VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Didžiųjų duomenų klasterizavimas ir klasifikavimas

Big Data Clustering and Classification

Bakalauro baigiamasis darbas

Atliko:	Lina Norkevičiūtė	(parašas)
Darbo vadovas:	prof. dr. Olga Kurasova	(parašas)
Darbo recenzentas:	lekt. dr. Tomas Plankis	(parašas)

Vilnius – 2020

Santrauka

Šiuolaikiniame pasaulyje skaitmeniniais įrenginiais kiekvieną dieną surenkama ir generuojama vis daugiau duomenų. Jie pasižymi ne tik išskirtiniu dydžiu ir greičiu, kuriuo turi būti saugomi ir apdorojami, bet ir savo įvairove. Dauguma tokių duomenų yra nestruktūrizuoti ar tik pusiau struktūrizuoti, norint išsaugoti jų teisingumą ir vertę, reikalingos didžiųjų duomenų technologijos ir technikos. Informacijos gavimui iš surinktos medžiagos gali būti taikomos įvairios duomenų tyrybos užduotys, kurių pavyzdžiais yra klasterizavimas ir klasifikavimas. Vis dėlto, įprasti klasterizavimo ir klasifikavimo algoritmai nėra pritaikyti didiesiems duomenims. Naudojant juos, duomenis reikia iš anksto paruošti sumažinant nagrinėjamų požymių poaibį arba pasirenkant analizuoti tik dalį turimos medžiagos. Klasterizavimo ir klasifikavimo algoritmus didiesiems duomenims galima pritaikyti ir jų realizacijas vykdant lygiagrečiai arba paskirstytai daugybės įrenginių tinkle. Šiam tikslui naudojamos ir didžiųjų duomenų technologijos, tokios kaip MapReduce programavimo modelis, Apache Hadoop bei Apache Spark didžiųjų duomenų variklis. Jos leidžia atlikti didžiųjų duomenų analizę rūpinantis ne skaičiavimų ir duomenų skirstymu, o tik reikalingos informacijos ieškančio funkcionalumo kūrimu.

Raktiniai žodžiai: didieji duomenys, klasterizavimas, klasifikavimas, klasterių analizė, klasių analizė, MapReduce, Hadoop paskirstyta failų sistema, Apache Spark, MLlib

Summary

In today's world more and more data are collected and generated by digital devices every day. They are characterized not only by the exceptional volume and velocity at which they have to be saved and processed, but by their variety too. Most of this data are unstructured or just semi-structured, in order to preserve their veracity and value, Big Data technologies and techniques have to be used. Various data mining tasks, such as data clustering and classification, can be utilised for extracting information from collected material. However, most of the regular clustering and classification algorithms are not well suited for Big Data analysis. When using them, data have to be preprocessed by reducing the data features subset or selecting just a sample of available material. Clustering and classification algorithms can be applied to Big Data by performing them in parallel or in a distributed network of multiple devices. Various Big Data technologies, such as MapReduce programming model, Apache Hadoop framework and Apache Spark Big Data engine, can be used for this purpose too. They allow to perform Big Data analysis without putting too much effort into distributing data or calculations and focusing only on developing functionality for finding useful information.

Keywords: Big Data, clustering, classification, cluster analysis, class analysis, MapReduce, Hadoop distributed file system, Apache Spark, MLlib

TURINYS

Įvadas

Pasaulyje surenkamų ir generuojamų duomenų kiekis vis didėja. Šie duomenys skiriasi savo kilmės šaltiniais bei forma, kuria yra saugomi ir pateikiami apdorojimui bei peržiūrai. Dauguma tokių duomenų yra nestruktūrizuoti ar tik iš dalies struktūrizuoti, todėl jų saugojimas ir apdorojimas schemose, turinčiose iš anksto apibrėžtą griežtą struktūrą, tampa gana sudėtingai įvykdoma užduotimi. Norint išsaugoti duomenų teisingumą ir vertę nuolat besikeičiančioje bei atsinaujinančioje aplinkoje, jie turi būti renkami, saugomi ir apdorojami pakankamai dideliu greičiu. Plečiantis ir vystantis kompiuterių tinklams bei daiktų internetui, greitis tampa vis svarbesne duomenų tyrybos charakteristika, kadangi medžiaga, gauta iš skaitmeninių įrenginių sensorių, kompiuterinių programų žurnalų ir socialinių tinklų įrašų, gali būti vertinama kaip nenutrūkstantis potencialiai vertingų duomenų srautas. Toks išskirtinis duomenų greitis, įvairovė ir dydis lemia tai, jog iki šiol plačiai naudotos tradicinės duomenų tvarkymo sistemos ir metodai nėra pritaikyti efektyviai atlikti šių duomenų saugojimo ir apdorojimo užduotis. Dėl to vis aktualesne tampa didžiųjų duomenų tema.

Vertingos informacijos gavimui iš duomenų gali būti taikomos tokios duomenų tyrybos užduotys, kaip klasterizavimas ir klasifikavimas. Vis dėlto, įprasti klasterizavimo ir klasifikavimo algoritmai bei šių algoritmų rezultatų vertinimo metodai yra skirti analizuoti pakankamai nedidelį kiekį duomenų, todėl darbui su didžiaisiais duomenimis nėra visiškai tinkami. Tokie algoritmai bei jų vertinimo metodai šiame darbe įvardijami kaip tradiciniai ir yra dalimi tyrimo objekto – tradicinių klasterizavimo ir klasifikavimo algoritmų pritaikymo didiesiems duomenims.

Šio darbo tikslas – įvertinti kokius vykdymo pakeitimus ir technologijas galima pritaikyti tradiciniams klasterizavimo bei klasifikavimo algoritmams, siekiant efektyviai analizuoti didžiuosius duomenis. Tokio tikslo siekiama sprendžiant penkis uždavinius:

- identifikuojant išskirtines didžiųjų duomenų savybes ir iššūkius, su kuriais susiduriama vykdant tokio pobūdžio duomenų tyrybą;
- išnagrinėjant tradicinius duomenų klasterizavimo ir klasifikavimo algoritmus, jų vertinimo metodus bei įvertinant pritaikymą didiesiems duomenims;
- identifikuojant bei išbandant technologijas ir kitą programinę įrangą, skirtą didžiųjų duomenų apdorojimui;
- nustatant galimus klasterizavimo ir klasifikavimo algoritmų vykdymo pakeitimus, skirtus juos pritaikyti didžiųjų duomenų tyrybai;
- palyginant skirtingų didiesiems duomenims pritaikytų algoritmų rezultatus analizuojant įvairius duomenų kiekius.

Laukiami rezultatai – daugumai nagrinėjamų tradicinių algoritmų bei jų vertinimo metodų galima identifikuoti būdus efektyvios didžiųjų duomenų analizės pritaikymui. Tikėtina, kad modifikuotas algoritmų vykdymas bus sudėtingesnis, o, analizuojant mažą kiekį duomenų, ir lėtesnis, tačiau duomenų kiekiui augant, tokie algoritmai išliks pakankamai spartūs ir tikslūs kokybiškam

duomenų tyrybos rezultatų gavimui. Be to, tikimasi, jog dauguma tradicinių klasterizavimo bei klasifikavimo algoritmų bus tinkami ir analizuojant didelius kiekius duomenų dėl didžiųjų duomenų technologijų pritaikymo.

Darbo atlikimui pasirinkta analizuoti dalį mokslinės literatūros didžiųjų duomenų, klasterizavimo ir klasifikavimo temomis, nagrinėti didiesiems duomenims naudojamų technologijų dokumentacijas bei informacinę medžiagą, kuri pateikiama įmonių, teikiančių duomenų saugojimo ir tvarkymo sprendimus. Dalies darbe naudojamų technologijų veikimui detalizuoti nagrinėjami moksliniai darbai, kuriuose šios technologijos yra pristatomos ir aptariamos.

Taip pat nuspręsta atlikti duomenų klasterizavimo ir klasifikavimo užduočių eksperimentus, lyginti jų rezultatus naudojant skirtingus algoritmus ir duomenų kiekius taikant didžiųjų duomenų technologijas. Šiems eksperimentams naudojami testiniai duomenų rinkiniai, sugeneruoti ir pritaikyti duomenų analizės technologijų išbandymui bei medžiaga iš viešai prieinamų duomenų saugyklų. Duomenų apdorojimui pasirinkta naudoti debesų kompiuterijos skaičiavimams skirtą Apache Spark platformą ir joje pateikiamos MLlib bibliotekos algoritmus:

- K-vidurkių algoritmą;
- Gauso sumaišymo algoritmą;
- Naivaus Bajeso algoritmą;
- sprendimų medžio algoritmą.

Darbe vykdomiems eksperimentams naudojami algoritmai vykdomi asmeniniu kompiuteriu, tačiau jie gali būti pritaikyti ir atlikimui debesų kompiuterijos platformomis. Taip pat darbe pateikiami klasterizavimo vizualizacijos rezultatai ir išbandomi algoritmai, esantys Elki duomenų tyrybos įrankyje:

- K-vidurkių algoritmas;
- lūkesčių maksimizavimo algoritmas.

Siekiant kuo efektyviau išnagrinėti gautus rezultatus, tyrimas bus atliekamas ribojant duomenų kiekius bei stebint algoritmų vykdymo laiką. Rezultatams pasirinkta atlikti lyginamąją analizę vertinant gautų įverčių, naudotų algoritmų bei technologijų sąsajas.

1. Didieji duomenys

1.1. Apibrėžimas

Kompiuteriais ir kitais skaitmeniniais įrenginiais surenkamų bei generuojamų duomenų kiekiai kiekvienais metais vis didėja. Skaičiuojama, kad 2012 metais kiekvieną dieną buvo sukuriama apie 2,5 eksabaito (10^{18} *baitų*) duomenų ir šis skaičius kas 40 mėnesių padvigubėdavo [BM12]. 2015 metais bendras surinktų skaitmeninių duomenų kiekis siekė 8 zetabaitus (10^{21} *baitų*) [Raj16], o 2020 metais jau turėtų išaugti iki 40 zetabaitų (10^{21} *baitų*) [KKA+16], kurių dauguma yra nestruktūrizuoti (angl. *unstructured*) ir turi skirtingus kilmės šaltinius. Šiuos duomenis galima apibūdinti didžiųjų duomenų (angl. *Big Data*) terminu.

Duomenų kiekio augimą lėmė didėjantis skaitmeninių įrenginių prieinamumas ir paplitimas. Atsirado ne tik daugiau kompiuterių, bet ir mobiliųjų telefonų, sensorių, atpažinimo radijo bangomis (angl. *RFID – radio-frequency identification*) įrenginių, kurie tarpusavyje gali būti sujungti ir vadinami daiktų internetu (angl. *Internet of Things*). 2009 metais šių įrenginių kiekis viršijo Žemėje gyvenančių žmonių skaičių, o 2020 turėtų siekti 26 milijardus [DGG15]. Galima manyti, jog sparčiai augantis didžiųjų duomenų kiekis turi vis daugiau potencialo suteikti naudingos informacijos moksliniams tyrimams, verslo rezultatų gerinimui bei kaštų mažinimui siekiant šių tikslų.

Vis dėlto tradicinės duomenų saugojimo ir apdorojimo sistemos nėra tinkamai pritaikytos darbui su didžiaisiais duomenimis. Tai lemia ne tik didelis duomenų kiekis, bet ir jų įvairovė. Didžiaisiais duomenimis gali būti laikomi socialinių tinklų įrašai, tekstiniai, audio ir video failai, vaizdinė informacija paveiksluose, kompiuterinių programų žurnalai (angl. *log files*), mobiliųjų įrenginių GPS signalai, sensorių surinkti duomenys skaitmeniniuose įrenginiuose ir daugelis kitų rūšių medžiagos, kuri gali būti paversta žinių teikiančia informacija. Duomenų analizė vertingai informacijai išgauti vykdoma ne tik technikomis, siejamomis su didžiaisiais duomenimis. Ją gali atlikti ir tradicinės duomenų saugyklų sistemos (angl. *data warehousing systems*), tačiau jos taiko kitokius analitinius ir duomenų valdymo metodus, kurie nėra tinkami informacijos struktūrizavimui ir paieškai nuolatiniame duomenų sraute, kuris gaunamas iš sensorių, mobiliųjų įrenginių siunčiamų signalų, naudotojų veiksmų žurnalų bei tekstų, pateiktų žmogiškąja kalba, ir kitur. Atlikti įvesties, gautos iš tokių šaltinių, analizę labiau pritaikytos yra didžiųjų duomenų technologijos, kurios kitaip negu tradicinės duomenų bazių ir saugyklų sistemos, gali įveikti iššūkius, susijusius su naujų rūšių duomenų srautais ir suteikti prie vertingų įžvalgų vedančios informacijos [KKA+16].

Poreikis didžiųjų duomenų apdorojimui naudoti kitokias sistemas ir metodus, nei gali pasiūlyti tradicinės duomenų bazės ir saugyklos, atsispindi ir viename iš šios sąvokos apibrėžimų, aprašytų De Mauro, Greco ir Grimaldi 2015 metais [DGG15]. Jame teigiama, kad „duomenys tampa didžiaisiais, kai jie viršija apdorojimo galimybes įprastose duomenų bazių sistemose“. Anot De Mauro, Greco ir Grimaldi, šie duomenys „reprezentuoja informacinį turtą, turintį tokį didelį kiekį, greitį ir įvairovę, kad jam reikia specialių technologijų ir analitinių metodų tam, kad būtų paverstas verte“.

1.2. Charakteristikos

1.2.1. Tradicinės charakteristikos

Literatūroje dažnai išskiriamos trys tradicinės didžiųjų duomenų charakteristikos. Jos papildo didžiųjų duomenų apibrėžimą, išryškindamos šios strities išskirtinumus ir kartu yra vadinamos 3V[1] modeliu, kurio komponentus 2011 metais aprašė Zikopoulos, Eaton ir kiti autoriai IBM korporacijoje [IZE11]:

- **Dydis (angl. *volume*)** apibūdina vis didėjančius duomenų kiekius, kuriuos reikia saugoti ir analizuoti. Šiais laikais jie yra per dideli tradicinėms duomenų apdorojimo sistemoms ir metodams. Nepaisant to, didelis kiekis gali reikšti potencialiai didelę vertę naujai informacijai gauti. Kaupiant ir bandant suprasti viską, ką galima užfiksuoti, verslas gali geriau suprasti savo klientus ir rinką, efektyviau prižiūrėti gamybos procesus ir įrenginių būklę. Vis dėlto galimybės apdoroti surinktus duomenis nedidėja taip greitai kaip pajėgumai juos surinkti. Dėl šios priežasties santykinė išanalizuotų duomenų dalis mažėja ir kelia vis didesnę riziką, kad vertinga informacija bus nepastebėta.

- **Greitis (angl. *velocity*)** skirtas apibūdinti spartą, kuria didieji duomenys yra generuojami ir apdorojami. Tradiciškai duomenų greičio apibrėžimas asocijuojamas su tuo kaip greitai duomenys atsiranda, gali būti išsaugoti ir surasti. Didžiųjų duomenų kontekste norint, kad informacija vis dar būtų aktuali ir vertinga, analizė turi būti atliekama beveik realiu laiku, kol duomenys yra generuojami ir net nėra patalpinti į saugyklas. Dėl šios priežasties didžiųjų duomenų apdorojimą reikėtų vertinti kaip darbą su pastoviu duomenų srautu, o ne sąlyginai statine informacija, kaip yra tradicinio duomenų apdorojimo kontekste.

- **Įvairovė (angl. *variety*)** apibūdina skirtingus didžiųjų duomenų tipus. Išaugęs skaitmeninių įrenginių, sensorių ir interneto naudotojų skaičius lėmė, kad duomenys, kuriuos reikia apdoroti, tapo daug labiau kompleksiškesni ir įvairesni nei bet kada anksčiau. Tradicinės reliacinės duomenų bazės yra pritaikytos darbui su pakankamai struktūrizuotais duomenimis, tačiau didžioji dalis šiandien surenkamų duomenų yra neapdoroti ir gryni (angl. *raw*), pusiau struktūrizuoti ar visai nestruktūrizuoti, tokie kaip: socialinių tinklų įrašai, elektroniniai laiškai, dokumentai, pateikti žmonėms suprantama kalba, audio, video medžiaga ir paveiksliukuose randama vaizdinė informacija. Juos sudėtinga patalpinti į reliacinių duomenų bazių lenteles tam, kad būtų galima apdoroti įprastiniais analitiniais metodais. Nepaisant to, organizacijos, kurios dirba su didžiaisiais duomenimis, turi gebėti analizuoti ir apdoroti visus duomenų tipus tam, kad galėtų surinkti vertingą informaciją ir išliktų konkurencingos rinkoje.

[1] 3V modelio pavadinimas sudarytas iš anglų kalba pateiktų trijų tradicinių charakteristikų pirmųjų raidžių: Volume, Velocity, Variety.

1.2.2. Papildomos charakteristikos

Kai kurie autoriai išskiria ir daugiau didžiųjų duomenų charakteristikų, kurios kartu su tradicinėmis suformuoja naujus modelius. Pavyzdžiui, vienas iš dažniau naudojamų yra 5V[2] modelis, kurį Marr aprašė 2014 metais [Mar14]. Šiam modeliui, kartu su dydžio, greičio ir įvairovės charakteristikomis, priskiriamos teisingumo bei vertės charakteristikos:

- **Teisingumas (angl. *veracity*)** naudojamas apibūdinti analizuojamų duomenų kokybę. Užfiksuotų duomenų kokybė gali ženkliai skirtis dėl skirtingų tipų ir kilmės šaltinių. Didelio teisingumo duomenys vertingi analizei, tuo tarpu mažas teisingumas indikuoja didelį kiekį triukšmo, trukdančio analitiniams metodams. Vis dėlto šį triukšmą galima kompensuoti didesniu duomenų kiekiu, būdingu šiai sričiai.

- **Vertė (angl. *value*)** viena svarbiausių didžiųjų duomenų charakteristikų, kuri apibūdina naudą, gautą duomenų rinkimo ir analizės metu. Ji gali parodyti ar investicijos į didžiųjų duomenų technologijas atsiperka ir ar verta tai vystyti toliau.

1.3. Duomenų tipai

Viena iš svarbiausių didžiųjų duomenų savybių yra jų įvairovė. Didėjantis surenkamų ir generuojamų duomenų kiekis lemia ne tik tai, kad skiriasi jų kilmės šaltiniai, bet ir patys duomenų tipai, kuriuos reikia saugoti, tvarkyti ir analizuoti. De Mauro, Greco ir Grimaldi 2019 metais [DGG19] išskyrė tris duomenų tipus su jų pavyzdžiais:

- **Struktūrizuotais (angl. *structured*)** duomenimis galima laikyti skaitiniu ir tekstiniu formatu pateiktą medžiagą, kuriai nereikia žmogaus kalbos ar konteksto, kuriame ji pateikta, interpretavimo;

- **Pusiau struktūrizuoti (angl. *semistructured*)** duomenys gali būti pateikti XML ar RSS[3] formatu, kuris gali būti standartizuotas ir suprantamas ne tik žmogui, bet ir kompiuteriui;

- **Nestruktūrizuoti (angl. *unstructured*)** duomenys, kartu su įvairialypės terpės (angl. *multimedia*) duomenimis sudaro didžiąją dalį surenkamos medžiagos kiekio. Šie duomenys apima audio ir video failus, paveikslėlius bei tekstus, kuriuose naudojama žmogaus kalba. Be to, vis daugiau duomenų gaunama iš internetinių puslapių, jų naudojimo žurnalų (angl. *log files*), socialinių tinklų įrašų, susirašinėjimų internete, skaitmeninių įrenginių sensorių [IZE11] bei stebėjimo kamerų [Raj16].

Skirtingų tipų apibrėžimas didžiųjų duomenų kontekste svarbus tuo, kad norint gauti kuo daugiau vertės iš surenkamų duomenų, reikia apdoroti ir pusiau struktūrizuotą bei nestruktūrizuotą

[2] 5V modelio pavadinimas sudarytas iš anglų kalba pateiktų penkių didžiųjų duomenų charakteristikų pirmųjų raidžių, į kurį įeina visos trys tradicinės charakteristikos (išvardintos 3V modelyje) ir dvi papildomos: Veracity, Value.

[3] RSS (angl. *Really Simple Syndication*) - duomenų formatas, skirtas reguliariai ir automatiškai pateikti turinį į interneto naudotojų žinių rinkiklius.

medžiagą, tačiau tradicinės duomenų saugojimo ir apdorojimo sistemos nėra pritaikytos darbui su ja. Šioms sistemoms reikalinga, kad duomenys būtų struktūrizuoti, pateikti tinkamu formatu ir atitinkantys iš anksto apibrėžtas griežtas taisykles. Vis dėlto didžioji dalis šiais laikais surenkamų duomenų yra neapdoroti, nestruktūrizuoti ar tik pusiau struktūrizuoti. Tradicinės sistemos nėra pajėgios apdoroti tokių tipų medžiagos, tam labiau pritaikytos yra didžiųjų duomenų technologijos [IZE11], kurių viena iš paskirčių gali būti mažai struktūrizuotus duomenis apdorojant paversti labiau struktūrizuotais.

1.4. Potenciali nauda

Perėjimas nuo tradicinių duomenų bazių ir duomenų apdorojimo sistemų prie didžiųjų duomenų technologijų reikalauja ne tik suinteresuotų asmenų iniciatyvos ir noro pokyčiams, bet ir gana reikšmingų finansinių investicijų, reikalingų žmonių apmokymui ir naujai įrangai. Dėl to verta atkreipti dėmesį ir į tai, kokia yra šių technologijų teikiama nauda mokslui, verslui, viešajam sektoriui ir kitoms duomenis naudojančioms sritims.

Pajusti gana ryškų pokytį, naudojant didžiųjų duomenų technologijas, gali verslas. MIT skaitmeninio verslo centro (angl. *MIT Center for Digital Business*) atlikto tyrimo metu paaiškėjo, jog kompanijos, savo veiklą labiau grindžiančios duomenimis, rodo geresnius finansinius ir veiklos rezultatus. Skaičiuojama, kad tokios įmonės priimdamos labiau duomenimis paremtus sprendimus uždirba 6 procentais daugiau pelno ir yra 5 procentais produktyvesnės nei jų rinkos konkurentai [BM12].

Šis fenomenas išryškėja elektroninės komercijos srityje, kur pasitelkiant didžiuosius duomenis paslaugų teikėjai bei elektroninės parduotuvės gali geriau suprasti savo klientus, teikti individualiai labiau pritaikytus pasiūlymus ir efektyviau išnaudoti savo resursus. Brynjolfsson ir McAfee 2012 metais [BM12] aprašė Sears Holdings kompanijos bandymą, naudojant surenkamus duomenis, klientams pateikti aktualesnius savalaikius paslaugų pasiūlymus. Tradicinės duomenų apdorojimo technologijos ir saugyklos, kurias iš pradžių bandė naudoti įmonė, nebuvo pajėgios tinkamai laiku atlikti šią užduotį dėl itin didelio duomenų kiekio ir įvairovės, tačiau pasitelkus didžiųjų duomenų technologijas, tai įvykdyti pavyko kokybiškiau, pigiau ir 8 kartus greičiau. Analizuojant surinktus duomenis, elektroninės parduotuvės gali tikėtis geresnių rezultatų. Jos gali geriau suprasti savo klientus ir pateikti kokybiškiau pritaikytus pasiūlymus pagal tai, kokiomis prekėmis žmogus domėjosi, kokią įtaką sprendimams padarė reklama, kitų klientų atsiliepimai bei puslapio išdėstymas. Toks gaunamų ir generuojamų duomenų srautas gali turėti nemažai vertingos informacijos, panaudojamos gerinti verslo rezultatus rinkoje, tačiau tokiu dideliu įvairių duomenų srautu tinkamai pasinaudoti ir tą informaciją išgauti galima tik pritaikius didžiųjų duomenų algoritmus ir metodus.

Didžiųjų duomenų nauda pastebima ir daugelyje kitų sričių, tokių kaip aviacija ir transportas. Oro uostai, naudodami didžiaisiais duomenimis paremtas informacines sistemas, gali geriau prognozuoti lėktuvų atvykimo laikus. Šiam tikslui naudojami daugybės sensorių renkami duomenys, lyginami su dideliu kiekiu istorinės informacijos, sukauptos prieš tai įvykusių skrydžių metu. 2012 metais Brynjolfsson ir McAfee [BM12] pateikė pavyzdį, kad tokio pobūdžio

PASSUR sistema, naudojant didžiuosius duomenis leido netikslumus ženkliai sumažinti. Be to, didžiųjų duomenų technologijos reikalingos ir analizuojant medžiagą, gaunamą iš tūkstančių sensorių naudojamų lėktuvuose. Greitas, tikslus ir savalaikis informacijos gavimas iš tokių duomenų aktualus ne tik efektyvesniam kuro sunaudojimui, bet ir lėktuvo dalių būklės bei pačio skrydžio stebėjimui, kuris gali lemti keleivių kelionės kokybę ir saugumą. Panašiu principu didieji duomenys taikomi ir kituose transporto sektoriuose mažinant spūstis, gerinant transporto priemonių priežiūrą ir kokybę [Raj16].

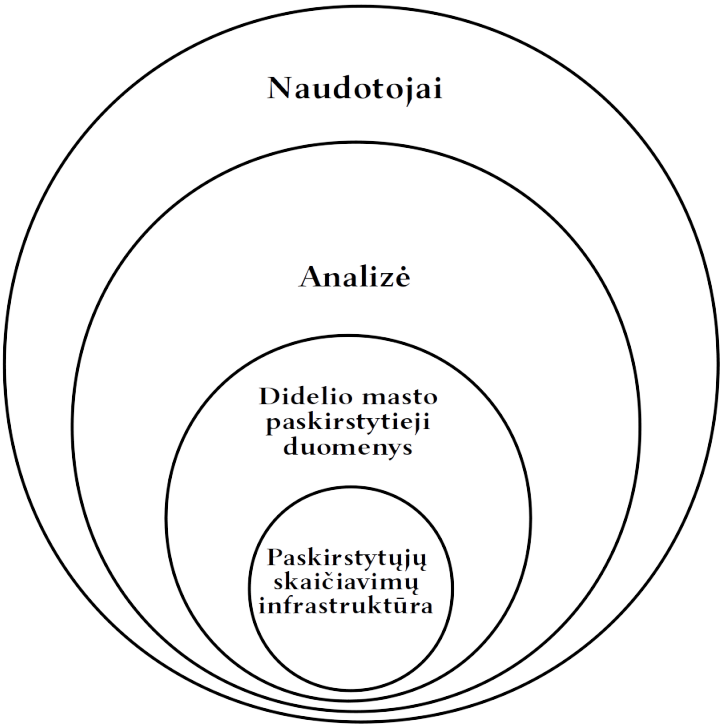
Technologijos, apdorojančios didžiuosius duomenis, naudojamos ir mokslinių tyrimų metu apdorojant didelius kiekius empirinių eksperimentų metu surinktos informacijos. Panašiai jos pritaikomos ir sveikatos apsaugos srityje nustatant diagnozes, saugant asmens sveikatos duomenis ar prognozuojant ligų protrūkius. Tokia didžiųjų duomenų svarba pastebima ir viešajame sektoriuje, kur tenka analizuoti visuomenės poreikius ir nuomonių tendencijas, tam kad būtų užtikrintas tinkamas sprendimų priėmimas.

1.5. Rizikos

Didžiųjų duomenų technologijos suteikia iki šiol neturėtas galimybes didžiulį kiekį renkamų duomenų paversti vertinga informacija, tačiau jos sukuria ir naujas rizikas šių dienų informaciniame pasaulyje. Viena iš jų – rizika asmens privatumui. Įrašai ir žinutės socialiniuose tinkluose bei elektroninių laiškų turinys gali būti panaudoti konfidencialios asmens informacijos atskleidimui ir kitai neetiškai veiklai, tokiai kaip žmonių skirstymas (angl. *profiling*) pagal jų pažiūras ar asmenines savybes [Raj16].

Privatumas tampa itin aktualus saugant bei apdorojant jautrius žmonių duomenis, pavyzdžiui, gautus iš medicininių įrašų. Tokie duomenų rinkiniai gali būti reikšmingi tikslesniam ligų diagnozavimui, ir moksliniams tyrimams sveikatos apsaugos srityje. Nepaisant to, viešas medicininių duomenų prieinamumas kelia grėsmę žmonių privatumui, todėl dažnai jie būna anonimizuojami asmens duomenis keičiant identifikaciniais numeriais. Vis dėlto, kaip 2016 metais rašė Rajaraman [Raj16], tokiais metodais apdoroti duomenys vis tiek išlieka pažeidžiami netinkamam naudojimui, nuo kurio juos bandoma apsaugoti ir kitais būdais.

Organizacijos, dirbančios su didžiaisiais duomenimis, turi skirti didelį dėmesį jų saugumui, kadangi netinkami sistemų apsaugos mechanizmai gali lemti daug kainuojančius įsilaužimus, duomenų nutekinimus bei praradimus. Kune ir kiti bendraautoriai 2016 metais [KKA+16] rašė, kad tradiciniai duomenų apsaugos metodai, pritaikyti nedidelėms sistemoms, nėra tinkami darbui su didžiaisiais duomenimis. Tai lemia ne tik išaugusio masto duomenų kiekiai, bet ir kitokia sistemų infrastruktūra, kurią gali sudaryti debesų kompiuterijos (angl. *cloud computing*) pagrindu sujungtų kompiuterių tinklai. Kune ir kiti bendraautoriai savo darbe pateikė tokioms sistemoms pritaikytą svogūno tipo apsaugos modelį didiesiems duomenims (angl. *Big Data Security Onion Model of Defense*), parodytą 1 paveikslėlyje. Šis modelis sudarytas iš keturių sluoksnių, kurie nurodo atskiruose sistemos lygiuose taikytinas apsaugos gaires, skirtas duomenų konfidencialumo ir privatumo išsaugojimui.



1 pav. Svogūno tipo apsaugos modelis didiesiems duomenims

- **Paskirstytųjų skaičiavimų infrastruktūros (angl. *distributed computing infrastructure*)** lygyje turi būti užtikrinta, kad lygiagrečiai vykstančių skaičiavimų metu konfidencialūs duomenys nebūtų atskleisti asmenims, neturintiems teisės šių duomenų žinoti;
- **Didelio masto paskirstytųjų duomenų (angl. *large-scale distributed data*)** lygyje nurodoma, kad duomenys turi būti greitai pasiekiami, šifruojami, privatūs ir pasiekiami tik tinkamas teises turintiems naudotojams;
- **Analizės (angl. *analytics*)** apsaugos lygyje pabrėžiama, kad analizės rezultatų peržiūrai turi būti taikomas kelių lygių naudotojų autentifikacijos mechanizmas;
- **Naudotojų (angl. *users*)** privatumo ir apsaugos lygis turi užtikrinti, kad duomenys apie pačius naudotojus išliks konfidencialūs ir validūs.

2. Klasterizavimas

2.1. Apibrėžimas ir uždavinys

Didžiuosius duomenis paversti vertinga informacija naudojamos įvairios duomenų tyrybos[4] (angl. *data mining*) technikos. Viena iš jų – klasterizavimas (angl. *clustering*), dar kitaip vadinama klasterių analize (angl. *cluster analysis*). Klasterizavimo metu duomenys yra suskirstomi į tarpusavyje panašių elementų grupes, vadinamas klasteriais. Verma ir kiti autoriai 2012 metais [VSC+12] rašė, kad „klasteris yra objektų kolekcija, kurie yra panašūs tarpusavyje ir nepanašūs į objektus, priklausančius kitiems klasteriams“.

Klasterizavimas svarbus apdorojant naujus duomenis, kuomet jie suskirstomi į iš anksto neapibrėžtas grupes ir randama informacija, kuri neapdorotuose (angl. *raw*) duomenyse įprastai būtų nepastebėta [TLC+15]. Didžiųjų duomenų kontekste, kur įvestis žinių gavybai surenkama realaus pasaulio sąlygomis, ši klasterizavimo savybė itin aktuali, nes dauguma surenkamų duomenų turi savybes, pagal kurias juos galima natūraliai sugrupuoti. Be to, klasterių analizės metu randami sąryšiai tarp duomenų ir modeliai (angl. *patterns*), kuriuos apbrėžia tik reikalavimai klasterių kūrimui, o ne atitikimas iš anksto apibrėžtoms kategorijoms ar savybėms. Dėl šios priežasties struktūrų ir modelių paieška, neturinti iš anksto apibrėžtų grupių, klasterių analizę leidžia vadinti neprižiūrimo mokymosi (angl. *unsupervised learning*) technika [VSC+12].

2.2. Algoritmai

Klasterių analizėje naudojami skirtingi algoritmai, kurie skiriasi ne tik jų vykdymo metu atliekamais žingsniais, bet ir klasterių tipais, randamais juos įvykdžius. Vienas iš kriterijų nurodo ar klasterizavimo metu gauti rezultatai suskirstyti į atskiras grupes ir klasterizavimas yra griežtas (angl. *hard partitioning*), ar elementas tam tikru lygiu priklauso ne vienam klasteriui ir gaunamas negriežtas klasterizavimas (angl. *soft (alt. fuzzy) clustering*) [VSC+12]. Be to, algoritmai skirstomi ir pagal tai, kaip tarpusavyje siejasi kiekviename klasteryje esantys duomenys, todėl gali būti paremti elementų hierarchija, jų susitelkimu apie centrą, pasiskirstymu ir tankumu.

Erman, Arlit ir Mahanti 2006 metais [EAM06] rašė, jog elementų tarpusavio panašumui klasteryje nustatyti, gali būti naudojami įvairūs matavimo vienetai, bet dažniausiai pasirenkamas Euklido atstumas. Mažas Euklido atstumas klasterių analizėje rodo, kad duomenys grupėje vienas su kitu yra itin panašūs, tuo tarpu dideli atstumai rodo menką panašumą tarp klasterio elementų. Kai duomenys klasterizuojami pagal n kriterijų kiekviename objekte, skirtumas tarp grupės elementų, pažymėtų p ir q gali būti apskaičiuojamas:

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

[4] Duomenų tyrybos (gavybos) (angl. *data mining*) terminas gali būti siejamas su aukso gavyba (angl. *gold mining*), kai purvas (duomenų tyryboje – neapdoroti duomenys) yra valomas gauti aukso grynuolius (duomenų tyryboje – žinias ir informaciją) [TLC+15].

2.2.1. K-vidurkių algoritmas

K-vidurkių (angl. *K-Means*) algoritmas priklauso susitelkimu apie centrą (angl. *centroid-based*) paremtų klasterizavimo algoritmų grupei. Jo metu suformuojama K kiekis sferos formos klasterių, o grupėje esančių narių panašumas užtikrinamas iteratyviai mažinant kvadratinį klaidos matą E:

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

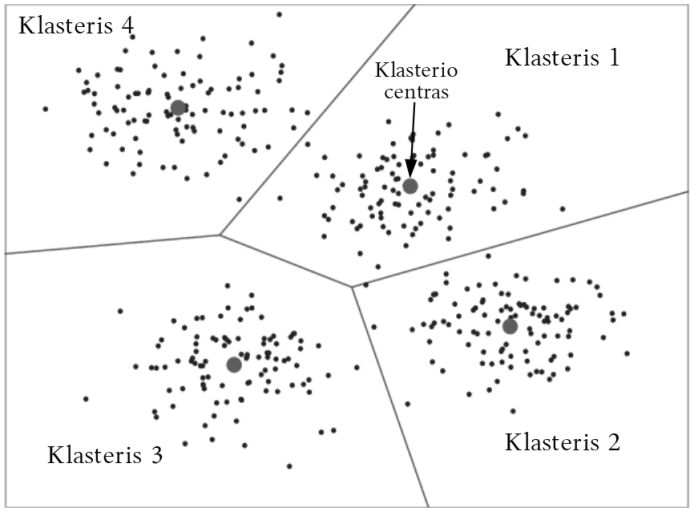
kur K yra ieškomų klasterių kiekis, n – analizuojamas objekto požymių skaičius, o atstumai d apskaičiuojami tarp kiekvieno objekto x ir klasterio centro c [EAM06]. K-vidurkių pseudoalgoritmo žingsnius 2012 metais aprašė Verma ir kiti [VSC+12]:

- atsitiktinai parenkama K klasterių centrų;
- suskaičiuojami atstumai tarp kiekvieno elemento ir klasterio centro;
- elementas priskiriamas klasteriui, nuo kurio centro yra mažiausiu atstumu;
- apskaičiuojami nauji klasterių centrai c, pagal formulę, kurioje k yra klasterio elementų kiekis:

$$c = \frac{1}{k} \sum_{i=1}^{k} x_i;$$

- perskaičiuojami atstumai tarp duomenų objektų ir naujai rastų klasterių centrų;
- jei atstumai nepasikeitė, tuomet klasteriai stabilizavosi ir algoritmas sustabdomas. Kitu atveju – skaičiavimai kartojami nuo elementų priskyrimo klasteriams žingsnio.

Atlikus klasterizavimą K-vidurkių algoritmu ir pavaizdavus jį grafiškai, kaip parodyta 2 paveikslėlyje, adaptuotame iš [EDS16], galima pamatyti, jog visi duomenų objektai priskiriami klasteriams, o erdvė padalinama į Voronojaus celes.



2 pav. Po klasterizavimo Voronojaus celėse išsidėstę klasteriai

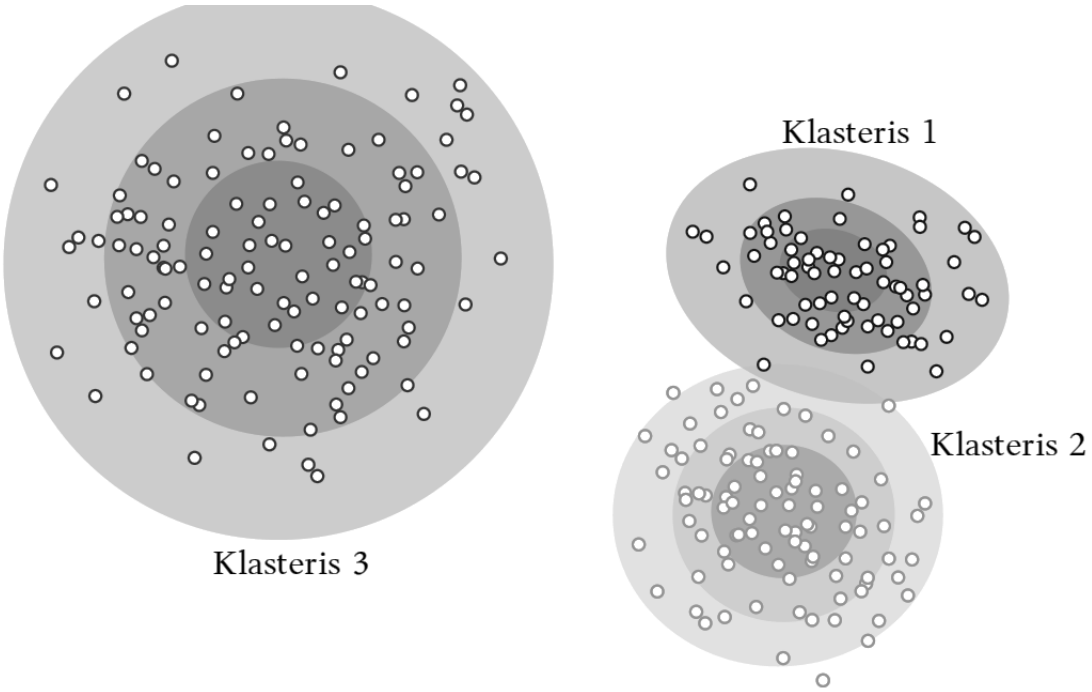
2.2.2. Lūkesčių maksimizavimo algoritmas

Lūkesčių maksimizavimo (angl. *expectation-maximization*, toliau – EM) algoritmas naudojamas, kai analizuojamų duomenų parametrų galimai trūksta. Šio algoritmo metu duomenys nėra griežtai klasterizuojami ir su tam tikra tikimybe priklauso kiekvienam klasteriui [AOO12]. Dėl šios priežasties jį galima priskirti negriežto klasterizavimo (angl. *soft clustering*) metodams. EM metodo pseudoalgoritmą sudaro lūkesčių (E) ir maksimizavimo (M) žingsniai:

- vykdant pirmąją iteraciją E žingsnio metu duomenims suteikiami atsitiktiniai įverčiai, reiškiantys tikimybes, kad duomuo priklauso tam tikram klasteriui [EAM06]. Tolimesnių iteracijų metu E žingsnis naudojamas apskaičiuoti kiekvieno duomens priklausymo klasteriams tikimybę;
- M žingsnio metu tikimybių pasiskirtymo parametrų vektorius, rastas E žingsnio metu, yra perskaičiuojamas taip, kad jo reikšmė palaipsniui artėtų prie lokalaus maksimumo.

Algoritmo žingsniai atliekami tol, kol pasiskirtymo parametrai susilieja arba pasiekiamas maksimalus iteracijų skaičius, todėl prieš vykdant algoritmą turi būti apibrėžtas ieškomų klasterių kiekis, leistina parametrų susiliejimo paklaida ir didžiausio iteracijų kiekio riba [AOO12].

Lūkesčių maksimizavimo algoritmas priskiriamas pasiskirstymu paremto (angl. *distribution-based*) klasterizavimo algoritmų grupei, todėl klasteris gali būti apibrėžtas kaip duomenų grupė, kurioje elementai yra pasiskirstę tankiau nei elementai, randami kituose klasteriuose. Tą pamatyti galima ir vizualiai pateiktame klasterių išsidėstyme, kurio pavyzdys pavaizduotas 3 paveikslėlyje[5].



3 pav. Lūkesčių maksimizavimo metodu gauti klasteriai

[5]Lūkesčių maksimizavimo (EM) metodu gautų klasterių vizualizacija, adaptuota iš `https://developers.google.com/machine-learning/clustering/clustering-algorithms`

2.2.3. DBSCAN algoritmas

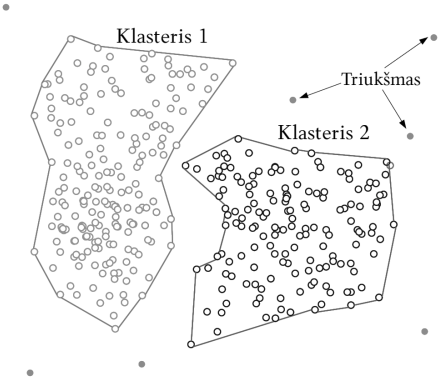
Klasterizavimo metu dažnai naudojami ir tankumu paremti (angl. *density-based*) algoritmai, iš kurių vienas yra vadinamas DBSCAN. Jo veikimas paremtas tankumo pasiekiamumu (angl. *density reachability*) ir tankumo sujungiamumu (angl. *density connectability*), kurie priklauso nuo dviejų DBSCAN algoritme naudojamų parametrų: minimalaus klasterio elementų kiekio m ir tarp jų esančios kaimynystės atstumo e [VSC+12].

DBSCAN klasterizavimo metu atstumas e apibrėžia objekto q kaimynystės dydį. Jei į tokiu matmeniu ribojamą erdvę patenka bent m kitų objektų, tuomet q laikomas šerdiniu (angl. *core*) objektu, kuris gali pasiekti visus kaimynystėje esančius elementus. Be to, objektui q pasiekiamais laikomi ir tie elementai, kurie patenka į q tiesiogiai ar tranzityviai pasiekiamų objektų kaimynystes. Jei tarp objektų p ir q egzistuoja jiems abiem pasiekiamas elementas, p ir q laikomi sujungtais DBSCAN vykdomo klasterių formavimo metu [EAM06].

Algoritmo, kurio metu DBSCAN randa klasterius, žingsniai 2006 metais buvo aprašyti Erman, Arlitt ir Mahanti [EAM06]:

- iš pradžių visi analizuojamos aibės objektai laikomi nepriskirti jokiam klasteriui;
- elementų aibėje parenkamas objektas q;
- jei q yra šerdinis objektas, remiantis kaimynystės atstumu e ir minimaliu klasterio elementų kiekiu m, randami visi su juo sujungti objektai ir priskiriami naujam klasteriui;
- jei q nėra šerdinis objektas, jis laikomas triukšmu ir analizei pasirenkamas kitas elementas;
- visus elementus priskyrus klasteriams arba triukšmui, algoritmas stabdomas.

Kaip ir kitų tankumu paremtų klasterių analizės metodų atveju, DBSCAN randamos grupės gali būti apibūdinamos kaip tankesnio elementų išsidėstymo erdvės, atskirtos retesnio išsidėstymo erdvėmis. Ši savybė lemia tai, kad DBSCAN algoritmu analizuojamų duomenų aibėje galima rasti objektus, kurie ženkliai skiriasi nuo visų klasterių ir gali būti laikomi triukšmu. Tai matoma ir 4 paveikslėlyje[6], kuriame pateikiamas pavyzdys, kaip tokie elementai, kartu su DBSCAN rastais klasteriais, atrodytų juos pateikus vizualiai.



4 pav. DBSCAN algoritmo rasti klasteriai

[6]DBSCAN algoritmo rastų klasterių pavyzdys, adaptuotas iš `https://developers.google.com/machine-learning/clustering/clustering-algorithms`

2.3. Algoritmų rezultatų vertinimas silueto koeficientu

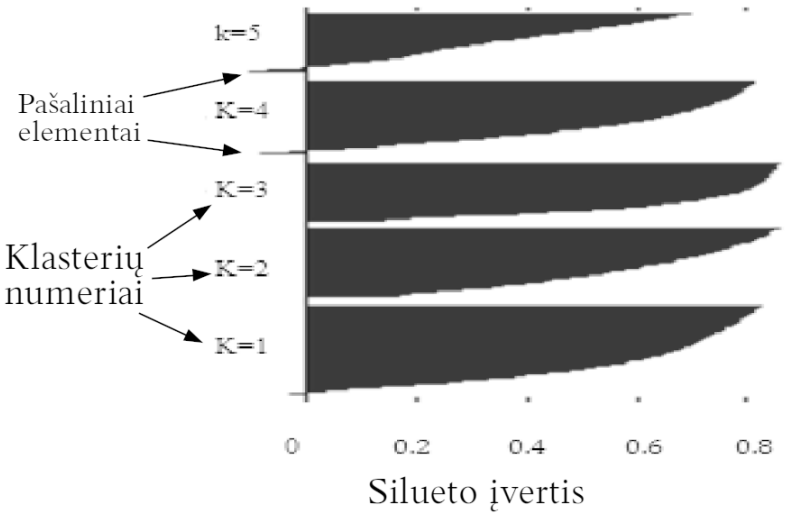
Klasterizavimo algoritmų kokybei nustatyti galima naudoti vidinio vertinimo (angl. *internal evaluation*) metodus, kurie klasterių analizės rezultatus vertina pagal tų pačių elementų informaciją, kurie buvo naudoti klasterizavimo metu.

Vienas iš tokių metodų vadinamas silueto koeficientu (angl. *silhouette coefficient*). Šis metodas gali parodyti, kurie objektai klasteriams buvo priskirti tinkamai, o kurie atsidūrė netoli kitų grupių ir gali būti laikomi pašaliniais (angl. *outliers*). Silueto koeficiento įvertis parodo silueto tankumą (angl. *tightness*) ir išskyrimą (angl. *separation*) pagal tai, kiek objektas yra arti kitų narių savo klasteryje ir kiek jis yra nutolęs nuo kitų klasterių. Be to, nagrinėjant vidutinius silueto įverčius, galima pamatyti kaip gerai veikia klasterizavimas ir naudoti juos parenkant tinkamą ieškomų grupių kiekį [BT14]. Burney ir Tarig 2014 metais aprašė silueto įverčio skaičiavimą:

- randamas vidutinis atstumų skirtumas *a(i)* tarp objekto *i* ir kitų tame pačiame klasteryje esančių objektų;
- ieškomi tarp objekto *i* ir kituose klasteriuose *C* esančių objektų vidutiniai atstumų skirtumai *d(i, C)*;
- parenkamas mažiausias iš atstumų *d(i, C)*, kuris pavadinamas *b(i)*;
- turint šiuos atstumus, skaičiuojamas silueto įvertis *s(i)* pagal formulę:

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

Randant daug pakankamai didelių įverčių yra laikoma, kad klasterizavimas ir jo konfiguracija parinkti tinkamai. Priešingu atveju, algoritmo konfigūraciją reikia keisti. Tokią informaciją galima pamatyti ir pavaizdavus siluetus vienoje erdvėje, kaip tai parodyta 5 paveikslėlyje [BT14].



5 pav. Analizės metu gautų klasterių siluetų įverčiai

2.4. Didžiųjų duomenų klasterizavimas

Klasterizavimas yra itin vertingas informacijos ir modelių (angl. *patterns*) radimo būdas įvairiuose duomenų rinkiniuose. Vis dėlto, nagrinėjant didžiuosius duomenis, ši tyrybos technika susiduria su iššūkiais, kurių pavyzdžiais gali būti didelis duomenų ir jų požymių kiekis, heterogeniškumas (angl. *heterogeneity*) bei pakankamai sudėtingas dalies algoritmų įgyvendinimas ir vykdymas. Be to, norint gauti kokybiškus klasterizavimo rezultatus, neretai yra reikalingos išankstinės su tiriamais duomenimis susijusios dalykinės srities žinios, kurios turi būti naudojamos parenkant tinkamus algoritmų vykdymo parametrus [BKL16].

Tradiciniai klasterizavimo metodai nėra visiškai pritaikyti didžiųjų duomenų analizei. Vykdant K-vidurkių algoritmą, kiekvieno kartojimo metu tikrinami visi nagrinėjamoje erdvėje esantys duomenys, siekiant nustatyti kuriems klasteriams jie priklauso ir perskaičiuojant visus klasterių centrus. Tuo tarpu lūkesčių maksimizavimo algoritmas iteratyviai tikrina kiekvieno duomens tikimybes priklausyti visiems erdvėje esantiems klasteriams. Ir nors tankumu paremtas DBSCAN algoritmas nevykdo nuolatinio klasterių centrų perskaičiavimo ar pakartotinio duomenų priskyrimo klasteriams, bet išlieka tikimybė, kad tokio algoritmo rasta duomenų grupė gali viršyti vieno įrenginio (angl. *single-machine*) atminties galimybes dėl potencialiai didelio kiekio tankiai pasiskirsčiusių nagrinėjamų duomenų. Su panašiais iššūkiais susiduriama ir vykdant algoritmų rezultatų vertinimą silueto koeficientu, kadangi jo metu tikrinant kiekvieno išnagrinėto duomens panašumą su kitais toje pačioje grupėje esančiais nariais ir skirtumą su duomenimis, esančiais kituose klasteriuose, šis metodas eksponentiškai sudėtingėja priklausomai nuo pradinės įvesties kiekio.

Poreikį didžiųjų duomenų klasterizavimui naudoti specialiai tam pritaikytus duomenų tyrybos algoritmus ir technologijas 2017 metais savo straipsnyje iškėlė Chen, Ludwig ir Li [CLL17]. Pasak jų „tradicinės klasterizavimo technikos negali susidoroti su tokiu dideliu kiekiu duomenų dėl jų sudėtingumo ir skaičiavimų kainos“. Dėl šios priežasties reikia ieškoti būdų kaip padidinti klasterizavimo algoritmų mastą (angl. *scale*) ir spartą neprarandant analizės rezultatų kokybės, tam kad didžiuosius duomenis būtų galima naudoti, nepaisant jų išskirtinio kiekio, dinamiškumo, sudėtingumo bei heterogeniškumo.

Didiesiems duomenims tinkami klasterizavimo algoritmai turi atitikti tam tikrą rinkinį savybių, reikalingų efektyviai kokybiškai tokio pobūdžio duomenų tyrybai. Dalis jų buvo pateikta Chen, Ludwig ir Li 2017 metais publikuotame straipsnyje [CLL17]:

- **plečiamumas (angl. *scalability*)** – algoritmo sudėtingumas ir vykdymo laikas neturėtų neproporcingai išaugti analizuojant didelius duomenų rinkinius;
- **patikimumas (angl. *robustness*)** – kraštutiniai duomenys (angl. *outliers*) neturėtų iškreipti įprastų klasterizavimo rezultatų;
- **eiliškumo ignoravimas (angl. *order insensitivity*)** – įvesties duomenų eiliškumas neturėtų paveikti galutinių algoritmo rezultatų;

- **minimali naudotojo įvestis (angl. *minimum user-specified input*)** – naudotojo įvedamų parametrų kiekis turi būti kiek įmanoma mažesnis;
- **nepriklausoma klasterių forma (angl. *arbitrary-shaped clusters*)** – klasterių forma neturi būti priklausoma nuo iš anksto pateiktų sąlygų;
- **duomens proporcijos leistinumas (angl. *point proportion admissibility*)** – pateikus skirtingas savybes, algoritmas turėtų pateikti skirtingus klasterizavimo rezultatus.

Tokias savybes atitinkantys algoritmai nebus paveikti didžiųjų duomenų srityje pasitaikančių įvesties dubliavimosi ir poreikio duomens klasterizavimą pakartoti, atliekant analizę ne vieną kartą.

Klasterizavimo algoritmus pritaikyti didesniems duomenų rinkiniams galima naudojant skirtingas technikas, gerinančias jų greitį ir plečiamumą (angl. *scalability*). Shirkhorshidi ir kitų autorių 2014 metais publikuotame darbe [SAW+14] šios technikos skirstomos į dvi rūšis:

- **klasterizavimas viename įrenginyje (angl. *single-machine clustering*)** – algoritmas vykdomas viename įrenginyje ir naudojami tik vieno įrenginio skaičiavimo ir atminties resursai;
- **klasterizavimas daugelyje įrenginių (angl. *multiple-machine clustering*)** – algoritmas atliekamas ne viename įrenginyje, todėl vykdymo metu gali naudoti daugiau resursų.

2.4.1. Klasterizavimas viename įrenginyje

Viena iš populiariausių technikų analizuoti didžiuosius duomenis vadinama klasterizavimu viename įrenginyje (angl. *single-machine clustering*). Nors šios grupės algoritmų vykdymui pasitelkiami viename skaitmeniniame įrenginyje prieinami skaičiavimų pajėgumai, analizuojami duomenys gali būti saugomi kitoje vietoje laikomoje atmintyje [CLL17].

Algoritmų plečiamumui ir greičiui pagerinti gali būti naudojamos imties ėmimo (angl. *sampling-based*) bei matmenų mažinimo (angl. *dimension reduction*) technikos. Imties ėmimo technika yra vienas iš paprasčiausių būdų klasterizavimą pritaikyti vis labiau didėjančiai analizuojamų duomenų erdvei. Šiai grupei priklausantys algoritmai nevykdo viso duomenų rinkinio analizės, vietoj to, jie klasterizavimą atlieka tik tam tikram poaibiui duomenų ir gautus rezultatus pritaiko likusiai rinkinio daliai. Toks būdas leidžia ženkliai paspartinti algoritmo vykdymo greitį ir sumažinti jo sudėtingumą, kadangi mažesnio įvesties kiekio analizavimas reikalauja ne tiek daug atminties ir skaičiavimų resursų [SAW+14].

Imties ėmimas nėra vienintelis būdas atlikti didžiųjų duomenų klasterizavimą viename įrenginyje. Praktikoje naudojama ir kita technika, vadinama matmenų mažinimu (angl. *dimension reduction*). Ji remiasi prielaida, kad algoritmo vykdymo spartai ir sudėtingumui įtakos turi ne tik analizuojamame rinkinyje esančių duomenų skaičius, bet ir jų požymių (angl. *feature*) kiekis [SAW+14]. Požymių skaičiui didėjant, algoritmo vykdymo laikas ilgėja, o sudėtingumas (angl. *complexity*) išauga. Šią problemą galima spręsti pasitelkus duomenų projekcijos (angl. *projection*) analizę, kuomet daugelio matmenų (angl. *high-dimensional*) duomenys yra pateikiami mažesnio matmenų skaičiaus (angl. *lower-dimensionality*) erdvėje. Vienas iš būdų tokiems rezultatams

gauti, yra klasterizavimas duomenų požymių poaibyje, kuomet panašius klasterius siekiama rasti analizuojant keletą mažesnių duomenų požymių rinkinių. Tokios technikos naudojimas grindžiamas prielaida, jog analizuojant didelį požymių kiekį turinčius duomenis, nemažai jų nėra itin reikšmingi galutiniams rezultatams ir panašūs klasteriai gali būti gaunami atliekant klasterizavimą skirtingose erdvėse, atspindinčiose nagrinėjamus duomenų požymius [CLL17].

Matmenų mažinimo technikai naudojamas ne tik atsitiktinis klasterių analizėje nagrinėjamų požymių parinkimas. 2014 metų darbe, paskelbtame Shirkhorshidi ir kitų autorių [SAW+14], siūloma duomenis ir jų požymius išreikšti matrica ir klasterizavimo metu nagrinėti transformuotą jos versiją, kurioje kiekvienas naujas elementas atspindėtų apytikslę originalios matricos nario reikšmę (angl. *approximation*), išreikštą mažesnių matmenų erdvėje. Vis dėlto, šis būdas kelia papildomus iššūkius vykdant duomenų tyrybą, kadangi tokiam metodui reikalinga tinkama funkcija, kuri išlaikytų kaip įmanoma panašesnį santykinį atstumą tarp analizuojamoje aibėje esančių duomenų ir galėtų rasti minimalų skirtumą $\|A' - A\|$ tarp pradinės matricos A ir jos transformuotos versijos A'.

2.4.2. Klasterizavimas daugelyje įrenginių

Didžiųjų duomenų klasterių analizei gali būti naudojama ir klasterizavimo daugelyje įrenginių (angl. *multiple-machine clustering*) technika, kurios algoritmų vykdymui gali būti panaudoti daugiau nei vieno įrenginio skaičiavimų bei atminties resursai [BGB18]. Praktikoje ši technika naudojama dažniau nei klasterizavimas viename įrenginyje, kadangi naudotojams gali pasiūlyti didesnes plečiamumo galimybes ir geresnį atsako laiką (angl. *response time*) [CLL17].

Išaugus surenkamų duomenų kiekiui, klasterizavimo daugelyje įrenginių reikšmė duomenų tyrybai stipriai išaugo, kadangi šis duomenų kiekio augimas buvo daug spartesnis nei pažanga kompiuterinės atminties ar procesorių srityje. Dėl šios priežasties atsirado poreikis ieškoti būdų kaip klasterizavimui panaudoti didesnius nei vieno įrenginio procesoriaus ir atminties pajėgumus. Kaip teigia Chen, Ludwig ir Li savo 2017 metų straipsnyje [CLL17], tokiu būdu duomenis galima saugoti ir analizuoti skirtinguose įrenginiuose, tam išnaudojant didesnius sistemos skaičiavimo pajėgumus.

Klasterių analizė ne viename įrenginyje gali būti atliekama vykdant lygiagretųjį klasterizavimą (angl. *parallel clustering*). Šiai technikai vykdyti gali būti naudojami įprasti lygiagretaus programavimo modeliai, kuriuose duomenys paskirstomi į atskiras dalis ir lygiagrečiai apdorojami daugelio procesorių ar jų branduolių, naudojantis tokiomis technologijomis kaip OpenMP ar MPI [CLL17]. Vis dėlto, nors tokia technika ir gali ženkliai padidinti duomenų tyrybos pajėgumus, lygiagretaus klasterizavimo įgyvendinimas yra gana sudėtingas dėl poreikio analizę vykdantiems specialistams rūpintis ne tik duomenų ir apkrovos paskirstymu (angl. *load balancing*), atsparumu gedimams (angl. *fault tolerance*) [SAW+14], bet ir gautų rezultatų sinchronizavimu [BKL16].

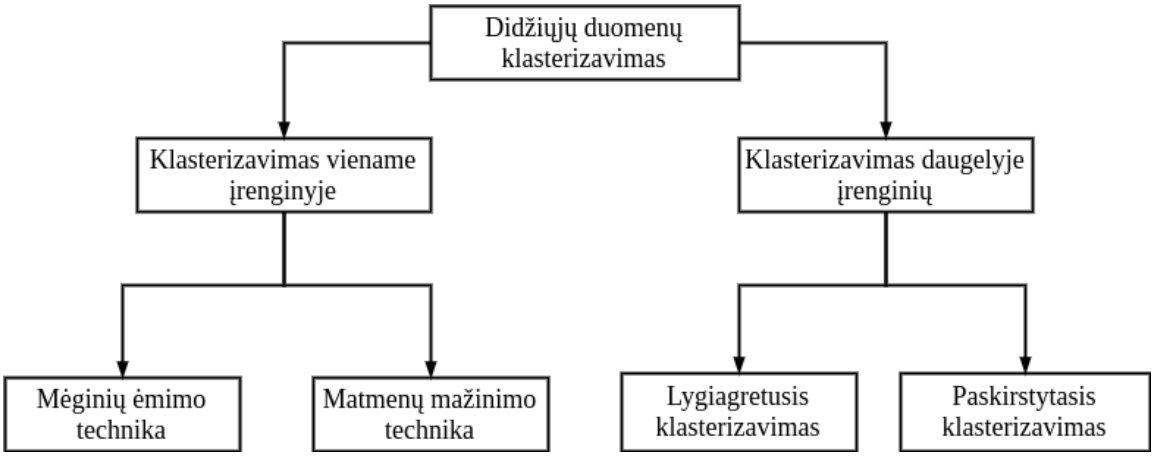
Klasterių analizei, kuri atliekama daugelyje įrenginių, taikoma ir kita technika, vadinama paskirstytuoju klasterizavimu (angl. *distributed clustering*). Vienas iš jos teikiamų privalumų yra tas, kad vykdant tokio pobūdžio duomenų tyrybą, naudojami modeliai ir sistemos abstrahuoja algoritmų lygiagretaus vykdymo detales ir leidžia specialistams susikoncentruoti į pačią duomenų

analizę. Paskirstytosios sistemos (angl. *distributed systems*) taip pat leidžia išvengti didelių kaštų, susijusių su poreikiu prižiūrėti ir išlaikyti superkompiuterius, reikalingus efektyviems lygiagretiesiems skaičiavimams. Šių sistemų dėka klasterių analizę galima atlikti tarpusavyje sujungtų įprastų kompiuterių tinkle. Tokiame tinkle esantys įrenginiai gali būti skirtingi ir vykdyti atskirus skaičiavimus. Be to, jų skaičius gali kisti dinamiškai, todėl atsiranda galimybė esant poreikiui naudoti didesnius skaičiavimų ar atminties resursus bei juos taupyti, kai tokio poreikio nėra [BKL16].

Viena iš pagrindinių paskirstytųjų sistemų funkcijų yra klasterizavimo darbų paskirstymas skirtingiems skaičiavimus atliekantiems sistemos mazgams (angl. *system node*), kuriuo siekiama sumažinti bendrą klasterių analizei reikalingo laiko kiekį ir pagreitinti naudotojo laukiamo rezultato gavimą. Toks apkrovos skirstymas gali būti vykdomas tarp mazgų nuolat dalijantis žinutėmis, skirtomis darbų sinchronizavimui, arba šiuos mazgus padalijus į šeimininko (angl. *master*) ir darbininko (angl. *slave*) kategorijas ir atliekant dviejų dalių klasterizavimą, kai išskirstytuose darbininkų mazguose yra atliekama lokali klasterių analizė, o jos rezultatai yra panaudojami atliekant globalųjį klasterizavimą šeimininko mazge [BKL16]. Vis dėlto, vykdant paskirstytą didelio kiekio duomenų analizę, išryškėja pakankamai rimtas tokios technikos trūkumas. Paskirstytųjų sistemų naudojamas komunikavimas tarp skirtingų jos dalių gali ženkliai sulėtinti veikimą, kadangi didžiąją algoritmo vykdymo laiko dalį reikia paskirti ne tik darbų sinchronizavimui, bet duomenų perkėlinėjimui iš vieno mazgo į kitą [CLL17].

Siekiant sumažinti tinklo apkrovą ir pagreitinti didžiųjų duomenų tyrybą klasterizavimo metodais Bendechache, Kechadi ir Le-Khac 2016 metais publikuotame straipsnyje [BKL16] pasiūlė atlikti dviejų žingsnių klasterių analizę, sudarytą iš lygiagrečiai paskirstytoje sistemoje atliekamo lokalių klasterių radimo ir jų suliejimo vykdant globalų klasterizavimą centriniame sistemos mazge. Bendechache, Kechadi ir Le-Khac sugebėjo tinklu perduodamų duomenų apimtį sumažinti 98 procentais globaliam klasterizavimui perduodant tik lokaliai rastų klasterių atstovus (angl. *representative*), parodančius šių duomenų grupių centrus ir jų kraštinių kontūrus. Be to, vykdant tokio pobūdžio duomenų analizę K-vidurkių ir DBSCAN algoritmais, naudotojui nereikia iš anksto pateikti ieškomo duomenų klasterių kiekio K, nes antrame metodo žingsnyje vykdomo galutinių rezultatų surinkimo metu, šis skaičius randamas automatiškai pagal lokaliai rastų duomenų grupių susiliejimą globalioje erdvėje. Globalaus klasterizavimo žingsnio metu, klasteriai yra sujungiami, kai persidengia jų atstovų išsidėstymu nurodomi kontūrai.

Klasterizavimo viename įrenginyje ir daugelyje įrenginių algoritmų grupei priklausančios technikos atitinka hierarchiją, kurios vizualizacija pateikta 6 paveikslėlyje [CLL17].



6 pav. Didiesiems duomenims naudojamų technikų hierarchija

2.5. Pritaikymas praktikoje

Klasterizavimas gali būti naudojamas duomenų analizei, kai nėra iš anksto žinomų duomenų grupių. Suradus natūralų objektų pasiskirstymą pagal jų panašumą, šioms grupėms galima priskirti identifikatorių ir naudoti jas kaip modelį duomenų klasifikavimui. Be to, jei analizės metu randama duomenų, kurie itin skiriasi nuo klasteriuose esančių elementų, juos galima vertinti kaip anomalijas imtyje ir atitinkamai į jas reaguoti [Sur17].

Klasterių analizė svarbi ir neuroninių tinklų, dirbtinio intelekto srityse. Ja galima aptikti pasikartojančius duomenų modelius (angl. *patterns*), apdoroti vaizdinę medžiagą ir pritaikyti mašiniame matyme (angl. *machine vision*) [BT14]. Biologijoje ši duomenų tyrybos metodika naudojama skirstyti organizmus į rūšis, analizuoti genomą ir ieškoti genų grupių sąsajų su organizmo savybėmis ar ligomis. Versle klasterizavimo būdu randamas bendrus interesus turinčių žmonių pasiskirstymas, kuris tampa itin vertingas potencialiems klientams teikiant geresnius savalaikius pasiūlymus bei aktualesnę informaciją.

3. Klasifikavimas

3.1. Apibrėžimas ir uždavinys

Informacijos gavimui analizuojant duomenis gali būti taikomi įvairūs duomenų tyrybos metodai. Vienas iš dažniau naudojamų duomenų apdorojimo metodų yra vadinamas klasifikavimu (angl. *classification*), kurio metu duomenys yra priskiriami tam tikroms tikslinėms grupėms (angl. *target groups*), dar kitaip vadinamoms klasėmis. Vykdant tokią analizę siekiama nustatyti kuriai iš galimų klasių gali priklausyti objektas iš naujai nagrinėjamų duomenų aibės [Sur17].

Viena iš pagrindinių klasifikavimo paskirčių yra sąryšių radimas tarp žinomos informacijos savybių ir naujai analizuojamų duomenų. Aggarwal 2014 metais išleistoje knygoje [Agg14] rašė, kad duomenų klasifikavimo uždavinys yra „turint aibę mokymo duomenų charakteristikų su priskirtomis mokymo etiketėmis, nustatyti klasės etiketę jos neturinčiam testuojamam objektui". To siekiama algoritmais, kuriuos įprastai sudaro dvi dalys: mokymo fazė (angl. *training phase*), kurios metu iš jau žinomų, identifikuotų duomenų konstruojamas analizės modelis ir testavimo fazė (angl. *testing phase*), kuri naudoja sukonstruotą modelį priskirti testuojamus duomenis potencialioms klasėms [Agg14]. Šie modelių rinkiniai dar kitaip gali būti vadinami klasifikatoriais (angl. *classifiers*). Jų rinkinius suformavus mokymo metu, algoritmai pateikia nuo klasifikatorių priklausančius rezultatus [TLC+15].

3.2. Algoritmai

Klasifikavimo algoritmai analizuojamam duomenų objektui gali priskirti vieną konkrečią klasę arba skaitinį įvertinimą, parodantį kiek duomuo atitinka kiekvieno analizėje naudojamo klasifikatoriaus kriterijus. Tokio pobūdžio vertinimu galima rasti analizuojamos medžiagos tapatumą su svarbesnės grupės kriterijais, kai modelio klasifikatorių svarba yra nelygiavertė. Be to, skaitiniai rezultatai gali būti naudojami identifikuoti elementus, labiau atitinkančius retas klases, kai jos sunkiai aptinkamos, bet yra itin vertingos. Nepaisant to, esant poreikiui klasifikuojamus duomenis griežtai suskirstyti į grupes, skaitinį vertinimą galima gana lengvai transformuoti į diskrečios klasės (angl. *discrete class*) priskyrimą, nagrinėjamam objektui priskiriant grupę, atitinkančią klasifikatorių su maksimalia rasta verte [Agg14].

Vykdant algoritmus, mokymo fazės metu, klasifikavimo modelis gali būti pritaikomas taip, kad analizė rastų duomenis, labiausiai atitinkančius iš anksto apibrėžtas ieškomas savybes. Tai vienas ryškiausių klasifikavimo skirtumų kitai duomenų tyrybos metodų grupei, vadinamai klasterizavimu, kurio metu duomenys grupuojami pagal naujos įvesties tarpusavio panašumą [TLC+15]. Klasių priskyrimo atveju analizavimo modelis suformuojamas dar prieš pradedant nagrinėti duomenis. Jį apibrėžti bei koreguoti gali ir žmogus pagal tai, kokių rezultatų tikimasi ar kokios informacijos ieškoma. Dėl šios priežasties klasifikavimas priskiriamas prižiūrimam mokymuisi (angl. *supervised learning*), kuris leidžia nagrinėjant duomenis rasti naudingą informaciją, išskiriant svarbesnias ieškomas savybes [Agg14].

Algoritmai skiriasi ir pagal tai, kaip vyksta apdorojamų duomenų skirstymas į klases. Išskiriamos dviejų klasių (angl. *binary classification*) ir daugelio klasių (angl. *multiclass*) analizės rūšys. Dviejų klasių klasifikavimas yra daug paprastesnis, tačiau suteikia ne tiek daug informacijos kiek daugelio klasių analizės metu. Vis dėlto daugelis sudėtingesnių algoritmų remiasi konceptais, suformuotais dviejų pasirinkimų analizei [Sur17].

3.2.1. Naivaus Bajeso algoritmas

Duomenų klasifikavimui gali būti naudojami tikimybiniai (angl. *probabilistic*) analizės metodai, kurie testuojamam objektui tinkamiausią klasę suranda pagal statistinius skaičiavimus. Tokio pobūdžio algoritmai gali pateikti po skaičiavimo rastą tikimybę (angl. *posterior probability*), kuri rodo klasifikavimo skaitinį įvertį, kiek duomuo atitinka kiekvienos modelio klasės savybes. Radus tokias tikimybes, priskyrimas klasėms atliekamas pritaikant tinkamą sprendimų strategiją [Agg14]. Vienas iš tokių klasifikavimo metodų yra vadinamas Naivaus Bajeso (angl. *Naive Bayes*) algoritmu, paremtu sąlyginėmis tikimybėmis (angl. *conditional probabilities*) ir Bajeso teorema, kuri tikimybes randa skaičiuojant reikšmes ir jų kombinacijų dažnumą istoriniuose duomenyse [Sur17]. Naivus Bajeso algoritmas aprašytas 2014 metų Aggarwal knygoje [Agg14]:

- jei testuojamas duomuo *T* turi *d* skirtingų požymių su *d* reikšmių ir norima nustatyti, kad jo tikimybė priklausyti klasei *Y(T)* yra *i*, tuomet ši priklausoma tikimybė (angl. *posterior probability*) žymima $P(Y(T) = i \mid x_1...x_d)$;
- skaičiavimams pritaikoma Bajeso teorema ir gaunama formulė:

$$P(Y(T) = i \mid x_1...x_d) = P(Y(T) = i) \cdot \frac{P(x_1...x_d \mid Y(T) = i)}{P(x_1...x_d)};$$

- dešinės skaičiavimų pusės supaprastinimui panaudojama naivi Bajeso prielaida ir skaičiavimai išreiškiami kaip duomens požymių sąlyginių tikimybių produktas:

$$P(x_1...x_d \mid Y(T) = i) = \prod_{j=1}^{d} P(x_j \mid Y(T) = i).$$

Algoritmas laikomas naiviu, nes remiasi prielaida, kad kiekviena analizuojamo objekto savybės tikimybė yra nepriklausoma nuo kitų. Nors realiame pasaulyje dažnai pasitaiko atvejų, kuomet tai nėra tiesa, praktikoje ši prielaida nedaro itin didelės įtakos metodo klasifikavimo tikslumui, tačiau jį padaro efektyvesniu ir patrauklesniu klasifikuojant didesnį kiekį sudėtingesnės struktūros duomenų [Sur17]. Supaprastinimas klasifikavimo modelį leidžia efektyviau pritaikyti nagrinėjant tikimybes analizės metu. Be to, mokymo fazės metu nesant pakankamam kiekiui duomenų apskaičiuoti tam tikros klasifikavimo savybės tikimybę, Naivaus Bajeso algoritmas kompensuoja šias reikšmes.

3.2.2. Sprendimų medžio algoritmas

Kai duomenų analizės metu yra svarbu žinoti kokie kriterijai buvo taikyti nagrinėjamą objektą priskiriant vienai iš modelio klasių, itin naudingu tampa sprendimų medžio (angl. *decision tree*) algoritmas. Jis generuoja taisykles, kurios transformuojamos į sąlyginius sakinius (angl. *conditional statements*) ir gali būti suprantamos žmogui, norinčiam išsiaiškinti analizės vykdymo metu taikomus pasirinkimus. Dalykinės srities ekspertai šią savybę gali panaudoti validuodami modelį ir sudarydami detalų klasifikatorių aprašymą, nurodantį kokias savybes turintys objektai buvo priskirti tam tikroms klasėms. Toks modelio skaidrumas (angl. *model transparency*) ne tik suteikia detalesnį jo aprašymą, bet ir padeda lengviau nuspėti kokioms grupėms bus priskirti naujai nagrinėjami elementai [Sur17].

Sprendimų medžio būdu duomenų analizė vykdoma hierarchiškai. Kiekviename struktūros lygyje duomenų rinkinio padalijimas vykdomas naudojant skilimo kriterijų (angl. *split criterion*), priklausantį nuo vieno arba kelių bendrai nagrinėjamų duomens požymių. Mokymo fazės metu objektai rekursiškai dalinami taip, kad jie kuo labiau pasiskirstytų tarp skirtingas klases reiškiančių medžio viršūnių. To siekiama ieškant didžiausio skirtingų elementų klasių asimetriškumo (angl. *skew*), esančio vienoje sprendimų medžio viršūnėje. Šiam rodikliui surasti praktikoje naudojami gini indekso ir entropijos įverčiai [Agg14].

Jeigu medžio viršūnėje N yra k skirtingų klasių ir p kiekis analizuojamų duomenų, tai gini indeksas $G(N)$ ir entropija $E(N)$ apskaičiuojami [Agg14] pateiktomis formulėmis:

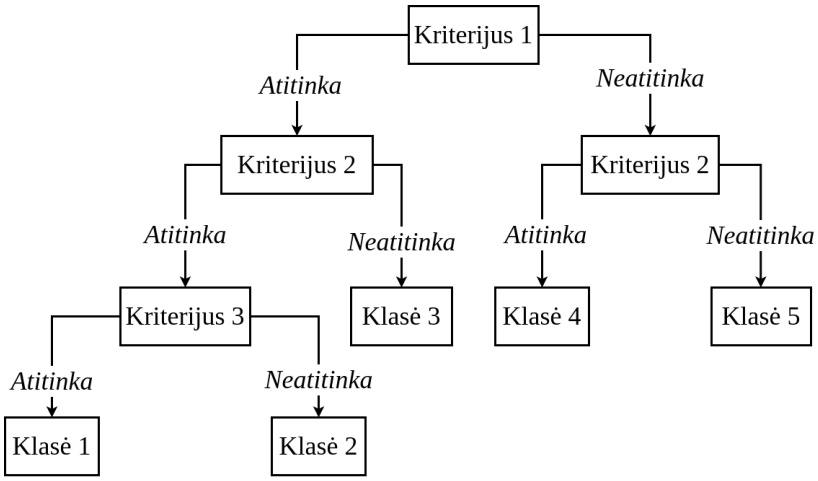
$$G(N) = 1 - \sum_{i=1}^{k} p_i^2;$$

$$E(N) = -\sum_{i=1}^{k} p_i \cdot \log(p_i).$$

Kuo mažesni gini indekso $G(N)$ ir entropijos $E(N)$ rodikliai, tuo didesnis yra klasių asimetriškumas medžio viršūnėje, reiškiantis geresnį konstruojamo modelio klasių padalijimą sprendimų medyje.

Dar viena metrika apibrėžti skilimo kriterijų, naudojamą klasifikavimo modelyje, vadinama homogeniškumu arba grynumu (angl. *purity*). Ši savybė apibrėžia tai, kaip gerai medžio viršūnės vaikus sudarančios elementų grupės atitinka tą pačią modelio klasę [Sur17]. Grynumas bei asimetriškumas naudojamas ir nustatant algoritmo mokymosi sustabdymo kriterijų, kuris reikalingas norint išvengti klasifikavimo modelio per didelio pritaikymo mokymo duomenims, dar vadinamo persimokymu (angl. *overfitting*). Vis dėlto dažnai nėra aišku kada sprendimų medžio augimą reikia stabdyti, todėl įprastai taikoma genėjimo (angl. *pruning*) procedūra, kai skirtingas klases atitinkantys medžio lapai sujungiami į vieną tikrinant skirtingose viršūnėse esančių elementų tarpusavio skirtumus arba išbandant tokio veiksmo įtaką modelio tikslumui [Agg14].

Testavimo fazės metu sprendimų medžio analizuojamas objektas priskiriamas tikslo klasei, vertinant jo atitikimą sekai hierarchiškai išdėstytų kriterijų, suformuotų metodo mokymo fazės metu [Sur17]. Tokio struktūros pavyzdys parodytas 7 paveikslėlyje.



7 pav. Sprendimų medžio modelis su klasifikavimo kriterijais

3.2.3. K-artimiausių kaimynų algoritmas

Vienas iš paprasčiausių duomenų klasifikavimo metodų yra vadinamas K-artimiausių kaimynų (angl. *K-Nearest Neighbors*) algoritmu. Tai tingus klasių analizės būdas, kuriame nėra aiškiai atskirtos modelio mokymo ir testavimo fazės. Vietoj to, lokalus klasifikavimo modelis yra sukuriamas atskirai kiekvienam nagrinėjamam duomenų objektui jo analizės metu.

K-artimiausių kaimynų algoritmas, analizuojant duomenų objektą, nustato k arčiausiai esančių modelio elementų su jau žinomomis klasėmis ir pritaikius pasirinktą funkciją, parenka kokią klasę priskirti nagrinėjamam duomeniui. Vienas iš tokių funkcijų pavyzdžių gali būti klasės priskyrimas pagal tai, kokios klasės elementai tarp k kiekio duomens kaimynų sudaro daugumą. Toks būdas vadinamas daugumos pasirinkimu (angl. *majority vote*) ir tinka, kai visi klasifikatoriai yra vienodai svarbūs vykdant analizę. Jei aptikti kai kurias klases yra svarbiau negu kitas, pavyzdžiui ieškant neatitikimų normai medicininiuose duomenyse, tuomet gali būti naudojami klasifikatoriai su svoriais ir šį matmenį grupės parinkimui įtraukiančios funkcijos [Agg14].

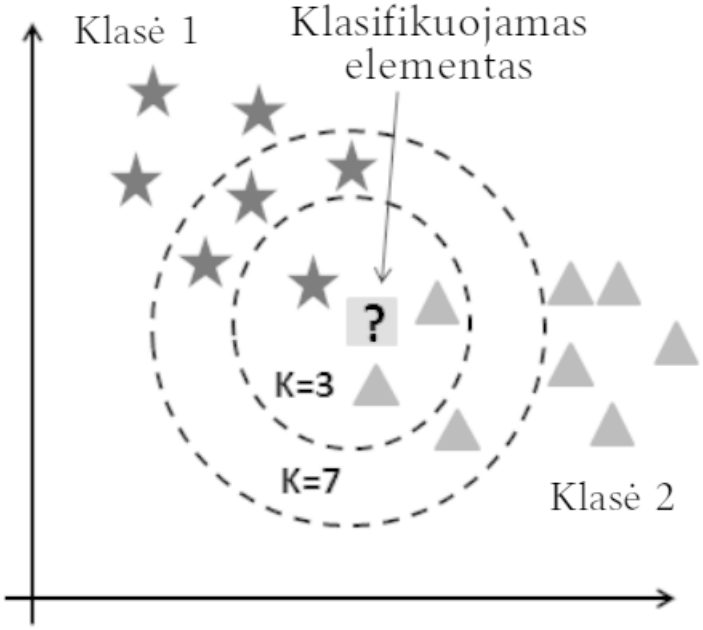
Kaimynų parinkimas duomenų klasifikavimo metu atliekamas naudojant pasirinktą elementų atstumo funkciją. Tai gali būti klasterizavimo skyriuje aprašytas Euklido atstumo skaičiavimas arba kita pagal ieškomų rezultatų reikalavimus parinkta metodika. Tokiu būdu duomenys yra suskirstomi į vienoje erdvėje pasiskirsčiusius elementų rinkinius. Jei klasifikuojama pagal daugiau nei vieną savybę, tokiu atveju kaimynystę apibrėžiantys atstumų įverčiai skaičiuojami su proporcija. Vis dėl to, kaip rašoma 2014 metų Aggarwal knygoje [Agg14], tokių skaičiavimų metu randamos tik sferos formos kaimynystės. Dėl šios priežasties algoritmas sunkiai identifikuoja retų klasių elementus ir nesubalansuotų duomenų rinkinių narius. Dažnai tokie objektai tiesiog laikomi pašaliniais (angl. *outliers*) ir nėra priskiriami jokiai klasei.

Nors K-artimiausių kaimynų algoritmas gali būti panaudotas daugumai duomenų tipų, kurių skirtumą galima apskaičiuoti atstumo funkcija, šio klasifikavimo modelio konstravimas anali-

zės metu tampa gana ryškiu trūkumu, kai duomenų yra daug arba nagrinėjamas didesnis kiekis objekto požymių. Taip yra dėl to, kad norint rasti *k* artimiausių kaimynų, reikia įvertinti visus erdvėje jau esančius elementus, atliekant skaičiavimus su jų turimomis charakteristikomis. Siekiant kompensuoti šį trūkumą gali būti kuriami duomenų panašumo indeksai arba klasifikavimui parenkami ne visų duomens požymių poaibiai [Agg14].

Vykdant K-artimiausių kaimynų algoritmą, kaip ir kitus pavyzdžiais paremtus (angl. *instance based*) klasifikavimo metodus, sudaromas modelių koncepto aprašymas (angl. *concept description*), kuris parodo koks yra sąryšis tarp rastų klasių ir jiems priskirtų duomenų. Be to Aggarwal darbe [Agg14] nurodyta, jis gali būti naudojamas ir optimizuojant algoritmo veikimą iš anksto paruošus ieškomų klasių modelį.

K-artimiausių kaimynų algoritmu rastų klasių vizuali reprezentacija gali būti panaši į parodytą 8 paveikslėlio[7] pavyzdyje.



8 pav. Klasifikavimo pavyzdys K-artimiausių kaimynų algoritmu

3.3. Algoritmų rezultatų vertinimai

Klasifikavimo algoritmų rezultatams įvertinti gali būti taikomi įvairūs metodai, naudojantys duomenis su jau žinomomis klasėmis. Dažniausiai tai yra atskira dalis mokymo aibės duomenų, kurie nėra naudojami klasifikavimo modelio formavimui ir gali būti laikomi auksiniu standartu, kurį norima pasiekti klasių analizės metu. Tokie poaibiai išimami iš mokymo aibės ir visiškai nenaudojami klasifikavimo modelių formavime [Agg14].

[7] Klasifikavimo pavyzdys k arčiausių kaimynų algoritmu adaptuotas iš `https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn`

3.3.1. Klasifikavimo matrica

Įvertinti algoritmų rezultatus, naudojant testinius duomenis, galima naudojant klasifikavimo matricą (angl. *confusion matrix*), kurioje pateikiama informacija apie tai, kiek klasifikavimo metu buvo rasta kiekvienos klasių elementų, ir kiek testiniuose duomenyse kiekvienos klasės narių buvo iš tikrųjų. Šios reikšmės klasifikavimo matricoje atitinkamai vadinamos spėtomis (angl. *predicted*) ir tikromis (angl. *actual*) [BT14].

Turint binarinį klasifikavimo modelį, matricoje pasirenkama viena iš dviejų klasių ir tikrinama kiek objektų analizės metu tikrinamai klasei buvo priskirti, o kiek – ne. Kaip nurodyta Burnay ir Tarig 2014 metų darbe [BT14], jei algoritmo buvo nustatyta, kad duomuo priklauso klasei ir tokią pat klasę objektas turi testinėje aibėje, rezultatas pažymimas kaip teisingai priskirtas TP (pagal angl. *true positive*). Kai duomuo klasei nepriskiriamas nei algoritmo išvestyje, nei testinėje aibėje, jam naudojamas TN žymėjimas (pagal angl. *true negative*). Jeigu remiantis testiniais duomenimis klasifikavimas klaidingai nustatė, kad duomuo priklauso klasei, naudojama reikšmė FP (pagal angl. *false positive*), jeigu klaidingai nepriklauso – FN (pagal angl. *false negative*). Tokios reikšmės sudaro klasifikavimo matricos modelį, pavaizduotą 1 lentelėje.

1 lentelė. Klasifikavimo matricos modelis

		Tikros	
		Priklauso	Nepriklauso
Spėtos	Priklauso	TP	FP
	Nepriklauso	FN	TN

Naudojant klasifikavimo matricą, tampa įmanoma vertinti algoritmų rezultatų tikslumą (angl. *accuracy*), kuris žymimas *ac* ir randamas pagal [BT14] pateiktą formulę:

$$ac = \frac{TP + TN}{TP + TN + FP + FN}.$$

Siekiant apskaičiuoti klasifikavimo specifiškumą (angl. *specificity*), rodantį teisingai nepriskiriamų elementų proporciją *s*, naudojama formulė:

$$s = \frac{TN}{TN + FP}.$$

3.3.2. F-įvertis

Dar vienas iš klasifikavimo vertinimo metodų pavyzdžių gali būti F-įvertis (angl. *F-measure*), kuris parodo kiek elementų klasėje yra narių, kurie neturėtų jai priklausyti, ir kiek elementų, nesančių grupėje, turėtų būti jai priskirti. Šį įvertį galima gauti panaudojus precizijos *p* (angl. *precision*) ir jautrumo *r* (angl. *recall*) matus:

$$p = \frac{TP}{TP + FP};$$

$$r = \frac{TP}{TP + FN}.$$

Apskaičiavus precizijos ir jautrumo reikšmes, F-įvertis gaunamas pagal formulę, kuri pateikta [TLC+15]:

$$F = \frac{2pr}{p+r}.$$

3.4. Didžiųjų duomenų klasifikavimas

Klasifikavimas gali būti naudojamas ir ieškant vertingos informacijos didžiuosiuose duomenyse. Vis dėlto realiame pasaulyje surenkami didieji duomenys pasižymi ne tik išskirtiniu jų dydžio, bet ir turimų požymių kiekiu [Sut16]. Tai apsunkina šių duomenų apdorojimą, kuris, norint išsaugoti informacijos vertę, turi būti atliekamas pakankamai dideliu greičiu. Be to, svarbu ir tai, kad klasifikavimo vykdymo metu, gali pasikeisti ieškomų tikslinių grupių (angl. *target group*) savybės, todėl atsiranda poreikis dinamiškai keisti mokymo fazėje (angl. *training phase*) sukonstruotus ir testavimo fazėje (angl. *testing phase*) naudojamus klasifikatorius (angl. *classifier*). Dėl šių priežasčių dauguma tradicinių klasifikavimo algoritmų ir jų rezultatų vertinimo metodų nėra visiškai pritaikyti didžiųjų duomenų analizei.

K-artimiausių kaimynų algoritmas neturi aiškiai atskirtų mokymo ir testavimo fazių. Jis kiekvienos iteracijos metu konstruoja klasifikavimo modelį, todėl nors ir gali prisitaikyti prie nagrinėjamos srities pasikeitimų, tačiau, esant dideliam kiekiui duomenų, yra pakankamai neefektyvus laiko prasme. Tuo tarpu sprendimų medžio algoritmas testavimo fazės metu analizuojamus duomenis klasėms priskiria gana greitai, bet jo mokymo fazės metu sukonstruoti modeliai yra statiniai ir nekintantys pagal tai, kokie duomenys yra nagrinėjami testavimo metu. Su panašiomis problemomis susiduriama ir vykdant Naivaus Bajeso algoritmą, kurio mokymo ir testavimo fazės taip pat yra atskirtos.

Nagrinėjant didžiuosius duomenis, analizuojama medžiaga gali būti vertinama kaip nenutrūkstantis duomenų srautas (angl. *stream*). Be to, praktikoje pasitaiko atvejų, kad analizuojant didelius duomenų kiekius gali pasikeisti ir ieškomų klasių savybės. Dėl to atsiranda poreikis testavimo fazės metu turėti galimybę dinamiškai keisti klasifikavimo modelį. Daugumos tradicinių klasifikavimo algoritmų vykdymo metu jis suformuojamas mokymo fazėje ir testuojant naujai pateiktą įvestį yra nekintantis. Vis dėlto literatūroje yra siūlomi algoritmų pakeitimai, sprendžiantys šį klasių analizėje pasitaikantį trūkumą. Vienas iš jų yra testavimo metu klasifikuoto duomens panaudojimas toliau mokyti klasifikavimo metu naudotą modelį. Tokiu būdu algoritmas gali dinamiškai prisitaikyti prie nagrinėjamoje srityje atsiradusių pokyčių. Taip pat vienas iš tokio metodo privalumų yra tas, kad taip gali būti analizuojami srautu pateikiami duomenys, kadangi algoritmas elemento klasifikavimą gali įvykdyti bet kuriuo metu pagal tai, kokie duomenys klasių analizei iki tol buvo pateikti ne tik mokymo, bet ir testavimo fazės metu [Mur13].

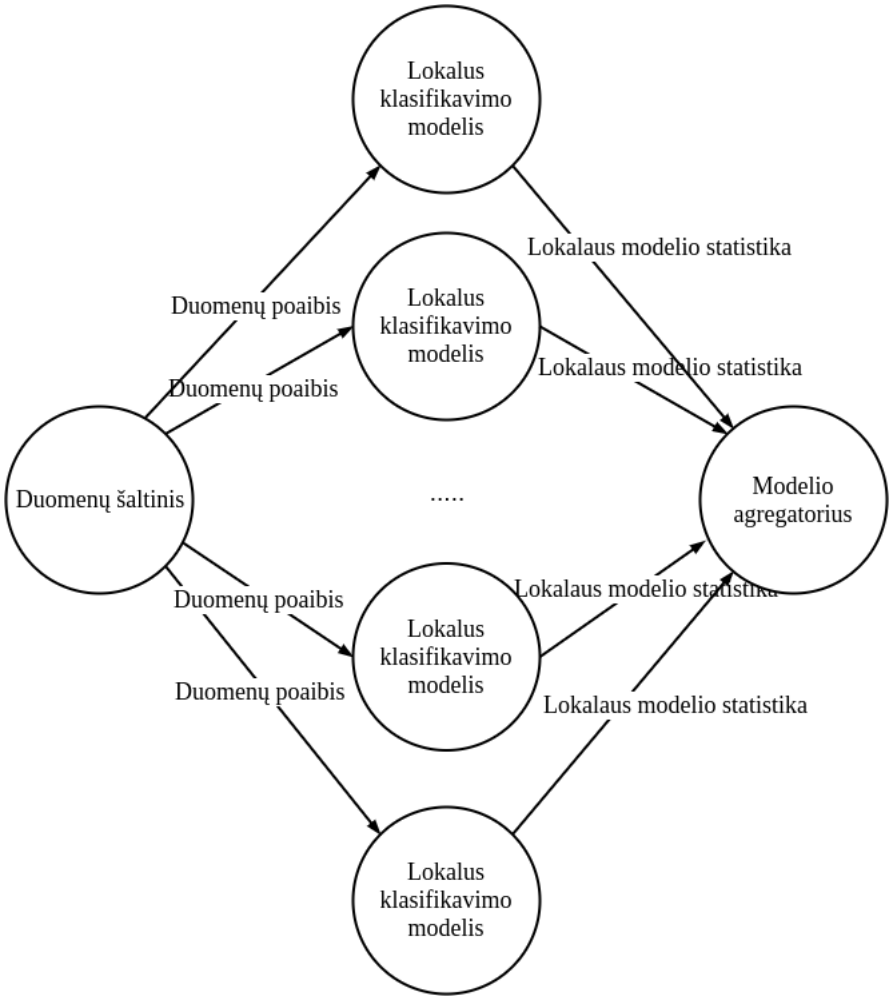
Siekiant klasifikavimo algoritmus pritaikyti didiesiems duomenims, kitas iš literatūroje siūlomų būdų yra lygiagretus jų vykdymas. Šis metodas gana efektyvus vykdant klasifikavimą Naivaus Bajeso algoritmu ir gali pademonstruoti gerą tikslumą ir pralaidumą (angl. *throughput*) net ir analizuojant itin didelius kiekius duomenų įrašų [LBC+13]. Tokio klasifikavimo metu remiamasi naivia prielaida, jog kiekviena duomens tikimybė priklausyti tam tikrai klasei yra nepriklausoma nuo kitų tikimybių. Dėl šios priežasties atsiranda galimybė jas apskaičiuoti skirtinguose įrengi-

niuose ir taip ženkliai sumažinti algoritmo vykdymo laiką. Be to, kaip teigiama De Fortuny, Martens ir Provost 2013 metais publikuotame straipsnyje [DMP13], realiomis sąlygomis surenkami didieji duomenys požymių prasme gali būti gana reti (angl. *sparse*). Jų siūlomame sprendime siekiama išnaudoti šią didžiųjų duomenų ypatybę ir klasifikuojant duomenis Naivaus Bajeso algoritmu sutaupyti nemažai skaičiavimo resursų analizuojant tik egzistuojančius duomens požymius. Pasak De Fortuny, Martens ir Provost, klasifikavimo metu tai gali suteikti galimybes ištirti didesnį kiekį duomenų ir taip pagerinti vykdomos analizės rezultatų tikslumą. Šiuo atveju remiamasi prielaida, jog tokiose didžiuosius duomenis naudojančiose verslo ar mokslo srityse, kaip žmonių elgsenos tyrimai, kiekvienas tiriamas subjektas turi galimybes pademonstruoti tik dalį tiriamųjų požymių ir siekiant išplėsti (angl. *scale up*) klasių analizės mastą, tiriant tik poaibį aktualių požymių, gauti rezultatai gali būti netikslūs. Dėl šios priežasties Naivaus Bajeso klasifikavimo metu tikslingiau yra tirti kiekvieno duomens demonstruojamus požymius didesniame duomenų rinkinyje [DMP13].

Lygiagretus vykdymas gali būti taikomas ir klasifikuojant duomenis sprendimų medžio algoritmu. Tokiu būdu padidinama mokymo fazės sparta ir plečiamumas, kadangi klasifikavimo modelio formavimas atliekamas paskirstytai ir turi galimybes panaudoti didesnius skaičiavimo ir atminties pajėgumus. Sprendimų medžio algoritmą vykdant lygiagrečiai formavimas atliekamas dviem žingsniais ir pirmo žingsnio metu formuojamas lokalus sprendimų medis (angl. *local decision tree*), o bendrą klasifikavimo modelį atspindintis globalus sprendimų medis (angl. *global decision tree*) atnaujinamas tik periodiškai sinchronizuojant jį su iki to suformuotais lokaliais modeliais [Mur13]. Klasifikavimo mokymo fazės lygiagretinimui galima naudoti skirtingus būdus, iš kurių tris Murdopo aprašė 2013 metais paskelbtame savo darbe ir įvardijo juos kaip horizontalųjį, vertikalųjį ir užduočių lygiagretinimą.

3.4.1. Horizontalusis lygiagretinimas

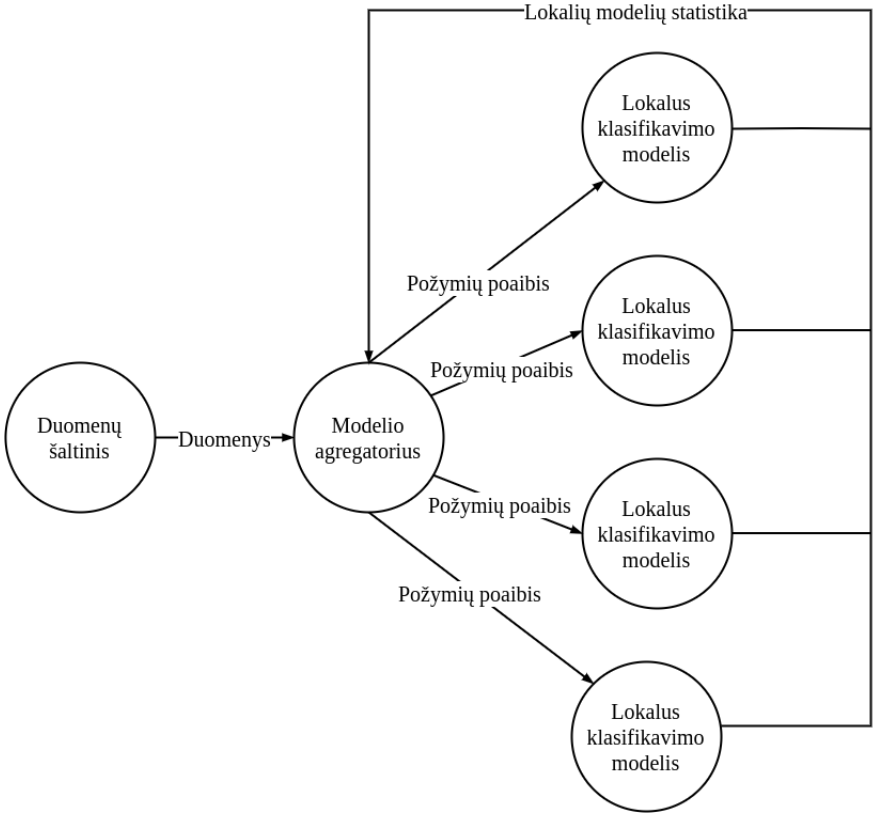
Horizontaliojo lygiagretinimo (angl. *horizontal parallelism*) atveju paskirstytasis algoritmas apmoko lokalųjį sprendimų medį tik su dalimi nagrinėjamų duomenų ir periodiškai siunčia surinktą statistiką į globalųjį sprendimų medį formuojantį mazgą (angl. *node*), vadinamą modelio agregatoriumi (angl. *model-aggregator*). Kai statistika surenkama iš visų lokalių klasifikatorių, vykdoma jos agregacija ir suformuojamas globalus sprendimų medis, pagal kurį atnaujinami ir lokalūs klasifikavimo modeliai. Tokio proceso pavyzdys pavaizduotas 9 paveikslėlyje, adaptuotame iš [Mur13]. Vykstant pasirengimui atlikti horizontaliai išlygiagretintą algoritmą, klasifikatoriams efektyviai paskirstyti duomenų poaibius galima ir šios medžiagos kiekvieną elementą iš eilės priskiriant skaičiavimus atliekantiems mazgams.



9 pav. Horizontaliai išlygiagretinto klasifikavimo vykdymo schema

3.4.2. Vertikalusis lygiagretinimas

Vertikaliu lygiagretinimu (angl. *vertical parallelism*) didiesiems duomenims pritaikytas sprendimų medžio algoritmas yra paremtas ne nagrinėjamų duomenų kiekio, o jų požymių paskirstymu. Tokiu atveju kiekvienas klasifikavimą atliekantis mazgas neformuoja lokalaus sprendimų medžio, o tik atlieka skaičiavimus su jam priskirtais duomenų požymiais ir siunčia rezultatus modelio agregatoriui, kuris ir formuoja globalų klasifikavimo modelį, panašiai kaip tai pavaizduota 10 paveikslėlyje, adaptuotame iš [Mur13]. Vertikalus lygiagretinimas tinkamas, kai analizuojami duomenys turi pakankamai didelį kiekį klases apsprendžiančių požymių, tokių kaip skirtingi žodžiai klasifikuojamame tekste.

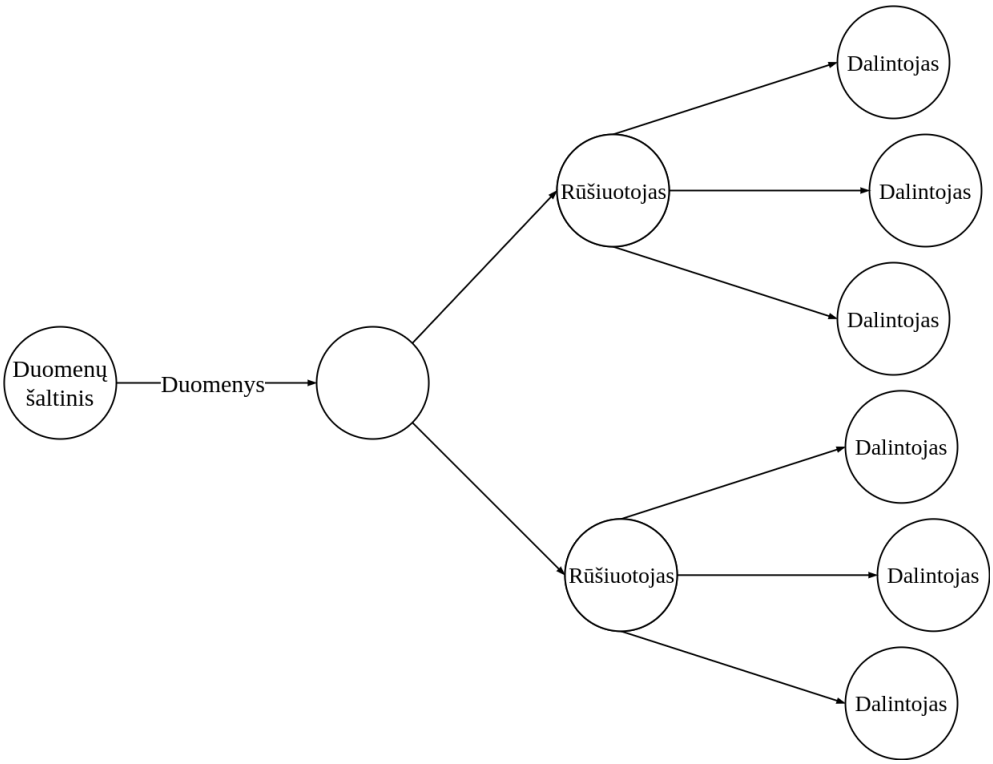


10 pav. Vertikaliai išlygiagretinto klasifikavimo vykdymo schema

3.4.3. Užduočių lygiagretinimas

Pritaikius užduočių lygiagretinimą (angl. *task parallelism*), sprendimų medžio algoritmas padalinamas į dalis, kurios gali būti vykdomos atskirai viena nuo kitos. Šias dalis gali sudaryti įvesties duomenų rūšiavimas (angl. *sorting*) ir paskirstymas klasifikatorius formuojantiems mazgams, nagrinėjamo elemento požymių analizavimas bei sprendimų medžio auginimas, kai pasiekiamos iš anksto nustatytos šakos dalijimosi (angl. *splitting*) ribos. 11 paveikslėlyje pateiktame pavyzdyje, adaptuotame iš [Mur13], matoma, kad užduočių lygiagretinimą naudojančią sistemą įprastai sudaro rūšiuotojai (angl. *sorter*) ir statistiką atnaujinantys dalintojai (angl. *updater-splitter*). Kai mokymo fazės metu į sistemą patenka naujas klasifikatorių formavimui skirtas duomuo, rūšiuotojas jį perduoda vienam iš savo pasiekiamų dalintojų. Jei nei vienas iš šių dalintojų neatitinka ieškomų duomens kriterijų, elementas perduodamas kitam rūšiuotojui. Radus tinkamą medžio

dalintoją, jis atlieka požymių analizę ir įvykdo skaičiavimus, reikalingus klasifikavimo modelio formavimui ir potencialiam sprendimų medžio auginimui. Pasiekus klasifikavimo modelio statistiką atnaujinančio dalintojo atminties ribas, šis transformuojamas į rūšiuotoją, o iki tol suformuoto sprendimų medžio lapai (angl. *leaf nodes*) tampa naujais klasifikatorius formuojančiais dalintojais.



11 pav. Užduotimis išlygiagretinto klasifikavimo vykdymo schema

3.5. Pritaikymas praktikoje

Klasifikavimą galima naudoti, kai duomenys analizės metu yra skirstomi į iš anksto žinomas grupes. Versle tokia metodika taikoma analizuojant klientų elgesį, poreikius ir atsaką į teikiamus pasiūlymus, remiantis jų asmeninėmis ir socialinėmis savybėmis. Taip pat modeliuojant verslo svarstomus sprendimus ir apskaičiuojant rizikas finansų srityje [Sur17].

Klasifikavimas taip pat taikomas medicinoje identifikuojant ligas pagal duomenis, gautus tyrimų metu, analizuojant įvykius stebėjimų metu ir ieškant neįprastos arba nusikalstamos veiklos. Klasių analizė gali būti pritaikyta tvarkant didelius kiekius dokumentų, apdorojant vaizdinę ar garsinę medžiagą, kuri turi savo semantiką, bei analizuojant žmonių veiklą internete ir socialiniuose tinkluose [Agg14].

4. Didžiųjų duomenų technologijos

Analizuojant didžiuosius duomenis dažnai naudojamos šiam tyrybos pobūdžiui skirtos technologijos. Jos įgalina duomenų specialistus analizę atlikti paskirstytame daugybės įrenginių tinkle, skiriant dėmesį ne skaičiavimų lygiagretinimui, duomenų išskaidymui ir klaidų valdymui, o tik ieškomą informaciją randančiam funkcionalumui. Tą pasiekti leidžia galingos sistemų sąsajos (angl. *interface*), kurių dėka automatiniu lygiagretinimu bei paskirstymu galima pasinaudoti net ir didelio masto kompiuterių tinkluose [DG04].

Daugumos šių technologijų tikslas – tiriant didžiuosius duomenis pasiūlyti geresnį lankstumą, plečiamumą ir greitį nei tradicinėse duomenų tvarkymo sistemose. Be to, atsiranda poreikis tvarkyti duomenis, gautus iš skirtingų šaltinių ir juos tinkamai tarpusavyje jungti (angl. *aggregate*), užtikrinant pastovų jų teisingumą, pasiekiamumą ir saugumą [OBA+18].

4.1. MapReduce programavimo modelis

Viena iš plačiausiai naudojamų didžiųjų duomenų technologijų yra MapReduce programavimo modelis (angl. *programming model*). Jis naudojamas lygiagrečiam itin didelių duomenų kiekių apdorojimui ir yra paremtas keletu konceptų, kuriuos sudaro iteravimas (angl. *iteration*) per įvestį, rakto ir reikšmės poros radimas kiekvienam įvesties elementui, tarpinių rezultatų grupavimas pagal raktą, iteravimas per rastas grupes ir kiekvienos iš šių grupių redukcija (angl. *reduction*) [Läm07].

MapReduce, kaip ir kitos didžiųjų duomenų technologijos, pateikia paprastą ir galingą sąsają, kuri leidžia didelio masto (angl. *large scale*) analizę vykdyti klasteryje įprastų, rinkoje prieinamų įrenginių [LBC+13]. Šis modelis leidžia gana paprastai išnaudoti paskirstytą ir lygiagretų užduočių vykdymą, kadangi jo realizacija (angl. *implementation*) pati pasirūpina duomenų perdavimu tinklu, apkrovos valdymu bei atsparumu gedimams (angl. *fault tolerance*). Be to, duomenų specialistai gali skaičiavimus išskaidyti į atskiras problemas, aprašytas MapReduce darbais [Läm07]. Tokie darbai susideda iš transformavimo (angl. *map*) ir redukcijos (angl. *reduce*) funkcijų, atitinkamai vadinamų transformatoriumi (angl. *mapper*) ir reduktoriumi (angl. *reducer*). Transformatorius kaip įvestį priima rakto ir reikšmės porą ir formuoja tokių porų rinkinį, tuomet šie rinkiniai surūšiuojami pagal rakto reikšmę ir perduodami skirtingiems reduktoriams. Kiekvienas reduktoriaus gautas rinkinys turi tą patį raktą, dėl to MapReduce gali būti efektyviai panaudotas tokioms užduotims kaip duomenų rūšiavimas ir kiekio skaičiavimas (angl. *counting*). Vis dėlto, šią sistemą naudojančiam specialistui reikia įgyvendinti pačias transformavimo ir redukcijos funkcijas pagal tai, kokio funkcionalumo norima [LBC+13].

MapReduce skaičiavimų metu vykdomas funkcijas 2018 metais publikuotame straipsnyje aprašė Oussous ir kiti bendraautoriai [OBA+18]:

- transformavimo funkcija padalina įvestį į nepriklausomas duomenų dalis (angl. *partitions*), susidedančias iš rakto ir reikšmės porų;
- MapReduce sistema po vieną siunčia visas rastas rakto ir reikšmės poras, kurios išdalinamos į keletą lygiagrečiai klasteryje vykdomų transformavimo užduočių. Po šio žingsnio atlieka-

mas maišymas (angl. *shuffling*), kuomet pradedamos surinkinėti tarpinės raktų ir reikšmių poros, jos yra rūšiuojamos ir grupuojamos tarpusavyje, taip gaunant raktų ir jiems priskirtų reikšmių rinkinius;

- redukcijos funkcija apdoroja tarpinėje išvestyje gautus rinkinius ir kiekvienam raktui pritaiko nustatytą jungimo funkciją, tokią kaip filtravimas, sumavimas, sąmaišos skaičiavimas (angl. *hashing*) ar kita. Po skaičiavimų gaunama viena ar daugiau galutinių rakto ir reikšmės porų, atspindinčių MapReduce skaičiavimų rezultatą;

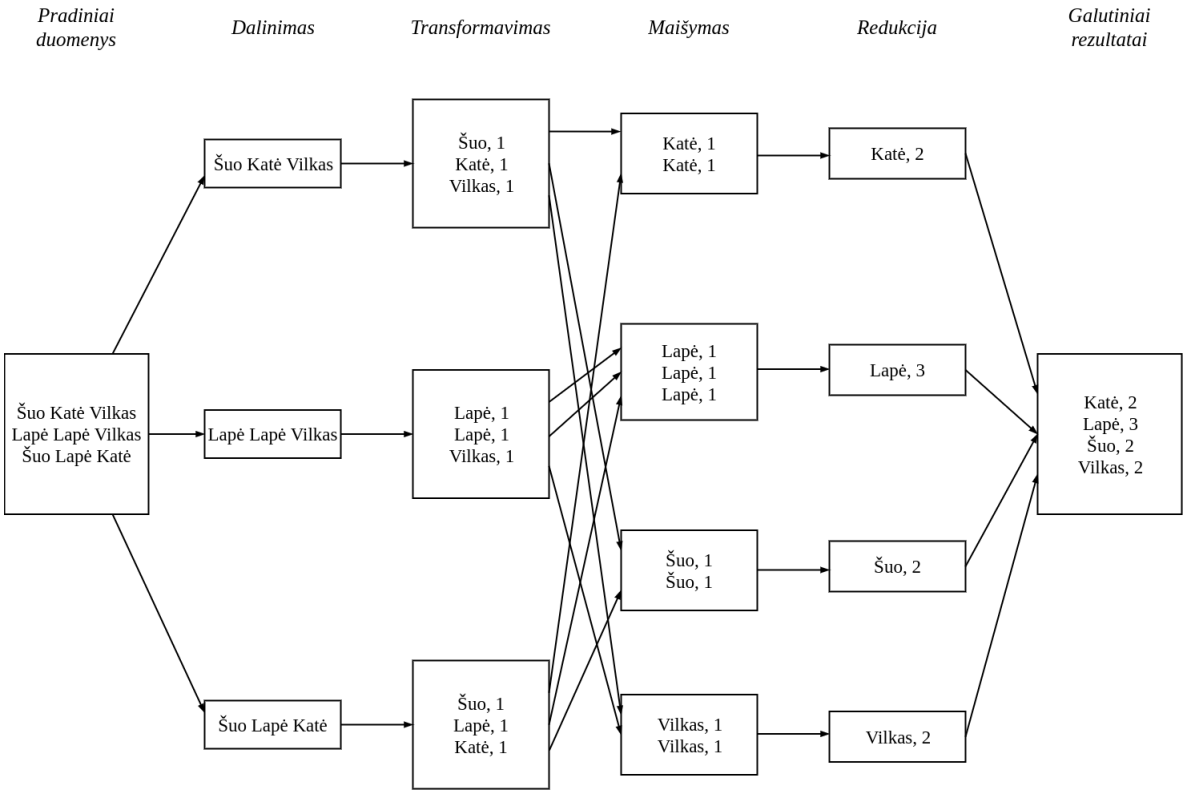
- visos rastos rakto ir reikšmės poros sistemos išsaugomos į failą, kurį gali pasiekti naudotojas arba toliau vykdomi kiti MapReduce darbai.

MapReduce architektūra sudaryta iš darbų stebėtoją (angl. *JobTracker*) vykdančio šeimininko ir daugelio užduočių stebėtojus (angl. *TaskTracker*) vykdančių darbininkų. Šeimininkas organizuoja darbus ir dalina užduotis darbininkams. Jis stebi užduočių vykdymą ir, skaičiavimams nepavykus, paskiria užduočiai naują vykdytoją [OBA+18]. Šeimininkas užduotims pasirenka darbininkus, tuo metu neturinčius darbo, ir jiems priskiria atlikti transformavimą arba redukciją. Transformavimą atliekantis darbininkas tarpinius rezultatus rašo į atmintį bei periodiškai išsaugo į failą lokaliame diske. Baigęs skaičiavimus, jis perduoda informaciją apie šį failą šeimininkui, kuris apie ją praneša redukciją vykdantiems darbininkams. Gavę pranešimą apie transformavimo pabaigą, jie naudoja nuotolinį procedūrų kvietimą (angl. *remote procedure call*) nuskaityti duomenis, esančius transformatoriaus atmintyje, juos surūšiuoja ir su gauta medžiaga atlieka redukciją, kurios rezultatai palaipsniui yra pridedami į galutinį išvesties failą [DG04]. MapReduce sistemą įprastai sudaro daugybė tarpusavyje sujungtų įrenginių, kurių dalis vykdymo metu gali tapti nepasiekiami. Siekiant užtikrinti nenutrūkstamą skaičiavimų vykdymą šeimininkas periodiškai paprašo atsako (angl. *ping*) iš kiekvieno sistemoje veikiančio darbininko. Jei per tam tikrą laiką jokio atsako nebuvo gauta, darbininkas pažymimas kaip neveikiantis ir visos jo vykdomos užduotys perduodamos kitoms sistemos dalims. Tokiu atveju, iš naujo atliekamos net ir užbaigtos nepasiekiamų darbininkų transformavimo užduotys, kadangi jų rezultatai saugomi ne globalioje failų sistemoje, o transformatorių atmintyje. Jeigu sutrikimas atsiranda šeimininko mazge, sistemą gali tekti perkrauti ir skaičiavimus atlikti pradedant nuo išsaugoto atskaitos taško (angl. *checkpoint*) [DG04].

Viename iš pirmųjų MapReduce modelį aprašiusių darbų, publikuotame Dean ir Ghemawhat 2004 metais [DG04], buvo nagrinėta realizacija, naudojanti Google failų sistemą (angl. *Google File System*). Siekiant išsaugoti duomenų integralumą, failai MapReduce sistemoje padalinami į blokus ir keliuose skirtinguose įrenginiuose išsaugomos jų kopijos. Taip užtikrinamas duomenų pasiekiamumas net ir nustojus veikti tam tikriems klasteryje esantiems įrenginiams. Vis dėlto, norint sumažinti tinklo apkrovą ir pagerinti duomenų perdavimo greitį, šeimininkas transformavimo užduotis darbininkams paskirsto taip, kad dauguma skaičiavimų būtų atliekami tame pačiame įrenginyje, kuriame laikoma ir įvesties medžiaga. Jeigu to padaryti neįmanoma pasirenkamas kitas transformavimo funkciją galintis atlikti darbininkas, esantis arčiausiai analizuojamų duomenų. Ši strategija padeda ženkliai sumažinti tinklu perduodamų duomenų kiekį, net ir vykdant didelio

masto skaičiavimus. Tokią pat problemą padeda spręsti ir kombinavimo (angl. *combiner*) funkcija, kuri transformatoriuje atlieka dalinę agregaciją, panašią į pateiktą redukcijos funkcijai ir leidžia tarp sistemos dalių perduoti mažesnį kiekį informacijos, taip pagreitinant veikimą. [DG04] Be to, kaip nurodyta Dean ir Ghemawhat, MapReduce sistema dėl savo paskirstytojo veikimo gali spręsti ir atsilikusio mazgo (angl. *straggler*) problemą, atsirandančią dėl itin lėto tam tikros sistemos dalies veikimo. Tokiai situacijai išvengti MapReduce skaičiavimų pabaigoje vis dar nebaigtos skaičiavimų užduotys pradedamos vykdyti kitų darbininkų ir MapReduce darbas užbaigiamas kai gaunamas atsakymas iš bet kurio tą pačią užduotį vykdančio įrenginio.

MapReduce darbo metu vykdomų žingsnių pavyzdys pateiktas 12 paveikslėlyje[8]. Jame demonstruojamas pradininiuose duomenyse pateikto teksto dalijimas, transformavimas, maišymas vykdant rūšiavimą ir grupavimą, redukcija skaičiuojant žodžių pasikartojimų kiekius bei galutinio rezultato surinkimas.



12 pav. MapReduce darbo metu vykdomi žingsniai skaičiuojant tekste pasikartojančius žodžius

4.2. Apache Hadoop ir Hadoop paskirstyta failų sistema

Praktikoje didžiųjų duomenų analizei naudojama ir kita technologija, vadinama Apache Hadoop. Tai atviro kodo (angl. *open source*) MapReduce modelį įgyvendinanti sistema, pakankamai plačiai naudojama tiriant didžiuosius duomenis įvairiose mokslo ir verslo srityse [LBC+13]. Hadoop dalis sudaro Hadoop MapReduce realizacija, galinti atlikti skaičiavimus naudojant įrenginių klasterių resursus, bei Hadoop paskirstyta failų sistema (angl. *Hadoop Distributed File System*, toliau – HDFS), kurioje saugoma įvestis transformavimo funkcijai bei pateikiami redukcijos

[8]MapReduce darbo metu vykdomų žingsnių pavyzdys, adaptuotas iš `https://whatsbigdata.be/mapreduce/`

funkcijos vykdymo metu gautų rezultatų failai. Ši sistema pritaikyta duomenis apdoroti grupėmis (angl. *batch*) ir negali būti naudojama pasiekti tarpinius MapReduce darbo rezultatus, kadangi jie yra laikomi kiekvieno skaičiavimus atliekančio mazgo lokalioje atmintyje [CCA+19].

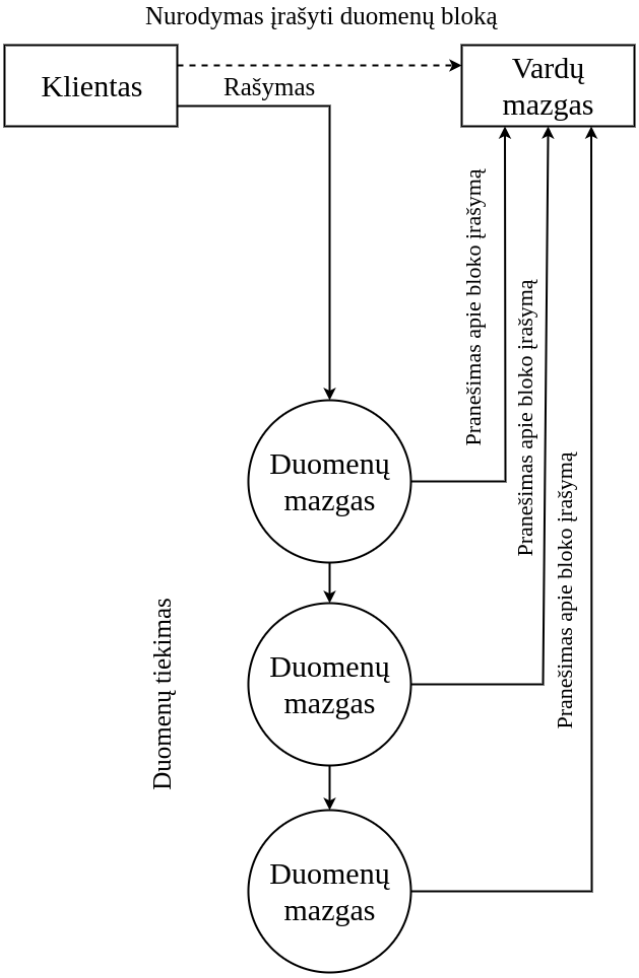
Hadoop architektūra panaši į įprastą MapReduce sistemą. Ją sudaro vienas šeimininko mazgas, vykdantis darbų stebėjimo procesą ir daugelis darbininkų mazgų, atliekančių užduočių stebėjimo funkcijas. Šeimininkas iš klientų priima darbus, padalina juos į užduotis ir paskirsto darbininkams, kurių mazgai šias užduotis atlieka. Įprastai kiekvienas darbininkas turi galimybes vykdyti keletą transformavimo ir redukcijos funkcijų. Vis dėlto, kai vienos iš jų įvestimi yra kitos funkcijos išvestis, atlikimas negali persidengti. Tokiu atveju reduktorius apie transformacijos metu gautus tarpinius rezultatus sužino tik tada, kai šeimininkas yra informuojamas apie transformatoriaus atliktą darbą ir pradedama organizuoti redukcijos užduotis. Persidengimas nevyksta ir reduktoriaus skaičiavimus atliekant viename darbe, o jo rezultatus naudojant kito darbo transformavimo funkcijos vykdymui. Tam reikalinga žinoti HDFS saugomo atminties bloko vietą, o redukcijos funkcija jį suformuoja tik tada, kai yra užbaigtos visos reduktoriui duomenis teikiančios transformavimo užduotys [CCA+19].

HDFS yra paskirstyta failų sistema, galinti veikti klasteryje įprastų rinkoje prieinamų įrenginių [Apa19]. Ši sistema leidžia patikimai saugoti didelius duomenų rinkinius bei didele sparta juos perduoti apdorojimui. Didelių klasterių sistemos yra sudarytos iš daugybės tarpusavyje sujungtų įrenginių, galinčių ne tik atlikti skaičiavimus, bet ir saugoti duomenis. Išskirsčius atmintį po klasterio įrenginius, atsiranda galimybė turimų resursų kiekį pakankamai paprastai išplėsti prijungiant į sistemą daugiau serverių [SKR+10].

HDFS duomenys ir informacija apie juos, vadinama metaduomenimis (angl. *metadata*), yra saugomi atskirai. Metaduomenys yra saugomi šeimininko serveryje, vadinamame vardų mazgu (angl. *NameNode*), o skaičiavimams ir analizei naudojami duomenys laikomi daugelyje kitų serverių, vadinamų duomenų mazgais (angl. *DataNode*). Visi jie tarpusavyje yra sujungti ir komunikuoja TCP paremtu protokolu. HDFS duomenų apsaugojimui nėra naudojami sudėtingi apsaugos mechanizmai, vietoj to, duomenų blokai yra kopijuojami ir jų kopijos išdalinamos skirtingiems duomenų mazgams. Tai leidžia ne tik patikimiau saugoti duomenis, bet pagerina tinklo pralaidumą dėl geresnio apkrovos išskirstymo ir galimybės skaičiavimus atlikti esant arčiau analizei reikalingų duomenų [SKR+10].

Hadoop naudojama paskirstyta failų sistema atitinka šeimininko ir daugelio darbininkų architektūrą. Šeimininko serveryje esantis vardų mazgas valdo visą failų sistemos vardų sritį (angl. *namespace*) ir klientų prieigą prie sistemoje esančių duomenų [Apa19]. Vardų sritis susideda iš hierarchiškai išdėstytų katalogų ir failų. Ji vardų mazge laikoma medžio pavidalu ir atspindi informaciją apie tai kokiuose duomenų mazguose saugomi tam tikri duomenų blokai. Kai klientas nori perskaityti medžiagą, jis vardų mazgo užklausia apie ieškomo duomens buvimo vietą ir skaitymą atlieka tiesiai iš duomenų mazgo. Rašant duomenis, vardų mazgas išrenka tris skirtingus duomenų mazgus ir medžiaga į juos įrašoma tiekimo (angl. *pipeline*) principu. Tokiu atveju klientas duomenis pateikia pirmajam mazgui, o tolesnis kopijų dalijimasis vyksta tarp mazgų. Kai įrašomas pirmas duomenų blokas, klientas gali prašyti vardų mazgo paskirti vietą tolimesniam medžiagos

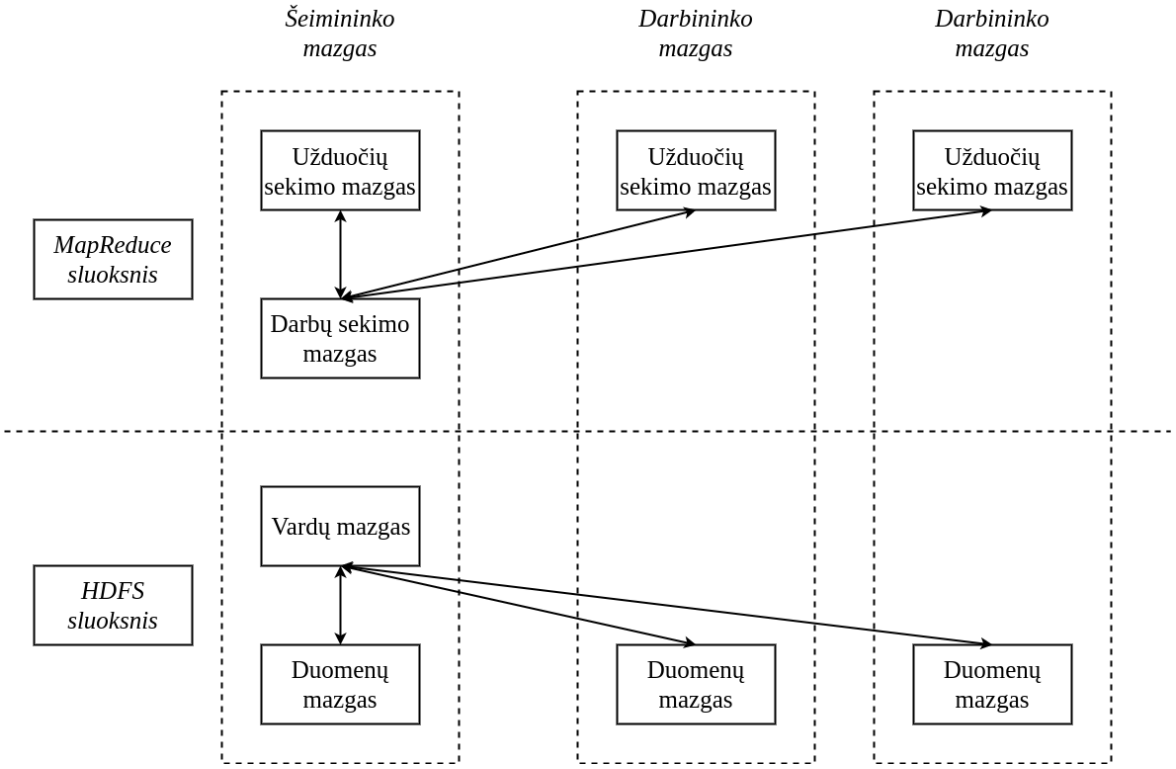
rašymui. Tokio duomenų valdymo vizualizacija pateikta 13 paveikslėlyje [SKR+10].



13 pav. HDFS vykdomas kopijavimo mechanizmas duomenų rašymo metu

Vienas iš svarbiausių didžiųjų duomenų technologijų iššūkių yra atsparumas klaidoms net ir apdorojant didelius duomenų rinkinius. HDFS tai užtikrina savo duomenų blokų ir jų kopijavimo (angl. *replication*) mechanizmu. Visi į sistemą įrašomi failai išsaugomi kaip seka vienodo dydžio atminties blokų ir jų kopijos išdalinamos skirtingiems mazgams. Kiekvienas duomenų mazgas periodiškai siunčia savo „širdies dūžį“ (angl. *heartbeat*) vardų mazgui, kartu su visų turimų duomenų blokų sąrašu. Jeigu šeimininkas kurį laiką iš darbininko negauna jokio signalo, duomenų mazgas pradedamas laikyti neaktyviu, sustabdomos duomenų įrašymo operacijos ir pradedamas jame buvusių blokų kopijavimas į naujai pasirinktus mazgus [Apa19]. Sistemoje esančių duomenų integralumas išsaugojamas ir nustojus veikti šeimininko serveryje esančiam vardų mazgui. Tai užtikrinama mazgo įjungimo metu sukuriant sistemos atvaizdą, vadinamą atskaitos tašku (angl. *checkpoint*) ir kaupiant po jo sukūrimo įvykusių duomenų pasikeitimų žurnalą. Tokią funkciją gali atlikti ir specialiai tam skirtas atskaitos taško mazgas (angl. *CheckpointNode*), kuris periodiškai sukuria sistemos atvaizdą ir išvalo žurnalą tam, kad būtų atlaisvinta dalis atminties, reikalingos vardų srities valdymui [SKR+10].

Darbe aprašytos Hadoop architektūros modelis pateiktas 14 paveikslėlyje[9]. Modelyje nurodytas skirtinguose sistemos sluoksniuose esančių mazgų išdėstymas: skaičiavimus ir analizę atliekančiame MapReduce sluoksnyje galima rasti darbų ir užduočių stebėjimo mazgus, o HDFS sluoksnyje, atsakingame už duomenų saugojimą, randami vardų ir duomenų mazgai, paskirstyti po sistemos klasterį.



14 pav. Hadoop architektūros modelio vizualizacija

4.3. Apache Spark variklis ir MLlib biblioteka

Didžiųjų duomenų tyryboje neretai tenka iteratyviai analizuoti iš skirtingų šaltinių gautus duomenis. Viena iš tokioms užduotims skirtų technologijų yra vadinama Apache Spark varikliu. Tai kompiuterio atmintyje (angl. *in-memory*) skaičiavimus atliekantis didžiųjų duomenų variklis, pasižymintis savo greičiu ir plečiamumu. Jis leidžia duomenų analizės įrankius kurti Java, Python, Scala ir R kalbomis bei palengvina šių programų kūrimą galimybe tyrimams panaudoti iš anksto paruoštas duomenų tyrybos priemonių bibliotekas [ABL+17].

Apache Spark yra atviro kodo (angl. *open-source*) variklis, galintis iteratyviai atlikti skaičiavimus paskirstytose klasterių sistemose. Jis suteikia galimybę skaičiavimams panaudoti optimizuotus tyrybos algoritmus ir dėl pasikartojimais paremto (angl. *iterative*) vykdymo yra gerai pritaikytas didelio masto duomenų analizei [MBY+16].

Apache Spark programavimo modelis panašus į siūlomą MapReduce technologijos, tačiau yra paremtas kitomis duomenų struktūromis, kurios vienas iš pavyzdžių yra vadinamas lanksčiu paskirstytuoju duomenų rinkiniu (angl. *Resilient Distributed Dataset*, toliau – RDD). RDD yra

[9]Hadoop architektūros modelio vizualizacija, adaptuota iš `http://a4academics.com/tutorials/83-hadoop/835-hadoop-architecture`

klaidoms atspari po klasterį išdalintų duomenų kolekcija, kuri gali būti sukurta funkcinio stiliaus transformacijomis ir apdorojama lygiagrečiai. Šią duomenų struktūrą Spark variklis apdoroja įrenginio atmintyje, todėl išvengiama nemažai pakankamai lėtų duomenų rašymo ir skaitymo diske operacijų. Toks apdorojimas atliekamas tingiai (angl. *lazily*), sudarant efektyvų skaičiavimų vykdymo planą. Jei tas pats RDD analizės metu yra keletą kartų iš eilės transformuojamas, Spark gali peržvelgti visą transformacijų seką ir šias operacijas įvykdyti vienu kartu, taip sutaupydamas dalį įrenginio resursų ir tinklo apkrovos. Be to, lankstieji paskirstytieji duomenų rinkiniai gali būti naudojami keleto skirtingų skaičiavimų. Nors įprastai transformuojant jie kiekvieną kartą yra perskaičiuojami, bet duomenis tiriantys specialistai gali nurodyti RDD išsaugoti atmintyje tam, kad tolimesnių skaičiavimų metu šias struktūras būtų galima panaudoti vėl. Jeigu visi duomenys atmintyje negali būti sutalpinti, dalis šių RDD struktūrų gali būti išsaugotos diske [ZXW+16].

Šis atviro kodo duomenų variklis taiko kitokius metodus apsaugoti duomenims ir jų integralumui. Kitaip nei MapReduce ar Hadoop sistemose, duomenims išsaugoti nėra kuriamos jų kopijos ar sistemos atskaitos taškai (angl. *checkpoint*). Vietoj to, kiekvienas RDD seka jo sukūrimui skirtas transformacijas ir, jei tam tikra duomenų dalis tampa nepasiekiama, šias operacijas atlieka iš naujo tam, kad būtų atstatyta prarasta analizuojamos medžiagos dalis. Toks procesas Apache Spark variklyje vadinamas kilme paremtu atsigavimu (angl. *lineage-based recovery*) ir yra efektyvesnis už MapReduce ar Hadoop sistemų apsaugos mechanizmus dėl to, kad esant dideliam duomenų kiekiui, jų perdavimas tinklu būtų lėtesnis nei perskaičiavimas atmintyje, kuris Apache Spark variklyje, kaip ir kitos operacijos, atliekamas lygiagrečiai [ZXW+16].

Apache Spark variklis suteikia galimybę pasinaudoti iš anksto paruoštomis įrankių bibliotekomis, skirtomis efektyviai didžiųjų duomenų tyrybai. Jų pavyzdžiais gali būti nenutrūkstamiems duomenų srautams valdyti skirta Spark Streaming biblioteka, įrankius nuskaityti medžiagai iš skirtingų šaltinių suteikianti Spark SQL biblioteka, GraphX biblioteka, leidžianti panaudoti optimizuotą sąsają grafais pateiktų struktūrų analizei, bei MLlib bilblioteka, kurioje įgyvendinta daugybė duomenų analizės algoritmų, kurių dalis skirta klasterizavimo ir klasifikavimo užduotims atlikti [ZXW+16]. MLlib leidžia pasinaudoti Apache Spark variklio teikiamais duomenų skaidymo ir procesų lygiagretinimo mechanizmais. Joje įgyvendinti klasterizavimo, klasifikavimo, matmenų mažinimo bei taisyklių radimo algoritmai padeda paprasčiau atlikti optimizuotas didžiųjų duomenų tyrybos užduotis [ABL+17], o išvystyta integracija su likusia Spark ekosistema leidžia iteratyviai ir išlygiagretintai atlikti informacijos paieškoms skirtus duomenų analizės algoritmus. Ši savybė aktuali atliekant didelio masto duomenų tyrybą, kadangi daugelis joje naudojamų būdų paremti pasikartojančiu skaičiavimų vykdymu [MBY+16].

5. Didžiųjų duomenų analizės eksperimentai

Siekiant ištirti didžiųjų duomenų technologijų tinkamumą klasterizavimo ir klasifikavimo užduotims, buvo atlikti eksperimentai, vykdantys tokią duomenų tyrybą ir apdorojantys didelius kiekius testinių duomenų. Klasterizavimo bandymuose pasirinkta tirti algoritmų vykdymo laiką analizuojant skirtingus duomenų kiekius ir keičiant ieškomų klasterių skaičių. Klasifikavimo bandymų metu buvo analizuojama ne tik laikas, per kurį algoritmas yra apmokomas ir įvykdomas naujų elementų testavimas, bet ir rezultatai, rodantys kaip tiksliai algoritmas suveikė, lyginant jo rastas testuojamų duomenų klases su iš anksto žinomomis jų etiketėmis. Bandant geriau suprasti klasifikavimo algoritmų mokymo fazės svarbą, buvo atlikti eksperimentai su skirtingais mokymo aibės dydžiais ir lyginami tokiu būdu gauti algoritmų vertinimo metodų rezultatai.

5.1. Sistemos konfigūracija

Eksperimentai atlikti naudojantis Apache Spark didžiųjų duomenų variklio 2.4.5 versija. Taip pat pasinaudota 2.4.5 versijos MLlib bibliotekoje esančių klasterizavimo ir klasifikavimo algoritmų realizacijomis. Šių technologijų panaudojimui Java programavimo kalba buvo sukurtas Apache Maven projektas, vykdantis MLlib bibliotekos pateikiamą funkcionalumą. Klasterių analizei nuspręsta naudoti šioje bibliotekoje randamus K-vidurkių bei Gauso sumaišymo (angl. *Gaussian mixture*) algoritmus. Pastarasis pasirinktas dėl to, kad savo veikimu ir gaunamais rezultatais yra panašus į darbe nagrinėjamą lūkesčių maksimizavimo algoritmą. Klasifikavimo eksperimentams panaudoti Naivaus Bajeso ir sprendimų medžio algoritmai, ieškota jų tikslumo, precizijos, jautrumo ir F-įverčio. Šie analizės metodai vykdyti naudojantis MLlib bibliotekoje prieinama skaičiavimų realizacija. Be to, atkreiptas dėmesys ir į tai, kad, nors Apache Spark variklis leidžia eksperimentus gana paprastai atlikti paskirstytose skaičiavimų sistemose, algoritmų vykdymas jose yra mokamas ir, naudojant daugiau resursų, gali kainuoti šimtus eurų. Dėl šios priežasties nuspręsta eksperimentus vykdyti asmeniniame kompiuteryje, kurio techniniai duomenys nurodyti 15 paveikslėlyje. Eksperimentams taip pat panaudotas 0.7.5 versijos Elki duomenų tyrybos įrankis[10], kuris siūlo geresnes klasterių vizualizacijos galimybes nei Apache Spark variklis ir MLlib biblioteka.

Atmintis	7,7 GiB
Procesorius	Intel® Core™ i7-9750H CPU @ 2.60GHz × 12
Grafika	GeForce GTX 1650/PCIe/SSE2
Operacinė sistema	Ubuntu 18.04.4 LTS
Operacinės sistemos tipas	64-bit
Diskas	110,1 GB

15 pav. Eksperimentams naudoto įrenginio techniniai duomenys

[10]Elki yra atviro kodo duomenų tyrybos įrankis, sukurtas Java programavimo kalba ir teikiantis duomenų klasterizavimo viename įrenginyje funkcionalumą. Jo dokumentacija pateikta `https://elki-project.github.io/`

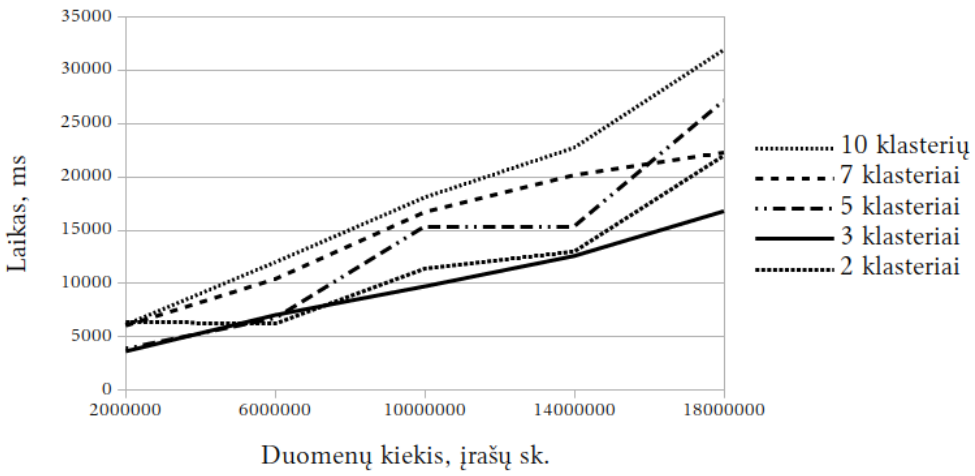
5.2. Klasterizavimo eksperimentai

Dalyje darbe aprašytų klasterizavimo eksperimentų naudoti dirbtinai generuoti testiniai duomenys, atspindintys duomenų įrašų išsidėstymą klasteriais dvimatėje požymių erdvėje. Toks būdas leido geriau valdyti technologijų išbandymui skirtų eksperimentų vykdymą, suteikiant daugiau galimybių keisti duomenų kiekį, jų išsidėstymą erdvėje ir formą. Generavimo žingsnis buvo pasiruošimo vykdymams dalis ir pakeitė tokioje tyryboje įprastą duomenų išankstinio apdorojimo (angl. *preprocessing*) žingsnį, kurio metu nagrinėjami duomenys gali būti valomi, normalizuojami ir pakeičiami skaitine ar kita algoritmams suprantama forma.

Pirmajame eksperimente buvo siekiama nustatyti kokią įtaką vykdymo laikui turi skirtingas analizuojamų duomenų ir ieškomų klasterių kiekis, klasterizavimą atliekant MLlib bibliotekoje prieinamu K-vidurkių algoritmu. Eksperimente naudoti generuoti duomenys, dauguma jų išdėstyti apie keletą centrų susitelkusių elementų grupėmis. Kiekvieno vykdymo metu algoritmui buvo pateikiama skirtingą duomenų kiekį turinti įvestis, keičiamas buvo ir ieškomų klasterių skaičius. 2 lentelėje ir 16 paveikslėlyje pateiktuose rezultatuose matoma, kad nagrinėjamam duomenų kiekiui didėjant, algoritmo vykdymo laikas augo beveik tiesiškai. Kai kuriais atvejais vykdymo laikas išliko toks pats net ir įvesties kiekiui padidėjus, tačiau tokius rezultatus galima vertinti kaip anomalijas, kurias galėjo sukelti kiti kompiuteryje vykdomi procesai. Taip pat gautuose rezultatuose matoma tendencija, jog ieškomų klasterių kiekiui didėjant, šiek tiek išauga ir algoritmo vykdymo laikas. Tai gali būti susiję su poreikiu kiekvieno pakartojimo metu perskaičiuoti klasterių centrus.

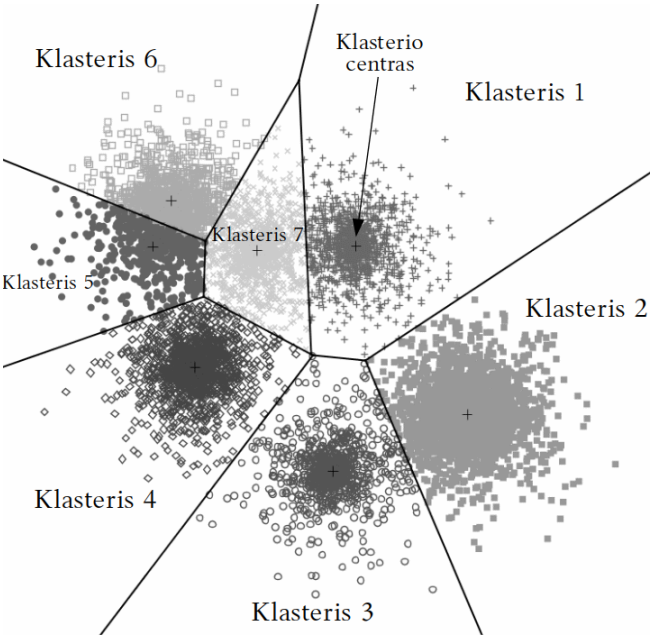
2 lentelė. Klasterizavimo vykdymo laikas naudojant K-vidurkių algoritmą

Duomenų kiekis	2 klasteriai	3 klasteriai	5 klasteriai	7 klasteriai	10 klasterių
	Laikas, ms	Laikas, ms	Laikas, ms	Laikas, ms	Laikas, ms
2000000	6390	3631	3896	6044	6130
6000000	6224	7033	6749	10413	11993
10000000	11391	9720	15367	16699	18081
14000000	13000	12579	15373	20164	22740
18000000	21997	16775	27164	22278	31920



16 pav. Klasterizavimo K-vidurkių algoritmu vykdymo laiko grafikas

Pirmojo eksperimento duomenų klasterių vizualizacija pateikta 17 paveikslėlyje. Klasterizavimo metu apdorota 1800000 elementų, turinčių po 2 požymius, ir ieškota 7 klasterių.

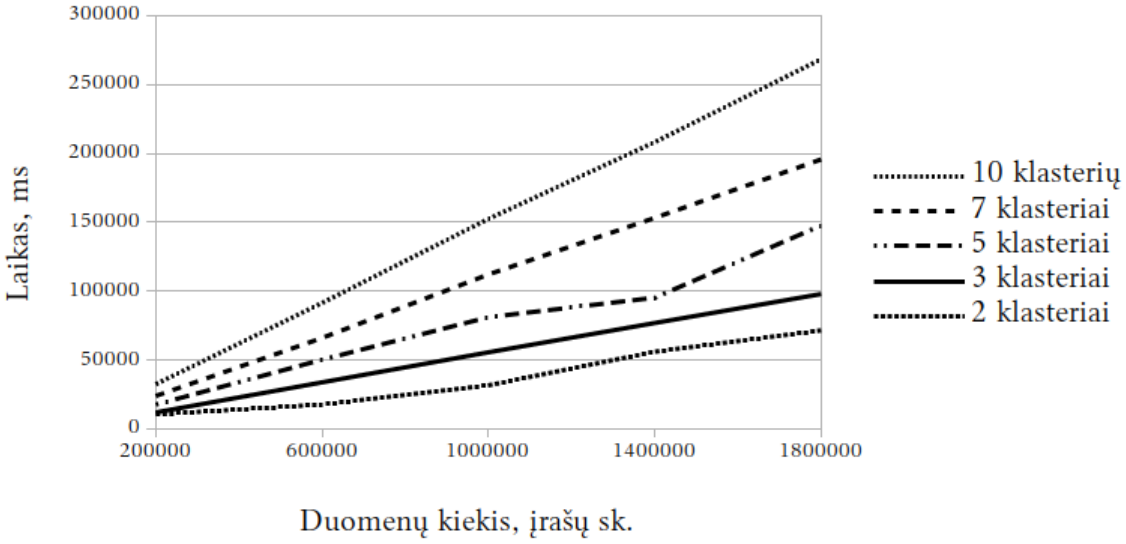


17 pav. K-vidurkių algoritmu rastų klasterių vizualizacija

Antrajame eksperimente buvo nagrinėjama kaip klasterizavimo vykdymo laikas keičiasi analizuojant skirtingą kiekį duomenų MLlib bibliotekos Gauso sumaišymo algoritmu. Taip pat skaičiavimuose buvo keičiamas ir algoritmo ieškomų klasterių kiekis. Kaip ir pirmojo eksperimento metu, analizei naudoti generuoti testiniai duomenys, o kiekvieno vykdymo metu metu buvo keičiamas analizuojamos jų dalies kiekis bei klasterių skaičius. 3 lentelės rezultatuose ir 18 paveikslėlyje galima įžvelgti, kad klasterizuojamų duomenų kiekiui didėjant, algoritmo vykdymo laikas didėjo santykinai stabiliai. Panašus veikimo trukmės augimas pastebėtas ir didinant ieškomų klasterių kiekį. Nepaisant to, Gauso sumaišymo algoritmas per kelias minutes sugebėjo atlikti net ir milijonų elementų analizę. Vis dėlto verta atkreipti dėmesį į tai, kad, net ir sumažinus įvesties kiekį, jis demonstravo prastesnį vykdymo laiką nei K-vidurkių algoritmas.

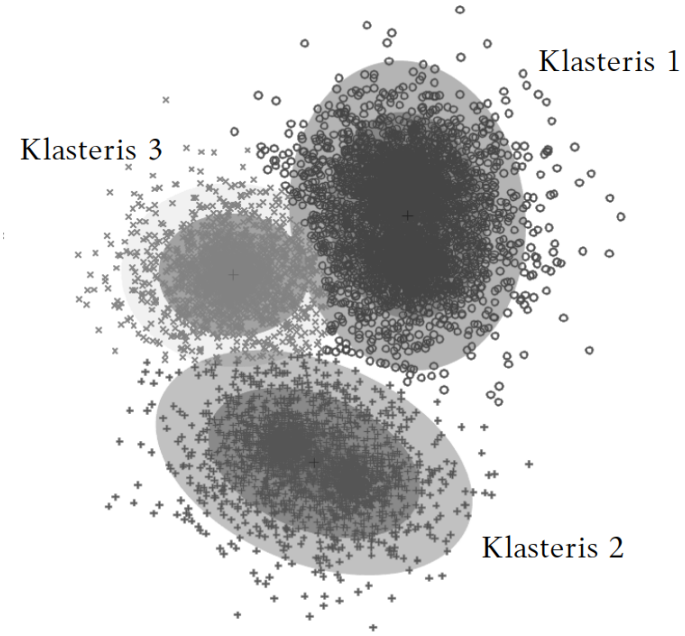
3 lentelė. Klasterizavimo vykdymo laikas naudojant Gauso sumaišymo algoritmą

Duomenų kiekis	2 klasteriai	3 klasteriai	5 klasteriai	7 klasteriai	10 klasterių
	Laikas, ms	Laikas, ms	Laikas, ms	Laikas, ms	Laikas, ms
200000	10448	11807	17509	23758	32338
600000	17608	33693	50150	65930	91344
1000000	31538	55494	81036	112051	152448
1400000	55948	76796	95199	153173	208206
1800000	71402	97820	147656	195632	268422



18 pav. Klasterizavimo Gauso sumaišymo algoritmu vykdymo laiko grafikas

Duomenų, kurie buvo nagrinėti antrojo eksperimento metu, klasterizavimo rezultatų vizualizacija pateikta 19 paveikslėlyje. Klasterių analizė atlikta vykdant Elki duomenų tyrybos įrankyje randamą lūkesčių maksimizavimo algoritmą, apdorota 200000 duomenų, turinčių po 2 požymius, bei ieškota 3 klasterių.



19 pav. Lūkesčių maksimizavimo algoritmu rastų klasterių vizualizacija

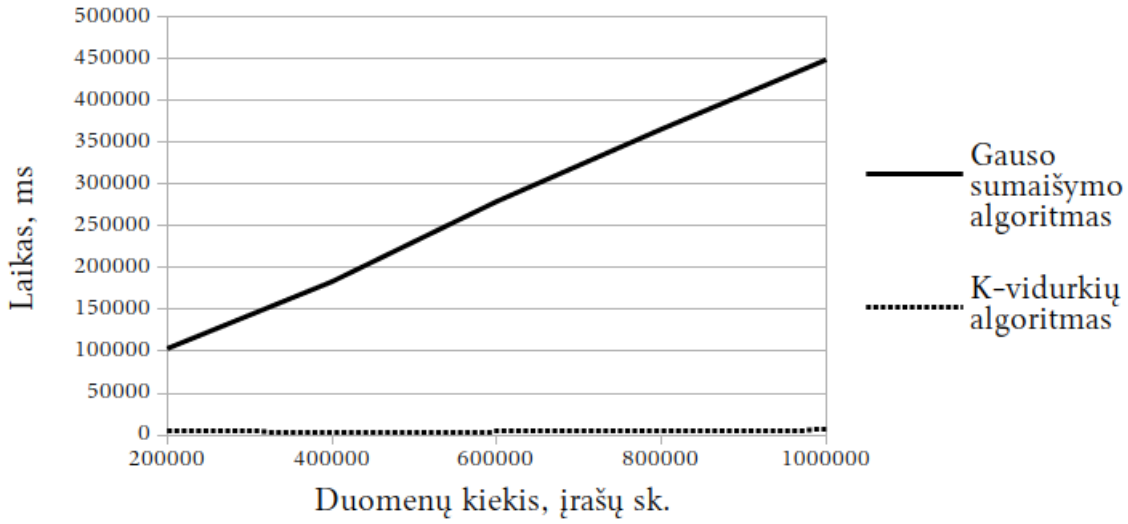
Trečiojo eksperimento metu klasterių analizė buvo vykdoma realiomis sąlygomis surinktų duomenų rinkinyje, gautame iš UCI mašininio mokymo saugyklos (angl. *machine learning repository*). Šis rinkinys sudarytas iš duomenų, atspindinčių individualių namų elektros energijos suvartojimą[11], ir yra surinktas 4 metus kas minutę tikrinant elektros matavimų rodmenis.

[11]Individualių namų elektros energijos suvartojimo duomenų rinkinys gali būti rastas `https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption`

Rinkinį sudaro 2075259 elementai, bet eksperimentams nuspręsta panaudoti pusę jų. Siekiant ištirti algoritmų spartą klasterizuojant įvairus duomenų kiekius, įvestis buvo dalinama į 5 poaibius, turinčius skirtingą kiekį įrašų: 200000, 400000, 600000, 800000 ir 1000000. Originaliame duomenų rinkinyje visi elementai turi 9 požymius, kuriuose pateikiami skirtingo pobūdžio matavimų kiekiai. Vis dėlto, prieš klasterizavimą 2 požymiai, atspindintys matavimo datą ir laiką, buvo pašalinti ir palikti 7 skaitinėmis reikšmėmis išreikšti požymiai. Nagrinėjant 4 lentelėje ir 20 paveiklslėlyje pateiktus klasterizavimo rezultatus, galima įžvelgti, kad tiek K-vidurkių, tiek Gauso sumaišymo algoritmas klasterių analizę sugebėjo įvykdyti per pakankamai trumpą laiką. Vis dėlto, klasterizuojant duomenis Gauso sumaišymo būdu, vykdymo laikas buvo daug kartų didesnis nei analizę vykdant K-vidurkių algoritmu. Didinant įvesties kiekį šio eksperimento apibrėžtose ribose, analizei reikalingas laikas augo santykinai tiesiškai. Tikėtina, kad tokia tendencija gauta ir dėl pasirinktų nedidelių analizuojamo duomenų kiekio rėžių.

4 lentelė. Realiomis sąlygomis surinktų duomenų klasterizavimo laikas K-vidurkių ir Gauso sumaišymo algoritmais

Duomenų kiekis	K-vidurkių algoritmas	Gauso sumaišymo algoritmas
	Laikas, ms	Laikas, ms
200000	5907	103231
400000	2834	183131
600000	4116	279052
800000	5079	365474
1000000	6147	448394



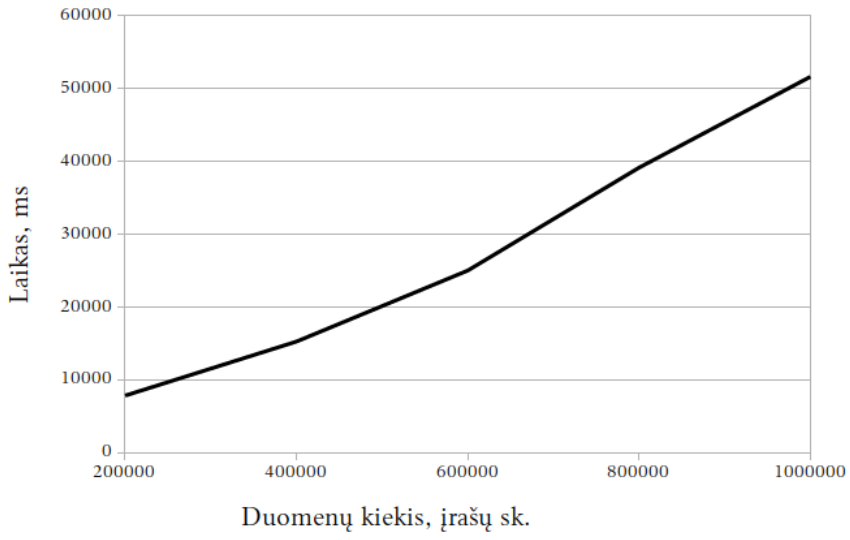
20 pav. Realiomis sąlygomis surinktų duomenų klasterizavimo vykdymo laikas

5.3. Klasifikavimo eksperimentai

Ketvirtojo eksperimento metu buvo tirta, kokią įtaką vykdymo laikui ir rezultatų kokybiniams įverčiams turi Naivaus Bajeso algoritmo klasifikuojamas duomenų kiekis. Klasių analizei buvo naudojama MLlib bibliotekoje randama algoritmo realizacija, o pradinė įvestis gauta iš URL reputacijos duomenų rinkinio[12], esančio UCI mašininio mokymo saugykloje. Šį rinkinį sudaro 2396130 elementų, turinčių po 3231961 požymį. Duomenys suskirstyti į dvi klases, iš kurių viena reiškia saugius internetinių puslapių adresus, o kita – pavojingų ir skirtų interneto naudotojų apgavystėms bei programinės įrangos (angl. *software*) pažeidimams. Bandant išsiaiškinti klasifikuojamų duomenų kiekio įtaką tikslumui ir vykdymo laikui, iš rinkinio suformuoti 5 duomenų poaibiai sudaryti iš 200000, 400000, 600000, 800000 ir 1000000 duomenų. Atlikus jų klasifikavimą Naivaus Bajeso algoritmu, 5 lentelėje ir 21 paveikslėlyje pateiktuose rezultatuose galima matyti, kad analizės vykdymo laikas didėjo santykinai greičiau nei klasifikuojamų elementų kiekis. Nepaisant to, algoritmas su visais įvesties kiekiais demonstravo gerus ir pastovius tikslumo, precizijos, jautrumo ir F-įverčio rezultatus. Šie įverčiai galėjo būti nulemti ir to, kad net 80 procentų duomenų buvo panaudoti klasifikavimo modelio formavimui mokymo fazėje.

5 lentelė. Naivaus Bajeso algoritmu vykdyto klasifikavimo laikai ir kokybiniai įverčiai keičiantis duomenų kiekiui

Duomenų kiekis	Vykdymo laikas, ms	Tikslumas	Precizija	Jautrumas	F-įvertis
200000	7838	0,913	0,947	0,947	0,869
400000	15261	0,926	0,938	0,853	0,893
600000	25006	0,925	0,930	0,862	0,894
800000	39134	0,926	0,919	0,860	0,889
1000000	51602	0,926	0,915	0,866	0,890



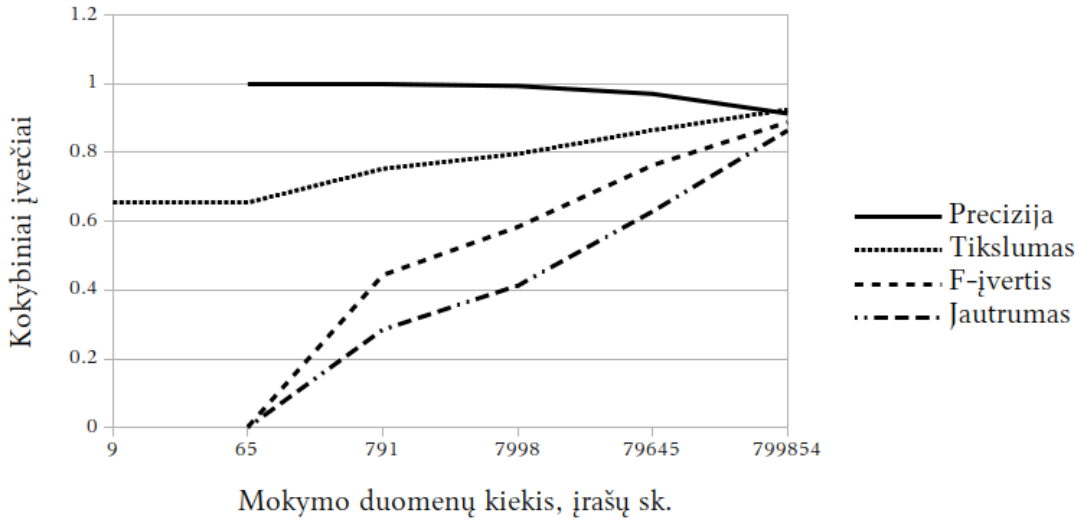
21 pav. Klasifikavimo Naivaus Bajeso algoritmu vykdymo laikas keičiantis duomenų kiekiui

[12]Eksperimentams panaudotas URL reputacijos duomenų rinkinys rastas `https://archive.ics.uci.edu/ml/datasets/URL+Reputation`

Penktajame eksperimente nagrinėta kaip keičiasi Naivaus Bajeso algoritmu vykdytos klasifikacijos kokybiniai įverčiai pagal tai, kokia duomenų dalis buvo panaudota modelio apmokymui. Taip pat buvo matuotas ir bendras algoritmo vykdymo laikas. Kiekvienos klasifikacijos metu naudota po 1000000 URL reputacijos duomenų rinkinio elementų, iš kurių skirtingos dalys buvo paskirstytos mokymo ir testavimo fazėms. 6 lentelėje ir 22 paveikslėlyje pateikti rezultatai rodo, kad tikslumas, precizija, jautrumas ir F-įvertis sparčiai didėjo augant mokymo duomenų kiekiui. Jam esant itin mažam, precizijos, jautrumo ir F-įverčio rodmenys net nebuvo tiksliai surasti. Verta atkreipti dėmesį ir į tai, kad nuo mokymo duomenų kiekio priklausė ir algoritmo vykdymo laikas, kuris augo šių elementų kiekiui didėjant.

6 lentelė. Naivaus Bajeso algoritmu vykdyto klasifikavimo laikai ir kokybiniai įverčiai keičiantis mokymo duomenų kiekiui

Mokymo duomenų kiekis	Vykdymo laikas, ms	Tikslumas	Precizija	Jautrumas	F-įvertis
9	29231	0,6541	Nerado	Nerado	Nerado
65	30527	0,6544	1,0	0,0009	0,0018
791	31805	0,7527	0,9996	0,2850	0,4435
7998	35888	0,7960	0,9935	0,4130	0,5835
79645	40996	0,8649	0,9706	0,6287	0,7631
799854	45972	0,9249	0,9136	0,8641	0,8882



22 pav. Klasifikavimo Naivaus Bajeso algoritmu kokybiniai įverčiai keičiantis mokymo duomenų kiekiui

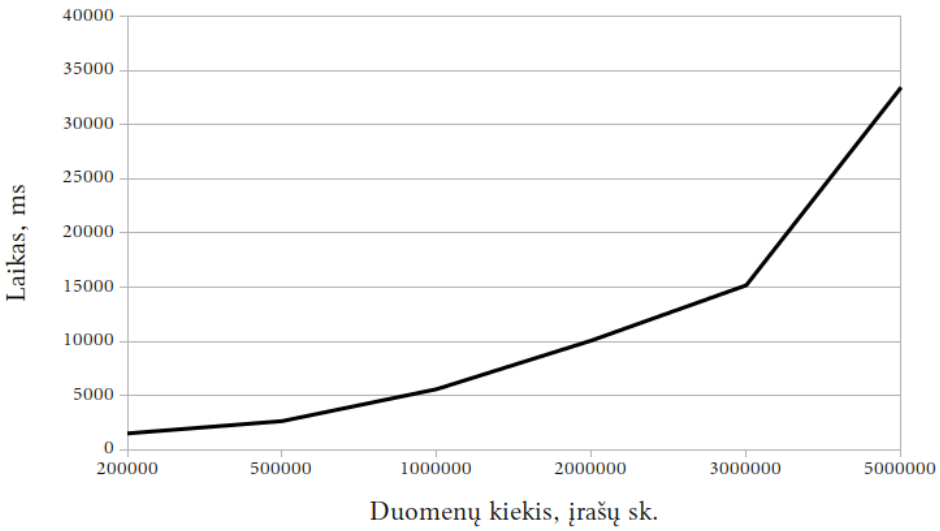
Šeštojo eksperimento metu tirta klasifikuojamų duomenų kiekio įtaka sprendimų medžio algoritmu vykdomai klasifikacijai. Buvo fiksuojami ir kokybiniai rezultatų įverčiai, apskaičiuoti MLlib bibliotekoje esančiais algoritmais. Panaudota ir šios bibliotekos sprendimų medžio realizacija, kuria analizuoti UCI mašininio mokymo saugykloje randamo SUSY duomenų rinkinio[13] elementai. Šį rinkinį sudaro 5000000 duomenų, suskirstytų į dvi klases ir pagal 18 požymių reiškiančių dalelių priklausymą signalui. Prieš klasifikaciją šie duomenys buvo suskirstyti į poaibius

[13]SUSY duomenų rinkinys gali būti rastas `https://archive.ics.uci.edu/ml/datasets/SUSY`

turinčius 200000, 500000, 1000000, 2000000, 3000000 ir 5000000 elementų. Įvykdžius skaičiavimus, iš rezultatų, pateiktų 7 lentelėje ir 23 paveikslėlyje, matyti, jog sprendimų medžio algoritmo tikslumas, precizija, jautrumas ir F-įvertis buvo mažesni nei Naivaus Bajeso algoritmo. Vis dėlto jie išliko pakankamai pastovūs net ir klasifikuojamų duomenų kiekiui kintant. Tuo tarpu, algoritmo įvesčiai didėjant turimuose duomenų kiekio rėžiuose, vykdomo klasifikavimo laikas taip pat augo.

7 lentelė. Sprendimų medžio algoritmu vykdyto klasifikavimo laikai ir kokybiniai įverčiai keičiantis duomenų kiekiui

Duomenų kiekis	Vykdymo laikas, ms	Tikslumas	Precizija	Jautrumas	F-įvertis
200000	1502	0,7678	0,8089	0,6465	0,7187
500000	2639	0,7718	0,7672	0,7226	0,7442
1000000	5589	0,7737	0,7692	0,7219	0,7448
2000000	10080	0,7720	0,8043	0,6634	0,7271
3000000	15176	0,7717	0,7628	0,7251	0,7435
5000000	33440	0,7719	0,7621	0,7297	0,7456

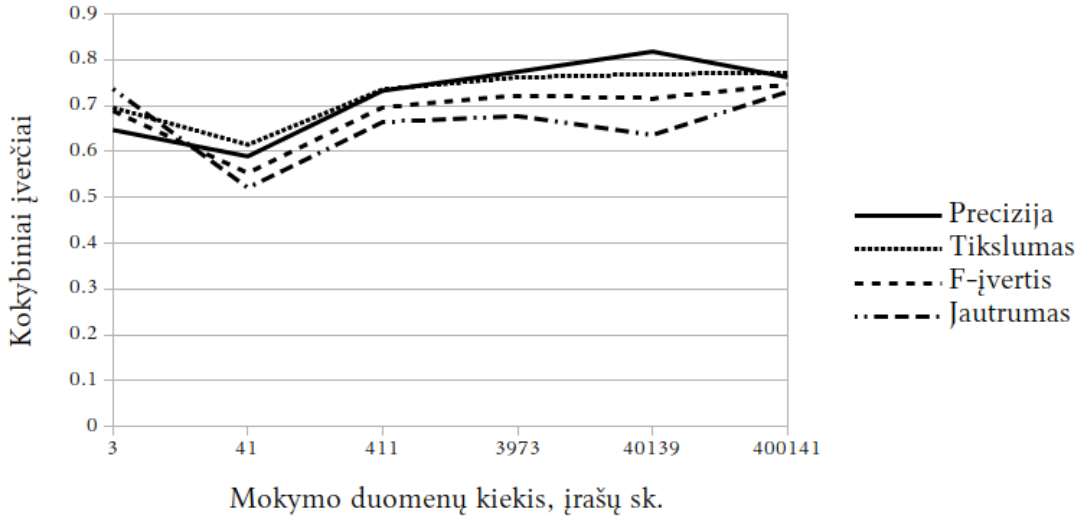


23 pav. Klasifikavimo sprendimų medžio algoritmu vykdymo laikas keičiantis duomenų kiekiui

Septintojo eksperimento tikslas buvo nustatyti, kokią įtaką klasifikavimo rezultatams sprendimų medžio algoritmu turi klasifikatoriams apmokyti skirtų duomenų kiekis. Bandymams pasirinkta naudoti 5000000 SUSY duomenų rinkinio elementų, kurių viena dalis buvo paskirta klasifikavimo modelio formavimui, o kita – testavimui. Vykdant klasių analizę buvo fiksuojami algoritmo vykdymo laikas, tikslumas, precizija, jautrumas ir F-įvertis. Kaip matoma 8 lentelės rezultatuose ir 24 paveikslėlyje, algoritmo tikslumas augo kartu su mokymo duomenų kiekiu. Vis dėlto, esant mažai mokymo duomenų aibei, jis buvo gana nedidelis ir tik neženkliai viršijo 50 procentų ribą, kurią būtų galima pasiekti klases spėjant atsitiktiniu būdu. Panašūs rezultatai matomi ir nagrinėjant precizijos, jautrumo ir F-įverčio matavimus. Verta dėmesį atkreipti ir į tai, kad algoritmo vykdymo laikas, nors ir nedaug, bet didėjo kartu su mokymo duomenų kiekiu.

8 lentelė. Sprendimų medžio algoritmu vykdyto klasifikavimo laikai ir kokybiniai įverčiai keičiantis mokymo duomenų kiekiui

Mokymo duomenų kiekis	Vykdymo laikas, ms	Tikslumas	Precizija	Jautrumas	F-įvertis
3	29231	0,6955	0,6469	0,7365	0,6888
41	30527	0,6148	0,5895	0,5209	0,5531
411	31805	0,7354	0,7326	0,6641	0,6967
3973	35888	0,7618	0,7739	0,6773	0,7224
40139	40996	0,7686	0,8178	0,6360	0,7156
400141	45972	0,7721	0,7621	0,7301	0,7457



24 pav. Klasifikavimo sprendimų medžio algoritmu kokybiniai įverčiai keičiantis mokymo duomenų kiekiui

Rezultatai ir išvados

Atlikus darbą gauti rezultatai:

1. nustatyta, kad pagrindiniai iššūkiai, su kuriais susiduriama vykdant didžiųjų duomenų tyrybą, susiję su išskirtiniu jų dydžiu, greičiu ir įvairove. Tokių duomenų saugojimas ir apdorojimas viršija įprastų sistemų galimybes, todėl norint išsaugoti informacijos teisingumą ir vertę, reikalingos specialios technologijos ir analitiniai metodai;

2. pastebėta, jog dauguma didžiųjų duomenų yra tik pusiau struktūrizuoti ar visai nestruktūrizuoti. Tai apsunkina jų analizę ir išsaugojimą tradicinėse duomenų tvarkymo sistemose, kuriose apdorojama medžiaga turi atitikti iš anksto apibrėžtas griežtas taisykles ir struktūras;

3. išsiaiškinta, kad klasterizavimo metu nagrinėjami elementai suskirstomi į tarpusavyje panašių objektų grupes. Vis dėlto, tradiciniai klasterių analizės algoritmai ir jų rezultatų vertinimo metodai nėra visiškai tinkami didžiųjų duomenų tyrybai:

    3.1. K-vidurkių algoritmas kiekvieno pakartojimo metu visus nagrinėjamos erdvės elementus priskiria klasteriams ir perskaičiuoja šių grupių centrus;

    3.2. didelis skaičiavimų kiekis atliekamas ir vykdant klasterizavimą lūkesčių maksimizavimo būdu. Tokios klasterių analizės metu visiems duomenims daugybę kartų yra skaičiuojamos tikimybės priklausyti kiekvienam erdvės klasteriui;

    3.3. DBSCAN algoritmas elemento priskyrimą klasterių įvykdo vieno pakartojimo metu, tačiau jame nėra ribojamas maksimalus grupėje esančių duomenų skaičius, todėl išlieka tikimybė, kad esant dideliam nagrinėjamos medžiagos kiekiui ir tankumui, gali būti viršytos algoritmą vykdančio vieno įrenginio atminties ir skaičiavimų ribos;

4. sužinota, jog klasifikavimas analizuojamus duomenis testavimo fazės metu priskiria tikslinėms grupėms, vadinamoms klasėmis ir suformuojamoms mokymo fazėje. Nors šis duomenų tyrybos būdas geba rasti ieškomą informaciją naujai pateiktoje įvestyje, tradicinių klasifikavimo algoritmų naudojimas nėra visiškai pritaikytas didžiųjų duomenų tyrybai:

    4.1. duomenis klasifikuojant Naivaus Bajeso algoritmu suskaičiuojamos priklausymo visoms klasėms tikimybės ir, nors šis būdas remiasi prielaida, kad jos tarpusavyje nėra susijusios, norint efektyviai tirti didžiuosius duomenis, algoritmą reikia tinkamai išlygiagretinti;

    4.2. sprendimų medžio algoritmas elementus nagrinėja mokymo fazėje suformuotais sąlyginiais sakiniais, nurodančiais kiekvienos klasės duomenų kriterijus. Nepaisant to, jog toks algoritmas elementų testavimą atlieka gana greitai, mokymo fazėje suformuotas klasifikavimo modelis yra statiškas ir neprisitaikantis prie analizuojamos srities pasikeitimų, galinčių įvykti srautu pateikiamų didžiųjų duomenų tyryboje;

4.3. K-artimiausių kaimynų algoritmas neturi aiškiai atskirtų mokymo ir testavimo fazių, jis kiekvieno pakartojimo metu formuoja analizei reikalingą klasių modelį, todėl gali lengvai prisitaikyti prie tiriamos srities pasikeitimų. Vis dėlto, didžiųjų duomenų tyrybai šis būdas nėra tinkamas, kadangi nagrinėjant daug elementų, kiekvieną kartą vykdomas klasifikavimo modelio apmokymas tampa sudėtinga ir neefektyvia procedūra;

5. nustatyta, kad klasterizuojant didžiuosius duomenis gali būti taikomos imties ėmimo ir matmenų mažinimo technikos, sudarančios galimybes vienu įrenginiu analizuoti didesnį kiekį duomenų nei įprastai. Tuo tarpu, klasterių analizę atliekant daugelyje įrenginių, algoritmus galima vykdyti lygiagrečiai arba naudojantis paskirstytomis sistemomis;

6. pastebėta, jog didžiųjų duomenų klasifikavimui gali būti taikomas duomenų padalijimu mazgams paremtas horizontalus lygiagretinimas. Taip pat klasių analizėje yra galimi vertikalus lygiagretinimas, kuris gaunamas skirtinguose sistemos mazguose nagrinėjant dalį objekto požymių bei užduočių lygiagretinimas, kuriame įvesties rūšiavimą ir analizę atlieka skirtingos sistemos dalys. Siekiant geresnio klasifikatorių prisitaikymo besikeičiančiai dalykinei sričiai ir srautu pateikiamai medžiagai, nagrinėjant didžiuosius duomenis, tęstiniam modelio apmokymui gali būti naudojami ir pačio algoritmo suklasifikuoti elementai;

7. išsiaiškinta, kad didžiųjų duomenų analizei įprastai naudojamos šiai tyrybai sukurtos paskirstytosios sistemos:

   7.1. išnagrinėtas MapReduce programavimo modelis, kuris skaičiavimus atlieka paskirstytame įrenginių tinkle, atskirai vykdydamas daug transformavimo ir redukcijos funkcijų, paremtų rakto ir reikšmės porų įvestimis. Transformavimo metu duomenys yra sugrupuojami pagal rakto panašumą ir redukcijos metu su šiomis grupėmis atliekami skaičiavimai;

   7.2. aprašyta Hadoop paskirstyta failų sistema, kuri leidžia didelius kiekius duomenų saugoti ir sparčiai pasiekti daugelio įrenginių tinkle. Siekiant pagerinti saugomos medžiagos pasiekiamumą ir apsaugą nuo praradimo, atminties blokai kopijuojami į skirtingus sistemos mazgus;

   7.3. išanalizuotas Apache Spark didžiųjų duomenų variklis, kuris naudojant lanksčius paskirstytuosius duomenų rinkinius leidžia atlikti skaičiavimus paskirstytose sistemose. Be to, jame prieinama MLlib biblioteka suteikia galimybę klasterizavimui ir klasifikavimui naudoti jau sukurtas algoritmų realizacijas;

8. išbandytas Apache Spark variklis ir MLlib bibliotekos algoritmai. Jais atliktos klasterizavimo ir klasifikavimo užduotys su medžiaga gauta iš mašininio mokymo duomenų rinkinių saugyklų. Nustatyta, kad ir esant dideliam duomenų kiekiui, Apache Spark variklis ir MLlib pasiekiamas funkcionalumas gali efektyviai atlikti duomenų analizę, tačiau jos vykdymo laikas priklauso ne tik nuo nagrinėjamų duomenų kiekio, bet ieškomų klasterių skaičiaus ar klasifikavimo modelio apmokymo dydžio.

Atlikus darbą suformuluotos tokios išvados:

1. didžiųjų duomenų analizę tradiciniais klasterizavimo ir klasifikavimo algoritmais galima vykdyti taikant ne algoritmų pakeitimus, o paruošiant nagrinėjamus duomenis ir analizuojant tik jų dalį ar požymių poaibį bei gautus rezultatus pritaikant visai nagrinėjamos medžiagos erdvei;

2. nagrinėjant didžiuosius duomenis, galima taikyti įprastų klasterizavimo ir klasifikavimo algoritmų pakeitimus, leidžiančius jų dalis vykdyti lygiagrečiai arba paskirčius daugelio įrenginių tinkle. Nors tai apsunkina analizės funkcionalumo įgyvendinimą, daug skaičiavimų atliekančiuose algoritmuose taip galima ženkliai pagerinti vykdymo laiką randant lokalius rezultatus ir juos jungiant tik globalioje erdvėje. Vis dėlto, verta atkreipti dėmesį, kad toks vykdymas yra labiau tinkamas algoritmams, atliekantiems daug atskirų skaičiavimų. Be to, paskirstytas ir lygiagretus vykdymas reikalauja papildomų pastangų ir dalies sistemos resursų skaičiavimų sinchronizavimui ir juos vykdančių įrenginių komunikacijai;

3. dažniausiai taikomas būdas atlikti didžiųjų duomenų klasterizavimą ir klasifikavimą yra algoritmus vykdant šiai duomenų tyrybai skirtomis sistemomis, tokiomis kaip MapReduce, Hadoop ar Spark. Pritaikius jas, analizę vykdantys specialistai gali visą dėmesį skirti informacijos ieškančio funkcionalumo kūrimui ir nesirūpinti duomenų ar skaičiavimų paskirstymu. Darbe atlikti skaičiavimai ir nagrinėta literatūra parodė, kad tokios sistemos gali efektyviai ir kokybiškai atlikti klasterių ir klasių analizes net ir esant dideliam kiekiui nagrinėjamų duomenų;

4. klasterizavimo vykdymo laikui įtakos turi ne tik analizuojamų duomenų kiekis, bet ir ieškomų klasterių skaičius. Atliekant eksperimentus pastebėta, jog ieškomų grupių kiekiui didėjant, didėjo ir analizei reikalingas laikas. Ši tendecija išliko tiek klasterizuojant elementus K-vidurkių algoritmu, tiek Gauso sumaišymo būdu. Tikėtina, kad tai buvo susiję su poreikiu kiekvieno pakartojimo metu perskaičiuoti klasterių centrus bei priklausymo šioms grupėms tikimybes. Vis dėlto, Apache Spark didžiųjų duomenų variklyje klasterizavimas Gauso sumaišymo būdu yra lėtesnis nei K-vidurkių algoritmu net ir analizuojant tuos pačius duomenis;

5. Apache Spark didžiųjų duomenų variklyje vykdant klasifikavimą Naivaus Bajeso algoritmu gaunami geresni kokybiniai įverčiai nei sprendimų medžio algoritmu, tačiau jie priklauso nuo to, kiek duomenų naudojama klasifikavimo modelio apmokymui. Eksperimentuose taip pat pastebėta, kad mokymo aibės kiekis didesnę įtaką gali turėti klasifikuojant duomenis ne sprendimų medžio, o Naivaus Bajeso algoritmu.

Nagrinėjant šią temą tolesniuose darbuose būtų galima įvertinti kokios technologijos ir metodai gali būti taikomi ankstyvam dalinai išanalizuotų didžiųjų duomenų analizės rezultatų pateikimui ir vizualizavimui, kuris yra reikalingi kokybiškam ir savalaikiui informacijos interpretavimui tiriant didelius kiekius medžiagos ir nenutrūkstamus duomenų srautus, būdingus didžiųjų duomenų sričiai.

Šaltiniai

[ABL+17] Mehdi Assefi, Ehsun Behravesh, Guangchi Liu ir Ahmad P. Tafti. Big data machine learning using apache spark MLlib. *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, 2018-January:3492–3498, 2017. DOI: 10.1109/BigData.2017.8258338.

[Agg14] Charu C Aggarwal. *Data classification: algorithms and applications*. CRC press, 2014, p. 2–36, 160–169. ISBN: 1466586753.

[AOO12] Adigun Adebisi, Omidiora Olusayo ir Olabiyisi Olatunde. An Exploratory Study of K-Means and Expectation Maximization Algorithms. *British Journal of Mathematics & Computer Science*, 2(2):62–71, 2012. DOI: 10.9734/bjmcs/2012/1036.

[Apa19] Apache Software Foundation. Apache Hadoop 3.2.0 – HDFS Architecture, 2019. URL: https://hadoop.apache.org/docs/r3.2.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. Tikrinta 2020-05-08.

[BGB18] Hind Bangui, Mouzhi Ge ir Barbora Buhnova. Exploring big data clustering algorithms for internet of things applications. *IoTBDS 2018 - Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security*, 2018-March(IoTBDS 2018):269–276, 2018. DOI: 10.5220/0006773402690276.

[BKL16] Malika Bendechache, M. Tahar Kechadi ir Nhien An Le-Khac. Efficient large scale clustering based on data partitioning. *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016*:612–621, 2016. DOI: 10.1109/DSAA.2016.70. arXiv: 1704.03421.

[BM12] Erik Brynjolfsson ir Andrew McAfee. Big Data : The Management Review. *Harvard Business Review*, (October):1–12, 2012. ISSN: 0017-8012. URL: http://tarjomefa.com/wp-content/uploads/2017/04/6539-English-TarjomeFa-1.pdf.

[BT14] S M Aqil Burney ir Humera Tarig. K-Means Cluster Analysis for Image Segmentation. *International Journal of Computer Applications*, 96(4):1–8, 2014.

[CCA+19] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy ir Russell Sears. MapReduce online. *Proceedings of NSDI 2010: 7th USENIX Symposium on Networked Systems Design and Implementation*:313–327, 2019.

[CLL17] Min Chen, Simone A. Ludwig ir Keqin Li. Clustering in big data. *Big Data Management and Processing*:333–346, 2017. DOI: 10.1201/9781315154008.

[DG04] Jeffrey Dean ir Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *OSDI 2004 - 6th Symposium on Operating Systems Design and Implementation*:137–149, 2004. ISSN: 23487852. DOI: 10.21276/ijre.2018.5.5.4.

[DGG15] Andrea De Mauro, Marco Greco ir Michele Grimaldi. What is big data? A consensual definition and a review of key research topics. *AIP Conference Proceedings*, 1644:97–104, 2015. ISSN: 15517616. DOI: 10.1063/1.4907823.

[DGG19] Andrea De Mauro, Marco Greco ir Michele Grimaldi. Understanding Big Data Through a Systematic Literature Review: The ITMI Model. *International Journal of Information Technology and Decision Making*, 18(4):1433–1461, 2019. ISSN: 02196220. DOI: 10.1142/S0219622019300040.

[DMP13] Enric Junqué De Fortuny, David Martens ir Foster Provost. Predictive modeling with big data: Is bigger really better? *Big Data*, 1(4):215–226, 2013. ISSN: 2167647X. DOI: 10.1089/big.2013.0037.

[EAM06] Jeffrey Erman, Martin Arlitt ir Anirban Mahanti. Traffic classification using clustering algorithms. *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data, MineNet'06*, 2006:281–286, 2006. DOI: 10.1145/1162678.1162679.

[EDS16] David Eckold, Karl Dearn ir Duncan Shepherd. Wear debris from total joint replacements: evaluation of automated categorisation by scale-invariant feature transforms. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging and Visualization*, 2016:1–7, 2016. DOI: 10.1080/21681163.2016.1230075.

[IZE11] IBM, Paul Zikopoulos ir Chris Eaton. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 1st leid., 2011. ISBN: 0071790535, 9780071790536.

[KKA+16] Raghavendra Kune, Pramod Kumar Konugurthi, Arun Agarwal, Raghavendra Rao Chillarige ir Rajkumar Buyya. The anatomy of big data computing. *Software: Practice and Experience*, 46(1):79–105, 2016. DOI: 10.1002/spe.2374. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2374.

[Läm07] Ralf Lämmel. Google's MapReduce programming model - Revisited. *Science of Computer Programming*, 68(3):208–237, 2007. ISSN: 01676423. DOI: 10.1016/j.scico.2007.07.001.

[LBC+13] Bingwei Liu, Erik Blasch, Yu Chen, Dan Shen ir Genshe Chen. Scalable sentiment classification for Big Data analysis using Naïve Bayes Classifier. *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*:99–104, 2013. DOI: 10.1109/BigData.2013.6691740.

[Mar14] Bernard Marr. Big data: the 5 vs everyone must know. https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know/, 2014. Tikrinta 2019-12-03.

[MBY+16] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks ir k.t. MLlib: Machine learning in Apache Spark. *Journal of Machine Learning Research*, 17:1–7, 2016. ISSN: 15337928.

[Mur13] Arinto Murdopo. Distributed decision tree learning for mining big data streams. *Master of Science Thesis, European Master in Distributed Computing*, (July):55, 2013.

[OBA+18] Ahmed Oussous, Fatima Zahra Benjelloun, Ayoub Ait Lahcen ir Samir Belfkih. Big Data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, 30(4):431–448, 2018. ISSN: 22131248. DOI: 10.1016/j.jksuci.2017.06.001. URL: https://doi.org/10.1016/j.jksuci.2017.06.001.

[Raj16] V Rajaraman. Big data analytics. *Resonance*, 21(8):695–716, 2016-08. ISSN: 0973-712X. DOI: 10.1007/s12045-016-0376-7. URL: https://doi.org/10.1007/s12045-016-0376-7.

[SAW+14] Ali Seyed Shirkhorshidi, Saeed Aghabozorgi, Teh Ying Wah ir Tutut Herawan. Big data clustering: A review. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8583 LNCS(PART 5):707–720, 2014. ISSN: 16113349. DOI: 10.1007/978-3-319-09156-3_49.

[SKR+10] K Shvachko, H Kuang, S Radia ir R Chansler. The Hadoop Distributed File System. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, p. 1–10, 2010. DOI: 10.1109/MSST.2010.5496972.

[Sur17] Sarika Surampudi. Oracle data mining concepts, 12c release 1. https://docs.oracle.com/database/121/DMCON/toc.htm, 2017. Tikrinta 2019-12-31.

[Sut16] Shan Suthaharan. Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, 36:1–12, 2016.

[TLC+15] Chun Wei Tsai, Chin Feng Lai, Han Chieh Chao ir Athanasios V. Vasilakos. Big data analytics: a survey. *Journal of Big Data*, 2(1):1–32, 2015. ISSN: 21961115. DOI: 10.1186/s40537-015-0030-3.

[VSC+12] Manish Verma, Mauly Srivastava, Neha Chack, Atul Kumar Diswar ir Nidhi Gupta. A Comparative Study of Various Clustering Algorithms in Data Mining. *International Journal of Engineering Research and Applications www.ijera.com*, 2(3):1379–1384, 2012.

[ZXW+16] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das ir k.t. Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016. ISSN: 15577317. DOI: 10.1145/2934664.