UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matjaž Majnik

Nadgradnja mere AUC pri analizi klasifikatorjev s krivuljami ROC

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2011

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01767/2011
Datum: 02.09.2011

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

Kandidat: **MATJAŽ MAJNIK**

Naslov: **NADGRADNJA MERE AUC PRI ANALIZI KLASIFIKATORJEV S KRIVULJAMI ROC**

IMPROVEMENT OF THE AUC METRIC IN CLASSIFIER ROC ANALYSIS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Površina pod krivuljo ROC (Receiver Operating Characteristics curve) se v strojnem učenju najpogosteje uporablja za ocenjevanje avtomatsko zgrajenih modelov. Z izračunom površine pod krivuljo (AUC) lahko kvantitativno primerjamo točnosti modelov. Zaradi pristranosti te mere v odvisnosti od učnih podatkov so raziskovalci predlagali njene izboljšave.

Kandidat naj v diplomskem delu naredi pregled izboljšav osnovne mere AUC in naj analizira njihove prednosti in slabosti. V nadaljevanju naj sam razvije in predlaga novo mero, ki odpravlja nadalje šibkosti opisanih izboljšav. Vse mere naj medseboj tudi primerja in ovrednoti na umetnih problemih.

Mentor:

doc. dr. Zoran Bosnić

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Matjaž Majnik,

z vpisno številko 63020101,

sem avtor/-ica diplomskega dela z naslovom:

Nadgradnja mere AUC pri analizi klasifikatorjev s krivuljami ROC

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a 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 30. 09. 2011 Podpis avtorja/-ice:

Zahvala

Rad bi se zahvalil mentorju doc. dr. Zoranu Bosniću za motivacijo, strokovne nasvete ter njegovo dostopnost tekom nastajanja tega dela.

Zahvaljujem se Marini, svojemu dekletu, za vso podporo in veliko mero potrpljenja.

Obenem se zahvaljujem mami in očetu za njuno razumevanje in pomoč tekom študija.

Mojemu atiju v spomin.
Neizmerno sem hvaležen za vso pozornost, ki si mi jo namenil v življenju. Vedno boš v mojem srcu.

Kazalo

Seznam uporabljenih kratic in simbolov

ROC - receiver operating characteristics
AUC - area under the ROC curve
TP - true positive
FP - false positive
TN - true negative
FN - false negative
TPR - true positive rate
FPR - false positive rate
TNR - true negative rate
FNR - false negative rate
probAUC - probabilistic AUC
scorAUC - scored AUC
sondAUC - softened AUC
softAUC - soft AUC

Povzetek

Mera AUC, ki se uporablja na področju vrednotenja klasifikatorjev in predstavlja eno glavnih orodij analize ROC, ima določene pomanjkljivosti. Ne upošteva namreč vrednosti točkovnih ocen (angl. *scores*) primerov, temveč le njihovo razvrstitev. Posledica tega dejstva je njena nezanesljivost pri ocenjevanju množic primerov, pri katerih so razlike med točkovnimi ocenami primerov zanemarljive. Slabost mere AUC pa je tudi njena neinformativnost pri medsebojnem primerjanju množic, ki vsebujejo enako število napak.

Raziskovalci so iz teh razlogov predlagali izboljšave mere AUC, ki upoštevajo tudi vrednosti točkovnih ocen. V tem delu obravnavamo štiri tovrstne mere. Ugotovljeno pa je bilo, da tudi te izpeljanke ne odpravijo vseh slabosti oz. celo vpeljejo nove. Pri njih se namreč lahko pojavi neprimeren vpliv lastnosti obravnavanih množic primerov na obnašanje teh različic.

Temeljni namen tega diplomskega dela je bil iz osnovne AUC in njenih izpeljank pridobiti novo mero, ki bi zgoraj naštete pomanjkljivosti odpravila in tako bolj informativno in obenem tudi verodostojno ocenjevala sposobnosti klasifikatorjev. Pri tem smo nameravali nadzirati in prilagoditi tudi omenjene vplive množic primerov.

Poglavitni rezultat diplomskega dela je torej nova mera za ocenjevanje klasifikatorjev, ki je bila preizkušena v nadzorovanem okolju. Glede na raznolikost točkovnih ocen, dobljenih od klasifikatorjev, smo z namenom splošne uporabnosti nove mere v njeno formulo vgradili parametre, s katerimi je mogoče prilagajati njeno delovanje. Na podlagi predstavljene končne primerjave različic ocenjujemo, da ima novopredlagana mera, ob upoštevanju nekaterih predpostavk, določene prednosti pred obstoječimi različicami.

Ključne besede:

mera AUC, izboljšava, krivulje ROC, vrednotenje klasifikatorjev

Abstract

AUC measure, which is used for classifier evaluation and represents one of the main tools of ROC analysis, has certain shortcomings. It ignores scores of instances and considers only their ranking order. The consequence is its unreliability in evaluating instance sets, where the differences between scores are negligible. Another disadvantage of the AUC is its low informative quality when mutually comparing sets containing the same number of errors.

For these reasons, researchers have proposed improvements of the AUC, which also take score values into account. In this thesis we address four such measures. However, it has been ascertained that they do not solve all the problems and may even introduce new ones. In the case of these variants, the improper influence of the attributes of considered instance sets on their behavior may arise.

The essential purpose of this thesis has been to acquire a new measure from the basic AUC and its derived variants eliminating the disadvantages mentioned and offering trustworthy and more informative evaluation of classifier performance. To achieve the goal, we also have intended to control and adjust the abovementioned influences of the instance sets.

Thus, the main result of this thesis is the new measure for the evaluation of classifiers, which has been tested in supervised environment. Considering the diversity of scores obtained from different classifiers we have included parameters allowing us to adjust the performance of the new measure. As a result, the measure should be more generally applicable. On the basis of final comparison of the variants and taking some presumptions into account we believe that the newly proposed measure has certain advantages over the existing variants.

Key words:

AUC measure, improvement, ROC curves, classifier evaluation

Poglavje 1

Uvod

Analiza ROC (angl. *Receiver Operating Characteristics*, karakteristika sprejemnika) je metodologija za vrednotenje, primerjanje in izbiro klasifikatorjev na osnovi njihovih sposobnosti napovedovanja. Njeno glavno orodje predstavljajo krivulje ROC ter mera AUC. Krivulja ROC je krivulja na dvodimenzionalnem grafu, ki prikazuje sposobnost klasifikatorja za izdajanje dobrih točkovnih ocen (angl. *scores*). Ker je vizualno primerjanje krivulj lahko težavno, se je začela uporabljati mera AUC, ki v obliki številske informacije povzema posamezne krivulje ROC. Primerjanje kakovosti klasifikatorjev se torej omeji na primerjanje številskih vrednosti.

Obravnava kakovosti klasifikatorjev je pomembna, saj so le-ti danes prisotni na širokem spektru področij, od strojnega učenja, podatkovnega rudarjenja, računalniškega vida ter drugih vej umetne inteligence do ekonomije (med drugim bančništva in zavarovalništva), družbenih ved in medicine.

Mera AUC ima pomanjkljivosti. Izkaže se kot nezanesljiva pri vrednotenju klasifikatorjev, ki primere v določeni množici ocenijo z zelo podobnimi točkovnimi ocenami. Poleg tega mera AUC ni uporabna pri medsebojnem primerjanju klasifikatorjev, ki na dani množici primerov naredijo enako število napak. Opisani lastnosti izhajata iz dejstva, da mera AUC ne upošteva samih velikosti točkovnih ocen primerov, temveč le njihovo razvrstitev.

Z namenom odprave zgornjih pomanjkljivosti so se pojavile mere, izpeljane iz mere AUC, ki velikosti točkovnih ocen upoštevajo. Kot se je izkazalo, te izpeljanke težave rešijo le delno, saj vpeljejo nekatere nove slabosti.

Namen tega dela je iz osnovne mere AUC oziroma njenih že obstoječih izpeljank razviti novo mero, ki bi odpravila omenjene pomanjkljivosti. Pri tem želimo pridobiti bolj informativno in hkrati verodostojno metodo za ocenjevanje sposobnosti klasifikatorjev. Zaradi splošnosti pri ocenjevanju klasifikatorjev

z mero AUC in njenimi različicami je lahko tudi sam pojem kakovosti klasifikatorja precej subjektiven. Da bi ta problem omilili, naredimo nekaj predpostavk. Vse obravnavane mere podrobneje preučimo in opazujemo njihovo obnašanje na konkretnih množicah primerov. Pri tem bolj natančno določimo njihove slabosti. Analiziramo tudi lastnosti množic primerov ter vplive teh lastnosti na obravnavane različice. Pri razvoju nove mere te vplive poskušamo nadzirati in prilagajati.

Delo je urejeno na sledeč način. Poglavje 2 vsebuje uvod v področje analize ROC. V poglavju 3 so predstavljene štiri že obstoječe različice osnovne mere AUC, ki temeljijo na točkovnih ocenah, poglavje 4 pa vsebuje opis različice, ki je bila predlagana v okviru tega dela. Primerjava uspešnosti vseh obravnavanih različic se nahaja v poglavju 5. V poglavju 6 so strnjene sklepne ugotovitve, v dodatku A pa je prikazana izvorna koda testnega programa.

Poglavje 2

Analiza ROC

Prva znana uporaba analize ROC izhaja iz obdobja druge svetovne vojne, ko je ta predstavljala pripomoček za obdelavo radarskih signalov. Njena uporaba se je pozneje nadaljevala v teoriji detekcije signalov za ponazoritev kompromisa med pogostostjo zadetka (angl. *hit rate*) in pogostostjo lažnega alarma (angl. *false alarm rate*) klasifikatorjev [8, 4]. Ostala področja, v katera je analiza ROC bila vpeljana, so med drugim psihofizika [8], medicina (različne tehnike medicinske vizualizacije za namene diagnosticiranja, vključno z računalniško tomografijo, mamografijo, rentgenom [15] in magnetno resonanco [13], ter razne metode v epidemiologiji [11]) in družbene vede. Objavljen je bil obširen seznam virov s področja analize ROC, ki obravnavajo odločanje v medicini. Ta seznam vsebuje preko 350 vpisov, ki so razdeljeni na več razdelkov [19].

Več kot desetletje analiza ROC bolj izrazito pridobiva na priljubljenosti tudi na področju umetne inteligence. Prvi primeri uporabe segajo v pozna osemdeseta leta prejšnjega stoletja, ko je bila prikazana koristnost krivulj ROC (podrobneje opisane v nadaljevanju) pri ocenjevanju algoritmov [14]. Krivulje ROC so med tem postale priljubljen način za ocenjevanje algoritmov strojnega učenja. Uvod v uporabo analize ROC v raziskovanju je predstavljen v [5].

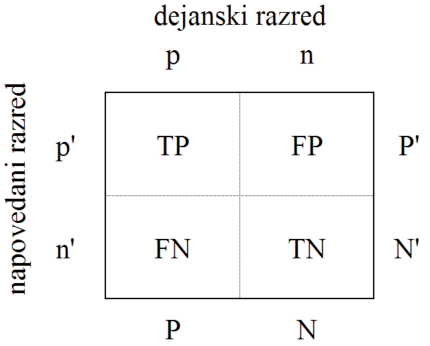
Raziskave na področju analize ROC so se živahno odvijale v različnih smereh. Med drugim je bila tako dvorazredna metodologija ROC posplošena na obravnavo večrazrednih problemov, predlagane so bile razširitve osnovnih krivulj ROC in osnovne mere AUC, pojavile so se druge oblike vizualizacije kot alternative osnovnim grafom ROC itd.

Analiza ROC v svoji izvirni dvorazredni obliki se uporablja za obravnavo dvorazrednih klasifikacijskih problemov. Podana mora biti množica primerov z znanimi razredi, v kateri vsak primer pripada pozitivnemu ali negativnemu razredu. Termina *pozitivni* in *negativni* izhajata iz zgodnje uporabe v medicini.

Tam so se primeri, ki so opisovali paciente, pri katerih je bil določen medicinski pojav zaznan (npr. bolezen), označevali kot pozitivni, preostali primeri pa kot negativni. Po zaključeni učni fazi naj bi bil klasifikator sposoben napovedati vrednost razreda pri novih (še nevidenih) primerih.

2.1 Kontingenčna tabela

Napovedani razredi danih primerov ne ustrezajo nujno dejanskim razredom. V kolikor se to zgodi, tak dogodek imenujemo napovedna napaka. Za predstavitev števila pravilno in napačno klasificiranih primerov se uporablja matrika. Ta matrika se imenuje *kontingenčna tabela* (angl. *contingency table* ali *confusion matrix*) in je prikazana na sliki 2.1. Pri klasifikaciji vsakega primera so možni štirje izidi. Primer, ki je pozitiven in je kot takšen tudi klasificiran, imenujemo *pravilno uvrščen pozitivni primer* (angl. *true positive*, *TP*). V kolikor je klasifikator naredil napako in pozitiven primer klasificiral kot negativnega, le-tega imenujemo *napačno uvrščen pozitivni primer* (angl. *false negative*, *FN*). Podobno, če je primer negativen in ga je tudi klasifikator prepoznal kot negativnega, ga označimo kot *pravilno uvrščen negativni primer* (angl. *true negative*, *TN*), v primeru napačne klasifikacije pa kot *napačno uvrščen negativni primer* (angl. *false positive*, *FP*). Pravilne odločitve klasifikatorja torej ležijo na glavni diagonali kontingenčne tabele, medtem ko ostali elementi tabele predstavljajo število napačnih klasifikacij.



Slika 2.1: Struktura kontingenčne tabele za binarne klasifikacijske probleme

Na osnovi kontingenčne tabele se izračunajo tudi druge mere za vrednotenje znanja, vključno s *TPR* (angl. *true positive rate*, delež pravilno uvrščenih pozitivnih primerov med vsemi pozitivnimi primeri) in *FPR* (angl. *false positive rate*, delež napačno uvrščenih negativnih primerov med vsemi negativnimi

primeri), ki sta opredeljeni na sliki 2.2. Na nekaterih področjih se TPR imenuje tudi *senzitivnost* oz. *občutljivost* (angl. *sensitivity*, *recall*), namesto *TNR* (angl. *true negative rate*, delež pravilno uvrščenih negativnih primerov med vsemi negativnimi primeri) pa se uporablja izraz *specifičnost* (angl. *specificity*).

$$\text{senzitivnost} = \text{TPR} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{FPR} = \frac{\text{FP}}{\text{N}}$$
$$\text{specifičnost} = \text{TNR} = \frac{\text{TN}}{\text{N}} = \frac{\text{TN}}{\text{FP} + \text{TN}} = 1 - \text{FPR}$$
$$\text{FNR} = \frac{\text{FN}}{\text{P}}$$

Slika 2.2: Mere učinkovitosti pomembne v analizi ROC

2.2 Grafi ROC in krivulje ROC

Graf ROC za izvirne dvorazredne probleme je opredeljen kot dvodimenzionalen diagram, ki prikazuje TPR (senzitivnost) na y osi v odvisnosti od FPR (=1-specifičnost) na x osi. Učinkovitost posameznega klasifikatorja, podanega z njegovima senzitivnostjo in specifičnostjo, je predstavljena kot točka na grafu ROC. Na tovrstnem grafu imamo nekaj osnovnih značilnih točk. Točka s koordinatami (0,0) ($TPR = 0$, $FPR = 0$) predstavlja klasifikator, ki nikoli ne napove pozitivnega razreda. Kljub temu, da tovrsten klasifikator nikoli napačno ne klasificira negativnega primera kot pozitivnega, ni dobra izbira, saj prav tako ne opravi nobene pravilne klasifikacije pozitivnega primera. Njegov sorodnik v točki (1,1) predstavlja nasprotno stanje ($TPR = 1$, $FPR = 1$), saj vse primere klasificira kot pozitivne, tako pa ustvari tudi visoko število napačno uvrščenih negativnih primerov. Klasifikatorja v točkah $(0, 0)$ in $(1, 1)$ se imenujeta tudi *privzeta klasifikatorja* (angl. *default classifiers*). V točki $(0, 1)$ se nahaja popoln klasifikator ($TPR = 1$, $FPR = 0$). Od klasifikatorja, ki deluje na resničnih podatkih, takšne učinkovitosti seveda ni mogoče pričakovati. Kljub temu pa obnašanje popolnega klasifikatorja predstavlja oporno točko, h kateri naj bi stremeli pri gradnji klasifikatorjev. Klasifikatorji, ki se nahajajo na glavni diagonali grafa ROC (ob predpostavki uravnotežene porazdelitve po razredih oz. enake cene napačnih klasifikacij), imajo enako učinkovitost, kot

jo ima naključno ugibanje. Za takšne klasifikatorje rečemo, da nimajo nobene informacije o problemu. Uporabni klasifikatorji ležijo nad glavno diagonalo. Tisti pod njo se obnašajo slabše od naključnega ugibanja. Omenjeni klasifikatorji pa lahko postanejo uporabni na zelo enostaven način, in sicer tako, da obrnemo njihove napovedi. Za takšne klasifikatorje rečemo, da imajo uporabno informacijo, vendar pa jo uporabljajo na napačen način [7].

Krivulja ROC je krivulja na grafu ROC z začetkom v točki $(0, 0)$ in koncem v točki $(1, 1)$. Postopek risanja te krivulje se razlikuje glede na vrsto klasifikatorjev, ki jih želimo oceniti. Klasifikatorje lahko glede na količino vrnjene informacije v grobem razdelimo na dve skupini: diskretne ter verjetnostne (angl. *probability estimating*) oz. točkovalne (angl. *scoring*). Za vsak primer diskretni klasifikator vrne samo napoved razreda, medtem ko verjetnostni klasifikator vrne vrednost, ki je lahko verjetnost razreda (v strogem pomenu) ali pa točkovna ocena (angl. *score*) razreda (torej nekalibrirana). Pri tem napoved razreda vsebuje najmanj informacije, napoved verjetnosti razreda pa največ. Napoved točkovne ocene razreda je po informativnosti nekje vmes. Razlog za uporabo točkovnih ocen je v tem, da v nekaterih primerih ni mogoče dobiti dobrih ocen verjetnosti (npr. v primeru majhne količine učnih podatkov). Pomen točkovnih ocen je pri tem naslednji: če klasifikator vrne točkovni oceni za dva primera, kjer je ocena prvega primera večja od ocene drugega primera, je to znak, da ima prvi primer tudi večjo verjetnost. Slabost pri tem je, da točkovnih ocen različnih klasifikatorjev ne moremo medsebojno primerjati, za razliko od napovedanih verjetnosti, ki imajo enotno interpretacijo. Pomembna značilnost krivulj ROC je, da merijo sposobnost klasifikatorjev za izdajanje dobrih točkovnih ocen [5]. Obravnavanim klasifikatorjem tako ni potrebno vračati točnih verjetnosti. Vse kar morajo narediti je, da razločujejo pozitivne primere od negativnih.

Zelo uporabna lastnost krivulj ROC je, da ostanejo nespremenjene pri menjavanju porazdelitve po razredih (angl. *class distribution*). Krivulja ROC je osnovana na TPR in FPR vrednostih. Porazdelitev po razredih je razmerje med številom pozitivnih primerov (levi stolpec na sliki 2.1) in številom negativnih primerov (desni stolpec na sliki 2.1). Ker sta tako TPR kot FPR izračunana iz vrednosti enega stolpca, so krivulje ROC neodvisne od porazdelitve po razredih.

2.2.1 Gradnja krivulje ROC verjetnostnega klasifikatorja

Za gradnjo krivulje ROC verjetnostnega klasifikatorja moramo najprej razvrstiti primere glede na njihove točkovne ocene. Nato izberemo primer, ki ima

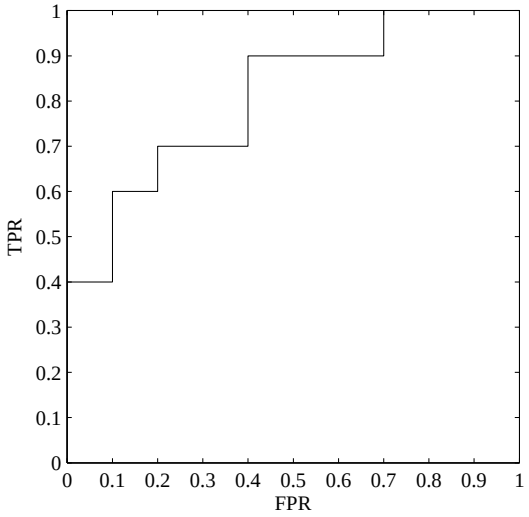
najvišjo točkovno oceno, na grafu ROC pa se postavimo na točko $(0, 0)$. Preverimo, ali je dejanski razred primera pozitiven ali negativen. Če je pozitiven, na grafu naredimo pomik za eno enoto navzgor, če je negativen, pa pomik za eno enoto v desno. Vodoravna oziroma navpična velikost enote sta pri tem obratno sorazmerni s številom negativnih oziroma pozitivnih primerov v podatkovni množici. Ta korak ponavljamo, tako da jemljemo naslednje primere po padajočem vrstnem redu ter se pri tem ustrezno pomikamo po grafu ROC. Postopek se zaključi, ko dosežemo zgornji desni kot v točki $(1, 1)$. Vse točke, pridobljene na ta način, na koncu povežemo in tako tvorimo krivuljo ROC. Postopek lahko razumemo tudi kot postavljanje različnega *praga* na točkovnih ocenah. S spreminjanjem tega praga torej lahko dobimo različne (FPR,TPR) točke, kar se tolmači tudi kot risanje krivulje ROC.

Tabela 2.1: Primer porazdelitve po razredih

primer št.	točkovna ocena klasifikatorja	dejanski razred	primer št.	točkovna ocena klasifikatorja	dejanski razred
1	0.95	p	11	0.57	n
2	0.92	p	12	0.55	p
3	0.90	p	13	0.54	p
4	0.86	p	14	0.52	n
5	0.80	n	15	0.50	n
6	0.73	p	16	0.48	n
7	0.71	p	17	0.47	p
8	0.64	n	18	0.44	n
9	0.61	p	19	0.38	n
10	0.60	n	20	0.35	n

Tabela 2.1 in slika 2.3 prikazujeta ilustrativen primer postopka risanja krivulje ROC. Tabela 2.1 vsebuje 20 primerov s točkovno oceno, ki jo je napovedal klasifikator, in dejanskim razredom. S črko 'p' je označen pozitivni razred, z 'n' pa negativni razred. Podatkovna množica je uravnotežena ter zajema 10 pozitivnih in 10 negativnih primerov. Tako je gradnja vzorčne krivulje ROC bolj enostavna, saj je velikost tako vodoravne kot navpične enote enaka 0.1. V našem primeru določitev pragu med vrednostima točkovnih ocen primerov 4 in 5 izraža stanje s štirimi pravilno uvrščenimi pozitivnimi primeri in nobenim napačno uvrščenim negativnim primerom, kar predstavimo s točko $(0, 0.4)$ na grafu ROC. Ko prag pomaknemo med vrednosti točkovnih ocen primerov 5 in 6, dobimo točko $(0.1, 0.4)$, saj ima primer 5, katerega dejanski razred je nega-

tivni, višjo točkovno oceno od več pozitivnih primerov. Na podoben način se pomikamo navzgor oziroma desno po grafu, dokler ne dosežemo točke $(1, 1)$, ter tako korak za korakom izrišemo krivuljo ROC na sliki 2.3. Predpostavili smo enako ceno napačnih klasifikacij, z njihovo vrednostjo enako 1. V primeru neuravnotežene porazdelitve po razredih ali neenakih cen napačnih klasifikacij se lastnosti prostora ROC spremenijo, tako se npr. pričakovane cene klasifikatorjev na diagonali med seboj razlikujejo.

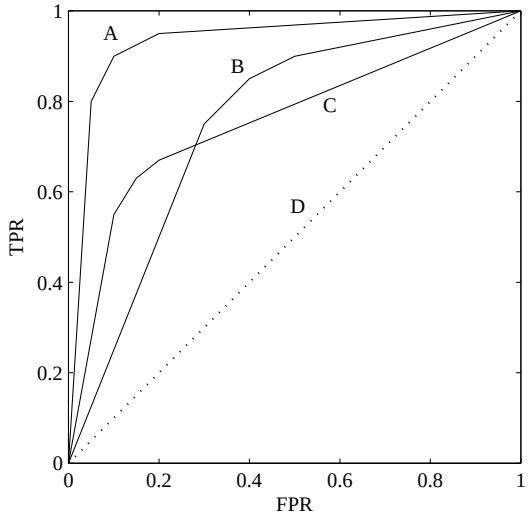


Slika 2.3: Krivulja ROC za vzorčno porazdelitev po razredih

Primer grafa ROC s štirimi različnimi (verjetnostnimi) klasifikatorji, ki so na grafu predstavljeni s štirimi krivuljami ROC, je podan na sliki 2.4. Klasifikator A je veliko boljši od ostalih treh klasifikatorjev. Krivulji ROC klasifikatorjev B in C se križata – vsak od njiju prekaša drugega v določenih okoliščinah uporabe. Klasifikator D ni uporaben, saj je njegova učinkovitost enaka kot pri naključnem ugibanju.

2.2.2 Gradnja krivulje ROC diskretnega klasifikatorja

Glede na to, da je diskretni klasifikator predstavljen z eno samo točko na grafu ROC, lahko zelo približno krivuljo ROC narišemo tako, da to točko povežemo s točkama privzetih klasifikatorjev. Veliko boljša možnost pa je, da analiziramo klasifikatorjev postopek odločanja in ga priredimo tako, da poleg



Slika 2.4: Graf ROC s štirimi različnimi klasifikatorji

napovedi razredov vrača tudi točkovne ocene. Ko imamo točkovne ocene, pa uporabimo isti postopek za gradnjo krivulje ROC kot v primeru verjetnostnega klasifikatorja.

2.3 Mera AUC

Primerjava dveh klasifikatorjev s pomočjo njunih krivulj ROC je lahko netrivialno opravilo, kadar noben od klasifikatorjev ni enoznačno boljši od drugega. V ta namen se v analizi ROC uporablja drugačna mera učinkovitosti klasifikacijskega modela: *AUC* (angl. *Area Under the ROC Curve*, površina ploskve pod krivuljo ROC) [9]. Statistični pomen mere AUC je naslednji: AUC klasifikatorja je enaka verjetnosti, da bo klasifikator naključno izbranemu pozitivnemu primeru dodelil višjo oceno kot naključno izbranemu negativnemu primeru [5]. Ta statistična lastnost se označuje tudi kot verjetnostna oblika mere AUC. Uporaba AUC kot mere učinkovitosti za algoritme strojnega učenja je obravnavana v [1], kjer je narejena tudi primerjava med AUC in merami, ki temeljijo na točnosti. Pokazano je, da ima AUC nekaj prikladnih lastnosti: standardna napaka se zmanjšuje, ko se AUC in število testnih primerov povečujeta; je neodvisna od odločitvenega pragu; je invariantna na apriorne verjetnosti razredov; in nakazuje, do kakšne stopnje sta negativni in pozitivni razred ločena.

Vrednost mere AUC se nahaja na intervalu med 0 in 1. Ker naj bi katerikoli uporaben klasifikacijski model ležal nad diagonalo grafa ROC, vrednost AUC tovrstnih modelov presega 0.5.

Formula mere AUC je:

$$AUC = \frac{\sum_{\text{preko vseh parov}} if(razlika \geq 0; 1; 0)}{\check{s}t.\ vseh\ parov} \tag{2.1}$$

kjer gremo v vsoti preko vseh parov enega pozitivnega in enega negativnega primera. Vrednost spremenljivke *razlika* je enaka razliki med točkovnima ocenama pozitivnega in negativnega primera (v natanko tem vrstnem redu) v posameznem paru. Pogojni stavek ima obliko

$$if(pogoj; a; b)$$

kjer je a vrnjena vrednost ob izpolnjenem pogoju, b pa vrnjena vrednost ob neizpolnjenem pogoju.

Kot zgled izračunajmo vrednost mere AUC za množico primerov iz tabele 2.1. V množici se nahaja 10 pozitivnih in 10 negativnih primerov. Pozitivni primeri imajo številke 1, 2, 3, 4, 6, 7, 9, 12, 13, 17, negativni primeri pa 5, 8, 10, 11, 14, 15, 16, 18, 19, 20. Obstaja $10 \cdot 10 = 100$ parov točkovnih ocen enega pozitivnega in enega negativnega primera. To obenem pomeni, da je ustrezni graf ROC, ki ga vidimo na sliki 2.3, razdeljen na 100 enakih kvadratkov. Vsoto v števcu ulomka v formuli 2.1 ter končno vrednost mere AUC za obravnavano množico primerov dobimo na spodaj opisan način.

1. Razlika med točkovno oceno primera 1 (p) in točkovno oceno kateregakoli negativnega primera je vedno pozitivna, saj je točkovna ocena primera 1 večja od točkovnih ocen vseh negativnih primerov. Delna vsota je enaka 10. Podobno velja tudi za pare, v katerih nastopajo točkovne ocene primerov 2 (p), 3 (p) in 4 (p). Skupna vsota se pri tem poveča na 40. Če graf ROC s slike 2.3 razdelimo na 10 vrstic in 10 stolpcev, lahko rezultat opisanega opazimo na spodnjih štirih vrsticah grafa (tj. vrstice so "polne").
2. Razlika med točkovno oceno primera 6 (p) in točkovnimi ocenami posameznih negativnih primerov je pozitivna za vse negativne primere razen primera 5. Slednji ima namreč večjo točkovno oceno od primera 6. Delna vsota, ki jo prištejemo, je 9, skupna vsota pa je enaka 49. Tu torej "izgubimo" en kvadratek na grafu (peta vrstica od spodaj navzgor). Podobno velja za pare, v katerih nastopa točkovna ocena primera 7 (p). Skupna vsota se poveča na 58.

3. Na enak način nadaljujemo do primera 17 (p), kjer je delna vsota enaka 3. Končna skupna vsota se povzpne na 81. To število delimo še s številom vseh parov, ki je v našem primeru enako 100. Za obravnavano množico primerov torej velja $AUC = 0.81$.

AUC je v sorodu z več dobro znanimi merami. Ima enak pomen kot Wilcoxonova statistika [9] in Mann-Whitneyeva statistika. Poleg tega je v zvezi z Ginijevim indeksom [2].

Mera AUC predstavlja teoretično osnovo tako za obstoječe različice, opisane v poglavju 3, kot tudi za nove različice, razvite v okviru tega dela in opisane v poglavju 4. V nadaljevanju jo zato imenujemo tudi *osnovna (različica) AUC*.

Poglavje 3

Različice AUC na osnovi točkovnih ocen

Osnovna AUC ne upošteva vrednosti točkovnih ocen (aposteriornih verjetnosti pozitivnega razreda), temveč le njihovo razvrstitev. Ker se informacije ne uporabi v celoti, bi končni rezultat lahko bil suboptimalen. Tako lahko na primer pri izbiri klasifikatorjev z visoko vrednostjo AUC pride do prevelikega prileganja podatkom. Po drugi strani je prednost osnovne mere AUC ta, da je neodvisna od kakršnihkoli predpostavk o porazdelitvi. V določenih primerih bi lahko bila koristna kakšna druga, prilagojena rešitev, kjer bi se poleg razvrstitve upoštevala tudi sama vrednost točkovnih ocen.

Za vrednotenje uspešnosti učenja lahko namesto osnovne AUC uporabimo eno izmed mer, ki so iz nje izpeljane in upoštevajo tako vrednosti točkovnih ocen kot razvrstitev. Tovrstne mere v tem delu imenujemo *izpeljane različice* oz. *izpeljanke*. Le-te delimo na že obstoječe in novorazvite (v okviru tega diplomskega dela). Štiri obstoječe izpeljanke so opisane v nadaljevanju tega poglavja. Obstajajo seveda tudi druge možnosti, katerih pa v okviru tega diplomskega dela ne bomo obravnavali. Za mero bi tako na primer lahko izbrali informacijski prispevek, Brierjevo oceno ali pa LogLoss, ki bi vrednosti točkovnih ocen upoštevale. Potrebno pa je vedeti, da vse naštete mere obenem ignorirajo razvrstitev primerov. Nadalje bi bila možna uporaba pasov zaupanja (angl. *confidence bands*) za krivulje ROC [12], vendar pa naj bi njihov vpliv na AUC v primeru neenakomerne porazdelitve verjetnosti bil nejasen [6].

Enako kot v primeru osnovne mere AUC, lahko tudi vrednosti izpeljanih različic izračunamo neposredno brez tvorbe ustreznih različic krivulj ROC. Mero AUC je mogoče posplošiti, kar omogoča, da osnovno mero in njene izpeljanke izrazimo v enotni obliki. Tovrstna posplošitev je bila izvedena v

[16]. Za izračun vrednosti katerekoli od različic AUC za dano množico primerov gremo preko vseh možnih parov enega pozitivnega in enega negativnega primera ter pri tem akumuliramo vrednost *funkcije razlike* (angl. *difference function*), poimenovane tudi *modifikacijska funkcija* (angl. *modifier function*) v [16]. Funkcija razlike obravnava razliko točkovnih ocen dveh primerov v posameznem paru. Končno vsoto nato delimo s številom možnih parov. Posplošeno obliko mere AUC tako lahko tolmačimo kot srednjo vrednost funkcije razlike za množico primerov. Iz posplošitve sledi, da se osnovna AUC in izpeljane različice med seboj razlikujejo le v funkciji razlike, torej načinu obravnave razlike točkovnih ocen. Funkcija razlike osnovne AUC je enotina stopnica.

Izraze "mera", "različica" in "izpeljanka" v nadaljevanju ponekod (kjer je to seveda smiselno) uporabljamo medsebojno zamenljivo. Pri tem so mere, obravnavane v tem in naslednjih poglavjih, izpeljanke osnovne različice mere AUC.

Spodaj so jedrnato predstavljene štiri že obstoječe izpeljanke mere AUC. Formule so pri tem podane v enotni obliki in se po zapisu nekoliko razlikujejo od formul v izvorni literaturi. Obnašanje izpeljank je zatem ponazorjeno z zgledom.

Metode risanja krivulj *probROC* in *scorROC* so bile predstavljene skupaj z ustreznimi različicami AUC, medtem ko pojma krivulj *sondROC* in *softROC* nista bila izrecno omenjena. Vse štiri različice so mehkejše od osnovne AUC, saj je njihov namen glajenje funkcije razlike osnovne AUC. Kot take naj bi intuitivno bile bolj robustne za majhne podatkovne množice. Funkcije razlike izpeljank $probAUC$, $scorAUC$ in $softAUC$ so upodobljene v [16].

3.1 Izpeljanka probAUC

V [6] je predstavljena prva izpeljanka, imenovana $probAUC$ (angl. *probabilistic AUC*, verjetnostna AUC). Njena funkcija razlike temelji na verjetnosti in enakomerni porazdelitvi. Pomen mere $probAUC$ lahko tolmačimo kot srednjo vrednost (i) povprečne napovedane verjetnosti pripadnosti pozitivnemu razredu za pozitivne primere in (ii) povprečne napovedane verjetnosti pripadnosti negativnemu razredu za negativne primere. Čeprav mera $probAUC$ v večini slučajev podcenjuje vrednost osnovne AUC, se lahko zgodi tudi nasprotno (kot lahko vidimo pri množicah 11, 12 in 14 v tabeli 3.1). Ker učinkovitosti ni več mogoče upodobiti z osnovno krivuljo ROC, je v istem viru podan tudi pristop za risanje krivulj probROC (s površino ploskve pod krivuljo enako $probAUC$). Avtorjem se je zdelo koristno ohraniti pojem krivulj ROC, saj le-

te ponujajo način izbire klasifikacijskega pragu. Tako kot krivulje ROC se tudi krivulje probROC gradijo v prostoru ROC. Pri gradnji krivulj probROC se za verjetnosti predpostavlja, da nimajo točno določene vrednosti, temveč ležijo na nekem intervalu. Krivulje probROC so običajno bolj gladkih oblik, glavne razlike so še posebej opazne takrat, ko je osnovna krivulja ROC nezanesljiva (malo primerov, majhne razlike med točkovnimi ocenami). Za večje množice primerov se krivulja probROC obnaša podobno kot osnovna krivulja ROC. Formula izpeljanke probAUC je:

$$probAUC = \frac{\frac{\sum_{\text{preko vseh poz. primerov}} \textit{točk. ocena}}{\textit{št. poz. primerov}} + \frac{\sum_{\text{preko vseh neg. primerov}} \textit{(1-točk. ocena)}}{\textit{št. neg. primerov}}}{2} \tag{3.1}$$

3.2 Izpeljanka scorAUC

Druga različica je *scorAUC* (angl. *scored AUC*, točkovana AUC) [17, 18]. Njena vrednost je enaka površini ploskve pod krivuljo *scorROC*. Krivulja scorROC ponazarja, kako hitro se AUC poslabša, če pozitivne točkovne ocene znižamo, oziroma z drugimi besedami, kako občutljiv je klasifikator na pomik vrednosti točkovnih ocen. *scorAUC* akumulira to informacijo v obliki številske mere. Krivuljo scorROC se tvori v prostoru, ki je popolnoma drugačen od prostora ROC. V tem primeru os x označuje vrednost parametra τ, ki predstavlja stopnjo znižanja pozitivnih točkovnih ocen, os y pa vrednost AUC tako spremenjene množice. Funkcija razlike, uporabljena pri meri *scorAUC*, je enotina stopnica, utežena z vrednostjo razlike točkovnih ocen. Različica *scorAUC* vedno podcenjuje vrednost osnovne mere AUC. Njune vrednosti so enake le takrat, ko klasifikator vrača popolne točkovne ocene za množico primerov, torej ko napove vrednost 1 za vsak pozitivni primer in vrednost 0 za vsak negativni primer (kar je izraženo tudi v tabeli 3.1). Zaradi dejstva, da *scorAUC* uporablja tako točkovne ocene kot razvrstitev, avtorji navajajo, da mera lahko služi kot statistika za testiranje različnosti dveh vzorcev (podobno kot Wilcoxon-Mann-Whitneyeva statistika). Formula izpeljanke scorAUC je:

$$scorAUC = \frac{\sum_{\text{preko vseh parov}} (if(razlika \geq 0; 1; 0) \cdot razlika)}{\textit{št. vseh parov}} \tag{3.2}$$

kjer gremo v vsoti preko vseh parov enega pozitivnega in enega negativnega primera. Pomen spremenljivke *razlika* in stavka *if* je podan v opisu formule 2.1.

3.3 Izpeljanka sondAUC

Tretja različica z imenom *sondAUC* (angl. *softened AUC*, zmehčana AUC) je bila predlagana v [10]. Mera *sondAUC* je pravzaprav posplošena različica mere *scorAUC*. Funkciji razlike sta enaki z naslednjo izjemo: v primeru *sondAUC* je uporabljeno potenciranje, kjer razlika med točkovnimi ocenami predstavlja osnovo, eksponent pa je parameter q. Namen eksponenta q je uravnavanje občutljivosti in robustnosti na spreminjanje razvrstitve, kar uporabniku omogoča bolj primerno izbiro klasifikatorjev. Z večanjem vrednosti q mera *sondAUC* običajno postane bolj občutljiva in manj robustna na spremembe v razvrstitvi. V kolikor vrednost parametra q postavimo na 0, mera *sondAUC* preide v osnovno mero AUC. Formula izpeljanke sondAUC je:

$$sondAUC = \frac{\sum_{\text{preko vseh parov}}(if(razlika \geq 0; 1; 0) \cdot razlika^q)}{št.\ vseh\ parov} \tag{3.3}$$

3.4 Izpeljanka softAUC

Zadnja različica se imenuje *softAUC* (angl. *soft AUC*, mehka AUC) [3]. Poleg upoštevanja vrednosti točkovnih ocen sta bili zaželeni lastnosti nove mere tudi zveznost in odvedljivost, kar za mero *softAUC* tudi velja. Funkcija razlike izpeljanke *softAUC* je sigmoidna funkcija (točneje, logistična funkcija) s parametrom β. Sigmoida ima vlogo zglajenega približka enotine stopnice, v katero skonvergira, ko $\beta \to \infty$. V takšnem primeru torej *softAUC* preide v osnovno AUC. Formula izpeljanke softAUC je:

$$softAUC = \frac{\sum_{\text{preko vseh parov}} \frac{1}{1+e^{((-1)\cdot\beta\cdot razlika)}}}{št.\ vseh\ parov} \tag{3.4}$$

kjer gremo v vsoti preko vseh parov enega pozitivnega in enega negativnega primera. Pomen spremenljivke *razlika* je podan v opisu formule 2.1.

3.5 Opredelitev novih pojmov

Sledi opredelitev pojmov, ki nastopajo tako v zgledu kot v naslednjih poglavjih:

- *razpon*: razlika med najvišjo in najnižjo točkovno oceno v množici. Velikost razpona točkovnih ocen leži na intervalu [0.00, 1.00],
- *rob*: meja oz. pas med pozitivnimi in negativnimi primeri,

- *absolutna širina roba*: razlika med točkovnima ocenama najmanjšega pozitivnega primera in največjega negativnega primera v množici. Če so primeri v množici pravilno razvrščeni, je absolutna širina roba pozitivna (klasifikator razločuje pozitivne in negativne primere), v nasprotnem primeru pa je negativna (klasifikator ne razločuje pozitivnih in negativnih primerov). Če imata najmanjši pozitivni primer in največji negativni primer enaki točkovni oceni, je absolutna širina roba enaka nič (tudi v tem primeru klasifikator ne razločuje pozitivnih in negativnih primerov). Velikost absolutne širine roba leži na intervalu [-1.00, 1.00],
- *relativna širina roba*: relativna širina roba = $\frac{\text{absolutna širina roba}}{\text{razpon}}$.

3.6 Izračun obstoječih izpeljank na zgledu in opis njihovega obnašanja

Na tem mestu želimo s pomočjo tabele 3.1 ilustrirati obnašanje obravnavanih različic AUC.

Recimo, da je podanih šest primerov, po trije pozitivni in trije negativni. Za vsak primer je torej poznan njegov dejanski razred. Te primeri imajo določene lastnosti oz. atribute. Le-te niso podani, saj za ilustracijo niso pomembni, vsekakor pa morajo obstajati. Vsak primer je predstavljen z verjetnostjo pripadnosti pozitivnemu razredu, izraženo v obliki realnega števila, kot jo je napovedal klasifikator (tj. s točkovno oceno), ter črko, ki sporoča dejanski razred (pri tem 'p' pomeni pozitivni razred, 'n' pa negativni razred). Vrednosti točkovnih ocen so sicer v danem zgledu izmišljene.

V tabeli lahko vidimo petnajst množic točkovnih ocen, v vsaki od njih pa nastopa istih šest, zgoraj omenjenih primerov. Glede na to, da gre za iste primere, bom namesto izraza *množica primerov* uporabljal izraz *množica točkovnih ocen*. Vsaka množica prikazuje obnašanje nekega drugega klasifikatorja. Vsak klasifikator pri tem za podane primere vrne točkovne ocene. Primeri so v vsaki množici razvrščeni glede na dodeljene točkovne ocene, od primera z največjo točkovno oceno do tistega z najmanjšo. Drugi stolpec tabele tako podaja napovedane točkovne ocene in dejanske razrede primerov. V nadaljnjih stolpcih pa lahko za vsako množico točkovnih ocen vidimo, kako na njeni podlagi osnovna mera AUC in že obstoječe izpeljanke ovrednotijo delo klasifikatorjev (tj. kakovost klasifikacije). Vrednosti vseh mer so izračunane na podlagi formul 2.1, 3.1, 3.2, 3.3 in 3.4. Vrednosti ocen posameznih različic zaradi različnih formul seveda medsebojno niso primerljive (primerjamo lahko le vrstni red).

Tabela 3.1: Množice točkovnih ocen (ki jih klasifikatorji napovejo za šest primerov) z naknadno izračunanimi vrednostmi osnovne mere AUC in njenih izpeljank za vsako množico

#	množica točkovnih ocen	AUC	probAUC	scorAUC	sondAUC	softAUC	q	β
1a	1.00p 1.00p 1.00p 0.00n 0.00n 0.00n	1.000	1.000	1.000	1.000	1.000	1/7	20.0
1b	1.00p 1.00p 1.00p 0.00n 0.00n 0.00n	1.000	1.000	1.000	1.000	0.999	1/7	7.0
1c	1.00p 1.00p 1.00p 0.00n 0.00n 0.00n	1.000	1.000	1.000	1.000	0.881	1/7	2.0
1d	1.00p 1.00p 1.00p 0.00n 0.00n 0.00n	1.000	1.000	1.000	1.000	0.731	1/7	1.0
1e	1.00p 1.00p 1.00p 0.00n 0.00n 0.00n	1.000	1.000	1.000	1.000	0.599	1/7	0.4
2	0.97p 0.95p 0.92p 0.09n 0.06n 0.05n	1.000	0.940	0.880	0.982	0.998	1/7	7.0
3	0.94p 0.94p 0.94p 0.58n 0.58n 0.58n	1.000	0.680	0.360	0.864	0.926	1/7	7.0
4	0.94p 0.88p 0.82p 0.61n 0.59n 0.55n	1.000	0.648	0.297	0.839	0.883	1/7	7.0
5	0.90p 0.70p 0.60p 0.40n 0.10n 0.00n	1.000	0.783	0.567	0.912	0.955	1/7	7.0
6	1.00p 1.00p 1.00p 0.90n 0.90n 0.90n	1.000	0.550	0.100	0.720	0.668	1/7	7.0
7	0.60p 0.57p 0.56p 0.54n 0.52n 0.51n	1.000	0.527	0.053	0.651	0.592	1/7	7.0
8	0.95p 0.83p 0.77n 0.75p 0.69n 0.40n	0.889	0.612	0.226	0.707	0.766	1/7	7.0
9a	0.61p 0.61p 0.61p 0.60n 0.60n 0.60n	1.000	0.505	0.010	0.215	0.517	1/3	7.0
9b	0.61p 0.61p 0.61p 0.60n 0.60n 0.60n	1.000	0.505	0.010	0.398	0.517	1/5	7.0
9c	0.61p 0.61p 0.61p 0.60n 0.60n 0.60n	1.000	0.505	0.010	0.518	0.517	1/7	7.0
9d	0.61p 0.61p 0.61p 0.60n 0.60n 0.60n	1.000	0.505	0.010	0.736	0.517	1/15	7.0
9e	0.61p 0.61p 0.61p 0.60n 0.60n 0.60n	1.000	0.505	0.010	0.995	0.517	1/1001	7.0
10	1.00p 0.80n 0.60p 0.25n 0.20p 0.00n	0.667	0.625	0.344	0.593	0.681	1/7	7.0
11	1.00p 0.90n 0.65n 0.56p 0.43p 0.00n	0.556	0.573	0.271	0.487	0.574	1/7	7.0
12	0.61n 0.61n 0.61n 0.60p 0.60p 0.60p	0.000	0.495	0.000	0.000	0.483	1/7	7.0
13	1.00p 1.00p 1.00p 1.00n 1.00n 1.00n	0.500	0.500	0.000	0.000	0.500	1/7	7.0
14	0.90n 0.77p 0.65n 0.56p 0.43p 0.22n	0.444	0.498	0.136	0.368	0.482	1/7	7.0
15	1.00n 1.00n 1.00n 0.00p 0.00p 0.00p	0.000	0.000	0.000	0.000	0.001	1/7	7.0

Množici 1 in 9 sta uporabljeni večkrat (in pri tem označeni s črkami od a do e) za različne vrednosti parametrov q in β, ki sta potrebna pri izračunu različic *sondAUC* oziroma *softAUC*.

Množica 1a-1e torej razkriva učinek spreminjanja parametra β. Ko gre $\beta \to \infty$, mera *softAUC* skonvergira v osnovno AUC. V nasprotnem primeru, ko gre $\beta \to 0$, pa se vrednost mere *softAUC* približuje vrednosti 0.5. Za preostale množice točkovnih ocen smo izbrali fiksno vrednost $\beta = 7$, saj se pri tej vrednosti parametra mera *softAUC* obnaša podobno kot osnovna AUC, vseeno pa zadosti drugače, da osmisli svoj obstoj.

Množica 9a-9e pa prikazuje vpliv parametra q na obnašanje mere *sondAUC*. Ko je $q = 1$, je mera *sondAUC* istovetna z mero *scorAUC*. Ko gre $q \to 0$, pa postaja *sondAUC* vse bolj podobna osnovni meri AUC. Pri izbiri vrednosti parametra q kot racionalnega števila med 0 in 1 je potrebna previdnost, saj se pri sodih imenovalcih ulomka pojavi problem z izračunom korena negativnega števila (kar se zgodi vedno, ko je kateri izmed primerov nepravilno razvrščen). Za vrednost parametra q se, na primer, lahko izbere $\frac{1}{3}$ ali $\frac{1}{5}$, ne pa tudi $\frac{1}{2}$. Za ostale množice točkovnih ocen smo izbrali fiksno vrednost $q = \frac{1}{7}$, saj po našem

mnenju ponuja primerno ravnovesje med robustnostjo in občutljivostjo.

Množice točkovnih ocen so urejene padajoče po kakovosti klasifikatorjevih napovedi od intuitivno najboljše do intuitivno najslabše. Pri tem množica 1a-1e predstavlja popolno stanje, kjer klasifikator vrne točkovno oceno 1.00 za vsak pozitivni primer in točkovno oceno 0.00 za vsak negativni primer. V množicah 2-7 so vsi primeri še vedno popolnoma pravilno razvrščeni, čeprav se je rob med pozitivnimi in negativnimi primeri zožal. Množica 13 predstavlja stanje, ko opazovani klasifikator ni uporaben. Le-ta vrne točkovno oceno 1 za vsak primer, ki ga vidi, in očitno ne razlikuje pozitivnih primerov od negativnih. Tudi klasifikator v množici 14 ne loči med pozitivnimi in negativnimi primeri ter daje vtis, da so njegove točkovne ocene naključne. Množica 15 je najslabša možna, vendar pa lahko popolno množico, tj. množico 1, dobimo na trivialen način, tako da obrnemo klasifikatorjeve odločitve (tj. tolmačimo 0 kot znak pozitivnega primera in 1 kot znak negativnega).

Med drugim je iz tabele tudi razvidno, da osnovne mere AUC ne moremo uporabiti za medsebojno primerjanje klasifikatorjev, ki vse primere pravilno razvrstijo. AUC namreč klasifikatorju v tem primeru vedno dodeli najvišjo možno oceno (tj. 1), ne glede na velikost absolutne širine roba, razpona, itd. Točkovne ocene takšnih klasifikatorjev predstavljajo množice 1, 2, 3, 4, 5, 6, 7 in 9. Enako pravzaprav velja za vsako medsebojno primerjanje klasifikatorjev, ki na neki množici primerov naredijo isto število napak, saj AUC takšnim klasifikatorjem vedno da identično oceno (kljub temu, da se le-ti med seboj morda razlikujejo glede na velikost izdanih točkovnih ocen). Ker imamo pri enaki kakovosti razvrščanja v splošnem raje klasifikator z npr. večjim razponom in večjo absolutno širino roba, opisana lastnost očitno predstavlja pomanjkljivost osnovne mere AUC.

Najbolj očitna opažanja so predstavljena spodaj:

- Ob pogledu na pravilno razvrščeni množici 4 in 5 lahko opazimo, da ima slednja večji razpon. Po drugi strani je absolutna širina roba v množici 4 rahlo večja ($0.82 - 0.61 = 0.21$ v primerjavi z $0.60 - 0.40 = 0.20$), zato lahko rečemo, da ta klasifikator nekoliko bolje razloči pozitivne primere od negativnih. Kljub temu pa so pri vrednotenju vse izpeljanke AUC množici 5 bolj naklonjene kot množici 4. Tukaj ima torej *razpon večji vpliv od absolutne širine roba*. Tudi za pravilno razvrščeno množico 3 so vrednosti vseh izpeljank AUC precej manjše kot za množico 5, čeprav se absolutna širina roba poveča na omembe vrednih 0.36. Zdi se, da se med vsemi različicami na razpon najbolj zanaša mera $scorAUC$.

- Pri pravilno razvrščeni množici 6 vse izpeljane različice AUC kot nezaželena

dojemajo sorazmerno majhno absolutno širino roba in/ali sorazmerno ozek razpon (oba velikosti 0.10). *Majhna absolutna širina roba in/ali ozek razpon torej negativno vplivata na višino ocen*, s katerimi izpeljanke AUC ocenijo množico. Klasifikatorjeve napovedi so pri tem popolnoma konsistentne, vsi pozitivni primeri namreč dobijo točkovno oceno 1.00, vsi negativni primeri pa točkovno oceno 0.90. Klasifikatorjeva sposobnost razvrščanja je torej tudi za dano množico najboljša možna, pomanjkljivost pa je sama velikost točkovnih ocen, katere posledica sta majhen razpon ter majhna absolutna širina roba. Ker za točen klasifikator ni potrebno, da je le-ta tudi dobro kalibriran (da torej vrača točne verjetnosti), poleg tega pa obstajajo metode za kalibriranje klasifikatorjev, lahko rečemo, da osnovna AUC odraža kakovost klasifikatorja te množice bolj verodostojno od izpeljanih različic.

- Zgoraj opisane težave postanejo bolj resne, ko med seboj primerjamo množice, ki niso vse pravilno razvrščene. Pri vrednotenju je namreč lahko *vpliv majhne absolutne širine roba močnejši celo od vpliva pravilne razvrstitve*. Kljub temu, da je nepravilno razvrščena množica 14 bistveno slabše kakovosti od pravilno razvščenih množic 6 in 7, opazimo da mera $scorAUC$ bolje oceni prav množico 14. Podobno tudi izpeljanki $probAUC$ in $scorAUC$ množico 11 ocenita bolje od množice 6.

- Pri primerjanju množic, ki niso vse pravilno razvrščene, se lahko zgodi še ena neprijetnost. Množica 8 je nepravilno razvrščena, klasifikator namreč napačno oceni dva primera ter pozitivnemu primeru dodeli malenkost nižjo točkovno oceno kot negativnemu. Ne glede na to, da je napaka sicer majhna, bi pričakovali, da bodo izpeljanke AUC bolje kot množico 8 ocenile pravilno razvrščeno množico 7. Vendar pa to ne drži in množica 8, ki ima večji razpon, je ovrednotena kot bolj kakovostna s strani vseh izpeljanih AUC različic, med katerimi najbolj izstopa $scorAUC$. Tudi *razpon ima torej lahko večji vpliv od pravilne razvrstitve*. Pojav prav tako opazimo ob primerjavi ocen mere $softAUC$ za nepravilno razvrščeno množico 10 in pravilno razvrščeno množico 6.

- Pravilno razvrščena množica 9 in nepravilno razvrščena množica 12 razkrivata glavno slabost mere AUC, ki jo izpeljane različice poskušajo odpraviti. To je *nezanesljivost AUC za (majhne) množice, kjer so razlike med napovedanimi točkovnimi ocenami zanemarljive*. Kot vidimo, se lahko drobna sprememba velikosti točkovnih ocen odrazi v znatni (v našem primeru celo največji možni) spremembi vrednosti mere AUC (saj

AUC popolnoma zaupa razvrstitvi). Po drugi strani izpeljane različice AUC vrnejo precej konsistentne vrednosti za obe, zelo podobni množici. To je tipičen primer situacije, ko je morda bolj primerno zaupati rezultatom vrednotenja, ki jih podajo izpeljane različice AUC.

3.7 Podrobnejša analiza

Veliko bolj obsežna analiza treh izpeljank AUC, *probAUC*, *scorAUC* in *softAUC*, je bila opravljena v [16], kjer navajajo, da naj bi njihova učinkovitost bila dvomljiva. Nobena od izpeljanih različic naj namreč ne bi prekašala osnovne AUC, vsaj ne v primeru njihove uporabe za vrednotenje in izbiro klasifikatorjev. Izpeljane različice naj bi vse bile pristranske z varianco v eni od obeh smeri, kar njihove teoretične temelje postavlja pod vprašaj. Kljub temu pa sta različici *probAUC* in *softAUC*, ob ustrezno izbranih vrednostih parametrov, spoznani za natančna približka osnovne mere AUC.

Poglavje 4

Predlagane različice mere AUC

To poglavje vsebuje opis nove izpeljanke AUC, ki jo predlagamo v okviru diplomskega dela, in analizira njene lastnosti ter njene (domnevne) prednosti pred že obstoječimi izpeljankami.

Podlaga za razvoj nove mere so ugotovitve iz poglavja 3, in sicer (i) nezanesljivost osnovne AUC za množice, pri katerih so razlike med točkovnimi ocenami zanemarljive, (ii) neinformativnost osnovne AUC pri primerjanju množic, katere vsebujejo enako število napak, ter (iii) neprimeren medsebojni vpliv dejavnikov, kot so pravilnost razvrstitve, velikost razpona in velikost absolutne širine roba pri obstoječih izpeljanih različicah.

Končni cilj je torej pridobiti mero, pri katerih bi vpliv omenjenih dejavnikov bilo mogoče prilagajati glede na podane množice primerov. Ta mera naj bi vračala tudi bolj verodostojne ocene za množice z majhnimi razlikami med točkovnimi ocenami in bila uporabna pri medsebojnem primerjanju poljubnih množic.

Predpostavljamo, da je kljub temu, da upoštevamo tudi velikost točkovnih ocen, najpomembnejši kriterij pravilna razvrstitev primerov, tj. da klasifikator (po možnosti) vsem pozitivnim primerom dodeli višjo točkovno oceno kot kateremukoli negativnemu primeru. Množice lahko torej glede na razvrstitev opišemo kot pravilno oziroma nepravino razvrščene.

Nove (vmesne) izpeljanke označujemo kot mm1AUC, mm2AUC, mm3AUC itd. Pri tem so nekatere številke izpuščene, saj smo pri razvoju občasno zašli v slepo ulico. V tem delu podajamo le tiste različice, ki se nahajajo neposredno na poti razvoja končne različice mm7AUC. Take izpeljanke so štiri in jih opisujemo v nadaljevanju. Pri tem je izpeljanka mm4AUC nadgradnja izpeljanke mm1AUC, mm6AUC nadgradnja izpeljanke mm4AUC, mm7AUC pa nadgradnja izpeljanke mm6AUC, tj. $mm1AUC \rightarrow mm4AUC \rightarrow mm6AUC \rightarrow$

mm7AUC.

Najprej smo si podrobneje ogledali lastnosti množic točkovnih ocen, podanih v tabeli 3.1 in jih povzemamo v tabeli 4.1. Pojem *absolutna velikost napake* je razložen v nadaljevanju, pri opisu izpeljanke mm7AUC.

Tabela 4.1: Lastnosti množic točkovnih ocen iz tabele 3.1

#	razpon ocen	absolutna širina roba	relativna širina roba	mediana ocen	absolutna velikost napake
1b	1.00	1.00	1.00	0.50	0
2	0.92	0.83	0.90	0.51	0
3	0.36	0.36	1.00	0.76	0
4	0.39	0.21	0.54	0.72	0
5	0.90	0.20	0.22	0.50	0
6	0.10	0.10	1.00	0.95	0
7	0.09	0.02	0.22	0.55	0
8	0.55	-0.02	-0.04	0.76	1
9c	0.01	0.01	1.00	0.61	0
10	1.00	-0.60	-0.60	0.43	3
11	1.00	-0.47	-0.47	0.61	4
12	0.01	-0.01	-1.00	0.61	9
13	0.00	0.00	–	1.00	9
14	0.68	-0.47	-0.69	0.61	5
15	1.00	-1.00	-1.00	0.50	9

Pred razvojem novih različic smo naredili nekaj dodatnih predpostavk, ki so skladne z intuitivnim dojemanjem kakovosti množic primerov. Ob odsotnosti spodnjih predpostavk (točneje, prvih dveh) namreč težko rečemo, da je klasifikator množice 1 (v tabeli 3.1) v kateremkoli pogledu boljši od klasifikatorjev množic 7 ali 9.

- Predpostavljamo, da je zaželen čim večji razpon. V popolnem primeru naj bi imel vrednost 1.00.
- Predpostavljamo, da je zaželena čim večja absolutna širina roba. V popolnem primeru naj bi imela vrednost 1.00.
- Predpostavljamo, da je zaželena čim manjša absolutna velikost napake. V popolnem primeru naj bi bila enaka 0.

Za mediano točkovnih ocen smo sicer pozneje ugotovili, da njena vrednost, vsaj na prvi pogled, ne vpliva niti na osnovno AUC niti na nobeno izmed izpeljank. Na primer, če vzamemo naslednji množici

množica 1: 0.60p 0.57p 0.56p 0.54n 0.52n 0.51n
množica 2: 0.90p 0.87p 0.86p 0.84n 0.82n 0.81n,

namreč ugotovimo, da vsaka izmed obravnavanih različic množici 1 dodeli enako oceno kot množici 2. Sklepamo lahko, da nobena od mer pravzaprav ni

odvisna od same velikosti točkovnih ocen, temveč vse mere temeljijo na razliki med ocenami. Mediane točkovnih ocen v nadaljevanju tako ne upoštevamo.

Po podrobnem pregledu vseh različic AUC smo se odločili, da bomo za začetek izhajali iz scorAUC in iz nje poskusili izpeljati izboljšano mero. Pri scorAUC je bilo, po našem mnenju, namreč precej možnosti za nadgradnjo.

4.1 Izpeljanka mm1AUC

Na ta način je najprej nastala izpeljanka *mm1AUC*. Funkcijo stopnice, utežene z razliko vrednosti, smo dodatno utežili z razponom ocen. Na ta način smo želeli *odpraviti kaznovanje klasifikatorjev, pri katerih so razlike med ocenami pozitivnih in negativnih primerov majhne, pod pogojem, da je majhen tudi sam razpon ocen.* Formula te izpeljanke je

$$mm1AUC = \frac{\sum_{\text{preko vseh parov}}(if(razlika \geq 0; 1; 0) \cdot (\frac{razlika}{razpon}))}{\check{s}t.\ vseh\ parov} \tag{4.1}$$

kjer gremo v vsoti preko vseh parov enega pozitivnega in enega negativnega primera. Pomen spremenljivke *razlika* in stavka *if* je podan v opisu formule 2.1. mm1AUC se od scorAUC razlikuje le po prisotnosti spremenljivke *razpon*, katere pomen je skladen z opisom v podpoglavju 3.5. Cilj je dosežen, saj nova mera klasifikatorjev z majhnim razponom ocen ne kaznuje preveč. V kolikor je *relativna širina roba enaka* 1, mera takega klasifikatorja niti malo ne kaznuje in mu dodeli najvišjo možno oceno (če je seveda razvrstitev brezhibna). Na ta način klasifikatorji v množicah 3, 6 in 9c dobijo oceno 1.00 (za razliko od ocen 0.36, 0.10 in 0.01 pri scorAUC).

4.2 Izpeljanka mm4AUC

Izpeljanka *mm4AUC* predstavlja nadgradnjo različice mm1AUC. Slednja je dopolnjena tako, da prispevek nobenega para, v primeru, da je razlika za ta par pozitivna, ne more biti manjši od 0.50. Naključni klasifikator pri izpeljanki mm1AUC sicer ne dobi ocene 0.50 (kot v primeru mere AUC) in ni možno z obratno razvrstitvijo klasifikatorjevih točkovnih ocen dobiti rezultata $1 - mm1AUC$, kjer je mm1AUC dobljen pri prvotni razvrstitvi. So pa na ta način *ocene boljših in slabših klasifikatorjev, ki jih dobimo z mero mm4AUC, nekoliko bolj ločene med seboj*, saj ima vsaka pravilna razvrstitev prispevek vsaj 0.50, vsaka napačna pa točno 0.00. Pri mm1AUC ima namreč pravilna razvrstitev

pri poljubno majhni razliki med pozitivnim in negativnim primerom v danem paru prispevek poljubno blizu 0.00. Namesto izbrane vrednosti 0.50 se lahko na tem mestu pozneje uporabi tudi nastavljiv parameter. Formula te izpeljanke je torej

$$mm4AUC = \frac{\sum_{\text{preko vseh parov}}(if(a \geq 0.5; a; (if(a = 0; 0; 0.5))))}{\check{s}t.\ vseh\ parov} \tag{4.2}$$

kjer je

$$a = if(razlika \geq 0; 1; 0) \cdot (razlika/razpon) \tag{4.3}$$

Pomen spremenljivke *razlika* in stavka *if* je podan v opisu formule 2.1, pomen spremenljivke *razpon* pa v podpoglavju 3.5.

4.3 Izpeljanka mm6AUC

Različica *mm6AUC* je bila dobljena z dopolnitvijo mere mm4AUC, in sicer tako, da je končni rezultat utežen s funkcijo absolutne širine roba. Namen je *kaznovanje ozkega absolutnega roba*. Za to funkcijo smo izbrali potenciranje (natančneje korenjenje), kjer je stopnja npr. 1/16 (torej šestnajsti koren). Stopnjo potence pri absolutni širini roba smatramo kot izrecni parameter (označimo ga kot n), ki odraža velikost kazni za ozek absolutni rob. Tovrstno uteževanje se uporabi le, če je absolutni rob pozitiven. Če je absolutni rob negativen, je vsaj en primer nepravilno razvrščen in kaznovanje ozkega roba ne bi imelo smisla (kakršnekoli širine že je takšen rob, je slab). Ker se pojavi problem v formuli zaradi šestnajstega korena negativnega števila, dodamo pravilo, da je $(\text{absolutna širina roba})^n = 1$, če je absolutna širina roba negativna. mm6AUC torej kaznuje le pravilno razvrščene množice (v kolikor je to potrebno).

Poleg tega vrednost mere mm4AUC predhodno prav tako potenciramo (stopnjo smatramo kot izrecni parameter m). S slednjim lahko povečamo oz. zmanjšamo vpliv mere mm4AUC. Končna formula je

$$mm6AUC = mm4AUC^m \cdot if(abs.\ šir.\ roba > 0; (abs.\ šir.\ roba)^n; 1) \tag{4.4}$$

kjer je pomen stavka *if* podan v opisu formule 2.1.

Na tem mestu povzemamo najbolj očitne značilnosti v obnašanju mere mm6AUC. Izračunane vrednosti za vseh 15 že obravnavanih množic se nahajajo v tabeli 4.2.

- mm6AUC, za razliko od AUC, oceno 1.00 dodeli le najboljšemu možnemu klasifikatorju (v množici 1b).

- mm6AUC, podobno kot obstoječe izpeljane različice, množico 2 oceni le malenkost slabše od najboljše možne.
- mm6AUC, za razliko od probAUC in scorAUC, množici 3 in 6 oceni kot zelo dobri. Ti dve množici si, zaradi svoje pravilne razvrstitve, maksimalne relativne širine roba in konsistentnosti, visoko oceno tudi zaslužita.
- *mm6AUC kot edina izmed izpeljanih različic množico 9c (s pravilno razvrstitvijo vendar majhno absolutno širino roba) oceni kot boljšo od množic 8 in 10. Enako sicer naredi tudi osnovna AUC, vendar pa je njena ocena manj informativna od ocene mm6AUC. Osnovna AUC namreč ne upošteva širine roba, zato sta zanjo klasifikatorja v 1b in 9c popolnoma enake kakovosti.*
- mm6AUC, prav tako kot skoraj edina od izpeljanih različic (izjema je sondAUC), pravilno razvrščeno množico 9c oceni kot boljšo od množice 11 (ki ima skoraj naključne ocene).
- mm6AUC, podobno kot AUC, scorAUC in sondAUC (za razliko od probAUC in softAUC), vse napake smatra kot slabe (ne glede na njihovo velikost). Tako množico 12 oceni z 0.00.

4.4 Izpeljanka mm7AUC

Izpeljanko mm6AUC poskušamo nadalje nadgraditi, da bi bolj primerno ocenila tudi pravilno razvrščeno množico 7. Doseči želimo, da množica 7 dobi višjo oceno, saj je le-ta nižja od ocene množice 8, ki ni popolnoma pravilno razvrščena. Lahko pa gremo za dosego istega cilja tudi v obratno smer in množicam z nepravilno razvrstitvijo (med njimi tudi množici 8) oceno znižamo. *Kot utež lahko namreč v tem primeru uporabimo število napak oz. velikost napake.*

Napako opredelimo takole: napaka je vsaka nepozitivna (tj. negativna ali ničelna) razlika med točkovnima ocenama pozitivnega in negativnega primera (v natanko tem vrstnem redu). Število napak je torej število nepozitivnih razlik. To število smo poimenovali tudi *absolutna velikost napake*. Če se število nepozitivnih razlik deli z največjim možnim številom nepozitivnih razlik (tj. največjim možnim številom napak), lahko takšno količino poimenujemo tudi *relativna velikost napake*. Največje možno število nepozitivnih razlik je enako številu vseh možnih parov enega pozitivnega in enega negativnega primera.

Dobimo torej formulo:

$$rel.\ vel.\ napake = \frac{število\ nepozitivnih\ razlik}{št.\ dej.\ poz.\ primerov \cdot št.\ dej.\ neg.\ primerov} \tag{4.5}$$

Če mero mm6AUC utežimo z relativno velikostjo napake, na ta način uvedemo *linearno kaznovanje klasifikatorja zaradi storjenih napak*. S tem se dodatno pripomore k boljšemu ločevanju dobrih in slabih klasifikatorjev (za kar si prizadeva že mera mm4AUC).

Formula za relativno velikost napake pa je tesno povezana s formulo za osnovno mero AUC. Velja:

$$rel.\ vel.\ napake = 1 - AUC \tag{4.6}$$

Različico *mm7AUC* dobimo tako, da mm6AUC pomnožimo s faktorjem (1 − *relativna velikost napake*) (da bi v primeru, ko ni napake, dobili vrednost 1). To pa je ravno AUC. Formula mere mm7AUC je torej

$$mm7AUC = mm6AUC \cdot AUC \tag{4.7}$$

4.5 Izračun vrednosti novorazvitih izpeljank

Vrednosti novorazvitih mer za 15 množic iz tabele 3.1 se nahajajo v tabeli 4.2. Tako pri meri mm6AUC kot mm7AUC sta uporabljeni vrednosti parametrov $m = 9/10$ in $n = 1/16$. Tudi pri novorazvitih izpeljankah vrednosti ocen posameznih izpeljank zaradi različnih formul ne moremo neposredno primerjati med seboj. Primerjamo lahko le vrstni red.

Tabela 4.2: Rezultati meritev kakovosti na podlagi novih mer za množice iz tabele 3.1

#	mm1AUC	mm4AUC	mm6AUC	mm7AUC
1b	1.000	1.000	1.000	1.000
2	0.957	0.957	0.950	0.950
3	1.000	1.000	0.938	0.938
4	0.761	0.761	0.709	0.709
5	0.630	0.679	0.638	0.638
6	1.000	1.000	0.866	0.866
7	0.593	0.648	0.530	0.530
8	0.410	0.546	0.581	0.516
9c	1.000	1.000	0.750	0.750
10	0.344	0.428	0.466	0.310
11	0.271	0.340	0.379	0.210
12	0.000	0.000	0.000	0.000
13	0.000	0.000	0.000	0.000
14	0.199	0.257	0.294	0.131
15	0.000	0.000	0.000	0.000

Ugotavljamo, da je različica mm7AUC izpolnila temeljna pričakovanja, ki smo jih navedli na začetku poglavja. Glede na izračunane rezultate mera

mm7AUC, po našem mnenju, obravnavanih 15 množic po kakovosti razvrsti bolje od obstoječih štirih izpeljank, ob tem pa nudi več informacije kot osnovna AUC. Slednje se nanaša na to, da mera mm7AUC množice z enakim številom napak, za razliko od osnovne AUC, razlikuje med seboj po kakovosti. To je razvidno tudi iz ocen množic 1–7 v tabelah 3.1 in 4.2.

Na podlagi tega smo se odločili, da z razvojem nove mere končamo. Tudi v primeru, da rezultati ne bi bili zadovoljivi, pa nadaljnje nadgrajevanje izpeljanke mm7AUC najbrž ne bi bilo smiselno, in sicer zaradi nevarnosti, da bi tako dobljene nove mere bile preveč prilagojene izbranim 15 množicam. Tedaj bi pri nadaljnjem razvoju bilo smotrno poiskati nova (ali vsaj delno spremenjena) izhodišča, ob tem pa seznam množic točkovnih ocen dopolniti z novimi množicami.

4.6 Opuščena smer razvoja

Naknadno smo poskusili vpeljati še nekoliko drugačno izboljšavo. Ob preučitvi formule izpeljanke scorAUC lahko opazimo naslednjo značilnost. Če (verjetnostni) klasifikator pozitivni primer v nekem paru oceni z nižjo oceno kot negativni primer, postane vrednost spremenljivke *razlika* v formuli scorAUC negativna. Velikost prispevka danega para k skupni vsoti je tako, ne glede na velikost razlike, enaka 0. Ta asimetrija pri obravnavi pozitivne in negativne razlike je omenjena že v [16] in se iz različice scorAUC seveda prenese tudi v vse naše izpeljanke (tj. mere mm1AUC–mm7AUC). Opisano asimetrijo smo na različici scorAUC poskusili odpraviti na naslednji način.

Izvor asimetrije je faktor $if(razlika \geq 0; 1; 0)$ v formuli za scorAUC (ki se dejansko prenese iz osnovne AUC). V primeru, da ta faktor izpustimo, bi se mera scorAUC morala obnašati simetrično. Formula tako dobljene izpeljanke *mm11AUC* bi tedaj bila

$$mm11AUC = \frac{\sum_{\text{preko vseh parov}} razlika}{\check{s}t.\ vseh\ parov} \tag{4.8}$$

kjer gremo v vsoti preko vseh parov enega pozitivnega in enega negativnega primera. Pomen spremenljivke *razlika* je podan v opisu formule 2.1.

Različica mm11AUC ima zalogo vrednosti na intervalu [-1, 1], ki se razlikuje od intervala [0, 1] pri vseh dosedaj obravnavanih merah. Formulo zato malenkostno spremenimo. Izpeljanka *mm12AUC* z zalogo vrednosti [0, 1] je opredeljena takole:

$$mm12AUC = \frac{mm11AUC + 1}{2} \tag{4.9}$$

Zanimivo je, da mm12AUC na prvi pogled vrača popolnoma enake vrednosti kot že obstoječa različica probAUC. mm12AUC smo nameravali dopolniti na enak način kot različice mm1AUC–mm7AUC. Pri tem pa smo prišli do naslednjih ugotovitev:

- dodatno uteževanje z razponom ocen (iz mm1AUC): To dopolnilo v primeru mere mm12AUC ne prinese izboljšanja, saj tako nadgrajena mera dobro oceni vse množice z veliko absolutno vrednostjo relativne širine roba, tj. ne glede na to, ali je razvrstitev pravilna ali ne.
- vpeljava najmanjšega prispevka za pozitivno razliko (iz mm4AUC): Zaradi zveznosti mere mm12AUC, ki je posledica odsotnosti stopnice v njeni formuli, to dopolnilo ne pride v poštev.
- dodatno uteževanje z absolutno širino roba (iz mm6AUC): Mera mm12AUC že sama po sebi primerno kaznuje majhno absolutno širino roba, zato to dopolnilo ne pride v poštev.
- dodatno uteževanje s številom napak (iz mm7AUC): To dopolnilo smo pri različicah mm1AUC–mm7AUC uporabili predvsem zaradi vpliva predhodnih dopolnil 2 in 3 s tega seznama. Ker nobeno od omenjenih tu ne pride v poštev, tudi uteževanje s številom napak ni smiselno.

Poglavje 5

Primerjava različic

V tem poglavju se nahaja nekoliko obsežnejši preizkus uspešnosti vseh obravnavanih različic AUC, tako osnovne kot tudi obstoječih in novorazvitih izpeljank. Preizkušanje se izvede programsko. Program je napisan v jeziku Java s pomočjo okolja NetBeans IDE. Predstavljeni so rezultati. Ugotovljeno je, kako se mere obnašajo in ali so se pričakovanja iz faze načrtovanja posameznih novorazvitih izpeljank uresničila. Testni program predstavlja, poleg novorazvite različice mm7AUC, dodatni rezultat diplomskega dela, in sicer kot pripomoček za razvoj nadaljnjih izboljšav. Delovanje testnega programa je opisano v nadaljevanju.

Predpostavljamo, da so podane množice primerov (kot npr. v tabeli 3.1), radi pa bi dobili oceno kakovosti vsake množice. Vsaka množica vsebuje poljubno število primerov, kjer je vsak primer podan kot točkovna ocena (ki jo vrne klasifikator) in dejanski razred (pozitivni ali negativni). To naj bi bili tudi edini znani podatki.

Na podlagi kriterija o pravilnosti razvrstitve z začetka poglavja 4 naj bi veljalo, da različice AUC vse pravilno razvrščene množice ovrednotijo kot boljše od katerekoli nepravilno razvrščene množice. V kolikor se zgodi, da določena različica neko pravilno razvrščeno množico ovrednoti kot slabšo od neke nepravilno razvrščene množice, to štejemo za *napako različice*. Glede na zgornji kriterij se osnovna mera AUC nikoli ne zmoti. Za napako različice bi sicer bilo možno šteti tudi dogodke, ko določena različica neko množico z eno napako klasifikatorja ovrednoti kot slabšo od neke množice z dvema napakama klasifikatorja, itd. Tovrstnih napak ne obravavamo, saj je lahko ena napaka klasifikatorja hujša in manj zaželena od dveh drugih, manjših napak klasifikatorja, kar bi zelo zapletlo postopek.

Na tem mestu bi želeli poudariti razliko med naslednjimi dogodki. Klasifi-

katorji ocenijo primere, rezultat so točkovne ocene teh primerov. Ilustrativen seznam množic primerov (oz. množic točkovnih ocen) se nahaja v tabeli 3.1. Obravnava samih klasifikatorjev ni predmet tega diplomskega dela, zato smo tudi predpostavili, da so množice primerov že znane. Različice AUC nato s pomočjo formul, predstavljenih v poglavjih 2, 3 in 4, ocenijo množice primerov in s tem klasifikatorje. Na koncu program, opisan v sklopu tega poglavja, na podlagi omenjene predpostavke o pravilnosti razvrstitve, da je najpomembnejši kriterij pravilna razvrstitev primerov, oceni različice AUC. Zaporedje dogodkov je torej sledeče:

1. klasifikatorji uvrstijo primere
2. z različicami AUC ocenimo klasifikatorje
3. program oceni različice AUC

5.1 Opis delovanja testnega programa

Testni program skupaj s komentarji obsega nekaj čez 1500 vrstic, pri tem se določeni vzorci v kodi tudi ponavljajo. Delovanje programa je opisano v naslednjih razdelkih.

S pomočjo generiranja novih množic lahko obnašanje različic AUC hitro preverimo na veliki količini množic, kar je zelo uporabno pri razvoju morebitnih novih izboljšav. Kakovost različic je izražena v obliki števila napak, kot so opredeljene na začetku tega poglavja (ob predpostavki, da morajo pravilno razvrščene množice biti ocenjene bolje od nepravilno razvrščenih). Točkovne ocene novogeneriranih množic so dobljene sistematično s spreminjanjem lastnosti množic, slednje tako med seboj niso neodvisne. Kljub temu pa je opisani postopek povsem uporaben za preizkušanje različic v tem delu.

Katerokoli izmed opisanih treh metod za generiranje novih množic lahko v programu po potrebi seveda izključimo. Če ne uporabimo nobene izmed njih, program obravnava le množice, podane v vhodni datoteki.

Celoten postopek vrednotenja različic AUC je popolnoma deterministično, rezultati so tako ponovljivi.

5.1.1 Vhodni podatki

Obstajati mora vhodna datoteka z množicami primerov. Vsaka vrstica naj bi vsebovala eno množico primerov. Število množic primerov ter število primerov v vsaki množici je pri tem poljubno. Vsak primer je podan s točkovno oceno in oznako dejanskega razreda. Vsebina vhodne datoteke se prebere, rezultat je

seznam množic primerov, ki vsebuje vse podatke iz vhodne datoteke. Množica primerov s šestimi elementi lahko recimo izgleda takole: (0.85p 0.80p 0.75p 0.70n 0.65n 0.60n).

Vhodna datoteka mora zadoščati naslednjim omejitvam. V kolikor so lete za strukturo že obstoječe vhodne datoteke neprimerne, je kodo programa potrebno ustrezno spremeniti.

- Vsaka množica mora biti zapisana po padajočih točkovnih ocenah (ne glede na dejanski razred primerov).
- Točkovna ocena vsakega primera mora biti podana na točno dve decimalni mesti natančno.

5.1.2 Generiranje novih množic na osnovi oženja razpona

Iz vsake obstoječe množice primerov se generirajo nove množice primerov. Vse nove množice se dodajo na skupni seznam množic primerov. Generiranje temelji na sistematičnem spreminjanju treh lastnosti množic, in sicer absolutne širine roba, razpona ter porazdelitve primerov po razredih. Na ta način je mogoče hitro preveriti, koliko napak posamezna različica naredi na večjem številu (sicer sistematično spremenjenih) množic. Postopek generiranja je popolnoma deterministično.

Zaradi preglednosti je vsak od omenjenih treh postopkov opisan v lastnem razdelku. Ta razdelek tako na kratko predstavlja generiranje na osnovi oženja razpona.

Razpon vsake podane množice se po korakih zmanjšuje s prvotnega na nek delež prvotnega. Število korakov je pri tem parameter. Nove točkovne ocene se po absolutni vrednosti v vsakem koraku vedno bolj približujejo srednji vrednosti največjega in najmanjšega primera v podani množici (se torej zgoščajo okoli nje). Vzemimo zgornjo množico (0.85p 0.80p 0.75p 0.70n 0.65n 0.60n) z razponom 0.25. Po npr. 100 korakih dobimo množico (0.72625p 0.72575p 0.72525p 0.72475n 0.72425n 0.72375n) z razponom 0.0025. Razvrstitev primerov se pri tem seveda ohrani. Tako iz vsake množice nastane 100 množic.

5.1.3 Generiranje novih množic na osnovi zmanjševanja absolutne širine roba

Za absolutno širino roba zahtevamo, da se zmanjšuje pri *nespremenjenem* razponu. Absolutna širina roba vsake podane množice se po korakih zmanjšuje s

prvotne na nek delež prvotne. Število korakov je pri tem parameter. Ohraniti se mora razvrstitev vseh primerov, kar pomeni tudi, da mora absolutna širina roba ohraniti predznak.

Izvedba zmanjševanja absolutne širine roba je nekoliko bolj zapletena od oženja razpona. Točkovne ocene največjega pozitivnega, najmanjšega pozitivnega, največjega negativnega in najmanjšega negativnega primera označimo kot maxPOZ, minPOZ, maxNEG oz. minNEG. Za način izračuna novih množic je pomembnih šest osnovnih tipov razvrstitev množic primerov. Pri tipu 1 je absolutna širina roba pozitivna, pri vseh ostalih pa negativna. Tipi so naslednji:
- tip 1: $0.00 \leq minNEG \leq maxNEG < minPOZ \leq maxPOZ \leq 1.00$
- tip 2: $0.00 \leq minNEG < minPOZ < maxNEG < maxPOZ \leq 1.00$
- tip 3: $0.00 \leq minPOZ < minNEG \leq maxNEG < maxPOZ \leq 1.00$
- tip 4: $0.00 \leq minNEG < minPOZ \leq maxPOZ < maxNEG \leq 1.00$
- tip 5: $0.00 \leq minPOZ < minNEG < maxPOZ < maxNEG \leq 1.00$
- tip 6: $0.00 \leq minPOZ \leq maxPOZ < minNEG \leq maxNEG \leq 1.00$

Pri tipu 1 med zmanjševanjem absolutne širine roba točkovni oceni maxPOZ in minNEG ostaneta fiksni. Točkovne ocene vseh ostalih primerov se spremenijo, in sicer po formuli, ki je odvisna od srednje vrednosti točkovnih ocen minPOZ in maxNEG, prav tako pa tudi od velikosti točkovnih ocen maxPOZ in minNEG. Razvrstitev vseh primerov se ohrani. Vse našteto velja tudi za tip 2. V formuli slednjega je potrebno le upoštevati, da je rob negativen. Razpon tako pri tipu 1 kot tipu 2 ostane nespremenjen. Oba tipa torej izpolnjujeta postavljene zahteve.

Zmanjševanje absolutne širine roba se pri tipih 5 in 6 pravzaprav degenerira v oženje razpona, saj tako absolutno širino roba kot razpon določata primera maxNEG in minPOZ. Absolutne širine roba namreč v tem primeru ne moremo zmanjšati brez oženja razpona. Ker razpon ožimo že na poprej opisan način, teh tipov ne bomo obravnavali. Prav tako se razpon spreminja pri tipih 3 in 4 (sicer le z ene strani). Ker pogoj nespremenjenega razpona tudi tu ni izpolnjen, ne bomo obravnavali niti teh dveh tipov.

Generiranje novih množic je torej smiselno dopustiti le v primeru, da je izvorna množica tipa 1 ali 2. To pomeni, da mora veljati naslednji pogoj: $(maxPOZ > maxNEG) AND (minNEG < minPOZ)$.

Vzemimo zgornjo množico (0.85p 0.80p 0.75p 0.70n 0.65n 0.60n) z absolutno širino roba 0.05. Po npr. 100 korakih dobimo množico (0.85p 0.787625p 0.72525p 0.72475n 0.662375n 0.60n) z absolutno širino roba 0.0005. Tako iz vsake množice nastane 100 množic.

5.1.4 Generiranje novih množic na osnovi spreminjanja porazdelitve po razredih

Število korakov je tukaj odvisno od števila primerov v množici. Vseh možnih porazdelitev je namreč $2^{\text{št. primerov}}$. Pri tem se seveda spreminja razmerje med številom pozitivnih in številom negativnih primerov v množici. Porazdelitve, kjer nastopajo samo pozitivni ali samo negativni primeri, izpustimo, saj želimo, da sta vedno zastopana oba razreda. Iz množice (0.85p 0.80p 0.75p 0.70n 0.65n 0.60n) torej dobimo $2^6 - 2 = 62$ množic:

(0.85n 0.80n 0.75n 0.70n 0.65n 0.60p),
(0.85n 0.80n 0.75n 0.70n 0.65p 0.60n),
(0.85n 0.80n 0.75n 0.70n 0.65p 0.60p),
...
(0.85p 0.80p 0.75p 0.70p 0.65p 0.60n).

5.1.5 Izračun vrednosti vseh različic AUC

Izračunajo se vrednosti vseh devetih mer (tj. osnovne, štirih obstoječih izpeljank in štirih novorazvitih) za vse množice s seznama (torej, tako podane kot vse novogenerirane). Vrednosti se izračunajo na podlagi formul, predstavljenih v poglavjih 2, 3 in 4.

5.1.6 Ugotavljanje števila napak

Prešteje se napake, ki jih naredijo posamezne mere, tj. kolikokrat posamezna mera neko pravilno razvrščeno množico oceni slabše od neke nepravilno razvrščene množice. Izpiše se tudi število vseh množic ter koliko od teh je pravilno razvrščenih.

Kot zgled si oglejmo število napak pri naslednjih dveh množicah:

množica 1: 1.00p 0.90p 0.80p 0.20n 0.10n 0.00n
množica 2: 1.00p 0.90p 0.80n 0.20p 0.10n 0.00n

Množica 1 je pravilno, množica 2 pa nepravilno razvrščena. Pri njunem primerjanju nobena od devetih mer ne naredi napake, vse mere množico 1 ocenijo kot boljšo od množice 2. Drugače pa je v primeru oženja razpona. Recimo, da na podlagi zgornjih dveh množic generiramo po 99 novih množic, kjer se razpon postopoma manjša z 1.00 na 0.01. Število napak, ki jih mere naredijo pri ocenjevanju tako dobljenih 200 množic (pri tem jih je 100 razvrščenih pravilno, 100 pa nepravilno), je podano v tabeli 5.1. Kot dodatna informacija sta

pri vsaki od mer navedeni tudi najmanjša ocena, ki jo mera dodeli pravilno razvrščeni množici, in največja ocena, ki jo mera dodeli nepravilno razvrščeni množici. Kot lahko vidimo, je prva od ocen pri vseh štirih obstoječih izpeljankah manjša od druge (in to celo bistveno), kar je tudi vzrok za napake teh različic.

Tabela 5.1: Število napak obravnavanih različic AUC na opisanih 200 množicah

različica	št. napak	min prav.	max nepr.
AUC	0	1.000	0.889
probAUC	49	0.504	0.700
scorAUC	58	0.008	0.467
sondAUC	21	0.501	0.774
softAUC	21	0.514	0.772
mm1AUC	0	0.800	0.467
mm4AUC	0	0.800	0.622
mm6AUC	0	0.777	0.652
mm7AUC	0	0.777	0.580

5.2 Preizkus različic

V tabeli 5.2 so podani rezultati preizkušanja za 40 različnih nastavitev oz. skupin množic. Začetnih množic je pri tem 5 in so izmišljene. Vidimo lahko, da se mere (vsaj pri danih nastavitvah) bistveno razlikujejo po številu napak.

Ob upoštevanju predpostavke o pomembnosti pravilne razvrstive z začetka poglavja 4 osnovna mera AUC ne more narediti napake. V tabeli 5.2 je zato ne navajamo. Kljub temu pa ne smemo pozabiti na pomanjkljivosti te mere, ki so predstavljene v poglavjih 3 in 4 in so bile tudi osnova za razvoj novih različic.

Stolpci AŠR, R in P vsebujejo informacijo o tem, ali so metode generiranja novih množic na podlagi zmanjševanja absolutne širine roba (AŠR), oženja razpona (R) oz. spreminjanja porazdelitve po razredih (P) bile uporabljene. V kolikor je pri neki nastavitvi katera izmed prvih dveh metod bila uporabljena, se v vrstici, ki pripada tej nastavitvi, v stolpcih AŠR oz. R nahajata vrednosti parametrov $k1$ oz. $k2$. Slednja predstavljata število korakov pri dotičnih dveh metodah. V primeru, da katera od metod ni bila uporabljena, se v ustreznem stolpcu nahaja znak '–'. Vrednosti omenjenih parametrov so se v postopku preizkušanja torej spreminjale.

Vrednosti preostalih parametrov so bile konstantne, in sicer: $q = 1/7$ (sondAUC), $\beta = 7$ (softAUC), $n = 1/100$ (mm6AUC in mm7AUC), $m = 9/10$ (mm6AUC in mm7AUC).

Formul za mm6AUC in mm7AUC nismo optimizirali glede na vrednosti parametrov m in n. Zanju smo uporabili zgornji vrednosti, ki sta se nam zdeli smiselni. Izpeljanki mm6AUC in mm7AUC se pri drugih vrednostih parametrov gotovo lahko obnašata slabše (naredita več napak), pri določenih (optimalnih) vrednostih parametrov pa verjetno tudi boljše. Vrednosti parametrov je pred uporabo v vsakem primeru potrebno prilagoditi problemu oz. seznamu množic, ki jih bomo obravnavali. Podobno velja za že obstoječi različici sondAUC in softAUC, pri katerih vrednosti parametrov q in β prav tako nista optimalni. Njuni vrednosti smo pustili enaki kot v opisu tabele 3.1.

V stolpcih 6 in 7 tabele 5.2 sta navedeni števili vseh množic oz. pravilno razvrščenih množic za posamezno nastavitev.

V vsaki vrstici je s krepko pisavo v modri barvi označena različica z najmanjšim številom napak. Pri tem izmed novorazvitih različic upoštevamo le mm7AUC. V primeru, da si v neki vrstici najmanjše število napak deli več različic, ni označena nobena izmed njih.

Pri vseh nastavitvah (A1-E8) je uporabljena metoda, ki generira nove množice na osnovi spreminjanja porazdelitve po razredih. Ker se s to metodo iz dane množice generira po ena množica za vsako možno porazdelitev, podajanje dejanskih razredov ni potrebno. Množice primerov so torej v tabeli 5.2 določene samo s točkovnimi ocenami.

Osnovne lastnosti izvornih množic v nastavitvah A1-E8 so naslednje:

- Izvorna množica A ima maksimalen razpon, točkovne ocene pa so pri tem popolnoma enakomerno porazdeljene po celotnem intervalu.
- Pri izvorni množici B se razpon zoži. Poveča se sicer tudi mediana točkovnih ocen, ki pa naj ne bi imela vpliva na obnašanje različic.
- Izvorna množica C ima maksimalen razpon. Točkovne ocene se po velikosti delijo na dve skupini, pri čemer ena teži k zgornji, druga pa k spodnji meji intervala.
- Izvorna množica D ima maksimalen razpon, točkovne ocene pa so neenakomerno porazdeljene. Pet točkovnih ocen ima namreč podobne vrednosti, vrednost šeste pa zelo odstopa.
- Izvorna množica E ima maksimalen razpon in je podobna množici C, s tem, da sta obe skupini točkovnih ocen še bolj izraziti.

Pri nastavitvah A in B se mera mm7AUC niti enkrat ne zmoti, pri nastavitvah C pa naredi 4 napake. Število napak mm7AUC se bistveno poveča pri nastavitvah D in E. Enak trend je opazen tudi pri vseh obstoječih izpeljankah.

Po kriteriju števila napak je mera mm7AUC boljša od ostalih mer pri 24 nastavitvah od 40. Pri 4 nastavitvah je najboljša mera sondAUC (C4, E2-E4),

od tega je pri 3 tudi mera softAUC boljša od mm7AUC (E2-E4). Pri ostalih 12 nastavitvah pa so mere med seboj delno ali popolnoma izenačene. Tako imajo pri 8 nastavitvah (A1-A4, B1-B4) enako število napak kot mm7AUC vse štiri obstoječe izpeljanke, pri 4 nastavitvah (C1-C3, E1) pa to velja za vsaj eno obstoječo izpeljanko.

Glede na število napak je pri obravnavanih nastavitvah najslabša različica scorAUC. Za odtenek bolje se obnaša mera probAUC. Ti dve meri pri teh nastavitvah veliko večino pravilno razvrščenih množic ocenita z nižjo oceno od določenih nepravilno razvrščenih množic. Precej bolje se obnese različica softAUC, najboljša od obstoječih različic pa je mera sondAUC. Pri zadnjih dveh je potrebno upoštevati, da njuno obnašanje ni optimalno, saj sta bila parametra β in q izbrana po občutku.

Tabela 5.2: Primerjava uspešnosti obravnavanih različic AUC

#	izvorna množica	AŠR	R	P	vseh mn.	pr. mn.	probAUC	scorAUC	sondAUC	softAUC	mm1AUC	mm4AUC	mm6AUC	mm7AUC
A1	1.00 0.80 0.60 0.40 0.20 0.00	–	–	da	62	5	0	0	0	0	0	0	0	0
A2	kot v A1	30	–	da	1860	150	0	0	0	0	0	0	0	0
A3	kot v A1	100	–	da	6200	500	0	0	0	0	0	0	2	0
A4	kot v A1	1000	–	da	62000	5000	0	0	0	0	0	0	26	0
A5	kot v A1	–	30	da	1860	150	115	120	55	69	0	0	0	**0**
A6	kot v A1	–	100	da	6200	500	385	405	192	236	0	0	0	**0**
A7	kot v A1	–	1000	da	62000	5000	3885	4070	1945	2374	0	0	0	**0**
A8	kot v A1	30	30	da	55800	4500	3757	3762	1863	2889	0	0	42	**0**
B1	0.90 0.88 0.86 0.81 0.77 0.76	–	–	da	62	5	0	0	0	0	0	0	0	0
B2	kot v B1	30	–	da	1860	150	0	0	0	0	0	0	3	0
B3	kot v B1	100	–	da	6200	500	0	0	0	0	0	0	10	0
B4	kot v B1	1000	–	da	62000	5000	0	0	0	0	0	0	108	0
B5	kot v B1	–	30	da	1860	150	121	125	57	119	0	0	0	**0**
B6	kot v B1	–	100	da	6200	500	411	426	193	406	0	0	0	**0**
B7	kot v B1	–	1000	da	62000	5000	4135	4274	1957	4077	0	0	0	**0**
B8	kot v B1	30	30	da	55800	4500	3896	3915	1990	3869	0	0	153	**0**
C1	1.00 0.90 0.80 0.20 0.10 0.00	–	–	da	62	5	0	2	0	0	2	0	0	0
C2	kot v C1	30	–	da	1860	150	30	97	0	2	97	37	55	0
C3	kot v C1	100	–	da	6200	500	108	325	0	12	325	130	185	0
C4	kot v C1	1000	–	da	62000	5000	1075	3262	**0**	142	3262	1327	1881	3
C5	kot v C1	–	30	da	1860	150	132	136	63	86	60	0	0	**0**
C6	kot v C1	–	100	da	6200	500	448	456	218	288	200	0	0	**0**
C7	kot v C1	–	1000	da	62000	5000	4523	4579	2201	2910	2000	0	0	**0**
C8	kot v C1	30	30	da	55800	4500	4250	4345	1905	2838	2910	1170	1918	**1**
D1	1.00 0.99 0.98 0.97 0.96 0.00	–	–	da	62	5	3	3	2	3	3	3	3	**1**
D2	kot v D1	30	–	da	1860	150	90	90	76	90	90	90	90	**30**
D3	kot v D1	100	–	da	6200	500	300	300	253	300	300	300	300	**100**
D4	kot v D1	1000	–	da	62000	5000	3000	3000	2534	3000	3000	3000	3000	**1001**
D5	kot v D1	–	30	da	1860	150	134	134	102	115	90	90	90	**39**
D6	kot v D1	–	100	da	6200	500	448	448	345	384	300	300	303	**131**
D7	kot v D1	–	1000	da	62000	5000	4492	4495	3462	3850	3000	3000	3030	**1319**
D8	kot v D1	30	30	da	55800	4500	4020	4020	3160	3480	2700	2700	2700	**1151**
E1	1.00 0.99 0.98 0.02 0.01 0.00	–	–	da	62	5	2	2	0	2	2	2	2	0
E2	kot v E1	30	–	da	1860	150	130	130	**2**	30	130	122	128	55
E3	kot v E1	100	–	da	6200	500	438	441	**2**	106	441	411	432	187
E4	kot v E1	1000	–	da	62000	5000	4405	4422	**10**	1098	4422	4127	4347	1888
E5	kot v E1	–	30	da	1860	150	140	140	95	110	60	60	64	**0**
E6	kot v E1	–	100	da	6200	500	470	473	325	370	200	200	214	**0**
E7	kot v E1	–	1000	da	62000	5000	4727	4733	3262	3706	2000	2000	2152	**0**
E8	kot v E1	30	30	da	55800	4500	4420	4421	2113	3047	3900	3660	3958	**1795**

Poglavje 6

Zaključek

V tem delu so bile preučene mere s področja analize ROC za ocenjevanje klasifikatorjev. Na enem mestu so obravnavane osnovna AUC in njene štiri obstoječe izpeljanke. Pri tem so njihove formule zapisane v enotni obliki. Predstavljen je bil zgled, iz katerega je razvidno, kako različice ocenjujejo konkretne množice primerov. Navedene so slabosti različic, ki so pri tem bile odkrite. Analizirane so bile tudi lastnosti množic primerov ter njihov vpliv na obnašanje različic.

Ena od težav pri ocenjevanju klasifikatorjev je subjektivnost. Da bi to težavo ublažili, smo naredili nekaj predpostavk. Razvita je bila nova mera, ki naj bi klasifikatorje ocenjevala verodostojno ter bila obenem bolj informativna od obravnavanih obstoječih različic. Glede na raznolikost točkovnih ocen, dobljenih od klasifikatorjev, smo z namenom splošne uporabnosti nove mere v njeno formulo vgradili dva parametra, ki vplivata na njeno občutljivost na določene lastnosti. Optimalni vrednosti teh parametrov pa je potrebno določiti eksperimentalno (po možnosti na osnovi konkretnih množic primerov, na katerih bo ta mera uporabljena), kar je lahko netrivialno opravilo.

Implementiran je bil program za primerjavo različic, ki se lahko uporabi tudi kot razvojno orodje pri morebitnem nadaljnjem izboljševanju. Pravilnost delovanja programa smo preverjali tako med samim razvojem kot tudi ob koncu le-tega. Kljub temu možnosti napak ne moremo popolnoma izključiti.

Na osnovi tabele 5.2 in ugotovitev iz poglavja 4 lahko sklenemo, da je novorazvita različica mm7AUC izpolnila zastavljene cilje z začetka omenjenega poglavja. Na uspešnost mere mm7AUC najbrž v manjši meri vpliva dejstvo, da smo množice sistematično generirali tudi na podlagi lastnosti, za katere smo pri novorazvitih različicah želeli doseči robustnost. Zavedati se je potrebno, da uspešnost nove različice velja v okviru že omenjenih predpostavk. Za ugotavljanje nadaljnjih zakonitosti pri obnašanju različic AUC bi bilo potrebno

opraviti obsežnejše testiranje, kar presega okvir tega dela.

Bodoče delo bi lahko zajemalo iskanje morebitnih pomembnih lastnosti množic, ki sedaj niso bile upoštevane in bi lahko izboljšale kakovost vrednotenja klasifikatorjev. Glede na to, da smo pri razvoju v večji meri izhajali iz različice scorAUC, bi možnosti izboljšav bolj podrobno lahko preverili tudi pri ostalih obstoječih izpeljankah, tj. probAUC, sondAUC in softAUC. Veliko možnosti za nadgradnjo je tudi pri testnem programu. Izdelalo bi se lahko (boljši) uporabniški vmesnik, prav tako bi se rezultate lahko vizualno upodobilo.

Različice bi lahko primerjali še z druge perspektive. Ogledali bi si, kako se le-te obnašajo pri določenem razponu ter rezultate primerjali s tistimi pri večjem oz. manjšem razponu. Število napak posameznih različic bi torej spremljali za vsak razpon posebej. Podobno bi lahko naredili tudi z absolutno širino roba in porazdelitvijo po razredih. Pri tem bi lahko dodali šum in tako v postopek vnesli naključnost. Primerjavo bi lahko izvedli tudi za različne velikosti množic primerov (npr. 20, 50 ali 100 primerov). Poleg tega bi za množice primerov lahko uporabili podatke iz javno dostopnih baz.

Dodatek A

Testni program

V tem dodatku podajamo zanimivejše dele izvorne kode testnega programa iz poglavja 5.

A.1 Podatkovna struktura

Spodaj se nahaja razred MnozicaPrimerov. Vsak objekt tega razreda vsebuje vso informacijo o posamezni (podani ali generirani) množici primerov.

```
1  public class MnozicaPrimerov {
2      List mnozica;
3
4      private int steviloPozitivnihPrimerov;
5      private int steviloNegativnihPrimerov;
6      private double najvecjiPozitivniPrimer;
7      private double najmanjsiPozitivniPrimer;
8      private double najvecjiNegativniPrimer;
9      private double najmanjsiNegativniPrimer;
10     private double absolutnaSirinaRoba;
11     boolean razvrscenaPravilno;
12     private double najvecjiPrimer;
13     private double najmanjsiPrimer;
14     private double razpon;
15     private double pozitivniRazpon;
16     private double negativniRazpon;
17
18     private double AUC;
19     private double probAUC;
20     private double scorAUC;
21     private double sondAUC;
22     private double softAUC;
```

```
23      private double mm1AUC;
24      private double mm4AUC;
25      private double mm6AUC;
26      private double mm7AUC;
27
28      public MnozicaPrimerov(List mn) {
29          mnozica = mn;
30
31          List pozPrimeri = (ArrayList)mnozica.get(0);
32          List negPrimeri = (ArrayList)mnozica.get(1);
33
34          steviloPozitivnihPrimerov = pozPrimeri.size();
35          steviloNegativnihPrimerov = negPrimeri.size();
36
37          najvecjiPozitivniPrimer = ((Primer)pozPrimeri.get(0)).
                getTockovnaOcena();
38          najmanjsiPozitivniPrimer = ((Primer)pozPrimeri.get(
                steviloPozitivnihPrimerov-1)).getTockovnaOcena();
39          najvecjiNegativniPrimer = ((Primer)negPrimeri.get(0)).
                getTockovnaOcena();
40          najmanjsiNegativniPrimer = ((Primer)negPrimeri.get(
                steviloNegativnihPrimerov-1)).getTockovnaOcena();
41
42          absolutnaSirinaRoba = najmanjsiPozitivniPrimer -
                najvecjiNegativniPrimer;
43
44          if(absolutnaSirinaRoba>0)
45              razvrscenaPravilno = true;
46          else
47              razvrscenaPravilno = false;
48
49          najvecjiPrimer = Math.max(((Primer)pozPrimeri.get(0)).
                getTockovnaOcena(), ((Primer)negPrimeri.get(0)).
                getTockovnaOcena());
50          najmanjsiPrimer = Math.min(((Primer)pozPrimeri.get(
                steviloPozitivnihPrimerov-1)).getTockovnaOcena(), ((
                Primer)negPrimeri.get(steviloNegativnihPrimerov-1)).
                getTockovnaOcena());
51
52          razpon = najvecjiPrimer - najmanjsiPrimer;
53
54          pozitivniRazpon = najvecjiPozitivniPrimer -
                najmanjsiPozitivniPrimer;
55
56          negativniRazpon = najvecjiNegativniPrimer -
                najmanjsiNegativniPrimer;
57      }
```

```
58
59      ...
60
61  }
```

A.2 Generiranje množic

V tem podpoglavju sta prikazani dve metodi od skupno treh za generiranje novih množic primerov iz že obstoječih množic.

```
1  public class Main {
2      static List vseMnozice = new ArrayList();
3
4      ...
5
6      // generiranje novih množic na osnovi spreminjanja
           porazdelitve po razredih
7      public static boolean noveMnozicePorazdelitev() {
8          // seznam novogeneriranih množic
9          List noveVseMnozice = new ArrayList();
10         // število korakov (predstavlja tudi število novih množic)
               . Število korakov je odvisno od števila primerov v
               množici.
11         int steviloKorakov;
12
13         // preko vseh obstoječih množic (tj. množic, podanih z
               vhodno datoteko)
14         for(int i=0; i<vseMnozice.size(); i++) {
15             MnozicaPrimerov trenutnaMnozica = (MnozicaPrimerov)
                   vseMnozice.get(i);
16             List trenutniPozPrimeri = (ArrayList)trenutnaMnozica.
                   mnozica.get(0);
17             List trenutniNegPrimeri = (ArrayList)trenutnaMnozica.
                   mnozica.get(1);
18
19             int stPoz = trenutnaMnozica.
                   getSteviloPozitivnihPrimerov();
20             int stNeg = trenutnaMnozica.
                   getSteviloNegativnihPrimerov();
21
22             // vseh možnih porazdelitev je 2^(število primerov)
23             steviloKorakov = (int) Math.pow(2, (stPoz + stNeg));
24
25             // polje točkovnih ocen vseh primerov trenutne množice
                   , urejeno po velikosti od največje točkovne ocene
```

```
                do najmanjše točkovne ocene
26              Double[] tockOcene = new Double[stPoz + stNeg];
27              int stevec = 0;
28
29              // napolnimo polje
30              for(int j=0; j<trenutniPozPrimeri.size();j++) {
31                  Primer trenutniPrimer = (Primer)trenutniPozPrimeri
                        .get(j);
32
33                  tockOcene[stevec] = trenutniPrimer.
                        getTockovnaOcena();
34                  stevec++;
35              }
36
37              for(int j=0; j<trenutniNegPrimeri.size();j++) {
38                  Primer trenutniPrimer = (Primer)trenutniNegPrimeri
                        .get(j);
39
40                  tockOcene[stevec] = trenutniPrimer.
                        getTockovnaOcena();
41                  stevec++;
42              }
43
44              // polje tudi dejansko uredimo po velikosti (do zdaj
                    ni bilo)
45              Arrays.sort(tockOcene, Collections.reverseOrder());
46
47              // števec h, zapisan v dvojiški obliki, predstavlja
                    porazdelitev po razredih (pri tem morajo biti
                    zapisane tudi začetne ničle). Ničla označuje
                    pripadnost negativnemu razredu, enica pa pripadnost
                     pozitivnemu razredu. Porazdelitve, kjer nastopajo
                    samo pozitivni ali pa samo negativni primeri,
                    izpustimo, saj želimo, da sta vedno zastopana oba
                    razreda. Spodnjo mejo zanke zato povečamo za 1,
                    zgornjo pa zmanjšamo za 1 (tako nikoli ne dobimo
                    samih ničel oz. samih enic). Gremo torej od 00...01
                     (predstavlja nn...np) do 11...10 (predstavlja pp
                    ...pn).
48              for(int h=1; h<steviloKorakov-1; h++) {
49                  // vsi primeri nove množice primerov
50                  List noviVsiPrimeri = new ArrayList();
51                  // vsi pozitivni primeri nove množice primerov
52                  List noviPozPrimeri = new ArrayList();
53                  // vsi negativni primeri nove množice primerov
54                  List noviNegPrimeri = new ArrayList();
55
```

```
56                  // dvojiški zapis, ki predstavlja porazdelitev po
                        razredih
57                  String dvojiskiZapis = Integer.toString(h, 2);
58
59                  int dolzina = dvojiskiZapis.length();
60
61                  // dodamo začetne ničle
62                  if(dolzina < (stPoz + stNeg))
63                      for(int k=0; k<(stPoz+stNeg-dolzina); k++)
64                          dvojiskiZapis = ''0'' + dvojiskiZapis;
65
66                  // primere uvrstimo med pozitivne oz. negativne na
                        podlagi dvojiške vrednosti
67                  for(int k=0; k<dvojiskiZapis.length();k++) {
68                      if(dvojiskiZapis.charAt(k) == '1') {
69                          // novi primer
70                          Primer noviPrimer = new Primer(tockOcene[k
                                ].doubleValue(), ''p'');
71
72                          // novi primer dodamo na seznam pozitivnih
                                primerov nove množice primerov
73                          noviPozPrimeri.add(noviPrimer);
74                      }
75                      else if(dvojiskiZapis.charAt(k) == '0'){
76                          // novi primer
77                          Primer noviPrimer = new Primer(tockOcene[k
                                ].doubleValue(), ''n'');
78
79                          // novi primer dodamo na seznam negativnih
                                primerov nove množice primerov
80                          noviNegPrimeri.add(noviPrimer);
81                      }
82                  }
83
84                  // na seznam vseh primerov nove množice primerov
                        dodamo seznam pozitivnih primerov in seznam
                        negativnih primerov
85                  noviVsiPrimeri.add(noviPozPrimeri);
86                  noviVsiPrimeri.add(noviNegPrimeri);
87
88                  // ustvarimo novo množico primerov
89                  MnozicaPrimerov novaMnozica = new MnozicaPrimerov(
                        noviVsiPrimeri);
90
91                  // novo množico primerov dodamo na seznam vseh
                        novogeneriranih množic primerov
92                  noveVseMnozice.add(novaMnozica);
```

```
93              }
94          }
95
96          // vse obstoječe množice odstranimo, saj so bile (ponovno)
                generirane skupaj z novimi množicami
97          vseMnozice.clear();
98
99          // novogenerirane množice primerov dodamo na seznam vseh
                množic primerov (na seznamu vseh množic primerov so
                sedaj tako prvotne množice kot tudi novogenerirane
                množice)
100         return vseMnozice.addAll(noveVseMnozice);
101     }
102
103     ...
104
105     // generiranje novih množic na osnovi oženja absolutne širine
            roba
106     public static boolean noveMnoziceAbsolutnaSirinaRoba(int stKor
            ) {
107         // seznam novogeneriranih množic
108         List noveVseMnozice = new ArrayList();
109         // število korakov (število novih množic pa bo za ena
                manjše, saj prvotne množice ni potrebno generirati)
110         int steviloKorakov = stKor;
111         // relativna velikost spremembe točkovnih ocen v vsakem
                koraku
112         double velikostKoraka = 1.0/steviloKorakov;
113         // aritmetična sredina roba, izračunana na podlagi
                najmanjšega pozitivnega in največjega negativnega
                primera
114         double aritmeticnaSredina;
115
116         // preko vseh obstoječih množic (tj. množic, podanih z
                vhodno datoteko)
117         for(int i=0; i<vseMnozice.size(); i++) {
118             MnozicaPrimerov trenutnaMnozica = (MnozicaPrimerov)
                    vseMnozice.get(i);
119             List trenutniPozPrimeri = (ArrayList)trenutnaMnozica.
                    mnozica.get(0);
120             List trenutniNegPrimeri = (ArrayList)trenutnaMnozica.
                    mnozica.get(1);
121
122             // če kateri od naslednjih dveh pogojev ni izpolnjen,
                    iz trenutne množice ne generiramo novih množic (jo
                    torej preskočimo):
123             // - največji pozitivni primer mora biti večji od
```

```
                največjega negativnega primera
124             // - najmanjši negativni primer mora biti manjši od
                    najmanjšega pozitivnega primera
125             if(!((trenutnaMnozica.getNajvecjiPozitivniPrimer() >
                    trenutnaMnozica.getNajvecjiNegativniPrimer()) &&
126                  (trenutnaMnozica.getNajmanjsiNegativniPrimer() <
                        trenutnaMnozica.getNajmanjsiPozitivniPrimer()))
                        )
127                  continue;
128
129             aritmeticnaSredina = (trenutnaMnozica.
                    getNajmanjsiPozitivniPrimer() + trenutnaMnozica.
                    getNajvecjiNegativniPrimer()) / 2.0;
130
131             for(int h=0; h<steviloKorakov-1; h++) {
132                  // vsi primeri nove množice primerov
133                  List noviVsiPrimeri = new ArrayList();
134                  // vsi pozitivni primeri nove množice primerov
135                  List noviPozPrimeri = new ArrayList();
136                  // vsi negativni primeri nove množice primerov
137                  List noviNegPrimeri = new ArrayList();
138
139                  // preko vseh pozitivnih primerov dane množice
                        primerov
140                  for(int j=0; j<trenutniPozPrimeri.size();j++) {
141                       Primer trenutniPrimer = (Primer)
                            trenutniPozPrimeri.get(j);
142                       double trenutnaTockovnaOcena = trenutniPrimer.
                            getTockovnaOcena();
143
144                       double novaTockovnaOcena;
145                       // v vsakem koraku se točkovna ocena
                            najmanjšega pozitivnega primera spremeni
146                       double noviNajmanjsiPozitivniPrimer =
                            aritmeticnaSredina + ((trenutnaMnozica.
                            getNajmanjsiPozitivniPrimer() -
                            aritmeticnaSredina) * (1 - (h+1)*
                            velikostKoraka));
147                       // tako se v vsakem koraku spremeni tudi
                            pozitivni razpon
148                       double noviPozRazpon = trenutnaMnozica.
                            getNajvecjiPozitivniPrimer() -
                            noviNajmanjsiPozitivniPrimer;
149
150                       // formula, po kateri se izračuna nova
                            točkovna ocena za vsak pozitivni primer
                            vsake množice primerov v vsakem koraku. Pri
```

```
                    tem se nove točkovne ocene enakomerno
                    razporedijo preko novega pozitivnega
                    razpona. Spodnja formula obravnava največji
                     pozitivni primer.
151                 if(trenutnaTockovnaOcena == trenutnaMnozica.
                        getNajvecjiPozitivniPrimer()) {
152                     novaTockovnaOcena = trenutnaTockovnaOcena;
153                 }
154                 // Spodnja formula obravnava najmanjši
                        pozitivni primer.
155                 else if (trenutnaTockovnaOcena ==
                        trenutnaMnozica.getNajmanjsiPozitivniPrimer
                        ()) {
156                     novaTockovnaOcena =
                            noviNajmanjsiPozitivniPrimer;
157                 }
158                 // Spodnja formula obravnava vse ostale
                        primere.
159                 else {
160                     novaTockovnaOcena =
                            noviNajmanjsiPozitivniPrimer + (
                            trenutnaTockovnaOcena - trenutnaMnozica
                            .getNajmanjsiPozitivniPrimer()) * (
                            noviPozRazpon / trenutnaMnozica.
                            getPozitivniRazpon());
161                 }
162
163                 // novi primer
164                 Primer noviPrimer = new Primer(
                        novaTockovnaOcena, trenutniPrimer.
                        getDejanskiRazred());
165
166                 // novi primer dodamo na seznam pozitivnih
                        primerov nove množice primerov
167                 noviPozPrimeri.add(noviPrimer);
168             }
169
170             // preko vseh negativnih primerov dane množice
                    primerov
171             for(int j=0; j<trenutniNegPrimeri.size();j++) {
172                 Primer trenutniPrimer = (Primer)
                        trenutniNegPrimeri.get(j);
173                 double trenutnaTockovnaOcena = trenutniPrimer.
                        getTockovnaOcena();
174
175                 double novaTockovnaOcena;
176                 // v vsakem koraku se točkovna ocena
```

```
                    največjega negativnega primera spremeni
177                 double noviNajvecjiNegativniPrimer =
                        aritmeticnaSredina + ((trenutnaMnozica.
                        getNajvecjiNegativniPrimer() -
                        aritmeticnaSredina) * (1 - (h+1)*
                        velikostKoraka));
178                 // tako se v vsakem koraku spremeni tudi
                        negativni razpon
179                 double noviNegRazpon =
                        noviNajvecjiNegativniPrimer -
                        trenutnaMnozica.getNajmanjsiNegativniPrimer
                        ();
180
181                 // formula, po kateri se izračuna nova
                        točkovna ocena za vsak negativni primer
                        vsake množice primerov v vsakem koraku. Pri
                         tem se nove točkovne ocene enakomerno
                        razporedijo preko novega negativnega
                        razpona. Spodnja formula obravnava
                        najmanjši negativni primer.
182                 if(trenutnaTockovnaOcena == trenutnaMnozica.
                        getNajmanjsiNegativniPrimer()) {
183                     novaTockovnaOcena = trenutnaTockovnaOcena;
184                 }
185                 // Spodnja formula obravnava največji
                        negativni primer.
186                 else if (trenutnaTockovnaOcena ==
                        trenutnaMnozica.getNajvecjiNegativniPrimer
                        ()) {
187                     novaTockovnaOcena =
                            noviNajvecjiNegativniPrimer;
188                 }
189                 // Spodnja formula obravnava vse ostale
                        primere.
190                 else {
191                     novaTockovnaOcena =
                            noviNajvecjiNegativniPrimer - (
                            trenutnaMnozica.
                            getNajvecjiNegativniPrimer() -
                            trenutnaTockovnaOcena) * (noviNegRazpon
                             / trenutnaMnozica.getNegativniRazpon()
                            );
192                 }
193
194                 // novi primer
195                 Primer noviPrimer = new Primer(
                        novaTockovnaOcena, trenutniPrimer.
```

```
                        getDejanskiRazred());
196
197                     // novi primer dodamo na seznam negativnih
                            primerov nove množice primerov
198                     noviNegPrimeri.add(noviPrimer);
199                 }
200
201                 // na seznam vseh primerov nove množice primerov
                        dodamo seznam pozitivnih primerov in seznam
                        negativnih primerov
202                 noviVsiPrimeri.add(noviPozPrimeri);
203                 noviVsiPrimeri.add(noviNegPrimeri);
204
205                 // ustvarimo novo množico primerov
206                 MnozicaPrimerov novaMnozica = new MnozicaPrimerov(
                        noviVsiPrimeri);
207
208                 // novo množico primerov dodamo na seznam vseh
                        novogeneriranih množic primerov
209                 noveVseMnozice.add(novaMnozica);
210             }
211         }
212
213         // novogenerirane množice primerov dodamo na seznam vseh
                množic primerov (na seznamu vseh množic primerov so
                sedaj tako prvotne množice kot tudi novogenerirane
                množice)
214         return vseMnozice.addAll(noveVseMnozice);
215     }
216
217     ...
```

A.3 Računanje vrednosti različic

Metoda izracunmm4AUC() izračuna vrednost različice mm4AUC za vse množice primerov. Podobne metode obstajajo tudi za preostalih osem različic AUC. Zaradi zagotavljanja čim večje svobode pri razvoju so metode izvedene kot povsem samostojne (dedovanje ni uporabljeno).

```
1     ...
2
3     // metoda za izračun vrednosti mm4AUC za vsako izmed množic
          primerov (za vse množice naenkrat)
4     public static void izracunmm4AUC() {
5         double vrednost;
```

```
6
7           for(int k=0; k<vseMnozice.size(); k++) {
8               MnozicaPrimerov trenutnaMnozica = (MnozicaPrimerov)
                    vseMnozice.get(k);
9               List trenutniPozPrimeri = (ArrayList)trenutnaMnozica.
                    mnozica.get(0);
10              List trenutniNegPrimeri = (ArrayList)trenutnaMnozica.
                    mnozica.get(1);
11
12              vrednost = 0;
13              int stPoz = trenutnaMnozica.
                    getSteviloPozitivnihPrimerov();
14              int stNeg = trenutnaMnozica.
                    getSteviloNegativnihPrimerov();
15
16              double razpon = trenutnaMnozica.getRazpon();
17
18              if(razpon == 0)
19                  System.out.println(``Izračun mm4AUC: razpon je
                        enak 0!'');
20
21              for(int i=0; i<stPoz; i++) {
22                  for(int j=0; j<stNeg; j++) {
23                      double razlika = ((Primer)trenutniPozPrimeri.
                            get(i)).getTockovnaOcena() - ((Primer)
                            trenutniNegPrimeri.get(j)).getTockovnaOcena
                            ();
24
25                      if(razlika > 0) {
26                          double a = razlika / razpon;
27
28                          if(a>=0.5)
29                              vrednost = vrednost + a;
30                          else if (a!=0)
31                              vrednost = vrednost + 0.5;
32                      }
33                  }
34              }
35
36              vrednost = vrednost / (stPoz * stNeg);
37
38              trenutnaMnozica.setmm4AUC(vrednost);
39          }
40      }
41
42      ...
43
```

```
44 } // konec razreda Main
```

A.4 Štetje napak

Spodaj se nahaja metoda mm7AUCpreizkus(), ki ob koncu testiranja prešteje število pravilno razvrščenih množic, ki jim je mm7AUC dodelila nižjo oceno kot neki nepravilno razvrščeni množici. Podobne metode obstajajo tudi za preostalih osem različic AUC.

```
1  public class mm7AUCpreizkus {
2      private double najmanmm7AUCpravRazvMnozice = 1;
3      private double najvecmm7AUCnepravRazvMnozice = 0;
4
5      private int steviloNapak = 0;
6
7      public mm7AUCpreizkus() {
8          for(int i=0; i<Main.vseMnozice.size(); i++) {
9              MnozicaPrimerov trenutnaMnozica = (MnozicaPrimerov)
                   Main.vseMnozice.get(i);
10
11             if(trenutnaMnozica.getRazvrscenaPravilno()) {
12                 if(trenutnaMnozica.getmm7AUC() <
                       najmanmm7AUCpravRazvMnozice)
13                     najmanmm7AUCpravRazvMnozice = trenutnaMnozica.
                           getmm7AUC();
14             }
15             else {
16                 if(trenutnaMnozica.getmm7AUC() >
                       najvecmm7AUCnepravRazvMnozice)
17                     najvecmm7AUCnepravRazvMnozice =
                           trenutnaMnozica.getmm7AUC();
18             }
19         }
20
21         for(int i=0; i<Main.vseMnozice.size(); i++) {
22             MnozicaPrimerov trenutnaMnozica = (MnozicaPrimerov)
                   Main.vseMnozice.get(i);
23
24             if(trenutnaMnozica.getRazvrscenaPravilno()) {
25                 if(trenutnaMnozica.getmm7AUC() <
                       najvecmm7AUCnepravRazvMnozice)
26                     steviloNapak = steviloNapak + 1;
27             }
28         }
29     }
```

```
30
31      public int getSteviloNapak() {
32          return steviloNapak;
33      }
34
35      public void setSteviloNapak(int stNap) {
36          steviloNapak = stNap;
37      }
38  }
```

Slike

Tabele

Literatura

[1] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, št. 7, zv. 30, str. 1145-1159, 1997.

[2] L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and regression trees*, Belmont, CA, ZDA: Wadsworth International Group, 1984.

[3] T. Calders, S. Jaroszewicz, "Efficient AUC optimization for classification," v zborniku *Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2007, str. 42-53.

[4] J. P. Egan, *Signal detection theory and ROC analysis*, New York, NY, ZDA: Academic Press, 1975.

[5] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters, special issue on ROC analysis*, št. 8, zv. 27, str. 861-874, 2006.

[6] C. Ferri, P. Flach, J. Hernández-Orallo, A. Senad, "Modifying ROC curves to incorporate predicted probabilities," v zborniku *Proceedings of the Second Workshop on ROC Analysis in Machine Learning*, 2005.

[7] P. Flach, S. Wu "Repairing concavities in ROC curves," v zborniku *Proceedings of the 2003 UK Workshop on Computational Intelligence*, 2003, str. 38-44.

[8] D. M. Green, J. A. Swets, *Signal detection theory and psychophysics*, New York, NY, ZDA: John Wiley and Sons, 1966.

[9] J. A. Hanley, B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, št. 1, zv. 143, str. 29-36, 1982.

[10] J. Huang, C. X. Ling, "Partial ensemble classifiers selection for better ranking," v zborniku *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005, str. 653-656.

[11] T. D. Koepsell, N. S. Weiss, *Epidemiologic methods: Studying the occurrence of illness*, New York, NY, ZDA: Oxford University Press, 2003.

[12] S. A. Macskassy, F. Provost, "Confidence bands for ROC curves: Methods and an empirical study," v zborniku *Proceedings of the First Workshop on ROC Analysis in Artificial Intelligence*, 2004, str. 61-70.

[13] N. A. Obuchowski, "Receiver operating characteristic curves and their use in radiology," *Radiology*, št. 1, zv. 229, str. 3-8, 2003.

[14] K. A. Spackman, "Signal detection theory: Valuable tools for evaluating inductive learning," v zborniku *Proceedings of the Sixth International Workshop on Machine Learning*, 1989, str. 160-163.

[15] J. A. Swets, "Measuring the accuracy of diagnostic systems," *Science*, št. 4857, zv. 240, str. 1285-1293, 1988.

[16] S. Vanderlooy, E. Hüllermeier, "A critical analysis of variants of the AUC," *Machine Learning*, št. 3, zv. 72, str. 247-262, 2008.

[17] S. Wu, P. A. Flach, "Scored and weighted AUC metrics for classifier evaluation and selection," v zborniku *Proceedings of the Second Workshop on ROC Analysis in Machine Learning*, 2005.

[18] S. Wu, P. Flach, C. Ferri, "An improved model selection heuristic for AUC," v zborniku *Proceedings of the Eighteenth European Conference on Machine Learning*, 2007, str. 478-489.

[19] (2002) Receiver operating characteristic (ROC) literature research. Dostopno na:
http://splweb.bwh.harvard.edu:8000/pages/ppl/zou/roc.html