VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

EKONOMETRIJOS BAKALAURO STUDIJŲ PROGRAMA

**Bakalauro baigiamasis darbas**

# YouTube vaizdo įrašų populiarumo vertinimas naudojant viešai prieinamus metaduomenis

## Assessing the Popularity of YouTube Videos Using Publicly Available Metadata

Gabrielė Ruminavičiūtė

Darbo vadovas Dr. Jolita Bernatavičienė

**Vilnius, 2021**

# Turinys

## Padėka

Autorė dėkoja Vilniaus universiteto Matematikos ir informatikos fakulteto Informacinių technologijų atviros prieigos centrui už suteiktus HPC išteklius šio darbo skaičiavimams atlikti.

# YouTube vaizdo įrašų populiarumo vertinimas naudojant viešai prieinamus metaduomenis


### Santrauka

YouTube vaizdo įrašų duomenų augimo tempai kelia daug problemų sprendžiant vaizdo įrašų populiarumo vertinimo ir klasifikavimo uždavinius. Vis dar nėra nusistovėjusios metodologijos, kokiais kriterijais remiantis, turi būti apibrėžiamas vaizdo įrašų populiarumas. Šiame darbe analizuojami YouTube duomenų programos programavimo sąsajos pagalba surinkti vaizdo įrašų metaduomenys ir išvestiniai jų parametrai. Įvertinus reikšmingus vidurkių skirtumus tarp populiarumo grupių statistiniais testais, atlikti duomenų klasterizavimo algoritmai: $k$-vidurkių ir saviorganizuojančių neurorinių tinklų, skirti patikrinti populiarumo grupių susidarymo tendencijas. Taip pat atliktas vaizdo įrašų populiarumo grupių klasifikavimas naudojant tris mašininio mokymosi metodus: atraminių vektorių, atsitiktinio miško ir daugialypės logistinės regresijos klasifikatorius. Atlikta algoritmų lyginamoji analizė bei atrinkti populiarumo lygį tiksliausiai klasifikuojantys požymių rinkiniai.



**Raktiniai žodžiai:** YouTube, populiarumas, klasterizavimas, klasifikavimas, atraminių vektorių klasifikatorius, atsitiktinis miškas, daugialypė logistinė regresija, mašininis mokymasis, statistinė analizė


# Assessing the Popularity of YouTube Videos Using Publicly Available Metadata


### Abstract

The growth rate of YouTube video data poses a number of challenges to assess and classify the popularity of YouTube videos. There is still no established methodology for defining the popularity of videos according to which criteria. This paper analyses the video metadata and derived parameters collected by the YouTube Data Application Programming Interface. After assessing significant differences in means among popularity groups by using statistical tests, data clustering algorithms: $k$-means and self-organising neural networks were applied, to search for trends in the formation of popularity groups. Classification of video popularity was also performed using three machine learning methods: support vector machine, random forest and multinomial logistic regression classifiers. A comparative analysis of the algorithms was carried out and the feature sets that most accurately classify the popularity level were selected.



**Key words:** YouTube, popularity, clustering, classification, support vector machine, random forest, multinomial logistic regression, machine learning, statistical analysis

# Iliustracijų sąrašas



# Lentelių sąrašas

# 1 Įvadas

YouTube tinklalapis yra viena populiariausių vaizdo įrašų platinimo platformų, turinčių virš dviejų milijardų naudotojų, žiūrinčių vaizdo įrašų turinį virš milijardo valandų kiekvieną dieną. Taip pat, kiekvieną minutę į šį tinklalapį yra įkeliama apie 500 valandų įvairaus vaizdo įrašų turinio [29]. Neabejotina, kad toks greitai augantis vaizdo įrašų duomenų kiekis sudaro vis sunkesnes galimybes vaizdo įrašus surūšiuoti ir atskirti, kurie vaizdo įrašai yra populiarūs bei nustatyti, kokiais kriterijais remiantis turi būti apibrėžiamas populiarumas. Taip pat nėra aiškiai apibrėžtos populiarumo sąvokos - kiekvienas tyrėjas tyrime populiarumo apibrėžimą pateikia savaip. Tiek vaizdo įrašo kūrėjams, tiek YouTube kompanijai informacija apie įrašų populiarumą yra svarbi, kadangi abi šalys uždirba iš reklamų - kuo vaizdo įrašas populiaresnis ir pasiekia daugiau auditorijos, tuo daugiau pinigų uždirbama iš tame vaizdo įraše leidžiamos reklamos. Ši informacija reikalinga ir įvairioms reklamos kompanijoms - populiariuose vaizdo įrašuose rodoma reklama matoma daugiau kartų gali padidinti įmonės pelną.

Viena iš pagrindinių metrikų, pagal kurią vaizdo įrašas yra laikomas populiariu yra vaizdo įrašo peržiūros. Kita vertus, populiarumas ir vaizdo įrašų peržiūrų skaičius nėra sinonimai. Labai svarbus kriterijus populiarumui nusakyti yra ir laikas. Dviejų vaizdo įrašų klasifikavimas į tą pačią populiarumo grupę, turint tą patį įrašų peržiūrų skaičių, bet skirtingą įrašų gyvavimo viešoje erdvėje trukmę, būtų neteisingas. Dėl šios priežasties, populiarumas gali būti apibrėžtas skirtingomis kategorijomis (nepopuliarūs, populiarūs, labai populiarūs ir t.t.) atsižvelgiant į vaizdo įrašo peržiūrų skaičių bei gyvavimo trukmę internete. Tačiau kyla ir kitas klausimas - kokie kiti papildomi požymiai gali daryti įtaką vaizdo įrašo populiarumui?

Šio darbo tikslas yra klasifikuoti YouTube vaizdo įrašus pagal populiarumo grupes, naudojant viešai prieinamus metaduomenis bei atlikti įvairių klasifikavimo algoritmų lyginamąją analizę. Tyrimui atlikti išsikelti tokie uždaviniai:

1. Išsiaiškinti, kokie panašaus pobūdžio tyrimai atlikti YouTube vaizdo įrašų populiarumo tematika.
2. Naudojantis viešai prieinama YouTube Duomenų programos programavimo sąsaja surinkti įvairių kanalų vaizdo įrašų metaduomenis bei iš jų sukonstruoti papildomus aiškinamuosius vaizdo įrašų kintamuosius.
3. Gautiems vaizdo įrašų metaduomenims atlikti statistinę analizę bei suskirstyti vaizdo įrašus į populiarumo grupes pagal pasirinktus požymius bei metrikas.
4. Naudojant grupavimo algoritmus išsiaiškinti klasių susidarymo tendencijas, įvertinti skirtumus tarp sudarytų klasių bei ryšį su vaizdo įrašų populiarumu.
5. Taikant mašininio mokymosi algoritmus atlikti YouTube vaizdo įrašų klasifikavimą bei įvertinti, kokie požymiai daro įtaką vaizdo įrašų populiarumui.
6. Apibendrinti atlikto tyrimo rezultatus bei parengti išvadas ir rekomendacijas ateities darbams.

Šiame tyrime, išsikeltam tikslui pasiekti, naudoti metodai ir modeliai:

- Mokslinės literatūros analizė.
- Pirminė duomenų analizė: statistinė duomenų analizė, dispersinė analizė MANOVA ir ANOVA;
- Klasterinė vaizdo įrašų populiarumo analizė, naudojant $k$-vidurkių ir saviorganizuojančių neuroninių tinklų algoritmus.
- Klasifikavimo algoritmai YouTube duomenims: atraminių vektorių klasifikatorius, atsitiktinio miško ir daugialypės logistinės regresijos klasifikatoriai.

Tyrimą sudaro dvi pagrindinės dalys: teorinė ir praktinė. Teorinėje dalyje pateikiama literatūros apžvalga, kurioje pristatomi atlikti tyrimai, nagrinėjantys YouTube populiarumo tematiką; aprašomi tyrime naudojami duomenys bei kokiu būdu jie surinkti; metodologijos skyrius yra skirtas taikomų metodų ir algoritmų aprašymui.

Praktinė dalis yra metodologijos pritaikymas surinktiems duomenims: ji susideda iš pirminės duomenų analizės (aprašomosios statistikos ir statistinių testų rezultatų) bei klasterizavimo ir klasifikavimo algoritmų rezultatų. Darbo pabaigoje pateiktos atlikto tyrimo išvados bei rekomendacijos, naudotas literatūros sąrašas ir priedai.

Surinktų duomenų pagrindu, dalis atliktų populiarumo tyrimo rezultatų pristatyti konferencijoje "Lietuvos magistrantų informatikos ir IT tyrimai" 2021 gegužės 14 dieną bei paruošta publikacija "YouTube vaizdo įrašų populiarumo vertinimas naudojant viešai prieinamus metaduomenis" publikuota konferencijos leidinyje [11].

## 1.1 Literatūros apžvalga

YouTube vaizdo įrašų populiarumo klausimas nagrinėjant papildomus metaduomenis nėra lengva užduotis. Daugelis svarbių metrikų, kurios iš dalies galėtų pasakyti, kokie požymiai lemia konkretaus vaizdo įrašo populiarumą, nėra viešai prieinami, todėl publikuoti tyrimai atlikti panašia tema taip pat yra labai riboti. Kita vertus, vaizdo įrašų populiarumas įtraukiant įvairius metaduomenis yra plačiai nagrinėjama tema. Vienas iš klasifikatorių, naudojamas YouTube vaizdo įrašų populiarumo prognozavimui yra atraminių vektorių klasifikatorius (angl. *Support Vector Machine*, SVM). [28] pateiktas tyrimas parodė, kad naudojant vizualinius vaizdo įrašų požymius (vaizdo įrašų trukmė, kadrų skaičius, rezoliucija, spalvos, veido ir teksto pasirodymas kadruose, miniatiūros kokybė ir t.t) galima nuspėti vaizdo įrašo populiarumą, o norint pagerinti šio klasifikatoriaus prognozavimo tikslumą, dinaminiai požymiai, tokie kaip peržiūrų, „patinka“, pasidalinimų ir komentarų skaičius taip pat turėtų būti įtraukti. Naudojant įvairius mašininio mokymosi metodus nustatyta, jog penki požymiai daro įtaką vaizdo įrašų populiarumui – peržiūros pirmąją dieną, kanalo prenumeratorių skaičius, miniatiūros kontrastas, paieškos sistemos „Google“ paspaudimai ir raktinių žodžių kiekis [8].

Kita vertus, populiarumo prognozavimas yra labai svarbi užduotis ankstyvajame vaizdo įrašo gyvavimo laikotarpyje, todėl [23] naudojami daugialypės tiesinės regresijos modeliai, skirti nuspėti ateities vaizdo įrašo peržiūras, naudojant praeities reikšmes. Vaizdo įrašų populiarumas yra vertinamas ir neturint apie vaizdo įrašus jokios auditorijos įsitraukimo istorijos arba dinaminių požymių. Tokiam populiarumo prognozavimui naudojami sudėtingesni mašininio mokymosi klasifikatoriai [2]: klasifikatorių junginys (angl. *Ensemble of Classifiers*) tiksliau nustato populiarumo klases negu individualūs Naiviojo Bajeso, SVM, logistinės regresijos, neuroninių tinklų ir atsitiktinio miško (angl. *Random Forest*, RF) klasifikatoriai.

Nagrinėjant mokslinę literatūrą pastebėta, kad vaizdo įrašų metaduomenų požymių rinkiniai skiriasi, todėl apibendrinti gautus rezultatus sudėtinga. Tyrimuose naudojami metaduomenų rinkiniai yra labai įvairūs: jie rūšiuojasi į dinaminius - kintančius laike - požymius (peržiūrų, komentarų, „patinka“, „nepatinka“ paspaudimų skaičius), miniatiūrų kokybės vertinimo požymių grupę (miniatiūros kokybė, šviesumas, išsiliejimas ir kontrastas), ir tekstinio turinio kokybės vertinimo požymių grupes (simbolių skaičius pavadinime, aprašymas, raktinių žodžių kiekis). Todėl analizė gali būti atliekama naudojant bendrus požymių rinkinius, arba pasirenkant tik vieną tikslinę požymių grupę, pavyzdžiui YouTube duomenų klasifikavimui [12] naudojami pasitelkiant tekstinius metaduomenis (aprašymą ir pavadinimą).

Vaizdo turinio atskyrimas į populiarumo klases yra sunkus uždavinys ne tik YouTube platformoje. Naudojant Netflix - internetinės kino filmų ir serialų transliavimo platformos - duomenis, šios platformos naudotojai sugrupuoti $k$ - vidurkių algoritmu tam, kad kiekvienoje grupėje identifikuoti populiariausius vaizdo įrašus jų ankstyvoje gyvavimo fazėje [16]. Kita vertus, grupavimo algoritmai gali būti naudojami ne tik populiarumo klasių identifikavimui: YouTube vaizdo įrašų platforma pasižymi dideliu kiekiu duomenų - įvairūs kanalai publikuoja vaizdo įrašus įvairiose kategorijose. Todėl kyla natūralus klausimas - kokių kategorijų vaizdo įrašai sulaukia palankios auditorijos vertinimo? [24] autoriai parodė, jog naudojant $k$-vidurkių algoritmą, daugiausia "patinka" paspaudimų

skaičiaus sulaukia muzikos kategorijoje esantys vaizdo įrašai.

# 2 Duomenys

Duomenys tyrimui yra surinkti YouTube Duomenų programavimo sąsajos pagalba (angl. *YouTube Data API*). Ši sąsaja leidžia prieigą prie įvairių viešų YouTube kanalų ir vaizdo įrašų statistikos. Duomenys yra pateikiami kaip JSON (angl. *Java Script Object Notation*) objektai.

Kiekvienam kreipimuisi į šį API reikia pateikti Google Developer Console sugeneruotą OAuth 2.0 žetoną. OAuth 2.0 leidžia vartotojui pasidalinti arba gauti tam tikrus duomenis nepasidalinant jautrios asmeninės informacijos (slaptažodžių, vartotojo vardo ir panašiai). Duomenų rinkimas vykdomas R programavimo kalba parašyta programėle (angl. *script*), kuri leidžiama kiekvieną dieną iki kol surinktas reikiamas kiekis duomenų.

Šios programavimo sąsajos naudojimasis vartotojui yra ribojamas. Kiekvienas kvietimasis į YouTube Duomenų API "kainuoja" tam tikras kvotas. Išnaudojus suteiktą kvotų skaičių - 10000 per dieną - sąsaja naudotis galima praėjus parai. Verta paminėti, kad 10000 kvotų nereiškia galimą gauti duomenų kiekį. Išnaudojus leistiną kiekį kvotų, vidutiniškai surenkama apie 3500 eilučių vaizdo įrašų metaduomenų.

Taigi, naudojantis viešai prieinama YouTube duomenų aplikacijos programavimo sąsaja, surinkti šie vaizdo įrašų metaduomenys:

- vaizdo įrašo pavadinimas;
- aprašymas;
- raktažodžiai;
- peržiūrų, komentarų, "patinka" ir "nepatinka" paspaudimų skaičius;
- miniatiūra;
- vaizdo įrašo įkėlimo data;
- vaizdo įrašo kategorija;
- kanalo, kuriame patalpintas vaizdo įrašas, pavadinimas;
- kanalo, kuriame patalpintas vaizdo įrašas, prenumeratorių skaičius.

Iš šių duomenų sukonstruoti papildomi kintamieji, kurie yra naudojami tyrime (žr. 1 lent.).

| Kintamasis | Apibūdinimas | Tipas | Galimos reikšmės (imties plotis) |
|---|---|---|---|
| channel_subscriber_count | Kanalo prenumeratorių skaičius | Kiekybinis | [0; 54800000] |
| age | Vaizdo įrašo amžius dienomis | Kiekybinis | [1; 5569] |
| avg_daily_views | Vidutinis vaizdo įrašo peržiūrų skaičius per dieną | Kiekybinis | [0; 3543465] |
| avg_daily_likes | Vidutinis "patinka" paspaudimų skaičius per dieną | Kiekybinis | [0; 207946] |
| avg_daily_dislikes | Vidutinis "nepatinka" paspaudimų skaičius per dieną | Kiekybinis | [0; 24110] |
| avg_daily_comments | Vidutinis komentarų skaičius per dieną | Kiekybinis | [0; 20661] |
| description_exists | Egzistuoja vaizdo įrašo aprašymas | Dichotominis | 1 arba 0 |
| chr_title | Simbolių pavadinime skaičius | Kiekybinis | [1; 100] |
| title_words | Žodžių pavadinime skaičius | Kiekybinis | [1; 25] |
| keywords_count | Raktinių žodžių skaičius | Kiekybinis | [0; 115] |
| thumbnail_quality | Miniatiūros kokybė | Kiekybinis | [0,0028, 99,9595] |
| thumbnail_blurriness | Miniatiūros išsiliejimas | Kiekybinis | [0; 42075] |
| thumbnail_contrast | Miniatiūros kontrastas | Kiekybinis | [0; 124,72] |
| thumbnail_brightness | Miniatiūros šviesumas | Kiekybinis | [0; 255] |
| popularity | Populiarumo grupė | Nominalus | 0, 1, 2, 3 |

1 lentelė: Kintamieji

Taip pat du kokybiniai kintamieji:

- video_category - vaizdo įrašo kategorija (žr. 17 lent.);
- channel - kanalas, kuriame patalpintas vaizdo įrašas (nuo 1 iki 629).

Taigi, iš viso turima 17 kintamųjų, kurių 13 - kiekybiniai, o likę 4 - kategoriniai. Visą duomenų imtį sudaro virš 130 tūkstančių vaizdo įrašų, patalpintų 629-iuose skirtinguose kanaluose. Lentelėje 2 pateiktas duomenų rinkinio apibendrinimas.

| | |
|---|---|
| Vaizdo įrašai | 132321 |
| Kanalai | 629 |
| Vidutinis vaizdo įrašo amžius (dienomis) | 1089 |
| Vidutinis peržiūrų skaičius (vienam vaizdo įrašui) | 4064744 |

2 lentelė: Duomenų rinkinio apibendrinimas

Atsižvelgiant į vaizdo įrašų peržiūrų ir gyvavimo trukmės internete ryšį su populiarumu, visa duomenų imtis yra suskirstyta į keturias populiarumo grupes, naudojant vidutinio vaizdo įrašo peržiūrų skaičiaus per dieną kvartilius tam, kad išvengti nesubalansuotos duomenų imties, jog kiekviena grupė duomenyse nepasitaikytų dažniau negu kita. Pirmąją grupę - nepopuliarų - sudaro vaizdo įrašai, kurių vidutinės peržiūros yra ne didesnės negu 48,6. Antrosios - vidutinio populiarumo - grupės vidutinės peržiūros per dieną patenka tarp (48,6; 494,4]. Populiarių vaizdo įrašų vidutinės peržiūros per dieną (trečiosios grupės) patenka į intervalą (494,4; 3209,4]. Visi likusieji vaizdo įrašai sudaro ketvirtąją - labai populiarių - grupę, kurių vidutinės peržiūros per dieną yra didesnės už 3209,4.

# 3 Metodologija

## 3.1 Metaduomenų konstravimas

Pagrindinių kintamųjų, paaiškinančių vaizdo įrašo žiūrovų įsitraukimą, apskaičiavimas:

1. vidutinės peržiūros per dieną: santykis visų vaizdo įrašo peržiūrų ir vaizdo įrašo amžiaus (skirtumas tarp vaizdo įrašo duomenų nusiskaitymo dienos ir vaizdo įrašo publikavimo viešai dienos);

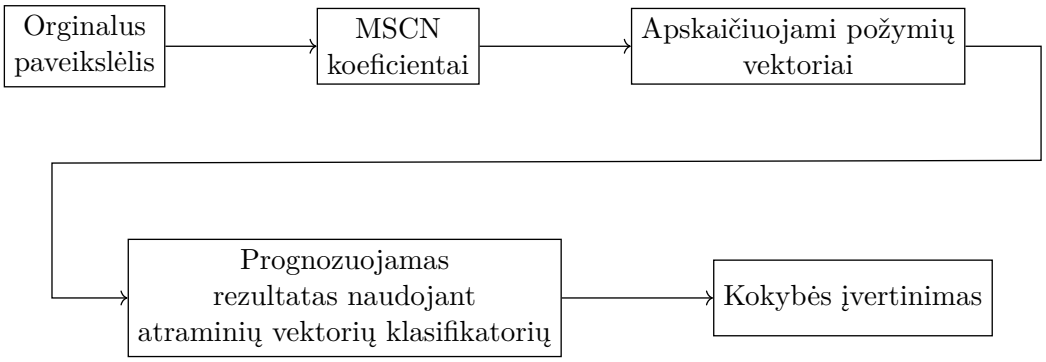
2. vidutinis komentarų skaičius per dieną: santykis visų komentarų ir vaizdo įrašo amžiaus;

3. vidutinis "patinka" paspaudimų skaičius per dieną: santykis visų "patinka" paspaudimų skaičiaus ir vaizdo įrašo amžiaus;

4. vidutinis "nepatinka" paspaudimų skaičius per dieną: santykis visų "nepatinka" paspaudimų skaičiaus ir vaizdo įrašo amžiaus;

Miniatiūrų įvertinimui naudojami kokybės, išsiliejimo, kontrasto ir šviesumo kintamieji.

Miniatiūros kokybė įvertinta naudojant aklojo/be nuorodų vaizdų erdvinės kokybės vertinimo metodiką (angl. *Blind/Referenceless Image Spatial* Quality Evaluator, trumpai BRISQUE) [21]. Šis kokybės vertinimas remiasi 1 pav. algoritmu.




1 pav.: BRISQUE algortimas

BRISQUE algoritmui pateiktas paveikslėlis normalizuojamas remiantis vidutiniu išskaičiuotu kontrasto normalizavimo balu (angl. *Mean Substracted Contrast Normalization*, MSCN). Norint suskaičiuoti MSCN koeficientus, paveikslėlio intensyvumas $I(i,j)$ kiekvienam pikseliui $(i,j)$ yra transformuojamas į skaistumą (angl. *luminance*) $\hat{I}(i,j)$.

$$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma^2(i,j) + C},$$

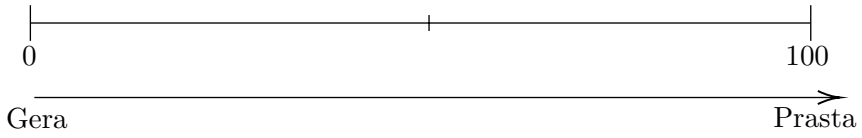
čia $i \in 1,2,...M$, $j \in 1,2,...N$ ($M$ ir $N$ yra paveikslėlio aukštis ir plotis atitinkamai). $\mu(i,j)$ ir $\sigma^2(i,j)$ yra atitinkamai vietinio lauko vidurkis ir dispersija, $C = 1$ konstanta. Vietinis lauko vidurkis ($\mu$) yra originalaus paveikslėlio Gauso išsiliejimas, o dispersija ($\sigma^2$) yra kvadratinis skirtumas tarp originalaus paveikslėlio intensyvumo ir $\mu$.

Taip pat, apskaičiuojami kaimyninių pikselių santykiai, naudojant poras tarp MSCN ir paslinktų MSCN koeficientų:

$$\begin{aligned} H(i,j) &= \hat{I}(i,j)\hat{I}(i,j+1), \\ V(i,j) &= \hat{I}(i,j)\hat{I}(i+1,j), \\ D1(i,j) &= \hat{I}(i,j)\hat{I}(i+1,j+1), \\ D2(i,j) &= \hat{I}(i,j)\hat{I}(i+1,j-1), \end{aligned}$$

kur $H, V, D1, D2$ yra atitinkamai horizontali, vertikali, kairiosios įstrižainės ir dešiniosios įstrižainės kryptys.

Naudojant gautus penkis paveikslėlius yra suskaičiuojami požymių vektoriai, $36 \times 1$ dydžio: pirmieji du vektoriaus elementai yra MSCN vaizdo pritaikymas apibendrintam Gauso pasiskirstymui. Kitiems šešiolikai vektoriaus elementams yra apskaičiuojamas asimetrinis apibendrintas Gauso pasiskirstymas naudojant keturis porinius kintamuosius. Turint iš viso 18 vektoriaus elementų, gautas paveikslėlis yra sumažinimas per pusę originalaus paveikslėlio dydžio ir procesas vėl kartojamas, kol gaunama 18 naujų vektoriaus elementų. Taigi, iš viso turimi 36 elementai. Tuomet naudojant autorių A. Mittal, Anish, A. Moorthy ir A. Bovik apmokytą atraminių vektorių klasifikatorių, apskaičiuojama paveikslėlio kokybė. 2 grafikas nurodo paveikslėlio kokybės vertinimo skalę: kuo mažesnis įvertinimas, tuo kokybė yra geresnė.




2 pav.: Paveikslėlio kokybės įvertinimas

Miniatiūros išsiliejimas skaičiuojamas naudojant Laplaso operatorių [22]. Paveikslėlis paverčiamas į pilkų tonų atvaizdą, tuomet apskaičiuojamas turimo paveikslėlio ir branduolio konvoliucijos rezultatas, naudojant branduolį:

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Galutiniame rezultate yra apskaičiuojama dispersija. Laplaso operatorius naudojamas briaunų nustatymui, nes pabrėžia atvaizdo regionų staigų kitimą, todėl aukštesnės dispersijos paveiksliukai turi daugiau staigių kitimų, o tai reiškia didesnį skaičių briaunų. Kuo daugiau nustatoma paveikslėlyje kraštinių, tuo paveikslėlis yra ryškesnis ir mažiau išsiliejęs.

Paveikslėlio kontrastas yra apskaičiuojamas remiantis pilkų tonų atvaizdo pikselio intensyvumo kvadratiniu vidurkiu (angl. *Root Mean Square*):

$$\sqrt{\frac{1}{MN}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\Big(I(i,j)) - \bar{I}\Big)^2}, \tag{3.1}$$

čia $\bar{I}$ yra pikselių intensyvumo vidurkis.

Paveikslėlio šviesumas yra apskaičiuojamas tokiu pat principu kaip ir 3.1, tačiau naudojamas ne pikselio intensyvumas, o šviesumas:

$$\sqrt{\frac{1}{MN}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\Big(\delta(i,j)) - \bar{\delta}\Big)^2},$$

čia $\bar{\delta}$ yra pikselių šviesumo vidurkis.

## 3.2 Statistiniai testai

Darbe naudojami statistiniai testai, skirti išsiaiškinti skirtumus tarp populiarumo grupių.

### 3.2.1 Daugiamatė dispersinė analizė MANOVA

MANOVA (angl. *Multivariate Analysis of Variance*) [10] tikrina skirtumus tarp dviejų arba daugiau vidurkių vektorių, t.y vienfaktorinė dispersinė analizė skirta palyginti kelių kokybinių faktoriaus reikšmėmis nusakomų populiacijų (grupių) vienmačius kiekybinio atsako vidurkius. Tikrinama hipotezė, jog visų (daugiamačių) grupių vidurkiai yra lygūs:

$$\begin{cases} H_0: & \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \dots \boldsymbol{\mu}_k \\ H_1: & \boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j \text{ nors vienai porai } (i,j), \end{cases} \tag{3.2}$$

$\boldsymbol{\mu}$ - daugiamačių grupių vidurkiai, $k$ - lyginamų grupių skaičius.

Šioje analizėje $F$ kriterijaus, skirto 3.2 hipotezei tikrinti, statistika gaunama skaidant visą kvadratų sumą $SST$ (angl. *Total Sum of Squares*) į dvi dalis: $SSW$ - vidinę tarpusavio kvadratų sumą (angl. *Within Sum of Squares*) ir $SSB$ - tarpgrupinę kvadratų sumą (angl. *Between Sum of Squares*):

$$SST = \sum_{l=1}^{g}\sum_{j=1}^{n_l}\left(Y_{lj}-\bar{Y}\right)^2 = \underbrace{\sum_{l=1}^{g}\sum_{j=1}^{n_l}\left(Y_{lj}-\bar{Y}_l\right)\left(Y_{lj}-\bar{Y}_l\right)'}_{SSW} + \underbrace{\sum_{l=1}^{g} n_l\left(Y_{lj}-\bar{Y}_l\right)\left(Y_{lj}-\bar{Y}_l\right)'}_{SSB},$$

čia $\bar{Y}_l$ - imties $l$ empirinis vidurkis, $l = 1, \dots, g$, $g$ - grupių skaičius, $n_l$ - imties $l$ dydis.

Apibendrinta 3 lentelė, vedanti prie testo statistikos:

| Kvadratų sumų ir tarpusavio sandaugų matricos | Laisvės laipsniai |
|---|---|
| $B = \sum_{l=1}^{g} n_l\left(Y_{lj}-\bar{Y}_l\right)\left(Y_{lj}-\bar{Y}_l\right)'$ | $g-1$ |
| $W = \sum_{l=1}^{g}\sum_{j=1}^{n_l}\left(Y_{lj}-\bar{Y}_l\right)\left(Y_{lj}-\bar{Y}_l\right)'$ | $\sum_{l=1}^{g} n_l - g$ |
| $T = B + W = \sum_{l=1}^{g}\sum_{j=1}^{n_l}\left(Y_{lj}-\bar{Y}\right)^2$ | $\sum_{l=1}^{g} n_l - 1$ |

3 lentelė: MANOVA lentelė

Daugiamačiams vidurkiams lyginti darbe naudojama Vilkso (angl. *Wilks*) statistika:

$$\Lambda = \frac{|W|}{|T|} = \frac{|W|}{|B+W|}.$$

$H_0$ yra atmetama, jeigu $F = \left(\frac{1-\Lambda^{1/b}}{\Lambda^{1/b}}\right)\left(\frac{ab-c}{p(g-1)}\right) > F_{p(g-1),ab-c}$, čia

$$a = \sum n_l - g - \frac{p-g+2}{2},$$

$p$ - kintamųjų skaičius;

$$b = \begin{cases} \sqrt{\frac{p^2(g-1)^2-4}{p^2+(g-1)^2-5}}, & \text{jei } p^2+(g-1)^2-5 > 0 \\ 1, & \text{jei } p^2+(g-1)^2-5 \leq 0 \end{cases}$$

$$c = \frac{p(g-1)-2}{2}.$$

Tam, kad būtų galima taikyti šį statistinį testą, reikalingos šių prielaidų išpildymas:

- atsako kintamieji yra tolygūs;
- daugiamatis normalusis pasiskirstymas - priklausomi kintamieji, turi būti normaliai pasiskirstę kiekvienoje grupėje;
- tiesiškumas - daroma prielaida, jog yra tiesinė priklausomybė tarp visų priklausomų kintamųjų porų;
- dispersijų homogeniškumas - grupėse dispersijos lygios;
- dispersijų homogeniškumas kovariacijų matricose.

Daroma prielaida, jog analizės duomenys yra tolygūs, normaliai pasiskirstę bei yra tiesinė priklausomybė tarp kintamųjų porų.

### 3.2.2 Dispersijos analizė ANOVA

ANOVA modelio prielaidos:

1. grupės turi tenkinti normalumo sąlygą;
2. grupių dispersijos yra lygios;
3. grupės nepriklausomos tarpusavyje.

Grupių dispersijų lygumo (dispersijų homogeniškumo) prielaida patikrinama Livyno testu (angl. *Levene*) [6], kur tikrinamos hipotezės:

$$\begin{cases} H_0: & \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2 \\ H_1: & \sigma_i^2 \neq \sigma_j^2 \text{ nors vienai porai } (i,j), \end{cases}$$

čia $\sigma^2$ - grupių dispersijos, $k$ - lyginamų grupių skaičius. Kriterijaus statistika yra apskaičiuojama:

$$F_{Levene} = \frac{N-k}{k-1} \frac{\sum_{i=1}^{k}(\bar{d_{i\cdot}} - \bar{d_{\cdot\cdot}})^2}{\sum_{i=1}^{k}\sum_{j=1}^{n_i} n_i (d_{ij} - \bar{d_{i\cdot}})^2},$$

čia $d_{ij} = |x_{ij} - \bar{x_i}|$, $i = 1, \ldots, k$, $j = 1, \ldots, n$, $\bar{x_i}$ - empirinis $i$-tosios grupės vidurkis.

$H_0$ yra atmetama, jeigu $F_{Levene} > F_\alpha(k-1, N-k)$, čia $F_\alpha(k-1, N-k)$ Fišerio skirstinio su $(k-1)$ ir $(N-k)$ laisvės laipsnių $\alpha$ lygmens kritinė reikšmė.

Kadangi dispersijų lygybė duomenims yra pažeista, taikomas ANOVA Velšo (angl. *Welch*) [19] kriterijus. Tikrinama hipotezė:

$$\begin{cases} H_0: & \mu_1 = \mu_2 = \cdots = \mu_k \\ H_1: & \mu_i \neq \mu_j \text{ nors vienai porai } (i,j), \end{cases}$$

čia $\mu$ - grupių vidurkiai, $k$ - lyginamų grupių skaičius. Kriterijaus statistika yra apskaičiuojama:

$$F_{Welch} = \frac{\frac{\sum_{i=1}^{k} W_i(\bar{X}_{i\cdot} - X'_{\cdot\cdot})}{k-1}}{1 + \frac{2}{3}(k-2)\Lambda},$$

čia

$$W_i = \frac{n_i}{s_i^2},$$

$n_i$ - imties dydis, $s_i$ - stebėjimų dispersija.

$$X'_{\cdot\cdot} = \frac{\sum_{i=1}^{k} W_i \bar{X}_i}{\sum_{i=1}^{k} W_i},$$

$$\Lambda = \frac{3 \sum \left(1 - \frac{W_i}{\sum_{i=1}^{k} W_i}\right)^2}{i^2 - 1}.$$

$H_0$ yra atmetama, jeigu $F_{Welch} > F_\alpha(k-1, \frac{1}{\Lambda})$, čia $F_{1-\alpha}(k-1, \frac{1}{\Lambda})$ Fišerio skirstinio su $(k-1)$ ir $(\frac{1}{\Lambda})$ laisvės laipsnių $\alpha$ lygmens kritinė reikšmė.

Velšo kriterijus nenustato, kurių grupių vidurkiai skiriasi (nustatoma ar bent dviejuose grupėse vidurkiai skiriasi iš bendro grupių $k$ skaičiaus), todėl jų identifikavimui naudojami aposterioriniai - Post-hoc kriterijai. ANOVA Velšo atveju taikomas Geimso-Hauvelio (angl. *Games-Howell*) - išplėstinis Tukio-Kramerio (angl. *Tukey-Kramer*) testas [25], kuris parodo, kurių grupių vidurkių skirtumai yra statistiškai reikšmingi. Tikrinamos hipotezės:

$$\begin{cases} H_0: & \mu_i - \mu_j = 0 \\ H_1: & \mu_i - \mu_j \neq 0. \end{cases}$$

Testo statistika yra apskaičiuojama:

$$t_{GH} = \frac{\bar{x}_i - \bar{x}_j}{\sqrt{\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}}},$$

$H_0$ yra atmetama, jeigu $t_{GH} > \frac{\sqrt{2}q_{\alpha,k,\nu}}{2}$, čia $q_{\alpha,k,\nu}$ kritinė reikšmė, kur

$$\alpha = \sqrt{\frac{\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}}{2}},$$

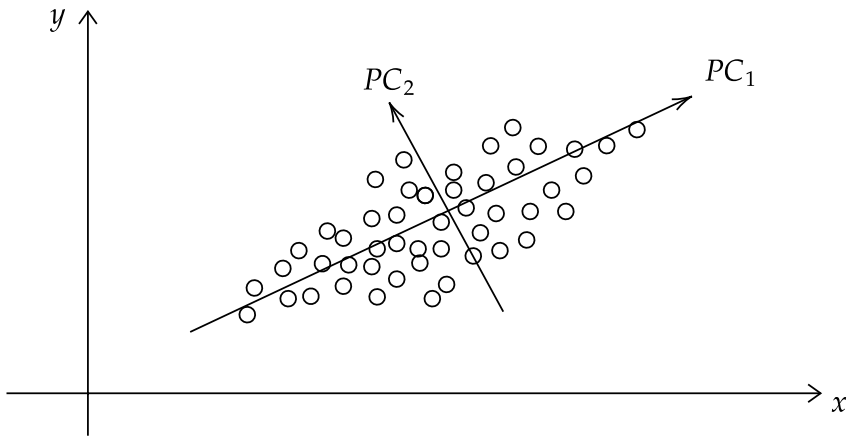
$$\nu = \frac{\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)^2}{\frac{\left(\frac{s_i}{n_i}\right)^2}{n_i - 1} + \frac{\left(\frac{s_j}{n_j}\right)^2}{n_J - 1}},$$

čia

- $k$ - grupių skaičius;
- $\bar{x}_i$, $\bar{x}_j$ imčių $i$ ir $j$ vidurkiai atitinkamai, $i, j \in 1, \ldots, k$;
- $s_i^2$, $s_j^2$ imčių $i$ ir $j$ dispersijos atitinkamai;
- $n_i$, $n_j$ atitinkamai $i$ ir $j$ imčių dydžiai.

### 3.3 Principinių komponenčių analizė

Principinių komponenčių analizė (angl. *Principal Components Analysis*, PCA) yra vienas iš būdų, siekiantis sumažinti duomenų dimensionalumą, neprarandant didelio kiekio informacijos. Atliekant PCA, yra ieškomos projekcijos, kurios maksimizuoja dispersiją. Jeigu požymių rinkinį sudaro $p$ požymių, tai iš viso yra gaunama $p$ principinių komponenčių. Pirmoji principinė komponentė yra kryptis erdvėje, kurios projekcija turi didžiausią dispersiją ($PC_1$). Antroji principinė ($PC_2$) komponentė yra kryptis, maksimizuojanti dispersiją tarp visų ortogonalių pirmajai komponentei krypčių. $p$ komponentė yra dispersiją maksimizuojanti kryptis, ortogonali ankstesnei $p-1$ principinei komponentei (žr. 3 pav.)




3 pav.: Pagrindinių komponenčių kryptys

Principinių komponenčių algoritmas:

1. Prieš atliekant principinių komponenčių analizę, kiekvienas kintamasis $x_{ij}$ yra standartizuojamas:

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x_j}}{s_j}, \tag{3.3}$$

čia $\bar{x_j}$ kintamojo $x_{ij}$ empirinis vidurkis, o $s_j$ - standartinis nuokrypis.

2. Tegul $\mathbf{X} = \{X_1, X_2, \ldots, X_m\} = \{\tilde{x}_{ij}, i = 1, \ldots, m, j = 1, \ldots, n\}$ duomenų matrica, kurios $i$-toji eilutė yra vektorius $X_i = (\tilde{x}_{i1}, \ldots, \tilde{x}_{in})$ žymintis objektus, o stulpeliai - objektus apibūdinančius parametrus $\tilde{x}_1, \ldots \tilde{x}_n$.

3. Apskaičiuojami kovariacijos koeficientai $c_{kl}$ bei suformuojama simetrinė kovariacinė matrica $C = \{c_{kl}, k = 1, \ldots, n, l = 1, \ldots, n\}$

4. Apibrėžiami kovariacinės matricos $C$ tikriniai vektoriai $E_k$ (angl. *Eigenvector*) ir jų tikrinės reikšmės $\lambda_k$ (angl. *Eigenvalues*) išsprendžiant

$$CE_k = \lambda_k E_k,$$

čia $E_k$ - vektorių stulpelis, $C$ - kovariacinė matrica, $\lambda_k$ - charakteringos lygties $|C - \lambda_k \mathbb{1}| = 0$ sprendinys, kur $\mathbb{1}$- vienetinė matrica, $|\cdot|$ - determinantas.

5. Tikriniai vektoriai $E_k$ surūšiuojami juos atitinkančių tikrinių reikšmių mažėjimo tvarka ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$).

6. Duomenų vektorius $X_i$, $i = 1, \ldots, m$ transformuojamas pagal formulę

$$Y_i = (X_i - \mu)A,$$

čia

- Matrica $A = (E_1, \ldots, E_n)$ - pagrindinių komponenčių matrica;
- $X_i = (\tilde{x}_{i1}, \ldots, \tilde{x}_{in})$;
- $\mu = (\mu_1, \ldots, \mu_n)$;

7. Gauti $Y_i = (y_{i1}, \ldots y_{in})$ vektoriai naujoje ortogonalioje koordinačių sistemoje $(y_1, \ldots y_n)$.

## 3.4 Klasterizavimas

Klasterizavimas yra mašininio mokymosi be priežiūros metodas, kuomet identifikuojamos duomenų grupės arba klasteriai, turintys panašumo matą (pvz. Euklidinį atstumą). Klasterizavimas padeda nustatyti panašumus ir ryšius tarp analizuojamų duomenų.

### 3.4.1 *k*-vidurkių metodas

Duomenų klasterizavimui naudojamas J.A.Hartigan ir M.A.Wong $k$-vidurkių algoritmas [7], kuris minimizuoja kvadratinę paklaidą: tegul $K_i$ yra klasteris, o $\{X_I^1, X_I^2, \ldots, X_i^\mu\}$, $i \in (1, \ldots, k)$ - šiam klasteriui priskirti objektai, $\mu_i$ - objektų klasteryje $K_i$ skaičius, $X_i^j = (x_{i1}^j, x_{i2}^j, \ldots, x_{in}^j)$, $j \in (1, \ldots, \mu_i)$. Kvadratinė paklaida klasteriui $K_i$ apskaičiuojama:

$$SSE_{K_i} = \sum_{j=1}^{\mu_i} \left\| X_i^j - C_i \right\|^2, \tag{3.4}$$

čia $C_i, i \in 1, \ldots, k$ klasterio centras. Tuomet kvadratinė paklaida klasterių aibei yra apskaičiuojama:

$$SSE_k = \sum_{i=1}^{k} SSE_{K_i} \tag{3.5}$$

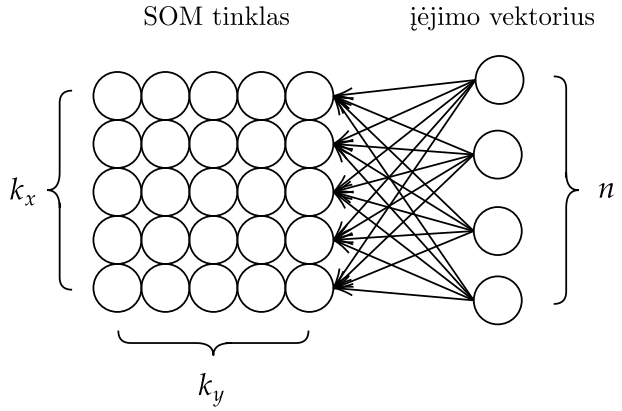
$k$-vidurkių algoritmas:

1. nurodomas klasterių skaičius $k$;
2. atsitiktinai inicijuojami klasterių centrai: $C_i, i \in 1, \ldots, k$;
3. kiekvienas duomenų taškas, esantis arčiausiai klasterio centro, priskiriamas klasteriui;
4. sudarytam klasteriui perskaičiuojami centrai;
5. skaičiuojama kvadratinė paklaida (pagal 3.5 formulę) tarp klasterio centro ir tam klasteriui priskirtų duomenų taškų;
6. 3-5 žingsniai kartojami tol, kol centrai nusistovi, t.y duomenys nebepasiskirsto kitiems klasteriams.

Prieš naudojant šį algoritmą, reikia žinoti klasterių skaičių $k$. Vienas iš populiariausių ir seniausių metodų optimaliam klasterių skaičiui parinkti yra alkūnės metodas (angl. *Elbow method*) [13]. Šiame metode yra vaizduojama dispersija - bendra klasterių kvadratų suma (angl. *total within-cluster Sum of Squares*, WCSS) ir klasterių skaičius. Pirmieji klasteriai paaiškina didžiąją dalį dispersijos ir informacijos, tačiau kažkuriame taške, ta informacija sumažėja ir tokiu būdu grafike susidaro kampas. Tas kampas yra vadinamas alkūne ir šis taškas yra laikomas optimaliu klasterių skaičiumi.

### 3.4.2 Saviorganizuojantys neuroniniai tinklai

Saviorganizuojančių neuroninių tinklų (angl. *Self-organizing maps*, SOM) dar kitaip vadinamų Kohoneno neuroninių tinklų idėja yra transformuoti sudėtingus aukštos dimensijos duomenis į paprastesnę mažesnės dimensijos (dažniausiai dviejų dimensijų) erdvę, išlaikant ryšius tarp duomenų taškų [14]. SOM yra neuronų $M = \{m_{ij}, i = 1, \ldots, k_x, j = 1, \ldots, k_y\}$, išdėstytų dvimačio tinklelio mazguose, masyvas [26]. Kiekvienas tinklo neuronas yra sujungiamas su kiekvieno įėjimo (angl. *input*) vektoriumi. 4 paveikslėlyje pavaizduota dvimačio neuroninio tinklo pavyzdinė schema.




4 pav.: Dvimačio SOM tinklo pavyzdinė schema

SOM tinklo mokymosi algoritmas [4]:

1. $k_x$ yra tinklo lentelės eilučių skaičius, o $k_y$ - stulpelių skaičius. Kiekvienas $n$-matis mokymosi aibės vektorius $X \in \{X_1, X_2, \ldots, X_m\}$ mokymosi metu yra susiejamas su tinklo neuronu, kuris yra $n$-matis vektorius, kaip ir mokymosi aibės vektorius. Mokymosi pradžioje vektorių $m_{ij}$ komponentės generuojamos atsitiktinai ir kiekviename mokymosi žingsnyje vienas vektorius iš mokymosi aibės $X \in \{X_1, X_2, \ldots, X_m\}$ pateikiamas į tinklą.

2. Kiekvienas pateiktas vektorius palyginamas su visais neuronais $m_{ij}$: randamas mažiausias pateikto vektoriaus $X_k$ Euklidinis atstumas ($\arg\min_{i,j}\{\|X_k - m_{ij}\|\}$) iki neurono-nugalėtojo $m_c$.

3. Visos neuronų komponentės yra keičiamos pagal formulę:

$$m_{ij}(t+1) = m_{ij}(t) + h_{ij}(t)^c \left(X_k - m_{ij}(t)\right),$$

čia $t$-iteracijų skaičius $h_{ij}^c = \exp\left(-\frac{\|R_c - R_i\|^2}{2\sigma^2(t)}\right)$, $R_c$ ir $R_{ij}$ yra dvimačiai vektoriai, sudaryti iš $m_c$ ir $m_{ij}$ indeksų, nusakančių vektoriaus $X_k$ neurono nugalėtojo $m_c$ ir perskaičiuojamo neurono $m_{ij}$ vietą SOM tinkle. $\eta_{ij}^c$ - kaimynystės tarp $m_c$ ir $m_{ij}$ eilė.

4. Kiekvienos epochos metu perskaičiuojami tie neuronai $m_{ij}$, kuriems galioja nelygybė:

$$\eta_{ij}^c \leq \max\left(\alpha \max(k_x, k_y), 1\right).$$

Epocha - mokymosi proceso dalis, kurios visi mokymosi aibės vektoriai $X_1, X_2, \ldots, X_m$ po vieną kartą pateikiami į tinklą nuosekliai arba atsitiktine tvarka.

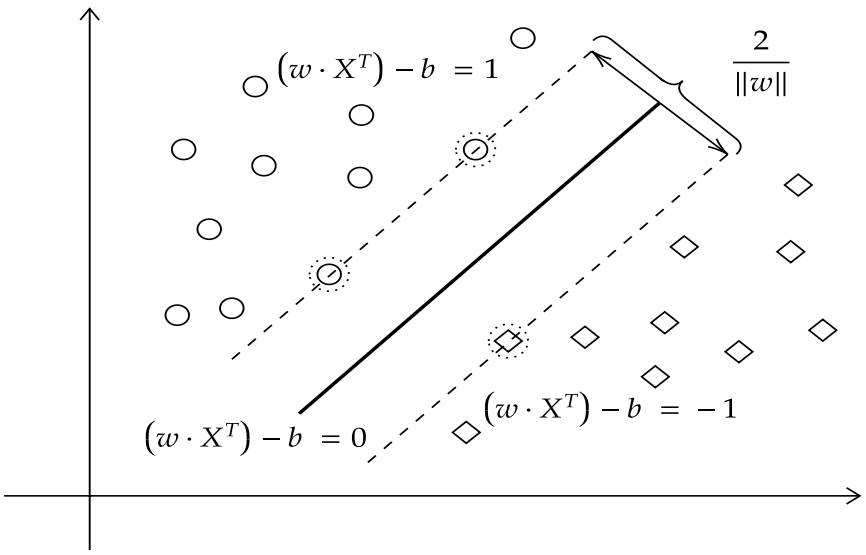
Po šio apmokymo, į tinklą pateikiami nauji (tinklui dar nematyti) duomenų vektoriai. Randamas kiekvieno vektoriaus neuronas nugalėtojas ir jis pažymimas SOM žemėlapyje neurono nugalėtojo vietoje taip sumažinant aukštos dimensijos įėjimo erdvė į mažesnės dimensijos išėjimo (angl. *output*) erdvę.

## 3.5 Klasifikavimas

Klasifikavimas - duomenų analizės metodas, kurio tikslas yra objekto, naudojant skaitmeninius duomenis, priskyrimas kokiai nors klasei. Tyrime naudojami trys klasifikavimo algoritmai: atraminių vektorių klasifikatorius, atsitiktinio miško ir daugialypės logistinės regresijos klasifikatoriai. Visais algoritmais yra sprendžiamas uždavinys: kokiai populiarumo grupei priklauso vaizdo įrašas su turimais metaduomenimis.

### 3.5.1 Atraminių vektorių klasifikatorius

Atraminių vektorių klasifikatorius [1] yra klasifikavimo su priežiūra metodas, kuris gali būti taikomas tiek klasifikavimo, tiek regresinės analizės uždaviniams spręsti. Klasifikavimo metu, naudojant SVM algoritmą, yra sukuriama hiperplokštuma, kuri geriausiai atskiria duomenis į klases. SVM klasifikatoriaus algoritmo veikimo iliustracija pavaizduota 5 paveikslėlyje.




5 pav.: Tiesinio branduolio SVM klasifikatoriaus pavyzdinė schema

Tegul $X_i$ yra mokymosi duomenų aibė, kur kiekvienas aibės objektas turi nustatytą klasę $Z_i$. Sudaromos tokios poros $(X_i, Z_i)$, $i = 1, \ldots, m$ ir $Z_i = \{1, -1\}$. SVM sukuria hiperplokštumą:

$$(w \cdot X^T) - b = 0, \quad w \in \mathbb{R}^n,\ b \in \mathbb{R},$$

atitinkančią tikslo funkciją

$$f(X) = \operatorname{sgn}\left((w \cdot X^T) - b\right),$$

čia $w = (w_1, w_2, \ldots, w_n)$, $X = (x_1, x_2, \ldots, x_n)$ yra vektoriai eilutės, o $w \cdot X^T = \sum_{i=1}^{n} w_i x_i$.

Sukurta hiperplokštuma priklauso nuo mokymosi aibės poaibio, sudaryto iš atraminių vektorių. Konstruojant geriausią hiperplokštumą yra sprendžiamas optimizavimo uždavinys, turintis tam tikrus ribojimus:

$$\min_{w,b} \frac{1}{2} \|w\|^2,$$

apribojimai: $Z_i \left((w \cdot X_i^T) - b\right) \geq 1, \quad i = 1, \ldots, m.$

Sudaroma Lagranžo forma:

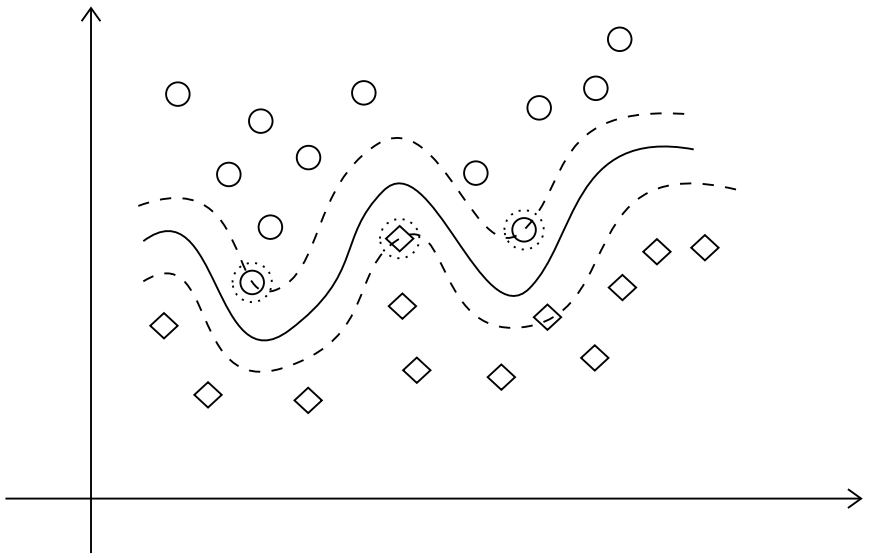
$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(Z_i \left(\left(w \cdot X_i^T\right) - b\right) - 1\right).$$

Atraminiais vektoriais yra vadinami tokie vektoriai, kuriems $\alpha_i \neq 0$. Maksimalus atstumas tarp vienos ir kitos klasių paviršių yra lygus $\frac{2}{\|w\|}$.

Bendru atveju tikslo funkcija yra:

$$f(X) = \operatorname{sgn}\left(\sum_{i=1}^{m} Z_i \alpha_i (k(X, X_i)) - b\right),$$

kur $k(X, X_I)$ yra branduolio (angl. *kernel*) funkcija. Kuomet duomenys negali būti atskiriami tiesiškai (žr. 6 pav.), naudojami branduoliai. Šiame darbe naudojamas radialinių bazių funkcijų (RBF) branduolys, kur $k(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|}{\gamma}\right)$.



6 pav.: Netiesinio branduolio SVM klasifikatoriaus pavyzdinė schema

Kadangi darbe yra sprendžiama kelių klasių $Z \in (1, 2, \ldots N)$ klasifikavimo problema, naudojamas *one-vs-one* metodas, kurį aprašė R. Debnath, N. Takahide ir H. Takahashi [3].

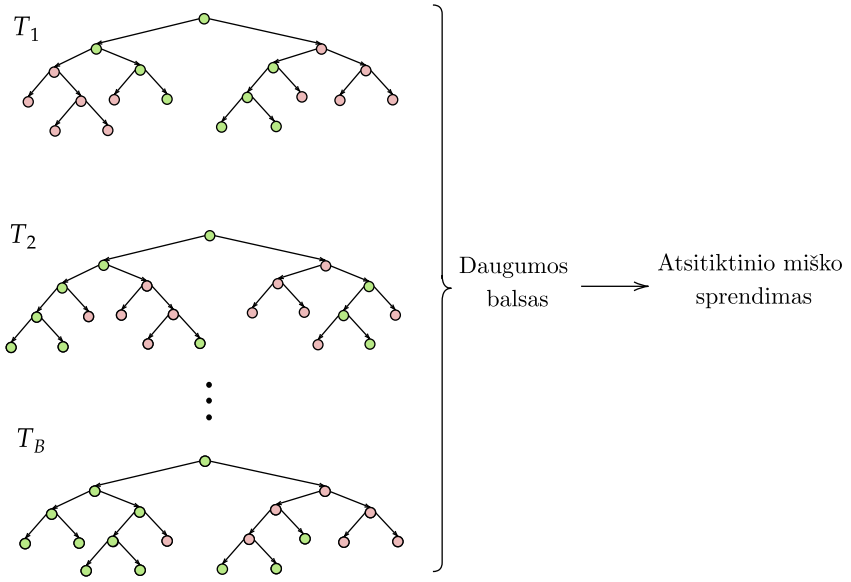
### 3.5.2 Atsitiktinio miško klasifikatorius

Atsitiktinio miško klasifikatorius yra kolektyvinis klasifikatorius, kuriame naudojami sprendimų medžių algoritmai (angl. *decision trees*) kartu su *Bagging* (angl. *Bootstrap Aggregating*), kur rezultatai gaunami daugumos balsavimu. Šis klasifikatorius yra ansamblinis, turintis mažą poslinkį (angl. *bias*), yra greitai apmokomas bei atsparus triukšmui [20], [18]. Atsitiktinio miško klasifikatoriaus klasių sudarymo idėja yra klasių formavimas, apjungiant sprendimus daugumos balsu iš sprendimų medžių, sukurtų sujungiant skirtingus duomenų poaibius iš visos duomenų aibės, naudojant atsitiktinai parinkus požymių poaibius iš požymių aibės. Todėl, atsitiktinio miško klasifikatorius grąžina tą klasę, kuri buvo sprendimų medžių parinkta daugiausia kartų. RF veikimo iliustracija pavaizduota 7.

Atsitiktinio medžio algoritmas klasifikavimui:

1. Parenkama $N$ dydžio atsitiktinė su grąžinimu imtis $Z^*$ iš mokymosi aibės (angl. *bootstrap*).
2. Auginamas atsitiktinio miško medis $T_b$, rekursiškai kartojant vėlesnius veiksmus visiems to medžio mazgams (angl. *node*) tol, kol minimalus mazgo dydis $n_{min}$ yra pasiektas:
   (a) Pasirenkamas $m$ kiekis požymių iš $p$ galimų.
   (b) Pasirenkamas geriausias kintamasis ir atskyrimo taškas tarp $m$ kintamųjų, naudojant Gini indeksą (angl. *Gini Impurity*) (žr. 3.6 formulę).
   (c) Mazgas atskiriamas į dar du dukterinius mazgus.
3. Gaunamas medžių $\{T_b\}_1^B$ ansamblis.
4. Klasė $\hat{C}_b(x)$ yra $b$-tojo atsitiktinio miško medžio prognozė. Tuomet

$$\hat{C}_{RF}^B(c) = \text{daugumos balsas} \left\{ \hat{C}_b(x) \right\}_1^B .$$




7 pav.: Atsitiktinio miško pavyzdinė schema

Vienas iš atsitiktinio miško klasifikatoriaus privalumų yra galimybė įvertini požymių svarbą (angl. *Feature Importance*, FI). Požymių svarba nusako, kurie požymiai algoritmui yra svarbūs

- kiek kartų klasifikatorius panaudoja tam tikrą požymį tikslesniam klasės priskyrimui. Viena iš sudėtingesnių priemonių, naudojamų ir darbe norint išmatuoti požymių svarbą, yra Gini indekso:

$$I_G(p) = \sum_{i=1}^{K} p_i (1 - p_i), \tag{3.6}$$

čia $i \in \{1, 2, ..., J\}$, $J$ – imtyje esančių kategorijų skaičius, o $p_i$ yra $i$-tosios kategorijos proporcija imtyje, pokytis, kuomet medyje atliekamas atskyrimas remiantis tam tikru požymiu.

Kiekviename atsitiktinio miško klasifikatoriaus medžio mazge siekiama geriausio skilimo naudojant Gini indeksą - matuojama, kaip gerai potencialus skilimas atskiria klases tame taške. Tai reiškia, kad mazge skilimas vykdomas pagal tam tikrą kintamąjį ir tokiu atskyrimo tašku, kad sukurtos naujos imtys kitos klasės atžvilgiu būtų grynesnės.

### 3.5.3 Daugialypės logistinės regresijos klasifikatorius

Daugialypė logistinė regresija (angl. *Multinomial Logistic Regression*, MLR) logistinės regresijos apibendrinimas, kuomet priklausomas kintamasis įgyja daugiau nei dvi skirtingas reikšmes, skirtas tiek regresijos, tiek klasifikavimo uždaviniams spręsti [5].

Kategorijų priskyrimas šiame klasifikatoriuje yra vykdomas skaičiuojant visų galimų $Y$ kategorijų įgijimo tikimybes bei prognozuojant $Y$ tokią reikšmę, kurios įgijimo tikimybė yra didžiausia. Matematinis modelis gali būti nusakomas [9]:

$$\log\left(\frac{p}{1-p}\right) = \alpha_j + \beta_1 x_1 + \ldots \beta_j x_j, \tag{3.7}$$

Lygtis 3.7 gali būti perrašoma:

$$p = P(Y_i = j \mid x_i) = \frac{\exp\{z_j\}}{1 + \sum_h^{J-1} \exp\{z_h\}},$$

kur $j \in \{1, 2 \ldots, J\}$, $J$ - imtyje esančių kategorijų skaičius, $z_j = \alpha_j + \beta_j x_i$, $x_i$ - požymiai.

Kintamųjų atrankai naudojama Voldo (angl. *Wald*) [17] kriterijaus statistika (arba Voldo Z statistika). Šis kriterijus padeda nuspręsti, ar kintamąjį reikia šalinti iš modelio. Hipotezės:

$$\begin{cases} H_0: & \beta_m = 0 \\ H_1: & \mu_m \neq 0, \end{cases}$$

kuri tikrinama statistika

$$Z = \frac{\hat{\beta}_m}{se(\hat{\beta}_m)},$$

čia *se* - standartinė paklaida.

Kaip ir atsitiktinio miško, taip ir daugialypės logistinės regresijos klasifikatoriuje, galima apskaičiuoti požymių svarbą. Ji apskaičiuojama sudedant gautas kiekvieno kintamojo $z$ reikšmes kiekvienoje kategorijoje. Kuo $z$ reikšmė yra didesnė, tuo požymio svarba klasifikatoriui yra svarbesnė.

### 3.5.4 *k*-dalių kryžminis patikrinimas

Klasifikatorių apmokymui yra naudojama $k$-dalių kryžminis patikrinimas (angl. *k-fold Cross Validation*). Duomenų imtis yra padalinama į lygiai $k$ dalių, kur šios dar yra skaidomos į dvi dalis - vertinimo ir patikrinimo: atliekamas klasifikatoriaus apmokymas naudojant vertinimo $k-1$ imties daliai ir testavimas naudojant likusią duomenų imties dalį. Tuomet, patikrinimo duomenys, kurie nenaudoti klasifikatoriaus mokymui, pateikiami klasifikatoriui ir apskaičiuojami klasifikavimo bendri tikslumo matai. Iš viso vykdoma $k$ iteracijų, užtikrinant, jog kiekviena iš $k$ dalių yra panaudota tikrinimui [27].

Darbe pasirinktas $k = 10$ dalių parametras. Dažniausiai pasirenkami $k = 5$ arba $k = 10$, tačiau griežtos taisyklės nėra. Kita vertus, kuomet imamas vis didesnis $k$, skirtumas tarp vertinimo ir patikrinimo imčių dydžio tampa vis mažesnis, o tai ir lemia poslinkio mažėjimą [15]. Kita vertus, didesnis $k$ reikalauja didesnių atlikimo kaštų, todėl tokio modelio apmokymas vyksta lėčiau.

### 3.5.5 Klasifikatorių įvertinimas

Tam, kad nustatyti, ar klasifikatorius tinkamai klasifikuoja objektus į reikiamas klases, tyrime naudojamas bendras klasifikavimo tikslumas (angl. *Accuracy*), kuris apskaičiuojamas kaip santykis tarp teisingai klasifikuotų duomenų skaičiaus ir visų duomenų skaičiaus:

$$\text{bendras klasifikavimo tikslumas} = \frac{\text{Teisingi spėjimai}}{\text{Visi spėjimai}}. \tag{3.8}$$

Kadangi bendras klasifikavimo tikslumas nepasako apie klasifikavimo tikslumą kiekvienai klasei atskirai, tyrime naudojamos papildomos dvi metrikos klasifikavimo algoritmų įvertinimui: jautrumas (angl. *recall/sensitivity*) ir tikslumas (angl. *precision*). Šias metrikas išreiškiamos prieš tai apibrėžus reikiamas sąvokas:

- tikrai teigiamas (TT) (angl. *true positive*) - klasės $J$ objektas priskiriamas klasei $J$.
- tikrai neigiamas (TN) (angl. *true negative*) - ne klasės $J$ objektas nepriskiriamas klasei $J$.
- klaidingai teigiamas (KT) (angl. *false positive*) - ne klasės $J$ objektas priskiriamas klasei $J$.
- klaidingai neigiamas (KN) (angl. *false negative*) - klasės $J$ objektas nepriskiriamas klasei $J$.

Todėl klasifikavimo kokybė išreiškiama formulėmis:

$$\text{jautrumas (klasė} = J) = \frac{\text{TT}}{\text{TT} + \text{KN}};$$

$$\text{tikslumas (klasė} = J) = \frac{\text{TT}}{\text{TT} + \text{KT}}.$$

Remiantis aukščiau apibrėžtomis sąvokomis, 3.8 formulė gali būti išreikšta:

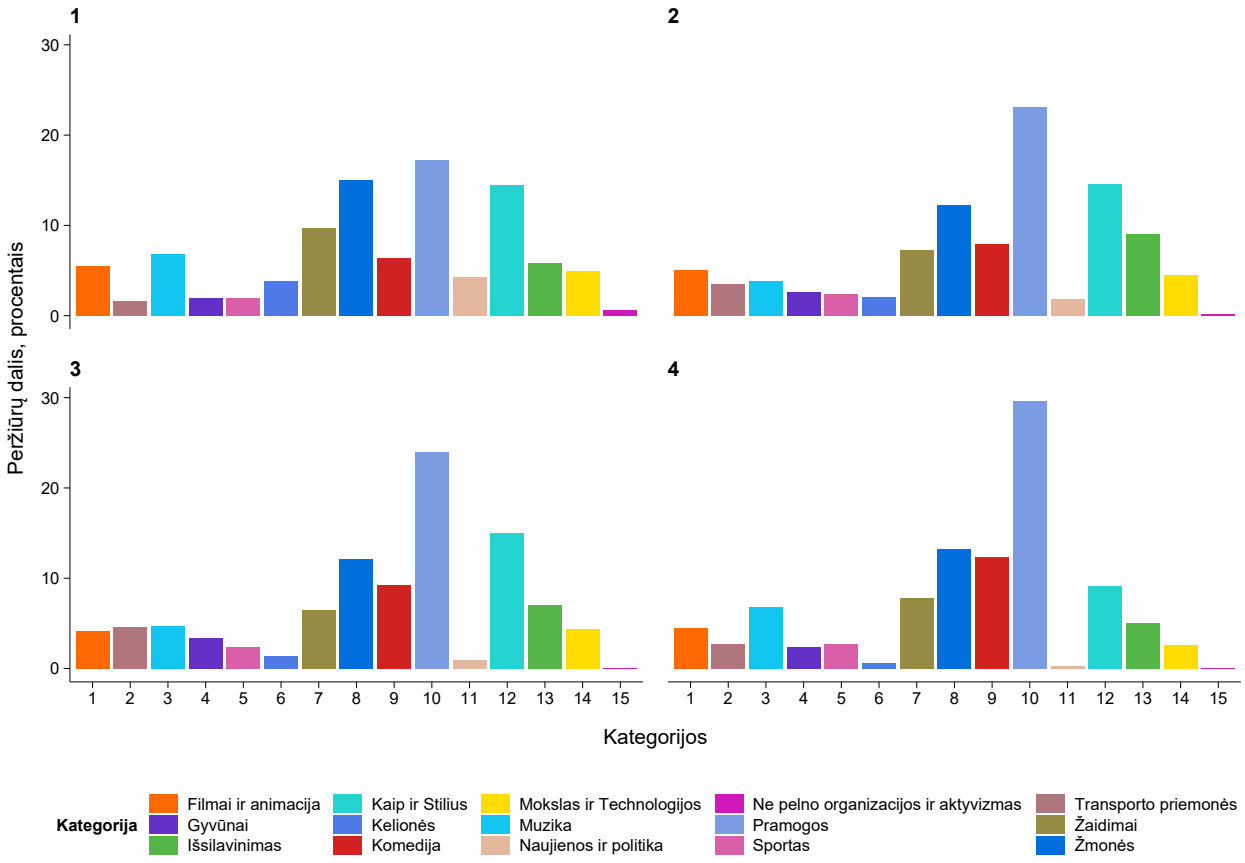
$$\text{bendras klasifikavimo tikslumas} = \frac{\text{TT} + \text{TN}}{\text{TT} + \text{TN} + \text{KN} + \text{KT}}.$$

# 4 Praktinė dalis

## 4.1 Pirminė duomenų analizė

Kiekvienos populiarumo grupės kiekybinių kintamųjų aprašomoji statistika pateikta priedų skyriuje atitinkamai 18, 20, 22 ir 24 lentelėse. Pagrindiniai skirtumai yra matomi tarp vidutinių peržiūrų, "patinka", "nepatinka" paspaudimų ir komentarų skaičiaus kintamųjų: kuo vaizdo įrašas populiaresnis, tuo yra didesnis auditorijos įsitraukimas, t.y minėtų kintamųjų vidurkis populiarumo grupėse yra didesnis. Panaši tendencija yra ir su simbolių pavadinime bei raktinių žodžių skaičiais - populiaresni vaizdo įrašai vidutiniškai turi daugiau simbolių pavadinime bei daugiau raktinių žodžių. Taip pat galima pastebėti, jog populiaresnių vaizdo įrašų turinčių kanalų prenumeratorių skaičius yra vidutiniškai didesnis - kas yra natūralu - didesni kanalai turi didesnę žiūrovų pradinę auditoriją, todėl tokiame kanale išleistas vaizdo įrašas, tikėtina, kad gali būti populiaresnis, negu tokiame kanale, kur prenumeratorių skaičius yra nedidelis. Vidutinis amžius, priešingai, populiarumo grupėse yra mažesnis tų vaizdo įrašų, kurie priskirti labai populiarių vaizdo įrašų grupei. Kadangi didžiąją peržiūrų dalį vaizdo įrašas dažniausiai surenka per pirmuosius mėnesius, toks įrašas gali būti populiaresnis, nes seni vaizdo įrašai po tam tikro laiko peržiūrų nerenka iš viso, arba renka labai mažai.

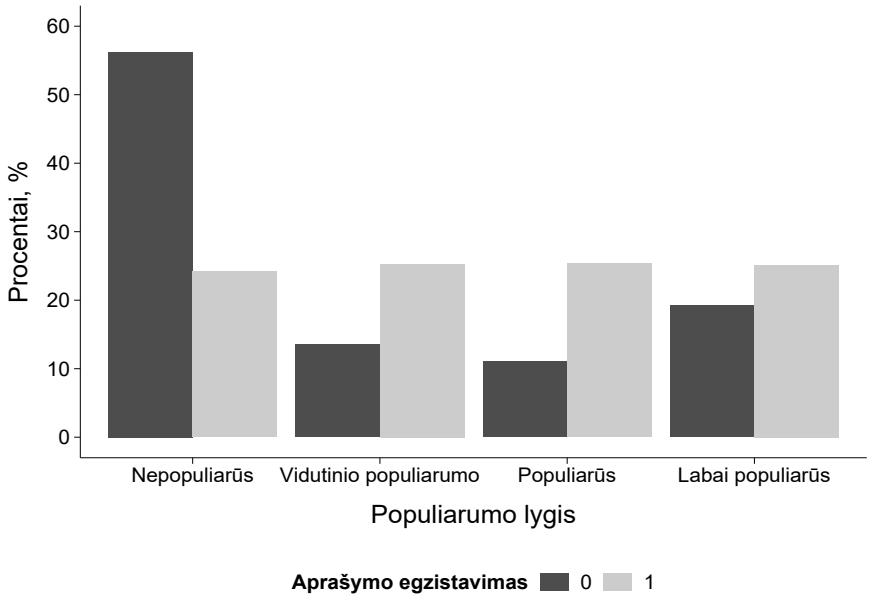
Kadangi rinkta informacija ir apie kiekvieną vaizdo įrašo kategoriją, pasižiūrėta kaip kiekviena kategorija yra pasiskirsčiusi tarp populiarumo grupių (žr. 8 pav.). Šiame paveikslėlyje 1 žymi nepopuliarių vaizdo įrašų grupę, 2 - vidutinio populiarumo, 3 - populiarių, o 4 - labai populiarių vaizdo įrašų grupę. Matoma, kad "Pramogų" kategorija visose populiarumo grupėse sudaro didžiausią procentinę dalį. Mažiausią dalį sudaro kategorija "Ne pelno organizacijos ir aktyvizmas". Kita vertus visos kategorijos populiarumo grupėse pasiskirsčiusios panašiai.




8 pav.: Kategorijų pasiskirstymas populiarumo grupėse

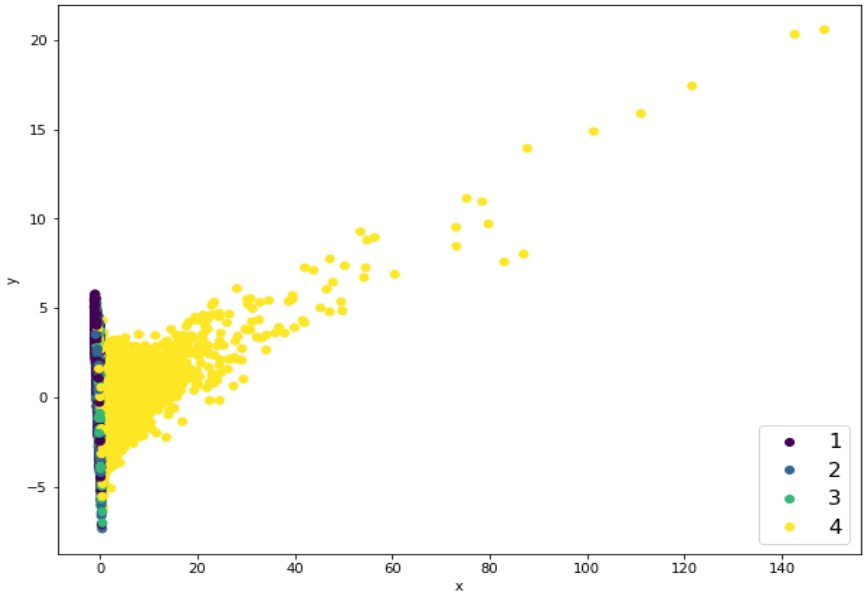
Vienas iš nagrinėtų kokybinių kintamųjų yra `description_exists` - aprašymo egzistavimas. Tikėtina, kad nepopuliariuose vaizdo įrašuose didesnę dalį sudarys vaizdo įrašai, kurie neturi jokio

aprašymo. Tą iliustruoja 9 grafikas. Iš jo matoma, kad didžiąją dalį nepopuliarių vaizdo įrašų dalį sudaro tokie vaizdo įrašai, kuriuose aprašymas neegzistuoja (reikšmė 0). Kita vertus, aprašymo egzistavimo (reikšmė 1) procentinė dalis visose populiarumo grupėse yra išsidėsčiusi tolygiai. Tai reiškia, kad jei vaizdo įrašas ir turi aprašymą, šio tikimybė patekti į tam tikrą populiarumo grupę yra tokia pati.




9 pav.: Aprašymo egzistavimas populiarumo grupėse

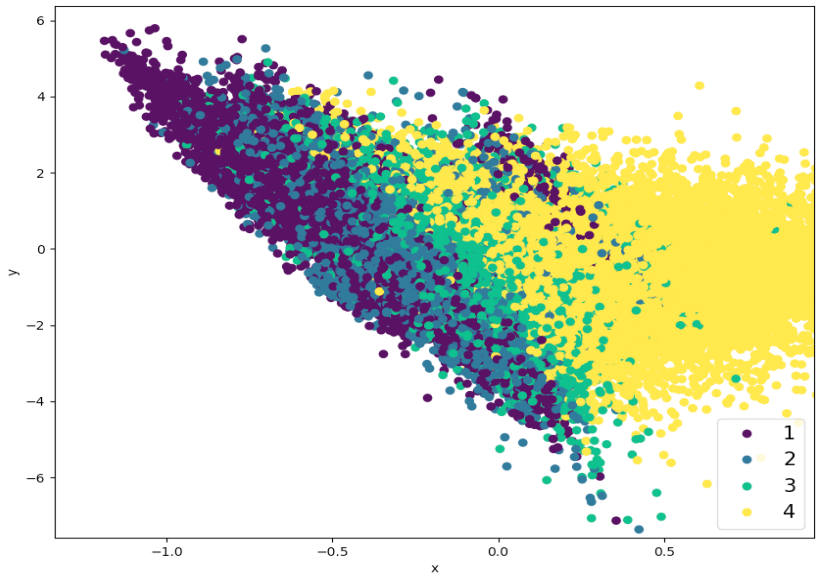
Tam, kad pasižiūrėti, kaip atrodo sudarytų populiarumo grupių išsidėstymas, klasės pavaizduotos plokštumoje, sumažinus duomenų dimensiją principinių komponenčių analize (žr. 10 pav., klasių žymėjimas toks pats kaip ir 8 pav.). Iš šio paveikslėlio matoma, kad geriausiai atsiskyrusi nuo visų grupių yra ketvirtoji populiarumo grupė.




10 pav.: Populiarumo klasių vizualizavimas naudojant dimensijos mažinimą PCA

Kita vertus, 10 grafike nesimato likusių klasių išsidėstymo. Todėl 11 grafike pavaizduotas išdidintas ankstesnio grafiko vaizdas. Iš šio paveikslėlio matoma, kad pirmoji, antroji ir trečioji grupė yra labiau persidengusios. Tačiau vis tiek galima pastebėti tam tikrą specifinį grupių išsidėstymo tendenciją: pirmoji - nepopuliarių vaizdo įrašų grupė yra susitelkusi aukščiau, antrosios grupės

vaizdo įrašai yra labiau išsibarstę negu pirmosios, tačiau didesnis taškų kiekis yra susitelkęs žemesnėje dalyje. Trečioji populiarumo grupė yra išsidėsčiusi lygiagrečiai tarp pirmosios ir antrosios bei ketvirtosios grupės.




11 pav.: Populiarumo klasių vizualizavimas naudojant dimensijos mažinimą PCA. Išdidintas vaizdas

### 4.1.1 Statistinių testų rezultatai

Kadangi grupės yra gana persidengusios, verta patikrinti ar egzistuoja skirtumai tarp populiarumo grupių. Skirtumai populiarumo grupėse tarp atsako kintamųjų tikrinami MANOVA statistiniu testu. Tikrinama hipotezė:

$$\begin{cases} H_0: & \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \boldsymbol{\mu}_3 = \boldsymbol{\mu}_4 \\ H_1: & \boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j \text{ nors vienai porai } (i, j). \end{cases}$$

Testo rezultatai matomi 4 lentelėje:

| | **Laisvės laipsniai** | $\Lambda$ | **F** | **p-reikšmė** |
|---|---|---|---|---|
| `popularity` | 36, 257268 | 0,29642 | 3638,6 | $2{,}2{\cdot}10^{-16}$ |

4 lentelė: MANOVA testo rezultatai

Remiantis pastarosios lentelės rezultatais, egzistuoja statistiškai reikšmingas skirtumas tarp populiarumo grupių (p-reikšmė $= 2{,}2{\cdot}10^{-16} < \alpha = 0.05$).

Taip pat atlikti ANOVA statistiniai testai, naudojant Velšo kriterijų, prieš tai patikrinus hipotezę apie dispersijų lygybę, naudojant Livyno testą, kur tikrinama hipotezė:

$$\begin{cases} H_0: & \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \sigma_4^2 \\ H_1: & \sigma_i \neq \sigma_j^2 \text{ nors vienai porai } (i, j). \end{cases}$$

Livyno testo rezultatai matomi 5 lentelėje. Galima teigti, jog dispersijos homogeniškumo prielaida yra pažeista (p-reikšmės $\approx 0 < 0,05$).

| **Kintamasis** | **Laisvės laipsniai**, n | **Laisvės laipsniai**, d | **Statistika** | **p -reikšmė** |
|---|---|---|---|---|
| avg_daily_views | 3 | 87084 | 17444,01 | 0 |
| avg_daily_comments | 3 | 87084 | 16599,89 | 0 |
| avg_daily_dislikes | 3 | 87084 | 11780,05 | 0 |
| avg_daily_likes | 3 | 87084 | 13203,74 | 0 |
| channel_subscriber_count | 3 | 87084 | 13578,69 | 0 |
| chr_title | 3 | 87084 | 65,18 | $4{,}32{\cdot}10^{-42}$ |
| keywords_count | 3 | 87084 | 8,93 | $6{,}52\cdot10^{-6}$ |
| thumbnail_quality | 3 | 87084 | 217,95 | $7{,}21\cdot10^{-141}$ |
| thumbnail_blurriness | 3 | 87084 | 51,66 | $2{,}37{\cdot}10^{-33}$ |
| thumbnail_brightness | 3 | 87084 | 397,5 | $1{,}7{\cdot}10^{-256}$ |
| thumbnail_contrast | 3 | 87084 | 233,9 | $3{,}66{\cdot}10^{-151}$ |
| age | 3 | 87084 | 3311,12 | 0 |

5 lentelė: Livyno testo rezultatai

Atlikus ANOVA testą, naudojant minėtą Velšo kriterijų, gauti 6 lentelėje pateikti rezultatai. Remiantis jais, galima teigti, jog statistiškai reikšmingi skirtumai yra visuose kiekybiniuose kintamuosiuose tarp vaizdo įrašo populiarumo grupių.

| **Kintamasis** | **Laisvės laipsniai**, n | **Laisvės laipsniai**, d | **Statistika** | **p -reikšmė** |
|---|---|---|---|---|
| avg_daily_views | 3 | 37311,84 | 55008,68 | 0 |
| avg_daily_comments | 3 | 37250,93 | 23599,92 | 0 |
| avg_daily_dislikes | 3 | 37274,65 | 27842,52 | 0 |
| avg_daily_likes | 3 | 37178,36 | 32317,01 | 0 |
| channel_subscriber_count | 3 | 39937,11 | 14893,54 | 0 |
| chr_title | 3 | 48118,04 | 252,49 | 0 |
| keywords_count | 3 | 48181,38 | 1444,16 | 0 |
| thumbnail_quality | 3 | 48002,75 | 373,67 | 0 |
| thumbnail_blurriness | 3 | 48325,00 | 1177,87 | 0 |
| thumbnail_brightness | 3 | 47876,57 | 942,68 | 0 |
| thumbnail_constrast | 3 | 47978,82 | 149,35 | 0 |
| age | 3 | 46461,12 | 4560,48 | 0 |

6 lentelė: ANOVA Velšo kriterijaus rezultatai

Tam, kad identifikuoti, tarp kurių grupių ir kintamųjų vidurkiai nesiskiria, atlikta Post Hoc analizė, kurioje naudojamas Geimso-Hauvelio testas. Testo rezultatai pateikiami priedų 26 lentelėje. Iš šio kriterijaus rezultatų matoma, kad nereikšmingi skirtumai tarp vidurkių yra šiuose požymiuose:

- Simbolių pavadinime skaičiuje tarp pirmosios ir antrosios bei pirmosios ir trečiosios populiarumo grupių (p-reikšmė $> 0{,}05$);
- Miniatiūros kokybėje tarp antros ir trečios populiarumo grupių;
- Miniatiūros išsiliejime tarp trečios ir ketvirtos populiarumo grupių;
- Miniatiūros kontraste tarp antros ir trečios populiarumo grupių.

Apibendrinant šiuos rezultatus, galima teigti, jog nors ir grupės yra gana stipriai persidengusios, tačiau skirtumai tarp populiarumo klasių vidurkių yra statistiškai reikšmingi. Todėl kvartilių metodu suskirstyti duomenys į populiarumo grupės yra tinkamas būdas ir klasių sujungimas yra nereikalingas.

## 4.2 Klasterizavimo rezultatai

Tam, kad patikrinti ar sudarytos populiarumo grupės naudojant statistinius metodus yra tinkamas būdas, atlikti du grupavimo algoritmai - $k$-vidurkių metodas ir SOM - siekiant įvertinti klasių susidarymo tendencijas ir ryšį su vaizdo įrašų populiarumu.

Prieš atliekant klasterizavimo algoritmus, kiekybiniai požymiai standartizuojami pagal 3.3 formulę ir sumažinta duomenų dimensija naudojant principinių komponenčių analizę tam, kad būtų galima geriau identifikuoti klasterius.

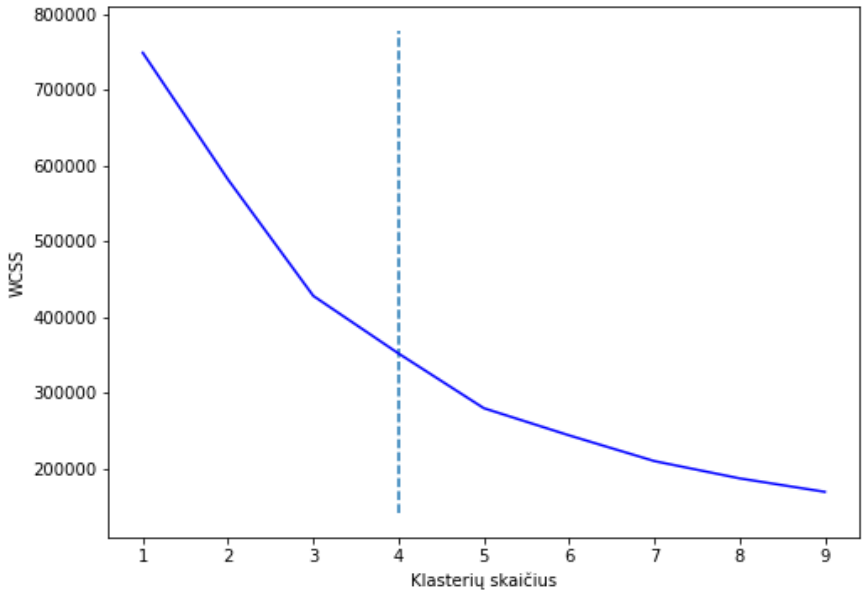
Klasterizavimo algoritmai atlikti šiam požymių rinkiniui: vidutinis peržiūrų, "patinka", "nepatinka" paspaudimų ir komentarų skaičius, miniatiūros kokybė, šviesumas, išsiliejimas ir kontrastas, prenumeratorių skaičius, vaizdo įrašo amžius, simbolių pavadinime skaičius, raktažodžių skaičius ir aprašymo egzistavimas.

Atlikus klasterizavimo algoritmus rezultatai vizualizuojami, siekiant išsiaiškinti grupių išsidėstymą koordinačių plokštumoje. Norint įvertinti grupių susidarymo tendencijas, kiekybiniams kintamiesiems suskaičiuoti vidurkiai grupėse.

Tam, kad palyginti visų trijų - kvartilių, $k$-vidurkių ir SOM - populiarumo grupių sudarymo metodus, pasirinkta viena grupė (nepopuliarių vaizdo įrašų klasė) ir palyginami vidurkiai visuose kintamuosiuose.

### 4.2.1 *k*-vidurkių metodas

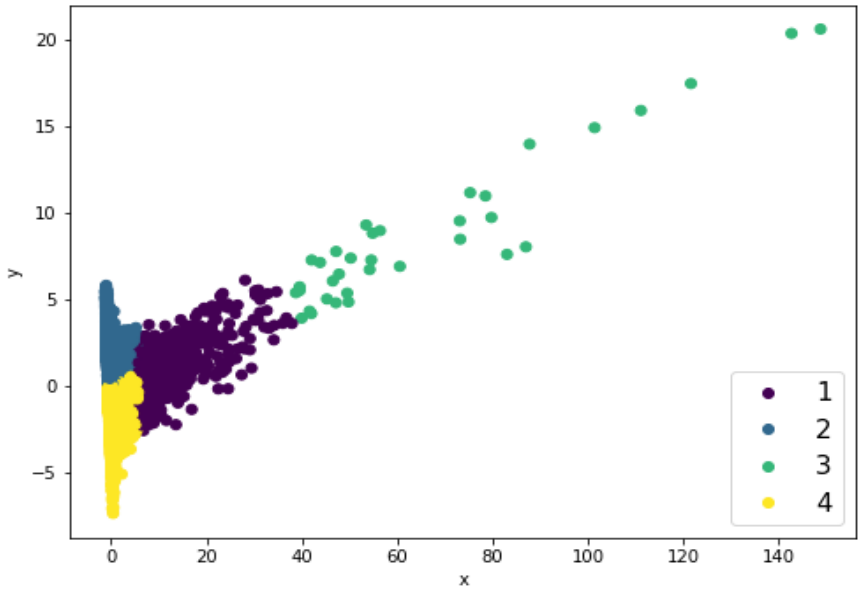
Prieš atliekant klasterinę analizę, yra ieškomas optimalus klasterių skaičius, naudojant alkūnės metodą tam, kad patikrinti ar statistiniu būdu sudarytos keturios populiarumo grupės atspindi duomenis. Alkūnės metodu nustatytas optimalus klasterių skaičius yra $k = 4$ (žr. 12 pav.)




12 pav.: Optimalus klasterių skaičius alkūnės metodu

Atlikus klasterinę analizę sumažintos dimensijos duomenims naudojant $k$-vidurkių metodą su parinktu klasterių skaičiumi $k = 4$, klasterių išsidėstymas matomas 13 paveikslėlyje. Iš šio paveikslėlio galima pastebėti, kad visos keturios grupės yra atsiskyrusios, tačiau tam, kad įvertinti kiekvieno klasterio skirtumus, suskaičiuoti kiekybinių požymių vidurkiai kiekvienoje grupėje.

Nagrinėjant grupių pasiskirstymą iš 13 pav., galima pastebėti, kad grupių dydžiai skiriasi - pirmąją bei trečiąją grupes sudaro mažesnis skaičius duomenų, negu antrąją ir ketvirtąją. Patikrinimui, ar antrosios grupės dalis yra nedengiama pirmosios ir antrosios, suskaičiuotas kiekvienos grupės imties dydis. Pirmąją grupę sudaro 675 vaizdo įrašai, antrąją - 59374, trečiąją tik 35, o ketvirtąją likę 72237 vaizdo įrašai, todėl galima teigti, kad $k$-vidurkiu algoritmu sugrupuoti duomenys yra nesubalansuoti.




13 pav.: $k$-vidurkių klasteriai su PCA

Remiantis 7 lentelės dinaminių požymių vidurkių rezultatais bei 13 paveikslėliu, teigiama, kad pirmoji grupė yra populiarių vaizdo įrašų grupė, antroji - nepopuliarių (dinaminių požymių vidurkiai grupėje yra mažiausi), trečioji - labai populiarių (dinaminių požymių vidurkiai grupėje yra didžiausi), o ketvirtoji - vidutinio populiarumo vaizdo įrašų grupė.

Pirmieji - dinaminiai požymiai - vidutinis peržiūrų, "patinka", "nepatinka", komentarų skaičius grupėse skiriasi. Matoma, kad kuo vaizdo įrašas populiaresnis, tuo didesnis yra auditorijos įsitraukimas. Tokia pati tendencija yra ir pradiniame duomenų suskirstyme pagal vidutinių peržiūrų kvartilius - kuo vaizdo įrašas populiaresnis, tuo dinaminių požymių vidurkis yra didesnis. Nagrinėjant vaizdo įrašų amžių, galima teigti, jog naujesniems vaizdo įrašams tapti populiaresniu yra didesnė tikimybė, negu senesniems. Ta pati tendencija matoma ir populiarumo grupių suskirstyme pagal kvartilius, tačiau $k$-vidurkių grupėse šis skirtumas yra ryškesnis.

Taip pat verta pastebėti, kad nepopuliarių ir vidutiniškai populiarių (antra ir ketvirta grupė atitinkamai) vaizdo įrašų prenumeratorių skaičius yra panašus - šiose grupėse esantys vaizdo įrašai patalpinti kanaluose, kurie vidutiniškai turi apie 5,6 mln. prenumeratorių. Kita vertus, pirmosios grupės vidutinis prenumeratorių skaičius skiriasi 3 kartus - vidutinis šių vaizdo įrašų prenumeratorių skaičius yra apie 16 milijonų, o labai populiarių vaizdo įrašų vidutinis prenumeratorių skaičius yra apie 21 mln., nuo antros ir ketvirtos populiarumo grupių skiriasi apie 4 kartus.

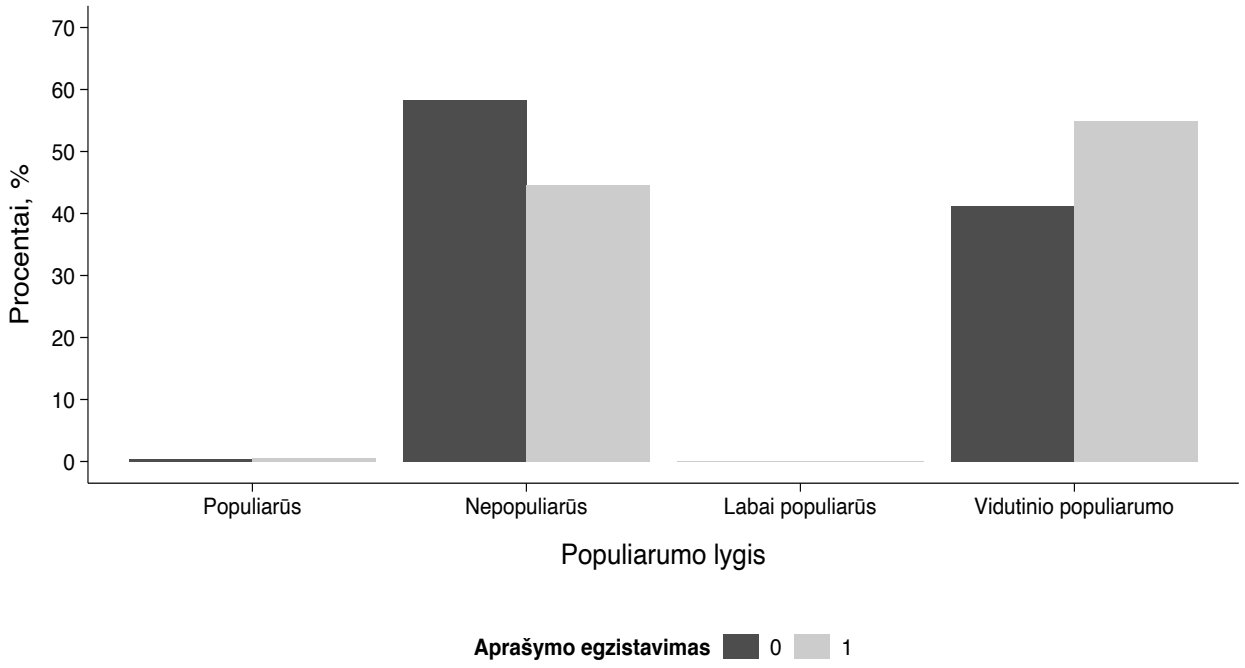
Nagrinėjant tekstinę informaciją, galima teigti, jog mažesnis skaičius raktinių žodžių ir simbolių pavadinime taip pat daro įtaką vaizdo įrašų populiarumui - vaizdo įrašas su mažesniu raktinių žodžių ir simbolių skaičiumi pavadinime, tikėtina, kad bus mažiau populiarus. Kita vertus, pernelyg didelis šių požymių skaičius vaizdo įraše taip pat nėra labai populiarų įrašų rodiklis - populiarių vaizdo įrašų (pirma grupė) vidutinis raktinių žodžių ir simbolių pavadinime skaičius yra mažesnis negu labai populiarų vaizdo įrašų (trečia grupė).

Miniatiūrą apibūdinantys požymiai - kokybė, šviesumas ir kontrastas - grupėse stipriai neišsiskiria, todėl galima pamanyti, jog tobulėjant technologijų amžiui - gerėjant kamerų, fotoaparatų bei vaizdo apdorojimo programų kokybei ir didėjant jų galimybėms, vaizdinė vaizdo įrašų kokybė taip pat gerėja - nuotraukos šviesios, ryškios ir geros kokybės. Vienintelis parametras, kuris išsiskiria yra miniatiūros išsiliejimas. Matoma, kad mažiausiai išsiliejusios miniatiūros yra vidutinio populiarumo vaizdo įrašų ($\mu_4 = 5271,32$), o daugiau išsiliejusių miniatiūrų turi nepopuliarūs vaizdo įrašai ($\mu_2 = 2080,87$).

| Kintamasis | **Vidurkis** | | | |
|---|---|---|---|---|
| | $\mu_1$ | $\mu_2$ | $\mu_3$ | $\mu_4$ |
| avg_daily_views | 431200 | 3766,8 | $1{,}32{\cdot}10^6$ | 8394,2 |
| avg_daily_comments | 567,5 | 4,79 | 5853,06 | 10,03 |
| avg_daily_likes | 8742,67 | 78,68 | 68954,1 | 154,81 |
| avg_daily_dislikes | 659,34 | 3,26 | 2724,87 | 10,16 |
| age | 207,42 | 1577,68 | 133,94 | 695,51 |
| channel_subscriber_count | $1,58 \cdot 10^7$ | $5,66 \cdot 10^6$ | $2,12 \cdot 10^7$ | $5,5 \cdot 10^6$ |
| chr_title | 54,96 | 40,39 | 44,26 | 58,36 |
| keywords_count | 39,7 | 26,22 | 35,03 | 46,04 |
| thumbnail_quality | 24,77 | 25,88 | 28,92 | 26,39 |
| thumbnail_blurriness | 3881,27 | 2080,87 | 3329,84 | 5271,32 |
| thumbnail_brightness | 141,64 | 124,25 | 138,36 | 151,47 |
| thumbnail_contrast | 63,08 | 57,6 | 62,64 | 69,91 |

7 lentelė: Vidurkiai $k$-vidurkių klasteriuose

Nagrinėjant aprašymo egzistavimo pasiskirstymą populiarumo grupėse (žr. 14 pav.), galima matyti, jog nepopuliarūs vaizdo įrašai turi daugiau vaizdo įrašų be jokio aprašymo, o vidutinio populiarumo vaizdo įrašų didesnę dalį sudaro tie vaizdo įrašai, kurie aprašymą turi. Kadangi populiarių ir labai populiarių vaizdo įrašų kiekis yra nedidelis, didelių skirtumų grafike nematyti. Tačiau nagrinėjant šias dvi imtis atskirai, populiariuose vaizdo įrašuose iš 675 visų vaizdo įrašų, tik 12 iš jų neturi vaizdo įrašo aprašymo, o labai populiariuose vaizdo įrašuose nėra nė vieno vaizdo įrašo be aprašymo.



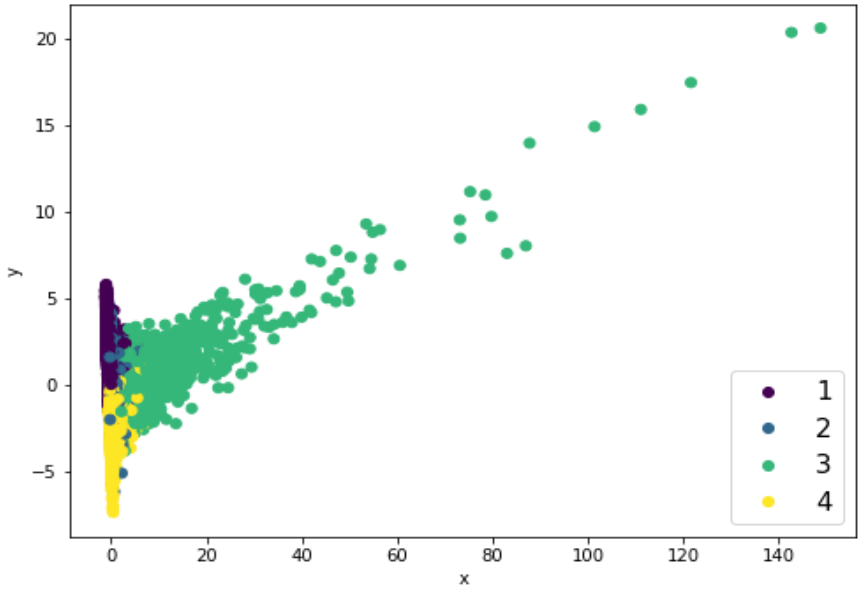

14 pav.: Aprašymo egzistavimas $k$-vidurkių populiarumo grupėse

Lyginant statistiniu metodu ir $k$-vidurkių algoritmu sudarytas populiarumo grupes, galima teigti, jog dinaminių požymių atžvilgiu grupių susidarymo tendencija yra panaši - kuo vaizdo įrašas populiaresnis, tuo dinaminių požymių vidurkis yra didesnis. Kadangi sudarytų grupių imčių dydžiai skiriasi nuo pradinių populiarumo grupių, palyginti, kokia dalis $k$-vidurkių sudarytų populiarumo grupės sutampa su kvartilių populiarumo grupėmis, yra sudėtinga. Tačiau iš sudarytų $k$-vidurkių populiarumo grupių, visi 35 labai populiarūs vaizdo įrašai yra taip pat labai populiarių vaizdo įrašų, kvartilių populiarumo grupės įrašai. Kita vertus, $k$-vidurkių populiarių vaizdo įrašų grupė

nėra tapati kvartilių populiarių vaizdo įrašų grupei - nė vienas $k$-vidurkių populiarios grupės įrašas nesutampa su kvartilių populiarių vaizdo įrašų grupe.

### 4.2.2 SOM

Saviorganizuojančių neurorinių tinklų algoritmui naudoti tokie pat požymiai kaip $k$-vidurkių algoritme. Tinklelio dydis parinktas $(1, 4)$ - tam, kad būtų galima palyginti su $k$-vidurkių algoritmu. Klasterių išsidėstymas pavaizduotas 15 paveikslėlyje. Čia matoma, kad pirmoji, antroji ir ketvirtoji grupės persidengia, todėl tikėtina, kad populiarumo grupių vidurkiai šiose grupėse mažai skirsis. Kita vertus, suskaičiavus grupių imties dydžius, teigiama, kad grupių dydžiai skiriasi - pirmąją grupę sudaro 62702, antrąją 15367, trečiąją 1635, o ketvirtąją grupę likę 52617 vaizdo įrašai.




15 pav.: SOM klasteriai

Kaip ir $k$-vidurkių klasterizavimo algoritme, taip ir SOM, apskaičiuotas kiekvieno požymių vidurkis grupėse. Kadangi pačių grupių išsidėstymas labai panašus kaip ir ankstesnio algoritmo (13 pav. ir 15 pav. atitinkamai), galima tikėtis, kad ir vidurkių rezultatai bus panašūs.

| Kintamasis | **Vidurkis** | | | |
|---|---|---|---|---|
| | $\mu_1$ | $\mu_2$ | $\mu_3$ | $\mu_4$ |
| `avg_daily_views` | 2059,3 | 14891,2 | 315527 | 5577.52 |
| `avg_daily_comments` | 3,53 | 12,48 | 367,82 | 11,06 |
| `avg_daily_likes` | 55,92 | 207,79 | 5853,05 | 150,15 |
| `avg_daily_dislikes` | 1,63 | 15,25 | 502,09 | 5,9 |
| `age` | 1500,58 | 1291,74 | 288,395 | 563,475 |
| `channel_subscriber_count` | $3,02 \cdot 10^6$ | $2,56 \cdot 10^7$ | $1,75 \cdot 10^7$ | $2,55 \cdot 10^6$ |
| `chr_title` | 40,83 | 48,3 | 60,78 | 61,78 |
| `keywords_count` | 24,1 | 47,9 | 41,61 | 49,33 |
| `thumbnail_quality` | 26,86 | 25,54 | 23,59 | 25,58 |
| `thumbnail_blurriness` | 2437,33 | 3833,78 | 3792,2 | 5495,01 |
| `thumbnail_brightness` | 127,43 | 154,61 | 146,73 | 148,5 |
| `thumbnail_contrast` | 59,88 | 61,47 | 61,82 | 70,7 |

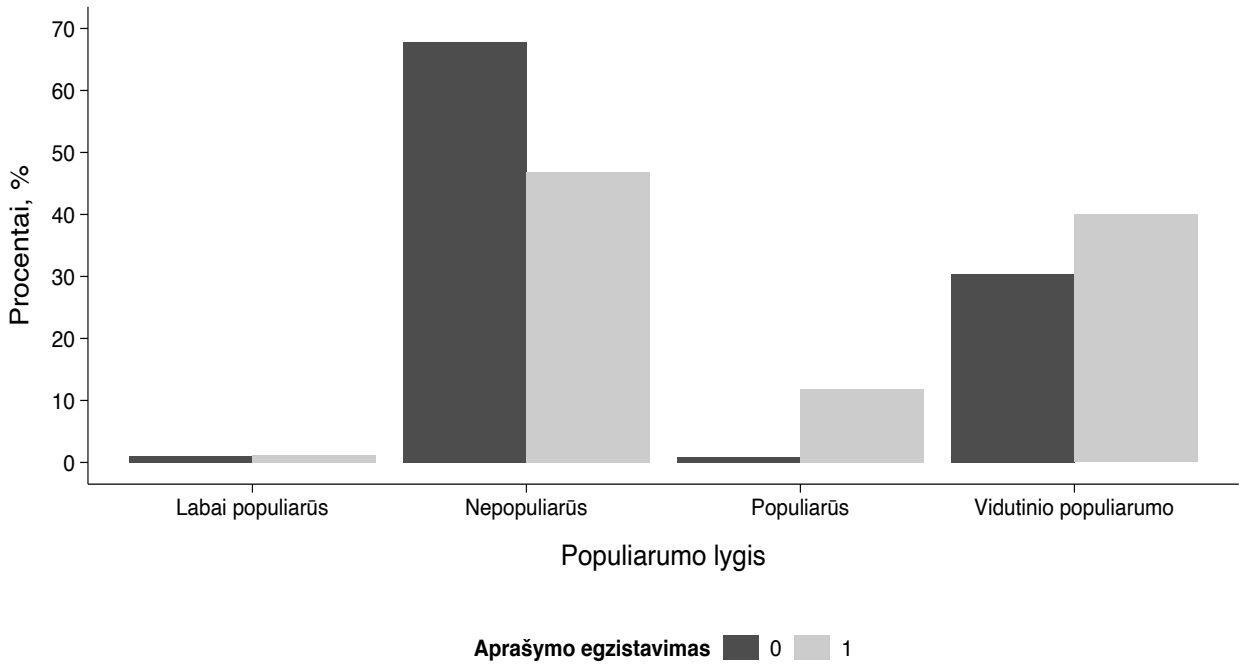
8 lentelė: Vidurkiai SOM klasteriuose

Remiantis 8 lentelės ir 15 paveikslėlio rezultatais, laikoma, kad pirmoji grupė yra nepopuliarių, antroji - populiarių, trečioji labai populiarių, o ketvirtoji - vidutinio populiarumo vaizdo įrašų gru-

pė, remiantis dinaminiais - vidutiniais peržiūrų, "patinka", "nepatinka" paspaudimų ir komentarų skaičiais - požymiais. Kaip ir $k$-vidurkių, taip ir SOM grupavimo algoritme rezultatai panašūs - kuo vaizdo įrašas populiaresnis, tuo dinaminiai požymiai vidurkiuose yra didesni - auditorijos įsitraukimas į vaizdo įrašą yra didesnis. Populiaresnis vaizdo įrašas sulaukia daugiau peržiūrų, yra daugiau komentuojamas, susilaukia daugiau "patinka" ir "nepatinka" paspaudimų. Taip pat populiaresnis vaizdo įrašas yra naujesnis, turi daugiau kanalo prenumeratorių. Kita vertus, verta paminėti, kad populiarių vaizdo įrašų kanalo prenumeratorių skaičius yra didesnis negu labai populiarių vaizdo įrašų, tačiau skirtumas nėra labai žymus.

Žiūrint į miniatiūrų kokybės vertinimo požymių (miniatiūros kokybė, šviesumas, išsiliejimas ir kontrastas) ir tekstinio turinio kokybės vertinimo požymių (raktažodžių ir simbolių pavadinime skaičių) grupių vidurkius, rezultatai gana panašūs kaip ir $k$-vidurkių grupavimo algoritme: nepopuliarūs vaizdo įrašai turi mažiau tekstinės informacijos apie vaizdo įrašą - mažiau simbolių pavadinime ir raktažodžių. Taip pat galima pastebėti, kad ir miniatiūros kokybės rodikliai yra prastesni - vidutiniškai miniatiūros yra prastesnės kokybės, daugiau išsiliejusios, mažiau šviesios ir ne tokios kontrastingos, nors šie vidurkių skirtumai nėra labai dideli.

16 paveikslėlyje matomas aprašymo egzistavimo pasiskirstymas SOM populiarumo grupėse. Rezultatų tendencija panaši kaip ir $k$-vidurkių algoritmo - nepopuliarūs vaizdo įrašai turi daugiau vaizdo įrašų be aprašymo, o vidutinio populiarumo tokių vaizdo įrašų dalis yra mažesnė. Tačiau nagrinėjant labai populiarius ir populiarius vaizdo įrašus, čia aprašymo egzistavimo tendencija yra kitokia - iš 1635 labai populiarių vaizdo įrašų, 32 vaizdo įrašai neturi aprašymo, o populiarių vaizdo įrašų be aprašymo vaizdo įrašų dalis yra mažesnė - iš 15367 vaizdo įrašų, tik 29 neturi aprašymo.




16 pav.: Aprašymo egzistavimas SOM populiarumo grupėse

Kadangi, kaip ir $k$-vidurkių algoritme, taip ir SOM populiarumo grupių imties dydžiai skiriasi nuo pradinių populiarumo grupių imties dydžių, palyginti, kokia dalis vaizdo įrašų SOM grupėse sutampa su kvartilių grupėmis, yra lygiai taip pat sudėtinga. Tačiau kitaip, negu $k$-vidurkių grupių, čia visose SOM grupėse yra sutampančių vaizdo įrašų su kvartilių populiarumo grupėmis.

### 4.2.3 Grupių sudarymo metodų palyginimas

Tam, kad palyginti visas populiarumo grupių susidarymo tendencijas, nagrinėjama nepopuliarių vaizdo įrašų grupė, kadangi tiek $k$-vidurkių, tiek SOM algoritme ši grupė yra viena didžiausių. Lyginant visų trijų algoritmų (kvartilių, SOM ir $k$-vidurkių) požymių statistiką (vidurkį $\mu$ ir standartinį nuokrypį $\sigma$) 9 lentelėje matoma, kad požymių reikšmės šiek tiek skiriasi.

| Kintamasis | Kvartilių metodas | | $k$-vidurkių metodas | | SOM metodas | |
|---|---|---|---|---|---|---|
| | $\mu_{\text{Q}}$ | $\sigma_{\text{Q}}$ | $\mu_{k\text{-mean}}$ | $\sigma_{k\text{-mean}}$ | $\mu_{\text{SOM}}$ | $\sigma_{\text{SOM}}$ |
| avg_daily_views | 13,28 | 13,51 | 3766,9 | 16102,43 | 2059,30 | 7796,6 |
| avg_daily_comments | 0,1 | 4,9 | 4,79 | 22,07 | 3,53 | 15,78 |
| avg_daily_likes | 0,63 | 31,03 | 78.68 | 327.94 | 55,92 | 235,42 |
| avg_daily_dislikes | 0,03 | 0,29 | 3,2592 | 18,75 | 1,6289 | 8,56 |
| age | 1499 | 1229,64 | 1578 | 1151,15 | 1501 | 1136,13 |
| channel_subscriber_count | $1{,}8{\cdot}10^6$ | $1{,}6{\cdot}10^6$ | $5{,}7{\cdot}10^6$ | $9{,}4{\cdot}10^6$ | $3{\cdot}10^6$ | $4{\cdot}10^6$ |
| chr_title | 48,64 | 22,03 | 40,39 | 17,43 | 40,83 | 17,43 |
| keywords_count | 28,65 | 24,09 | 26,22 | 21,41 | 24,1 | 19,88 |
| thumbnail_quality | 28,28 | 15,64 | 25,88 | 14,91 | 26,86 | 15,3 |
| thumbnail_blurriness | 3107 | 2760,7 | 2080,9 | 1516,97 | 2437,3 | 1868,84 |
| thumbnail_brightness | 131 | 39,24 | 124,3 | 36,74 | 127,4 | 37,34 |
| thumbnail_contrast | 63,16 | 15,43 | 57,69 | 13,11 | 59,88 | 13,78 |

9 lentelė: Nepopuliarių vaizdo įrašų statistika požymiuose

Nagrinėjant dinaminius požymius, jų vidurkiai metoduose skiriasi. $k$-vidurkių vaizdo įrašai apima platesnę imtį - vaizdo įrašai turi daugiau peržiūrų ir "nepatinka" paspaudimų. Kita vertus, amžiaus, vidutiniai miniatiūrų kokybės ir tekstiniai požymiai visose metoduose skiriasi mažiau. Aprašymo egzistavimo nepopuliariuose vaizdo įrašuose visuose metoduose tendencija taip pat yra panaši: didžioji dalis vaizdo įrašų šioje grupėje neturi aprašymo (9, 14, 16 paveikslėliai).

Taigi, skirtingi algoritmai duoda gana skirtingus populiarumo grupių susidarymo rezultatus su tam tikromis panašiomis charakteristikomis - populiaresniuose vaizdo įrašuose auditorijos įsitraukimas yra didesnis, mažesnė dalis vaizdo įrašų neturi aprašymo. Tačiau teigti, jog statistiniu būdu sudarytos populiarumo grupės yra netinkamos, negalima. Nors šiuo būdu atsižvelgiama tik į vaizdo įrašo amžių ir peržiūras, kiti požymiai taip pat turi panašų ryšį su populiarumo grupėmis kaip ir $k$-vidurkių bei SOM.

## 4.3 Klasifikavimo rezultatai

Tam, kad nuspėti vaizdo įrašų populiarumo klases naudojant metaduomenų požymių rinkinį, atliktas vaizdo įrašų klasifikavimas. Duomenys klasifikuojami trimis - atraminių vektorių, atsitiktinio miško ir daugialypės logistinės regresijos klasifikatoriais. SVM, atsitiktinio miško ir daugialypės regresijos klasifikatoriams apmokyti naudojama atsitiktiniu būdu parinkta subalansuota (kiekvieną klasę sudaro vienodas skaičius vaizdo įrašų) imtis, sudaranti 70% duomenų. Rezultatų patikrinimui naudojamas 10 dalių kryžminis patikrinimas. Likusi 30% duomenų imtis sudaro testavimo aibę, kuri skirta patikrinti, kaip gerai klasifikatorius klasifikuoja nematytus duomenis.

Klasifikavimo algoritmai atliekami kvartilių metodu suskirstytai aibei, naudojant vienuolikos požymių rinkinį: kanalo prenumeratorių skaičius, vidutinis "patinka", "nepatinka" paspaudimų skaičius per dieną, vidutinis komentarų skaičius per dieną, vaizdo įrašo aprašymo egzistavimas, simbolių pavadinime skaičius, raktinių žodžių skaičius, miniatiūros kokybė, išsiliejimas, kontrastas ir miniatiūros šviesumas. Į šiuos klasifikatorius nėra įtraukiami du kintamieji - vidutinis peržiūrų skaičius per dieną ir vaizdo įrašo amžius tam, kad išvengti tiesinio ryšio tarp populiarumo grupių.

### 4.3.1 Atraminių vektorių klasifikatorius

Prieš atliekant SVM klasifikatorių, kiekybiniai duomenys yra standartizuojami pagal 3.3 formulę. Kintamųjų standartizacija yra būtina šiam klasifikatoriui, kadangi tokiu būdu visų kintamųjų rėžiai turės panašią įtaką skaičiuojant atstumus hiperplokštumos konstravimui.

Kadangi darbe naudojamas SVM radialinis branduolys (naudojant tiesinį SVM branduolį, pasiektas labai žemas 0,43 bendras klasifikavimo tikslumas), šiam klasifikatoriui reikalingi du specifiniai parametrai $C$ (bendras parametras kiekvieno tipo branduoliams) ir $\gamma$ (išskirtinis radialinio branduolio parametras).

- $C > 0$ - parametras, apibrėžiantis tolerantiškumą klaidingam klasifikavimui. Kuomet parametras $C$ parinktas per mažas, hiperplokštumos paraštė (angl. *margin*) bus platesnė, modelis parenka mažiau taškų kaip atraminius vektorius ir taip gaunama mažesnė dispersija ir egzistuoja didesnis reikšmių poslinkis. Tačiau per mažas $C$ parametras gali lemti per didelį prisitaikymą prie mokymosi imties (angl. *overfitting*). Kuomet $C$ parametras parenkamas didesnis, tuomet hiperplokštumos paraštė bus siauresnė, modelis parenka daugiau taškų kaip atraminius vektorius, o tai reiškia didesnę dispersiją ir mažesnį reikšmių poslinkį. Kita vertus, per didelis $C$ parametras gali lemti prastesnį bendrą klasifikatoriaus tikslumą mokymosi imčiai.

- $\gamma$ - parametras, atsakingas už sprendimo ribos sklandumą ir kontroliuoja modelio dispersiją. Kuomet $\gamma$ yra labai didelis, tuomet sprendimo riba yra labai vingiuota, dėl to modelio dispersija gali būti didesnė ir tas gali lemti per didelį prisitaikymą prie mokymosi imties. Kuomet parametras $\gamma$ yra pernelyg mažas, riba yra glotnesnė ir dėl to modelio dispersija yra mažesnė, todėl toks modelis gali neteisingai klasifikuoti sudėtingiau išsidėsčiusius duomenų taškus.

Radialinio branduolio optimalūs parametrai $C$ ir $\gamma$ parinkti naudojant parametrų tinklelinę paiešką (angl. *grid-search*). Kryžminiu patikrinimu išbandytos kombinacijos matomos lentelėje:

| **Parametras** | **Išbandytos reikšmės** | **Geriausio modelio parametro reikšmė** |
|---|---|---|
| $C$ | 0,1; 1; 10; 100; 1000 | 1000 |
| $\gamma$ | 1; 0,1; 0,01; 0,001; 0,0001 | 1 |

10 lentelė: SVM klasifikatoriaus parametrai

Remiantis 10 lentelės parametrais, optimalūs parametrai yra $C = 1000$ ir $\gamma = 1$, o bendras klasifikavimo tikslumas naudojant šiuos parametrus testavimo duomenims yra 0,67. Testavimo imčiai pritaikytas klasifikatorius pasiekė 0,68 bendrą klasifikavimo tikslumą.

### 4.3.2 Atsitiktinio miško klasifikatorius

Priešingai negu SVM klasifikatorius, atsitiktinio miško klasifikatoriui duomenų transformacija yra nereikalinga. Tačiau lygiai taip pat kaip ir SVM klasifikatorius, RF klasifikatoriui reikalingi parinkti tam tikri parametrai:

- `n_estimator` - sprendimų medžių kiekis miške.

- `max_features` - kiekvieno padalijimo metu atsitiktinai atrinktų požymių skaičius. Galimos reikšmės:

  - `log2` - $\log_2$(požymių skaičius).
  - `sqrt` - $\sqrt{\text{požymių skaičius}}$.

- `max_leaf_nodes` - bendras mazgų skaičius kiekviename sprendimų medyje;

- `min_samples_split` - minimalus duomenų kiekis taškų, reikalingas toliau skaidyti medžio lapams;

- `min_samples_leaf` - minimalus duomenų taškų kiekis lapui.

- `max_depth` - maksimalus sprendimų medžio gylis;

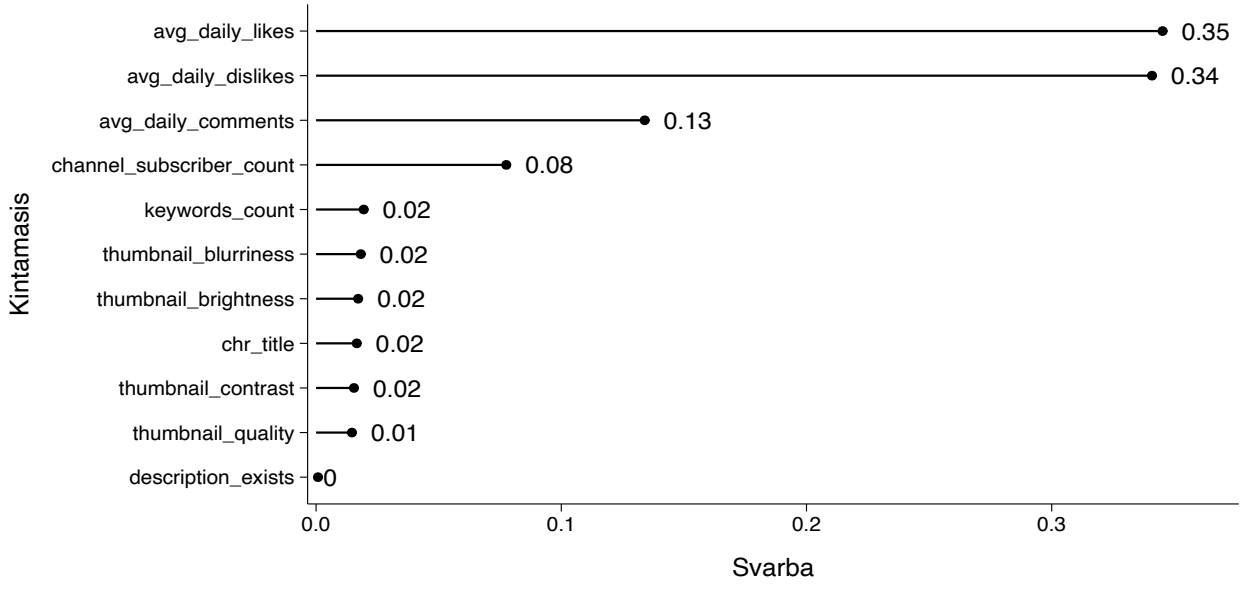
Iš viso buvo apmokyti 648 atsitiktinio miško modeliai su 11 lentelėje išbandytomis reikšmėmis. Atsitiktinio miško klasifikatorius, įgyjantis aukščiausią bendrą klasifikatoriaus tikslumą (0,87) yra sudarytas iš 600 sprendimų medžių, turintis 2500 bendrą mazgų skaičių kiekviename sprendimų medyje (žr. 11 lent.). Testavimo imties bendras klasifikavimo tikslumas yra 0,87.

| Parametras | Išbandytos reikšmės | Geriausio modelio parametro reikšmė |
|---|---|---|
| `n_estimator` | 300, 400, 500, 600 | 600 |
| `max_features` | `log2`, `sqrt` | `log2` |
| `max_leaf_nodes` | 2200, 2300, 2500 | 2500 |
| `min_samples_split` | 2, 5, 10 | 10 |
| `min_samples_leaf` | 1, 2, 5 | 2 |
| `max_depth` | 30, 40, 50 | 40 |

11 lentelė: Atsitiktinio miško klasifikatoriaus parametrai

Atlikus atsitiktinio miško klasifikatorių, įvertinta požymių svarba klasifikatoriaus tikslumui. Požymio svarbumas yra visų medžių sprendimų vidurkis, tai reiškia, kad požymio svarba yra vidutinė tarp visų medžių. Visų kintamųjų FI suma yra lygi 1. Remiantis 17 grafiku, pagrindiniai trys kintamieji yra:

1. vidutinis "patinka" paspaudimų skaičius per dieną - $\text{FI}_{(\texttt{avg\_daily\_likes})} = 0,35$.
2. vidutinis "nepatinka" paspaudimų skaičius per dieną - $\text{FI}_{(\texttt{avg\_daily\_dislikes})} = 0,34$.
3. vidutinis komentarų skaičius per dieną - $\text{FI}_{(\texttt{avg\_daily\_comments})} = 0,13$.




17 pav.: Atsitiktinio miško klasifikatoriaus požymių svarba

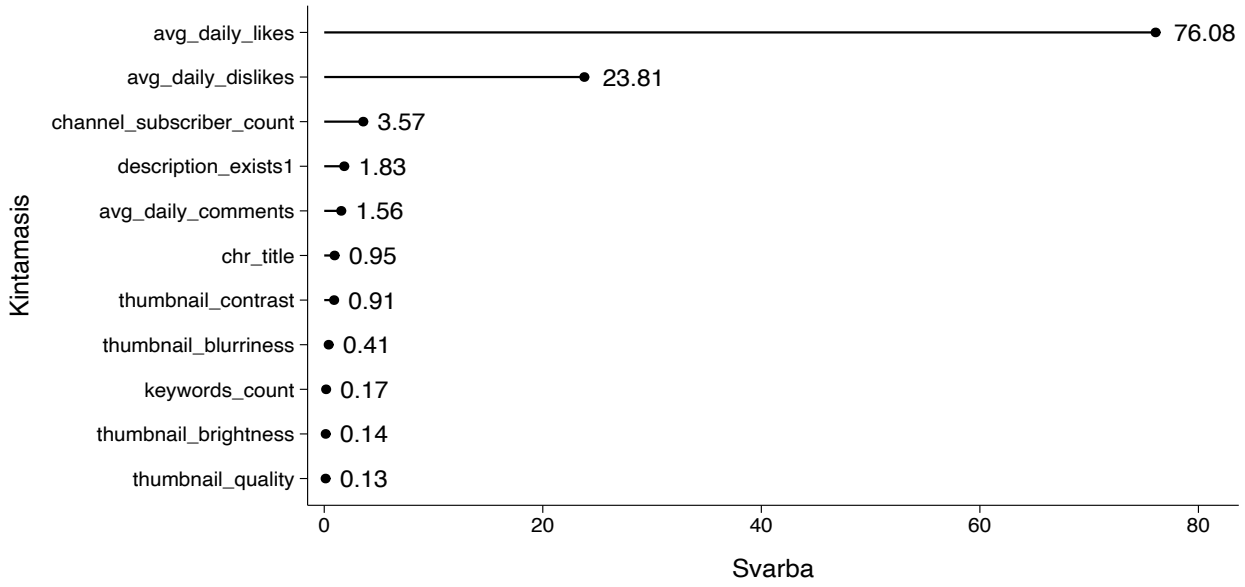
### 4.3.3 Daugialypės logistinės regresijos klasifikatorius

MLR klasifikatoriui parametrų parinkimas yra nereikalingas, tačiau šiam klasifikatoriui yra naudojami tik statistiškai reikšmingi kintamieji, todėl prieš konstruojant daugialypės logistinės regresijos klasifikatorių, atliekama išsami požymių analizė. Taip pat kaip ir SVM, šiam klasifikatoriui reikalingas duomenų transformavimas, tam kad pašalinti vidurkio efektą kintamiesiems - duomenys standartizuojami. Kita vertus, standartizavimas nepaveikia asimetrijos ir eksceso, todėl kiekybiniams priklausomiems kintamiesiems pritaikoma kvadratinės šaknies transformacija, kadangi kintamieji yra stipriai asimetriški.

Šiame klasifikatoriaus algoritme pirma populiarumo grupė yra laikoma kaip kontrolinė kategorija. Apmokius modelį, vykdoma kintamųjų atranka. Kadangi nė vienas koeficientas prie kintamųjų visose grupėse nebuvo reikšmingai nesiskiriantis nuo nulio su $\alpha = 0,05$ pasikliovimo lygmeniu, visi kintamieji modelyje yra paliekami ir antras modelis neapmokomas (27, 28 ir 29 lentelės).

Mokymosi duomenims, atliktas daugialypės logistinės regresijos klasifikatorius pasiekė 0,83 bendrą klasifikavimo tikslumą. Pritaikius MLR testavimo duomenims, gautas toks pats bendras klasifikavimo tikslumas kaip ir mokymosi duomenims – 0,83.

Kaip ir atsitiktinio miško klasifikatoriuje, taip ir MLR, svarbiausi požymiai sutampa: vidutinis "patinka" paspaudimų skaičius per dieną (`avg_daily_likes`) ir vidutinis "nepatinka" paspaudimų skaičius per dieną (`avg_daily_dislikes`). Kita vertus, trečiasis požymis MLR klasifikatoriuje yra kanalo prenumeratorių skaičius (`channel_subscriber_count`).




18 pav.: Daugialypės logistinės regresijos klasifikatoriaus požymių svarba

### 4.3.4 Klasifikatorių palyginimas

12 lentelėje pateikti visų naudotų klasifikatorių rezultatai testavimo duomenims. Kaip matoma iš pateiktos lentelės, SVM klasifikatoriumi prasčiausiai identifikuojama antra klasė, o tiksliausiai ketvirta. Įvertinus RF klasifikatoriaus jautrumą ir tikslumą, galima daryti išvadą, kad šis klasifikatorius gana tiksliai suskirstė objektus į reikiamas klases. Nagrinėjant MLR klasifikatoriaus jautrumą, matoma, kad šis klasifikatorius yra mažiau jautrus antros ir trečios populiarumo klasės atžvilgiu, tačiau gana tiksliai klasifikuoja kitas klases.

Lyginant visų atliktų klasifikatorių rezultatus testavimo imčiai, tiksliausiai duomenis suskirstė į populiarumo klases atsitiktinio miško klasifikatorius, kurio bendras klasifikatoriaus tikslumas yra 0,87. Žemiausią tikslumą turi SVM klasifikatorius – testavimo duomenims šis klasifikavimo algoritmas pasiekė 0,68 bendrą klasifikavimo tikslumą.

| | **SVM** | | **Atsitiktinis miškas** | | **MLR** | |
|---|---|---|---|---|---|---|
| **Klasė** | Jautrumas | Tikslumas | Jautrumas | Tikslumas | Jautrumas | Tikslumas |
| 1 | 0,71 | 0,63 | 0,93 | 0,91 | 0,91 | 0,86 |
| 2 | 0,57 | 0,54 | 0,83 | 0,83 | 0,77 | 0,77 |
| 3 | 0,62 | 0,68 | 0,82 | 0,82 | 0,77 | 0,77 |
| 4 | 0,81 | 0,89 | 0,90 | 0,92 | 0,86 | 0,91 |
| Bendras klasifikavimo tikslumas | 0,68 | | 0,87 | | 0,83 | |

12 lentelė: Klasifikavimo rezultatai

### 4.3.5 *k*-vidurkių ir SOM sudarytų grupių klasifikavimas

Išsamesnis klasifikavimas klasterizavimo algoritmų duomenims, skirtas pasižiūrėti kaip gerai klasifikatorius klasifikuoja vaizdo įrašus į klases, atliktas naudojant atsitiktinio miško klasifikatorių. Šis klasifikatorius parinktas todėl, jog jis tiksliausiai klasifikuoja vaizdo įrašus, padalintus pagal kvartilių metodą.

Kadangi tiesinio ryšio tarp populiarumo grupių ir amžiaus bei vaizdo įrašų peržiūrų skaičiaus nėra, šiuose klasifikatoriuose naudojamas papildytas požymių rinkinys: kanalo prenumeratorių skaičius, vaizdo įrašų amžius dienomis, vidutinis peržiūrų, "patinka", "nepatinka" paspaudimų skaičius per dieną, vidutinis komentarų skaičius per dieną, vaizdo įrašo aprašymo egzistavimas, simbolių pavadinime skaičius, raktinių žodžių skaičius, miniatiūros kokybė, išsiliejimas, kontrastas ir miniatiūros šviesumas. Taigi, naudojama trylika kintamųjų, taikant tokią pat klasifikavimo metodiką kaip ir kvartilių grupių žymėjimo atveju. Duomenų aibė suskirstyta į 70% mokymosi ir 30% testavimo imtis. Klasifikatoriaus apmokymui naudojamas 10 dalių kryžminis patikrinimas.

#### 4.3.5.1 *k*-vidurkių grupių klasifikavimas

Šiai duomenų imčiai išbandytos 13 lentelėje nurodytos reikšmės. Pritaikius mokymosi duomenims aukščiausią bendrą klasifikavimo tikslumą turintį klasifikatorių su geriausiomis parametro reikšmėmis testavimo duomenims, gautas 0,97 bendras klasifikavimo tikslumas.

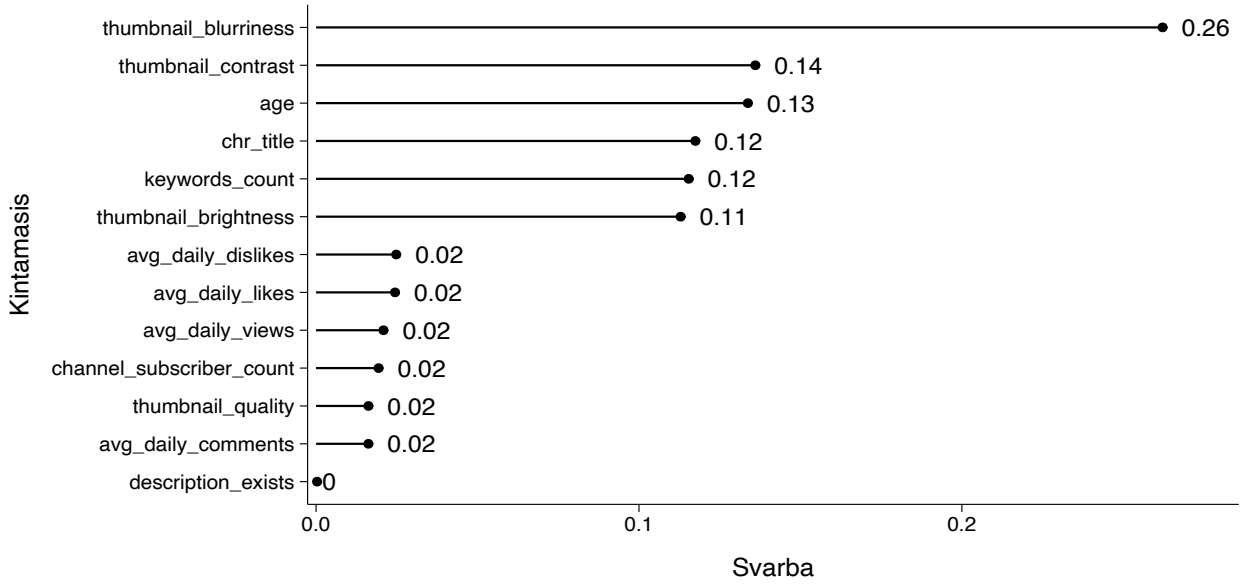
| **Parametras** | **Išbandytos reikšmės** | **Geriausio modelio parametro reikšmė** |
|---|---|---|
| `n_estimator` | 200, 250, 300, 500 | 500 |
| `max_features` | `log2`, `sqrt` | `log2` |
| `min_samples_split` | 2, 5, 10, 12, 20 | 2 |
| `min_samples_leaf` | 1, 2, 5, 10, 15 | 2 |
| `max_depth` | 10, 20, 30, 40, 50 | `None` |

13 lentelė: Atsitiktinio miško klasifikatoriaus parametrai *k*-vidurkių duomenų imčiai

Kaip ir kvartilių metodu suskirstytų duomenų atveju, taip ir *k*-vidurkių metodu suskirstytų duomenų aibei, įvertinta atsitiktinio miško klasifikatoriaus požymių svarba bendram klasifikavimo tikslumui. Remiantis 19 grafiku, matoma, kad trys pagrindiniai kintamieji yra

1. miniatiūros išsiliejimas - $\mathrm{FI}_{(\texttt{thumbnail\_blurriness})} = 0,26$.
2. miniatiūros kontrastas - $\mathrm{FI}_{(\texttt{thumbnail\_contrast})} = 0,14$.
3. vaizdo įrašų amžius - $\mathrm{FI}_{(\texttt{age})} = 0,13$.

Priešingai negu kvartilių atsitiktinio miško klasifikatoriuje, svarbiausi požymiai yra ne dinaminiai - atspindintys auditorijos įsitraukimą, o miniatiūros kokybės įvertinimo požymiai - miniatiūros išsiliejimas ir kontrastas. Taip pat matoma, kad svarbus kintamasis yra ir vaizdo įrašo amžius.




19 pav.: Atsitiktinio miško klasifikatoriaus požymių svarba $k$-vidurkių duomenų imčiai

#### 4.3.5.2 SOM grupių klasifikavimas

Kaip ir $k$-vidurkių, taip SOM duomenų aibei išbandyti tokie pat atsitiktinio miško klasifikatoriaus parametrai. Geriausio modelio parametro reikšmės yra nurodytos 14 lentelėje. Pritaikius klasifikatorių testavimo duomenis, gautas 0,96 bendras klasifikavimo tikslumas.

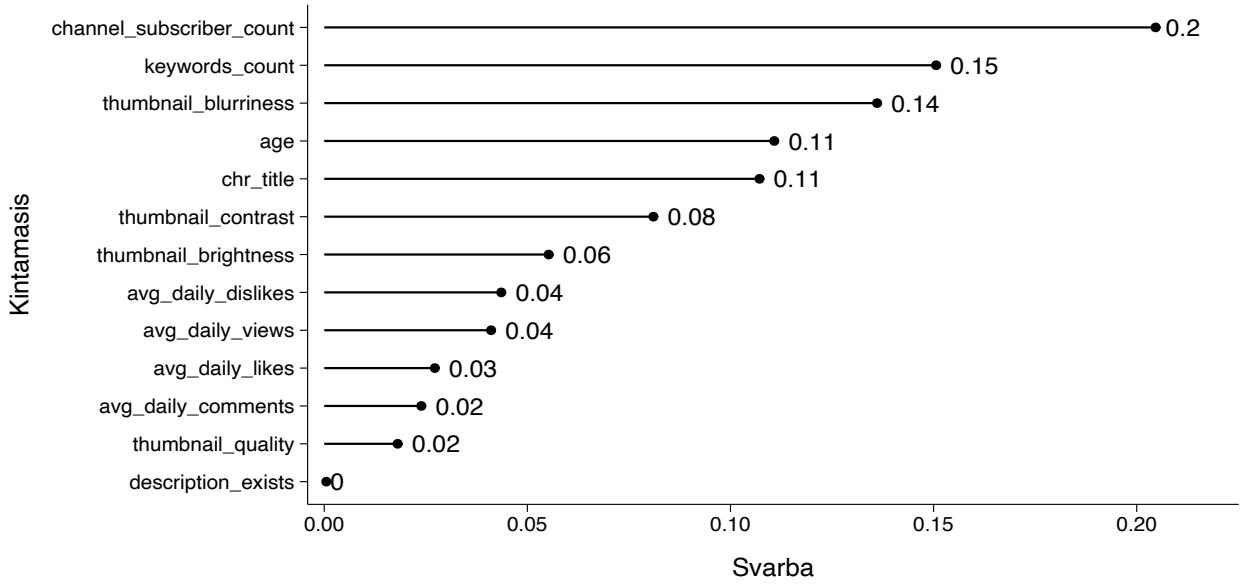
| **Parametras** | **Išbandytos reikšmės** | **Geriausio modelio parametro reikšmė** |
|---|---|---|
| `n_estimator` | 200, 250, 300, 500 | 250 |
| `max_features` | `log2`, `sqrt` | `sqrt` |
| `min_samples_split` | 2, 5, 10, 12, 20 | 2 |
| `min_samples_leaf` | 1, 2, 5, 10, 15 | 1 |
| `max_depth` | 10, 20, 30, 40, 50 | 40 |

14 lentelė: Atsitiktinio miško klasifikatoriaus parametrai SOM duomenų imčiai

Išnagrinėjus SOM metodu sugrupuotiems duomenims pritaikyto RF klasifikatoriaus požymių svarbą (žr. 20 pav.), matoma, kad rezultatai nuo kvartilių ir $k$-vidurkių RF klasifikatoriaus požymių svarbos skiriasi. Čia svarbiausi kintamieji yra

1. kanalo prenumeratorių skaičius - $\mathrm{FI}_{(\texttt{channel\_subscriber\_count})} = 0,2$.
2. raktinių žodžių skaičius - $\mathrm{FI}_{(\texttt{keywords\_count})} = 0,15$.
3. miniatiūros išsiliejimas - $\mathrm{FI}_{(\texttt{thumbnail\_blurriness})} = 0,14$.

Kita vertus, vaizdo įrašų amžius taip pat yra gana svarbus kintamasis kaip ir $k$-vidurkių RF klasifikatoriaus požymių svarbos atveju. Tačiau čia, jo svarba yra ketvirtoje vietoje. Taip pat verta paminėti, jog kintamasis, nusakantis ar egzistuoja aprašymas vaizdo įrašuose, visuose atsitiktinio miško modeliuose yra mažiausiai svarbus.




20 pav.: Atsitiktinio miško klasifikatoriaus požymių svarba SOM duomenų imčiai

#### 4.3.5.3 Klasifikatorių palyginimas

15 lentelėje pateikti atsitiktinio miško klasifikavimo rezultatai, kuomet populiarumo klasės žymimos remiantis klasterizavimo algoritmais.

| | **$k$-vidurkių RF klasifikatorius** | | **SOM RF klasifikatorius** | |
|---|---|---|---|---|
| **Klasė** | Jautrumas | Tikslumas | Jautrumas | Tikslumas |
| Nepopuliarūs | 0,96 | 0,97 | 0,97 | 0,96 |
| Vidutinio populiarumo | 0,97 | 0,97 | 0,96 | 0,96 |
| Populiarūs | 0,76 | 0,88 | 0,94 | 0,96 |
| Labai populiarūs | 0 | 0 | 0,94 | 0,98 |
| Bendras klasifikavimo tikslumas | 0,97 | | 0,96 | |

15 lentelė: Atsitiktinio miško klasifikavimo rezultatai klasterizavimo duomenims

Kadangi $k$-vidurkių ir SOM populiarumo klasių žymėjimo imtis yra nesubalansuota, vertinant kaip gerai klasifikatorius suskirstė objektus į reikiamas klases, yra remiamasi kiekvienos klasės jautrumu ir tikslumu atskirai. Iš šios lentelės galima pastebėti, jog $k$-vidurkių RF klasifikatoriaus labai populiarių vaizdo įrašų klasės jautrumas ir tikslumas yra 0. Tai reiškia, kad klasifikatorius šiai klasei priklausančius duomenų taškus priskyrė populiarių vaizdo įrašų klasei. Dėl to, populiarių vaizdo įrašų identifikavimas yra taip pat prastesnis (jautrumas 0,76 ir tikslumas 0,88) nepopuliarių ir vidutinio populiarumo klasių atžvilgiu. Geriausiai identifikuojama nepopuliarių ir vidutinio populiarumo klasės (jautrumas 0,96 ir tikslumas 0,97 bei jautrumas 0,97 ir tikslumas 0,97 atitinkamai). Šiuo atveju galima daryti išvadą, kad $k$-vidurkių labai populiarių vaizdo įrašų klasė yra panaši į populiarių vaizdo įrašų klasę, todėl tokiu atveju vertėtų šias pastarąsias dvi klases sujungti.

Žiūrint į SOM RF klasifikatoriaus rezultatus matoma, kad visos klasės yra identifikuojamos panašiai gerai, o tai reiškia, kad visos keturios šio žymėjimo klasės yra labiau atskiriamos ir tarp klasių egzistuoja didesni skirtumai, nors kaip ir $k$-vidurkių atveju, taip SOM populiarumo klasių žymėjimo imtis yra nesubalansuota.

### 4.3.6 Klasifikavimo rezultatai naudojant skirtingą populiarumo grupių žymėjimą

Klasifikavimo rezultatų patikrinimui kiekvienu atskiru klasių žymėjimų atveju, visi klasifikavimo algoritmai atlikti naudojant prieš tai aprašytą metodologiją. $k$-vidurkių ir SOM populiarumo klasių žymėjime klasifikavimui naudojamas požymių rinkinys: kanalo prenumeratorių skaičius, vaizdo įrašų amžius dienomis, vidutinis peržiūrų, "patinka", "nepatinka" paspaudimų skaičius per dieną, vidutinis komentarų skaičius per dieną, vaizdo įrašo aprašymo egzistavimas, simbolių pavadinime skaičius, raktinių žodžių skaičius, miniatiūros kokybė, išsiliejimas, kontrastas ir miniatiūros šviesumas. Kvartilių populiarumo klasių žymėjimo požymių rinkinys klasifikavimui naudotas be vaizdo įrašų amžiaus ir vidutinių peržiūrų skaičiaus.

Kadangi požymių rinkiniai ir populiarumo klasių imties dydžiai, naudoti populiarumo grupių žymėjimo klasifikatoriuose skiriasi, lyginti šių klasifikatorių bendrą klasifikavimo tikslumą kiekvieno klasifikatoriaus lygmenyje būtų neteisingas. Todėl lyginamas tik skirtingas klasifikatorių bendras klasifikavimo tikslumas skirtinguose populiarumo klasių žymėjimo algoritmuose (žr. 16 lent.). Atsižvelgiant į tai, kad $k$-vidurkių ir SOM populiarumo klasių žymėjimo duomenys yra nesubalansuoti, prieduose pateiktas ir atliktų klasifikatorių - SVM, RF ir MLR kiekvienos klasės jautrumas ir tikslumas (lentelės 30, 31 ir 32 atitinkamai)

| | **Klasifikatorius** | | |
|---|---|---|---|
| **Populiarumo grupių žymėjimas** | Atraminių vektorių klasifikatorius | Atsitiktinio miško klasifikatorius | Daugialypės logistinės regresijos klasifikatorius |
| $k$-vidurkių populiarumo grupės | 1 | 0,97 | 0,99 |
| SOM populiarumo grupės | 0,99 | 0,96 | 0,97 |
| Kvartilių populiarumo grupės | 0,68 | 0,87 | 0,83 |

16 lentelė: Klasifikavimo rezultatai naudojant skirtingą populiarumo grupių žymėjimą

Iš 16 lentelės matoma, kad klasifikatorių bendro klasifikatorių tikslumo $k$-vidurkių ir SOM grupių žymėjimuose tendencija yra panaši: geriausi rezultatai $k$-vidurkių ir SOM algoritmais suskirstytiems duomenis pasiekti taikant atraminių vektoriaus klasifikatorių, o prastesni taikant RF klasifikatorių. Kita vertus skirtumas tarp klasifikatorių $k$-vidurkių ir SOM populiarumo klasių žymėjimuose yra labai mažas. Kvartilių populiarumo klasių žymėjime skirtumai tarp bendro klasifikavimo tikslumo ir skirtingų klasifikatorių yra didesni. Tačiau apibendrinant šiuos rezultatus, galima teigti, jog trijų skirtingų klasifikatorių pritaikymas duomenims yra gana aukštas pagal kiekvieną populiarumo klasių žymėjimo metodą.

# 5 Išvados bei rekomendacijos

1. Darbe atliktas vaizdo įrašų populiarumo tyrimas paremtas YouTube duomenų programos programavimo sąsajos pagalba surinktais vaizdo įrašų metaduomenimis ir išvestiniais jų parametrais:

   - Pirminė duomenų analizė - išsiaiškinti pagrindiniai skirtumai tarp kvartilių statistiniu būdu sudarytų populiarumo grupių.
   - Klasterinė duomenų analizė - įvertintos klasių susidarymo tendencijos ir ryšys su vaizdo įrašų populiarumu.
   - Įgyvendinti klasifikavimo algoritmai - populiarumo grupių priskyrimas turint fiksuotą požymių rinkinį.

2. Atlikus dispersinę analizę gauti tokie rezultatai:

   - įvertintas jungtinių kiekybinių požymių vidurkių reikšmingas skirtumas tarp populiarumo grupių naudojant MANOVA Vilkso kriterijumi paremtą statistinį testą ($F(36, 257268) = 3638{,}6$; $\Lambda = 0{,}3$; p – reikšmė $= 2{,}2 \cdot 10^{-16}$);
   - egzistuoja statistiškai reikšmingi skirtumai visuose kintamuosiuose tarp vaizdo įrašų populiarumo grupių (p-reikšmės $< 0{,}05$), naudojant ANOVA Velšo kriterijų;
   - nustatyti reikšmingi vidurkių skirtumai daugumoje požymių tarp kiekvienos populiarumo grupės porų naudojant Geimso-Hauvelio kriterijų;
   - statistinių testų rezultatai leidžią daryti išvadą, jog kvartilių metodu sudarytos populiarumo klasės yra tinkamas būdas populiarumo grupių vertinimui.

3. Pritaikius du grupavimo algoritmus $k$-vidurkių ir SOM populiarumo grupių sudarymui bei palyginus visus tris populiarumo grupių sudarymo metodus (kvartilių, $k$-vidurkių ir SOM), gauti skirtingi populiarumo grupių susidarymo rezultatai, tačiau su panašiomis charakteristikomis:

   - populiaresniuose vaizdo įrašuose auditorijos įsitraukimas yra didesnis;
   - mažesnė dalis vaizdo įrašų neturi aprašymo;
   - nepopuliarūs vaizdo įrašai turi mažiau tekstinės informacijos apie vaizdo įrašą bei prastesni miniatiūros kokybės rodikliai.

4. Panaudojus tris mašininio mokymosi algoritmus vaizdo įrašų klasifikavimui į populiarumo grupes pagal kvartilius ir įvertinus jų tikslumą pagal bendro klasifikavimo tikslumo metriką, klasifikatorius galima išrikiuoti tokia tvarka:

   - Atsitiktinio miško klasifikatorius (bendras klasifikavimo tikslumas = 0,87).
   - Daugialypės logistinės regresijos klasifikatorius (bendras klasifikavimo tikslumas = 0,83).
   - Atraminių vektorių klasifikatorius (bendras klasifikavimo tikslumas = 0,68).

5. Klasifikuojat vaizdo įrašus į populiarumo grupes sudarytas kvartilių metodu, svarbiausi kintamieji yra šie (išdėstyti pagal svarbą tiek RF, tiek MLR klasifikatoriams):

   - vidutinis "patinka" paspaudimų skaičius per dieną;
   - vidutinis "nepatinka" paspaudimų skaičius per dieną.

6. Atlikus $k$-vidurkių ir SOM metodais sudarytų grupių vaizdo įrašų klasifikavimą, susidurta su nesubalansuoto duomenų rinkinio problema - tokiu atveju naudingiau nagrinėti atskirų grupių jautrumą ir tikslumą, kadangi šios metrikos yra informatyvesnės už bendrą klasifikavimo tikslumą.

7. Palyginus visų trijų populiarumo grupių suskirstymo algoritmus, gauti skirtingi klasifikavimo rezultatai, todėl populiarumo klasių žymėjimo algoritmo pasirinkimas populiarumo vertinime priklauso nuo eksperto analizės tyrimo ir tikslų. Kita vertus, ateityje reikia ištirti, kuris populiarumo grupių priskyrimo metodas tinkamiausiai apibrėžia vaizdo įrašų populiarumą.

8. Ateities darbuose būtų prasminga išplėsti požymių rinkinį, panaudojant natūralios kalbos apdorojimo algoritmus, įvertinus sentimentų analizę komentaruose bei raktinių žodžių su vaizdo įrašų pavadinimu ryšį. Kadangi klasterizavimo algoritmuose naudojamas Euklidinio atstumo panašumo matas, verta išbandyti kitus panašumo matus, pavyzdžiui, panaudoti Manheteno atstumą ir palyginti rezultatus grupių susidaryme. Klasifikavimo analizėje prasminga išplėsti klasifikatorių aibę, panaudojant rekurentinius neuroninius tinklus ir palyginti rezultatus su atliktais klasifikatorių algoritmais.

# 6 Literatūra


[1] J. Bernatavičienė. *Vizualios žinių gavybos metodologija ir jos tyrimas.* disertacija, Vilniaus Gedimino Technikos Universitetas, 2008.

[2] Y.-L. Chen and C.-L. Chang. Early prediction of the future popularity of uploaded videos. *Expert Systems with Applications*, 133:59–74, 2019.

[3] R. Debnath, N. Takahide, and H. Takahashi. A decision based one-against-one method for multi-class support vector machine. *Pattern Anal. Appl.*, 7:164–175, 07 2004.

[4] G. Dzemyda, O. Kurasova, and J. Žilinskas. *Daugiamačių duomenų vizualizavimo metodai.* Mokslo aidai, 2008.

[5] A. El-Habil. An application on multinomial logistic regression model. *Pakistan Journal of Statistics and Operation Research*, 8, 03 2012.

[6] J. Gastwirth, Y. Gel, and W. Miao. The impact of levene's test of equality of variances on statistical theory and practice. *Statistical Science*, 24, 10 2010.

[7] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[8] W. Hoiles, A. Aprem, and V. Krishnamurthy. Engagement dynamics and sensitivity analysis of youtube videos, 2016.

[9] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.

[10] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis.* Applied Multivariate Statistical Analysis. Pearson Prentice Hall, 2007.

[11] B. Jolita, D. Gintautas, K. Olga, and Žilinskas Julius. Konferencijos "Lietuvos magistrantų informatikos ir IT tyrimai" darbai. *Vilnius University Open Series*, pages 1–119, geg. 2021. doi: `10.15388/LMITT.2021`.

[12] G. S. Kalra, R. S. Kathuria, and A. Kumar. Youtube video classification based on title and description text. In *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 74–79, 2019.

[13] T. Kodinariya and P. Makwana. Review on determining of cluster in k-means clustering. *International Journal of Advance Research in Computer Science and Management Studies*, 1:90–95, 01 2013.

[14] T. Kohonen, M. R. Schroeder, and T. S. Huang. *Self-Organizing Maps.* Springer-Verlag, Berlin, Heidelberg, 3rd edition, 2001.

[15] M. Kuhn and K. Johnson. *Applied predictive modeling.* Springer, New York, NY, 2013.

[16] Y.-T. Lin, C.-C. Yen, and J.-S. Wang. Video popularity prediction: An autoencoder approach with clustering. *IEEE Access*, PP:1–1, 07 2020.

[17] X. Liu. Chapter 3 - linear mixed-effects models. In X. Liu, editor, *Methods and Applications of Longitudinal Data Analysis*, pages 61–94. Academic Press, Oxford, 2016.

[18] G. Louppe. *Understanding Random Forests: From Theory to Practice.* disertacija, University of Liège, 2014.

[19] M. Mendes and E. Akkartal. Comparison of anova f and welch tests with their respective permutation versions in terms of type i error rates and test power. *Kafkas Universitesi Veteriner Fakultesi Dergisi*, 16:711–716, 01 2010.

[20] J. Minelga. *Skaitinio intelekto metodai neinvazinei gerklų ligų diagnostikai.* disertacija, Kauno Technologijos Universitetas, Vilniaus Gedimino Technikos Universitetas, 2018.

[21] A. Mittal, A. Moorthy, and A. Bovik. No-reference image quality assessment in the spatial domain. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 21, 08 2012.

[22] J. L. Pech Pacheco, G. Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia. Diatom autofocusing in brightfield microscopy: A comparative study. In *Pattern Recognition, Proceedings. 15th International Conference on*, volume 3, pages 314–317 vol.3, 02 2000.

[23] H. Pinto, J. M. Almeida, and M. A. Gonçalves. Using early view patterns to predict the popularity of youtube videos. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, page 365–374, New York, NY, USA, 2013. Association for Computing Machinery.

[24] G. Ravali, P. Moulika, A. parna, J.Abhinaya, and T.Anuradha. Analysing youtubedata using k-means clustering. *nternational Journal of Emerging Trends & Technology in Computer Science*, 6, 04 2017.

[25] M. C. Shingala and A. Rajyaguru. Comparison of post hoc tests for unequal variance. *International Journal of New Technologies in Science and Engineering*, 2, 11 2015.

[26] P. Stefanovič and O. Kurasova. Tekstinių dokumentų panašumų paieška naudojant saviorganizuojančius neuroninius tinklus ir k vidurkių metodą. *Informacijos mokslai*, 65:24–33, Jan. 2013.

[27] J. Tichonov. *Klasifikavimo metodais grindžiami skaitmeninių vaizdų glaudinimo sprendimai.* disertacija, Vilniaus Universitetas, 2018.

[28] T. Trzcinski and P. Rokita. Predicting popularity of online videos using support vector regression. *IEEE Transactions on Multimedia*, 19(11):2561–2570, Nov 2017.

[29] Youtube. Youtube for press. `https://blog.youtube/press/`.

# 7 Priedai

| Reikšmė | Kategorija |
|---|---|
| 1 | Filmai ir animacija |
| 2 | Transporto priemonės |
| 3 | Muzika |
| 4 | Gyvūnai |
| 5 | Sportas |
| 6 | Kelionės |
| 7 | Žaidimai |
| 8 | Žmonės |
| 9 | Komedijos |
| 10 | Pramogos |
| 11 | Naujienos ir politika |
| 12 | Kaip ir Stilius |
| 13 | Išsilavinimas |
| 14 | Mokslas ir technologijos |
| 15 | Ne pelno ir organizacijos ir aktyvizmas |

17 lentelė: Kategorija

| Kintamasis | Vidurkis | Dispersija | Standartinis nuokrypis | Minimali reikšmė | Maksimali reikšmė |
|---|---|---|---|---|---|
| channel_subscriber_count | 1839828 | $3{,}62 \cdot 10^{13}$ | 1564206 | 0 | 51500000 |
| age | 1499 | 1512014 | 1229,64 | 1 | 5406 |
| avg_daily_views | 13,28 | 182,62 | 13,51 | 0 | 48,6 |
| avg_daily_likes | 0,63 | 962,78 | 31,03 | 0 | 5600,6 |
| avg_daily_dislikes | 0,03 | 0,083 | 0,29 | 0 | 31 |
| avg_daily_comments | 0,1 | 24,08 | 4,9 | 0 | 888,6 |
| chr_title | 48,64 | 485,2 | 22,03 | 1 | 100 |
| keywords_count | 28,65 | 580,3 | 24,09 | 0 | 110 |
| thumbnail_quality | 28,28 | 244,67 | 15,64 | 0,007 | 99,92 |
| thumbnail_blurriness | 3107 | 7621454 | 2760,7 | 0 | 36697 |
| thumbnail_contrast | 63,16 | 238,03 | 15,43 | 0 | 124,63 |
| thumbnail_brightness | 131 | 1539,84 | 39,24 | 0 | 255 |

18 lentelė: Pirmosios grupės kiekybinių kintamųjų aprašomoji statistika

| Kintamasis | Q1 | Q2 | Q3 |
|---|---|---|---|
| channel_subscriber_count | 35200 | 290000 | 1110000 |
| age | 433 | 1244 | 2229 |
| avg_daily_views | 1,98 | 8,06 | 21,7 |
| avg_daily_likes | 0,03 | 0,16 | 0,52 |
| avg_daily_dislikes | 0,0007 | 0,0048 | 0,0161 |
| avg_daily_comments | 0,0014 | 0,0153 | 0,0588 |
| chr_title | 32 | 46 | 62 |
| keywords_count | 8 | 23 | 47 |
| thumbnail_quality | 17,74 | 25,38 | 35,31 |
| thumbnail_blurriness | 1032,53 | 2347,43 | 4405,4 |
| thumbnail_contrast | 53,47 | 63,44 | 73,11 |
| thumbnail_brightness | 105,89 | 130,46 | 154,92 |

19 lentelė: Pirmosios populiarumo grupės kvartiliai

| Kintamasis | Vidurkis | Dispersija | Standartinis nuokrypis | Minimali reikšmė | Maksimali reikšmė |
|---|---|---|---|---|---|
| channel_subscriber_count | 3429504 | $3{,}89{\cdot}10^{13}$ | 6235151 | 5 | 54800000 |
| age | 1250 | 927265,3 | 962,95 | 1 | 5569 |
| avg_daily_views | 203,83 | 15674,51 | 125,2 | 48,60 | 494,40 |
| avg_daily_likes | 4,16 | 48,28 | 6,95 | 0 | 93,09 |
| avg_daily_dislikes | 0,09 | 0,75 | 0,87 | 0 | 46,44 |
| avg_daily_comments | 0,68 | 2,53 | 1,59 | 0 | 96,2 |
| chr_title | 48,88 | 436,21 | 20,89 | 1 | 100 |
| keywords_count | 37,4 | 546,40 | 23,38 | 0 | 112 |
| thumbnail_quality | 25,95 | 211,85 | 14,56 | 0,0028 | 99,82 |
| thumbnail_blurriness | 3845 | 8593188 | 2931,41 | 0 | 42075 |
| thumbnail_contrast | 64,93 | 195,59 | 13,99 | 0 | 122,38 |
| thumbnail_brightness | 137,8 | 1191,361 | 34,52 | 0 | 254,1 |

20 lentelė: Antrosios grupės kiekybinių kintamųjų aprašomoji statistika

| Kintamasis | Q1 | Q2 | Q3 |
|---|---|---|---|
| channel_subscriber_count | 431000 | 1210000 | 3580000 |
| age | 458 | 1079 | 1855 |
| avg_daily_views | 95,11 | 170,72 | 296,4 |
| avg_daily_likes | 1,87 | 4,16 | 8,5 |
| avg_daily_dislikes | 0,044 | 0,095 | 0,2 |
| avg_daily_comments | 0,1242 | 0,32 | 0,7424 |
| chr_title | 33 | 46 | 62 |
| keywords_count | 18 | 35 | 59 |
| thumbnail_quality | 16,33 | 23,37 | 32,23 |
| thumbnail_blurriness | 1667,28 | 3207,63 | 5292,56 |
| thumbnail_contrast | 56,165 | 65,16 | 73,92 |
| thumbnail_brightness | 116,87 | 137,77 | 159,04 |

21 lentelė: Antrosios populiarumo grupės kvartiliai

| Kintamasis | Vidurkis | Dispersija | Standartinis nuokrypis | Minimali reikšmė | Maksimali reikšmė |
|---|---|---|---|---|---|
| channel_subscriber_count | 5827060 | $6{,}35{\cdot}10^{13}$ | 7969533 | 24 | 54800000 |
| age | 976,7 | 726432,7 | 852,31 | 1 | 5380,0 |
| avg_daily_views | 1451,3 | 561249,6 | 749,17 | 494,4 | 3209,4 |
| avg_daily_likes | 32,23 | 1898,74 | 43,57 | 0 | 803,62 |
| avg_daily_dislikes | 0,73 | 31,35 | 5,6 | 0 | 491,5 |
| avg_daily_comments | 2,15 | 107,92 | 10,39 | 0 | 579,5 |
| chr_title | 49,44 | 425,37 | 20,62 | 2 | 100 |
| keywords_count | 40,64 | 563,56 | 23,74 | 0 | 115 |
| thumbnail_quality | 25,83 | 204,48 | 14,3 | 0,008 | 99,9 |
| thumbnail_blurriness | 4185 | 8958352 | 2993,05 | 0 | 38989 |
| thumbnail_contrast | 65,22 | 180,8 | 13,45 | 0 | 124,22 |
| thumbnail_brightness | 142,4 | 1182,1 | 34,38 | 0 | 253,4 |

22 lentelė: Trečiosios grupės kiekybinių kintamųjų aprašomoji statistika

| Kintamasis | Q1 | Q2 | Q3 |
|---|---|---|---|
| channel_subscriber_count | 870000 | 2950000 | 7080000 |
| age | 277 | 766 | 1462 |
| avg_daily_views | 801,61 | 1272,03 | 1989,25 |
| avg_daily_likes | 16,54 | 32,23 | 59,19 |
| avg_daily_dislikes | 0,38 | 0,73 | 1,4 |
| avg_daily_comments | 0,89 | 2,15 | 4,41 |
| chr_title | 34 | 47 | 62 |
| keywords_count | 21 | 41 | 62 |
| thumbnail_quality | 16,30 | 23,2 | 31,95 |
| thumbnail_blurriness | 2022,73 | 3585,86 | 5651,45 |
| thumbnail_contrast | 56,852 | 65,62 | 74,01 |
| thumbnail_brightness | 121,454 | 142,22 | 163,61 |

23 lentelė: Trečiosios populiarumo grupės kvartiliai

| Kintamasis | Vidurkis | Dispersija | Standartinis nuokrypis | Minimali reikšmė | Maksimali reikšmė |
|---|---|---|---|---|---|
| channel_subscriber_count | 11422580 | $1{,}52{\cdot}10^{14}$ | 12344365 | 24 | 54800000 |
| age | 629,6 | 476564,8 | 690,34 | 1 | 5064,0 |
| avg_daily_views | 33619 | 9287706901 | 96372,75 | 3209 | 207945,46 |
| avg_daily_likes | 678,48 | 10376957 | 3221,33 | 0 | 24109,06 |
| avg_daily_dislikes | 42,77 | 59021,26 | 242,94 | 0 | 207945,46 |
| avg_daily_comments | 43,53 | 79109,5 | 281,26 | 0 | 20660,38 |
| chr_title | 54,14 | 502,85 | 22,42 | 3 | 100 |
| keywords_count | 41,75 | 591,88 | 24,33 | 0 | 114 |
| thumbnail_quality | 24,56 | 176,8603 | 13,3 | 0,01 | 99,96 |
| thumbnail_blurriness | 4191,83 | 8086903 | 2843,75 | 5,14 | 35440,17 |
| thumbnail_contrast | 64,254 | 159,77 | 12,64 | 7,71 | 124,72 |
| thumbnail_brightness | 145,69 | 1077,623 | 32,83 | 14,14 | 251,74 |

24 lentelė: Ketvirtosios grupės kiekybinių kintamųjų aprašomoji statistika

| Kintamasis | Q1 | Q2 | Q3 |
|---|---|---|---|
| channel_subscriber_count | 1910000 | 7080000 | 17700000 |
| age | 101 | 391 | 936 |
| avg_daily_views | 5533,91 | 10580,48 | 25626,54 |
| avg_daily_likes | 85,51 | 208,92 | 491,36 |
| avg_daily_dislikes | 3,19 | 7,63 | 23,03 |
| avg_daily_comments | 1,62 | 9,14 | 26,7 |
| chr_title | 37 | 51 | 70 |
| keywords_count | 21 | 44 | 64 |
| thumbnail_quality | 15,9 | 22,22 | 30,18 |
| thumbnail_blurriness | 2137,82 | 3609,6 | 5597,18 |
| thumbnail_contrast | 56,06 | 64,23 | 72,53 |
| thumbnail_brightness | 126,2 | 147,4 | 166,84 |

25 lentelė: Ketvirtosios populiarumo grupės kvartiliai

| Kintamasis | Grupė | | p - reikšmė |
|---|---|---|---|
| avg_daily_views | 1 | 2 | 0 |
| avg_daily_views | 1 | 3 | 0 |
| avg_daily_views | 1 | 4 | 0 |
| avg_daily_views | 2 | 3 | 0 |
| avg_daily_views | 2 | 4 | 0 |
| avg_daily_views | 3 | 4 | 0 |
| avg_daily_comments | 1 | 2 | 0 |
| avg_daily_comments | 1 | 3 | 0 |
| avg_daily_comments | 1 | 4 | 0 |
| avg_daily_comments | 2 | 3 | 0 |
| avg_daily_comments | 2 | 4 | 0 |
| avg_daily_comments | 3 | 4 | 0 |
| avg_daily_dislikes | 1 | 2 | 0 |
| avg_daily_dislikes | 1 | 3 | 0 |
| avg_daily_dislikes | 1 | 4 | 0 |
| avg_daily_dislikes | 2 | 3 | 0 |
| avg_daily_dislikes | 2 | 4 | 0 |
| avg_daily_dislikes | 3 | 4 | 0 |
| avg_daily_likes | 1 | 2 | 0 |
| avg_daily_likes | 1 | 3 | 0 |
| avg_daily_likes | 1 | 4 | 0 |
| avg_daily_likes | 2 | 3 | 0 |
| avg_daily_likes | 2 | 4 | 0 |
| avg_daily_likes | 3 | 4 | 0 |
| channel_subscriber_count | 1 | 2 | 0 |
| channel_subscriber_count | 1 | 3 | 0 |
| channel_subscriber_count | 1 | 4 | 0 |
| channel_subscriber_count | 2 | 3 | 0 |
| channel_subscriber_count | 2 | 4 | 0 |
| channel_subscriber_count | 3 | 4 | 0 |
| chr_title | 1 | 2 | 8,32e-01 |
| chr_title | 1 | 3 | 8,80e-02 |
| chr_title | 1 | 4 | 0 |
| chr_title | 2 | 3 | 4,00e-03 |
| chr_title | 2 | 4 | 0 |

| Kintamasis | Grupė | | p - reikšmė |
|---|---|---|---|
| chr_title | 3 | 4 | 0 |
| keywords_count | 1 | 2 | 0 |
| keywords_count | 1 | 3 | 0 |
| keywords_count | 1 | 4 | 0 |
| keywords_count | 2 | 3 | 0 |
| keywords_count | 2 | 4 | 0 |
| keywords_count | 3 | 4 | 0 |
| thumbnail_quality | 1 | 2 | 0 |
| thumbnail_quality | 1 | 3 | 0 |
| thumbnail_quality | 1 | 4 | 0 |
| thumbnail_quality | 2 | 3 | 0,2 |
| thumbnail_quality | 2 | 4 | 0 |
| thumbnail_quality | 3 | 4 | 0 |
| thumbnail_blurriness | 1 | 2 | 0 |
| thumbnail_blurriness | 1 | 3 | 0 |
| thumbnail_blurriness | 1 | 4 | 0 |
| thumbnail_blurriness | 2 | 3 | 0 |
| thumbnail_blurriness | 2 | 4 | 0 |
| thumbnail_blurriness | 3 | 4 | 0,03 |
| thumbnail_brightness | 1 | 2 | 0 |
| thumbnail_brightness | 1 | 3 | 0 |
| thumbnail_brightness | 1 | 4 | 0 |
| thumbnail_brightness | 2 | 3 | 0 |
| thumbnail_brightness | 2 | 4 | 0 |
| thumbnail_brightness | 3 | 4 | 0 |
| thumbnail_constrast | 1 | 2 | 0 |
| thumbnail_constrast | 1 | 3 | 0 |
| thumbnail_constrast | 1 | 4 | 0 |
| thumbnail_constrast | 2 | 3 | 0,35 |
| thumbnail_constrast | 2 | 4 | 0 |
| thumbnail_constrast | 3 | 4 | 0 |
| age | 1 | 2 | 0 |
| age | 1 | 3 | 0 |
| age | 1 | 4 | 0 |
| age | 2 | 3 | 0 |
| age | 2 | 4 | 0 |
| age | 3 | 4 | 0 |

26 lentelė: Geimso-Hauvelio testo rezultatai

| Kintamasis | Koeficientas | Voldo kriterijus | p-reikšmė |
|---|---|---|---|
| Laisvasis narys | 17,24 | 81,06 | <0,05 |
| channel_subscriber_count | 0,56 | 27,04 | <0,05 |
| avg_daily_likes | 19,69 | 75,07 | <0,05 |
| avg_daily_dislikes | 4,5 | 28,94 | <0,05 |
| avg_daily_comments | 0,77 | 6,81 | <0,05 |
| chr_title | 0,17 | 10,26 | <0,05 |
| keywords_count | -0,0007 | -0,04 | 0,97 |
| thumbnail_quality | -0,03 | -1,86 | 0,06 |
| thumbnail_blurriness | -0,07 | -3,64 | <0,05 |
| thumbnail_contrast | -0,16 | -9,16 | <0,05 |
| thumbnail_brightness | 0,05 | 3,29 | <0,05 |
| description_exists1 | 0,43 | 3,64 | <0,05 |

27 lentelė: Antros populiarumo grupės daugialypės logistinės regresijos koeficientai

| Kintamasis | Koeficientas | Voldo kriterijus | p-reikšmė |
|---|---|---|---|
| Laisvasis narys | 20,8 | 80,57 | <0,05 |
| channel_subscriber_count | 1,22 | 45,62 | <0,05 |
| avg_daily_likes | 26,91 | 95,66 | <0,05 |
| avg_daily_dislikes | 8,42 | 49,23 | <0,05 |
| avg_daily_comments | -0,11 | -0,87 | 0,39 |
| chr_title | 0,3 | 12,94 | <0,05 |
| keywords_count | 0,05 | 1,923 | 0,05 |
| thumbnail_quality | 0,02 | 0,94 | 0,35 |
| thumbnail_blurriness | -0,15 | -6,27 | <0,05 |
| thumbnail_contrast | -0,3 | -12,42 | <0,05 |
| thumbnail_brightness | 0,05 | 2,39 | 0,02 |
| description_exists1 | 0,66 | 3,56 | <0,05 |

28 lentelė: Trečios populiarumo grupės daugialypės logistinės regresijos koeficientai

| Kintamasis | Koeficientas | Voldo kriterijus | p-reikšmė |
|---|---|---|---|
| Laisvasis narys | 18,84 | 65,1 | <0,05 |
| channel_subscriber_count | 1,79 | 57,32 | <0,05 |
| avg_daily_likes | 29,48 | 103,42 | <0,05 |
| avg_daily_dislikes | 10,89 | 62,18 | <0,05 |
| avg_daily_comments | -0,68 | -5,28 | <0,05 |
| chr_title | 0,48 | 16,69 | <0,05 |
| keywords_count | 0,13 | 4,32 | <0,05 |
| thumbnail_quality | 0,08 | 2,98 | 0,003 |
| thumbnail_blurriness | -0,19 | -6,09 | <0,05 |
| thumbnail_contrast | -0,45 | -15,1 | <0,05 |
| thumbnail_brightness | 0,03 | 0,99 | 0,32 |
| description_exists1 | 0,74 | 3,27 | <0,05 |

29 lentelė: Ketvirtos populiarumo grupės daugialypės logistinės regresijos koeficientai

| | $k$-vidurkių SVM klasifikatorius | | SOM SVM klasifikatorius | | Kvartilių SVM klasifikatorius | |
|---|---|---|---|---|---|---|
| **Klasė** | Jautrumas | Tikslumas | Jautrumas | Tikslumas | Jautrumas | Tikslumas |
| Nepopuliarūs | 1 | 1 | 1 | 0,99 | 0,71 | 0,63 |
| Vidutinio populiarumo | 1 | 1 | 0,99 | 0,99 | 0,57 | 0,54 |
| Populiarūs | 0,80 | 0,96 | 0,99 | 0,98 | 0,62 | 0,68 |
| Labai populiarūs | 0,71 | 1 | 0,82 | 0,99 | 0,81 | 0,89 |
| Bendras klasifikavimo tikslumas | 1 | | 0,99 | | 0,68 | |

30 lentelė: SVM klasifikatoriaus rezultatai

| | $k$-vidurkių RF klasifikatorius | | SOM RF klasifikatorius | | Kvartilių RF klasifikatorius | |
|---|---|---|---|---|---|---|
| **Klasė** | Jautrumas | Tikslumas | Jautrumas | Tikslumas | Jautrumas | Tikslumas |
| Nepopuliarūs | 0,96 | 0,97 | 0,97 | 0,96 | 0,93 | 0,91 |
| Vidutinio populiarumo | 0,97 | 0,97 | 0,96 | 0,96 | 0,83 | 0,83 |
| Populiarūs | 0,76 | 0,88 | 0,94 | 0,96 | 0,82 | 0,82 |
| Labai populiarūs | 0 | 0 | 0,94 | 0,98 | 0,90 | 0,92 |
| Bendras klasifikavimo tikslumas | 0,97 | | 0,96 | | 0,87 | |

31 lentelė: RF klasifikatoriaus rezultatai

| | $k$-vidurkių MLR klasifikatorius | | SOM MLR klasifikatorius | | Kvartilių MLR klasifikatorius | |
|---|---|---|---|---|---|---|
| **Klasė** | Jautrumas | Tikslumas | Jautrumas | Tikslumas | Jautrumas | Tikslumas |
| Nepopuliarūs | 0,99 | 0,99 | 0,98 | 0,98 | 0,91 | 0,86 |
| Vidutinio populiarumo | 0,98 | 0,99 | 0,39 | 0,41 | 0,77 | 0,77 |
| Populiarūs | 0,1 | 0,24 | 0,97 | 0,97 | 0,77 | 0,77 |
| Labai populiarūs | 0 | 0 | 0,94 | 0,94 | 0,86 | 0,91 |
| Bendras klasifikavimo tikslumas | 0,99 | | 0,97 | | 0,83 | |

32 lentelė: MLR klasifikatoriaus rezultatai

## 7.1 R programos kodas

```
# Script for Data
library(tuber)
library(jsonlite)
library(tidyverse)

yt_oauth(
  app_id ="",
  app_secret = "",
  token = "")

C_MAIN_JSON_FILE_NAME = "~data.json"
```

```
data_df <- read_json(
  path = C_MAIN_JSON_FILE_NAME,
  simplifyVector = T)

data_df <- data.frame()

isVideoWritten = F

for(channelId in youtube_channels){
  print(paste("channelId: ", channelId))
  if(!channelId %in% data_df$channel_id){
  channel_stats <- get_channel_stats(channel_id = channelId)
  if(as.numeric(channel_stats[["statistics"]]$videoCount) <= 600){
    channel_video_stats <- get_all_channel_video_stats(channel_id = channelId)
    channel_video_stats <- as.data.frame(channel_video_stats, stringsAsFactors = F)
    for(video_idx in 1:nrow(channel_video_stats)){
      videoId = as.character(channel_video_stats[video_idx, "id"])
      print(paste("  videoId: ", videoId))
      if(!videoId %in% data_df$video_id){
      video_stats <- get_video_details(video_id = videoId)
      tags_concat = NA
      tags = video_stats$items[[1]]$snippet$tags

      if(!is.null(tags)){
        tags_concat = paste(data.frame(matrix(unlist(tags),
        nrow=length(tags), byrow=T))[,1],
        collapse = "; ")
      }

      video_likes = channel_video_stats[video_idx, "likeCount"]
      if(!is.null(video_likes) && !is.na(video_likes)){
        video_likes = as.numeric(video_likes)
      } else {
        video_likes = NA
      }

      commentCount = channel_video_stats[video_idx, "commentCount"]
      if(!is.null(commentCount) && !is.na(commentCount)){
        commentCount = as.numeric(commentCount)
      } else {
        commentCount = NA
      }

      video_dislikes = channel_video_stats[video_idx, "dislikeCount"]
      if(!is.null(video_dislikes) && !is.na(video_dislikes)){
        video_dislikes = as.numeric(video_dislikes)
      } else {
        video_dislikes = NA
      }

      if("subscriberCount" %in% names(channel_stats$statistics)){
        channel_subscriber_count =
```

```
      as.numeric(channel_stats[["statistics"]]$subscriberCount)
      if(!is.null(channel_subscriber_count) && !is.na(channel_subscriber_count)){
        channel_subscriber_count = as.numeric(channel_subscriber_count)
      }
    } else{
      channel_subscriber_count = NA
    }

    as.Date(channel_video_stats[video_idx, "publication_date"], "%Y-%m-%d")

    data_temp_df <- data.frame(
      video_id = as.character(channel_video_stats[video_idx,"id"]),
      video_title = as.character(channel_video_stats[video_idx, "title"]),
      video_publication_date = as.Date(channel_video_stats[video_idx,
      "publication_date"],
                                       format = "%Y-%m-%d"),
      video_description = as.character(channel_video_stats[video_idx, "description"]),
      video_likes = video_likes,
      video_dislikes = video_dislikes,
      video_comments = commentCount,
      video_views = as.numeric(channel_video_stats[video_idx, "viewCount"]),
      video_keywords = as.character(tags_concat),
      video_category  = as.numeric(video_stats[["items"]][[1]][["snippet"]]
      [["categoryId"]]),
      video_thumbnail = paste("https://i.ytimg.com/vi/",
                              videoId,
                              "/mqdefault.jpg",
                              sep = ""),
      channel_id = as.character(channel_stats$id),
      channel_title = as.character(channel_stats[["snippet"]]$title),
      channel_subscriber_count = channel_subscriber_count,
      channel_videos_count = as.numeric(channel_stats[["statistics"]]$videoCount),
      channel_video_view_count = as.numeric(channel_stats[["statistics"]]$viewCount),
      extract_date = Sys.Date(),
      stringsAsFactors = F)

    isVideoWritten = T
    data_df <- rbind(data_df, data_temp_df)
    print(paste("    - added."))
  }
} else {
  list_channel_video <- list_channel_videos(channel_id = channelId, max_results = 60)
  list_channel_video <- list_channel_video[1:500,]
  for(video_idx in 1:nrow(list_channel_video)){
    videoId = as.character(list_channel_video[video_idx, "contentDetails.videoId"])
    print(paste("videoId: ", videoId))

    if(!videoId %in% data_df$video_id){
    video_stats <- get_video_details(video_id = videoId)
    channel_video_stats <- get_stats(video_id = videoId)
    channel_video_stats <- as.data.frame(channel_video_stats, stringsAsFactors = F)
    tags_concat = NA
    tags = video_stats$items[[1]]$snippet$tags
```

```
if(!is.null(tags)){
  tags_concat = paste(data.frame(matrix(unlist(tags), nrow=length(tags),
  byrow=T))[,1],
  collapse = "; ")
}

if("commentCount" %in% colnames(channel_video_stats)){
  commentCount = channel_video_stats[["commentCount"]]
  if(!is.null(commentCount) && !is.na(commentCount)){
    commentCount = as.numeric(commentCount)
  }
} else{
  commentCount = NA
}

if("subscriberCount" %in% names(channel_stats$statistics)){
  channel_subscriber_count =
  as.numeric(channel_stats[["statistics"]]$subscriberCount)
  if(!is.null(channel_subscriber_count) && !is.na(channel_subscriber_count)){
    channel_subscriber_count = as.numeric(channel_subscriber_count)
  }
} else{
  channel_subscriber_count = NA
}

if("likeCount" %in% names(channel_video_stats)){
  video_likes  = channel_video_stats[["likeCount"]]
  if(!is.null(video_likes) && !is.na(video_likes)){
    video_likes = as.numeric(video_likes)
  }
} else{
  video_likes = NA
}

if("dislikeCount" %in% names(channel_video_stats)){
  video_dislikes = channel_video_stats[["dislikeCount"]]
  if(!is.null(video_dislikes) && !is.na(video_dislikes)){
    video_dislikes = as.numeric(video_dislikes)
  }
} else{
  video_dislikes = NA
}

data_temp_df1 <- data.frame(
  video_id = video_stats[["items"]][[1]][["id"]],
  video_title =  video_stats[["items"]][[1]][["snippet"]][["title"]],
  video_publication_date =
  video_stats[["items"]][[1]][["snippet"]][["publishedAt"]],
  video_description = video_stats[["items"]][[1]][["snippet"]][["description"]],
  video_likes = video_likes,
  video_dislikes = video_dislikes,
  video_comments = commentCount,
```

```
        video_views = channel_video_stats[["viewCount"]],
        video_keywords = as.character(tags_concat),
        video_category  =
        as.numeric(video_stats[["items"]][[1]][["snippet"]][["categoryId"]]),
        video_thumbnail = paste("https://i.ytimg.com/vi/",
        videoId,
        "/mqdefault.jpg",
        sep = ""),
        channel_id = as.character(channel_stats$id),
        channel_title = video_stats[["items"]][[1]][["snippet"]][["channelTitle"]],
        channel_subscriber_count = channel_subscriber_count,
        channel_videos_count = as.numeric(channel_stats[["statistics"]]$videoCount),
        channel_video_view_count = as.numeric(channel_stats[["statistics"]]$viewCount),
        extract_date = Sys.Date(),
        stringsAsFactors = F)

      isVideoWritten = T
      data_df <- rbind(data_df, data_temp_df1)
      print(paste("    - added."))
    }
  }
}

 } else {
   print("Tokio kanalo video jau yra duomenyse.")
 }

if (isVideoWritten){
  write_json(
    x = data_df,
    path = C_MAIN_JSON_FILE_NAME)
}

# Data cleaning, preparations, main statistics
data <- data %>%
mutate(popularity = case_when(
    (avg_daily_views <= quantile(avg_daily_views)[2]) ~ 1,
    (avg_daily_views > quantile(avg_daily_views)[2] &
    avg_daily_views <= quantile(avg_daily_views)[3]) ~ 2,
    (avg_daily_views > quantile(avg_daily_views)[3] &
    avg_daily_views <= quantile(avg_daily_views)[4]) ~ 3,
    TRUE ~ 4))

data$channel <- as.factor(data$channel_title)
datachannel <- factor(data$channel)
nlevels(data$channel)

nrow(data)

mean(data$age)

mean(data$video_views)
```

```
data_description <- data %>%
    group_by(popularity, description_exists) %>%
    count()

data_description <- data_description %>%
    group_by(description_exists) %>%
    mutate(N = sum(n))

data_description <- data_description %>%
    group_by(description_exists) %>%
    mutate(Procentas = (n*100)/N)

data_description <- data_description %>%
mutate(Popular =
case_when(popularity == 1 ~ "Nepopuliarūs",
          popularity == 2 ~ "Vidutinio populiarumo",
          popularity == 3 ~ "Populiarūs",
          popularity == 4 ~ "Labai populiarūs"))

data_description$Popular <- factor(data_description$Popular,
levels = c("Nepopuliarūs",
          "Vidutinio populiarumo",
          "Populiarūs",
          "Labai populiarūs"))

ggplot(data_description, aes(fill = description_exists, y = Procentas, x = Popular)) +
  geom_bar(position = "dodge", stat = "identity") +
  theme_publish() +
  scale_fill_grey(start = 0.3, end = 0.8) +
  labs(x = "Populiarumo lygis", fill = "Aprašymo egzistavimas", y = "Procentai, %") +
  scale_y_continuous(breaks = seq(0, 60, 10), limits = c(0, 60))

TC <- data_df %>%
group_by(popularity,video_category) %>%
summarise(TotalTrending = n()) %>%
arrange(popularity, desc(TotalTrending))
Totals <- TC %>% group_by(popularity) %>% summarise(totals = sum(TotalTrending))
TC <- merge(TC,Totals, by = "popularity")
TC <- TC %>% select(popularity, totals, video_category, TotalTrending)
TC <- TC %>% group_by(popularity, totals, video_category) %>%
summarise(Totals = sum(TotalTrending))
TC$Percentage <- (TC$Totals / TC$totals)*100

TC <- TC %>% mutate(Category_title =
case_when(video_category == 1 ~ "Filmai ir animacija",
        video_category == 2 ~ "Transporto priemonės",
        video_category == 3 ~ "Muzika",
        video_category == 4 ~ "Gyvūnai",
        video_category == 5 ~ "Sportas",
        video_category == 6 ~ "Kelionės",
        video_category == 7 ~ "Žaidimai",
        video_category == 8 ~ "Žmonės",
        video_category == 9 ~ "Komedija",
```

```
        video_category == 10 ~ "Pramogos",
        video_category == 11 ~ "Naujienos ir politika",
        video_category == 12~ "Kaip ir Stilius",
        video_category == 13 ~ "Išsilavinimas",
        video_category == 15 ~ "Ne pelno organizacijos ir aktyvizmas",
        TRUE ~ "Mokslas ir Technologijos"))

nb.cols <- 18
mycolors <- colorRampPalette(brewer.pal(8, "Set2"))(nb.cols)

ggplot(data = TC, aes(
  x = video_category,
  y = Percentage,
  fill = factor(Category_title))) +
  geom_bar(stat = "identity",
           position = "dodge",
           show.legend = TRUE) +
  scale_fill_manual(values = mycolors)  +
  ylab("Peržiūrų dalis, procentais") +
  xlab("Kategorijos") +
  theme(
    axis.text = element_text(size = 15),
    legend.text = element_text(size = 15),
    legend.title = element_text(size = 15))  +
  facet_wrap( ~ popularity, scales = "free", ncol = 2) +
  labs(fill = "Kategorija") +
  theme_publish()

dataf$video_keywords[is.na(dataf$video_keywords)] <- 0
data$keywords_count <- ifelse(data$video_keywords == 0, 0,
sapply(data_df$video_keywords, function(x) length(unlist(strsplit(as.character(x),
"\\W+")))))

# Statistical tests
library(jsonlite)
library(tidyverse)
library(reshape2)
library(mvnormtest)
library(rstatix)

data <- read_json(
  path = "~data.json",
  simplifyVector = T)

data_manova <- data %>% select(popularity,
                               thumbnail_quality,
                               channel_subscriber_count,
                               chr_title,
                               thumbnail_contrast,
                               thumbnail_brightness,
                               thumbnail_blurriness,
                               avg_daily_likes,
                               avg_daily_dislikes,
```

```
                                    avg_daily_comments,
                                    keywords_count,
                                    avg_daily_views,
                                    age,
                                    description_exists)

data_manova$popularity <- as.factor(data_manova$popularity)
data_manova$description_exists <- as.factor(data_manova$description_exists)

outlier_thumbnail_quality <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(thumbnail_quality)

outlier_channel_subscriber_count <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(channel_subscriber_count)

outlier_chr_title <-   data_manova %>%
  group_by(popularity) %>%
  identify_outliers(chr_title)

outlier_thumbnail_contrast <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(thumbnail_contrast)

outlier_thumbnail_brightness <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(video_thumbnail_brightness)

outlier_thumbnail_blurriness <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(thumbnail_blurriness)

outlier_avg_daily_likes <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(avg_daily_likes)

outlier_avg_daily_dislikes <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(avg_daily_dislikes)

outlier_avg_daily_comments <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(avg_daily_comments)

outlier_avg_daily_views <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(avg_daily_views)

outlier_age <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(age)
```

```
outlier_keywords_count <- data_manova %>%
  group_by(popularity) %>%
  identify_outliers(keywords_count)

outliers <- union_all(outlier_thumbnail_quality, outlier_channel_subscriber_count) %>%
            union_all(outlier_chr_title) %>%
            union_all(outlier_thumbnail_contrast) %>%
            union_all(outlier_thumbnail_brightness) %>%
            union_all(outlier_thumbnail_blurriness) %>%
            union_all(outlier_avg_daily_likes) %>%
            union_all(outlier_avg_daily_dislikes) %>%
            union_all(outlier_avg_daily_comments) %>%
            union_all(outlier_keywords_count) %>%
            union_all(outlier_avg_daily_views) %>%
            union_all(outlier_age)

outliers <- outliers %>% select(-c(is.outlier, is.extreme))

data_manova_outliers <- anti_join(data_manova, outliers)

outcome <- data_manova_outliers[,-c(1,14)]

outcome<- as.matrix(outcome)

# MANOVA
dataModel <- manova(outcome ~ popularity, data=data_manova_outliers)

summary(dataModel, test = "Wilks")

# Levene test
data_manova_outliers %>%
  gather(key = "variable", value = "value",
         thumbnail_quality,
         channel_subscriber_count,
         chr_title,
         thumbnail_contrast,
         thumbnail_brightness,
         thumbnail_blurriness,
         avg_daily_likes,
         avg_daily_dislikes,
         avg_daily_comments,
         keywords_count,
         age,
         avg_daily_views) %>%
  group_by(variable) %>%
  levene_test(value ~ popularity)

grouped.data <- data_manova_outliers %>%
  gather(key = "variable", value = "value",
         thumbnail_quality,
         channel_subscriber_count,
         chr_title,
         thumbnail_contrast,
```

```
          thumbnail_brightness,
          thumbnail_blurriness,
          avg_daily_likes,
          avg_daily_dislikes,
          avg_daily_comments,
          keywords_count,
          age,
          avg_daily_views) %>%
  group_by(variable)

# ANOVA Welch
grouped.data %>% welch_anova_test(value ~ popularity)

# Games-Howell
data_manova_outliers %>%
  gather(key = "variables", value = "value",
          thumbnail_quality,
          channel_subscriber_count,
          chr_title,
          thumbnail_contrast,
          thumbnail_brightness,
          thumbnail_blurriness,
          avg_daily_likes,
          avg_daily_views,
          avg_daily_dislikes,
          avg_daily_comments,
          keywords_count,
          age) %>%
  group_by(variables) %>%
  games_howell_test(value ~ popularity) %>%
  select(-estimate, -conf.low, -conf.high)

# MLR
library(caret)
library(dplyr)
library(moments)
library(ggplot2)
library(ggcorrplot)
library(e1071)
library(doParallel)
library(reshape2)
library(jsonlite)

data <- read_json(path = "~data.json",simplifyVector = T)

data <- data %>% select(
    popularity,
    thumbnail_quality,
    channel_subscriber_count,
    chr_title,
    thumbnail_contrast,
    thumbnail_brightness,
    thumbnail_blurriness,
```

```
    avg_daily_likes,
    avg_daily_dislikes,
    avg_daily_comments,
    keywords_count,
    description_exists)

data_scale <- data
data_scale[-c(1,12)] <- scale(data_scale[-c(1,12)])

stats <- as.data.frame(cbind(Mean = sapply(data, mean),
                             Median = sapply(data, median),
                             Std_Dev = sapply(data, sd),
                             CV = sapply(data, sd) / sapply(data, mean),
                             Skewness = sapply(data, skewness),
                             Kurtosis = sapply(data, kurtosis)))

stats

stats2 <- as.data.frame(cbind(Mean = sapply(data_scale[ , c(2:11)],
                              Median = sapply(data_scale[ , c(2:11)], median),
                              Std.Dev = sapply(data_scale[ , c(2:11)], sd),
                              Skewness = sapply(data_scale[ , c(2:11)], skewness),
                              Kurtosis = sapply(data_scale[ , c(2:11)], kurtosis)))

stats2

data_scale[ ,-c(1,12)] <- scale(sqrt(data[ ,-c(1,12)]))

trControl_mnl <- trainControl(
  method = "cv",
  number = 10,
  search = "grid",
  classProbs = FALSE,
  summaryFunction = multiClassSummary)

tuneGrid_mnl <- expand.grid(decay = seq(0, 1, by = 0.1))

set.seed(2091)

data_scale$popularity <- as.factor(data_scale$popularity)
data_scale$description_exists <- as.factor(data_scale$description_exists)

index <- caret::createDataPartition(data_scale$popularity,
                                    p = 0.7,
                                    list = FALSE)

train <- data_scale[index, ]
test <- data_scale[-index, ]

model_mnl <- caret::train(popularity ~ .,
                          data = train,
                          method = "multinom",
                          maxit = 100,
```

```
                          trace = FALSE,
                          tuneGrid = tuneGrid_mnl,
                          trControl = trControl_mnl)

model_mnl$bestTune

model_mnl$results %>% select(decay, Accuracy)

sum.mnl.c <- summary(model_mnl)
z <- sum.mnl.c$coefficients / sum.mnl.c$standard.errors
p <- (1 - pnorm(abs(z), 0, 1)) * 2

sum.mnl.c

# Coefficients for popularity = 2
coeff.mnl.c1 <- rbind(sum.mnl.c$coefficients[1,],
                      sum.mnl.c$standard.errors[1,],
                      z[1,],
                      p[1,])

rownames(coeff.mnl.c1) <- c("Coefficient", "Std. Errors",
                            "z stat", "p value")

# Coefficients for popularity = 3
coeff.mnl.c2 <- rbind(sum.mnl.c$coefficients[2,],
                      sum.mnl.c$standard.errors[2,],
                      z[2,],
                      p[2,])

rownames(coeff.mnl.c2) <- c("Coefficient", "Std. Errors",
                            "z stat", "p value")

# Coefficients for popularity = 4
coeff.mnl.c3 <- rbind(sum.mnl.c$coefficients[3,],
                      sum.mnl.c$standard.errors[3,],
                      z[3,],
                      p[3,])

coeff.mnl.c1
coeff.mnl.c2
coeff.mnl.c3

rownames(coeff.mnl.c3) <- c("Coefficient", "Std. Errors",
                            "z stat", "p value")

caret::confusionMatrix(predict(model_mnl,
                               newdata = test,
                               type = "raw"),
                       reference = test$popularity,  mode = "prec_recall")

varImp(model_mnl, scale = FALSE)
```

## 7.2 Python programos kodas

```
# Thumbnails
import pandas as pd
import imquality.brisque as brisque
from PIL import Image
from PIL import ImageStat
import requests
from io import BytesIO
import time
import warnings
import random
import urllib.request
from urllib.error import HTTPError
import cv2
import os
from colorthief import ColorThief
import webcolors
import numpy as np
from datetime import datetime
warnings.simplefilter(action='ignore', category=FutureWarning)

def getVideoThumbnailURL(video_id):
    return "https://i.ytimg.com/vi/"+video_id+"/mqdefault.jpg"

def downloadJPG(idx, url):
    print("idx:" + str(idx) + " url: " + url)
    try:
        urllib.request.urlretrieve(url, "C:\\Projects\\Python\\Bakalauras\\images\\"
+idx+".jpg")
    except HTTPError:
        print("  - nerastas image, http error")

def briske(idx, url):
    print("idx: " + str(idx) + " url: " + url)
    time.sleep(0.05)
    try:
        return brisque.score(Image.open(BytesIO(requests.get(url).content)));
    except AssertionError:
        print("klaida:")
        return 1000;

def brightness(im_file):
    print("brightness file: " + im_file)
    try:
        im = Image.open(im_file).convert('L')
        stat = ImageStat.Stat(im)
        return stat.rms[0]
    except FileNotFoundError:
        print("  - not found.")
        return 1000

def contrast(im_file):
```

```
    print("contrast file: " + im_file)
    if(os.path.isfile(im_file)):
        img = cv2.imread(im_file)
        img_grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        contrast = img_grey.std()
        return contrast
    else:
        print("  - not found.")
        return 1000

def variance_of_laplacian(im_file):
    print("blurriness file: " + im_file)
    if(os.path.isfile(im_file)):
        img = cv2.imread(im_file)
        return cv2.Laplacian(img, cv2.CV_64F).var()
    else:
        print("  - not found.")
        return -1

data = pd.read_json(r'data.json')

data['row_index'] = range(len(data))

data['video_thumbnail'] = data.apply(lambda row: getVideoThumbnailURL(row['video_id']),
axis=1)

data.apply(lambda row: downloadJPG(str(row['row_index']), row['thumbnail']), axis=1)

data['thumbnail_brightness'] = data.apply(lambda
row: brightness("C:\\Projects\\Python\\Bakalauras\\images\\"+ str(row['row_index'])
+ ".jpg"),
axis=1)

data['thumbnail_contrast'] = data.apply(lambda
row: contrast("C:\\Projects\\Python\\Bakalauras\\images\\"+ str(row['row_index']) +
".jpg"),
axis=1)

data['thumbnail_quality'] = data.apply(lambda
row: briske(row['row_index'], row['video_thumbnail']),
axis=1)

data['thumbnail_blurriness'] = data.apply(lambda
row: variance_of_laplacian("C:\\Projects\\Python\\Bakalauras\\images\\"+
str(row['row_index']) + ".jpg"),
axis=1)

# k-means
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

```
from sklearn.preprocessing import StandardScaler
from kneed import  KneeLocator
import matplotlib.pyplot as plt

data = pd.read_json(r"data.json")

data_features = data[["avg_daily_views",
                      "thumbnail_quality",
                      "channel_subscriber_count",
                      "chr_title",
                      "thumbnail_contrast",
                      "thumbnail_brightness",
                      "thumbnail_blurriness",
                      "avg_daily_likes",
                      "avg_daily_dislikes",
                      "avg_daily_comments",
                      "age",
                      "keywords_count",
                      "description_exists"]]

data_features['description_exists'] = pd.to_numeric(data_features.description_exists)

scalar = StandardScaler()

features = ["avg_daily_views",
            "thumbnail_quality",
            "channel_subscriber_count",
            "chr_title",
            "thumbnail_contrast",
            "thumbnail_brightness",
            "thumbnail_blurriness",
            "avg_daily_likes",
            "avg_daily_dislikes",
            "avg_daily_comments",
            "age",
            "keywords_count"]

standardized_features = pd.DataFrame(
    scalar.fit_transform(data_features[features].copy()),
columns = features)

old_shape = data_features.shape

data_features.drop(features, axis = 1, inplace = True)

X_std = pd.concat([data_features, standardized_features], axis= 1)

pca = PCA(n_components=13)
principalComponents = pca.fit_transform(X_std)
features = range(pca.n_components_)
PCA_components = pd.DataFrame(principalComponents)

ks = range(1, 10)
```

```
inertias = []
for k in ks:
    model = KMeans(n_clusters=k)
    model.fit(X_std)
    inertias.append(model.inertia_)

kl = KneeLocator(range(1, 10), inertias, curve="convex", direction="decreasing")
kl.elbow

kl.plot_knee()
plt.legend().remove()
plt.ylabel("WCSS")
plt.xlabel("Klasterių skaičius")
plt.gcf().set_size_inches(8, 6)
plt.gca().set_title("")
plt.show()

model = KMeans(n_clusters=4)
model.fit(X_std)
fig = plt.figure(figsize=(8, 6))
labels = model.predict(X_std)
scatter = plt.scatter(PCA_components[0], PCA_components[1], c=labels)
plt.legend(handles=scatter.legend_elements()[0], labels=[1,2,3,4], prop={'size': 15},
loc = 4)
plt.title("")
plt.ylabel("y")
plt.xlabel("x")
plt.show()

data_features = data[["avg_daily_views",
                      "thumbnail_quality",
                      "channel_subscriber_count",
                      "chr_title",
                      "thumbnail_contrast",
                      "thumbnail_brightness",
                      "thumbnail_blurriness",
                      "avg_daily_likes",
                      "avg_daily_dislikes",
                      "avg_daily_comments",
                      "age",
                      "keywords_count",
                      "description_exists"]]

labels_kmeans = pd.DataFrame(labels)

data_clusters_kmeans = pd.concat([labels_kmeans, data_features], axis= 1)

data_clusters_kmeans = data_clusters_kmeans.rename(columns={0: "Cluster"})

mean_k_means = data_clusters_kmeans.groupby(['Cluster']).mean()

data_clusters_kmeans.groupby(['Cluster']).size()
```

```
# SOM
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from minisom import MiniSom

data = pd.read_json(data.json")

data_features = data[["avg_daily_views",
                      "thumbnail_quality",
                      "channel_subscriber_count",
                      "chr_title",
                      "thumbnail_contrast",
                      "thumbnail_brightness",
                      "thumbnail_blurriness",
                      "avg_daily_likes",
                      "avg_daily_dislikes",
                      "avg_daily_comments",
                      "age",
                      "keywords_count",
                      "description_exists"]]

data_features['description_exists'].copy =
pd.to_numeric(data_features['description_exists'])

scalar = StandardScaler()

features = ["avg_daily_views",
            "thumbnail_quality",
            "channel_subscriber_count",
            "chr_title",
            "video_thumbnail_contrast",
            "video_thumbnail_brigtness",
            "video_thumbnail_blurriness",
            "avg_daily_likes",
            "avg_daily_dislikes",
            "avg_daily_comments",
            "age",
            "keywords_count"]

standardized_features = pd.DataFrame(
    scalar.fit_transform(data_features[features].copy()), columns = features)

old_shape = data_features.shape
data_features.drop(features, axis = 1, inplace = True)
X_std = pd.concat([data_features, standardized_features], axis= 1)
data_scaled = X_std.values

som_shape = (1, 4)

som = MiniSom(som_shape[0], som_shape[1], data_scaled.shape[1], sigma=.5,
```

```
    learning_rate=.5,
    neighborhood_function='gaussian', random_seed=10)

som.train_batch(data_scaled, 1000, verbose=True)

winner_coordinates = np.array([som.winner(x) for x in data_scaled]).T
cluster_index = np.ravel_multi_index(winner_coordinates, som_shape)

pca = PCA(n_components=13)
principalComponents = pca.fit_transform(X_std)
features = range(pca.n_components_)
PCA_components = pd.DataFrame(principalComponents)

fig = plt.figure(figsize=(8, 6))
scatter = plt.scatter(PCA_components[0], PCA_components[1], c=cluster_index)
plt.legend(handles=scatter.legend_elements()[0], labels=[1,2,3,4], prop={'size': 15},
loc = 4)
plt.title("")
plt.ylabel("y")
plt.xlabel("x")
plt.show()

data_features = data[["avg_daily_views",
                      "thumbnail_quality",
                      "channel_subscriber_count",
                      "chr_title",
                      "video_thumbnail_contrast",
                      "video_thumbnail_brigtness",
                      "video_thumbnail_blurriness",
                      "avg_daily_likes",
                      "avg_daily_dislikes",
                      "avg_daily_comments",
                      "age",
                      "keywords_count",
                      "description_exists"]]

labels_som = pd.DataFrame(cluster_index)

data_clusters_som = pd.concat([labels_som, data_features], axis= 1)

data_clusters_som = data_clusters_som.rename(columns={0: "Cluster"})

mean_som = data_clusters_som.groupby(['Cluster']).mean()

data_clusters_som.groupby(['Cluster']).size()

# SVM classification
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVC
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

```
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.model_selection import GridSearchCV

mms = MinMaxScaler()

df = pd.read_json (r'data.json')

df = df[['popularity', 'thumbnail_quality', 'channel_subscriber_count', 'chr_title',
'thumbnail_contrast', 'thumbnail_brightness', 'thumbnail_blurriness', 'avg_daily_likes',
'avg_daily_dislikes', 'avg_daily_comments', 'keywords_count', 'description_exists']]

y = df[['popularity']]

df_scaled = df

df_scaled[['thumbnail_quality', 'channel_subscriber_count', 'chr_title',
'thumbnail_contrast','thumbnail_brightness', 'thumbnail_blurriness', 'avg_daily_likes',
'avg_daily_dislikes', 'avg_daily_comments',  'keywords_count']]  = mms.fit_transform(
df_scaled[['thumbnail_quality', 'channel_subscriber_count', 'chr_title',
'thumbnail_contrast', 'thumbnail_brightness', 'video_thumbnail_blurriness',
'avg_daily_likes', 'avg_daily_dislikes', 'avg_daily_comments', 'keywords_count']])

X_train, X_test, y_train, y_test = train_test_split(df_scaled, y, test_size = 0.30)

labels = np.array(y_train['popularity'])
features= X_train.drop('popularity', axis = 1)
feature_list = list(features.columns)
features = np.array(features)

labels_test = np.array(y_test['popularity'])
features_test= X_test.drop('popularity', axis = 1)
features_list_test = list(features_test.columns)
features_test = np.array(features_test)

tuned_parameters = [{'kernel': ['rbf'],
                     'gamma': [1e0, 1e-1, 1e-2, 1e-3,1e-4],
                     'C': [1e3, 1e2, 1e1, 1e0, 1e-1]}]

mdl = SVC(random_state = 1,  max_iter=- 1)

grid_search = GridSearchCV(mdl,
                           tuned_parameters,
                           refit = True,
                           verbose = 0,
                           n_jobs = -1,
                           cv = 10)

grid_search.fit(features, labels)
grid_search.cv_results_
grid_search.best_params_
```

```
model= SVC(C=1000, kernel='rbf', gamma=1, max_iter=- 1)
model.fit(features, labels)
model_predict = model.predict(features_test)

print(classification_report(labels_test, model_predict))

# RF classification
import pandas as pd
import numpy as np
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import multilabel_confusion_matrix

df = pd.read_json (r'data.json')

df = df[['popularity', 'thumbnail_quality', 'channel_subscriber_count', 'chr_title',
'thumbnail_contrast', 'thumbnail_brightness', 'thumbnail_blurriness', 'avg_daily_likes',
'avg_daily_dislikes', 'avg_daily_comments', 'keywords_count', 'description_exists']]

y = df[['popularity']]

X_train, X_test, y_train, y_test = train_test_split(df, y, test_size = 0.30)

labels = np.array(y_train['popularity'])
features= X_train.drop('popularity', axis = 1)
feature_list = list(features.columns)
features = np.array(features)

label_test = np.array(y_test['popularity'])
features_test= X_test.drop('popularity', axis = 1)
features_list_test = list(features_test.columns)
features_test = np.array(features_test)

n_estimators = [300, 400, 500, 600]
max_features = ["log2",  "sqrt"]
max_depth = [30,40,50]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 5]
max_leaf_nodes = [2200, 2300, 2500]

grid_param = {'n_estimators': n_estimators,
'max_features': max_features,
'max_depth': max_depth,
'min_samples_split': min_samples_split,
'min_samples_leaf': min_samples_leaf,
'max_leaf_nodes': max_leaf_nodes}

mdl = RandomForestClassifier()
```

```
grid_search = GridSearchCV(estimator = mdl,
                           param_distributions = grid_param,
                           verbose = 0,
                           n_jobs = -1,
                           cv = 10)

grid_search.fit(features, labels)
grid_search.best_params_
grid_search.best_params_

regressor = RandomForestClassifier(max_depth=40,
                                   max_features='log2',
                                   n_estimators=200,
                                   random_state=1,
                                   min_samples_split = 10,
                                   min_samples_leaf = 2,
                                   max_leaf_nodes = 2500)

regressor.fit(features, labels)
model_predict = model.predict(features_test)

print(classification_report(labels_test, model_predict))

feature_imp = pd.Series(regressor.feature_importances_,
index=feature_list).sort_values(ascending=False)

feature_imp
```