UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Biček

Grafični gradnik za merjenje kvalitete klasifikatorja s pomočjo krivulj

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Janez Demšar

Ljubljana, 2009

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: 01555/2009
Datum: 05.04.2009

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

Kandidat: **MIHA BIČEK**

Naslov: **GRAFIČNI GRADNIK ZA MERJENJE KVALITETE KLASIFIKATORJA S POMOČJO KRIVULJ**

WIDGET FOR MEASURING CLASSIFIER PERFORMANCE BASED ON CURVES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V zadnjem času je tudi strojnem učenju popularno merjenje kvalitete klasifikatorja s krivuljami ROC, ki kažejo primerjavo med senzitivnostjo in specifičnostjo klasifikatorja pri različnih klasifikacijskih pragih. Poleg njih obstaja tudi več drugih krivulj, npr. krivulja dviga, ki kaže dobiček pri različnih pragih, kalibracijska krivulja, ki kaže razmerje med napovedano in pravo verjetnostjo razreda, in druge.

Sestavite grafični gradnik za okolje Orange, ki bo omogočal risanje krivulj, ki jih definira uporabnik. Kot vnaprej definirane krivulje naj pozna nekaj splošnih krivulj (npr. ROC, krivulja dviga in podobne), uporabniku pa naj omogoča, da za izračun vrednosti, ki jih kažeta osi grafa, poda poljubno formulo, v kateri nastopajo verjetnosti, vrednosti atributov, deleži pravilnih in napačnih pozitivni in negativnih primerov in podobno.

Uporabo gradnika preskusite na praktičnih primerih.

Mentor:

doc. dr. Janez Demšar

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

Zahvala

Zahvaljujem se doc. dr. Janezu Demšarju, ki me je navdušil za področje umetne inteligence in mi omogočil opravljanje diplomskega dela.

Zahvaljujem se tudi prof. Petru Palu, ki je lektoriral diplomsko delo in Mihu Štajdoharju za praktične nasvete pri programiranju v okolju Orange.

Kazalo

Slovarček uporabljenih angleških strokovnih izrazov

Receiver Operating Characheristics (ROC)	karakteristika delovanja sprejemnika
Precision-Recall (PR)	preciznost-priklic
Lift Curve	krivulja dviga
true positive	resnični pozitivni primer
false positive	lažni pozitivni primer
true negative	resnični negativni primer
false negative	lažni negativni primer
true positive rate	delež resničnih pozitivnih primerov
false positive rate	delež lažnih pozitivnih primerov
recall	priklic
precision	preciznost
accuracy	natančnost
threshold	prag
example clasification accuracy (eCA)	klasifikacijska točnost primera
clasification accuracy	klasifikacijska točnost
vertical averaging	navpično povprečenje
treshold averaging	povprečenje glede na prag

Povzetek

V strojnem učenju se pogosto srečamo z nalogo ocenjevanja in primerjanja različnih klasifikatorjev med sabo. V ta namen se poleg skalarnih mer uporabljajo tudi različne grafične metode. Grafične metode, kot so karakteristike delovanja sprejmnika (v nadaljevanju *ROC*, *Receiver Operating Characheristic Curve)* ali preciznost-priklic (*Precision-Recall, PR*) so že dobro znane in uporabljane v krogih, ki se ukvarjajo s strojnim učenjem.

To diplomsko delo preučuje možnosti uporabe alternativnih grafičnih metod, ki bi nam lahko pomagale pri ocenjevanju in primerjanju klasifikatorjev.

V prvem poglavju sta predstavljeni dve izmed klasičnih grafičnih metod in algoritem za računanje točk teh metod, poleg tega pa tudi dva primera alternativnih grafičnih metod. Na koncu poglavja so prikazani načini povprečenja točk, ki zadevajo vse metode in nam pomagajo ugotoviti mero variance.

V drugem poglavju je predstavljen splošen algoritem, ki omogoča definiranje krivulj, ki jih lahko uporabimo za ocenjevanje klasifikatorjev. V tem poglavju so opisane tudi lastnosti grafičnega gradnika v okolju Orange, ki implementira omenjeni algoritem, njegov uporabniški vmesnik in navodila za uporabo grafičnega gradnika.

V tretjem poglavju so prikazani primeri uporabe grafičnega gradnika.

Četrto poglavje je zaključek diplomskega dela in govori o doseženih ciljih in nekaterih pomanjkljivostih grafičnega gradnika.

Ključne besede: strojno učenje, ROC, ocenjevanje klasifikatorjev, klasifikator, vizualizacija

Abstract

In machine learning we are often faced with a task of evaluating and comparing classifiers based on their performance. Alongside scalar measures various graphical methods are being used. Graphical methods such as Receiver Operating Characheristic Curve (ROC) or Precision-Recall (PR) are already well known and used in the machine learning community.

This thesis explores the options of using alternative graphical methods to help us explore and compare classifiers.

The first chapter describes two of most commonly used graphical methods along with the algorithm to produce the set of points for this methods. Later on it depicts examples of two less conventional methods that could be used along side ROC and PR curve.
End of this chapter covers methods that help us estimate the measure of variance.

The second chapter begins with a description of a generic algorithm that can be used to define methods to evaluate and compare classifiers. Afterwards it describes an implementation of this algorithm as a widget in a data mining software called Orange.
It also describes the user interface of the widget and some guidelines how to use it.

The third chapter gives some examples how the widget could be used.

The fourth chapter is the conclusion of the thesis. It consists of assessing if the goals of the thesis were met and pointing out some flaws of the widget.

Key words: data mining, ROC, classifier evaluation, clasification, visualization

Uvod

V strojnem učenju se za ocenjevanje uspešnosti klasifikatorja poleg skalarnih mer (klasifikacijska točnost, informacijski dobitek, Brierjeva mera) uporabljajo tudi različne grafične metode. Med bolj znanimi in uporabljanimi metodami so grafi: krivulja karakteristike delovanja sprejemnika (v nadaljevanju *ROC*, *Receiver Operating Characheristic Curve*), preciznost-priklic (*Precision-Recall*) in krivulja dviga (*Lift Curve*), poleg njih pa obstajajo še številne druge.

Vsaka od teh metod nam na nek način pomaga pri vizualizaciji, razvrščanju in izbiri klasifikatorjev glede na njihovo uspešnost. Katero od metod bomo uporabili, je odvisno od problema, ki ga rešujemo in od tega, katera od njih nam je najbližja.

Med vsemi temi metodami so podobnosti in seveda tudi razlike. Naloga te diplomske naloge je bila poiskati te podobnosti in razlike ter jih posplošiti v enotni algoritem, ki bi pokril vse naštete krivulje in omogočal na preprost način definirati nove.

Praktični del diplomske naloge je bil napisati grafični gradnik (*widget*) v okolju Orange [4], ki bi poleg tega, da bi že vseboval vnaprej definirane standardne krivulje, naštete zgoraj, uporabniku omogočal tudi definiranje lastnih. V okolju Orange že obstajajo gradniki, ki omogočajo risanje nekaterih od zgoraj naštetih krivulj. Prvotna ambicija je bila, da bi te gradnike nadomestili z novim gradnikom, ki bi znal vse in še več, kot že obstoječi. Iz razlogov, ki jih bomo podrobneje opisali kasneje, se namen ni uresničil in tako bo gradnik dodan kot dopolnilo obstoječim.

1. Ocenjevanje klasifikatorjev

1.1 Predstavitev problema

Naj bo C množica primerov. Vsak element iz C se preslika v enega izmed elementov iz množice {p,n}. Temu pravimo, da je primer pozitiven ali negativen oz. pravimo, da pripada pozitivnemu ali negativnemu razredu.

Klasifikator je preslikava, ki vsakemu primeru iz množice C priredi napoved, kateremu razredu ta primer pripada. Za te napovedi bomo uporabljali oznake {Y,N}, kjer Y pomeni, da je klasifikator napovedal, da bo primer pozitiven, N pa pomeni, da je klasifikator napovedal, da bo primer negativen.

Nekateri klasifikatorji dajejo samo diskretno oceno, torej napoved, ki samo napoveduje, kateremu razredu pripada določen primer, drugi klasifikatorji poleg tega dajejo tudi oceno verjetnosti, da primer pripada določenemu razredu. V svojem delu se bomo omejili na slednje. Poleg tega obstajajo postopki, ki omogočajo pridobitev ocen verjetnosti iz klasifikatorjev, ki podajajo samo diskretno oceno [1].

Če imamo podan klasifikator in primer, so možni štirje izidi:

- Če je primer pozitiven in klasifikator to pravilno napove, potem ga štejemo kot resničnega pozitivnega (*true positive, TP*).
- Če je primer pozitiven in ga klasifikator napove kot negativnega, ga štejemo za lažnega negativnega (*false negative, FN*).
- Če je primer negativen in ga klasifikator pravilno napove kot negativnega, ga štejemo kot resničnega negativnega (*true negative, TN*).
- Če je primer negativen in ga klasifikator napove kot pozitivnega, ga štejemo kot lažnega pozitivnega (*false positive, FP*).

		pravi razred	
		p	**n**
napovedani razred	**Y**	Resnični pozitivni (*True Positives, TP*)	Lažni pozitivni (*False Positives, FP*)
	N	Lažni negativni (*False Negatives, FN*)	Resnični negativni (*True Negatives, TN*)

Tabela 1: možni izidi pri napovedih klasifikatorja

Odnos med možnimi izidi najbolje predstavlja tabela 1. Iz nje pa izhajajo tudi nekatere pogosto uporabljane metrike:

$$FPR = \frac{FP}{N} \qquad TPR = \frac{TP}{P} \qquad recall = \frac{TP}{P}$$

$$precision = \frac{TP}{TP + FP} \qquad accuracy = \frac{TP + TN}{P + N}$$

Delež resničnih pozitivnih primerov (*true positive rate, TPR*) klasifikatorja je razmerje med številom vseh pozitivnih primerov, ki jih pravilno napovemo in številom vseh pozitivnih primerov. Za to razmerje se dostikrat uporablja tudi ime priklic (*recall*).

Podobno je delež lažnih pozitivnih primerov (*false positive rate, FPR*) enak razmerju med številom vseh negativnih primerov, ki smo jih napačno napovedali in številom vseh negativnih primerov.

Preciznost (*precision*) je razmerje med vsemi pozitivnimi primeri, ki smo jih pravilno napovemo in številom vseh primerov, za katere smo napovedali, da pripadajo pozitivnemu razredu.

Natančnost (*accuracy, CA*) je razmerje med vsemi pravilno napovedanimi primeri in vsemi primeri.

1.2 Grafi, ki temeljijo na merah TP, FP, TN, FN

Graf ROC je dvodimenzionalen graf, ki prikazuje razmerje med deležem resničnih pozitivnih primerov, ki jih odkrijemo in deležem lažnih pozitivnih primerov. Dobimo ga tako, da na osi x prikazujemo delež lažnih pozitivnih primerov, na osi y pa delež resničnih pozitivnih primerov.

Ker so razmeroma preprosti, poleg tega pa pogosto uporabljani , bomo na njih predstavili osnovne koncepte grafov, ki temeljijo na merah TP, FP, TN, FN.

Diskretni klasifikator daje za vsak primer samo napoved razreda. Tak klasifikator lahko v ROC grafu prikažemo kot eno samo točko.

Za nas so zanimivejši klasifikatorji, ki poleg napovedi razreda, napovedujejo tudi verjetnost, da primer pripada določenemu razredu.Tak klasifikator lahko s pomočjo praga (*threshold*) uporabimo na tak način, da dobimo diskretni klasifikator. Če je napoved verjetnosti pozitivnega razreda, ki jo poda klasifikator, večja ali enaka pragu, ki ga določimo, potem diskretni klasifikator napove Y. Če je verjetnost pod tem pragom, diskretni klasifikator daje napoved N.

Pri različnih pragih dobimo različni delež resničnih in lažnih pozitivnih primerov, TPR in FPR, ki jih lahko povežemo v krivuljo ROC.

1.2.1 Generiranje ROC grafa

Na prvi pogled se zdi najlažje generirati ROC graf tako, da prag spreminjamo od -∞ do +∞ z nekim določenim korakom in pri tem vsakič izračunamo vrednosti deleža resničnih pozitivnih primerov in deleža lažnih pozitivnih primerov.

V tem poglavju bomo opisali način, ki je precej bolj učinkovit kot prej omenjena možnost, poleg tega pa je ta algoritem prilagojen nekaterim posebnostim ROC grafov. Razumevanje principa, po katerem deluje algoritem, olajšuje razumevanje splošnega algoritma, ki je prikazan v poglavju 2.1.

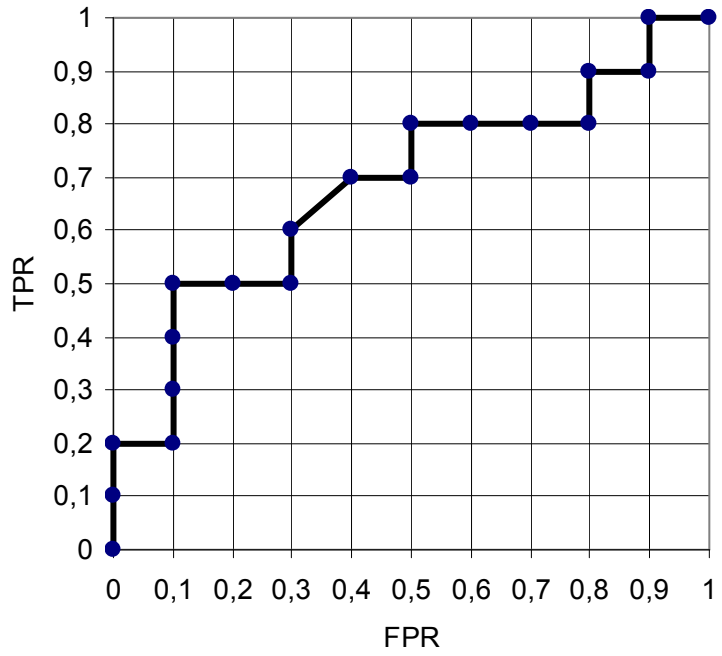
Algoritem izkorišča naslednjo lastnost klasifikatorjev: vsak primer, ki je bil ocenjen kot pozitiven glede na trenutni prag, bo prav tako ocenjen kot pozitiven za vse prage, ki so nižji od trenutnega.Tako lahko primere preprosto uredimo po padajočih ocenah verjetnosti in posodabljamo vrednosti TPR in FPR medtem, ko se pomikamo po urejenem seznamu navzdol.

Preden podrobno definiramo algoritem, naredimo primer, ki bo pokazal njegovo delovanje. V tabeli je dvajset primerov, za katere je podan razred, kateremu v resnici pripadajo in napoved verjetnosti, da bo primer pozitiven, ki jo vrača klasifikator. Imamo 10 pozitivnih in 10 negativnih primerov, kar olajša računanje: P = 10, N = 10.

Zap. št	Razred	Verjetnost	Zap. št.	Razred	Verjetnost
1	p	0.9	11	p	0.505
2	p	0.8	12	n	0.39
3	n	0.7	13	p	0.38
4	p	0.6	14	n	0.37
5	p	0.55	15	n	0.36
6	p	0.54	16	n	0.35
7	n	0.53	17	p	0.34
8	n	0.52	18	n	0.33
9	p	0.51	19	p	0.30
10	n	0.505	20	n	0.1

Tabela 2: primeri in ocene verjetnosti, da bo dani primer pozitiven

Na sliki 1 je ROC graf za primere, podane v tabeli 2. V nadaljevanju si bomo ogledali, kako pridemo do njega.



Slika 1: primer ROC grafa

V tabeli so primeri že urejeni po padajočih ocenah verjetnosti, da primer pripada pozitivnemu razredu.

Če prag postavimo pri ∞, to pomeni, da nobenega primera ne bomo ocenili kot pozitivnega. Zaradi tega ni mogoče, da bi katerega od primerov napačno proglasili za pozitivnega. Vrednost FP je potemtakem enaka 0, prav tako je FPR enak 0.
Ker nobenega primera ne ocenimo pozitivno, je tudi število resničnih pozitivnih enako 0, zato sta vrednosti TP in TPR enaki 0. Tako dobimo točko (0,0) , ki je značilna za vsak ROC graf.

Če prag premaknemo na 0,9, potem naš klasifikator kot pozitivnega oceni samo prvi primer. Ker je ta tudi v resnici pozitiven, se število resničnih pozitivnih poveča za 1:
TP = 0 + 1 TPR = 1 / 10 = 0,1. Število lažnih pozitivnih (*FP*) se ne spremeni, prav tako FPR. Iz tega dobimo točko (0, 0,1).

Ko ta prag postopoma zmanjšujemo, dobimo funkcijo, ki je na zgornji sliki.
Točka (1, 1) na grafu ustreza stanju, ko prag zmanjšamo na 0, to pomeni, da za vse primere napovemo, da bodo pozitivni. V tem primeru je vrednost resničnih pozitivnih enaka 10, prav tako je vrednost lažnih negativnih enaka 10. Vrednosti TPR in FPR sta zato enaki 1. Tako dobimo točko (1,1).
Omeniti je potrebno še prehod iz točke (0,3 , 0,6) v točko (0,4 , 0,7). Povsod drugje je funkcija stopničasta, tu pa se zgodi drugačen prehod. Razlog je v tem, da imata točki 10 in 11 enako napoved verjetnosti. Ko prag zmanjšamo na to verjetnost, moramo obe točki obravnavati naenkrat. Primer 10 je v resnici negativen, primer 11 pa pozitiven, zaradi tega se vrednosti TP in FP obe povečata za ena.

Točk ne smemo obravnavati posebej zato, ker bi bila v tem primeru oblika grafa odvisna od tega, katero od teh točk bi obravnavali prvo.

Da bi si bralec lažje predstavljal pomen in uporabo ROC krivulj, podajmo še primer uporabe. Recimo, da zgornji podatki pripadajo klasifikatorju, ki zdravniku pomaga napovedati, ali bo obravnavani pacient zbolel, ali ne.

Če napove, da bo pacient zbolel, je potrebno bolnika poslati še na dodatne boleče in drage preiskave. Če napove, da je pacient zdrav, teh preiskav ni, obstaja pa tveganje (čeprav mogoče majhno), da je napoved napačna.

Os x prikazuje FPR, oziroma v našem primeru delež zdravih, ki jih bomo po nepotrebnem dodatno preiskovali. Os y prikazuje TPR, oz. v našem primeru delež bolnih, ki jih bomo v resnici odkrili. Če zdaj prag postavimo pri ∞, ne bomo nobenemu pacientu napovedali, da bo zbolel. Sicer ne bo nobene pritožbe zaradi nepotrebnih preiskav, bodo pa pritožbe vseh, ki bodo zboleli, to ustreza točki (0,0) na grafu. Če gremo v drugo skrajnost, lahko ta prag postavimo na 0 in s tem vse proglasimo za bolne. Sicer bomo odkrili vse zares bolne, vendar bo zaradi tega število po nepotrebnem pregledanih naraslo čez vse meje.

V praksi moramo ta prag določiti v sodelovanju z uporabnikom s potrebnim ekspertnim znanjem, v našem primeru z zdravnikom.

1.2.2 Definicija ROC algoritma

Definicijo bomo podali v psevdokodi, ki je povzeta po Fawcetu[1].

```
Vhodi :
   L          množica testnih primerov;
    f(i)      ocena verjetnosti, da je primer pozitiven ;
   P          število pozitivnih primerov
   N          število negativnih primerov
Izhodi:
   R          seznam ROC točk urejen po naraščajočem FPR
Zahteva:
   P > 0 in N > 0

L_sorted  ←  L urejen po padajočih ocenah verjetnosti
FP ← TP ← 0
R ← []
f_prev ← -∞
i ← 1
while i ≤ |Lsorted| do
   if f(i) != fprev then
      push ( FP/ N, TP /P)  onto R
      fprev ← f(i)
   end if
   if L_sorted [i] is a positive example then
      TP ← TP +1
   else
      FP ← FP +1
   end if
   i ← i + 1
end while
push ( FP/ N, TP /P)  onto R
end
```

1.2.3 Graf preciznost-priklic

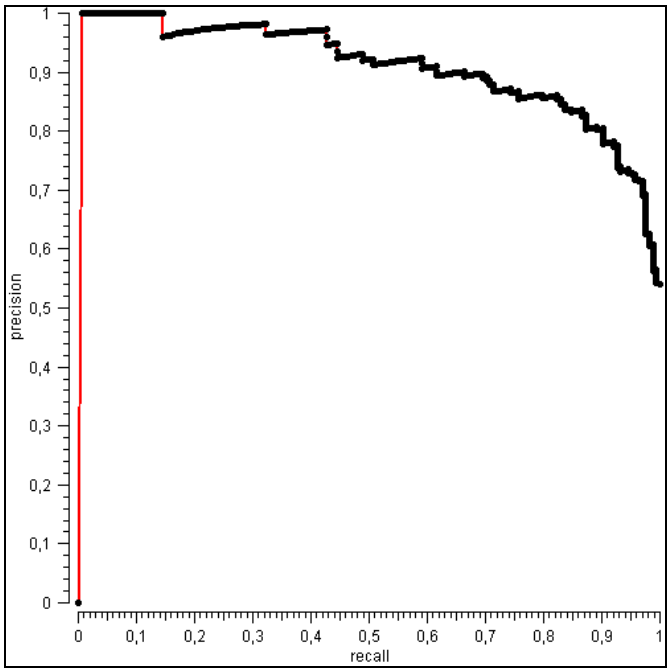
Graf preciznost-priklic (*precision–recall,PR*) je dvodimenzionalen graf, kjer na osi y prikazujemo natančnost (*precison*), na x osi pa priklic (*recall*), ki je enak deležu resničnih pozitivnih primerov.

Krivulje preciznost-priklic se pogosto uporabljajo pri sistemih za poizvedovanje (*Information Retrieval*) [7], le te se ukvarjajo s pridobivanjem informacij iz besedil.
Nekateri raziskovalci jih priporočajo kot alternativo ROC grafom, še posebej v primerih, ko je porazdelitev razredov zelo neenakomerna. [3].

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{P}$$

Algoritem, ki omogoča, da dobimo točke tega grafa, je popolnoma enak algoritmu za ROC, s to razliko, da namesto FPR računamo preciznost.

Preciznost lahko interpretiramo kot delež resničnih pozitivnih med vsemi primeri, ki jih proglasimo za pozitivne, priklic pa kot delež vseh resničnih pozitivnih primerov, ki jih odkrijemo.Primer grafa preciznost-priklic je na sliki 2.



Slika 2: primer grafa precision - recall

Iz grafa na sliki 2 lahko vidimo, da je doseganje visoke vrednosti priklica običajno povezano z manjšo preciznostjo.

1.3 Grafi, ki ne temeljijo na merah TP, FP, TN, FN

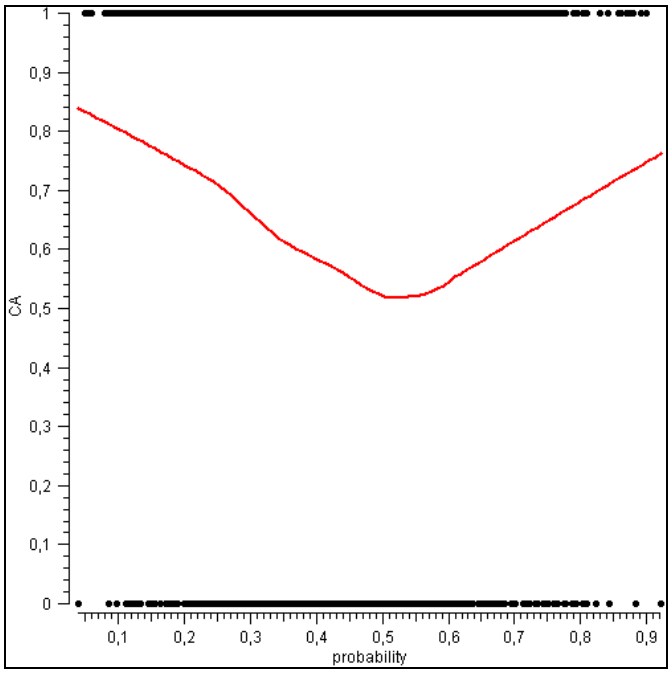
Grafi, ki temeljijo na merah TP,FP,TN,FN so pogosto uporabljani pri ocenjevanju uspešnosti klasifikatorja, grafa, ki smo ju omenili v prejšnjem poglavju, pa sta le dve izmed najbolj uporabljanih metod.

Poleg omenjenih mer, lahko uspešnost ocenjujemo tudi na druge načine. V tem poglavju bomo pokazali dva primera grafov, ki nista sestavljena iz zgoraj omenjenih mer.

1.3.1 Graf klasifikacijske točnosti primera glede na napovedano verjetnost

Graf klasifikacijske točnosti primera (*example clasification accuracy, eCA*) glede na napovedano verjetnost, prikazuje zvezo med verjetnostjo, ki jo napove klasifikator in točnostjo diskretne napovedi istega klasifikatorja. Način, kako ta graf dobimo, se razlikuje od tistega, ki smo ga opisali v prejšnjem poglavju.

Algoritem mora iti po vseh primerih in si pri vsakemu zapomniti napovedano verjetnost in ujemanje napovedanega razreda z dejanskim. Če imamo ujemanje, temu priredimo vrednost 1, drugače je ta vrednost 0. Te točke lahko vidimo na grafu s slike 3 zgoraj in spodaj. Ker nam točke same po sebi ne povejo veliko, uporabimo lokalno regresijo. Metodo bomo podrobneje opisali v poglavju 1.4.5, zaenkrat povejmo samo to, da nam omogoča med danimi točkami potegniti krivuljo, ki se jim kar najbolj prilagaja.



Slika 3: primer grafa klasifikacijske točnosti glede na verjetnost, ki jo poda klasifikator

Iz slike 3 lahko razberemo, da se dani diskretni klasifikator najbolje odreže, ko je napovedana verjetnost visoka ali nizka. V primeru, ko je verjetnost blizu 50%, postanejo tudi napovedi precej bolj negotove. Vidimo lahko še to, da klasifikator malenkost bolje napoveduje, ko je verjetnost nizka , kot takrat, ko je le ta visoka. Omenimo še, da tovrstne krivulje niso ravno pogosto uporabljane, opisujemo pa jih kot primer krivulje, ki jo je na podlagi dela, opravljenega v okviru te diplomske naloge, preprosto definirati in narisati.

1.3.2 Graf dobička glede na prag

Pri večini problemov, s katerimi se srečujemo, lahko na nek način opišemo koristi, ki jih imamo ob pravilnih napovedih in škodo ob nepravilnih. Dostikrat nas pravzaprav najbolj zanima dobiček in kako ga čimbolj povečati.

Za ponazoritev lahko vzamemo problem, s katerim smo se študentje srečali pri predmetu umetna inteligenca in je na nek način vplival na celotno diplomsko nalogo.

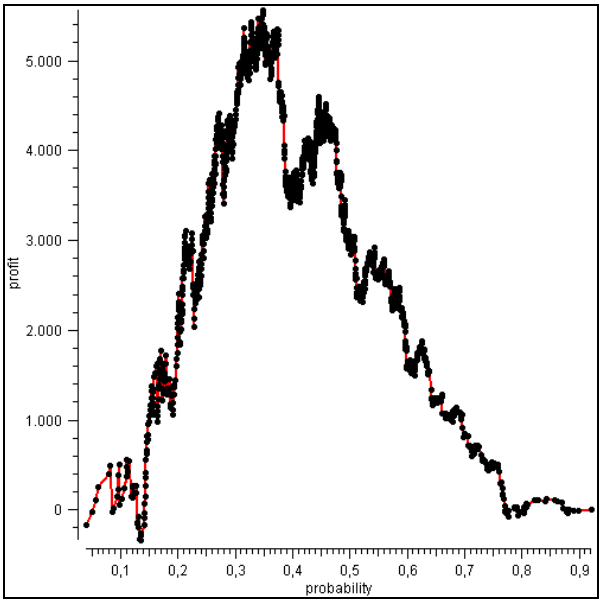
Imamo klasifikator, ki vrača verjetnost, da bo posojilojemalec vrnil posojilo, ki ga je najel pri banki. Banka za posojeni znesek zahteva obresti po formuli:

$$obresti = \frac{posojeni\ \ znesek^{\,2}}{30}$$

V primeru, da stranka posojila ne vrne, denarja ni mogoče izterjati, tako da banka posojeni znesek izgubi. Banka bo denar posodila strankam, za katere je verjetnost, da bodo posojilo vrnile čim večja. Zanima nas, kje postaviti mejo, da bo bankin dobiček čim večji.
Najprej opazimo, da lahko na podlagi napovedane verjetnosti vračanja kredita (če je le-ta seveda dovolj točno ocenjena) izračunamo pričakovani dobiček pri posamezni stranki. Ta znaša

$d = \frac{p * želeni\ znesek^{\,2}}{30} - (1-p) * želeni\ znesek$, kjer je p napovedana verjetnost vračanja kredita.

Da bi narisali krivuljo, ki kaže odvisnost med pragi in pričakovanimi dobički pri posameznih pragih, moramo stranke razvrstiti po padajoči napovedani verjetnosti vračanja kredita. Če mejo za odobritev postavimo pri ∞, bo dobiček enak 0. Nato podobno kot pri krivulji ROC postopno nižamo prag. Za vsako stranko iz urejenega seznama izračunamo dobiček po gornji formuli in ga dodamo k dobičku, ki bi ga dobili, če bi prag nastavili na verjetnost vračanja pri tej stranki. Tako dobimo funkcijo dobička, ki je na sliki 4. Pri krivulji na sliki, ki se nanaša na podatke in klasifikator, ki smo ga uporabljali pri omenjeni seminarski nalogi, razberemo, da je dobiček največji, ko prag odobritve postavimo na približno 35 %.



Slika 4: graf dobička glede na prag odobritve

1.4 Načini povprečenja

Pri ocenjevanju klasifikatorjev pogosto uporabljamo različna vzorčenja podatkov, kot je npr. prečno preverjanje. Pri tem množico primerov T razdelimo na podmnožice T_1, T_2, ...T_n. Nato n-krat poženemo algoritem učenja, vsakič nad drugo množico $T\backslash T_i$, in ga testiramo s primeri iz množice T_i. Tako ne dobimo ene same krivulje, temveč n (praviloma različnih) krivulj. Te lahko prikažemo na različne načine.

1.4.1 Prikazovanje krivulj brez povprečenja

Pri tej metodi preprosto narišemo vse krivulje, ki jih dobimo iz testnih množic T_i. Ko narišemo vse krivulje, lahko vidimo, kakšen je maksimum in minimum, vidimo pa tudi kakšna so odstopanja med posameznimi testnimi množicami. Če so ta odstopanja velika, lahko sklepamo, da metoda ni robustna.

1.4.2 Zlitje brez povprečenja

Vse testne množice T_i lahko zlijemo v eno samo veliko testno množico T, ki jo nato obravnavamo, kot da je bila dobljena z enim samim učenjem. Iz takšne krivulje ni mogoče razbrati odstopanj in s tem sklepati o robustnosti metode. Metoda je nekoliko problematična zaradi dejstva, da so bile verjetnosti, ki so prirejene testnim primerom, dobljene z različnimi klasifikatorji (saj smo postopek učenja ponovili n-krat), zato med seboj niso nujno primerljive.

1.4.3 Navpično povprečenje

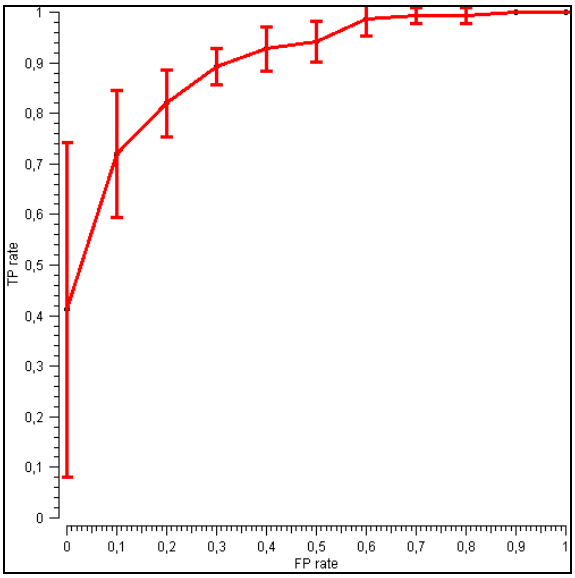
Pri navpičnem povprečenju (*vertical averaging*) si na osi x izberemo m vrednosti in nato za te fiksne vrednosti x_i vzorčimo vrednosti $y_{i,j}$ za posamezno krivuljo (i : 1 ... m; j: 1 .... n kjer je n število krivulj, ki jih želimo povprečiti). V primeru, ko ne poznamo vrednosti $y_{i,j}$, to vrednost pridobimo s pomočjo interpolacije. Pri tem uporabimo prvo točko krivulje, ki ima x koordinato manjšo od x_i in prvo točko, ki ima x koordinato večjo od x_i.

Za vsako točko x_i izračunamo povprečno vrednost pripadajočih $y_{i,j}$ in tako dobimo točko navpično povprečene krivulje (x_i,y_i).

$$y_i = \frac{\sum_{j=1}^{n} y_{i,j}}{n}$$

Poleg povprečij se prikazuje tudi standardni odklon. Le ta se za posamezni y_i izračuna po formuli:

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{n} (y_{i,j} - y_i)^2}{n-1}}$$



Slika 5: primer vertikalnega povprečenja

Podrobnejši opis navpičnega povprečenja skupaj s psevdokodo algoritma za računanje točk se nahaja v [1].

1.4.4 Povprečenje glede na prag

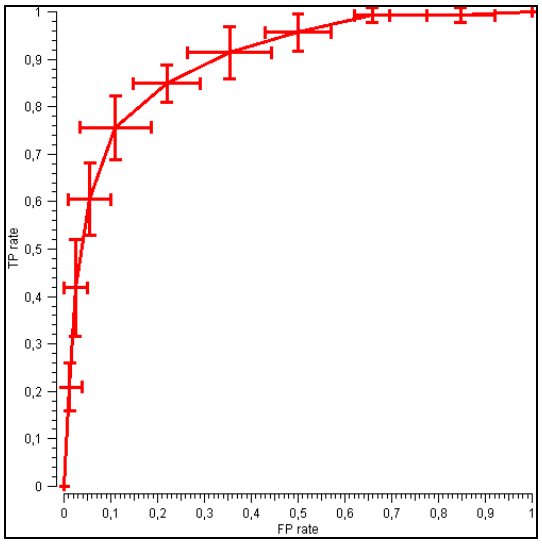
Prednost navpičnega povprečenja (*Treshold averaging*) je v njegovi preprostosti, saj imamo vedno določen x, za katerega računamo povprečje vrednosti y-a. Holte [5] je opozoril na to, da spremenljivka x ni pod neposrednim nadzorom uporabnika, zato v resnici prihaja do variance, tako pri x, kot pri y.

Povprečenje glede na prag ne računa povprečij glede na vrednost koordinate x ali y, ampak glede na vrednost spremenljivke, po kateri smo uredili podatke, torej v krivuljah, o kakršnih smo pisali doslej, po verjetnosti. Zaradi tega potrebuje metoda za vsako točko poleg obeh koordinat še vrednost spremenljivke, po kateri smo podatke uredili.

Metoda mora najprej sestaviti vzorec pragov (treshold), po katerih bomo izvajali povprečenje. Najpreprosteje to storimo tako, da uporabnik določi število pragov, i. Celoten razpon možnih pragov razbijemo na i intervalov in nato za vsak tak prag poiščemo na vsaki izmed krivulj točko, ki mu je najbližja.

Če je n enako številu krivulj, ki jih želimo povprečiti, tako za vsak prag dobimo n točk. Tako dobljene točke nato povprečimo tako po x kot po y kordinati, prav tako izračunamo standardni odklon po obeh oseh. Povprečje po osi x oziroma y ter standardni odklon izračunamo po naslednjih enačbah:

$$x_i = \frac{\sum_{j=1}^{n} x_{i,j}}{n} \qquad y_i = \frac{\sum_{j=1}^{n} y_{i,j}}{n} \qquad \sigma x_i = \sqrt{\frac{\sum_{j=1}^{n} (x_{i,j} - x_i)^2}{n-1}} \qquad \sigma y_i = \sqrt{\frac{\sum_{j=1}^{n} (y_{i,j} - y_i)^2}{n-1}}$$



Slika 6: primer povprečenja glede na prag

1.4.5 Glajenje

LOESS (*locally weighted scatterplot smoothing*) [2] je različica regresije, ki se uporablja predvsem za glajenje krivulj. Denimo, da imamo podano množico točk (x_i, y_i), prek katerih želimo povleči zglajeno krivuljo. To storimo tako, da izberemo vzorec koordinat x (te imajo lahko vrednosti, ki se pojavljajo v podatkih, lahko pa gre za, na primer, določeno število enakomerno porazdeljenih vrednosti iz obsega x, ki se pojavlja v podatkih). Za vsak x iz podane množice točk vzamemo določen delež točk (tipično 30% ali 50%) z najbolj podobnimi koordinatami x_i. Za točke v tem oknu izračunamo običajno linearno regresijo, s tem, da vsako točko utežimo glede na razliko $|x-x_i|$, pri čemer uporabimo določeno jedrno funkcijo. Najobičajnejša jedrna funkcija je

$$e^{\frac{(x - x_i)^2}{\sigma}}.$$

Nato, končno, uporabimo linearno regresijo kot model, s katerim ocenimo y pri danem x. Ko to ponovimo za vse x iz vzorca koordinat, dobimo dokaj gladko krivuljo ("gladkost" je odvisna od gostote izbranega vzorca).

V diplomski nalogi smo glajenje z LOESS potrebovali predvsem v primerih, na kakršnega smo naleteli v razdelku 1.3.1. Tam smo imeli pri vsakem posameznem pragu x točko pri y=0 ali pri y=1. S pomočjo LOESS lahko povlečemo krivuljo, ki se spušča in dviga, glede na to, ali imamo v danem področju več točk pri y=0 ali pri y=1.

2. Grafični gradnik za risanje splošnih krivulj karakteristik klasifikatorjev

V prvem poglavju smo opisali nekatere izmed metod za ocenjevanje uspešnosti klasifikatorjev. Praktični del diplomske naloge je realiziran kot grafični gradnik v okolju Orange. To poglavje opisuje grafični gradnik in njegovo delovanje.

2.1 Algoritem

Algoritem za generiranje točk je nastajal postopoma in se prilagajal tipu grafov, ki smo jih želel risati. Algoritem potrebuje naslednje podatke, ki povedo način računanja točk:

a) **Urejenost**: podatke lahko uredimo po kateremkoli atributu; naraščujoče ali padajoče. Po želji pa lahko tudi ohranimo urejenost vhodnih podatkov (možnost »No sorting«).

b) **Združevanje podatkov**, ki imajo enako vrednost atributa. Združevanje je možno samo po atributu, po kateremu smo podatke uredili. Definirali smo tri možnosti: »None« , »Last value«, »Average value« .

Kadar uporabljamo možnost »None« , združevanja ni. Algoritem gre po vseh primerih in za vsakega izračuna vrednost podane formule. Vsakič, ko se to zgodi, se na graf doda nova točka.Ta tip je, recimo, uporabljen pri primeru v poglavju 1.3.1.

Možnost »Last value« smo definirali predvsem zaradi ROC algoritmov. Kot smo videli, je potrebno pri njih primere, ki imajo enake ocene verjetnosti, obravnavati naenkrat. Algoritem za primere z isto vrednostjo atributa doda samo vrednost formule, ki jo je izračunal pri zadnjem primeru z isto vrednostjo atributa.

Možnost »Average value« doda za vse primere z isto vrednostjo izbranega atributa samo eno točko. Posamezni koordinati točke sta povprečji vrednosti formule koordinate za primere z isto vrednostjo atributa.

c) **Količini na koordinatnih oseh**: za risanje grafa je potrebno določiti količini, ki ju bosta predstavljali koordinatni osi. To storimo s formulama, ki jo vpiše uporabnik, in sicer eno za vrednost koordinate x , drugo za vrednost koordinate y. Formule so lahko preproste, npr. vrednost nekega atributa, ali pa tudi precej kompleksni izrazi. Več o pisanju formul pa bomo napisali v razdelku 2.1.2.

Uporabnik poleg tega lahko izbira med štirimi načini povprečenja in izbira testne množice, iz katerih algoritem računa točke, vendar to na sam algoritem nima nobenega vpliva.

2.1.1 Psevdokoda algoritma

Algoritem bomo za boljše razumevanje podali še s psevdokodo. V praktični implementaciji je še nekaj tehničnih trikov, s katerimi pospešimo izvajanje kode ali pa se izognemo nekaterim možnim napakam zaradi posebnosti v podatkih.

```
Vhodi:
   L                 množica testnih primerov in napovedanih verjetnosti ciljnega
                     razreda;
   sortBy            ime atributa po katerem sortiramo;
   sortDirection     smer sortiranja, {ascending,descending};
   mergeType         način združevanja  {None,lastValue,averageValue};
   formulaX          formula za izračun količine, ki se nanaša na koordinato x;
   formulaY          formula za izračun količine, ki se nanaša na koordinato y;
   targetClass       razred, ki ga obravnavamo kot pozitivnega
Izhodi:
   R                 seznam s koordinatami točk krivulje

ROCvariables = ["TP","FP","TN","FN","CA"]

if sortBy  != »No sort« then
   if sortDirecton = »ascending« then
      L_sorted   ←  L  sorted ascending by attribute sortBy
   elsif sortDirecton = »descending« then
      L_sorted   ←  L  sorted descending by attribute sortBy
   end if
else
   L_sorted   ←  L
end if

#ROC spremenljivke se vedno računajo, v primeru da podatki niso urejeni po verjetnosti so
napačne in se zato ne smejo uporabljati
if sortBy != »probability« and ( formulaX.containsOneOf(ROCvariables)   or
formulaY.containsOneOf(ROCvariables) ) then
  exit -1
end if

previousSortValue  ← None
R ← []

P = total("trueClass = targetClass",targetClass)
N = total("trueClass != targetClass",targetClass)

if sortDirection = "descending" then
   TP ← FP ← 0   TN ← N  FN ← P
else
   TN ← FN ← 0   TP ← P  FP ← N
end if

eP ← eN ← eCA ← eTP ← eFP ← eTN ← eFN ← None #še niso def.
xAvg ← xVal ← evaluate(formulaX)
yAvg ← yVal ← evaluate(formulaY)

if xVal is not None and yVal is not None then
   n ← 1
   xSum ← xVal
   ySum ← yVal
else
   xSum ← ySum ← n ← 0
end if

for example in L_sorted :
   eP ← float(example.actualClass = targetClass)
   eN ← float(example.actualClass != targetClass)
   eCA ← float(example.actualClass = example.predictedClass)
   eTP ← float(example.predictedClass=targetClass=example.actualClass)
   eFP ← float(example.predictedClass=targetClass!=example.actualClass)
   eTN← float(example.predictedClass!=targetClass!=example.actualClass)
   eFN ← float(example.predictedClass!=targetClass=example.actualClass)

   if sortDirection = "descending" then
      TP ← TP + eP
      FP ← FP + eN
      TN ← TN - eN
      FN ← FN - eP
      PP ← PP + 1
      NP ← NP - 1
   else
      TP ← TP - eP
      FP ← FP - eN
      TN ← TN + eN
      FN ← FN + eP
      PP ← PP - 1
      NP ← NP + 1
   end if
```

```
   sortValue  ← example.sortBy

   if typeOfMerge is None then
      if (xVal is not None and yVal is not None) then
         push ( xVal, yVal)  onto R
      end if

   else
      if sortValue != previousSortValue then
         if typeOfMerge = 'lastValue' then
            if (xVal is not None and yVal is not None) then
               push ( xVal, yVal)  onto R
            end if
         elsif typeOfMerge = 'averageValue' then
            if (xAvg is not None and yAvg is not None) then
               push ( xAvg, yAvg)  onto R)
               xSum ←ySum ← n ← 0
               yAvg ← xAvg ← None
             end if
         end if
      end if
   end if


   xVal ← evaluate(formulaX)
   yVal ← evaluate(formulaY)

   if xVal is not None and yVal is not None then
      n ← n + 1
      xSum  ← xSum + xVal
      ySum  ← ySum + yVal
      xAvg  ← xSum / n
      yAvg  ← ySum / n
   end if

   previousSortValue ← sortValue

end for

#dodamo se zadnjo tocko
if Merge = 'averageValue' then
   xVal ← xAvg
   yVal ← yAvg
end if
push ( xVal, yVal)  onto R)
return R
```

#funkcija total
#total(formula,targetClass)
Vhodi :

formula	formula, katere vsoto računamo;
targetClass	razred ki, ga obravnavamo kot pozitivnega

Izhodi:

sum	vsota podane formule po vseh primerih

```
sum ← 0
for example in examples
   sum ← sum + evaluate(formula)
return sum
```

2.1.2 Pisanje formul

Glavna značilnost grafičnega gradnika je možnost, da uporabnik piše svoje formule. Uporabnik ima pri pisanju izrazov na voljo naslednje možnosti:

a) **Atributi primera in spremenljivke, ki pripadajo primeru:**

Uporabnik lahko uporabi vsak atribut, ki pripada primeru. Poleg tega ima vsak primer dodane štiri atribute, ki so posledica ocenjevanja primerov:

probability
V spremenljivki je shranjena verjetnost, da primer pripada ciljnemu razredu. Verjetnost je podana kot decimalno število z vrednostjo med 1 in 0.

iteration
V spremenljivki je shranjen podatek, kateri testni množici pripada primer. Zavzame lahko vrednosti od 0 do n -1, kjer je n število testnih množic.

actualClass
Spremenljivka hrani podatek, kateremu razredu pripada primer. Zavzame lahko vse vrednosti razredov, katerim lahko pripada primer.

predictedClass
Spremenljivka hrani napoved diskretnega klasifikatorja, kateremu razredu pripada primer. Zavzame lahko vse vrednosti razredov, katerim lahko pripada primer.

Vedno moramo definirati razred, ki je pozitiven, vsi ostali pa so obravnavani kot negativni. V primeru posojil iz poglavja 1.3.1, bi za ciljni razred lahko določili tiste primere, ki posojilo vrnejo. Če imamo problem z več kot dvema razredoma, za pozitivnega lahko vzamemo samo enega izmed njih. Če primer pripada kateremukoli drugemu razredu, je negativen.

Iz tako definiranih atributov lahko izpeljemo še naslednje diskretne spremenljivke, ki pripadajo ocenjenemu primeru. Vrednosti teh spremenljivk so 0 ali 1.

eP (*example positive*)
Vrednost te spremenljivke je enaka 1, kadar je razred primera enak ciljnemu, v nasprotnem primeru je enaka 0 (trueClass == targetClass).

eN (*example negative*)
Vrednost te spremenljivke je enaka 1, kadar se razred primera in ciljni razred razlikujeta. V nasprotnem primeru je enaka 0 (trueClass!= targetClass).

eCA (*example accuracy*)
Vrednost te spremenljivke je enaka 1, kadar se napoved razreda, kot jo poda diskretni klasifikator, ujema z razredom primera. V nasprotnem primeru je vrednost spremenljivke enaka 0 (trueClass == predictedClass).

eTP (*example TP*)
Vrednost spremenljivke je enaka 1, kadar se napoved diskretnega klasifikatorja ujema z razredom primera in je obenem enaka ciljnemu razredu (predictedClass == targetClass == actualClass). V nasprotnem primeru je njena vrednost enaka 0.

eTN (*example TN*)
Vrednost spremenljivke je enaka 1, kadar je razred primera različen od ciljnega razreda in diskretni klasifikator to pravilno napove. V nasprotnem je vrednost spremenljivke enaka 0 (predictedClass != targetClass != actualClass).

eFP (*example FP*)
Vrednost spremenljivke je enaka 1, kadar je napoved diskretnega klasifikatorja enaka ciljnemu razredu, razred primera pa se od njiju razlikuje. V nasprotnem je njena vrednost enaka 0 (predictedClass == targetClass != actualClass).

eFN (*example FN*)
Vrednost spremenljivke je enaka 1, kadar je razred primera enak ciljnemu razredu, napoved diskretnega klasifikatorja pa se od njiju razlikuje. V nasprotnem primeru je njena vrednost enaka 0 (predictedClass != targetClass == actualClass)

b) **Spremenljivke, ki se nanašajo na celotno populacijo:**

Druga skupina spremenljivk se nanaša na celotno množico, ki jo ocenjujemo. To so spremenljivke:

P (*positives*)
Vrednost te spremenljivke je enaka številu vseh pozitivnih primerov v testni podmnožici. Zapišemo jo lahko tudi kot ∑ eP.

N (*negatives*)
Vrednost te spremenljivke je enaka številu vseh negativnih primerov v testni podmnožici. Zapišemo jo lahko tudi kot ∑ eN.

NN
Vrednost te spremenljivke je enaka številu vseh primerov v testni podmnožici.

c) **Spremenljivke, vezane na prag:**

Tretja skupina spremenljivk ni neposredno povezana s posameznim primerom, ampak je vezana na postavitev praga. To so naslednje spremenljivke:

TP (true positives)
Vrednost spremenljivke je enaka številu resničnih pozitivnih primerov pri danem pragu.

TN (true negatives)
Vrednost spremenljivke je enaka številu resničnih negativnih primerov pri danem pragu.

FP (false positives)
Vrednost spremenljivke je enaka številu lažnih pozitivnih primerov pri danem pragu.

FN (false negatives)
Vrednost spremenljivke je enaka številu lažnih negativnih primerov pri danem pragu.

PP
Vrednost spremenljivke je enaka številu vseh primerov, ki jih pri danem pragu proglasimo za pozitivne.

NP
Vrednost spremenljivke je enaka številu vseh primerov, ki jih pri danem pragu proglasimo za negativne.

Iz teh spremenljivk izhajajo še naslednje spremenljivke, ki so prav tako vezane na prag. Čeprav jih je mogoče enostavno izračunati iz zgoraj navedenih, so definirane zaradi lažjega in bolj preglednega pisanja formul:

- `acc = CA = (TP+ TN) / NN`
- `recall = sensitivity = TPR = TP / P`
- `FPR = FP / N`
- `specificity = 1 - FPR`
- `precision = PPV = TP / PP`
- `NPV = TN / NP`
- `FDR = FP / PP`

Opozoriti je treba, da so vse formule iz skupine c smiselne samo v primeru, ko so podatki urejeni po napovedi verjetnosti. Če so podatki urejeni kakorkoli drugače, bodo formule, ki vsebujejo kateregakoli izmed zgornjih izrazov, neveljavne.

d) **Funkcije**

Funkcija cumm:

Pogosto nas ne zanima samo trenutna vrednost atributa, ampak želimo vedeti tudi vsoto formule. Za te primere je definirana funkcija cumm(formula). Vrednost te funkcije je enaka vsoti vrednosti podane formule po vseh do sedaj obravnavanih primerih.

Kot primer se lahko spomnimo funkcije iz poglavja 1.3.2. Če primere razvrstimo po padajoči napovedi verjetnosti vračanja kredita, lahko funkcijo dobička s pomočjo funkcije cumm definiramo kot: `cumm( amount_requested **2/ 30 if eP else - amount_requested)`, kjer je amount_requested atribut stranke, ki pove višino zaprošenega kredita.

Če kot formulo za koordinato x, uporabimo probability, izberemo urejanje podatkov po padajočih napovedih verjetnosti, brez združevanja in z zlitjem dobimo graf iz poglavja 1.3.2.

Funkcija total:

Kadar nas zanima vsota ne samo po dosedaj obravnanih primerih, ampak po vseh primerih, lahko uporabimo funkcijo total(formula). Njena vrednost je enaka vsoti vrednosti formule po vseh primerih. Tako recimo lahko dobimo število vseh pozitivnih primerov s formulo total(actualClass == targetClass). Vrednost funkcije total je enaka pri vseh pragih.

Ostale funkcije

Uporabnik lahko uporablja vse funkcije iz pythonovega modula math, uporablja pa lahko tudi pogojne stavke, na način, ki smo ga prikazali na primeru v točki d.

2.2 Uporabniški pogled

Zdaj, ko smo definirali osnovne lastnosti grafičnega gradnika, je čas, da si pogledamo uporabniški vmesnik gradnika. Uporabnik preko njega nastavlja vse parametre algoritma, ki smo jih opisali do sedaj.

Ker grafični gradnik deluje v okolju Orange, najprej pokažimo način njegove uporabe z drugimi komponentami tega okolja.

Na sliki 7 vidimo preprost način uporabe grafičnega gradnika.

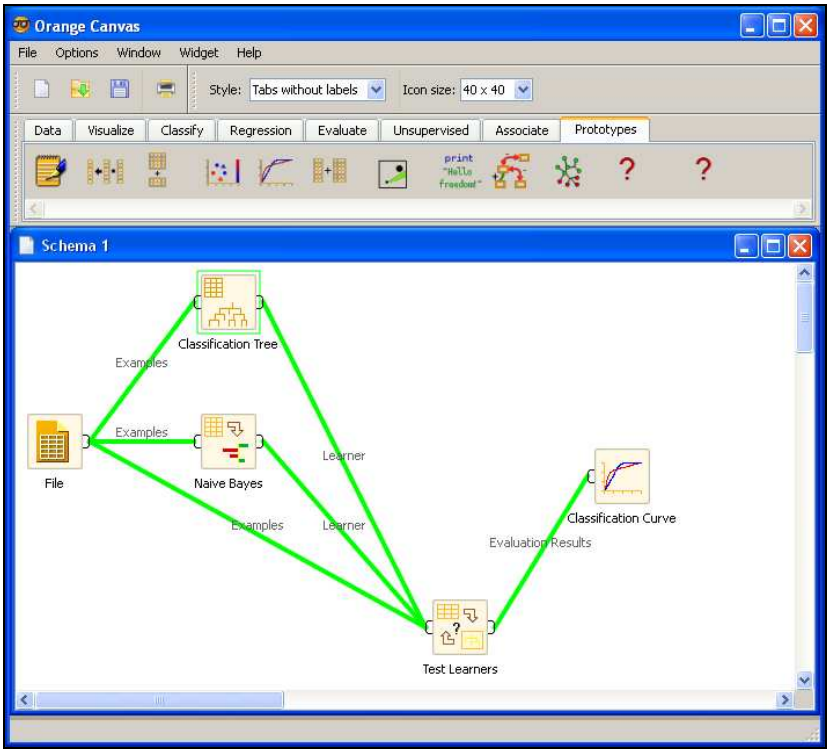
Na levi vidimo grafični gradnik »File«. Z njim naložimo podatke, ki jih želimo obravnavati.

Izhod grafičnega gradnika »File« je vhod za tri druge gradnike. Gradnika »Naive Bayes« in »Classification Tree« sprejmeta podatke iz gradnika »File« in na njihovi osnovi zgradita vsak svoj klasifikator.

Grafični gradnik »Naive bayes« zgradi klasifikator, ki deluje kot naivni bayesov klasifikator, gradnik »Classification Tree«, pa zgradi klasifikator, ki se odloča na podlagi drevesa, vendar poleg diskretne napovedi napoveduje še verjetnost.

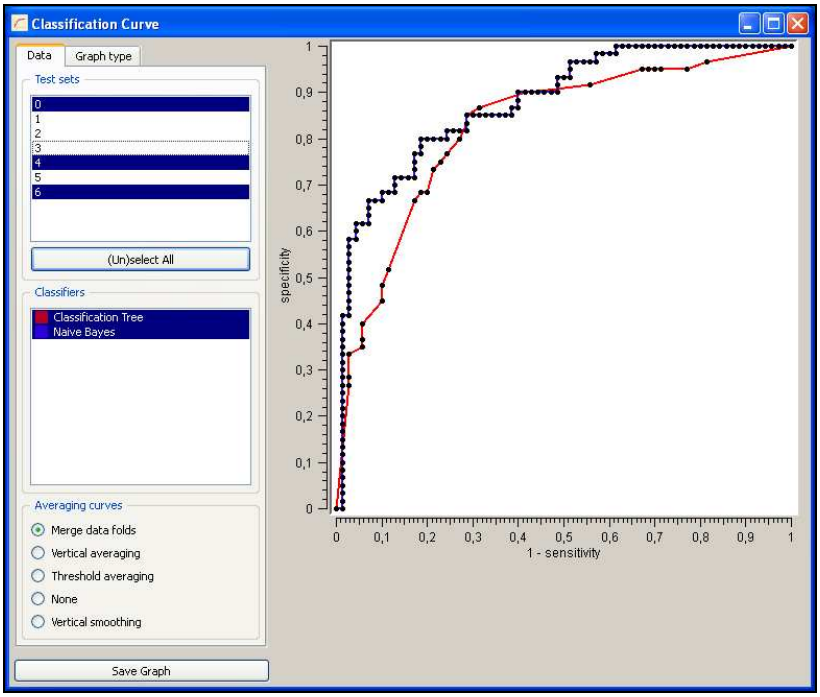
Grafični gradnik »Test Learners« kot vhod dobi podatke iz gradnika »File« in oba klasifikatorja. Gradnik klasifikatorja oceni primer za primerom in kot izhod daje tabelo ocenjenih primerov.

To tabelo kot vhod dobi grafični gradnik »Classification Curve«. Iz nje dobi vse podatke, ki jih potrebuje za risanje grafov.



Slika 7: primer uporabe grafičnega gradnika

Zdaj pa si podrobneje poglejmo sam grafični gradnik »Classification Curve«, ki je prikazan na sliki 8.



Slika 8: grafični gradnik »Classification Curve«, zavihek »Data«

Na desni strani je slika grafa, ki ga želi risati uporabnik, na levi pa so kontrole, ki so razporejene v dva zavihka.

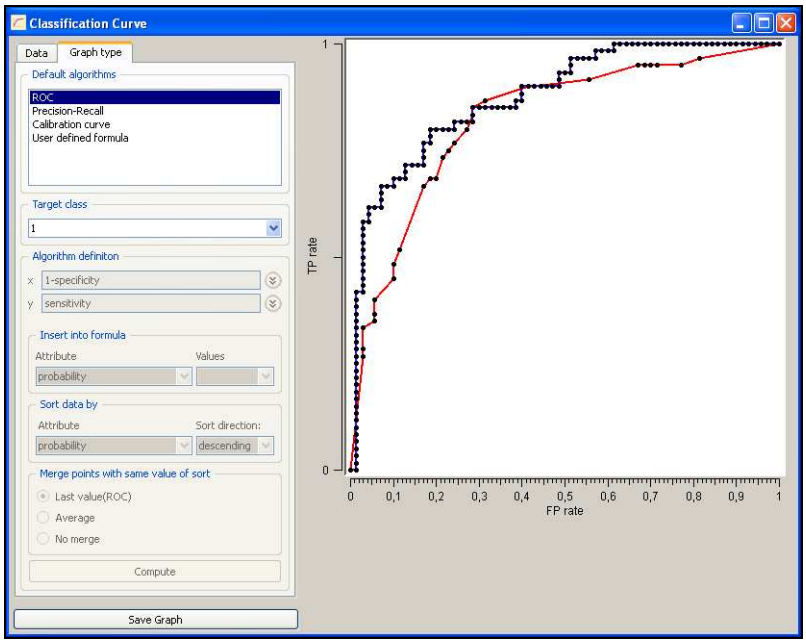
Izven zavihka je gumb »Save Graph«, s katerim graf lahko shranimo.

V zavihku »Data«, ki ga prikazuje slika 8, se nahajajo naslednji elementi:

- »Test sets«; s to kontrolo določamo, katere izmed testnih množic bodo upoštevane pri risanju grafov. Izbrane testne množice so obarvane temno. Temu kontrolnemu elementu pripada še gumb »(Un)select all«, ki pomaga pri hitrejšem izboru vseh kontrolnih množic

- S kontrolo »Classifiers« izbiramo klasifikatorje, katerih krivulje bomo risali. Na sliki sta izbrana oba klasifikatorja.

- »Averaging curves«; s to kontrolo določamo način prikaza krivulje. Izbiramo lahko med naslednjimi možnostmi:
  - »Merge data folds« ustreza preprostemu zlitju.
  - »Vertical averaging« je metoda navpičnega povprečenja.
  - »Treshold averaging« je metoda povprečenja glede na prag.
  - »None« preprosto prikazuje krivulje vseh izbranih testnih setov in jih ne povpreči.
  - »Vertical smoothing« ustreza metodi glajenja.



Slika 9: grafični gradnik »Classification Curve«, zavihek »Graph type«

Elementi zavihka »Graph type«, ki ga lahko vidimo na sliki 9, so:

- »Default alghoritms«; uporabnik ima na voljo nekatere izmed najbolj uporabljanih grafov, kot so ROC in Precision-Recall. V primeru, da si želi risati krivulje po svojih željah, mora izbrati možnost »User defined formula«.

- Spustni kombinirani seznam (*drop-down combo box*) »Target class« je kontrola, s katero določimo, kateri izmed možnih razredov bo obravnavan kot pozitiven, oziroma kot targetClass. Vsi ostali se obravnavajo kot negativni.

- Kontrola »Algorithm definition« je sestavljena iz več elementov. V primeru, ko v kontroli »Default algorithms« ni izbrana možnost »User defined formula«, je ta kontrola onemogočena in je uporabnik ne more spreminjati. Kontrola je sestavljena iz naslednjih gradnikov:
  - prva gradnika sta okenci, v kateri vpisujemo formulo po kateri algoritem računa x oziroma y koordinato
  - poleg okenca za koordinato x oziroma y je gumb. Ob kliku na gumb se odpre okence, v katero lahko vpisujemo naziv osi
  - naslednji gradnik je »Insert into formula«. Vsebuje dva spustna kombinirana seznama. S prvim lahko uporabnik v formulo doda ime atributa ali spremenljivke, z drugim pa vanjo lahko dodaja vrednosti teh atributov
  - gradnik »Sort data by« prav tako vsebuje dva spustna kombinirana seznama. S prvim določimo, po katerem atributu naj se uredijo podatki, z drugim pa določimo ali naj bo to urejanje po naraščujočem ali padajočem vrstnem redu
  - s kontrolo »Merge elements with same value of sort« uporabnik določa, kako naj algoritem obravnava primere, ki imajo enako vrednost atributa, po katerem smo primere uredili. Na voljo so tri možnosti: »Last value(ROC)«, »Average«, »No merge«
  - zadnji element je gumb »Compute«, s katerim povzročimo, da se točke krivulj ponovno izračunajo.

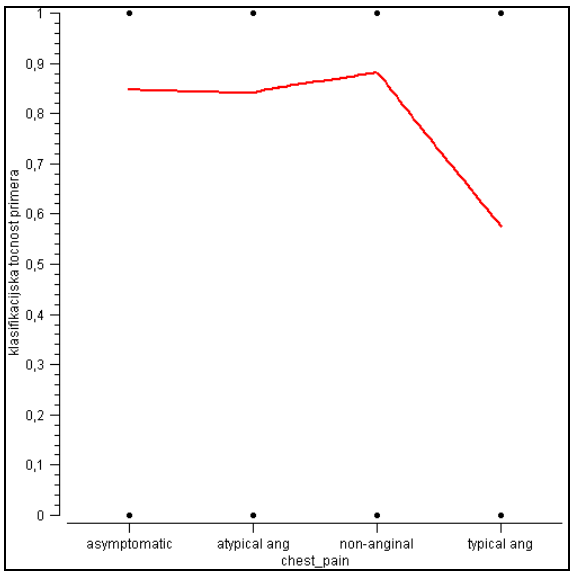
3. Primeri

Do sedaj smo opisali delovanje algoritma in grafičnega gradnika ter njegov uporabniški vmesnik, zdaj pa je čas, da na primerih pokažemo nekatere možnosti njegove uporabe.

3.1 Primeri s podatki o srčnih boleznih

Prvi del primerov prikazuje obnašanje bayesovega klasifikatorja na podatkih - heart_disease.tab, ki so priloženi programskemu paketu Orange. Ti podatki prikazujejo obolevanje pacientov za srčnimi boleznimi glede na parametre, kot so spol, krvni tlak in drugo. Kot ciljni razred si bomo izbrali »1« kar pomeni, da pacient zboli.

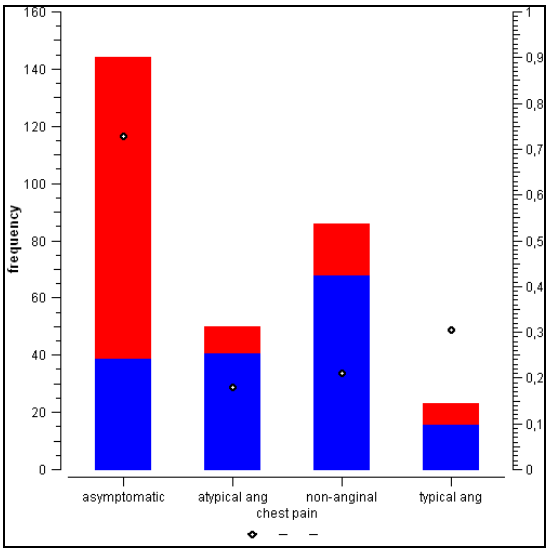
Če preizkušamo različne poglede lahko postanemo pozorni na naslednji graf. Na osi x prikazujemo atribut chest_pain, na osi y klasifikacijsko točnost primera (*eCA*), za način povprečenja pa izberemo metodo glajenja (*vertical smoothing*). Tako dobimo graf, ki je prikazan na sliki 10.



Slika 10: graf klasifikacijske točnosti primera v odvisnosti od atributa chest_pain

Graf kaže, da klasifikator bistveno slabše napoveduje v primeru, ko je vrednost atributa chest_pain enaka »typcal_ang«. V tem primeru klasifikator pravilno napove samo približno 58% vseh primerov. Zanima nas, zakaj je klasifikacijska točnost primera v tem primeru toliko slabša.

V iskanju odgovora si poglejmo graf porazdelitve bolnih in zdravih glede na atribut chest_pain, ki je prikazan na sliki 11. Kot modri so označeni pacienti, ki ne zbolijo, kot rdeči pa tisti, ki zbolijo. Graf smo dobili s pomočjo grafičnega gradnika »Distributions«, ki je del okolja Orange.



Slika 11: graf porazdelitve bolnih in zdravih pacientov v odvisnosti od atributa chest_pain

Opazimo lahko, da je delež bolnih, ki imajo vrednost atributa chest_pain enako »typical_ang«, majhen. Poleg tega je ta skupina bolnikov najmanša med vsemi.

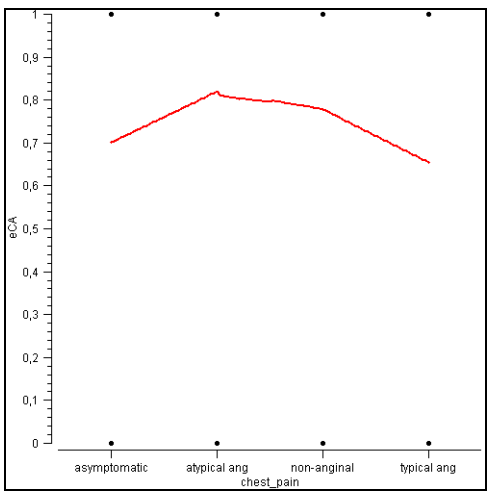
Klasifikacijsko natančnost primera, kot jo vidimo na sliki 10, lahko s pomočjo formul izrazimo kot:

```
1 if  ( ( probability >= 0.5 and actualClass == targetClass ) or
        ( probability <  0.5 and actualClass != targetClass ) )
else 0 .
```

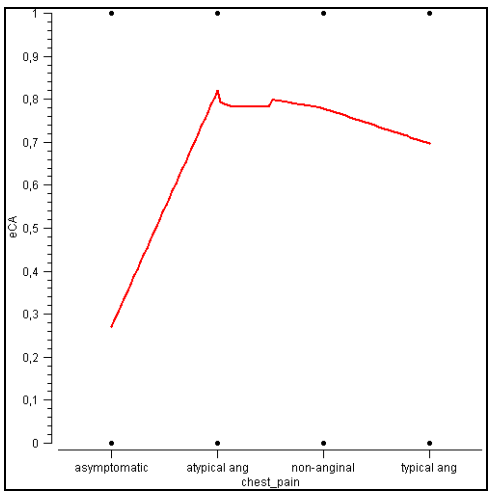
Vrednost 0.5 pomeni, da klasifikator primere, ki so pod to mejo, oceni za negativne oziroma zdrave, tiste nad to mejo pa za pozitivne oziroma bolne.

Ker je porazdelitev pri skupini, ki ima atribut chest pain enak »typical ang«, izrazito v prid zdravih, se vprašajmo, ali lahko klasifikacijsko točnost izboljšamo, če to mejo pomaknemo višje.

Na slikah 12 in 13 sta klasifikacijski točnosti, ki ju dobimo, če to mejo pomaknemo na 0.99 oziroma 1.0 .



Slika 12



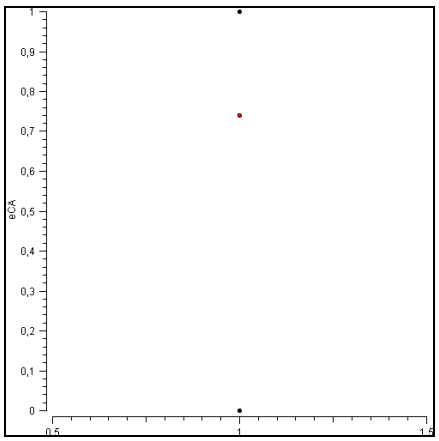
Slika 13

Na sliki 12 lahko vidimo, kaj se zgodi, če mejo, kjer pacienta proglasimo za bolnega, povišamo na 99%. Klasifikacijska točnost primera za primere, kjer je vrednost atributa chest_pain enaka »typical_ang«, se bistveno izboljša. Znaša približno 65 % in je tako za sedem odstotkov večja kot klasifikacijska točnost v prejšnjem primeru.

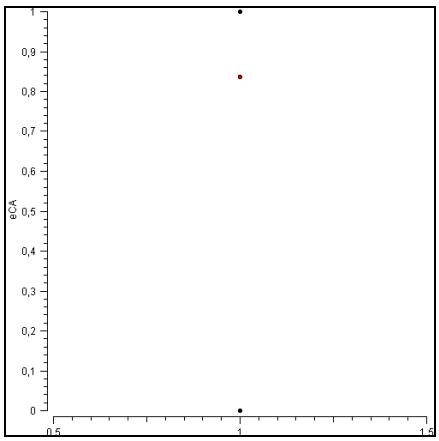
Zdi se, da bi bilo za primere, kjer je vrednost atributa chest_pain enaka »typcal_ang«, najbolje, če to mejo postavimo kar na 100%. Tako bomo za bolne proglasili samo paciente, za katere bo klasifikatorjeva ocena verjetnosti, da je pacient bolan, enaka 100%.
Na sliki 13 vidimo, da se v tem primeru klasifikacijska točnost primera za primere, kjer je vrednost atributa chest_pain enaka »typcal_ang«, izboljša na 70%.

Tako povečana klasifikacijska točnost primera, za primere, kjer je vrednost atributa chest_pain enaka »typical_ang«, na žalost ni brez posledic za preostale primere.
Na slikah 12 in 13 lahko opazimo, da se klasifikacijska točnost pri ostalih primerih poslabša.

To poslabšanje klasifikacijske točnosti pri vseh primerih skupaj lahko vidimo iz grafov na slikah 14 in 15. Grafe dobimo tako, da za formulo, po kateri računamo koordinato x, namesto atributa chest_pain vpišemo 1. Tako dobimo klasifikacijsko točnost primera za vse primere.
Na sliki 14 je klasifikacijska točnost primera, če prag postavimo pri 100%, na sliki 15 pa klasifikacijska točost primera v primeru, ko je ta prag enak 50%.



Slika 14

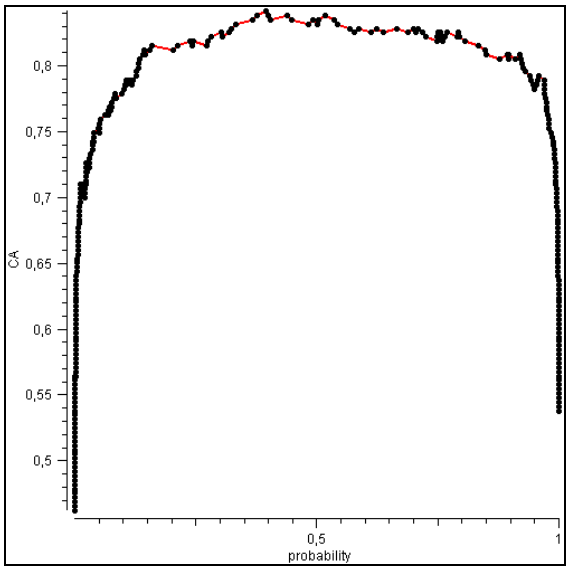


Slika 15

Vidimo, da je za skupno klasifikacijsko točnost precej bolje, če meja ostane na 50 %.

Vprašamo se lahko še, kako postaviti prag, da bo klasifikacijska točnost največja. V ta namen lahko uporabimo graf klasifikacijske točnosti glede na prag. Dobimo ga tako, da na osi x prikazujemo napovedano verjetnost, da bo pacient zbolel, na osi y pa klasifikacijsko točnost. Ta graf lahko vidimo na sliki 16.

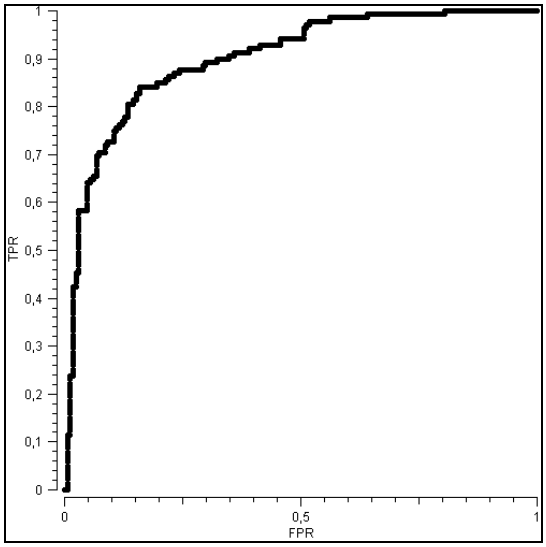
Iz grafa lahko razberemo, da največjo klasifikacijsko točnost dosežemo, če prag postavimo pri približno 40 %. V tem primeru znaša klasifikacijska točnost približno 84 %. Lahko vidimo tudi to, da je klasifikacijska točnost pri pragu 50% približno 83%, pri 99% pa približno 74%, kar se ujema s slikama 14 in 15, kjer prikazujemo klasifikacijsko točnost primera.



Slika 16: graf klasifikacijske točnosti v odvisnosti od praga

Vprašanje, ki si ga lahko zastavimo, je koliko je klasifikacijska točnost v danem primeru pomembna. Pove nam namreč, v koliko odstotkih je naša napoved pravilna, za nas pa je pravzaprav bolj zanimivo, kolikšen odstotek bolnikov odkrijemo.

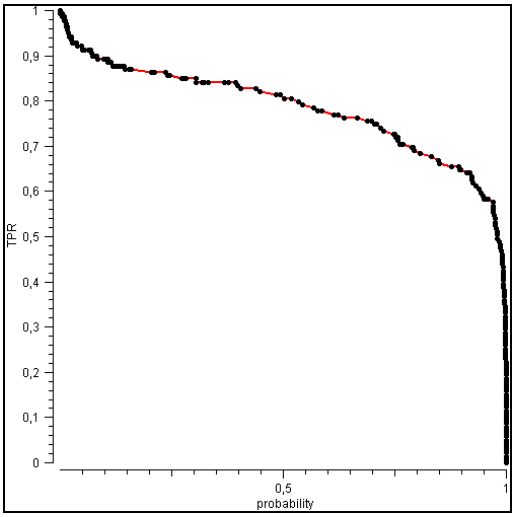
ROC graf s slike 17 prikazuje prav to. Na osi y prikazuje delež bolnikov, ki jih odkrijemo, na osi x pa delež zdravih pacientov, ki jih po nepotrebnem prestrašimo.



Slika 17: ROC graf

Iz grafa lahko vidimo, da je potrebno za to, da odkrijemo visok odstotek bolnih, potrebno po nepotrebnem preiskati tudi veliko zdravih pacientov. Kje točno postaviti mejo, je seveda naloga uporabnika s strokovnim znanjem. Pri odločanju bi mu lahko pomagala še naslednja grafa.

Graf na sliki 18 prikazuje razmerje med pragom, ki ga izberemo pri odločanju katerega pacienta bomo poslali na dodatne preiskave, in deležem bolnikov, ki jih pri tem pragu odkrijemo.



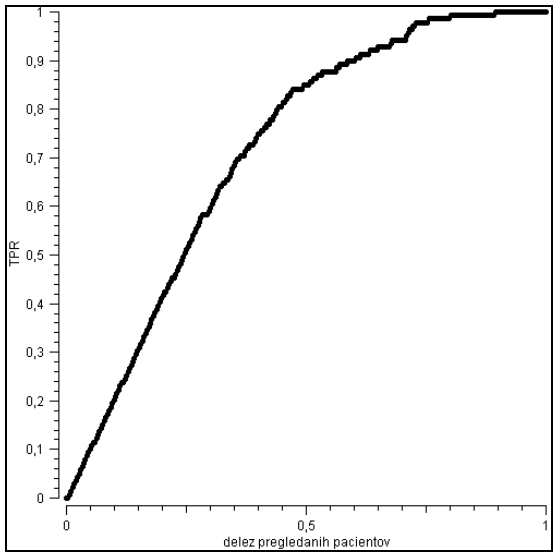
Slika 18: graf deleža odkritih bolnikov v odvisnosti od praga

Iz grafa lahko razberemo naslednje ugotovitve: Če prag postavimo zelo visoko (blizu 1), bo delež odkritih bolnikov nizek. Takoj ko prag zmanjšamo, se ta delež bistveno poveča, vendar samo do točke, ko odkrijemo približno polovico vseh bolnikov. Od te točke naprej se delež povečuje počasneje. Če želimo odkriti 85 % bolnikov, moramo ta prag spustiti na 30 %. To pomeni, da na dodatne preiskave pošljemo paciente, ki jim klasifikator napove 30% ali več odstotkov možnosti bolezni.

Uporabnik , ki ima omejene vire, kot je na primer EKG aparat, se utegne vprašati tudi, kolikšen delež ljudi naj pregleda. Če se za prag določi meja, ki bo prenizka in bodo pregledani praktično vsi pacienti, bodo preiskave zamujale. Zaradi tega se lahko zgodi, da bodo bolniki, ki bi ta pregled potrebovali bolj nujno, umrli zaradi nepotrebnega čakanja.

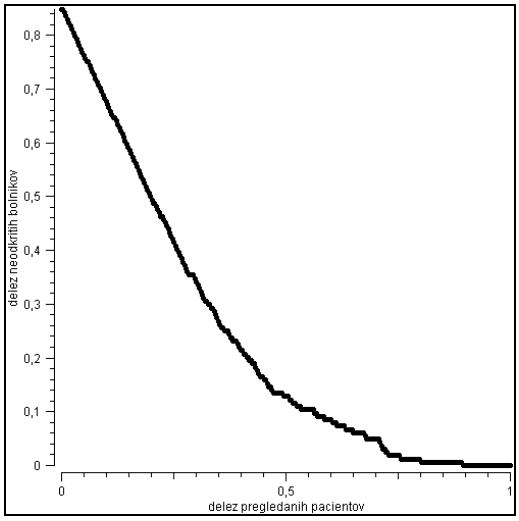
Enako se lahko uporabnik vpraša, kakšne zmogljivosti naprav bi potreboval, da bi se pregledi lahko izvedli v primernem času.

Pri odločitvi bi si lahko pomagal z grafom, kjer je na osi x delež pacientov, ki jih pregledamo, na osi y pa delež bolnih, ki jih s temi pregledi odkrijemo. Graf najlažje dobimo tako, da za formulo koordinate x uporabimo izraz `cumm(1) / NN`, za formulo koordinate y izraz TPR in izberemo možnost urejanje podatkov po padajoči verjetnosti.
To je potrebno zato, ker bomo v vsakem primeru najprej pregledovali bolnike, ki jih ocenimo kot bolj verjetno bolne. Opisani graf lahko vidimo na sliki 19.



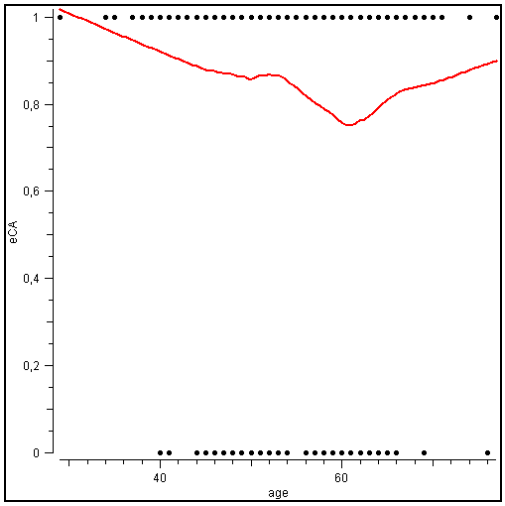
Slika 19: graf deleža odkritih bolnikov v odvisnosti od deleža pregledanih pacientov

Spet vidimo, da je za to, da odkrijemo večino bolnih, potrebno preiskati velik delež pacientov. Če si želimo odkriti 85 % vseh bolnikov, moramo preiskati približno polovico pacientov.
Isto informacijo lahko na malce drugačen način predstavimo z grafom, kjer na osi y namesto deleža odkritih bolnikov prikazujemo delež bolnikov, ki jih ne odkrijemo.
Formula za os y se zato spremeni v FN / P, vse ostalo pa ostane enako. Graf je prikazan na sliki 20.



Slika 20: graf deleža neodkritih bolnikov glede na delež pregledanih pacientov

Če ostanemo pri podatkih o srčnih bolnikih, lahko analiziramo še graf s slike 21. Na osi x je prikazana starost pacientov, na osi y pa klasifikacijska točnost primera (*eCA*). Kot način povprečenja je uporabljeno glajenje.

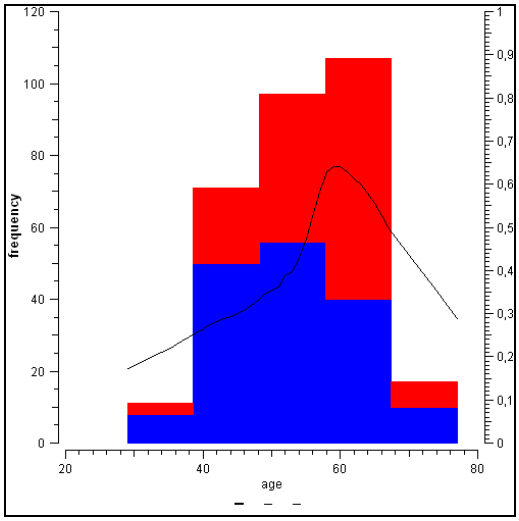


Slika 21: graf klasifikacijske točnosti primera v odvisnosti od starosti

Glajenje iz točk, ki jih vidimo v zgornjem in spodnjem delu grafa, izračuna rdečo krivuljo, ki prikazuje povezanost teh točk. Vsaka točka nosi podatek o starosti posameznega primera in pravilnosti njegove klasifikacije. Če je bila napoved klasifikatorja pravilna, je klasifikacijska točnost primera enaka 1, v nasprotnem primeru pa je enaka 0. Glajenje iz teh točk izračuna krivuljo, ki jo lahko interpretiramo kot verjetnost pravilne klasifikacije za določeno starostno skupino.

Vidimo, da se naš klasifikator precej dobro odreže pri bolnikih, mlajših od 50 let ali starejših od 65 let, v vmesnem obdobju pa se verjetnost, da bo klasifikator napovedal pravilni razred, zmanjša.

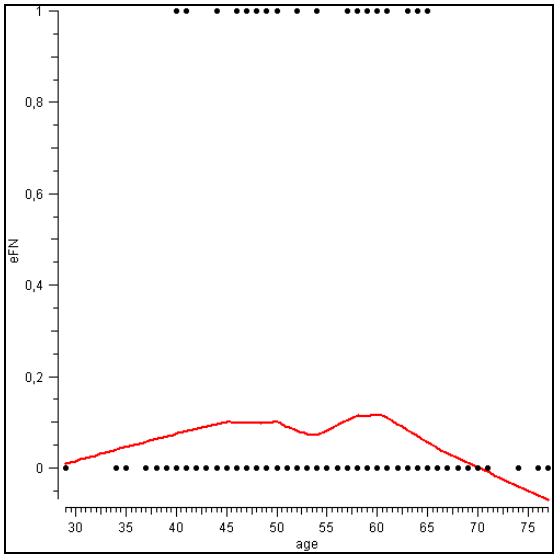
Če si pogledamo graf s slike 22, nas lahko stanje še bolj skrbi. Na sliki je prikazana porazdelitev bolnih glede na starost, z modro barvo so prikazani zdravi, z rdečo pa bolni pacienti. Vidimo lahko, da se delež bolnih drastično poveča ravno v obdobju, kjer se naš klasifikator odreže bistveno slabše.



Slika 22: graf porazdelitve bolnih pacientov glede na starost

Zanima nas seveda spet predvsem to, koliko bolnikov bomo poslali v prezgodnji grob, še posebej v tem očitno precej kritičnem življenjskem obdobju med petdesetim in petinšestdesetim letom.

To lahko ugotovimo iz grafa na sliki 23. Na njem je namesto klasifikacijske točnosti primera prikazan eFN primera. Ta je enak 1, če je pacient bolan in tega ne ugotovimo, v vseh drugih primerih je enak 0.



Slika 23: graf deleža neodkritih bolnikov v odvisnosti od starosti

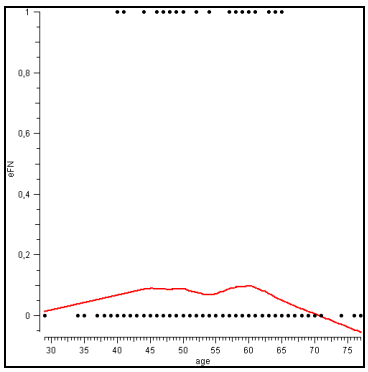
Graf nam kaže razmere nekoliko drugače kot tisti s slike 21. Oba grafa nakazujeta, da se klasifikator razmeroma dobro odreže pri mlajših in starejših pacientih. Razlika je v tem, da sta na grafu s slike 23 vidni dve območji, kjer je delež neodkritih pacientov večji, na grafu s slike 21 pa samo eno območje, kjer je klasifikacijska natančnost izrazito slabša.

Po grafu s slike 23 lahko sklepamo, da se delež neodkritih bolnikov poveča že okrog petinštiridestega leta, se nato od petdesetega do petinpetdesetega leta nekoliko zmanjša in doseže nov vrhunec okrog šestdesetega leta. Razlike so zelo majhne, zato je dokaj verjetno, da so naključne.

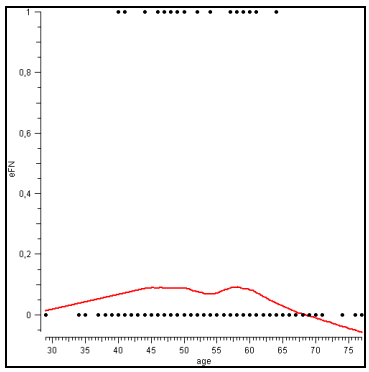
V območju, kjer je delež neodkritih bolnikov največji, ta znaša približno 12 %. To pomeni, da bi klasifikator kot zdrave ocenil 12 odstotkov bolnikov v tej starostni skupini.

Na srečo stanje ni tako dramatično. Pri grafih smo namreč v obeh primerih uporabljali spremenljivke, ki so vezane na mejo, postavljeno pri 50%. Poglejmo si še, kaj se zgodi, če to mejo postavimo nižje.

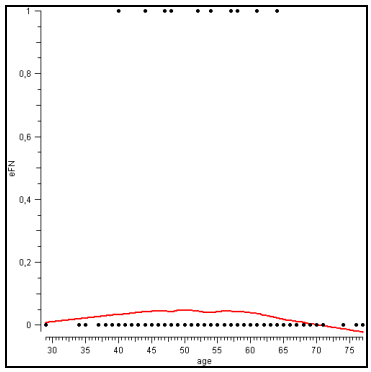
Najprej moramo definirati eFN, ki je v zvezi s pragom, ki ga bomo določili. Izraz `predictedClass != targetClass == actualClass` nam opredeljuje eFN, kjer je prag postavljen na 50%. To je posledica tega, da klasifikator v primeru, da ne zahtevamo drugače, prag postavi na 50 %. Če hočemo to izraziti s pomočjo verjetnosti, dobimo izraz:
`1 if ( targetClass == actualClass and probability <= 0.5) else 0.`



Slika 24



Slika 25



Slika 26

Na sliki 24, 25 in 26 vidimo grafe deleža neodkritih bolnikov v odvisnosti od starosti, če prag postavimo pri 40, 30 oziroma 5 odstotkih.

Graf s slike 24, kjer je prag enak 40%, je po obliki zelo podoben tistemu s slike 23. Delež neodkritih bolnikov je še vedno največji okrog šestdesetega leta in znaša približno 9.7%. Podobno lahko rečemo za graf s slike 25, kjer je oblika še vedno zelo podobna, delež neodkritih bolnikov pri starosti šestdeset let pa se zmanjša na 9.2%. Največjo razliko lahko opazimo, če prag spustimo na samo 5%, to prikazuje graf s slike 26. V tem primeru sta dva vrhova precej neopazna, delež neodkritih bolnikov pa v nobeni starostni skupini ne preseže 5%.

To izboljšanje ima, na žalost svojo ceno. Ker smo prag postavili tako nizko, to pomeni, da moramo dodatno pregledati vsakega pacienta, za katerega klasifikator ugotovi vsaj pet odstotkov možnosti, da bo zbolel. Kot je razvidno iz grafa s slike 20, moramo zato preiskati približno 68 odstotkov pacientov.

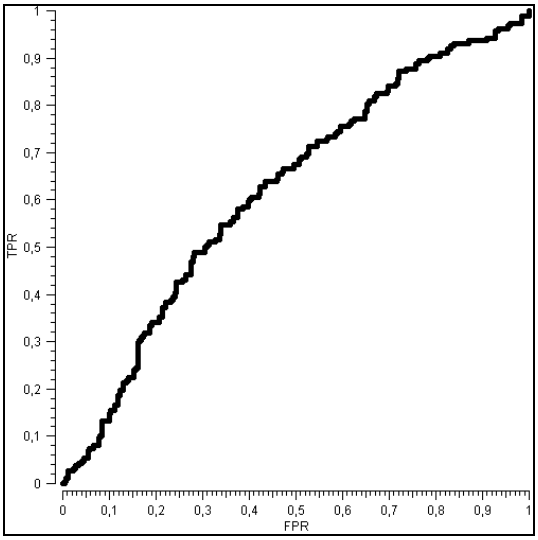
3.2 Primeri, povezani s podatki o prosilcih za posojilo

Za naslednje primere si oglejmo problem, ki smo ga opisali v poglavju 1.2.2. Kot ciljni razred si izberemo razred »yes«, kar pomeni, da prosilec posojilo odplača.

Pri prejšnjem problemu cilja nismo popolnoma definirali, pravzaprav so nas bolj zanimala razmerja med odkritimi in po nepotrebnem preiskanimi pacienti, točne meje pa nismo znali postaviti. To je posledica tega, da je težko določiti, kolikšen delež bolnikov smo pripravljeni izgubiti.

Tokrat je cilj reševanja problema jasen. Ugotoviti moramo, kako klasifikator uporabiti tako, da bo banki prinesel čim večji dobiček.

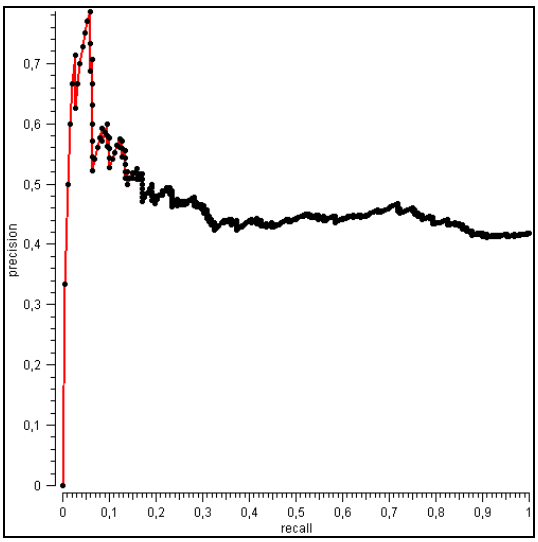
Pri analiziranju problema najprej poglejmo, kaj lahko ugotovimo z uporabo metod ROC in Precision- Recall, ki sta med bolj znanimi metodami za ocenjevanje učinkovitosti klasifikatorjev.



Slika 27 : ROC graf

Na sliki številka 27 je ROC graf. Na osi y prikazujemo TPR, kar ustreza deležu prosilcev, ki nameravajo kredit vrniti in jim ga odobrimo. Na osi x prikazujemo FPR, kar ustreza deležu prosilcev, ki kredita ne nameravajo vrniti, a jim ga vseeno odobrimo. Prav ti dve količini sta za nas najpomembnejši, problem je le v tem, da je na prvi pogled težko oceniti, kje postaviti mejo.

Graf s slike 28 je graf preciznost–priklic (*precision –recall*). Na osi x prikazuje priklic, na osi y pa preciznost. Preciznost nam v danem primeru pove razmerje med prosilci, ki jim odobrimo posojilo in ga vrnejo ter med vsemi prosilci, ki jih pravilno ocenimo. Priklic nam pove, kolikšen delež prosilcev, ki bodo posojilo vrnili, smo odkrili.



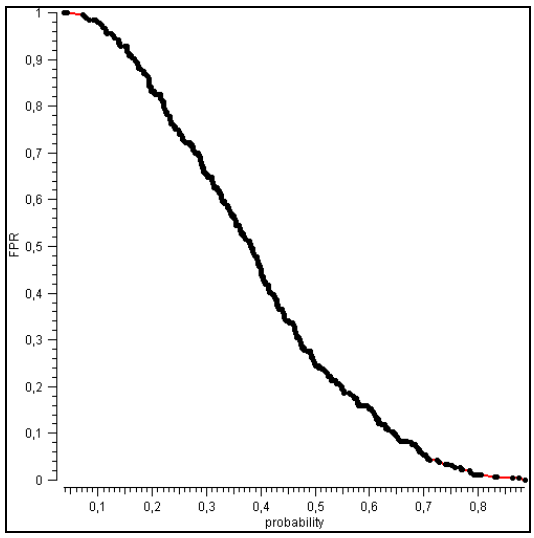
Slika 28 : graf preciznost - priklic

Iz grafa lahko ugotovimo, da je razmerje ugodno, ko je odkriti delež poštenih prosilcev nizek, torej takrat, ko je napovedana verjetnost visoka. Če mejo spustimo nižje, se razmerje poslabša.

S pomočjo prvih dveh grafov smo prišli do nekega temeljnega razumevanja problema. Problem leži v tem, da se še vedno ne znamo odločiti, kje postaviti mejo, komu odobriti posojilo in koga zavrniti.

Iz grafa na sliki 27 bi po občutku lahko to mejo postavili v točki, kjer je FPR približno 28%. V tem delu se krivulja po obdobju hitre rasti umiri. Druga možnost bi bila, da mejo postavimo v točki, kjer je FPR enak 63 %, zato ker po tej točki pride del, kjer krivulja raste počasneje.

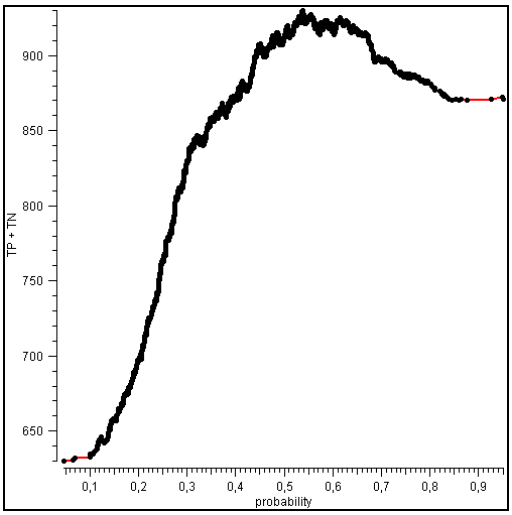
Teh točk je seveda na grafu veliko, vendar za nobeno ne moremo biti popolnoma prepričani, da je najprimernejša kot meja za odobritev posojila. Iz grafa na sliki 29 lahko vidimo, da bi bili verjetnosti, ki bi ustrezali zgoraj navedenenima FPR, približno 50 % oziroma 30 %.



Slika 29: graf FPR v odvisnosti od napovedane verjetnosti

Oglejmo si še graf s slike 30. Na osi x je verjetnost, na osi y pa vsota TP in TN. Zanima nas, kje bo ta vsota največja. Iz grafa lahko razberemo, da imamo največ pravilnih napovedi v primeru, ko mejo postavimo na približno 54 %.

Po tem grafu se zdi najpametneje mejo postaviti na približno 48 %, saj bi s tem najbolj povečali število pravilnih odločitev. Isto sliko dobimo, če namesto `TP + TN` prikazujemo `CA`.



Slika 30: Graf pravilno napovedanih primerov v odvisnosti od praga

Problem leži v tem, da imajo pravilne pozitivne in pravilne negativne odločitve različne nagrade, ki smo jih opisali že v poglavju 1.2.2.

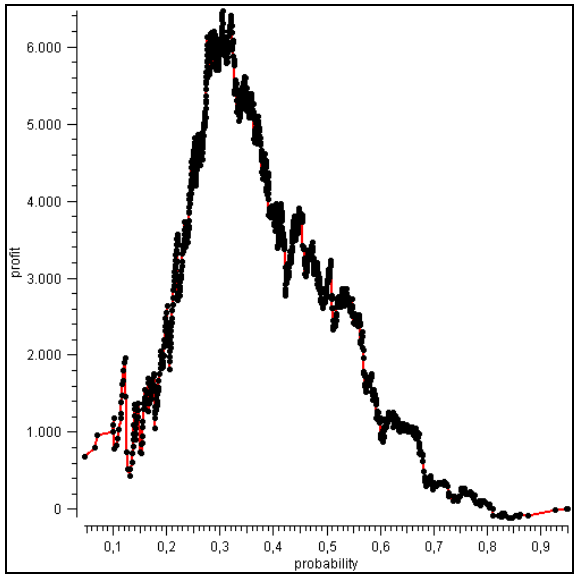
Ker nas v našem primeru zanima dobiček, je najbolje risati kar dobiček v odvisnosti od praga. To dosežemo tako, da na osi x prikazujemo napovedano verjetnost, na osi y pa dobiček.

Formulo za računanje dobička moramo seveda definirati sami. Po formulah iz poglavja 1.2.2 lahko ugotovimo, da ga lahko izrazimo kot:

```
cumm( ( amount_requested **2 / 30) * eP - (amount_requested * eN) ).
```

Izbrati moramo urejanje podatkov po padajočih verjetnostih in opcijo »No merge«.
Če za način povprečenja izberemo možnost »Merge data folds«, dobimo graf s slike 31.

Po tem grafu lahko sklepamo, da je najboljša meja za odobritev približno 30%, v tem primeru je dobiček največji. To pomeni, da posojilo odobrimo prosilcem, za katere klasifikator napove več kot tretjino možnosti za vračilo posojila.



Slika 31: graf dobička v odvisnosti od praga

V resnici je boljša rešitev ta, da se za odobritev odločamo za vsak posamezni primer posebej. Za vsak posamezni primer lahko izračunamo potencialni dobiček po naslednji formuli:

$$d = \frac{p * \text{želeni znesek}^{2}}{30} + (1 - p) * \text{želeni znesek},$$

kjer je p verjetnost, da bo prosilec posojilo vrnil. Če je dobiček primera pozitiven, prosilcu posojilo odobrimo, v nasprotnem primeru pa ga zavrnemo.

4. Zaključek

V diplomski nalogi smo opisali način delovanja grafičnega gradnika in nekatere primere njegove uporabe. Za konec se je potrebno še vprašati, če so cilji, ki smo si jih zastavili, izpolnjeni.

Prvi in najbolj bistven cilj je bil narediti orodje, s katerim bo uporabnik lahko analiziral obnašanje klasifikatorjev na način, ki si ga bo sam izbral. Ta cilj je v večjem delu izpolnjen. Uporabnik lahko uporablja nekatere standardne grafe, poleg tega pa lahko algoritem računanja točk prilagaja svojim potrebam.

Na žalost ostajajo nekatere omejitve. Uporabnik lahko riše grafe, dokler ostaja znotraj omejitev osnovnega algoritma, ki smo ga opisali v poglavju 2.1.1. Spreminja lahko samo parametre algoritma, ne pa tudi samega algoritma.

Edini način, kako odpraviti to pomanjkljivost, bi bil, da uporabniku dopustimo pisanje lastnih algoritmov, vendar je bilo treba to zamisel opustiti. Ne samo, da bi bila zelo težavna s stališča implementacije in bi že tako na nekaterih mestih zapletenemu gradniku dodala popolnoma novo dimenzijo zapletenosti, temveč bi bila najverjetneje tudi prezapletena za veliko večino uporabnikov.

Kljub temu, da z njim ni mogoče risati čisto vseh grafov, ki si jih uporabnik zamisli, dodaja grafični gradnik »Classification Curve« v okolje Orange nov pogled na ocenjevanje klasifikatorjev. Uporabnik si lahko delovanje klasifikatorja ogleda s plati, ki ga najbolj zanima in pri tem ni več omejen s samo štirimi obstoječimi grafi.

Prav ta svoboda uporabnika lahko predstavlja problem. Gradnik dopušča uporabniku veliko svobode pri določanju parametrov algoritma in ga poskuša čim manj omejevati.
Razlog za to svobodo je v tem, da avtor ne more predvideti vseh različnih možnosti, ki jih lahko porodi človeška domišlija. Problemi v realnem svetu so različni in prav tako pristopi posameznikov pri reševanju teh problemov.

Nekatere možnosti, ki jih gradnik ponuja, se tako zdijo neuporabne. Dosti jih bo verjetno takih tudi ostalo, za nekatere pa se utegne zgoditi, da jih nekdo nekega dne prepozna kot rešitev svojega problema.

Iz svobode uporabnika seveda sledi tudi možnost napačne uporabe. Hitro se namreč lahko zgodi, da si uporabnik napačno predstavlja, kaj prikazuje graf, ki ga je definiral.
Proti temu na žalost ni prave rešitve. Vsako močno orodje lahko namreč v primeru, da ga ne znamo uporabljati, povzroči več škode kot koristi.

Prvotni namen, nadomestitev grafičnih gradnikov »ROC Curve«, »Lift Chart« in »Calibration Plot« z gradnikom »Classification Curve« ni uspel.

Že pri primerjavi ROC krivulje, ki jo riše grafični gradnik »ROC Curve«, s to, ki jo riše gradnik »Classification Curve«, lahko opazimo, da ima prvi nekaj dodatnih možnosti, kot je na primer, risanje konveksne lupine. Čeprav bi v gradnik lahko dodali nekatere od teh možnosti, bi to dodatno zapletlo ne samo programiranja gradnika, ampak tudi sam izgled

uporabniškega vmesnika. Če bi te posebne značilnosti dodali za ROC krivulje, potem seveda ni vzroka, da jih ne bi dodali še za krivulje Precision-Recall.

Vse to bi gradnik naredilo nepregleden in težko razumljiv. Zaradi tega je mogoče z gradnikom risati samo osnovne krivulje. Uporabnik, ki si želi bolj natančno ogledati ROC krivulje, lahko še naprej uporablja obstoječi gradnik, ki se ukvarja izključno s tem tipom krivulj.

Zadnji razlog za njihovo ohranitev je v tem, da nima smisla ukiniti orodij, ki jih uporabniki okolja Orange že dobro poznajo in znajo uporabljati.

Tako je grafični gradnik »Classification Curve« dodan samo kot dopolnilo že obstoječim gradnikom.

Upam, da bodo uporabniki nov gradnik dobro sprejeli in ga s pridom uporabljali.

5. Literatura

[1] T. Fawcet, ROC *Graphs: Notes and Practical Considerations for Researchers*, Netherlands: Kluwer Academic Publishers,2004 .

[2] (2008) Local regression. Dostopno na http://en.wikipedia.org/wiki/Local_regression

[3] J. Davis and M. Goadrich, »*The Relationship Between Precision-Recall and ROC Curves*,« v zborniku 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006.

[4] J. Demsar, B. Zupan, and G.Leban, »*Orange: From Experimental Machine Learning to Interactive Data Mining*,« Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, White Paper (www.ailab.si/orange) 2004.

[5] R.Holte, *Personal Communication*, 2002.

[6] (2008) *Cumulative Gains and Lift Charts*. Dostopno na: http://www2.cs.uregina.ca/~dbd/cs831/notes/lift_chart/lift_chart.html

[7] (2008) *Information retrieval*. Dostopno na: http://en.wikipedia.org/wiki/Information_retrieval

Izjava o avtorstvu

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod mentorstvom doc. dr. Janeza Demšarja. Sodelavce, ki so mi pri delu pomagali, sem navedel v zahvali.

Miha Biček