmukų klasteryje nebuvo pastebėta jokių tendencijos pasikeitimų. Šiame klasteryje didžiausią surinktų filmukų dalį sudarė 3 ir 4 temų vaizdo

	Minimumas	Maksimumas	Vidurkis	Mediana	Stand. nuok.
Truk. min	1.000	480.000	14.829	3.000	66.421
PR	0.060	26.050	1.458	0.655	2.121
Per.	4.000e+02	2.571918e+08	1.437499e+07	4.988026e+06	2.483609e+07
Vid. žiūr.	7.480	79.150	33.062	32.485	9.228
Kom.	0.000	193974	3285.158	400.500	11506.531
Mėg.	3.000	4.626729e+06	1.091261e+05	2.352650e+04	2.871582e+05
Pasidal.	0.000	895894.000	16352.573	1922.500	55281.547
Valanda	0.000	23.000	7.938	8.000	3.946
Sav. d.	1.000	7.000	3.325	3.000	1.802
Tema	1.000	5.000	3.115	3.000	1.133

5 lentelė: Statistinių parametrų lentelė, apibūdinanti jungtinių požymių rinkinio klasterį, kuriame surinkti populiariausi įrašai. Pažymejimai: Truk. min - filmuko trukmė minutėmis, PR - vaizdo įrašų parodymų ir paspaudimų ant jo skaičių santykis, Per.- įrašo peržiūrų skaičius, Vid. žiūr. - vidutinis žiūrėjimo laikas sekundėmis, Kom. - komentarų skaičius, Mėg. - mėgti mygtukų skaičius, Pasidal. - pasidalinimų skaičius, Valanda - įrašo publikavimo valanda, Sav. d. - įrašo publikavimo savaitės diena, Tema - vaizdo įrašo tematika.

įrašai.

Suklasterizavus kiekvieno kanalo vaizdo įrašus pagal antrąjį požymių rinkinį ir išanalizavus populiariausių filmukų klasterį pastebėta, kad kaip ir ankstesniuose rezultatuose didžiausią surinktų filmukų dalį sudarė 3 ir 4 temų vaizdo įrašai. Taip pat labiausiai dominuoja įrašai, esantys 3 min. trukmės. Galiausiai pastebėta, jog populiariausi vaizdo įrašai mažiausiai publikuojami savaitgaliais, o populiariausi įrašai daugiausiai skelbiami 7-9 val.

Galiausiai kiekvieno kanalo filmukai buvo klasterizuojami atskirai pagal atskirus požymių rinkinius. Dėl paprastesnio rezultatų palyginimo, visais atvejais buvo pasirinkta sudaryti 5 klasterius. Atlikus vaizdo įrašų klasterizaciją pirmajam požymių rinkiniui kiekviename nagrinėjame puslapyje ir peržvelgus klasterius, kuriuose sugrupuoti populiariausi įrašai, pastebėti šie rezultatai:

1. „puslapis1“ - populiariausių įrašų klasteryje didžiausią dalį sudarė 3 ir 4 temų vaizdo įrašai.

2. „puslapi2“ - analizuojant šio kanalo klasterį pastebėta tendencijos pasikeitimų. Klasteryje, kuriame buvo pateikti populiariausi kanalo vaizdo įrašai, didžiausią dalį sudarė 2 temos filmukai.

3. „puslapis3“ - Tarp populiariausių sugrupuotų filmukų didžiausią dalį sudarė tik 3 temos vaizdo įrašai.

Peržvelgus, kaip kiekviename kanale klasterizuojami vaizdo įrašai pagal pirmą požymių rinkinį, buvo atlikta filmukų klasterizaciją pagal kokybės matus ir surinktas peržiūras. Išanalizavus kiekvieno kanalo populiarių filmukų klasterius buvo pastebėta:

1. „puslapis1“ - Šio kanalo populiariausių vaizdo įrašų surinktame klasteryje labiausiai dominavo 3 ir 4 temų vaizdo įrašai, taip pat didžiausią dalį įrašų sudaro 3 min. trukmės filmukai. Pastebėta, jog daugiausiai įrašų paskelbta 7-9 val. ryto, tačiau mažiausiai įrašų publikuojama savaitgaliais.

2. „puslapis2“ - Klasteryje, kuriame buvo pateikti populiariausi kanalo vaizdo įrašai, didžiausią surinktų filmukų dalį sudarė 1 ir 2 temų vaizdo įrašai. Taip pat daugiausiai priskiriami įrašai, kurie yra 3 min. arba virš 20 min. trukmės. Dar išsiaiškinta, kad tarp populiariausių filmukų, mažiausiai jų buvo papublikuota savaitgaliais. Galiausiai nustatyta, jog daugiausiai šiame klasteryje esančių įrašų paskelbti 7-9 val. ryto.

3. „puslapis3“ - Šio kanalo populiariausių vaizdo įrašų surinktame klasteryje labiausiai dominavo 4 temos vaizdo įrašai, taip pat filmukai, kurie yra 3 min. trukmės. Mažiausią populiarių įrašų dalį sudarė filmukai, papublikuoti savaitgaliais. Galiausiai pastebėta, jog daugiausiai šio klasterio įrašų paskelbti 7-9 val. ryto.

Atlikus vaizdo įrašų klasterizavimą hierarchiniu metodu aiškiai matyti, jog tarp „puslapis1“ ir „puslapis3“ kanalų populiariausių filmukų labiausiai dominavo 3 ir 4 temų vaizdo įrašai. „puslapis2“ kanale pamatyta tendencija, jog daugiausiai populiariausių filmukų buvo 2 tematikos filmukai, tačiau taip pat nemaža dalis populiariausių įrašų buvo 1 tematikos. Taip pat pamatyta, jog populiariausių vaizdo įrašų klasteriuose dominuoja įrašai, paskelbti darbo dienomis. Galiausiai nustatyta, jog daugiausiai populiarių įrašų yra skelbiami 7-9 val. ryto.

Šiame skyriuje buvo atliktas vaizdo įrašų klasterizavimas K-vidurkių ir hierarchiniu metodu. Kiekvienas klasterizavimo metodas buvo taikomas visiems kanalų filmukams sujungus du požymių rinkinius, taip pat ir atskiriems rinkiniams. Galiausiai buvo klasterizuojami kiekvieno kanalo vaizdo įrašai pagal abu požymių rinkinius. Žvelgiant į gautus metodų rezultatus galima manyti, jog hierarchiniu metodu atliktas klasterizavimas atskleidė daugiau populiariam turiniui būdingų bruožų. Vis dėlto, abu klasterizavimo metodai atskleidė, jog tarp „puslapis1“ kanalo populiariausių filmukų labiausiai dominavo 3 ir 4 temų vaizdo įrašai, „puslapis2“ - 2 tematikos filmukai, „puslapis3“ - 3 tematikos įrašai. Taip pat hierarchinio klasterizavimo metodo rezultatai indikavo, jog populiariausi vaizdo įrašai daugiausia yra skelbiami darbo dienomis bei 7-9 val. ryto.

Lyginant K-vidurkių bei hierarchiniu metodu gautus klasterizavimo rezultatus ir rezultatus, gautus po hipotezių tikrinimo, pastebėta sutapimų ir prieštaravimų. Tiek po hipotezių tikrinimo, tiek po klasterizavimo buvo matyti, jog „puslapis1“ kanale labiausiai žiūrimi yra 3 ir 4 temų vaizdo įrašai. Taip pat klasterizavimo rezultatai paantrina išvadą, jog vaizdo įrašai daugiausiai sulaukia peržiūrų, kai yra publikuojami rytais. Nors hierarchinio klasterizavimo metodo rezultatai indikavo, kad darbo dienomis skelbiami vaizdo įrašai daugiausiai sulaukia peržiūrų, hipotezių tikrinimo metu šis teiginys buvo paneigtas.

7 Išvados

Dėl didelio susidomėjimo ir galimybių, socialiniai tinklai sparčiai populiarėja ir jų naudojimasis tampa neatsiejama žmonių gyvenimo dalimi. Nuo šios technologijos atsiradimo ženkliai pasikeitė daugumos žmonių kasdienybė - kaip yra bendraujama su kitais, kaip sužinoma, kokie svarbūs įvykiai įvyko ar vyksta visame pasaulyje. Šiame tyrime buvo susipažinta su socialinių tinklų samprata bei socialinių tinklų rūšimis, apžvelgta jų svarba visuomenei, komercijai, darbo rinkai bei politikai. Darbe plačiau apibūdintas socialinis tinklas „Facebook“, įvardinti šios technologijos pranašumai ir atliekamos funkcijos.

Darbe išanalizuota keliamo turinio strategijos sąvoka, įvardinti strategijų bruožai, kurie buvo tiriami analizuojant socialinio tinklo „Facebook“ kanalus. Keliamas turinys šiuose kanaluose buvo analizuojamas pagal publikuojamų vaizdo įrašų trukmę, kuriamo turinio tematiką, paskelbimo savaitės dieną bei laiką. Taip pat susipažinta su tyrimo srityje naudojamais metodais. Temų modeliavimui buvo pasirinkta taikyti latentinį Dirichlė paskirstymo algortimą, kuris yra vienas populiariausių temų modeliavime naudojamų metodų, leidžiančių išskirti iš žodžių rinkinių pagrindines temas be papildomos žmogaus intervencijos. Klasterizavimui buvo pasirinkta taikyti populiarius metodus kaip hierarchinį ir K-vidurkių klasterizavimo metodą. Statistinių hipotezių tikrinimui buvo pasinaudota Kruskal-Wallis ir Dunn testais. Jų pagalba buvo patikrintos įžvalgos, gautos išanalizavus puslapių keliamo turinio strategijas.

Tyrime buvo analizuojami trys „Facebook“ puslapiai: „puslapis1“, „puslapis2“, „puslapis3“. Tyrimui buvo susikomplektuota duomenų imtis, kuri buvo subalansuota ir sudaryta iš 188 kintamųjų bei 8264 skirtingų vaizdo įrašų. Visi nagrinėjami vaizdo įrašai buvo papublikuoti socialinėje platformoje 2021 m. laikotarpyje. Duomenų imtyje nagrinėjami kintamieji apibūdino filmukų pagrindinius vaizdo įrašų parametrus, valstybių gyventojų peržiūrų rodiklius, taip įrašų žiūrimumą pagal amžių ir lytį. Taip pat buvo sukurti papildomi išvestiniai kintamieji, kurie padėjo atlikti puslapių populiarumo vertinimą. Galiausiai patikrinus ir nustačius, kad duomenyse trūkstamų reikšmių kiekis sudaro sąlyginai nedidelę dalį visos imties, įrašai su trūkstama informacija buvo panaikinti.

Siekiant išsamiau suprasti duomenis, kiekvienam „Facebook“ puslapiui buvo atlikta aprašomoji statistika ir pirminė duomenų analizė. Pastebėta, jog vidutinis įrašų žiūrėjimo laikas kiekviename puslapyje buvo panašus ir skyrėsi tik keliomis sekundėmis, sužinota, kad visuose kanaluose labiausiai dominuoja labai trumpi vaizdo įrašai, kurių trukmė yra iki 5 min., tačiau nustatyta, jog kiekviename kanale buvo talpinamų ir labai didelės trukmės įrašų. Visuose kanaluose vidutiniškai atsispindėjo žemas įrašų parodymų ir paspaudimų ant jo skaičių santykis. Tokie rezultatai atskleidė įrašų patrauklumo trūkumą. Analizuojant filmukų paspaudimų ant įrašo skaičių bei vaizdo įrašų peržiūrų kiekį buvo atrastas dar vienas pastebėjimas - nemaža dalis peržiūrų buvo surenkama dėl „Facebook“ automatinio paleidimo funkcijos. Nagrinėjant kanalų auditorijas buvo išsiaiškinta, jog kanalų vaizdo įrašus labiausiai mėgsta stebėti Azijos, Arfikos, Pietų Amerikos ir Šiaurės Amerikos valstybės. Taip pat surasta, jog „puslapis2“ kanalo dominuojanti auditorija yra 18-54 m. moterys, o „puslapis1“ ir „puslapis3“ - 18-54 m. vyrai.

Norint geriau suprasti duomenis ir atlikti gilesnį populiarumo vertinimą, buvo atliktas temų modeliavimas. Temų modeliavimas padėjo apibendrinti duomenų imtyje turimą tekstinę informaciją ir išsiaiškinti, kokiomis tematikomis buvo kuriamas turinys. Prieš atliekant temų modeliavimą, pirma reikėjo sutvarkyti ir paruošti tekstinę informaciją. Vėliau buvo pasinaudota latentinio Dirichlė paskirstymo modeliu ir gauta temų žodžių matrica. Iš kiekvienai temai priskiriamų raktinių žodžių buvo sudarytos galutinės apibendrintos temos: „Pasidaryk pats, namų amatai ir pramogos“, „Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai“, „Statyba ir automobiliai“, „Namo ir kiemo restauravimas“ bei „Makiažas, grožis ir menas“. Svarbu paminėti, jog darbe taikomą temų modeliavimą galima būtų patobulinti: modelio tikslumą galima būtų pagerinti įtraukiant ir analizuojant daugiau hiperparametrų.

Išanalizavus nagrinėjamų kanalų keliamo turinio strategijas bei patikrinus išsikeltas hipotezes nu-

statyta, jog „puslapis1“ kanale labiausiai žiūrimi buvo „Statyba ir automobiliai“, „Namo ir kiemo restauravimas“ bei „Makiažas, grožis ir menas“ temų vaizdo įrašai. „puslapis2“ puslapyje mažiausiai peržiūrų sulaukė „Makiažas, grožis ir menas“ tematikos vaizdo įrašai, tuo tarpu kitos temos žiūrimumo atžvilgiu buvo panašios. „puslapis3“ kanale mažiausiai peržiūrų surinko „Statyba ir automobiliai“ tematikos filmukai, kai kitų temų įrašai žiūrimi buvo panašiai. Taip pat įsitikinta, kad „Namo ir kiemo restauravimas“ tematikos įrašai kanaluose sulaukdavo daugiausiai vartotojų interakcijų. Išsiaiškinta, jog daugiausiai įrašai peržiūrų surenka tada, kai yra publikuojami ryte arba vakare. Galiausiai sužinota, jog įrašų publikavimas skirtingomis savaitės dienomis neturi įtakos vaizdo įrašų peržiūroms ir sulaukiamam vartotojų interakcijų kiekiui. Siekiant patobulinti strategijų analizę, ją galima būtų išplėsti tiriant daugiau bruožų. Papildomi analizuojami bruožai galėtų apibūdinti, kokios kokybės yra keliami vaizdo įrašai, taip pat parodyti, ar filmukai yra kuriami puslapių moderatorių, ar įkeltų įrašų autorinės teisės priklauso kitiems kūrėjams. Papildomiems bruožams tirti reiktų papildomų duomenų, kurių nepavyko gauti šiame darbe dėl konfidencialumo ribojimų.

Galiausiai buvo atlikta kiekvieno nagrinėjamo puslapio vaizdo įrašų klasterizacija. Vaizdo įrašų klasterizavimas buvo atliktas K-vidurkių ir hierarchiniu metodu. Kiekvienas klasterizavimo metodas buvo taikomas visiems kanalų filmukams sujungus du požymių rinkinius, taip pat ir atskiriems rinkiniams. Paskutiniu etapu buvo klasterizuojami kiekvieno kanalo vaizdo įrašai pagal abu požymių rinkinius. Klasterizavimo etapu buvo siekta pasižiūrėti, kokie bruožai yra būdingi kiekvieno kanalo populiariausiems vaizdo įrašams, todėl didžiausias skiriamas dėmesys buvo klasteriams, kuriuose buvo surinkti žiūrimiausi filmukai. Abu klasterizavimo metodai atskleidė, jog tarp „puslapis1“ kanalo populiariausių filmukų labiausiai dominavo 3 ir 4 temų vaizdo įrašai, „puslapis2“ - 2 tematikos filmukai, „puslapis3“ - 3 tematikos įrašai. Taip pat hierarchinio klasterizavimo metodo rezultatai indikavo, jog populiariausi vaizdo įrašai daugiausia yra skelbiami darbo dienomis bei 7-9 val. ryto. Peržvelgus klasterizavimo rezultatus ir rezultatus, gautus po hipotezių tikrinimo, pastebėta sutapimų ir prieštaravimų. Atlikti testai ir populiariausių įrašų klasterių analizė atslkeidė, jog „puslapis1“ kanale labiausiai žiūrimi yra 3 ir 4 temų vaizdo įrašai. Taip pat šiuose klasteriuose pasimatė hipotezių metu patvirtintas teiginys, jog vaizdo įrašai daugiausiai sulaukia peržiūrų, kai yra publikuojami rytais. Galiausiai, nors hierarchinio klasterizavimo metodo rezultatai ir indikavo, kad įrašai daugiausiai sulaukia peržiūrų, kai yra skelbiami darbo dienomis, hipotezių tikrinimo metu šis teiginys buvo paneigtas.

Literatūra

[1] Brooks, S.K., Webster, R.K., Smith, L.E., Woodland, L., Wessely, S., Greenberg N. The psychological impact of quarantine and how to reduce it: rapid review of the evidence. 395:912-20, 2020.

[2] Kennedy, K. Facebook Page vs Profile: Everything You Need To Know. https://yourblogworks.com/facebook-page-vs-facebook-profile/.

[3] Amy, A. New York Times Study: „The Psychology of Sharing. Why Do People Share Online?“ https://www.businesswire.com/news/home/20110713005971/en/.

[4] Pew Research Center. Americans Who Mainly Get Their News on Social Media Are Less Engaged, Less Knowledgeable. https://www.pewresearch.org/journalism/2020/07/30/.

[5] The New York Times Social Media's Globe-Shaking Power. https://www.nytimes.com/2016/11/17/technology/social-medias-globe-shaking-power.html?_r=0.

[6] Fujiwara, T., Müller, K., Schwarz, C. How Twitter affected the 2016 presidential election. https://voxeu.org/article/how-twitter-affected-2016-presidential-election.

[7] Pew Research Center. Social Media Update 2016. https://www.pewresearch.org/internet/2016/11/11/social-media-update-2016/.

[8] Simplilearn. Social Media for Business: Tips for Generating Leads and Building Your Brand. https://www.simplilearn.com/social-media-for-business-article.

[9] Babu, S. 90% of Employers Consider an Applicant's Social Media Activity During Hiring Process https://smallbiztrends.com/2020/05/social-media-screening.html.

[10] CareerBuilder. More Than Half of Employers Have Found Content on Social Media That Caused Them NOT to Hire a Candidate, According to Recent CareerBuilder Survey. https://press.careerbuilder.com/.

[11] Visuotinė lietuvių enciklopedija. https://www.vle.lt/straipsnis/socialiniai-tinklai/.

[12] Merriam-Webster Dictionary. https://www.merriam-webster.com/dictionary/social%20network.

[13] Cambridge Dictionary. https://dictionary.cambridge.org/dictionary/english/social-networking.

[14] Overby, S. Augmented reality (AR) vs. virtual reality (VR): What's the difference? https://enterprisersproject.com/article/2019/10/ar-vs-vr-whats-difference.

[15] Bardi, J. What Is Virtual Reality: Definitions, Devices, and Examples. https://www.marxentlabs.com/what-is-virtual-reality/.

[16] Statista Research Department. Number of monthly active Facebook users worldwide as of 1st quarter 2022. https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/#:~:text=How.

[17] Lua, A. What is Facebook? https://edu.gcfglobal.org/en/facebook101/what-is-facebook/1/.

[18] Goodwill Community Foundation. 20 Top Social Media Sites to Consider for Your Brand in 2022 https://buffer.com/library/social-media-sites/.

[19] Newberry, C. How the Facebook Algorithm Works in 2022 and How to Make it Work for You. https://blog.hootsuite.com/facebook-algorithm/.

[20] Halvorson, K. The Discipline of Content Strategy. http://alistapart.com/article/thedisciplineofcontentstrategy/.

[21] Rach, M., Halvorson, K. Content strategy for the Web: content strategy Web _p2. New Riders, 2012.

[22] Blei, D. M., Ng, A. Y., Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, 3: 993-1022, 2003.

[23] Mcauliffe, J., Blei, D. Supervised topic models. Advances in neural information processing systems, 20, 2007.

[24] Li, D., He, B., Ding, Y., Tang, J., Sugimoto, C., Qin, Z., Yan, E., Li, J. and Dong, T. Community-based topic modeling for social tagging. In Proceedings of the 19th ACM international conference on Information and knowledge management, 1565-1568, 2010.

[25] Daud, A. Using time topic modeling for semantics-based dynamic research interest finding. Knowledge-Based Systems, 26: 154-163, 2012.

[26] Lim, K.W., Chen, C. and Buntine, W. Twitter-network topic model: A full Bayesian treatment for social network and text modeling, arXiv preprint arXiv:1609.06791, 2016.

[27] Ostrowski, D.A. Using latent dirichlet allocation for topic modelling in twitter. *In Proceedings of the 2015 IEEE 9th international conference on semantic computing (IEEE ICSC 2015)*, 493-497, 2015.

[28] Weng, J., Lim, E.P., Jiang, J. and He, Q. Twitterrank: finding topic-sensitive influential twitterers. *In Proceedings of the third ACM international conference on Web search and data mining*, 261-270, 2010.

[29] Alpaydin, E. Introduction to machine learning. 2nd ed., Massachusetts:The MIT Press, 2010.

[30] Mohammed, M., Khan, M. B., Bashier, E. B. M. Machine learning: algorithms and applications. Crc Press, 2016.

[31] Frigyik, B., Gupta, M., Chen, Y. PIntroduction to the dirichlet distribution and related processes. *Department of Electrical Engineering, University of Washignton*, UWEETR-2010-0006, 2010.

[32] Wikipedia, the free encyclopedia. Multinomial distribution. https://en.wikipedia.org/wiki/Multinomial_distribution.

[33] Liu, Z., Li, M., Liu, Y., Ponraj, M. Performance evaluation of Latent Dirichlet Allocation in text mining. *In 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, vol. 4, 2695-2698, 2011.

[34] Verma, M., Srivastava, M., Chack, N., Diswar, A. K., Gupta, N. A Comparative Study of Various Clustering Algorithms in Data Mining. *International Journal of Engineering Research and Applications*, 2(3):1379–1384, 2012.

[35] Tsai, C. W., Lai, C. F., Chao, H. C., Vasilakos, A. V. Big data analytics: a survey. Journal of Big Data 2, 21, 2015. https://doi.org/10.1186/s40537-015-0030-3.

[36] Dunham, M. H. Data Mining Introductory and Advanced Topics. Pearson Education, Inc. Prentice Hall. 315 p. 2003.

[37] Erman, J., Arlitt, M., Mahanti, A. Traffic classification using clustering algorithms. *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, MineNet'06, 2006: 281–286, 2006.

[38] Burney, S. A., Tariq, H. K-Means Cluster Analysis for Image Segmentation. *International Journal of Computer Applications*, 96(4):1–8, 2014

[39] Tubular. https://tubularlabs.com/.

[40] Krug, S. Reactions Now Available Globally. https://about.fb.com/news/2016/02/reactions-now-available-globally/.

[41] Facebook. Video requirements. https://www.facebook.com/business/m/one-sheeters/video-requirements.

[42] Facebook. About Facebook Game Stream Time Limits. https://www.facebook.com/business/help/313730735865902?id=648321075955172.

[43] MiniMatters. What's the Impact of Facebook Autoplay? https://www.minimatters.com/facebook-autoplay/.

[44] Jung, E. H., Sundar, S. S. Senior citizens on Facebook: How do they interact and why? *Computers in Human Behavior*, 61: 27-35, 2016.

[45] Yeung, C. Social Media Usage Statistics By Age: Marketing to Adults Aged 50+. http://synthesio.wpengine.com/blog/social-media-usage-statistics-by-age/?utm_medium=blog&utm_source=mktg&utm_campaign=socialmediausage.

[46] Menzli, A. Tokenization in NLP: Types, Challenges, Examples, Tools. https://neptune.ai/blog/tokenization-in-nlp.

[47] Gensim Tutorials. https://tedboy.github.io/nlps/generated/gensim.utils.html.

[48] Lang, N. Stemming vs. Lemmatization in NLP. https://towardsdatascience.com/stemming-vs-lemmatization-in-nlp-dea008600a0.

[49] Scikit-learn library. https://scikit-learn.org/stable/getting_started.html.

[50] LatentDirichletAllocation funkcijos dokumentacija. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html.

[51] Bouchrika, I. The Best Times to Post on Social Media: 2022 Studies & Statistics. https://research.com/tutorials/the-best-times-to-post-on-social-media.

[52] StandardScaler function. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[53] Silhouette_score function. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.

[54] Dendrogram function. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html.

[55] AgglomerativeClustering function. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

8 Priedai

	Minimumas	Maksimumas	Vidurkis	Mediana	Stand. nuok.
Truk. sek.	61.000	28809.000	247.802	187.000	997.735
Truk. min	1.000	480.00	3.937	3.000	16.609
Parod.	4.599e+03	5.855e+08	1.444e+07	4.087e+06	3.325e+07
Paspaud.	5.000e+00	1.677e+07	1.595e+05	1.331e+04	7.480e+05
PR	0.060	12.580	0.616	0.400	0.738
Per.	4.000e+02	2.571e+08	7.162e+06	1.890e+06	1.655e+07
Per. 60	1.270e+02	1.169e+08	2.482e+06	5.155e+05	6.449e+06
Vid. žiūr.	7.380	79.150	26.584	26.260	8.781
Inter.	1.140e+02	5.925e+07	7.233e+05	1.307e+05	2.272e+06
Kom.	0.000	93974.000	1474.883	159.000	7266.483
Mėg.	2.300e+01	4.626e+06	5.067e+04	7.500e+03	5.067e+04
Pasidal.	0.000	895894.000	7465.132	587.000	7465.132

6 lentelė: „puslapis1“ puslapio pagrindinių rodiklių bendra statistinė informacija. Lentelėse kinamųjų pavadinimai yra sutrumpinti: vaizdo įrašo trukmė, išreikšta sekundėmis - „Truk. sec.“, vaizdo įrašo trukmė, išreikšta minutėmis - „Truk. min.“, įrašo parodymo kiekis „Facebook" vartotojų ekranuose - „Parod.“, paspaudimų ant vaizdo įrašo skaičius - „Paspaud.“, vaizdo įrašų parodymų ir paspaudimų ant jo skaičių santykis - „PR“, 3 sekundžių vaizdo įrašo peržiūros - „Per“, 60 sekundžių vaizdo įrašo peržiūros - „Per 60“, vidutinis žiūrėjimo laikas sekundėmis - „Vid. žiūr.“, bendras vartotojų interakcijų skaičius - „Inter.“, komentarų kiekis - „Kom.“, paspaudimų mėgti kiekis - „Mėg.“ ir pasidalinimų kiekis - „Pasidal.“.

	Minimumas	Maksimumas	Vidurkis	Mediana	Stand. nuok.
Truk. sek.	61.000	28811.000	421.537	187.000	2378.246
Truk. min	1.000	480.000	6.795	3.000	39.626
Parod.	7.420e+02	2.084e+08	2.188e+06	2.461e+05	8.696e+06
Paspaud.	5.000	5.633e+06	2.759e+04	6.325e+02	2.311e+05
PR	0.040	26.050	0.687	0.350	1.452
Per.	1.580e+02	1.239e+08	1.077e+06	9.431e+04	4.737e+06
Per. 60	3.400e+01	6.125e+07	3.785e+05	1.987e+04	2.122e+06
Vid. žiūr.	4.380	64.780	22.455	21.595	8.738
Inter.	2.200e+01	1.315e+07	1.094e+05	5.978e+03	5.909e+05
Kom.	0.000	33833.000	292.667	8.000	1838.502
Mėg.	0.000	643843.000	6674.826	444.500	34237.495
Pasidal.	0.000	147317.000	764.598	22.000	5531.351

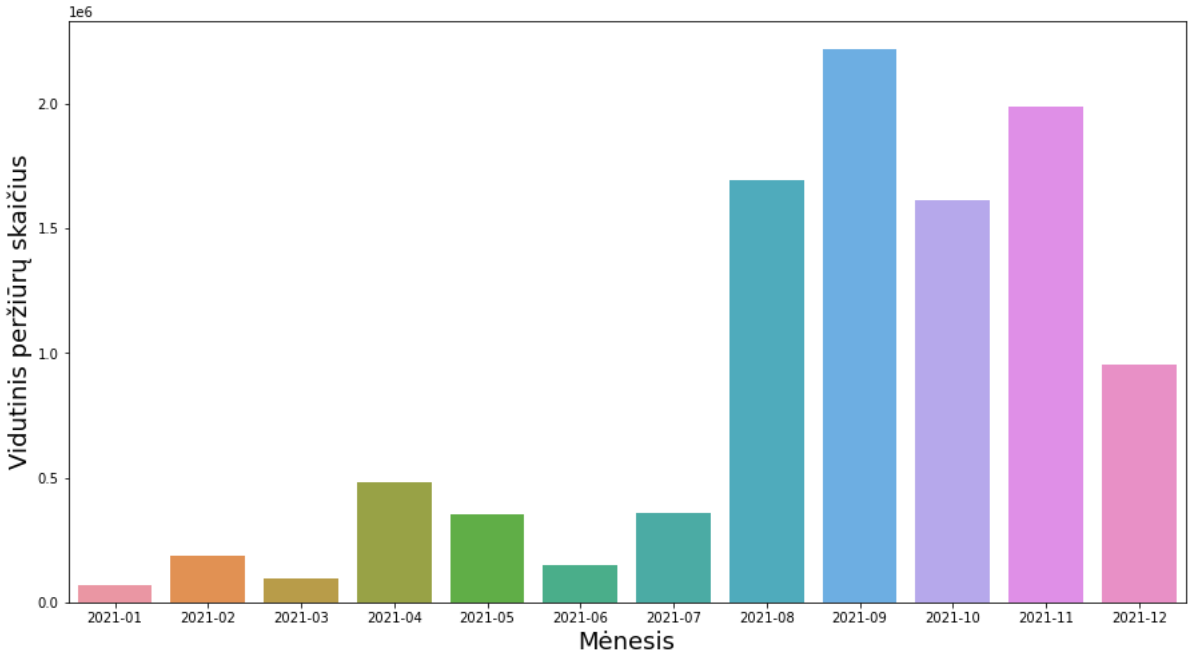
7 lentelė: „puslapis2“ puslapio pagrindinių rodiklių bendra statistinė informacija. Lentelėse kinamųjų pavadinimai yra sutrumpinti: vaizdo įrašo trukmė, išreikšta sekundėmis - „Truk. sec.“, vaizdo įrašo trukmė, išreikšta minutėmis - „Truk. min.“, įrašo parodymo kiekis „Facebook" vartotojų ekranuose - „Parod.“, paspaudimų ant vaizdo įrašo skaičius - „Paspaud.“, vaizdo įrašų parodymų ir paspaudimų ant jo skaičių santykis - „PR“, 3 sekundžių vaizdo įrašo peržiūros - „Per“, 60 sekundžių vaizdo įrašo peržiūros - „Per 60“, vidutinis žiūrėjimo laikas sekundėmis - „Vid. žiūr.“, bendras vartotojų interakcijų skaičius - „Inter.“, komentarų kiekis - „Kom.“, paspaudimų mėgti kiekis - „Mėg.“ ir pasidalinimų kiekis - „Pasidal.“.

	Minimumas	Maksimumas	Vidurkis	Mediana	Stand. nuok.
Truk. sek.	61.000	28813.000	555.801	187.000	3036.405
Truk. min	1.000	480.000	9.053	3.000	50.597
Parod.	2.911e+03	6.034e+07	1.703e+06	3.533e+05	5.419e+06
Paspaud.	3.700e+01	3.377e+06	5.516e+04	1.034e+03	3.074e+05
PR	0.050	20.090	1.057	0.440	1.653
Per.	4.900e+02	3.404e+07	8.226e+05	1.123e+05	2.878e+06
Per. 60	7.100e+01	1.107e+07	2.655e+05	2.420e+04	1.051e+06
Vid. žiūr.	7.480	65.710	21.730	20.310	7.675
Inter.	7.200e+01	4.951e+06	9.541e+04	7.201e+03	4.368e+05
Kom.	0.000	9015.000	74.075	8.000	394.625
Mėg.	4.000	312425.000	5559.288	629.000	23747.899
Pasidal.	0.000000	78965.000	669.657	35.000	4074.180

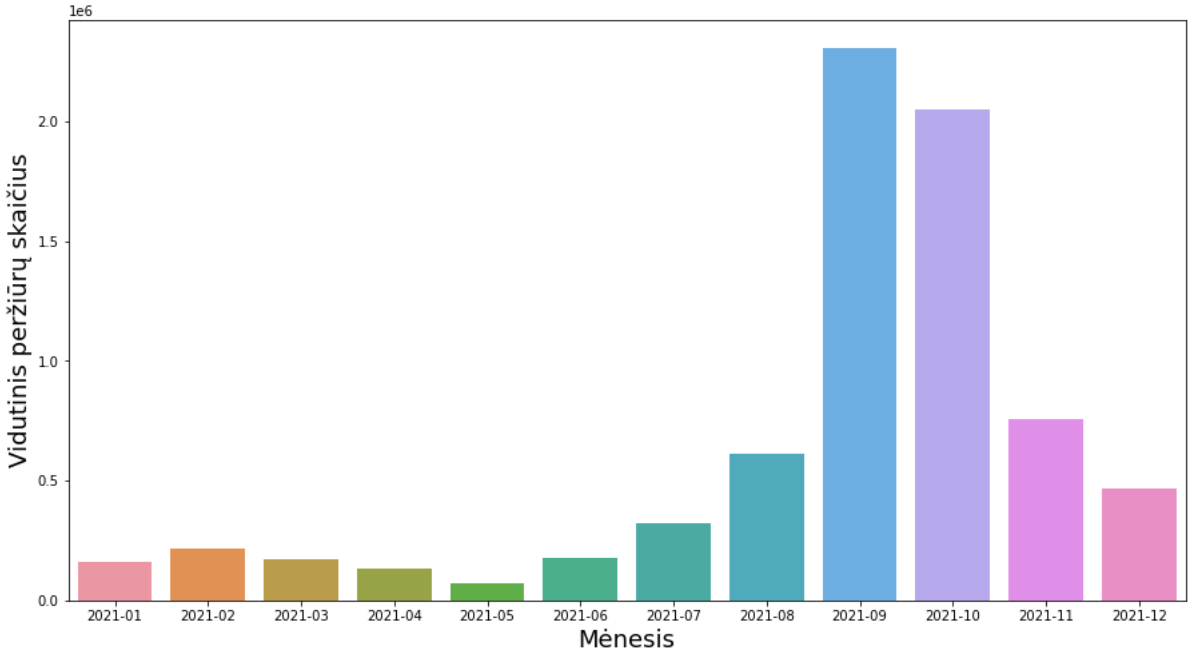
8 lentelė: „puslapis3“ puslapio pagrindinių rodiklių bendra statistinė informacija. Lentelėse kinamųjų pavadinimai yra sutrumpinti: vaizdo įrašo trukmė, išreikšta sekundėmis - „Truk. sec.“, vaizdo įrašo trukmė, išreikšta minutėmis - „Truk. min.“, įrašo parodymo kiekis „Facebook" vartotojų ekranuose - „Parod.“, paspaudimų ant vaizdo įrašo skaičius - „Paspaud.“, vaizdo įrašų parodymų ir paspaudimų ant jo skaičių santykis - „PR“, 3 sekundžių vaizdo įrašo peržiūros - „Per“, 60 sekundžių vaizdo įrašo peržiūros - „Per 60“, vidutinis žiūrėjimo laikas sekundėmis - „Vid. žiūr.“, bendras vartotojų interakcijų skaičius - „Inter.“, komentarų kiekis - „Kom.“, paspaudimų mėgti kiekis - „Mėg.“ ir pasidalinimų kiekis - „Pasidal.“.

	„puslapis1“	„puslapis2“	„puslapis3“
1.	25-34 m. vyrai	25-34 m. moterys	25-34 m. vyrai
2.	35-44 m. vyrai	18-24 m. moterys	35-44 m. vyrai
3.	18-24 m. vyrai	35-44 m. moterys	18-24 m. vyrai
4.	45-54 m. vyrai	25-34 m. moterys	45-54 m. vyrai
5.	25-34 m. moterys	45-54 m. moterys	25-34 m. moterys

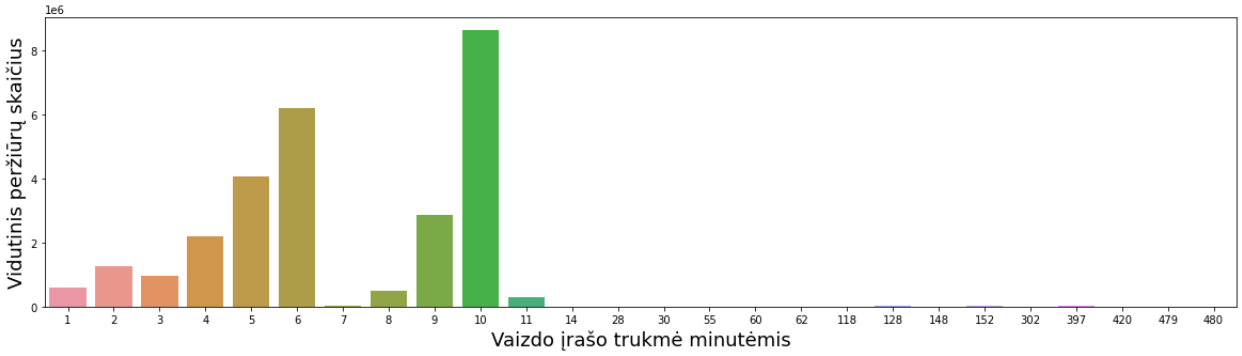
9 lentelė: Labiausiai žiūrinčios auditorijos pagal amžiaus grupę ir lytį.



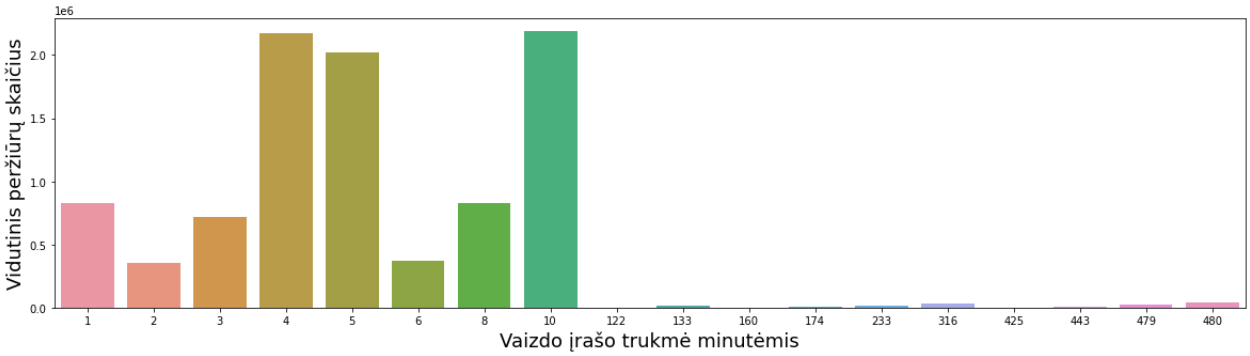
13 pav.: Vidutinis vaizdo įrašų peržiūrų skaičius „puslapis2“ puslapyje



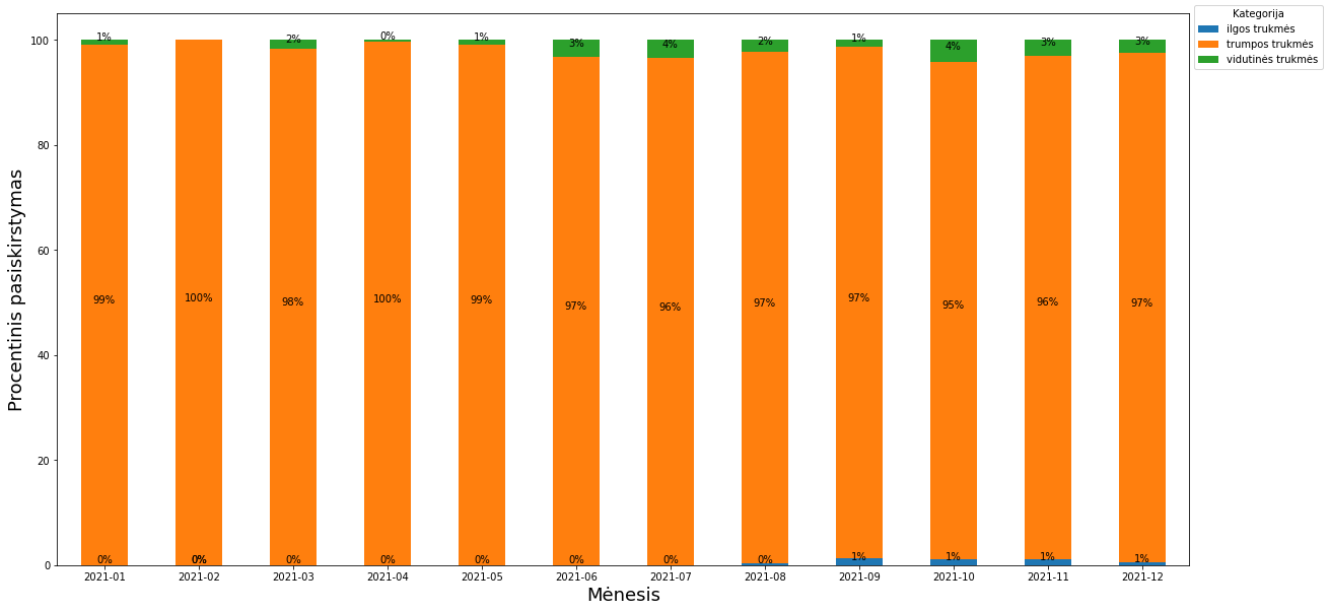
14 pav.: Vidutinis vaizdo įrašų peržiūrų skaičius „puslapis3“ puslapyje



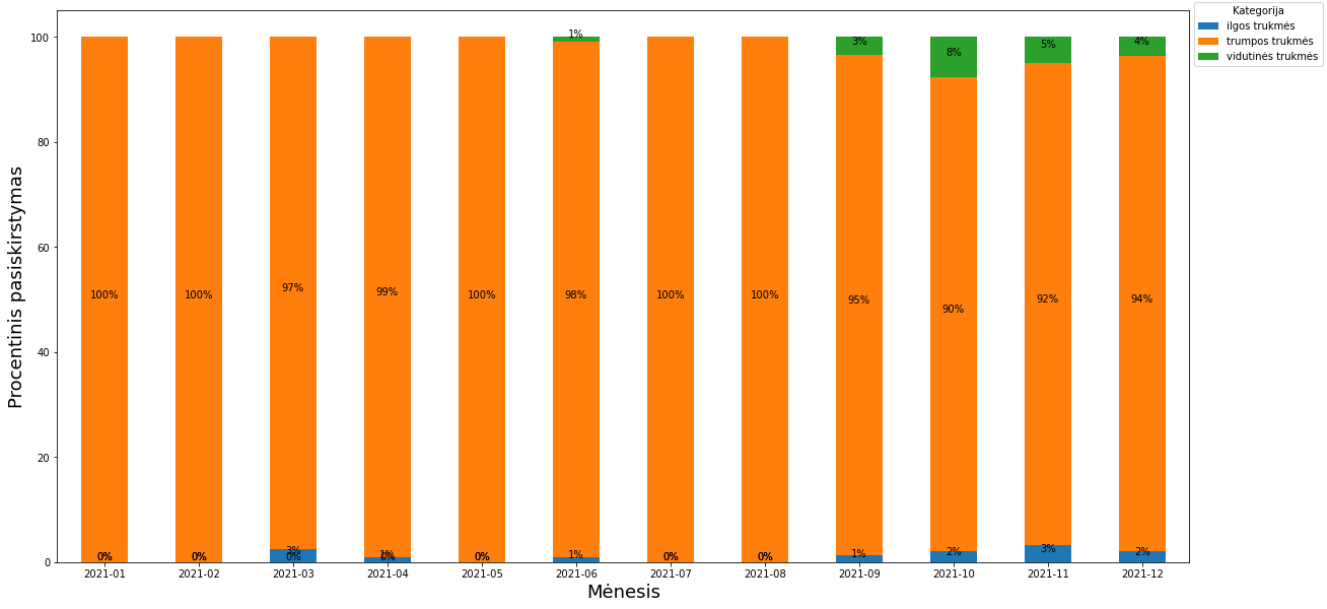
15 pav.: Vidutinis peržiūrų skaičius pagal vaizdo įrašų trukmę „puslapis2“ puslapyje.



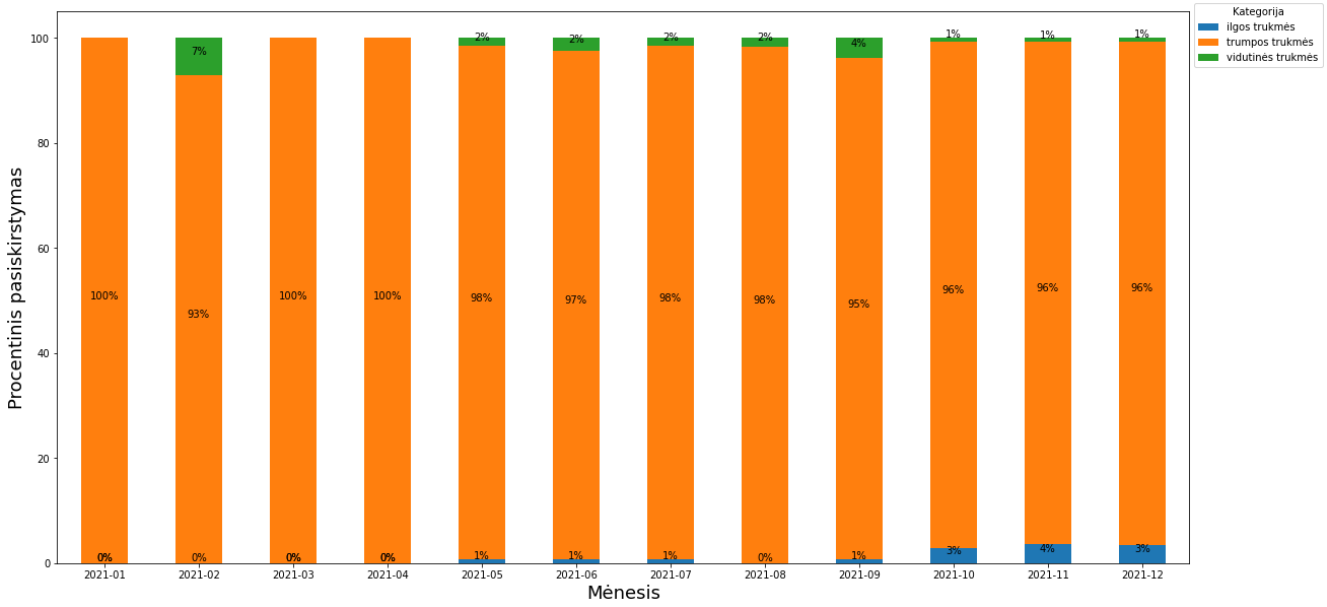
16 pav.: Vidutinis peržiūrų skaičius pagal vaizdo įrašų trukmę „puslapis3“ puslapyje.



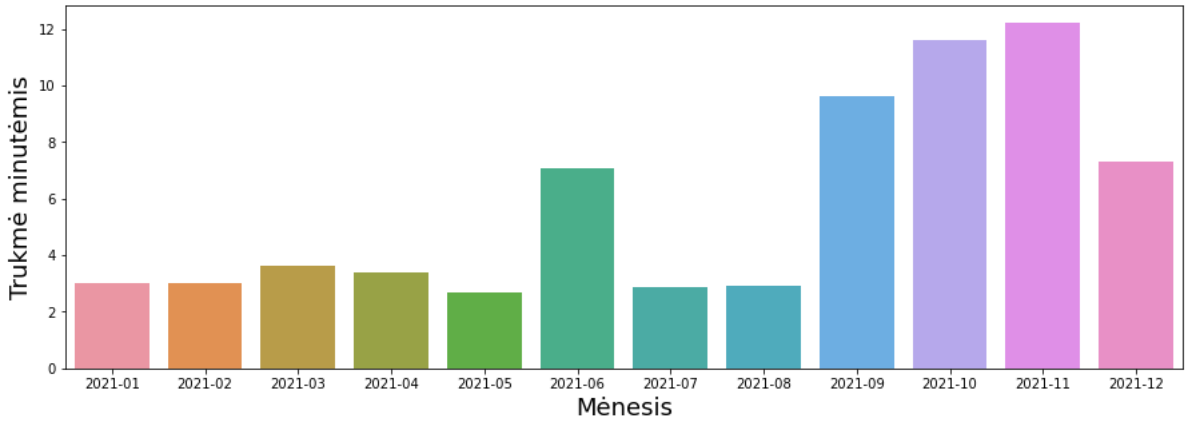
17 pav.: Keliamų vaizdo įrašų trukmės pasiskirstymas „puslapis1“ puslapyje.



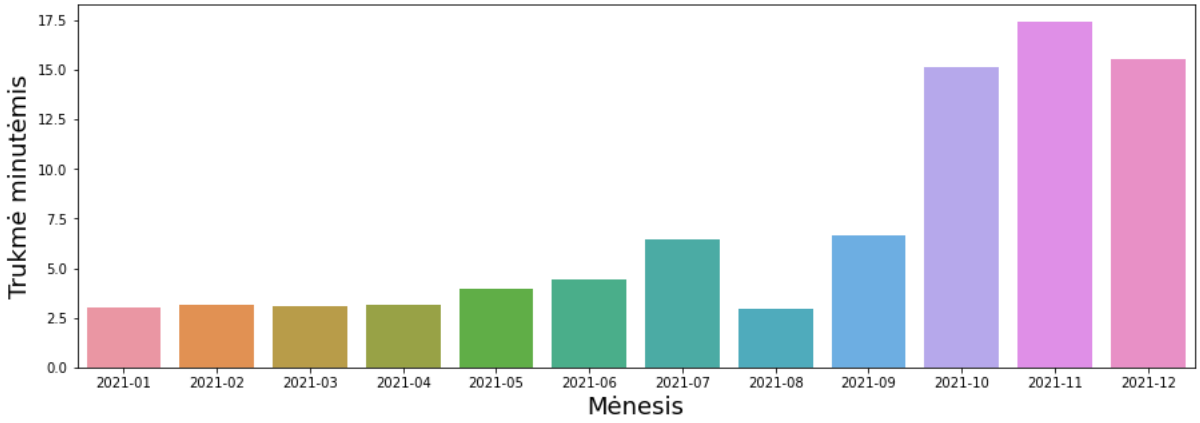
18 pav.: Keliamų vaizdo įrašų trukmės pasiskirstymas „puslapis2“ puslapyje.



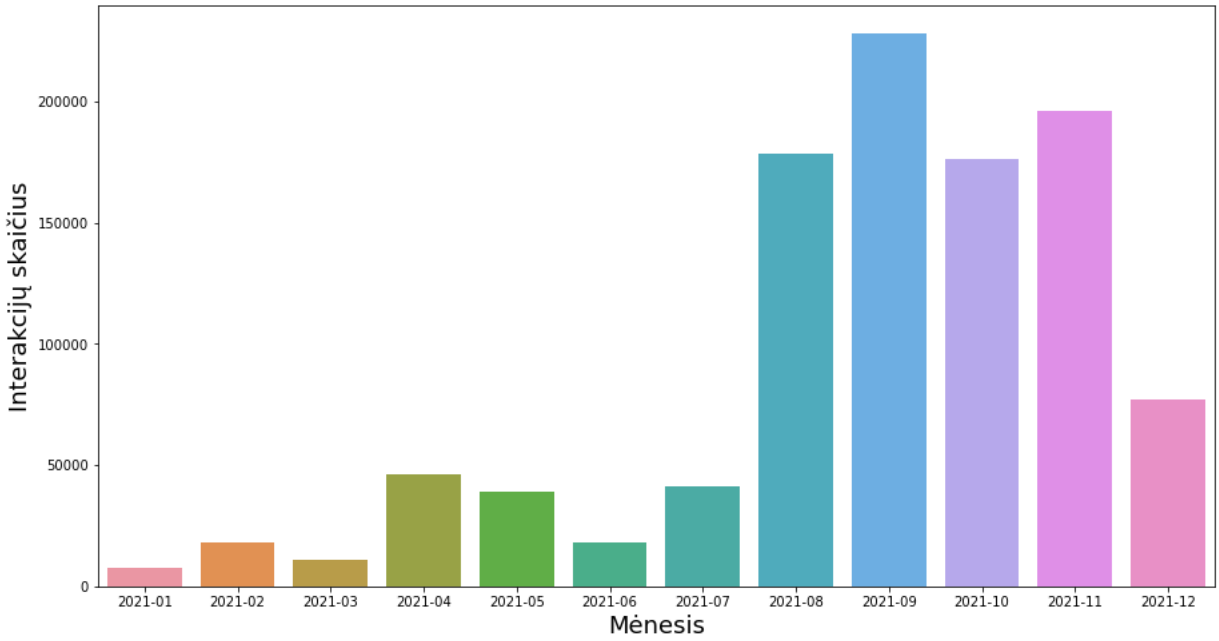
19 pav.: Keliamų vaizdo įrašu trukmės pasiskirstymas „puslapis3“ puslapyje.



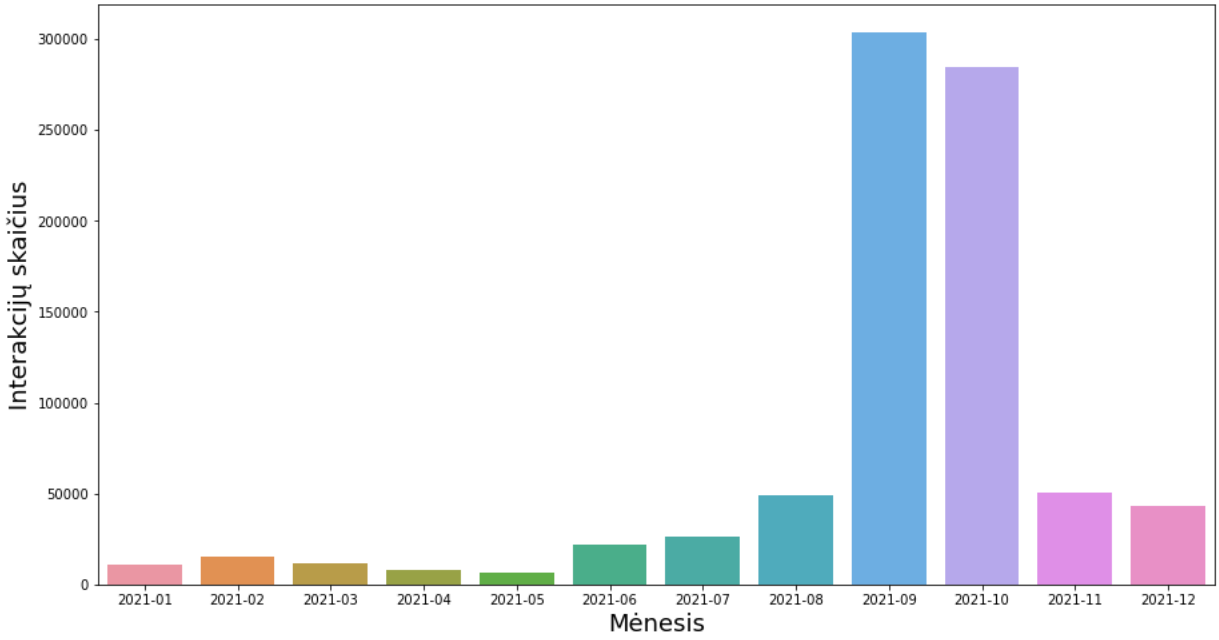
20 pav.: Vidutinė keliamų vaizdo įrašų trukmė „puslapis2“ puslapyje.



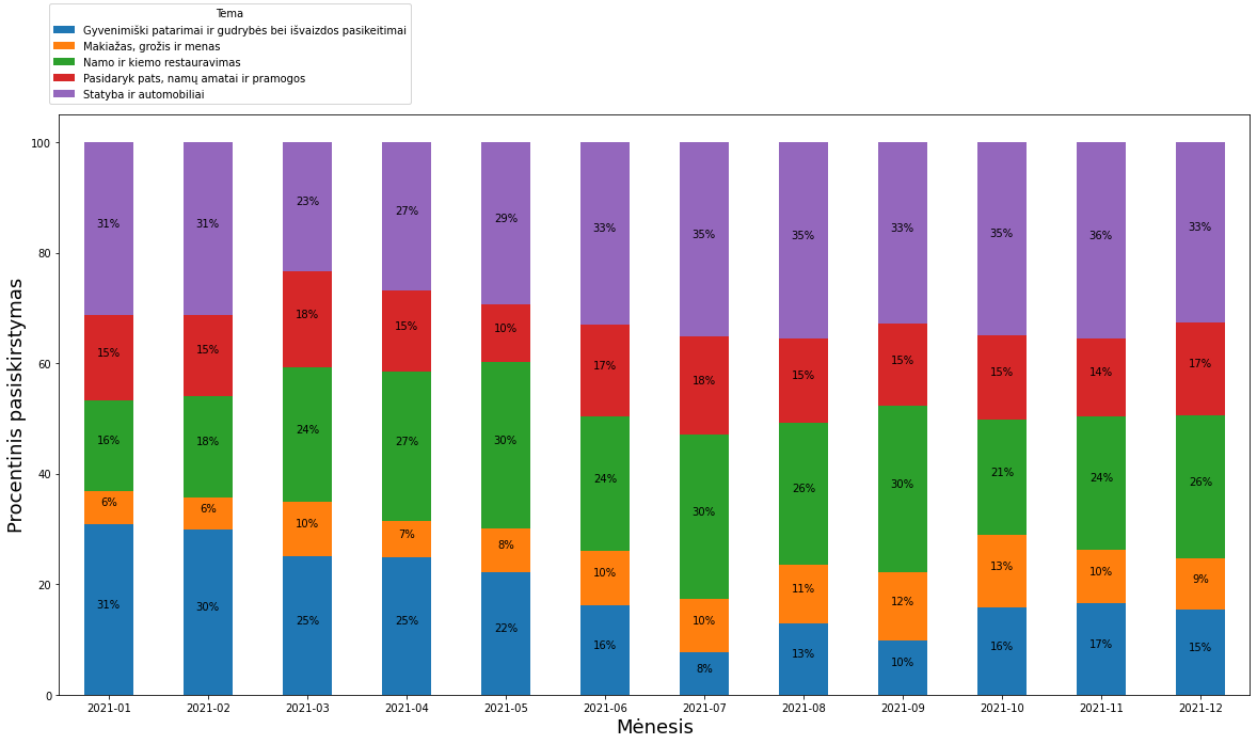
21 pav.: Vidutinė keliamų vaizdo įrašų trukmė „puslapis2“ puslapyje.



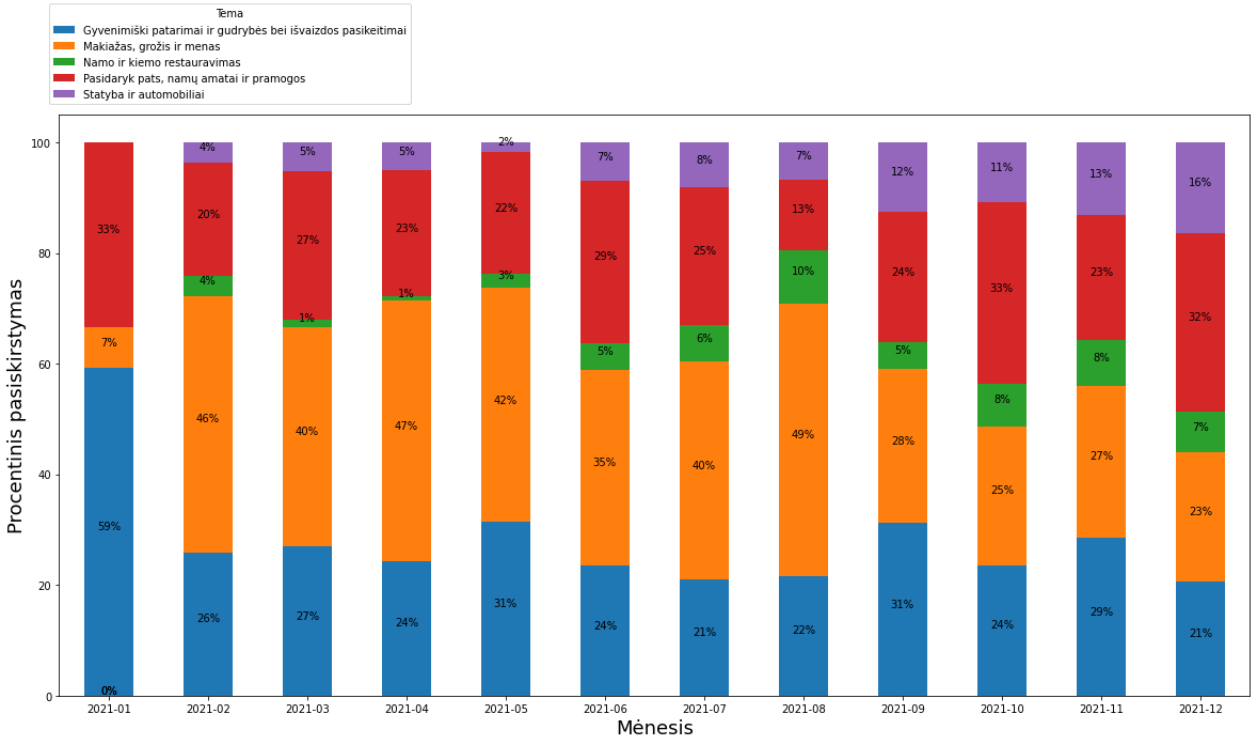
22 pav.: Vidutinis vartotojų interakcijų skaičius „puslapis2“ puslapyje.



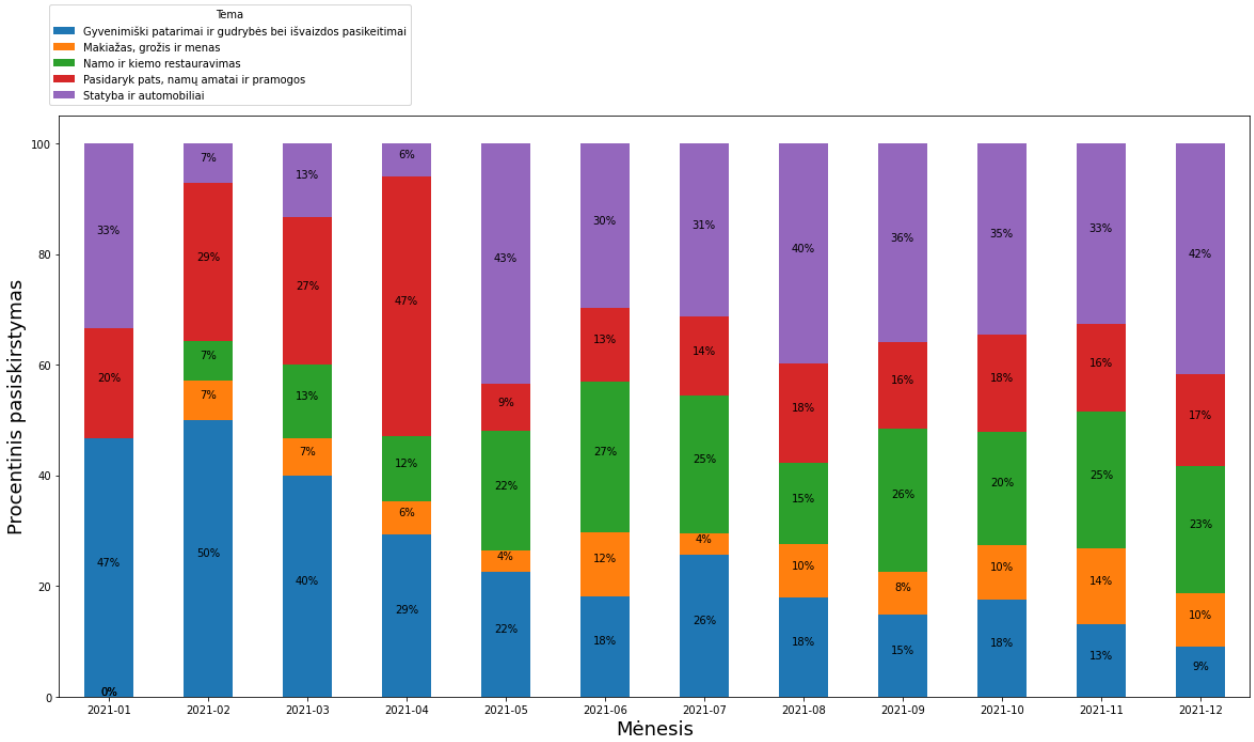
23 pav.: Vidutinis vartotojų interakcijų skaičius „puslapis3“ puslapyje.



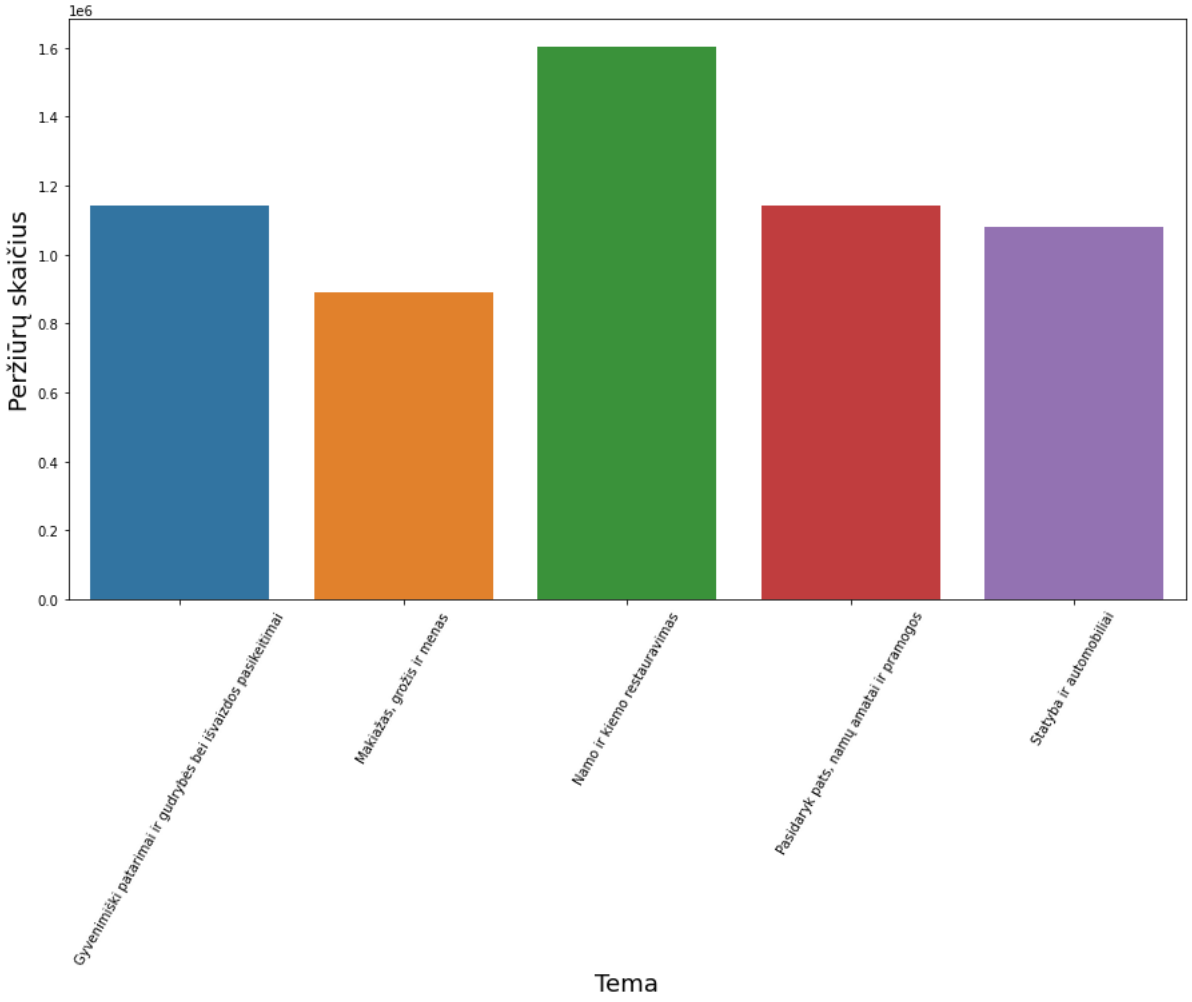
24 pav.: Publikuojamų vaizdo įrašų pasiskirstymas pagal temas „puslapis1“ kanale



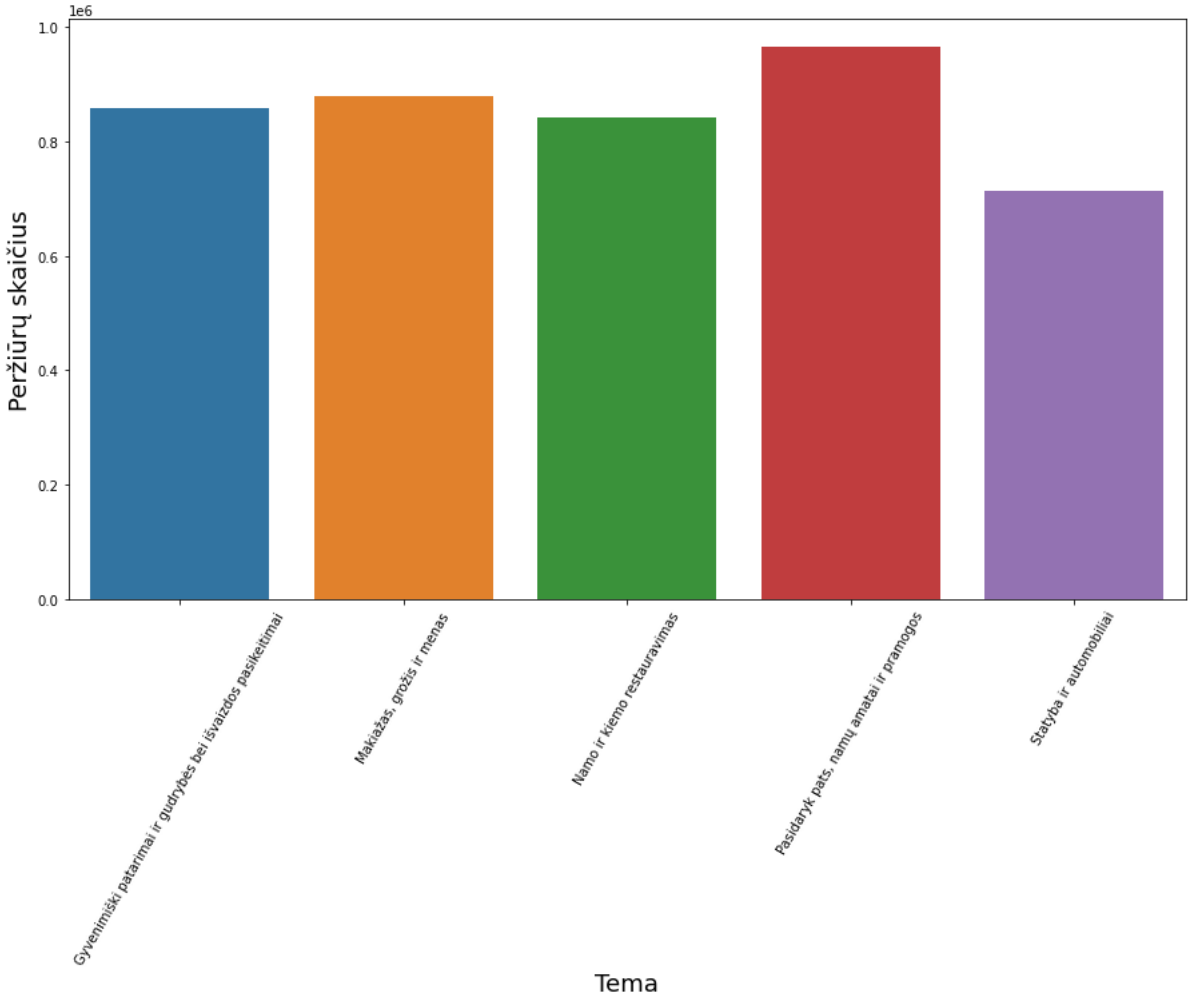
25 pav.: Publikuojamų vaizdo įrašų pasiskirstymas pagal temas „puslapis2“ kanale



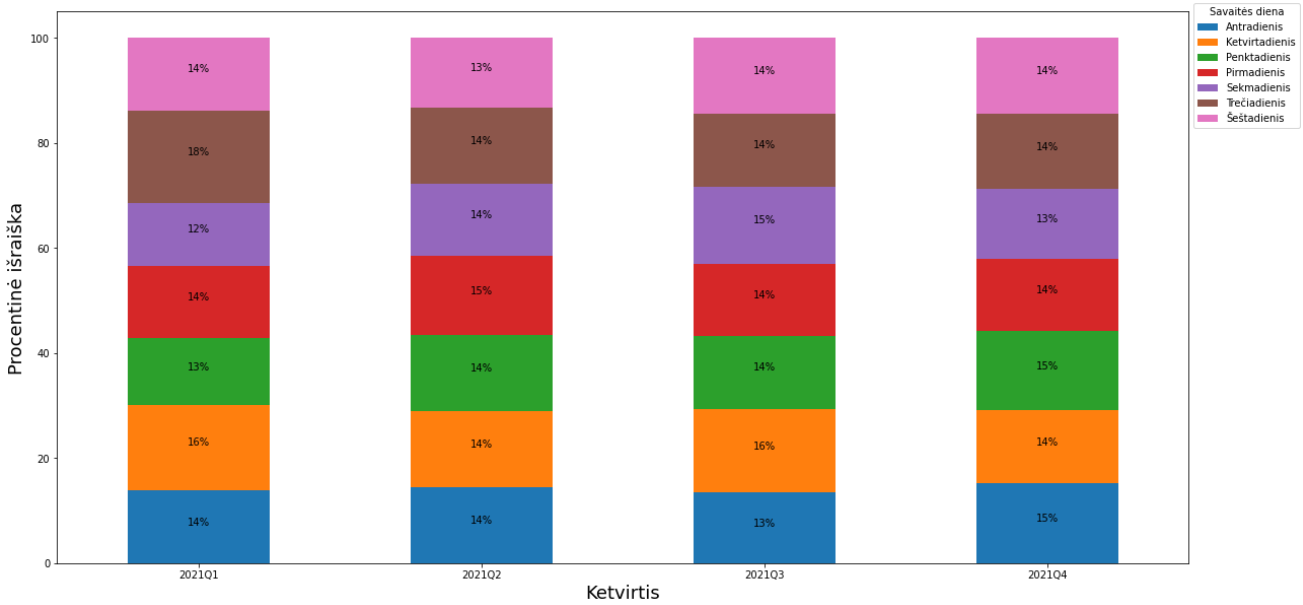
26 pav.: Publikuojamų vaizdo įrašų pasiskirstymas pagal temas „puslapis3“ kanale



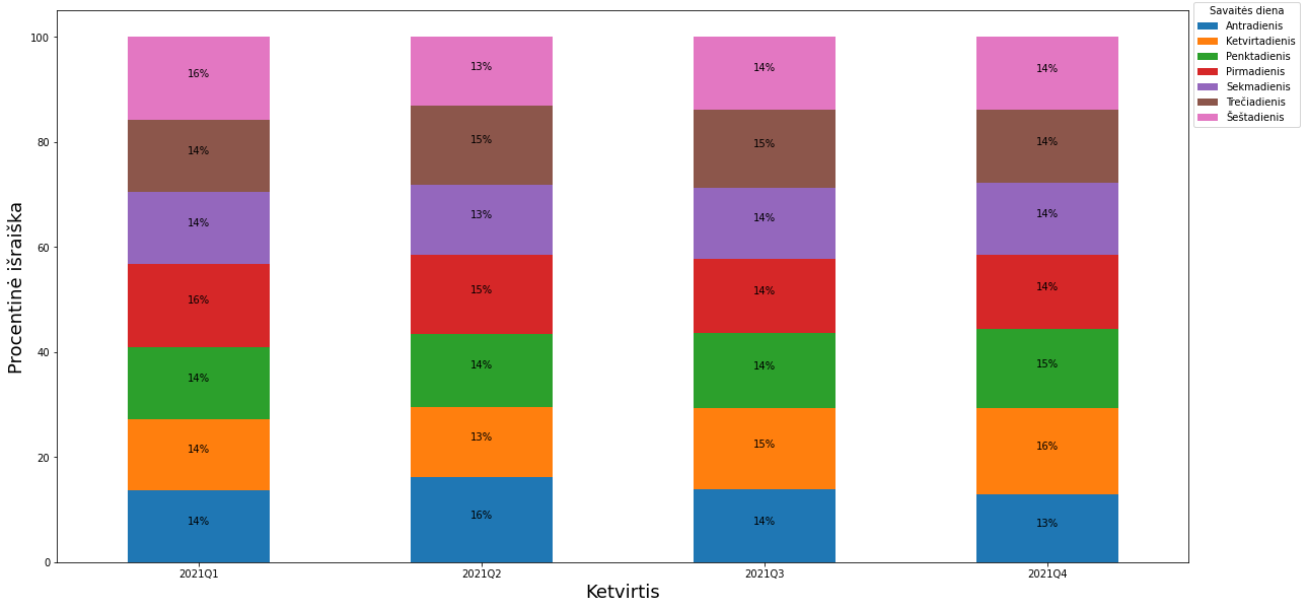
27 pav.: Vidutinis surenkamų peržiūrų skaičius pagal temas „puslapis2“ kanale.



28 pav.: Vidutinis surenkamų peržiūrų skaičius pagal temas „puslapis3“ kanale.



29 pav.: Savaitės dienomis publikuojamų įrašų pasiskirstymas „puslapis2“ kanale



30 pav.: Savaitės dienomis publikuojamų įrašų pasiskirstymas „puslapis3“ kanale

Temos nr.	1	2	3	4	5
1.	1.000000	8.691573e-02	6.346040e-03	3.220926e-03	0.193467
2.	0.086916	1.000000e+00	2.182484e-10	1.815677e-10	0.000029
3.	0.006346	2.182484e-10	1.000000e+00	1.000000e+00	1.000000
4.	0.003221	1.815677e-10	1.000000e+00	1.000000e+00	1.000000
5.	0.193467	2.865295e-05	1.000000e+00	1.000000e+00	1.000000

10 lentelė: Dunn testo rezultatai „puslapis1“ kanale.

Temos nr.	1	2	3	4	5
1.	1.000000	1.000000	0.781107	0.532945	0.353435
2.	1.000000	1.000000	1.000000	0.936152	0.101251
3.	0.781107	1.000000	1.000000	1.000000	0.009471
4.	0.532945	0.936152	1.000000	1.000000	0.013656
5.	0.353435	0.101251	0.009471	0.013656	1.000000

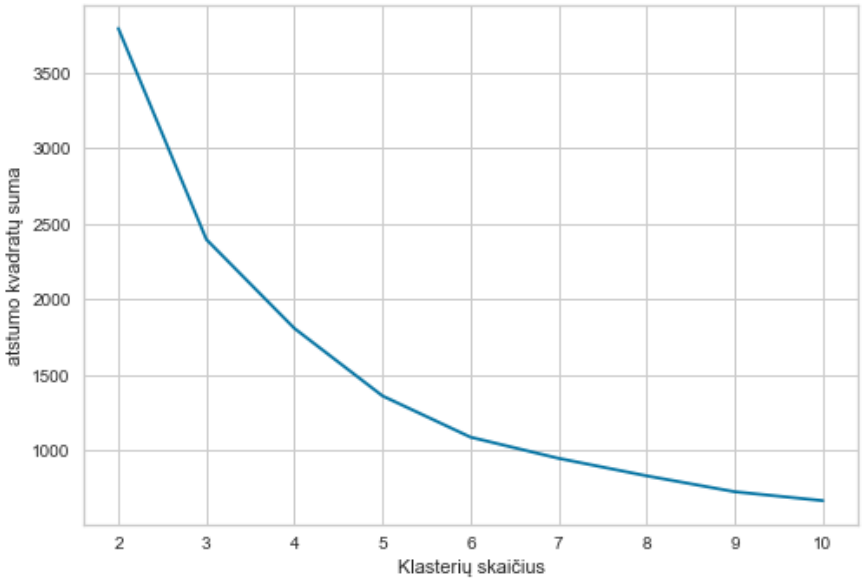
11 lentelė: Dunn testo rezultatai „puslapis2“ kanale.

Temos nr.	1	2	3	4	5
1.	1.000000	1.000000	0.000070	0.395326	1.000000
2.	1.000000	1.000000	0.025994	1.000000	0.506176
3.	0.000070	0.025994	1.000000	0.124585	0.000107
4.	0.395326	1.000000	0.124585	1.000000	0.135100
5.	1.000000	0.506176	0.000107	0.135100	1.000000

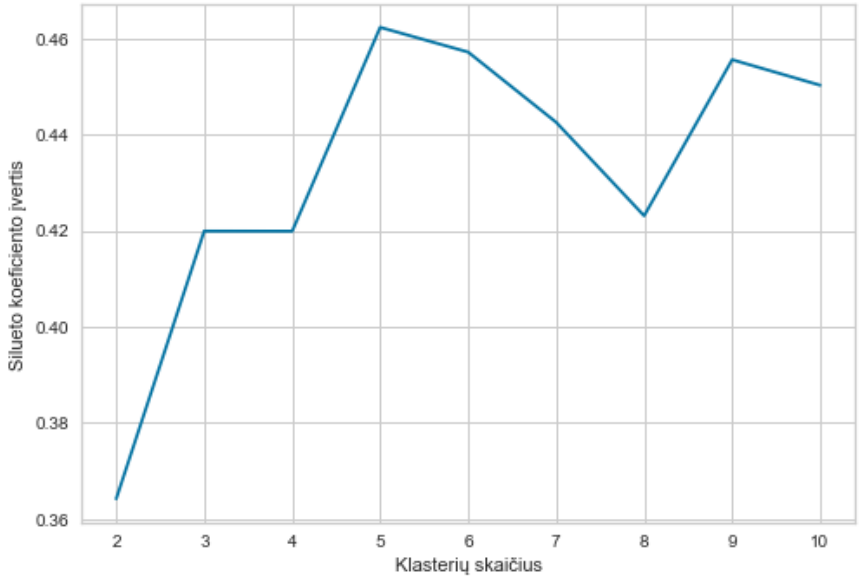
12 lentelė: Dunn testo rezultatai „puslapis3“ kanale.

Paros metas	Naktis	Rytas	Diena	Vakaras
Naktis	1.000000e+00	7.117992e-12	0.009728	5.339520e-07
Rytas	7.117992e-12	1.000000e+00	0.001318	1.000000e+00
Diena	9.728379e-03	1.318324e-03	1.000000	6.047303e-02
Vakaras	5.339520e-07	1.000000e+00	0.060473	1.000000e+00

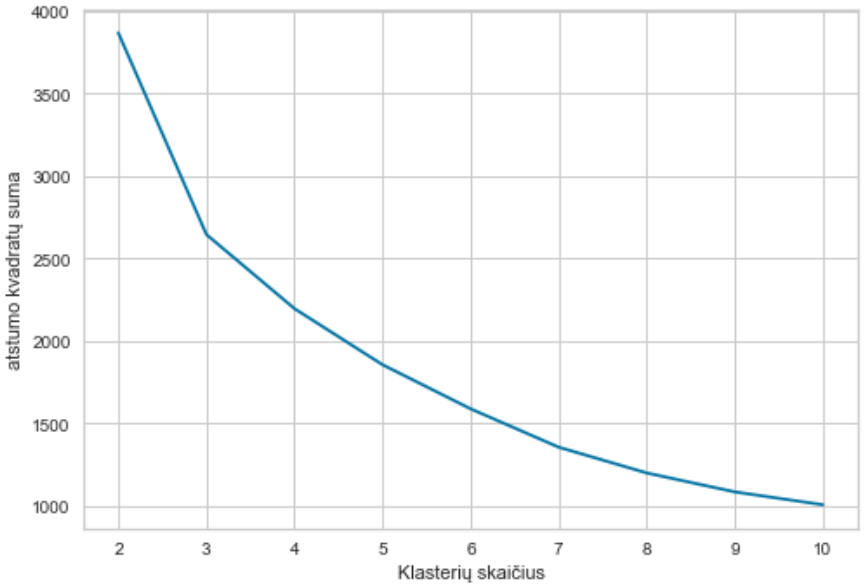
13 lentelė: Dunn testo rezultatai „puslapis1“ kanale.



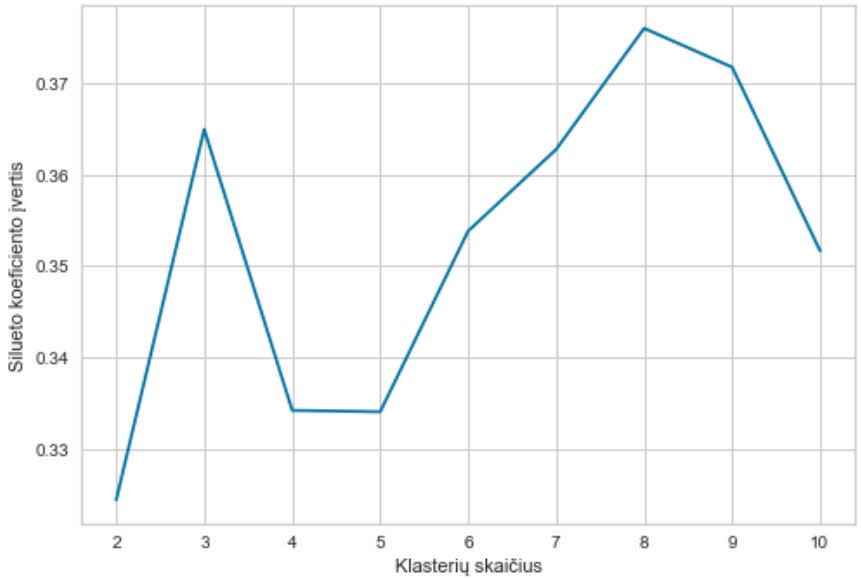
31 pav.: Ieškomas optimalus klasterių skaičius naudojant alkūnės metodą. Klasterizuojami visų kanalų filmukai pagal pirmąjį požymių rinkinį.



32 pav.: Ieškomas optimalus klasterių skaičius naudojant silueto koeficientą. Klasterizuojami visų kanalų filmukai pagal pirmąjį požymių rinkinį.



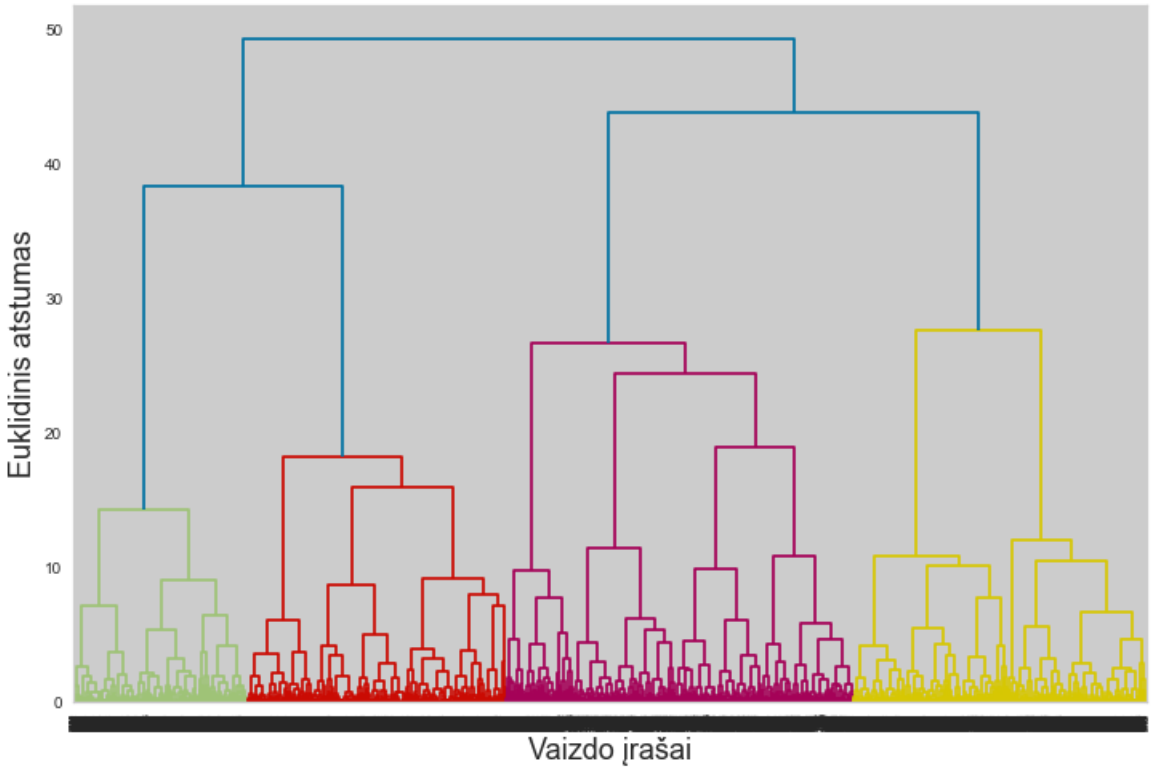
33 pav.: Ieškomas optimalus klasterių skaičius naudojant alkūnės metodą. Klasterizuojami visų kanalų filmukai pagal antrąjį požymių rinkinį.



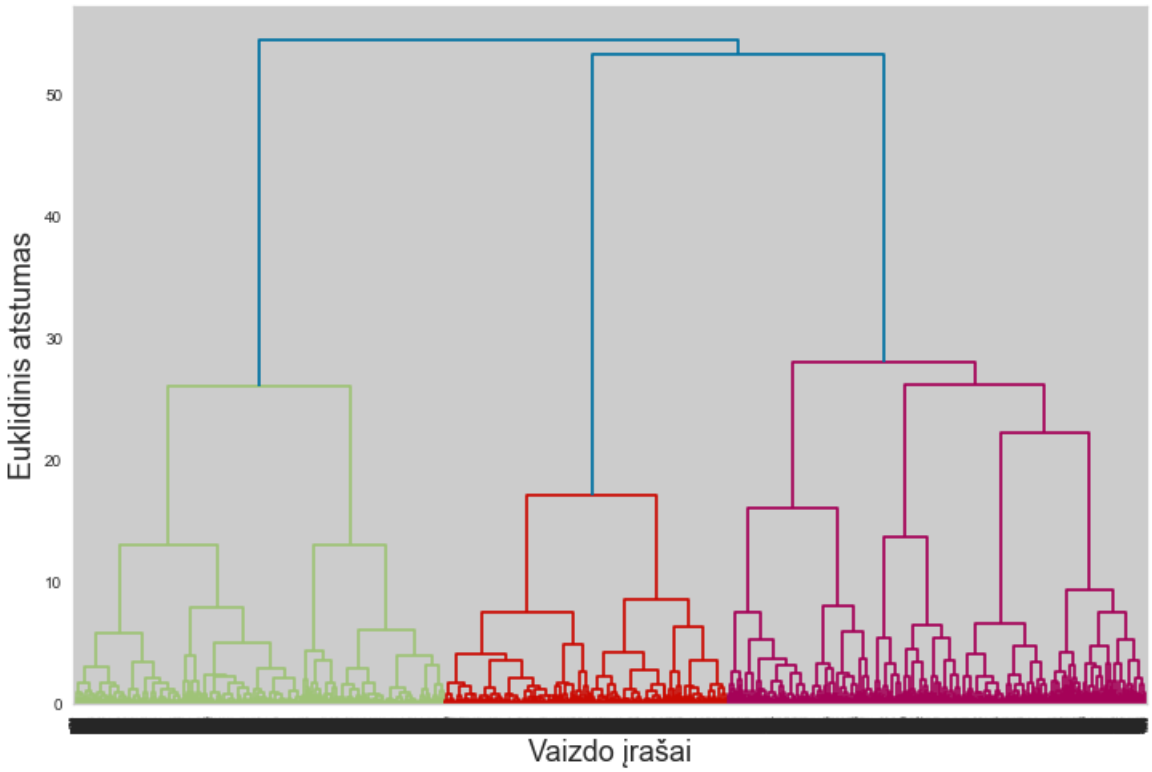
34 pav.: Ieškomas optimalus klasterių skaičius naudojant silueto koeficientą. Klasterizuojami visų kanalų filmukai pagal antrąjį požymių rinkinį.

	Minimumas	Maksimumas	Vidurkis	Mediana	Stand. nuok.
Truk. min	1.000	480.000	13.700	3.000	65.068
PR	0.040	26.050	1.656	0.790	2.318
Per.	4.000e+02	2.571e+08	1.914470e+07	1.116692e+07	2.642134e+07
Vid. žiūr.	9.180	79.150	37.197	36.310	7.879
Kom.	0.000	193974	4319.465	960	12781.035
Mėg.	5.000	4.626729e+06	1.449086e+05	5.753900e+04	3.154859e+05
Pasidal.	0.000	895894.000	21650.475	5022.000	61343.417
Valanda	0.000	21	7.945	8.000	4.219
Sav. d.	1.000	7.000	4.035	4.000	1.795
Tema	1.000	5.000	1.656	3.000	2.318

14 lentelė: Statistinių parametrų lentelė, apibūdinanti jungtinių požymių rinkinio klasterį, kuriame surinkti populiariausi įrašai. Pažymejimai: Truk. min - filmuko trukmė minutėmis, PR - vaizdo įrašų parodymų ir paspaudimų ant jo skaičių santykis, Per.- įrašo peržiūrų skaičius, Vid. žiūr. - vidutinis žiūrėjimo laikas sekundėmis, Kom. - komentarų skaičius, Mėg. - mėgti mygtukų skaičius, Pasidal. - pasidalinimų skaičius, Valanda - įrašo publikavimo valanda, Sav. d. - įrašo publikavimo savaitės diena, Tema - vaizdo įrašo tematika.



35 pav.: Dendrograma, sukurta pagal pirmąjį požymių rinkinį.



36 pav.: Dendrograma, sukurta pagal antrąjį požymių rinkinį.

Python kodas

```
#Užkraunamos reikiamos bibliotekos

import re
import numpy as np
import pandas as pd
from pprint import pprint
import gensim
import spacy
from textblob import TextBlob
from nltk.corpus import stopwords
stop = stopwords.words('english')
from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.pyplot import figure

import warnings
warnings.filterwarnings('ignore')



data= pd.read_csv('/Users/deirosius/Downloads/duomenys.csv', sep=",")

# Kuriami nauji kintamieji, pervardinamos reiksmės lietuvių kalba

data['Publish time'] = data['Publish time'].astype('datetime64[ns]') #ns=Nanosecond
#data['vid_len_mins'] = pd.to_timedelta(data['Duration (seconds)']).astype('timedelta64[m]')
.astype(int)
data['vid_len_mins'] = (data['Duration (sec)'].astype(int)/60).astype(int) #taking only
data['quarter'] = pd.PeriodIndex(data['Publish time'], freq='Q')
data['month'] = pd.PeriodIndex(data['Publish time'], freq='M')
data['CTR'] = round((data['Clicks to play']/data['Impressions'])*100 ,2)
data['Day_of_week'] = data['Publish time'].dt.day_name()

def f(row):
    if row['vid_len_mins'] in range(0,5):
        val = "trumpos trukmės"
    elif row['vid_len_mins'] in range(5,20):
```

```
        val = 'vidutinės trukmės'
    else:
        val = "ilgos trukmės"
    return val
data['video_length_category'] = data.apply(f, axis=1)


data.rename({'3-second video views': 'Video views'}, axis=1, inplace=True)
data['Day_of_week'] = data['Day_of_week'].str.replace('Friday', 'Penktadienis')
data['Day_of_week'] = data['Day_of_week'].str.replace('Wednesday', 'Trečiadienis')
data['Day_of_week'] = data['Day_of_week'].str.replace('Thursday', 'Ketvirtadienis')
data['Day_of_week'] = data['Day_of_week'].str.replace('Monday', 'Pirmadienis')
data['Day_of_week'] = data['Day_of_week'].str.replace('Tuesday', 'Antradienis')
data['Day_of_week'] = data['Day_of_week'].str.replace('Saturday', 'Šeštadienis')
data['Day_of_week'] = data['Day_of_week'].str.replace('Sunday', 'Sekmadienis')

#Tikrinimana, kiek trūkstamų reikšmių yra

total_missing = data.isnull().sum().sort_values(ascending=True)
#Percentage of missing values in the total data
percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(ascending=True)
missing_data = pd.concat([total_missing, percent], axis=1, keys=['Total Missing', 'Percent'])
missing_data.head(30)

#Kuriamos kiekvieno puslapio statistinės lentelės įviariems kintamiesiems

df = data[['Page name', 'Title', 'Duration (sec)', 'vid_len_mins', 'Publish time',
'quarter' ,'month', 'Impressions', 'Clicks to play','CTR','Video views',
'60-second video views', 'Average Seconds viewed', 'Engagements', 'Comments',
'Likes', 'Shares']]


df.describe().iloc[1:,]
df['video_length_category'].value_counts()
df.month.value_counts()

audience = data
audience = audience.drop(['Title', 'Duration (sec)', 'vid_len_mins', 'Publish time',
'quarter', 'month','Impressions', 'Clicks to play','CTR','Video views',
'60-second video views', 'Average Seconds viewed',
           'Engagements', 'Comments', 'Likes', 'Shares', 'Views by top audience (F)',
```

```
           'Views by top audience (M)', 'Views by top audience (65+)',
           'Views by top audience (55-64)', 'Views by top audience (45-54)',
           'Views by top audience (35-44)', 'Views by top audience (25-34)',
           'Views by top audience (18-24)', 'Views by top audience (13-17)',
           'Views by top audience (65+, M)', 'Views by top audience (65+, F)',
           'Views by top audience (55-64, M)', 'Views by top audience (55-64, F)',
           'Views by top audience (45-54, M)', 'Views by top audience (45-54, F)',
           'Views by top audience (35-44, M)', 'Views by top audience (35-44, F)',
           'Views by top audience (25-34, M)', 'Views by top audience (25-34, F)',
           'Views by top audience (18-24, M)', 'Views by top audience (18-24, F)',
           'Views by top audience (13-17, M)', 'Views by top audience (13-17, F)',
           'video_length_category','Post ID','People reached',
           'Unique 60-second video views',], axis=1)
ax = audience.groupby(['Page name']).mean().iloc[: , :]
lst1 = ax.iloc[:1 , :].sort_values(by =['Crafty Panda'],
axis=1, ascending=False).iloc[:, :5].columns
lst2 = ax.iloc[1:2 ,:].sort_values(by =['Handy Panda'],
axis=1, ascending=False).iloc[:, :5].columns
lst3 =ax.iloc[2:3 , :].sort_values(by =['Lady Panda'],
axis=1, ascending=False).iloc[:, :5].columns
ranking = pd.DataFrame(list(zip(lst1,lst3,lst2,)),
               columns =['Crafty Panda','Lady Panda','Handy Panda'])


age = data[['Page name', 'Views by top audience (65+)',
           'Views by top audience (55-64)', 'Views by top audience (45-54)',
           'Views by top audience (35-44)', 'Views by top audience (25-34)',
           'Views by top audience (18-24)', 'Views by top audience (13-17)']]
age = age.groupby(['Page name']).mean()
lst1 = age.iloc[:1 , :].sort_values(by =['Crafty Panda'],
axis=1, ascending=False).iloc[:, :].columns
lst2 = age.iloc[1:2 ,:].sort_values(by =['Handy Panda'],
axis=1, ascending=False).iloc[:, :].columns
lst3 =age.iloc[2:3 , :].sort_values(by =['Lady Panda'],
axis=1, ascending=False).iloc[:, :].columns
ranking_age = pd.DataFrame(list(zip(lst1,lst3,lst2,)),
               columns =['Crafty Panda','Lady Panda','Handy Panda'])


gender = data[['Page name','Views by top audience (F)',
           'Views by top audience (M)']]
```

```
gender = gender.groupby(['Page name']).mean()
lst1 = gender.iloc[:1 , :].sort_values(by =['Crafty Panda'],
axis=1, ascending=False).iloc[:, :].columns
lst2 = gender.iloc[1:2 ,:].sort_values(by =['Handy Panda'],
axis=1, ascending=False).iloc[:, :].columns
lst3 =gender.iloc[2:3 , :].sort_values(by =['Lady Panda'],
axis=1, ascending=False).iloc[:, :].columns

ranking_gender = pd.DataFrame(list(zip(lst1,lst3,lst2,)),
               columns =['Crafty Panda','Lady Panda','Handy Panda'])
ranking_gender

age_gender = age_gender.groupby(['Page name']).mean()
lst1 = age_gender.iloc[:1 , :].sort_values(by =['Crafty Panda'],
axis=1, ascending=False).iloc[:, :5].columns
lst2 = age_gender.iloc[1:2 ,:].sort_values(by =['Handy Panda'],
axis=1, ascending=False).iloc[:, :5].columns
lst3 =age_gender.iloc[2:3 , :].sort_values(by =['Lady Panda'],
axis=1, ascending=False).iloc[:, :5].columns
ranking_age_gender = pd.DataFrame(list(zip(lst1,lst3,lst2,)),
               columns =['Crafty Panda','Lady Panda','Handy Panda'])

# Visa duomenų imtis išskiriama į 3 kanalus

pls1 = data.loc[data['Page name'] == 'XXXXXX']
pls2 = data.loc[data['Page name'] == 'YYYYYY']
pls3 = data.loc[data['Page name'] == 'ZZZZZZ']
pls1_stat = pls1[['Page name', 'Title', 'Duration (sec)', 'vid_len_mins', 'Publish time',
'quarter', 'month','Impressions', 'Clicks to play','CTR',
'Video views', '60-second video views', 'Average Seconds viewed','Engagements',
'Comments', 'Likes', 'Shares']]
pls2_stat = pls2 [['Page name', 'Title', 'Duration (sec)', 'vid_len_mins', 'Publish time',
'quarter', 'month','Impressions', 'Clicks to play','CTR',
'Video views', '60-second video views', 'Average Seconds viewed','Engagements',
'Comments', 'Likes', 'Shares']]
pls3_stat = pls3[['Page name', 'Title', 'Duration (sec)', 'vid_len_mins', 'Publish time',
'quarter', 'month','Impressions', 'Clicks to play','CTR',
'Video views', '60-second video views', 'Average Seconds viewed','Engagements',
'Comments', 'Likes', 'Shares']]

pls1_stat.describe().iloc[1:,]
```

```
pls2_stat.describe().iloc[1:,]
pls3_stat.describe().iloc[1:,]

#Duomenu vizualizavimas

#Vidutinis vaizdo įrašų peržiūrų skaičius, atliekamas kodas su visais kanalais
avg_video_views = XXXXXX.groupby(by=["month"]).aggregate({"Video views":"mean"})
#pasidarom stulpelį, kur vid. video_views pagal mėnesius. Menesiai kaip indeksai padaryti
res = avg_video_views.reset_index() #gaunam du stulpelius: menesių ir peržiūrų
res_wide = res.melt(id_vars="month") #pasikeiciam i wide formata,
prisiskiriam nauja stulpeli variable su reiksmemis video_views, stulpelis video views
tampa value stulpelis
plt.figure(figsize=(15,8))
sns.barplot(x="month", y="value",data=res_wide)
plt.ylabel('Vidutinis peržiūrų skaičius', fontsize = 18)
plt.xlabel('Mėnesis', fontsize = 18)
#plt.title('', fontsize = 20)
plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')
#plt.show()

#Vidutinis peržiūrų skaičius pagal vaizdo įrašų trukmę, atliekamas kodas su visais kanalais
vid_length = XXXXXX.groupby(by=["vid_len_mins"]).aggregate({"Video views":"mean"})
res=vid_length.reset_index()
res_wide=res.melt(id_vars="vid_len_mins")
plt.figure(figsize=(20,5))
sns.barplot(x="vid_len_mins", y="value",data=res_wide)
plt.ylabel('Vidutinis peržiūrų skaičius', fontsize = 18)
plt.xlabel('Vaizdo įrašo trukmė minutėmis', fontsize = 18)
#plt.title('', fontsize=20)
#plt.show()
plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')

#Keliamų vaizdo įrašu trukmės pasiskirstymas, atliekamas kodas su visais kanalais
ax= pd.crosstab(XXXXXX['month'], XXXXXX['video_length_category']).
apply(lambda r: r/r.sum()*100, axis=1)
ax_1 = ax.plot.bar(figsize=(20,10),stacked=True, rot=0)
plt.legend(loc='upper left', bbox_to_anchor=(1,1.025), title="Kategorija")
plt.xlabel('Mėnesis', fontsize = 18)
#plt.title('', fontsize=20)
plt.ylabel('Procentinis pasiskirstymas', fontsize = 18)
for rec in ax_1.patches:
```

```
    height = rec.get_height()
    ax_1.text(rec.get_x() + rec.get_width() / 2,
                rec.get_y() + height / 2,
                "{:.0f}%".format(height),
                ha='center',
                va='bottom')
#plt.show()
plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')

#Vidutinis vartotojų interakcijų skaičius, atliekamas kodas su visais kanalais
interactions = XXXXXX.groupby(by=["month"]).aggregate({"Engagements":"mean"})
res = interactions.reset_index()
res_wide = res.melt(id_vars="month")
plt.figure(figsize=(15,8))
sns.barplot(x="month", y="value",data=res_wide)
plt.ylabel('Interakcijų skaičius', fontsize = 18)
plt.xlabel('Mėnesis', fontsize = 18)
#plt.title('', fontsize=20)
#plt.show()
plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')

# Vidutinė keliamų vaizdo įrašų trukmė, atliekamas kodas su visais kanalais
vid_length = XXXXXX.groupby(by=["month"]).aggregate({"vid_len_mins":"mean"})
res=vid_length.reset_index()
res_wide=res.melt(id_vars="month")
plt.figure(figsize=(15,5))
sns.barplot(x="month", y="value",data=res_wide)
plt.ylabel('Trukmė minutėmis', fontsize = 18)
plt.xlabel('Mėnesis', fontsize = 18)
#plt.title('', fontsize=20)
#plt.show()
plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')

#Savaitės dienomis publikuojamų įrašų pasiskirstymas, atliekamas kodas su visais kanalais
ax= pd.crosstab(XXXXXX['quarter'], XXXXXX['Day_of_week'])
.apply(lambda r: r/r.sum()*100, axis=1)
#procentine israiska, kiek per savaites dienas buvo publikuota video
ax_1 = ax.plot.bar(figsize=(20,10),stacked=True, rot=0)
plt.legend(loc='upper left', bbox_to_anchor=(1,1.025), title="Savaitės diena")
plt.xlabel('Ketvirtis', fontsize = 18)
#plt.title('', fontsize=20)
```

```
plt.ylabel('Procentinė išraiška', fontsize = 18)
#susikuriam procentines israiskas stulpeliuose
for rec in ax_1.patches:
    height = rec.get_height()
    ax_1.text(rec.get_x() + rec.get_width() / 2,
                rec.get_y() + height / 2,
                "{:.0f}%".format(height),
                ha='center',
                va='bottom')
plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')

#Temų modeliavimas

text = data['Description'].values.tolist()
text = [re.sub(r'\S*@\S*\s?', '', sent) for sent in text]
text = [re.sub(r'\s+', ' ', sent) for sent in text]
text = [re.sub(r"\'", "", sent) for sent in text]
text = [re.sub(r'https?:\/\/[A-Za-z0-9\.\/]+', '', sent) for sent in text]
emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U0001F1F2-\U0001F1F4"  # Macau flag
        u"\U0001F1E6-\U0001F1FF"  # flags
        u"\U0001F600-\U0001F64F"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U0001F1F2"
        u"\U0001F1F4"
        u"\U0001F620"
        u"\u200d"
        u"\u2640-\u2642"
        u'\U00010000-\U0010ffff'
        u"\u2600-\u2B55"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\u3030"
        u"\ufe0f"
```

```
        "]+", flags=re.UNICODE)

# text = emoji_pattern.sub(r'', text)
text = [re.sub(emoji_pattern, '', sent) for sent in text]

def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))
        # deacc=True removes punctuations
data_words = list(sent_to_words(text))
def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append(" ".join([token.lemma_ if token.lemma_ not in ['-PRON-']
        else '' for token in doc if token.pos_ in allowed_postags]))
    return texts_out

nlp = spacy.load('en_core_web_sm')
nlp.disable_pipes('ner', 'parser')
data_lemmatized = lemmatization(data_words, allowed_postags=['NOUN', 'VERB'])
vectorizer = CountVectorizer(analyzer='word',
                             min_df=15, # minimum reqd occurences of a word
                             stop_words='english',  # remove stop words
                             lowercase=True,    # convert all words to lowercase
                             token_pattern='[a-zA-Z0-9]{3,}',  # num chars > 3
                            )
data_vectorized = vectorizer.fit_transform(data_lemmatized)
data_dense = data_vectorized.todense()
print("Sparsity: ", ((data_dense > 0).sum()/data_dense.size)*100, "%")

# Build LDA Model
lda_model = LatentDirichletAllocation(n_components=20,
                                      # Number of topics
                                      max_iter=10,
                                      # Max learning iterations
                                      learning_method='online',
                                      random_state=100,
                                      # Random state
                                      batch_size=128,
```

```
                                    # n docs in each learning iter
                                    evaluate_every = -1,
                                    # compute perplexity every n iters, default: Don't
                                    n_jobs = -1,
                                    # Use all available CPUs
                                   )

lda_output = lda_model.fit_transform(data_vectorized)
print(lda_model)
LatentDirichletAllocation(batch_size=128,
                          doc_topic_prior=None,
                          evaluate_every=-1,
                          learning_decay=0.7,
                          learning_method='online',
                          learning_offset=10.0,
                          max_doc_update_iter=100,
                          max_iter=10,
                          mean_change_tol=0.001,
                          n_components=10,
                          n_jobs=-1, perp_tol=0.1,
                          random_state=100,
                          topic_word_prior=None,
                          total_samples=1000000.0, verbose=0
                         )
print("Log Likelihood: ", lda_model.score(data_vectorized))
print("Perplexity: ", lda_model.perplexity(data_vectorized))
pprint(lda_model.get_params())

search_params = {'n_components': [5, 6, 7, 8, 9, 10, 11, 15, 20],
'learning_decay': [.5, .7, .9]}
lda = LatentDirichletAllocation(max_iter=5, learning_method='online', learning_offset=50
,random_state=0)
model = GridSearchCV(lda, param_grid=search_params)
model.fit(data_vectorized)
GridSearchCV(cv=None, error_score='raise',
             estimator=LatentDirichletAllocation(learning_method=None,
             n_components=10, n_jobs=1),
             n_jobs=1,
             param_grid={'n_components': [5, 6, 7, 8, 9, 10, 11, 15, 20],
             'learning_decay': [0.5, 0.7, 0.9]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
```

```
            scoring=None, verbose=0)
best_lda_model = model.best_estimator_
print("Best Model's Params: ", model.best_params_)
print("Best Log Likelihood Score: ", model.best_score_)
print("Model Perplexity: ", best_lda_model.perplexity(data_vectorized))

n_topics = [5, 6, 7, 8, 9, 10, 11, 15, 20]
log_likelyhoods_5 = [round(model.cv_results_['mean_test_score'][index])
for index, gscore in enumerate(model.cv_results_['params']) if
gscore['learning_decay']==0.5]
log_likelyhoods_7 = [round(model.cv_results_['mean_test_score'][index])
for index, gscore in enumerate(model.cv_results_['params']) if
gscore['learning_decay']==0.7]
log_likelyhoods_9 = [round(model.cv_results_['mean_test_score'][index])
for index, gscore in enumerate(model.cv_results_['params']) if
gscore['learning_decay']==0.9]

# Show graph
plt.figure(figsize=(12, 8))
plt.plot(n_topics, log_likelyhoods_5, label='0.5')
plt.plot(n_topics, log_likelyhoods_7, label='0.7')
plt.plot(n_topics, log_likelyhoods_9, label='0.9')
plt.title("")
plt.xlabel("Temų skaičius", fontsize = 18)
plt.ylabel("Log-tikėtinumo funckijos reikšmės", fontsize = 18)
plt.legend(title='Mokinimosi greitis', loc='best')
#plt.show()
plt.savefig('Log-tikėtinumo_funckijos_reikšmės.png',facecolor=(1, 1, 1),
bbox_inches='tight')
lda_output = best_lda_model.transform(data_vectorized)
topicnames = ['Topic' + str(i) for i in range(best_lda_model.n_components)]
docnames = ['Doc' + str(i) for i in range(len(data))]
df_document_topic = pd.DataFrame(np.round(lda_output, 2), columns=topicnames,
index=docnames)
dominant_topic = np.argmax(df_document_topic.values, axis=1)
df_document_topic['dominant_topic'] = dominant_topic
df_topic_keywords = pd.DataFrame(best_lda_model.components_)
df_topic_keywords.columns = vectorizer.get_feature_names()
df_topic_keywords.index = topicnames
df_topic_keywords.head()
def show_topics(vectorizer=vectorizer, lda_model=lda_model, n_words=20):
```

```
    keywords = np.array(vectorizer.get_feature_names())
    topic_keywords = []
    for topic_weights in lda_model.components_:
        top_keyword_locs = (-topic_weights).argsort()[:n_words]
        topic_keywords.append(keywords.take(top_keyword_locs))
    return topic_keywords
topic_keywords = show_topics(vectorizer=vectorizer, lda_model=best_lda_model,
n_words=15)
df_topic_keywords = pd.DataFrame(topic_keywords)
df_topic_keywords.columns = ['Word '+str(i) for i in range(df_topic_keywords.shape[1])]
df_topic_keywords.index = ['Topic '+str(i) for i in range(df_topic_keywords.shape[0])]
df_topic_keywords
Topics = ["Pasidaryk pats, namų amatai ir pramogos",
          "Gyvenimiški patarimai ir gudrybės bei išvaizdos pasikeitimai",
          "Statyba ir automobiliai",
          "Namo ir kiemo restauravimas",
          "Makiažas, grožis ir menas"]
df_topic_keywords["Topics"]=Topics
df_topic_keywords

nlpPred = spacy.load('en_core_web_sm')
nlpPred.disable_pipes('ner', 'parser')
def predict_topic(text, nlp=nlpPred):
    global sent_to_words
    global lemmatization
    mytext_2 = list(sent_to_words(text))
    mytext_3 = lemmatization(mytext_2, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])
    mytext_4 = vectorizer.transform(mytext_3)
    topic_probability_scores = best_lda_model.transform(mytext_4)
    topic = df_topic_keywords.iloc[np.argmax(topic_probability_scores), 1:14]
    .values.tolist()
    infer_topic = df_topic_keywords.iloc[np.argmax(topic_probability_scores), -1]
    return infer_topic, topic, topic_probability_scores

def apply_predict_topic(text):
    text = [text]
    infer_topic, topic, prob_scores = predict_topic(text = text)
    return(infer_topic)
data["Topic"]= data['Description'].apply(apply_predict_topic)
data.head(5)
```

```
#Vizualizavimas pagal temas, atliekamas su visais kanalais

ax= pd.crosstab(XXXXX['month'], XXXXXX['Topic'])
.apply(lambda r: r/r.sum()*100, axis=1)
ax_1 = ax.plot.bar(figsize=(20,10),stacked=True, rot=0)
plt.legend(loc='upper left', bbox_to_anchor=(1,1.025), title="Keywords")
plt.xlabel('Mėnesis', fontsize = 18)
#plt.title('', fontsize=20)
plt.ylabel('Procentinis pasiskirstymas', fontsize = 18)
for rec in ax_1.patches:
    height = rec.get_height()
    ax_1.text(rec.get_x() + rec.get_width() / 2,
                rec.get_y() + height / 2,
                "{:.0f}%".format(height),
                ha='center',
                va='bottom')

plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')

followers_conversion = XXXXX.groupby(by=["Topic"])
.aggregate({"Video views":"mean"})
res = followers_conversion.reset_index()
res_wide = res.melt(id_vars="Topic")
plt.figure(figsize=(15,8))
sns.barplot(x="Topic", y="value",data=res_wide)
plt.ylabel('Peržiūrų skaičius', fontsize = 18)
plt.xlabel('Tema', fontsize = 18)
#plt.title('', fontsize=20)
plt.xticks(rotation=60)
#plt.show()
plt.savefig('',facecolor=(1, 1, 1), bbox_inches='tight')


#Statistiku sudarymas keliamo turinio strategijoms

data['Topic_number'] = data['Topic'].replace({"Pasidaryk pats, namų amatai ir pramogos": 1,
                                              "Gyvenimiški patarimai ir gudrybės
                                              bei išvaizdos pasikeitimai": 2,
                                              "Statyba ir automobiliai": 3,
                                              "Namo ir kiemo restauravimas": 4,
                                              "Makiažas, grožis ir menas": 5})
```

```
data['Page_number'] = data['Page name'].replace({'XXXXX': 1,
                                                 'YYYYY': 2,
                                                 'ZZZZZ': 3})
data['Day_of_week_nu'] = data['Day_of_week'].replace({'Monday': 1,'Tuesday': 2,
                                              'Wednesday': 3,'Thursday': 4,'Friday': 5,
                                                 'Saturday': 6,'Sunday': 7,})

def f(row):
    if row['hour'] in range(0,6):
        val = "Naktis"
    elif row['hour'] in range(6,12):
        val = 'Rytas'
    elif row['hour'] in range(12,18):
        val = "Diena"
    else:
        val = "Vakaras"
    return val

data['hour_category'] = data.apply(f, axis=1)
data['hour_category_nu'] = data['hour_category'].replace({'Naktis': 1,
'Rytas': 2,'Diena': 3,'Vakaras': 4})
crafty = data.loc[data['Page name'] == 'Crafty Panda']
lady = data.loc[data['Page name'] == 'Lady Panda']
handy = data.loc[data['Page name'] == 'Handy Panda']

df1 = data[['Page_number', 'Title', 'Duration (sec)','vid_len_mins','Publish time',
'quarter',
'month','Day_of_week_nu','hour', 'Impressions', 'Clicks to play','CTR',
'Video views', 'Average Seconds viewed','Engagements', 'Comments', 'Likes',
'Shares','Topic','Topic_number',]]

df2 = data[['Page_number', 'Topic','Topic_number','Views by top audience (65+, M)',
'Views by top audience (65+, F)','Views by top audience (55-64, M)',
'Views by top audience (55-64, F)', 'Views by top audience (45-54, M)',
'Views by top audience (45-54, F)', 'Views by top audience (35-44, M)',
'Views by top audience (35-44, F)','Views by top audience (25-34, M)',
'Views by top audience (25-34, F)', 'Views by top audience (18-24, M)',
'Views by top audience (18-24, F)', 'Views by top audience (13-17, M)',
'Views by top audience (13-17, F)']]

XXXXX = df1.loc[data['Page_number'] == 1]
```

```
YYYYY = df1.loc[data['Page_number'] == 2]
ZZZZZ = df1.loc[data['Page_number'] == 3]
XXXXX_1 = df2.loc[data['Page_number'] == 1]
YYYYY_1 = df2.loc[data['Page_number'] == 2]
ZZZZZ_1 = df2.loc[data['Page_number'] == 3]


XXXXX.groupby(['Topic_number']).mean()
XXXXX[(XXXXX['Topic_number'] == 2)]['Day_of_week_nu'].value_counts()
XXXXX[(XXXXX['Topic_number'] == 5)]['hour'].value_counts()
XXXXX_1.groupby(['Topic_number']).mean().iloc[:, :].sort_values(by = 5,
axis=1, ascending=False)
XXXXX.groupby(['Day_of_week_nu']).count()

YYYYY.groupby(['Topic_number']).mean()
YYYYY[(YYYYY['Topic_number'] == 2)]['Day_of_week_nu'].value_counts()
YYYYY[(YYYYY['Topic_number'] == 5)]['hour'].value_counts()
YYYYY_1.groupby(['Topic_number']).mean().iloc[:, :].sort_values(by = 5,
axis=1, ascending=False)
YYYYY.groupby(['Day_of_week_nu']).count()

ZZZZZ.groupby(['Topic_number']).mean()
ZZZZZ[(ZZZZZ['Topic_number'] == 2)]['Day_of_week_nu'].value_counts()
ZZZZZ[(ZZZZZ['Topic_number'] == 5)]['hour'].value_counts()
ZZZZZ_1.groupby(['Topic_number']).mean().iloc[:, :].sort_values(by = 5,
axis=1, ascending=False)
ZZZZZ.groupby(['Day_of_week_nu']).count()

#Hipoteziu tikrinimas
#Testai temoms pagal temų interakcijas
from scipy import stats
data_group1 = df.loc[df['Topic_number'] == 1]['Engagements']
data_group2 = df.loc[df['Topic_number'] == 2]['Engagements']
data_group3 = df.loc[df['Topic_number'] == 3]['Engagements']
data_group4 = df.loc[df['Topic_number'] == 4]['Engagements']
data_group5 = df.loc[df['Topic_number'] == 5]['Engagements']
result = stats.kruskal(data_group1, data_group2, data_group3,
data_group4, data_group5)
print(result)
data = [data_group1, data_group2, data_group3,data_group4, data_group5]
p_values = sp.posthoc_dunn(data, p_adjust = 'bonferroni')
```

```
print(p_values)
p_values > 0.05

ĘTestai peržiūroms pagal publikavimo laiką
data_group1 = df.loc[df['hour_category'] == 'Naktis']['Video views']
data_group2 = df.loc[df['hour_category'] == 'Rytas']['Video views']
data_group3 = df.loc[df['hour_category'] == 'Diena']['Video views']
data_group4 = df.loc[df['hour_category'] == 'Vakaras']['Video views']
result = stats.kruskal(data_group1, data_group2, data_group3,data_group4)
print(result)
data = [data_group1, data_group2, data_group3,data_group4]
p_values = sp.posthoc_dunn(data, p_adjust = 'bonferroni')
print(p_values)
p_values > 0.05

#Testai peržiūroms/interakcijoms pagal publikavimo dieną
data_group1 = df.loc[df['Day_of_week_nu'] == 1]['Video views']
data_group2 = df.loc[df['Day_of_week_nu'] == 2]['Video views']
data_group3 = df.loc[df['Day_of_week_nu'] == 3]['Video views']
data_group4 = df.loc[df['Day_of_week_nu'] == 4]['Video views']
data_group5 = df.loc[df['Day_of_week_nu'] == 5]['Video views']
data_group6 = df.loc[df['Day_of_week_nu'] == 6]['Video views']
data_group7 = df.loc[df['Day_of_week_nu'] == 7]['Video views']
result = stats.kruskal(data_group1, data_group2, data_group3,
data_group4, data_group5, data_group6, data_group7)
print(result)
data_group1 = df.loc[df['Day_of_week_nu'] == 1]['Engagements']
data_group2 = df.loc[df['Day_of_week_nu'] == 2]['Engagements']
data_group3 = df.loc[df['Day_of_week_nu'] == 3]['Engagements']
data_group4 = df.loc[df['Day_of_week_nu'] == 4]['Engagements']
data_group5 = df.loc[df['Day_of_week_nu'] == 5]['Engagements']
data_group6 = df.loc[df['Day_of_week_nu'] == 6]['Engagements']
data_group7 = df.loc[df['Day_of_week_nu'] == 7]['Engagements']
result = stats.kruskal(data_group1, data_group2, data_group3,
data_group4, data_group5, data_group6, data_group7)
print(result)



#Testai kiekvienam kanalui
#Testai temoms pagal žiūrimumą, atliekamas su kiekvienu kanalu
```

```
data_group1 = XXXXX.loc[XXXXX['Topic_number'] == 1]['Video views']
data_group2 = XXXXX.loc[XXXXX['Topic_number'] == 2]['Video views']
data_group3 = XXXXX.loc[XXXXX['Topic_number'] == 3]['Video views']
data_group4 = XXXXX.loc[XXXXX['Topic_number'] == 4]['Video views']
data_group5 = XXXXX.loc[XXXXX['Topic_number'] == 5]['Video views']
print(result)
data = [data_group1, data_group2, data_group3,data_group4, data_group5]
p_values = sp.posthoc_dunn(data, p_adjust = 'bonferroni')
print(p_values)
p_values > 0.05

data_group1 = YYYYY.loc[YYYYY['Topic_number'] == 1]['Video views']
data_group2 = YYYYY.loc[YYYYY['Topic_number'] == 2]['Video views']
data_group3 = YYYYY.loc[YYYYY['Topic_number'] == 3]['Video views']
data_group4 = YYYYY.loc[YYYYY['Topic_number'] == 4]['Video views']
data_group5 = YYYYY.loc[YYYYY['Topic_number'] == 5]['Video views']
result = stats.kruskal(data_group1, data_group2, data_group3,data_group4, data_group5)
print(result)
data = [data_group1, data_group2, data_group3,data_group4, data_group5]
p_values = sp.posthoc_dunn(data, p_adjust = 'bonferroni')
print(p_values)
p_values > 0.05

data_group1 = ZZZZZ.loc[ZZZZZ['Topic_number'] == 1]['Video views']
data_group2 = ZZZZZ.loc[ZZZZZ['Topic_number'] == 2]['Video views']
data_group3 = ZZZZZ.loc[ZZZZZ['Topic_number'] == 3]['Video views']
data_group4 = ZZZZZ.loc[ZZZZZ['Topic_number'] == 4]['Video views']
data_group5 = ZZZZZ.loc[ZZZZZ['Topic_number'] == 5]['Video views']
result = stats.kruskal(data_group1, data_group2, data_group3,data_group4, data_group5)
print(result)
data = [data_group1, data_group2, data_group3,data_group4, data_group5]
p_values = sp.posthoc_dunn(data, p_adjust = 'bonferroni')
print(p_values)
p_values > 0.05


#Klasterizavimas
#Klasterizavimas pagal du požymių rinkinius

df =  data[['vid_len_mins','Topic_number',
           'CTR','Video views', 'Average Seconds viewed',
```

```
          'Comments', 'Likes', 'Shares','hour','Day_of_week_nu',]]

# K-Vidurkių metodas
sc = StandardScaler()
df_scaled = sc.fit_transform(df)
df_normalized = normalize(df_scaled)
from sklearn.cluster import KMeans
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df_normalized)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 11), wcss)
plt.title('')
plt.xlabel('Klasterių skaičius')
plt.ylabel('atstumo kvadratų suma')
plt.savefig('alkune_bendri_irasai_visi_matai.png',facecolor=(1, 1, 1), bbox_inches='tight')
plt.show()
range_n_clusters = [2, 3, 4, 5, 6, 7, 8,9,10]
silhouette_avg = []
for num_clusters in range_n_clusters:
 # initialise kmeans
     kmeans = KMeans(n_clusters=num_clusters)
     kmeans.fit(df_normalized)
     cluster_labels = kmeans.labels_
 # silhouette score
     silhouette_avg.append(silhouette_score(df_normalized, cluster_labels))

plt.plot(range_n_clusters,silhouette_avg, 'bx-')
plt.xlabel('Klasterių skaičius')
plt.ylabel('Silueto koeficiento įvertis')
plt.title('')
plt.savefig('siluetas_bendri_irasai_visi_matai.png',facecolor=(1, 1, 1),
bbox_inches='tight')
plt.show()
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(df_normalized)
df['cluster'] = y_kmeans
df['cluster'].value_counts()
df[(df['cluster'] == 0)].describe()
df[(df['cluster'] == 1)]['vid_len_mins'].value_counts()
```

```
df[(df['cluster'] == 0)]['Topic_number'].value_counts().plot.bar(rot = 0)
plt.xlabel('Tema')
plt.ylabel('Įrašų skaičius klasteryje')
plt.savefig('Įrašų skaičius klasteryje.png',facecolor=(1, 1, 1), bbox_inches='tight')

# Hierarchinis klasterizavimas
sc = StandardScaler()
df_scaled = sc.fit_transform(df)
df_normalized = normalize(df_scaled)
#Using the dendrogram to find the optimal number of clusters
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(12, 8))
dendrogram = sch.dendrogram(sch.linkage(df_normalized, method = 'ward'))
plt.title('')
plt.xlabel('Vaizdo įrašai', fontsize = 18)
plt.ylabel('Euklidinis atstumas', fontsize = 18)
plt.savefig('Dendrograma_bendri_irasai_visi_matai.png',facecolor=(1, 1, 1),
bbox_inches='tight')
plt.show()
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(df_normalized)
df['cluster_h'] = y_hc
df[(df['cluster_h'] == 1)].describe()
df[(df['cluster_h'] == 1)]['Topic_number'].value_counts()
df[(df['cluster_h'] == 1)]['vid_len_mins'].value_counts()
df[(df['cluster_h'] == 1)]['Day_of_week_nu'].value_counts()
df[(df['cluster_h'] == 1)]['hour'].value_counts()

#Klasterizavimas pagal peržiūras ir įrašo kokybes matus
#K-vidurkių metodas
view_vs_video = data[['Page_number','vid_len_mins','hour',
'Video views','Day_of_week_nu','Topic_number']]
XXXXX_view_vs_video = view_vs_video.loc[data['Page_number'] == 1]
YYYYY_view_vs_video = view_vs_video.loc[data['Page_number'] == 2]
ZZZZZ_view_vs_video = view_vs_video.loc[data['Page_number'] == 3]
#Duomenu nomralizavimas ir standartizavimas
sc = StandardScaler()
df_scaled = sc.fit_transform(XXXXX_view_vs_video)
df_normalized = normalize(df_scaled)
from sklearn.cluster import KMeans
wcss = []
```

```
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df_normalized)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

range_n_clusters = [2, 3, 4, 5, 6, 7, 8,9,10]
silhouette_avg = []
for num_clusters in range_n_clusters:
 # initialise kmeans
     kmeans = KMeans(n_clusters=num_clusters)
     kmeans.fit(df_normalized)
     cluster_labels = kmeans.labels_
 # silhouette score
     silhouette_avg.append(silhouette_score(df_normalized, cluster_labels))

plt.plot(range_n_clusters,silhouette_avg, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()

kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(df_normalized)
XXXXX_view_vs_video['cluster'] = y_kmeans
XXXXX_view_vs_video['cluster'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster'] == 4 )].describe()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster'] == 5 )]['hour'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster'] ==4)]['Topic_number'].value_counts()

lady_view_vs_video = view_vs_video.loc[data['Page_number'] == 2]
sc = StandardScaler()
df_scaled = sc.fit_transform(YYYYY_view_vs_video)
df_normalized = normalize(df_scaled)
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
```

```
    kmeans.fit(df_normalized)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
range_n_clusters = [2, 3, 4, 5, 6, 7, 8,9,10]
silhouette_avg = []
for num_clusters in range_n_clusters:
 # initialise kmeans
     kmeans = KMeans(n_clusters=num_clusters)
     kmeans.fit(df_normalized)
     cluster_labels = kmeans.labels_
 # silhouette score
     silhouette_avg.append(silhouette_score(df_normalized, cluster_labels))
plt.plot(range_n_clusters,silhouette_avg, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(df_normalized)
YYYYY_view_vs_video['cluster'] = y_kmeans
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster'] == 4 )].describe()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster'] ==4)]['Topic_number'].value_counts()
sc = StandardScaler()
df_scaled = sc.fit_transform(ZZZZZ_view_vs_video)
df_normalized = normalize(df_scaled)
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df_normalized)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

range_n_clusters = [2, 3, 4, 5, 6, 7, 8,9,10]
```

```
silhouette_avg = []
for num_clusters in range_n_clusters:
 # initialise kmeans
     kmeans = KMeans(n_clusters=num_clusters)
     kmeans.fit(df_normalized)
     cluster_labels = kmeans.labels_
 # silhouette score
     silhouette_avg.append(silhouette_score(df_normalized, cluster_labels))

plt.plot(range_n_clusters,silhouette_avg, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(df_normalized)
ZZZZZ_view_vs_video['cluster'] = y_kmeans
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster'] == 4 )].describe()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster'] ==4)]['Topic_number'].value_counts()

#Hierarchinis klasterizavimas

sc = StandardScaler()
df_scaled = sc.fit_transform(XXXXX_view_vs_video)
df_normalized = normalize(df_scaled)
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(15, 10))
dendrogram = sch.dendrogram(sch.linkage(df_normalized, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Videos')
plt.ylabel('Euclidean distances')
plt.show()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(df_normalized)
XXXXX_view_vs_video['cluster_h'] = y_hc
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)].describe()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['vid_len_mins'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['Topic_number'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['hour'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['Day_of_week_nu'].value_counts()
```

```
sc = StandardScaler()
df_scaled = sc.fit_transform(YYYYY_view_vs_video)
df_normalized = normalize(df_scaled)
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(15, 10))
dendrogram = sch.dendrogram(sch.linkage(df_normalized, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Videos')
plt.ylabel('Euclidean distances')
plt.show()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(df_normalized)
YYYYY_view_vs_video['cluster_h'] = y_hc
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)].describe()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['vid_len_mins'].value_counts()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['Topic_number'].value_counts()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['hour'].value_counts()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['Day_of_week_nu'].value_counts()

sc = StandardScaler()
df_scaled = sc.fit_transform(ZZZZZ_view_vs_video)
df_normalized = normalize(df_scaled)
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(15, 10))
dendrogram = sch.dendrogram(sch.linkage(df_normalized, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Videos')
plt.ylabel('Euclidean distances')
plt.show()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(df_normalized)
ZZZZZ_view_vs_video['cluster_h'] = y_hc
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)].describe()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['vid_len_mins'].value_counts()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['Topic_number'].value_counts()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['hour'].value_counts()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['Day_of_week_nu'].value_counts()
```

```
#Klasterizavimas pagal populiarumo matus ir tematikas
#K-vidurkių metodas
pop_vs_topic = data[['CTR','Video views','Page_number',
                     'Average Seconds viewed', 'Comments', 'Likes', 'Shares','Topic_number']]
XXXXX_view_vs_video = view_vs_video.loc[data['Page_number'] == 1]
YYYYY_view_vs_video = view_vs_video.loc[data['Page_number'] == 2]
ZZZZZ_view_vs_video = view_vs_video.loc[data['Page_number'] == 3]
#Duomenu nomralizavimas ir standartizavimas
sc = StandardScaler()
df_scaled = sc.fit_transform(XXXXX_view_vs_video)
df_normalized = normalize(df_scaled)
from sklearn.cluster import KMeans
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df_normalized)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

range_n_clusters = [2, 3, 4, 5, 6, 7, 8,9,10]
silhouette_avg = []
for num_clusters in range_n_clusters:
 # initialise kmeans
     kmeans = KMeans(n_clusters=num_clusters)
     kmeans.fit(df_normalized)
     cluster_labels = kmeans.labels_
 # silhouette score
     silhouette_avg.append(silhouette_score(df_normalized, cluster_labels))

plt.plot(range_n_clusters,silhouette_avg, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()

kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
```

```
y_kmeans = kmeans.fit_predict(df_normalized)
XXXXX_view_vs_video['cluster'] = y_kmeans
XXXXX_view_vs_video['cluster'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster'] == 4 )].describe()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster'] == 5 )]['hour'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster'] ==4)]['Topic_number'].value_counts()

lady_view_vs_video = view_vs_video.loc[data['Page_number'] == 2]
sc = StandardScaler()
df_scaled = sc.fit_transform(YYYYY_view_vs_video)
df_normalized = normalize(df_scaled)
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df_normalized)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
range_n_clusters = [2, 3, 4, 5, 6, 7, 8,9,10]
silhouette_avg = []
for num_clusters in range_n_clusters:
 # initialise kmeans
     kmeans = KMeans(n_clusters=num_clusters)
     kmeans.fit(df_normalized)
     cluster_labels = kmeans.labels_
 # silhouette score
     silhouette_avg.append(silhouette_score(df_normalized, cluster_labels))
plt.plot(range_n_clusters,silhouette_avg, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(df_normalized)
YYYYY_view_vs_video['cluster'] = y_kmeans
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster'] == 4 )].describe()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster'] ==4)]['Topic_number'].value_counts()
sc = StandardScaler()
```

```
df_scaled = sc.fit_transform(ZZZZZ_view_vs_video)
df_normalized = normalize(df_scaled)
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df_normalized)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

range_n_clusters = [2, 3, 4, 5, 6, 7, 8,9,10]
silhouette_avg = []
for num_clusters in range_n_clusters:
 # initialise kmeans
     kmeans = KMeans(n_clusters=num_clusters)
     kmeans.fit(df_normalized)
     cluster_labels = kmeans.labels_
 # silhouette score
     silhouette_avg.append(silhouette_score(df_normalized, cluster_labels))

plt.plot(range_n_clusters,silhouette_avg, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(df_normalized)
ZZZZZ_view_vs_video['cluster'] = y_kmeans
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster'] == 4 )].describe()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster'] ==4)]['Topic_number'].value_counts()

#Hierarchinis klasterizavimas

sc = StandardScaler()
df_scaled = sc.fit_transform(XXXXX_view_vs_video)
df_normalized = normalize(df_scaled)
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(15, 10))
```

```
dendrogram = sch.dendrogram(sch.linkage(df_normalized, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Videos')
plt.ylabel('Euclidean distances')
plt.show()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(df_normalized)
XXXXX_view_vs_video['cluster_h'] = y_hc
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)].describe()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['vid_len_mins'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['Topic_number'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['hour'].value_counts()
XXXXX_view_vs_video[(XXXXX_view_vs_video['cluster_h'] == 4)]['Day_of_week_nu'].value_counts()


sc = StandardScaler()
df_scaled = sc.fit_transform(YYYYY_view_vs_video)
df_normalized = normalize(df_scaled)
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(15, 10))
dendrogram = sch.dendrogram(sch.linkage(df_normalized, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Videos')
plt.ylabel('Euclidean distances')
plt.show()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(df_normalized)
YYYYY_view_vs_video['cluster_h'] = y_hc
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)].describe()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['vid_len_mins'].value_counts()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['Topic_number'].value_counts()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['hour'].value_counts()
YYYYY_view_vs_video[(YYYYY_view_vs_video['cluster_h'] == 4)]['Day_of_week_nu'].value_counts()

sc = StandardScaler()
df_scaled = sc.fit_transform(ZZZZZ_view_vs_video)
df_normalized = normalize(df_scaled)
import scipy.cluster.hierarchy as sch
plt.figure(figsize=(15, 10))
```

```
dendrogram = sch.dendrogram(sch.linkage(df_normalized, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Videos')
plt.ylabel('Euclidean distances')
plt.show()
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(df_normalized)
ZZZZZ_view_vs_video['cluster_h'] = y_hc
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)].describe()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['vid_len_mins'].value_counts()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['Topic_number'].value_counts()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['hour'].value_counts()
ZZZZZ_view_vs_video[(ZZZZZ_view_vs_video['cluster_h'] == 4)]['Day_of_week_nu'].value_counts()
```