UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Mlakar

Primerjava točnosti diagnostičnih modelov raka zgrajenih s strojnim učenjem iz podatkov o genskih izrazih

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2010

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: 01605/2009
Datum: 15.10.2009

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

Kandidat: **MIHA MLAKAR**

Naslov: **PRIMERJAVA TOČNOSTI DIAGNOSTIČNIH MODELOV RAKA ZGRAJENIH S STROJNIM UČENJEM IZ PODATKOV O GENSKIH IZRAZIH**

ACCURACY OF CANCER DIAGNOSIS MODELS INFERRED BY MACHINE LEARNING FROM GENE EXPRESSION DATA SETS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V diplomski nalogi preučite, kakšna je napovedna točnost različnih tehnik strojnega učenja pri gradnji in uporabi napovednih modelov v diagnostiki raka. Modele gradite iz podatkov o genskih izrazih. Podatke pridobite iz spletne baze podatkov Gene Expression Omnibus. Pri vrednotenju skušajte uporabiti čim več ustreznih podatkovnih naborov. Rezultate primerno statistično obdelajte in poročajte o najbolj uspešnih tehnikah.

Mentor:

prof. dr. Blaž Zupan

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Miha Mlakar,

z vpisno številko 63030170,

sem avtor diplomskega dela z naslovom:

Primerjava uspešnosti tehnik strojnega učenja v diagnostiki raka iz podatkov o genskih izrazih.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  prof. dr. Blaža Zupana
- 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 15.4.2010 Podpis avtorja:

Zahvala

Na tem mestu se zahvaljujem mentorju prof. dr. Blažu Zupanu za pomoč, nasvete in predloge pri nastajanju diplomskega dela. Zahvaljujem se tudi Kristini ter družini za podporo in spodbudo tekom celotnega študija.

Kazalo

Seznam uporabljenih kratic in simbolov

DNK - deoksiribonukleinska kislina, nosilka genetske informacije v vseh živih organizmih

RNK - ribonukleinska kislina, prenašalka genske informacije med DNK in ribosomi

mRNK (angl. *messenger RNA*) - informacijski RNK

cDNK (angl. *complementary DNA*) - komplementarna DNK

GEO (angl. *Gene Expression Omnibus*) - javno skladišče množic podatkov o genskih izrazih

ROC (angl. *Receiver Operating Characteristic*) - pokazatelj karakteristike delovanja klasifikacijske metode

AUC (angl. *Area Under ROC Curve*) - mera za ugotavljanje klasifikacijske natančnosti

SVM (angl. *Support Vector Machines*) - metoda podpornih vektorjev

kNN (angl. *k-Nearest Neighbors*) - metoda bližnjih sosedov

Povzetek

Iz podatkov o izraženosti genov v celicah lahko s pomočjo strojnega učenja poskusimo napovedati, ali je določeno tkivo benigno ali maligno. V diplomskem delu smo testirali različne tehnike strojnega učenja na podatkih, ki smo jih dobili iz javne baze na internetu. Testirali smo na večih različnih množicah podatkov, da bi dobili čim bolj zanesljive rezultate. Za vsako metodo smo izračunali povprečno vrednost mere AUC preko vseh množic podatkov in izdelali graf kritičnih razdalj, ki pokaže povprečne range metod. Rezultati, ki smo jih dobili, so malo presenetljivi. Pričakovali smo, da se bo najbolje obnesla metoda podpornih vektorjev, vendar se je za nekoliko uspešnejšo izkazala metoda klasifikacija z naključnimi gozdovi. Standardni odklon mere uspeha je bil relativno visok, tako da bi lahko bil vrstni red metod na drugih podatkih drugačen.

Ključne besede:

strojno učenje, bioinformatika, diagnostika, genski izraz, mikročipi, rak

Abstract

Using machine learning on gene expression data we can try to predict if tissue is benign or malignant. We have evaluated different machine learning technique on the data that we have obtained from the public data base Gene Expression Omnibus. The algorithms were tested on different data sets to get more reliable results. The methods were scored using AUC measure and statistically compared in a critical distance graph. The results were a bit surprising. We expected that the best method would be support vector machines method, but it was method of random forests. Standard deviation was relatively high so the order of methods could be different on some other data.

Key words:

machine learning, bioinformatics, diagnostics, gene expression, microarray, cancer

Poglavje 1

Uvod

Vsaka celica v človeškem telesu vsebuje celoten nabor genov v genomu. Vendar pa so v danem trenutku aktivni samo nekateri geni in zato celice lahko opravljajo različne funkcije. Geni nosijo zapis za izdelavo zaporedja aminokislin, ki sestavljajo proteine, proteini pa opravljajo večino najpomembnejših funkcij v celici. Zato prav to, kateri geni so v določeni celici izraženi, določa edinstvene lastnosti za vsako celico.

Genski izraz je termin, ki se uporablja za opisovanje prepisa genskih informacij, ki se nahajajo v DNK, v sporočilno RNK (mRNK) molekulo. Informacija, ki jo nosi mRNK, se uporabi za izdelavo proteinov. Proučevanje mRNK, ki ga proizvede celica, vodi do spoznanja, kateri geni so izraženi, kar omogoča vpogled v to, kako celica deluje. Izražanje genov je zelo zapleten in strogo reguliran proces, ki omogoča celici, da se dinamično odziva tako na spodbude iz okolja kot tudi na svoje lastne spreminjajoče se potrebe. Ta mehanizem deluje kot nekakšno stikalo, ki glede na potrebe kontrolira, kateri geni so v celici izraženi in obenem v kolikšnem obsegu oziroma v kakšni intenziteti.

Motnja ali sprememba v izražanju genov je vzrok za mnoge bolezni [8]. Zato je proučevanje izražanja genov in vloge, ki jo imajo specifični geni za razvoj bolezni, pomembno za diagnosticiranje določenih bolezni in tudi za iskanje novih zdravil za zdravljenje teh bolezni.

Ker je rak bolezen, ki je tesno povezana z genskimi izrazi, je zato na podlagi podatkov o genskih izrazih moč prepoznavati njegov tip (diagnostika) ali pa napovedovati njegov potek (prognostika). Tako je s pomočjo teh podatkov na primer mogoče ugotoviti, ali je nek tumor benigen ali maligen. Vse, kar potrebujemo za določitev, je genska slika (odtis) tumorja. Vsaj za nekatere vrste rakavih obolenj naj bi bil ta način že sedaj boljši od obstoječih načinov določevanja, ki so bolj dolgotrajni in bolj nezanesljivi, kar močno pripomore

k skrbi za bolnika. Seveda je gensko sliko (izraženost genov) razen za diagnostiko raka mogoče uporabiti še za druge zdravstvene namene. Pomaga nam pri določitvi ocene tveganja, ugotavljanju odziva na zdravila, diagnostičnih preiskavah, prognostični stratifikaciji in izbiri zdravljenja [11].

Napovedovanje stanja in poteka raka s pomočjo genskih izrazov predstavlja velik potencial za prihodnost, ker bo prispevalo k izboljšanju odkrivanja bolezni in predvsem k učinkovitejšemu zdravljenju bolnikov in je zato zelo zanimivo za preučevanje. Upoštevati pa je potrebno, da je ta način še relativno nov in zato še v povojih. Problemi, ki se v zvezi s tem pojavljajo, so poleg zaenkrat še (pre)majhnega števila eksperimentov oziroma opazovalnih vzorcev tkiv za posamezen tip bolezni natančnost analize in pa obdelava podatkov, pridobljenih z novimi metodami (npr. metoda pridobivanja podatkov o genskih izrazih z mikročipi).

Zaradi tega je to področje zelo zanimivo za analize in nove raziskave na področju modeliranja oziroma strojnega učenja. Način, s katerim bomo v diplomi poskusili napovedovati bolezen, bo potekal s pomočjo strojnega učenja. Obstaja veliko različnih tehnik strojnega učenja, ki iz danih podatkov poskušajo čim bolj natančno ugotoviti razred, kateremu ti podatki pripadajo. Osredotočili se bomo na diagnostiko, torej na prepoznavanje stanja vzorca celičnega tkiva. Razred je v primeru diagnosticiranja raka lahko benigen ali maligen.

Primerjali bomo uspešnost različnih tehnik strojnega učenja v diagnostiki raka iz podatkov o genskih izrazih. Podatke za našo študijo smo pridobili iz javno dostopnih spletnih virov.

Poglavje 2

Opis problema

2.1 Namen in cilji

Namen diplomske naloge je ugotoviti, katera od tehnik strojnega učenja oziroma katera klasifikacijska metoda je najprimernejša za ugotavljanje, ali je vzorec tkiva (npr. vzet iz nekega tumorja) rakavega izvora ali ne. Vhodni podatki, na katerih bomo to ugotavljali, so podatki o genskih izrazih. Ob tem bo potrebno uporabiti znanje s področja strojnega učenja, delovanja klasifikacijskih metod in samega postopka, testiranja in interpretacije pridobljenih rezultatov.

Klasifikacijske metode, ki se bodo uporabljale, je potrebno natančno poznati in razumeti. Zato je vsaka od metod tu podrobno opisana. Vedeti je treba, katere so značilnosti posamezne metode, kaj je za posamezno metodo pomembno in katere so njene prednosti in pomanjkljivosti. Pri vsaki metodi je predstavljeno tudi, s kakšnimi parametri bomo metodo uporabili za testiranje. Od parametrov, ki jih nastavimo pri metodah, so namreč odvisni rezultati.

Za vsak sklop podatkov, na katerem bomo testirali klasifikacijske metode, bomo na koncu razporedili metode od najbolj do najmanj učinkovite glede na vrednost AUC (Area under ROC curve), ki je mera ki se navadno uporablja [1, 5]. Najboljša metoda bo dobila rang 1, druga rang 2 itd. Na koncu bomo za vsako metodo izrisali histogram vrednosti AUC in izračunali povprečno vrednost AUC za celotne podatke ter izrisali graf kritičnih razdalj, ki prikaže povprečne vrednosti rangov za posamezne metode in kritične razdalje med metodami.

Glede na graf rangov in na povprečno vrednost AUC bomo na koncu ocenili, katera tehnika je bila na danih podatkih najbolj uspešna. Pridobljene rezultate testiranja bomo nato primerjali z že znanimi rezultati iz nekaterih raziskav, ki

so javno objavljene. V primerjavi bomo poskusili razložiti, zakaj so rezultati podobni, oziroma kaj so razlogi, da se rezultati razlikujejo.

2.2 Pregled stanja na področju bioinformatike

V zadnjih letih potekajo številne raziskave na področju diagnosticiranja na osnovi podatkov o genskih izrazih z namenom, da bi dobili vpogled v mehanizme zapletenih celičnih dogajanj pri kompleksnih boleznih, med katere sodijo tudi rakava obolenja [10]. Namen raziskav je tudi pridobivanje znanja za razvoj ciljnih zdravil brez nepotrebnih stranskih učinkov in možnost prilagajanja zdravljenja glede na genetske posebnosti posameznika.

V raziskave in v pridobivanje primernih podatkov za analizo oz. obdelavo se vsako leto vlaga na desetine miljonov dolarjev [11]. Razlog, zakaj je pridobivanje novih podatkov tako drago, je v tem, da morajo biti podatki, ki jih pridobijo, izjemno visoke kvalitete, primerno označeni, imeti morajo informacijo o nadaljnem medicinskem poteku ter privoljenje pacienta, da se lahko njegovi podatki uporabijo v raziskovalne namene. Zaradi naštetih problemov in zaradi zagotovitve obvezne anonimnosti množina podatkov z vidika števila zbranih in obdelanih vzorcev ni velika.

Poleg problema z zaenkrat še premajhno količino podatkov za obdelavo pa se pojavljajo tudi problemi, povezani s kvaliteto teh podatkov. Genomski podatki so tipično pridobljeni s tehnologijo, s katero so meritve neprecizne, šumne. Veliko raziskovalcev tipično nima dovolj znanja za analitične korake, ki so potrebni, da se pretvori desettisoče podatkovnih točk s šumom, ki jih pridobijo z metodo pridobivanja podatkov o genskih izrazih z mikročipi, v zanesljive in uporabne informacije. Zaradi tega je potrebno pri obdelavi sodelovati z izkušenimi biostatistiki, ki pri tem postopku pomagajo.

Dodaten problem, ki se pojavi pri sami analizi podatkov, izhaja iz dejstva, da neka povprečna meritev vsebuje med 5000 in 50000 izmerjenih točk, ki so interpretirane kot spremenljivke pri klasifikaciji oz. določanju razreda, v katerega spada primer. Tako je v neki množici podatkov število spremenljivk genskih meritev veliko večje, kot je število primerov, iz katerih želimo narediti napoved, kar naredi mnogo klasifikacijskih metod neuporabnih.

Ker je raziskovanje oziroma ugotavljanje, katera klasifikacijska metoda je najboljša na taki vrsti podatkov, zelo aktualno, je tudi različnih študij s tega področja zelo veliko [5]. V teh študijah se po večini izkaže, da sta najbolj učinkoviti metoda podpornih vektorjev (angl. Support vector machines, SVM) in pa metoda naključnih gozdov [3, 5, 10]. Metoda SVM je pravzaprav

največkrat omenjena kot najboljša, vendar pa je treba upoštevati, da se metoda naključnih gozdov manjkrat pojavlja v testiranjih. Testiranja se izvajajo na zelo različnih podatkih in z različnimi nastavitvami parametrov klasifikacijskih metod. Večje razlike, do katerih prihaja med rezultati različnih študij, se pojavljajo zaradi načina obravnave velikega števila spremenljivk (izbira podmnožice pomembnih atributov ali ne), zaradi uporabe različnih strategij za izračun točnosti (prečno preverjanje, angl. cross-validation, naključni izbor primerov za učenje, angl. bootstrap...), zaradi pristranskega testiranja in zaradi različnosti podatkov, kar tudi zelo vpliva na rezultate. Nekatere objave so najbrž naklonjene prikazu moči določene metode in zaradi tega so primeri, na katerih se testira, izbrani tako, da ta metoda dosega boljše rezultate, kot bi jih dosegli na nevtralnih podatkih. Drugi znak, po katerem lahko sklepamo, da je testiranje pristransko, pa je, da parametri, ki so bili uporabljeni pri določenih metodah, niso predstavljeni. Zaradi tega je praktično nemogoče primerjati vsa testiranja med seboj ter tako dobiti neko zanesljivo oceno, katera metoda je najbolj uspešna.

V nalogi smo zato skušali primerjati metode strojnega učenja na velikem naboru podatkov in se na ta način vsaj deloma izogniti pristranskosti. Znanje in pridobljene izkušnje bodo koristne tudi za morebitno nadaljnjo bolj podrobno analizo z večimi metodami, različnimi nastavitvami parametrov teh metod in na večjem številu primerov.

2.3 Predstavitev podatkov

2.3.1 Izvor podatkov

Podatke o genskih izrazih, ki smo jih uporabill za testiranje, smo dobili od Nacionalnega centra za biotehnološke informacije, napredno znanost in zdravje (National Center for Biotechnology Information advances science and health (NCBI)), ki na svojih internetnih straneh nudi možnost dostopa do javnih baz biomedicinskih in genskih podatkov. Skladišče množic podatkov, ki smo ga uporabili, se imenuje Gene Expression Omnibus (GEO) [12]. GEO je javno skladišče, ki shranjuje in omogoča brezplačen dostop do podatkov, pridobljenih z metodo pridobivanja podatkov o genskih izrazih z mikročipi.

Na računalnik smo preko dodatka (add-on) Bioinformatics za program Orange [4] prenesli modul obiGEO. Ta modul priskrbi vmesnik za dostop do GEO skladišča. Trenutno vsebuje razreda GDSInfo in GDS. GDSInfo se uporablja za dostop do informacij o GEO množicah podatkov. Razred GDS pa zagotavlja metode za prenos podatkov za določeno GEO množico podatkov.

2.3.2 Metoda pridobivanja podatkov z mikročipi

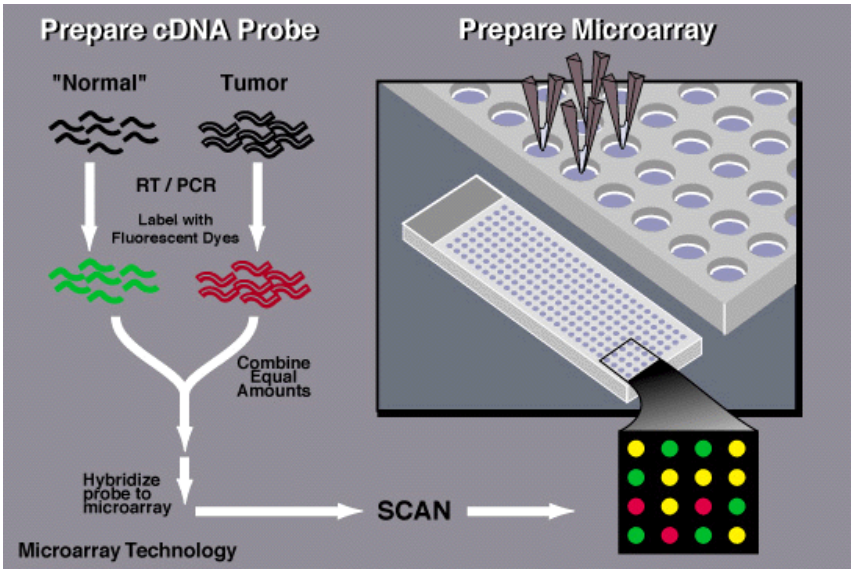
Z metodo pridobivanja podatkov o genskih izrazih z mikročipi (angl. microarray), ki spada med metode naslednje generacije vzorčenja (angl. next-generation sequencing), se pridobi velika večina vseh podatkov, ki se uporabljajo na področju bioinformatike genskih izrazov. Tako so bili tudi podatki, ki smo jih uporabili za diplomsko nalogo, pridobljeni s to metodo. Z njo lahko raziskovalci naenkrat hitro in učinkovito pridobijo podatke o genskih izrazih desettisočev genov [9].

Za merjenje genskih izrazov z mikročipi potrebujemo celice tkiva, ki ga bomo testirali, in pa kontrolne (zdrave) celice. Iz celic izoliramo informacijski RNK (mRNK), ker mRNK vsebuje informacijo o tem, kateri geni so v tej celici aktivni. Ta mRNK nato uporabimo kot osnovo za izdelavo komplementarne DNK (cDNK), ki ji nato dodamo fluorescentno barvo, da lahko ločimo testni vzorec od kontrolnega vzorca. Z zeleno barvo označimo genski material v kontrolnih celicah in z rdečo barvo v testnih celicah. Vzorca nato premešamo in nanesemo na mikročip.

Mikročip je steklena, plastična ali silicijeva ploščica, ki ima nekaj tisoč mikroskopsko majhnih jamic. V jamice so sintetizirani deli DNK molekul tako, da se bo na vsak del vezal točno določen gen oziroma cDNK, ki ima ta gen izražen. Molekule so razporejene v jamice tako, da se za vsako jamico ve, kateri gen se bo vezal nanjo in tako nam določena jamica na nek način predstavlja določen gen.

Po združitvi obarvane zmesi testnega in kontrolnega vzorca z mikročipom se ustvarijo vezi med molekulami cDNK in molekulami, ki so bile že prej na mikročipu. Molekule, ki so na mikročipu sintetizirane od začetka, so na mikročip tudi pritrjene. To je pomembno zato, ker po tem, ko vzorec združimo z mikročipom, mikročip speremo. Na ta način odstranimo nevezane molekule.

Sedaj vstavimo mikročip v čitalec - temen prostor, kjer se s posebnim laserjem odčita barvo za vsako jamico na mikročipu. Glede na to, na katerih mestih in s kakšno barvo so se obarvale jamice na mikročipu, lahko ugotovimo, kateri geni so izraženi pri testni in kateri pri kontrolni celici. Rumena barva predstavlja kombinacijo testne in kontrolne DNA. To pomeni, da sta gena v obeh celicah enako izražena. Glede na to, kako močna je barva, lahko ugotovimo tudi stopnjo izraženosti določenega gena. Poenostavljena slika pridobivanja podatkov o genskih izrazih z mikročipi je prikazana na sliki 2.1.



Slika 2.1: *Postopek pridobivanja podatkov o genskih izrazih z mikročipi*

2.3.3 Uporabljeni podatki

Podatkovno skladišče GEO, iz katerega smo črpali podatke, je v času črpanja podatkov vsebovalo 2008 različnih množic podatkov. Nekatere od teh množic podatkov so imele več skupin različnih tipov vzorcev. Izmed vseh 2008 množic podatkov smo za testiranje izbrali tiste skupine, ki so imele vsaj 2 razreda, s po vsaj 10 vzorci. Takih množic je bilo 227 s skupno 555 skupinami. Za tako omejitev smo se odločili, ker je od števila primerov (vzorcev) odvisna zanesljivost klasifikacijske napovedi. Več kot imamo primerov, bolj zanesljiv je rezultat, ki ga vrne klasifikacijska metoda. Ker bi nam skupine z manj kot 10 primeri lahko pokvarile povprečno klasifikacijsko točnost, smo take skupine raje izpustili iz testiranja.

V tabeli 2.1 je prikazanih nekaj množic podatkov, ki smo jih uporabili. Celoten nabor vključno z vsemi podrobnostmi je viden na spletnem naslovu (*http://www.ncbi.nlm.nih.gov/sites/GDSbrowser*). V tabeli je prikazana oznaka množice podatkov, kratek opis množice, število primerov (vzorcev), ki jih množica podatkov vsebuje, število atributov in število razredov.

Oznaka	**Kratek opis v angleščini**	**Št. primerov**	**Št. atributov**	**Št. razredov**
GDS997	Transcription factor CHOP null mutation effect on fibroblasts subjected to ER stress: time course	24	12488	7
GDS968	Radiation therapy toxicity association with abnormal transcriptional response to DNA damage	171	12651	64
GDS963	Macular degeneration and dermal fibroblast response to sublethal oxidative stress	36	12651	2
GDS960	Estrogen effect on lung: time course	28	22690	6
GDS946	Familial combined hyperlipidemia	24	22283	2
GDS887	Diabetic nephropathy: models of type 1 and type 2 diabetes	50	9568	8
GDS879	X-linked hypophosphatemia and low phosphate diet	20	12488	8
GDS874	Spinal cord injury and regeneration time course: above T9 (RG-U34C)	48	8789	10
GDS873	Spinal cord injury and regeneration time course: above T9 (RG-U34B)	47	8791	10
GDS872	Spinal cord injury and regeneration time course: above T9 (RG-U34A)	52	8799	11
GDS871	Spinal cord injury and regeneration time course: below T9 (RG-U34C)	43	8789	9
GDS870	Spinal cord injury and regeneration time course: below T9 (RG-U34B)	52	8791	10
GDS869	Spinal cord injury and regeneration time course: below T9 (RG-U34A)	55	8799	11
GDS868	MPSS transcriptome analysis project	87	71752	53
GDS843	Adult acute myeloid leukemia: bone marrow and peripheral blood expression profiles (SHDJ)	49	43008	7
GDS841	Adult acute myeloid leukemia: bone marrow and peripheral blood expression profiles (SHCO)	26	43008	6
GDS773	Retinoic acid teratogenic effect on cranial neural crest: time course	27	12488	6
GDS724	Kidney transplant rejection expression profiling	62	12651	6

Tabela 2.1: Tabela množic podatkov

Poglavje 3

Predstavitev uporabljenih metod strojnega učenja

3.1 Metoda bližnjih sosedov (k-NN)

Za metodo najbližjih sosedov, ki velja za najpreprostejšo metodo v strojnem učenju, je značilno, da spada med t.i. "lene klasifikatorje", ker tu učenja skorajda ni. Z drugimi besedami to pomeni, da klasifikator v fazi učenja na učni množici ne zgradi učnega modela (npr. drevesa), ampak da vsakič znova uporabi učne podatke pri napovedovanju vrednosti za nov primer. Metoda enostavno shrani podmnožico ali pa vse učne primere in ko je potrebno klasificirati nov primer, se poišče podmnožica podobnih učnih primerov, ki se uporabijo za napoved razreda novega primera. Zaradi takšnega načina je glavnina procesiranja potrebna pri klasifikaciji novega primera in ne pri grajenju učnega modela. Zaradi tega je časovna zahtevnost klasifikacije novega primera precej večja kot pri drugih metodah učenja, kar je lahko še posebej problematično pri velikih učnih množicah, saj mora algoritem preiskati celotno množico za vsak nov (testni) primer.

V našem primeru smo uporabili navadno metodo k-najbližjih sosedov ali na kratko kNN (angl. k-nearest neighbors). Pri tej varianti algoritma shranimo vse učne primere. Ko želimo napovedati razred r_x novemu primeru u_x, poiščemo med učnimi primeri k najbližjih $u_1, \ldots, u_k$ primerov in pri klasifikaciji napovemo večinski razred, t.j. razred, ki mu pripada največ izmed k najbližjih sosedov.
Formula za izračun razreda r_x novemu primeru:

$$r_x = argmax_{r \in \{V_1,\ldots,V_{n_0}\}} \sum_{i=1}^{k} \delta(r, r^{(i)})$$

kjer je

$$\delta(a,b) = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases}$$

Za parameter k običajno vzamemo neko liho število, npr. 1, 7, 15, 31, ker se s tem izognemo neodločenosti pri glasovanjih pri klasifikacijskih problemih z dvema razredoma. Za to, kakšen k bomo vzeli, se odločimo na podlagi stopnje šuma v učnih podatkih. Če v učnih primerih napak ni, potem se bo najbolje obnesel algoritem 1-NN. Če so v učnih primerih napake, lahko s povečevanjem parametra k povprečimo napovedi več bližnjih primerov in s tem zmanjšamo verjetnost, da je k učnih primerov napačnih. Negativna posledica povečevanja atributa k je, da h klasifikaciji prispevajo tudi tisti učni primeri, ki niso dovolj podobni novemu primeru. Zaradi tega je potrebno za vsak problem posebej eksperimentalno določiti optimalni k.

Pomembno je vedeti tudi, da sama izbira parametra k ne določa velikosti okolice novega primera, znotraj katere izbiramo učne primere, ampak se okolica dinamično spreminja glede na gostoto učnih primerov v danem prostoru primerov. S fiksno velikostjo okolice novega primera bi v nekaterih delih prostora lahko dobili preveč bližnjih sosedov in v drugih delih nobenega. Ker gostota učnih primerov služi za oceno dejanske verjetnostne gostote v prostoru primerov, s parametrom k algoritem elegantno reši problem gostejših oziroma redkejših delov prostora.

Osrednji del algoritma metode k-NN je določitev, kateri primeri so najbližji novemu primeru. Poudariti je potrebno, da je algoritem zelo občutljiv na izbrano metriko, tako da lahko sprememba metrike zelo hitro spremeni razred, v katerega se klasificira nov primer. Ponavadi se za metriko uporabi evklidska razdalja. Vsi zvezni atributi se normalizirajo na interval $[0, 1]$ in razdalja med dvema vrednostima je enaka absolutni razliki med njima. Za diskretne atribute pa je razdalja med različnima vrednostima 1, med enakima pa 0. Če za nek atribut ne obstaja vrednost, potem se za razdaljo med atributom z znano in atributom z neznano vrednostjo oziroma dvema atributoma z neznano vrednostjo, vzame razdalja 0,5.

Torej je razdalja med dvema primeroma podana z:

$$D(u_l, u_j) = \sqrt{\sum_{i=1}^{a} d(v^{(i,l)}, v^{(i,j)})^2}$$

kjer za zvezni atribut A_i velja:

$$d(v^{(i,l)}, v^{(i,j)}) = |v^{(i,l)}, v^{(i,j)}|$$

In za diskretnega:

$$d(v^{(i,l)}, v^{(i,j)}) = \begin{cases} 0, & v^{(i,l)} = v^{(i,j)} \\ 1, & v^{(i,l)} \neq v^{(i,j)} \end{cases}$$

Za testiranje smo v Pythonu uporabili algoritem k-NN s privzetimi lastnostmi (atributi). To pomeni, da je bil atribut $k = 10$, da nismo uporabili obteževanja primerov in da je bila za metriko izbrana Evklidska razdalja.

Inicializacija algoritma k-NN, kot smo ga uporabili pri testiranju:

$knn = orange.kNNLearner(name = \text{“}kNN\text{”})$

3.2 Naivni Bayesov klasifikator

Naivni bayesov klasifikator je najpreprostejši tip Bayesovega klasifikatorja. Temelji na predpostavki, da so vrednosti različnih atributov pri danem razredu neodvisne. Kljub temu, da je ta predpostavka v realnih primerih velikokrat prekršena, se klasifikator še vedno dobro odreže pri mnogih primerih. Formulo naivnega Bayesovega klasifikatorja izpeljemo s pomočjo Bayesovega pravila in je enaka:

$$P(r_k|V) = P(r_k) \prod_{i=1}^{a} \frac{P(r_k|v_i)}{P(r_k)}$$

Naloga učnega algoritma je s pomočjo učne množice podatkov aproksimirati apriorne verjetnosti razredov $P(r_k)$, $k = 1, \ldots, n_0$ in pogojne verjetnosti razredov r_k, $k = 1, \ldots, n_0$ pri dani vrednosti v_i atributa $A_i, i = 1 \ldots a$: $P(r_k|v_i)$. Za odkrivanje apriornih verjetnosti se uporablja formula:

$$P(r_k) = \frac{N_k}{N}$$

kjer je N_k število učnih primerov iz razreda r_k in N število vseh učnih primerov. Za ocenjevanje pogojnih verjetnosti se uporablja formula:

$$P(r_k|v_i) = \frac{N_{k,i}}{N_i}$$

kjer je $N_{k,i}$ število učnih primerov iz razreda r_k in z vrednostjo i-tega atributa v_i ter N_i število vseh učnih primerov z vrednostjo i-tega atributa v_i. Dobra lastnost naivnega Bayesovega klasifikatorja je, da pri učenju (izračunu pogojnih verjetnosti) lahko atribute, ki nimajo podane vrednosti, izpustimo iz formule. Naivni Bayesov klasifikator pri klasifikaciji uporabi le atribute, katerih vrednosti so podane za dani primer, ki ga klasificiramo. Atribute, katerih vrednosti za dani primer ne poznamo, preprosto ignoriramo.

Pri diskretnih atributih lahko naivni Bayesov klasifikator uporabljamo neposredno, pri zveznih atributih pa je potrebno atribut najprej diskretizirati. V našem primeru smo diskretizirali zvezne atribute tako, da smo jih razdelili na 5 enako gosto razporejenih intervalov. To pomeni, da se interval, na katerem se nahajajo vrednosti zveznega atributa, ne razdeli na enako dolge odseke, ampak da se razdeli na odseke tako, da je na vsakem odseku enako število vrednosti.

V praksi je naivni Bayesov klasifikator kljub naivnosti zelo uspešen. Izkaže se, da je pogojna neodvisnost pogosto sprejemljiva predpostavka. Poleg tega je ocenjevanje verjetnosti relativno zanesljivo in zato ne pride do prevelikega prilagajanja učni množici. Tretji razlog pa je v tem, da ima, kadar predpostavka o pogojni neodvisnosti ni popolnoma izpolnjena, naivni Bayes še vedno dovolj "rezerve". Razponi ocen verjetnosti med najbolj verjetnim in ostalimi razredi so namreč ponavadi dovolj veliki, da napaka zaradi predpostavke o neodvisnosti ne uspe "pokvariti" vrstnega reda verjetnosti razredov.

Zaradi zgoraj naštetih razlogov se naivni Bayesov klasifikator pogosto obnaša zelo dobro tudi na problemih, kjer predpostavka o pogojni neodvisnosti atributov ne drži popolnoma. Če pa obstajajo močne odvisnosti med atributi, pa naivni Bayesov klasifikator odpove. V takem primeru pa je bolje uporabiti kakšen drug klasifikator.

Za testiranje smo v programskem jeziku Python uporabili naslednje metode:

Inicializacija naivnega Bayesovega klasifikatorja:
$bayes = orange.BayesLearner(name = \text{“}Bayes\text{”})$

Klic metode za diskretizacijo podatkov:
$orange.Preprocessor\_discretize(data, method =$
$= orange.EquiNDiscretization(numberOfIntervals = 5))$

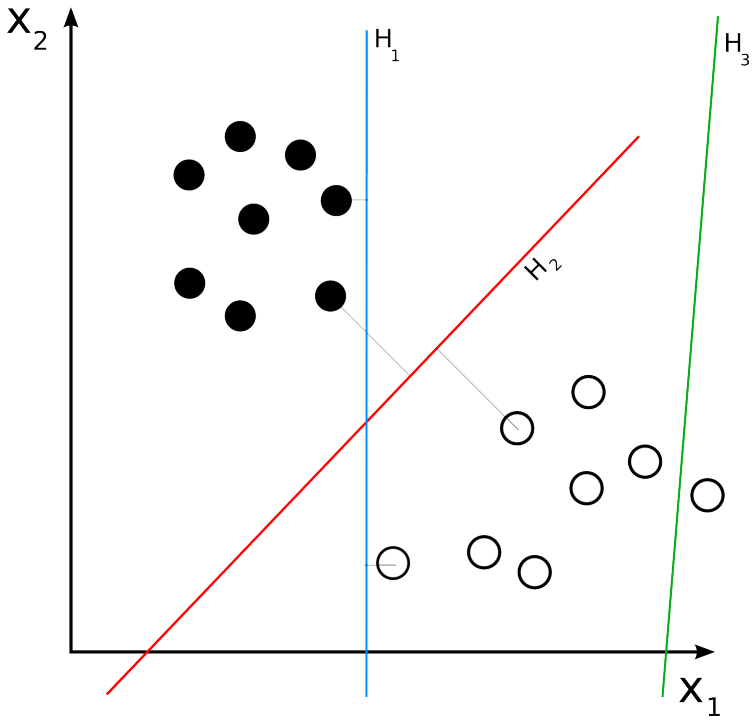
3.3 Metoda podpornih vektorjev (SVM)

Metode podpornih vektorjev spadajo med najbolj uspešne metode za klasifikacijo in regresijo. Pri tej metodi algoritem na danih testnih podatkih, ki pripadajo enemu od dveh razredov, naredi model, na podlagi katerega nato skuša nove primere pravilno uvrstiti v enega izmed dveh razredov. Vendar pa to naredi na drugačen način kot večina ostalih algoritmov, ki težijo k temu, da minimizirajo število atributov, tako da poiščejo ustrezno podmnožico pomembnih atributov, nad katerimi potem zgradijo neko ne preveč kompleksno funkcijo. Pri metodi SVM pa uporabimo čimveč razpoložljivih atributov, ki niso vsi zelo pomembni in ki jih z linearno kombinacijo lahko uporabimo za napovedovanje odvisne spremenljivke. Torej je pri metodi SVM pomembno predvsem to, na kakšen način pametno kombinirati atribute. Izbira atributov je precej manj pomembna, saj bo sama metoda z ustrezno kombinacijo izluščila želeno informacijo.

Metode SVM so primerne za učenje na množicah z velikim številom primerov, opisanih z velikim številom manj pomembnih atributov. Pozitivna stran SVM metode je, da dosega visoko točnost napovedi, slaba pa, da je interpretacija naučenega težavna, prav tako pa tudi razlaga posamezne odločitve.

Kot smo omenili zgoraj, je metoda SVM namenjena razločevanju dveh razredov med seboj. V primeru, da imamo več razredov, postopek ponovimo za vsak razred, ki ga skušamo ločiti od ostalih.

Za opis metode izhajamo iz tega, da imamo množico učnih primerov, za katere je znano, kateremu razredu pripadajo. Vsak primer predstavimo z vektorjem v vektorskem prostoru. Velikost vektorja je enaka številu atributov (n), ki jih imajo primeri. Naloga metode SVM je v tem n dimenzionalnem prostoru poiskati optimalno hiperravnino (hiperploskev), ki ločuje primere iz različnih razredov. Najbližjim primerom optimalne hiperravnine pravimo podporni vektorji, razdalji hiperravnine od podpornih vektorjev pa rob (angl. margin). Torej je optimalna hiperravnina tista, ki ima maksimalni rob, oz. z drugimi besedami, je optimalna hiperravnina tista, pri kateri je vsota oddaljenosti podpornih vektorjev maksimalna. Za optimalno hiperravnino je značilna tudi enaka oddaljenost do najbližjih primerov iz obeh razredov. To široko prazno območje med razredi nam kasneje omogoča, da lahko čim bolj zanesljivo razvrščamo tudi primere, ki niso čisto enaki učnim primerom. Pri postavljanju hiperravnine ni potrebno upoštevati vseh učnih vektorjev. Vektorji, ki so daleč od hiperravnine oz. so skriti za fronto ostalih, ne vplivajo na lego hiperravnine. Torej je lega hiperravnine odvisna le od podpornih vektorjev. Slika 3.1 prikazuje prostor učnih primerov. Vsaka točka prikazuje en učni primerek. Če



Slika 3.1: *Hiperravnina H3 sploh ne razdeli podatkov na dva razreda. Hiperravnina H1 jih razdeli, vendar pa z manjšim robom, kot jih razdeli hiperravnina H2. Iz tega sledi, da je hiperravnina H2 optimalna.*

primer spada v prvi razred, potem je označen s polnim krogom, če pa primer pripada drugemu razredu, je označen s praznim krogom.

Hiperravnine ni možno "zviti", zato je možno izvesti čisto ločitev razredov (kot jo zgoraj prikažeta H2 in H3) le v primeru, ko so učni primeri linearno ločljivi. V vsakdanji uporabi se pogosto zgodi, da v originalnem prostoru z učnimi primeri linearna hiperravnina ne zadošča za sprejemljivo klasifikacijsko točnost.

Ko vektorji niso linearno ločljivi, bi lahko uporabili zvijačo. Vektorje bi eksplicitno transformirali tako, da bi jim povečali dimenzijo (povečali bi število atributov). Če dimenzijo dovolj povečamo, postanejo vsi razredi vektorjev linearno ločljivi. Vendar pa pri tem nastaneta dve težavi. Transformiranje vektorjev v prostor z višjo dimenzijo je računsko zahtevna operacija in inverzna transformacija hiperravnine iz prostora z višjo dimenzijo v prostor z nižjo dimenzijo spremeni hiperravnino v zelo zapleteno. Problemu prevelikega števila atributov pravimo tudi "prekletstvo dimenzionalnosti".

Kot je opisano zgoraj, lahko z nelinearno transformacijo postane prostor

primeren za linearno ločitveno hiperravnino. Na ta način lahko z različnimi transformacijami rešujemo različne nelinearne probleme. Prava moč metode SVM je v tem, da če za transformacijo uporabi posebne funkcije, ki se jim reče jedro, se metoda lahko izogne težavam s prevelikim številom atributov. Transformacijo in inverzno transformacijo lahko opravimo brez tega, da bi jo dejansko računsko izvedli na vseh vektorjih. Zadošča uporaba le enega dela učnih vektorjev (podporni vektorji) za popolni opis meje med razredi.

Z različnimi jedrnimi funkcijami (K) dobimo različne transformacije prostora atributov in s tem različne variante metode SVM. Primeri jedrnih funkcij so:

- Linearna: v tem primeru ohranimo originalni atributni prostor:

$$K(u_j, u) = u.u_j$$

- Polinomska: za dano stopnjo polinoma d lahko uporabimo naslednjo funkcijo za konvolucijo skalarnega produkta:

$$K(u_j, u) = [(u.u_j) + 1]^d$$

- Radialna: za izbrano vrednost parametra γ je konvolucija podana z:

$$K(u_j, u) = e^{-\gamma|u-u_j|^2}$$

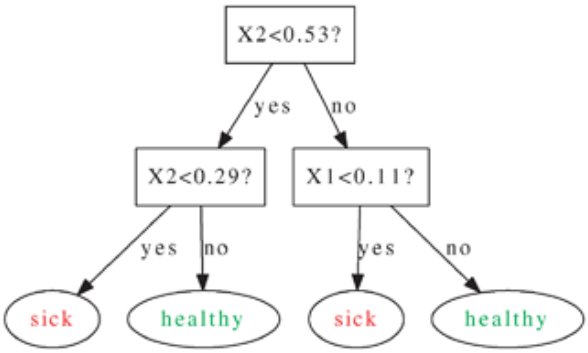
- Sigmoidna: pri dani sigmoidni funkciji S (npr $S = tangh$) dobim jedrno funkcijo za določene vrednosti parametrov v in c:

$$K(u_j, u) = S[v(u.u_j) + c]$$

3.4 Odločitvena drevesa

Pri metodi odločitvenih dreves moramo za potrebe klasifikacije najprej zgraditi odločitveno drevo. Odločitveno drevo zgradimo na osnovi učnih primerov. Vsak primer je opisan z vektorjem vrednosti atributov in z razredom, ki mu pripada. Atributi so lahko zvezni ali diskretni.

Odločitveno drevo je sestavljeno iz notranjih vozlišč, ki ustrezajo atributom, vej, ki ustrezajo podmnožicam vrednosti atributov, in listov, ki ustrezajo razredom. Ena pot v drevesu od korena do lista ustreza enemu odločitvenemu



Slika 3.2: *Primer klasifikacijskega drevesa.*

pravilu. Pri tem so pogoji (pari atribut - podmnožica vrednosti), ki jih srečamo na poti, konjunktivno povezani. Primer enostavnega odločitvenega pravila: "Če parameter X2 ni manjši od 0.53 in parameter X1 ni manjši od 0.11, potem je oseba, ki ji pripadajo ti podatki, zdrava."
Primer enostavnega odločitvenega drevesa, je prikazan na sliki 3.2

Osnovni algoritem učenja odločitvenih dreves po [6] je naslednji:
Imamo učno množico S.

1. Če je izpolnjen ustavitveni pogoj, potem je množica S list v našem drevesu
2. sicer
   (a) izberi najboljši atribut Ai
   (b) najboljši atribut postane vozlišče
   (c) množico S razdelimo na podmnožice Si glede na vrednosti atributa Ai
3. rekurzivno ponovi koraka 1-2 za vse podmnožice Si

Pri tem je ustavitveni pogoj lahko:

- dovolj "čista" učna množica (npr. vsi ali večina primerov pripada istemu razredu),
- premalo učnih primerov za zanesljivo nadaljevanje gradnje drevesa,
- odsotnost (dobrih) atributov

Ključnega pomena pa je vsekakor izbira "najboljšega atributa". Za izbiro atributa se najpogosteje uporabljajo mere: informacijski prispevek, razmerje informacijskega prispevka, Gini-indeks in RelieF [6]. Ko drevo zgradimo na učnih podatkih, ga nato lahko uporabljamo za klasifikacijo novih primerov. Pri klasifikaciji potujemo od korena drevesa po ustreznih vejah do lista. List vsebuje informacijo o številu učnih primerov, ki pripadajo posameznim razredom. Iz frekvenc učnih primerov se oceni verjetnostna porazdelitev razredov. Število učnih primerov pa kaže na zanesljivost te ocene. Poseben primer je prazen list, ki ne ustreza nobenemu učnemu primeru in zato ne omogoča nobene ocene verjetnosti posameznih razredov. V takem primeru ponavadi uporabimo naivni Bayesov klasifikator, ki nam vseeno vrne neko naivno verjetnost posameznih razredov.

Rezanje drevesa

Zaradi nezanesljivosti napovedovanja na nižjih nivojih drevesa, kjer vozliščem ustreza majhno število učnih primerov, se posveča tem vozliščem posebna pozornost. Ustavitveni pogoji poskušajo ustaviti gradnjo drevesa, ko le-ta postane nezanesljiva ali pa nepotrebna. Ker pa je to težko vnaprej napovedati, se pogosto gradnja nadaljuje ne glede na zanesljivost atributa in se drevo naknadno poreže. Osnovni algoritem naknadnega rezanja je naslednji:
Za vsa notranja vozlišča od spodaj navzgor je potrebno:

- oceniti povprečno pričakovano napako klasifikacije v poddrevesih
- oceniti pričakovano napako klasifikacije v trenutnem vozlišču
- če je povprečna pričakovana napaka poddreves večja od pričakovane napake vozlišča, se poreže poddrevesa in spremeni vozlišče v list.

Za ocenjevanje napake uporabljamo m-oceno verjetnosti [6].
Pri testiranju smo uporabili drevo, ki smo ga gradili tako, da je bilo minimalno število primerov v listih enako 2. Za ocenjevanje napake smo uporabili $m = 2$. Za izbiro "najboljšega" atributa smo uporabili mero informacijski prispevek.

$orngTree.TreeLearner(mForPruning = 2, minSubset = 2)$

3.5 Metoda naključnih gozdov

Metoda naključnih gozdov je metoda, ki je sestavljena iz veliko odločitvenih dreves [6]. Namenjena je izboljšanju napovedne točnosti drevesnih algoritmov.

Ideja je generirati zaporedje odločitvenih dreves tako, da se pri izbiri najboljšega atributa v vsakem vozlišču naključno izbere relativno majhno število atributov, ki vstopajo v izbor za najboljši atribut. Število tako zgrajenih dreves je ponavadi 100, lahko tudi več.

Drevesa, zgrajena na učni množici, se zatem uporabijo za klasifikacijo novega primera po metodi glasovanja. Vsako drevo ima en glas, ki ga nameni razredu, v katerega bi to drevo klasificiralo nov primer. Iz vseh glasov dobimo verjetnostno distribucijo po vseh razredih.

Metoda naključnih gozdov je robustna, saj zmanjša varianco drevesnih algoritmov. S to metodo dosežemo zelo visoko stopnjo klasifikacije. Slabe strani metode pa so, da je razlaga odločitev otežena, saj je množica dreves nepregledna, in da je metoda zelo dovzetna za preveliko prilagajanje podatkom, kar še posebno ob prisotnosti šuma zelo pokvari natančnost klasifikacije. Pri testiranju smo uporabili metodo, ki uporabi 100 dreves:

$$orngEnsemble.RandomForestLearner(trees = 100)$$

3.6 Večinski klasifikator

Za to metodo je značilno, da nov (testni) primer vedno klasificira v večinski razred. To pomeni, da primer uvrsti v razred, kateremu pripada največ učnih primerov. Če so učni primeri razporejeno enakomerno med vse razrede, potem se ta metoda odreže najslabše. V primeru, da imamo dva razreda z enakim številom primerov, potem lahko pričakujemo, da bo ta metoda imela 50% klasifikacijsko točnost.

Ta metoda se običajno uporablja kot kontrolna metoda. Ta metoda bi seveda morala doseči najslabše klasifikacijske rezultate od vseh metod, saj pravzaprav ne zgradi modela iz učnih primerov oz. je njena napoved neodvisna od vrednosti atributov testnega primera.

Poglavje 4

Testiranje in rezultati testiranja

4.1 Potek dela

Preden smo lahko začeli s testiranjem, je bilo najprej potrebno namestiti programski jezik Python in program Orange Canvas (odprtokodni program za vizualizacijo in analizo podatkov). Za Orange je bilo potrebno namestiti še dodatek za Bioinformatics, v katerem smo preko gradnika (widget) "GEO Dataset" lahko dostopali do Gene Expression Omnibus množic podatkov.

Nato smo se v Pythonu lotili pisanja skripte, ki bo izvedla potrebno testiranje klasifikacijskih metod, ki ga želimo izvesti. V skripti smo najprej definirali klasifikacijske metode (in njihove parametre), ki smo jih kasneje uporabili za testiranje. Nato smo se lotili pisanja glavne zanke, ki poteka čez celotno bazo podatkov, ki smo jo uporabili, torej čez vseh 2008 množic podatkov. Za vsako množico podatkov smo nato za vsak "sample type" preverili, ali vsebuje vsaj dva razreda, ki vsebujeta vsak vsaj deset primerov. "Sample type" s pripadajočimi primeri, ki je zadoščal pogojem, je prišel v poštev za testiranje. Ta pogoj smo postavili zato, ker manj kot imamo primerov, na katerih testiramo, manj natančno je testiranje klasifikacijskih metod na teh podatkih. In ker bi netočni rezultati lahko pokvarili skupne rezultate, smo take primere izpustili. Nato smo na primernih "sample type" izvedli testiranje.

Testiranje smo izvedli na vseh sedmih opisanih klasifikacijskih metodah. Pri testiranju smo morali biti pozorni, saj klasifikacijska metoda naivni Bayes ne more klasificirati zveznih atributov, ki jih podatki o genskih izrazih vsebujejo. Ostale klasifikacijske metode z zveznimi atributi nimajo problemov, tako da smo podatke diskretizirali le za naivni Bayes klasifikacijsko metodo. Atribute smo diskretizirali na pet intervalov tako, da je bilo na vsakem intervalu enako število vrednosti.

Če smo želeli preveriti učinkovitost klasifikacijske metode, smo morali pri klasificiranju množico primerov razdeliti na učno in testno množico. Na učni množici smo metodo "naučili" in nato njeno učinkovitost preverili na testni množici. Za razdelitev na učno in testno množico smo uporabili metodo bootstrap. Za to metodo je značilno, da iz množice primerov naključno izbira primere, ki bodo v učni množici. Ob vsakem izboru se primer ne izbriše iz začetne množice, tako da imamo lahko v učni množici več enakih primerov. Število primerov, ki jih metoda izbere iz začetne množice, je enaka velikosti množice, ker pomeni, da bi v teoriji lahko vsi primeri bili izbrati v učno množico. V povprečju pa učna množica po izboru vsebuje 63.2% primerov iz začetne množice vseh primerov. V testno množico tako uvrstimo primere, ki niso v učni množici.

Za vsako množico podatkov smo metodo bootstrap uporabili desetkrat in tako desetkrat dobili različno učno in testno množico. Za določitev učinkovitosti klasifikatorja smo uporabili mero Area under ROC Curve (AUC). Iz rezultatov vseh desetih ponovitev smo na koncu izračunali povprečno vrednost AUC.

Tako smo za vsako klasifikacijsko metodo izračunali povprečno vrednost AUC in range. Range smo dobili tako, da smo metode za vsako množico podatkov razvrstili od najboljše do tiste, ki se je pri napovedi rezultatov iskazala kot najslabša. Najboljši metodi smo dodelili rang ena, najslabši metodi pa rang sedem, kar je bilo v našem primeru število različnih klasifikacijskih metod, ki smo jih primerjali. Če sta npr. imeli dve klasifikacijski metodi enako vrednost AUC, sta obe metodi dobili sredinski rang. To pomeni, če sta to bili najboljši metodi, sta obe dobili rang 1,5.

Po izračunu povprečnih vrednosti AUC in rangov za vsako klasifikacijsko metodo na vseh množicah podatkov smo za vsako metodo posebej narisali histogram. Tipične vrednosti za mero AUC so od 0.5 do 1.0. 1.0 pomeni, da je klasifikacijska metoda vse testne primere klasificirala v pravilen razred, 0.5 pa da je bila napoved uvrstitve v pravilen razred 50%, kar je ponavadi značilno za večinski klasifikator. Histogram smo zaradi preglednosti razdelili na dvajset stolpcev, tako da vsak stolpec pokriva 5% možnih vrednosti AUC. Na histogramu smo predstavili razporeditev vrednosti AUC tako, da višina stolpca v histogramu predstavlja pogostost (število) pojavljanja vrednosti AUC pri testiranju te metode. Na ta način se iz histograma lepo vidi, katere vrednosti AUC se pri določeni klasifikacijski metodi največkrat pojavljajo.

Nato smo za vsako klasifikacijsko metodo izračunali povprečen AUC preko vseh množic podatkov. Ta vrednost predstavlja povprečje povprečnih vrednosti AUC, ki smo jih dobili na posameznih množicah podatkov. Ta vrednost nam pove, kakšno natančnost naj bi imela ta metoda pri klasificiranju. Skupaj

z grafom kritičnih razdalj nam ta vrednost da odgovor na to, katera klasifikacijska metoda je najbolj uspešna.

Na koncu smo izrisali še graf kritičnih razdalj. To je graf, ki za vsako klasifikacijsko metodo prikaže povprečni rang. Povprečni rang se izračuna na podlagi vseh rangov, ki smo jih izračunali za vsako množico podatkov posebej. Iz grafa se tako lahko takoj razbere, katera metoda ima najboljši povprečni rang. To z drugimi besedami pomeni, katera metoda je bila v povprečju na množicah podatkov bolj uspešna od ostalih. Tu je treba poudariti, da ima lahko neka metoda najboljši povprečni rang, a to še ne pomeni, da ima tudi najvišjo povprečno vrednost AUC. In seveda tudi obratno. To se zgodi, če je metoda na nekaterih podatkih zelo uspešna, na drugačnih podatkih pa izpade porazno.

Graf kritičnih razdalj [5] pa nam, kot že ime pove, pokaže tudi kritične razdalje. Kritične razdalje so na grafu prikazane z vodoravno odebeljeno črto. Če sta dve klasifikacijski metodi znotraj kritične razdalje, to pomeni, da sta statistično gledano enako uspešni in da je razlika med njima premajhna, da bi eno metodo lahko razglasili za bolj uspešno od druge.

4.2 Rezultati testiranja

Po izvedbi testiranja smo dobili kot rezultatt povprečno vrednost AUC za vsako metodo za vse podatke, histogram za vsako klasifikacijsko metodo in graf kritičnih razdalj.

Povprečne vrednosti AUC za vse klasifikacijske metode smo uredili od najbolj uspešne proti najmanj uspešni. Večja vrednost mere AUC pomeni, da je ta metoda pri napovedovanju bolj uspešna (več primerov uvrsti v pravilni razred). Povprečne vrednosti so izračunane iz 555 različnih AUC vrednosti, ki smo jih pridobili iz primernih skupin podatkov. Pri vsaki metodi smo izračunali tudi standardni odklon.

Povprečne vrednosti AUC in standardni odkloni za posamezne metode:

1. **Metoda naključnih gozdov:**
   AUC = 0.863
   Standardni odklon: 0.164

2. **Metoda SVM z linearnim jedrom (Linear Learner):**

AUC = 0.827
Standardni odklon: 0.190

3. **Metoda SVM z radialnim jedrom:**
   AUC = 0.799
   Standardni odklon = 0.190

4. **Metoda najbližjih sosedov (kNN):**
   AUC = 0.776
   Standardni odklon = 0.200

5. **Metoda odločitvenih dreves:**
   AUC = 0.742
   Standardni odklon = 0.152

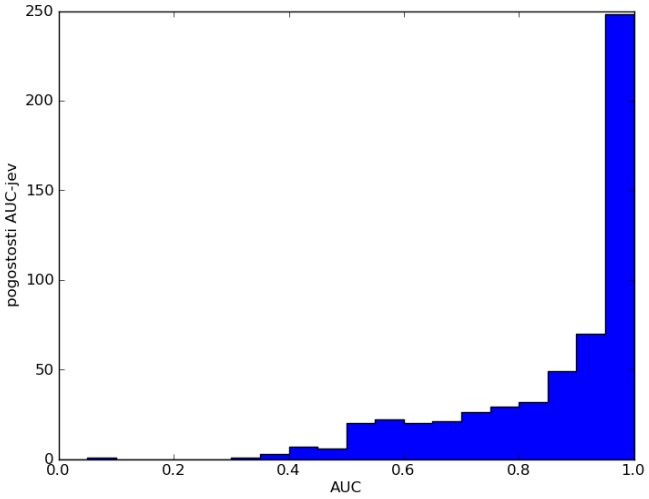
6. **Metoda Naivni Bayes:**
   AUC = 0.624
   Standardni odklon = 0.148

7. **Metoda večinskega klasificiranja:**
   AUC = 0.5
   Standardni odklon = 0.0

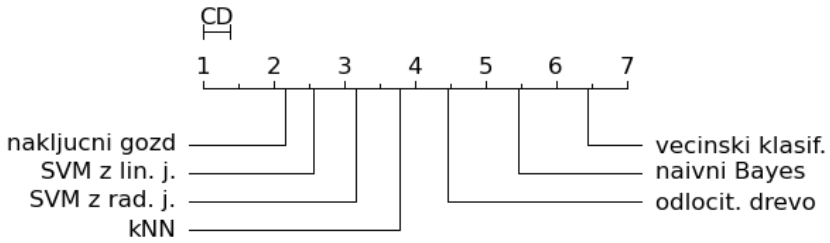
Histogram za klasifikacijsko metodo naključnih gozdov je prikazan na sliki 4.1. Iz histograma se lepo vidi, da je metoda veliko večino vseh množic podatkov klasificirala v celoti pravilno. Število k določeni verjetnosti klasificiranih množic podatkov pa se nato zmanjšuje skupaj z zmanjšanjem verjetnosti pravilne napovedi. Histogrami ostalih klasifikacijskih metod so prikazani v prilogi B.
Graf kritičnih razdalj vseh klasifikacijskih metod je prikazan na sliki 4.2.

4.3 Diskusija

Po opravljenem testiranju na vseh množicah podatkov so rezultati pokazali, da je od vseh uporabljenih klasifikacijskih metod najuspešnejša metoda naključnih gozdov. Ta metoda je imela najvišjo povprečno vrednost AUC (AUC = 0,863), prav tako pa je tudi na grafu kritičnih razdalj, ki prikazuje povprečno vrednost rangov, dosegla najboljši rezultat. Takoj za njo sta po rezultatih obe različici metode SVM. Izmed teh dveh je metoda z linearnim jedrom dosegla boljše rezultate kot metoda z radialnim jedrom. Za metodo SVM z radialnim jedrom



Slika 4.1: *Histogram uspešnosti metode naključnih gozdov*



Slika 4.2: *Graf kriticnih razdalj*

so se nato razvrstile metoda najbližjih sosedov, metoda odločitvenih dreves, naivni Bayes in na koncu pričakovano večinski klasifikator. Večinski klasifikator je dosegel povprečno vrednost AUC = 0.5, kar je bila tudi njegova pričakovana končna vrednost.

Rezultat je kljub temu, da sta zmagovalni metodi blizu skupaj, vseeno malo presenetljiv, saj smo pričakovali, da bo metoda SVM najbolj uspešna. Res pa je, da ja standardni odklon pri obeh zmagovalnih metodah relativno velik, kar nakazuje, da bi se lahko na drugačnih podatkih vrstni red zamenjal.

Metoda naključnih gozdov v literaturi, ki obravnava napovedovanje na podlagi genskih izrazov, ni tako razširjena, kljub temu da ima idealne značilnosti za takšne vrste podatkov, kot smo jih testirali. Uspešna je na podatkih, kjer imamo veliko več spremenljivk kot primerov, še posebej, če podatki vsebujejo

šum [7]. Šum pa se pri podatkih, pridobljenih s pomočjo mikročipov, pojavlja. Metoda je zelo uspešna tudi brez predhodne izbire pomembnih genov, ki sicer zmanjšuje šum v podatkih, kar za nekatere metode ne velja. V našem testiranju ni bila narejena predhodna izbira genov in to je za metodo naključnih gozdov ugodno v primerjavi z ostalimi metodami. Poleg tega je pri metodi naključnih gozdov pozitivno tudi to, da ni potrebno posebno uglaševanje parametrov kot pri nekaterih drugih metodah, kjer s spreminjanjem vrednosti parametrov izboljšujemo uspešnost metode. To ni potrebno, ker metoda že z osnovnimi nastavitvami ponavadi doseže najboljši rezultat. Ker smo testirali le enkrat in smo ob tem testiranju za vsako metodo izbrali parametre pred začetkom, ni bilo mogoče prilagajanje parametrov in s tem izboljševanje učinkovitosti pri določenih metodah.

Vendar je nekaj značilnosti testiranja takšnih, da je treba pridobljene rezultate vseeno vzeti z zadržkom. Pri testiranju smo za delitev množice primerov na učno in testno množico uporabili metodo bootstrap z desetimi iteracijami. Za bolj zanesljive rezultate bi bilo potrebno pri vsaki množici podatkov izvesti več iteracij. Vendar pa to ni bilo možno, ker je bilo samo izvajanje testiranja že sedaj zelo časovno potratno in da smo sploh prišli do rezultatov, smo morali uporabiti gručo računalnikov. Če bi imeli na voljo zmogljivejše računalnike, bi lahko metodo bootstrap izvedli z večimi iteracijami in s tem dosegli večjo zanesljivost klasifikacijske točnosti pri testiranju.

Druga značilnost testiranja, ki morda nekoliko postavi rezultate pod vprašaj, se nanaša na število ustreznih množic podatkov za testiranje. Pri testiranju smo postavili pogoj za vsako množico podatkov. Pogoj je bil, da mora biti število primerov, ki pripadajo določenemu razredu, večje od deset. Problem je, da je deset primerov za točno klasifikacijo zelo malo. Če predpostavimo, da se dve tretjini primerov določi za učno množico, to pomeni, da točnost napovedovanja preverimo na le treh primerih. To pa je malo. Ob naključnem faktorju, ki se pojavi pri določanju, kateri primeri gredo v učno in kateri v testno množico, lahko pri tako majhnem številu primerov to pomaga oziroma škodi določeni metodi. To se zgodi zaradi tega, ker imajo metode različen način napovedovanja na osnovi učnih primerov.

Razlog, zakaj nismo meje desetih primerov dvignili na več primerov, temelji v sami strukturi in količini podatkov. Vsaka množica podatkov, na kateri smo testirali, ima podatke, ki so neodvisni od drugih množic podatkov. Problem se je pojavil v tem, da so množice podatkov ponavadi vsebovale zelo malo primerov. Že od pogoju, da je deset primerov minimum, smo morali iz testiranja izpustiti več kot tri četrtine vseh množic podatkov. Tako smo se morali odločiti, ali naj dvignemo omejitev minimalnega števila primerov in s

tem pridobimo na zanesljivosti rezultatov, ampak hkrati še zmanjšamo število množic podatkov, na katerih smo testirali, ali ne. Odločili smo se, da bomo pustili omejitev, ker nas je zanimal rezultat na vseh množicah podatkov. Z dvigom omejitve bi se približevali primerjavi uspešnosti metod na le nekaterih množicah podatkov in ne na vseh, kot je bilo na začetku zamišljeno.

Za prihodnja testiranja bi bilo potrebno preveriti, ali bi sprememba parametrov pri določenih metodah kaj spremenila rezultate. Potrebno bi bilo povečati število iteracij pri metodi bootstrap in s tem še pridobiti na zanesljivosti rezultatov. Glavna stvar, ki bi jo bilo potrebno narediti, če bi želeli dejansko ugotoviti, katera metoda je najuspešnejša, pa je, da bi metode testirali še na drugih, večjih množicah podatkov.

Poglavje 5

Zaključek

Iz podatkov o genskih izrazih je mogoče napovedovati, ali je določeno obolenje rakavo ali ne. Razlika med zdravimi in bolnimi celicami je namreč v izražanju genov v celici. V zdravi celici je izražanje genov drugačno kot v celici, ki je zaradi različnih vplivov spremenjena in je zato njeno izražanje genov in s tem tvorjenje proteinov drugačno/napačno.

Pri tem, kako čim bolj zanesljivo postaviti diagnozo rakavega obolenja iz podatkov o genskih izrazih, pridobljenih s pomočjo mikročipov, je zaenkrat sicer še kar nekaj problemov, ki izhajajo iz pridobivanja in obdelave velike količine podatkov. Vendar je to način, ki veliko obeta in bo v prihodnosti pripomogel k hitrejšemu odkrivanju in zdravljenju bolezni.

Rezultati opravljenega testiranja na danih množicah podatkov o genskih izrazih so pokazali, da je od vseh uporabljenih tehnik strojnega učenja metoda naključnih gozdov v danih pogojih najuspešnejša klasifikacijska metoda za diagnosticiranje raka. Za njo sta se uvrstili obe različici metode SVM, nato pa metoda najbližjih sosedov, metoda odločitvenih dreves in na koncu naivni Bayes. Kot najuspešnejša se je metoda naključnih gozdov izkazala tako z najboljšo povprečno vrednostjo mere AUC, kot tudi z najboljšim povprečnim rangom. To pomeni, da če bi se morali pri diagnosticiranju raka odločiti za eno metodo, bi to bila metoda naključnih gozdov.

Prve tri najboljše metode so si bile resda zelo blizu po rezultatih, vseeno pa je to, da je bila metoda naključnih gozdov najbolj učinkovita, manjše presenečenje. Literatura o dosedaj opravljenih primerjavah med metodami govori o tem, da se pri večini različnih testiranj kot najuspešnejša izkaže metoda SVM. Vendar je pri naših rezultatih potrebno upoštevati, da bi se z različnim uglaševanjem metod in z večimi ponovitvami testiranja lahko rezultati tudi malo spremenili. To daje spodbudo, da se je potrebno s tem problemom ukvar-

jati še naprej in bolj v podrobnosti, saj ima takšno diagnosticiranje zelo svetlo prihodnost.

Dodatek A

Programska koda v jeziku Python

Primer programa(eden od dvajsetih), ki sem ga uporabil za testiranje:

```
# -*- coding: latin-1 -*-
import orange
import obiGEO
import orngTest
import orngStat
import orngEnsemble
import orngTree
import orngSVM
import orngFSS
import random
import stats

def bootstrap_indices(n):
 indices = []
 out_of_bag = n*[1]   #na zacetku so vsi primeri out of bag = testni
 for i in range(n):
        r = random.randint(0,n-1)
  indices.append(r)
  out_of_bag[r] = 0
 return indices, [i for i in range(n) if out_of_bag[i] == 1]

def make_bootstrap(data,learning_indices, testing_indices):
 train = orange.ExampleTable(data.domain)
 train.extend([data[i] for i in learning_indices])
 test = orange.ExampleTable(data.domain)
 test.extend([data[i] for i in testing_indices])
 return train, test

def attributeFilter(data):
 podatki = []  #tu zapisujemo attribute, ki jih bomo uporabili
 natt = len(data.domain.attributes)
 missing = [0.] * natt

 for i in data:
```

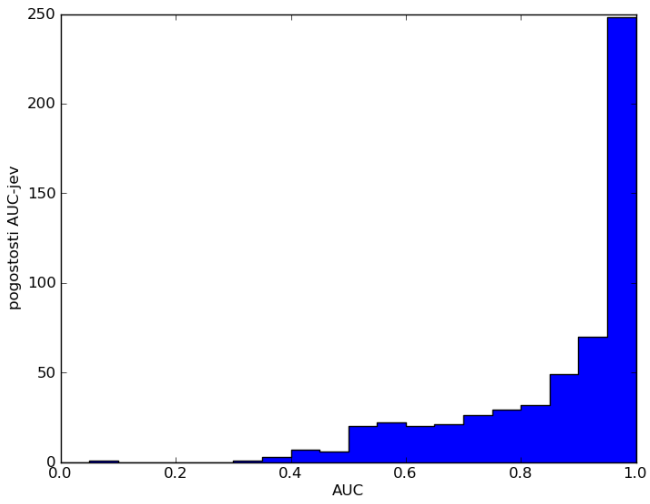
```
  for j in range(natt):
   if i[j].isSpecial():
    missing[j] += 1
 missing = map(lambda x, l=len(data):x/l*100., missing)

 for i in range(natt):
  if(missing[i]<50):
   podatki.append(data.domain.variables[i])
 podatki.append(data.domain.variables[natt]) #na koncu (classVar)

 domena=orange.Domain(podatki)
 data2= orange.ExampleTable(domena, data)

 return data2


#KLASIFIKATORJI
bayes = orange.BayesLearner(name = "Bayes")
ll = orange.LinearLearner(name= "LinearLearner")
svm= orngSVM.SVMLearnerEasy(name = "SVM", kernel_type=orngSVM.SVMLearner.RBF)
knn = orange.kNNLearner(name = "kNN")
tree=orngTree.TreeLearner(name = "tree", mForPruning=2,minSubset=2)
forest = orngEnsemble.RandomForestLearner(trees=100, name="Forest")
majority = orange.MajorityLearner(name="Majority")

learners = [ll, svm, knn, tree, forest, majority]
includeBayes=True    #če je true potem se upošteva tudi bayes

min_examples = 10
no_of_teoretical_iterations =3 #  kolikokrat ponovimo bootstrap(k=3)
learnersAUC=(len(learners))*[0]  #potrebno za izračun povpr. AUC za klasif.
bayesAUC=0  #kot zgoraj, le da za bayesa

averageLearnersAUCs = []  #sem bomo zapisovali pare (indeks od
#learners, povprečna vrednost AUC za ta learner)
averageBayesAUCs=[] # povpr. vrednosti AUC za bayesa za vsak ds

ranks=[]  #SEM ZAPISUJEMO TABELO RANGOV PRI DOLOČENEM DATASET-U
no_of_datasets=0 # ŠT RAZLIČNIH MNOŽIC NA KATERIH DELAMO KLASIF.

stev_dataSeta=0
zapStev=1
interval=100

gdsinfo = obiGEO.GDSInfo()
for gds_id, info in gdsinfo.items()[((zapStev-1)*interval):][:interval]:
   print "DataSet: ", gds_id, "    to je dataset st: ", stev_dataSeta
   stev_dataSeta=stev_dataSeta+1
   subset_types = set([s["type"] for s in info['subsets']])
   for st in subset_types:
 ex_subset = [(s["description"], s["sample_id"]) for s in info["subsets"]
if s["type"] == st if len(s["sample_id"]) >= min_examples]
  # ex_subset vsebuje ime razreda (v gds717 pod type strain je to R6/2 in pa
#wild type) ter id-je primerov podmn. je v datasetu GDS717 R6/2 in wild type

 if len(ex_subset) >= 2:
          ## TU SE TESTIRA
  no_of_datasets = no_of_datasets +1
  gds = obiGEO.GDS(gds_id)
```

```
  data = gds.getdata(report_genes=True, transpose=True, sample_type=st,
remove_unknown=True)
  print "---- ", st, " ----"

  #DOLOČIMO ZAČETNE VREDNOSTI SPREMENLJIVK ZA IZRAČUN POVPRAČNIH AUC-JEV
  no_of_iterations = no_of_teoretical_iterations
  bayesAUC=0
  for i in range(len(learners)):
   learnersAUC[i]=0;

  # ZANKA SE PONOVI KOLIKOKRAT SE PONOVI IZBIRA UČNE/TESTNE MNOŽICE
  for i in range(no_of_iterations):
   try:
    print "Iteracija: ", (i+1)
    data=attributeFilter(data)
    #ZA BAYESA JE POTREBNO DISKRETIZIRATI ATRIBUTE
    if(includeBayes):
     bayesData= orange.Preprocessor_discretize(data,
method=orange.EquiNDiscretization(numberOfIntervals=5))
     learning_indices, testing_indices = bootstrap_indices(len(bayesData))
     trainData, testData=make_bootstrap(bayesData,learning_indices,testing_indices)
     results = orngTest.learnAndTestOnTestData([bayes], trainData, testData)

     bayesAUC = bayesAUC + orngStat.AUC(results)[0]
     print "Metoda: ", bayes.name, "  AUC = %.3f" % orngStat.AUC(results)[0]


    #ZA OSTALE KLASIF. PREVERIMO REZ. POSEBEJ, KER NIMAJO DISKRETNIH ATRIBUTOV
    learning_indices, testing_indices = bootstrap_indices(len(data))
    trainData, testData = make_bootstrap(data,learning_indices, testing_indices)
    print "pred rezultati"
    results = orngTest.learnAndTestOnTestData(learners, trainData, testData)
    for i in range(len(learners)):
      learnersAUC[i] = learnersAUC[i] + orngStat.AUC(results)[i]

     print "Metoda: ",learners[i].name," AUC = %.3f"% orngStat.AUC(results)[i]

   except:
    no_of_iterations=no_of_iterations-1
    print
    print "Prislo je do napake"
    print

  #IZRAČUNAMO POVPREČNI AUC IN GA SHRANIMO ZA IZRIS HISTOGRAMA
  temp=[]
  if(includeBayes):
   averageBayesAUCs.append(bayesAUC/no_of_iterations)
   temp.append((bayesAUC/no_of_iterations))
   print "Metoda: ", bayes.name, " ima povprečni AUC: ",
(bayesAUC/no_of_iterations)

  for i in range(len(learners)):
   averageLearnersAUCs.append((i,(learnersAUC[i]/no_of_iterations)))
   temp.append((learnersAUC[i]/no_of_iterations))
   print "Metoda: ", learners[i].name, " ima povprečni AUC: ",
(learnersAUC[i]/no_of_iterations)

  temp[:]=[(1-i) for i in temp]  #RANK 1 BO IMEL TISTI, KI JE NAJBLJIŽJE 1
  ranks.append(stats.rankdata(temp)) # V RANKS SE DODAJO RANGI
```

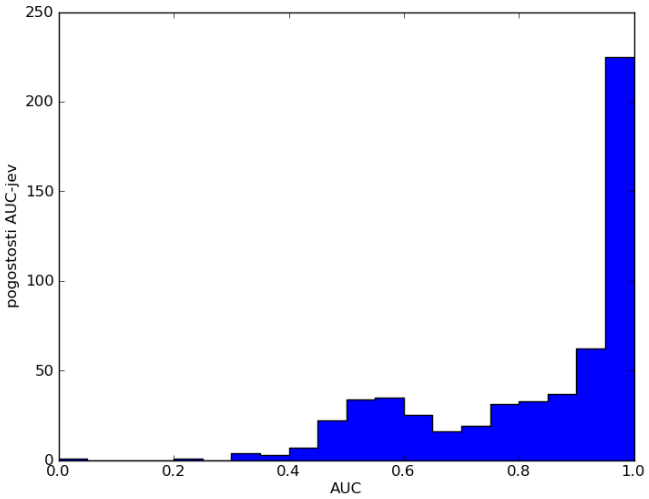
```
    print    #NAREDI NOVO VRSTO ZA NASLEDNJI DATASET


#IZRAČUN POVPREČNIH RANGOV
print "Rangi"
print ranks
averageRanks=[]

#OSTALI KLASIFIKATORJI
for i in range(len(learners)):
 temp=0
 for j in range(len(ranks)):
  temp=temp + ranks[j][(i)]
 temp=temp/(len(ranks))
 averageRanks.append(temp)

#BAYES
if(includeBayes):
 temp=0
 for i in range(len(ranks)):
  temp=temp + ranks[i][(len(learners))]
 temp=temp/(len(ranks))
 averageRanks.append(temp)


#V DATOTEKO SHRANIMO RANGE IN POVPREČNE AUC-JE

filename="rezultati" + str(zapStev) + ".txt"
file = open(filename, 'w')

#NAJPREJ NAPIŠEMO POVPREČNE AUC-JE
#BAYES
file.write(str(averageBayesAUCs))
file.write("\n")
#OSTALI KLASIFIKATORJI
file.write(str(averageLearnersAUCs))
file.write("\n")

#POVPREČNI RANGI
file.write(str(averageRanks))

file.close()
```

Dodatek B

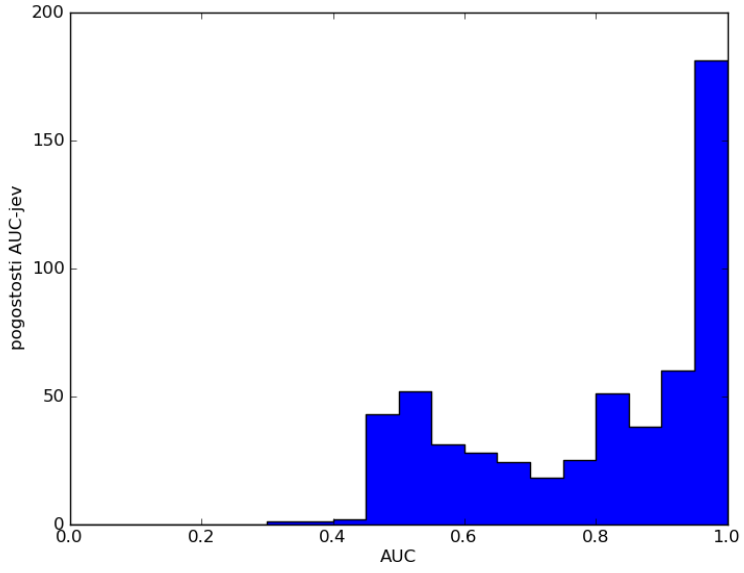
Histogrami uspešnosti klasifikacijskih metod



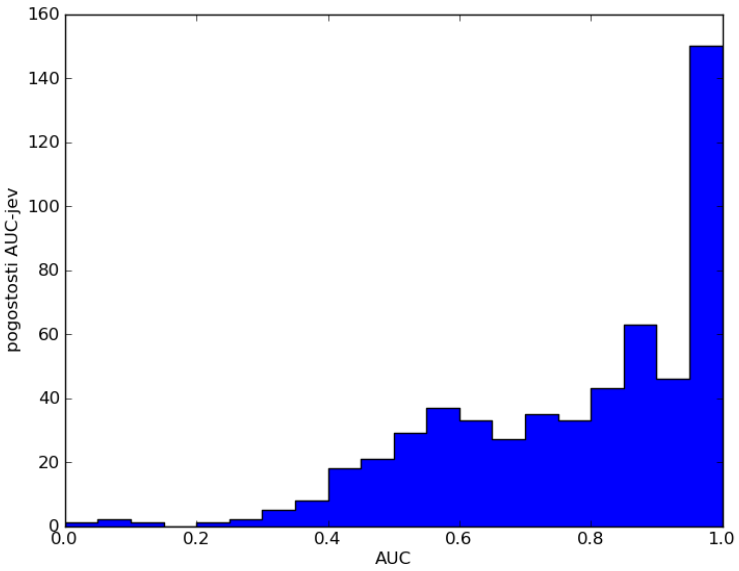
Slika B.1: *Histogram uspešnosti metode naključnih gozdov*



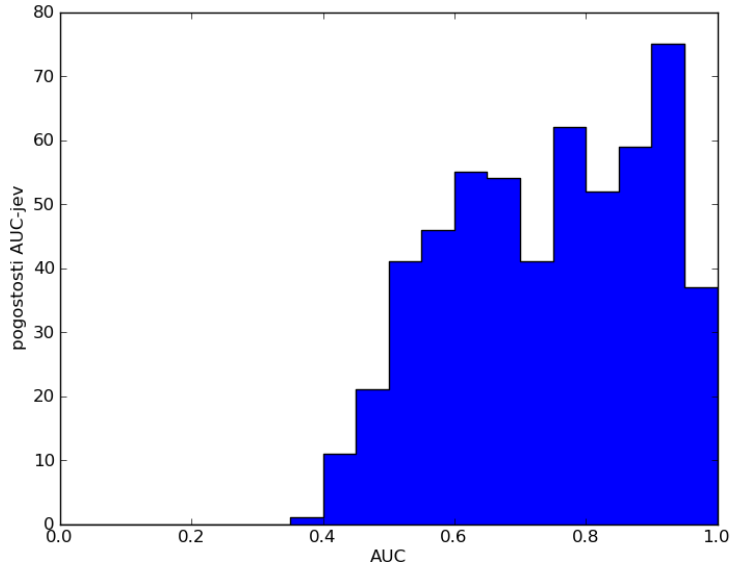
Slika B.2: *Histogram uspešnosti metode SVM z linearnim jedrom*



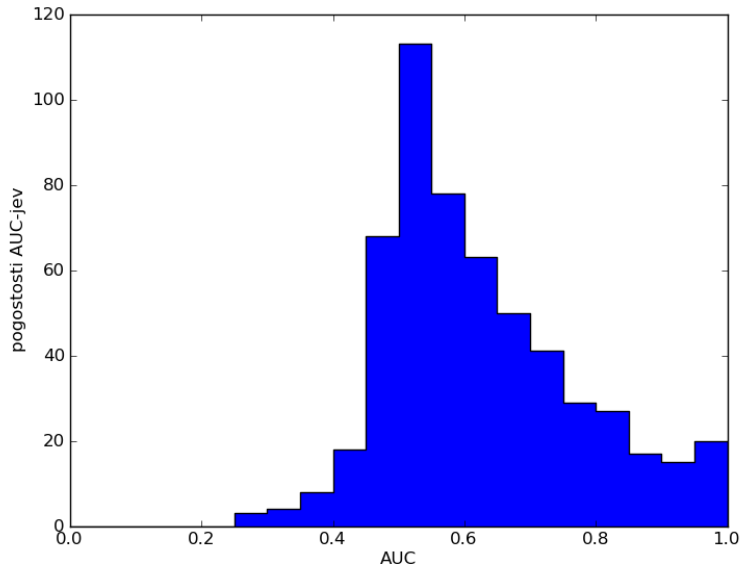
Slika B.3: *Histogram uspešnosti metode SVM z radialnim jedrom*



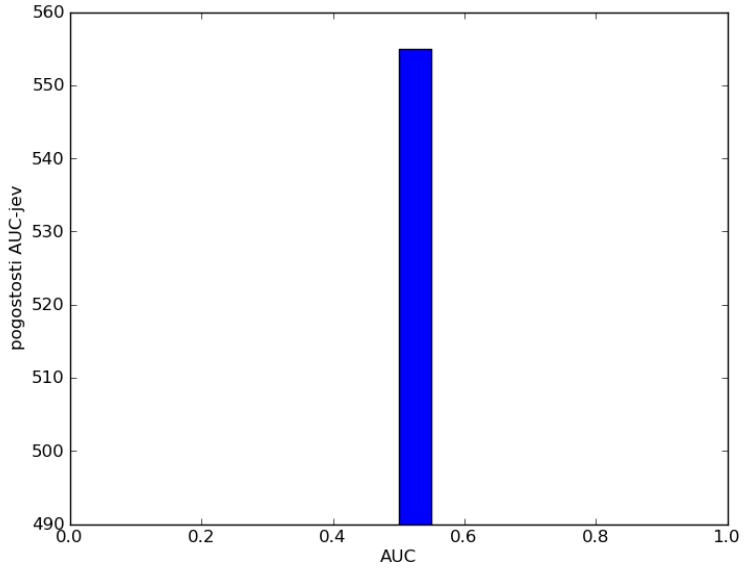
Slika B.4: *Histogram uspešnosti metode najbližjih sosedov*



Slika B.5: *Histogram uspešnosti metode odločitvenih dreves*



Slika B.6: *Histogram uspešnosti metode naivni Bayes*



Slika B.7: *Histogram uspešnosti metode večinskega klasificiranja*

Literatura

[1] J. R. Beck and E. K. Schultz, The use of ROC curves in test performance evaluation *Arch Pathol Lab Med*, 110: 13–20, 1986.

[2] A. L. Boulesteix, C. Strobl, T. Augustin, M. Daumer, Evaluating Microarray-based Classifiers: An Overview *Cancer Informatics*, 77-97, 2008

[3] M. P. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares Jr., D. Haussler, Knowledge-based analysis of microarray gene expression data by using support vector machines, *PNAS*,2000 (97): 262-267.

[4] T. Curk, J. Demsar, Q. Xu, G. Leban, U. Petrovic, I. Bratko, G. Shaulsky, B. Zupan, Microarray data mining with visual programming, *Bioinformatics*, 21(3): 396-398, 2005.

[5] J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, *Journal of Machine Learning Research*, 1-30, Jul. 2008.

[6] I. Konononko , *Strojno učenje*, Ljubljana : Fakulteta za računalništvo in informatiko, 2005

[7] A.Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lazano, R.Armananzas, G. Santafe, A. Perez, V. Robles, *Machine learning in bioinformatics*, Briefings in Bioinformatics, 7(1), pp. 86-112, 2006.

[8] (2007), *Microarrays: Chipping away at the mysteries of science and medicine*, Dostopno na:
http://www.ncbi.nlm.nih.gov/About/primer/microarrays.html

[9] R. Simon, M. D. Radmacher, K. Dobbin, L. M. McShane Pitfalls in the Use of DNA Microarray Data for Diagnostic and Prognostic Classification, *Journal of the National Cancer Institute*, 95(1): 14-18, 2003.

[10] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, S. Levy, A comprehensive evaluation of multicategory classification methods for microarray gene expresion cancer diagnosis, *Bioinformatics*, 21: 631-643, 2005.

[11] Gene expression and cancer: getting it together, *Editorial Nature Genetics*, 31(1): 1-2, 2002.

[12] T. Barrett, at al. NCBI GEO: archive for high-throughput functional genomic data, *Nucleic Acids Research*, 2008 (37).