KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Aistė Buivytė

POŽYMIŲ ATRANKOS METODŲ TAIKYMAS KLASIFIKAVIMO UŽDAVINIAMS SPRĘSTI

Baigiamasis magistro projektas

Vadovas

Doc. dr. Vytautas Janilionis

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

POŽYMIŲ ATRANKOS METODŲ TAIKYMAS KLASIFIKAVIMO UŽDAVINIAMS SPRĘSTI

Baigiamasis magistro projektas

Taikomoji matematika (kodas 621G10003)

Vadovas

Doc. dr. Vytautas Janilionis

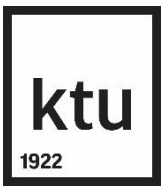
Recenzentas

Doc. dr. Tomas Ruzgas

Projektą atliko

Aistė Buivytė

KAUNAS, 2016



KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Aistė Buivytė

Taikomoji matematika (kodas 621G10003)

Baigiamojo projekto „Požymių atrankos metodų taikymas klasifikavimo uždaviniams spręsti“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2016 m. gegužės mėn. 30 d.

Kaunas

Patvirtinu, kad mano, **Aistės Buivytė,** baigiamasis projektas tema „Požymių atrankos metodų taikymas klasifikavimo uždaviniams spręsti“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymu nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

TURINYS

Buivytė, Aistė. Požymių atrankos metodų taikymas klasifikavimo uždaviniams spręsti. *Magistro* baigiamasis projektas / vadovas doc. dr. Vytautas Janilionis; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Mokslo kryptis ir sritis: Matematika, Fiziniai mokslai

Reikšminiai žodžiai: *klasifikavimas, atraminių vektorių klasifikatorius, rekursinė požymių atranka, genetinis algoritmas, modeliuoto atkaitinimo algoritmas.*

Kaunas, 2016. 81 p.

SANTRAUKA

Analizuojant įvairius duomenis, dažnai taikomi klasifikavimo metodai. Šiuo metu, klasifikavimo uždavinio sprendimą apsunkina tai, kad duomenų ir klasifikuojamų požymių kiekiai yra labai dideli, todėl efektyvi reikšmingų požymių atranka – aktualus uždavinys.

Baigiamojo darbo tikslas – pasiūlyti metodiką ir programines priemones, kurios atrinktų mažiau požymių, bet ženkliai nepablogintų klasifikavimo kokybės rodiklių. Sprendžiant praktinius uždavinius, svarbus veiksnys yra laikas, reikalingas klasifikavimo algoritmų vykdymui. Siekiant pagreitinti klasifikavimo uždavinių sprendimą, darbe jungiami genetinio algoritmo, modeliuoto atkaitinimo ir rekursinės požymių atrankos metodai su atraminių vektorių klasifikatoriumi. Gauti klasifikavimo rezultatai lyginami su atsitiktinių miškų vidiniu požymių atrankos metodu, panaudojant kelias klasifikavimo kokybės metrikas. Gauti klasifikavimo rezultatai vaizduojami, panaudojant šilumos grafikus. Pasiūlyta metodika realizuota programiškai, panaudojant R, Python ir SAS programavimo kalbas.

Sukurtos metodikos ir programinių priemonių taikymas parodė, kad jos sprendžia darbe suformuluotus uždavinius.

Baigiamojo darbo rezultatai pristatyti studentų konferencijoje „Matematika ir matematikos dėstymas – 2016“ – KTU. Gauti rezultatai buvo publikuoti konferencijos leidinio straipsnyje.

Aiste, Buivyte. *Application of Feature Selection Methods to Solve Classification Problems*: *Master's* thesis in Applied Mathematics / supervisor assoc. prof. Vytautas Janilionis. The Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Research area and field: Mathematics, Physical Sciences

Key words: *classification, support vector machine, recursive feature selection, genetic algorithm, simulated annealing algorithm*.

Kaunas, 2016. 81 p.

SUMMARY

Classification problems are widely applied on purpose to analize various data. Nowadays, if the number of data and features is large in a dataset, classification problems become more difficult, thus the effective selection of significant features is a relevant task.

The aim of master's work is to suggest a methodology and software tools which would let to select a smaller subset of original features but would not diminish classification quality measures significantly. Solving practical problems, an important factor is the time required for performance of classification algorithms in machine learning. In order to speed up the solution of classification problems, genetic algorithm, simulated annealing and recursive feature elimination methods are joined with support vector classifier. The results achieved by previous implementations are compared with interior features selection method of random forest using several classification quality metrics. The classification results are displayed using heat maps. The proposed methodology was implemented programmatically using R, Python and SAS programming languages.

Developed methodology and software tools has shown that they solve the tasks assigned in work.

Final results of work were presented at student conference called „Mathematics and Teaching of Mathematics – 2016" – KTU. The results were published in the article of conference.

SANTRUMPOS

AVK – atraminių vektorių klasifikatorius

GA – genetinis algoritmas (angl. *genetic algorithm*)

MA – modeliuotas atkaitinimas (angl. *simulated annealing*)

NP – polinominis sudėtingumas (angl. *nondeterministic polynomial*)

RPA – rekursinė požymių atranka (angl. *recursive feature elimination*)

ĮŽANGA

„Faktų klasifikavimas yra mokslo funkcija.“ – Karl Pearson

Duomenų tyrybos procesas apima tam tikrus žingsnius, tokius kaip duomenų rinkinio parinkimas, duomenų pirminis apdorojimas, duomenų analizė, rezultatų interpretavimas ir vertinimas. Klasifikavimas yra uždavinys, priskiriantis stebėjimus atitinkamoms klasėms. Daugelis realaus pasaulio uždavinių gali būti įvardijami kaip klasifikavimo uždaviniai, pavyzdžiui, kaip klasifikuoti gaunamus laiškus, paciento ligos diagnozę, genų alelius, klientus pagal mokumą ar kita. Požymių atranka priskiriama prie pirminio duomenų tyrybos etapo uždavinio. Dažnai požymių skaičius duomenų tyrybos uždaviniuose yra didelis. Turint daug požymių, atsiranda didelės atminties reikalavimas, lėtas vykdymo greitis, per didelis jautrumas triukšmui. Ne visi požymiai yra vienodai informatyvūs. Požymių atrankos tikslas – pašalinti nereikšmingus klasifikavimui požymius. Apgalvotas parinkimas, kuriuos požymius įtraukti į modelį, turi didesnį poveikį bendrai klasifikatoriaus kokybei, nei paties klasifikavimo metodo parinkimas. Analitika ugdo gilesnį supratimą apie sprendžiamą problemą, todėl kiekvienas matematinis algoritmas turi atitikti teoriją ir paaiškinti procesą.

Pirmoje baigiamojo darbo dalyje išanalizuoti literatūroje minimi požymių atrankos ir duomenų klasifikavimo metodai. Pastebėta, kad nėra daug publikacijų apie požymių atrankos ir klasifikavimo metodų junginius. Darbo tikslas yra pasiūlyti metodiką ir programines priemones, kurios atrinktų mažiau požymių, bet ženkliai nepablogintų klasifikavimo kokybės rodiklių. Antroje dalyje aprašoma metodika, derinanti originalių požymių atranką ir klasifikavimo metodus. Iš esmės sprendžiami du tarpusavyje susiję uždaviniai, atliekamas realizuotų metodų palyginimas. Pasiūlytai metodikai įgyvendinti buvo pasitelktos Python, R ir SAS programinių įrangų aplinkos, kurių realizacija pateikta 5, 6, 7 ir 8 prieduose (žr. 5, 6, 7 ir 8 priedus). Trečioje projekto dalyje išsamiai aprašomi gauti rezultatai. Požymių atrankai buvo pasirinktas genetinis algoritmas, rekursinis požymių atrankos algoritmas ir modeliuoto atkaitinimo algoritmas. Gauti požymiai buvo klasifikuojami, panaudojant atraminių vektorių klasifikavimo metodą. Kiekvienam klasifikavimo modeliui apskaičiuotos klasifikavimo kokybės metrikos, atskirai apmokymo ir testavimo imtims. Klasifikavimo rezultatai buvo lyginami su atsitiktinių miškų vidiniu požymių atrankos metodo gautais rezultatais.

1. LITERATŪROS APŽVALGA

Šioje dalyje pristatomi klasifikavimo uždavinių sprendimo iššūkiai, pateikiama požymių atrankos ir klasifikavimo metodų apžvalga, duomenų analizės atviro kodo programinės įrangos, pristatomas baigiamojo projekto darbo tikslas ir uždaviniai.

1.1. KLASIFIKAVIMO UŽDAVINIŲ SPRENDIMO PROBLEMOS

Klasifikavimas yra tikslo funkcijos *f* apmokymo uždavinys, sprendžiantis kaip kiekvienas duomenų aibės stebėjimas *x* yra priskiriamas vienai iš anksto nustatytai klasės kintamojo reikšmei *y*. Klasifikavimo uždaviniai yra priskiriami prie klasifikavimo su mokytoju (angl. *supervised learning*) metodų. Klasifikavimas dažnai suvokiamas kaip du atskiri uždaviniai, t. y. dvinaris (angl. *binary*) klasifikavimas arba kelių klasių (angl. *multiclass*) klasifikavimas. Klasę paaiškinantys kintamieji gali būti tolygūs (skaitinio tipo), kategoriniai, žodiniai, tekstiniai. Klasifikatorius – matematinė klasifikavimo metodo funkcija, kuri įvesties duomenis priskiria klasės kintamiesiems. Klasifikavimo uždavinio procesas yra dalinamas į kelis etapus: duomenų paruošimą (pradinių duomenų dalinimą į apmokymo ir testavimo imtis), požymių atrinkimą, klasifikatoriaus apmokymą, kur remiantis požymių vektoriumi atliekamas klasifikavimas.

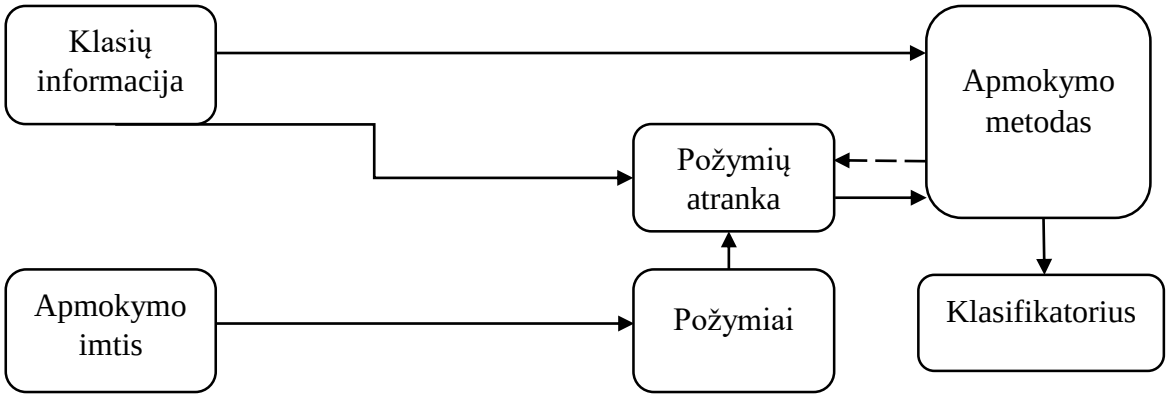
Klasifikuojant duomenis vis dažniau susiduriama su įvairiais iššūkiais. Didėjant duomenų dimensijai, daugelis duomenų analizės metodų ir klasifikavimo uždavinių tampa gerokai sunkesni. Duomenys su daug požymių gali sumažinti klasifikavimo tikslumą, duoti prastą kokybę tarp klasių, per didelias skaičiavimo sąnaudas ir atminties trūkumą. Gali būti per mažai duomenų objektų, o tai darytų įtaką nepatikimo modelio sukūrimui, priskiriant klasę visiems galimiems objektams. Kai požymių duomenyse yra daug, atributų erdvėje mažinimas supaprastina įvairių vizualizavimo metodų panaudojimą. Literatūroje pateikta nemažai metodų požymių atrankai ir dimensijos mažinimui. Požymių atrankos privalumas yra tas, kad neprarandama informacija apie atskiro požymio reikšmingumą, lengviau interpretuoti gautus rezultatus. Kita vertus, kai analizuojamas mažas požymių rinkinys ir pradiniai požymiai yra labai skirtingi, klasifikavimo informacija prarandama, jei bent vienas požymis yra praleidžiamas. Požymių atrankos metodų taikymas reikalauja gero analizuojamų duomenų išmanymo. Taikant dimensijos mažinimo metodus, požymių erdvės dydis gali ženkliai sumažėti, bet prarandama daug pradinės požymių erdvės informacijos. Svarbus dimensijos mažinimo metodų trūkumas yra tai, kad tiesiniai originalių požymių dariniai yra sunkiai interpretuojami, o informacija apie tai, kokį įnašą įneša pradiniai požymiai, prarandama [1].

1.2. POŽYMIŲ ATRANKOS IR KLASIFIKAVIMO METODŲ APŽVALGA

Technologinės naujovės turi poveikį visuomenei ir moksliniams tyrimams. Jos leidžia surinkti didžiulį duomenų kiekį su palyginti mažomis sąnaudomis. Požymių atranka ir klasifikavimas gali būti du tarpusavyje derinami metodai.

Požymių atrankos metodai

Požymių atranka klasifikuojant duomenis yra pageidautina dėl to, kad mažėja klasifikavimo uždavinio sprendimo laikas ir ženkliai palengvėja klasifikavimo rezultatų interpretacija. Požymių poaibio atranka (angl. *feature selection*), kitaip vadinama kintamųjų arba atributų atranka, tai procesas, kurio metu atrenkamas pradinių požymių poaibis, atmetant nereikšmingus arba perteklinius požymius pagal tam tikrus kriterijus [2]. Požymių atranka priskiriama mašininio mokymosi (angl. *machine learning*) ir duomenų tyrybos (angl. *data mining*) sritims. Požymių atranka – *NP* sunkus optimizavimo uždavinys. Požymių atranka daro didelę įtaką klasifikatorių apmokymui [3]. Vietoj tiesioginio duomenų klasifikavimo su visais požymiais, požymiai gali būti atrenkami. Klasifikavimo uždaviniui, įtraukiant požymių atranką, spęsti skirta schema yra vaizduojama 1.2.1 paveiksle (žr. 1.2.1 pav.).



1.2.1 pav. Požymių atrankos schema klasifikavimui [3]

Požymių atrankos metodai literatūroje [2] yra skirstomi į tris kategorijas (klases): filtro (angl. *filter*) metodus, aplanko (angl. *wrapper*) metodus, integruotus (angl. *embedded*) metodus. Požymių atrankos metodai skiriasi nuo dimensijos mažinimo metodų. Abu metodai siekia sumažinti originalių skaičių duomenų rinkinyje. Dimensijos mažinimo metodai sukuria naujus, „latentinius“ požymius, kuriuos sunku interpretuoti praktikoje, o požymių atrankos metodai sumažina originalių požymių skaičių. Dimensijos mažinimo metodai apima pagrindinių

komponenčių analizę (angl. *principal component analysis*), atskirosios reikšmės skaidymą (angl. *singular value decomposition*) ir Samono algoritmą (angl. *Sammon's mapping*). Požymių atrankos metodai yra naudingi, nes, turint mažiau požymių, atsiranda galimybė ženkliai nesumažinti klasifikavimo tikslumo ir palengvinti klasifikavimo rezultatų supratimą. Požymių atrankos metodai jungia ir atrankos algoritmą, ir modelio sudarymą.

Požymių atrankos filtro metodai [3] taiko statistinį matą ir taip įvertina įnašą kiekvienam požymiui. Požymiai yra rikiuojami pagal gaunamus rezultatus ir išlaikomi arba pašalinami iš duomenų. Šie metodai yra vienamačiai arba daugiamačiai. Jie apsvarsto kiekvieną požymį nepriklausomai arba atsižvelgiant į priklausomą kintamąjį. Pavyzdžiais galėtų būti Chi – kvadrato kriterijus, informacijos prieaugis, koreliacijos koeficientai. Filtro modelis atskiria požymių atranką nuo klasifikatoriaus apmokymo taip, kad šie du algoritmai nesąveikauja vienas su kitu. Filtro metodai yra veiksmingi dėl gana trumpo skaičiavimo laiko ir atsparūs persimokymui (angl. *overfitting*).

Aplanko metodai [3] požymių rinkinio atranką vertina kaip paieškos uždavinį, kuriame konstruojamos, apskaičiuojamos skirtingos kombinacijos, jos lyginamos su kitomis kombinacijomis, nustatomos galimos sąveikos tarp kintamųjų. Aplanko modelis naudoja nuspėjamą tikslumą, kurį iš anksto nustato mokymosi metodas, siekiant sužinoti pasirinktų požymių kokybę. Paieškos procesas gali būti metodinis (pavyzdžiui, kaip trumpiausio kelio paieška (angl. *best – first search*)), stochastinis (pavyzdžiui, laiptiniai algoritmai (angl. *hill – climbing*)), euristinis (pavyzdžiui, veiksnių įtraukimo algoritmas (angl. *forward*) arba veiksnių išmetimo (angl. *backward*) algoritmas). Aplanko metodai turi tris pagrindines komponentes: požymių atrankos paiešką (t. y., kaip rasti požymių poaibį iš visų galimų požymių), požymių įvertinimą (t. y., kaip įvertinti pasirinkto klasifikatoriaus įvykdymą) ir indukcijos algoritmą. Aplanko metoduose klasifikatoriaus apmokymas prilyginamas juodajai dėžei. Požymių paieškos komponentas parodo požymių aibę, o požymių įvertinimo komponentas naudoja klasifikatorių, įvertina efektyvumą, atlieka požymių poaibio atrankos iteraciją. Klasifikatoriui apmokyti parenkama tik aukščiausia reikšme įvertinta požymių aibė. Gautas klasifikatorius vertina nepriklausomą testavimo rinkinį, kuris nėra naudojamas apmokymo proceso metu. Aplanko metodai dažnai yra pernelyg „brangūs“ duomenims su dideliu požymių skaičiumi. Šis trūkumas kyla dėl ilgo skaičiavimo proceso. Aplanko metodams yra priskiriamas RPA metodas.

Integruoti metodai [3] leidžia sužinoti, kurie požymiai geriausiai prisideda prie kuriamo modelio tikslumo. Šis metodas naudoja statistinius kriterijus, parenka kelis kandidatinių požymių poaibius su tam tikru galingumu ir išskiria poaibį su didžiausiu klasifikavimo tikslumu. Integruoti metodai apjungia filtro ir aplanko metodus. Tuo pačiu metu vyksta ir modelio pritaikymas, ir požymių atranka. Labiausiai paplitęs integruotų požymių atrankos metodų tipas yra

reguliarizavimo (angl. *regularization*) metodai su tikslo funkcijomis, kurios mažina tinkamumo paklaidas, koeficientai tampa maži arba lygūs nuliui. Požymiai, kurių koeficientai yra arti nulio, pašalinami. Dėl gaunamų gerų rezultatų, reguliarizavimo modeliai sulaukia nemažai dėmesio. Reguliarizavimo metodai taip pat yra vadinami baudos metodais, kurie nustato papildomus optimizavimo apribojimus prognozavimo algoritmui, todėl modelio sudėtingumas mažėja. Reguliarizavimo metodų pavyzdžiais galėtų būti mažiausio absoliutaus susitraukimo ir atrankos operatorius (L1) (angl. *least absolute shrinkage and selection operator* (trump. *LASSO*)), elastinis tinklas (L1+L2) (angl. *elastic net*) ir gūbrinė regresija (angl. *ridge regression*).

Straipsnyje [4] teigiama, kad požymių atranka 50 % gali pagerinti klasifikavimo modelio tikslumą. Straipsnio autorius pateikia kelis algoritmus, skirtus požymių atrankai ir priskiriamus prie aplanko metodų: skruzdžių kolonijos optimizavimo (angl. *ant colony optimization*) algoritmą, atkaitinimo modeliavimą (angl. *simulated annealing*), dalelių spiečiaus optimizavimo (angl. *particle swarm optimization*) algoritmą.

Apie dalelių spiečiaus optimizavimo algoritmo taikymą požymiams parinkti kalbėjo Bing Xue [5]. Jis taikė kelių tikslo funkcijų dalelių spiečiaus optimizavimą, generuodamas nedominuojančių sprendinių (požymių pogrupių) Pareto frontą. Algoritmas buvo pritaikytas ant 12 kontrolinių duomenų rinkinių. Eksperimentai parodė, kad toks optimizavimas gali automatiškai vystyti daug nedominuojančių sprendinių rinkinių ir parinkti tinkamiausią požymių poaibį.

Keliose mokslinėse publikacijose kalbama apie didelius duomenų rinkinius, kuriuos norima klasifikuoti, bet požymių yra daug, todėl reikalinga efektyvi požymių atranka. Viename iš straipsnių [6], požymių atrankos skaičiavimams siūlo naudoti laikui imlias strategijas, kurios vieną po kito įtraukia požymius į požymių pogrupius, panaudojant baigtinį skaičių iteracijų. Ypatingai didelių matmenų duomenų požymių atranka yra vykdoma per požymių generavimo „mechanizmus“ (angl. *feature generating machine*). Vietoj optimizavimo atlikimo visiems įvesties požymiams, požymių generavimo „mašinos“ įtraukia informatyviausius požymius ir tada sprendžia sumažintą kelių branduolių apmokymo uždavinį. Siūloma optimizavimo schema imituoja persimokymo strategiją, mažinant požymių atrankos paklaidą ir neįdedant daug pastangų. Be to, požymių atrankos paklaida gali būti efektyviai „sušvelninama“, atskirai kontroliuojant sprendimo funkcijos sudėtingumą, kas ir yra pagrindinis šio metodo privalumas.

Keli autoriai yra pasiūlę evoliuciniais skaičiavimais pagrįstus metodus požymių atrankai. Guerra – Salcedo [7] rašė apie hibridinį, genetinį požymių atrankos algoritmą, kuris yra greitas ir tikslus. Landry su kitais autoriais [8] analizavo genetiniu programavimu pagrįstą požymių atrankos „techniką“. Bala ir kiti autoriai [9] naudojo GA požymių įvertinimui.

Straipsnyje [10] siūlomas evoliuciniu algoritmu pagrįstas požymių atrankos metodas, kuris naudoja „MapReduce“ paradigmą, t. y., programinės įrangos programavimo modelį, leidžiantį

lygiagrečiai apdoroti didelius duomenų rinkinius, išgaunant požymių pogrupius iš didžiųjų duomenų. Metodas išskaido originalius duomenis į blokus ir iš jų pasimoko. Po to, gauti daliniai rezultatai apjungiami į galutinį požymių svorių vektorių. Požymių atrankos metodas yra įvertinamas, naudojant tris gerai žinomus klasifikatorius: AVK, logistinę regresiją ir Naivaus Bajeso metodą. CHC metodas yra binarinis GA. Jis įtraukia pusiau tolyginį kryžminimą. Toks kryžminimas atsitiktinai parenka pusę bitų, kurie skiriasi tarp abiejų tėvų. Gaunami du palikuonys, kurie turi didžiausią Hemingo atstumą iš tėvų. Toliau vykdoma elitinė atranka. Čia kiekvienoje kartoje nauja populiacija yra sudaroma iš geriausių individų, renkamų tarp tiek dabartinių, tiek palikuonių populiacijų. Esant lygiosioms, parenkami tėvai. Po elitinės atrankos vykdomas kryžminimas. Dviem individams neleidžiama kryžmintis, jeigu Hemingo panašumas tarp jų viršija ribinę d (paprastai inicializuojamą $d = \frac{L}{2}$, kur L yra chromosomos ilgis). Ribinė vertė mažėja, kai negaunamas joks palikuonis, o tai rodo, jog algoritmas konverguoja. Procesas yra atnaujinamas. Kai $d = 0$ (taip atsitinka po kelių kartų negavus jokių naujų palikuonių), populiacija laikoma nekintančia. Tokiu atveju, nauja populiacija yra generuojama išlaikant geriausią individą, o likę individai turi tam tikrą procentinį išmestų bitų dydį. CHC metodas pagal klasifikaciją patenka į aplanko metodų grupę. CHC metodas natūraliai prisitaiko prie požymių atrankos uždavinio, nes kiekvienas požymis gali būti apibrėžtas kaip bitas sprendimo vektoriuje. Kiekviena vektoriaus pozicija nurodo, ar atitinkamas požymis yra atrenkamas. Tinkamumo funkcija, naudojama įvertinti naujus individus, pritaiko k – artimiausių kaimynų klasifikatorių duomenų rinkiniui, gautam pašalinus atitinkamus požymius. Tinkamumo reikšmė yra svertinė k – artimiausių kaimynų suma.

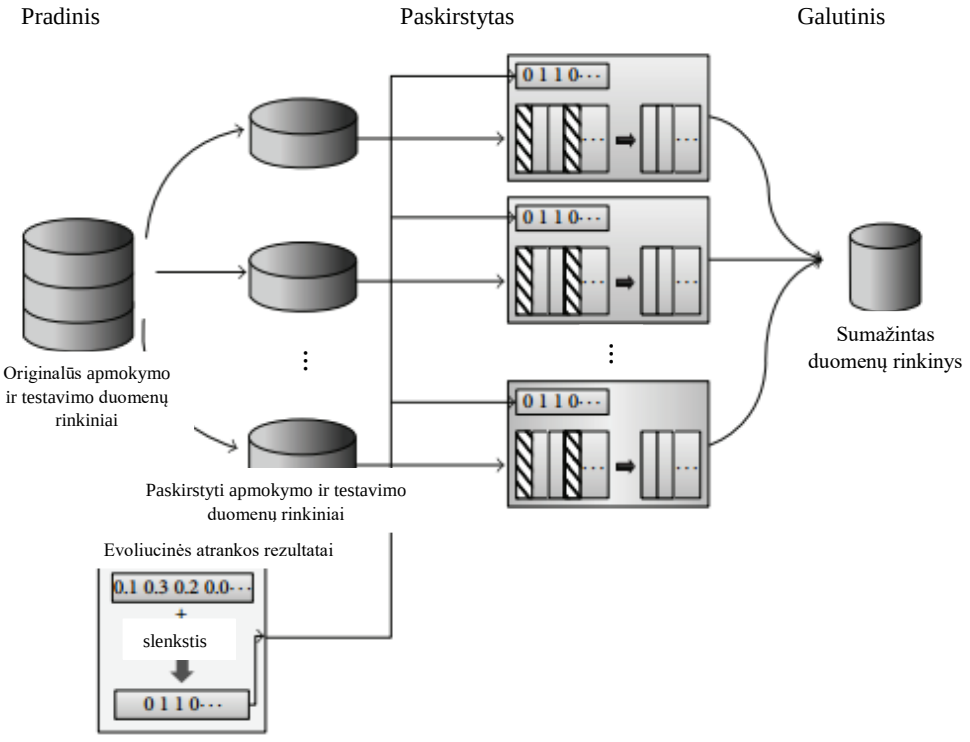
Straipsnyje [10] minimas CHC metodo sugretinimas, panaudojant „MapReduce“ procedūrą ir išgaunant svorių vektorius. Tegul T yra apmokymo rinkinys, saugomas „Hadoop‘o“ (t. y., atviro kodo programinėje įrangoje ir aplikacijų visumoje (angl. *framework*) didelių duomenų kaupimui ir apdirbimui) paskirstytoje failų sistemoje. Turime m skaičių duomenų padalinimo užduočių. „MapReduce“ suskaidymo procedūra dalina T į m nesikertančių virtualių mašinų arba aplikacijos objektų (angl. *instance*) poaibių. Kiekvienas T_i poaibis, kur $i \in \{1, \dots, m\}$, yra apdorojamas atitinkamos duomenų padalinimo užduoties. Kadangi šis padalinimas atliekamas nuosekliai, visi poaibiai turi apytiksliai tą patį virtualių mašinų skaičių ir atsitiktinė T failų atranka užtikrina tinkamą klasių pusiausvyrą. Duomenų dalinimo etapas po kiekvieno T_i susideda iš evoliucinio požymių atrankos algoritmo (šiuo atveju, remiantis CHC metodu). Vadinasi, kiekvienos duomenų dalinimo užduoties išvestis yra binarinis vektorius $f_i = \{f_{i1}, \dots, f_{iD}\}$, kur D – požymių skaičius, kuris nurodo CHC metodo atrinktus požymius. Redukcijos etape imamas visų binarinių vektorių vidurkis, išgaunant vektorių (žr. (1) formulę):

$$x = \{x_1, \dots, x_D\}, \tag{1}$$

kur $x_j = \frac{1}{m}\sum_{i=1}^{m} f_{ij}$ ir $j \in \{1, \dots, D\}$ yra dalis evoliucinės požymių atrankos, įtraukiančios j – ąjį požymį. Gaunamas bendras evoliucinės požymių atrankos rezultatas (žr. 1.2.2 pav.). Vektorius x_j naudojamas sukurti sumažintą duomenų rinkinį, kuris toliau būtų imamas mašininio apmokymo tikslais. Kai apskaičiuojamas vektorius x, sekantis tikslas yra iš originalaus duomenų rinkinio pašalinti mažiau perspektyvius požymius. Sukuriamas papildomas „MapReduce“ procesas. Vektorius x paverčiamas į binarinį pavidalą, naudojant slenkstį (angl. *threshold*) θ (žr. (2) formulę):

$$b = \{b_1, \dots, b_D\}, \tag{2}$$

kur $b_j = \begin{cases} 0, \text{jeigu } x_j \geq \theta, \\ 1, \text{kitu atveju.} \end{cases}$ Vektorius b parodo, kokie požymiai parenkami sumažintam duomenų rinkiniui. Parenkamų požymių skaičius $D' = \sum_{j=1}^{D} b_j$ kontroliuojamas su slenksčiu θ. Su aukštu slenksčiu parenkami tik keli požymiai, o žemesnis slenkstis leidžia atrinkti daugiau požymių. Slenksčio nustatymas leidžia pagerinti klasifikavimo modelio specifiškumą ir jautrumą prie tam tikrų sąlygų.



1.2.2 pav. Duomenų rinkinio dimesijos mažinimo proceso schema [10]

Klasifikavimo metodai

Klasifikatoriaus konstravimui ir vertinimui yra naudojamas 10 – lypis kryžminio patikrinimo (angl. *10 – fold cross – validation*) metodas, kuris pradinę imtį padalina, pavyzdžiui, į apmokymo ir testavimo duomenų rinkinius, t. y. į *k* tarpusavyje nesuderinamų poaibių arba blokų. Atliekamas perskaičiavimas ir jeigu tikslumas tampa priimtinas, procesas baigiamas. Literatūroje [11, 12, 13, 14, 15, 16, 17, 18] yra minimi keli duomenų klasifikavimui skirti metodai, kurie aprašomi žemiau.

***K* – artimiausių kaimynų klasifikatorius**

K – artimiausių kaimynų klasifikatorius priskiriamas prie neparametrinių klasifikavimo metodų. Kaimynai yra randami, remiantis atstumo metrika, pavyzdžiui, Euklido atstumu (žr. (3) formulę):

$$d(x, y) = \sqrt{\sum_j (x_i - y_i)^2}, \tag{3}$$

kur $x_i = x_1, x_2, \dots, x_m$ ir $y_i = y_1, y_2, \dots, y_m$ yra dviejų taškų *m* požymių reikšmės.

Naujas nesuklasifikuotas stebėjimas yra priskiriamas tai klasei, kuriai yra priskiriami artimiausi stebėjimai iš apmokymo duomenų rinkinio. Kai matuojamas atstumas, o tam tikri požymiai turi dideles reikšmes, lyginant su kitais požymiais, tuomet būtina požymių reikšmes normalizuoti, pavyzdžiui, atliekant *Z* – mato (angl. *Z – score*) standartizavimą (žr. (4) formulę):

$$X^* = \frac{X - vidurkis(X)}{standartinis\_nuokrypis(X)}. \tag{4}$$

Kategoriniams duomenims Euklido metrika netinka. Kategoriniams duomenims apibrėžiama funkcija, kuri leidžia palyginti *i* – tojo požymio vertes taip (žr. (5) formulę) [11]:

$$d(x_i, y_i) = \begin{cases} 0, \text{jei } x_i = y_{i,} \\ 1, \text{kitu atveju.} \end{cases} \tag{5}$$

Tiesinė diskriminantinė analizė

Pagrindinė tiesinės diskriminantinės analizės idėja yra rasti tiesinę transformaciją, kuri geriausiai diskriminuotų tarp klasių. Matematiškai tiesinė diskriminantinė analizė yra vykdoma per matricų skaidymo analizę. Naudojant tiesinę diskriminantinę analizę, siekiama projekcijos, kuri geriausiai atskiria duomenis. Nauji objektai remiasi apmokymo imtimi, o jų priklausymas klasėms nėra iš anksto žinomas [12].

Tarkime, kad turime *n* klasių (baigtinį skaičių), o požymiai (kintamieji) yra pasiskirstę pagal normalųjį pasiskirstymą ir išmatuoti intervalų skalėje. Tada vidinių klasių (angl. *intra – class*) matrica yra apskaičiuojama (žr. (6) formulę):

$$\sum_w = S_1 + \cdots + S_n = \sum_{i=1}^{n} \sum_{x \in c_i} (x - \overline{x_\iota})(x - \overline{x_\iota})^i, \quad (6)$$

kur c_i – klasės, x_i – p – dimensijos imtys, $\overline{x_\iota} = \frac{1}{m_i} \sum_{x \in c_i}$, m_i – stebėjimų skaičius klasėse.

Tarp klasių (angl. *inter – class*) skaidymo matrica yra apibrėžiama kaip (žr. (7) formulę):

$$\sum_b = \sum_{i=1}^{n} m_i (\overline{x_\iota} - \bar{x})(\overline{x_\iota} - \bar{x})', \quad (7)$$

kur m_i – apmokymo imties stebėjimų skaičius kiekvienai klasei, $\overline{x_\iota}$ – kiekvienos klasės vidurkis, $\bar{x} = \frac{1}{m} \sum_{i=1}^{n} m_i \overline{x_\iota}$ – bendro vidurkio vektorius.

Gavus 6 ir 7 lygtis (žr. (6) ir (7) formulę), tiesinė transformacija Φ turi būti maksimizuota. Transformacija Φ yra gaunama, sprendžiant apibendrintą tikrinių reikšmių uždavinį (žr. (8) formulę):

$$\sum_b \Phi = \lambda \sum_w \Phi. \quad (8)$$

Taigi, klasifikavimui yra naudojami diskriminantiniai požymiai. Turint Φ, klasifikavimas yra atliekamas transformuojant erdvę ir remiantis tam tikromis atstumo metrikomis. Tada naujas stebėjimas z yra priskiriamas (žr. (9) formulę):

$$\arg \min_k d(z\Phi, \overline{x_k}\Phi), \quad (9)$$

kur $\overline{x_k}$ yra k – tosios klasės centras [12].

Naivaus Bajeso klasifikatorius

Bajeso (angl. *Bayesian*) klasifikatoriai yra statistiniai klasifikatoriai. Jie prognozuoja klasės priklausomumo tikimybę, t. y. tikimybę, kad tam tikra aibė priklauso tam tikrai klasei (grupei). Bajeso klasifikavimas yra pagrįstas Bajeso teorema. Klasifikavimo metodus lyginantys tyrimai yra parodę, kad savo veikimu Bajeso klasifikatorius, žinomas kaip Naivaus Bajeso klasifikatoriumi, yra panašus į sprendimo medžių (angl. *decision tree*) ir dirbtinių neuroninių tinklų (angl. *artificial neural network*) klasifikatorius. Bajeso klasifikatoriai rodo aukštą tikslumą ir yra greiti, kai analizuojami dideli duomenų masyvai. Naivaus Bajeso klasifikatoriai leidžia manyti, kad požymių vertės tam tikroje klasėje poveikis yra nepriklausomas nuo kitų požymių reikšmių.

Tegul X – duomenų rinkinys. H – hipotezė, kad duomenų rinkinys X priklauso tam tikrai klasei C. $P(H|X)$ ir $P(X|H)$ yra aposteriorinės tikimybės, o $P(H)$ ir $P(X)$ – apriorinės tikimybės. Bajeso teorema apibrėžiama (žr. (10) formulę):

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}. \quad (10)$$

Naivaus Bajeso klasifikatoriaus veikimas yra apibūdinamas keliais etapas. Pirma, tegul D – apmokymo rinkinys su požymiais ir stebėjimus apibūdinančiais klasės kintamaisiais. Kaip įprasta,

kiekvienas duomenų rinkinys turi n – dimensijos požymių vektorių $X = (x_1, x_2, \dots, x_n)$, vaizduojantį n – matavimų, atitinkamai $A_1, A_2, \dots, A_n$. Antra, tarkime, kad yra m klasių $C_1, C_2, \dots, C_m$. Klasifikatorius prognozuoja, kad X yra siejama su klase, turinčia didžiausią aposteriorinę tikimybę. Naivaus Bajeso klasifikatorius prognozuoja, kad duomenų stebėjimas priklauso klasei C_i tada ir tik tada, jei (žr. (11) formulę):

$$P(C_i|X) > P\big(C_j|X\big), \tag{11}$$

kur $1 \leq j \leq m$ ir $j \neq i$. Taigi, $P(C_i|X)$ yra maksimizuojama. Klasė C_i yra vadinama maksimalia aposteriorine hipoteze, kur $P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$ apskaičiuojama pagal Bajeso teoremą. Trečia, kadangi $P(X)$ yra konstanta visoms klasėms, tik $P(X|C_i)P(C_i)$ yra maksimizuojama. Jeigu klasės apriorinė tikimybė yra nežinoma, tada bendrai manoma, jog klasės yra vienodai tikėtinos, t. y. $P(C_1) = P(C_2) = \cdots = P(C_m)$ ir tada tik maksimizuojama $P(X|C_i)$ tikimybė. Kitu atveju, maksimizuojama $P(X|C_i)P(C_i)$ tikimybė. Klasės apriorinė tikimybė gali būti apskaičiuojama kaip $P(C_i) = \frac{|C_{i,D}|}{|D|}$. Ketvirta, jeigu duotas duomenų rinkinys su daug požymių, tada sunku apskaičiuoti $P(X|C_i)$ tikimybę. Norint sumažinti tikimybės $P(X|C_i)$ skaičiavimus, daroma prielaida, kad požymių reikšmės yra sąlyginai nepriklausomos viena nuo kitos, atsižvelgiant į klasės kintamąjį. Taigi, $P(X|C_i)$ yra gaunama (žr. (12) formulę):

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i). \tag{12}$$

Jeigu A_k požymiai yra kategoriniai, tada $P(x_k|C_i)$ yra aibės D, turinčios reikšmes x_k požymiams A_k klasės C_i skaičius, padalinamas iš $\big|C_{i,D}\big|$. Jeigu A_k požymiai yra tolydūs, tada tolydūs požymiai turi Gauso pasiskirstymą su vidurkiu ir standartiniu nuokrypiu. Tikimybė $P(x_k|C_i)$ yra apibrėžiama (žr. (13) formulę):

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{13}$$

taip, kad (žr. (14) formulę):

$$P(x_k|C_i) = g\big(x_k, \mu_{C_i}, \sigma_{C_i}\big). \tag{14}$$

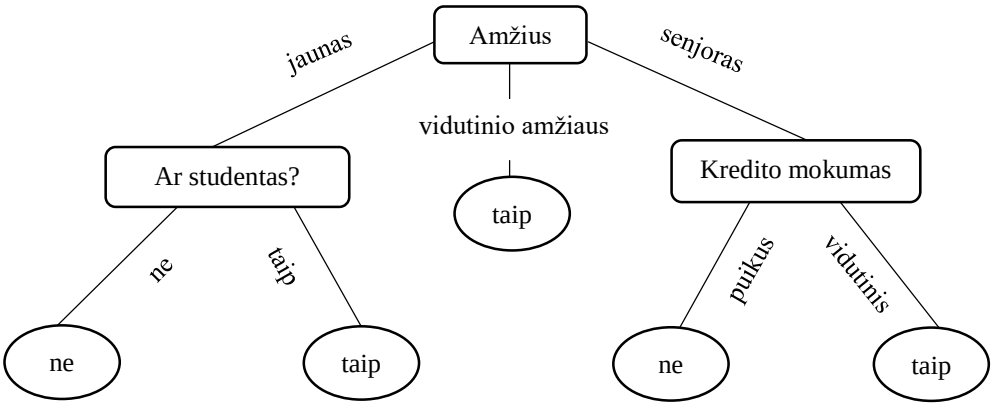
Penkta, norint nustatyti klasę stebėjimui, tikimybė $P(X|C_i)P(C_i)$ yra skaičiuojama kiekvienai klasei C_i. Klasifikatorius priskirs stebėjimą klasei C_i tada ir tik tada, jeigu (žr. (15) formulę):

$$P(X|C_i)P(C_i) > P\big(X|C_j\big)P\big(C_j\big), \tag{15}$$

kur $1 \leq j \leq m$ ir $j \neq i$. Duomenys klasifikuojami, kai tikimybė $P(X|C_i)P(C_i)$ yra maksimali [13].

Sprendimų medžiai

Sprendimo medis yra struktūrinė schema, panaši į medžio struktūrą, kurioje kiekvienas vidinis mazgas (viršūnė) reiškia testą, kiekviena šaka duoda testo rezultatus, o kiekvienas mazgo lapas – klasės pavadinimą. Viršutinis mazgas medyje yra šakninis mazgas. Sprendimo medžiai yra priskiriami prie neparametrinių klasifikavimo metodų. Tipinė sprendimo medžio struktūra yra pavaizduota 1.2.3 paveiksle (žr. 1.2.3 pav.). Čia sprendžiama, ar visi klientai gali įsigyti kompiuterį.



1.2.3 pav. Sprendimų medžio struktūra

Sprendimo medžio algoritmas turi tris parametrus: *D* (duomenų suskaldymą), požymių sąrašą, požymių atrankos metodą. Iš pradžių, turimas pilnas apmokymo duomenų kortežas (angl. *tuple*) ir su juo susiję klasių pavadinimai. Požymių atrankos metodas nurodo euristinės atrankos procedūros atributą, kuris geriausiai atskiria duotas klasifikavimo klases. Ši procedūra taiko požymių atrankos matus, tokius kaip informacijos prieaugis (angl. *information gain*) ir Gini indeksas (angl. *Gini index*). Gini indeksas reikalauja binarinio medžio. Informacijos prieaugis leidžia turėti kelias klases (iš mazgo išeina dvi arba daugiau šakų).

Kai sudaromas sprendimo medis, daugelis šakų atspindi apmokymo duomenų anomalijas, o taip yra dėl triukšmo ir išskirčių. Medžių genėjimo metodai leidžia išspręsti duomenų persimokymo problemą. Tokie metodai paprastai naudoja statistines priemones ir pašalina mažiausiai patikimas šakas. Apgenėti medžiai yra linkę būti mažesni ir ne tokie sudėtingi, todėl juos suprasti yra lengviau. Apgenėti medžiai yra greitesni ir geriau teisingai klasifikuoja nepriklausomus testavimo duomenis, nei negenėjami medžiai. Medžių genėjimas turi du bendrus metodus: prieš – genėjimas (angl. *prepuning*) ir po – genėjimas (angl. *postpruning*). Prieš – genėjimas reikalauja daugiau skaičiavimo laiko. Vietoj genėjamų medžių, kurie remiasi klaidų santykio nustatymu, galima genėti medžius, atsižvelgiant į bitus, reikalingus užkoduoti jiems. Geriausiais apgenėtas medis yra tas, kuris minimizuoja užkoduotų bitų skaičių. Šis metodas

remiasi minimalaus aprašymo ilgio (angl. *minimum description length*, trump. *MDL*) principu, kuris turi mažiausią skaičių bitų.

ID3 sprendimo medžio metodas naudoja informacijos prieaugio matą, tačiau neturi medžio genėjimo etapo. *ID3* metodas tinka klasifikuoti tik kategorinius kintamuosius. Tikėtina informacija, kurios reikia klasifikuoti kortežą *D* yra apskaičiuojama (žr. (16) formulę):

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i), \tag{16}$$

čia p_i yra nenulinė apriorinė tikimybė, gaunama $p_i = \frac{|C_{i,D}|}{|D|}$, kur C_i – klasė. Logaritmo pagrindas yra lygus dviem, nes informacija koduojama bitais. $Info(D)$ yra vadinama kortežo (arba aibės) *D* entropija (angl. *entropy*).

Tarkime, kad požymis *A* turi *v* skirtingas reikšmes, t. y. $\{a_1, a_2, \dots, a_v\}$, kurios stebimos iš apmokymo duomenų. Požymis *A* dalina aibę *D* į poaibius, t. y. į $\{D_1, D_2, \dots, D_v\}$, kur D_j apibūdina tuos *D* stebėjimus, kurie turi *A* išvestį a_j. Poaibiai atitiktų šakas, einančias iš mazgo *N*. Apie tai, kiek po padalinimo reikia informacijos, kad būtų atlikta tiksli klasifikacija, skaičiuojama pagal 17 formulę (žr. (17) formulę):

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j), \tag{17}$$

kur $\frac{|D_j|}{|D|}$ veikia kaip *j* – ojo dalinimo svoris. $Info_A(D)$ yra tikėtina informacija, kuri reikalauja klasifikuoti aibę *D* pagal požymius *A*. $Info_A(D)$ turi būti kiek įmanoma mažesnis.

Informacijos prieaugis yra apibrėžiamas kaip skirtumas tarp pirminio informacijos įvertinio (pagrįsto klasių proporcingumu) ir naujo informacijos kiekio (gauto po aibės padalinimo pagal požymį *A*) (žr. (18) formulę):

$$Gain(A) = Info(D) - Info_A(D). \tag{18}$$

Pasirenkamas tas požymis A, kuris turi didžiausią informacijos prieaugį ($Gain(A)$).

C4.5 metodas yra *ID3* metodo praplėtimas. Metodas medžio auginimo etape naudoja informacijos prieaugio santykį (angl. *gain ratio*). Taikoma padalinimo informacijos vertė, kuri gaunama (žr. (19) formulę):

$$SplitInfo_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right). \tag{19}$$

Taigi, informacijos prieaugio santykis yra apibrėžiamas kaip (žr. (20) formulę):

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}. \tag{20}$$

Požymis, su maksimaliu informacijos prieaugio santykiu yra pasirenkamas. Atkreipiamas dėmesys, kad, kai matas priartėja prie nulio, santykis tampa nestabilus. Tokiu atveju, įdedamas apribojimas, pagal kurį pasirinkto testo informacijos prieaugis privalo būti didelis, t. y. bent toks didelis, kaip vidutinis prieaugis, gautas per visus tirtus testus.

C4.5 metodas naudoja algoritmą, vadinamą pesimistiniu genėjimu. Jis skaičiuoja klaidų rodiklį ir daro išvadas apie po – medžio genėjimą. Pesimistinis genėjimas naudoja apmokymo imtį ir skaičiuoja paklaidos įverčius. Tikslumo ir klaidos įverčiai yra pagrįsti apmokymo duomenų rinkiniu, optimistiniai, todėl daro didelę įtaką klasifikavimui. Pesimistinis genėjimo būdas reguliuoja klaidų įvertį, įdedant baudą (angl. *penalty*).

Klasifikavimo ir regresijos medžiuose (angl. *classification and regression trees*, trump. ***CART***) požymių atrankai yra naudojamas Gini indeksas. Jo matematinis apibrėžimas yra (žr. (21) formulę):

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2, \tag{21}$$

kur p_i yra apriorinė tikimybė, gaunama $p_i = \frac{|C_{i,D}|}{|D|}$. Suma skaičiuojama per visas *m* klases.

Gini indeksas laikosi binarinio padalinimo kiekvienam požymiui. Jeigu požymis *A* dalina aibę D į D_1 ir D_2, D aibės Gini indeksas, duodantis tokį padalinimą, išreiškiamas (žr. (22) formulę):

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2). \tag{22}$$

Kiekvienam požymiui skaičiuojami visi įmanomi binariniai požymių padalinimai. Diskretiems požymiams pasirenkamas tas padalinimo poaibis, kuris požymiui duoda minimalų Gini indeksą. Tolydiems požymiams apsvarstomi visi galimi padalinimo taškai.

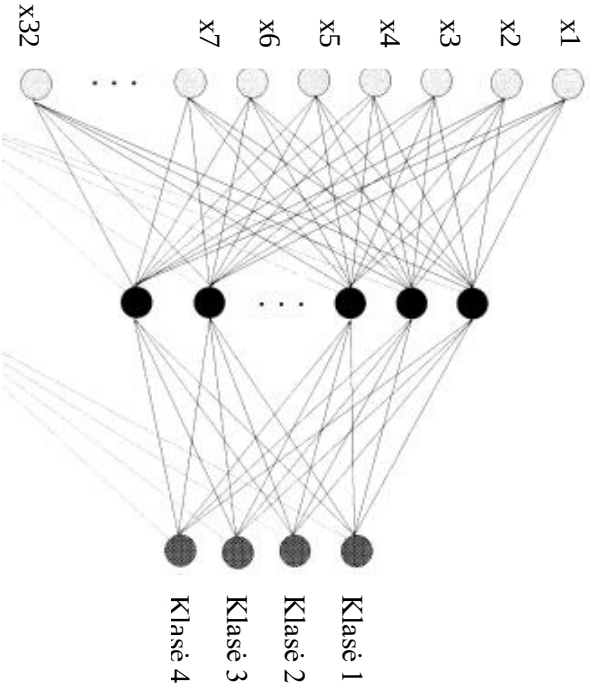
Požymių aibės parinkimui maksimizuojama išraiška (žr. (23) formulę):

$$\Delta Gini(A) = Gini(D) - Gini_A(D). \tag{24}$$

CART metodas naudoja medžio sudėtingumo skaičiavimui skirtą genėjimo algoritmą, kuris priskiriamas prie po – genėjimo rūšies. Medžio sudėtingumas – funkcija, kuri priklauso nuo lapų skaičiaus medyje ir medžio klaidų rodiklio. Medžių klaidų rodiklis (santykis) – neteisingai suklasifikuotų stebėjimų procentas. Medžio genėjimas prasideda nuo medžio apačios. Kiekvienam vidiniam mazgui *N*, apskaičiuojamas taip vadinamo po – medžio sudėtingumas. Jeigu po – medis keičiamas į lapą, sudėtingumas perskaičiuojamas. Dvi gautos reikšmės tarpusavyje palyginamos. Jeigu apgenėtas po – medis mazge *N* duoda mažesnį sudėtingumą, tada po – medis genamas. Priešingu atveju, paliekamas toks pat [13].

Neuroninių tinklų metodas

Dirbtiniai neuroniniai tinklai iš pradžių buvo sukurti, pamėgdžiojant žmogaus nervų sistemą. Tinklas susideda iš neuronų ir ryšių tarp jų, kurie kartu nustato tinklo elgesį. Neuroninis tinklas susideda iš trijų arba daugiau neuronų sluoksnių: įėjimo sluoksnis, išėjimo sluoksnis ir bent vienas paslėptasis sluoksnis (žr. 1.2.4 pav.). Perceptronu yra vadinamas dirbtinio tinklo neuronas. Kiekvieno sluoksnio neuronus jungia aksonas [14].



1.2.4 pav. Neuroninio tinklo struktūra [14]

Neuroniniai tinklai yra netiesinis metodas. Neuroninio tinklo pagalba klasifikavimo išėjimo reikšmę galima apskaičiuoti (žr. (25) formulę):

$$Y_j = \varphi\left(U_j + t_j\right), \tag{25}$$

kur $U_j = \sum\left(X_i \cdot w_{ij}\right)$, φ – aktyvavimo funkcija (dažnai sigmoidinė funkcija), t_j – slenksčio reikšmė. Čia kiekvienam j – tajam neuronui, kiekvienas įvesties kintamasis X_i yra dauginamas iš svorio w_{ij}. Jie kartu susumuojami. Modelio parametrai yra apskaičiuojami didžiausio tikėtinumo metodu [15]. Neuroniniai tinklai yra skaičiuojami be grįžtamojo ryšio.

Logistinė regresija

Logistinė regresija taikoma prognozuoti kategorinio klasės kintamojo reikšmes, tiksliau tikimybes. Logistinė regresija yra panaši į vieno perceptrono modelį. Logistinė regresija nereikalauja kintamųjų normalumo pasiskirstymo sąlygos. Logistinė regresija gali būti skirstoma

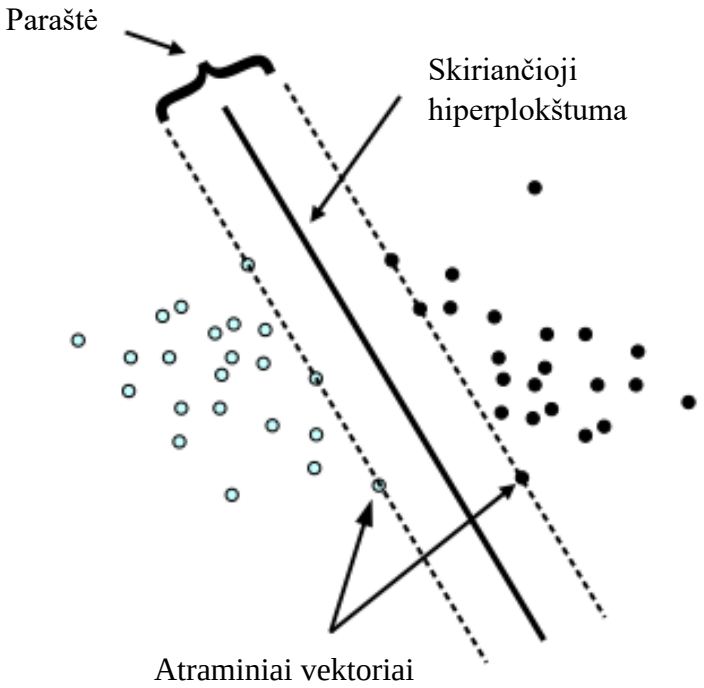
į dvinarę, daugianarę arba ranginę. Matematinė tikimybė klasės kintamajam įgyti tam tikrą reikšmė yra užrašoma (žr. (26) formulę):

$$p_i = \frac{e^{z(x_i)}}{1+e^{z(x_i)}} = \frac{1}{1+e^{-z(x_i)}}, \tag{26}$$

čia X_i – nepriklausomi kintamieji, $Z(X_i) = a + b_1X_{1i} + \cdots + b_kX_{ki}$. Kaip ir neuroninių tinklų atveju, logistinės regresijos parametrai yra apskaičiuojami didžiausio tikėtinumo metodu**.** Jeigu tikimybė prognozuojamam išvesties (klasės) kintamajam yra didesnė už 0,5, tada stebėjimas yra priskiriamas klasei [16]. Logistinės regresijos modelis yra realizuotas daugelyje programinių paketų. Logistinės regresijos modelis savyje įtraukia požymių atrankos algoritmus, todėl atskiras požymių atrankos uždavinys yra praleidžiamas.

Atraminių vektorių klasifikatorius

AVK bendrai yra priskiriamas prie netiesinio klasifikavimo metodų. Klasifikuojami duomenys atskiriami hiperplokštuma. Jeigu negalima rasti tiesinės hiperploštumos, duomenų taškai yra projektuojami į aukštesnės dimensijos erdvę. Toks projektavimas yra pasiekiamas per branduolio (angl. *kernel*) metodus. Visa užduotis gali būti formuluojama kaip kvadratinio optimizavimo uždavinys. AVK yra naudojamas rašysenos ranka atpažinimui. Žemiau pateikiama klasifikavimo, panaudojant atraminius vektorius, vizualizacija (žr. 1.2.5 pav.) [17].



1.2.5 pav. AVK klasifikavimo vizualizacija konkrečiam pavyzdžiui [17]

AVK trūkumas yra tai, kad klasifikavimo rezultatas yra grynai ideologinis, o klasės tikimybės nėra apskaičiuojamos. Jeigu apmokymo duomenų yra daug, procesas tampa lėtas [18].

1.3. PROGRAMINĖS ĮRANGOS APŽVALGA IR PARINKIMAS

Siekiant realizuoti požymių atrankos metodus ir atlikti duomenų klasifikavimą, buvo panaudotos Python ir R programinės įrangos. Apache Spark yra greita, įtraukianti interaktyvias užklausas ir bendro pobūdžio klasteriams skaičiuoti skirta platforma. Greitis yra svarbus, norint apdoroti didelius duomenų rinkinius, nes yra skirtumas, ar duomenis apdoroti interaktyviai, ar laukti nemažai valandų. Apache Spark suteikia galimybę paleisti skaičiavimus atmintyje. Apache Spark taikomųjų programų programavimo sąsaja (angl. *application programming interface*, trump. *API*) palaiko Python, R, Java, SQL kalbas. Apache Spark įtraukia MapReduce modelį, leidžiantį lygiagrečiai apdoroti didelius duomenų rinkinius. Dėl greitų skaičiavimų ir toliau aprašomų patogių naudojimui bibliotekų, buvo pasirinkta PySpark platforma, kuri paleidžiama per taip vadinamą „Git Bash“ komandinį lauką.

Pagrindinės naudotos Python programavimo kalbos bibliotekos:

- „Scikit – learn“ – biblioteka, skirta mašininiam apmokymui Python aplinkoje. Tai yra atviro kodo įrankis, sukurtas duomenų tyrybai ir analizei;
- „Pandas“ – atviro kodo biblioteka, teikianti aukštos kokybės, lengvo naudojimo duomenų struktūras. Tai yra duomenų analizės įrankis Python programavimo kalbai;
- „NumPy“ – plėtinys Python programavimo kalbai, bendrosios paskirties, masyvams apdoroti skirtas paketas. Jis efektyviai valdo didelių matmenų duomenų rinkinius ir leidžia neprarasti per daug greičio, naudojant mažų matmenų masyvus;
- „Matplotlib“ – grafikų braižymui skirta biblioteka;
- „Scipy“ – biblioteka, kuri teikia daug paprogramių skaitiniam integravimui ir optimizavimui. Ji sukurta dirbti kartu su „NumPy“ biblioteka.

R programavimo kalba suteikia platų statistinių ir grafinių technikų pasirinkimą. Ši programavimo kalba vartotojui yra nemokama ir patogiai išplečiama.

Pagrindinės naudotos R programavimo kalbos bibliotekos:

- „Caret“ – klasifikavimo ir regresijos apmokymui skirta biblioteka, įtraukianti duomenų padalinimą, pirminį apdorojimą, požymių atrankos šablonus, modelio parametrų nustatymą, kintamųjų svarbumo įvertinimą;
- „E1071“ – biblioteka, kuri naudojama Furjė transformacijoms, latentinei analizei, klasifikatorių sudarymui;

- „DoParralel“ – biblioteka, leidžianti atlikti lygiagrečius skaičiavimus;
- „GGplot2“ – biblioteka, skirta grafikų braižymui R programinėje įrangoje ir galinti išgauti sudėtingą duomenų vizualizavimą;
- „RandomForest“ – biblioteka, kuri įtraukia klasifikavimą ir regresiją, remiantis atsitiktiniais miškais ir naudojant atsitiktinius įvesties kintamuosius;
- „Lattice“ – biblioteka, kuri leidžia braižyti aukštos kokybės grafikus.

Duomenų požymių klasifikavimo uždaviniui spręsti gali būti naudojamas ir SAS (angl. *Statistical Analysis System*) Enterprise Miner programinis paketas. Šis paketas leidžia vykdyti medžiais pagrįstą požymių atrankos algoritmą. Prie šio paketo jungiamasi debesų kompiuterijos pagalba.

1.4. BAIGIAMOJO DARBO TIKSLAS IR UŽDAVINIAI

Baigiamojo darbo tikslas: pasiūlyti metodiką ir programines priemones, kurios atrinktų mažiau požymių, bet ženkliai nepablogintų klasifikavimo kokybės rodiklių.

Uždaviniai:

1. išanalizuoti literatūrą apie požymių atrankos metodus, klasifikavimo metodus, jų sprendimo problemas ir naudojamas programines įrangas;
2. pasiūlyti metodiką, kuri leistų parinkti geriausią požymių atrankos metodą, klasifikuojant konkrečius duomenis;
3. parinkti tinkamas programines priemones ir realizuoti metodiką programiškai;
4. panaudojus pasiūlytą metodiką ir sukurtas priemones, atlikti realių duomenų klasifikavimą. Palyginti uždavinio sprendimo laikus ir klasifikavimo kokybės metrikas, taikant skirtingus metodus.

2. TYRIMŲ METODAI IR METODIKA

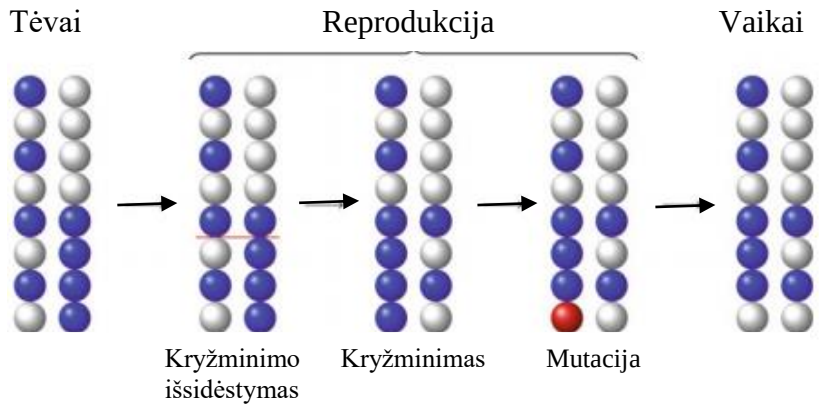
Šiame skyriuje pateikti požymių atrankai atlikti ir duomenų klasifikavimui modeliuoti naudojami metodai, jų programinės realizacijos, pristatoma R ir Python programinių kalbų sąsaja.

2.1. REIKŠMINGŲ KLASIFIKAVIMO POŽYMIŲ ATRANKOS MODELIAI

Duomenų masyvai nuolat didėja, todėl svarbus uždavinys yra požymių atranka klasifikavimo uždaviniams spręsti, kuri leidžia pagreitinti skaičiavimus ir išgauti geresnį klasifikavimo rezultatų interpretavimą. Darbe yra naudojamas GA, MA algoritmas, RPA algoritmas, atsitiktinių miškų klasifikavimo modeliai.

2.1.1. GENETINIO ALGORITMO TAIKYMAS POŽYMIŲ ATRANKAI

GA imituoja Darvino natūralios atrankos teoriją ir randa kai kurių funkcijų optimalias reikšmes. GA turi daug taikymo sričių. Šiame darbe GA yra taikomas reikšmingų požymių atrankai. GA priskiriamas prie aplanko metodų grupės. Iš pradžių sukuriamas pradinis sprendimų rinkinys su atitinkamomis tinkamumo reikšmėmis, kur didesnė reikšmė yra geresnė. Šis sprendimų rinkinys yra vadinamas populiacija (angl. *parents*), o kiekvienas sprendinys – individu (angl. *children*). Kuriama tokia populiacija, kuri kiekviename žingsnyje yra geresnė. Individai su geriausiomis tinkamumo reikšmėmis yra derinami atsitiktinai, gaminami palikuonys, kurie paveldi genus (chromosomas) iš abiejų tėvų ir sudaro kitą populiaciją. Individai yra kryžminami (angl. *crossover*) (atkartojama genetinė reprodukcija), pakeičiant nedidelę dalį genetinių duomenų ir taikomos atsitiktinės mutacijos (angl. *mutation*). Procesui kartojantis, pagaminama daug kartų su vis geresniais sprendimais. Bendra GA schema pateikiama 2.1.1.1 paveiksle (žr. 2.1.1.1 pav.).



2.1.1.1 pav. Reprodukcinio etapo GA schema [19]

Tarkime, kad turime požymių aibę $f = \{f_1, f_2, \dots, f_{m-1}, f_m\}$. Požymių atrankos uždavinyje individai yra prognozės kintamųjų poaibiai. Čia chromosomos (genų seka) yra užkoduojamos dvejetainiu formatu (1 arba 0), t. y. požymis yra įtraukiamas į poaibį arba ne. Tinkamumo reikšmė (E_i) yra modelio apibūdinimo matas, pavyzdžiui, *RMSE* (vidutinių kvadratinių paklaidų reikšmė) arba klasifikavimo kokybės metrikos. Tikslo funkcija yra optimizuojamas dydis. GA pagrįstas optimizavimo procesas gali būti labai agresyvus, galimas persimokymas, bet atliekama „gili“ paieška. Max Kuhn ir Kjell Johnson [19] yra aprašę pseudokodą (žr. 2.1.1.2 pav.) požymių atrankai pritaikytam GA:

1. Nustatomas algoritmo stabdymo kriterijus (generavimų skaičius), vaikų skaičius (*Vaik_sk*) kiekvienai generacijai (nebūtinai) ir mutacijos tikimybė (p_m);
2. Generuojamas pradinis, atsitiktinai parinktų m binarinių chromosomų rinkinys, kurių kiekvienos ilgis yra $l = 2$;
3. **kartojama**,
4. ▏ **kiekvienai** *chromosomai* **atliekama**:
5. ▏ ▏ modelio parametrų nustatymas ir kiekvienos chromosomos tinkamumo apskaičiavimas
6. ▏ **užbaigiama**
7. ▏ **kiekvienai** *reprodukcijai* $k = 1, \dots, \frac{Vaik\_sk}{2}$ **atliekama**:
8. ▏ ▏ parenkamos dvi tinkamumo kriterijumi pagrįstos chromosomos
9. ▏ ▏ kryžminimas: atsitiktinai parenkama padėtis (atsitiktinė pozicija) ir iš abiejų pusių pakeičiamas kiekvienas chromosomos genas
10. ▏ ▏ mutacija: kiekvienoje naujo vaiko chromosomoje, kiekviena geno binarinė reikšmė yra atsitiktinai pakeičiama su tikimybe p_m
11. ▏ **užbaigiama**
12. **kol** *tenkinamas sustojimo kriterijus*.

2.1.1.2 pav. Pseudokodas požymių atrankai atlikti, panaudojant GA

Atsižvelgiant į požymių atrankos procesą, chromosoma yra binarinis vektorius, kuris turi tokį patį ilgį, kaip ir nepriklausomi kintamieji duomenų rinkinyje. Kiekvienas chromosomos arba geno binarinis įėjimas parodo kiekvieno nepriklausomo kintamojo buvimą arba nebuvimą duomenyse. Chromosomos tinkamumas yra nustatomas pagal modelį, naudojant nepriklausomus kintamuosius kaip binarinius vektorius. GA randa optimalius sprendimus su visomis nepriklausomų kintamųjų rinkinių kombinacijomis. Kryžminimo etapas veda vėlesnes kartas link optimumo panašių genetinių duomenų poerdviuose. Kitaip tariant, paieškos poerdviai yra susiaurinami į erdvę, kurioje atsitiktinai apibrėžiamos labiausiai tinkamos chromosomos. Tai reiškia, kad algoritmas gali įstrigti į lokalų optimumą. Jeigu kontekstas yra požymių atranka, parinkti požymiai gali apibūdinti optimalų modelį, bet gali egzistuoti ir kiti optimalūs požymių porinkiniai. Mutacijos

etapas leidžia algoritmui išvengti lokalaus optimumo, atsitiktinai sutrikdant genetinius duomenis. Paprastai mutacijos tikimybė yra laikoma maža, tarkim, $p_m = 0{,}05$. Visgi, jeigu vartotojas svarsto apie lokalaus optimumo galimybę, tada mutacijos tikimybė gali būti padidinta. Mutacijos tikimybės kėlimo poveikis sulėtina konvergavimą į optimalų sprendimą [19].

GA realizacija yra pateikta R programinėje kalboje. GA atrinkti požymiai yra apmokomi su AVK.

2.1.2. MODELIUOTO ATKAITINIMO ALGORITMO TAIKYMAS POŽYMIŲ ATRANKAI

Egzistuoja kelios optimizuotos paieškos procedūros, kurios taikomos požymių atrankos uždaviniui. Viena iš jų – MA algoritmo pritaikymas. MA algoritmas yra paimtas iš aplanko metodų grupės. MA algoritmas yra globalus paieškos metodas, kuris daro mažus atsitiktinius pakeitimus pradiniam kandidatų spėjimui. Jeigu vykdymo vertė pakeistai reikšmei yra geresnė, priimamas naujas sprendimas. Priešingai, suboptimalus sprendimas išlaiko esamą požymį ir duoda geresnį sprendimą vėlesnėse iteracijose. Tarkime, kad turime m požymių aibę $f = \{f_1, f_2, \dots, f_{m-1}, f_m\}$. Požymių atrankos uždavinyje, sprendimas yra binarinis vektorius, kuris apibūdina turimą poaibį. Jeigu turime m požymių, sprendimas turi m elementų, kur kiekvienas elementas yra 0 arba 1 (požymis patenka į poaibį arba ne). Poaibis yra sutrikdomas atsitiktinai, pakeičiant nedidelį kiekį narių.

Tvirtinama, kad MA algoritmo pritaikymas požymių atrankai atlikti yra mažiau agresyvus, nei GA. Bet kuriuo atveju, tiriamajame darbe yra analizuojami abu metodai ir realizuojami R programinėje aplinkoje. MA algoritmo atrinkti požymiai irgi yra apmokomi su AVK.

MA algoritmas imituoja metalo šildymo ir aušinimo procesą, padeda rasti stiprias ir stabilias konfigūracijas. MA randa sprendimą atsitiktinai (parenkant pradinę temperatūrą) ir tada lėtai ieško modelio patobulinimų (vyksta aušinimo procesas). Iš pradžių parenkamas pradinis požymių poaibis ir įvertinama modelio kokybė (žymimas E_1). Po to, požymių poaibis šiek tiek pasikeičia (sumažėja) ir sukuriamas kitas modelis su E_2 įvertintu tinkamumu. Jeigu naujasis modelis yra geresnis, t. y. $E_2 < E_1$, tada priimamas naujasis požymių rinkinys. Jeigu yra priešingai, tuomet naujasis požymių rinkinys vis dar gali būti priimamas, bet jau remiamasi tam tikra tikimybe p_i^{α}, kur i yra proceso iteracija. Tikimybė suformuojama taip, kad laikui bėgant mažėtų. Kai i tampa didelė, tikimybė tampa mažai tikėtina, kad būtų priimama suboptimali konfigūracija. Procesas tęsiasi su tam tikru iš anksto numatytu iteracijų skaičiumi ir naudojami geriausi kintamųjų poaibiai visose iteracijose. Algoritmas sustoja, kai pradinė temperatūra pasiekia mažą reikšmę. Vadinasi, temperatūra modelyje nuolat kinta. Idėja yra išvengti lokalaus optimumo (sprendimo, kuris šiuo

metu yra geriausias, bet bendrai toks nėra). Įtraukiant „blogus“ sprendimus, algoritmas gali tęsti paiešką kitose erdvėse, todėl yra mažiau imlus laikui [19].

Žemiau detaliai aprašomas šio algoritmo veikimas (žr. 2.1.2.1 pav.) požymių atrankos požiūriu [19]:

1. Generuojamas pradinis atsitiktinių požymių poaibis;
2. **kiekvienai** *iteracijai* $i = 1, \dots, t$ **atliekama**:
3. ▏ atsitiktinai sutrikdomas pradinių geriausių požymių rinkinys (pradinė temperatūra)
4. ▏ (neprivaloma) perapdorojami duomenys
5. ▏ suderinamas ir apmokomas modelis, panaudojant nepriklausomų kintamųjų rinkinį
6. ▏ apskaičiuojamas modelio tinkamumas (E_i)
7. ▏ **jeigu** $E_i < E_{geriausias}$ **tada**
8. ▏ ▏ priimti požymių rinkinį kaip geriausią
9. ▏ ▏ nustatyti $E_{geriausias} = E_i$
10. ▏ **kitu atveju**
11. ▏ ▏ priimant požymių rinkinį, perskaičiuoti tikimybę $p_i^{\alpha} = e^{\frac{E_{geriausias} - E_i}{T}}$, čia T yra temperatūros reikšmė ($T_0 = 1$, $T = T_0 \cdot \alpha$), besikeičianti per iteracijas, o E – vykdymo (tinkamumo) matas, α – konstanta (aušinimo parametras), lygus 0,75.
12. ▏ ▏ generuoti atsitiktinį skaičių U tarp $[0, 1]$
13. ▏ ▏ **jeigu** $p_i^{\alpha} \leq U$, **tada**
14. ▏ ▏ ▏ priimti naują požymių rinkinį kaip geriausią
15. ▏ ▏ ▏ nustatyti $E_{geriausias} = E_i$
16. ▏ ▏ **kitu atveju**
17. ▏ ▏ ▏ išlaikyti dabartinį požymių rinkinį
18. ▏ ▏ **užbaigiama**
19. ▏ **užbaigiama**
20. **užbaigiama**
21. Nustatomas požymių rinkinys, susijęs su mažiausiu E_i per visas iteracijas;
22. Modelis užbaigiamas su šia atrinktų požymių aibe.

2.1.2.1 pav. Pseudokodas požymių atrankai atlikti, panaudojant MA algoritmą

2.1.3. REKURSINIS POŽYMIŲ ATRANKOS ALGORITMAS

RPA algoritmas yra panašus į nuoseklų veiksnių išmetimo algoritmą. Algoritmas pritaiko modelį visiems atsitiktiniams dydžiams ir priskiriamas prie aplanko metodų grupės. Kiekvienas kintamasis yra rikiuojamas pagal svarbumą modeliui. Tegul m yra mažėjanti seka, t. y. požymių aibė $f = \{f_1, f_2, \dots, f_{m-1}, f_m\}$. Kiekvienoje požymių atrankos iteracijoje i yra grąžinami m_i požymiai, modelis pertvarkomas ir veikimas įvertinamas, panaudojant klasifikavimo kokybės metrikas. Apibrėžiama m_i reikšmė su geriausia vykdymo charakteristika ir svarbiausiais m_i požymiais. Norint išgauti geresnį algoritmo atlikimo kokybę, galima įvesti perskaičiavimo žingsnį,

panaudojant 10 – lypį kryžminį patikrinimą. Nors kryžminis patikrinimas duoda geresnius rezultatus, dažnai turima problemų dėl sudėtingų skaičiavimų ir ilgo algoritmo veikimo laiko. Tada vartotojams rekomenduojama turėti prieigą prie kelių procesorių. Kitas kryžminio patikrinimo trūkumas yra tai, kad kiekvienoje iteracijoje yra grąžinama daug geriausių požymių aibių. Iš kitos pusės, gaunamas labiau tikimybinis vertinimas, nei rikiuojant vieną fiksuotą duomenų rinkinį. AVK svorio vektoriaus *w* koeficientai yra naudojami požymių išrikiavimui požymių svarbumo mažėjimo tvarka. Tie požymiai, kurie turi įvertį $c_i = (w_i)^2$, yra pašalinami, o w_i reiškia atitinkamą svorių vektoriaus *w* komponentą, kuris modelyje kinta. Algoritmo pabaigoje pateikiamas geriausias atrinktų požymių poaibis [19].

RPA algoritmas yra naudojamas su AVK. RPA algoritmo privalumas tas, kad jis yra atsparus persimokymui. Detalus RPA metodo su tinkamu perskaičiavimu algoritmas pateikiamas žemiau (žr. 2.1.3.1 pav.) [19]:

1. **Kiekvienai** *perskaičiavimo iteracijai* **atliekama**:
2. duomenys padalinami į apmokymo ir testavimo imtis
3. suderinamas ir apmokomas modelis apmokymo duomenims, panaudojant visus *P* nepriklausomus kintamuosius
4. apskaičiuojamas modelio įvykdymas
5. apskaičiuojamas kintamųjų svarbumas arba reitingai
6. **kiekvienam** *poaibių skaičiui* m_i, $i = 1, \dots, m$ **atliekama**:
7. išsaugomi m_i svarbiausi kintamieji
8. (neprivaloma) perapdorojami duomenys
9. suderinamas ir apmokomas modelis apmokymo duomenims, panaudojant m_i nepriklausomus kintamuosius
10. apskaičiuojamas modelio įvykdymas, panaudojant testavimui skirtą imtį
11. (neprivaloma) perskaičiuojami reitingai kiekvienam kintamajam
12. **užbaigiama**
13. **užbaigiama**
14. Apskaičiuojamas įvykdymo profilis per m_i požymius, naudojant testavimui skirtą imtį;
15. Nustatomas tinkamas požymių skaičius;
16. Nustatomi galutiniai reitingai kiekvienam išlikusiam požymiui;
17. Pritaikomas galutinis modelis, grindžiamas m_i požymiais ir naudojant pradinę apmokymo imtį.

2.1.3.1 pav. Pseudokodas požymių atrankai atlikti, panaudojant RPA algoritmą

2.1.4. ATSITIKTINIŲ MIŠKŲ VIDINIS POŽYMIŲ ATRANKOS METODAS

Atsitiktinių miškų modelis yra priskiriamas prie netiesinio klasifikavimo metodų. Miškas yra grafas, kurio visi komponentai yra medžiai. Atsitiktinių miškų algoritmas atsitiktinai generuoja

požymių poaibius. Atsitiktiniai miškai yra vienodai pasiskirstę klasifikavimo medžiai, kur kiekvienas medis yra konstruojamas pagal pasiskirstymo taisyklę [20]. Atsitiktinių miškų (angl. *random forests*) vidinis požymių klasifikavimo metodas yra susijęs su atsitiktinių miškų regresijos algoritmu, tik vietoj regresijos medžių naudojamas klasifikavimo medis. Kiekvienas medis miške apibūdina duomenų imties klasifikavimą. Individualus sprendimų medis yra konstruojamas, panaudojant požymių atsitiktinę atranką. Kiekvienas medis priklauso nuo atsitiktinių vektorių reikšmių. Kol medžių tipai algoritme keičiasi, kiekviename dalinime atsitiktinai parinktų nepriklausomų kintamųjų skaičius *m* lieka tas pats. Klasifikavimo atsitiktiniai miškai turi tokias savybes [19]:

- neparametrinis;
- tinka spręsti dideliems uždaviniams;
- automatiškai atrenka požymius;
- klasifikavimo pabaigoje skaičiuojami klasifikavimo kokybės rodikliai ir klasifikavimo rezultatų matrica (angl. *confusion matrix*), kuri vadinama sumaišymo arba klaidų matrica.

Medžių auginimui yra naudojama *CART* algoritmo metodologija. Medžiai auginami iki maksimalaus dydžio ir nėra genėjami. Atsitiktiniai miškai yra atsparūs klaidoms ir išskirtims. Atsitiktiniai miškai neturi problemų dėl persimokymo, tinka kategoriniams ir tolydiems kintamiesiems klasifikuoti. Atsitiktinio miško tikslumas priklauso nuo atskirų klasifikatorių tikslumo ir priklausomybės tarp jų mato. Atsitiktinius miškus yra sunku interpretuoti. Atsitiktiniai medžiai yra nejautrūs parinktų požymių skaičiui [13]. Atsitiktiniai medžiai yra modeliuojami, panaudojant tokią bendrą schemą [19]:

```
1.  Pasirenkamas atsitiktinis skaičius m kuriamų modelių;
2.  kiekvienam i = 1, ..., m atliekama:
3.      generuojama imtis iš pradinių duomenų
4.      šiai imčiai apmokomas medžio modelis
5.      kiekvienam dalinimui atliekama:
6.          atsitiktinai parenkami k (< m) pradiniai nepriklausomi kintamieji
7.          parenkami geriausi požymiai tarp k pradinių kintamųjų ir duomenys išskaidomi
8.      užbaigiama
9.      naudojamas medžio modelio sustojimo kriterijus, kuris nustato, ar medis yra baigtas
        (genėjimas neatliekamas)
10. užbaigiama
```

2.1.4.1 pav. Atsitiktinių miškų pseudokodas požymių atrankai atlikti

Atsitiktiniai miškai leidžia *m* – tajam požymiui apskaičiuoti santykinį svarbumą prognozuojamoms klasėms ir santykinį svarbumą visiems požymiams, naudojant apibendrintą

Gini indeksą. Svarbumo įvertis m – tajam požymiui viename sprendimo medyje T_k yra apskaičiuojamas pagal 27 formulę (žr. (27) formulę):

$$IS_k(f_m) = \sum_{D \in T_k} \Delta Gini(D). \quad (27)$$

Bendras svarbumas *K* medžiams atsitiktiniuose miškuose apibrėžiamas tokiu būdu (žr. (28) formulę):

$$IS(f_m) = \frac{1}{K} \sum_{k=1}^{K} IS_k(f_m). \quad (28)$$

Požymiai yra išdėliojami svarbumo įverčių mažėjimo tvarka. Atsitiktiniai miškai kiekviename dalinime parenka *m* geriausių originalių požymių, remiantis modelio tinkamumo duomenims vertinimu.

Bet kokio klasifikavimo modelio tikslumo įvertinimui yra naudojamos klasifikavimo korektiškumą nusakančios metrikos, vadinamos klasifikavimo kokybės rodikliais (žr. 2.1.4.1 lent.). Metrikų skaičiavimui įvedami atitinkami pažymėjimai, priklausantys nuo teisingai arba neteisingai suklasifikuotų stebėjimų santykinio kiekio (žr. 2.1.4.2 lent.). Neteisingai klasifikuotų stebėjimų santykinis dydis ir specifiškumas yra minimizuojami matai, o jautrumas, sutapimų dydis, tikslumas bei *f* – įvertis – maksimizuojami matai [13]. Teisingai klasifikuotų stebėjimų santykinis dydis (sutapimų dydis) parodo klasifikatoriaus efektyvumą, tikslumas – teisingai klasifikuotų klasių dydį, jautrumas – klasifikatoriaus efektyvumą, leidžiantį teisingai identifikuoti klases, *f* – įvertis – ryšį tarp duomenų klasių ir algoritmu klasifikuotų stebėjimų, specifiškumas – kaip efektyviai klasifikatorius identifikuoja neteisingai klasifikuotas klases, neteisingai klasifikuotų stebėjimų dydis – klasifikuojamo modelio klaidų kiekį. Klasifikavimo kokybės metrikos pateikiamos procentiniais dydžiais.

2.1.4.1 lentelė. Klasifikavimo kokybės įvertinimo metrikų pažymėjimai (sumaišymo matrica) [13]

		Numatoma klasė	
Tikroji klasė	**TAIP**	**NE**	**Iš viso**
TAIP	TP_i	FN_i	P_i
NE	FP_i	TN_i	N_i
Iš viso	P_i'	N_i'	$P_i + N_i$

2.1.4.2 lentelė. Klasifikavimo kokybės nustatymo metrikos [13]

Metrika	Formulė
Teisingai klasifikuotų stebėjimų santykinis dydis, sutapimų santykinis dydis (angl. *accuracy*, *recognition rate*)	$\frac{1}{N}\cdot\sum_{i=1}^{N}\frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i}$
Neteisingai klasifikuotų stebėjimų santykinis dydis (angl. *error rate*, *missclassification rate*)	$\frac{1}{N}\cdot\sum_{i=1}^{N}\frac{FP_i+FN_i}{TP_i+FN_i+FP_i+TN_i}$
Jautrumas (angl. *sensitivity*, *recall*)	$\frac{1}{N}\cdot\sum_{i=1}^{N}\frac{TP_i}{TP_i+FN_i}$
Specifiškumas (angl. *specificity*)	$\frac{1}{N}\cdot\sum_{i=1}^{N}\frac{TN_i}{FP_i+TN_i}$
Tikslumas (angl. *precision*)	$\frac{1}{N}\cdot\sum_{i=1}^{N}\frac{TP_i}{TP_i+FP_i}$
f – įvertis (angl. *f – score*)	$\frac{2\times tikslumas\times jautrumas}{tikslumas+jautrumas}$

2.2. ATRAMINIŲ VEKTORIŲ KLASIFIKATORIUS

Baigiamajame projekte kiekvieno algoritmo atrinktas požymių poaibis buvo klasifikuojamas, panaudojant AVK. AVK buvo pasirinktas dėl efektyvumo didelės dimensijos erdvėje. AVK naudoja duomenų apmokymo imtį sprendimo funkcijoje, vadinamoje atraminiais vektoriais, todėl veiksmingai išnaudojama atmintis. Kai stebėjimai tvarkingai neatsiskiria (sprendžiama iš grafinio vaizdo), tada naudojamos branduolio funkcijos (tiesinė (be branduolio) (angl. *linear*), polinominė (angl. *polinomial*), radialinė (angl. *radial basis function*)). AVK siekia optimizuoti hiperplokštumos ribos plotį [21].

AVK naudoja griežtos paraštės (angl. *hard margin*), lengvos paraštės (angl. *soft margin*) arba branduolio metodus. Tegul M m – dimensijos apmokymo įvesties kintamųjų x_i $(i = 1, \dots, M)$ priklauso klasėms y_i $(i = 1, \dots, M)$ (šiuo atveju, turimas 9 klasių klasifikavimo uždavinys). Sprendimo funkcijos yra (žr. (29) formulę):

$$D_i(x) = w_i^T \phi(x) + b_i, \tag{29}$$

kur w_i yra m – dimensijos vektorius, $\phi(x)$ – funkcija, kuri braižo x į m – dimensijos požymių erdvę, b_i – poslinkio koeficientas.

Jei hiperplokštuma $D_i(x) = 0$, randama optimali skiriančioji hiperplokštuma. Jeigu klasifikavimo uždavinys yra atskiriamas, apmokymo duomenys priklauso klasei i, tenkinančiai $D_i(x) \geq 1$. Tie duomenys, kurie priklauso kitai klasei, tenkina $D_i(x) \leq -1$. Jeigu duomenys yra neatskiriami, tuomet $|D_i(x)| = 1$ ir klasei i priklausantys atraminiai vektoriai tenkina $D_i(x) \leq 1$.

Priešingu atveju, $D_i(x) \geq -1$. Paraštė maksimizuojama, kai Euklido norma, t. y. $\frac{1}{2}\|w\|^2$ artėja į minimumą.

AVK esmė yra rasti koeficientus *w* (žr. (30) formulę) ir *b* (žr. (31) formulę):

$$w = \sum_{i=1}^{M} \alpha_i y_i x_i \text{ ir} \tag{30}$$

$$b = y_i - w^T x_i, \tag{31}$$

čia α_i yra neneigiami Lagranžo daugikliai [21].

C ir γ yra hiperparametrai, skirti optimizuoti klasifikavimo uždavinį. *C* yra minkštos paraštės reguliarizavimo parametras. Minkštos paraštės minimizavimo uždavinys apibrėžiamas 32 formule (žr. (32) formulę):

$$Q(w, b, \xi) = \frac{1}{2}\|w\|^2 + \frac{C}{p}\sum_{i=1}^{M} \xi_i^p, \tag{32}$$

kur *p* – bauda (angl. *penalty*) (*L1*, jei $p = 1$ ir *L2*, jei $p = 2$), o ξ_i – neneigiami laisvi kintamieji.

Darbe naudojamas radialinis branduolys (žr. (33) formulę), kuris naudoja Euklido atstumą.. Parametras γ – laisvas Gauso (angl. *Gaussian*) radialinio branduolio parametras.

$$K(x, x') = e^{-\gamma\|x - x'\|^2}, \tag{33}$$

kur γ – teigiamas parametras, kontroliuojantis spindulį. Perrašome (žr. (34) formulę):

$$K(x, x') = e^{-\gamma\|x\|^2} e^{-\gamma\|x'\|^2} e^{2\gamma x^T x'}, \tag{34}$$

kur $e^{2\gamma x^T x'} = 1 + 2\gamma x^T x' + 2\gamma^2 (x^T x')^2 + \frac{(2\gamma)^2}{3!}(x^T x')^3 + \cdots$ ir $e^{2\gamma x^T x'}$ – begalinė polinomo suma.

Čia sprendimo funkcija yra apskaičiuojama, naudojant 35 formulę (žr. (35) formulę):

$$D(x) = \sum_{i \in S} \alpha_i y_i \, e^{-\gamma\|x - x'\|^2} + b. \tag{35}$$

Branduolio $K(x, x')$ bei hiperparametrų *C* ir γ parinkimas leidžia išvengti persimokymo problemos [21].

2.3. PROGRAMINIAI MODELIAI

Požymių atrankos ir klasifikavimo metodai buvo realizuoti Python ir R programinėse aplinkose. Python programavimo kalba buvo paleista per virtualią Linux operacinę sistemą, panaudojant Apache Spark galimybes: greitį, paskirstytus skaičiavimus ir atmintį. Konkrečios realizacijos yra pateikiamos baigiamojo projekto prieduose (žr. 5, 6, 7, 8 priedus), o programinės realizacijos aprašomos 2.3.1 skyrelyje (žr. 2.3.1 skyrelį).

2.3.1. PROGRAMINIŲ MODELIŲ STRUKTŪRA

Pradinių požymių dalinimas į apmokymo ir testavimo imtis R programiniame pakete buvo atliktas, taikant `createDataPartition()` funkciją.

GA požymių atrankai atlikti buvo realizuotas R programinėje aplinkoje, panaudojant tokias funkcijas:

- `gafsControl()` – kontroliuoja skaičiavimo niuansus `gafs()` funkcijai, t. y. perskaičiavimo metodą, perskaičiavimo iteracijų skaičių, lygiagrečios posistemės parinkimą, modeliavimo funkcijos nurodymus;
- `gafs()` – nėra labai specifinė procedūra. Ji leidžia parinkti požymių imtį, klasės kintamąjį, algoritmo iteracijų skaičių, populiacijos dydį, kryžminimo ir mutacijos tikimybes. Visi veiksmai yra kontrolės funkcijoje.

MA algoritmas požymių atrankai atlikti irgi buvo realizuotas R programinėje aplinkoje. Naudotos funkcijos:

- `safsControl()` – kontroliuoja skaičiavimo niuansus `safs()` funkcijai, t. y. perskaičiavimo metodą, perskaičiavimo iteracijų skaičių, lygiagrečios posistemės parinkimą, modeliavimo funkcijos nurodymus;
- `safs()` – prižiūrima mokytojo požymių atrankos procedūra, naudojant MA modeliavimą. Ji leidžia atsitiktinai nurodyti požymių imtį, klasės kintamąjį, iteracijų skaičių.

Modeliuojant AVK apmokymą, R programinė kalba suteikia kelias patogias funkcijas:

- `svm()` – funkcija, naudojama apmokyti AVK, parinkti branduolio tipą, tikrines hiperparametrų reikšmes;
- `predict()` – funkcija, naudojama klasifikavimo klasėms prognozuoti;
- `tune()` – bendrinė funkcija, leidžianti parinkti statistinių metodų hiperparametrus ir naudojanti paiešką tinkleliuose su visais į sistemą pateiktais parametrų diapazonais (intervalais);
- `confusionMatrix()` – funkcija, kuri apskaičiuoja tikrų ir prognozuotų klasių porinius stebėjimus bei su klasifikavimo matrica susijusias statistikas.

Taikant atsitiktinių miškų vidinį požymių klasifikavimo algoritmą, pasitelkiamos tokios R programinės kalbos funkcijos:

- `train()` – funkcija, nustatanti modelio „tiuningo“ parametrus, matematinį klasifikavimo metodą, apibrėžianti klasės kintamąjį, pradinę požymių imtį, kryžminio patikrinimo blokus;
- `randomForest()` – funkcija, realizuojanti atsitiktinių miškų klasifikavimo metodą, duodanti išaugintų medžių skaičių, atrinktų požymių skaičių;
- `varImpPlot()` – funkcija, grąžinanti taškinį kintamųjų svarbumą, apskaičiuotą pagal atsitiktinius miškus.

Python programavimo kalboje buvo sukurta funkcija, leidžianti nusakyti klasifikavimo rezultatus per šilumos grafikų panaudojimą.

Šilumos grafikams nusakyti ir apibrėžti, R programinė kalba pateikia:

- `levelplot()` – funkciją, leidžiančią pateikti gautus rezultatus, panaudojant šilumos grafikus;
- `layer()` – funkcija, kurios pagalba koreguojami šilumos grafikai, įvedant papildomas grafines galimybes, pavyzdžiui, skaitinį duomenų formatą ant šilumos grafikų. Tai leidžia geriau interpretuoti šilumos grafikus, nes vienu metu matomas ne tik grafinis, bet ir skaitinis vaizdas.

Programos vykdymo laikui apskaičiuoti, imama R programinio paketo `proc.time()` funkcija. Python programiniame pakete skaičiavimų įvykdymo laiko įvertinimui naudojama `time.time()` funkcija.

Klasifikavimo tikslumui nusakyti, buvo skaičiuojamos bendros modelių metrikos: tikslumas, *f* – įvertis, jautrumas, teisingai klasifikuotų stebėjimų dydis (angl. *accuracy*). R programinis paketas leidžia naudotis tokiomis funkcijomis:

- `nrow()` – grąžina klaidų matricos stulpelių skaičių;
- `diag()` – konstruoja diagonalią matricą;
- `data.frame()` – leidžia sukonstruoti kelių tarpusavyje susijusių statistikų sąrašą ir yra vertinama kaip pagrindinė duomenų struktūra;
- `apply()` – taiko, šiuo atveju sumos, funkciją kiekvienai eilutei ir stulpeliui;
- `sum()` – apskaičiuoja sumą;
- `mean()` – apskaičiuoja vidurkį;
- `as.matrix()` – argumentus suveda į matricą.

RPA algoritmas su AVK buvo aprašytas, panaudojant Python programavimo kalbą. Parinktos funkcijos:

- `cross_validation.train_test_split()` – atsitiktinai padalina pradinius duomenis į apmokymo ir testavimo imtis.
- `fit()` – funkcija, kuri leidžia kuriamą metodą pritaikyti duomenų imčiai;
- `SVC()` - funkcija, naudojama apmokyti AVK, parinkti branduolio tipą, tikrines hiperparametrų reikšmes;
- `RFECV()` – rekursinei požymių atrankai skirta funkcija, kuri įtraukia kryžminį patikrinimą;
- `grid_search.GridSearchCV()` – funkcija, leidžianti išsamiai ieškoti modelio hiperparametrus;
- `predict()` – funkcija, kuri leidžia prognozuoti klasės kintamąjį;
- `confusion_matrix()` – funkcija, kuri išveda klasifikavimo rezultatų matricą;
- `accuracy_score()` – funkcija, kuri apskaičiuoja teisingai klasifikuotų stebėjimų dydį;
- `classification_report()` – funkcija, apskaičiuojanti bendras klasifikavimo kokybės metrikas.

Vidinės programinių paketų funkcijos buvo koreguotos pagal turimus duomenis, pritaikyti hiperparametrai, ieškoti optimalūs naudojamų metodų sprendimai, programiškai realizuotos klasifikavimo tikslumą nusakančios metrikos, atliktas požymių atrankos metodų ir AVK apjungimas, vizualizuoti sprendimai grafiškai.

2.3.2. R IR PYTHON SĄSAJA

Baigiamajame darbe naudojamos dvi programinės kalbos: R ir Python. Python kalba duomenų apdorojimui tinka labiau, nei R, bet ji neturi statistikos bibliotekų. Rekomenduojama vienu metu naudoti abi kalbas. Siekiant kalbas suvienodinti, buvo pasiūlyta panaudoti „rpy2“ Python biblioteką, leidžiančią R programinius kodus paleisti per Python programinės kalbos langą (žr. 2.3.2.1 pav.). Kiekviena R kalbos funkcija Python programinėje įrangoje yra užrašoma kaip r(*funkcija*).

```
import rpy2.robjects as rkalba
r = rkalba.r
s = r("read.csv('Pradinis.csv')")
```

2.3.2.1 pav. R kalba Python aplinkoje

3. POŽYMIŲ ATRANKOS METODŲ PALYGINIMAS

Šioje pristatomi tirti duomenys, palyginami eksperimentų rezultatai, taikant įvarius požymių atrankos algoritmus klasifikavimo uždaviniams spręsti, pateiktos klasifikavimo metodų kokybės metrikos ir rekomendacijos tolimesniems tyrimams.

3.1. DUOMENYS IR ATLIKTI EKSPERIMENTAI

Darbe tirti vienos iš didžiausių e – komercijos įmonės „Otto Group“, paimti iš „Kaggle“ atviros prieigos archyvų. „Kaggle“ yra didžiausia duomenų analizės konkursų, konsultacijų ir matematinio modeliavimo platforma. „Otto Group“ turi dukterines įmones daugiau nei 20 šalių, įskaitant Jungtines Amerikos Valstijas, Vokietiją, Prancūziją. Įmonė kiekvieną dieną parduoda milijonus produktų visame pasaulyje, įskaitant ir savo produkcijos liniją. Įmonei yra svarbi nuosekli produktų eksploatacinių savybių analizė. Dėl skirtingos globalinės infrastruktūros, nemažai produktų yra suklasifikuoti kitaip. Produktų kokybės analizė priklauso nuo sugebėjimo tiksliai klasifikuoti panašius produktus. Kuo geresnis klasifikavimas, tuo geresnės įžvalgos apie gaminių asortimento gerinimą.

Tiriami duomenys turi 93 skirtingus požymius (kintamuosius), klasifikavimo kintamąjį (9 kategorijos (visų produktų klasės)) ir virš 10 tūkstančių stebėjimų. Tikslas yra sukurti tokį klasifikavimo modelį, kuris lengvai galėtų priskirti produktus klasėms. Prieinamus ir atviro kodo šių duomenų tyrimus galima rasti „Kaggle“ (https://www.kaggle.com) internetiniame puslapyje. Bendra produktų klasifikavimo į klases idėja yra pateikta 3.1.1 paveiksle (žr. 3.1.1 pav.).

3.1.1 pav. Produktų klasifikavimas į „Otto Group“ klases (pavydžiui, elektronika, mada ar kita)

Kiekviena eilutė atitinka vieną produktą. Požymiai nurodo įvairių įvykių skaičių. Jie yra išmatuoti santykių matavimo skalėje.

Pradiniai duomenys buvo padalinti į apmokymo (70 % pradinių duomenų) ir testavimo (30 % pradinių duomenų) imtis. Klasifikuojamų duomenų požymių atrankai buvo naudoti tokie metodai: GA ir MA optimizavimo algoritmų pritaikymas požymių atrankai atlikti, RPA algoritmas ir atsitiktinių miškų vidinis požymių atrankos algoritmas, pradžioje įtraukiantis požymių atrankos etapą. Atrinkti požymiai buvo klasifikuojami AVK arba atsitiktinių miškų metodu. Siekiant išvengti persimokymo problemos, AVK modeliui buvo ieškoti hiperparametrai. Norint įvertinti klasifikavimo modelio kokybę, buvo skaičiuotos kelios metrikos (rodikliai): tikslumas, f – įvertis, jautrumas, teisingai klasifikuotų stebėjimų dydis. Tyrimo eigoje buvo sukurtos požymių atrankos, optimizavimo algoritmų ir klasifikavimo metodų kombinacijos. Į nagrinėjamą uždavinį buvo žiūrėta iš dviejų pusių: kaip gauti kuo mažiau originalių požymių ir kartu ženkliai nesumažinti klasifikavimo tikslumo. Vadinasi, vienu metu minimizuojamas požymių skaičius ir maksimizuojami klasifikavimo kokybės rodikliai. Atskirai matuojamas kiekvieno metodo įvykdymo laikas. Tyrimo pabaigoje buvo sukurta patogi R ir Python sąsaja, leidžianti apjungti abi programavimo kalbas, iškviečiant komandinę operacinės sistemos eilutę ir R programinį kodą skaitant per Python programavimo aplinką. Tyrimo rezultatai pateikiami kituose skyreliuose (žr. 3.2 ir 3.3 skyrelius).

3.2. KLASIFIKAVIMO KOKYBĖS ĮVERTINIMO METRIKOS

Tyrimo metu buvo realizuoti 4 modeliai, atlikta 16 eksperimentų: GA pritaikymas požymių atrankai atlikti ir apjungimas su AVK metodu, MA algoritmo pritaikymas požymių atrankai atlikti ir apjungimas su AVK metodu, RPA algoritmo ir AVK junginys, atsitiktinių miškų vidinis požymių atrankos ir klasifikavimo metodas. Siekiant įvertinti sukurtų modelių veikimą, atsižvelgiant į kiekvienos klasės rinkinį, apskaičiuojamos bendros klasifikavimo kokybės metrikos apmokymo ir testavimo duomenims (žr. 3.2.1 ir 3.2.2 lent.). Šie rodikliai yra naudingi, kai klasės nėra tolygiai pasiskirstę (matysime 3.3 skyrelyje). Tokiais atvejais, vien tik tikslumo apskaičiavimas gali būti klaidinantis sprendimas, nes visą laiką prognozuojama viena dominuojanti klasė, o pasiekiamas gana aukštas bendras teisingai klasifikuotų stebėjimų santykinis dydis, bet mažas jautrumas arba tikslumas. Reikšmingų požymių atrankos etape randamas fiksuotas požymių poaibio skaičius (žr. 3.2.3 lent.). Kiekvieno algoritmo, kuris jungia ir požymių atranką, ir klasifikavimo uždavinį, iteracijos pabaigoje fiksuojamas algoritmo vykdymui reikalingas laikas (žr. 3.2.3 lent.). Atliekant minėtų metodų realizaciją, pastebėta, kad analizuojami metodai yra imlūs laikui. Sumažinant požymių skaičių, atliekami skaičiavimai užtrunka trumpiau (žr. 3.2.4 lent.), todėl tokia analizė yra tikslinga. Hiperparametrai randami, nurodžius jų galimus kitimo intervalus.

3.2.1 lentelė. Klasifikavimo kokybės metrikos, panaudojant skirtingus metodus (apmokymo imtis)

Metodas	Iteracija	Teisingai klasifikuotų stebėjimų dydis	*f* – įvertis	Tikslumas	Jautrumas
GA + AVK	*i* = 1	94,07 %	93,58 %	91,67 %	96,25 %
	i = 2	93,34 %	92,34 %	89,95 %	95,91 %
	i = 3	96,69 %	96,64 %	95,51 %	98,02 %
	i = 4	93,99 %	93,74 %	91,98 %	96,17 %
MA + AVK	*i* = 1	59,47 %	42,58 %	40,97 %	76,94 %
	i = 2	69,96 %	57,28 %	52,63 %	77,67 %
	i = 3	56,17 %	–	36,12 %	–
	i = 4	62,13 %	48,91 %	45,69 %	77,12 %
RPA algoritmas + AVK	*i* = 1	87,68 %	88 %	89 %	88 %
	i = 2	88,90 %	89 %	90 %	89 %
	i = 3	90,32 %	90 %	91 %	90 %
	i = 4	83,44 %	84 %	85 %	83 %
Atsitiktiniai miškai	*i* = 1	100 %	100 %	100 %	100 %
	i = 2	100 %	100 %	100 %	100 %
	i = 3	100 %	100 %	100 %	100 %
	i = 4	100 %	100 %	100 %	100 %

3.2.2 lentelė. Klasifikavimo kokybės metrikos, panaudojant skirtingus metodus (testavimo imtis)

Metodas	Iteracija	Teisingai klasifikuotų stebėjimų dydis	*f* – įvertis	Tikslumas	Jautrumas
GA + AVK	*i* = 1	65,55 %	55,89 %	52,23 %	72,54 %
	i = 2	65,23 %	52,67 %	50,45 %	67,21 %
	i = 3	65,80 %	57,03 %	53,20 %	69,57 %
	i = 4	65,64 %	56,53 %	52,71 %	72,52 %
MA + AVK	*i* = 1	52,51 %	–	32,31 %	–
	i = 2	58,38 %	40,04 %	39,05 %	52,06 %
	i = 3	51,17 %	–	29,61 %	–
	i = 4	53,71 %	34,31 %	33,79 %	59,03 %
RPA algoritmas + AVK	*i* = 1	78,85 %	79 %	80 %	79 %
	i = 2	81,03 %	81 %	81 %	81 %
	i = 3	81,70 %	82 %	82 %	82 %
	i = 4	77,26 %	77 %	78 %	77 %
Atsitiktiniai miškai	*i* = 1	76,14 %	66,87 %	64,72 %	74,14 %
	i = 2	75,70 %	67,58 %	65,11 %	75,80 %
	i = 3	76,43 %	66,93 %	64,62 %	76,83 %
	i = 4	76,97 %	68,91 %	66,43 %	76,70 %

3.2.3 lentelė. Kiti klasifikavimo rezultatai, panaudojant skirtingus požymių atrankos metodus

Metodas	**Iteracija**	**Laikas**	**Atrinktų požymių skaičius**	**Hiperparametrai**	
				γ	C
GA + AVK	$i = 1$	264 min	84	0,1	2
	$i = 2$	257 min	81	0,1	2
	$i = 3$	268 min	84	0,1	5
	$i = 4$	271 min	85	0,1	2
MA + AVK	$i = 1$	169 min	20	0,1	5
	$i = 2$	174 min	22	0,1	5
	$i = 3$	180 min	22	0,1	2
	$i = 4$	172 min	24	0,1	2
RPA algoritmas + AVK	$i = 1$	477 min	65	0,01	10
	$i = 2$	395 min	81	0,01	5
	$i = 3$	317 min	75	0,01	10
	$i = 4$	357 min	63	0,01	5
Atsitiktiniai miškai	$i = 1$	664 min	47	–	–
	$i = 2$	693 min	47	–	–
	$i = 3$	870 min	47	–	–
	$i = 4$	872 min	47	–	–

Klasifikavimo, įtraukiant požymių atranką, rezultatai buvo palyginti su rezultatais, kai AVK klasifikuoja visus turimus duomenis, t. y., nevykdoma požymių atranka (žr. 3.2.4 ir 3.2.5 lenteles). Gaunamas virš 90 % tikslumas abiem imtims. Metodas suklasifikuoja rezultatus per 450 minučių.

3.2.4 lentelė. Klasifikavimo kokybės metrikos, nenaudojant požymių atrankos (apmokymo imtis)

Metodas	**Požymių skaičius**	**Teisingai klasifikuotų stebėjimų dydis**	***F* – įvertis**	**Tikslumas**	**Jautrumas**
AVK	93	92,60 %	93 %	93 %	93 %

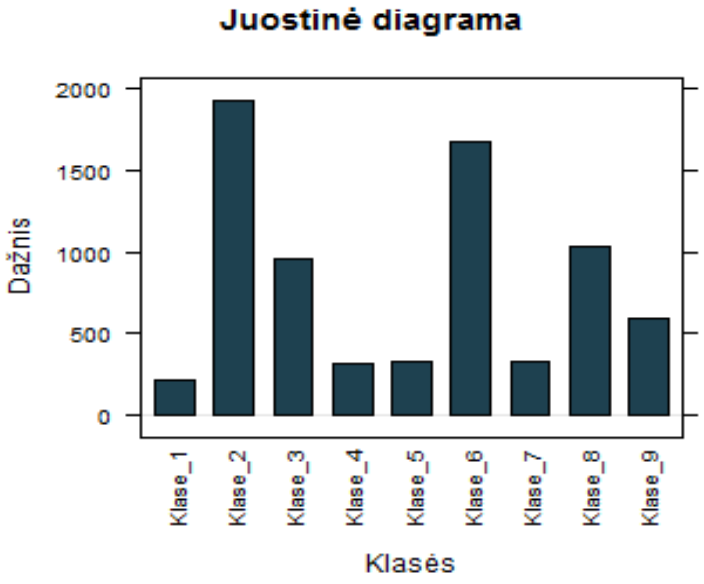
3.2.5 lentelė. Klasifikavimo kokybės metrikos, nenaudojant požymių atrankos (testavimo imtis)

Algoritmas	**Požymių skaičius**	**Teisingai klasifikuotų stebėjimų dydis**	***F* – įvertis**	**Tikslumas**	**Jautrumas**
AVK	93	74,36 %	74 %	75 %	74 %

Mažiausiai laiko požymių atrankai atlikti ir klasifikuoti reikalauja MA ir AVK kombinacija. MA algoritmas duoda mažus požymių poaibius, lyginant su kitais realizuotais metodais. MA algoritmas yra mažiau imlus laikui. Ilgiausiai kompiuterinių skaičiavimo resursų reikalauja atsitiktinių miškų modeliavimas. Kalbant apie klasifikavimo kokybės metrikas, prasčiausiai veikia MA ir AVK apjungimo algoritmas. Čia tiek apmokymo, tiek testavimo imtims, tikslumas gaunamas ~ 50 %. Nors GA pašalina nedaug požymių, tačiau atrinktus požymius klasifikuoja 90 % tikslumu (apmokymo imtis) ir 65 % tikslumu (testavimo imtis). RPA algoritmo ir AVK metodų junginys duoda gerus rezultatus testavimo imčiai, t. y. pasiekiamas ~ 80 % tikslumas, todėl tokia metodų variacija parenkama kaip geriausia testavimo imčiai. Apmokymo imčiai rekomenduojama naudoti atsitiktinių miškų vidinį požymių atrankos ir klasifikavimo metodą, kuriam nereikia atskiro požymių atrankos sprendimo.

3.3. POŽYMIŲ ATRANKOS METODŲ PALYGINIMO REZULTATAI

Taikant klasifikavimo modelius, apmokymo imtis sudaro 70 % pradinių duomenų, o 30 % duomenų paliekama testavimo imčiai. Atlikus pradinių duomenų tiriamąją analizę nustatyta (žr. 3.3.1 pav.), kad klasės nėra tolygiai pasiskirstę, 2 ir 6 klasės yra dominuojančios. Į kiekvieną klasę patenka daugiau, nei 10 procentų produktų. Požymiai atrenkami, naudojant skirtingus požymių atrankos algoritmus. Atrinkti poaibiai klasifikuojami, panaudojant AVK. Klasifikavimo rezultatų vizualizavimui naudojami šilumos grafikai (angl. *heat map*). Jeigu matrica nėra diagonali, tai rodo, kad yra klaidingai klasifikuotų duomenų. Šilumos grafikuose skaitinių reikšmių dydį apibūdina spalvos intensyvumas.



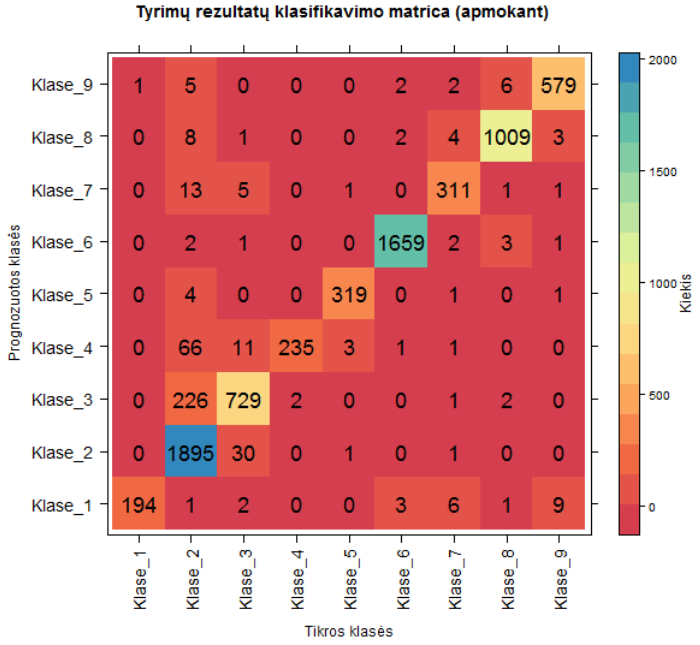
3.3.1 pav. Pradinių duomenų juostinė diagrama

GA taikymo požymių atrankai rezultatai

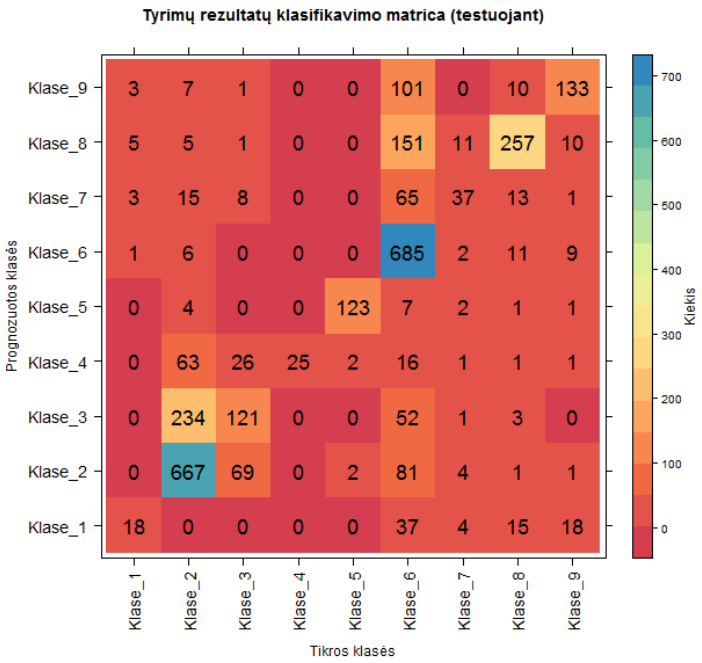
Tyrimas kartojamas 4 kartus. GA veikia su fiksuotu iteracijų skaičiumi. Chromosomos ilgis yra lygus požymių skaičiui, o kiekvienas bitas, pvz. 0, reiškia, kad požymis neatrenkamas, o 1, kad atrenkamas. Siekiant panaudoti lygiagrečius skaičiavimus, eksperimentai buvo atliekami kompiuteriu su 4 branduoliais. Atliekamas 10 – lypis kryžminio patikrinimo metodas, atsitiktinai parenkami populiacijų bei individų dydžiai, įvertinamas chromosomų efektyvumas kiekvienoje kartoje. Atliekant eksperimentus, kryžminimo tikimybė buvo 0,8, o mutacijos tikimybė buvo 0,1. GA atrinko požymius: 84 (pirmas bandymas), 81 (antras bandymas), 84 (trečias bandymas) ir 85 (ketvirtas bandymas). Pastebėta, kad atrinktų reikšmingų požymių skaičius beveik nesumažėja. Nustatyta, kad geriausios reikšmės pasiekiamos 3, 6 ir 8 iteracijose.

Atrinktiems požymių poaibiams buvo pritaikytas AVK metodas. Bandant išvengti persimokymo problemos, ne tik buvo taikytas kryžminio patikrinimo metodas, bet ir ieškoti modelio hiperparametrai C ir γ (galima rasti 3.2 skyrelyje). Norint, kad stebėjimai atsiskirtų tvarkingai, įvestos radialinės branduolio funkcijos.

3.3.2 paveiksle (žr. 3.3.2 pav.) yra pateikta klasifikavimo rezultatų matrica apmokymo imčiai, taikant genetinį požymių atrankos algoritmą. 3.3.3 paveiksle (žr. 3.3.3 pav.) yra vaizduojama klasifikavimo rezultatų matrica testavimo imčiai, taikant genetinį požymių atrankos algoritmą. Kitų iteracijų rezultatai pateikiami 1 priede (žr. 1 priedą). Iš 3.3.2 paveikslo (žr. 3.3.2 pav.) matome, kad apmokymo modeliui klasifikavimo rezultatai atskiria beveik visas klases. Validavimo atveju (žr. 3.3.3 pav.), klasifikavimo modelis daro klaidas 2 ir 6 klasėse. Klasifikavimo modelio statistikos buvo pateiktos 3.2. skyrelyje (žr. 3.2. skyrelį).



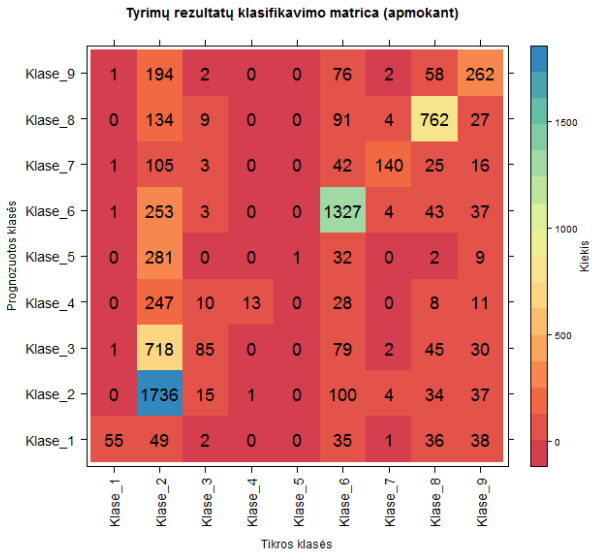
3.3.2 pav. GA ir AVK metodo klasifikavimo rezultatų matrica apmokymo imčiai



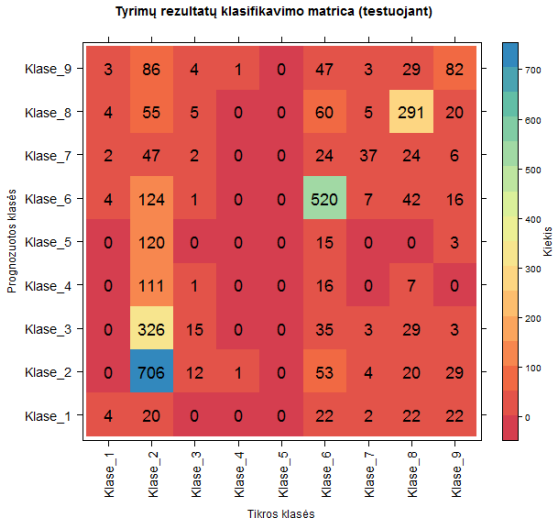
3.3.3 pav. GA ir AVK metodo klasifikavimo rezultatų matrica testavimo imčiai

MA taikymo požymių atrankai rezultatai

Toliau buvo atliekamas MA algoritmo pritaikymas požymių atrankai modeliuoti. Tyrimas kartojamas 4 kartus. Parenkamas pradinis paieškos erdvės taškas (viena galima požymių kombinacija) su aukšta „temperatūra“. Kiekvienoje iteracijoje poaibyje atliekami atsitiktiniai pakeitimai. Jeigu pakeitimai duoda geresnį požymių poaibį, jis priimamas. Priešingu atveju, taip pat priimamas, bet su tikimybe, kuri priklauso nuo „temperatūros“. Algoritmas sustoja, kai nebelieka patobulinimų. Pastebėta, kad MA algoritmas leidžia gauti mažus originalių požymių poaibius, kurių dydis gaunamas toks: 20 (pirmas bandymas), 22 (antras bandymas), 22 (trečias bandymas), 24 (ketvirtas bandymas). Geriausios reikšmės gaunamos 7 arba 8 iteracijose. Analogiškai, kaip ir aukščiau aprašytoje metodų aplikacijoje, atliekamas 10 – lypis kryžminis patikrinimas, klasifikavimo modelio hiperparametrų paieška, kurios rezultatai yra pateikti 3.2 skyrelyje (žr. 3.2 skyrelį), įvedama radialinio branduolio funkcija. MA, kuris taikomas požymių atrankai, ir AVK algoritmas pirmo eksperimento metu pateikia žemiau vaizduojamas sumaišymo matricas duomenų apmokymo ir testavimo imtims, į realizaciją įtraukiant šilumos grafikus (žr. 3.3.4 ir 3.3.5 pav.). Iš grafikų matyti, kad algoritmas neatskiria 5 klasės ir beveik visus stebėjimus klasifikuoja 2, 6 ir 8 klasėms. Kitų iteracijų rezultatai pateikiami 2 priede (žr. 2 priedą). Klasifikavimo modelio statistikos buvo skaičiuotos 3.2. skyrelyje (žr. 3.2. skyrelį).



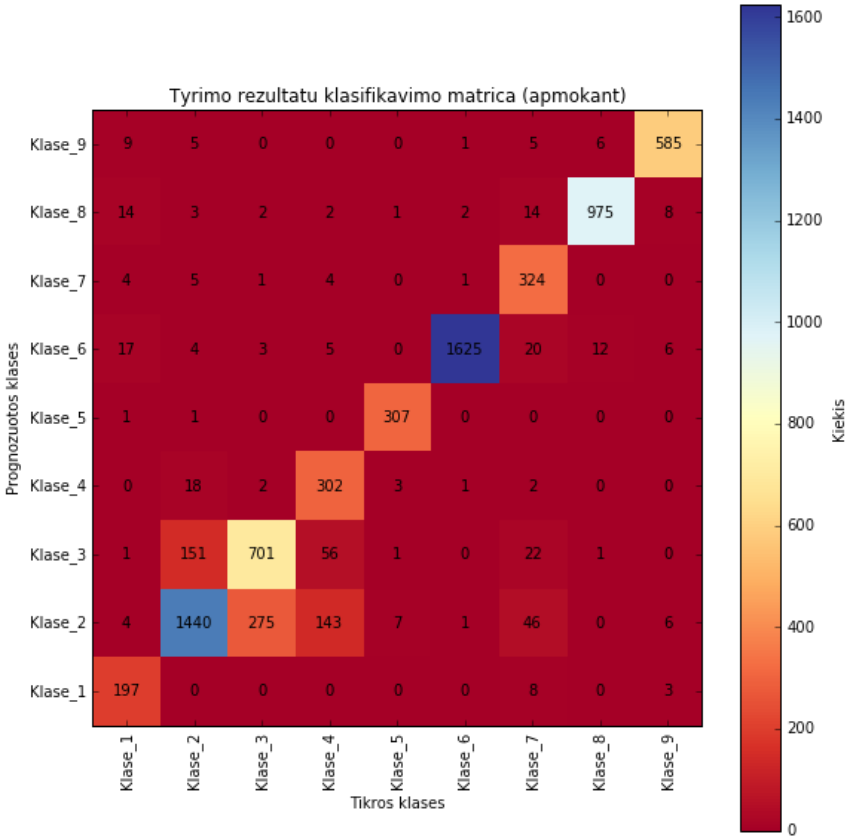
3.3.4 pav. MA algoritmo ir AVK metodo klasifikavimo rezultatų matrica apmokymo imčiai



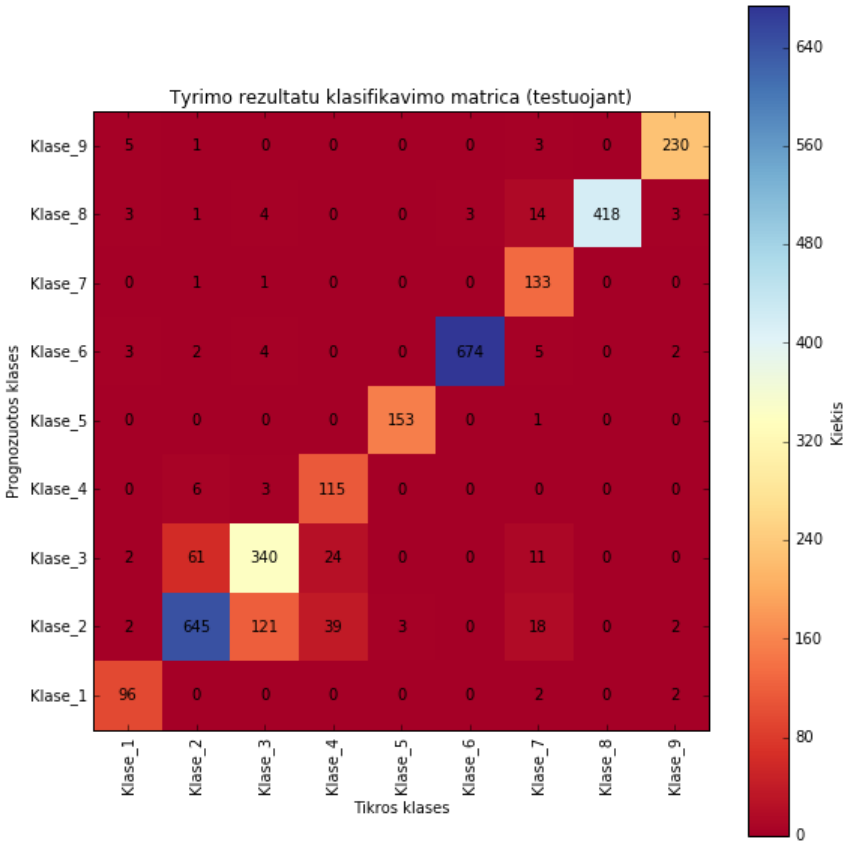
3.3.5 pav. MA algoritmo ir AVK metodo klasifikavimo rezultatų matrica testavimo imčiai

RPA taikymo požymių atrankai rezultatai

Po aukščiau aprašytų metodikų, buvo modeliuojama RPA algoritmo ir AVK sąsaja. RPA algoritmo panaudojimas leidžia suranguoti požymius, juos pašalinti iš sekos, pradedant nuo mažiausiai svarbaus kintamojo. RPA algoritmas iš pradžių įtrauka visus kintamuosius ir vienu metu pašalina po vieną požymį. Pirmame bandyme gaunami 65 požymiai. Jie klasifikuojami, ieškomi hiperparametrai (žr. 3.2 skyrelį), įvedamas 10 – lypis kryžminio patikrinimo metodas. Gaunama klasifikavimo rezultatų matrica dviem imtims (žr. 3.3.6 ir 3.3.7 pav.). Matoma, kad testavimo imčiai klasifikavimo metodas neteisingai atskiria 2 ir 3 klases. Kitų iteracijų rezultatai pateikiami 3 priede (žr. 3 priedą). Klasifikavimo modelio kokybės metrikos (statistikos) buvo apskaičiuotos 3.2. skyrelyje (žr. 3.2. skyrelį).



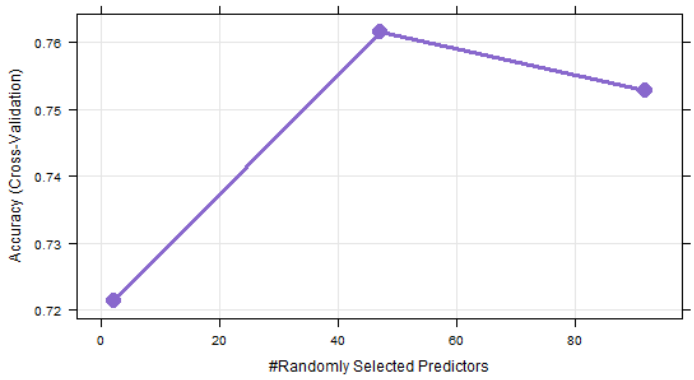
3.3.6 pav. RPA algoritmo ir AVK metodo klasifikavimo rezultatų matrica apmokymo imčiai



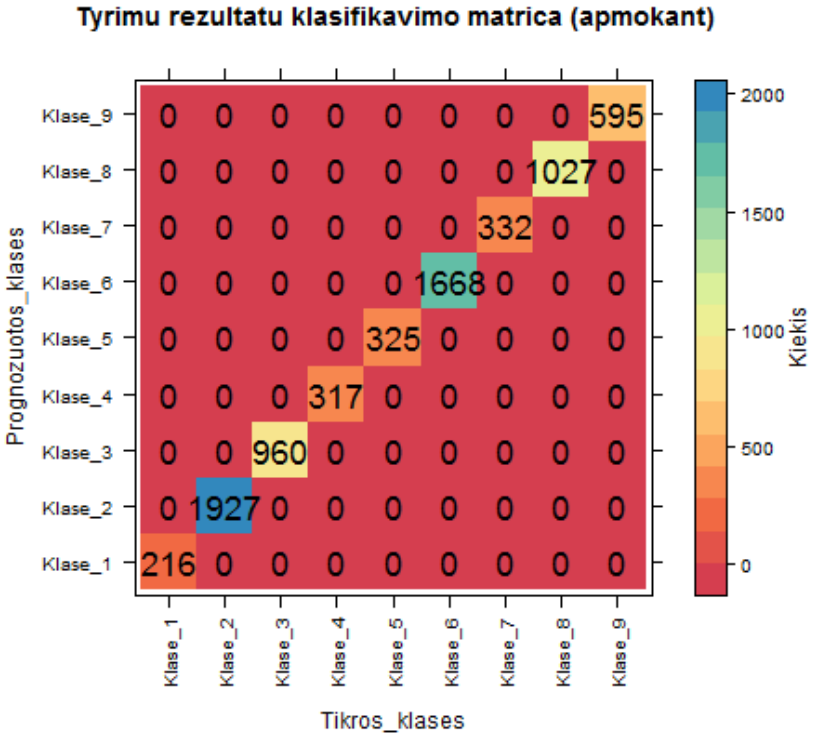
3.3.7 pav. RPA algoritmo ir AVK metodo klasifikavimo rezultatų matrica testavimo imčiai

Atsitiktinių miškų vidinio požymių atrankos ir klasifikavimo taikymo rezultatai

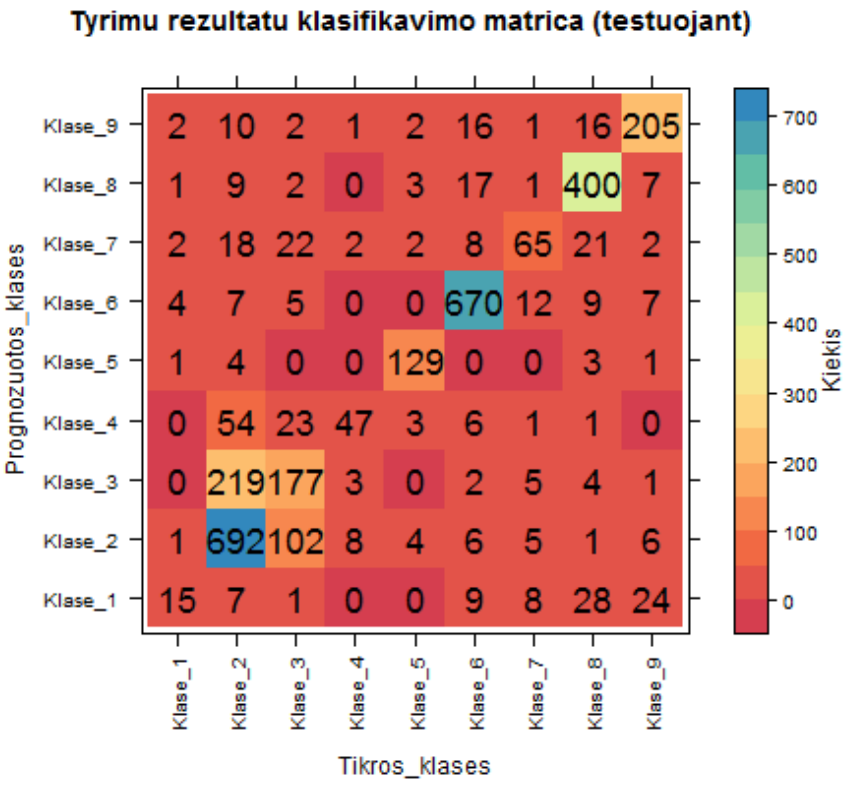
Gauti požymių atrankos ir klasifikavimo metodų apjungimo rezultatai lyginami su atsitiktinių miškų vidiniais požymių atrankos klasifikatoriais, kurie požymių atrankos etapo nesprendžia kaip atskiro uždavinio. Randami 47 svarbūs kintamieji – naujas požymių poaibis, kurį sudaro didžiausią tikslumą klasėms duodantys originalūs požymiai (žr. 3.3.8 pav.). Klasifikavimo metodas parodo itin gerus rezultatus apmokymo imčiai. Metodą testuojant, teisingai klasifikuojami 76 % stebėjimų. Žemiau vaizduojamos klasifikavimo rezultatų matricos apmokymo ir testavimo duomenims (žr. 3.3.9 ir 3.3.10 pav.). Kiti bandymų rezultatai yra pavaizduoti 4 priede (žr. 4 priedą). Klasifikavimo modelio statistikas galima rasti 3.2. skyrelyje (žr. 3.2. skyrelį).



3.3.8 pav. Tikslumas pagal atrinktus požymius

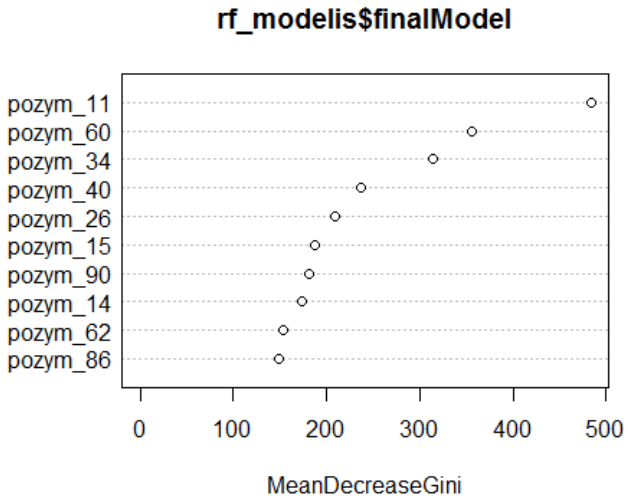


3.3.9 pav. Klasifikavimo, panaudojant atsitiktinius miškus, rezultatų matrica apmokymo imčiai

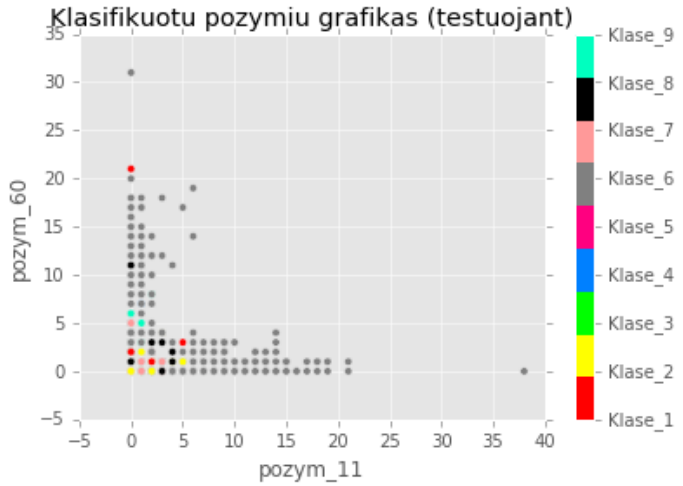


3.3.10 pav. Klasifikavimo, panaudojant atsitiktinius miškus, rezultatų matrica testavimo imčiai

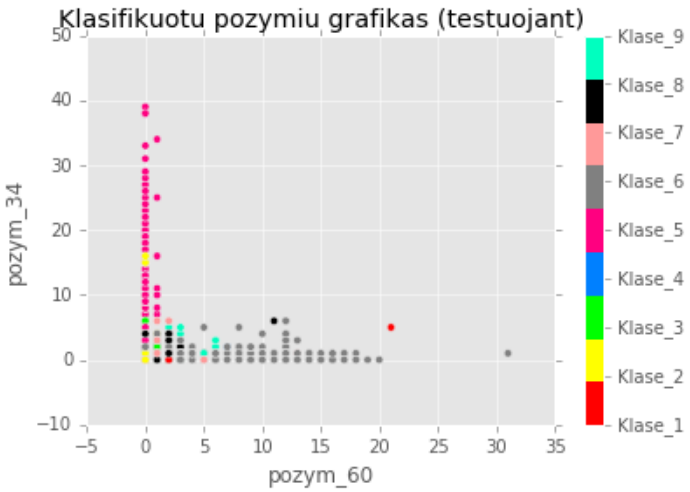
Siekiant visus gautus klasifikavimo rezultatus pavaizduoti taškinių grafikų (angl. *scatter plot*) pavidalu, reikalingos kryptys, kurios būtų logiškos. Atsitiktinių miškų klasifikavimo metodas išdėlioja atrinktus požymius pagal svarbumą (žr. 3.3.11 pav.). Pastebėta, kad visi analizuoti algoritmai išlaiko tuos kintamuosius, kuriuos pagal svarbumą išdėlioja atsitiktinių miškų klasifikavimo metodas. Tokie kintamieji yra, pavyzdžiui, *pozym_11*, *pozym, 60*, *pozym_34*, *pozym_40* ir kiti. 10 svarbių kintamųjų grafikas pateikiamas 3.3.10 paveiksle (žr. 3.3.11 pav.). Vizualizuojami 11, 60, 34 ir 40 požymiai ant testavimo imties pirmo bandymo metu, panaudojant RPA algoritmą ir taškų sklaidos diagramą (žr 3.3.11, 3.3.12 ir 3.3.13 pav.). Akivaizdu, sprendžiant iš 12 grafiko (žr. 3.3.12 pav.), kad dauguma stebėjimų patenka į 6 klasę. Tai reiškia, kad 11 ir 60 požymiai gerai nusako 6 klasę. Kiti grafikai (žr. 3.3.13 ir 3.3.14 pav.) parodo, kad 34 ir 60 požymiai pakankamai gerai atskiria 5 ir 6 klases, o 34 ir 40 požymiai – 2 ir 5 klases. Mažos 60 požymio ir didelės 34 požymio reikšmės gerai apibrėžia 5 klasę, priešingai – 6 klasę. Mažos 34 požymio ir didelės 40 požymio reikšmės gerai nusako 2 klasę, priešingai – 5 klasę.



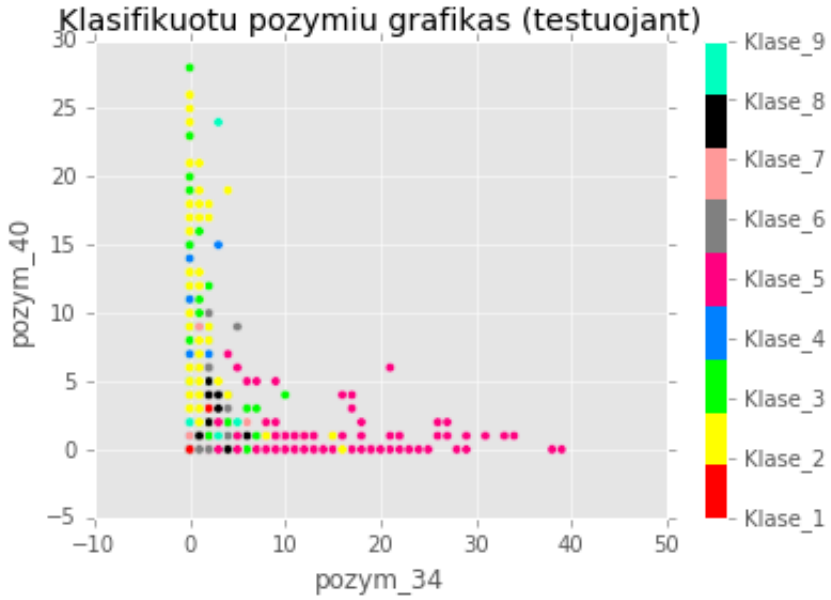
3.3.11 pav. Požymių svarbumas pagal Gini indeksą



3.3.12 pav. 11 ir 60 požymių taškų sklaidos diagrama pagal klases

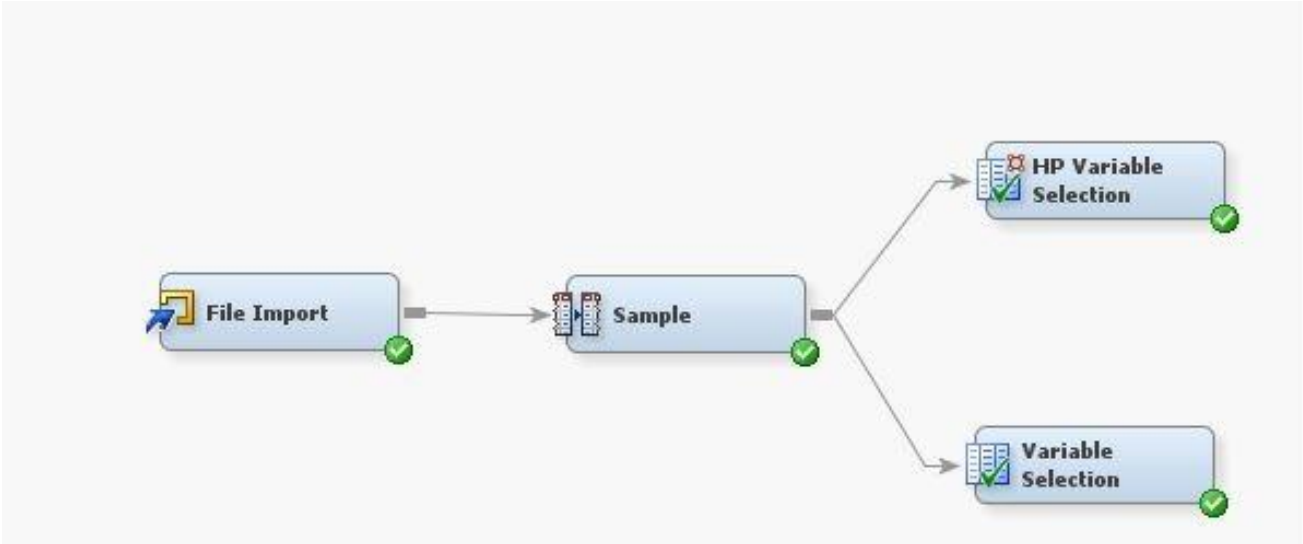


3.3.13 pav. 60 ir 34 požymių taškų sklaidos diagrama pagal klases



3.3.14 pav. 34 ir 40 požymių taškų sklaidos diagrama pagal klases

Svarbius požymius buvo bandoma surasti, panaudojant SAS (angl. *Statistical Analysis System*) „Enterprise Miner“ įrankį. Jis leidžia duomenų masyvams sudaryti tikslinę (pilną) proceso diagramą (šiuo atveju, požymių atranką) (žr. 3.3.15 pav.). Aukštos kokybės požymių atrankos etape automatiškai, be didelių vartotojo pastangų, sudaromi sprendimo medžiai. Jų gauti požymiai pagal svarbumą išsirikiuoja analogiškai, kaip ir 3.3.11 paveiksle (žr. 3.3.11 pav.). Iš to sprendžiame, kad programiškai modeliuoti atsitiktiniai miškai gerai atlieka požymių atrankos procesą.



3.3.15 pav. Požymių atrankos diagrama SAS „Enterprise Miner“ įrangoje

Rekomendacijos tolimesniems tyrimams:

Baigiamajame projekte siūlomos svarbių požymių atrankos ir jų klasifikavimo metodikos gali būti taikomos įvairiems klasifikavimo uždaviniams, kurie įtraukia didelį kiekį kintamųjų, prieš tai pakoreguojant duomenų nuskaitymo ir modelio hiperparametrų parinkimo programines eilutes R ir Python programinėse įrangose. Svarbu pabrėžti, kad modeliuojami algoritmai atima nemažai kompiuterinių resursų, todėl rekomenduojama analizę atlikti, panaudojant didesnių resursų kompiuterius.

IŠVADOS

1. Eksponentiškai didėjant analizuojamų duomenų kiekiams, aktualus uždavinys yra klasifikavimui reikšmingų požymių atranka. Išanalizavus literatūrą, nustatyta, kad požymių atrankai naudojami keli metodai, kurie skiriasi algoritmo įvykdymo greičiu, atrenkamų požymių dalimi ir klasifikavimo pagal atrinktus požymius kokybe. Klasifikavimo uždavinio sprendimą apsunkina tai, kad nėra metodikos ir programinių priemonių, leidžiančių patogiai palyginti kelis požymių atrankos metodus.
2. Darbe pasiūlyta metodika, integruojanti požymių atrankos metodus, optimizavimo algoritmus (genetinis algoritmas, modeliuotas atkaitinimas ir rekursinė požymių atranka) ir leidžianti palyginti klasifikavimo atraminių vektorių metodų kokybės metrikas, kai taikomi keli metodai. Gauti rezultatai lyginami su rezultatais, gautais, taikant atsitiktinių miškų vidinius požymių atrankos ir klasifikavimo modelius.
3. Panaudojus programinę įrangą Python, R ir SAS, sukurta metodikos programinė realizacija, kuri ženkliai sumažina darbo sąnaudas, parenkant geriausius požymių atrankos metodus.
4. Atliktas 4 požymių atrankos metodų palyginimas, klasifikuojant realų duomenų rinkinį. Nustatyta, kad:
    - Taikant požymių atrankai modeliuoto atkaitinimo algoritmą, pradinis požymių skaičius sumažėjo 74,19 – 78,49 %, bet reikšmingai sumažėjo ir klasifikavimo kokybė. Teisingo klasifikavimo tikslumas apmokymo imčiai buvo 60 – 70 %, *f* – įvertis – 42 – 57 %, jautrumas – 76 – 78 %. Testavimo imčiai teisingai klasifikuotų stebėjimų tikslumas buvo 50 – 60 %, *f* – įvertis – 34 – 40 %, jautrumas – 52 – 59 %. Modeliuoto atkaitinimo metodas netinka požymių atrankai, sprendžiant klasifikavimo uždavinius. Klasifikavimo vidutinė trukmė buvo 174 minutės.
    - Pritaikius atsitiktinius miškus, buvo teisingai suklasifikuojami visi apmokymo imties duomenys. Apmokymo imčiai klasifikavimo tikslumas sumažėjo iki 76 – 78 %, jautrumas – 74 – 78 %, *f* – įvertis – 67 – 69 %. Atsitiktinių miškų klasifikatorius geriausiai tiko apmokymo imčiai klasifikuoti. Požymių skaičius sumažėjo 49,46 %. Klasifikavimo vidutinė trukmė buvo 775 minutės.
    - Taikant rekursinę požymių atranką, buvo teisingai suklasifikuojami 83 – 93 % apmokymo imties stebėjimų. Testavimo imčiai gaunamas tikslumas siekė 78 – 82 %, *f* – įvertis – 77 – 82 %, jautrumas – 77 – 82 %, teisingai klasifikuotų stebėjimų santykinis dydis – 77 – 82 %. Šis metodų junginys gerai tiko atskirti testavimo imties

klases. Požymių skaičius buvo sumažinamas 12,9 – 32,26 %. Klasifikavimo vidutinė trukmė buvo 387 minutės.

- Pritaikius genetinį algoritmą, teisingai klasifikuotų stebėjimų santykinis dydis buvo 94 – 97 %, o analizuojant apmokymo imtį – apie 66 %. Apmokymo imčiai gaunamas tikslumas siekė 90 – 96 %, *f* – įvertis – 92 – 97 %, jautrumas – 96 – 98 %, o testavimo imčiai apskaičiuojamas tikslumas buvo 50 – 53 %, *f* – įvertis – 53 – 57 %, jautrumas – 67 – 73 %. Pradinė imtis buvo sumažinta 8,6 – 12,9 %. Metodo įvykdymo trukmė buvo beveik dvigubai ilgesnė, nei atliekant modeliuoto atkaitinimo algoritmą. Klasifikavimo vidutinė trukmė buvo 265 minutės.

5. Atraminių vektorių metodas teisingai klasifikavo 92,60 % apmokymo imties stebėjimų ir 74,36 % testavimo imties stebėjimų. Klasifikavimas vidutiniškai truko 450 minučių. Metodai, atrenkantys mažesnį pradinių požymių skaičių (rekursinė požymių atranka, genetinis algoritmas, modeliuoto atkaitinimo algoritmas, atsitiktiniai miškai) ženkliai sumažina klasifikavimo laiką, o klasifikavimo kokybė reikšmingai nesumažėja.

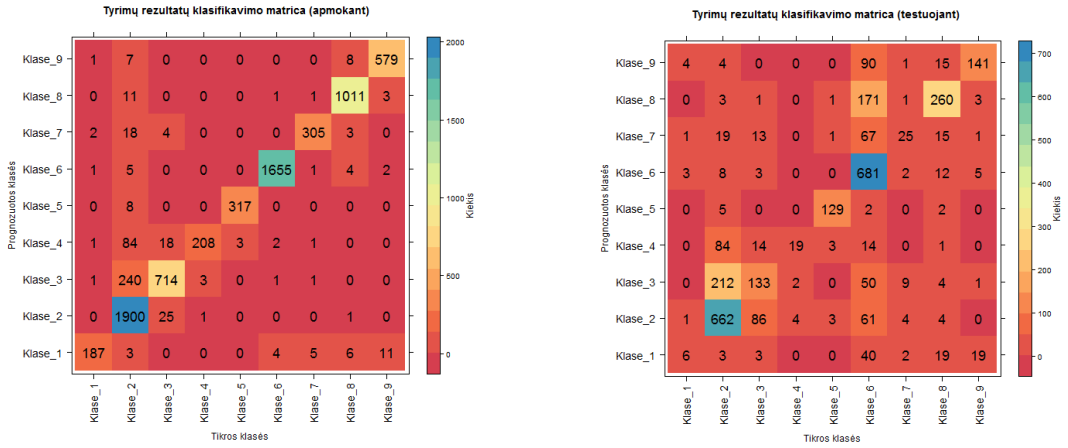
LITERATŪROS SĄRAŠAS

1. JANECEK, A., GANSTERER, W.N., DEMEL, M. and ECKER, G. *On the Relationship between Feature Selection and Classification Accuracy.* Citeseer, 2008.
2. WANG, A., et al. Accelerating Wrapper-Based Feature Selection with K-Nearest-Neighbor. *Knowledge-Based Systems*, 7, 2015, vol. 83. pp. 81-91 ISSN 0950-7051.
3. TANG, J., ALELYANI, S. and LIU, H. Feature Selection for Classification: A Review. *Data Classification: Algorithms and Applications*, 2014. pp. 37.
4. JAGANATHAN, P. and KUPPUCHAMY, R. A Threshold Fuzzy Entropy Based Feature Selection for Medical Database Classification. *Computers in Biology and Medicine*, 12/1, 2013, vol. 43, no. 12. pp. 2222-2229 ISSN 0010-4825.
5. XUE, B., ZHANG, M. and BROWNE, W.N. Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach. *Cybernetics, IEEE Transactions On*, 2013, vol. 43, no. 6. pp. 1656-1671.
6. TAN, M., TSANG, I.W. and WANG, L. Towards Ultrahigh Dimensional Feature Selection for Big Data. *The Journal of Machine Learning Research*, 2014, vol. 15, no. 1. pp. 1371-1429.
7. GUERRA-SALCEDO, C., CHEN, S., WHITLEY, D. and SMITH, S. *Fast and Accurate Feature Selection using Hybrid Genetic Strategies.* IEEE, 1999.
8. LANDRY, J., DA COSTA, L. and BERNIER, T. Discriminant Feature Selection by Genetic Programming: Towards a Domain Independent Multi-Class Object Detection System. *Journal of Systemics, Cybernetics and Informatics*, 2006, vol. 3, no. 1. pp. 76-81.
9. BALA, J., et al. *Hybrid Learning using Genetic Algorithms and Decision Trees for Pattern Classification.* , 1995.
10. PERALTA, D., et al. Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach. *Mathematical Problems in Engineering*, 2015, vol. 501. pp. 246139.
11. MARCOULIDES, G.A. Discovering Knowledge in Data: An Introduction to Data Mining. *Journal of the American Statistical Association*, 2005, vol. 100, no. 472. pp. 1465-1465.
12. LI, T., ZHU, S. and OGIHARA, M. *Using Discriminant Analysis for Multi-Class Classification.* IEEE, 2003.
13. HAN, J., KAMBER, M. and PEI, J. *Data Mining: Concepts and Techniques.* Elsevier, 2011.
14. REBY, D., et al. Artificial Neural Networks as a Classification Method in the Behavioural Sciences. *Behavioural Processes*, 1997, vol. 40, no. 1. pp. 35-43.
15. WU, C.H. and MCLARTY, J.W. *Neural Networks and Genome Informatics.* Elsevier, 2012.
16. VYDAS ČEKANAVIČIUS, G.M. *Statistika Ir Jos Taikymai 2.* Vilnius: TEV, 2002 ISBN 9955–491–16–7.
17. MEYER, D. and WIEN, F.T. Support Vector Machines. *The Interface to Libsvm in Package e1071*, 2015.
18. DREISEITL, S. and OHNO-MACHADO, L. Logistic Regression and Artificial Neural Network Classification Models: A Methodology Review. *Journal of Biomedical Informatics*, 2002, vol. 35, no. 5. pp. 352-359.

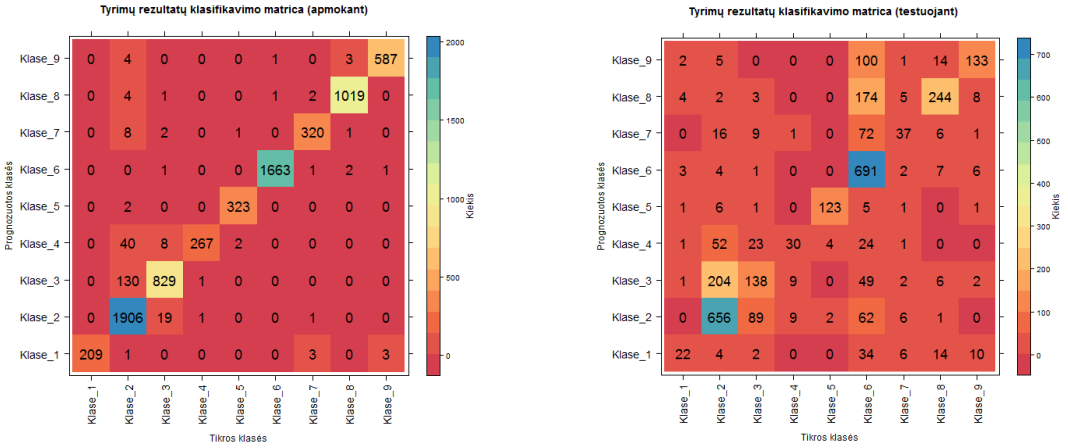
19. KUHN, M. and JOHNSON, K. *Applied Predictive Modeling.* Springer, 2013.
20. DU, K. and SWAMY, M. *Neural Networks and Statistical Learning.* Springer Science & Business Media, 2013.
21. ABE, S. *Support Vector Machines for Pattern Classification.* Springer, 2005.

1 PRIEDAS. Klasifikavimo rezultatų matricos, gautos taikant genetinį algoritmą ir atraminių vektorių klasifikatorių (apmokymo ir testavimo imtims)

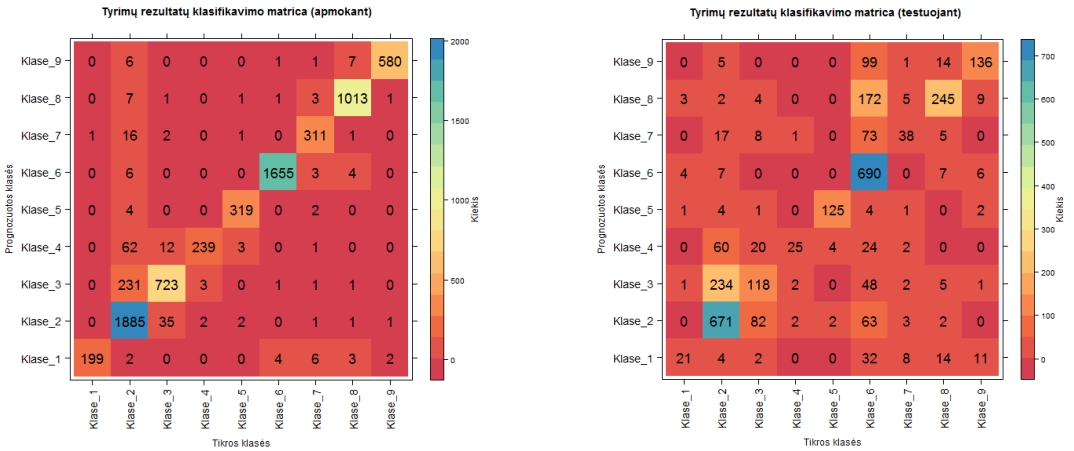
Iteracija = 2



Iteracija = 3



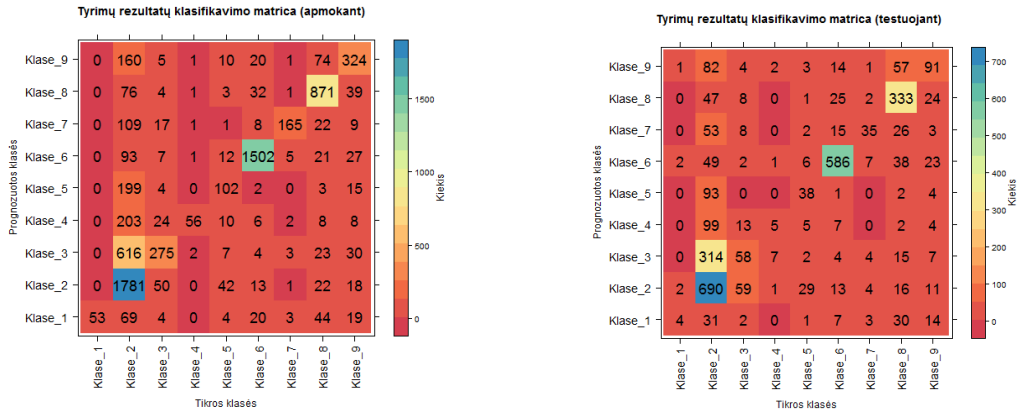
Iteracija = 4



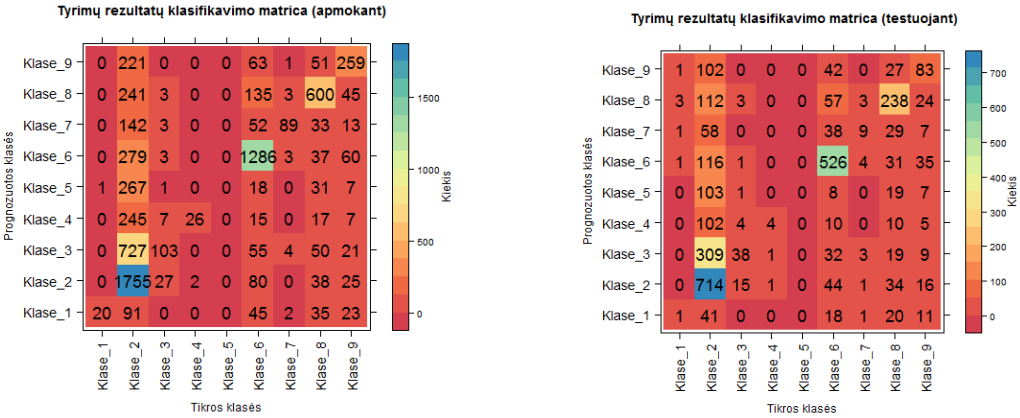
1 P. 1 pav. GA ir AVK klasifikavimo rezultatai

2 PRIEDAS. Klasifikavimo rezultatų matricos, gautos taikant modeliuoto atkaitinimo algoritmą ir atraminių vektorių klasifikatorių (apmokymo ir testavimo imtims)

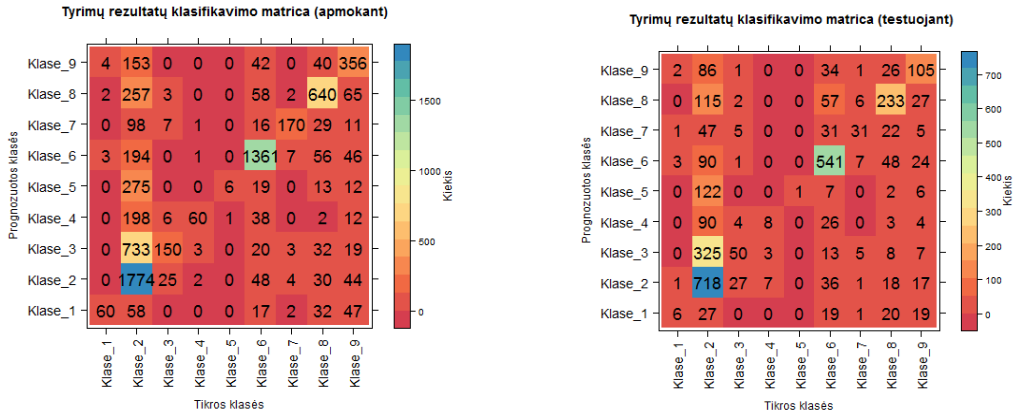
Iteracija = 2



Iteracija = 3



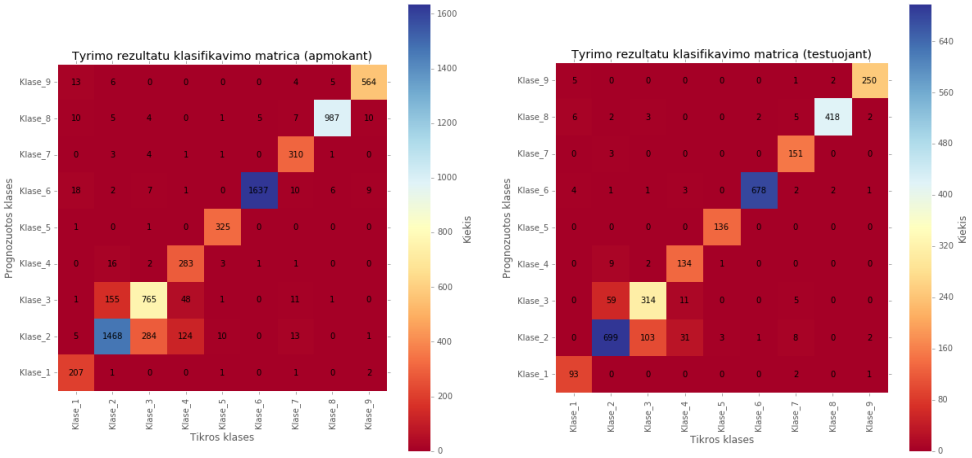
Iteracija = 4



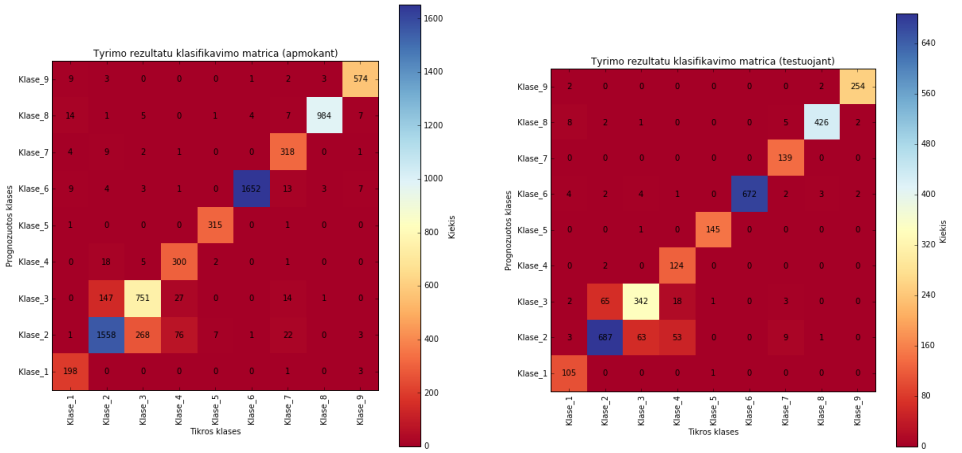
2 P. 1 pav. MA algoritmo ir AVK klasifikavimo rezultatai

3 PRIEDAS. Klasifikavimo rezultatų matricos, gautos taikant rekursinį požymių atrankos algoritmą ir atraminių vektorių klasifikatorių (apmokymo ir testavimo imtims)

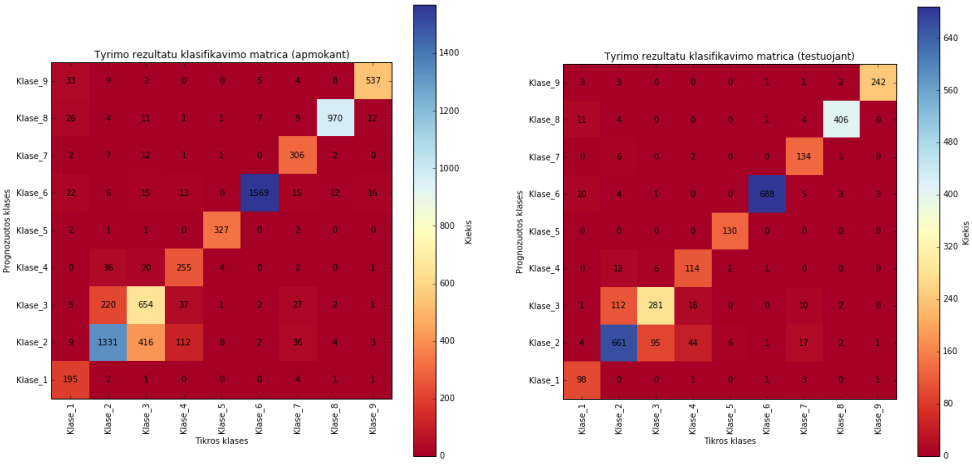
Iteracija = 2



Iteracija = 3



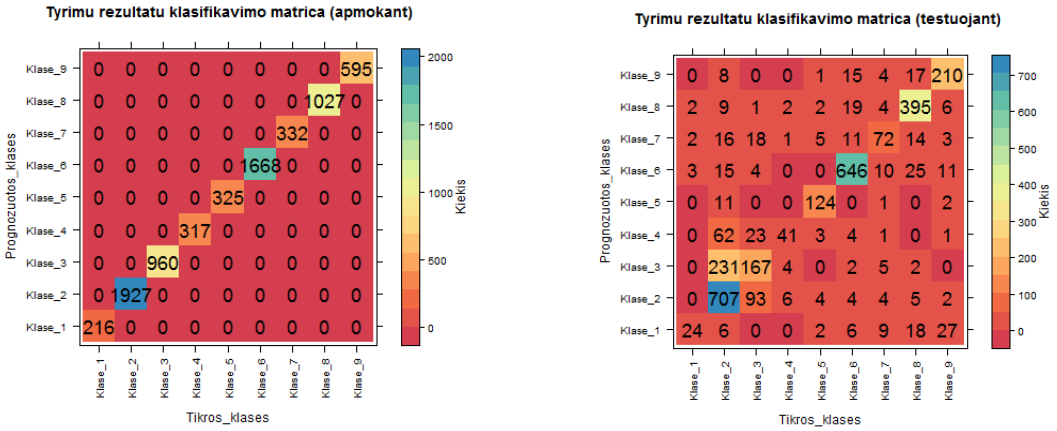
Iteracija = 4



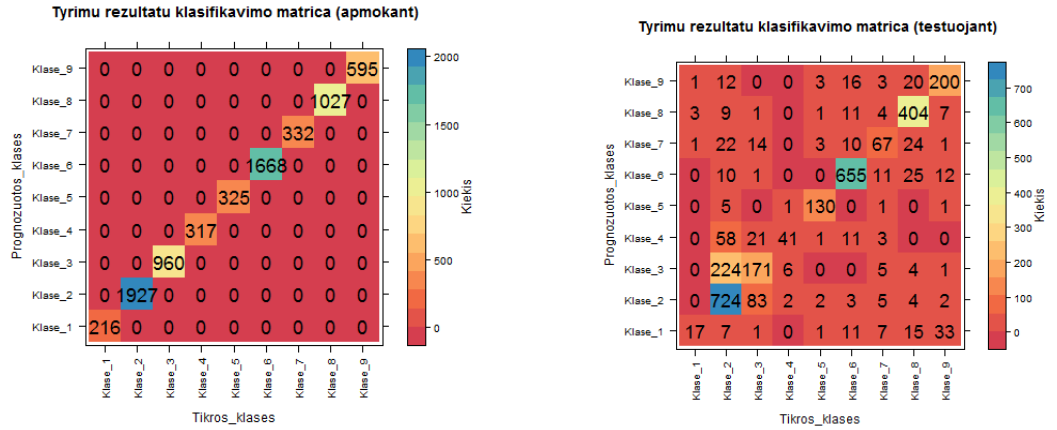
3 P. 1 pav. RPA algoritmo ir AVK klasifikavimo rezultatai

4 PRIEDAS. Klasifikavimo rezultatų matricos, gautos taikant atsitiktinių miškų klasifikatorių (apmokymo ir testavimo imtims)

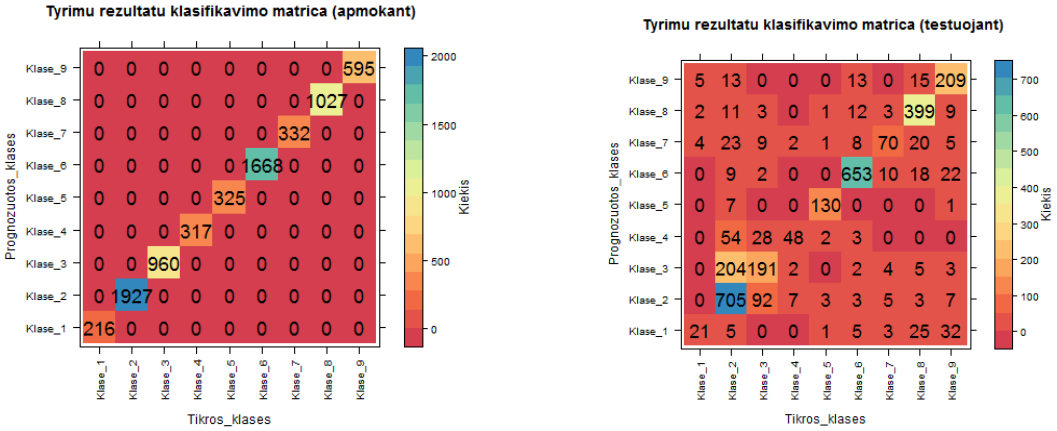
Iteracija = 2



Iteracija = 3



Iteracija = 4



4 P. 1 pav. Atsitiktinių miškų metodo klasifikavimo rezultatai

5 PRIEDAS. Genetinis požymių atrankos algoritmas atraminių vektorių klasifikavimo uždaviniui išspręsti, panaudojant R programinį paketą

```
1  # Nuskaitome duomenu faila
2  setwd("C:/Users/User/Desktop")
3  getwd()
4  duomenys <- read.csv("Pradinis.csv")
5  dim(duomenys)
6  head(duomenys, 3)

7  # Bibliotekos
8  library(caret)
9  library(doParallel)
10 library(dplyr)
11 library(pROC)
12 library(e1071)
13 library(lattice)
14 library(RColorBrewer)
15 library(latticeExtra)
16 library(grid)

17 # Isskiriame apmokymo ir testavimo imtis
18 Duomdalinimas <- createDataPartition(duomenys$kategorija,p = .7,list = FALSE)
19 Apmokymas <- duomenys[Duomdalinimas,-c(1,0)]
20 Testavimas <- duomenys[-Duomdalinimas,-c(1,0)]
21 y = Apmokymas$kategorija

22 barchart(Apmokymas$kategorija, main="Juostinė diagrama",
   count.labels=Apmokymas$kategorija, xlab = list(label="Klasės"), ylab =
   list(label="Dažnis"), scales=list(x=list(rot=90)), horizontal=F, par.settings =
   list(plot.polygon = list(col = c("#1E4150"))))

23 # Pozymiu atranka su GA
24 registerDoParallel(2)
25 getDoParWorkers()
```

```
Genetinis <- gafsControl(functions = rfGA, method = "cv", genParallel=TRUE, allowParallel = TRUE)
lygiai <- c("Klase_1","Klase_2","Klase_3","Klase_4","Klase_5","Klase_6","Klase_7","Klase_8","Klase_9")
system.time(Genetinis_main <- gafs(x = Apmokymas, y = y, iters = 10, popSize = 93, levels = lygiai, gafsControl = Genetinis))
plot(Genetinis_main, col=c("green", "red"))
print (Genetinis_main)


ptm <- proc.time()
# Atraminiu vektoriu metodas (SVM)
final <- Genetinis_main$ga$final # Pasiima GA parinktus pozymius
apmokymas_2 <- Apmokymas[,final] # Apmokymo imtis
head(apmokymas_2,2)
dim(apmokymas_2)
testavimas_2 <- Testavimas[,final] # Testavimo imtis


svm_modelis <- svm(kategorija~., data = apmokymas_2)
summary(svm_modelis)
spejimas <- predict(svm_modelis, apmokymas_2)
table(spejimas, apmokymas_2$kategorija)
# Parametru tuningas
svm_tuningas <- tune(svm, kategorija~., data = apmokymas_2, kernel = "radial", ranges = list(cost = c(0.5,1,2,5,10), gamma = c(.1,.5,1,2)))
print(svm_tuningas)
svm_modelis_po_tuningo <- svm(kategorija ~ ., data = apmokymas_2, kernel = "radial", cost = 2, gamma = 0.1)
summary(svm_modelis_po_tuningo)
spejimas_2 <- predict(svm_modelis_po_tuningo, apmokymas_2)
table(spejimas_2, apmokymas_2$kategorija)
svm_spejimas <- predict(svm_modelis_po_tuningo, testavimas_2)
table(svm_spejimas, testavimas_2$kategorija)
plot(spejimas_2)
plot(svm_spejimas)
```

```
53 proc.time() - ptm

54 # Rezultatas
55 confusionMatrix(spejimas_2, apmokymas_2$kategorija)
56 confusionMatrix(svm_spejimas, testavimas_2$kategorija)

57 # Statistikos (apmokymui)
58 c = apmokymas_2$kategorija
59 statistika1 = as.matrix(table(spejimas_2, c))
60 n = sum(statistika1)
61 Klasiu_sk = nrow(statistika1)
62 diagonali = diag(statistika1)
63 eilsuma = apply(statistika1, 1, sum)
64 stulsuma = apply(statistika1, 2, sum)
65 a = eilsuma / n
66 b = stulsuma / n
67 accuracy = sum(diagonali) / n
68 accuracy
69 precision = diagonali / stulsuma
70 recall = diagonali / eilsuma
71 f1 = 2 * precision * recall / (precision + recall)
72 data.frame(precision, recall, f1)
73 GalutinisPrecision = mean(precision)
74 GalutinisRecall = mean(recall)
75 GalutinisF1 = mean(f1)
76 data.frame(GalutinisPrecision, GalutinisRecall, GalutinisF1)

77 # Statistikos (testavimui)
78 v = testavimas_2$kategorija
79 statistika1 = as.matrix(table(svm_spejimas, v))
80 n = sum(statistika1)
81 Klasiu_sk = nrow(statistika1)
82 diagonali = diag(statistika1)
83 eilsuma = apply(statistika1, 1, sum)
84 stulsuma = apply(statistika1, 2, sum)
```

```
a = eilsuma / n
b = stulsuma / n
accuracy = sum(diagonali) / n
accuracy
precision = diagonali / stulsuma
recall = diagonali / eilsuma
f1 = 2 * precision * recall / (precision + recall)
data.frame(precision, recall, f1)
GalutinisPrecision = mean(precision)
GalutinisRecall = mean(recall)
GalutinisF1 = mean(f1)
data.frame(GalutinisPrecision, GalutinisRecall, GalutinisF1)


# Grafikai
taip <- table(spejimas_2, c)
ne <- table(svm_spejimas, v)
spalva <- colorRampPalette(brewer.pal(8, "Spectral"))
# Silumos grafikas
levelplot(taip, col.regions=spalva, scales=list(x=list(rot=90,cex=1.2), y=list(cex=1.2)), xlab=list(label="Tikros klasės"), ylab=list(label="Prognozuotos klasės"), main = "Tyrimų rezultatų klasifikavimo matrica (apmokant)") +
  layer(panel.text(c(1), c(1:9),nrow=2, taip[1,1:9], cex=1.5))+
  layer(panel.text(c(2), c(1:9),nrow=2, taip[2,1:9], cex=1.5))+
  layer(panel.text(c(3), c(1:9),nrow=2, taip[3,1:9], cex=1.5))+
  layer(panel.text(c(4), c(1:9),nrow=2, taip[4,1:9], cex=1.5))+
  layer(panel.text(c(5), c(1:9),nrow=2, taip[5,1:9], cex=1.5))+
  layer(panel.text(c(6), c(1:9),nrow=2, taip[6,1:9], cex=1.5))+
  layer(panel.text(c(7), c(1:9),nrow=2, taip[7,1:9], cex=1.5))+
  layer(panel.text(c(8), c(1:9),nrow=2, taip[8,1:9], cex=1.5))+
  layer(panel.text(c(9), c(1:9),nrow=2, taip[9,1:9], cex=1.5))
levelplot(ne, col.regions=spalva, scales=list(x=list(rot=90,cex=1.2), y=list(cex=1.2)), xlab=list(label="Tikros klasės"), ylab=list(label="Prognozuotos klasės"), main = "Tyrimų rezultatų klasifikavimo matrica (testuojant)") +
  layer(panel.text(c(1), c(1:9),nrow=2, ne[1,1:9], cex=1.5))+
  layer(panel.text(c(2), c(1:9),nrow=2, ne[2,1:9], cex=1.5))+
```

```
  layer(panel.text(c(3), c(1:9),nrow=2, ne[3,1:9], cex=1.5))+
  layer(panel.text(c(4), c(1:9),nrow=2, ne[4,1:9], cex=1.5))+
  layer(panel.text(c(5), c(1:9),nrow=2, ne[5,1:9], cex=1.5))+
  layer(panel.text(c(6), c(1:9),nrow=2, ne[6,1:9], cex=1.5))+
  layer(panel.text(c(7), c(1:9),nrow=2, ne[7,1:9], cex=1.5))+
  layer(panel.text(c(8), c(1:9),nrow=2, ne[8,1:9], cex=1.5))+
  layer(panel.text(c(9), c(1:9),nrow=2, ne[9,1:9], cex=1.5))
trellis.focus("legend", side="right", clipp.off=TRUE, highlight=FALSE)
grid.text(expression(paste("Kiekis",sep="")),1.1,0.50,rot=90)
trellis.unfocus()


plot(Apmokymas[,c(59,33)], col=as.numeric(spejimas_2), pch=16)
legend("topright",legend=unique(spejimas_2),
col=unique(as.numeric(spejimas_2)), pch=16)
title(main = "Klasifikuotų požymių grafikas(apmokant)")


plot(Testavimas[,c(9,16)], col=unclass(svm_spejimas),pch=16)
legend("topright",legend=unique(svm_spejimas),
col=unique(as.numeric(svm_spejimas)), pch=16)
title(main = "Klasifikuotų požymių grafikas(testuojant)")
rfGA$initial(vars = 93, popSize = 93)
```

6 PRIEDAS. Modeliuoto atkaitinimo požymių atrankos algoritmas atraminių vektoriui klasifikavimo uždaviniui išspręsti, panaudojant R programinį paketą

```
# Nuskaitome duomenu faila
setwd("C:/Users/User/Desktop")
getwd()
duomenys <- read.csv("Pradinis.csv")
dim(duomenys)
head(duomenys, 3)

# Bibliotekos
library(caret)
library(doParallel)
library(dplyr)
library(e1071)
library(lattice)
library(RColorBrewer)
library(latticeExtra)
library(grid)

# Isskiriame apmokymo ir testavimo imtis
Duomdalinimas <- createDataPartition(duomenys$kategorija,p = .7,list = FALSE)
Apmokymas <- duomenys[Duomdalinimas,-c(1,0)]
Testavimas <- duomenys[-Duomdalinimas,-c(1,0)]
y = Apmokymas$kategorija

# Pozymiu atranka su modeliuotu atkaitinimu (SA)
Atkaitinimas <- safsControl(functions = rfSA, method = "cv", allowParallel = TRUE)
lygiai <- c("Klase_1","Klase_2","Klase_3","Klase_4","Klase_5","Klase_6","Klase_7","Klase_8","Klase_9")
system.time(Mod_atkaitinimas <- safs(x = Apmokymas, y = y, iters = 8, levels = lygiai, safsControl = Atkaitinimas))
print(Mod_atkaitinimas)
plot(Mod_atkaitinimas)
```

```
27 names(Mod_atkaitinimas)
28 # Atraminiu vektoriu metodas (SVM)
29 taip <- Mod_atkaitinimas$resampled_vars
30 taip # Pasiima SA parinktus pozymius
31 final <- Mod_atkaitinimas$sa$final
32 apmokymas_22 <- Apmokymas[final] # Apmokymo imtis
33 head(apmokymas_22,2)
34 dim(apmokymas_22)
35 testavimas_22 <- Testavimas[,final] # Testavimo imtis

36 ptm <- proc.time()
37 svm_modelis <- svm(kategorija~., data = apmokymas_22)
38 summary(svm_modelis)
39 spejimas <- predict(svm_modelis, apmokymas_22)
40 table(spejimas, y)

41 # Parametru tuningas
42 svm_tuningas <- tune(svm, kategorija~., data = apmokymas_22, kernel = "radial",
   ranges = list(cost = c(0.5,1,2,5,10), gamma = c(.1,.5,1,2)))
43 print(svm_tuningas)
44 svm_modelis_po_tuningo <- svm(kategorija ~ ., data = apmokymas_22, kernel =
   "radial", cost = 1, gamma = 0.5)
45 summary(svm_modelis_po_tuningo)
46 spejimas_2 <- predict(svm_modelis_po_tuningo, apmokymas_22)
47 table(spejimas_2, apmokymas_22$kategorija)
48 svm_spejimas <- predict(svm_modelis_po_tuningo, testavimas_22)
49 table(svm_spejimas, testavimas_22$kategorija)
50 plot(spejimas_2)
51 plot(svm_spejimas)
52 proc.time() - ptm

53 # Rezultatas
54 confusionMatrix(spejimas_2, apmokymas_22$kategorija)
55 confusionMatrix(svm_spejimas, testavimas_22$kategorija)
```

```
# Statistikos (apmokymui)
c = apmokymas_22$kategorija
statistika1 = as.matrix(table(spejimas_2, c))
n = sum(statistika1)
Klasiu_sk = nrow(statistika1)
diagonali = diag(statistika1)
eilsuma = apply(statistika1, 1, sum)
stulsuma = apply(statistika1, 2, sum)
a = eilsuma / n
b = stulsuma / n
accuracy = sum(diagonali) / n
accuracy
precision = diagonali / stulsuma
recall = diagonali / eilsuma
f1 = 2 * precision * recall / (precision + recall)
data.frame(precision, recall, f1)
GalutinisPrecision = mean(precision)
GalutinisRecall = mean(recall)
GalutinisF1 = mean(f1)
data.frame(GalutinisPrecision, GalutinisRecall, GalutinisF1)


# Statistikos (testavimui)
v = testavimas_22$kategorija
statistika1 = as.matrix(table(svm_spejimas, v))
n = sum(statistika1)
Klasiu_sk = nrow(statistika1)
diagonali = diag(statistika1)
eilsuma = apply(statistika1, 1, sum)
stulsuma = apply(statistika1, 2, sum)
a = eilsuma / n
b = stulsuma / n
accuracy = sum(diagonali) / n
accuracy
precision = diagonali / stulsuma
recall = diagonali / eilsuma
```

```
f1 = 2 * precision * recall / (precision + recall)
data.frame(precision, recall, f1)
GalutinisPrecision = mean(precision)
GalutinisRecall = mean(recall)
GalutinisF1 = mean(f1)
data.frame(GalutinisPrecision, GalutinisRecall, GalutinisF1)


# Grafikai
taip <- table(spejimas_2, c)
ne <- table(svm_spejimas, v)
spalva <- colorRampPalette(brewer.pal(8, "Spectral"))
# Silumos grafikas
levelplot(taip, col.regions=spalva, scales=list(x=list(rot=90,cex=1.2), y=list(cex=1.2)),
xlab=list(label="Tikros klasės"), ylab=list(label="Prognozuotos klasės"), main =
"Tyrimų rezultatų klasifikavimo matrica (apmokant)") +
  layer(panel.text(c(1), c(1:9),nrow=2, taip[1,1:9], cex=1.5))+
  layer(panel.text(c(2), c(1:9),nrow=2, taip[2,1:9], cex=1.5))+
  layer(panel.text(c(3), c(1:9),nrow=2, taip[3,1:9], cex=1.5))+
  layer(panel.text(c(4), c(1:9),nrow=2, taip[4,1:9], cex=1.5))+
  layer(panel.text(c(5), c(1:9),nrow=2, taip[5,1:9], cex=1.5))+
  layer(panel.text(c(6), c(1:9),nrow=2, taip[6,1:9], cex=1.5))+
  layer(panel.text(c(7), c(1:9),nrow=2, taip[7,1:9], cex=1.5))+
  layer(panel.text(c(8), c(1:9),nrow=2, taip[8,1:9], cex=1.5))+
  layer(panel.text(c(9), c(1:9),nrow=2, taip[9,1:9], cex=1.5))
levelplot(ne, col.regions=spalva, scales=list(x=list(rot=90,cex=1.2), y=list(cex=1.2)),
xlab=list(label="Tikros klasės"), ylab=list(label="Prognozuotos klasės"), main =
"Tyrimų rezultatų klasifikavimo matrica (testuojant)") +
  layer(panel.text(c(1), c(1:9),nrow=2, ne[1,1:9], cex=1.5))+
  layer(panel.text(c(2), c(1:9),nrow=2, ne[2,1:9], cex=1.5))+
  layer(panel.text(c(3), c(1:9),nrow=2, ne[3,1:9], cex=1.5))+
  layer(panel.text(c(4), c(1:9),nrow=2, ne[4,1:9], cex=1.5))+
  layer(panel.text(c(5), c(1:9),nrow=2, ne[5,1:9], cex=1.5))+
  layer(panel.text(c(6), c(1:9),nrow=2, ne[6,1:9], cex=1.5))+
  layer(panel.text(c(7), c(1:9),nrow=2, ne[7,1:9], cex=1.5))+
  layer(panel.text(c(8), c(1:9),nrow=2, ne[8,1:9], cex=1.5))+
```

```
121   layer(panel.text(c(9), c(1:9),nrow=2, ne[9,1:9], cex=1.5))
122 trellis.focus("legend", side="right", clipp.off=TRUE, highlight=FALSE)
123 grid.text(expression(paste("Kiekis",sep="")),1.1,0.50,rot=90)
124 trellis.unfocus()

125 plot(Apmokymas[,c(33,39)], col=as.numeric(spejimas_2), pch=16)
126 legend("topright",legend=unique(spejimas_2),      col=unique(as.numeric(spejimas_2)),
127 pch=16)
128 title(main = "Klasifikuotų požymių grafikas(apmokant)")

129 plot(Testavimas[,c(33,39)], col=unclass(svm_spejimas),pch=16)
130 legend("topright",legend=unique(svm_spejimas),
131 col=unique(as.numeric(svm_spejimas)), pch=16)
132 title(main = "Klasifikuotų požymių grafikas(testuojant)")
```

7 PRIEDAS. Atsitiktinių miškų klasifikatorius, panaudojant R programinį paketą

```
1  # Nuskaitome duomenu faila
2  setwd("C:/Users/DELL/Desktop/GALUTINIS")
3  getwd()
4  duomenys <- read.csv("Pradinis.csv")
5  dim(duomenys)
6  head(duomenys, 3)

7  # Bibliotekos
8  require(caret)
9  require(ggplot2)
10 require(randomForest)
11 library(RColorBrewer)
12 library(latticeExtra)
13 library(grid)
14 library(lattice)

15 # Isskiriame apmokymo ir testavimo imtis
16 Duomdalinimas <- createDataPartition(duomenys$kategorija,p = .7,list = FALSE)
17 Apmokymas <- duomenys[Duomdalinimas,-c(1,0)]
18 Testavimas <- duomenys[-Duomdalinimas,-c(1,0)]
19 y = Apmokymas$kategorija

20 ptm <- proc.time()
21 # Atsitiktiniai medziai
22 # registerDoParallel(2)
23 # getDoParWorkers()
24 rf_modelis <- train(kategorija~., data = Apmokymas, method="rf",
   trControl=trainControl(method = "cv",number = 10), prox=TRUE)
25 print(rf_modelis)
26 print(rf_modelis$finalModel)
27 plot(rf_modelis, col = "mediumpurple3", lwd = 3, pch = 16, cex=2)
28 plot(rf_modelis$finalModel)
29 plot(varImp(rf_modelis))
```

```
varImpPlot(rf_modelis$finalModel, n = 47)
Apmokymo_data <- predict(rf_modelis, Apmokymas)
Testavimo_data <- predict(rf_modelis, Testavimas)
proc.time() – ptm

# Rezultatas
confusionMatrix(data = Apmokymo_data, Apmokymas$kategorija)
confusionMatrix(data = Testavimo_data, Testavimas$kategorija)

# Statistikos (apmokymui)
c = Apmokymas$kategorija
statistika1 = as.matrix(table(Apmokymo_data, c))
n = sum(statistika1)
Klasiu_sk = nrow(statistika1)
diagonali = diag(statistika1)
eilsuma = apply(statistika1, 1, sum)
stulsuma = apply(statistika1, 2, sum)
a = eilsuma / n
b = stulsuma / n
accuracy = sum(diagonali) / n
accuracy
precision = diagonali / stulsuma
recall = diagonali / eilsuma
f1 = 2 * precision * recall / (precision + recall)
data.frame(precision, recall, f1)
GalutinisPrecision = mean(precision)
GalutinisRecall = mean(recall)
GalutinisF1 = mean(f1)
data.frame(GalutinisPrecision, GalutinisRecall, GalutinisF1)

# Statistikos (testavimui)
v = Testavimas$kategorija
statistika1 = as.matrix(table(Testavimo_data, v))
n = sum(statistika1)
Klasiu_sk = nrow(statistika1)
```

```
diagonali = diag(statistika1)
eilsuma = apply(statistika1, 1, sum)
stulsuma = apply(statistika1, 2, sum)
a = eilsuma / n
b = stulsuma / n
accuracy = sum(diagonali) / n
accuracy
precision = diagonali / stulsuma
recall = diagonali / eilsuma
f1 = 2 * precision * recall / (precision + recall)
data.frame(precision, recall, f1)
GalutinisPrecision = mean(precision)
GalutinisRecall = mean(recall)
GalutinisF1 = mean(f1)
data.frame(GalutinisPrecision, GalutinisRecall, GalutinisF1)

# Grafikai
taip <- table(Apmokymo_data, c)
ne <- table(Testavimo_data, v)
spalva <- colorRampPalette(brewer.pal(8, "Spectral"))

# Silumos grafikas
levelplot(taip, col.regions=spalva, scales=list(x=list(rot=90)),
xlab=list(label="Tikros_klases"), ylab=list(label="Prognozuotos_klases"), main =
"Tyrimu rezultatu klasifikavimo matrica (apmokant)") +
  layer(panel.text(c(1), c(1:9),nrow=2, taip[1,1:9], cex=1.5))+
  layer(panel.text(c(2), c(1:9),nrow=2, taip[2,1:9], cex=1.5))+
  layer(panel.text(c(3), c(1:9),nrow=2, taip[3,1:9], cex=1.5))+
  layer(panel.text(c(4), c(1:9),nrow=2, taip[4,1:9], cex=1.5))+
  layer(panel.text(c(5), c(1:9),nrow=2, taip[5,1:9], cex=1.5))+
  layer(panel.text(c(6), c(1:9),nrow=2, taip[6,1:9], cex=1.5))+
  layer(panel.text(c(7), c(1:9),nrow=2, taip[7,1:9], cex=1.5))+
  layer(panel.text(c(8), c(1:9),nrow=2, taip[8,1:9], cex=1.5))+
  layer(panel.text(c(9), c(1:9),nrow=2, taip[9,1:9], cex=1.5))
```

```
93  levelplot(ne, col.regions=spalva, scales=list(x=list(rot=90)),
    xlab=list(label="Tikros_klases"), ylab=list(label="Prognozuotos_klases"), main =
    "Tyrimu rezultatu klasifikavimo matrica (testuojant)") +
94    layer(panel.text(c(1), c(1:9),nrow=2, ne[1,1:9], cex=1.5))+
95    layer(panel.text(c(2), c(1:9),nrow=2, ne[2,1:9], cex=1.5))+
96    layer(panel.text(c(3), c(1:9),nrow=2, ne[3,1:9], cex=1.5))+
97    layer(panel.text(c(4), c(1:9),nrow=2, ne[4,1:9], cex=1.5))+
98    layer(panel.text(c(5), c(1:9),nrow=2, ne[5,1:9], cex=1.5))+
99    layer(panel.text(c(6), c(1:9),nrow=2, ne[6,1:9], cex=1.5))+
100   layer(panel.text(c(7), c(1:9),nrow=2, ne[7,1:9], cex=1.5))+
101   layer(panel.text(c(8), c(1:9),nrow=2, ne[8,1:9], cex=1.5))+
102   layer(panel.text(c(9), c(1:9),nrow=2, ne[9,1:9], cex=1.5))
103 trellis.focus("legend", side="right", clipp.off=TRUE, highlight=FALSE)
104 grid.text(expression(paste("Kiekis",sep="")),1.1,0.50,rot=90)
105 trellis.unfocus()

106 plot(Apmokymas[,c(9,16)], col=as.numeric(Apmokymo_data), pch=16)
107 legend("topright",legend=unique(Apmokymo_data),
108 col=unique(as.numeric(Apmokymo_data)), pch=16)
109 title(main = "Klasifikuotų požymių grafikas(apmokant)")

110 plot(Testavimas[,c(9,16)], col=unclass(Testavimo_data),pch=16)
111 legend("topright",legend=unique(Testavimo_data),
112 col=unique(as.numeric(Testavimo_data)), pch=16)
113 title(main = "Klasifikuotų požymių grafikas(testuojant)")
114 rforest <- randomForest(kategorija~., data = Apmokymas)
```

8 PRIEDAS. Rekursinis požymių atrankos algoritmas atraminių vektorių klasifikavimo uždaviniui išspręsti, panaudojant Python programinį paketą

```
% load_ext autoreload
% autoreload 2
# Bibliotekos:
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
import pylab as pl
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.feature_selection import RFECV
from sklearn.svm import SVR
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import StratifiedKFold
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import ExtraTreesClassifier
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn import grid_search
from sklearn.metrics import classification_report
from sklearn.learning_curve import learning_curve
from scipy import interp
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from mlxtend.evaluate import plot_decision_regions
```

```
33 from sklearn import cross_validation
34 from scipy import ndimage
35 # Isskiriame apmokymo ir testavimo imtis
36 Pradinis = pd.read_csv("Pradinis.csv")
37 Apmokymastrain, Testavimastest = cross_validation.train_test_split(Pradinis, test_size
   = 0.3) # Apmokymui imame 70 % duomenu
38 Klases = LabelEncoder().fit_transform(Apmokymastrain['kategorija'])
39 Apmokymas = Apmokymastrain.drop('kategorija', axis=1)
40 Klases_test = LabelEncoder().fit_transform(Testavimastest['kategorija'])
41 Testavimas = Testavimastest.drop('kategorija', axis=1)
42 X1, y1 = Apmokymas, Klases
43 Xtest, ytest = Testavimas, Klases_test
44 # list(Apmokymas)
45 Apmokymas.shape
46 # Apmokymas
47 import time
48 start = time.time()
49 # Pozymiu parinkimas su rekursiniu pozymiu eliminavimu arba trump. RFE
50 Modelis = SVC(kernel = "linear")
51 Paleisti = RFECV(estimator = Modelis, step = 1, cv = StratifiedKFold(y1, 2), scoring =
   'accuracy')
52 Paleisti.fit(X1, y1)
53 print("Optimalus pozymiu skaicius: %d" % Paleisti.n_features_)
54 plt.figure()
55 plt.xlabel("Atrinktu pozymiu skaicius")
56 plt.ylabel("Kryzminio ivercio (angl. cross - validation) balas (teisingu klasifikavimu
   skaicius)")
57 plt.plot(range(1, len(Paleisti.grid_scores_) + 1), Paleisti.grid_scores_, 'g')
58 plt.savefig('RFE.png')
59 plt.show()
60 Pakeista = Paleisti.fit_transform(X1, y1)
61 Pakeista.shape
62 Naujas1 = Paleisti.transform(X1)
63 Naujas1.shape
64 Naujas1
```

```
X2, y2 = Naujas1, Klases
Paleisti.support_
data = X1.drop(['pozym_1',
                'pozym_8',
                'pozym_10',
                'pozym_13',
                'pozym_16',
                'pozym_18',
                'pozym_20',
                'pozym_22',
                'pozym_24',
                'pozym_27',
                'pozym_32',
                'pozym_35',
                'pozym_38',
                'pozym_44',
                'pozym_48',
                'pozym_54',
                'pozym_56',
                'pozym_63',
                'pozym_64',
                'pozym_66',
                'pozym_67',
                'pozym_69',
                'pozym_70',
                'pozym_74',
                'pozym_79',
                'pozym_80',
                'pozym_85',
                'pozym_86'
                 ],1)

Keisti = Xtest.drop(['pozym_1',
                'pozym_8',
                'pozym_10',
```

```
100             'pozym_13',
101             'pozym_16',
102             'pozym_18',
103             'pozym_20',
104             'pozym_22',
105             'pozym_24',
106             'pozym_27',
107             'pozym_32',
108             'pozym_35',
109             'pozym_38',
110             'pozym_44',
111             'pozym_48',
112             'pozym_54',
113             'pozym_56',
114             'pozym_63',
115             'pozym_64',
116             'pozym_66',
117             'pozym_67',
118             'pozym_69',
119             'pozym_70',
120             'pozym_74',
121             'pozym_79',
122             'pozym_80',
123             'pozym_85',
124             'pozym_86'
125              ],1)

126 print(list(data))
127 data.shape
128 # ATRAMINIU VEKTORIU METODAS
129 # Klasifikavimas = svm.LinearSVC(C = 1.0, class_weight = None, dual = True,
    fit_intercept = True,
130 #                   intercept_scaling = 1, loss = 'squared_hinge', max_iter = 1000,
131 #                   multi_class = 'ovr', penalty = 'l2', random_state = None, tol = 0.0001,
132 #                   verbose = 0).fit(data, y2)
```

```
133 #Kitaip = SVC(class_weight='balanced', kernel='poly')
134 # Kitaip = Kitaip.fit(data, y2)
135 #Prognoze  = Kitaip.predict(Keisti)
136 #print('Training accuracy Poly:', accuracy_score(y2, Kitaip))
137
138 Hiper = grid_search.GridSearchCV(SVC(class_weight='balanced', kernel='rbf'),
    param_grid={
        "C": [0.1, 1, 5, 10], 'gamma': [0.01, 0.10, 1, 2, 5]} , n_jobs=-1)
139 Hiper = Hiper.fit(data, y2)
140 print('Geriausi SVM parametrai:', Hiper.best_params_)
141 Lygiai = ['Klase_1', 'Klase_2', 'Klase_3', 'Klase_4', 'Klase_5', 'Klase_6', 'Klase_7',
    'Klase_8', 'Klase_9']
142 AVM = SVC(class_weight='balanced', kernel='rbf', C=10, gamma=0.01)
143 # plot_learning_curve(estimator, title, data, y2, cv=cv)
144 # scores = cross_validation.cross_val_score(AVM, data, y2, cv = 10)
145 AVM = AVM.fit(data, y2)
146 Prog_ AVM = AVM.predict(data)
147 print('Training accuracy SVM:', accuracy_score(y2, Prog_ AVM))
148 print(classification_report(y2, Prog_ AVM, target_names=Lygiai))
149 # VALIDAVIMAS
150 Kitai_imciai = AVM.fit(Keisti, ytest)
151 Tikrinimas = Kitai_imciai.predict(Keisti)
152 print('Testavimo tikslumas:', accuracy_score(ytest, Tikrinimas))
153 print(classification_report(ytest, Tikrinimas, target_names=Lygiai))
154 rezultatas1 = confusion_matrix(Tikrinimas, Klases_test)
155 rezultatas2 = confusion_matrix(Prog_ AVM, y2)
156 fig = pl.figure(figsize=(8,8))
157 ax = fig.add_subplot(111)
158 def plot_confusion_matrix(rezultatas1, title='Tyrimo rezultatu klasifikavimo matrica
    (testuojant)', cmap=pl.cm.RdYlBu):
159     Rotated_Plot = ndimage.rotate(rezultatas1, 90)
160     im= pl.imshow(Rotated_Plot, interpolation='nearest', cmap=cmap)
161     Spalva = fig.colorbar(im)
162     pl.title(title)
```

```
163     Spalva.set_label('Kiekis')
164     Pavadinimai1 = ['Klase_9', 'Klase_8', 'Klase_7', 'Klase_6', 'Klase_5', 'Klase_4', 'Klase_3', 'Klase_2', 'Klase_1']
165     Pavadinimai2 = ['Klase_1', 'Klase_2', 'Klase_3', 'Klase_4', 'Klase_5', 'Klase_6', 'Klase_7', 'Klase_8', 'Klase_9']
166     Paimame1 = np.arange(len(Pavadinimai1))
167     Paimame2 = np.arange(len(Pavadinimai2))
168     pl.xticks(Paimame2, Pavadinimai2, rotation=90)
169     pl.grid(False)
170     pl.yticks(Paimame1, Pavadinimai1)
171     pl.tight_layout()
172     pl.ylabel('Prognozuotos klases')
173     pl.xlabel('Tikros klases')
174 rezultatas1 = confusion_matrix(Tikrinimas, Klases_test)
175 plot_confusion_matrix(rezultatas1)
176 m = np.rot90(rezultatas1)
177 for(i,j), z in np.ndenumerate(m):
178     pl.text(j, i, '{:0.0f}'.format(z), ha='center', va='center')
179 pl.show()
180 rezultatas1
181 fig = pl.figure(figsize=(8,8))
182 ax = fig.add_subplot(111)
183 def plot_confusion_matrix(rezultatas2, title='Tyrimo rezultatu klasifikavimo matrica (apmokant)', cmap=pl.cm.RdYlBu):
184     Rotated_Plot = ndimage.rotate(rezultatas2, 90)
185     im= pl.imshow(Rotated_Plot, interpolation='nearest', cmap=cmap)
186     Spalva = fig.colorbar(im)
187     pl.title(title)
188     Spalva.set_label('Kiekis')
189     Pavadinimai1 = ['Klase_9', 'Klase_8', 'Klase_7', 'Klase_6', 'Klase_5', 'Klase_4', 'Klase_3', 'Klase_2', 'Klase_1']
190     Pavadinimai2 = ['Klase_1', 'Klase_2', 'Klase_3', 'Klase_4', 'Klase_5', 'Klase_6', 'Klase_7', 'Klase_8', 'Klase_9']
191     Paimame1 = np.arange(len(Pavadinimai1))
192     Paimame2 = np.arange(len(Pavadinimai2))
```

```
193     pl.xticks(Paimame2, Pavadinimai2, rotation=90)
194     pl.grid(False)
195     pl.yticks(Paimame1, Pavadinimai1)
196     pl.tight_layout()
197     pl.ylabel('Prognozuotos klases')
198     pl.xlabel('Tikros klases')
199 rezultatas1 = confusion_matrix(Prog_ AVM, y2)
200 plot_confusion_matrix(rezultatas2)
201 m = np.rot90(rezultatas2)
202 for(i,j), z in np.ndenumerate(m):
203     pl.text(j, i, '{:0.0f}'.format(z), ha='center', va='center')
204 pl.show()
205 rezultatas2
206 print(start - time.time())
207 from matplotlib import style
208 style.use("ggplot")
209 Pavadinimai3 = ['Klase_1', 'Klase_2', 'Klase_3', 'Klase_4', 'Klase_5', 'Klase_6',
    'Klase_7', 'Klase_8', 'Klase_9']
210 formatter1 = plt.FuncFormatter(lambda i, *args: Pavadinimai3[int(i)])
211 from matplotlib.colors import ListedColormap
212 spalvos1 = ListedColormap(['#ff0000', '#ffff00', '#00ff00', '#0080ff', '#ff0080',
    '#808080', '#ff9999', '#000000', '#00ffbf'])

213 plt.scatter(data["pozym_11"], data["pozym_60"], c = Prog_ AVM, cmap=spalvos1)
214 plt.xlabel("pozym_11")
215 plt.ylabel("pozym_60")
216 plt.title("Klasifikuotu pozymiu grafikas (apmokant)")
217 clb = plt.colorbar(ticks=[0,1,2,3,4,5,6,7,8], format=formatter1)
218 plt.show()
219 from matplotlib import style
220 style.use("ggplot")
221 Pavadinimai4 = ['Klase_1', 'Klase_2', 'Klase_3', 'Klase_4', 'Klase_5', 'Klase_6',
    'Klase_7', 'Klase_8', 'Klase_9']
222 formatter2 = plt.FuncFormatter(lambda i, *args: Pavadinimai4[int(i)])
223 from matplotlib.colors import ListedColormap
```

```
224 spalvos2 = ListedColormap(['#ff0000', '#ffff00', '#00ff00', '#0080ff', '#ff0080',
    '#808080', '#ff9999', '#000000', '#00ffbf'])

225 plt.scatter(Keisti["pozym_11"], Keisti["pozym_60"], c = Tikrinimas, cmap=spalvos2)
226 plt.xlabel("pozym_11")
227 plt.ylabel("pozym_60")
228 plt.title("Klasifikuotu pozymiu grafikas (testuojant)")
229 clb = plt.colorbar(ticks=[0,1,2,3,4,5,6,7,8], format=formatter2)
230 plt.show()
```