Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gregor Čepin

Večciljno učenje v klasifikaciji in regresiji

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Marko Robnik-Šikonja

Ljubljana, 2016

Izjava o avtorstvu magistrskega dela

Spodaj podpisani Gregor Čepin sem avtor magistrskega dela z naslovom:

Večciljno učenje v klasifikaciji in regresiji

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marka Robnika-Šikonje,
- so elektronska oblika magistrskega dela, naslov (slovenski, angleški), povzetek (slovenski, angleški) ter ključne besede (slovenske, angleške) identični s tiskano obliko magistrskega dela,
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki "Dela FRI".

V Ljubljani, 15. marca 2016 Podpis avtorja:

Kazalo

Povzetek

Večciljno učenje je metoda strojnega učenja, pri kateri je cilj, da se algoritem obenem nauči reševati več sorodnih problemov. Namesto več ločenih modelov poišče algoritem skupen model, ki je tipično manjši od vsote velikosti posameznih modelov, je lažje razumljiv in manj prilagojen učnim podatkom. Pri napovedovanju s skupnim modelom algoritem napoveduje vrednosti za več problemov hkrati. Uporabljeni problemi morajo biti med seboj sorodni, da lahko učenje enega problema pripomore k boljšemu učenju ostalih problemov. Do sedaj je bil pristop pri drevesnih metodah uspešno uporabljan za združevanje več klasifikacijskih ali več regresijskih problemov, v tem delu pa pristop posplošimo tako, da lahko uporabljamo klasifikacijske in regresijske naloge mešano. Pri gradnji drevesnih modelov uporabljata klasifikacija in regresija različne metode za izbiro atributov pri delitvi primerov v notranjih vozliščih. Vrednosti ocen med seboj niso primerljive, zato pri gradnji drevesa atribute rangiramo glede na obe metodi in izberemo atribut, ki je skupno najbolje rangiran. V nalogi implementiramo večciljno odločitveno in regresijsko drevo ter ansambelski metodi večciljni bagging in večciljni naključni gozd. Primerjamo jih z enociljnimi različicami algoritmov, z običajnimi večciljnimi drevesi in z večciljnimi nevronskimi mrežami. Uvedemo mero sorodnosti med nalogami, ki temelji na rangiranju atributov. Ta nam omogoča, da znotraj podatkovne množice poiščemo tiste naloge, ki so si najbolj sorodne in jih je smiselno obravnavati z večciljnim pristopom. Na eni od podatkovnih množic deluje implementirani večciljni naključni gozd statistično značilno bolje kot enociljni algoritmi. Na nekaterih podatkovnih množicah pa implementirani

algoritmi delujejo slabše kot enociljni.

Ključne besede

strojno učenje, odločitveno drevo, večciljno učenje, naključni gozd, bagging, klasifikacija, regresija, rangiranje

Abstract

Multitask learning is an approach to machine learning, in which algorithm learns to solve multiple related problems. It tries to find one common model instead of building multiple separate models. Such a model is usually smaller than the sum of separate models, easier to understand and less likely to overfit training data. In prediction stage the algorithm predicts values for several problems at the same time. Problems that are learned together must be related, so that learning of one problem can improve learning of other problems. Currently this approach is used with tree models for either multiple classification or multiple regression tasks. In this work we extend the approach to mixed classification and regression tasks. During construction of trees different attribute selection methods are used in regression and classification. The returned scores are not directly comparable, so in our scenario we rank attributes for each task and choose the attribute that is best ranked in total. We implement multitask regression and classification tree, multitask bagging and multitask random forest based on rankings of attributes. We compare these algorithms with their single task variants, with regular multitask tree and with multitask neural network. We propose task relatedness measure based on ranking of attributes. In this way we can find related tasks in a dataset and use them together in multitask approach. On one dataset implemented multitask random forest works statistically significantly better than single-task version. On some datasets implemented algorithms work worse than single-task versions.

Keywords

machine learning, decision tree, multitask tree, random forest, bagging, classification, regression, ranking

Poglavje 1

Uvod

Strojno učenje običajno uporabljamo za napovedovanje ene ciljne vrednosti. V kolikor nas zanimajo napovedi za več povezanih problemov, lahko zgradimo model za vsak problem posebej ali pa uporabimo večciljno učenje, pri katerem zgradimo skupen model za napovedovanje vrednosti več problemov. Tako namesto več ločenih modelov dobimo en skupen model, ki je tipično manjši od vsote velikosti posameznih modelov, ponavadi lažje razumljiv in manj prilagojen učnim podatkom. Pri večciljnem učenju lahko dosežemo tudi boljšo napovedno točnost, v kolikor učenje določenega problema pripomore k boljšemu učenju sorodnih problemov.

V tabeli 1.1 je prikazana podatkovna množica enociljnega učenja, kakršnega najpogosteje izvajamo v strojnem učenju. Podatkovna množica vsebuje za

Tabela 1.1: Prikaz običajne podatkovne množice.

	x_1	x_2	x_3	...	x_n	y
Učni primer 1	v_{11}	v_{12}	v_{13}	...	v_{1n}	w_1
Učni primer 2	v_{21}	v_{22}	v_{23}	...	v_{2n}	w_2
Učni primer 3	v_{31}	v_{32}	v_{33}	...	v_{3n}	w_3
...	...	...	...	...	...	
...	...	...	...	...	...	
Učni primer k	v_{k1}	v_{k2}	v_{k3}	...	v_{kn}	w_k

Tabela 1.2: Prikaz podatkovne množice večciljnega učenja.

	x_1	x_2	x_3	...	x_n	y_1	y_2	...	y_m
Učni primer 1	v_{11}	v_{12}	v_{13}	...	v_{1n}	w_{11}	w_{12}	...	w_{1m}
Učni primer 2	v_{21}	v_{22}	v_{23}	...	v_{2n}	w_{21}	w_{22}	...	w_{2m}
Učni primer 3	v_{31}	v_{32}	v_{33}	...	v_{3n}	w_{31}	w_{32}	...	w_{3m}
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
Učni primer k	v_{k1}	v_{k2}	v_{k3}	...	v_{kn}	w_{k1}	w_{k2}	...	w_{km}

vsak učni primer podatke o atributih x_i in eno napovedno vrednost v stolpcu y, ki se jo algoritem nauči in jo kasneje napoveduje. Tabela 1.2 prikazuje podatkovno množico večciljnega učenja. Ta vsebuje več stolpcev y_i, torej vsebuje za vsak primer več vrednosti, ki se jih mora algoritem naučiti in jih kasneje napovedati.

Osnovni pojmi povezani z večciljnim učenjem, ki so uporabljeni v tem delu, so:

Večciljno učenje - uporaba metod strojenega učenja za učenje več vrednosti, v nasprotju z večjim delom metod strojnega učenja, pri katerih napovedujemo eno vrednost.

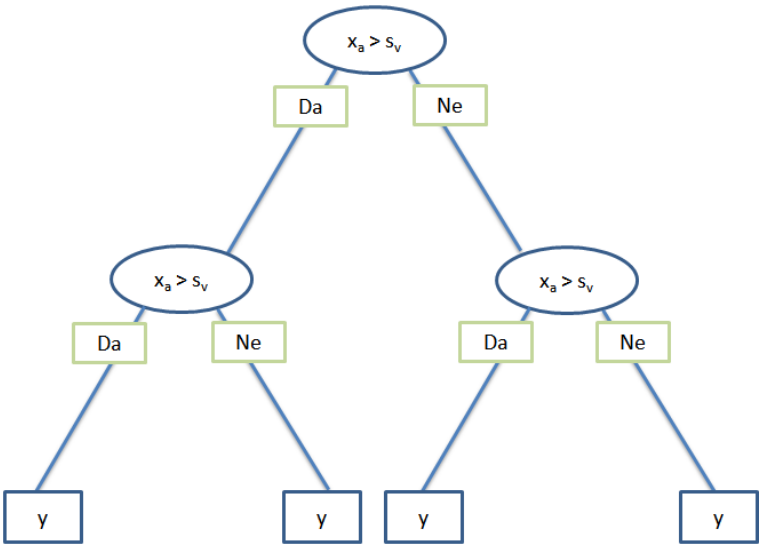
Ciljni atributi - vrednosti, ki jih algoritem napoveduje, imenujemo ciljni atributi. Njihove vrednosti v trenutku napovedi algoritma za nov primer niso znane, s pomočjo naučenega modela in vhodnih podatkov jih algoritem napove. V enociljnem učenju algoritem napoveduje en ciljni atribut, pri večciljnem učenju pa več ciljnih atributov.

Naloga - učenju ali napovedovanju vrednosti za posamezni ciljni atribut pravimo naloga. V procesu večciljnega učenja se algoritem nauči več nalog, število nalog je enako številu ciljnih atributov.

Rangiranje atributov - razvrščanje atributov glede na uspešnost posameznega atributa pri deljenju primerov v vozlišču drevesa po izbrani meri za ocenjevanje atributov.

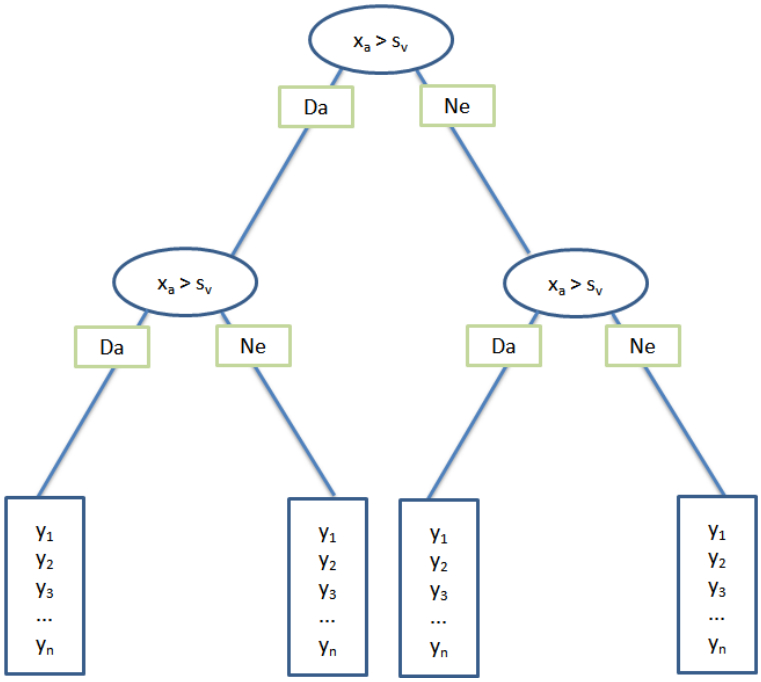
Odločitvena drevesa so ena osnovnejših metod strojnega učenja. Ob

učenju se zgradi drevo od zgoraj navzdol, naučeno znanje se hrani v drevesni obliki. Takšni modeli so dobro razumljivi. Ena od njihovih slabosti je, da se lahko preveč prilegajo učnim podatkom. Ansambelske metode zgradijo množico nekoliko različnih odločitvenih dreves, ob napovedovanju pa vsa drevesa glasujejo za končno napoved. Ansambelske metode zahtevajo več časa in prostora za učenje in napovedovanje, a zmanjšajo težave posameznih odločitvenih dreves s prekomernim prileganjem podatkom. Uporaba večciljnega učenja lahko prav tako zmanjša prekomerno prileganje učnim podatkom, saj se je algoritem primoran naučiti splošnejši model, ki upošteva vse ciljne atribute. Slika 1.1 prikazuje primer enociljnega drevesa, slika 1.2 pa primer večciljnega drevesa. Ko je drevo zgrajeno, je razlika med njima le ta, da večciljno drevo v vsakem listu napove vrednost večih ciljnih atributov, enociljno pa le vrednost enega.



Slika 1.1: Primer enociljnega drevesa.

V strojnem učenju uporabljamo eno od naslednjih dveh možnosti, glede na to kakšen ciljni atribut želimo napovedovati. Napovedujemo lahko številske vrednosti, tako napovedovanje imenujemo regresija. Lahko pa pri napovedovanju izbiramo med nekaj vnaprej določenimi vrednostmi, tak način imenu-



Slika 1.2: Primer večciljnega drevesa.

jemo klasifikacija. Pri gradnji odločitvenega drevesa želimo v vsakem vozlišču poiskati atribut, ki bo kar najbolje razdelil primere v vozlišču na dve ali več podvej. Pri klasifikaciji in regresiji se uporabljajo različne mere za ocenjevanje atributov, ki med seboj niso neposredno primerljive. Kadar imamo ob uporabi večciljnega drevesa naloge obeh tipov, lahko sicer za vsako nalogo ločeno ocenimo primernost posameznega atributa, ne moremo pa izbrati skupno najboljšega, ker mere med seboj niso primerljive. Posledično ne moremo enostavno uporabiti tako regresijskih kot klasifikacijskih nalog mešano znotraj enega večciljnega drevesa.

V tem magistrskem delu to težavo odpravimo tako, da za vsako nalogo atribute rangiramo glede na oceno, ki jo je vsak atribut dobil pri tej nalogi. S pomočjo rangiranj atributov za vse naloge lahko izberemo atribut, ki je skupno najbolje rangiran. Tako prilagojena drevesa nato uporabimo še v ansambelskih metodah bagging in naključni gozd.

V drugem poglavju predstavimo obstoječo literaturo s področja večciljnega

učenja. V tretjem poglavju opišemo večciljna odločitvena drevesa in večciljne ansambelske metode. V četrtem poglavju podrobneje opišemo naš pristop z rangiranjem. V petem poglavju so opisani poskusi, ki smo jih izvedli za primerjavo delovanja različnih algoritmov in primerjavo z obstoječimi metodami. Nato predstavimo dobljene rezultate. V šestem poglavju povzamemo naš prispevek in podamo predloge za nadaljnje delo.

Poglavje 2

Povezane raziskave

Na področju večciljnega učenja obstaja mnogo raziskav. Za pregled smo izbrali nekaj temeljnih del in del, ki se dotikajo naše teme.

V [1] so obsežneje opisali delovanje večciljnega učenja, motivacijo za njegovo uporabo in njegove prednosti. Večciljno učenje izvedejo z nevronskimi mrežami. Nevronske mreže prilagodijo za večciljno učenje z dodajanjem izhodov iz nevronske mreže, za vsak dodatni ciljni atribut en dodaten izhod. Naloge imajo del notranjih povezav nevronske mreže skupen, tako pride do deljenja informacij med nalogami. Del povezav se med procesom učenja lahko specializira za posamezno nalogo. V kolikšni meri si posamezne naloge delijo notranje povezave, je odvisno tudi od kapacitete nevronske mreže. Delovanje večciljnih nevronskih mrež je preizkušeno na treh testnih domenah, pri čemer primerjajo doseženo napovedno napako z napovedno napako običajnih nevronskih mrež. V prvem primeru preverjajo uspešnost napovedi 9 ciljnih atributov v primerjavi z uspešnostjo učenja vseh ciljnih atributov ločeno. V drugem primeru primerjajo uspešnost napovedi dveh ciljnih atributov, uporabijo pa 10 ciljnih atributov, torej so ostali ciljni atributi uporabljeni le za izboljšanje napovedi dveh glavnih ciljnih atributov. V tretjem primeru uporabijo za dodatne ciljne atribute tiste atribute, ki so na voljo le v učni množici, ne pa tudi kasneje pri uporabi algoritma. Podatkovna množica je z medicinskega področja, dodatni ciljni atributi so podatki, ki jih lahko prido-

bijo le z nadaljnjimi preiskavami. Tako so preverjali uspešnost glavne naloge, ostale naloge so le v pomoč pri učenju. Predstavljen je tudi predlog, kako uporabiti večciljno učenje pri gradnji odločitvenih dreves in pri algoritmu K najbližjih sosedov. Vecčciljno drevo, ki je predlagano, je prilagojeno le za klasifikacijo, medtem ko smo v našem delu združili regresijske in klasifikacijske naloge v večciljnih drevesih.

Opisane prednosti večciljnega učenja so manjša časovna in prostorska zahtevnost v primerjavi z gradnjo modelov za vsako nalogo ločeno, algoritem se lahko nauči boljši model tudi z manj podatki, tak model je manj občutljiv na šum enega ciljnega atributa. V primeru, ko imamo v učni množici atribut, ki ni na voljo v testni množici oziroma pri uporabi modela, lahko ta atribut poskusimo uporabiti kot dodatni ciljni atribut. Primer tega je podatkovna množica o bolnikih z določeno boleznijo. V učni množici so podatki o osnovnih in nadaljnjih preiskavah za vsakega bolnika. Novi bolniki sprva pridobijo le podatke iz osnovne preiskave, podatki iz nadaljnjih preiskav pa takrat še niso na voljo. Ob napovedovanju ciljnega atributa za nove bolnike teh podatkov še ne moremo uporabiti kot vhodne atribute, lahko pa si z njimi pomagamo kot z dodatnimi ciljnimi atributi pri učenju in jih naučen model napoveduje. Delo predstavi tudi primer, pri katerem je ob uporabi večciljnih nevronskih mrež nekatere atribute bolje uporabiti kot izhodne atribute namesto kot vhodne.

Pri večciljnem učenju je pomembno, da so vse naloge oz. vsi ciljni atributi med seboj sorodni, saj tako pripomorejo k učenju skupnega splošnega modela. V nasprotnem primeru se lahko napovedna točnost tudi bistveno zmanjša. V primerih, ko nas zanima le napovedna točnost enega ciljnega atributa, lahko poiščemo primerne uteži za ostale naloge glede na sorodnost posamezne naloge z glavno nalogo. Na ta način z utežjo določimo, koliko vsak ciljni atribut prispeva pri izgradnji modela. Na primer glavna naloga dobi največjo utež, saj želimo, da ima največji vpliv pri izgradnji modela, ostale naloge pa dobijo uteži sorazmerne s sorodnostjo teh nalog in glavne naloge.

V [2] združijo koncepta odločitvenih dreves in združevanja v skupine. Vsak list in vsako vozlišče v drevesu lahko obravnavamo kot skupino primerov. Izbrati moramo dve funkciji, prva za vsako skupino izračuna enega predstavnika, imenovanega prototip, druga funkcija pa računa razdaljo med dvema prototipoma in s tem razdaljo med skupinama, katerima pripadata prototipa. Ob izgradnji drevesa se v vsakem vozlišču za razdelitev primerov na podskupine izbere tisti atribut, pri katerem bosta imeli novonastali skupini največjo medsebojno razdaljo. Kot primer uporabe takih dreves opišejo in preizkusijo večciljno klasifikacijsko drevo, ki je po rezultatih primerljivo z napovedovanjem z običajnimi drevesi vsakega ciljnega atributa ločeno.

V [3] predstavijo večciljni ansambelski algoritem, ki temelji na algoritmu *Boosting*. Za šibke klasifikatorje algoritem uporablja večciljna odločitvena drevesa. Osredotočijo se na binarne klasifikacijske probleme. Delovanje preizkusijo na ugotavljanju pripadnosti besedil določeni tematiki. Preizkusijo tudi delovanje z različnimi utežmi za posamezne dodatne naloge. Merijo le natančnost napovedi ciljnega atributa glavne naloge, ostale naloge so uporabljene le kot pomoč glavni nalogi. V našem delu bomo uporabili večciljne ansambelske metode, ki bodo poleg klasifikacijskih reševale tudi regresijske probleme.

V [4] predstavijo večciljno učenje z uporabo večciljnih odločitvenih dreves kot šibkih klasifikatorjev pri algoritmu *Adaboost*. Informacijski prispevek uporabljen pri gradnji dreves priredijo za uporabo pri gradnji večciljnih odločitvenih dreves. Osredotočijo se na klasifikacijske probleme.

V [5] prilagodijo ansamble odločitvenih pravil za delovanje na večciljnih problemih. Opisani algoritmi rešujejo le regresijske probleme, ne pa tudi klasifikacijskih. Predlagan algoritem doseže boljšo napovedno točnost kot večciljna regresijska drevesa, a slabšo kot večciljni naključni gozd.

V [6] se osredotočijo na ugotavljanje podobnosti med regresijskimi problemi, ki jih lahko uporabimo skupaj pri večciljni regresiji za izboljšanje natančnosti napovedi. Podobnost med nalogami ocenijo s pomočjo kovariance med nalogami. V našem delu bomo predstavili način merjenja podobnosti

med nalogami, ki bo deloval tudi na mešanih regresijskih in klasifikacijskih problemih v večciljnih drevesih.

V praksi pri večciljnih podatkovnih množicah ne moremo pričakovati, da bodo vse naloge med seboj sorodne. Za boljše rezultate se lahko naloge razvrsti v skupine po sorodnosti in nato le naloge znotraj skupine uporabljajo skupen model znanja. V [7] to storijo za večciljne nevronske mreže, v [8] pa za metodo SVM. V obeh primerih za iskanje skupin uporabijo notranje stanje uporabljenega algoritma. V [9] združujejo v skupine naloge pred uporabo algoritma k najbližjih sosedov. V tem delu bomo preizkusili način iskanja sorodnih nalog in združevanja nalog v skupine, ki bo deloval v uporabljenih drevesnih metodah.

Poglavje 3

Drevesne metode

3.1 Odločitvena in regresijska drevesa

Odločitvena in regresijska drevesa so razširjena metoda strojnega učenja. Drevo sestoji iz vozlišč, ki se rekurzivno delijo na podvozlišča. Ko nadaljnja delitev ni več mogoča ali željena se ustvari list, v katerem se izračuna končna napoved. V vsakem vozlišču je pogoj, glede na katerega se primeri razdelijo v eno od podvozlišč. V fazi učenja se zgradi odločitveno drevo, v fazi napovedovanja se sledi vejam glede na pogoje v vozliščih, dokler ne prispemo do lista.

Postopek gradnje drevesa pričnemo v korenskem vozlišču, ki je vstopna točka v drevo. V korenskem vozlišču začnemo z vsemi učnimi primeri ter jih nato delimo po vejah glede na izbrane pogoje v vozliščih. Med gradnjo drevesa ponavljamo rekurzivni postopek, pri katerem primere v vozlišču razdelimo na dve podmnožici, v nekaterih sistemih pa tudi na več podmnožic. S pomočjo učnih primerov poiščemo atribut, ki glede na izbrano mero primere razdeli na najboljši način. Ta atribut se shrani v vozlišču in ga kasneje pri napovedovanju uporabljamo za delitev primerov, ki prispejo v vozlišče. V kolikor je izbran zvezni številski atribut, v naslednjem koraku poiščemo še najprimernejša vrednost, pri kateri se primeri delijo. Za vsako nastalo podmnožico ustvarimo novo vozlišče, kjer se postopek rekurzivno ponovi. V

vsakem vozlišču tudi preverimo, ali je izpolnjen kateri od ustavitvenih pogojev. Če je, vozlišča ne delimo naprej, temveč ga spremenimo v list. S pomočjo učnih primerov v listu izračunamo vrednost, ki bo pri napovedovanju napovedana za primere, ki bodo prispeli v list.

Drevo se lahko pretirano prilagodi učnim podatkom, kar poslabša delovanje na testnih množicah. Po izgradnji drevesa lahko drevo porežemo (angl. pruning), pri tem odstranimo vozlišča, za katera postopek oceni, da povzročajo pretirano prileganje podatkom.

Poleg samostojne uporabe se odločitvena in regresijska drevesa uporabljajo tudi kot osnovni gradniki ansambelskih metod. Ansambelske metode uporabljajo množico nekoliko različnih dreves, ki skupaj napovedujejo končni rezultat in lahko izboljšajo napovedno točnost v primerjavi s posameznim drevesom. Pri ansambelskih metodah ne uporabljamo rezanja dreves in ne omejujemo velikosti drevesa pri izgradnji, saj se problem pretiranega prileganja podatkom rešuje z množico raznolikih dreves.

3.1.1 Klasifikacija

Klasifikacijsko drevo uporabimo takrat, ko pri napovedovanju ciljnega atributa izbiramo med elementi iz omejene množice vrednosti. Možne vrednosti za napovedovanje niso številsko primerljive in jih ne moremo razvrstiti po velikosti ali napovedati vmesnih vrednosti. Pri preverjanju uspešnosti klasifikacijskega algoritma želimo, da algoritem v čim več primerih pravilno izbere vrednost za ciljni atribut.

Pri klasifikaciji se kot mere za izbiro najboljšega atributa za deljenje primerov v vozlišču najpogosteje uporabljajo *gini indeks*, *informacijski prispevek* in *razmerje informacijskega prispevka*.

Gini indeks za množico primerov izračunamo po enačbi (3.1), pri čemer w zavzame vse možne vrednosti ciljnega atributa Y, $p(Y = w)$ pa predstavlja verjetnost, da ima učni primer vrednost Y enako trenutno obravnavani vrednosti w. Z enačbo (3.2) izberemo atribut, ki doseže najmanjšo uteženo vsoto *gini* vrednosti za vse podmnožice primerov, ki nastanejo pri delitvi

primerov z izbranim atributom. n je število primerov v vozlišču.

$$Gini(Y) = 1 - \sum_{w \in values(Y)} p(Y = w)^2 \tag{3.1}$$

$$\arg\min_{A} \left(\sum_{v \in values(A)} \left(\frac{|\{X_A = v\}|}{n} \cdot Gini(Y|X_A = v) \right) \right) \tag{3.2}$$

Informacijski prispevek (angl. information gain) primere razdeli na podmnožice z najmanjšo entropijo.

Z enačbo (3.3) izračunamo entropijo ciljnega atributa.

$$H(Y) = - \sum_{v \in values(Y)} p(Y = v) \log_2 p(Y = v) \tag{3.3}$$

Z enačbo (3.4) izračunamo informacijski prispevek za množico primerov Y in atribut A, s katerim razdelimo množico primerov na podmnožice.

$$Gain(Y, A) = H(Y) - \sum_{v \in values(A)} \left(\frac{|\{X_A = v\}|}{n} \cdot H(Y|X_A = v) \right) \tag{3.4}$$

Informacijski prispevek lahko izračuna pretirano dobre rezultate za diskretne atribute z več možnimi vrednostmi. Mera, ki to težavo omili, je *razmerje informacijskega prispevka* (angl. Information Gain Ratio). Z enačbo (3.5) izračunamo korekcijsko vrednost, s katero nato v enačbi (3.6) popravimo vrednost informacijskega prispevka.

$$H(A) = - \sum_{v \in values(A)} \left(\frac{|\{X_A = v\}|}{n} \cdot \log_2 \frac{|\{X_A = v\}|}{n} \right) \tag{3.5}$$

$$GainRatio(A) = \frac{Gain(A)}{H(A)} \tag{3.6}$$

3.1.2 Regresija

Regresija pravimo postopku strojnega učenja, pri katerem je ciljni atribut številska vrednost. Pri gradnji drevesa v vozliščih izberemo atribut, ki primere razdeli na taki podmnožici, da se kar najbolj zmanjša varianca vrednosti

ciljnega atributa primerov znotraj podmnožic.

$$Var(Y) = \frac{1}{|Y|} \sum_{i=1}^{|Y|} (y_i - \overline{y})^2 \tag{3.7}$$

Enačba (3.7) prikazuje izračun variance znotraj množice primerov Y, y_i je vrednost ciljnega atributa i-tega primera znotraj množice, $\overline{y}$ pa povprečna vrednost ciljnega atributa vseh primerov znotraj množice Y. Pri izbiri atributa v vozlišču poiščemo atribut, pri uporabi katerega je utežena vsota varianc nastalih podmnožic kar najmanjša.

3.2 Bagging

Pri metodi bagging [10] uporabimo namesto enega odločitvenega in regresijskega drevesa množico dreves. Pri učenju vsako drevo prejme nekoliko drugačno množico učnih primerov ter posledično zgradi nekoliko drugačno drevo od ostalih dreves v množici. Pri napovedovanju vsa drevesa glasujejo za končno napoved ali pa uporabimo uteženo vsoto porazdelitve ciljnega razreda. Za vsako drevo pred postopkom učenja za N učnih primerov naključno generiramo novo učno množico N primerov, ki so iz prvotne učne množice izbrani naključno z vračanjem. S takim načinom izbiranja je v povprečju v učni množici zajetih $(1 - \frac{1}{e}) = 63.2\%$ primerov iz prvotne učne množice, nekateri izmed njih so vsebovani večkrat. Zaradi nekoliko drugačnih učnih primerov vsako od dreves pri gradnji izbere nekoliko drugačne atribute za delitev primerov v vozliščih. Število uporabljenih dreves pri metodi bagging je običajno okvirno med 50 in 1000. Bagging pomaga zmanjšati prekomerno prilagoditev podatkom. Odločitvena in regresijska drevesa se namreč lahko prekomerno prilagodijo podanim učnim podatkom, saj zaradi mnogih delitev primerov v liste drevesa pride le malo učnih primerov. Model, ki je pretirano prilagojen učnim podatkom, ni dovolj splošen za dani problem in dosega slabše rezultate pri testiranju. Pri baggingu dobi vsako drevo nekoliko drugačno množico učnih primerov. Četudi se posamezno drevo preveč prilagodi svojim učnim podatkom, se pri napovedovanju, ko glasujejo vsa drevesa,

ta pomanjkljivost zmanjša. Drevesa uporabljena za bagging posledično niso porezana in jim ne omejujemo globine izgradnje.

V primerjavi s posameznim odločitvenim in regresijskim drevesom je slabost ansamblov težja razumljivost modela in njegovih odločitev ljudem, saj je množico dreves težje pregledati in razumeti, končna odločitev pa je izbrana na podlagi vseh dreves skupaj. Ker je potrebno zgraditi več dreves, sta večji tudi časovna in prostorska zahtevnost.

3.3 Naključni gozd

Naključni gozd [11] je nadgradnja algoritma bagging. Uporablja enak način sestavljanja učne množice za vsako drevo posebej kot bagging. Namesto običajnih dreves pa uporablja naključna drevesa. Ta pri učenju v vsakem vozlišču ne iščejo najboljšega atributa med vsemi možnimi atributi, pač pa le med naključno generirano podmnožico atributov. Velikost podmnožice atributov, ki jih algoritem v posameznem vozlišču obravnava, običajno izberemo glede na velikost množice vseh atributov $|A|$. Velikost podmnožice atributov v vsakem vozlišču $|A_{sub}|$ je najpogosteje velikosti $|A_{sub}| = \left\lfloor \sqrt{|A|} + 1 \right\rfloor$ ali $|A_{sub}| = \lfloor \log_2(|A|) + 1 \rfloor$. V kolikor je velikost izbrane podmnožice atributov enaka velikosti množice vseh atributov $|A_{sub}| = |A|$, postane algoritem naključnih gozdov enak algoritmu bagging [12].

3.4 Večciljna drevesa

Pri večciljnih drevesih namesto enega ciljnega atributa napovedujemo več ciljnih atributov. Pri gradnji večciljnega drevesa moramo uporabiti prilagojene mere za izbiro najboljšega atributa v vozliščih drevesa, saj želimo, da bo drevo dobro napovedovalo ciljne vrednosti za vse ciljne atribute. Pri do sedaj uporabljenih odločitvenih ali regresijskih večciljnih drevesih so bili vsi ciljni atributi iste vrste, bodisi vsi klasifikacijski bodisi vsi regresijski. Kadar so vsi ciljni atributi iste vrste, lahko pri izbiri atributa v vozlišču izračunamo pov-

prečno vrednost mere preko vseh nalog in izberemo skupno najboljši atribut. V tem delu poskušamo združiti oba načina, tako da se lahko uporabljajo tudi klasifikacijske in regresijske naloge mešano znotraj ene podatkovne množice.

3.5 Večciljni ansambli

Bagging in naključni gozd pri večciljnih problemih delujeta podobno kot pri enociljnih problemih, le da so drevesa, ki jih uporabimo, večciljna. Vsako drevo se zgradi z večciljnim odločanjem v vozliščih. Pri napovedih se za vsako nalogo pri klasifikaciji izbere najpogostejše napovedana vrednost vseh dreves ali utežena vsota porazdelitev ciljnih razredov, pri regresiji pa povprečna vrednost napovedi vseh dreves ali utežena srednja vrednost.

Poglavje 4

Pristop z rangiranjem ocen atributov

4.1 Regresijska in klasifikacijska večciljna drevesa

Večciljna drevesa običajno delujejo bodisi na podatkovnih množicah z vsemi klasifikacijskimi nalogami bodisi na podatkovnih množicah z vsemi regresijskimi nalogami, na pa tudi na podatkovnih množicah z nalogami obeh vrst. Pri klasifikaciji in regresiji se namreč uporabljajo različne mere za izbiro atributov, ki pa niso primerljive med seboj. Zato v tem delu predlagamo pristop, kjer za vsako nalogo, ne glede na to ali je klasifikacijska ali regresijska, atribute rangiramo po pripadajoči meri za izbiro atributov. Atribut, ki bi bil v enociljnem učenju v takem vozlišču izbran kot najboljši, je na prvem mestu, drugi najboljši na drugem mestu in tako naprej. Tako dobimo za vsako nalogo vrstni red za vse atribute. Nato združimo vse uvrstitve posameznega atributa preko vseh nalog znotraj podatkovne množice, torej seštejemo vsa dosežena mesta atributa in kot najboljšega za vse naloge skupaj izberemo tistega, ki je najboljši v skupnem seštevku.

Preprost primer rangiranja atributov je prikazan v tabeli 4.1. V zgornjem delu tabele so zapisane uvrstitve atributov na posameznih nalogah glede na

dosežene vrednosti mere za izbiro atributa v vozlišču. V spodnjem delu sledita vsota vseh uvrstitev in vrstni red glede na doseženo vsoto vseh uvrstitev. Kot skupno najboljši atribut bi bil v tem primeru izbran atribut 2, saj doseže najmanjšo vsoto uvrstitev.

Tabela 4.1: Oris delovanja algoritma za izbiro atributov v večciljnih drevesih z rangiranjem.

	Atribut 1	Atribut 2	Atribut 3	Atribut 4	Atribut 5
Rangiranje pri 1. nalogi	2.	1.	4.	5.	3.
Rangiranje pri 2. nalogi	3.	2.	1.	4.	5.
Rangiranje pri 3. nalogi	3.	1.	2.	5.	4.
Vsota uvrstitev	8	4	7	14	12
Končno mesto	3.	1.	2.	5.	4.

Enačba (4.1) prikazuje način izbire atributa po rangiranju, iščemo torej atribut z najmanjšo vsoto rangiranj. w_i je utež posamezne naloge, ki jo uporabimo, kadar želimo naloge različno utežiti.

$$\arg\min_{A} \left(\sum_{i \in naloge} (w_i \times Rang(A, i)) \right) \tag{4.1}$$

Izbran atribut se uporabi v vozlišču za delitev primerov. V kolikor gre za numerični atribut, je potrebno izbrati še vrednost, pri kateri se primeri razdelijo v dve skupini. Postopek je podoben kot pri izbiri atributov, le da namesto atributov rangiramo možna mesta delitve za atribut.

Pri enociljnih drevesih pri napovedovanju novega primera sledimo vozliščem glede na vrednosti atributov tega primera, dokler ne pridemo do lista. V listu končno napoved pri klasifikacijskih problemih najpreprosteje izračunamo tako, da poiščemo najpogostejšo vrednost ciljnega atributa med vsemi učnimi primeri, ki so bili razvrščeni v ta list. Pri regresiji za končno napoved izračunamo povprečno vrednost napovedi učnih primerov, ki so bili razvrščeni v ta list. Pri večciljnih problemih lahko preprosto razširimo ta postopek in končno vrednost za posamezen ciljni atribut izračunamo enako

kot pri enociljnem načinu, saj je napovedana vrednost za posamezno nalogo neodvisna od ostalih nalog. Razlika je le v tem, da postopek izračuna napovedane vrednosti ponovimo za vsako nalogo.

Pri ustavitvenih pogojih, ki določajo, v katerih primerih vozlišče postane list, sta uporabni omejitvi najmanjšega števila učnih primerov v vozlišču in največje globine drevesa. Ti dve meri namreč nista odvisni od števila nalog. Za uporabo naprednejših metod bi morali zasnovati sistem združevanja mer med več nalogami in med nalogami različnega tipa, torej med regresijskimi in klasifikacijskimi nalogami. Pri ansambelskih metodah se teh omejitev ne uporablja, tako da za nas niso pomembne.

Podobno velja za rezanje dreves (angl. pruning), ki ga nismo uporabljali pri odločitvenem in regresijskem drevesu, saj običajne metode za rezanje dreves delujejo na enociljnih problemih in bi jih bilo potrebno prilagoditi za večciljno delovanje. Pri ansambelskih metodah rezanja dreves ne uporabljamo.

4.2 Mere za primerjavo večciljnih algoritmov

Pri podatkovnih množicah s samo regresijskimi ali samo klasifikacijskimi nalogami lahko za primerjavo različnih algoritmov izračunamo povprečno uspešnost algoritma preko vseh nalog in primerjamo te vrednosti. Pri uporabi večciljnih dreves na podatkovnih množicah, v katerih se nahajajo tako regresijske kot klasifikacijske naloge, vrednosti ne moremo povprečiti, saj mere za ocenjevanje uspešnosti za klasifikacijske naloge in za regresijske naloge niso neposredno primerljive. Za primerjanje uspešnosti večciljnih algoritmov zato algoritme primerjamo med seboj znotraj vsake naloge in jih rangiramo po uspešnosti. Nato seštejemo za vsak algoritem njegove uvrstitve preko vseh nalog in na koncu algoritme razvrstimo glede na vsoto doseženih uvrstitev. Preprost primer je prikazan v tabeli 4.2. Najboljšo skupno uvrstitev preko vseh nalog v tem primeru doseže algoritem 2.

Za preverjanje delovanja algoritmov na regresijskih nalogah uporabimo

Tabela 4.2: Oris primerjave večciljnih algoritmov.

	Algoritem 1	Algoritem 2	Algoritem 3	Algoritem 4
Uvrstitve za 1. nalogo	3.	2.	1.	4.
Uvrstitve za 2. nalogo	2.	1.	3.	4.
Uvrstitve za 3. nalogo	1.	2.	4.	3.
Vsota uvrstitev	6	5	8	11
Končno mesto	2.	1.	3.	4.

meri relativna srednja kvadratna napaka in relativna srednja absolutna napaka.

Relativna srednja kvadratna napaka (angl. Relative Mean Squared Error - RMSE) za vsak testni primer izmeri kvadrat razlike med napovedano vrednostjo in pravilno vrednostjo ciljnega atributa. Sešteje izračunane kvadrate razlik za vse testne primere in jih deli z vsoto kvadratov razlik med povprečno vrednostjo ciljnih atributov v vseh učnih primerih in ciljnimi atributi testnih primerov. Dobljeno številsko vrednost na koncu še koreni, da dobimo napako izraženo v istih enotah kot ciljno spremenljivko. Enačba (4.2) prikazuje izračun napake. N je število testnih primerov, $\hat{y}_i$ je z algoritmom napovedana vrednost za i-ti testni primer, y_i je pravilna vrednost ciljnega atributa za i-ti testni primer in $\bar{y}$ je povprečna vrednost ciljnega atributa v učnih primerih.

$$RMSE(\hat{y}, y) = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{N} (\bar{y} - y_i)^2}} \tag{4.2}$$

Relativna srednja absolutna napaka (angl. Relative Mean Absolute Error - RMAE) za vsak testni primeri izmeri absolutno razliko med napovedano vrednostjo in pravilno vrednostjo ciljnega atributa. Sešteje izračunane absolutne razlike za vse testne primere in jih deli z vsoto absolutnih razlik med povprečno vrednostjo ciljnih atributov v vseh učnih primerih in ciljnimi atributi testnih primerov. Enačba (4.3) prikazuje izračun napake. N je število testnih primerov, $\hat{y}_i$ je z algoritmom napovedana vrednost za i-ti testni pri-

mer, y_i je pravilna vrednost ciljnega atributa za i-ti testni primer in $\bar{y}$ je povprečna vrednost ciljnega atributa v učnih primerih.

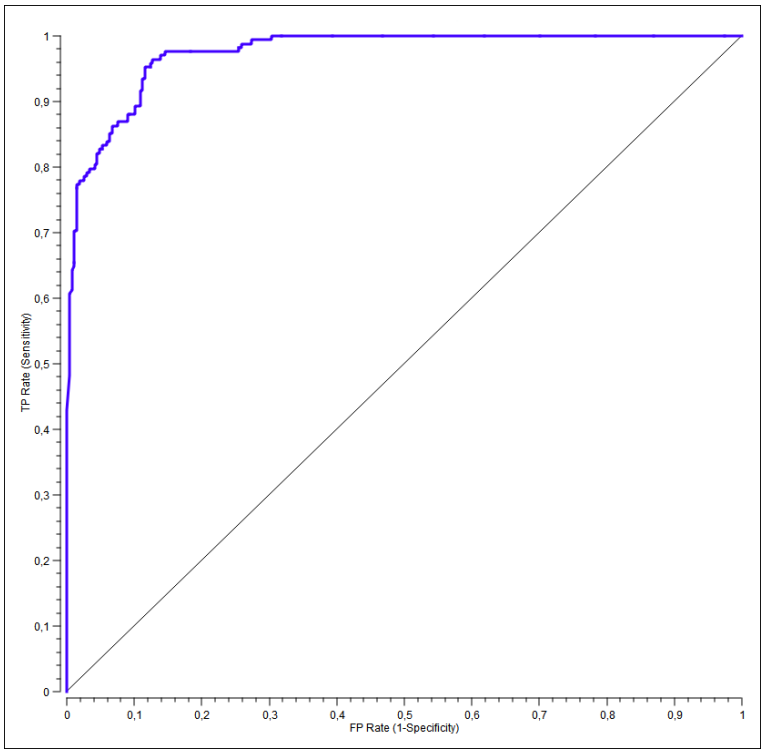
$$RMAE(\hat{y}, y) = \frac{\sum_{i=1}^{N} |\hat{y}_i - y_i|}{\sum_{i=1}^{N} |\bar{y} - y_i|} \tag{4.3}$$

Za preverjanje delovanja algoritmov na klasifikacijskih nalogah uporabimo meri klasifikacijska točnost in AUC. Klasifikacijska točnost predstavlja delež pravilnih napovedi in jo izračunamo kot število testnih primerov, za katere je algoritem pravilno napovedal vrednost, deljeno s številom vseh testnih primerov. Enačba (4.4) prikazuje izračun klasifikacijske točnosti. N je število testnih primerov, $\hat{y}_i$ je z algoritmom napovedana vrednost za i-ti testni primer, y_i je pravilna vrednost ciljnega atributa za i-ti testni primer.

$$točnost(\hat{y}, y) = \frac{\sum_{i=1}^{N} 1\,(\hat{y}_i = y_i)}{N} \tag{4.4}$$

Druga uporabljena mera za klasifikacijske naloge je površina pod krivuljo (angl. Area Under ROC Curve - AUC). Pri algoritmih, ki poleg same napovedi podajo tudi zanesljivost izbranega razreda, lahko izračunamo površino pod krivuljo ROC. Krivulja ROC ima na x osi delež lažnih pozitivnih primerov (angl. False Positive - FP) na intervalu med 0 in 1, na y osi pa delež resnično pozitivnih primerov (angl. True Positive - TP) na intervalu med 0 in 1. Krivulja nam pokaže, kako se deleža resnično pozitivnih in lažno pozitivnih primerov spreminjata pri različnih mejnih vrednostih za klasifikacijo. Željena je čim večja površina pod krivuljo, saj to pomeni veliko število resnično pozitivnih primerov v primerjavi s številom lažno pozitivnih primerov.

Slika 4.1 prikazuje primer krivulje ROC.



Slika 4.1: Primer krivulje ROC.

4.3 Izvedba

Opisane algoritme smo implementirali v orodju za strojno učenje Weka [13]. Weka je razširjeno odprto-kodno orodje za strojno učenje napisano v programskem jeziku Java. Ponuja interakcijo preko grafičnega vmesnika ali preko programske kode. Za podatkovne množice uporablja datoteke tipa *arff*. Glavna prednost Weke v primerjavi z orodjema Orange in Scikit-learn je v tem, da so algoritmi v celoti implementirani v enem programskem jeziku. Razvoj večciljnega odločitvenega in regresijskega drevesa smo začeli iz obstoječega enociljnega drevesa, prav tako sta v Weki že implementirani enociljni različici bagginga in naključnega gozda. Orodje nam olajša tudi obdelovanje vhodnih podatkov in analizo dobljenih rezultatov.

V Weki je implementiranih nekaj različic odločitvenih dreves. Med njimi smo poiskali takšno, ki podpira tako klasifikacijo kot regresijo in ki bi bilo primerno za izhodiščno točko nadaljnje razširitve. Izbrali smo regresijsko in klasifikacijsko odločitveno drevo v Weki poimenovano *REPTree*, ki je opisano kot hitro odločitveno drevo, saj numerične vrednosti atributov sortira le enkrat, pred gradnjo drevesa. Sortirane vrednosti numeričnih atributov se uporabijo pri iskanju najboljšega mesta razcepa atributa v vozlišču. Med gradnjo drevesa v vozlišča na nižjih nivojih pridejo le učni primeri, ki so bili tja razvrščeni na višjih nivojih vozlišč. Vrstni red vrednosti numeričnih atributov se pri tem ne spremeni, tako da sortiranje v nadaljnjih vozliščih ni potrebno. *REPTree* za ocenjevanje atributov za klasifikacijo uporablja informacijski prispevek, za regresijo pa varianco. Poimenovan je po načinu rezanja drevesa imenovanem *reduced-error pruning (with backfitting)*. Rezanja dreves ne uporabljamo, saj ni primerno za ansambelske metode, poleg tega bi bilo potrebno rezanje dreves prilagoditi za večciljna drevesa.

Večje spremembe drevesa so potrebne pri izbiri atributov, ki pri nas temelji na rangiranju atributov, ter pri izbiri mest razcepa za numerične atribute, ki prav tako potrebuje rangiranje. Spremembe so potrebne tudi pri sprejemanju vhodnih podatkov, saj podatkovne množice vsebujejo več ciljnih atributov. Nekatere ustavitvene pogoje, ki so bili vezani na en ciljni

atribut, odstranimo, saj za nas niso bistveni in bi jih morali pred uporabo prilagoditi za večciljni algoritem. Popravki so potrebni tudi pri napovedovanju vrednosti, saj mora algoritem napovedati vrednosti za vse ciljne atribute obenem. Bagging v Weki uporablja *REPTree* drevesa kot osnovne prediktorje. Za večciljno delovanje prilagodimo bagging implementacijo za uporabo večciljnih dreves kot osnovnih prediktorjev. Naključni gozd še nekoliko razširi bagging ter namesto odločitvenih dreves *REPTree* uporablja odločitvena drevesa *RandomTree*, ki pri izgradnji uporabijo naključno podmnožico atributov za iskanje najboljšega atributa v vozlišču. Za implementacijo večciljnega naključnega gozda smo omenjeno delovanje drevesa *RandomTree* prenesli v naše večciljno drevo. Razlika pri gradnji večciljnega naključnega drevesa v primerjavi z običajnim večciljnim drevesom je v tem, da obravnavamo in rangiramo le naključno izbrano podmnožico vseh atributov ter izberemo najboljši atribut iz te podmnožice. Tako prilagojena drevesa smo uporabili kot osnovni prediktor naključnih gozdov.

Za primerjavo implementiranih večciljnih algoritmov z večciljnimi nevronskimi mrežami uporabimo njihovo implementacijo v orodju Orange. Omenjena implementacija večciljnih nevronskih mrež ne omogoča uporabe mešanih nalog, zato primerjamo delovanje le na podatkovnih množicah z enakim tipom nalog. Za primerjavo implementiranih večciljnih dreves z običajnimi večciljnimi drevesi uporabimo njihovo implementacijo v orodju Scikit-learn. V obeh primerih uporabimo privzete nastavitve parametrov. Testiranje poteka tako, da najprej v Weki pripravimo učne in testne množice za 5x2 testiranje in jih shranimo na disk. V omenjenih dveh orodjih napovemo vrednosti za testne primere in jih shranimo v datoteko. Preverjanje vseh rezultatov se nato izvaja v Weki.

4.4 Sorodne naloge

Enociljni algoritem se pri učenju popolnoma osredotoči na eno nalogo, vse izbire atributov v vozliščih in izbire mest razcepa pri zveznih atributih delujejo

z namenom najboljšega rezultata pri eni nalogi. Pri večciljnem algoritmu je namen poiskati nek kompromisen model za več nalog hkrati. Če se naloge med seboj preveč razlikujejo, se rezultati za vse naloge poslabšajo. Ena od možnih razlag slabšega delovanja večciljnega algoritma na nekaterih podatkovnih množicah je ta, da si naloge med seboj niso dovolj podobne in druga drugi slabšajo rezultate. Ker naš večciljni algoritem deluje na klasifikacijskih in regresijskih nalogah mešano z uporabo rangiranja, se pri rangiranju izgubi del informacije o uspešnosti atributov. Posledično pri nalogah, ki so si preveč različne, uspešnost hitreje pade pod nivo enociljnih algoritmov.

Z merjenjem sorodnosti med nalogami znotraj podatkovne množice bi lahko poiskali naloge, ki se jih splača obravnavati skupaj. V podatkovni množici ima vsaka naloga svoj stolpec s ciljnimi atributi. Neposredna primerjava ciljnih atributov ni mogoča. Ciljni atributi imajo lahko različne vrednosti, so različnega tipa - klasifikacijski ali regresijski, četudi sta nalogi podobni. Potrebujemo mero, s katero bomo lahko primerjali tudi klasifikacijske naloge z regresijskimi, saj v nasprotnem primeru ne bi mogli primerjati vseh nalog. Pri večciljnih algoritmih smo za vsako nalogo rangirali atribute v vsakem vozlišču. Namen uporabe rangiranja atributov pri izgradnji drevesa je bila neodvisnost od tipa nalog. Podobno velja tudi pri iskanju sorodnosti, dve nalogi sta lahko sorodni, četudi sta različnega tipa. Od sorodnih nalog pričakujemo, da podobno rangirajo atribute, torej lahko podatke o tem, kako naloge rangirajo atribute, uporabimo tudi za primerjavo sorodnosti med njimi.

Za pridobitev podatkov o sorodnosti nalog najprej zgradimo večciljno odločitveno drevo, pri katerem pa nas ne zanima končna napoved, želimo zbrati le podatke, kako naloge rangirajo atribute. Zabeležimo razvrstitve atributov v nekaj zgornjih nivojih drevesa. Tako dobimo dovolj podatkov za primerjavo. V korenskem vozlišču so atributi rangirani glede na delitev vseh učnih primerov, nižje kot gremo v drevesu, manj primerov pride do vozlišča in bolj rangiranje atributov postaja odvisno od delitev v višjih vozliščih. Delitve v višjih vozliščih predpostavljajo, da so vse naloge enako podobne. Ker je

naš cilj ugotoviti podobnost nalog, je smiselno upoštevati rangiranja v višjih vozliščih. Razliko med rangiranji lahko merimo z absolutno razdaljo med uvrstitvami atributov. Bolj ko sta rangiranji podobni, manjša bo absolutna razdalja med njima.

S pomočjo mere razdalje med nalogami lahko naloge znotraj podatkovne množice združimo v skupine in pri uporabi večciljnega algoritma skupaj obravnavamo le naloge znotraj skupine. V tem primeru lahko pričakujemo boljše rezultate kot pri uporabi vseh nalog, saj so si naloge znotraj skupine bolj podobne kot vse naloge skupaj.

Uporabljene mere za izbiro atributov v drevesih so kratkovidne, zato izračunamo še razdalje med rangiranji atributov pri uporabi mere za ocenjevanje atributov Relief [14]. Za klasifikacijske naloge je namenjena različica ReliefF in za regresijske RReliefF, uporabimo njuno implementacijo v Weki. Za vsako nalogo rangiramo atribute glede na vrednost mere Relief in izračunamo absolutno razdaljo med rangiranji atributov pri nalogah.

Poglavje 5

Poskusi

S poskusi primerjamo uspešnost razvitih večciljnih algoritmov v primerjavi z običajnimi enocliljnimi algoritmi. Večciljni algoritmi zgradijo model za vse naloge naenkrat in lahko podajo skupno napoved za vse naloge z enim odločitvenim drevesom oziroma ansamblom, medtem ko enociljni algoritmi za vsak ciljni atribut zgradijo ločeno drevo oziroma ansambel. Preizkušeni večciljni algoritmi so večciljno regresijsko in odločitveno drevo (v tabelah rezultatov krajše imenovan *Tree MT*), večciljni bagging (krajše *Bagging MT*) in večciljni naključni gozd (krajše *RF MT*). Uporabljeni enociljni algoritmi so odločitveno in regresijsko drevo (v tabelah rezultatov imenovan *Tree*), bagging in naključni gozd (krajše *RF*).

Za merjenje uspešnosti uporabljamo za regresijske naloge meri relativna srednja kvadratna napaka in relativna srednja absolutna napaka, za klasifikacijske naloge pa klasifikacijsko točnost in AUC. Regresijski meri predstavljata napako, manjša vrednost pomeni boljši rezultat, klasifikacijski pa uspešnost delovanja in večja vrednost pomeni boljši rezultat. Ker pri klasifikaciji in regresiji uporabljamo različne mere in so napake pri različnih nalogah različno velike, za vsako nalogo rangiramo vse algoritme glede na uspešnost pri tej nalogi. Nato primerjamo algoritme glede na skupni seštevek vseh uvrstitev preko vseh nalog za neko podatkovno množico. Tako lahko primerjamo delovanje algoritmov tudi na podatkovnih množicah, ki imajo mešano klasi-

fikacijske in regresijske ciljne atribute.

Na dveh podatkovnih množicah, na katerih implementirani večciljni algoritmi delujejo bolje od enociljnih algoritmov, primerjamo algoritme še z običajnimi večciljnimi drevesi in z večciljnimi nevronskimi mrežami. Običajna večciljna drevesa ne uporabljajo rangiranja atributov in posledično ne delujejo na mešanih klasifikacijskih in regresijskih množicah, zato je mogoča le omejena primerjava. Uporabimo implementacijo večcijnih dreves iz orodja Scikit-learn in implementacijo večciljnih nevronskih mrež iz orodja Orange. Ker so v tem primeru težave s pridobivanjem podatkov o zanesljivosti napovedi, teh ne uporabimo in opravimo le en način merjenja uspešnosti.

5.1 Opis podatkovnih množic

Večina podatkovnih množic, ki se uporabljajo za preizkušanje večciljnih algoritmov, ima vse ciljne atribute istega tipa, bodisi vse regresijske ali vse klasifikacijske. Naši algoritmi delujejo na obeh tipih podatkovnih množic in tudi na podatkovnih množicah z mešanimi regresijskimi in klasifikacijskimi ciljnimi atributi. Nekaj obstoječih podatkovnih množic zato prilagodimo in spremenimo tip delu ciljnih atributov ter tako pripravimo podatkovne množice z mešanimi regresijskimi in klasifikacijskimi ciljnimi atributi.

Podatkovne množice za večciljne probleme pridobimo iz zbirke *Mulan for multitarget regression (MTR)* [15], iz zbirke Dragija Koceva [16] in iz zbirke *UCI Machine Learning Repository*. Nekatere podatkovne množice so bile v obstoječi literaturi najprej uporabljene za regresijsko večciljno napovedovanje, kasneje pa prilagojene še za klasifikacijsko večciljno napovedovanje.

Pri podatkovni množici *Water Quality* [17] napovedujemo prisotnost različnih bioindikatorskih vrst rastlin in živali v rečni vodi. Atributi so količine posameznih kemijskih spojin v vodi in fizikalne lastnosti vode. Indikatorji imajo tri možne vrednosti glede na količino organizmov v vodi: malo, srednje ali veliko. Ciljnih atributov je 14. Prvotno so bili ciljni atributi uporabljeni kot regresijski, a ker zavzemajo le tri možne vrednosti, jih lahko brez bistve-

nih sprememb uporabimo tudi kot klasifikacijske.

Podatkovna množica *Bridges* [18] vsebuje podatke o mostovih v mestu Pittsburgh. Atributi so oznaka reke, nad katero je postavljen most, leto izgradnje, namen, dolžina, število pasov in višina mostu. Ciljni atributi so klasifikacijski: vrsta uporabljenega materiala, razpon mostu, tip mostu, položaj ceste in razpon mostu v primerjavi s širino reke.

Podatkovna množica *Bike Sharing* [19] vsebuje podatke o izposojah koles v nekem mestu. Vsak primer v učni množici predstavlja en dan. Atributi vsebujejo podatke o tem, ali je bil to delovni dan ali ne, podatke o temperaturi, vremenu, vlažnosti, vetru in letnem času. Napovedujemo sledeče regresijske ciljne atribute: število izposoj koles neregistriranih uporabnikov, število izposoj koles registriranih uporabnikov in število izposoj koles vseh uporabnikov skupaj v tem časovnem obdobju.

Podatkovna množica *Atp1d* (angl. The Airline Ticket Price) [20] vsebuje podatke o cenah letalskih kart. Atributi so čas do poleta, dan v tednu na dan poleta ter trenutne in pretekle cene letalskih kart pri različnih ponudnikih. Ciljni atributi so regresijski: najnižje cene letalskih kart za naslednji dan pri štirih različnih letalskih družbah, najnižja cena za katerokoli letalsko družbo za direktni let in najnižja cena za katerokoli letalsko družbo brez omejitve števila vmesnih postankov.

Pri podatkovni množici *Yeast* [21] vsak primer predstavlja en gen pri kvasovkah. Ciljni atributi povedo, ali gen sodeluje pri določeni funkciji, npr. pri metabolizmu, pri tvorjenju beljakovin, rasti celic, deljenju celic in obrambi celic. Vseh funkcij je 14. Posamezen gen lahko sodeluje pri več funkcijah.

V tabeli 5.1 so prikazane lastnosti podatkovnih množic, ki smo jih uporabili za testiranje. Za pridobitev mešanih podatkovnih množic smo delu ciljnih atributov spremenili tip iz regresijskega v klasifikacijskega oz. obratno.

Tabela 5.1: Lastnosti podatkovnih množic.

Ime podatkovne množice	Število primerov	Št. diskretnih atributov	Št. numeričnih atributov	Razredi	Regresijske napovedi
Water Quality	1060	0	16	0	14
Water Quality *mešana*	1060	0	16	7	7
Bridges	108	3	3	5	0
Bridges *mešana*	108	3	3	2	3
Bike Sharing	731	5	7	0	3
Atp1d	337	0	411	0	6
Yeast	2417	0	103	14	0
Yeast *mešana*	2417	0	103	7	7

5.2 Uporabljeni parametri

Za testiranje smo na vsaki podatkovni množici izvedli 5x2 prečno preverjanje (5 kratno ponovitev z delitvijo na učni in testni del v velikosti 50% celotnih podatkov). Rezanja dreves pri odločitvenem in regresijskem drevesu ne uporabljamo, saj metode za rezanje niso prilagojene za večciljna drevesa. Bagging uporabi 50 dreves za učenje in napoved, medtem ko jih naključni gozd uporabi 100. Pri naključnem gozdu se velikost podmnožice atributov izračuna po formuli $|A_{sub}| = \lfloor \log_2(|A|) + 1 \rfloor$, kjer je $|A|$ število atributov podatkovne množice in $|A_{sub}|$ število ocenjenih atributov v vsakem vozlišču. Za enociljne algoritme smo uporabili njihove implementacije v Weki. To so odločitveno drevo *REPTree*, *bagging* z *REPTree* kot osnovnim prediktorjem in naključni gozd z osnovnim prediktorjem *RandomTree*. Za večciljne različice smo uporabljali algoritme izpeljane iz enociljnih, ki smo jih prilagodili za delovanje na večciljnih problemih, kot je opisano v 4. poglavju.

5.3 Rezultati

V tabeli 5.2 so prikazani rezultati različnih algoritmov na podatkovni množici *Water Quality*. V posamezni vrstici so prikazane napake, ki jih različni algo-

Tabela 5.2: Rezultati na podatkovni množici *Water Quality*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	102,07 (6.)	94,56 (4.)	94,15 (3.)	101,74 (5.)	93,82 (2.)	93,40 (1.)
N2 (R)	101,01 (5.)	97,20 (3.)	96,77 (1.)	102,37 (6.)	96,89 (2.)	98,98 (4.)
N3 (R)	96,18 (5.)	89,83 (3.)	89,51 (2.)	97,12 (6.)	87,60 (1.)	90,49 (4.)
N4 (R)	87,01 (6.)	82,63 (1.)	83,00 (2.)	84,42 (4.)	84,53 (5.)	84,30 (3.)
N5 (R)	96,70 (5.)	88,62 (2.)	88,48 (1.)	99,17 (6.)	88,84 (3.)	89,18 (4.)
N6 (R)	84,63 (5.)	79,03 (4.)	78,82 (3.)	88,95 (6.)	78,51 (2.)	77,86 (1.)
N7 (R)	96,74 (5.)	94,09 (2.)	94,03 (1.)	98,67 (6.)	95,77 (3.)	96,08 (4.)
N8 (R)	89,27 (5.)	84,57 (2.)	84,47 (1.)	93,24 (6.)	86,31 (4.)	86,20 (3.)
N9 (R)	86,76 (6.)	79,73 (3.)	80,08 (4.)	81,86 (5.)	76,06 (2.)	73,82 (1.)
N10 (R)	94,57 (6.)	89,88 (3.)	90,03 (4.)	93,09 (5.)	85,38 (2.)	84,06 (1.)
N11 (R)	102,42 (5.)	93,17 (4.)	92,81 (2.)	103,21 (6.)	93,11 (3.)	92,68 (1.)
N12 (R)	86,84 (5.)	80,66 (2.)	80,34 (1.)	89,99 (6.)	82,27 (4.)	81,48 (3.)
N13 (R)	100,83 (5.)	94,32 (1.)	94,34 (2.)	101,98 (6.)	96,28 (4.)	95,64 (3.)
N14 (R)	82,17 (5.)	79,27 (4.)	78,27 (3.)	85,29 (6.)	78,21 (2.)	77,19 (1.)
Vsota uvrstitev	74	38	30	79	39	34
Uvrstitev	5	3	1	6	4	2

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	106,47 (5.)	94,07 (4.)	93,55 (3.)	115,11 (6.)	93,54 (2.)	93,43 (1.)
N2 (R)	111,32 (5.)	98,66 (2.)	98,18 (1.)	122,72 (6.)	98,74 (3.)	100,31 (4.)
N3 (R)	110,45 (5.)	95,29 (2.)	95,18 (1.)	115,82 (6.)	95,76 (3.)	97,45 (4.)
N4 (R)	100,99 (5.)	90,15 (3.)	89,83 (1.)	102,04 (6.)	90,31 (4.)	89,99 (2.)
N5 (R)	102,53 (5.)	92,06 (3.)	91,94 (2.)	110,87 (6.)	92,22 (4.)	91,37 (1.)
N6 (R)	96,29 (5.)	84,67 (2.)	84,13 (1.)	108,81 (6.)	85,10 (4.)	84,95 (3.)
N7 (R)	107,94 (5.)	96,33 (2.)	96,10 (1.)	109,70 (6.)	97,26 (3.)	98,23 (4.)
N8 (R)	104,28 (5.)	91,71 (2.)	91,67 (1.)	114,28 (6.)	92,88 (3.)	92,92 (4.)
N9 (R)	99,41 (5.)	84,70 (4.)	84,56 (3.)	103,87 (6.)	83,17 (2.)	81,12 (1.)
N10 (R)	107,23 (5.)	92,90 (4.)	92,55 (3.)	116,58 (6.)	90,45 (2.)	89,53 (1.)
N11 (R)	108,38 (5.)	93,33 (2.)	92,92 (1.)	116,21 (6.)	93,71 (4.)	93,47 (3.)
N12 (R)	105,05 (5.)	90,99 (2.)	90,46 (1.)	112,63 (6.)	92,14 (4.)	91,83 (3.)
N13 (R)	110,09 (5.)	95,84 (2.)	95,72 (1.)	115,25 (6.)	96,80 (4.)	96,17 (3.)
N14 (R)	96,56 (5.)	86,18 (3.)	85,29 (1.)	107,38 (6.)	86,23 (4.)	85,71 (2.)
Vsota uvrstitev	70	37	21	84	46	36
Uvrstitev	5	3	1	6	4	2

ritmi storijo pri napovedovanju vrednosti ciljnega atributa pripadajoče naloge. Poleg napake je v oklepajih zapisana uvrstitev algoritma pri posamezni nalogi. Vsaka naloga ima v prvem stolpcu v oklepaju zapisano oznako R ali C. R pomeni, da gre za regresijsko nalogo in C pomeni, da gre za klasifikacijsko nalogo.

Predzadnja vrstica prikazuje vsoto vseh doseženih uvrstitev vsakega algoritma, zadnja vrstica pa končno uvrstitev algoritma, pri kateri so algoritmi uvrščeni glede na doseženo skupno vsoto uvrstitev preko vseh nalog na testirani podatkovni množici. V zgornji polovici tabele so rezultati regresijskih nalog merjeni z mero *RMAE*, rezultati klasifikacijskih pa s klasifikacijsko točnostjo. V spodnji polovici tabele so rezultati regresijskih nalog merjeni z mero *RMSE*, rezultati klasifikacijskih pa z mero *AUC*. Pri obeh regresijskih merah pomeni nižja vrednost manjšo napako in s tem boljšo uvrstitev, pri klasifikacijskih pa višja vrednost pomeni boljši rezultat in s tem boljšo uvrstitev pri razvrstitvi algoritmov. Pri merah *RMAE*, *RMSE* in klasifikacijski točnosti so navedene vrednosti pomnožene s 100, zapisane so v odstotkih.

Na podatkovni množici *Water Quality* (tabela 5.2) doseže najboljši skupni rezultat večciljni naključni gozd, sledi enociljni naključni gozd, na tretjem mestu pa je večciljni bagging. Vse večciljne metode dosegajo boljši rezultat od pripadajočih enociljnih. V tem primeru so rezultati enaki pri obeh uporabljenih načinih merjenja uspešnosti algoritmov.

V tabeli 5.3 so prikazani rezultati na prilagojeni podatkovni množici *Water Quality* z mešanimi regresijskimi in klasifikacijskimi ciljnimi atributi. Množico smo predelali tako, da smo vse sode ciljne atribute spremenili v klasifikacijske, lihi pa so ostali regresijski kot v prvotni množici. Tako preizkusimo delovanje tudi na mešano klasifikacijskih in regresijskih podatkovnih množicah. Najbolje uvrščen je večciljni naključni gozd, nadaljnje uvrstitve pa se nekoliko razlikujejo glede na uporabljene mere. V prvem primeru sledita enociljni bagging in večciljni bagging, v drugem pa enociljni naključni gozd in enociljni bagging.

Tabela 5.3: Rezultati na podatkovni množici *Water Quality* z mešanimi regresijskimi in klasifikacijskimi ciljnimi atributi.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	101,54 (5.)	94,97 (4.)	94,48 (2.)	106,58 (6.)	94,56 (3.)	93,85 (1.)
N2 (C)	68,75 (5.)	73,43 (2.)	73,43 (2.)	59,77 (6.)	73,49 (1.)	73,09 (4.)
N3 (R)	94,07 (5.)	90,04 (2.)	90,17 (3.)	99,00 (6.)	88,23 (1.)	91,14 (4.)
N4 (C)	76,09 (5.)	79,73 (2.)	79,73 (2.)	70,60 (6.)	79,79 (1.)	79,35 (4.)
N5 (R)	98,43 (5.)	88,71 (2.)	88,53 (1.)	101,42 (6.)	89,20 (3.)	90,20 (4.)
N6 (C)	47,35 (5.)	52,03 (2.)	52,22 (1.)	43,84 (6.)	51,96 (3.)	50,67 (4.)
N7 (R)	96,74 (5.)	94,03 (2.)	93,19 (1.)	98,16 (6.)	94,53 (3.)	94,59 (4.)
N8 (C)	67,28 (5.)	72,56 (1.)	72,56 (1.)	60,81 (6.)	72,13 (3.)	71,75 (4.)
N9 (R)	90,80 (6.)	80,88 (4.)	80,38 (3.)	84,20 (5.)	76,65 (2.)	74,36 (1.)
N10 (C)	64,03 (5.)	68,45 (3.)	68,45 (3.)	58,64 (6.)	69,22 (2.)	69,54 (1.)
N11 (R)	101,88 (5.)	93,07 (4.)	92,68 (2.)	102,37 (6.)	93,02 (3.)	92,31 (1.)
N12 (C)	65,37 (5.)	68,67 (4.)	68,75 (3.)	61,13 (6.)	69,09 (2.)	69,13 (1.)
N13 (R)	100,83 (5.)	94,89 (2.)	94,50 (1.)	101,42 (6.)	96,59 (4.)	95,75 (3.)
N14 (C)	67,33 (5.)	71,01 (3.)	70,94 (4.)	60,84 (6.)	71,84 (1.)	71,75 (2.)
Vsota uvrstitev	71	37	29	83	32	38
Uvrstitev	5	3	1	6	2	4

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	106,78 (5.)	94,29 (4.)	93,88 (2.)	119,72 (6.)	94,06 (3.)	93,59 (1.)
N2 (C)	0,54 (5.)	0,60 (4.)	0,60 (3.)	0,53 (6.)	0,62 (1.)	0,62 (2.)
N3 (R)	108,20 (5.)	96,02 (2.)	95,85 (1.)	121,04 (6.)	96,87 (3.)	98,09 (4.)
N4 (C)	0,66 (5.)	0,76 (4.)	0,76 (3.)	0,64 (6.)	0,76 (2.)	0,77 (1.)
N5 (R)	105,59 (5.)	91,51 (1.)	91,72 (2.)	115,96 (6.)	92,29 (4.)	92,19 (3.)
N6 (C)	0,70 (5.)	0,78 (4.)	0,79 (3.)	0,67 (6.)	0,79 (2.)	0,79 (1.)
N7 (R)	109,25 (5.)	97,37 (3.)	96,83 (1.)	115,02 (6.)	97,33 (2.)	98,53 (4.)
N8 (C)	0,66 (5.)	0,74 (3.)	0,75 (1.)	0,59 (6.)	0,75 (2.)	0,74 (4.)
N9 (R)	102,89 (5.)	84,97 (4.)	84,39 (3.)	103,93 (6.)	82,86 (2.)	80,80 (1.)
N10 (C)	0,61 (6.)	0,69 (4.)	0,70 (3.)	0,63 (5.)	0,77 (2.)	0,78 (1.)
N11 (R)	107,78 (5.)	93,17 (3.)	92,76 (1.)	115,22 (6.)	93,38 (4.)	92,81 (2.)
N12 (C)	0,67 (5.)	0,76 (4.)	0,77 (2.)	0,64 (6.)	0,77 (3.)	0,78 (1.)
N13 (R)	109,33 (5.)	96,58 (3.)	96,13 (1.)	114,84 (6.)	97,30 (4.)	96,15 (2.)
N14 (C)	0,67 (5.)	0,76 (4.)	0,76 (2.)	0,63 (6.)	0,77 (1.)	0,76 (3.)
Vsota uvrstitev	71	47	28	83	35	30
Uvrstitev	5	4	1	6	3	2

Pri izdelavi mešanih množic lahko izberemo, katere naloge bodo klasifikacijske in katere regresijske. Za temeljitejše preverjanje delovanja na mešanih množicah smo zgradili še tri različice podatkovne množice *Water Quality* z različnimi izbirami klasifikacijskih in regresijskih nalog. V tabelah 5.4, 5.5 in 5.6 so prikazane uvrstitve na teh treh množicah. Za boljšo preglednost v teh treh primerih navajamo le končne uvrstitve algoritmov.

Tabela 5.4: Rezultati na podatkovni množici *Water Quality* s klasifikacijskimi nalogami od N1 do N7 in regresijskimi nalogami N8 do N14.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Vsota uvrstitev	75	30	24	79	43	41
Uvrstitev	5	2	1	6	4	3

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Vsota uvrstitev	74	40	29	80	40	31
Uvrstitev	5	3	1	6	3	2

Tabela 5.5: Rezultati na podatkovni množici *Water Quality* s klasifikacijskimi nalogami N1, N3, N4, N6, N8, N9, N13 in regresijskimi nalogami N2, N5, N7, N10, N11, N12, N14.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Vsota uvrstitev	74	39	25	79	39	37
Uvrstitev	5	3	1	6	3	2

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Vsota uvrstitev	73	44	26	81	37	33
Uvrstitev	5	4	1	6	3	2

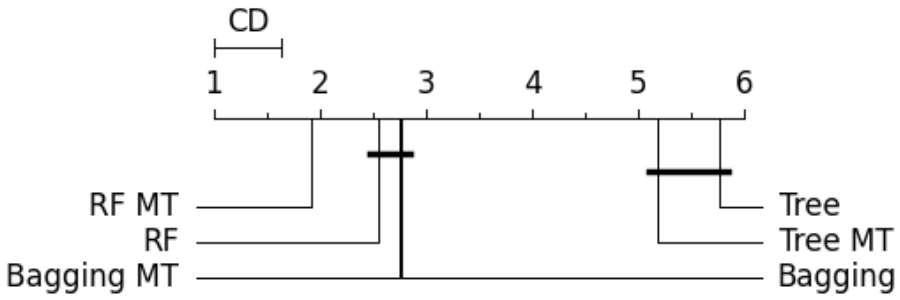
Tabela 5.6: Rezultati na podatkovni množici *Water Quality* s klasifikacijskimi nalogami N1, N2, N5, N7, N8, N11, N14 in regresijskimi nalogami N3, N4, N6, N9, N10, N12, N13.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Vsota uvrstitev	69	33	30	80	38	43
Uvrstitev	5	2	1	6	3	4

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Vsota uvrstitev	74	40	26	80	39	35
Uvrstitev	5	4	1	6	3	2



Slika 5.1: Povprečne uvrstitve algoritmov na petih testnih različicah podatkovne množice *Water Quality* s prikazano kritično razdaljo.

Statistično pomembnost razlik pri skupnih rezultatih na vseh zgoraj navedenih različicah podatkovne množice *Water Quality* preverimo s Friedmanovim testom [22] z Nemenyi post-hoc testom za izračun kritične razdalje pri $\alpha = 0.05$. Za 5 zgoraj omenjenih različic podatkovne množice smo uporabili dva različna načina metrik, v vsaki podatkovni množici se nahaja 14 nalog, torej smo skupno opravili 140 razvrstitev algoritmov. Povprečne uvrstitve algoritmov so prikazane v tabeli 5.7. Slika 5.1 prikazuje povprečne uvrstitve s kritično razdaljo. Pod opisanimi pogoji je večciljni naključni gozd statistično značilno boljši od ostalih algoritmov. Sledijo enociljni naključni gozd, večciljni bagging in enociljni bagging, med katerimi pa ni statistično značilnih razlik.

Tabela 5.7: Povprečne uvrstitve algoritmov na petih testiranih različicah podatkovne množice *Water Quality*.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Povprečna uvrstitev	5.18	2.75	1.91	5.77	2.77	2.56

V tabeli 5.8 so prikazani rezultati implementiranih večciljnih algoritmov v primerjavi z običajnim večciljnim drevesom in z večciljno nevronsko mrežo na podatkovni množici *Water Quality*. Običajno večciljno drevo deluje nekoliko bolje od našega, a slabše od ostalih algoritmov. Razlog boljšega delovanje običajnega večciljnega drevesa od našega je verjetno v tem, da ta ne upora-

blja rangiranja atributov, pri katerem se izgubi vrednost razlik med atributi. Tako doseže malo boljšo skupno vsoto uvrstitev kot naše večciljno drevo. Nevronska mreža se uvrsti med večciljni bagging in večciljni naključni gozd. S testiranjem parametrov nevronske mreže bi verjetno lahko še izboljšali njen rezultat.

Tabela 5.8: Primerjava implementiranih večciljnih algoritmov, običajnega večciljnega drevesa in večciljne nevronske mreže na podatkovni množici *Water Quality*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree MT classic	Neural network
Naloga 1 (R)	100,58 (4.)	94,01 (2.)	93,52 (1.)	125,97 (5.)	96,17 (3.)
Naloga 2 (R)	98,43 (5.)	96,23 (2.)	96,52 (3.)	97,49 (4.)	67,37 (1.)
Naloga 3 (R)	95,57 (4.)	90,21 (3.)	89,93 (2.)	109,68 (5.)	69,45 (1.)
Naloga 4 (R)	89,02 (4.)	83,41 (3.)	83,31 (2.)	106,56 (5.)	60,96 (1.)
Naloga 5 (R)	100,73 (4.)	89,25 (2.)	88,73 (1.)	119,45 (5.)	97,56 (3.)
Naloga 6 (R)	83,28 (3.)	78,88 (1.)	79,00 (2.)	137,69 (5.)	126,24 (4.)
Naloga 7 (R)	96,01 (5.)	92,59 (3.)	92,78 (4.)	81,02 (2.)	72,60 (1.)
Naloga 8 (R)	90,34 (4.)	84,08 (2.)	84,22 (3.)	126,18 (5.)	68,04 (1.)
Naloga 9 (R)	87,35 (3.)	79,96 (2.)	79,92 (1.)	106,87 (4.)	109,23 (5.)
Naloga 10 (R)	90,68 (5.)	89,69 (4.)	89,03 (3.)	71,95 (1.)	71,95 (1.)
Naloga 11 (R)	100,17 (3.)	92,96 (2.)	92,73 (1.)	124,22 (5.)	102,29 (4.)
Naloga 12 (R)	88,46 (5.)	80,48 (4.)	80,29 (3.)	79,96 (2.)	73,61 (1.)
Naloga 13 (R)	102,73 (5.)	94,38 (3.)	94,26 (2.)	99,40 (4.)	78,87 (1.)
Naloga 14 (R)	84,96 (3.)	80,09 (2.)	79,88 (1.)	97,87 (4.)	101,17 (5.)
Vsota uvrstitev	57	35	29	56	32
Uvrstitev	5	3	1	4	2

V tabeli 5.9 so prikazani rezultati različnih algoritmov na podatkovni množici *Bridges*. Najboljši rezultat pri obeh merah doseže večciljni naključni gozd, v prvem primeru sta na drugem mestu enociljni bagging in enociljni naključni gozd, v drugem primeru pa le enociljni naključni gozd.

Tabela 5.9: Rezultati na podatkovni množici *Bridges*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Naloga 1 (C)	85,74 (5.)	86,94 (1.)	86,57 (3.)	83,98 (6.)	85,74 (4.)	86,67 (2.)
Naloga 2 (C)	81,39 (6.)	85,56 (3.)	87,31 (1.)	83,24 (5.)	86,67 (2.)	85,09 (4.)
Naloga 3 (C)	62,50 (6.)	67,50 (4.)	70,83 (1.)	63,33 (5.)	69,72 (3.)	70,37 (2.)
Naloga 4 (C)	61,85 (5.)	68,89 (3.)	72,31 (1.)	61,02 (6.)	71,67 (2.)	66,94 (4.)
Naloga 5 (C)	50,93 (6.)	54,63 (4.)	57,13 (3.)	51,20 (5.)	57,22 (2.)	59,44 (1.)
Vsota uvrstitev	28	15	9	27	13	13
Uvrstitev	6	4	1	5	2	2

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Naloga 1 (C)	0,76 (5.)	0,81 (3.)	0,83 (2.)	0,70 (6.)	0,78 (4.)	0,84 (1.)
Naloga 2 (C)	0,87 (5.)	0,94 (3.)	0,97 (1.)	0,84 (6.)	0,94 (4.)	0,97 (2.)
Naloga 3 (C)	0,64 (5.)	0,79 (2.)	0,83 (1.)	0,59 (6.)	0,79 (3.)	0,76 (4.)
Naloga 4 (C)	0,75 (5.)	0,80 (2.)	0,80 (1.)	0,73 (6.)	0,78 (4.)	0,79 (3.)
Naloga 5 (C)	0,87 (5.)	0,94 (4.)	0,97 (1.)	0,84 (6.)	0,97 (2.)	0,96 (3.)
Vsota uvrstitev	25	14	6	30	17	13
Uvrstitev	5	3	1	6	4	2

V tabeli 5.10 so prikazani rezultati različnih algoritmov na podatkovni množici *Bridges* z mešanimi klasifikacijskimi in regresijskimi nalogami. Najboljši rezultat doseže večciljni naključni gozd, sledi enociljni naključni gozd.

V tabeli 5.11 so prikazani rezultati implementiranih večciljnih algoritmov v primerjavi z običajnim večciljnim drevesom in z večciljno nevronsko mrežo na podatkovni množici *Bridges*. Najboljši rezultat doseže algoritem večciljnih nevronskih mrež, ki doseže malo boljšo skupno vsoto uvrstitev kot večciljni naključni gozd. Ker gre za povsem različna algoritma, bi morali za natančnejšo primerjavo testirati njune rezultate pri optimiziranih parametrih, na primer poiskati primerno velikost nevronske mreže in primerno

Tabela 5.10: Rezultati na podatkovni množici *Bridges* z mešanimi klasifikacijskimi in regresijskimi nalogami.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Naloga 1 (R)	74,85 (1.)	80,05 (4.)	78,96 (3.)	80,08 (5.)	90,17 (6.)	78,26 (2.)
Naloga 2 (C)	83,43 (5.)	86,76 (3.)	87,41 (1.)	82,96 (6.)	86,94 (2.)	85,65 (4.)
Naloga 3 (R)	94,85 (5.)	94,88 (6.)	93,13 (4.)	90,11 (3.)	88,55 (1.)	89,14 (2.)
Naloga 4 (C)	61,85 (5.)	69,17 (2.)	69,81 (1.)	59,26 (6.)	67,69 (3.)	66,20 (4.)
Naloga 5 (R)	79,18 (6.)	70,73 (4.)	68,16 (2.)	75,73 (5.)	70,08 (3.)	65,66 (1.)
Vsota uvrstitev	22	19	11	25	15	13
Uvrstitev	5	4	1	6	3	2

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Naloga 1 (R)	105,36 (5.)	89,20 (2.)	86,31 (1.)	114,99 (6.)	94,64 (4.)	92,05 (3.)
Naloga 2 (C)	0,87 (5.)	0,95 (4.)	0,96 (2.)	0,85 (6.)	0,95 (3.)	0,96 (1.)
Naloga 3 (R)	101,17 (6.)	90,44 (3.)	88,22 (1.)	101,01 (5.)	88,23 (2.)	91,33 (4.)
Naloga 4 (C)	0,74 (5.)	0,78 (3.)	0,79 (2.)	0,73 (6.)	0,79 (1.)	0,76 (4.)
Naloga 5 (R)	84,31 (5.)	69,99 (4.)	66,72 (2.)	84,65 (6.)	69,89 (3.)	66,36 (1.)
Vsota uvrstitev	26	16	8	29	13	13
Uvrstitev	5	4	1	6	2	2

število naključno izbranih atributov pri izgradnji dreves naključnega gozda. Tako pa dobimo le okvirno primerjavo rezultatov. Običajno večciljno drevo je boljše kot naše večciljno drevo, a slabše kot bagging in naključni gozd.

Tabela 5.11: Primerjava implementiranih večciljnih algoritmov, običajnega večciljnega drevesa in večciljne nevronske mreže na podatkovni množici *Bridges*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree MT classic	Neural network
Naloga 1 (C)	84,07 (4.)	86,11 (3.)	87,59 (1.)	82,22 (5.)	87,22 (2.)
Naloga 2 (C)	80,74 (5.)	85,74 (3.)	87,22 (2.)	81,85 (4.)	89,07 (1.)
Naloga 3 (C)	60,74 (5.)	63,89 (3.)	67,04 (1.)	61,11 (4.)	66,30 (2.)
Naloga 4 (C)	61,48 (5.)	67,78 (3.)	70,56 (2.)	62,04 (4.)	70,74 (1.)
Naloga 5 (C)	51,11 (5.)	55,19 (3.)	56,30 (2.)	51,67 (4.)	60,19 (1.)
Vsota uvrstitev	24	15	8	21	7
Uvrstitev	5	3	2	4	1

Tabela 5.12: Rezultati na podatkovni množici *Bike Sharing*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	62,34 (6.)	48,03 (5.)	45,61 (3.)	46,49 (4.)	38,59 (2.)	37,14 (1.)
N2 (R)	47,05 (6.)	36,15 (4.)	35,11 (2.)	42,20 (5.)	35,74 (3.)	32,52 (1.)
N3 (R)	43,05 (6.)	33,72 (3.)	33,59 (2.)	41,22 (5.)	34,44 (4.)	31,43 (1.)
Vsota uvrstitev	18	12	7	14	9	3
Uvrstitev	6	4	2	5	3	1

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	71,06 (6.)	52,80 (4.)	50,09 (3.)	54,24 (5.)	45,03 (2.)	43,27 (1.)
N2 (R)	53,98 (6.)	42,11 (3.)	40,74 (2.)	50,71 (5.)	42,34 (4.)	38,70 (1.)
N3 (R)	50,35 (6.)	40,18 (3.)	39,88 (2.)	47,98 (5.)	40,45 (4.)	37,58 (1.)
Vsota uvrstitev	18	10	7	15	10	3
Uvrstitev	6	3	2	5	3	1

Tabela 5.12 prikazuje rezultate na podatkovni množici *Bike Sharing*. Najbolje deluje enociljni naključni gozd, sledi večciljni naključni gozd. Vsi enociljni algoritmi so uvrščeni pred ali na enako mesto kot pripadajoči večciljni algoritmi pri obeh načinih merjenja rezultatov.

V tabeli 5.13 so prikazani rezultati na podatkovni množici *Atp1d*. Najbolje se odrežeta enociljni naključni gozd in enociljni bagging.

Tabela 5.13: Rezultati na podatkovni množici *Atp1d*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	54,18 (5.)	44,16 (3.)	45,57 (4.)	54,50 (6.)	42,74 (2.)	42,13 (1.)
N2 (R)	50,18 (6.)	41,03 (3.)	42,45 (5.)	41,28 (4.)	35,78 (1.)	37,54 (2.)
N3 (R)	44,55 (6.)	36,95 (2.)	37,54 (4.)	41,94 (5.)	37,10 (3.)	35,65 (1.)
N4 (R)	43,06 (6.)	33,47 (4.)	34,76 (5.)	23,13 (1.)	24,06 (2.)	30,08 (3.)
N5 (R)	56,87 (6.)	47,38 (3.)	49,30 (4.)	53,85 (5.)	44,22 (2.)	43,50 (1.)
N6 (R)	42,36 (6.)	32,70 (4.)	33,86 (5.)	19,31 (2.)	19,16 (1.)	28,27 (3.)
Vsota uvrstitev	35	19	27	23	11	11
Uvrstitev	6	3	5	4	1	1

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	60,09 (5.)	48,86 (3.)	49,28 (4.)	63,88 (6.)	47,97 (2.)	47,33 (1.)
N2 (R)	61,05 (6.)	47,78 (3.)	48,61 (4.)	55,06 (5.)	43,57 (1.)	44,78 (2.)
N3 (R)	54,20 (5.)	44,21 (3.)	43,87 (2.)	55,52 (6.)	45,60 (4.)	43,33 (1.)
N4 (R)	54,74 (6.)	40,00 (5.)	38,40 (4.)	31,58 (2.)	26,74 (1.)	34,68 (3.)
N5 (R)	63,47 (6.)	50,19 (3.)	50,32 (4.)	61,54 (5.)	48,75 (2.)	47,27 (1.)
N6 (R)	55,75 (6.)	40,78 (5.)	39,13 (4.)	26,74 (2.)	23,12 (1.)	33,98 (3.)
Vsota uvrstitev	34	22	22	26	11	11
Uvrstitev	6	3	3	5	1	1

Tabela 5.14: Rezultati na podatkovni množici *Yeast*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (C)	68,18 (5.)	74,47 (3.)	70,31 (4.)	67,49 (6.)	77,11 (2.)	77,50 (1.)
N2 (C)	56,69 (6.)	60,37 (3.)	59,83 (4.)	56,83 (5.)	62,92 (2.)	64,04 (1.)
N3 (C)	61,80 (6.)	70,54 (3.)	67,01 (4.)	62,57 (5.)	72,06 (2.)	72,37 (1.)
N4 (C)	66,86 (5.)	71,98 (3.)	71,45 (4.)	65,54 (6.)	73,47 (2.)	74,13 (1.)
N5 (C)	70,05 (5.)	76,09 (3.)	75,67 (4.)	68,23 (6.)	77,95 (2.)	78,84 (1.)
N6 (C)	70,91 (5.)	76,92 (3.)	75,75 (4.)	67,86 (6.)	77,68 (2.)	77,74 (1.)
N7 (C)	78,75 (5.)	83,50 (3.)	82,69 (4.)	75,40 (6.)	83,75 (1.)	83,67 (2.)
N8 (C)	74,61 (5.)	81,20 (1.)	80,67 (4.)	72,24 (6.)	81,12 (2.)	81,12 (2.)
N9 (C)	90,45 (5.)	92,91 (1.)	92,91 (1.)	85,97 (6.)	92,91 (1.)	92,91 (1.)
N10 (C)	85,62 (5.)	89,48 (2.)	89,48 (2.)	82,25 (6.)	89,48 (2.)	89,61 (1.)
N11 (C)	84,18 (5.)	88,02 (2.)	88,00 (4.)	79,80 (6.)	88,02 (2.)	88,11 (1.)
N12 (C)	66,43 (5.)	75,03 (2.)	75,04 (1.)	64,98 (6.)	74,91 (4.)	74,95 (3.)
N13 (C)	65,96 (5.)	74,38 (1.)	74,38 (1.)	64,32 (6.)	74,27 (4.)	74,33 (3.)
N14 (C)	98,19 (5.)	98,48 (1.)	98,48 (1.)	97,24 (6.)	98,48 (1.)	98,48 (1.)
Vsota uvrstitev	72	31	42	82	29	20
Uvrstitev	5	3	4	6	2	1

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (C)	0,62 (6.)	0,75 (4.)	0,76 (3.)	0,63 (5.)	0,78 (2.)	0,79 (1.)
N2 (C)	0,56 (5.)	0,65 (4.)	0,66 (3.)	0,56 (6.)	0,67 (2.)	0,69 (1.)
N3 (C)	0,62 (5.)	0,77 (4.)	0,78 (3.)	0,61 (6.)	0,79 (2.)	0,80 (1.)
N4 (C)	0,65 (5.)	0,77 (4.)	0,77 (3.)	0,63 (6.)	0,79 (2.)	0,80 (1.)
N5 (C)	0,63 (5.)	0,76 (4.)	0,76 (3.)	0,63 (6.)	0,77 (2.)	0,78 (1.)
N6 (C)	0,59 (5.)	0,71 (4.)	0,71 (3.)	0,57 (6.)	0,71 (2.)	0,73 (1.)
N7 (C)	0,59 (5.)	0,72 (3.)	0,72 (4.)	0,58 (6.)	0,72 (2.)	0,74 (1.)
N8 (C)	0,57 (5.)	0,68 (2.)	0,68 (4.)	0,56 (6.)	0,68 (3.)	0,70 (1.)
N9 (C)	0,51 (5.)	0,60 (1.)	0,59 (2.)	0,50 (6.)	0,56 (4.)	0,59 (3.)
N10 (C)	0,53 (6.)	0,64 (4.)	0,66 (3.)	0,55 (5.)	0,66 (2.)	0,67 (1.)
N11 (C)	0,54 (5.)	0,64 (4.)	0,65 (1.)	0,53 (6.)	0,64 (3.)	0,65 (2.)
N12 (C)	0,53 (5.)	0,62 (2.)	0,62 (4.)	0,52 (6.)	0,62 (3.)	0,63 (1.)
N13 (C)	0,54 (5.)	0,62 (3.)	0,62 (4.)	0,54 (6.)	0,63 (2.)	0,63 (1.)
N14 (C)	0,52 (5.)	0,60 (3.)	0,63 (2.)	0,47 (6.)	0,56 (4.)	0,64 (1.)
Vsota uvrstitev	72	46	42	82	35	17
Uvrstitev	5	4	3	6	2	1

V tabeli 5.14 so prikazani rezultati na podatkovni množici *Yeast*. Najboljši rezultat doseže enociljni naključni gozd, sledi enociljni bagging.

V tabeli 5.15 so prikazani rezultati na podatkovni množici *Yeast* z mešanimi klasifikacijskimi in regresijskimi ciljnimi atributi. Lihi ciljni atributi so bili pretvorjeni v regresijske, medtem ko so sodi ciljni atributi ostali klasifikacijski. Najboljši rezultat dosežeta enociljni naključni gozd in enociljni bagging.

Na podatkovnih množicah *Water Quality* in *Bridges* večciljni algoritmi delujejo bolje kot enociljni, pri ostalih obravnavanih podatkovnih množicah pa so boljši enociljni algoritmi. V nekaterih primerih so bagging metode boljše od naključnega gozda, kar je deloma lahko posledica relativno velikega števila dreves uporabljenega pri baggingu, pri naključnem gozdu namreč uporabimo dvakrat toliko dreves kot pri baggingu. V nadaljevanju bomo skušali poiskati razloge za slabše delovanje večciljnih algoritmov na nekaterih podatkovnih množicah in izboljšati rezultate vsaj za nekatere naloge.

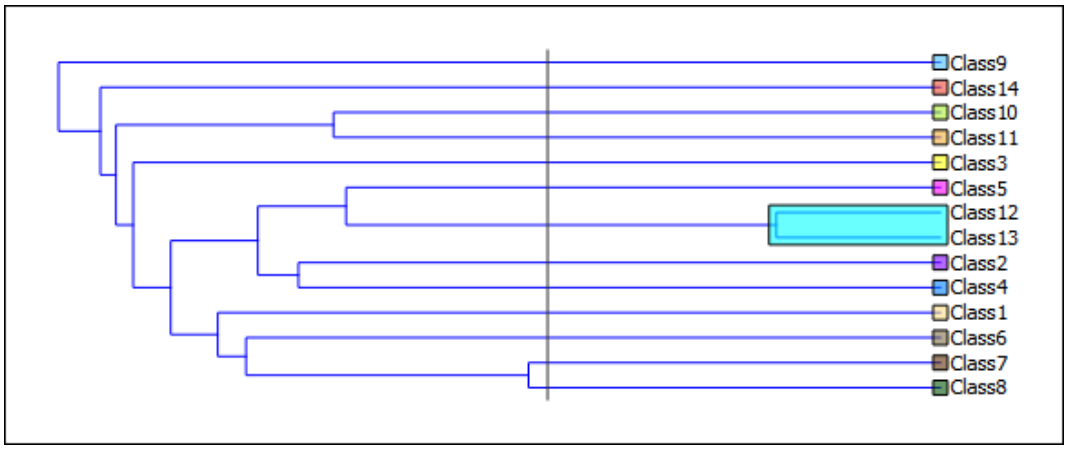
Tabela 5.15: Rezultati na podatkovni množici *Yeast* z mešanimi klasifikacijskimi in regresijskimi ciljnimi atributi.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	82,66 (4.)	84,75 (5.)	88,04 (6.)	73,75 (1.)	79,02 (2.)	82,08 (3.)
N2 (C)	56,22 (6.)	61,20 (3.)	60,16 (4.)	57,23 (5.)	63,42 (2.)	64,23 (1.)
N3 (R)	82,16 (3.)	85,44 (5.)	87,95 (6.)	77,54 (1.)	82,38 (4.)	81,54 (2.)
N4 (C)	66,27 (5.)	71,47 (4.)	71,70 (3.)	65,29 (6.)	73,31 (2.)	73,97 (1.)
N5 (R)	82,12 (4.)	83,60 (5.)	85,86 (6.)	77,25 (1.)	81,03 (2.)	81,25 (3.)
N6 (C)	70,08 (5.)	76,37 (3.)	75,44 (4.)	68,64 (6.)	77,40 (1.)	77,35 (2.)
N7 (R)	89,31 (2.)	91,67 (4.)	93,70 (6.)	85,81 (1.)	90,47 (3.)	92,50 (5.)
N8 (C)	74,48 (5.)	80,52 (1.)	80,04 (4.)	70,69 (6.)	80,44 (2.)	80,36 (3.)
N9 (R)	100,97 (4.)	99,70 (2.)	100,03 (3.)	101,53 (5.)	98,38 (1.)	104,19 (6.)
N10 (C)	85,28 (5.)	89,65 (3.)	89,65 (3.)	82,64 (6.)	89,70 (1.)	89,70 (1.)
N11 (R)	97,51 (5.)	97,35 (4.)	97,19 (3.)	94,67 (1.)	96,76 (2.)	100,49 (6.)
N12 (C)	66,67 (5.)	75,41 (1.)	75,41 (1.)	64,03 (6.)	75,39 (3.)	75,36 (4.)
N13 (R)	96,28 (3.)	96,22 (2.)	96,90 (5.)	94,43 (1.)	96,52 (4.)	97,17 (6.)
N14 (C)	98,24 (5.)	98,65 (1.)	98,65 (1.)	97,00 (6.)	98,65 (1.)	98,65 (1.)
Vsota uvrstitev	61	43	55	52	30	44
Uvrstitev	6	2	5	4	1	3

Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.

	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
N1 (R)	108,70 (5.)	89,44 (3.)	90,87 (4.)	114,67 (6.)	87,55 (2.)	87,34 (1.)
N2 (C)	0,56 (6.)	0,65 (4.)	0,67 (3.)	0,57 (5.)	0,68 (2.)	0,68 (1.)
N3 (R)	108,05 (5.)	89,27 (3.)	90,25 (4.)	117,57 (6.)	88,20 (2.)	86,89 (1.)
N4 (C)	0,65 (5.)	0,76 (3.)	0,76 (4.)	0,62 (6.)	0,79 (2.)	0,80 (1.)
N5 (R)	107,77 (5.)	88,90 (3.)	89,40 (4.)	116,42 (6.)	88,63 (2.)	86,97 (1.)
N6 (C)	0,58 (5.)	0,69 (4.)	0,70 (3.)	0,57 (6.)	0,71 (2.)	0,71 (1.)
N7 (R)	115,75 (5.)	94,15 (2.)	95,12 (4.)	119,93 (6.)	94,64 (3.)	93,86 (1.)
N8 (C)	0,57 (5.)	0,68 (2.)	0,68 (3.)	0,54 (6.)	0,67 (4.)	0,69 (1.)
N9 (R)	115,86 (5.)	99,62 (1.)	99,69 (2.)	117,34 (6.)	100,64 (3.)	100,70 (4.)
N10 (C)	0,52 (6.)	0,65 (4.)	0,67 (2.)	0,55 (5.)	0,66 (3.)	0,67 (1.)
N11 (R)	120,51 (5.)	97,92 (2.)	97,82 (1.)	124,75 (6.)	99,11 (4.)	98,37 (3.)
N12 (C)	0,53 (5.)	0,62 (4.)	0,62 (3.)	0,53 (6.)	0,63 (2.)	0,64 (1.)
N13 (R)	120,39 (5.)	97,73 (1.)	97,97 (3.)	130,90 (6.)	98,41 (4.)	97,93 (2.)
N14 (C)	0,48 (6.)	0,58 (3.)	0,58 (4.)	0,49 (5.)	0,63 (1.)	0,60 (2.)
Vsota uvrstitev	73	39	44	81	36	21
Uvrstitev	5	3	4	6	2	1



Slika 5.2: Hierarhično razvrščanje nalog podatkovne množice *Yeast*.

5.3.1 Podobne naloge

V nadaljevanju preizkusimo mero podobnosti med nalogami, ki kot razdaljo med nalogami uporablja absolutno razdaljo med rangiranji atributov. S pomočjo te mere želimo poiskati najbližje naloge in izboljšati rezultate za te naloge.

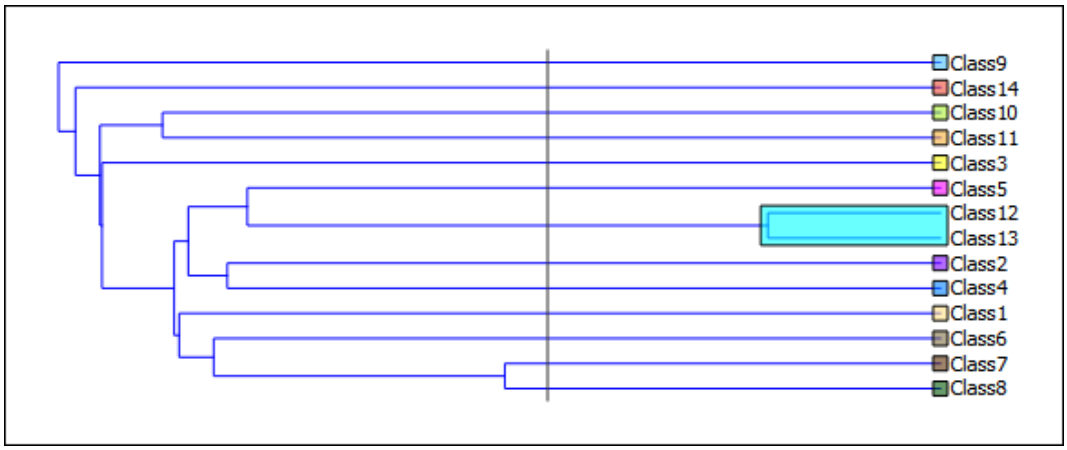
Slika 5.2 prikazuje primer hierarhičnega razvrščanja nalog pri podatkovni množici *Yeast*. Najbolj sta si podobni nalogi *Class 12* in *Class 13*. V tabeli 5.16 so prikazani rezultati na podatkovni množici *Yeast*, z uporabljenima le omenjenima dvema nalogama, ki sta najbližji glede na to mero podobnosti. V tem primeru je rezultat dosežen z večciljnim algoritmom boljši ali enak od rezultata enociljnih algoritmov. To je edina podskupina nalog, pri kateri je večciljni algoritem boljši od enociljnega pri tej podatkovni množici. Pri preizkusih ostalih najbližjih parov večciljni algoritem ni dosegel boljšega rezultata kot enociljni algoritem. Razlike med ostalimi nalogami so očitno prevelike, da bi naš algoritem uspel doseči boljše rezultate kot enociljni algoritem.

Slika 5.3 prikazuje primer hierarhičnega razvrščanja nalog pri podatkovni množici *Yeast mešana*, torej podatkovni množici z mešanimi klasifikacijskimi in regresijskimi nalogami. Razdalje med nalogami in hierarhija nalog so podobni kot pri navadni podatkovni množici *Yeast*. Enako sta med seboj

Tabela 5.16: Rezultati na podatkovni množici *Yeast*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.						
	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Class12 (C)	64,30 (6.)	75,87 (2.)	75,90 (1.)	65,30 (5.)	75,79 (3.)	75,76 (4.)
Class13 (C)	63,93 (5.)	75,24 (2.)	75,27 (1.)	63,81 (6.)	75,13 (3.)	74,95 (4.)
Vsota uvrstitev	11	4	2	11	6	8
Uvrstitev	5	2	1	5	3	4
Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.						
	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Class12 (C)	0,54 (6.)	0,63 (4.)	0,64 (1.)	0,54 (5.)	0,63 (3.)	0,63 (2.)
Class13 (C)	0,54 (5.)	0,63 (3.)	0,64 (2.)	0,54 (6.)	0,62 (4.)	0,64 (1.)
Vsota uvrstitev	11	7	3	11	7	3
Uvrstitev	5	3	1	5	3	1

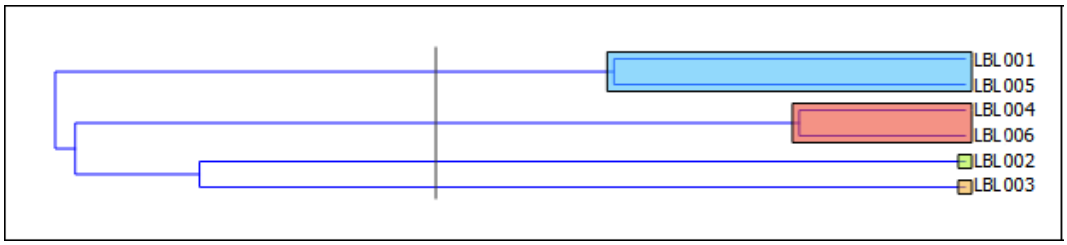
najbolj podobni nalogi *Class 12* in *Class 13*. V tabeli 5.17 so prikazani rezultati na podatkovni množici *Yeast mešana* za ti dve nalogi. Rezultati so podobni kot pri običajni podatkovni množici *Yeast*. Podobni rezultati so pričakovani, saj smo v tem primeru le spremenili tip nekaterih nalog iz klasifikacijskih v regresijske, algoritem pa je narejen z namenom, da deluje v obeh primerih, neodvisno od tipa naloge. V obeh primerih izstopa slabše delovanje enociljnega naključnega gozda pri prvem načinu meritve. Deloma je to verjetno posledica načina razvrščanja algoritmov z rangiranjem, saj se izgubijo dejanske velikosti razlik med algoritmi.



Slika 5.3: Hierarhično razvrščanje nalog podatkovne množice *Yeast mešana*.

Tabela 5.17: Rezultati na podatkovni množici *Yeast mešana*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.						
	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Class 12 (C)	63,76 (6.)	75,27 (2.)	75,28 (1.)	64,40 (5.)	75,17 (3.)	75,17 (3.)
Class 13 (R)	97,01 (6.)	95,61 (2.)	95,91 (3.)	94,58 (1.)	96,36 (4.)	96,89 (5.)
Vsota uvrstitev	12	4	4	6	7	8
Uvrstitev	6	1	1	3	4	5
Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.						
	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
Class 12 (C)	0,52 (6.)	0,63 (4.)	0,63 (2.)	0,53 (5.)	0,63 (3.)	0,64 (1.)
Class 13 (R)	134,82 (6.)	98,41 (4.)	97,85 (1.)	130,30 (5.)	98,36 (3.)	98,07 (2.)
Vsota uvrstitev	12	8	3	10	6	3
Uvrstitev	6	4	1	5	3	1



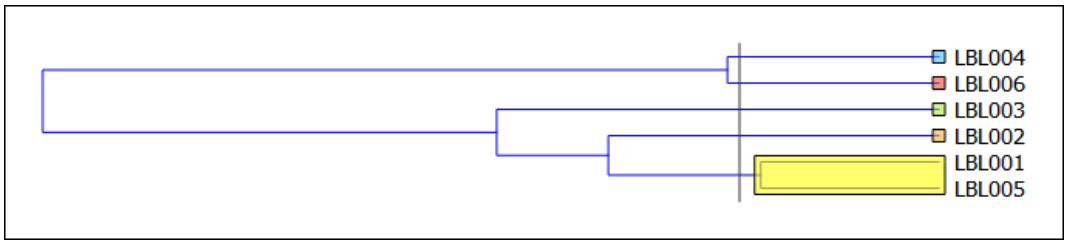
Slika 5.4: Hierarhično razvrščanje nalog podatkovne množice *Atp1d*.

Slika 5.4 prikazuje primer hierarhičnega razvrščanja nalog pri podatkovni množici *Atp1d*. Najbolj sta si podobni nalogi *LBL004* in *LBL006*. V tabeli 5.18 so prikazani rezultati na podatkovni množici *Atp1d* za ti dve nalogi. Večciljni bagging pri prvi meri doseže boljše, pri drugi meri pa enake rezultate kot enociljni bagging. V tem primeru izstopa slabo delovanje obeh naključnih gozdov pri obeh načinih meritev. Pri nalogah *LBL004* in *LBL006* je bil enociljni naključni gozd že pri preizkusu vseh nalog naenkrat najslabše uvrščen od vseh enociljnih algoritmov, le da se to v tem primeru ni poznalo na končni uvrstitvi, saj je bil pri ostalih nalogah dobro uvrščen. Tukaj pa to zaradi le dveh nalog pride bolj do izraza.

S pomočjo mere podobnosti med nalogami smo pri omenjenih treh podatkovnih množicah uspeli poiskati pare nalog, pri katerih lahko dosežemo boljše rezultate z večciljnim algoritmom kot z enociljnim. V vseh treh primerih so bili to pari nalog, ki so si bile najbolj podobne znotraj podatkovne množice. Pri vseh ostalih parih in podskupinah nismo uspeli izboljšati rezultatov enociljnega algoritma.

Tabela 5.18: Rezultati na podatkovni množici *Atp1d*.

Uporaba mere RMAE za regresijske naloge in klasifikacijske točnosti za klasifikacijske.						
	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
LBL004 (R)	24,54 (4.)	21,34 (1.)	30,73 (6.)	24,35 (3.)	23,79 (2.)	29,12 (5.)
LBL006 (R)	20,79 (4.)	18,31 (1.)	29,60 (6.)	19,28 (3.)	18,82 (2.)	27,92 (5.)
Vsota uvrstitev	8	2	12	6	4	10
Uvrstitev	4	1	6	3	2	5
Uporaba mere RMSE za regresijske naloge in mere AUC za klasifikacijske.						
	Tree MT	Bagging MT	RF MT	Tree	Bagging	RF
LBL004 (R)	29,97 (3.)	25,08 (1.)	34,43 (6.)	31,33 (4.)	26,52 (2.)	34,20 (5.)
LBL006 (R)	28,57 (4.)	23,40 (2.)	34,67 (6.)	26,53 (3.)	23,33 (1.)	34,31 (5.)
Vsota uvrstitev	7	3	12	7	3	10
Uvrstitev	3	1	6	3	1	5

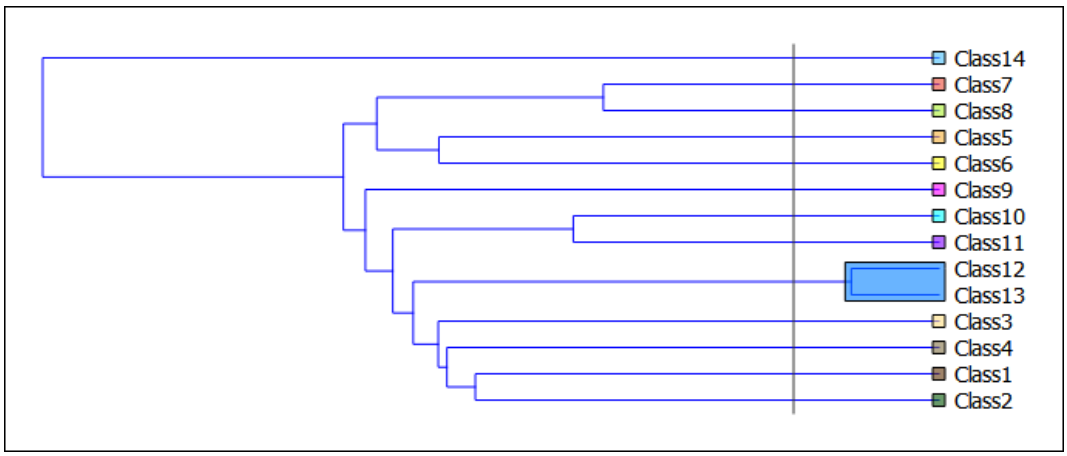


Slika 5.5: Hierarhično razvrščanje nalog podatkovne množice *Atp1d* z uporabo mere Relief.

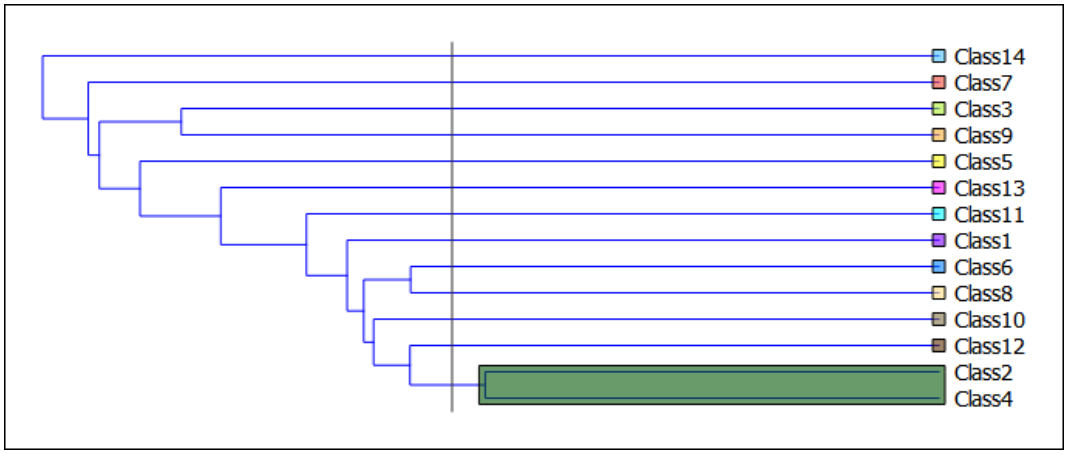
Preizkusili smo tudi rangiranje atributov z mero Relief. Pri uporabi te mere dobimo pri podatkovni množici *Yeast* (slika 5.6) podobne najbližje naloge kot pri prejšnjem pristopu. Najbližji nalogi sta si *Class 12* in *Class 13*, sledita para *Class 7* in *Class 8* ter *Class 10* in *Class 11*, kar je enako kot v zgornjem pristopu.

Pri podatkovni množici *Atp1d* (slika 5.5) sta najbližja para nalog *LBL001* in *LBL005* ter *LBL004* in *LBL006*, kar je ravno zamenjan vrstni red kot pri zgornjem pristopu. To je nekoliko slabša napoved, saj le pri paru *LBL004* in *LBL006* večciljni algoritem doseže boljši rezultat kot enociljni algoritem.

Pri podatkovni množici *Yeast* z mešanimi regresijskimi in klasifikacijskimi nalogami (slika 5.7) se na najnižjem nivoju povezujejo le naloge istega tipa, bodisi regresijske (liha števila) ali klasifikacijske (soda števila), zato dobimo precej drugačne pare najbližjih nalog kot z zgornjim pristopom. Pri mešanih klasifikacijskih in regresijskih nalogah se z uporabo mere Relief pojavi težava, da imajo klasifikacijske naloge preveč različna rangiranja od regresijskih, zato so si najbližje naloge le naloge istega tipa. Za podatkovne množice z nalogami istega tipa je uporaba podobnosti na podlagi rangiranja atributov z metodo Relief dokaj blizu zgoraj opisani metodi z uporabo mer uporabljenih pri izgradnji drevesa, medtem ko za podatkovne množice z mešanimi tipi nalog brez dodatnih prilagajanj ni primerna.



Slika 5.6: Hierarhično razvrščanje nalog podatkovne množice *Yeast* z uporabo mere Relief.



Slika 5.7: Hierarhično razvrščanje nalog podatkovne množice *Yeast* z mešanimi klasifikacijskimi in regresijskimi nalogami z uporabo mere Relief.

Poglavje 6

Zaključek

Algoritme odločitveno in regresijsko drevo, bagging ter naključni gozd smo prilagodili za delovanje na večciljnih problemih in jih implementirali s prilagoditvijo obstoječih algoritmov v orodju *Weka*. Algoritmi lahko delujejo na podatkovnih množicah z regresijskimi in klasifikacijskimi nalogami znotraj ene podatkovne množice. Za izbiro atributov v vozlišču pri izgradnji drevesa smo uporabili rangiranje atributov. Pri vsaki nalogi smo atribute rangirali ter izbrali atribut, ki je bil skupno najbolje rangiran. Uspešnost algoritmov smo primerjali s standardnimi različicami omenjenih algoritmov. Na nekaterih podatkovnih množicah naš algoritem dobro deluje, na različicah podatkovne množice *Water Quality* deluje večciljni naključni gozd statistično značilno bolje od enociljnih algoritmov. Na nekaterih podatkovnih množicah pa delujejo implementirani večciljni algoritmi slabše od enociljnih algoritmov. Pri rangiranju se izgubi nekaj informacije o dejanskih razlikah med ocenami atributov, tako da s tem združevanje nalog izgubi nekaj moči.

Eden od razlogov za slabše delovanje na nekaterih podatkovnih množicah je lahko premajhna sorodnost nalog znotraj podatkovne množice, da bi jih naš algoritem uspel izkoristi za boljše rezultate. Težavo smo poskusili omiliti z mero podobnosti med nalogami in združevanjem podobnih nalog v skupine. Izkaže se, da lahko večciljni algoritem izboljša delovanje nekaterih parov nalog znotraj podatkovne množice, četudi na vseh nalogah naenkrat tega ni

zmožen.

Pri nadaljnjem delu bi lahko preizkusili še kakšen način rangiranja atributov in način izbire skupno najboljšega atributa. Preučili bi lahko, kakšna razlika med nalogami je še sprejemljiva, da je koristno nalogi uporabiti skupaj za boljši rezultat. Tako bi že pri izračunu sorodnosti lahko ocenili, ali bi izboljšali rezultat z združitvijo teh nalog v skupino. Mero za sorodnost bi lahko uporabili neposredno znotraj algoritma, tako bi algoritem lahko sam izbral, katere naloge bi obravnaval skupaj z uporabo večciljnega algoritma in katere posamezno z uporabo enociljnega algoritma. Z enociljnim algoritmom bi obravnaval naloge, ki niso dovolj sorodne ostalim in z večciljnim tiste naloge, ki so si dovolj sorodne med seboj.

Literatura

[1] R. Caruana, "Multitask learning", *Machine Learning*, št. 28, zv. 1, str. 41–75, 1997.

[2] H. Blockeel, L. De Raedt, J. Ramon, "Top-down induction of clustering trees", *Proceedings of the 15th International Conference on Machine Learning*, str. 55–63, 1998.

[3] P. Huang, G. Wang, S. Qin, "A novel learning approach to multiple tasks based on boosting methodology", *Pattern Recognition Letters*, št. 31, zv. 12, str. 1693–1700, 2010.

[4] J. B. Faddoul, B. Chidlovskii, R. Gilleron, F. Torre, "Learning multiple tasks with boosted decision trees", *Machine Learning and Knowledge Discovery in Databases*, str. 681–696, 2012.

[5] T. Aho, B. Ženko, S. Džeroski, T. Elomaa, "Multi-target regression with rule ensembles", *The Journal of Machine Learning Research*, št. 13, zv. 1, str. 2367–2407, 2012.

[6] M. Solnon, S. Arlot, F. Bach, "Multi-task regression using minimal penalties", *The Journal of Machine Learning Research*, št. 13, zv. 1, str. 2773–2812, 2012.

[7] B. Bakker, T. Heskes, "Task clustering and gating for bayesian multitask learning", *The Journal of Machine Learning Research*, št. 4, str. 83–99, 2003.

[8] T. Evgeniou, C. A. Micchelli, M. Pontil, "Learning multiple tasks with kernel methods", *Journal of Machine Learning Research*, str. 615–637, 2005.

[9] S. Thrun, J. O'Sullivan, "Clustering learning tasks and the selective cross-task transfer of knowledge", *Learning to learn*, str. 235–257, Kluwer Academic Publishers, 1998.

[10] L. Breiman, "Bagging predictors", *Machine learning*, št. 24, zv. 2, str. 123–140, 1996.

[11] L. Breiman, "Random forests", *Machine learning*, št. 45, zv. 1, str. 5–32, 2001.

[12] D. Kocev, C. Vens, J. Struyf, S. Džeroski, "Tree ensembles for predicting structured outputs", *Pattern Recognition*, št. 46, zv. 3, str. 817–833, 2013.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The weka data mining software: an update", *ACM SIGKDD explorations newsletter*, št. 11, zv. 1, str. 10–18, 2009.

[14] M. Robnik-Šikonja, I. Kononenko, "Theoretical and empirical analysis of ReliefF and RReliefF", *Machine learning*, št. 53, zv. 1-2, str. 23–69, 2003.

[15] "Mulan multi-target regression dataset." `http://mulan.sourceforge.net/datasets-mtr.html`. [Dostop: 1.12.2015].

[16] D. Kocev, "Multi-Target Classification Datasets." `http://kt.ijs.si/DragiKocev/PhD/resources/doku.php?id=multi_target_classification`. [Dostop: 1.12.2015].

[17] S. Džeroski, D. Demšar, J. Grbović, "Predicting chemical parameters of river water quality from bioindicator data", *Applied Intelligence*, št. 13, zv. 1, str. 7–17, 2000.

[18] Y. Reich, "Design knowledge acquisition: task analysis and a partial implementation", *Knowledge Acquisition*, št. 3, zv. 3, str. 237–254, 1991.

[19] H. Fanaee-T, J. Gama, "Event labeling combining ensemble detectors and background knowledge", *Progress in Artificial Intelligence*, št. 2, zv. 2-3, str. 113–127, 2014.

[20] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, I. Vlahavas, "Multi-label classification methods for multi-target regression", *arXiv preprint arXiv:1211.6581*, 2012.

[21] A. Elisseeff, J. Weston, "A kernel method for multi-labelled classification", *Advances in neural information processing systems*, str. 681–687, 2001.

[22] J. Demšar, "Statistical comparisons of classifiers over multiple data sets", *The Journal of Machine Learning Research*, št. 7, str. 1–30, 2006.