Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Saša Saftić

Primerjava pristopov k večznačni in večciljni klasifikaciji

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Zoran Bosnić

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00135/2013
Datum: 15.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **SAŠA SAFTIĆ**

Naslov: **PRIMERJAVA PRISTOPOV K VEČZNAČNI IN VEČCILJNI KLASIFIKACIJI**

COMPARISON OF APPROACHES TO MULTILABEL AND MULTITARGET CLASSIFICATION

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Pri klasifikaciji obstajajo problemske domene, kjer moramo posamezni primer klasificirati v več razredov hkrati (večznačna klasifikacija) ali pa z enim klasifikatorjem simulirati več posameznih klasifikatorjev in s tem napovedovati več razrednih spremenljivk hkrati (večciljna klasifikacija).
V diplomski nalogi naj kandidat opravi pregled metod s področij večciljne in večznačne klasifikacije. Predstavi naj zadnje dosežke na tem področju in prikaže različne načine, pri katerih lahko z enoznačnimi modeli simuliramo večznačno/večciljno napovedovanje. Kandidat naj izbrane algoritme testira na danih podatkih iz pedagoške domene, namenjenih klasifikaciji učnih gradiv v multimedijske kategorije.

Mentor:

doc. dr. Zoran Bosnić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Saša Saftić, z vpisno številko **63100435**, sem avtor diplomskega dela z naslovom:

Primerjava pristopov k večznačni in večciljni klasifikaciji.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Zorana Bosnića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 12. septembra 2013 Podpis avtorja:

Želel bi se zahvaliti staršem za čustveno in finančno podporo na poti do diplome in mentorju Zoranu Bosniću za hitre ter obsežne odgovore na moja vprašanja.

Kazalo

Povzetek

V diplomskem delu smo primerjali delovanje enoznačnih, večznačnih in večrazrednih klasifikacijskih metod. Testiranja so bila izvedena na podatkih, ki predstavljajo problem iz pedagoške domene in so primerni za večciljno klasifikacijo. Za uporabo enoznačnih klasifikatorjev na enakih podatkih smo z različnimi metodami za transformacijo podatkov transformirali podatke v obliko, ki je primerna za enoznačno klasifikacijo. Za vsak algoritem smo izračunali mere uspešnosti in jih primerjali med seboj. Za najboljše algoritme smo izrisali ROC-krivulje. Dobljeni rezultati kažejo, da lahko za dobre rezultate uporabimo več različnih pristopov (CLUS, HOMER, nevronske mreže). Kljub temu, da pokažejo algoritmi, ki so namenjeni večciljni klasifikaciji, boljše napovedovanje v več razredov, so zadovoljivi tudi rezultati, pridobljeni z ostalimi metodami.

Ključne besede

enoznačna klasifikacija, večznačna klasifikacij, večciljna klasifikacija, mere uspešnosti, ROC-krivulja

Abstract

In this research we compare different algorithms for single-label, multi-label and multi-class classification. The testing was performed using data representing a problem from educational domain, and are appropriate for multi-class classification. We used different methods for transforming multi-label classification to single-label classification. We compared algorithms with evaluation measures and ROC curves. Results shows that several methods can be used for this task. Algorithms that are specialized for multi-class classification have better predictions for multi-class problems, although other algorithms also show good performance.

Keywords

single-label classification, multi-label classification, multi-class classification, evaluation measures, ROC curve

Poglavje 1

Uvod

Klasifikacija oz. razvrščanje različnih primerov je lahko enostaven problem (npr. razvrščanje oseb po spolu). Takšno razvrščanje opravljamo dnevno in se nam zdi enostavno. Kompleksnost problema se povečuje s številom parametrov, ki jih potrebujemo za uspešno razvrščanje. Obdelava podatkov postane kompleksen problem, ko se soočamo z večjim številom primerov z več parametri. Zaradi tega so razvili algoritme, ki zmorejo obdelavo podatkov, ki vsebujejo veliko primerov in veliko atributov.

V preteklosti so se večinoma ukvarjali z enoznačno klasifikacijo (vsakemu primeru so pripisali samo en razred). Danes pa imamo vedno več problemov, ki potrebujejo klasifikacijo v več razredov naenkrat. Eden od razlogov za to je verjetno svetovni splet, ki nam je omogočil dostop do velike množice podatkov. Za reševanje teh problemov uporabljamo algoritme za večznačno klasifikacijo. Tipična predstavnika takšnega problema sta razvrščanje besedil (recimo novica v časopisu je lahko povezana s kategorijama šport in medicina) ali filmov v več kategorij (npr. isti film je lahko drama in komedija hkrati). Včasih pa želimo iste podatke razvrstiti v več izključujočih se kategorij.

V diplomskem delu se ukvarjamo s primerjavo uporabe različnih metod klasifikacije nad izbranimi podatki, ki predstavljajo problem iz pedagoške domene. V poglavju 2 opišemo različne metode strojnega učenja in uporabljene algoritme. V poglavju 3 opišemo različne mere uspešnosti, ki smo jih upora-

bili za primerjavo uspešnosti med algoritmi. V okviru poglavja 4, ki vsebuje opis eksperimentalnega postopka in podatkov, opišemo različna okolja in pristope, ki smo jih uporabili za implementacijo algoritmov. V tem poglavju opišemo tudi strukturo in vsebino podatkov. V poglavju 4.6 predstavimo rezultate. Na koncu sledi še zaključek.

Poglavje 2

Opisi metod strojnega učenja

2.1 Kaj je strojno učenje?

V diplomi se ukvarjamo s področjem nadzorovanega učenja, in sicer s klasifikacijo. Klasifikacija je proces učenja funkcije, ki preslika podatek v enega izmed vnaprej določenih razredov [1]. V nadaljevanju bomo opisali različne klasifikacijske pristope za ta problem.

2.2 Enoznačna klasifikacija

V strojnem učenju je enoznačna klasifikacija pogost problem. Cilj pri enoznačni klasifikaciji je, da se algoritem uči iz množice primerov, kjer je vsak povezan z oznako iz množice oznak. Cilj je klasifikacijski model (angl. classifier), ki na osnovi vrednosti vhodnih atributov napove razred za neki dani primer. Z drugimi besedami, klasifikacija je proces, kjer nekemu neoznačenemu primeru določimo neko diskretno vrednost (oznako). Klasifikacijski model je model (rezultat klasifikacije), ki pridobi en atribut (oznako) s pomočjo ostalih [2]. V diplomi smo uporabili in primerjali več različnih algoritmov za enoznačno klasifikacijo.

2.2.1 Naivni Bayesov klasifikator

Naivni Bayesov klasifikator predpostavlja pogojno neodvisnost vrednosti atributov pri danem razredu:

$$p(v_1, v_2, ...v_n|c) = \prod_i p(v_i|c) \tag{2.1}$$

Naivna Bayesova formula:

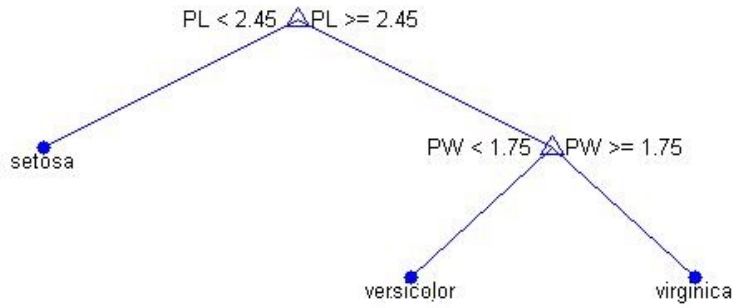
$$p(c|v_1, v_2, ...v_n) = \prod_i \frac{p(c|v_i)}{p(c)} \tag{2.2}$$

Naloga učnega algoritma je s pomočjo učne množice podatkov aproksimirati verjetnosti na desni strani enačbe [3].

Primer: recimo, da je objekt kolo, če ima dve kolesi, dva pedala in sedež. Bayesov klasifikator upošteva vsako lastnost nekega primera ločeno od ostalih. Tako verjetnosti, da je objekt kolo, lastnost *dve kolesi* prinese enako, ne glede na to, ali ima ostale lastnosti ali ne.

2.2.2 Odločitveno drevo

Odločitvena drevesa delujejo običajno tako, da algoritem pregleda celotno množico učnih podatkov in poišče *najbolj informativen* atribut. Učna množica je nato razdeljena glede na vrednost tega atributa. *Najbolj informativen* atribut pomeni, da je podmnožica, ki je ustvarjena glede na vrednost tega atributa, v povprečju bolj homogena od ostalih možnih podmnožic. Pri podmnožici, ki ni popolnoma homogena, algoritem postopek ponovi. Kadar najde najbolj informativen atribut, razdeli množico glede na njegovo vrednost. Algoritem to ponavlja, dokler ni vsaka podmnožica popolnoma homogena (sestavljena iz enega razreda) ali pa je zadoščeno kakšnemu drugemu ustavitvenemu pogoju. Hierarhijo, ki jo dobimo s tem postopkom, imenujemo odločitveno drevo. Drevo lahko uporabimo za klasifikacijo tako, da nov primer uvrstimo v list drevesa glede na vrednosti njegovih atributov. Primeru določimo razred, ki se najpogosteje pojavi v listu [7]. Strategija



Slika 2.1: Odločitveno drevo

garantira enostavne, a ne nujno najenostavnejše rešitve [1]. Prikaz enostavnega odločitvenega drevesa, ki prikazuje problem klasificiranja v domeni iris, je prikazan na sliki 2.1. S slike lahko sklepamo, da bi nov primer, ki bi imel atribut $PL < 2.45$, klasificirali v razred *setosa*. Primer, ki bi imel atribut $PL >= 2.45$, bi nato klasificirali glede na atribut PW. Če ima primer atribut $PW < 1.75$, ga klasificiramo v razred *versicolor*, drugače pa v razred *virginica*.

2.2.3 Metoda k najbližjih sosedov

Metoda k najbližjih sosedov klasificira primere glede na njihovo podobnost z ostalimi primeri. Za primere, ki so blizu eden drugega, pravimo, da so sosedje. Ko dodamo nov primer, izračunamo njegovo razdaljo od ostalih primerov. Nov primer je klasificiran v kategorijo, s katero je označeno največje število najbližjih sosedov. Število najbližjih sosedov, ki jih želimo pogledati, določimo sami (vrednost označimo s k)[4].

2.2.4 Nevronske mreže

Nevronske mreže so zasnovane po vzoru nevronov v možganih. V osnovi predstavljajo nevronske mreže kompleksen algoritem, ki se uporablja tudi na drugih področjih, ne le pri enoznačni klasifikaciji. Topologija nevronske mreže običajno vključuje tri sloje, ki si prenašajo podatke (angl. feed forward). Vhodni sloj (angl. input layer) je sloj, ki mu podamo parametre našega primera. Vozlišča (nevroni) vhodnega sloja so povezana z vsemi vozlišči skritega sloja (angl. hidden layer). Vozlišča skritega sloja so prav tako povezana z vsemi vozlišči izhodnega sloja (angl. output layer). Izhodni sloj vrača odgovor nevronske mreže glede na vrednosti vhodnih nivojev. Podatki se lahko prenašajo preko enega ali več skritih slojev, lahko pa se ne prenašajo skozi noben skrit sloj. [5].

2.3 Večznačna klasifikacija

Tradicionalna enoznačna klasifikacija se ukvarja z učenjem iz množice primerov, ki so označeni z eno oznako λ iz množice oznak $L, |L| > 1$. Če je $|L| = 2$, potem gre za problem binarne klasifikacije (ali filtriranje v primeru obdelave besedila in spleta). Pri večznačni klasifikaciji so primeri označeni s skupino oznak $Y \subseteq L$ [6]. Z drugimi besedami, večznačna klasifikacija dodeli vsakemu primeru skupino oznak. Te oznake med seboj niso izključujoče. Neki članek je lahko označen s kategorijami religija, politika, finance, izobrazba ... (nekateri članki lahko pripadajo samo eni izmed kategorij, nekateri pa več kategorijam).

V diplomskem delu smo uporabili in primerjali dva različna algoritma za večznačno klasifikacijo.

2.3.1 HOMER

HOMER deluje po metodi deli in zmagaj (angl. divide and conquer). Glavna ideja je, da večznačno klasifikacijo, ki ima veliko oznak L, transformiramo v

drevesno strukturo, ki je sestavljena iz več enostavnejših večznačnih problemov. Vsak enostavnejši primer ima manjše število oznak $k << |L|$. Vsako vozlišče n tega drevesa vsebuje skupino oznak $L_n \subseteq L$. Vsako notranje vozlišče n vsebuje unijo znakov svojih otrok (vozlišč pod njim), $L_n = \bigcup L_c, c \in otrok(n)$. Vozlišče, ki je koren drevesa, vsebuje vse oznake $L_{root} = L$. Pri algoritmu definiramo koncept metaoznake (angl. meta-data) vozlišča n,μ_n kot disjunkcijo oznak, ki so v tem vozlišču, $\mu_n \equiv \vee\lambda_j, \lambda_j \in L_n$. Metaoznake imajo naslednji pomen: učni primer ima lahko metaoznako samo, če pripada vsaj eni izmed oznak iz L_n. Vsako notranje vozlišče n vsebuje večznačen klasifikator h_n. h_n napoveduje eno ali več metaoznak svojih otrok. Tako je skupina oznak za h_n enaka $M_n = \{\mu_c | c \in otroci(n)\}$. Za večznačno klasifikacijo novega primera x začne HOMER algoritem pri h_{root} in rekurzivno posreduje x večznačim klasifikatorjem h_c (kjer je c otrok trenutnega vozlišča) samo, če je μ_c med predikcijskimi vrednostmi od $h_{oce(c)}$. Ta proces lahko pripelje do predikcije enega ali več enoznačnih prediktorjev tik nad izbranim/mi listom/listi. Unija napovedanih enoznačnih oznak je rezultat algoritma HOMER [6].

2.3.2 BP-MLL

BP-MLL je model nevronske mreže, ki je prilagojen za večznačno klasifikacijo. Uporablja znani *back-propagation* algoritem s prilagojeno funkcijo napake. Napaka algoritma BP-MLL je prilagojena tako, da upošteva značilnosti večznačne klasifikacije [9].

2.4 Večciljna klasifikacija

Večciljna klasifikacija pomeni, da želimo klasificirati več razrednih spremenljivk naenkrat (besede razred zato ne smemo zamešati z besedo oznaka). Za lažje razumevanje ponazorimo razliko med večznačno in večciljno klasifikacijo s primerom. Denimo, da želimo klasificirati fotografijo, na kateri je predmet, ki ima lahko več barv. Medtem ko bomo z večznačno klasifikacijo lahko

primeru pripisali več vrednosti razreda hkrati (npr. "zelena" in "rumena"), bomo z večciljno klasifikacijo lahko fotografijo klasificirali v več različnih pomenskih razrednih spremenljivk, od katerih npr. prva določa barvo objekta, druga pa velikost fotografije (torej "zelena"-"velika").

2.4.1 CLUS

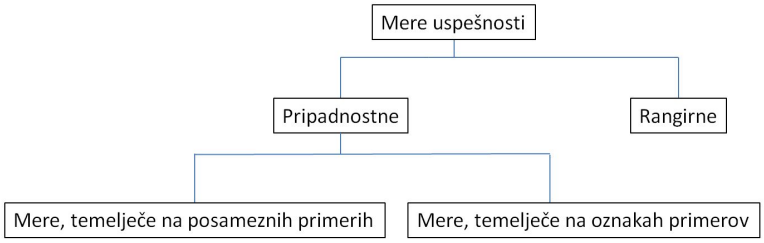
Clus je različica odločitvenega drevesa. V nadaljevanju bomo opisali, kako lahko podoben postopek uporabimo za večciljno klasifikacijo. Prva sprememba je uporaba hevristike (merimo *razdaljo* med primerom in povprečjem razdalj primerov podmnožice, ki ji pripada). Razdaljo lahko merimo npr. s kvadratno napako (angl. squared error). Hevristika je prva razlika med običajnim odločitvenim drevesom in odločitvenim drevesom, prilagojenim za večciljno klasifikacijo. Druga razlika je odločanje. Ko primer uvrstimo v list, se moramo odločiti, katerim razredom pripada. Standardni postopek (pri običajnem odločitvenem drevesu) je, da izberemo razred, ki se v listu najpogosteje pojavi. V primeru odločitvenega drevesa za večciljno klasifikacijo lahko primeru določimo več razredov. To naredimo tako, da predpostavimo, da primer pripada nekemu razredu c_i, če delež primerov p_i, ki pripadajo razredu c_i, v listu preseže mejo (angl. treshold) t_i. CLUS vrednosti t_i med izvajanjem ne spreminja, ampak shranjuje vrednosti p_i in omogoča naknadno spremembo vrednosti t_i.

Ustavitvena pogoja, ki ju uporablja CLUS, sta parameter *minclass*, ki določa minimalno število razredov v listu, in statistični *F-test*, ki preverja, ali je nadaljnja delitev lista statistično smiselna [7].

Poglavje 3

Mere uspešnosti učenja

Mere uspešnosti se za večznačne/večciljne sisteme razlikujejo od tistih, ki so namenjene ocenjevanju uspešnosti enoznačnega sistema. Zaradi večje kompleksnosti napovednega problema je zaželeno, da vključimo več različnih mer uspešnosti. Mere uspešnosti nekega predikcijskega algoritma delimo v dve skupini, pripadnostne (angl. bipartiton-based) in rangirne (angl. ranking-based). Pripadnostne mere računamo tako, da primerjamo napovedane in pravilne oznake. To skupino (pripadnostne) delimo še na mere, temelječe na posameznih primerih (angl. example-based), in mere, temelječe samo na oznakah primerov (angl. label-based). Mere, temelječe na posameznih primerih, upoštevajo povprečno razliko dejanskih in napovedanih skupin v celotni množici primerov. Mere, temelječe samo na oznakah primerov, pa ocenjujejo predikcijsko uspešnost za vsako oznako posebej in na koncu povprečijo uspešnosti vseh oznak. V diplomskem delu uporabimo štiri mere, temelječe na posameznih primerih: točnost (angl. accuracy), preciznost (angl. precision), priklic (angl. recall) in F-mero (angl. F-measure). Rangirne mere uspešnosti primerjajo razporeditev (angl. ranking) oznak s pravilno razporeditvijo oznak [10]. Na sliki 3.1 je grafični prikaz razdelitve mer uspešnosti. V nadaljevanju so opisane uporabljene mere uspešnosti. V definicijah je z Y_i označena skupina pravilnih oznak primera x_i. S $h(x_i)$ je označena skupina napovedanih oznak za primer X_i. Vse definicije se nanašajo na podatke, ki



Slika 3.1: Kategorizacija mer uspešnosti [10]

so primerni za večznačno klasifikacijo.

3.1 Točnost

Točnost (angl. accuracy) je za posamezni primer x_i definirana kot Jaccardova podobnost koeficientov med dvema skupinama oznak $h(x_i)$ in Y_i. Točnost nato povprečimo po vseh primerih [10]:

$$accurracy(h) = \frac{1}{N} \sum_{i=1}^{N} \frac{|h_{(x_i)} \cap Y_i|}{|h_{(x_i)} \cup Y_i|} \tag{3.1}$$

Z Y_i je označena skupina pravilnih oznak primera x_i. S $h(x_i)$ je označena skupina napovedanih oznak za primer X_i. Z N je označeno število primerov.

Jaccardova podobnost (ali razdalja) meri podobnost med dvema množicama tako, da deli število elementov, ki jih imata množici v preseku s številom elementov, ki jih imata v uniji.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.2}$$

Z A in B sta označeni množici podatkov.

3.2 Preciznost

Preciznost (angl. precision) je za posamezni primer x_i definirana kot število primerov, ki so v preseku med množico napovedanih oznak in pravilnih oznak,

deljeno s številom elementov v množici pravilnih oznak. Točnost je nato povprečena po vseh primerih[10]:

$$precision(h) = \frac{1}{N} \sum_{i=1}^{N} \frac{|h_{(x_i)} \cap Y_i|}{|Y_i|} \tag{3.3}$$

Z Y_i je označena skupina pravilnih oznak primera x_i. S $h(x_i)$ je označena skupina napovedanih oznak za primer X_i. Z N je označeno število primerov.

3.3 Priklic

Priklic (angl. recail) je za posamezni primer x_i, definiran kot število primerov, ki so v preseku med množico napovedanih oznak in pravilnih oznak. To število delimo s številom elementov v množici napovedanih oznak[10].

$$recall(h) = \frac{1}{N} \sum_{i=1}^{N} \frac{|h_{(x_i)} \cap Y_i|}{|h_{(x_i)}|} \tag{3.4}$$

Z Y_i je označena skupina pravilnih oznak primera x_i. S $h(x_i)$ je označena skupina napovedanih oznak za primer X_i. Z N je označeno število primerov.

3.4 F-mera

F-mera (angl. F-measure) je definirana kot harmonična sredina med preciznostjo in priklicem. Definiramo jo tako, da dvakratno število primerov, ki so v preseku med množico napovedanih oznak in pravilnih oznak, delimo z vsoto števila primerov v množici napovedanih oznak in številom elementov v množici pravilnih oznak [10].

$$F = \frac{1}{N} \sum_{i=1}^{N} \frac{2 * |h_{(x_i)} \cap Y_i|}{|h_{(x_i)}| + |Y_i|} \tag{3.5}$$

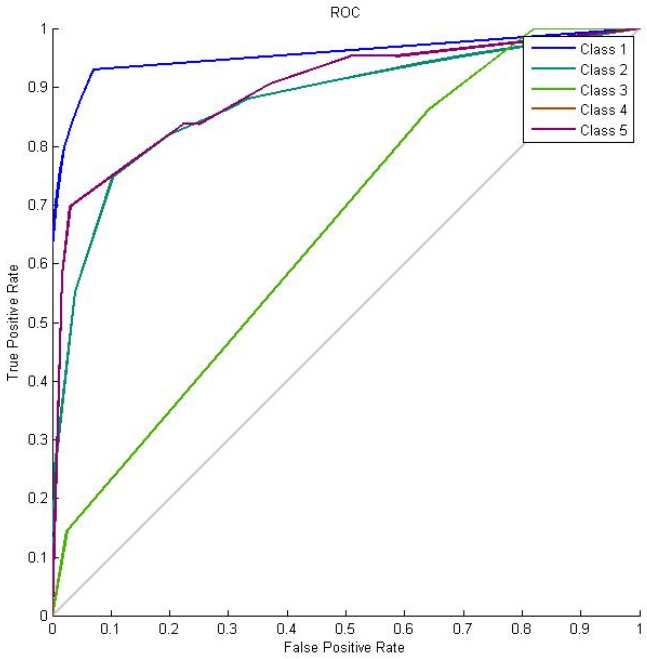
Z Y_i je označena skupina pravilnih oznak primera x_i. S $h(x_i)$ je označena skupina napovedanih oznak za primer X_i. Z N je označeno število primerov. F doseže najboljšo oceno pri vrednosti 1 in najslabšo oceno pri vrednosti 0.

3.5 ROC-krivulja

Analiza ROC (angl. Receiver Operating Characteristics, karakteristika sprejemnika) je zelo popularna tehnika grafične analize klasifikacijskih metod. ROC-analiza se uporablja na področjih radiologije, medicine, statistike, bioinformatike, strojnega učenja, prepoznavanja vzorcev itd. Prav tako so pogosto uporabljene meritve, ki so izpeljane iz ROC-krivulje, kot npr. ploščina pod krivuljo (AUC).

V klasifikaciji nam ROC-krivulja pokaže delovanje modela na dva načina. Na x-koordinati prikazuje napačno pozitivno frekvenco FPR (angl. false positive rate), na y-koordinati pa prikazuje pravilno pozitivno frekvenco TPR (angl. true positive rate). TPR določa, koliko je pravilnih napovedi v pozitiven razred (npr. 1), FPR pa, koliko je napačnih napovedi v pozitivni razred. Krivulja ROC enostavno vizualizira TPR in FPR. [19]. Najboljša točka je v koordinati (0,1), ki nam pove, da ni napačnih negativnih napovedi in da ni napačnih pozitivnih napovedi. Nad diagonalo, ki deli ROC-prostor, so klasifikacije, ki so boljše od naključnih. Prostor pod diagonalo pa predstavlja klasifikacije, ki so slabše od naključnih.

Primer ROC-krivulj vidimo na sliki 3.2. Na sliki je razvidno dobro delovanje algoritma pri napovedovanju dveh razredov in slabše pri treh.



Slika 3.2: Več ROC-krivulj v istem koordinatnem sistemu

Poglavje 4

Evalvacija in rezultati

V tem poglavju je opisan potek dela. Kratkemu opisu delovnega okolja (katere programe in knjižnice smo uporabili pri raziskovanju) sledi definicija parametrov, s katerimi smo poganjali posamezne algoritme. V nadaljevanju so opisane zgradba podatkov in potrebne prilagoditve podatkov za to diplomsko delo.

4.1 Delovno okolje

Za primarno delovno okolje smo izbrali programski jezik Matlab [11]. Ker smo uporabili veliko algoritmov s področja večznačne/večciljne klasifikacije, smo uporabili tudi knjižnico Mulan [12], ki je napisna v programskem jeziku Java [13]. V Javi smo uporabili knjižnico CLUS.

4.1.1 Matlab

Matlab je visokonivojski programski jezik, ki ga uporabljamo v interaktivnem in grafičnem okolju. Namenjen je grafičnemu prikazu podatkov, razvijanju algoritmov, ustvarjanju matematičnih modelov itd. Matlab vsebuje programski jezik, analitična orodja in veliko že vključenih matematičnih funkcij [11].

Za potrebe raziskovalnega dela diplomske naloge smo uporabili statistično orodje (angl. statistical toolbox), ki ni prisotno v osnovni verziji programa

Matlab. Matlab smo uporabili za implementacijo algoritma nevronskih mrež in za uporabo algoritmov odločitveno drevo, naivni Bayes, k najbližjih sosedov in za izračun nekaterih mer uspešnosti.

4.1.2 Mulan

Mulan je odprtokodna knjižnica Java za delo z večznačnimi podatki. Večznačne podatke uporablja za gradnjo večznačnih predikcijskih modelov. Knjižnica Mulan vsebuje več algoritmov s področja strojnega učenja [12].

Za potrebe raziskovalnega dela diplomske naloge smo uporabili Mulanovo implementacijo algoritmov HOMER in BP-MLL.

4.1.3 CLUS

Knjižnico z algoritmom CLUS so implementirali v programskem jeziku Java na Univerzi K. U. Leuven in na Institutu "Jožef Štefan"[8].

4.2 Parametri uporabljenih algoritmov

Pri posameznih algoritmih strojnega učenja smo izbrali naslednje parametre:

- Naivni Bayes: za testiranje algoritma naivni Bayes smo uporabili implementacijo v programskem jeziku in okolju Matlab. Dodatni argument, ki ga lahko določimo, je distribution, ki določa, kakšna porazdelitev najbolje opisuje naše podatke. Na voljo imamo razdelitve normal (Gaussian), kernel, mvmn (Multivariate multinomial distribution) in mn (Multinomial distribution). Nastavili smo delovanje algoritma z multinomsko porazdelitvijo verjetnosti.

- Odločitveno drevo: za testiranje algoritma odločitveno drevo smo uporabili implementacijo v programskem jeziku in okolju Matlab. Odločitveno drevo smo ustvarili tako, da smo objektu *classregtree* podali učno matriko X in vektor oznak y. Če je naš vektor oznak sestavljen

iz veliko različnih vrednosti, se izvede regresija. Če je y sestavljen iz kategoričnih podatkov, kot v našem primeru, pa klasifikacija. Uporabili smo atribut *prune*, ki vključi rezanje drevesa.

- K najbližjih sosedov: za testiranje algoritma k najbližjih sosedov smo uporabili implementacijo v programskem jeziku in okolju Matlab. Pri tem algoritmu nismo spreminjali privzetih vrednosti ostalih atributov. Vrednost k smo nastavili na 5.

- Nevronske mreže: implementacijo nevronske mreže smo naredili v okolju Matlab. Za algoritem smo napisali ocenitveno (angl. cost) in gradientno funkcijo. Uporabili smo knjižnico *fmincg*, ki optimizira postopek in skrbi, da ne pride do prevelike prilagoditve učnih podatkov (angl. overfit). Pri implementaciji smo algoritmu določili več atributov. Velikost vhodnega sloja je enaka številu primerov v učni množici, velikost skritega sloja je enaka številu atributov in velikost izhodnega sloja je enaka številu unikatnih oznak. Algoritmu smo omogočili maksimalno 800 iteracij pri iskanju najmanjše napake med pravilnimi in napovedanimi vrednostmi.

- HOMER: za testiranje z algoritmom HOMER smo uporabili implementacijo iz odprtokodne knjižnice MULAN. Začeli smo z osnovnim konstruktorjem *HOMER(MultiLabelLearner mll, int clusters, HierarchyBuilder.Method method)*, ki ustvari novo instanco algoritma HOMER. Kot lahko vidimo iz konstruktorja, lahko nastavimo tri argumente. MultiLabelLearner predstavlja model. Ker je algoritem HOMER model, s katerim želimo delati, ne potrebujemo navesti tega parametra, saj je ta izbran privzeto. Ob izbiri modela HOMER lahko izberemo drugačen konstruktor, in sicer *HOMER()*, ki ne potrebuje dodatnih argumentov. Ko smo dobilo instanco algoritma HOMER, smo ji z metodo *build(trainDataSet)* določili učno množico primerov. Argument *trainDataSet* je naša učna množica primerov.

- BP-MLL: enako kot pri testiranju algoritma HOMER smo tudi pri testiranju algoritma BP-MLL uporabili odprtokodno implementacijo iz knjižnice MULAN. Za gradnjo osnovne instance smo uporabili konstruktor *BPMLL()*, ki smo mu nastavili lastnosti: z metodo *setHiddenLayers(new int[])* število skritih plasti, z metodo *setNormalizeAttributes(boolean normalize)* pa normalizacijo podatkov pred gradnjo klasifikatorja. Algoritmu BP-MLL smo na enak način kot pri algoritmu HOMER podali učno množico primerov, in sicer z metodo *build(trainDataSet)*, kjer je *trainDataSet* naša učna množica primerov.

- CLUS: za testiranje algoritma CLUS smo uporabili implementacijo univerze Katholieke universiteit Leuven in Instituta "Jožef Štefan". Algoritem lahko poganjamo v konzoli Java ali ga uporabimo znotraj kode Java. Osnovno instanco smo dobili z osnovnim konstruktorjem *Clus()*. Za kasnejšo gradnjo klasifikatorja potrebujemo še objekt *ClusInductionAlgorithmType*, ki smo ga nastavili na *ClusDecisionTree*, kar je osnovna izbira. Izbira tega parametra določa delovanje algoritma CLUS na način, ki smo ga opisali v podpoglavju 2.4.1. CLUS-klasifikator zgradimo z metodo *initialize(cargs, clss)*, kjer mora argument *cargs* vsebovati vsaj pot do datoteke z opisom atributov, ki jih bo klasiifikator napovedoval. Drugi argument metode *initialize(cargs, clss)* je *ClusInductionAlgorithmType*.

4.3 Opis podatkov

Za evalvacijo opisanih algoritmov smo uporabili podatke, ki so bili pridobljeni v okviru študije [20], v kateri je sodelovalo 272 dodiplomskih študentov različnih študijskih programov. Avtorji so v študiji preučevali povezanost med učnim stilom posameznika in preferencami po uporabi različnih multimedijskih vrst študijskih gradiv. V okviru študije so o vsakem izmed 272 študentov zapisali 31 atributov. Prvi je identifikacijska številka študenta, ki za to diplomsko delo ni relevantna. Sledijo trije demografski atributi (spol,

mesec rojstva in leto rojstva), nato še študijska smer in povprečna ocena. Naslednjih dvajset atributov predstavlja podatke o različnih učnih stilih (Kolbov učni stil, Rancourtov učni stil, Hemisferični učni stil in učni stil VAK), njihove dimenzije in nekatere razlike med njimi. Na koncu imamo še pet atributov, označenih od A1 do A5, ki predstavljajo ciljne atribute (atributi, ki jih algoritmi napovedujejo):

- A1: pogostost uporabe animacij in videogradiv,
- A2: pogostost uporabe simulacij in izobraževalnih računalniških iger,
- A3: pogostost uporabe besedil z barvno diskriminacijo (npr. pomembni elementi so barvno označeni),
- A4: pogostost uporabe strukturalnih gradiv, kjer si elementi sledijo v logičnem vrstnem redu,
- A5: pogostost uporabe avdiogradiv.

Razredni atributi od A1 do A5 lahko zavzamejo vrednosti od 1 do 4.

4.3.1 Diskretizacija podatkov in različni formati zapisa podatkov

Ker je vsak izmed razredov, v katere napovedujemo (od A1 do A4), predstavljen z nebinarnimi vrednostmi (od 1 do 4), smo za potrebe nekaterih algoritmov podatke diskretizirali. To smo naredili tako, da smo vsako vrednost $v, v \in \{1-4\}$ spremenili v 0, če je $v <= 2$, in v 1, če je $v > 2$. Ta diskretizacija je bila npr. potrebna pri algoritmu HOMER. Algoritme, ki diskretizacije ne potrebujejo, smo učili z dejanskimi atributi.

Pri uporabi algoritmov HOMER in BP-MLL smo zapisali datoteki v dveh formatih. Prvo v formatu *arff (Attribute-Relation File Format)* in drugo v formatu *xml (EXtensible Markup Language)*. Datoteka arff je tekstovna datoteka ASCII, ki opisuje seznam primerov z atributi [14]. V datoteki smo definirali množico primerov. V glavi datoteke smo definirali relacijo, seznam

atributov in njihov tip. Relacijo smo definirali z oznako @$RELATION$ in za vsak atribut definirali njegovo ime ter tip. Vsako definicijo novega atributa smo zapisali v novi vrstici. Primer vrstice: @$ATRIBUTE$ *letoRojstva numeric*. Za atribute, ki lahko zavzamejo samo določene vrednosti, definiramo množico možnih vrednosti. Primer vrstice: @$ATTRIBUTE$ $v1\_KOLB$ $\{AKOM, ASIM, DIVE, KONV\}$. Ker smo uporabili diskretizacijo podatkov, smo ciljne atribute od A1 do A5 definirali z množico možnih vrednosti $\{0, 1\}$. Za potrebe algoritma HOMER smo definirali XML-datoteko, ki vsebuje informacije o tem, kateri atributi so ciljni.

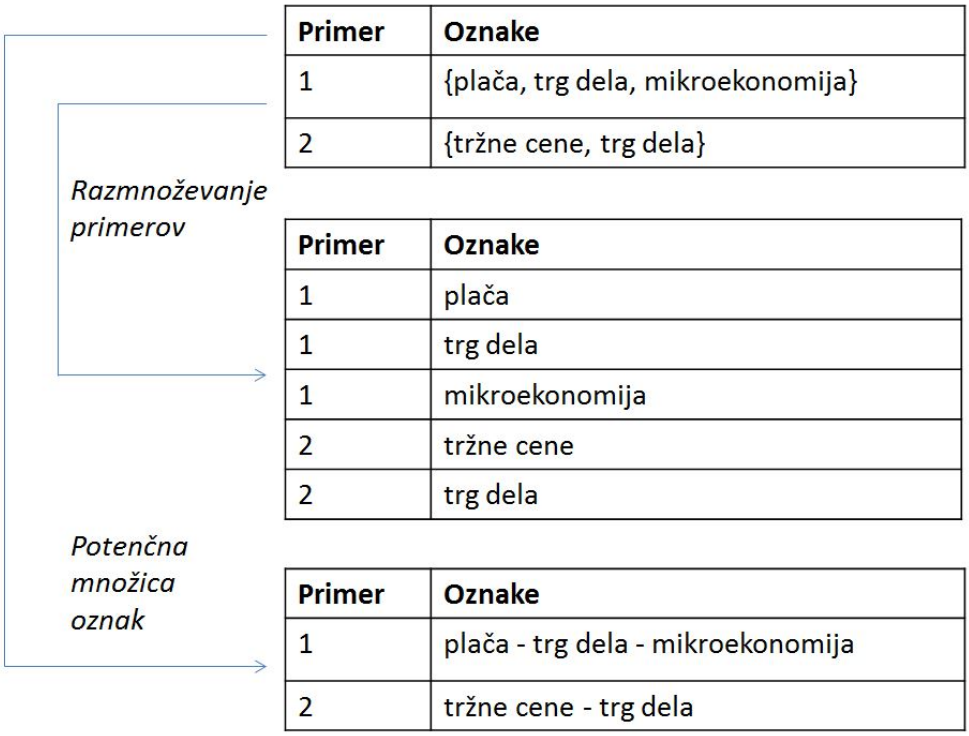
Pri uporabi algoritma CLUS smo morali zapisati datoteko v formatih *s* in *arff*. Datoteka v formatu *s* vsebuje nastavitve za algoritem in opis podatkov. V datoteki smo napisali, kateri atributi so ciljni in katere naj upošteva. Napisali smo, katera drevesa (porezano in neporezano) naj izriše in ali naj izpiše napovedi. Zgradba datoteke *arff* je pri uporabi algoritma CLUS enaka kot pri uporabi algoritma HOMER ali BP-MLL, le da podatkov nismo diskretizirali in je zato množica možnih vrednosti za ciljne atribute *numeric*.

4.4 Transformacija podatkov za enoznačno klasifikacijo

Ker podatki niso primerni za uporabo enoznačne klasifikacije, smo uporabili tri metode za transformacijo podatkov, ki so primerni za enoznačno klasifikacijo. Takšni podatki so potrebni za gradnjo enoznačnih klasifikatorjev, ki smo jih uporabili pri primerjavi različnih načinov klasifikacije.

Tsoumakas and Katakis [15] delita metodo transformacije za enoznačno klasifikacijo na dve kategoriji: (a) prilagoditev algoritmov in (b) transformacija problema. Z adaptacijo algoritmov spremenimo algoritem tako, da lahko deluje s podatki za večznačno klasifikacijo. Transformacija problema pa prilagodi podatke v enega ali več enoznačnih problemov [10].

Te metode omogočajo, da problem večznačne klasifikacije razdelimo na



Slika 4.1: Prikaz uporabe dveh tehnik transformacij podatkov [17]

več manjših podproblemov in nato združimo predikcije podproblemov za končno klasifikacijo [16].

Slika 4.1 je prikaz uporabe dveh tehnik transformacij podatkov. Prikazuje metodo razmnoževanja primerov (angl. instance-copy) in metodo potenčne množice oznak (angl. label powerset).

V nadaljevanju so opisane uporabljene transformacijske metode.

4.4.1 Eden proti vsem

Metodo transformacije podatkov eden proti vsem (angl. one-vs-all) smo uporabili tako, da smo zgradili toliko klasifikacijskih modelov, kolikor je razredov. Na takšen način smo dobili rezultate, kot bi jih dobili pri večznačni

klasifikaciji.

4.4.2 Razmnoževanje primerov

Metoda razmnoževanje primerov (angl. instance-copy) zamenja vsak večznačen primer (x_i, Y_i) z $|Y_i|$ primeri (X_i, λ_i), za vsak $\lambda_i \in Y_i$. Enoznačno klasifikacijo, ki nam vrne distribucijo razredne verjetnosti, lahko uporabimo za učenje in predikcijo relevantnih oznak za naše primere [17].

4.4.3 Potenčne množice oznak

Osnova metode potenčne množice oznak (angl. label powerset) je združevanje celotne skupine oznak (enega primera) v eno oznako. Na tak način dobimo problem z enoznačnimi podatki. Množica možnih oznak za enoznačno klasifikacijo so vse unikatne oznake primerov (združene oznake). Problem, ki se lahko pojavi pri tej metodi, je veliko število možnih oznak. To lahko rešimo na tak način, da izberemo samo tiste oznake, ki se pojavijo več kot t-krat, kjer je t neko število, ki ga določimo pred izvedbo transformacije [10].

4.5 Evalvacija

Med razvojem smo implementirane algoritme testirali na podatkih, ki so jih ponudili na Courseri. Na tak način smo lahko preverili, ali implementirane metode delujejo pravilno. Ko smo se prepričali, da je implementacija metode dobra, smo jo uporabili za klasifikacijo podatkov. Preden smo lahko izvedli klasifikacijo podatkov, smo odstranili prvo vrstico (kjer so imena posameznih atributov) in prvi stolpec (identifikacijske številke študentov).

Glede na to, da je implementacija algoritma primerna za enoznačno klasifikacijo, podatki pa za večznačno klasifikacijo, smo v Matlabu implementirali funkciji *getCopyInstance* in *getLabelPowerSet*. Funkcija *getCopyInstance* zgradi matriko primerov, kjer ima vsak primer samo eno oznako (torej primerno za enoznačno klasifikacijo). Funkcija deluje po metodi *razmnoževanja*

primerov (kot je razvidno iz imena funkcije). Tudi funkcija *getLabelPowerSet* zgradi matriko primerov, primerno za enoznačno klasifikacijo, in deluje po metodi *potenčne množice oznak*. Za uporabo metode *eden proti vsem* nismo napisali posebne funkcije, temveč smo jo naredili v glavni skripti.

Da smo lahko preverili delovanje algoritma in ga primerjali z ostalimi algortimi, smo izračunali mere uspešnosti, za kar smo uporabili funkcijo *Evaluate(ACTUAL,PREDICTED)* [18], ki sprejema dva argumenta. *ACTUAL* je matrika pravilnih oznak, *PREDICTED* pa matrika napovedanih oznak. Funkcija *Evaluate* vrne naslednje mere uspešnosti: točnost, preciznost, priklic in F-mero.

Zaradi kredibilnosti oz. zanesljivosti algoritmov smo uporabili 10-kratno prečno preverjanje. To pomeni, da smo predikcijski model zgradili desetkrat, in sicer vsakič na drugih 9/10 podatkov in ga preverili na 1/10 ostalih podatkov. Vsakič smo izračunali mere uspešnosti in na koncu povprečje. Na tak način smo dobili manj pristrane rezultate zaradi povprečenja preko različnih možnih delitev na učno in testno množico ter s tem izogibanja prevelikemu prileganju. Na podoben način smo uporabili in preverili vse uporabljene metode.

Algoritme za večznačno in večciljno klasifikacijo smo uporabili v programskem jeziku Java. Preden smo lahko uporabili podatke, smo jih zapisali v formatu *arff* in pripravili datoteki *xml* in *s*.

4.6 Rezultati

V tem poglavju so predstavljeni rezultati in primerjave algoritmov, ki smo jih uporabili v diplomskem delu. Začnemo s primerjavo enoznačnih klasifikatorjev glede na različne transformacije v enoznačne podatke. Vsako izmed mer uspešnosti prikažemo v svoji tabeli in meritve analiziramo. Nato primerjamo algoritme za večznačno klasifikacijo, na koncu pa še analiziramo algoritme za večciljno klasifikacijo. Zaključimo z interpretacijo rezultatov. Vse rezultate smo pridobili z diskretiziranimi podatki zaradi možnosti primerjave (ker je

	Točnost		
	one-vs-all	copy-instance	label powerset
kNN	0.707	0.481	0.953
naivni Byes	0.730	0.682	0.835
odločitvena drevesa	0.691	0.711	**0.982**
nevronske mreže	0.768	0.679	0.981

Tabela 4.1:

pri algoritmu HOMER in BP-MM diskretizacija nujno potrebna).

Tabela 4.1 prikazuje algoritme za enoznačno klasifikacijo pri različnih metodah transformacije za mero uspešnosti *točnost*. Točnost je ena izmed osnovnih mer uspešnosti in nam pove, kolikšen delež oznak je bil pravilno napovedan. V tabeli lahko opazimo, da se je najbolje odrezala metoda *potenčne množice oznak* z algoritmom odločitvenih dreves, druga najboljša metoda je *eden proti vsem* z algoritmom nevronske mreže. Stopnja točnosti je najmanjša pri metodi *razmnoževanja primerov*. V povprečju se je najbolje odrezal algoritem nevronske mreže.

Tabela 4.2 prikazuje algoritme enoznačne klasifikacije pri različnih metodah transformacije za mero uspešnosti *preciznost*. V tabeli lahko opazimo, da se je najbolje odrezala metoda *potenčne množice oznak* z algoritmom odločitvenih dreves, druga najboljša metoda je *eden proti vsem* pri uporabi z nevronsko mrežo. Nizke stopnje preciznosti ne vidimo pri nobeni metodi. V poprečju se je najbolje odrezal algoritem nevronske mreže.

Tabela 4.3 prikazuje algoritme enoznačne klasifikacije pri različnih metodah transformacije za mero uspešnosti *priklic*. V tabeli lahko opazimo, da se je najbolje odrezala metoda *potenčne množice oznak* z algoritmom odločitvenih dreves in nevronske mreže, druga najboljša je metoda *razmnoževanja primerov* z algoritmom odločitvenih dreves. Nizko stopnjo priklica opazimo pri algoritmu k najbližjih sosedov pri metodi *razmnoževanja primerov*. V poprečju se je najbolje odrezal algoritem odločitvenih dreves.

	Preciznost		
	one-vs-all	copy-instance	label powerset
kNN	0.732	0.737	0.982
naivni Byes	0.742	0.743	0.860
odločitvena drevesa	0.689	0.731	**0.982**
nevronske mreže	0.782	0.735	0.981

Tabela 4.2:

	Priklic		
	one-vs-all	copy-instance	label powerset
kNN	0.716	0.450	0.970
naivni Byes	0.767	0.862	0.851
odločitvena drevesa	0.777	0.956	**1.000**
nevronske mreže	0.788	0.879	**1.000**

Tabela 4.3:

	F-mera		
	one-vs-all	copy-instance	label powerset
kNN	0.722	0.551	0.975
naivni Byes	0.751	0.795	0.855
odločitvena drevesa	0.729	0.828	**0.990**
nevronske mreže	0.783	0.799	0.990

Tabela 4.4:

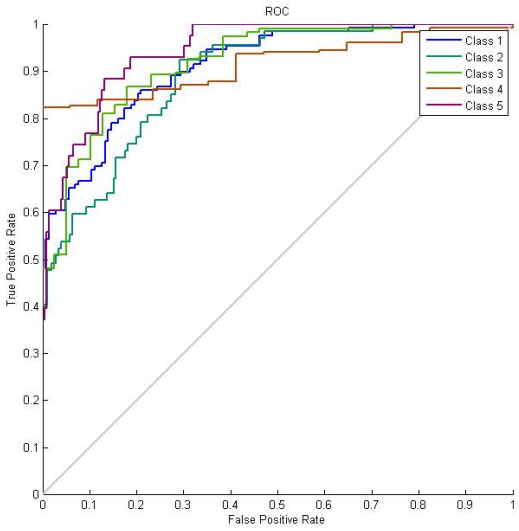
	HOMER	BP-MLL	CLUS
Točnost	0.853	0.877	0.897
Preciznost	0.891	0.952	0.997
Priklic	0.951	0.919	0.810
F-mera	0.903	0.920	0.893

Tabela 4.5:

Tabela 4.4 prikazuje algoritme za enoznačno klasifikacijo pri različnih metodah transformacije za mero *F-mera*. F-mera je definirana kot harmonična sredina med natačnostjo in priklicem. Najboljšo vrednost ima v 1 in najslabšo v 0. V tabeli lahko opazimo, da se je najbolje odrezala metoda *potenčne množice oznak* pri uporabi z odločitvenimi drevesi, druga najboljša je metoda *razmnoževanja primerov* z algoritmom odločitvenih dreves. Nizko stopnjo priklica opazimo pri algoritmu k najbližjih sosedov pri metodi *razmnoževanja primerov*. V povprečju so se najbolje odrezale nevronske mreže.

Tabela 4.5 prikazuje rezultate algoritmov večznačne klasifikacije in večciljne klasifikacije ter njihove mere uspešnosti. Najbolj točen in precizen algoritem je CLUS, najbolj priklican HOMER, najboljšo F-mero pa ima BP-MLL.

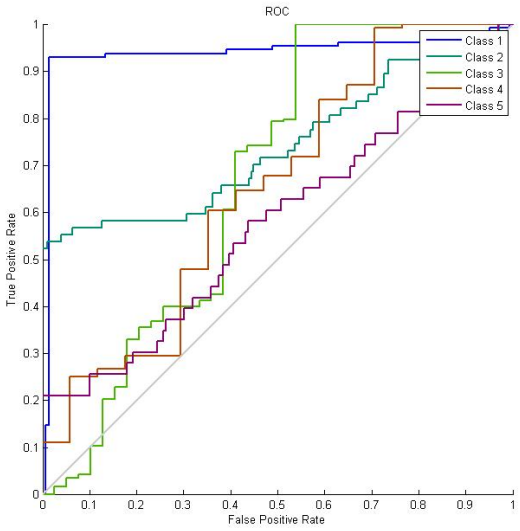
Slika 4.5 predstavlja ROC-krivulje za CLUS-algoritem. Po pričakovanjih ta algoritem kot ostali napoveduje v več razredov. S slike je razvidno, da algoritem dobro napoveduje v štiri razrede.



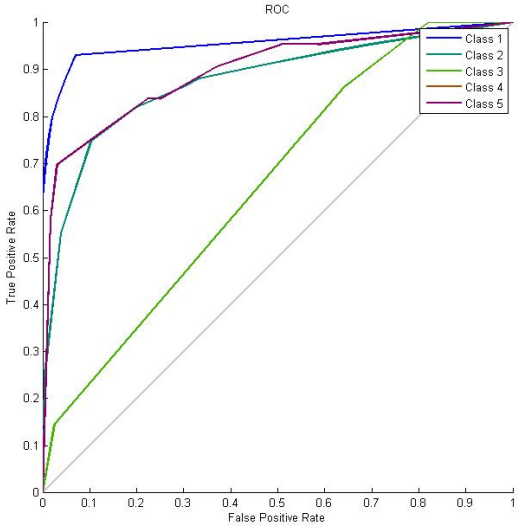
Slika 4.2: ROC-krivulja za enoznačno nevronsko mrežo

Ker so se v povprečju najbolje odrezale nevronske mreže, bomo primerjali nevronske mreže z algoritmi HOMER, BP-MLL in CLUS tako, da bomo izrisali ROC-krivulje. Za vsak algoritem podajamo eno sliko, na kateri so štiri ROC-krivulje, za vsak razred ena. Slika 4.2 prikazuje ROC-krivulje za nevronske mreže. S slike vidimo, da algoritem napoveduje dobro v en razred. Slika 4.3 prikazuje ROC-krivulje za algoritem BP-MLL. S slike vidimo, da algoritem dobro napoveduje v en razred, slabo pa v štiri in pet razredov. Slika 4.4 prikazuje ROC-krivulje za algoritem HOMER. S slike vidimo, da algoritem dobro napoveduje v en razred, slabo pa v štiri in pet razredov. Slika 4.5 prikazuje ROC-krivulje za algoritem CLUS. Po pričakovanjih ta algoritem bolje kot ostali napoveduje v več razredov. Vidimo zelo dobro napovedovanje v štiri razrede.

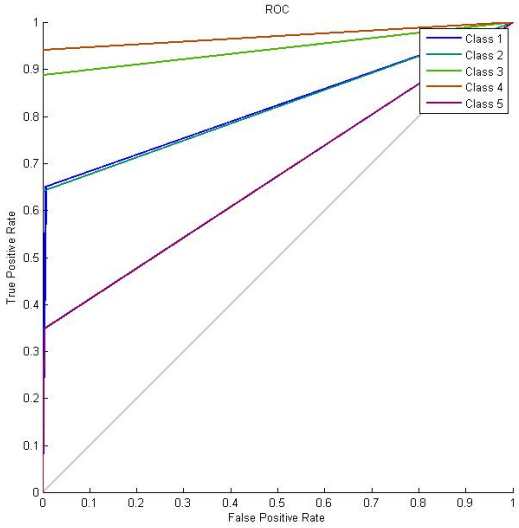
Kljub jasnosti dobljenih rezultatov je treba opozoriti, da lahko pri večji podatkovni množici dobimo drugačne rezultate. Razlog za to je v že prej omenjeni metodi transformacije *potenčne množice oznak*, ki pri veliki množici podatkov slabše deluje zaradi veliko različnih oznak.



Slika 4.3: ROC-krivulja za algoritem BP-MLL



Slika 4.4: ROC-krivulja za algoritem HOMER



Slika 4.5: ROC-krivulja za algoritem CLUS

Poglavje 5

Zaključek

V tem diplomskem delu smo se ukvarjali s primerjavo metod za večciljno in večznačno klasifikacijo. Ti dve vrsti klasifikacije smo simulirali s pristopi za enoznačno klasifikacijo (nevronske mreže, naivni Bayes, odločitvena drevesa in k najbližjih sosedov), testirali specializirane algoritme za večznačno klasifikacijo (HOMER in BP-MLL) in specializirane algoritme za večciljno klasifikacijo (CLUS). Rezultati so pokazali, da metoda CLUS in simulacija z enoznačno metoda nevronsko mrežo delujeta najbolje.

Kot sklep lahko poudarimo, da smo dobili podobne uspešnosti učenja, če uporabljamo metode za transformacijo podatkov v enoznačno klasifikacijo ali pa specializirane algoritme za ta namen. Kadar je število razredov večje, pokažejo boljše napovedovanje specializirani algoritmi za večciljno klasifikacijo, kot je CLUS.

Možno nadaljnje delo na tem področju vključuje primerjavo uporabe dobljenih smernic tudi na podatkih iz drugih problemskih domen in evalvacijo tudi drugih možnih algoritmov za večznačno in večciljno klasifikacijo.

Literatura

[1] Mehmed Kantardzic, "Decision Trees and Decision Rules', v knjigi: "Data Mining Concepts, Models, Methods, and Algorithms", A John Wiley and sons, inc., 2003.

[2] Mohammad S. Sorower , "A Literature Survey on Algorithms for Multi-label Learning", Oregon State University. Dostopno na: http://people.oregonstate.edu/ sorowerm/pdf/Qual-Multilabel-Shahed-CompleteVersion.pdf

[3] Petra Kralj, "Naivni Bayesov klasifikator", Inštitut Jožef Štefan. Dostopno na:
http://kt.ijs.si/PetraKralj/UNGKnowledgeDiscovery/Bayes.pdf

[4] "KNN Algorithms', v priročniku: "SPSS statictics", IBM. Dostopno na: http://pic.dhe.ibm.com/infocenter/spssstat/v20r0m0/index.jsp?topic=%2Fcom.ibm.spss.statistics.help%2Falg_knn.htm

[5] Dr. N. Ganesan ANSI, Dr.K. Venkatesh, Dr. M. A. Rama, A. Malathi Palani, "Application of Neural Networks in Diagnosing Cancer Disease Using Demographic Data", v reviji: International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 26, Dostopno na: http://www.ijcaonline.org/journal/number26/pxc387783.pdf

[6] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas, "Effective and Eficient Multilabel Classification in Domains

with Large Number of Labels", Aristotle University of Thessaloniki. Dostopno na: http://bradblock.com.s3-website-us-west-1.amazonaws.com/Effective_and_Efficient_Multilabel_Classification_in_Domains_with_Large_Number_of_Labels.pdf

[7] Hendrik Blockeel, Leander Schietgat, Jan Struyf, Amanda Clare, Sašo Džeroski, "Hierarchical Multilabel Classification Trees for Gene Function Prediction", Katholieke Universiteit Leuven, University of Wales, Jozef Stefan Institute, University of Wisconsin. Dostopno na: http://eprints.pascal-network.org/archive/00003710/01/42257.pdf

[8] Inštitut Jožef Štefan, "Uradna spletna stran", Inštitut Jožef Štefan, Dostopno na: http:/http://www.ijs.si/

[9] Marios Ioannou, George Sakkas, Grigorios Tsoumakas and Ioannis Vlahavas, "Obtaining Bipartitions from Score Vectors for Multi-Label Classification", Aristotle University of Thessaloniki. Dostopno na: http://lpis.csd.auth.gr/publications/ioannou-ictai10.pdf

[10] Gjorgji Madjarov and Dragi Kocev and Dejan Gjorgjevikj and Sašo Džeroski, "An extensive experimental comparison of methods for multi-label learning", Methodius University, University of Wales, Jozef Stefan Institute. Dostopno na: ttp://www.sciencedirect.com/science/article/pii/S0031320312001203

[11] MathWorks, "MATLAB", Dostopno na: http://www.mathworks.com/products/matlab/

[12] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, "Mulan: A Java Library for Multi-Label Learning", Dostopno na: http://mulan.sourceforge.net/

[13] Oracle, "Java", Dostopno na: http://java.com/

[14] WEKA, "ARFF (book version)", The University of Waikado, Dostopno na: http://weka.wikispaces.com/ARFF

[15] G. Tsoumakas, I. Katakis, "Multi label classification: an overview", International Journal of Data Warehouse and Mining 3, 2007

[16] Yi Zhang, Jeff Schneider, "A Composite Likelihood View for Multi-Label Classification", Carnegie Mellon University. Dostopno na: http://www.cs.cmu.edu/ schneide/Composite_V2.pdf

[17] Sergeja Vogrinčič, Zoran Bosnić, "Ontology-based multi-label classification of economic articles", v zbirki: "Computer Science and Information Systems, Volume 8, Number 1", ComSIS, 2011

[18] Barnan Das, "Performance Measures for Classification", The MathWorks, Inc., Dostopno na: http://www.mathworks.com/matlabcentral/fileexchange/37758-performance-measures-for-classification/content/Evaluate.m

[19] José Hernández-Orallo, "ROC curves for regression, Pattern Recognition", Volume 46, Issue 12, December 2013, Pages 3395-3411, ISSN 0031-3203, Dostopno na: http://dx.doi.org/10.1016/j.patcog.2013.06.014

[20] Uroš Ocepek, Zoran Bosnić, Irena Nančovska Šerbec, Jože Rugelj, "Exploring the Relation between Learning Style Models and Preferred Multimedia Types", Computers & Education, 2013, vol. 69, str. 343-355.