VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS

Bakalauro baigiamasis darbas

Klasifikavimo uždavinių sprendimas naudojant „Weka“ ir „R“

(*Solution of Classification Problems Using ‘Weka‘ and ‘R‘*)

Atliko: 4 kurso studentas
Domantas Mincė (parašas)

Darbo vadovas:
Doc., dr. Algirdas Mačiulis (parašas)

Vilnius
2020

Turinys

Įvadas

Klasifikacija yra tam tikrų duomenų aibės elementų klasės numatymo procesas. Klasės kartais vadinamos tikslais, etiketėmis ar kategorijomis. Pavyzdžiui, šlamšto aptikimas elektroniniame pašte gali būti vadinamas klasifikavimo uždaviniu. Tai yra dvejetainė klasifikacija, nes yra tik dvi klasės: šlamštas ir ne šlamštas. Klasifikatorius naudoja tam tikrus mokymo duomenis, kad suprastų, kaip duoti įvesties atributai yra susiję su klase. Tokiu atveju kaip mokymo duomenys turi būti naudojami žinomi šlamštui ir ne šlamštui priskiriami laiškai. Kai klasifikatorius yra tinkamai išmokytas, jis gali būti naudojamas nežinomo elektroninio laiško priskyrimui klasei. Klasifikacija priklauso prižiūrimo mokymosi (angl. Supervised learning) kategorijai. Šioje kategorijoje klasės pateikiamos kartu su įvesties duomenimis. Klasifikavimas yra pritaikomas daugelyje sričių, tokių kaip kredito patvirtinimas, medicininė diagnozė, tikslinė rinkodara. Yra du besimokančiųjų (angl. Learners) tipai: tingūs besimokantieji (angl. Lazy learners) ir nekantrūs besimokantieji (angl. Eager learners) [Asi18]. Tingūs besimokantieji tiesiog laukia, kol pasirodys duomenys, kuriuos reikia klasifikuoti. Kai tai įvyksta, klasifikacija atliekama remiantis labiausiai susijusiais mokymo aibės duomenimis. Palyginus su nekantriais besimokančiaisiais, tingūs besimokantieji mažiau laiko skiria treniravimuisi, tačiau daugiau laiko užima klasifikacija. Prieš imdamiesi duomenų klasifikavimo, nekantrūs besimokantieji sukuria klasifikavimo modelį, pagrįstą pateiktais mokymo duomenimis. Jis turi sugebėti pateikti vieną hipotezę, apimančią visą duomenų aibės erdvę. Dėl modelio konstravimo, nekantrūs besimokantieji ilgai užtrunka treniruodamiesi, bet greičiau numato rezultatus.

Dabar yra daugybė klasifikavimo algoritmų, tačiau neįmanoma nuspręsti, kuris iš jų pranašesnis už kitus. Tai priklauso nuo turimo duomenų rinkinio taikymo ir pobūdžio. Pavyzdžiui, jei klases galima tiesiškai atskirti, tai tiesiniai klasifikatoriai, tokie kaip logistinė regresija, linijinis Fišerio diskriminacija gali pranokti sudėtingesnius modelius ir atvirkščiai [Asi18].

Pagrindinis baigiamojo darbo uždavinys buvo susieti „Weka“ su „R“ programine įranga. Toks poreikis kilo todėl, jog sprendžiant klasifikavimo uždavinius yra kuriami sprendimų medžiai, kurių „Weka“ gražiai atvaizduoti nemoka, o „R“ tam turi labai plačias galimybes. Pagrindinis uždavinys buvo surasti, kaip sujungti abi programas bei kaip pakeisti algoritmo atiduodamą duomenų struktūrą, kad grafiniai redaktoriai, tokie kaip „graphviz” sugebėtų ją apdoroti.

Darbo tikslas – sukurti „R” paketą, kuris sujungtų „Weka” programinės įrangos pateikiamus algoritmus su „R” grafinėmis galimybėmis braižant sprendimų medžius.

1. Sprendimų medžių klasifikatoriai realizuoti „Weka" programoje

Sprendimų medžių pagalba yra sprendžiami klasifikavimo ir regresijos uždaviniai. Pastaruoju metu sprendimų medžiai dar vadinami klasifikavimo ir regresijos medžiais (angl. Classification and regression trees (CART)). Jie sukuria medį tam, kad būtų galima įvertinti duomenų pavyzdį, pradėdami nuo medžio šaknies ir juda lapais, kol bus galima nuspėti rezultatą. Sprendimo medžio kūrimo procesas veikia godžiai išrenkant geriausią skilimo tašką, kad būtų galima nuspėti rezultatą ir procesas yra kartojamas, kol medis tampa fiksuoto gylio. Sukūrus medį jis yra „genimas" siekiant pagerinti modelio gebėjimą apibendrinti naujus duomenis [Bro16]. Sprendimų medžių algoritmų taikymo sritys apima duomenų tyrimą, modelio atpažinimą, pasirinkimo galimybių nustatymą ir kainų nustatymą finansuose, ligų bei rizikos tendencijų nusakymą ir daug kitų sričių [Hus19].

1.1 ADTree

Kintamas sprendimų medis (angl. Alternating decision tree (ADTree)) yra sėkminga klasifikavimo technika, sujungianti sprendimų medžius su numatomu didinimo (angl. Predictive boosting) tikslumu į klasifikavimo taisyklių rinkinį. Originalus indukcijos algoritmo formulavimas atkreipė dėmesį į dvejetainės klasifikacijos problemas. Kintami sprendimų medžiai suteikia mechanizmą, kuris sujungia silpnas hipotezes, sugeneruotas didinimo (angl. Boosting) metu, į vieną aiškią interpretaciją. Tikėdami originaliu įgyvendinimu, mes naudojame nelygybės sąlygas, kurios lygina vieną atributą su konstanta - silpna hipoteze, sugeneruota kiekvienos didinimo iteracijos metu. Kiekvienos didinimo iteracijos t metu algoritmas palaiko dvi aibes: išankstinių sąlygų rinkinį ir taisyklių rinkinį, atitinkamai pažymėtus P_t ir R_t. Tada, kiekvienos didinimo iteracijos metu yra sukuriama silpnų hipotezių aibė C [HPK+02].

1.1.1 Daugiaklasiai ADTree

Išplečiant bet kurį algoritmą nuo dvejetainės prie daugiaklasės klasifikacijos, yra dvi galimybės. Paprasčiausias būdas yra daugiaklases problemas paversti keliomis dvejetainės klasifikacijos problemomis. Šis bendras požiūris gali būti pritaikytas bet kuriam klasifikavimo algoritmui, todėl gaunamas balsavimo modelių (angl. Voting model) rinkinys. Paprastai toks požiūris lemia daugybės modelių sukūrimą. Kita galimybė – bandyti sukurti vieną medį, galintį tiesiogiai numatyti kiekvieną klasę [HPK+02].

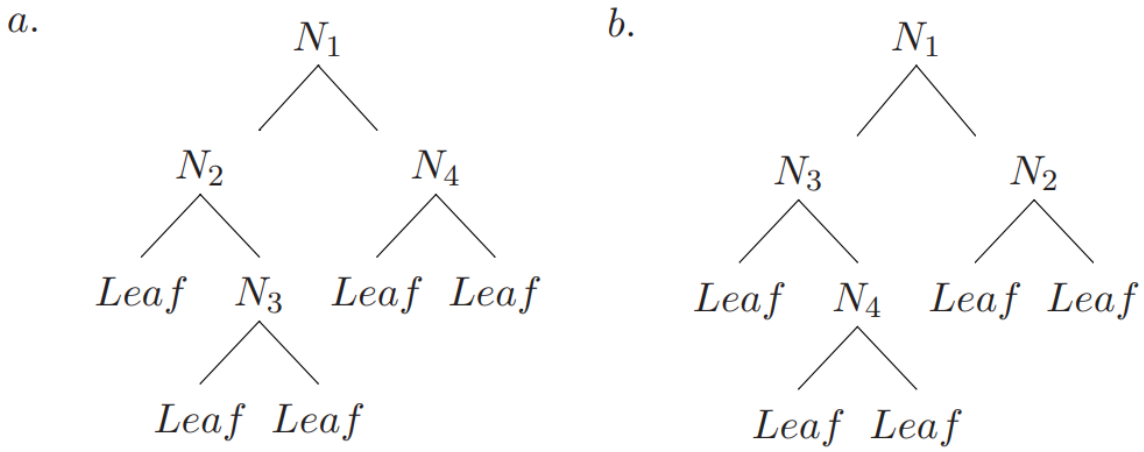
Kelių „ADTree" klasių reikšmių transformavimas į dvi klases gali būti pasiektas keliais būdais. Kadangi „ADtrees" gali būti sujungti, gautas daugiaklasis modelis gali būti vienas medis, gautas iš dviejų klasių balsavimo medžių rinkinio. Standartinis metodas yra klasių reikšmių pogrupio traktavimas kaip A klasės, o likusių reikšmių rinkinys – B klasės. Tokiu atveju problema yra sumažinama iki dviejų klasių, iš kurių galima sudaryti modelį. Tada tai yra pakartojama skirtingiems pogrupiams ir modeliai balsuoja dėl juos atstovaujančių klasių reikšmių [HPK+02].

1.2 BFTree

Sprendimų medžiai yra potencialiai galingi numatytojai, kurie aiškiai parodo duomenų rinkinio struktūrą. Standartinių sprendimų medžių besimokantieji pradžioje išplečia šakas pagal eiliškumą, tuo tarpu besimokantiesiems pagal geriausia sprendimą (angl. Best-first decision), medyje pirmiausia išplečiama „geriausia" šaka. „Geriausia" šaka yra tokia, kurios padalijimas lemia maksimalų klaidų sumažėjimą tarp visų šakų, kurias galima suskaidyti. Gautas medis bus tas pats, kai bus visiškai užaugęs, skiriasi tik tvarka, kuria jis sudarytas. Žiūrint praktiškai, kai kurios medžio šakos iš tikrųjų neatspindi pagrindinės srities informacijos. Ši problema vadinama

perpildymu (angl. Overfitting) ir ją sukelia triukšmingi duomenys. Genėjimas yra būtinas, kad treniruočių duomenys nebūtų perpildyti. Taip pat, reikia išmesti tas duomenų dalis, kurios nėra informatyvios ir nepadės rezultatų nuspėjime. „Geriausias pirmas“ algoritmo šakos išplėtimas leidžia ištirti naujus genėjimo būdus nustatant atliktų išplėtimų skaičių remiantis kryžminiu tikrinimu (angl. Cross-validation) [Shi07].

„Geriausio pirmo“ sprendimo medžiai yra sukonstruoti „skaldyk ir valdyk“ principu, panašiai kaip standartiniai medžiai, kuriuose ieškoma giliausia šaka. Pagrindinė mintis, kaip sudaryti pirmąjį medį yra tokia: pirmiausia pasirenkame atributą, kurį norime patalpinti šaknyje ir, remiantis tam tikrais kriterijais, padarome keletą šio atributo šakų. Tada padaliname mokymo duomenis į pogrupius, po vieną kiekvienai šakai, besitęsiančiai nuo šaknies. Šis žingsnis kartojamas pasirinktai šakai, naudojant tik tuos duomenų aibės pavyzdžius, kurie ją iš tikrųjų pasiekia. Kiekviename žingsnyje mes pasirenkame „geriausią“ poaibį iš visų pogrupių, kuriuos galima išplėsti. Šis procesas tęsiasi tol, kol visos šakos yra grynos arba yra pasiektas tam tikras išplėtimų skaičius. Svarbu prisiminti, kad „geriausias pirmas“ medžiams viršūnių numeravimas gali būti skirtingas, kol „giliausias pirmas“ visada turi tokį patį [Shi07].



Pav 1. Sprendimų medžiai: (a) hipotetinis „pirmas giliausias" sprendimo medis, (b) hipotetinis „geriausias pirmas" sprendimo medis.

1.3 Iterative Dichotomiser 3 (ID3)

„Iterative Dichotomiser 3” algoritmas yra naudojamas sprendimų medžiams kurti. Jo autorius yra Johnas Rossas Quinlanas. ID3 sprendimų medžiai yra naudojami klasifikavimui, o algoritmo tikslas – sukurti kuo „negilesnius“ sprendimų medžius.

Pradedant nuo medžio šaknies, ID3 algoritmas sukuria sprendimų medį po vieną vidinę šaką. Tuo pačiu metu kiekvienoje šakoje pasirenkamas informatyviausias atributas, jei duomenis skirstytume į pogrupius pagal atributo vertes. Kaip nustatoma, kuris atributas yra informatyviausias? Tai nustatoma pagal entropijos mažinimo idėją, kuri yra Šenono informacijos teorijos dalis. Entropija yra sistemos netvarkos matas. Duomenų rinkinys, kuriame yra daug netvarkos ar neapibrėžtumo, nesuteikia daug informacijos. „Iterative Dichotomiser 3” veikimas: pirmas žingsnis yra apskaičiuoti pirminę duomenų rinkinio entropiją. Antra, reikia apskaičiuoti kiekvieno atributo informatyvumą. Toliau pasirenkame atributą, kurio informatyvumas yra didžiausias. Sekantis veiksmas yra skirstyti duomenų rinkinį pagal atributo, kuris turėjo didžiausia informatyvumą, reikšmes. Tada duomenų rinkinys yra padalijamas remiantis vertėmis, kurias gali užimti informatyviausias atributas. Pertvaros tampa išeinančiomis šakomis iš šaknies viršūnės. Šios šakos baigiasi nepaženklintomis viršūnėmis. Iš šių skaidinių pašalinamas informatyviausias atributas ir jis nebenaudojamas tolimesniame naujų šakų skaidyme. Atributas gali būti naudojamas

kaip sprendimo kintamasis tik vieną kartą tam tikroje medžio dalyje, bet gali būti naudojamas kelis kartus visame medyje. Paskutinis reikalavimas – kartoti visus aukščiau paminėtus veiksmus kiekvienai medžio šakai naudojant atitinkamą skaidinį. Algoritmas baigia savo darbą kai kiekvienas skaidinio duomenų rinkinys yra tos pačios klasės dalis, nebeliko daugiau atributų arba skirstinyje nėra nei vieno duomenų pavyzdžio [Add19].

1.4 C4.5 arba J48

C4.5 arba J48 yra Johno Rosso Quinlano sukurtas algoritmas skirtas kurti sprendimų medžius, kurie yra naudojami klasifikavimo problemoms spręsti. Tai patobulina ir praplečia aukščiau minėtą ID3 algoritmą, naudojant tiek nuolatinius, tiek diskrečius požymius, trūkstamas reikšmes ir medžių genėjimą po sukūrimo. Komercinis jo įpėdinis yra C5.0 / See5, kuris veikia daug greičiau nei C4.5, atmintis naudojama efektyviau ir kuriami sprendimų medžiai yra mažesni. Kadangi algoritmas yra prižiūrimo mokymosi, jam yra reikalingas mokymo pavyzdžių rinkinys. Kiekvienas pavyzdys gali būti vertinamas kaip pora: įvesties objektas ir norima išvesties vertė - klasė. Algoritmas analizuoja mokymo rinkinį ir sukuria klasifikatorių, kuris turi sugebėti teisingai klasifikuoti mokymo ir testų pavyzdžius. Bandomasis pavyzdys yra įvesties objektas, o algoritmas turi numatyti išvesties vertę, kitaip sakant, pavyzdys turi būti priskirtas klasei. Klasifikatorius, kurį naudoja C4.5 yra sprendimų medis, o šis medis kuriamas nuo šaknies iki lapų, atsižvelgiant į Occamo skustuvą (angl. Occam‘s razor). Šis skustuvas sako, kad atsižvelgiant į du teisingus tam tikros problemos sprendimus, turėtume pasirinkti paprastesnį sprendimą.

Medžių genėjimas atliekamas siekiant gauti mažesnius medžius ir išvengti perpildymo (angl. Overfitting). Yra dvi genėjimo rūšys: išankstinis genėjimas ir genėjimas po medžio sukūrimo. Jei pasirenkamas išankstinis genėjimas, medžio sudarymas yra sustabdomas, kai atributai yra nebesvarbūs (neinformatyvūs). Tai atlikti yra sunku, bet veikia greičiau nei kita genėjimo rūšis. Atliekant genėjimą po medžio sukūrimo mes pašaliname tik tam tikras šakas. Tai galime padaryti pasitelkdami sub-medžių auginimą.

C4.5 pranašumai: sukuria modelius, kuriuos galima lengvai suprasti, lengva įgyvendinti, gali naudoti tiek kategorines, tiek tęstines reikšmes, kovoja su triukšmingais duomenimis. Algoritmas taip pat turi ir trūkumų: dėl nedidelių duomenų svyravimų medžiai gali skirtis, ypač kai kintamųjų reikšmės yra arti viena kitos. Prastai veikia, kai yra mažas treniruočių duomenų rinkinys [Oct11].

1.5 M5P

M5P yra Johno Rosso Quinlano M5 algoritmo, skirto indukuoti regresijos modelių medžius, rekonstrukcija. M5P derina įprastą sprendimų medį su tiesinės regresijos funkcijų galimybėmis. Pirma, medžiui sukurti naudojamas sprendimų medžio indukcijos algoritmas, tačiau vietoj to, kad viršūnėje būtų maksimaliai padidintas informacijos prieaugis, naudojamas padalijimo kriterijus, kuris iki minimumo sumažina kiekvienos klasės verčių pokyčius poaibiuose. Padalijimo procedūra sustabdoma, jei visų duomenų aibės įrašų, pasiekusių viršūnę, klasės reikšmės labai nedaug skiriasi arba išlieka tik keli duomenų aibės pavyzdžiai. Antra, medis genimas aukštyn iš kiekvieno lapo. Atliekant genėjimą vidinė viršūnė paverčiama lapu su regresine plokštuma. Trečia, siekiant išvengti nelygumų tarp sub-medžių, taikoma lyginimo procedūra, kuri sujungia lapo numatymą su kiekviena viršūne pakeliui į šaknį, išlyginanti jį kiekvienoje viršūnėje. Lyginimas vyksta derinant modelį su verte, kurią numatė tiesinis modelis.

Breiman sukurta klasifikavimo ir regresijos medžių (angl. Classification and regression trees (CART)) metodika pritaikyta išvardintiems atributams ir trūkstamoms vertėms. Visi išvardinti atributai paverčiami dvejetainiais kintamaisiais, kad visos M5P skiltys būtų dvejetainės. Dėl trūkstamų verčių M5P naudoja metodą, vadinamą „surogatiniu padalijimu“, kuris randa kitą atributą, kurį reikia padalyti ir naudoti vietoje originalaus. Mokymo metu M5P, kaip surogatinį atributą, naudoja klasės vertę tikėdamas, kad tai yra atributas, geriausiai koreliuojantis su padalijimui naudojamu atributu. Pasibaigus padalijimo procedūrai, visos trūkstamos vertės

pakeičiamos vidutinėmis treniruočių duomenų reikšmėmis. Testuojant nežinoma atributo reikšmė pakeičiama vidutine to atributo verte visose mokymo instancijose, pasiekiančiose viršūnę, todėl visada pasirenkamas didžiausias pogrupis. M5P generuoja kompaktiškus ir palyginti suprantamus modelius [Kra].

1.6 RandomForest

„RandomForest“ yra populiariausias klasifikavimo algoritmas. Jis sprendžia tiek klasifikavimo, tiek regresijos uždavinius. Šis algoritmas yra prižiūrimo mokymosi ir, kaip rodo pavadinimas, sukuria mišką su daug medžių. Apskritai, kuo daugiau medžių miške, tuo tvirtesnis miškas atrodo. Lygiai taip pat „RandomForest“ klasifikatoriuje – kuo didesnis medžių skaičius miške, tuo tikslesni rezultatai yra gaunami [Pol17].

Šio algoritmo pranašumai – kaip jau minėjau, jis gali būti naudojamas tiek klasifikacijai, tiek regresijos užduotims. Taip pat, mums nereikia rūpintis trūkstamomis reikšmėmis – už mus jomis pasirūpina klasifikatorius. Kai miške turime daug medžių, „RandomForest“ klasifikatorius neleis perkrauti (angl. Overfit) modelio. Šis algoritmas yra naudojamas banko operacijose, farmacijoje, akcijų biržose bei sparčiai populiarėjančioje elektroninėje komercijoje (spręsti pirkėjo krepšelio uždavinius) [Pol17].

„RandomForest“ turi beveik tokius pačius parametrus, kaip sprendimų medis ar pakavimo (angl. Bagging) klasifikatorius. Laimei, nereikia derinti sprendimų medžio su pakavimo klasifikatoriumi, nes galima lengvai naudoti „RandomForest“ klasifikatoriaus klasę. „RandomForest“ prideda papildomo atsitiktinumo modeliui auginant medžius. Užuot ieškojęs svarbiausio atributo, algoritmas skaido duomenis ir ieško geriausių savybių tarp atsitiktinių funkcijų pogrupio. Tai lemia didelę įvairovę, kuri paprastai lemia geresnį modelį. Todėl, „RandomForest“ algoritmas atsižvelgia tik į atsitiktinį funkcijų pogrupį. Įmanoma patiems padaryti medžius dar labiau atsitiktiniais papildomai naudojant kiekvienos funkcijos atsitiktinius slenksčius, o ne ieškant geriausių įmanomų slenksčių, kaip tai daro įprastas sprendimų medis [Don19].

1.7 RandomTree

„RandomTree“ yra prižiūrimas klasifikatorius. Tai yra mokymosi algoritmas, sukuriantis daugybę besimokančiųjų. Jame įgyvendinama idėja supakuoti (angl. Bag) atsitiktinį duomenų rinkinį sprendimų medžiui sudaryti. Standartiniame medyje kiekviena viršūnė yra padalijama naudojant geriausią padalijimą tarp visų kintamųjų. „RandomTree“ kiekviena viršūnė padalijama naudojant geriausią tarp tos pačios viršūnės atsitiktinai parinktų prediktorių segmentą.

Atsitiktinius medžius pristatė Leo Breimanas ir Adele Cutler. Algoritmas gali spręsti tiek klasifikavimo, tiek regresijos problemas. Atsitiktiniai medžiai – tai prediktorių rinkinys, vadinamas medžiu. Klasifikacija veikia taip: atsitiktinių medžių klasifikatorius paima įvesties duomenų aibę, klasifikuoja ją pagal kiekvieną miško medį ir išveda klasę, kuri gavo daugiausiai „balsų“. Regresijos atveju klasifikatoriaus rezultatas yra visų miško medžių atsakymų vidurkis.

„RandomTree” iš esmės yra dviejų esamų mašinų mokymosi algoritmų derinys: vieno modelio medžiai derinami su „RandomForest“ idėjomis. „RandomForest” parodė, kad galima žymiai pagerinti sprendimų medžių našumą. Pirmiausia treniruočių duomenys gaunami imant kiekvieno medžio pakeitimą. Antra, auginant medį, užuot visada apskaičiavus geriausią įmanomą kiekvieno atributo padalijimą, kiekvienoje šakoje atsižvelgiama tik į atsitiktinį visų atributų pogrupį ir apskaičiuojamas geriausias to pogrupio padalijimas. Tokie medžiai pirmą kartą buvo klasifikuojami pagal atsitiktinių modelių medžius, kuriuose yra pavyzdiniai medžiai. „RandomTree” naudoja šia produkciją dalijamajai atrankai ir tokiu būdu skatina pagrįstai subalansuotus medžius, kai vienas bendras šaknies nustatymas veikia visus lapus ir taip supaprastinamas optimizavimo procesas [WFH11] [Epp] [Pfa11] [Wis13].

1.8 REPTree

„REPTree“ naudoja regresijos medžio logiką ir sukuria kelis medžius skirtingomis iteracijomis. Po to iš visų sugeneruotų medžių išrenkamas geriausias. Tai bus laikoma „atstovu“. Medžio genėjimo metu naudojamas matas yra medžio padarytų prognozių vidutinė kvadratinė paklaida. Išgenėtas medis (angl. Basically reduced error pruning tree (REPT)) yra greitas sprendimų medžio mokymasis ir jis sukuria medį, pagrįstą gauta informacija arba sumažinta dispersija. „REPTree“ algoritmas yra greito apsisprendimo medis, kuris sukuria: sprendimų arba regresijos medį naudodamas atributų informatyvumą, kaip padalijimo kriterijų ir išlygina jį naudodamas klaidų genėjimą. Skaitinių atributų reikšmės yra rūšiuojamos tik vieną kartą. Trūkstamos vertės yra nagrinėjamos naudojant C4.5 metodą, naudojant trupmeninius egzempliorius. „REPTree“ algoritmo pavyzdys yra taikomas „UCI“ saugykloje – yra sukuriama neatitikimų matrica (angl. Confusion matrix) [WFH11] [SM14] [DPR+14].

2 „Weka” klasifikavimo rezultatų duomenų struktūra

Natūralu, kad kiekvieną kartą sprendžiant klasifikavimo uždavinius yra tikimasi suprantamo rezultato. Šiame skyriuje aptarsime visus iš „Weka” po klasifikavimo gautus duomenis ir jų reikšmes. Visame skyriuje rezultatai iliustruojami naudojant Fišerio irisų duomenis. „Iris“ gėlių duomenų rinkinys yra daugiamatis duomenų rinkinys, kurį 1936 m. pristatė britų statistikas ir biologas Ronaldas Fisheris [FIS36]. Jis kartais vadinamas Andersono „Iris” duomenų rinkiniu, nes Edgaras Andersonas rinko duomenis, kad nustatytų trijų susijusių rūšių irisų žiedų morfologinius pokyčius [AND36].

2.1 Algoritmo paleidimo informacija

Tai yra pats pirmas gautos informacijos skyrius, kuriame galime sužinoti komandą, kurią paleidus buvo gauti šie rezultatai (angl. Scheme), duomenis, kurie buvo naudoti (angl. Relation), įrašų (angl. Instances) kiekį, kurie buvo klasifikuoti bei atributų (angl. Attributes) skaičių ir pavadinimus (žr. pav 2).

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===
```

Pav 2. pirmasis programos paleidimo rezultatų blokas gautas „Weka"

Taip pat, šiame bloke dar matomas pasirinktas testo variantas, šiuo atveju tai yra kryžminis tikrinimas (angl. Test mode).

2.2 Gautasis medis

Sekanti iš „Weka" gautoji duomenų dalis yra medis tekstiniame formate (žr. pav. 3). Iš to, jau be grafinio vaizdo, galima skaityti, koks medis buvo sudarytas: jei žiedlapio plotis (angl. Petal Width) yra mažiau arba lygu 0.6, iškart žinome, kad augalo rūšis yra *„iris-setosa"*. Taip pat kairėje yra matomi skaičiai (pvz. 48.0/1.0), jie reiškia, kiek įrašų buvo klasifikuota teisingai ir neteisingai (atitinkamai kairė ir dešinė pasvirojo brūkšnio pusės). Dar matome, kiek buvo gauta lapų bei suskaičiuotas visas medžio dydis (vizualizuojant lapai žymimi kvadratais, o viršūnės apskritimais) bei kiek laiko užtruko sukurti klasifikavimo modelį.

```
J48 pruned tree
------------------

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9


Time taken to build model: 0.03 seconds
```

Pav 3. „Weka" grąžintas J48 algoritmo sudarytas medis „iris" duomenims

2.3 Klasifikavimo kokybės charakteristikos

Šiame bloke „Weka" mums primena kiek sumoje buvo įrašų, kurie buvo klasifikuoti (angl. Total Number of Instances), skaičių, kiek įrašų buvo suklasifikuoti teisingai, t.y įrašas priskirtas teisingai klasei (angl. Correctly Classified Instances). Taip pat pastebime, kad teisingai bei neteisingai klasifikuotus (angl. Incorrectly Classified Instances) įrašus pateikia tiek skaičiumi, tiek procentais. Toliau galime matyti „Kappa" statistiką (angl. Kappa statistic). „Kappa" statistika dažnai naudojama patikrinti vertintojo patikimumą (angl. Interrater reliability). Vertintojo patikimumo svarba yra ta, kad jis parodo, kiek tyrime surinkti duomenys teisingai atspindi išmatuotus kintamuosius [McH19]. Kitaip tariant, ši statistika parodo, kiek klasifikavimo modelis yra tikslesnis už atsitiktinį klasifikavimą (kuo arčiau 1 – tuo modelis tikslesnis). „Kappa" statistika apskaičiuojamą pagal formulę matomą apačioje, kur p_o teisingai klasifikuotų įrašų dalis, o p_e - hipotetinė sutapimo tikimybė [Coh].

$$k = \frac{p_o - p_e}{1 - p_e}$$

Toliau galime matyti vidutinę absoliučiąją paklaidą (angl. Mean absolute error) įvertį. Vidutinė absoliučioji paklaida matuoja vidutinį klaidų dydį spėjimų rinkinyje, neatsižvelgdama į jų kryptį. Tai yra absoliučių skirtumų tarp numatymo ir faktinio stebėjimo vidurkis, lyginant su bandiniu, kai visi individualūs skirtumai turi vienodą svorį [MAE]. Šis įvertis apskaičiuojamas pagal formulę:

$$MAE = \frac{|p_1 - a_1| + ... + |p_n - a_n|}{n}$$

kur p yra spėjimo reikšmė, a – tikroji reikšmė, o n – įrašų skaičius [WF11] 178 puslapis.

Kitas atributas yra šaknies vidutinė kvadratinė paklaida (angl. Root mean squared error). Vidutinė kvadratinė paklaida - standartinis likučių nuokrypis (numatymo klaidos). Likučiai yra atstumo nuo regresijos linijos taškų matas. Taigi, vidutinė kvadratinė paklaida yra šių liekanų pasiskirstymo matas. Kitaip tariant, jis parodo, kaip koncentruoti duomenys yra pagal jų tinkamumą [Bar92]. Šio parametro formulė yra:

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{n}}$$

kur p yra spėjimo reikšmė, a – tikroji reikšmė, o n – įrašų skaičius [WF11] 178 puslapis.

Tolesnis matas yra santykinė paklaida (angl. Relative absolute error). Trumpai tariant, ši metrika parodo koks yra matavimo santykis su matuojamo daikto dydžiu. Ji apskaičiuojama pagal formulę:

$$RAE = \frac{|p_1 - a_1| + ... + |p_n - a_n|}{|a_1 - \bar{a}| + ... + |a_n - \bar{a}|}$$

kur p yra spėjimo reikšmė, a – tikroji reikšmė, o ā – tikrųjų reikšmių vidurkis [WF11] 178 puslapis.

Santykinė absoliuti paklaida (angl. Root relative squared error) - yra santykinis dydis, kuris būtų gautas, jei būtų naudojamas paprastas klasifikavimas. Ji randama pagal formulę:

$$RRSE = \sqrt{\frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + ... + (a_n - \bar{a})^2}}$$

kur p yra spėjimo reikšmė, a – tikroji reikšmė, o ā – tikrųjų reikšmių vidurkis [WF11] 178 puslapis, (žr. pav. 4).

```
=== Summary ===

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150
```

Pav 4. J48 algoritmo veikimo santrauka „iris" duomenims

2.4 Išsamus tikslumas pagal klases

Šiame rezultatų bloke galime matyti, koks yra klasifikacijos tikslumas kiekvienai klasei. Klasių skaičius yra kintantis. Kadangi mes naudojame „iris" duomenis, tai klasės yra trys. Toliau nagrinėkime kiekvieną stulpelį: pirmasis – procentas įrašų, kurie buvo teisingai priskirti teigiamai klasei (angl. True Positive) [Eva].

Kitas stulpelis – procentas įrašų, kurie buvo klaidingai priskirti teigiamai klasei (angl. False Positive) [Eva].

Sekančiame stulpelyje galime matyti tikslumą arba teigiamą prognozinę vertę (angl. Precision). Požymis yra teigiamų ir neigiamų rezultatų proporcija. Ji yra gaunama iš dviejų praeitų matų pagal formulę:

$$P = TP/(TP+FP)$$

kur TP yra procentas įrašų teisingai priskirtų teigiamai klasei, o FP - yra procentas įrašų klaidingai priskirtų teigiamai klasei [Eva].

Sekantis įvertis vadinamas teisingai teigiamų įverčiu rodiklis (angl. Recall or True Positive Rate). Požymis nurodo, kiek teigiamų įverčių buvo priskirta teisingai klasei. Jis apskaičiuojamas pagal formulę:

$$TPR = \frac{TP}{TP + FN}$$

kur TP - yra procentas įrašų teisingai priskirtų teigiamai klasei, o FN – procentas neteisingai neigiamai klasei priskirtų įrašų [Eva].

F matas (angl. F-Measure), dar vadinamas F1 balu arba F balu, yra testavimo tikslumo matas ir yra apibrėžiamas, kaip testo tikslumo (angl. Precision) ir teisingai teigiamų įverčių rodiklio (angl. Recall) svertinis vidurkis [Mea] [Sco]:

$$F1 = 2\,x\,\frac{precission\,x\,recall}{precission + recall}$$

„Matthews" koreliacijos koeficientas (MCC) naudojamas kaip dvejetainių klasifikacijų kokybės matas. Jis apskaičiuojamas pagal formulę:

$$MCC = \frac{TP\,x\,TN - FP\,x\,FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

kur TP yra procentas įrašų teisingai priskirtų teigiamai klasei, FP - yra procentas įrašų klaidingai priskirtų teigiamai klasei, FN – procentas neteisingai neigiamai klasei priskirtų įrašų, o FP - procentas neteisingai teigiamai klasei priskirtų įrašų [Mat].

AUC - ROC kreivė yra klasifikavimo problemos našumo matavimas esant įvairiems slenksčių parametrams. ROC yra tikimybės kreivė, o AUC žymi atskyrimo laipsnį ar matą. Ji nurodo, kiek modelis yra pajėgus atskirti klases [Nar18].

PRC Area (angl. Precision-recall curves) kreivės apibendrina kompromisą tarp teisingai teigiamų įverčiu rodiklio ir teigiamos numatomosios vertės numatomajam modeliui, naudojant skirtingas tikimybės ribas. [Bro18]. (žr. pav 5).

```
=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.980	0.000	1.000	0.980	0.990	0.985	0.990	0.987	Iris-setosa
	0.940	0.030	0.940	0.940	0.940	0.910	0.952	0.880	Iris-versicolor
	0.960	0.030	0.941	0.960	0.950	0.925	0.961	0.905	Iris-virginica
Weighted Avg.	0.960	0.020	0.960	0.960	0.960	0.940	0.968	0.924	

Pav 5. „iris" duomenų klasės tikslumo matai po J48 klasifikavimo

2.5 Nesutapimų matrica

Paskutiniame rezultatų bloke matome nesutapimų matricą (angl. Confusion matrix). Nesutapimų matrica yra lentelė, kuri dažnai naudojama apibūdinti klasifikavimo modelio (arba klasifikatoriaus) veikimą tiriamųjų duomenų rinkinyje, kurio tikrosios vertės yra žinomos. Tai leidžia vizualizuoti algoritmo atlikimą. Taip pat, tai leidžia lengvai nustatyti nesutapimus tarp klasių, pvz. viena klasė paprastai klaidingai paženklinama kaip kita. Dauguma algoritmo rodiklių apskaičiuojami iš nesutapimų matricos. (žr. pav 6).

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Pav 6. J48 algoritmo nesutapimų matrica „iris" duomenims.

3 „Weka" ir „RWeka" pristatymas

Dvi iš atvirojo kodo aplinkų, pasiekiamų kompiuteriniam / statistiniam mokymuisi duomenų kasimo (angl. Data mining) ir žinių atradimo (angl. Knowledge discovery) srityse, yra programinės įrangos paketai „Weka" ir „R". Jie atsirado mašininio mokymosi ir statistikų bendruomenėse. Kad vienoje sistemoje būtų pasiekiami skirtingi abiejų paketų įrankiai, siūlomas „RWeka" paketas, kuris sujungia „Weka" funkcionalumą su „R". Naudojant tik ploną (daugiausia „R") kodo sluoksnį, pateikiamos bendrosios sąsajos rinkinys, kuris gali sukurti „Weka" sąsajas su „R" grafinėmis galimybėmis [HBZ07] [Tea07].

3.1 „Weka"

Įvairi patirtis rodo, kad neegzistuoja tokia mašininio mokymosi schema, kuri išspręstų visus duomenų gavybos uždavinius. Universalus besimokantysis yra tik idealistinė fantazija, o duomenų gavyba yra eksperimentinis mokslas [WF11] puslapiai 365-367.

„Weka" darbastalis yra moderniausių mašininio mokymosi algoritmų ir duomenų išankstinio apdorojimo įrankių rinkinys. Programinė įranga sukurta taip, kad būtų galima greitai bei lanksčiai

išbandyti egzistuojančius metodus naujai pateiktiems duomenims. Ji suteikia didžiulę pagalbą visam eksperimentinių duomenų gavybos procesui, įskaitant pradinių įvesties duomenų ruošimą, statistinį mokymosi schemų vertinimą ir įvesties duomenų bei mokymosi rezultato vizualizavimą. „Weka" nėra tik gausus mokymosi algoritmų pasirinkimas, bet ir platus išankstinio apdorojimo įrankių asortimentas. Į šį įvairiapusį ir išsamų priemonių rinkinį galima patekti naudojantis bendra grafine sąsaja, kad jos vartotojai galėtų palyginti įvairius metodus ir nustatyti, kurie labiausiai tinka nagrinėjamai problemai spręsti [WF11] puslapiai 365-367.

„Weka" buvo sukurta „Waikato" universitete Naujojoje Zelandijoje, o „Weka" vardas reiškia „Waikato" aplinka žinių analizei (angl. Waikato Environment for Knowledge Analysis). Už universiteto ribų „Weka" yra tariama rimu su Meka (angl. Mecca). Meka yra neskraidantis ir labai smalsios prigimties paukštis, kuris yra randamas tik Naujosios Zelandijos salose [WF11] puslapiai 365-367.

„Weka" yra parašyta „Java" kalba ir yra platinama pagal „GNU" viešosios licencijos sąlygas. Sistema veikia beveik bet kurioje platformoje ir buvo išbandyta naudojant „Linux", „Windows" ir „Macintosh" operacines sistemas. Tai suteikia vienodą sąsają su daugeliu skirtingų mokymosi algoritmų, taip pat metodais, taikomais duomenų apdorojimui prieš ir po mokymosi bei mokymo schemų rezultatams įvertinti pagal bet kurį duomenų rinkinį [WF11] puslapiai 365-367.

„Weka" teikia mokymosi algoritmų, kuriuos galima lengvai pritaikyti duomenų rinkinyje, įgyvendinimus. Tai taip pat apima įvairias duomenų rinkinių pertvarkymo priemones, tokias kaip diskretizavimo algoritmai. Galima iš anksto apdoroti duomenų rinkinius, įtraukti juos į mokymosi schemą ir išanalizuoti gautą klasifikatorių bei jo veikimą – visa tai daroma neparašant nei vienos kodo eilutės. Darbastalį sudaro visų standartinių duomenų gavybos problemų metodai: regresija, klasifikavimas, grupavimas, asociacijos taisyklių pasirinkimas ir atributų parinkimas. Susipažinimas su duomenimis yra neatsiejama darbo dalis, todėl „Weka" suteikia daugybę duomenų pirminio apdorojimo priemonių. Visų algoritmų paimami duomenys turi būti pateikiami kaip viena santykinė lentelė ARFF formatu, kurią galima nuskaityti iš failo arba sugeneruoti iš duomenų bazės užklausos [WF11] puslapiai 365-367.

Vienas iš „Weka" naudojimo būdų yra pritaikyti mokymosi metodą duomenų rinkiniui, analizuoti gautus rezultatus ir taip daugiau išmokti apie turimus duomenis. Kitas būdas yra naudoti apmokytą modelį, kad būtų sukurtos prognozės naujiems duomenų įrašams. Kitaip tariant, dalis duomenų yra panaudojama modeliui apmokyti ir tada modelis yra panaudojamas nepanaudotai daliai duomenų rinkinio įrašų. Trečias būdas, kaip efektyviai naudoti „Weka", yra kelių skirtingų besimokančiųjų priskyrimas ir jų rezultatų palyginimas, taip nustatant efektyviausią numatytoją [WF11] puslapiai 365-367.

Mokymosi metodai yra vadinami klasifikatoriais, o interaktyviojoje „Weka" sąsajoje galima pasirinkti norimą klasifikatorių naudojantis meniu. Dauguma klasifikatorių turi modifikuojamus parametrus [WF11] puslapiai 365-367.

Faktinis mokymosi schemų įgyvendinimas yra pats vertingiausias „Weka" resursas. Duomenų pirminio apdorojimo įrankiai, vadinami filtrais, suveikia per sekundės dalį. Kaip ir klasifikatorius, iš meniu galima pasirinkti filtrus ir pritaikyti juos savo poreikiams [Wek].

3.2 „RWeka" paketas

„RWeka" paketas, kaip sako pavadinimas, yra „R" sąsaja su „Weka". Šis paketas yra sukurtas naudojantis „rJava" paketu, skirtu žemo lygio tiesioginiams „R" / „Java" ryšiams, kad būtų pasiektas pagrindinis „Weka" funkcionalumas. Kadangi, kaip jau minėta aukščiau, „Weka" besimokantiesiems teikia „pagrindines" klases ir nuoseklią metodų sąsają, šioms besimokančiųjų klasėms galima pateikti bendrųjų sąsajų generatorius, kurie pakartotinai naudoja „Weka" metodus. Šis požiūris leidžia pasiekti tiek apibendrinamumą, nes naujos sąsajos gali būti generuojamos bet kuriuo metu, tiek palaikomumą (nes pakartotinai naudojamos tik „eksportuotos" „Weka" funkcijos) [HBZ07] [Tea07].

Lentelė 1: „R" / "RWeka“ funkcijų / metodų ir jų atitikmenų „Wekoje“ apžvalga [HBZ07].

	R/**RWeka**	**Weka**
classifier interface	`make_Weka_classifier()`	**Weka** class (in JNI notation)
	$f_{I,W}$	`buildClassifier()`
	`print()`	`globalInfo()`, `technicalInformation()`
	`WOW()`	`listOptions()`
fitted classifier	`print()`	`toString()`
	`fitted()`	`classifyInstance()`
	`predict()`	`classifyInstance()`, `distributionForInstance()`
	`summary()`	'Evaluation' class
	`plot()` ('Weka_tree')	–
	`write_to_dot()`	`graph()`

Sąsajos generatorius „make_Weka_classifier” sukuria sąsajos funkciją tam tikrai „Weka” klasei (pvz. „weka.classifiers.trees.REPTree”), kurios visiškai identiškas pavadinimas yra nurodytas „Weka” notacijoje [Wek]. Sąsajos funkcija iš tikrųjų yra paveldima iš sąsajos klasės „R_Weka_classifier_interface”. Kai algoritmas panaudojamas duomenų rinkiniams, jis grąžina objektus, paveldėtus iš nurodytų klasių, kuriuos klasifikatoriaus sąsajos funkcijos visada paveldi. Visos klasifikatoriaus sąsajos funkcijos turi įprastą formulę, duomenis ir poaibį, taip pat oficialią kontrolę, kad būtų nurodyti reikiami valdymo argumentai, kurie privalo būti perduoti „Weka”. Spausdinant tokias sąsajos funkcijas yra naudojami „Weka” *globalInfo()* ir *TechnicalInformation()* metodai, kad būtų pateiktas sąsajos funkcionalumas [HBZ07] [Tea07]. Kai yra iškviečiama klasifikatoriaus sąsajos funkcija, „R” yra sukuriamas modelio rėmas (angl. Model frame), kuris perduodamas „Weka” objektui. Tada šiems objektams yra iškviečiamas „Weka” metodas *buildClassifier()*. Tinkamos vertės (modelio prognozės treniruočių duomenims) yra gaunamos iškvietus „Weka” klasifikavimo metodą *classifyInstances()*, skirtą klasifikatoriui ir kiekvienam treniravimo duomenų įrašui [HBZ07] [Tea07].

Be klasifikatorių, „RWeka” teikia sąsajų generavimo įrankius klasterizatoriams, filtrams bei asociatoriams. Kai kurių svarbiausių algoritmų sąsajos funkcijos yra pateikiamos lengvai. Naudotojai gali naudoti sąsajos generatoriaus funkcijas norėdami sukurti papildomas sąsajos funkcijas savo nuožiūra arba netgi sukurti sąsajos funkcijas, skirtingas nuo numatytųjų, paprastai modifikuodami grąžinimo struktūrą, kad ji būtų išsiųsta į skirtingus, galimai vartotojo apibrėžtus, *plot()* arba *summary()* metodus [HBZ07] [Tea07].

Verta paminėti, jog nors „RWeka” paketas ir turi pritaikytą *plot()* funkciją, ji veikia, pavyzdžiui, „J48”, „M5P” ar „LMT” algoritmams, tačiau jei yra sukurta algoritmo „REPTree” ar „RandomTree” sąsaja, paprastas „RWeka” atvaizdavimo būdas neveikia. Gera žinia ta, jog „RWeka” pakete yra implementuota „write_to_dot” funkcija, kuri sukuria „DOT” kalbos reprezentacijas sukurtiems klasifikatoriams. Tada šios reprezentacijos apdorojamos naudojant „Graphviz” programą. Taip pat, „DOT” reprezentacija gali būti perduota atgal į „R” ir vizualizuota naudojant „Rgraphviz” paketą [HBZ07].

Į „RWeka” paketą yra įtrauktas nemodifikuotas „Weka” jar failas, kad šis būtų maksimaliai prižiūrimas: su kiekviena nauja „Weka” versija taip pat išleidžiama ir nauja „RWeka” versija, kurios nereikia papildomai modifikuoti. Be jar failo „RWeka” kodo bazę sudaro du komponentai: pirma, pagrindinis komponentas yra aukšto lygio „R” kodas sąsajų generatoriams ir raportavimui

bei naudingi metodai, kuriuos atiduoda „rJava" paketas. Antra, yra tam tikras „Java" sąsajos kodas, skirtas efektyvumui pagerinti, pavyzdžiui, skirtingai nei R duomenų rėmai (angl. Data frames), „Weka" duomenų įrašai yra suskirstyti į eilę, o numatymai yra pavieniams įrašams: našumui pagerinti yra naudojamas „Java" kodas visiems įrašams nustatyti [HBZ07] [Tea07].

4 Naudotos priemonės ir rezultatai

Šio skyriaus tikslas yra pristatyti baigiamojo darbo tikslų bei uždavinių įgyvendinimą bei tam atlikti naudotas priemones. Pagrindinis tikslas – sukurti „R" paketą, kuris supaprastintų sprendimų medžių atvaizdavimą. „Weka" programinė įranga turi grafines galimybes, tačiau jos yra skurdžios ir atvaizduojami medžiai dažniausiai neįskaitomi dėl persidengiančių lapų ar viršūnių, jei medis yra didelis. Todėl buvo nuspręsta surasti būdą, kaip naudotis standartiniais „Weka" algoritmais, tačiau grafiniam medžių atvaizdavimui naudoti nemokamą „R" programinę įrangą. Sukurto paketo pagalba galima spręsti klasifikavimo uždavinius naudojantis lygiai tokiomis pačiomis galimybėmis, kurias turi „Weka" tačiau gauti gražius grafikus nepaliekant „R" aplinkos. Šioje dalyje taip pat pristatoma darbo eiga, naudoti duomenys bei gauti rezultatai.

4.1 Priemonės

4.1.1 Klasifikavimo uždavinio sprendimui naudoti duomenys

Baigiamajam darbui buvo ieškomi duomenys, kurie būtų susiję su biologija arba medicina. Šiam scenarijui puikiai tiko duomenys iš 130 Jungtinių Amerikos Valstijų ligoninių apie pacientus sergančius diabetu [Dia]. Ligoninėje hospitalizuotų pacientų hiperglikemijos valdymas turi didelę įtaką rezultatams tiek sergamumo, tiek mirštamumo atžvilgiu. Tačiau yra nedaug nacionalinių diabeto priežiūros ligoninėje atliktų vertinimų, kurie galėtų būti pagrindas pokyčiams. Ši didelės klinikinės duomenų bazės (17 milijonų unikalių pacientų) analizė buvo atlikta siekiant pateikti tokį vertinimą ir rasti būsimas kryptis, kurios galėtų padėti pagerinti pacientų saugą. Buvo nustatyta beveik 70 000 stacionarinių diabeto atvejų, kurių analizė buvo pakankamai išsami [SDG+14].

Šiame tyrime naudota „Health Facts" duomenų bazė - nacionalinis duomenų sandėlis, kuriame kaupiami išsamūs klinikiniai įrašai iš ligoninių visose JAV valstijose. „Health facts" yra savanoriška programa, prieinama medicinos įstaigoms, naudojančioms „Cerner" elektroninę sveikatos įrašų sistemą. Duomenų bazėje kaupiami duomenys yra sistemingai gaunami iš programoje dalyvaujančių sveikatos priežiūros įstaigų elektroninių medicinos įrašų, taip pat duomenys apie pacientų apsilankymus įstaigose (skubios pagalbos, ambulatoriniai ir stacionariniai duomenys), duomenų teikėjo demografija - amžius, lytis ir rasė - diagnozės ir procedūros atliktos ligoninėje, patvirtintos „ICD-9-CM" kodais, laboratoriniai duomenys, vaistinių duomenys, mirtingumas ligoninėje ir ligoninės charakteristikos [SDG+14].

Duomenys, kuriuos naudojau savo baigiamajam darbui rodo diabetu sergančių pacientų 10 metų (1999–2008) klinikinę priežiūrą 130 ligoninių ir integruotų gimdymo tinklų visose Jungtinėse Amerikos Valstijose: vidurio vakaruose 18, šiaurės rytuose 58, pietuose 28 ir vakaruose 16 ligoninių. Daugelyje ligoninių (78) lovų yra nuo 100 iki 499, 38 ligoninėse mažiau nei 100, o 14 ligoninių - daugiau nei 500 [SDG+14].

Duomenų bazę sudaro 41 lentelė ir iš viso 117 atributų. Duomenų bazėje yra 74 036 643 unikalūs vizitai, kurie atitinka 17 880 231 unikalų pacientą. Kadangi šie duomenys apima ne tik atskiras ligonines, bet ir integruoto gimdymo tinklo sveikatos priežiūros sistemas, duomenyse yra tiek stacionarinių, tiek ambulatorinių duomenų, įskaitant skubios pagalbos skyrių [SDG+14].

Mano naudotas duomenų rinkinys buvo sukurtas dviem žingsniais. Pirma, įrašai buvo paimti iš duomenų bazės su 55 atributais (žr. priedas nr. 1). Antras žingsnis, buvo atlikta preliminari

duomenų analizė ir pirminis apdorojimas, išlaikant tik atributus, kurie galėtų būti panaudoti tolesnėje analizėje, t.y. turi pakankamai informacijos [SDG+14].

Iš duomenų bazės buvo paimta informacija apie pacientus atitinkančius šiuos kriterijus:

1. Pacientas buvo paguldytas į ligoninę (stacionarinis gydymas).
2. Pacientas turi bet kokio tipo ,,diabetinę" diagnozę įvestą į sistemą.
3. Gulėjimo ligoninėje laikas buvo tarp 1 ir 14 dienų.
4. Gulėjimo ligoninėje metu buvo atlikti laboratoriniai tyrimai.
5. Gulėjimo ligoninėje metu buvo skiriami vaistai.

Trečias ir ketvirtas scenarijai buvo taikomi tam, kad būtų pašalinti pacientai, kurie atvyko į ligoninę tik procedūroms, kurios truko trumpiau nei parą ir kuriuose diabeto gydymo pokyčiai buvo mažiau pastebimi. Būtina paminėti, jog ne visi duomenų rinkinyje esantys pacientai serga diabetu, bet jie į jį pateko todėl, nes bent viena iš jiems priskirtų galimų diagnozių buvo diabetas. Buvo nustatyta, kad 101766 įrašai atitinka visus aukščiau paminėtus scenarijus ir buvo naudojami tolimesnei analizei. Atributų bei savybių atranką atliko pirminiame tyrime dalyvavę klinikiniai ekspertai. Buvo išsaugoti tie požymiai, kurie gali būti potencialiai susiję su diabeto būkle ar gydymu. Iš duomenų bazėje esančios informacijos buvo išgauti 55 požymiai, kurie apibūdina diabetą, įskaitant demografinius duomenis, diagnozes, vaistus nuo diabeto, apsilankymų skaičių tai ir mokėtojų informaciją [SDG+14].

Pirmame priede pateikiamas atributo pavadinimas originalo kalba, atributo aprašymas ir jo tipas. Taip pat, atributai, kurie turi nemažai reikšmių yra sugrupuoti taip, kad būtų patogiau spręsti klasifikavimo uždavinius, pavyzdžiui, amžiaus atributas originaliai turėjo daug mažesnių intervalų, bet kad sprendimų medis nebūtų per daug sudėtingas intervalų skaičių sumažinau ir priskyriau jiems skaitmenį, kad medis būtų lengviau skaitomas.

4.1.2 „R" paketo sukūrimui naudotos priemonės

Tam, kad būtų įgyvendintas baigiamojo darbo tikslas – sukurtas ,,R" paketas, kuris automatizuoja sprendimų medžių sukūrimą ir atvaizdavimą, reikėjo panaudoti keletą jau egzistuojančių ,,R" paketų bei juose esančių funkcijų, kurias aprašysiu šiame skyrelyje.

- read.arff() - ,,RWeka" paketo funkcija, kuri nuskaito ,,Weka" .arff (angl. Attribute Relation File Format). Kintamasis, kuris turi būti perduotas funkcijai – kelias iki failo, kurį norime nuskaityti, pavyzdžiui: readData <- read.arff(file="/Users/Desktop/iris.arff")
- make_Weka_classifier() - ,,RWeka" paketo funkcija, kuri sukuria sąsają iš ,,R" į jau egzistuojantį ,,Weka" klasifikatorių. Kad sukurti sąsają, šiai funkcijai užtenka nurodyti klasifikatoriaus pavadinimą pagal oficialią ,,Weka" JNI notifaciją [Wek], pavyzdžiui REPTreeClass <- make_Weka_classifier(name = "weka.classifiers.trees.REPTree"). Įvykdžius šią komandą, kintamasis ,,REPTreeClass" tampa sąsaja tarp ,,Weka" ir ,,R" ir naudojant jį galime klasifikuoti duomenis naudojantis visomis ,,Weka" galimybėmis: results <- REPTreeClass(formula = class ~ ., data = readData, control = Weka_control(M = 5)). Kintamasis ,,formula" nurodo tikslo atributo pavadinimą (šiuo atveju ,,class") bei pagal kuriuos kitus atributus bus vykdoma klasifikacija. Šiuo atveju tai nurodo taško simbolis, kuris reiškia, kad bus klasifikuojama pagal visus atributus, bet tai galima pakeisti išvardinant tik norimus. ,,Data" kintamasis nurodo duomenis, kuriuos klasifikuosime (jie turi būti nuskaityti su aukščiau minėta read.arff() funkcija). ,,Control" kintamasis leidžia naudoti visus klasifikatoriaus parametrus, kuriuos galima nustatyti ,,Weka" programinėje įrangoje. ,,M = 5" reiškia, kad medyje bus atvaizduojami tik tie lapai, kuriuose yra ne mažiau 5 įrašų.
- tempfile() – ,,R" funkcija, kuri sukuria laikiną failą sistemoje, į kurį vėliau talpinamas sprendimų medis.

- write_to_dot – dar viena „RWeka" paketo funkcija, kuri sukuria „DOT" kalbos reprezentaciją objektui, kad šį galėtų apdoroti „Rgraphviz" paketas ir atvaizduoti sprendimų medį.
- rainbow() – paketo „RPMG" funkcija, kuri lapams bei viršūnėms su sutampančiais pavadinimais priskiria spalvas, kad medis būtų lengviau skaitomas.
- evaluate_Weka_classifier() - „RWeka" paketo funkcija naudojama įvertinti klasifikatoriaus tikslumui. Naudojant šią funkciją galima įvertinti klasifikatoriaus tikslumą naudojant kryžminį tikrinimą (kintamasis „numFolds") bei pateikiant kontrolines imtis (kintamasis „newdata").
- make_Weka_filter – taip pat „RWeka" paketo funkcija, kuria galima pasiketi bet kurį „Weka" filtrą. Pavyzdžiui, ją galima naudoti norint patikrinti klasifikavimo tikslumą pateikiant kontrolines imtis. Tą galima padaryti iš pradinių duomenų pasigaminus du failus, o tam yra reikalingas filtras „Resample". Jį padarius ir naudojant jo opcijas, galima padaryti du failus – vieną pateikiamų duomenų failą, kitą testavimo imtį ir taip patikrinti klasifikatoriaus tikslumą. Užtenka paduoti filtro pavadinimą pagal JNI [Wek] notaciją, pavyzdžiui: make_Weka_filter(weka.filters.unsupervised.instance.Resample").

4.2 Sukurto paketo funkcijų aprašymas

Baigiamojo darbo metu sukūriau „R" paketą, kurį pavadinau „RWekaGraphics". Jis sudarytas iš 5 funkcijų:

1. REPTreeClass – funkcija, kuri sukuria sąsają su „REPTree" algoritmu ir įvykdo algoritmą pasirinktiems duomenims. Kad ši funkcija veiktų jai yra būtini du parametrai, tai „fileName" ir „classAttribute". Jie nurodo failą, kurį reikia klasifikuoti bei tikslo atributą, pavyzdžiui, REPTreeClass(fileName = "/Users/Desktop/iris.arff", classAttribute = "class"). Šiuo atveju algoritmas būtų įvykdytas „iris" duomenims. Be šių dviejų atributų, funkcija turi kontrolės variantus, kuriais naudojantis galima manipuliuoti klasifikavimo tikslumu [Wek]:

    a. MinNum – minimalus lapo „svoris". Jei nustatome, kad šis parametras yra lygus 10, į medį nepateks tie lapai, kurie turi mažiau nei 10 įrašų. Jei neįrašyta, reikšmė pagal nutylėjimą yra 2.
    b. MinVarianceProp – mažiausia visų duomenų, kurie turi būti mazge, dispersijos dalis, kad būtų galima vykdyti padalijimą regresijos medžiuose. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0.001.
    c. NumFolds - nustatomas genėjimui naudojamas duomenų kiekis. Viena klostė naudojama genėjimui, kita - taisyklėms auginti. Jei neįrašyta, reikšmė pagal nutylėjimą yra 3.
    d. Seed – naudojama atsitiktiniams duomenims parinkti. Jei neįrašyta, reikšmė pagal nutylėjimą yra 1.
    e. NoPrunning – nusako ar medžių genėjimas turi būti naudojamas. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    f. MaxDepth – didžiausias medžio gylis. Jei neįrašyta, reikšmė pagal nutylėjimą yra -1 (medžio gylis neapribotas).
    g. InitialCount – pradinis klasės vertės skaičius. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0.0.
    h. SpreadInitialCount – pradinį skaičių skaičiuoti visoms vertėms, o ne atskirai kiekvienai. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    i. OutputDebug – klasifikatorius gali išvesti papildomą informaciją į konsolę. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.

j. doNotCheckCapabilities – klasifikatoriaus galimybės nėra tikrinamos prieš sukuriant klasifikatorių (reikia naudoti atsargiai norint sumažinti programos veikimo laiką). Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
k. numDecimalPlaces – dešimtainių skaitmenų skaičius naudojamas modeliui išvesti. Jei neįrašyta, reikšmė pagal nutylėjimą 2.
l. batchSize - pageidaujamas apdoroti įrašų skaičius, jei atliekamas dalies duomenų numatymas. Gali būti pateikta daugiau ar mažiau egzempliorių, tačiau tai suteikia galimybę nurodyti pageidaujamą partijos dydį. Jei neįrašyta, reikšmė pagal nutylėjimą yra 100.

„REPTreeClass" naudojimo pavyzdys: results <- REPTreeClass(fileName = "/Users/Desktop/iris.arff", classAttribute = "class", MinNum = 10, MinVarianceProp = 0.001, NumFolds = 2, Seed = 1, NoPrunning = TRUE, MaxDepth = 6, InitialCount = 0.0, SpreadInitialCount = FALSE, OutputDebug = FALSE, doNotCheckCapabilities = TRUE, numDecimalPlaces = 2, batchSize = 150).

2. „RandomTreeClass" funkcija, kuri sukuria sąsają su „RandomTree" algoritmu ir jį įvykdo pasirinktiems duomenims. Kad ši funkcija veiktų jai yra būtini du parametrai - „fileName" ir „classAttribute". Jie nurodo failą, kurį reikia klasifikuoti bei tikslo atributą, pavyzdžiui, RandomTreeClass(fileName = "/Users/Desktop/iris.arff", classAttribute = "class"). Šiuo atveju algoritmas būtų įvykdytas „iris" duomenims. Be šių dviejų atributų, funkcija turi kontrolės variantus, kuriais naudojantis galima manipuliuoti klasifikavimo tikslumu [Wek]:

a. Kvalue – nustato atsitiktinai pasirinktų atributų klasifikavimui skaičių. Jei neįrašyta, reikšmė pagal nutylėjimą 0.
b. MinNum – minimalus lapo „svoris". Jei nustatome, kad šis parametras yra lygus 10, į medį nepateks tie lapai, kurie turi mažiau nei 10 įrašų. Jei neįrašyta, reikšmė pagal nutylėjimą yra 2.
c. MinVarianceProp – mažiausia visų duomenų, kurie turi būti mazge, dispersijos dalis, kad būtų galima vykdyti padalijimą regresijos medžiuose. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0.001.
d. Seed – naudojama atsitiktiniams duomenims parinkti. Jei neįrašyta, reikšmė pagal nutylėjimą yra 1.
e. MaxDepth – didžiausias medžio gylis. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0 (medžio gylis neapribotas).
f. NumFolds - Nustatomas duomenų, naudojamų „backfitting" algoritmui, kiekis. Viena klostė naudojama „backfitting", kitos - medžio auginimui. Jei neįrašyta, reikšmė pagal nutylėjimą 0 („backfitting" nenaudojamas).
g. allowUnclassifiedInstances – ar leisti neklasifikuotus įrašus. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
h. breakTiesRandomly – nutraukti ryšius atsitiktine tvarka, kai keli atributai yra vienodai geri. Jei neįrašyta, reikšmė pagal nutylėjimą FALSE.
i. OutputDebug – klasifikatorius gali išvesti papildomą informaciją į konsolę. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
j. doNotCheckCapabilities – klasifikatoriaus galimybės nėra tikrinamos prieš sukuriant klasifikatorių (reikia naudoti atsargiai norint sumažinti programos veikimo laiką). Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
k. numDecimalPlaces – dešimtainių skaitmenų skaičius naudojamas modeliui išvesti. Jei neįrašyta, reikšmė pagal nutylėjimą 2.
l. batchSize - pageidaujamas apdoroti įrašų skaičius, jei atliekamas dalies duomenų numatymas. Gali būti pateikta daugiau ar mažiau egzempliorių, tačiau tai suteikia

galimybę nurodyti pageidaujamą partijos dydį. Jei neįrašyta, reikšmė pagal nutylėjimą yra 100.

„RandomTreeClass" naudojimo pavyzdys: results <- RandomTreeClass(fileName = "/Users/Desktop/iris.arff", classAttribute = "class", KValue = 0, MinNum = 10, MinVarianceProp = 0.001, Seed = 1, MaxDepth = 5, NumFolds = 0, allowUnclassifiedInstances = FALSE, breakTiesRandomly = TRUE, OutputDebug = FALSE, doNotCheckCapabilities = TRUE, numDecimalPlaces = 1, batchSize = 100).

3. „RandomForestClass" funkcija, kuri sukuria sąsają su „RandomForest" algoritmu ir jį įvykdo pasirinktiems duomenims. Kad ši funkcija veiktų jai yra būtini du parametrai - „fileName" ir „classAttribute". Jie nurodo failą, kurį reikia klasifikuoti bei tikslo atributą, pavyzdžiui, RandomTreeClass(fileName = "/Users/Desktop/iris.arff", classAttribute = "class"). Šiuo atveju algoritmas būtų įvykdytas „iris" duomenims. Be šių dviejų atributų, funkcija turi daug kontrolės variantų, kuriais naudojantis galima manipuliuoti klasifikavimo tikslumu [Wek]

    a. bagSizePercent - kiekvieno krepšio dydis procentais nuo treniruočių komplekto dydžio. Jei neįrašyta, reikšmė pagal nutylėjimą yra 100.
    b. calcOutOfBag – ar apskaičiuoti „out-of-bag" klaidą. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    c. storeOutOfBagPredictions – ar išsaugoti „out-of-bag" prognozes vidiniame vertinimo objekte. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    d. outputOutOfBagComplexityStatistics - ar reikia pateikti sudėtingumą statistiką, kai atliekamas „out-of-bag" vertinimas. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    e. printClassifiers - atspausdina atskirus klasifikatorius išvestyje. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    f. computeAttributeImportance – apskaičiuoja ir išveda atributų svarbą. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    g. numIterations – vykdymų skaičius. Jei neįrašyta, reikšmė pagal nutylėjimą yra 100.
    h. numExecutionSlots - vykdymo gijų skaičius, naudojamas ansambliui sukonstruoti. Jei neįrašyta, reikšmė pagal nutylėjimą yra 1.
    i. numFeatures – atsitiktinai vertinamų atributų skaičius. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0.
    j. minNum - minimalus lapo „svoris". Jei nustatome, kad šis parametras yra lygus 10, į medį nepateks tie lapai, kurie turi mažiau nei 10 įrašų. Jei neįrašyta, reikšmė pagal nutylėjimą yra 2.
    k. MinVarianceProp – mažiausia visų duomenų, kurie turi būti mazge, dispersijos dalis, kad būtų galima vykdyti padalijimą regresijos medžiuose. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0.001.
    l. seed - naudojama atsitiktiniams duomenims parinkti. Jei neįrašyta, reikšmė pagal nutylėjimą yra 1.
    m. maxDepth - didžiausias medžio gylis. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0 (medžio gylis neapribotas).
    n. numFolds - Nustatomas duomenų, naudojamų „backfitting" algoritmui, kiekis. Viena klostė naudojama „backfitting", kitos - medžio auginimui. Jei neįrašyta, reikšmė pagal nutylėjimą 0 („backfitting" nenaudojamas).
    o. allowUnclassifiedInstances - ar leisti neklasifikuotus įrašus. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    p. breakTiesRandomly – nutraukti ryšius atsitiktine tvarka, kai keli atributai yra vienodai geri. Jei neįrašyta, reikšmė pagal nutylėjimą FALSE.

q. OutputDebug – klasifikatorius gali išvesti papildomą informaciją į konsolę. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
r. doNotCheckCapabilities – klasifikatoriaus galimybės nėra tikrinamos prieš sukuriant klasifikatorių (reikia naudoti atsargiai norint sumažinti programos veikimo laiką). Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
s. numDecimalPlaces – dešimtainių skaitmenų skaičius naudojamas modeliui išvesti. Jei neįrašyta, reikšmė pagal nutylėjimą 2.
t. batchSize - pageidaujamas apdoroti įrašų skaičius, jei atliekamas dalies duomenų numatymas. Gali būti pateikta daugiau ar mažiau egzempliorių, tačiau tai suteikia galimybę nurodyti pageidaujamą partijos dydį. Jei neįrašyta, reikšmė pagal nutylėjimą yra 100.

„RandomForestClass" naudojimo pavyzdys: results <- RandomForestClass(fileName = "/Users/Desktop/iris.arff", classAttribute = "class", bagSizePercent = 100, calcOutOfBag = FALSE, storeOutOfBagPredictions = TRUE, outputOutOfBagComplexityStatistics = FALSE, printClassifiers = FALSE, computeAttributeImportance = TRUE, numIterations = 50, numExecutionSlots = 2, numFeatures = 0 MinNum = 10, MinVarianceProp = 0.001, Seed = 1, MaxDepth = 5, NumFolds = 0, allowUnclassifiedInstances = FALSE, breakTiesRandomly = TRUE, OutputDebug = FALSE, doNotCheckCapabilities = TRUE, numDecimalPlaces = 1, batchSize = 100).

4. „ J48Class" funkcija, kuri sukuria sąsają su „J48" algoritmu ir jį įvykdo pasirinktiems duomenims. Kad ši funkcija veiktų jai yra būtini du parametrai - „fileName" ir „classAttribute". Jie nurodo failą, kurį reikia klasifikuoti bei tikslo atributą, pavyzdžiui, J48Class(fileName = "/Users/Desktop/iris.arff", classAttribute = "class"). Šiuo atveju algoritmas būtų įvykdytas „iris" duomenims. Be šių dviejų atributų, funkcija turi kontrolės variantus, kuriais naudojantis galima manipuliuoti klasifikavimo tikslumu [Wek]:

    a. unpruned – nustato ar medis turi būti genimas. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    b. collapseTree – nustato ar pašalinti medžio dalis, kurios nesumažina mokymo klaidų skaičiaus. Jei neįrašyta, reikšmė pagal nutylėjimą yra TRUE.
    c. confidenceFactor – pasitikėjimo koeficientas naudojamas medžių genėjimui – kuo mažesnis, tuo labiau medis genimas. Jei neįrašyta, reikšmė pagal nutylėjimą yra 0.25.
    d. minNumObj – minimalus lapo „svoris". Jei nustatome, kad šis parametras yra lygus 10, į medį nepateks tie lapai, kurie turi mažiau nei 10 įrašų. Jei neįrašyta, reikšmė pagal nutylėjimą yra 2.
    e. reducedErrorPruning – nustato ar vietoj „C4.5" genėjimo naudojamas sumažintos klaidos (angl. Reduced-error) genėjimas. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    f. binarySplits – nustato ar kuriant medžius naudojami dvejetainiai skilimai nominaliems atributams. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    g. subtreeRaising – nustato ar gali būti svarstoma „sub medžio" (angl. subtree) auginimo galimybė vykdant pagrindinio medžio genėjimą. Jei neįrašyta, reikšmė pagal nutylėjimą yra TRUE.
    h. saveInstanceData – nustato ar reikia išsaugoti treniruočių duomenis sprendimų medžio vizualizacijai. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    i. useLaplace – nustato ar lapų skaičius yra išlygintas remiantis Laplaso formule. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
    j. MDLcorrelation – nustato ar „MDL" korekcija naudojama ieškant skaitinių atributų skilimo vietų. Jei neįrašyta, reikšmė pagal nutylėjimą yra TRUE.

k. seed – naudojama atsitiktine tvarka atrenkant duomenis, kai vykdomas naudojant sumažintos klaidos (angl. Reduced error) genėjimas. Jei neįrašyta, reikšmė pagal nutylėjimą yra 1.
l. doNotmakeSplitPointActualValue – jei pasirinktas variantas TRUE, padalijimo taškas neperkeliamas į faktinę duomenų vertę. Tai gali žymiai pagreitinti didelių duomenų rinkinių su skaitiniais atributais klasifikavimą. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
m. OutputDebug – klasifikatorius gali išvesti papildomą informaciją į konsolę. Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
n. doNotCheckCapabilities – klasifikatoriaus galimybės nėra tikrinamos prieš sukuriant klasifikatorių (reikia naudoti atsargiai norint sumažinti programos veikimo laiką). Jei neįrašyta, reikšmė pagal nutylėjimą yra FALSE.
o. numDecimalPlaces – dešimtainių skaitmenų skaičius naudojamas modeliui išvesti. Jei neįrašyta, reikšmė pagal nutylėjimą 2.
p. batchSize - pageidaujamas apdoroti įrašų skaičius, jei atliekamas dalies duomenų numatymas. Gali būti pateikta daugiau ar mažiau egzempliorių, tačiau tai suteikia galimybę nurodyti pageidaujamą partijos dydį. Jei neįrašyta, reikšmė pagal nutylėjimą yra 100.

„J48Class" naudojimo pavyzdys: results <- J48Class(fileName = "/Users/Desktop/iris.arff", classAttribute = "class", unpruned = FALSE, collapseTree = TRUE, confidenceFactor = 0.23, minNumObj = 2, reducedErrorPruning = FALSE, binarySplits = TRUE, subtreeRaising = FALSE, saveInstanceData = TRUE, useLaplace = FALSE, MDLcorrelation = TRUE, seed = 1, doNotmakeSplitPointActualValue = FALSE, OutputDebug = FALSE, doNotCheckCapabilities = FALSE, numDecimalPlaces = 3, batchSize = 80).

5. „TreePlot" funkcija skirta sprendimų medžių grafiniam atvaizdavimui. Šios funkcijos veikimui yra reikalingas tik vienas atributas - „results". Šis kintamasis turi būti gaunamas atlikus klasifikavimą su vienu iš aukščiau paminėtų „RWekaGraphics" paketo funkcijų. „TreePlot" funkcija gavusi rezultatų kintamąjį, išanalizuoja visus medyje esančius objektus (lapus, šaknį) ir atributams, kurių pavadinimai sutampa, suteikia vienodą spalvą. Tokiu būdu sprendimų medis tampa dar skaitomesnis.

   Taip pat, verta paminėti, jog gautasis medis gali būti grąžinamas įvairiais formatais, tokiais kaip: „PNG", „JPEG", „TIFF", „BMP", „EPS" ar „PDF". Savo darbe medžius piešiau „PDF" formatu, nes taip išsaugoma aukščiausia raiška.

4.3 Rezultatai

Šio skyriaus tikslas yra pristatyti aukščiau aprašytų funkcijų rezultatus. Testuoti funkcijas buvo naudojami 130 Junginių Amerikos Valstijų ligoninių duomenys (žr. 4.1.1 skyrių).

4.3.1 „J48Class" ir „TreePlot" funkcijų rezultatai

Pirmasis analizuotas scenarijus, naudojant naujai sukurtas funkcijas, buvo naudojant „readmitted" kaip tikslo atributą. Buvo tikrinama, kaip paciento grįžimas į stacionarinį gydymą priklauso nuo paciento rasės, lyties, amžiaus, kaip pacientas pateko į ligoninę, kur pacientas buvo išrašytas, kas rekomendavo pacientui kreiptis į gydymo įstaigą, kiek laiko praleido ligoninėje, „A1C" testo rezultatų, 23 skirtų vaistų dozių pokyčio („metamorfin"-„ metformin-pioglitazone"), ar buvo skirti vaistai nuo diabeto ir ar jų dozės buvo keičiamos (žr. priedas nr. 1).

Šis sprendimų medis atvaizduotas naudojant mano sukurto paketo funkciją „TreePlot". Paveikslėlis grąžinamas „PDF" formatu:

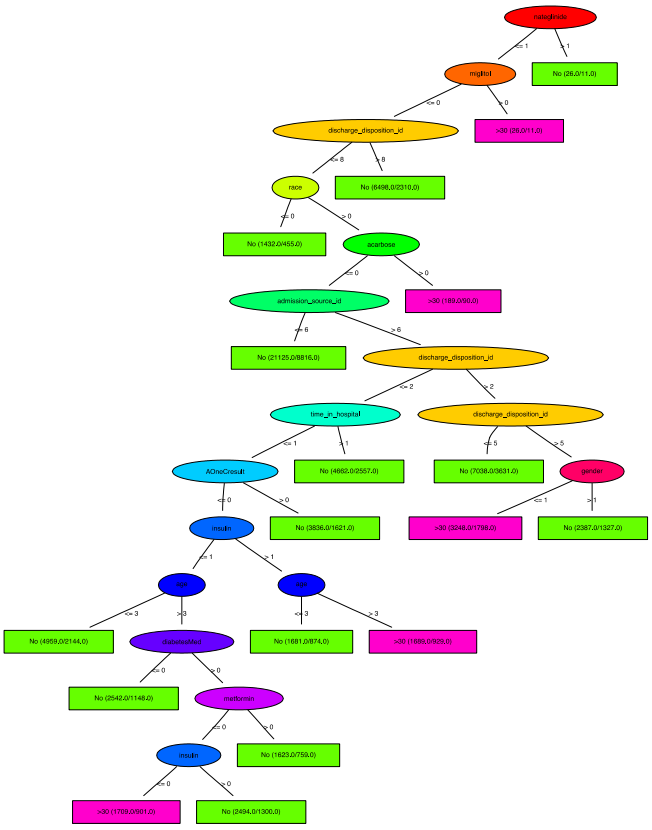
```
library(RWekaGraphics)
```

```
Classified <- J48Class("/Users/Desktop/J4866.arff",
"readmitted", minNumObj = 1300)
DecisionTree <- TreePlot(Classified)
pdf('/Users/Desktop/J48.pdf')
plot(DecisionTree)
dev.off()
```

Ši funkcija reikalauja vieno kintamojo „results” kuris gaunamas panaudojus vieną iš klasifikavimo algoritmų, mano paketa tai būtų „REPTreeClass”, „J48Class”, „RandomTreeClass” arba „RandomForestClass” funkcijos.

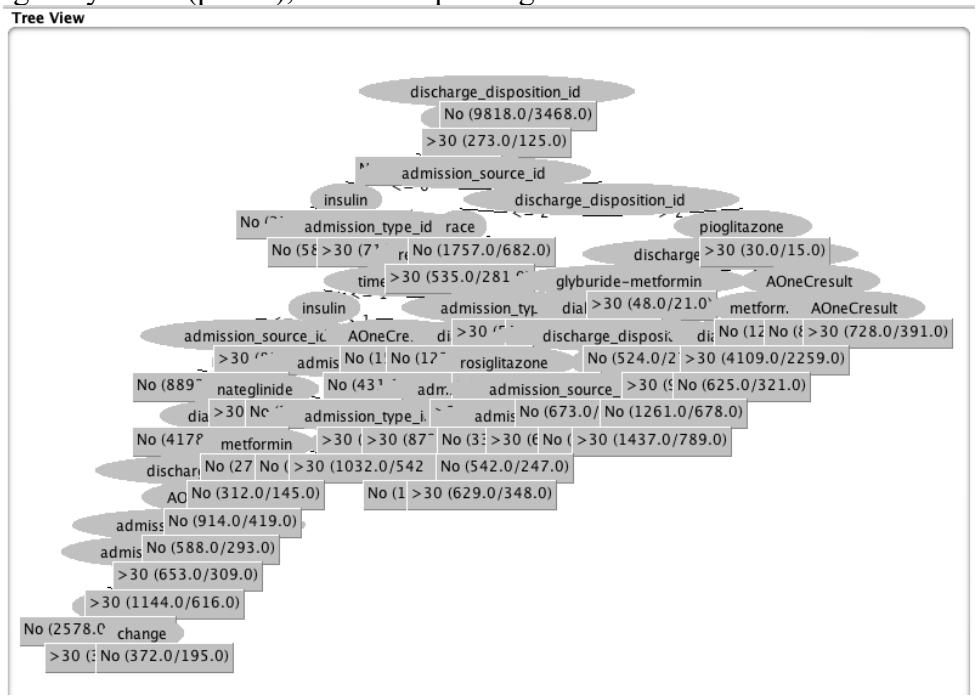
Įdomu tai, kad šiuo atveju informatyviausias atributas apie paciento grįžimą į stacionarinį gydymą buvo, kur pacientas keliavo po gydymo ligoninėje („discharge_disposition _id). Matome, kad jei „discharge_disposition_id” reikšmė yra didesnė už 8, pacientai į stacionarinį gydymą nebesugrįžo. Tačiau galima pastebėti, jog jei reikšmė, kur pacientas keliavo iš ligoninės yra mažesnė arba lygi 8, jo rasė yra žinoma (atributo reikšmė > 0) bei jam buvo skirta akrabozės – pacientas grįžo į stacionarinį gydymą po daugiau nei 30 dienų (pav. 7). Taip pat, aiškiai matoma, jog tai, ar pacientas grįžo į stacionarą priklauso nuo amžiaus (<=3 – negrįžo, >3 – grįžo po daugiau nei 30 dienų).

Taip pat, buvo vertinamas klasifikatoriaus tikslumas jį testuojant trimis būdais: kryžminio tikrinimo, kontrolinių imčių bei naudojant treniruočių duomenis (angl. Use Training Set). Pirmu atveju, buvo naudojamos kontrolines imtys – pradinis duomenų failas buvo suskaldytas į duomenų rinkinį (66%) ir mokymo imtį (34%). Tai buvo atlikta naudojant „Resample” filtrą. Šio testo rezultatas – klasifikatorius teisingai suklasifikavo 54.2165% įrašų. Naudojant kryžminį tikrinimą su 10 klosčių teisingai suklasifikuota 54.1823% irašų, o kai buvo naudojami mokymo duomenys (angl. Use Training Set) – 54.3178% irašų.



pav 7. „J48Class” ir „TreePlot” funkcijos. Tikslo atributas „readmitted”.

Matome, jog paveikslėlis yra spalvotas (vienodos lapų etiketės nuspalvintos tomis pačiomis spalvomis) ir lapai nepersidengia. Taip pat, artinant paveikslėlį jo kokybė nenukenčia. Palyginus su „Weka" galimybėmis (pav 8.), medis tampa daug labiau skaitomesnis.



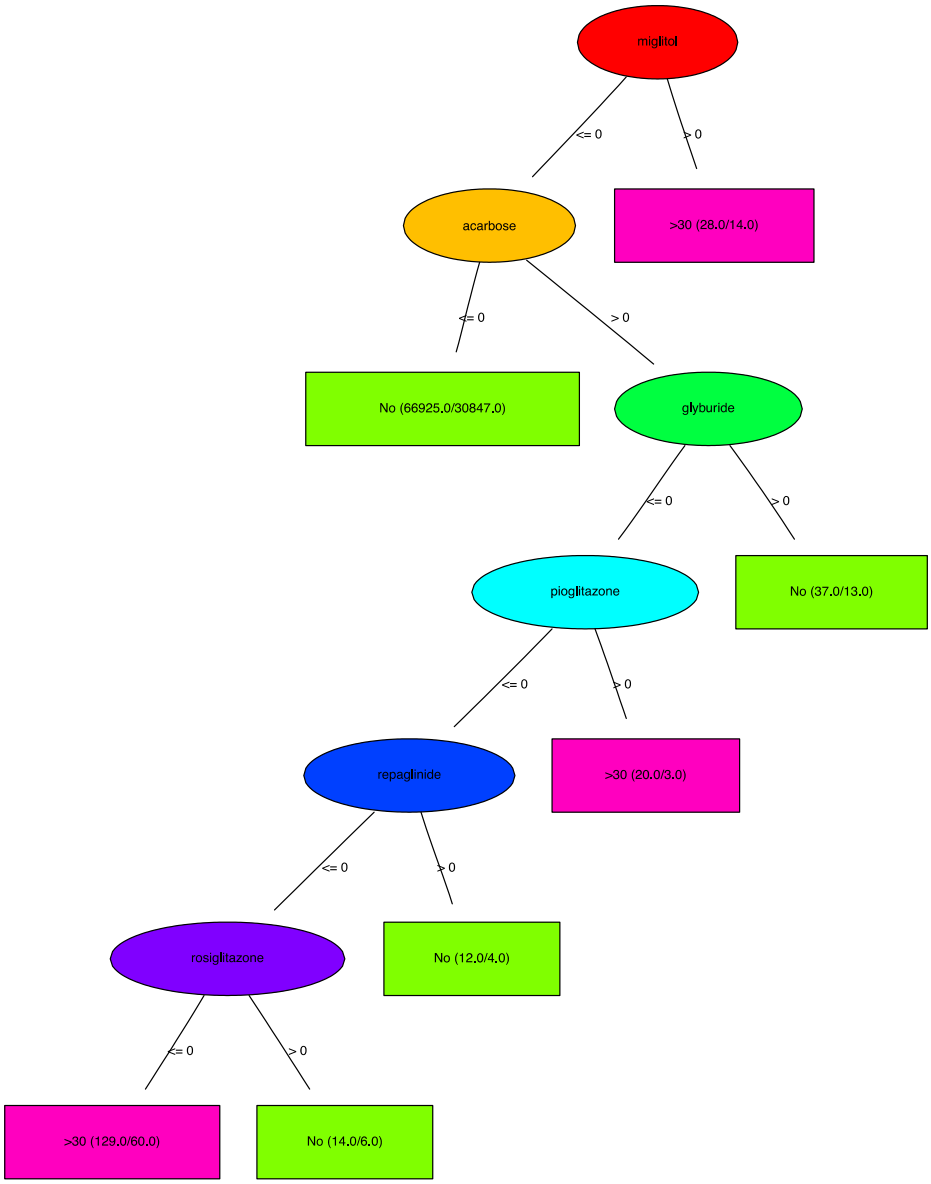
pav 8. „J48" ir grafinis atvaizdavimas atliktas „Weka". Tikslo atributas „readmitted".

Antrasis scenarijus taip pat susijęs su „readmitted" atributu. Šiuo atveju buvo paimta šiek tiek siauresnė klasifikuojamų atributų aibė – A1C testo rezultatai bei 23 gydymo metu skirti arba neskirti vaistai („metamorfin"-„ metformin-pioglitazone").

Šiuo atveju, informatyviausias atributas yra „miglitol". Matome, jog jei gydymo metu pacientams buvo skirta miglitolio, jie sugrįžo į stacionarinį gydymą po daugiau nei 30 dienų. Kitu atveju, jei miglitolio nebuvo skirta, tačiau buvo skirta akrabozė – pacientai nebesugrįžo (pav. 9). Kontrolinės imtys teisingai suklasifikavo 54.0534%, kryžminis tikrinimas 53.9424%, o naudojant mokymo duomenis (angl. Use Training Set) - 54.0505% įrašų.

Šis medis gaunamas panaudojant tokį kodo segmentą:

```
library(RWekaGraphics)
Classified <- J48Class("/Users/Desktop/J48Readmitted66.arff",
"readmitted", minNumObj = 10)
DecisionTree <- TreePlot(Classified)
pdf('/Users/Desktop/J48Readmitted.pdf')
plot(DecisionTree)
dev.off()
```



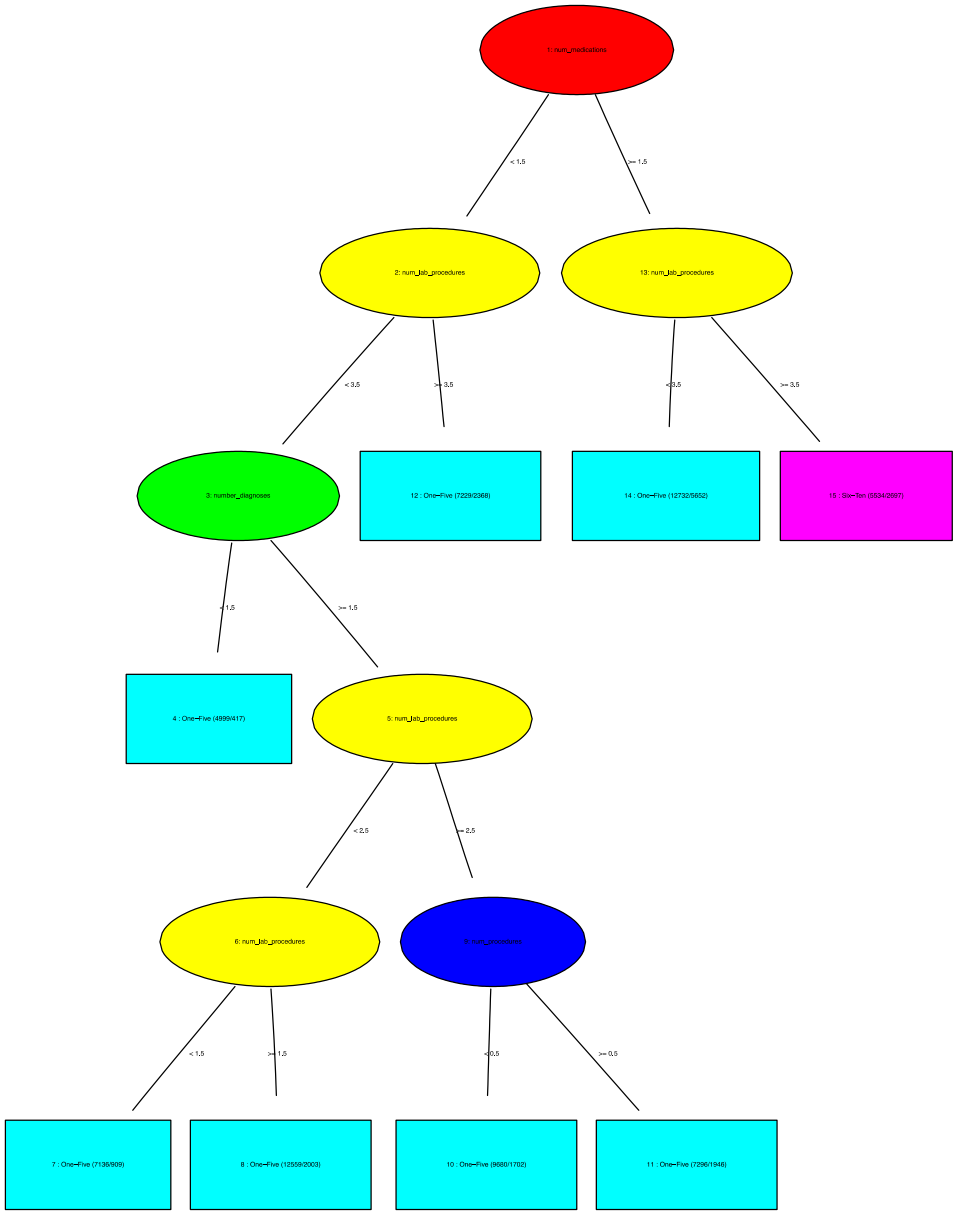
pav 9. „J48Class" ir „TreePlot" funkcijos. Tikslo atributas „readmitted", kiti atributai – 23 medikamentai ir A1C tyrimai.

4.3.2 „RandomTreeClass" ir „TreePlot" funkcijų rezultatai

Tam, kad ištestuoti „RandomTreeClass" funkcijos veikimą buvo pasirinktas kitas tikslo atributas „time_in_hospital" – kiek laiko pacientas buvo hospitalizuotas. Kiti tyrimui naudoti atributai – laboratorinių tyrimų skaičius, kitų procedūrų (ne laboratorinių tyrimų) skaičius, kiek skirtingų vaistų buvo skirta gydymo metu ir kiek skirtingų diagnozių buvo įvesta į sistemą (pav. 10). Medžiui nupiešti naudojamos komandos:

```
library(RWekaGraphics)
Classified <-
RandomTreeClass("/Users/Desktop/time_in_hospital66Percent",
"time_in_hospital", MinNum = 7000)
DecisionTree <- TreePlot(Classified)
pdf('/Users/Desktop/time_in_hospital_classified.pdf')
plot(DecisionTree)
```

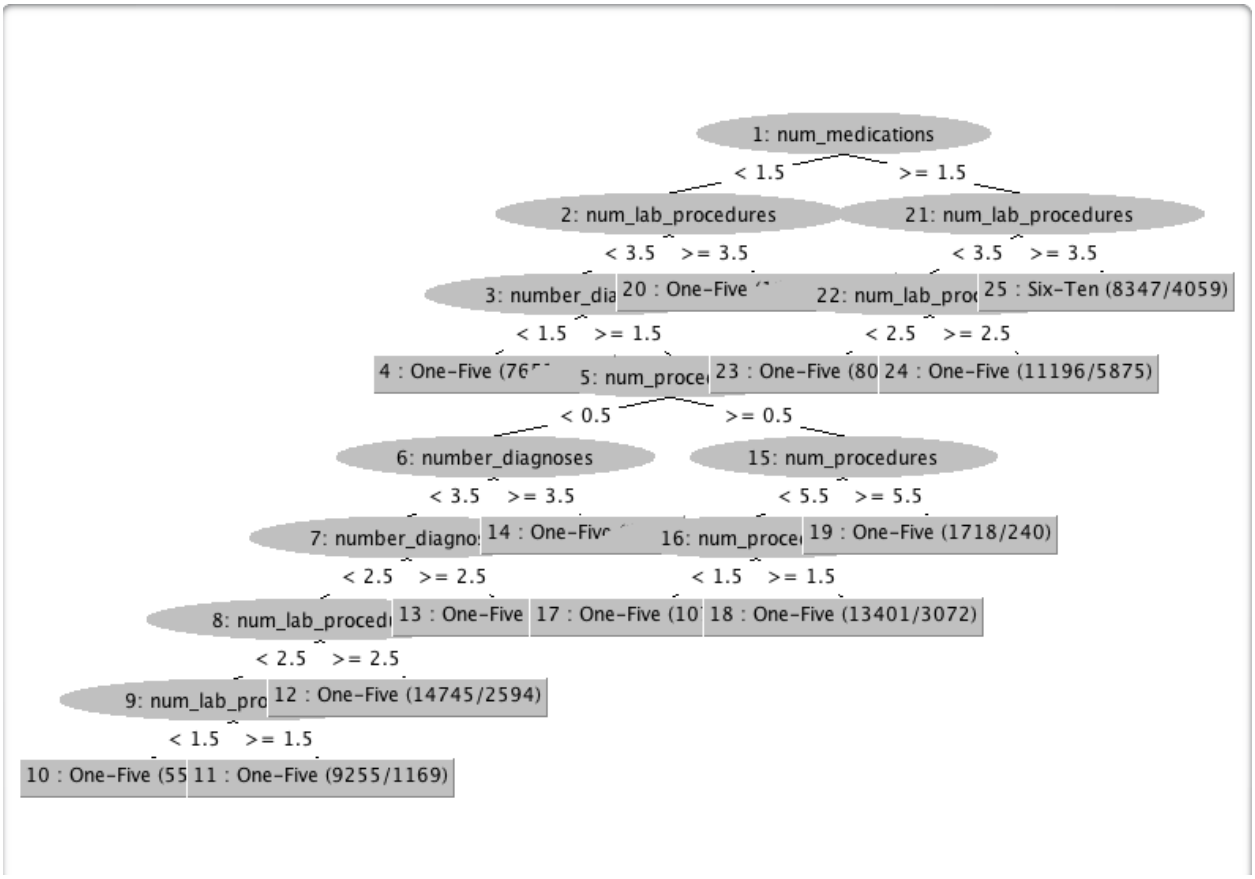
```
dev.off()
```



pav 10. ,,RandomTreeClass" ir ,,TreePlot" funkcijos. Tikslo atributas ,,time_in_hospital”

Atributas ,,num_medications” ir šiuo atveju yra informatyviausias, todėl yra medžio viršūnė. Kaip matoma, beveik visais atvejais klasifikuojant aukščiau išvardintus atributus, pacientų vizitas ligoninėje truko nuo vienos iki penkių dienų, pavyzdžiui, jei pacientui buvo skirta iki 30 medikamentų ir atlikta daugiau nei 70 laboratorinių testų, pacientai ligoninėje praleido nuo vienos iki penkių dienų. Vienu atveju pacientai išbuvo ligoninėje nuo šešių iki dešimties dienų – kai buvo skirta daugiau nei 30 medikamentų ir atlikta ne daugiau nei 70 laboratorinių tyrimų. Naudojant kontrolines imtis, klasifikatorius teisingai suklasifikavo 74.0817% įrašų, kryžminis tikrinimas - 73.5029%, o naudojant mokymo duomenis (angl. Use Training Set) - 73.8017% įrašų.

Palyginimui, ,,RandomTree” algoritmas bei vizualizacija atlikta ,,Weka” programoje:



pav 11. „RandomTree" algoritmas ir grafinis atvaizdavimas „Weka" programoje.

Naudojantis algoritmų turimais kontrolės atributais medį galima supaprastinti. Pavyzdžiui, pav. 10 kairėje pusėje visi klasifikavimo rezultatai yra „one-five", t.y. pacientai ligoninėje buvo nuo vienos iki penkių dienų. Todėl yra logiška panaikinti antrą viršūnę ir ją pakeisti lapu su reikšme „One-Five".

Tai vyksta taip: pirma įvykdomas klasifikavimo algoritmas „RandomTreeClass" su nurodytais parametrais. Tada į naują kintamąjį įdedame medį, kurį sugeneruoja funkcija „TreePlot".

```
library(RWekaGraphics)
Classified <-
RandomTreeClass("/Users/Desktop/time_in_hospital66Percent",
"time_in_hospital", MinNum = 7000, maxDepth = 3)
DecisionTree <- TreePlot(Classified)
```

Sugeneruotą medį įrašome į failą naudojant „Rgraphviz" paketo funkciją „agwrite". Modifikuojame failą pagal save, nustatome viršūnių ar lapų pavadinimus, formą, spalvas (pav. 12).

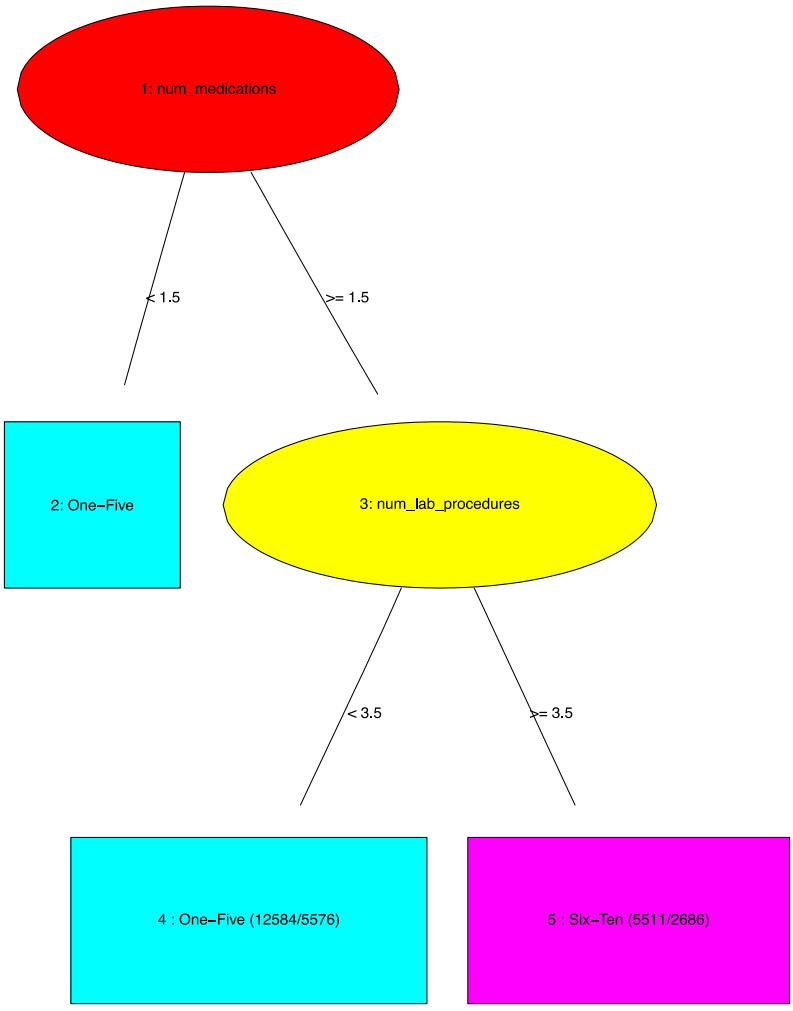
```
agwrite(DecisionTree, "/Users/Desktop/DecisionTreeFile.txt")
```

```
digraph RandomTree {
        node [label="\N"];
        edge [style=bold];
        graph [bb="0 0 798 306"];
        N6aaceffd [label="1: num_medications", fillcolor="#FF0000FF"];
        Nc86b9e3 [label="2: One-Five", shape=box, fillcolor="#00FFFFFF"];
        N4c309d4d [label="3: num_lab_procedures", fillcolor="#FFFF00FF"];
        N38102d01 [label="4 : One-Five (12584/5576)", shape=box, fillcolor ="#00FFFFFF"];
        N37883b97 [label="5 : Six-Ten (5511/2686)", shape=box, fillcolor ="#FF00FFFF"];
        N6aaceffd -> Nc86b9e3 [label=" < 1.5"];
        N6aaceffd -> N4c309d4d [label=" >= 1.5"];
        N4c309d4d -> N38102d01 [label=" < 3.5"];
        N4c309d4d -> N37883b97 [label=" >= 3.5"];
}
```

pav 12. Agwrite() atiduotas funkcijos medis įrašytas į failą. Pakeistas taip, kad pav. 10 esantis medis būtų redukuotas.

Nuskaitome medį naudodami funkciją „agread” ir išsaugome medį norimu formatu. Rezultatas – redukuotas medis, kuris turi mažiau lapų ir viršūnių, bet jo reikšmė nepakitusi (pav. 13).

```
DecisionTree <-agread("/Users/Desktop/DecisionTreeFile.txt")
pdf("/Users/Desktop/DecisionTree.pdf”)
plot(DecisionTree)
dev.off()
```



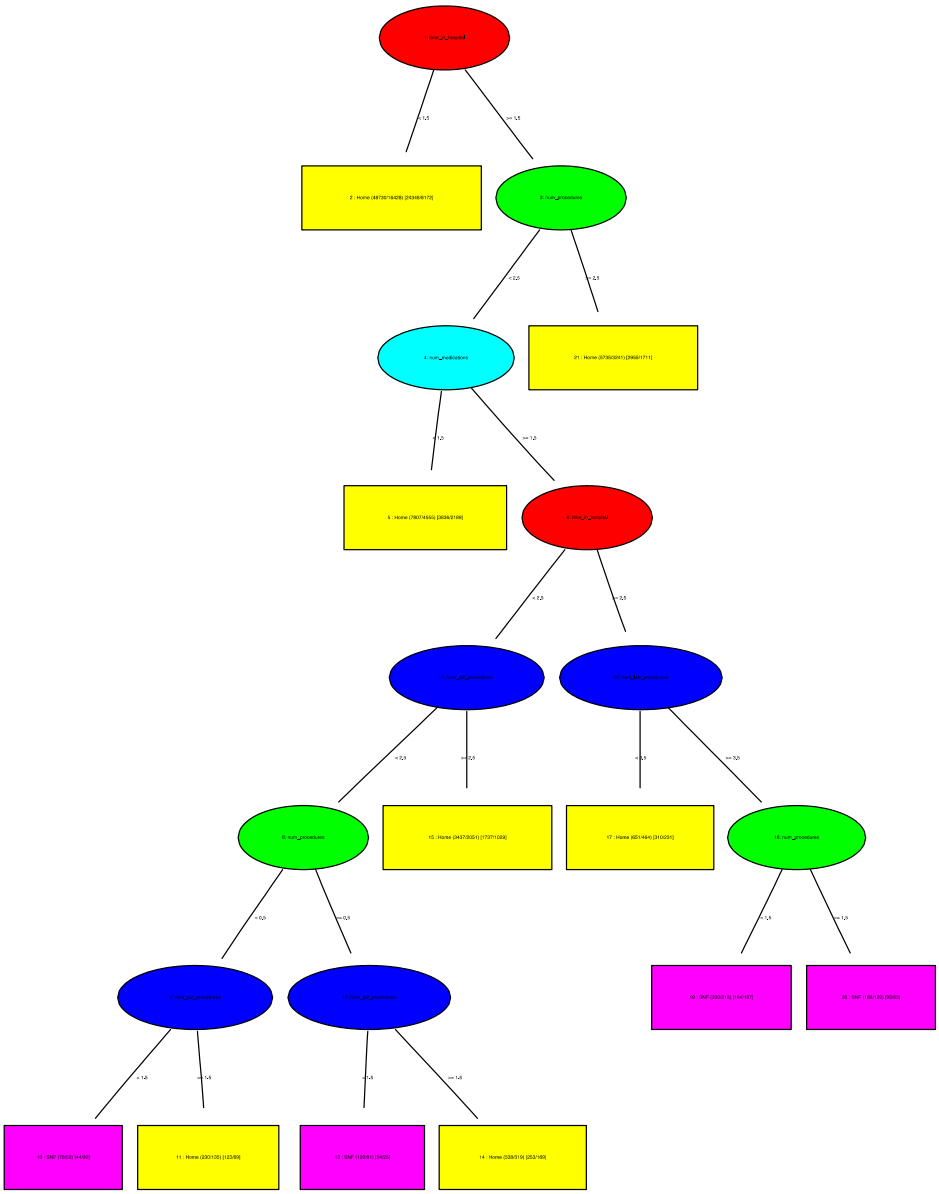
pav 13. Redukuotas pav. 10 medis.

4.3.3 „REPTreeClass” ir „TreePlot” funkcijų rezultatai

„REPTreeClass” funkcijai patikrinti buvo panaudotas dar kitoks duomenų pjūvis – tikslo atributas „discharge_disposition_id” arba kur pacientas buvo išrašytas iš stacionarinio gydymo. Pirmajam scenarijui papildomi atributai buvo: kiek laiko pacientas praleido ligoninėje, kiek testų buvo atlikta laboratorijoje ir skaičius ne laboratorinių testų, kiek medikamentų buvo skirta bei diagnozių įvestų į sistemą skaičius (pav. 14). Medis gaunamas naudojant šį kodą:

```
library(RWekaGraphics)
Classified <-
REPTreeClass("/Users/Desktop/discharge_disposition_id",
"time_in_hospital", MinNum = 70)
```

```
DecisionTree <- TreePlot(Classified)
pdf('/Users/Desktop/ discharge_disposition_id.pdf')
plot(DecisionTree)
dev.off()
```

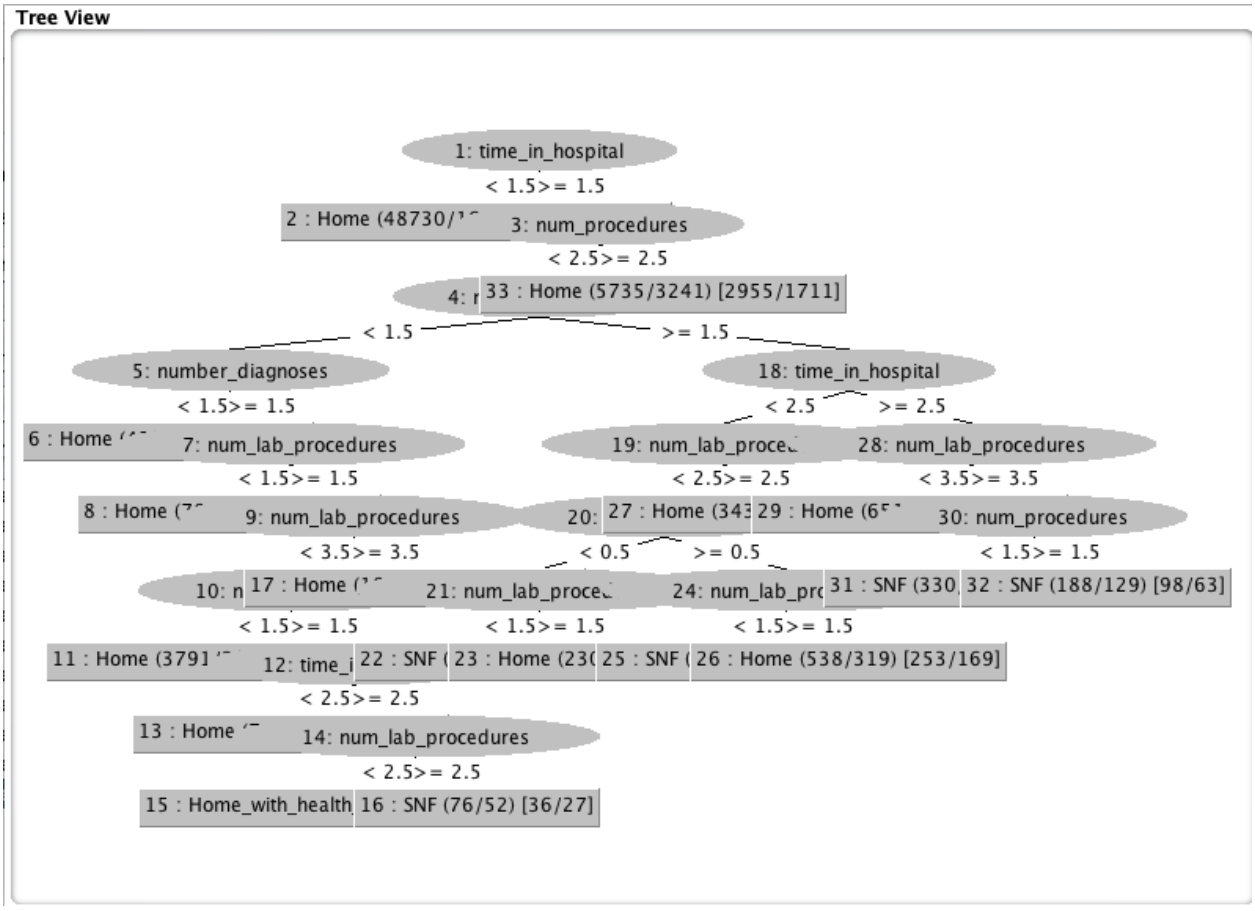


pav 14. ,,REPTreeClass" ir ,,TreePlot" funkcijos. Tikslo atributas ,,discharge_disposition_id".

Šiame scenarijuje informatyviausias atributas yra ,,time_in_hospital” – kiek laiko pacientas praleido ligoninėje. Jau pačiame medžio viršuje galima matyti, kad jei pacientas ligoninėje praleido ne daugiau nei 10 dienų, jis buvo išrašytas namo. Kitu atveju pacientai buvo išrašyti į kvalifikuotas slaugos įstaigas (angl. SNF – Skilled nursing facility).

Klasifikatoriai buvo testuojami naudojant kryžminį tikrinimą, kontrolines imtis bei naudojant mokymo duomenis (angl. Use Training Set). Šiuo atveju kontrolinių imčių klasifikavimas – suklasifikavo 59.4% įrašų teisingai, kryžminis tikrinimas 59.2103%, o naudojant mokymo duomenis (angl. Use Training Set) - 59.2516%.

Palyginimui, ,,REPTree” algoritmas bei vizualizacija atlikta ,,Weka” programoje:



pav 15. ,,REPTree" algoritmas ir vizualizacija atlikta ,,Weka" programoje

Išvados

Bakalauro baigiamajame darbe buvo pristatyti įvairūs klasifikavimo algoritmai, kurių rezultate gaunami sprendimų medžiai, aptarti „Weka“ programinės įrangos grąžinami klasifikavimo rezultatai, detalizuotos kokybės charakteristikos, išsamus tikslumas pagal klases bei nesutapimų matrica. Taip pat, pristatytas rezultatų vaizdavimui naudotas duomenų rinkinys, „RWeka“ paketas, „Weka“ programinė įranga bei naujai sukurtas „R“ paketas.

„Weka“ programinė įrangą yra labai gera tuo, jog turi daugybę mašininio mokymosi algoritmų vienoje vietoje. Deja, jos vizualizacijos galimybės yra labai skurdžios, todėl vienas iš uždavinių buvo sukurti „R“ paketą, kurio pagalba būtų galima spręsti klasifikavimo užduotis ir gražiai atvaizduoti sprendimų medžius. „R“ buvo pasirinktas dėl to, jog yra nemokama, atviro kodo programinė įranga turinti plačias vizualizacijos galimybes.

Sukurtas „R“ paketas susideda iš 5 funkcijų: 4 skirtingi klasifikavimo algoritmai ir viena medžio atvaizdavimui. Atvaizduojamas medis yra daug gražesnis, nei jį nubraižo „Weka“, lapai nepersidengia ir yra lengvai skaitomas. Taip pat, jei medyje yra lapų, kurių pavadinimas yra toks pat, jiems yra priskiriama spalva, kuri padaro medį dar suprantamesnį. Naujasis paketas gali būti plečiamas pridedant kitus klasifikavimo algoritmus, taip pat, galima naudoti visus „Weka“ programinės įrango filtrus ar atributų vertintojus neišeinant iš „R“ aplinkos.

Baigiamajame darbe nemažai dėmesio skirta gilesniam susipažinimui su „Weka“ klasifikavimo algoritmais bei pačios programinės įrangos veikimu. Taip pat, pagilintos „R“ programavimo kalbos bei duomenų paruošimo ir analizės žinios.

Literatūra

[Add19] Automatic Addison, “Iterative Dichotomiser 3 (ID3) algorithm from scratch”, https://automaticaddison.com/iterative-dichotomiser-3-id3-algorithm-from-scratch/, 2019, tikrinta 2020-05-14.

[AND36] Edgar Anderson, "The species problem in Iris”, Annals of the Missouri Botanical Garden, Vol. 23, 1936, pp. 457-509.

[Asi18] Sidath Asiri, “Machine Learning Classifiers”, https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623, 2018, tikrina 2020-05-16.

[Bar92] Anthony G. Barnston, “Correspondence among the Correlation, RMSE, and Heidke Forecast Verification Measures; Refinement of the Heidke Score”, Weather and Forecasting, Vol. 7, 1992, pp. 699-700.

[Bro16] Jason Brownlee, How to use Classification Machine Learning Algorithms in Weka, https://machinelearningmastery.com/use-classification-machine-learning-algorithms-weka/, 2016, tikrinta 2020-05-16.

[Bro18] Jason Brownlee, How to Use ROC Curves and Precision-Recall Curves for Classification in Python, https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/, 2018, tikrinta 2020-05-14.

[Coh] Cohen’s kappa, https://en.wikipedia.org/wiki/Cohen%27s_kappa, tikrinta 2020-05-16.

[Dia] Diabetes 130-US hospitals for years 1999-2008 Data Set, https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008, tikrinta 2020-05-10.

[Don19] Niklas Donges, A complete guide to random forest algorithm, https://builtin.com/data-science/random-forest-algorithm#how, 2019, tikrinta 2020-05-15.

[DPR+14] Payal P.Dhakate, Suvarna Patil, K. Rajeswari, Deepa Abin, “Preprocessing and Classification in WEKA Using Different Classifier”, Journal of Engineering Research and Applications, Vol. 4, 2014, pp. 91-93.

[Epp] David Eppstein, Random tree, https://en.wikipedia.org/wiki/Random_tree, tikrinta 2020-05-14.

[Eva] Evaluation of binary classifiers, https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers, tikrinta 2020-05-14.

[FIS36] R.A. Fisher, "The Use of Multiple Measures in Taxonomic Problems", Ann. Eugenics, vol. 7, 1936, pp. 179-188.

[HBZ07] Kurt Hornik, Christian Buchta, Achim Zeileis, ,, Open-Source Machine Learning: R Meets Weka”, Computational Statistics. Vol. 24, 2009, p. 225-232.

[HPK+02] Geoffrey Holmes, Bernhard Pfahringer, Richard Kirkby, Eibe Frank and Mark Hall, "Multiclass Alternating Decision Trees", ECML, 2002, pp. 161-172.

[Hus19] Shadab Hussain, Use Cases of Different Machine Learning Algorithms, https://blog.usejournal.com/machine-learning-algorithms-use-cases-72646df1245f, 2019, tikrinta 2020-05-16.

[Kra] Stefan Kramer, M5P, http://old.opentox.org/dev/documentation/components/m5p, tikrinta 2020-05-14.

[MAE] MAE and RMSE – Which Metric is Better? https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d, tikrinta 2020-05-16.

[Mat] Matthews correlation coefficient, https://en.wikipedia.org/wiki/Matthews_correlation_coefficient, tikrinta 2020-05-17.

[McH19] Mary L. McHugh, Interrater reliability: the kappa statistic, 2019, p. 276-282.

[Mea] F measure, https://metacademy.org/graphs/concepts/f_measure, tikrinta 2020-05-17.

[Nar18] Sarang Narkhede, Understanding AUC – ROC curve, https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5, 2018, tikrinta 2020-05-16.

[Oct11] Octavian's blog, Decision trees – C4.5, https://octaviansima.wordpress.com/2011/03/25/decision-trees-c4-5/, 2011, tikrinta 2020-05-14.

[Pfa11] Bernhard Pfahringer, "Random model trees: an effective and scalable regression method", University of Waikato, Department of Computer Science, 2010.

[Pol17] Saimadhu Polamuri, How the random forest algorithm works in machine learning, https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/, 2017, tikrinta 2020-05-14.

[Sco] F1 score, https://en.wikipedia.org/wiki/F1_score, tikrinta 2020-05-17.

[SDG+14] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records, BioMed Research International, Vol. 2014, 2014, pp. 781670.

[Shi07] Haijan Shi, Best-first Decision Tree Learning, The University of Waikato, 2007.

[SM14] Dr. B. Srinivasan, P.Mekala, "Mining Social Networking Data for Classification Using REPTree", International Journal of Advance Research in Computer Science and Management Studies, Vol 2, 2014, pp. 155-160.

[Tea07] R Development Core Team, "R: A Language and Environment for Statistical Computing.", 2007.

[Wek] Weka functions, https://weka.sourceforge.io/doc.stable/?fbclid=IwAR2E-xhfDl3S1PvgTMMmjGfTBM28FgyRFsJ81cxHwUZd_2bN3sQPT_MR2KU, tikrinta 2020-05-17.

[WF11] Ian H. Witten, Eibe Frank, ‘‘Data Mining: Practical Machine Learning Tools and Techniques“, Morgan Kaufmann Publishers, Second Edition, 2005.

[WFH11] Ian H. Witten, Eibe Frank & Mark A. Hall., “Data Mining Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers, Third Edition.”, 2011.

[Wis13] K. Wisaeng , “A Comparison of Different Classification Techniques for Bank Direct Marketing”, Computer Science, 2013, pp. 116-119.

Santrauka

Darbe analizuojamas įvairių klasifikavimo algoritmų veikimas, „Weka“ bei „R“ programinės įrangos galimybės sprendžiant klasifikavimo uždavinius bei vizualizuojant rezultatus. Darbe pateikiamas devynių klasifikavimo algoritmų aprašymas, „Weka“ klasifikavimo algoritmo rezultatas bei klasifikavimo metu gauti įverčiai. Praktinėje darbo dalyje pristatomas naujai sukurtas „R“ paketas, kuris yra pritaikomas tiek klasifikavimo algoritmų vykdymui, tiek sprendimų medžių atvaizdavimui. Taip pat, pateikiamas sukurto paketo rezultatai naudojantis dideliu duomenų kiekiu.

Raktažodžiai: Weka, RWeka, klasifikavimas, klasifikatoriai, besimokantysis, sprendimų medis, duomenų kasimas, duomenų analizė, duomenų modelis.

Summary

The work analyses the operation of various classification algorithms, the possibilities of "Weka" and "R" software in solving classification problems and visualizing the results. The paper presents a description of nine classification algorithms, the result of the Weka classification algorithm and the estimates obtained during the classification. In the practical part of the work, a newly developed "R" package is presented, which is applicable both for the implementation of classification algorithms and for the representation of decision trees. Also, the results of the created package using a large amount of data are presented.

Keywords: Weka, RWeka, classification, classificators, learner, decision tree, data mining, data analysis, data model.

Priedas nr. 1

Atributo pavadinimas	Atributo apibūdinimas	Atributo tipas
Encounter_id	Priėmimo į medicinos įstaigą numeris	Skaitinis
patient_nbr	Unikalus numeris suteikiamas pacientui	Skaitinis
race	Paciento rasė	Nominalus
0	Unknown	
1	AfricanAmerican	
2	Asian	
3	Caucasian	
4	Hispanic	
5	Other	
Gender	Paciento lytis	Nominalus
1	Female	
2	Male	
3	Unknown/Invalid	
Age	Paciento amžius	Nominalus
[0-20)	1	
[20-40)	2	
[40-60)	3	
[60-80)	4	
[80-100)	5	
Weight	Paciento svoris svarais	Skaitinis
?	Unknown	
[0-25)	0	
[25-50)	1	
[50-75)	2	
[75-100)	3	
[100-125)	4	
[125-150)	5	
[150-175)	6	
[175-200)	7	
>200	8	
admission_type_id	Kaip pacientas pateko į ligoninę	Nominalus
1	Emergency	
2	Urgent	
3	Elective	
4	Newborn	
5	Not Available	
6	NULL	
7	Trauma Center	

8	Not Mapped	
discharge_disposition_id	Kur pacientas keliavo po ligoninės	Nominalus
1	Discharged to home	
2	Discharged/transferred to another short term hospital	
3	Discharged/transferred to SNF	
4	Discharged/transferred to ICF	
5	Discharged/transferred to another type of inpatient care institution	
6	Discharged/transferred to home with home health service	
7	Left AMA	
8	Discharged/transferred to home under care of Home IV provider	
9	Admitted as an inpatient to this hospital	
10	Neonate discharged to another hospital for neonatal aftercare	
11	Expired	
12	Still patient or expected to return for outpatient services	
13	Hospice / home	
14	Hospice / medical facility	
15	Discharged/transferred within this institution to Medicare approved swing bed	
16	Discharged/transferred/referred another institution for outpatient services	
17	Discharged/transferred/referred to this institution for outpatient services	
18	NULL	
19	Expired at home. Medicaid only, hospice.	
20	Expired in a medical facility. Medicaid only, hospice.	
21	Expired, place unknown. Medicaid only, hospice.	
22	Discharged/transferred to another rehab fac including rehab units of a hospital .	
23	Discharged/transferred to a long term care hospital.	
24	Discharged/transferred to a nursing facility certified under Medicaid but not certified under Medicare.	
25	Not Mapped	
26	Unknown/Invalid	
30	Discharged/transferred to another Type of Health Care Institution not Defined Elsewhere	
27	Discharged/transferred to a federal health care facility.	
28	Discharged/transferred/referred to a psychiatric hospital of psychiatric distinct part unit of a hospital	
29	Discharged/transferred to a Critical Access Hospital (CAH).	
admission_source_id	Kas rekomendavo pacientui kreiptis į gydymo įstaigą	Nominalus
1	Physician Referral	
2	Clinic Referral	
3	HMO Referral	
4	Transfer from a hospital	

5	Transfer from a Skilled Nursing Facility (SNF)	
6	Transfer from another health care facility	
7	Emergency Room	
8	Court/Law Enforcement	
9	Not Available	
10	Transfer from critial access hospital	
11	Normal Delivery	
12	Premature Delivery	
13	Sick Baby	
14	Extramural Birth	
15	Not Available	
17	NULL	
18	Transfer From Another Home Health Agency	
19	Readmission to Same Home Health Agency	
20	Not Mapped	
21	Unknown/Invalid	
22	Transfer from hospital inpt/same fac reslt in a sep claim	
23	Born inside this hospital	
24	Born outside this hospital	
25	Transfer from Ambulatory Surgery Center	
26	Transfer from Hospice	
time_in_hospital	Kiek dienų pacientas praleido ligoninėje	Skaitinis
[1-5)	1	
[6-10)	2	
[10-14)	3	
Payer code	Kaip pacientas atsiskaitė už paslaugas	Nominalus
Medical specialty	Priimančio mediko specialybė	Nominalus
num_lab_procedures	Skaičius testų, kurie buvo atlikti laboratorijoje	Skaitinis
[1-20)	1	
[20-40)	2	
[40-60)	3	
[60-80)	4	
[80-100)	5	
[100-120)	6	
[120-140)	7	
num_procedures	Procedūrų ar testų (atliekamų ne laboratorijoje) skaičius	Skaitinis
[0-20)	1	
[20-40)	2	
[40-60)	3	

[60-80)	4	
[80-100)	5	
[100-120)	6	
num_medications	Skaičius medikamentų skirtų gydymo ligoninėje metu	Skaitinis
[0-20)	1	
[20-40)	2	
[40-60)	3	
[60-80)	4	
[80-100)	5	
number_outpatient	Kiek kartų pacientas gydėsi ambulatoriškai tais pačiais metais	Skaitinis
[0-10)	1	
[10-20)	2	
[20-30)	3	
[30-40)	4	
[40-50)	5	
number_emergency	Kiek kartų pacientas nenumatytai tais pačiais metais lankėsi gydymo įstaigose	Skaitinis
[0-20)	1	
[20-40)	2	
[40-60)	3	
[60-80)	4	
number_inpatient	Kiek kartų pacientas gydėsi stacionariai tais pačiais metais	skaitinis
[0-10)	1	
[10-20)	2	
[20-30)	3	
diag_1	Pirminė diagnozė	Nominalus
diag_2	Antrinė diagnozė	Nominalus
diag_3	Papildoma diagnozė	Nominalus
number_diagnoses	Diagnozių įvestų į sistemą skaičius	Skaitinis
[0-5)	1	
[5-10)	2	
[10-15)	3	
[15-20)	4	
max_glu_serum	Gliukozės kiekio serume tyrimo rezultatai	Nominalus
0	None (not measured)	
1	Norm	
2	>200	
3	>300	

A1Cresult	Glikuoto hemoglobino (HbA1c) tyrimo rezultatai	Nominalus
0	None (not measured)	
1	Norm	
2	>7	
3	>8	
metformin	Metamorfino vaistų dozės gydymo ligoninėje metu (no - neskirta)	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
repaglinide	Repaglinido vaistų dozės gydymo ligoninėje metu (no - neskirta)	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
nateglinide	Nateglinido vaistų dozės gydymo ligoninėje metu (no - neskirta)	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
chlorpropamide	Chloropropamido vaistų dozės gydymo ligoninėje metu (no - neskirta)	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
glimepiride	Glimepirido vaistų dozės gydymo ligoninėje metu (no - neskirta)	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
acetohexamide	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
glipizide	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
glyburide	No - not prescribed	Nominalus
0	No (not prescribed)	

1	Steady	
2	Up	
3	Down	
tolbutamide	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
pioglitazone	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
rosiglitazone	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
acarbose	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
miglitol	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
troglitazone	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
tolazamide	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
examide	No - not prescribed	Nominalus
0	No (not prescribed)	
citoglipton	No - not prescribed	Nominalus
0	No (not prescribed)	
insulin	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	

3	Down	
glyburide-metformin	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
2	Up	
3	Down	
glipizide-metformin	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
glimepiride-pioglitazone	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
metformin-rosiglitazone	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
metformin-pioglitazone	No - not prescribed	Nominalus
0	No (not prescribed)	
1	Steady	
change	Parodo ar buvo keisti medikamentai (dozė arba vistas) diabeto gydymui	Nominalus
0	No	
1	Changed	
diabetesMed	Parodo ar buvo skirta meikamentu diabeto gydymui	Nominalus
0	No	
1	Yes	
readmitted	Dienų skaičius iki pacientas grįžo į stacionarinį gydymą	Nominalus
0	No	
1	>30	
2	<30	