Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Marko Robnik Šikonja

# RAZVOJ HEVRISTIK ZA USMERJANJE UČENJA REGRESIJSKIH DREVES

magistrsko delo

Mentor: prof.dr. Igor Kononenko

Ljubljana, marec 1997

# Povzetek

Analizirali smo nekatere ključne elemente v znanih sistemih za učenje regresijskih dreves ter jih poskušali dopolniti in izboljšati. V ta namen smo razvili učni sistem in vanj vgradili nekatere znane pristope, nato pa smo jih dopolnili in dodali postopke, ki smo jih razvili v tem delu.

Pri strojnem učenju se je za enega ključnih elementov izkazala hevristična ocena kvalitete atributov. Za klasifikacijske učne probleme in za relacijsko učenje obstajajo variante algoritma Relief z mnogimi teoretičnimi prednostmi pred hevristikami, ki temeljijo na nečistoči. Za učenje regresijskih funkcij takšne hevristike še ne obstajajo, zato obstoječi regresijski učni sistemi uporabljajo regresijske variante funkcij nečistoče. Na podlagi analize algoritma ReliefF smo izpeljali nekratkovidno hevristiko za oceno atributov v regresijskih problemih in empirično preverili njeno obnašanje glede na število učnih primerov, stopnjo šuma v podatkih in število naključnih atributov v opisu problema. Novi algoritem RReliefF (Regresijski ReliefF) smo preizkusili tudi v okviru učnega sistema za gradnjo regresijskih dreves.

Pri klasifikaciji in v induktivnem logičnem programiranju se je na nekaterih vrstah problemov pokazalo, da obstoječi atributi in relacije ne zadoščajo za razumljiv opis danega koncepta. Tipično se problemi takšne vrste rešujejo z avtomatskim ali ročnim dodajanjem novih, vmesnih konceptov. Ta pristop, imenovan konstruktivna indukcija, smo poskusili tudi pri učenju regresijskih dreves. Uporabili smo operatorje konjunkcije, seštevanja in množenja. V duhu principa najkrajše dolžine opisa (MDL) smo izpeljali kodiranje ocene kvalitete konstruktov za RReliefF in MSE.

Princip MDL smo uporabili pri gradnji linearnih modelov v listih, za kar smo razvili kodiranje modelov, ter ga testirali z nekaj optimizacijskimi metodami.

Kodiranje konstruktov in modelov smo uporabili pri rezanju dreves po principu MDL. Novo rezanje smo primerjali z uveljavljeno metodo rezanja z $m$-oceno verjetnosti.

Vse uvedene novosti smo testirali na več množicah umetnih in realnih podatkov.

# The developement of the heuristics for guiding the learning of the regression trees

## Abstract

We have analysed and tried to improve some key procedures used in the learning of the regression trees. For this we have developed the regression trees learning system and incorporated some of the methods used in the known systems and the new methods presented in this thesis.

The problem of estimating the quality of attributes seems to be an important issue in machine learning. Algorithm Relief and its variants used in classification and inductive logic programming have many theoretical advantages over impurity based estimators. There is no such heuristic for regressional problems. We present the analysis of ReliefF which lead us to adapt it to continuous class problems. The behaviour of Regressional ReliefF (RReliefF) was tested with different number of learning examples, with noisy data and with different numbers of random attributes. The experiments show that it can be used for non-myopic learning of the regression trees.

In classification problems and in inductive logic programming we often employ constructive induction when the existing set of attributes is not enough for the description of the target concept. We have tried constructive induction in regression and implemented conjunction, addition and multiplication as constructive operators. A coding scheme was developed for constructs as well as for the estimates of the attribute's quality for RReliefF and MSE. In the spirit of the Minimum Description Length (MDL) principle these codings were used for the selection of the best construct.

The coding scheme for linear models was derived and the MDL principle was used also for optimization of the linear models in the leaves of the regression tree.

We propose a MDL-based pruning algorithm, which uses the codings of constructs and models and compare it with well known $m$-estimate pruning method.

All new methods were tested on several artificial and real world domains.

# Ključne besede

# Keywords

| | |
|---|---|
| umetna inteligenca | artificial intelligence |
| strojno učenje | machine learning |
| regresija | regression |
| induktivno učenje | inductive learning |
| ocenjevanje atributov | attribute estimation |
| konstruktivna indukcija | constructive induction |
| regresijska drevesa | regression trees |
| funkcije nečistoče | impurity functions |
| linearni modeli | linear models |
| rezanje regresijskih dreves | pruning of the regression trees |
| najkrajša dolžina opisa | mimimal description length (MDL) |

# Zahvala

Najprej bi se rad zahvalil mentorju prof. dr. Igorju Kononenku, ki me je v teku podiplomskega študija usmerjal in spodbujal, pri tem pa ni ostal zgolj mentor in profesor.

Zahvala gre vodji naše raziskovalne skupine prof. dr. Ivanu Bratku, ki me je sprejel v svoje učinkovito raziskovalno in delovno okolje.

Sodelavci obeh ljubljanskih laboratorijev za umetno inteligenco, še posebej Matjaž Kukar in Uroš Pompe so prispevali ustvarjalno vzdušje ter odgovorili na mnoga moja vprašanja.

Ljubezen mojih bližjih in še posebej Žo je osmišljala mene in moje delo. Hvala.

# Kazalo

# 1

# Uvod

*Vse je bilo že povedano. Toda, ker nihče ne posluša, je potrebno vedno znova začenjati.*

*André Gide*

Učenje je gotovo ena od aktivnosti, ki so najmočneje oblikovale človeka, in fenomen, ki ga proučujejo številne znanstvene discipline. V tem delu gledamo na učenje z vidika umetne inteligence, oziroma natančneje z vidika strojnega učenja.

Predstavljajmo si, da se ukvarjamo z nekim pojavom. O njem in o stvareh v zvezi z njim smo sistematično zbirali podatke, poznamo nekaj preteklih pojavitev tega pojava in okoliščin, v katerih se je zgodil. Iz zbranih podatkov se želimo česa naučiti. S pridobljenim znanjem želimo napovedovati značilnosti prihodnjih pojavitev in/ali pojav bolje razumeti.

Področje našega dela je še ožje definirano. Ukvarjali se bomo z atributnim učenjem iz primerov, kar pomeni, da smo podatke zbirali v obliki značilnosti oziroma atributov pojava. Napovedovati želimo le eno značilnost naenkrat in sicer predpostavljamo, da jo lahko izrazimo s številko.

Zgornji odstavek je poskušal enostavno predstaviti področje našega dela, ki je atributno učenje z zveznim razredom. Na učenje zveznega razreda lahko gledamo tudi kot na proces regresije - iskanja regresijske (aproksimacijske) funkcije. Poleg klasičnih statističnih pristopov k temu problemu (multidimenzionalna regresija, faktorska analiza) je znana in vse bolj uveljavljena tudi drevesno strukturirana regresija oziroma gradnja regresijskih dreves. Znanje je tu predstavljeno v obliki drevesa. Pri napovedovanju vrednosti nekega primera začnemo v korenu. V vsakem vozlišču drevesa se nahaja atribut. Odvisno od vrednosti atributa našega primera sledimo eni od vej iz vozlišča. Ko prispemo do lista, najdemo tu predpis,

ki primeru določi vrednost. Znani tovrstni sistemi so CART (Breiman in sod., 1984), Retis (Karalič, 1991) in M5 (Quinlan, 1993).

Ena od ključnih nalog pri gradnji drevesa je izbira atributov v vozliščih. Izbiro opravimo s pomočjo hevristične funkcije, ki pri danih podatkih oceni kvaliteto oziroma primernost atributov. Ocenjevanje kvalitete atributov ni pomembno le za gradnjo regresijskih dreves. Tovrstna informacija je koristna tudi drugje pri analizi podatkov, npr. pri predprocesiranju, ko izbiramo koristno množico atributov.

Danes uporabljane hevristike temeljijo na fukcijah nečistoče, ki ocenjujejo, kako dobro bi delitev primerov pri neki vrednosti atributa razdelila primere na tiste z večjo in na tiste z manjšo vrednostjo.[1] Mere temelječe na funkcijah nečistoče predpostavljajo medsebojno neodvisnost atributov. Tam, kjer je predpostavka kršena, je njihova ocena neprimerna. Osnovna mera nečistoče v regresiji je srednja kvadratna napaka (MSE).

Za klasifikacijske probleme obstaja algoritem Relief (Kira in Rendell, 1992), ki se zaveda odvisnosti med atributi. Njegova razširitev ReliefF (Kononenko, 1994), ki je primerna za večrazredne probleme, ter šumne in manjkajoče podatke, se je v praksi odlično obnesla (Kononenko in sod., 1997), prilagojena pa je bila tudi za induktivno logično programiranje (Pompe in Kononenko, 1995; Kononenko in sod., 1996). Tudi za regresijske probleme bi želeli ocenjevalno hevristiko s podobnimi kvalitetami. Naj omenimo, da lahko namesto atributov v vozlišča drevesa postavimo tudi kak drug test, ki kombinira informacijo več osnovnih atributov. S sestavljanjem testov v vozliščih se ukvarja konstruktivna indukcija.

Na področju klasifikacije (diskreten razred) in induktivnega logičnega programiranja smo v zadnjih letih opazili številne uspešne primere uporabe principa najkrajše dolžine opisa (MDL) (Quinlan in Rivest, 1989; Kovačič, 1994; Mehta in sod., 1995; Kononenko, 1995). Čeprav ima pristop številne dobre lastnosti in je teoretično dobro utemeljen, v regresiji še ni bil uporabljen. Raziskali smo uporabo principa v konstruktivni indukciji, gradnji modelov v listih in pri rezanju regresijskih dreves.

## 1.1 Pregled opravljenega dela

Zgradili smo sistem za učenje regresijskih dreves. Vanj smo vključili znane metode ocenjevanja atributov in rezanja. V listih smo dopustili uporabo linearnih modelov ali srednje vrednosti razreda.

[1] Pri Retisu hevristika ocenjuje delitev glede na to, kako dobro lahko primere opišemo z linearno funkcijo, kar pa je še vedno oblika funkcije nečistoče.

Proučili smo obstoječe načine ocenjevanja atributov v regresiji ter na podlagi algoritma ReliefF izpeljali novo nekratkovidno regresijsko hevristiko, ki jo imenujemo RReliefF. Preverili smo njeno obnašanje pri različnem številu učnih primerov, pri dodanem šumu ter z različnim številom naključnih atributov. Vključili smo jo v učni sistem ter testirali njeno obnašanje pri učenju regresijskih dreves.

Učni sistem smo nadgradili s konstruktivno indukcijo. V vozliščih drevesa smo uporabljali operatorje konjunkcije, seštevanja in množenja. Razvili in analizirali smo vključitev principa najmanjše dolžine opisa v ocenjevanje kvalitete konstruktov z algoritmoma RReliefF in MSE.

Princip MDL smo uporabili pri gradnji modelov v listih, za kar smo razvili kodiranje modelov, ter ga testirali z nekaj optimizacijskimi metodami.

Kodiranje konstruktov in modelov smo uporabili pri rezanju dreves po principu MDL. Novo rezanje smo primerjali z uveljavljeno metodo rezanja z $m$-oceno verjetnosti.

Vse uvedene novosti smo testirali na več množicah umetnih in realnih podatkov.

## 1.2 Pregled vsebine

Drugo poglavje vsebuje opis in značilnosti nekratkovidnega regresijskega algoritma RReliefF. Najprej predstavi osnovni algoritem Relief in njegovo razširitev ReliefF, nato izpelje regresijsko inačico. Pokažemo nekaj lastnosti algoritma RReliefF in ga empirično primerjamo s kratkovidnim algoritmom na problemih ocenjevanja kvalitete atributov in gradnje regresijskih dreves.

Tretje poglavje se ukvarja s konstruktivno indukcijo. Po krajši predstavitvi tematike pokažemo uporabo principa najkrajše dolžine opisa v kombinaciji z algoritmoma RReliefF in MSE ter predstavimo rezultate.

Četrto poglavje predstavi gradnjo linearnih modelov v listih regresijskega drevesa. Predstavimo znane pristope in uvedemo pristop temelječ na principu najkrajše dolžine opisa.

V petem poglavju se dotaknemo rezanja regresijskih dreves. Primerjamo že uveljavljen pristop z novim, ki uporablja formule za dolžino opisa iz tretjega in četrtega poglavja.

Šesto poglavje sklene delo, povzame glavne dosežke, nakaže odprta vprašanja in predstavi smernice za nadaljnje delo.

Dodatek A vsebuje grafe odvisnosti ocen atributov algoritmov RReliefF in MSE od števila učnih primerov za uporabljene umetne probleme.

Dodatek B prikaže odvisnosti ocen atributov od šuma v podatkih.

Dodatek C grafično ponazarja odvisnost ocen algoritma RReliefF od števila atributov z naključnimi vrednostmi v opisu problema.

# 2

# Nekratkovidna hevristika

*Bistvo je očem nevidno. Kdor hoče videti, mora gledati s srcem.*
*Antoine de Saint-Exupéry*

Ocenjevanje atributov je eden od pomembnih problemov v strojnem učenju. Pojavlja se pri napovedovanju tako diskretnega kot zveznega razreda z različnimi simboličnimi formalizmi (drevesa, pravila, relacije), pri konstruktivni indukciji in pri izbiri podmnožice atributov (feature subset selection).

Večina hevrističnih ocen kvalitete atributov predpostavlja medsebojno neodvisnost atributov. Pri diskretnem razredu so takšne vse mere, ki temeljijo na funkcijah nečistoče, na primer informacijski prispevek (information gain) (Hunt in sod., 1966) in Gini-indeks (Breiman in sod., 1984), pa tudi mera razdalje (distance measure) (Mantaras, 1989) in J-ocena (Smyth in Goodman, 1990). Primera za zvezni razred sta srednja kvadratna in srednja absolutna napaka (Breiman in sod., 1984). Zaradi te predpostavke so te mere manj primerne pri problemih z močnimi odvisnostmi med atributi.

Za diskretni razred obstaja nekaj ocenjevalnih funkcij, ki se zavedajo možnih odvisnosti med atributi in jih zaznavajo. Najbolj znana med njimi sta algoritem Relief (Kira in Rendell, 1992) in njegova razširitev ReliefF (Kononenko, 1994), ki pravilno ocenita kvaliteto atributov v klasifikacijskih problemih z močnimi odvisnostmi med atributi. Podobni sta jima kontekstna vrednost (contextual merit) (Hong, 1994) in geometrijska ocena (Elomaa in Ukkonen, 1994).

V tem poglavju si bomo zaradi lažjega razumevanja najprej pogledali osnovni algoritem Relief in njegovo razširitev ReliefF. Ukvarjali se bomo z analizo algoritma ReliefF, ki nas bo pripeljala do njegove razširitve na zvezni razred, nato pa bomo z nekaj poskusi ilustrirali lastnosti novega algoritma.

## 2.1 Osnovni algoritem Relief

Namen originalnega algoritma Relief (Kira in Rendell, 1992) in vseh njegovih izpeljank je oceniti kvaliteto atributov, glede na to kako dobro vrednosti atributov ločijo med primeri, ki so si podobni. Algoritem se nahaja na sliki 2.1.

*Algoritem* Relief
*Vhod:* za vsak učni primer vektor vrednosti atributov in vrednost razreda
*Izhod:* vektor ocen kvalitete atributov $W$

1. postavi vse uteži $W[A]$ := 0.0;
2. **for** i := 1 **to** $m$ **do begin**
3. naključno izberi učni primer $R$;
4. poišči najbljižji zadetek $H$ in najbližji pogrešek $M$;
5. **for** $A$ := 1 **to** število atributov **do**
6. $W[A]$ := $W[A]$ - diff($A$,$R$,$H$)/$m$ + diff($A$,$R$,$M$)/$m$;
7. **end**;

Slika 2.1: Osnovni algoritem Relief.

Algoritem najprej naključno izbere učni primer $R$ (3. vrstica) in poišče dva, njemu najbljižja soseda (4. vrstica): enega iz istega razreda, ki ga imenujemo blužnji zadetek $H$, in drugega iz različnega razreda, ki ga imenujemo bližnji pogrešek $M$. Glede na vrednosti atributov pri primerih $R$, $H$ in $M$ (5. in 6. vrstica) algoritem popravi ocene kvalitete atributov v vektorju $W$. Ves proces se ponovi $m$ krat, pri čemer vrednost $m$ določi uporabnik.

Funkcija $diff(A, I_1, I_2)$ izračuna razliko vrednosti atributa $A$ med dvema primeroma $I_1$ in $I_2$. Za diskretne atribute je definirana kot

$$diff(A, I_1, I_2) = \left\{ \begin{array}{ll} 0 & ; vrednost(A, I_1) = vrednost(A, I_2) \\ 1 & ; sicer \end{array} \right\} \tag{2.1}$$

za zvezne pa kot:

$$diff(A, I_1, I_2) = \frac{|vrednost(A, I_1) - vrednost(A, I_2)|}{max(A) - min(A)} \tag{2.2}$$

Funkcijo $diff$ uporabljamo tudi za izračun razdalje dveh primerov, ko iščemo najbližje primere. Razdalja med dvema primeroma je definirana kot vsota razdalj po vseh atributih.

Ocena kvalitete atributa $W[A]$, ki jo izračuna Relief je približek razlike naslednjih verjetnosti (Kononenko, 1994):

$$\begin{aligned} W[A] \;=\; & P(\text{različna vrednost A}|\text{najbližja primera iz različnega razreda}) \\ & - P(\text{različna vrednost A}|\text{najbližja primera iz istega razreda}) \end{aligned} \tag{2.3}$$

Časovna kompleksnost algoritma Relief za $N$ učnih primerov in $A$ atributov je $O(m \times N \times A)$. Z originalnim algoritmom Relief lahko ocenjujemo zvezne in diskretne atribute, vendar pa smo omejeni le na probleme z dvema razredoma in brez neznanih vrednosti.

## 2.2 Razširjeni ReliefF

Kononenko (1994) je razširil originalni Relief tako, da je sposoben delovati na nepopolnih podatkih in na večrazrednih problemih, bistveno manj pa je občutljiv tudi na šumne podatke. Razširitev, ki jo je poimenoval ReliefF, se nahaja na sliki 2.2.

*Algoritem* ReliefF
*Vhod:* za vsak učni primer vektor vrednosti atributov in vrednost razreda
*Izhod:* vektor ocen kvalitete atributov $W$

1. postavi vse uteži $W[A]$ := 0.0;
2. **for** i := 1 **to** $m$ **do begin**
3. naključno izberi učni primer $R$;
4. poišči $k$ najbljižjih zadetkov $H_j$;
5. **for** vsak razred $C \neq razred(R)$ **do**
6. iz razreda $C$ poišči $k$ najbližjih pogreškov $M_j(C)$;
7. **for** $A$ := 1 **to** število atributov **do**
8. $W[A] := W[A] - \sum_{j=1}^{k} diff(A, R, H_j)/(m \times k) +$
9. $\sum_{C \neq razred(R)} [\frac{P(C)}{1-P(razred(R))} \sum_{j=1}^{k} diff(A, R, M_j(C))]/(m \times k)$;
10. **end**;

Slika 2.2: Razširjeni algoritem ReliefF.

Za regresijo najpomembnejša razširitev je upoštevanje $k$ bližnjih zadetkov in pogreškov namesto enega. Ta sprememba prispeva tudi k robustnosti algoritma in njegovi neobčutljivosti za šum. Razširitev na več razredov je dosežena z uteženo vsoto prispevkov pogreškov iz vseh razredov (9.vrstica).

Manjkajoče vrednosti upoštevamo glede na njihovo verjetnost in sicer razširimo definicijo funkcije $diff$. V primeru, da ima neznano vrednost diskretnega atributa en primer (npr. $I_1$):

$$diff(A, I_1, I_2) = 1 - P(vrednost(A, I_2)|razred(I_1)) \tag{2.4}$$

če pa sta neznani obe vrednosti diskretnega atributa, je $diff$ definiran kot

$$diff(A, I_1, I_2) = 1 - \sum_{i=1}^{št.\ vred.\ A} (P(V_i|razred(I_1)) \times P(V_i|razred(I_2))) \quad (2.5)$$

kjer $V_i$ predstavlja $i$-to vrednost atributa $A$, pogojne verjetnosti pa so ocenjene z njihovo relativno frekvenco na učni množici. Manjkajoče vrednosti zveznih atributov obravnavamo zelo podobno. Namesto verjetnosti upoštevamo neko aproksimacijo gostote verjetnosti npr. jedrne funkcije (kernel functions) (Smyth in Mellstrom, 1992; Redner in Walker, 1984).

Ocene kvalitete atributov, ki jih izračuna Relief, so močno povezane z ocenami funkcij nečistoče (Kononenko, 1994). To lastnost bomo podrobneje analizirali in uporabili v poglavju 3.

Moč algoritma Relief in njegovih izpeljank je njegova sposobnost, da izrabi lokalno informacijo ter upošteva kontekst, toda kljub temu poda globalno sliko.

## 2.3 Regresijski ReliefF - RReliefF

Upoštevajoč zvezo (2.3) in razširitve uporabljene v algoritmu ReliefF smo izpeljali regresijsko verzijo algoritma ReliefF, ki smo jo poimenovali regresijski ReliefF ali kratko RReliefF (Robnik Šikonja in Kononenko, 1996; Kononenko in sod., 1996).

V regresijskih problemih je razred zvezen zato ne moremo uporabiti (bližnjih) zadetkov in pogreškov. Namesto, da bi enolično določili pripadnost razredu, uvedemo raje neke vrste verjetnost, da dva primera pripadata različnima razredoma. To verjetnost modeliramo z relativno razdaljo med vrednostima razreda obeh primerov.

Za oceno $W[A]$ v enačbi (2.3) potrebujemo še predznaka obeh členov. V sledeči izpeljavi bomo preoblikovali enačbo (2.3), tako da jo bomo lahko ovrednotili z uporabo verjetnosti, da dva primera pripadata različnemu razredu. Če zapišemo

$$P_{diffA} = P(\text{različna vrednost A}|\text{bližnja primera}) \quad (2.6)$$

$$P_{diffC} = P(\text{različen razred}|\text{bližnja primera}) \quad (2.7)$$

and

$$P_{diffC|diffA} = P(\text{različen razred}|\text{različna vrednost A in bližnja primera}) \quad (2.8)$$

dobimo z uporabo Bayesovega pravila iz (2.3) naslednjo enačbo:

$$W[A] = \frac{P_{diffC|diffA} P_{diffA}}{P_{diffC}} - \frac{(1 - P_{diffC|diffA}) P_{diffA}}{1 - P_{diffC}} \quad (2.9)$$

Torej lahko ocenimo $W[A]$ tako, da aproksimiramo izraze (2.6), (2.7) in (2.8). To naredi algoritem na sliki 2.3.

*Algoritem* RReliefF
*Vhod:* za vsak učni primer vektor vrednosti atributov in vrednost razreda
*Izhod:* vektor ocen kvalitete atributov $W$

1. postavi vse $N_{dC}$, $N_{dA}[A]$, $N_{dC\&dA}[A]$, $W[A]$ na 0;
2. **for** i := 1 **to** m **do begin**
3. naključno izberi primer $R_i$;
4. izberi **k** primerov $I_j$, ki so najbližji $R_i$;
5. **for** j := 1 **to** k **do begin**
6. $N_{dC} := N_{dC} + |razred(R_i) - razred(I_j)| * f(i,j)$;
7. **for** A := 1 **to** število atributov **do begin**
8. $N_{dA}[A] := N_{dA}[A] + diff(A, R_i, I_j) * f(i,j)$;
9. $N_{dC\&dA}[A] := N_{dC\&dA}[A] + |razred(R_i) - razred(I_j)| *$
10. $diff(A, R_i, I_j) * f(i,j)$;
11. **end**;
12. **end**;
13. **end**;
14. **for** A := 1 **to** število atributov **do**
15. $W[A] := N_{dC\&dA}[A]/N_{dC}$ - $(N_{dA}[A] - N_{dC\&dA}[A])/(m - N_{dC})$;

Slika 2.3: Algoritem RReliefF.

Uteži za različen razred, različen atribut ter za različna razred in atribut zbiramo v $N_{dC}$, $N_{dA}[A]$ in $N_{dC\&dA}[A]$. Oceno vsakega atributa $W[A]$ (enačba (2.9)) izračunamo v 14. in 15. vrstici.

Z uporabo člena $f(i,j)$ na sliki 2.3 (vrstice 6, 8 in 10) upoštevamo razdaljo med dvema primeroma $R_i$ and $I_j$. Bližji primeri naj bi imeli večji vpliv, zato z naraščajočo razdaljo od primera $R_i$ eksponentno manjšamo vpliv primera $I_j$:

$$f(i,j) = \frac{f_1(i,j)}{\sum_{l=1}^{k} f_1(i,l)} \quad \text{in} \quad f_1(i,j) = e^{-\left(\frac{rank(R_i,I_j)}{\sigma}\right)^2} \tag{2.10}$$

kjer $rank(R_i, I_j)$ pomeni rang (vrstni red) primera $I_j$ v padajoče urejenem zaporedju razdalj primerov od primera $R_i$, parameter $\sigma$ pa uravnava hitrost padanja vpliva z razdaljo in ga določi uporabnik.

Poskušali smo tudi z uporabo konstantnega vpliva vseh $k$, primeru $R_i$ najbližjih primerov $I_j$, tako, da smo vzeli $f_1(i,j) = 1/k$, toda rezultati se niso statistično značilno razlikovali. Ker menimo, da je uporaba eksponentno padajočega vpliva primernejša in splošnejša, bomo v tem delu podajali le tovrstne rezultate.

Časovna kompleksnost algoritma RReliefF je enaka časovni kompleksnosti originalnega algoritma Relief in je $O(m \times N \times A)$. Najzahtevnejša operacija v glavni zanki **for** je izbira **k** najbližjih sosedov $I_j$. Zanjo moramo izračunati razdalje od $I_j$ do $R_i$, kar lahko za $N$ primerov storimo v $O(N \times A)$ korakih. To je časovno najbolj zahtevno, medtem ko za izgradnjo kopice potrebujemo $O(N)$ operacij, $k$ najbližjih pa iz nje izločimo v $O(k \log n)$ korakih, toda to je manj kot $O(N \times A)$.

Opozorimo naj še, da tako ReliefF v klasifikaciji kot RReliefF v regresiji izračunavata približke enačbe (2.9), kar nam daje enoten pogled na ocenjevanje atributov v klasifikaciji in regresiji.

## 2.4 Preizkušanje algoritma RReliefF

V tem razdelku bomo najprej preizkusili zmožnosti algoritma RReliefF, da prepozna pomembne atribute in jih razvrsti po pomembnosti, nato pa ga bomo uporabili pri gradnji regresijskih dreves.

RReliefF bomo primerjali s srednjo kvadratno napako (mean squared error - MSE) kot mero kvalitete atributov (Breiman in sod., 1984). To je standardna mera v sistemih z regresijskimi drevesi. Glede na ta kriterij je najboljši tisti atribut, ki minimizira izraz:

$$MSE(A) = p_L \cdot s(t_L) + p_D \cdot s(t_D), \tag{2.11}$$

kjer sta $t_L$ and $t_D$ podmnožici učnih primerov, ki gredo v levo oziroma v desno vejo drevesa, glede na njihovo vrednost atributa $A$, $p_L$ in $p_D$ pa predstavljata deleža primerov, ki gredo levo oziroma desno. $s(t)$ je standardna deviacija vrednosti razreda $c_i$ učnih primerov v podmnožici $t$:

$$s(t) = \sqrt{\frac{1}{N(t)} \sum_{i=1}^{N(t)} (c_i - \overline{c(t)})^2}. \tag{2.12}$$

$\overline{c(t)}$ predstavlja povprečno vrednost razreda v podmnožici $t$.

Minimum izraza (2.11) glede na vse mogoče delitve pri danem attributu vzamemo za oceno kvalitete atributa $A$ in ga upoštevamo v vseh sledečih rezultatih.

### 2.4.1 Umetni problemi

Za preverjanje različnih lastnosti algoritma v raznih okoliščinah smo uporabljali nekaj različnih skupin testnih problemov, ki vsaka vsebuje enega ali več problemov.

**FRAKCIJA:** vsak problem je opisan z zveznimi atributi z vrednostmi od 0 do 1. Vrednost razreda je definirana kot neceli (frakcijski) del vsote $I$ pomembnih attributov: $C = \sum_{j=1}^{I} A_j - \lfloor \sum_{j=1}^{I} A_j \rfloor$. Ti problemi so zvezna posplošitev koncepta parnosti reda $I$ in so opisani z močno odvisnimi zveznimi atributi.

**MODULO-8:** problemi so opisani z množicami atributov, akterih vrednosti so cela števila med 0 in 7. Polovico atributov obravnavamo kot diskretne, drugo polovico pa kot zvezne; vsak zvezen atribut je natančna kopija enega od diskretnih. Vrednost razreda je določena kot vsota $I$ pomembnih atributov po modulu 8: $C = (\sum_{i=j}^{I} A_j) \bmod 8$. Ti problemi so celoštevilčna posplošitev koncepta parnosti (ki je vsota po modulu 2) reda $I$. Pokazali naj bi, kako dobro RReliefF razpoznava močno odvisne atribute in kako rangira enakovredne diskretne in zvezne atribute.

**PARNOST:** vsak problem opisujejo diskretni, logični atributi. $I$ pomembnih atributov definira koncept parnosti: če je njihov parnostni bit enak 0, je vrednost razreda naključno število med 0 in 0.5, sicer ima razred naključno vrednost med 0.5 in 1.

$$C = \left\{ \begin{array}{lcl} rand(0, 0.5) & ; & (\sum\limits_{j=1}^{I} A_j) \bmod 2 = 0 \\ rand(0.5, 1) & ; & (\sum\limits_{j=1}^{I} A_j) \bmod 2 = 1 \end{array} \right\}$$

Tovrstni problemi predstavljajo malo zamegljen koncept parnosti (reda $I$). Testirali naj bi obnašanje algoritma na diskretnih atributih.

**LINEAR:** problem opisujejo zvezni atributi z vrednostmi med 0 in 1, vrednost razreda pa je izračunana z naslednjo linearno formulo: $C = A_1 - 2A_2 + 3A_3 - 3A_4$. Ta problem smo izbrali, da bi primerjali uspešnost algoritma RReliefF z algoritmom MSE, za katerega vemo, da razpoznava linearne odvisnosti.

**COSINUS:** vsebuje zvezne atribute z vrednostmi od 0 do 1. Vrednost razreda določa obrazec: $C = (-2A_2 + 3A_3)\cos(4\pi A_1)$. Ta problem je izbran zaradi nelinearne odvisnosti atributov.

V poskusih, ki so opisani v nadaljevanju, smo uporabljali $I = \{2, 3, 4\}$ pomembnih atributov. Vsakemu od problemov smo dodali tudi nekaj nepomembnih (naključnih) atributov z vrednostmi v istem obsegu, kot jih imajo pomembni atributi.

Za vsak problem smo generirali $N$ primerov in izračunali ocene kot povprečje 10 kratnega prečnega preverjanja. S tem smo zbrali zadosti podatkov, da smo

eliminirali vpliv verjetnosti, ki ga povzroči naključna izbira primerov v algoritmu RReliefF (3. vrstica na sliki 2.3), omogočilo pa nam je tudi izračun statistične pomembnosti razlik med ocenami z dvostranskim t-testom (pri stopnji značilnosti 0.05).

Vse poskuse smo poganjali z istim naborom parametrov (konstanta $m$ v glavni zanki = 250, k-najbližjih= 200, $\sigma = 20$ (glej enačbo (2.10))).

### 2.4.2 Vpliv števila učnih primerov

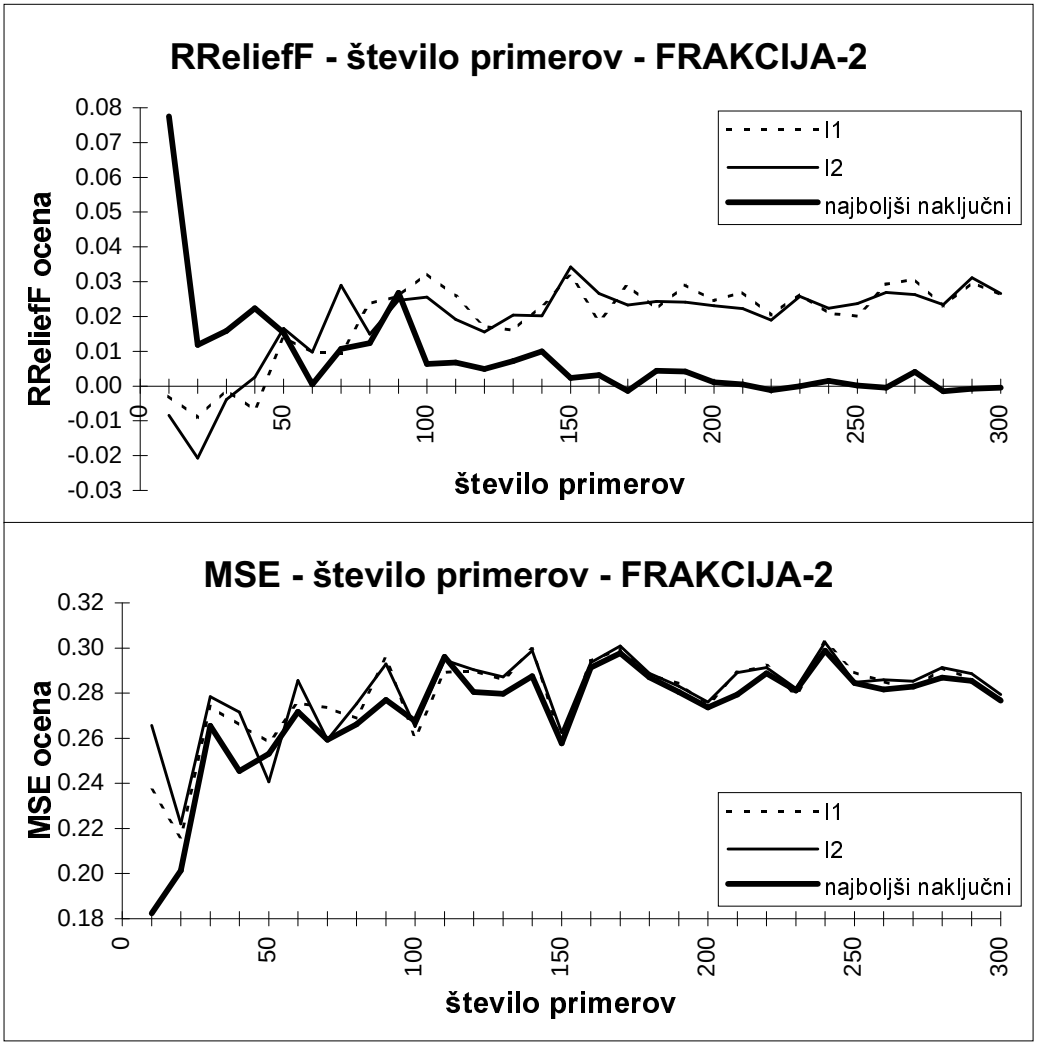
Najprej smo poskusili, kako na oceno kvalitete vpliva število učnih primerov. Generirali smo probleme z $I = \{2, 3, 4\}$ pomembnimi atributi in jim dodali $R = 10 - I$ atributov z naključnimi vrednostmi v istem številskem obsegu. Vsak od problemov je tako opisan z 10 atributi (COSINUS in LINEAR imata $I$ fiksiran na 3 oziroma 4). Ocene kvalitete atributov v skupno 11 problemih smo 10 kratno prečno preverjali, spreminjajoč število primerov od 10 do 1000 v korakih po 10.

Slika 2.4 prikazuje odvisnost ocene kvalitete atributov od števila učnih primerov pri problemu FRAKCIJA z dvema pomembnima atributoma ($I = 2$). *Opozorimo naj, da pripisuje RReliefF večja števila boljšim atributom, MSE pa ravna ravno obratno.*

Slika kaže, da je pri malo primerih (pod 50) naključni atribut z najvišjo oceno (najboljši naključni) ocenjen kot boljši od obeh pomembnih atributov ($I_1$ in $I_2$). S povečanjem števila primerov na 100 smo dosegli mejo, ko sta bila oba pomembna atributa statistično značilno bolje ocenjena kot najboljši naključni atribut. Spodaj vidimo, da MSE ne razloči med pomembnimi in naključnimi atributi, ampak vedno določi enemu od 8 naključnih atributov boljšo (nižjo) oceno kvalitete kot $I_1$ ali $I_2$.

Obnašanje algoritmov RReliefF in MSE je podobno na drugih problemih iz skupin FRAKCIJA, MODULO in PARNOST. Grafe njihovih odvisnosti podajamo v dodatku A, tukaj pa si poglejmo povzetek rezultatov, ki se nahaja v tabeli 2.1. Za vsakega od obeh algoritmov podajamo dve števili, ki pomenita število potrebnih primerov, da je ocena pomembnega atributa, ki je bil ocenjen kot najslabši ($I_w$) oziroma najboljši ($I_b$) med pomebnimi atributi značilno presegla oceno atributa, ki je bil ocenjen kot najboljši med naključnimi atributi. Znak '-' pomeni, da hevristika ni uspela značilno razlikovati med obema skupinama atributov. Opazimo, da število potrebnih primerov narašča z naraščajočo kompleksnostjo (številom pomembnih atributov) problemov.

Medtem, ko so skupine PARNOST, FRAKCIJA in MODULO-8 rešljive za RReliefF, je MSE popolnoma odpovedal. Problem MODULO-8-4 (s štirimi pomembnimi atributi) je pretežak tudi za RReliefF. Zdi se, da 1000 primerov ne zadostuje za tako kompleksen problem, kajti kompleksnost narašča eksponentno: število vrhov v problemskem prostoru za problem MODULO-m-p je namreč $m^p$.




Slika 2.4: Spreminjanje števila pomembnih atributov pri problemu FRAKCIJA z dvema pomembnima atributoma. Pozor: RReliefF pripiše večja števila boljšim atributom, medtem ko MSE dela obratno.

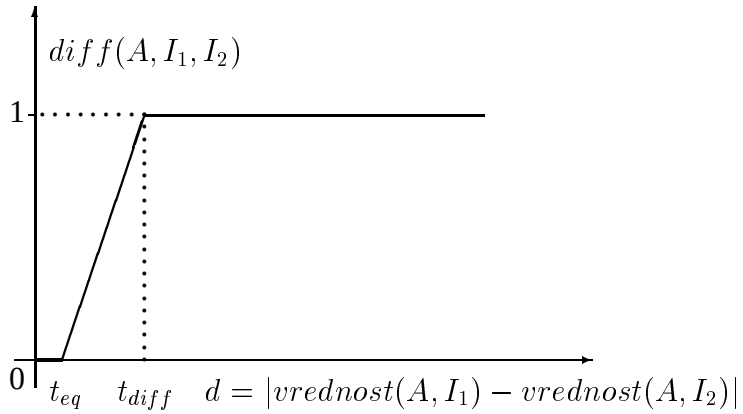
Z dodatnim poskusom z 8000 primeri smo se prepričali, da je RReliefF zmožen ločiti pomembne od naključnih atributov.

V skupini problemov MODULO (slike A.4, A.6 in A.7) je zanimivo tudi, da so diskretni atributi ocenjeni bolje kot njim enaki zvezni atributi. To obnašanje si lahko pojasnimo z definicijo funkcije $diff$ (glej izraza (2.1) in (2.2)). Za primer vzemimo dva učna primera, ki imata vrednost atributa $A_i$ 2 oziroma 5. Če je

Tabela 2.1: Rezultati spreminjanja števila učnih primerov. Številke predstavljajo mejno število primerov, ki jih je hevristika potrebovala, da je lahko značilno ločila najslabši ($I_w$) oziroma najboljši ($I_b$) pomembni atribut od najboljšega naključnega atributa.

| | | RReliefF | | MSE | |
|---|---|---|---|---|---|
| problem | I | $I_w$ | $I_b$ | $I_w$ | $I_b$ |
| FRAKCIJA | 2 | 100 | 100 | - | - |
| | 3 | 300 | 220 | - | - |
| | 4 | 950 | 690 | - | - |
| MODULO-8 | 2+2 | 80 | 70 | - | - |
| | 3+3 | 370 | 230 | - | - |
| | 4+4 | - | - | - | - |
| PARNOST | 2 | 50 | 50 | - | - |
| | 3 | 100 | 100 | - | - |
| | 4 | 280 | 220 | - | - |
| LINEAR | 4 | - | 10 | 340 | 20 |
| COSINUS | 3 | - | 50 | 490 | 90 |

$A_i$ diskreten atribut, potem velja $diff(A_i, 2, 5) = 1$, kajti diskretni vrednosti sta različni. Če pa je $A$ zvezen atribut, velja $diff(A_i, 2, 5) = \frac{|2-5|}{7} \approx 0.43$. Očitno je torej, da so ob takšni definiciji funkcije $diff$ zvezni atributi podcenjeni. Ta problem lahko obidemo s pragovno funkcijo kot jo predlaga (Hong, 1994). Definicijo funkcije $diff$ za zvezne atribute posplošimo, kot to ilustrirata slika 2.5:




Slika 2.5: Pragovna funkcija za zvezne atribute.

Definicijo zapišemo

$$diff(A, I_1, I_2) = \begin{cases} 0 & ; d \leq t_{eq} \\ 1 & ; d > t_{diff} \\ \frac{d - t_{eq}}{t_{diff} - t_{eq}} & ; t_{eq} < d \leq t_{diff} \end{cases} \tag{2.13}$$

kjer $d = |vrednost(A, I_1) - vrednost(A, I_2)|$ pomeni razdaljo med vrednostima atributa dveh primerov, $t_{eq}$ in $t_{diff}$ pa sta pragova, ki ju definira uporabnik. $t_{eq}$ predstavlja maksimalno razdaljo, ko dve vrednosti atributa še vedno upoštevamo za enaki, $t_{diff}$ pa minimalno razdaljo, da jemljemo vrednosti kot različne. Če postavimo $t_{eq} = 0$ in $t_{diff} = \max(A) - \min(A)$ dobimo zopet (2.2). Vrednosti pragov lahko postavi uporabnik za vsak atribut posebej, kar je še posebej smiselno pri merjenih podatkih, lahko se jih naučimo vnaprej upoštevajoč kontekst (Ricci in Avesani, 1995) ali pa jim avtomatsko določimo smiselne vrednosti (Domingos, 1997). Pogled na sliko 2.5 nam porodi misel, da bi bila uporaba sigmoidne funkcije morda še splošnejša, vendar smo se njeni uporabi odrekli, ker njenih parametrov ne znamo interpretirati na tako preprost način. Ocene diskretnih in zveznih atributov v problemih MODULO postanejo pri avtomatski nastavitvi vrednosti pragov popolnoma identične. Rezultate za problem MODULO z dvema pomembnima atributoma in uporabo pragovne funkcije prikazuje slika A.5. Slik za ostale probleme ne navajamo, saj so zelo podobne tistim brez uporabe pragovne funkcije (razmerja ostajajo ista, le številčne vrednosti se spremenijo). Pri drugih poskusih v nadaljevanju tega poglavja rezultatov z uporabo pragovne funkcije ne prikazujemo, ker se ne razlikujejo bistveno od tistih brez nje.

Rezultati pri problemih LINEAR in COSINUS kažejo (glej tudi sliki A.11 in A.12), da imata tako RReliefF kot MSE kar nekaj težav pri ločevanju najslabšega pomembnega atributa ($A_1$ oziroma $A_2$) od najboljšega naključnega, toda MSE je bil vendarle uspešnejši. RReliefF je sicer uspel značilno ločiti obe skupini z 100 ali več učnih primerov, vendar je občasna konica pri oceni naključnih atributov povzročila, da je t-vrednost padla pod mejo značilnosti. MSE je v glavnem ločil obe skupini toda manjše varianca mu daje rahlo prednost. Pri ločevanju najboljšega pomembnega od naključnih atributov (kar je naloga, ki jo zahteva npr. gradnja regresijskih dreves) sta bili uspešni obe hevristiki, vendar je RReliefF potreboval manj učnih primerov. To dejstvo verjetno kompenzira slabši rezultat RReliefFa pri ločevanju najslabšega pomembnega atributa. Razlika med hevristikama je tudi pri določanju vrstnega reda pomembnosti atributov pri problemu COSINUS. Pravilen padajoči vrstni red, ki ga določi RReliefF je $A_1$, $A_3$, $A_2$, medtem ko MSE, ki ne razpozna nelinearnih odvisnosti, razvrsti atribute takole: $A_3$, $A_2$, $A_1$.

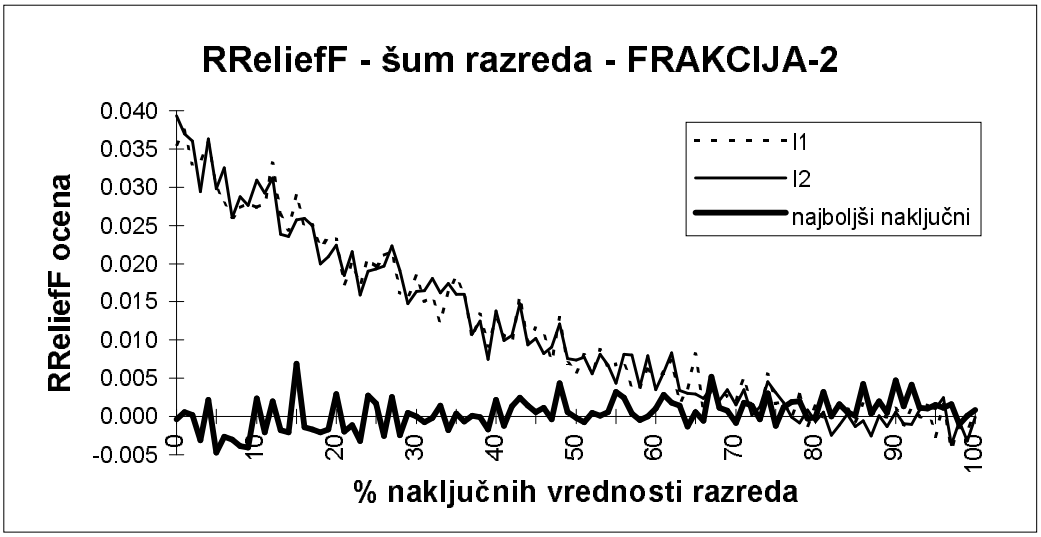
Rezultati na problemih LINEAR in COSINUS pričajo o primerljivi uspešnosti obeh hevristik na relativno preprostih problemih.

Testirali smo tudi druge vrste nelinearnih odvisnosti med atributi (logaritemsko, eksponentno, polinomsko, trigonometrično, ...) in RReliefF se je vedno pokazal kot boljši ali enak MSE.

### 2.4.3 Dodajanje šuma s spreminjanjem vrednosti razreda

Robustnost algoritma RReliefF smo preverili z istimi nastavitvami kot prej (problemi z $I = \{2, 3, 4\}$ pomembnimi atributi, skupaj 10 atributov), le število učnih primerov smo fiksirali na 1000. Podatkom smo dodajali šum tako, da smo določenemu odstotku primerov naključno določili vrednost razreda v istem intervalu, kot so se vrednosti gibale sicer. Dodajali smo od 0 do 100% šuma v korakih po 1%.

Slika 2.6 prikazuje odvisnost ocene RReliefFa za problem FRAKCIJA z $I = 2$ pomembnima atributoma. MSE je bil neuspešen celo brez šuma, zato smo graf izpustili.




Slika 2.6: Ocene kvalitete, ki jih vrne RReliefF, ko podatkom za problem FRAKCIJA z dvema pomembnima atributoma, dodajamo šum v obliki sprememb vrednosti razreda.

Vidimo, da so ocene, ki jih vrne RReliefF robustne, saj loči najslabši pomemben atribut od najboljšega diskretnega celo s 50% pokvarjenih vrednosti razreda.

Grafi odvisnosti za ostale probleme se nahajajo v dodatku B, tabela 2.2 pa povzema rezultate.

Tabela 2.2: Rezultati dodajanja šuma vrednostim razreda. Številke pomenijo mejni odstotek primerov, ki smo jim lahko vrednost razreda določili naključno pri tem pa še vedno dobili statistično pomembne razlike v ocenah med najslabšim ($I_w$) oziroma najboljšim ($I_b$) pomembnim atributom in najboljšim naključnim atributom.

| | | RReliefF | | MSE | |
|---|---|---|---|---|---|
| problem | I | $I_w$ | $I_b$ | $I_w$ | $I_b$ |
| FRAKCIJA | 2 | 53 | 59 | - | - |
| | 3 | 16 | 35 | - | - |
| | 4 | 3 | 14 | - | - |
| MODULO-8 | 2+2 | 64 | 75 | - | - |
| | 3+3 | 52 | 70 | - | - |
| | 4+4 | - | - | - | - |
| PARNOST | 2 | 66 | 70 | - | - |
| | 3 | 60 | 71 | - | - |
| | 4 | 50 | 67 | - | - |
| LINEAR | 4 | - | 66 | 50 | 85 |
| COSINUS | 3 | - | 46 | 36 | 63 |

Številke podajajo maksimalen odstotek pokvarjenih vrednosti razreda, da so bile ocene najslabšega ($I_w$) oziroma najboljšega ($I_b$) pomembnega atributa še vedno značilno boljše od ocen najboljšega naključnega atributa. Znak '-' označuje nezmožnost hevristike, da bi značilno ločila med obema skupinama atributov, celo v primeru, ko šuma sploh ni bilo.

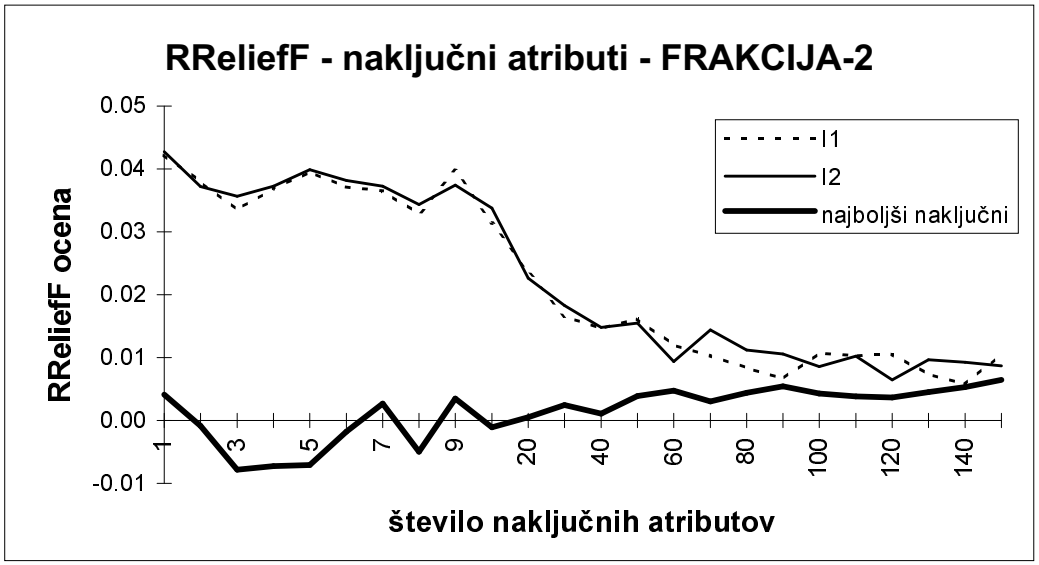
### 2.4.4 Dodajanje naključnih atributov

Za razliko od MSE je RReliefF občutljiv na šum v obliki dodatnih naključnih atributov. To občutljivost smo testirali s podobno testno konfiguracijo kot prej (problemi z $I = \{2, 3, 4\}$ pomembnimi atributi, število učnih primerov fiksirano na 1000), le da smo problemom dodali od 1 do 150 naključnih atributov.

Slika 2.7 prikazuje odvisnost ocene za problem FRAKCIJA z dvema pomembnima atributoma.

Vidimo, da je RReliefF precej neobčutljiv za tovrsten šum, saj mu niti 70 naključnih atributov ni preprečilo, da bi značilno ločil med najslabšim pomembnim in najboljšim naključnim atributom.

Tabela 2.3 povzema rezultate za vse probleme. Stolpiča $I_w$ in $I_b$ podajata




Slika 2.7: Ocene kvalitete, ki jih vrne RReliefF, ko v opis problema FRAKCIJA z dvema pomembnima atributoma, dodajamo naključne atribute.

Tabela 2.3: Rezultati dodajanja naključnih atributov. Številke povedo koliko naključnih atributov smo lahko največ dodali, da smo še vedno statistično značilno ločili najslabši oz. najboljši pomembni atribut od najboljšega naključnega atributa.

| problem | I | $I_{worst}$ | $I_{best}$ |
|---|---|---|---|
| FRACTION | 2 | 70 | 80 |
| | 3 | 10 | 20 |
| | 4 | 4 | 5 |
| MODULO-8 | 2+2 | 50+50 | 100+100 |
| | 3+3 | 7+7 | 20+20 |
| | 4+4 | - | - |
| PARITY | 2 | $>$150 | $>$ 150 |
| | 3 | 50 | 70 |
| | 4 | 10 | 20 |
| LINEAR | 4 | - | $>$150 |
| COSINUS | 3 | - | $>$150 |

največje število naključnih atributov, ki smo jih lahko dodali problemu, da so bile ocene za najslabši oziroma najboljši pomembni atribut še vedno značilno boljše od ocene najboljšega naključnega atributa. Znak '-' pomeni, da RReliefF ni uspel ločiti pomembnih od naključnega atributov niti pri samo enem naključnem atributu.

### 2.4.5 Gradnja regresijskih dreves

Razvili smo sistem za učenje regresijskih dreves, ki rekurzivno od zgoraj navzdol gradi binarna regresijska drevesa. V vsakem vozlišču s hevristiko izberemo atribut in učne primere razdelimo na levo in desno vejo glede na njihovo vrednost pri izbranem atributu. Ta standardni pristop uporabljajo znani sistemi za učenje regresijskih dreves CART (Breiman in sod., 1984), Retis (Karalič, 1991) in M5 (Quinlan, 1993). Njegove podrobnosti in razširitve so opisane v naslednjih poglavjih.

Kot hevristiko za izbiro atributa smo uporabili RReliefF ali MSE (2.11). Naš sistem smo poganjali z dvema različnima naboroma parametrov in postopkov, pri katerih smo se zgledovali po znanih učnih sistemih. Poimenovali smo ju

**točka:** kot model v listih uporablja povprečno vrednost razreda primerov v listu, drevesa režemo z $m$-oceno verjetnosti in

**premica:** ki uporablja v listih porezane linearne modele ter glajenje in rezanje kot v M5.

Učni sistem smo poganjali na umetno generiranih podatkih in na podatkih z zveznim razredom, ki so dosegljivi na repozitoriju na UCI (Murphy in Aha, 1995). Uporabljali smo umetne probleme opisane v tem poglavju (11 problemov s po 1000 primeri, vsak problem opisuje 10 atributov, od teh so 2, 3, ali 4 pomembni, ostali so naključni). Na vsakem problemu smo izvedli 10 kratno prečno preverjanje. Rezultate podaja tabela 2.4.

Primerjamo relativno srednjo kvadratno napako zgrajenih modelov $\phi$:

$$RE_t(\phi) = \frac{R_t(\phi)}{R_t(\mu)}, \text{ kjer je } R_t(\tau) = \frac{1}{N_t}\sum_{i=1}^{N_t}(c_i - \tau(x_i))^2. \tag{2.14}$$

Pri tem predpostavljamo, da je $N_t$ število testnih primerov, da je $i$-ti primer zapisan kot urejen par $(c_i, x_i)$, in je $x_i$ vektor vrednosti atributov, $\tau(x_i)$ je vrednost razreda, ki jo napove model $\tau$, ter $\mu$ model, ki vedno napove povprečno vrednost razreda. Za smiselne modele velja $RE(\phi) < 1$.

Poleg napake smo vključili tudi mero kompleksnosti drevesa $C$, ki jo definiramo kot število vseh pojavitev vseh atributov (ter konstantnih členov v listih

Tabela 2.4: Rezultati gradnje regresijskih dreves, kjer smo za izbor atributov v vozliščih uporabljali RReliefF ali MSE. Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | premica | | | | | točka | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | RReliefF | | MSE | | | RReliefF | | MSE | | |
| problem | $RE$ | $C$ | $RE$ | $C$ | S | $RE$ | $C$ | $RE$ | $C$ | S |
| Frakcija-2 | .34 | 112 | .86 | 268 | + | .52 | 87 | 1.17 | 160 | + |
| Frakcija-3 | .73 | 285 | 1.08 | 440 | + | 1.05 | 174 | 1.62 | 240 | + |
| Frakcija-4 | 1.05 | 387 | 1.10 | 392 | 0 | 1.51 | 250 | 1.65 | 219 | 0 |
| Modulo-8-2 | .22 | 98 | .77 | 329 | + | .06 | 58 | .81 | 195 | + |
| Modulo-8-3 | .58 | 345 | 1.08 | 436 | + | .59 | 166 | 1.58 | 251 | + |
| Modulo-8-4 | 1.05 | 380 | 1.07 | 439 | 0 | 1.52 | 253 | 1.52 | 259 | 0 |
| Parnost-2 | .28 | 125 | .55 | 208 | + | .27 | 7 | .38 | 103 | 0 |
| Parnost-3 | .31 | 94 | .82 | 236 | + | .27 | 15 | .61 | 213 | + |
| Parnost-4 | .35 | 138 | .96 | 283 | + | .25 | 31 | .88 | 284 | + |
| Linear | .02 | 4 | .02 | 4 | 0 | .19 | 59 | .19 | 55 | 0 |
| Cosinus | .27 | 334 | .41 | 364 | + | .36 | 105 | .29 | 91 | 0 |
| Auto-mpg | .13 | 97 | .14 | 102 | 0 | .21 | 27 | .21 | 19 | 0 |
| Auto-price | .14 | 48 | .12 | 53 | 0 | .28 | 16 | .17 | 10 | 0 |
| CPU | .12 | 33 | .15 | 47 | 0 | .42 | 16 | .31 | 10 | 0 |
| Housing | .17 | 129 | .15 | 177 | 0 | .31 | 33 | .23 | 23 | 0 |
| Servo | .25 | 53 | .28 | 55 | 0 | .24 | 7 | .33 | 9 | 0 |

drevesa pri linearni formuli ali točkovnem modelu), kjerkoli v drevesu. Stolpič označen s S kaže na signifikantnost razlike med modeli zgrajenimi z MSE in RReliefFom. 0 pomeni, da razlika ni pomembna pri stopnji značilnosti 0.05, '+' pomeni, da je model z algoritmom RReliefF značilno boljši, (in '-' bi označeval značilno boljši rezultat modela z MSE).

V načinu premica (leva stran tabele 2.4) so modeli generirani s pomočjo RRelieFa na umetnih problemih večinoma značilno boljši kot modeli zgrajeni z MSE. Na UCI podatkih so modeli primerljivi (RReliefF je boljši na treh, MSE pa na dveh problemih, toda z neznačilnimi razlikami). Kompleksnost modelov induciranih z RReliefFom je večinoma manjša tako na umetnih kot na UCI problemih.

V točkovnem načinu je slika na umetnih problemih podobna, na UCI podatkih pa je bil z neznačilnimi razlikami trikrat boljši MSE in enkrat RReliefF. V povprečju so bila drevesa dobljena z MSE manjša na realnih in lažjih umetnih (LINEAR, COSINUS) problemih.

V tem poglavju smo pokazali primernost nekratkovidne regresijske hevristike RReliefF za ocenjevanje atributov in gradnjo regresijskih dreves, v naslednjem pa se bomo ukvarjali s konstruktivno indukcijo.

# 3

# Konstruktivna indukcija

*Ocenjevati, pomeni uničevati in slediti toku časa; ustvarjati, pomeni graditi in obrniti tok časa.*

*Jacques Attali*

Algoritmi, ki se učijo opisov konceptov iz primerov, v določeni meri predvidevajo razporeditev primerov v problemskem prostoru (Utgoff, 1986) in na podlagi teh predpostavk, ki jih imenujemo pristranost algoritma, izberejo podprostore, ki predstavljajo iskani koncept. Tako je za uspešno učenje koncepta nujno, da se primeri, ki ga predstavljajo, nahajajo v enem ali več območjih problemskega prostora, ki smo jih sposobni opisati z danim opisnim jezikom.

Problem nastopi, ko primeri niso strnjeni v območjih, ki jih je dani opisni jezik sposoben opisati. Takrat je inducirani opis koncepta zapleten, težko razumljiv, nepopoln, preveč prilagojen učnim primerom in zato nepravilen. Vzroka za to sta dva:

- šum v učnih primerih in/ali
- neprimerna opisni jezik in formalizem.

S problemom neprimernega opisnega jezika se ukvarja konstruktivna indukcija.

Pojem *konstruktivna indukcija* je prvič uporabljen v (Michalski, 1986b). Po njem konstruktivna indukcija v nasprotju s selektivno indukcijo, ki za indukcijo opisa koncepta izbira le med njegovimi že znanimi lastnostmi, generira tudi nove lastnosti.

Tej definiciji lahko oporekamo, saj ni povsem jasno, kaj so nove lastnosti: le-te bodo nujno izražene s starimi, gre torej zgolj za vprašanje opisnega jezika.

Na konstruktivno indukcijo lahko gledamo tudi kot na poskus transformacije originalnega problemskega prostora v prostor, kjer so učni primeri razporejeni bolj urejeno oziroma bolj ugodno glede na dano pristranost našega učnega algoritma. Iščemo primeren jezik za predstavitev danega učnega problema (Pfahringer, 1994).

Kakorkoli, konstruktivna indukcija lahko bistveno pripomore k uspehu učenja (Kibler in Langley, 1990). Naštejmo nekaj prednosti (Ragavan in Rendell, 1993; Robnik, 1993; Yang in sod., 1991):

- z gradnjo vmesnih konceptov (atributov, jezikovnih gradnikov) pripomoremo k večji kompaktnosti in razumljivosti končnega koncepta,
- dobimo izraznejši opisni jezik,
- dosežemo večjo točnost klasifikacije.

Pri tem pa se nam obetajo tudi težave (Pfahringer, 1994; Robnik, 1993):

- mogoča je prevelika prilagoditev učnim primerom, kar pomeni, da so konstrukti pretirano kompleksni in zapleteni in zato teže razumljivi, ter
- večja računska kompleksnost učenja, ki sledi iz poskusov generiranja novih koristnih konstruktov.

Različna področja strojnega učenja se različno lotevajo konstruktivne indukcije in jo tudi različno poimenujejo. Ločimo v grobem dva pristopa (Robnik, 1995).

**Transformacijska indukcija:** dano množico pravil postopno preoblikujemo z uporabo transformacijskih operatorjev.

**Operatorska indukcija:** konstrukte gradimo z uporabo operatorjev na osnovnih atributih ali že izpeljanih konstruktih.

Pregled tehnik, ki se uporabljajo na različnih področjih, najdemo v (Robnik, 1995), tukaj podajamo le kratek pregled literature nekaj najpomembnejših pristopov.

Transformacijski pristop uporabljata pri učenju odločitvenih pravil znana sistema INDUCE (Michalski in Dietterich, 1983; Michalski, 1986a) in Duce (Muggleton, 1990). Predstavnik tega pristopa pri odločitvenih pravilih je opisan v (Yang in sod., 1991), pa tudi razvrščanje konceptov v skupine (conceptual clustering) lahko uvrstimo v to skupino.

Operatorska indukcija je pri učenju pravil uporabljena v sistemu CiPF (Pfahringer, 1994), pri učenju odločitvenih dreves v sistemu LFC (Ragavan in Rendell, 1993), poskus z Bayesovim klasifikatorjem pa je opisan v (Kononenko, 1991).

V induktivnem logičnem programiranju ločimo rajši med preoblikovalnim pristopom, ko uvedemo nov predikat zaradi njegove zanimivosti ali večje kompaktnosti teorije, ter pristopom na zahtevo, ko poskušamo s hevristikami odkriti primere, kjer opisni jezik ne zadošča za opis popolne in pravilne teorije. Pregled konstruktivne indukcije v zvezi z induktivnim logičnim programiranjem najdemo v (Stahl, 1993).

Pri nevronskih mrežah nevroni na skritih nivojih pomenijo konstrukte oziroma vmesne koncepte. Malo drugačen pristop je nevronska mreža s funkcijsko povezavo (Pao, 1989).

V literaturi nismo zasledili uporabe konstruktivne indukcije v regresiji, vendar se nam zdi operatorska konstrukcija primernejša za to področje. V nadaljevanju bomo opisali odprta vprašanja uporabe konstruktivne indukcije v regresiji, nakazali in opisali nekaj rešitev ter predstavili poizkuse, s katerimi smo preverjali naše odločitve.

## 3.1 Operatorji

Očitno se nekaterih konceptov brez pravih operatorjev ne bomo mogli naučiti in narobe: nekateri koncepti bodo z uporabo pravih operatorjev preprosto naučljivi. Na to, katere operatorje bomo uporabili za učenje nekega koncepta, lahko gledamo kot na izbiro potrebnega predznanja za dani problem.

Možnosti za izbiro operatorjev je mnogo, npr:

- **logični:** konjunkcija, disjunkcija, negacija, implikacija, ekvivalenca, ...,
- **aritmetični:** seštevanje, odštevanje, množenje deljenje, polinomi, trigonometrične operacije, logaritmi, eksponenti, ...,
- **pragovni:** koliko pogojev iz množice pogojev je resničnih, ali je to število večje, manjše od praga,
- **preštevanje**: število primerov, ki zadoščajo nekemu pogoju,
- **veriga:** če imamo neko tranzitivno relacijo in tvorijo primeri zaradi tega neko zaporedje (verigo), tvori operator atribut, ki določi posebne elemente v verigi: prvega, zadnjega, srednjega, N-tega, ali pa dolžino verige,
- **odvisnost:** konstrukt opisuje relacije med posameznimi atributi: monotono povezanost, monotono povezanost pri nekem pogoju, približno monotono povezanost, korelacijo, korelacijo pri pogoju, ...,

- **aritmetične relacije:** enakost, neenakost, večji, manjši,
- **diskretizacija:** konstrukt označuje, da lahko vrednost atributa pade v nek interval,
- **združitev vrednosti** nominalnih atributov v množice,
- $\mathbf{3}\sigma$: numeričnim atributom določimo ali pripada vrednost "zdravemu" intervalu $[\mu - 3\sigma, \mu + 3\sigma]$ (odstopanje $3\sigma$ od srednje vrednosti),
- **kartezični produkt:** vse kombinacije vrednosti dveh atributov itd.

Ker v literaturi nismo zasledili uporabe konstruktivne indukcije v regresiji, smo se odločili za uporabo splošnih operatorjev. Glede na izkušnje s klasifikacijskimi sistemi smo se odločili za konjunkcijo, dodali pa smo še seštevanje in množenje. Poglejmo si algoritem, s katerim smo gradili konstrukte.

## 3.2 Postopek gradnje konstruktov

Gradnja konstruktov je zaradi tipično izredno velikega problemskega prostora časovno zelo zahtevna operacija, zato jo moramo omejiti. Uporabili smo omejeno iskanje v širino (beam search) z omejeno globino, ki je kompromis med požrešnimi metodami, ki pogosto padejo v lokalni ekstrem, in metodo izčrpnega preiskovanja, ki je časovno nesprejemljiva. Globino iskanja smo omejili s tem, da smo dolčili največjo velikost konstrukta. Psevdo kodo procedure za gradnjo konstruktov najdemo na sliki 3.1.

Konstrukte gradimo postopno in ločeno - vsak operator posebej. To ni smiselno le zaradi manjšega problemskega prostora pač pa tudi zaradi razumljivosti konstruktov. Izkušnje so namreč pokazale, da si ljudje ne znamo predstavljati pomena konstruktov, sestavljenih iz različnih operatorjev. Vsak konstrukt poskušamo razširiti z vsemi konstruktivnimi gradniki (atributi oziroma vrednostmi atributov) in ocenimo kvaliteto dobljene množice. Izberemo najboljše ter z njimi ponovimo postopek na naslednji globini. Poglabljanje ustavimo, ko doseže največjo dovoljeno globino $maxDepth$. Konstrukt z najboljšo vrednostjo kriterijske funkcije, kar smo jih našli v postopku poglabljanja, vrnemo kot rezultat.

Bistvena za dobro delovanje omejenega preiskovanja v širino je ocena kvalitete konstruktov, ki mora izbrati najperspektivnejše konstrukte za razširjanje.

## 3.3 Ocenjevanje konstruktov

Pri hevristiki za ocenjevanje konstruktov si želimo podobne lastnosti kot pri dobrih hevristikah za ocenjevanje atributov. V konstruktivni indukciji imamo zaradi

*Postopek:* gradnja konstruktov
*Vhod:* množica atributov $A$, konstruktivni operator $O$
*Izhod:* najboljši konstrukt $c_{best}$

1. $c_{best}$ = najboljši atribut ;
2. v $G$ pripravi gradnike (atribute oz. vrednosti atributov iz $A$)
3. v $C$ pripravi $beamSize$ najboljših gradnikov iz $G$ ;
4. **for** $depth$ := 1 **to** $maxDepth$ **do begin**
5. $newC = O(C, G)$ ;
6. oceni konstrukte iz $newC$ ;
7. **if** najboljši iz $newC$ je boljši od $c_{best}$ **then**
8. $c_{best}$ = najboljši iz $newC$ ;
9. v $C$ izberi $beamSize$ najboljših konstruktov $newC$ ;
10. **end** ;

Slika 3.1: Psevdo koda procedure za izgradnjo konstruktov.

večje izrazne moči in večjega prostora, ki ga preiščemo (oversearching) (Quinlan in Cameron-Jones, 1995), še večje težave s problemom prevelike prilagoditve učnim podatkom. Želimo si, da bi zato hevristika upoštevala tudi velikost preiskanega prostora.

Niti RReliefF niti MSE ne zadoščata našim zahtevam, zato smo ju poskušali prilagoditi. Dela smo se lotili s principom najkrajše dolžine opisa.

### 3.3.1 Princip MDL

Princip najmanše dolžine opisa (minimum description length - MDL) izhaja iz Occamovega rezila, ki pravi, da je nesmiselno z več početi to, kar lahko storimo z manj oziroma, da je najpreprostejša razlaga tudi najverjetnejša. Teoretično je utemeljen s kompleksnostjo Kolmogorova (Li in Vitányi, 1993).

Princip je bil že mnogokrat uporabljen v strojnem učenju npr. (Quinlan in Rivest, 1989; Kovačič, 1994; Mehta in sod., 1995; Kononenko, 1995). Bistvo pristopa je zakodirati dani problem kar najkrajše in izbrati tisto rešitev problema, ki ima najkrajšo kodo.

To v konstruktivni indukciji pomeni, da moramo izbrati učinkovito kodiranje konstruktov in njihove kvalitete. Zanima nas le dolžina kode, ne pa konkretna koda (Rissanen, 1983; Kovačič, 1994). Ker delamo z regresijskimi problemi, se nujno srečamo tudi s problemom kodiranja realnih vrednosti. Uporabljamo rešitev uporabljeno v (Karalič, 1995) in si vnaprej določimo številčno preciznost s katero

delamo. Realno število $x$ se takole spremeni v celo:

$$Z(x) = \lfloor \frac{x}{preciznost} \rfloor, \tag{3.1}$$

ki ga nato zakodiramo s formulo (Rissanen, 1983):

$$\begin{aligned} Rissanen(0) &= 1 \\ Rissanen(n) &= 1 + \log_2(n) + \log_2(\log_2(n)) + ... + \log(2.865064..) \end{aligned} \tag{3.2}$$

kjer vsota vsebuje le pozitivne člene.

Izbrali smo naslednje kodiranje konstruktov:

$$MDL_{konstrukt} = koda(konstrukt) + koda(Ocena_{konstrukta}) \tag{3.3}$$

Pri konstruktu najprej posebej zakodiramo, s kašnim tipom konstrukta imamo opravka (konjunkcija, vsota, produkt, posamičen atribut), zato je lahko specifična koda za vsak tip krajša, kot če bi izbrali splošnejše kodiranje poljubnih izrazov (v npr. drevesni ali RPN notaciji):

$$koda(konstrukt) = koda(Tip_{konstrukta}) + koda(Specifičen_{konstrukt}) \tag{3.4}$$

Dolžina kode tipa konstrukta je logaritem števila operatorjev bitov:

$$koda(Tip_{konstrukta}) = \log_2(število_{operatorjev}) \tag{3.5}$$

Specifične kode za posamezne vrste konstruktov sestavimo po naslednjih pravilih.

**Posamičen atribut:** različno kodiramo diskretne in zvezne atribute:

- pri diskreten atributu najprej zakodiramo izbrani atribut, nato pa še izbiro podmnožice njegovih vrednosti, ki gredo v levo vejo drevesa ($A$ je število atributov):

$$koda(Posamičen_{atribut}) = \log_2(A) + \log_2 \binom{V_A}{v_A} \tag{3.6}$$

- pri zveznem atributu prav tako zakodiramo izbrani atribut, nato pa še mejo vrednosti, ki gredo v levo poddrevo. Ker želimo, da so vse mejne točke enako verjetne (in naj imajo enako dolgo kodo), uporabimo namesto Rissanenove formule za naravna števila (3.2), ki daje

večjim številom daljše kode, logaritem števila različnih vrednosti atributa. Število različnih vrednosti atributa je določeno z dolžino intervala vrednosti, ki jih lahko zavzame atribut (kar ugotovimo iz učne množice), ter s preciznostjo, ki jo želimo. Dolžina kode je naslednja:

$$koda(Posamičen_{atribut}) = \log_2(A) + \log_2 \frac{Interval_A}{preciznost} \tag{3.7}$$

**Konjunkcija:** ker je konjunkcija vedno sestavljena iz zaporedja členov, ki so lahko bodisi vrednost atributa (pri diskretnih atributih) ali interval vrednosti atributa (pri zveznih atributih), vzamemo naslednjo dolžino kode:

$$koda(Konjunkcija) = \log_2(Dolžina_{konj.}) + \sum_{i=1}^{Dolž.\ konj.} koda(i\text{-}ti\ člen) \tag{3.8}$$

pri tem posamične člene kodiramo takole:

- pri diskreten atributu kodiramo za kateri atribut gre in njegovo vrednost

$$koda(Člen_{konjunkcije}) = \log_2(A) + \log_2(V_A) \tag{3.9}$$

- pri zveznem atributu zakodiramo za kateri atribut gre in dve meji (spodnjo in zgornjo), ki določita pripadnost intervalu vrednosti atributa

$$koda(Člen_{konjunkcije}) = \log_2(A) + 2 \cdot \log_2 \frac{Interval_A}{preciznost} \tag{3.10}$$

Vrednosti, ki gredo levo v drevesu, nam ni treba kodirati, saj ima konjunkcija le dve možni vrednosti in lahko predpostavimo, da gre npr. prva vedno v levo vejo.

**Vsota in produkt:** izraz je sestavljen iz zaporedja zveznih atributov, vzamemo naslednjo dolžino kode ($A_{zv}$ je število zveznih atributov):

$$koda(Izraz) = \log_2(Dolž_{izraza}) + \sum_{i=1}^{Dolž.} [\log_2(A_{zv}) + \log_2 \frac{Interval_{izraza}}{preciznost}] \tag{3.11}$$

Pri kodiranju celotnega konstrukta (izraz (3.3)) nam ostane še del, s katerim kodiramo oceno kvalitete konstrukta.

### 3.3.2 Kodiranje ocene kvalitete konstrukta

Pri kodiranju kvalitete konstrukta bi morali upoštevati tudi kvaliteto poddreves, ki bi ju lahko zgradili z delitvijo primerov glede na vrednost pri konstruktu. V času, ko gradimo konstrukt in izbiramo najboljšega, te informacije še ni na voljo in si z vidika računske kompleksnosti tudi ne moremo privoščiti takšnega izračuna. Zadovoljiti se moramo s približkom, to je z oceno kvalitete konstrukta, ki jo vrne hevristična funkcija.

Za RReliefF to intuitivno ni čisto neosnovano. Predstavljamo si namreč lahko, da RReliefFova ocena izraža podobnost gostot porazdelitve konstrukta in razreda. Če v vozlišče izberemo dobro ocenjeni konstrukt, torej povečamo možnost, da bosta porazdelitve modelov, ki ju opisujeta poddrevesi (kakršna koli sta že) še bližje porazdelitvi razreda.

MSE je ocena, ki predpostavlja v levi in desni veji list s točkovnim modelom (povprečno vrednostjo razreda), zato je približek informacije o morebitnih celotnih poddrevesih zelo verjetno slabši.

Kodiranje ocene kvalitete, ki jo vrne RReliefF, temelji na dejstvu, da je ta ocena močno povezana s funkcijami nečistoče. Poglejmo si izpeljavo za klasifikacijski ReliefF (Kononenko, 1994).

Izhajamo iz enačbe (2.3), ter pri pogojnem delu obeh verjetnosti izpustimo to, da gre za najbližja primere. Dobimo

$$\begin{aligned} W'[A] &= P(\text{različna vrednost A}|\text{različen razred}) \\ &- P(\text{različna vrednost A}|\text{isti razred}) \end{aligned} \tag{3.12}$$

kar zaradi preprostejše izpeljave zapišemo malo drugače:

$$\begin{aligned} W'[A] &= P(\text{ista vrednost A}|\text{isti razred}) \\ &- P(\text{ista vrednost A}|\text{različen razred}) \end{aligned} \tag{3.13}$$

Po analogiji z definicijami (2.6), (2.7) in (2.8) zapišemo:

$$P_{equalA} = P(\text{ista vrednost A}) \tag{3.14}$$

$$P_{equalC} = P(\text{isti razred}) \tag{3.15}$$

$$P_{equalC|equalA} = P(\text{isti razred}|\text{ista vrednost A}) \tag{3.16}$$

ter analogno z (2.9) z Bayesovim pravilom dobimo:

$$W'[A] = \frac{P_{equalC|equalA}P_{equalA}}{P_{equalC}} - \frac{(1 - P_{equal|equalA})P_{equalA}}{1 - P_{equalC}} \tag{3.17}$$

Za izbiranje z vračanjem v strogem smislu veljajo za verjetnosti naslednje enakosti:

$$P_{equalC} = \sum_C P(C)^2 \tag{3.18}$$

$$P_{equalC|equalA} = \sum_V \left( \frac{P(V)^2}{\sum_V P(V)^2} \times \sum_C P(C|V)^2 \right) \tag{3.19}$$

Z njihovo uporabo dobimo:

$$W'[A] = \frac{P_{eqvalA}}{P_{equalC}(1 - P_{equalC})} \times Ginigain'(A) \tag{3.20}$$

$$= const \times \sum_V P(V)^2 \times Ginigain'(A) \tag{3.21}$$

kajti za vse atribute $A$ je člen $\frac{1}{P_{equalC}(1-P_{equalC})}$ konstanten, Ginigain' pa je definiran kot

$$Ginigain'(A) = \sum_V \left( \frac{P(V)^2}{\sum_V P(V)^2} \times \sum_C P(C|V)^2 \right) - \sum_C P(C)^2 \tag{3.22}$$

in je za razred $C$ ter vrednosti $V$ atributa $A$ močno koreliran s prispevkom Gini indeksa (Breiman in sod., 1984). Ta namesto faktorja

$$\frac{P(V)^2}{\sum_V P(V)^2} \tag{3.23}$$

uporablja

$$\frac{P(V)}{\sum_V P(V)} = P(V) \tag{3.24}$$

Enačba 3.21 kaže na močno povezanost ocene, ki jo vrneta ReliefF in RReliefF s prispevkom Gini indeksa, ta pa je spet močno povezan z entropijo. Za kodiranje pa potrebujemo prav entropijo!

Iz enačbe 3.20 dobimo

$$Ginigain'(A) = \frac{P_{equalC}(1 - P_{equalC})}{P_{eqvalA}} \times W'[A] \tag{3.25}$$

Člena $P_{equalC}$ in $P_{equalA}$ smo dejansko že izračunali z algoritmom RReliefF (slika 2.3), zato lahko zapišemo

$$koda(Ocena_{RReliefF}) = E_{RReliefF} \times GiniGain' \tag{3.26}$$

kjer je $E_{RReliefF}$ konstanta, ki pomeni približek pretvorbe med prispevkom Gini indeksa in entropijo.

Za MSE prav tako velja, da je njegova ocena povezana z Gini indeksom (Kononenko in sod., 1996), oziroma prispevkom gini indeksa. Definirajmo prispevek MSE (glej tudi enačbi (2.11) in (2.12), ki definirata $MSE(A)$ in $s(t)$):

$$MSEgain(A) = s(t) - MSE(A), \tag{3.27}$$

in definirajmo dolžino kode

$$koda(Ocena_{MSE}) = E_{MSE} \times MSEgain \tag{3.28}$$

kjer je $E_{MSE}$ spet konstanta, ki pomeni približek pretvorbe med prispevkom MSE in entropijo.

Določili smo vse potrebno za kodiranje, zato si zdaj empirično oglejmo, kako deluje konstruktivna indukcija.

## 3.4 Poizkusi s konstrukcijo

Konstrukcijo smo preizkušali po podobnem scenariju kot uporabo hevristik pri gradnji regresijskih dreves v razdelku 2.4.5. Uporabili smo dve množici parametrov za gradnjo drevesa (v razdelku 2.4.5 poimenovani točka in premica) ter štiri različne hevristike za ocenjevanje konstruktov: RReliefF, MSE, MDL z RReliefF ter MDL z MSE.

Za konstrukcijo smo vedno uporabili isti nabor parametrov: širino preiskovanja $beamSize = 20$, največjo dolžino konstruktov 3, konstruke smo generirali le v korenu drevesa, najboljših 5 pa smo lahko uporabili tudi drugje v drevesu. Pri uporabi principa MDL smo določili $preciznost = 0.01$, obe konstanti za pretvorbo v entropijo $E_{RReliefF}$ in $E_{MSE}$ pa smo postavili na 1000. Z nekaj predhodnimi poskusi smo ugotovili, da so te vrednosti smiselne, njihova analiza in avtomatsko določanje pa nam bosta zadali še nekaj dela v prihodnosti.

Rezultati zbrani v tabelah 3.1 in 3.2 so povprečje desetkratnega prečnega preverjanja.

Najprej primerjajmo rezultate dreves s konstrukcijo s tistimi brez nje, ki se nahajajo v tabeli 2.4. Ugotovimo, da je konstrukcija v nekaterih primerih (FRAKCIJA, MODULO) bistveno pomagala k uspehu učenja tako z RRelifFom kot z MSE. Tam, kjer smo zajeli najpomembnejše odvisnosti, so zgrajena drevesa bistveno manjša, sicer pa so nekoliko večja.

Kot smo že omenili, pomeni konstrukcija pogled naprej, in če je širina iskanja zadostna, s slepim preiskovanjem najdemo pomembne kombinacije atributov, ki jih razpozna tudi MSE. Pri bolj zapletenih odvisnostih (reda 3), širina preiskovanja ne zadošča, da bi slepo sestavili najpomembnejše odvisnosti, zato MSE ne uspe.

Velike razlike med napakami pri problemih reda 3 in reda 4 si razlagamo z omejitvijo najdaljših konstruktov na 3 člene. Inačici z RReliefFom sta na ta način

Tabela 3.1: Rezultati gradnje regresijskih dreves s konstruktivno indukcijo v načinu premica. Kot hevristike za oceno kvalitete konstruktov smo uporabljali RReliefF, MSE, MDL z RReliefFom (MdlRFF) ter MDL z MSE (MdlMSE). Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | premica | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RReliefF | | MSE | | MdlRRF | | MdlMSE | |
| problem | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ |
| Frakcija-2 | .04 | 41 | .01 | 9 | .06 | 45 | .01 | 9 |
| Frakcija-3 | .07 | 71 | .81 | 514 | .06 | 66 | .80 | 473 |
| Frakcija-4 | .66 | 343 | 1.08 | 677 | .74 | 351 | 1.11 | 648 |
| Modulo-8-2 | .11 | 17 | .11 | 9 | .12 | 17 | .11 | 9 |
| Modulo-8-3 | .14 | 36 | .09 | 27 | .19 | 41 | .09 | 27 |
| Modulo-8-4 | .99 | 275 | 1.04 | 681 | 1.00 | 237 | .98 | 646 |
| Parnost-2 | .28 | 114 | .55 | 224 | .28 | 114 | .55 | 224 |
| Parnost-3 | .31 | 88 | .82 | 278 | .31 | 90 | .82 | 278 |
| Parnost-4 | .35 | 135 | .96 | 316 | .35 | 135 | .96 | 316 |
| Linear | .02 | 4 | .02 | 4 | .02 | 4 | .02 | 4 |
| Cosinus | .29 | 400 | .42 | 563 | .29 | 384 | .42 | 560 |
| Auto-mpg | .14 | 117 | .14 | 298 | .14 | 117 | .14 | 295 |
| Auto-price | .13 | 68 | .23 | 163 | .15 | 69 | .23 | 163 |
| CPU | .11 | 45 | .11 | 70 | .11 | 40 | .11 | 73 |
| Housing | .20 | 191 | .15 | 286 | .18 | 199 | .15 | 286 |
| Servo | .26 | 46 | .28 | 45 | .33 | 46 | .28 | 45 |

sestavili delne rešitve iz treh členov, kar pa ni zadoščalo za bistveno manjšo napako.

Ko primerjamo uspeh različnih ocen za ocenjevanje konstruktov vidimo, da ostaja razmerje med MSE in RReliefF enako kot pri ocenjevanju atributov. MDL z RReliefFom daje približno enako dobre rezultate kot RReliefF tako glede relativne napake kot glede kompleksnosti dreves. Isto velja tudi za primerjavo med MSE in MDL z MSE, kar morda kaže na ne najbolj posrečeno izbiro parametrov $E_{RReliefF}$ in $E_{MSE}$.

Na realnih problemih pomembnih razlik pri napaki ni bilo, v oči pa padejo precej manj kompleksna drevesa inducirana z RReliefFom in MDL z RReliefFom v načinu premica, kar kaže na to, da sta hevristiki zajeli pomembne značilnosti problemov in sta jih lahko precej krajše izrazili.

Tabela 3.2: Rezultati gradnje regresijskih dreves s konstruktivno indukcijo v načinu točka. Kot hevristike za oceno kvalitete konstruktov smo uporabljali RReliefF, MSE, MDL z RReliefFom (MdlRFF) ter MDL z MSE (MdlMSE). Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | točka | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RReliefF | | MSE | | MdlRRF | | MdlMSE | |
| problem | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ |
| Frakcija-2 | .14 | 34 | .02 | 36 | .11 | 35 | .02 | 36 |
| Frakcija-3 | .06 | 69 | 1.31 | 267 | .06 | 66 | 1.28 | 246 |
| Frakcija-4 | 1.19 | 278 | 1.72 | 327 | 1.22 | 296 | 1.76 | 315 |
| Modulo-8-2 | .00 | 37 | .00 | 37 | .00 | 37 | .00 | 37 |
| Modulo-8-3 | .02 | 67 | .02 | 60 | .01 | 61 | .02 | 60 |
| Modulo-8-4 | 1.04 | 226 | 1.61 | 382 | 1.24 | 183 | 1.45 | 357 |
| Parnost-2 | .27 | 7 | .38 | 105 | .27 | 7 | .38 | 105 |
| Parnost-3 | .27 | 15 | .58 | 217 | .27 | 15 | .58 | 218 |
| Parnost-4 | .25 | 30 | .88 | 289 | .25 | 31 | .88 | 289 |
| Linear | .20 | 63 | .15 | 58 | .16 | 55 | .15 | 57 |
| Cosinus | .30 | 116 | .32 | 105 | .29 | 116 | .32 | 105 |
| Auto-mpg | .25 | 33 | .24 | 22 | .22 | 36 | .24 | 22 |
| Auto-price | .21 | 17 | .59 | 149 | .23 | 19 | .59 | 148 |
| CPU | .26 | 12 | .30 | 13 | .24 | 12 | .29 | 13 |
| Housing | .33 | 38 | .20 | 30 | .27 | 35 | .20 | 30 |
| Servo | .23 | 7 | .33 | 9 | .23 | 15 | .33 | 9 |

Ogledali smo si konstrukcijo v notranjih vozliščih, zdaj si poglejmo še modeliranje v listih drevesa.

# 4

# Modeli v listih drevesa

*Mar ne pokvarimo stvari s tem, ko jih izrazimo?*

*Virginia Woolf*

Prvi sistem za učenje regresijskih dreves CART (Breiman in sod., 1984) je v listih za napovedovanje vrednosti regresijske funkcije uporabljal točkovni model - povprečje vrednosti razreda učnih primerov v listu.

V sistemu Retis (Karalič, 1991) so bili v listih uporabljeni linearni modeli, kar se je v večini primerov izkazalo kot dobro. Linearni model predstavlja hiper ravnino v prostoru učnih primerov in je definiran kot:

$$C = a_0 + \sum_{i=1}^{\text{št. zveznih atributov}} a_i \cdot A_i \tag{4.1}$$

V postopku gradnje regresijskega drevesa določimo koeficiente $a_i$ z enim od algoritmov za linearno regresijo, npr. s postopkom dekompozicije singularnih vrednosti (Press in sod., 1988). Slabost tega pristopa je, da v linearni formuli vedno nastopajo vsi zvezni atributi, čeprav morda vsi nimajo vpliva na vrednost razreda. To tudi poveča občutljivost modelov na šum.

Sistem M5 (Quinlan, 1993) se je lotil tega problema tako, da je popravil oceno srednje absolutne napake modela $\tau$ s hevrističnim členom:

$$error_t(\tau) = A_t(\tau) \times \frac{N_t + \nu}{N_t - \nu} \text{ kjer je } A_t(\tau) = \frac{1}{N_t} \sum_{i=1}^{N_t} |c_i - \tau(x_i)|. \tag{4.2}$$

kjer je $N_t$ število učnih primerov v lisu $t$, $\nu$ pa število koeficientov $a_i$ v linearni formuli 4.1. Na ta način se poveča ocena napake modelom z mnogo koeficienti v listih z malo primeri. M5 nato izvede požrešni algoritem, ki na vsakem koraku iz formule (4.1) izpusti člen, ki najmanj poveča napako (4.2). Kot model izbere formulo z najmanjšo napako, kar jih je našel med preiskovanjem. Na ta

način so formule v listih tipično precej krajše in bolj razumljive. Hevristični člen $\frac{N_t+\nu}{N_t-\nu}$ v izrazu (4.2) je intuitivno sicer jasen, vendar teoretično neutemeljen, zato smo v način izbire linearnih modelov poskusili uvesti princip MDL. Izbrali smo naslednje kodiranje.

$$MDL_{model} = koda(model) + \sum_{i=1}^{N_t} koda(napaka_i), \tag{4.3}$$

Napako zakodiramo z Rissanenovo formulo 3.2:

$$koda(napaka_i) = 1 + Rissanen(\frac{|c_i - \tau(x_i)|}{preciznost_{napake}}) \tag{4.4}$$

linearen model pa tako, da najprej zakodiramo $\omega$ atributov, ki nastopajo v njej, nato pa še njim pripadajoče koeficiente:

$$koda(model) = \log_2 \binom{A}{\omega} + \sum_{i=1}^{\omega}(1 + Rissanen(\frac{|a_i|}{preciznost_{modela}})) \tag{4.5}$$

En bit dodajamo za predznak, saj smo ga izgubili pri pretvorbi v naravno število.

Zato, da poiščemo model z najkrajšo kodo, smo uporabili Powellovo minimizacijo, ki je eden najbolj uporabljanih algoritmov za večdimenzionalno minimizacijo (Press in sod., 1988).

Z uporabo MDL principa koeficientov ne eliminiramo direktno, pač pa jih postavimo na 0.

Poglejmo si zdaj obnašanje našega postopka in ga primerjajmo z metodama v Retisu in M5.

## 4.1 Preizkušanje modelov

Poskuse smo izvajali podobno kot pri gradnji regresijskih dreves v razdelku 2.4.5. Uporabili smo parametre za gradnjo drevesa, ki smo jih v razdelku 2.4.5 poimenovali premica, za ocenjevanje atributov smo uporabljali RReliefF in MSE ter tri postopke gradnje linearnih formul: Retis, M5 ter MDL. Konstruktivne indukcije nismo uporabljali, drevesa smo rezali z $m$-oceno ($m = 2$), za MDL pa smo postavili parametra natančnosti na smiselne vrednosti ($preciznost_{napake} = 0.01$, $preciznost_{modela} = 0.1$).

Rezultati, zbrani v tabelah 4.1 in 4.2, so povprečje desetkratnega prečnega preverjanja.

Razlike med postopki pri relativni srednji kvadratni napaki statistično niso pomembne, pač pa je v večini primerov signifikantno večja kompleksnost dreves

Tabela 4.1: Rezultati gradnje regresijskih dreves z različnimi postopki gradnje linearnih modelov. Za oceno kvalitete atributov smo uporabljali RReliefF. Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | RReliefF | | | | | |
|---|---|---|---|---|---|---|
| | Retis | | M5 | | MDL | |
| problem | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ |
| Frakcija-2 | .27 | 161 | .31 | 103 | .32 | 77 |
| Frakcija-3 | .78 | 311 | .72 | 227 | .74 | 217 |
| Frakcija-4 | 1.53 | 384 | 1.50 | 305 | 1.54 | 355 |
| Modulo-8-2 | .06 | 183 | .03 | 149 | .25 | 84 |
| Modulo-8-3 | 1.19 | 343 | 1.17 | 228 | 1.25 | 283 |
| Modulo-8-4 | 1.29 | 259 | 1.33 | 213 | 1.38 | 273 |
| Parnost-2 | .25 | 7 | .27 | 7 | .29 | 9 |
| Parnost-3 | .27 | 15 | .27 | 15 | .32 | 23 |
| Parnost-4 | .27 | 31 | .25 | 31 | .29 | 33 |
| Linear | .00 | 11 | .00 | 4 | .00 | 4 |
| Cosinus | .18 | 227 | .18 | 143 | .18 | 165 |
| Auto-mpg | .15 | 29 | .14 | 28 | .16 | 17 |
| Auto-price | .20 | 43 | .28 | 22 | .22 | 46 |
| CPU | .16 | 16 | .15 | 11 | .21 | 10 |
| Housing | .21 | 54 | .20 | 42 | .20 | 33 |
| Servo | .23 | 14 | .25 | 10 | .23 | 11 |

zgrajnih z Retisom. Razlike med načinoma M5 in MDL v glavnem niso signifikantne. Z analizo zgrajenih dreves in modelov smo ugotovili, da je MDL postavil kar nekaj koeficientov pri naključnih atributih na 0 (oziroma na vrednosti blizu 0), nakaj pa jih je precej zmanjšal, vendar ne dovolj. Očitno ima naša kriterijska funkcija (dolžina kode) zelo veliko lokalnih minimumov, kjer se postopek minimizacije ustavi. Ta problem smo skušali rešiti s simuliranim ohlajanjem (simmulated annealing), vendar rešitve niso bile bistveno boljše, le čas minimizacije je skokovito narastel.

Zaključimo lahko, da za izgradnjo linearnih modelov v listih regresijskega drevesa trenutno še ne moremo določiti najboljše rešitve. Postopek uporabljen v M5 s požrešnim izločanjem parametrov učinkovito gradi kratke modele in dosega napake, ki ne zaostajajo za drugimi pristopi. MDL zaenkrat še ni upravičil vseh

Tabela 4.2: Rezultati gradnje regresijskih dreves z različnimi postopki gradnje linearnih modelov. Za oceno kvalitete atributov smo uporabljali MSE. Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | MSE | | | | | |
|---|---|---|---|---|---|---|
| | Retis | | M5 | | MDL | |
| problem | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ |
| Frakcija-2 | 1.12 | 312 | 1.15 | 238 | 1.14 | 243 |
| Frakcija-3 | 1.72 | 400 | 1.66 | 282 | 1.69 | 352 |
| Frakcija-4 | 1.77 | 401 | 1.79 | 309 | 1.79 | 355 |
| Modulo-8-2 | .84 | 340 | .84 | 216 | .81 | 268 |
| Modulo-8-3 | 1.58 | 428 | 1.64 | 279 | 1.70 | 407 |
| Modulo-8-4 | 1.62 | 441 | 1.56 | 295 | 1.68 | 424 |
| Parnost-2 | .38 | 105 | .38 | 105 | .42 | 95 |
| Parnost-3 | .58 | 218 | .58 | 218 | .69 | 184 |
| Parnost-4 | .88 | 289 | .88 | 289 | .97 | 224 |
| Linear | .00 | 11 | .00 | 4 | .00 | 4 |
| Cosinus | .30 | 259 | .29 | 150 | .29 | 189 |
| Auto-mpg | .15 | 46 | .15 | 39 | .15 | 24 |
| Auto-price | .19 | 52 | .20 | 32 | .21 | 48 |
| CPU | .21 | 9 | .24 | 8 | .25 | 5 |
| Housing | .20 | 57 | .20 | 44 | .19 | 40 |
| Servo | .34 | 17 | .28 | 13 | .49 | 16 |

naših pričakovanj. Težava je verjetno v načinu kodiranja. Dolžina kode je zaradi nezveznosti težavna za optimizacijske metode, vprašljiva pa je tudi uporaba Rissanenove formule ter uporabe parametra za preciznost, ki spremeni zvezno vrednost v naravno število. Slednje je vsekakor širše (tudi filozofsko) vprašanje, ki zadeva količino informacije v realnem številu in računanje realnih števil z diskretnim (Turingovim) strojem.

Dosedaj smo opisali celoten postopek gradnje regresijskega drevesa, v naslednjem poglavju si poglejmo še rezanje.

# 5

# Rezanje regresijskih dreves

*Negujete svoj vrtiček!*
*Voltaire*

Klasičen pristop k učenju tako regresijskih kot odločitvenih dreves najprej zgradi nekoliko preveliko drevo, nato pa poreže veje brez zadostne statistične podpore. Razlog za takšno početje je v tem, da smo v procesu gradnje drevo preveč specializirali oziroma prilagodili učnim podatkom. Ker si želimo drevo, ki je splošno in bo dobro delovalo na neznanih primerih, dodamo še korak generalizacije, to pa je rezanje. Drug pogled na rezanje pravi, da smo s tem, ko smo zgradili preveliko drevo naredili pogled naprej (lookahead), po koncu gradnje pa to preveliko drevo analiziramo in odrežemo neperspektivne veje. V literaturi se pojavljajo trije pristopi k rezanju regresijskih dreves.

CART (Breiman in sod., 1984) uporablja za rezanje parameterski kriterij kompleksnosti drevesa. Za nastavitev parametra $\alpha$ moramo izvesti večkratno gradnjo celotnega drevesa s prečnim preverjanjem ali uporabiti posebno, dovolj veliko množico učnih podatkov. V našem primeru, ko uporabljamo pri gradnji drevesa kar nekaj časovno precej zahtevnih tehnik (izbira atributov, konstruktivna indukcija, linearni modeli v listih), se nam zdi večkratno prečno preverjanje za nastavitev parametra nesprejemljivo, za realne probleme pa sploh nimamo na voljo zadosti podatkov, še posebej, ker obstaja enako dobro ali boljše, manj kompleksno rezanje (Karalič, 1991; Karalič in Cestnik, 1991).

Retis (Karalič, 1991) uporablja algoritem naknadnega rezanja z $m$-oceno verjetnosti, ki temelji na algoritmu za rezanje odločitvenih dreves (Niblett in Bratko, 1990), in je bil mnogokrat uspešno uporabljen v praksi. Gre za ocenitev pričakovane verjetnosti napake na neznanih primerih. V vsakem vozlu ocenimo napako, ki bi jo naredili, če bi bil ta vozel list, in napako, če temu vozlu pustimo poddrevesi. Če je ocena prve napake manjša, drevo na tem mestu obrežemo (odstranimo poddrevesi), sicer ne. Postopek rekurzivno, od listov proti korenu, izve-

demo na celotnem drevesu. Pri tem postopku je zelo pomembna zanesljiva ocena pričakovane napake. Ocenjujemo jo z bayesovskim pristopom k ocenjevanju verjetnosti (Cestnik, 1991), ki pride še posebej do izraza pri ocenjevanju verjetnosti iz majhnega števila podatkov. Z $m$-oceno iz $N$ poskusov ocenimo verjetnost $p$ dogodka $X$ takole:

$$p = \frac{N}{N+m} p_X + \frac{m}{N+m} p_a \tag{5.1}$$

kjer je $p_X$ relativna frekvenca dogodka $X$ v $N$ poskusih, $p_a$ pa vnaprejšnja (a priorna) verjetnost dogodka $X$. Parameter $m$ določa zaupanje v vnaprejšnjo verjetnost $p_a$. S spreminjanjem tega parametra kontroliramo velikost drevesa (čim večji je $m$, tem bolj zaupamo vnaprejšnjim verjetnostim in je zato drevo manjše). Retis uporablja $m$-oceno verjetnosti tudi pri klasifikaciji. V naši implementaciji smo se temu izognili in smo z $m$-oceno ocenjevali le napako. S tem smo se izignili anomaliji opisani v (Cestnik in Bratko, 1991; Karalič, 1991), ko lahko drevo pri velikih vrednostih parametra $m$ začne naraščati in ga je zato potrebno rezati zaporedno z naraščajočim zaporedjem vrednosti $m$.

M5 (Quinlan, 1993) uporablja podoben rekurziven algoritem rezanja kot Retis in prav tako primerja oceno napake, če poddrevesa ohrani ali poreže. Napako ocenjuje s srednjo absolutno napako, ki jo pomnoži s hevrističnim členom $\frac{N_t+\nu}{N_t-\nu}$ (glej izraz 4.2). M5 nima parametra, s katerim bi kontrolirali velikost drevesa.

## 5.1 Uporaba principa MDL pri rezanju

Po zgledu klasifikacijskih dreves (Kononenko, 1997) smo poskušali uvesti princip MDL tudi v rezanje regresijskih dreves.

Za rezanje smo uporabili podoben rekurziven algoritem kot Retis, vendar v duhu principa MDL, v vsakem notranjem vozlišču drevesa namesto napake primerjamo dolžino kodiranja poddreves z dolžino kode modela, ki bi ga uporabili, če bi drevo na tem mestu porezali. Dolžino kode modela izračunamo s formulo (4.3), za kodiranje poddrevesa pa si predstavljamo, da zapišemo vsa vozlišča po nekem dogovorjem redu (na primer najprej levo v globino - preorder). Kodi vsakega vozlišča dodamo 1 bit, ki pove, ali gre za notranje vozlišče ali za list. Za notranje vozlišče zakodiramo atribut oziroma konstrukt (obrazec (3.4)), v listih pa kodiramo uporabljeni model (obrazec (4.3)).

Določiti moramo tudi vrednosti obeh parametrov za preciznost (napake in modela). Z njima kontroliramo velikost drevesa.

## 5.2 Poizkusi z rezanjem

Poizkuse smo izvajali podobno kot pri gradnji regresijskih dreves v razdelku 2.4.5. Uporabili smo dva nabora parametrov za gradnjo drevesa, ki smo ju v razdelku 2.4.5 poimenovali točka in premica, za ocenjevanje atributov smo uporabljali RReliefF in MSE, dodali pa smo tudi konstruktivno indukcijo, kjer smo konstrukte ocenjevali po principu MDL (z RReliefFom).

Primerjali smo rezanje z $m$-oceno in rezanje po principu MDL. Želeli smo preveriti, kakšno je najboljše možno rezanje z dano proceduro. Najprej smo za vsak problem zgradili deset dreves po metodi desetkratnega prečnega preverjanja, nato pa smo jih obrezali za vsak nabor parametrov za rezanje. Izračunali smo povprečje in vzeli minimalno povprečno napako za oceno najboljšega možnega rezanja. V tabelah 5.1, 5.2 in 5.3 poleg napake podajamo tudi povprečno kompleksnost drevesa, ki smo jo dosegli z istimi parametri.

Za $m$-oceno smo že poznali smiselen nabor vrednosti, zato smo spreminjali vrednost paramera $m$ po logaritemski skali od 0.0001 do 10000 ($80$ vrednosti in $80 \times 10 = 800$ rezanj). Pri MDL rezanju smo po logaritemski skali spreminjali $preciznost_{modela}$ od 0.0002 do 100 (30 vrednosti), ter $preciznost_{napake}$ relativno glede na interval vrednosti razreda od 0.0002 do 1 (20 vrednosti), torej skupno $20 \times 30 \times 10 = 6000$ rezanj.

Splošen vtis, ki ga dobimo v vseh treh tabelah je, da dajeta obe rezanji drevesa s približno enako relativno napako (brez statistično pomembnih razlik), vendar so pri enaki napaki drevesa obrezana z MDL večinoma precej manj kompleksna. Sklepamo, da z MDL rezanjem lahko dobimo kvalitetna in manjša drevesa kot z $m$-oceno. Težava ostaja določitev parametrov. Pri rezanju z $m$-oceno poznamo smiselne vrednosti in nastavitev parametra na 2, daje zanesljive (čeprav ne najboljše možne) rezultate v mnogih problemih. Za MDL rezanje še nimamo tovrstnih izkušenj. Najbolje bi bilo, če bi obe preciznosti nastavil strokovnjak za dani problem. To ostaja za zdaj odprto vprašanje.

Tabela 5.1: Rezultati rezanja regresijskih dreves z $m$ oceno in s principom MDL ter z variranjem njunih parametrov. Drevesa smo gradili v načinu premica, atribute smo ocenjevali s hevristikama RReliefF in MSE. Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | premica | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RReliefF | | | | MSE | | | |
| | $m$ | | MDL | | $m$ | | MDL | |
| problem | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ |
| Frakcija-2 | .26 | 119 | .26 | 118 | 1.00 | 3 | .78 | 74 |
| Frakcija-3 | .65 | 347 | .64 | 277 | 1.00 | 3 | 1.00 | 4 |
| Frakcija-4 | 1.00 | 3 | 1.00 | 3 | 1.00 | 3 | 1.00 | 3 |
| Modulo-8-2 | .15 | 108 | .15 | 108 | .77 | 333 | .63 | 167 |
| Modulo-8-3 | 1.00 | 3 | .98 | 52 | 1.00 | 4 | 1.01 | 2 |
| Modulo-8-4 | 1.00 | 2 | 1.00 | 12 | 1.00 | 3 | 1.00 | 4 |
| Parnost-2 | .27 | 7 | .27 | 7 | .38 | 117 | .40 | 76 |
| Parnost-3 | .27 | 15 | .27 | 16 | .57 | 248 | .63 | 94 |
| Parnost-4 | .25 | 31 | .25 | 31 | .86 | 347 | .83 | 91 |
| Linear | .02 | 4 | .02 | 4 | .02 | 4 | .02 | 4 |
| Cosinus | .14 | 416 | .11 | 200 | .30 | 205 | .25 | 160 |
| Auto-mpg | .16 | 27 | .15 | 15 | .13 | 24 | .14 | 30 |
| Auto-price | .22 | 65 | .20 | 12 | .15 | 55 | .11 | 36 |
| CPU | .11 | 21 | .13 | 21 | .18 | 13 | .18 | 11 |
| Housing | .18 | 41 | .18 | 31 | .19 | 61 | .17 | 45 |
| Servo | .16 | 53 | .16 | 38 | .28 | 13 | .26 | 42 |

Tabela 5.2: Rezultati rezanja regresijskih dreves z $m$ oceno in s principom MDL ter z variranjem njunih parametrov. Drevesa smo gradili v načinu točka, atribute smo ocenjevali s hevristikama RReliefF in MSE. Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | točka | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | RReliefF | | | | MSE | | | |
| | $m$ | | MDL | | $m$ | | MDL | |
| problem | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ |
| Frakcija-2 | .24 | 269 | .25 | 222 | .94 | 40 | .84 | 97 |
| Frakcija-3 | .86 | 110 | .87 | 95 | 1.00 | 1 | 1.00 | 1 |
| Frakcija-4 | 1.00 | 3 | 1.00 | 1 | 1.00 | 1 | 1.00 | 2 |
| Modulo-8-2 | .00 | 127 | .00 | 127 | .80 | 144 | .71 | 138 |
| Modulo-8-3 | 1.00 | 8 | 1.00 | 1 | 1.00 | 1 | 1.01 | 4 |
| Modulo-8-4 | 1.00 | 14 | 1.00 | 8 | 1.00 | 1 | 1.01 | 2 |
| Parnost-2 | .27 | 7 | .27 | 8 | .38 | 117 | .41 | 65 |
| Parnost-3 | .27 | 15 | .27 | 15 | .57 | 248 | .64 | 109 |
| Parnost-4 | .25 | 31 | .25 | 31 | .86 | 347 | .83 | 100 |
| Linear | .14 | 546 | .13 | 169 | .12 | 334 | .13 | 278 |
| Cosinus | .20 | 272 | .23 | 185 | .26 | 145 | .26 | 91 |
| Auto-mpg | .19 | 86 | .18 | 45 | .19 | 120 | .19 | 53 |
| Auto-price | .26 | 75 | .31 | 33 | .14 | 50 | .15 | 16 |
| CPU | .23 | 51 | .25 | 18 | .18 | 47 | .21 | 19 |
| Housing | .27 | 90 | .26 | 30 | .21 | 34 | .21 | 27 |
| Servo | .16 | 40 | .18 | 22 | .32 | 8 | .25 | 5 |

Tabela 5.3: Rezultati rezanja regresijskih dreves z $m$ oceno in s principom MDL ter z variranjem njunih parametrov. Pri gradnji dreves smo uporabljali konstruktivno indukcijo, konstrukte smo ocenjevali po principu MDL z RReliefFom, v listih smo uporabljali točkovni ali linearen model. Predstavljeni sta relativna srednja kvadratna napaka ($RE$) in kompleksnost regresijskega drevesa ($C$).

| | konstrukcija - MDL z RReliefF | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | premica | | | | točka | | | |
| | $m$ | | MDL | | $m$ | | MDL | |
| problem | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ | $RE$ | $C$ |
| Frakcija-2 | .03 | 23 | .02 | 26 | .04 | 269 | .04 | 147 |
| Frakcija-3 | .06 | 80 | .04 | 61 | .03 | 138 | .03 | 124 |
| Frakcija-4 | .98 | 116 | .83 | 151 | .96 | 66 | .86 | 78 |
| Modulo-8-2 | .11 | 14 | .11 | 15 | .00 | 43 | .00 | 43 |
| Modulo-8-3 | .17 | 39 | .17 | 39 | .00 | 84 | .00 | 84 |
| Modulo-8-4 | 1.00 | 2 | .94 | 66 | 1.00 | 2 | .97 | 34 |
| Parnost-2 | .27 | 7 | .27 | 7 | .27 | 7 | .27 | 7 |
| Parnost-3 | .27 | 15 | .27 | 16 | .27 | 15 | .27 | 15 |
| Parnost-4 | .25 | 31 | .25 | 31 | .25 | 31 | .25 | 32 |
| Linear | .02 | 4 | .02 | 4 | .11 | 561 | .12 | 147 |
| Cosinus | .18 | 404 | .19 | 307 | .24 | 296 | .28 | 151 |
| Auto-mpg | .14 | 28 | .14 | 26 | .18 | 43 | .18 | 52 |
| Auto-price | .21 | 64 | .19 | 25 | .17 | 90 | .24 | 26 |
| CPU | .09 | 23 | .12 | 28 | .15 | 27 | .23 | 15 |
| Housing | .23 | 138 | .18 | 56 | .25 | 77 | .26 | 27 |
| Servo | .15 | 59 | .16 | 40 | .15 | 47 | .19 | 25 |

# 6

# Sklep

*To pismo je tako dolgo zato, ker nisem imel časa, da bi napisal krajšega.*
*Blaise Pascal*

Analizirali smo obstoječe hevristike za ocenjevanje atributov in na podlagi algoritma ReliefF razvili nekratkovidno hevristiko RReliefF za ocenjevanje atributov v regresiji. Analiza nas je privedla tudi do enotnega pogleda na ocenjevanje atributov v klasifikaciji in regresiji. Empirično smo preverili delovanje novega algoritma glede na različno število učnih primerov, glede na šum pri vrednostih razreda in glede na število naključnih atributov, ki nastopajo v opisu problema. Rezultati so pokazali, da je RReliefF sposoben odkriti močne odvisnosti med atributi, če te obstajajo, sicer pa je enakovreden najpogosteje uporabljani hevristiki MSE. Izkazalo se je, da potrebuje za uspešno delovanje enako število učnih primerov kot MSE in da je robusten glede šuma in dodanih naključnih atributov.

Vse te lepe lastnosti so nas prepričale, da smo preizkusili uporabo nove hevristike pri gradnji regresijskih dreves. Zgradili smo sistem za učenje regresijskih dreves, ki vključuje znane načine ocenjevanja atributov, gradnje modelov v listih in rezanja. Pokazalo se je, da je uporaba hevristike RReliefF smiselna in da lahko prinese pomembno zmanjšanje napake in kompleksnosti dreves.

Učni sistem smo nadgradili s konstruktivno indukcijo. V vozliščih drevesa smo uporabljali operatorje konjunkcije, seštevanja in množenja. V konstrukcijo smo uvedli princip MDL in v ta namen razvili in analizirali kodiranje ocenjevanja kvalitete konstruktov z algoritmoma RReliefF in MSE. Rezultati so pokazali, da lahko uporaba konstruktov v vozliščih pomembno zmanjša napako ter kompleksnost induciranih dreves. Z uporabo konstruktivne indukcije ter linearne regresije v listih smo dobili z algoritmom RReliefF na realnih problemih pri isti napaki precej manjša drevesa kot z algoritmom MSE, kar kaže na sinergijsko delovanje konstruktivne indukcije in algoritma RReliefF. Uporaba principa MDL ni upravičila naših pričakovanj o manjših konstruktih. V prihodnosti bomo morali

preučiti druge načine kodiranja in predstavitve konstruktov.

Primerjali smo različne načine gradnje linearnih modelov v listih. Zaradi možnosti uporabe MDL smo razvili kodiranje modelov. Pri testiranju smo v primerjavi z neporezanimi modeli dobili signifikantno manjše modele pri isti napaki, vendar nam da kodiranje realnih števil močno nezvezno kriterijsko funkcijo z mnogimi vrhovi. Njena optimizacija je zato težavna, rezultati pa ne tako zanesljivi, kot bi si želeli. Metodo bo potrebno še dopolniti.

Razviti kodiranji konstruktov in modelov smo vključili v kodiranje regresijskih dreves, ki smo ga razvili zaradi rezanja dreves po principu MDL. Novo rezanje smo primerjali z uveljavljeno metodo rezanja z $m$-oceno verjetnosti. Drevesa obrezana z obema načinoma napovedujejo s približno enako napako, vendar so tista, obrezana z metodo po principu MDL, večinoma manj kompleksna. Težava pri novi metodi je nastavitev parametrov za preciznost, kar pa se bo, upamo, rešilo, ko bomo imeli z novo metodo več izkušenj.

Povzetek našega dela nakazuje smernice tudi za nadaljnje delo.

Z vse večjo popularnostjo analize velikih podatkovnih zbirk z metodami umetne inteligence (data mining), je postala važna določitev podmnožico pomembnih atributov. Menimo, da je RReliefF dorastel tej nalogi, treba pa bo preučiti različne načine vzorčenja podatkov, določanja števila najbližjih primerov in njihovo učinkovitejše iskanje s k-d drevesi.

V konstruktivno indukcijo je potrebno vgraditi še več različnih operatorjev. Poskušali bomo razviti hevristike, ki bodo avtomatsko detektirale potrebnost uporabe posameznih operatorjev. Z uporabo našega sistema v praksi si bomo pridobili izkušnje glede določanja različnih parametrov sistema, več pozornosti pa bomo namenili tudi avtomatski nastavitvi teh vrednosti.

RReliefF želimo uporabiti v inkrementalnem učenju in pri analizi časovnih vrst. Menimo, da bi lahko z njim detektirali spremembo konteksta.

Uporaba principa MDL v regresiji ostaja najpomembnejše odprto vprašanje tega dela. V literaturi ne obstaja kodiranje realnih števil, ki bi v regresijo pripeljalo lepoto tega principa, kot jo poznamo iz kodiranj diskretnih vrednosti. V zadnjem času se je uveljavil pogled, ki napako učnega sistema dekomponira na napako zaradi pristranosti in napako zaradi variance (Geman in sod., 1992; Kohavi in Wolpert, 1996). Ta pogled in navidezni paradoks pretiranega iskanja (Quinlan in Cameron-Jones, 1995), je postavil v drugo luč tudi princip MDL. V duhu kompleksnosti Kolmogorova (Li in Vitányi, 1993) in Occamovega reza je potrebno upoštevati ne le kompleksnost opisa hipoteze, pač pa tudi algoritma, ki jo je generiral in celo kompleksnost izračuna, ki jo je opravil na njej. V teh smereh bodo šle tudi naše raziskave.

# Literatura


Breiman, L., Friedman, L., Olshen, R., in Stone, C. (1984). *Classification and regression trees*. Wadsworth Inc., Belmont, California.

Cestnik, B. (1991). *Ocenjevanje verjetnosti v avtomatskem učenju*. Doktorska disertacija, Univerza v Ljubljani, Fakulteta za elektrotehniko in računalništvo.

Cestnik, B. in Bratko, I. (1991). On estimating probabilities in tree pruning. V Kodratoff, Y., urednik, *Proceedings of European working session on learning (EWSL-91)*, strani 138–150, Porto, Portugal. Springer-Verlag.

Domingos, P. (1997). Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*. (to appear).

Elomaa, T. in Ukkonen, E. (1994). A geometric approach to feature selection. V De Raedt, L. in Bergadano, F., ,urednika *Proceedings of European Conference on Machine Learning*, strani 351–354. Springer Verlag.

Geman, S., Bienenstock, E., in Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–48.

Hong, S. J. (1994). Use of contextual information for feature ranking and discretization. Technical Report RC19664, IBM.

Hunt, E., Martin, J., in Stone, P. (1966). *Experiments in Induction*. Academic Press, New York.

Karalič, A. (1991). Avtomatsko učenje regresijskih dreves iz nepopolnih podatkov. Magistersko delo, Univerza v Ljubljani, Fakulteta za elektrotehniko in računalništvo.

Karalič, A. (1995). *First Order Regression*. Doktorska disertacija, University of Ljubljana, Faculty of Computer and Information Science.

Karalič, A. in Cestnik, B. (1991). The bayesian approach to tree-structured regression. V *Proceedings of ITI-91*, strani 155–160, Cavtat, Croatia.

Kibler, D. in Langley, P. (1990). Machine learning as an experimental science. V Dietterich, T. G. in Shavlik, J. W., urednika, *Readings in Machine Learning*, strani 45–59. Morgan Kaufman.

Kira, K. in Rendell, L. A. (1992). A practical approach to feature selection. V D.Sleeman in P.Edwards, urednika, *Proceedings of International Conference on Machine Learning*, strani 249–256. Morgan Kaufmann.

Kohavi, R. in Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss function. V *Proceedings of the XIII International Conference on Machine Learning*. Morgan Kaufmann.

Kononenko, I. (1991). Semi-naive bayesian classifier. V Kodratoff, Y., urednik, *Proceedings of European working session on learning (EWSL 91)*, Porto, Portugal. Springer-Verlag.

Kononenko, I. (1994). Estimating attributes: analysis and extensions of Relief. V De Raedt, L. in Bergadano, F., urednika, *Machine Learning: ECML-94*, strani 171–182. Springer Verlag.

Kononenko, I. (1995). On biases in estimating multi-valued attributes. V *Proceedings of the IJCAI-95*, strani 1034–1040. Morgan Kaufmann.

Kononenko, I. (1997). The minimum description length based decision tree pruning. Technical report, University of Ljubljana, Faculty of Information and Computer Science.

Kononenko, I., Robnik Šikonja, M., in Pompe, U. (1996). ReliefF for estimation and discretization of attributes in classification, regression and ILP problems. V Ramsay, A., urednik, *Artificial Intelligence: Methodology, Systems, Applications: Proceedings of AIMSA'96*, strani 31–40. IOS Press.

Kononenko, I., Šimec, E., in Robnik-Šikonja, M. (1997). Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence*, 7:39–55.

Kovačič, M. (1994). *Stohastic Inductive Logic Programming*. Doktorska disertacija, University of Ljubljana, Faculty of Computer and Information Science.

Li, M. in Vitányi, P. (1993). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York.

Mantaras, R. (1989). ID3 revisited: A distance based criterion for attribute selection. V *Proceedings of Int. Symp. Methodologies for Intelligent Systems*, Charlotte, North Carolina, USA.

Mehta, M., Rissanen, J., in Agrawal, R. (1995). MDL-based decision tree pruning. V *Proceedings of KDD-95*, strani 216–221.

Michalski, R. (1986a). A theory and methodology of inductive learning. V Michalski, R., Carbonnel, J., in Mitchell, T., uredniki, *Machine Larning: An Artificial Intelligence Approach*, strani 83–134. Kaufman.

Michalski, R. (1986b). Understanding the nature of learning: issues and research directions. V Michalski, R., Carbonnel, J., in Mitchell, T., uredniki, *Machine Larning: An Artificial Intelligence Approach, Volume II*, strani 3–25. Kaufman.

Michalski, R. S. in Dietterich, T. G. (1983). A comparative review of selected methods for learning from examples. V Michalski, R., Carbonnel, J., in Mitchell, T., uredniki, *Machine Larning: An Artificial Intelligence Approach*, strani 41–82. Kaufman.

Muggleton, S. (1990). *Inductive Acquisition of Expert Knowledge*. Addison-Wesley, Workingham, England.

Murphy, P. in Aha, D. (1995). UCI repository of machine learning databases. (http://www.ics.uci.edu/ mlearn/MLRepository.html).

Niblett, T. in Bratko, I. (1990). Learning decision rules in noisy domains. V Bramer, M., urednik, *Developement in Expert Systems*. Cambridge University Press.

Pao, Y.-H. (1989). *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley.

Pfahringer, B. (1994). Controlling constructive induction in CiPF: An MDL approach. V De Raedt, L. in Bergadano, F., urednika, *Machine Learning: ECML-94*, strani 242–256. Springer Verlag.

Pompe, U. in Kononenko, I. (1995). Linear space induction in first order logic with relief. V Della Riccia, G., Kruse, R., in Viertl, R., uredniki, *Mathematical and Statistical Methods in Artificial Intelligenc, CISM Courses and Lectures No.363*. Springer Verlag.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., in Flannery, B. P. (1988). *Numerical recipes in C*. Cambridge University Press.

Quinlan, J. R. (1993). Combining instance-based and model-based learning. V *Proceedings of the X. International Conference on Machine Learning*, strani 236–243. Morgan Kaufmann.

Quinlan, J. R. in Cameron-Jones, R. M. (1995). Oversearching and layered search in empirical learning. V *Proceedings of the IJCAI-95*, strani 1019–1024. Morgan Kaufmann.

Quinlan, J. R. in Rivest, R. L. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248.

Ragavan, H. in Rendell, L. (1993). Lookahead feature construction for learning hard concepts. V *Proceedings of the X. International Machine Learning Conference*, strani 252–259.

Redner, R. R. in Walker, H. F. (1984). Mixture densities, maximum likehood and the EM algorithm. *SIAM Review*, 26(2):195–239.

Ricci, F. in Avesani, P. (1995). Learning a local similarity metric for case-based reasoning. V *Proceedings of the international conference on case-based reasoning (ICCBR-95)*, Sesimbra, Portugal.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–4310.

Robnik, M. (1993). Konstruktivna indukcija z odločitvenimi drevesi. Univerza v Ljubljani, Fakulteta za elektrotehniko in računalništvo. (diplomsko delo).

Robnik, M. (1995). Konstruktivna indukcija v strojnem učenju. *Elektrotehniški vestnik*, 62(1):43–49.

Robnik Šikonja, M. in Kononenko, I. (1996). Context sensitive attribute estimation in regression. V Kubat, M. in Widmer, G., urednika, *Proceedings of ICML'96 workshop on Learning in context sensitive domains*, strani 43–52. Morgan Kaufmann.

Smyth, P. in Goodman, R. (1990). Rule induction using information theory. V Piatetsky-Shapiro, G. in Frawley, W., urednika, *Knowledge Discovery in Databases*. MIT Press.

Smyth, P. in Mellstrom, J. (1992). Detecting novel classes with applications to fault diagnosis. V Sleeman, D. in Edwards, P., urednika, *Machine Learning, Proceedings of the IX. International Workshop*, strani 416–425.

Stahl, I. (1993). An overview of predicate invention techniques in ILP. Technical Report BRA 6020: Inductive Logic Programming, ESPRIT.

Utgoff, P. E. (1986). Shift of bias for inductive concept learning. V Michalski, R., Carbonnel, J., in Mitchell, T., uredniki, *Machine Larning: An Artificial Intelligence Approach, Volume II*, strani 107–148. Kaufman.

Yang, D., Rendell, L., in Blix, G. (1991). A scheme for feature construction and a comparison of empirical methods. V *Proceedings of the Twelft International Joint Conference on Artificial Intelligence*, strani 699–704.

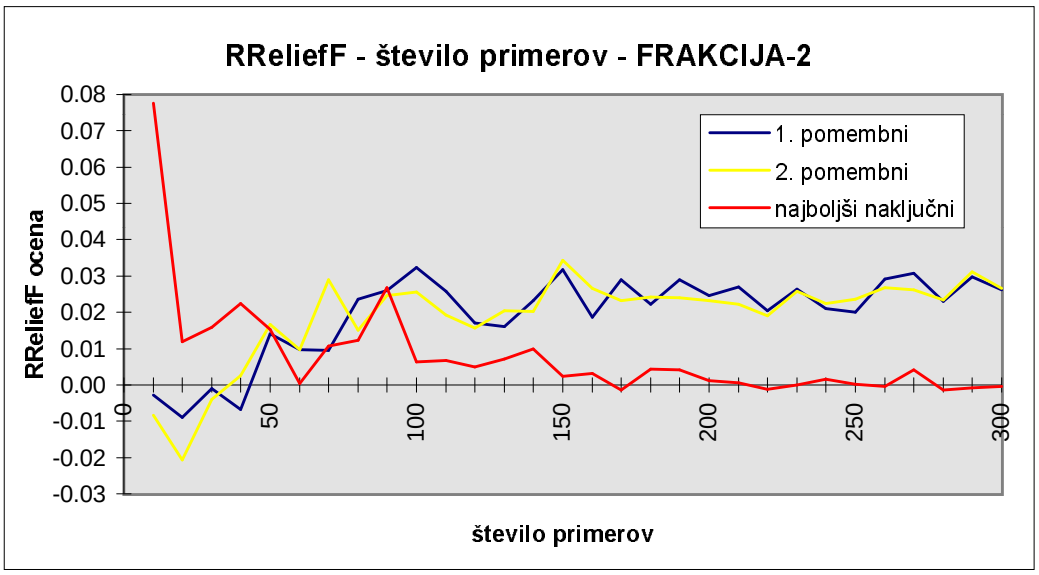
# A

# Ocene glede na število učnih primerov

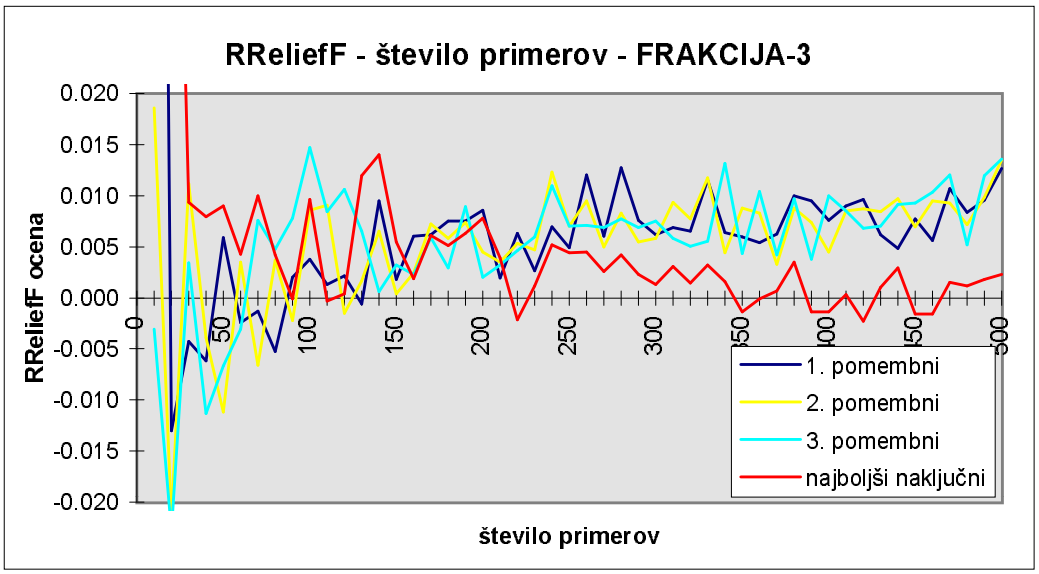
*Izbira! To je preblisk razuma. Oklevate? Vse je povedano, zmotili ste se.*
*Honoré de Balzac*

Grafi predstavljajo odvisnosti ocen kvalitete atributov od števila učnih primerov za RReliefF. Že iz slike 2.4 in tabele 2.1 je razvidno, da je MSE popolnoma neuspešen v problemih FRAKCIJA, MODULO in PARNOST, zato teh grafov ne podajemo. Za COSINUS in LINEAR so ocene, ki jih vrača MSE smiselne in smo jih vključili.

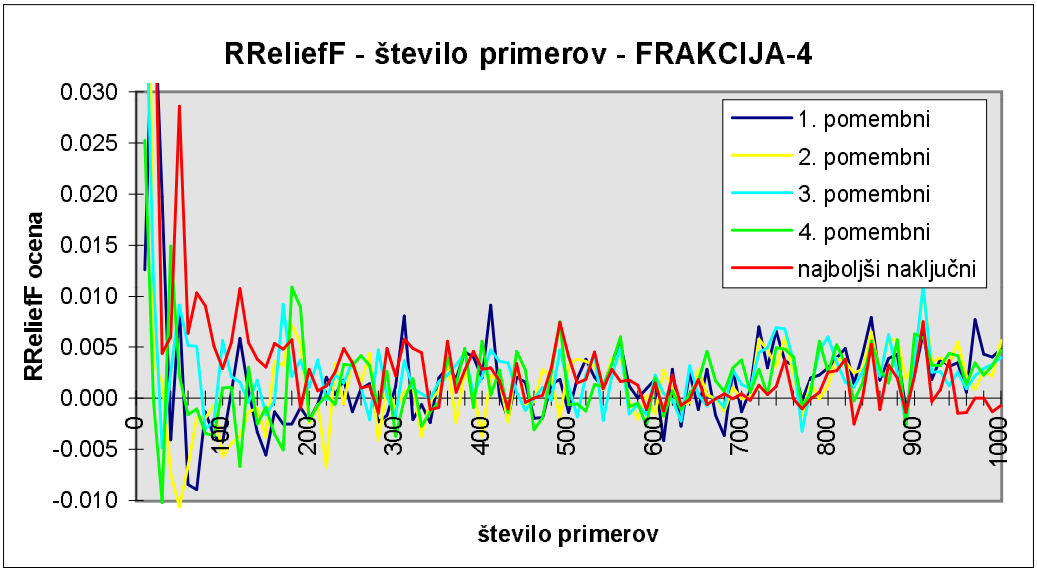
Pri problemu MODULO-8 z dvema pomembnima atributoma podajamo tudi graf ocen za varianto RReliefFa, ki uporablja pragovno funkcijo za uravnoteženje ocen diskretnih in zveznih atributov. Pri ostalih problemih rezultatov s pragovno funkcijo ne nevajamo, saj so zelo podobni tistim brez pragovne funkcije (razmerje med atributi ostaja enako, spremenijo se le številčne vrednosti).



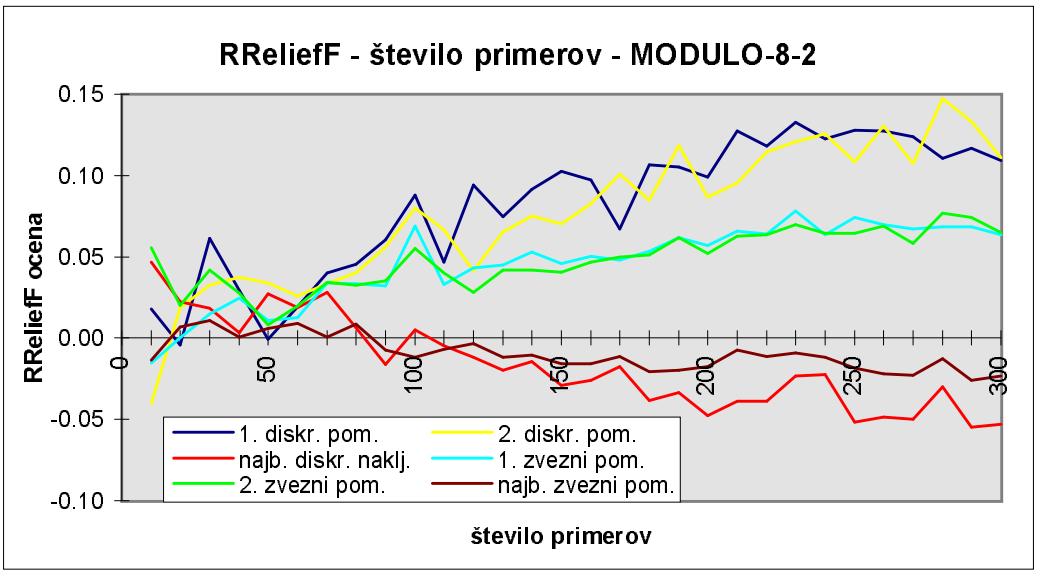

Slika A.1: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: FRAKCIJA z dvema pomembnima atributoma.



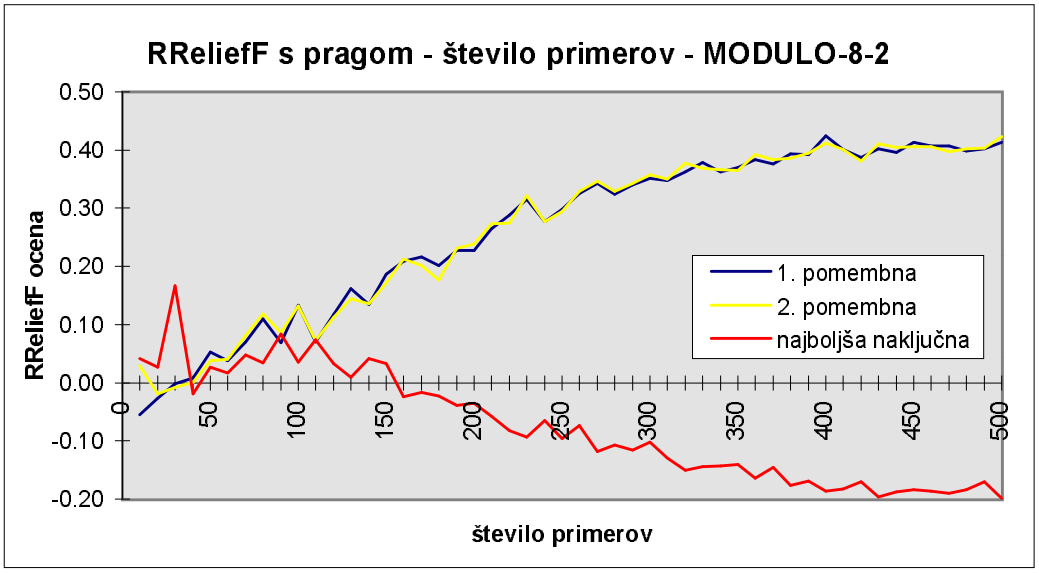

Slika A.2: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: FRAKCIJA s tremi pomembnimi atributi.



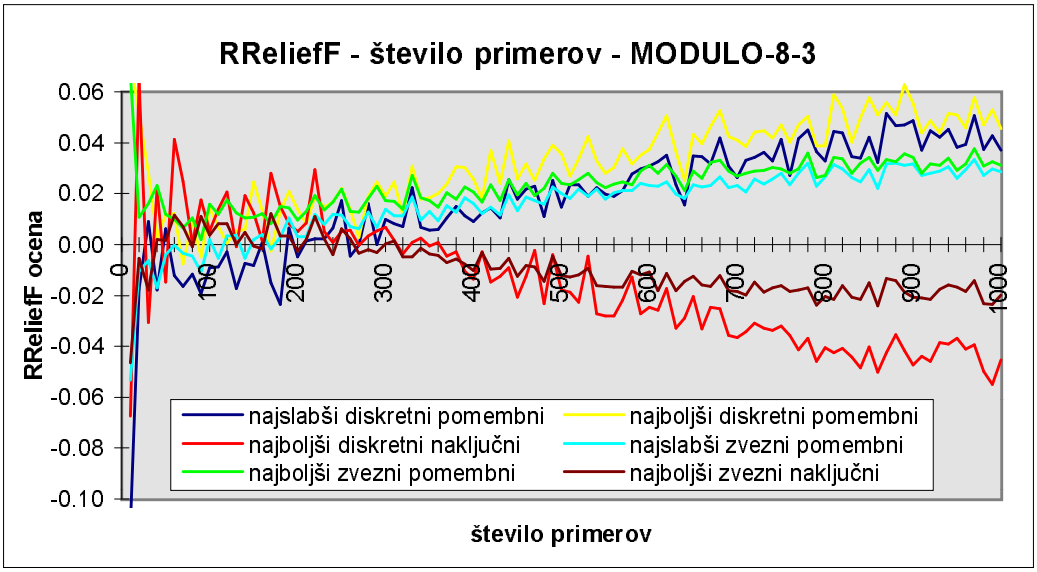

Slika A.3: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: FRAKCIJA s štirimi pomembnimi atributi.



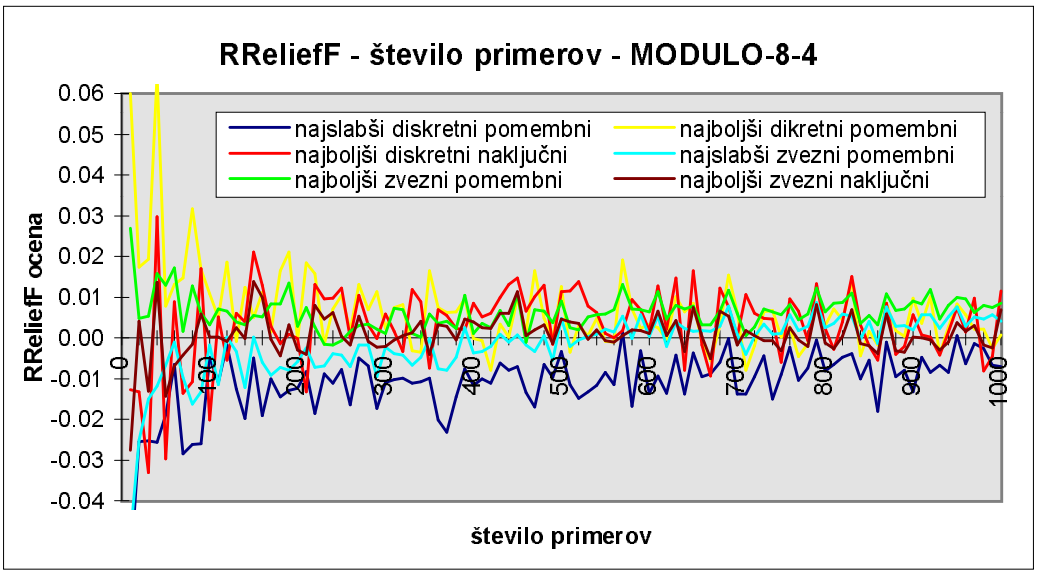

Slika A.4: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: MODULO-8 z dvema pomembnima atributoma.



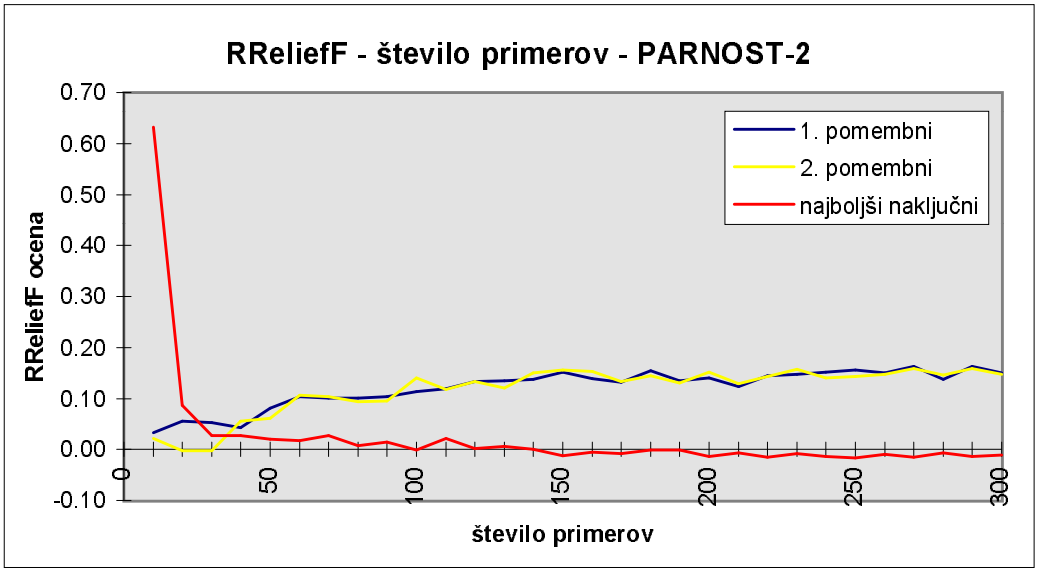

Slika A.5: Ocene kvalitete atributov pri spreminjanju števila učnih primerov z uporabo pragovne funkcije. Problem: MODULO-8 z uporabo pragovne funkcije z dvema pomembnima atributoma. Pri avtomatskih nastavitvah pragov postanejo ocene paroma enakih diskretnih in zveznih atributov popolnoma enake. Zanimive so številčne vrednosti ocen, če jih primerjamo s prejšnjo sliko A.4.



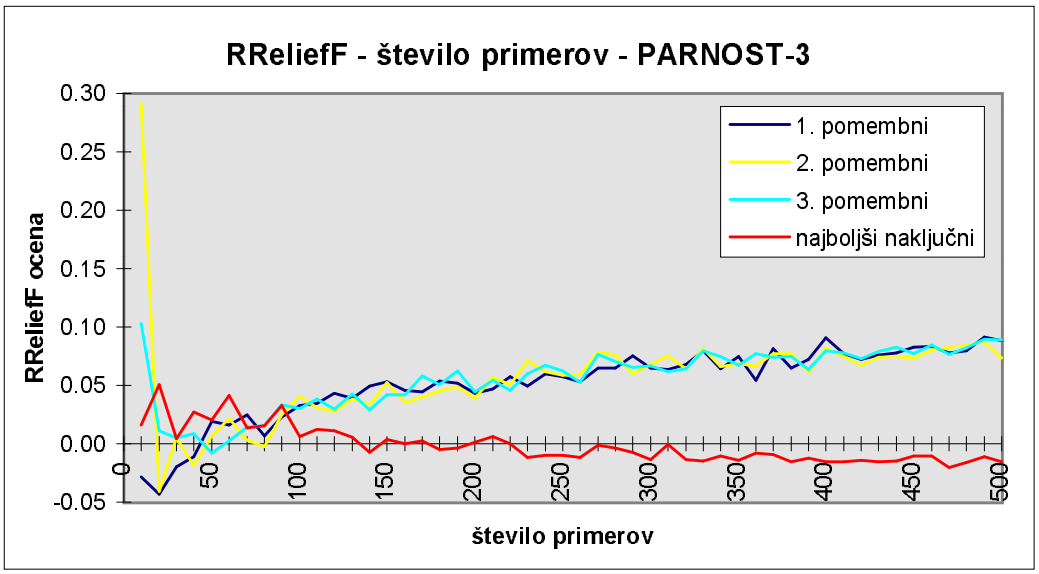

Slika A.6: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: MODULO-8 s tremi pomembnimi atributi.



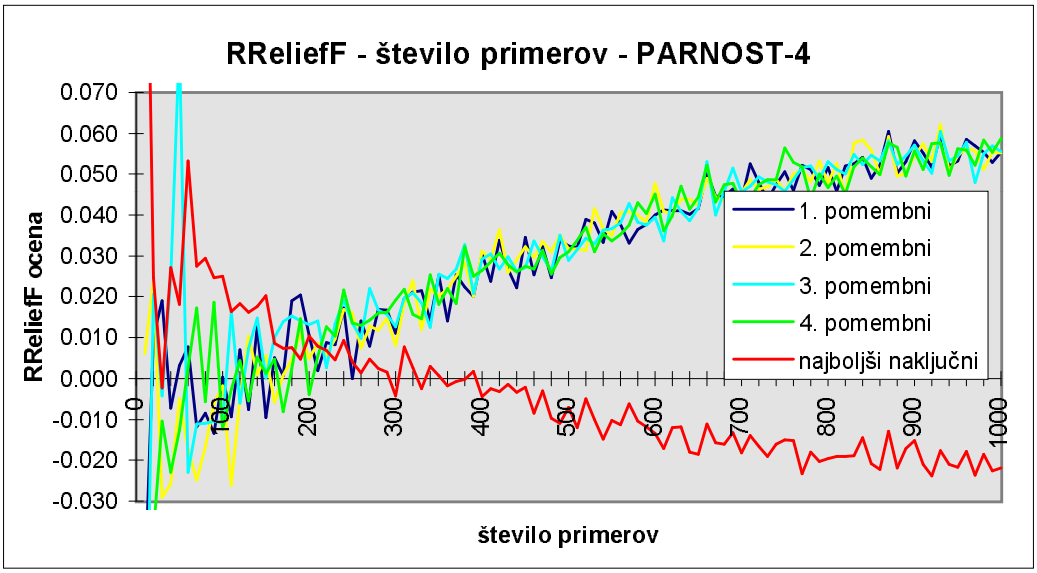

Slika A.7: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: MODULO-8 s štirimi pomembnimi atributi.



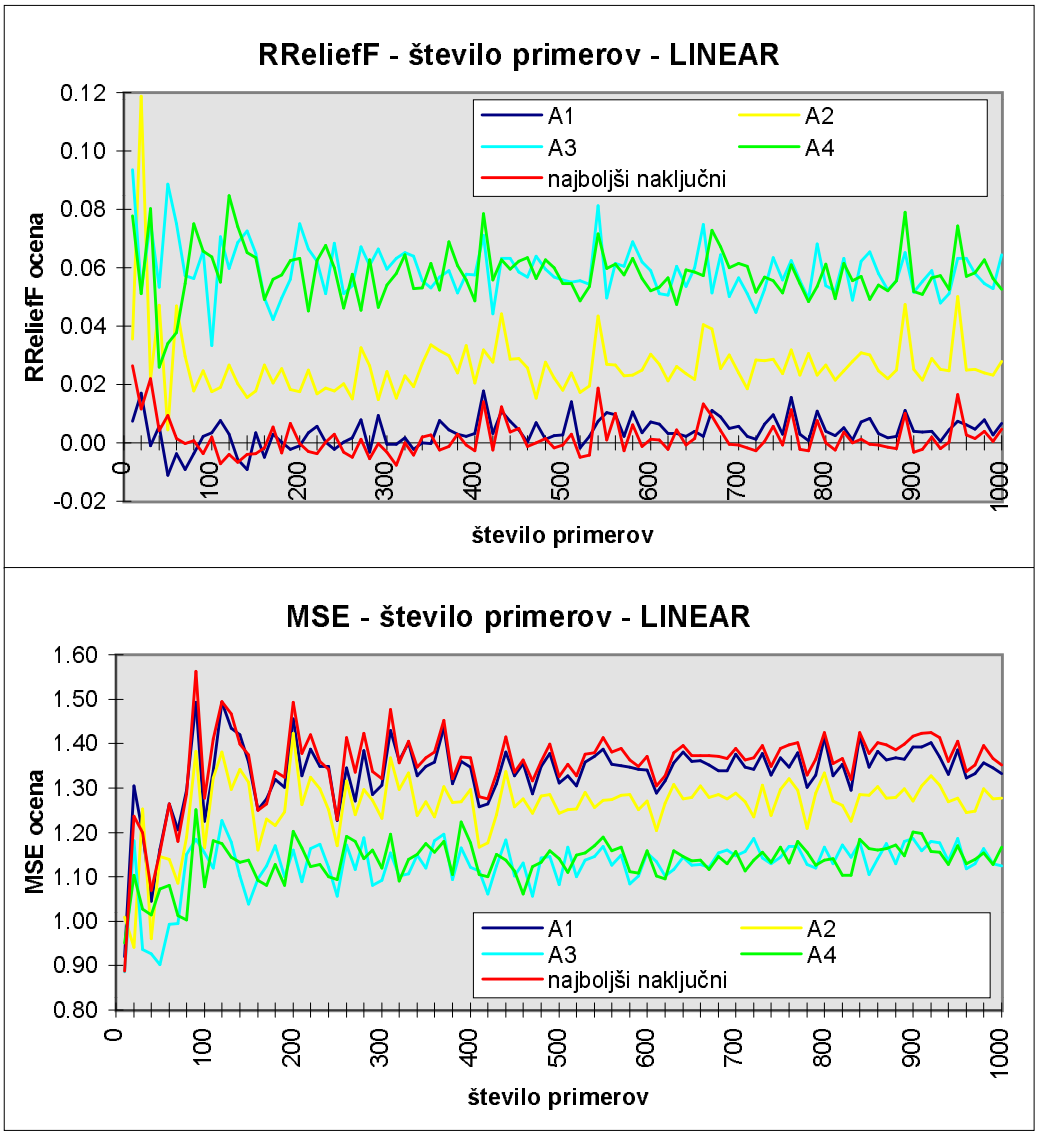

Slika A.8: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: PARNOST z dvema pomembnima atributoma.



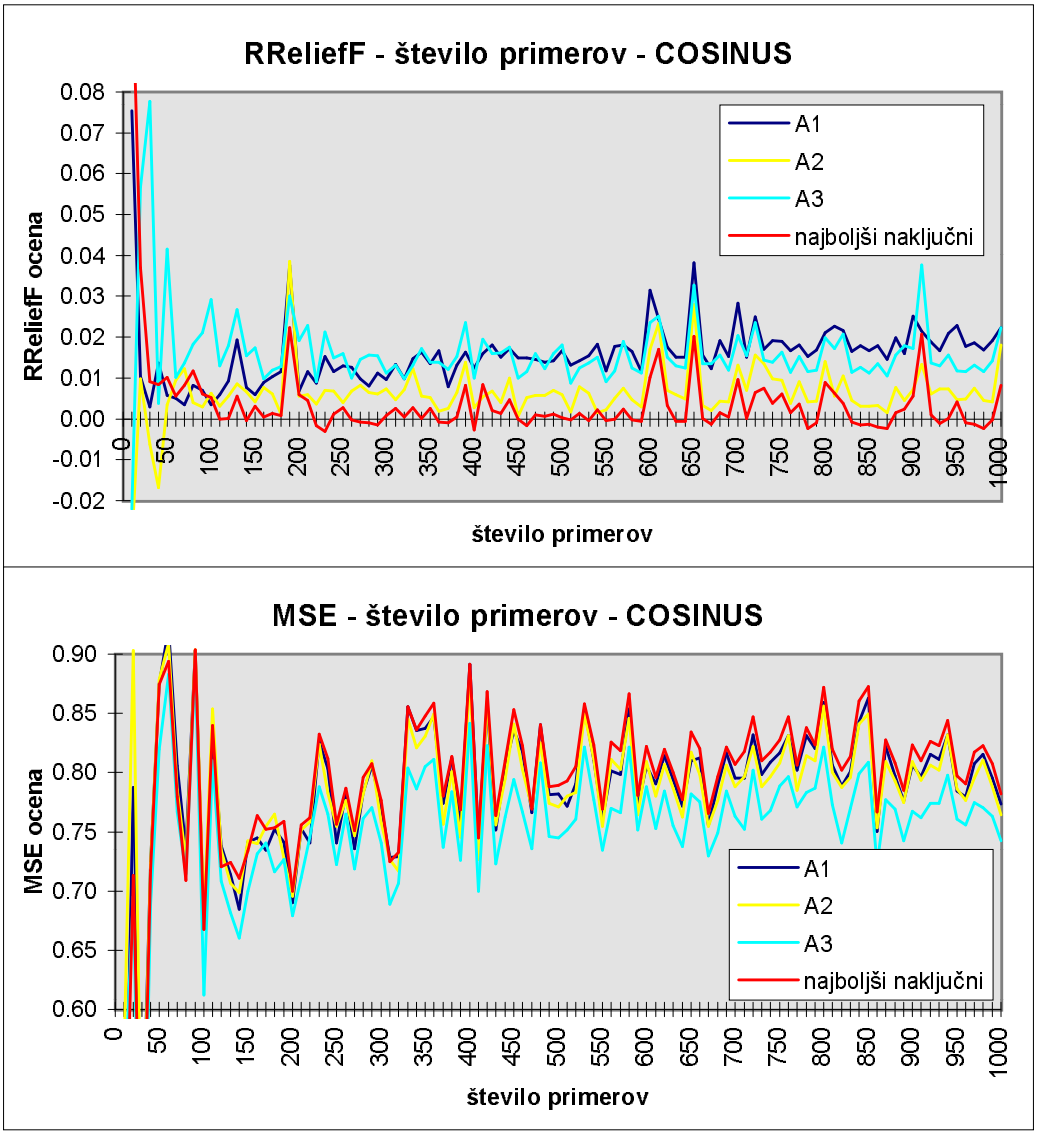

Slika A.9: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: PARNOST s tremi pomembnimi atributi.




Slika A.10: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem: PARNOST s štirimi pomembnimi atributi.




Slika A.11: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem LINEAR: $C = A_1 - 2A_2 + 3A_3 - 3A_4$



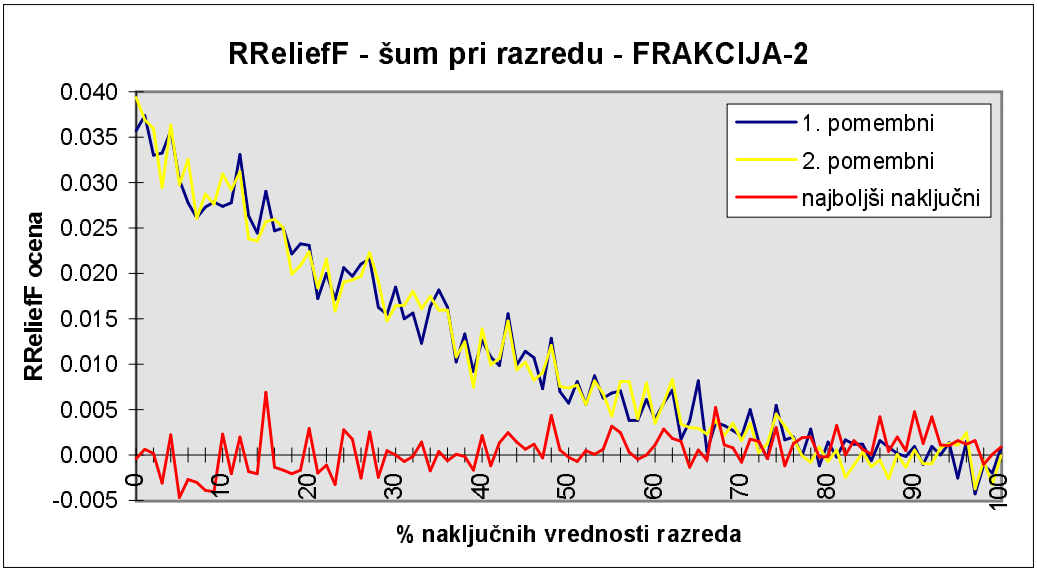

Slika A.12: Ocene kvalitete atributov pri spreminjanju števila učnih primerov. Problem COSINUS: $C = (-2A_2 + 3A_3)\cos(4\pi A_1)$

# B

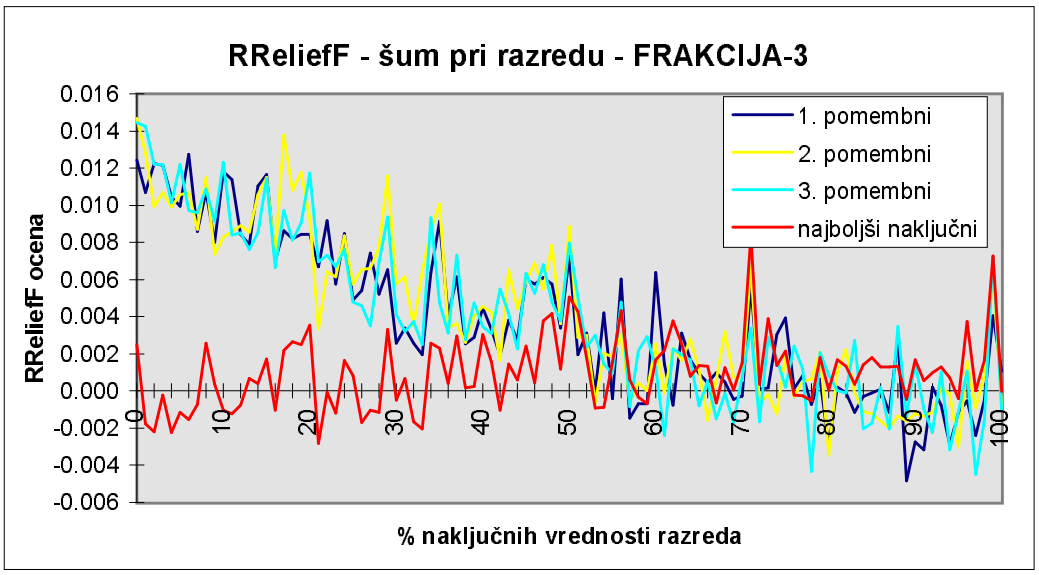
# Ocene pri napačnem razredu

*Če zaprete vrata vsem napakam, bo tudi resnica ostala zunaj.*
*Rabindranath Tagore*

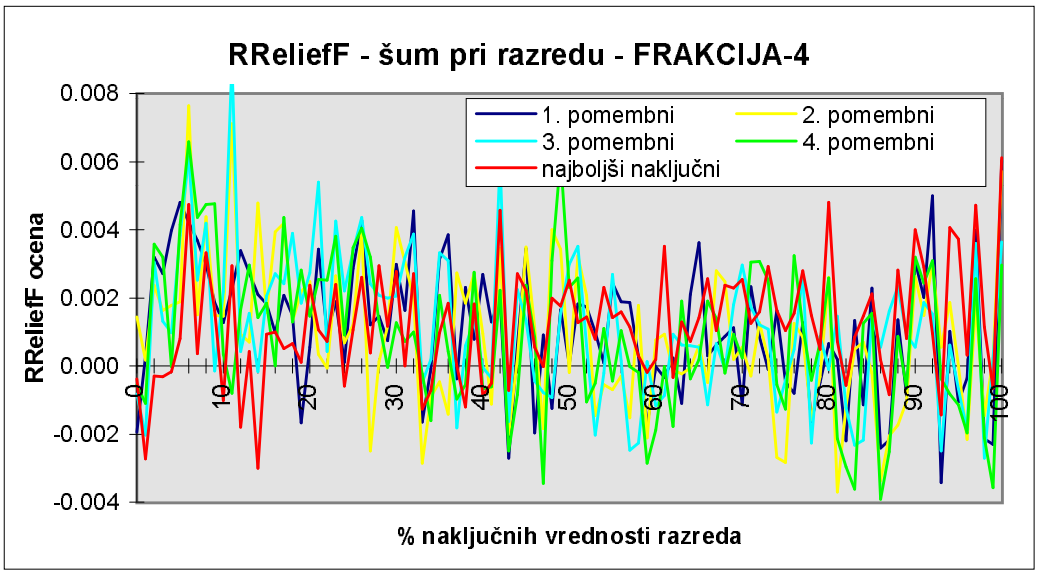
Grafi predstavljajo odvisnosti ocen kvalitete atributov od deleža naključnih vrednosti razreda. Za probleme FRAKCIJA, MODULO in PARNOST podajamo le ocene, ki jih izračuna RReliefF, saj je MSE izgubljen tudi brez šuma. Pri LINEAR in COSINUS vrača MSE smiselne ocene in jih predstavljamo.



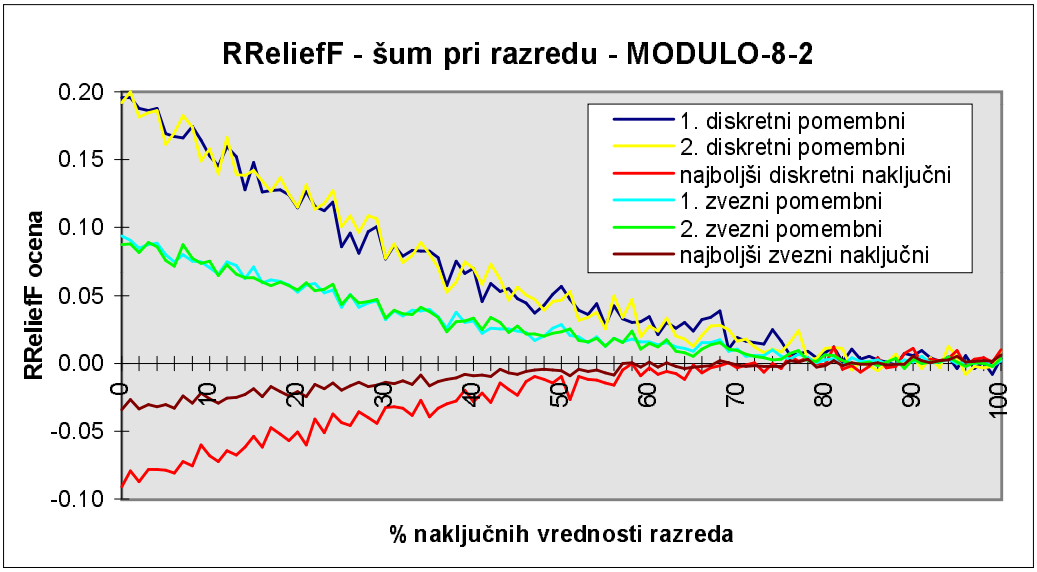

Slika B.1: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem: FRAKCIJA z dvema pomembnima atributoma.



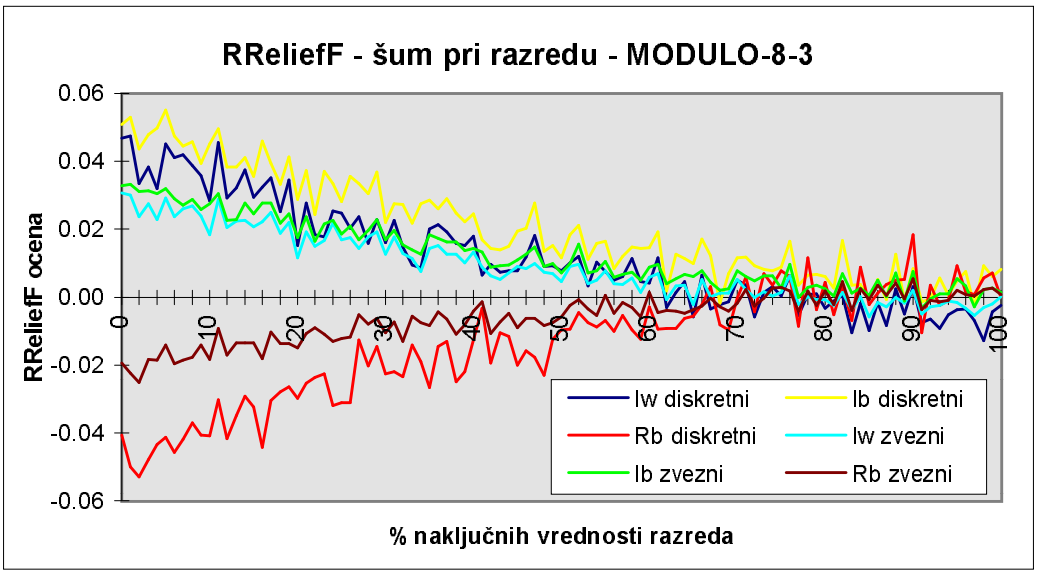

Slika B.2: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem: FRAKCIJA s tremi pomembnimi atributi.



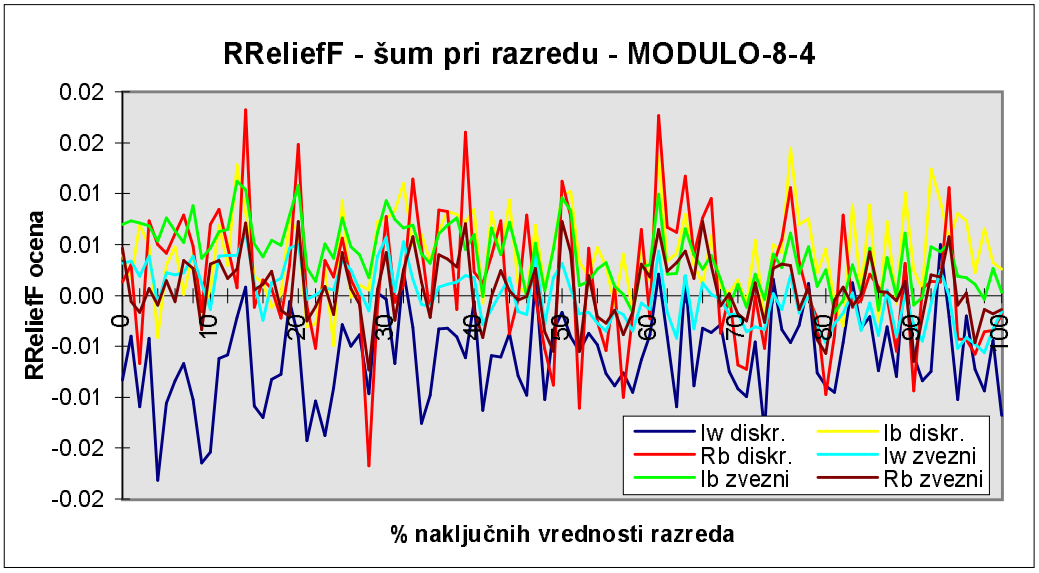

Slika B.3: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem: FRAKCIJA s štirimi pomembnimi atributi.



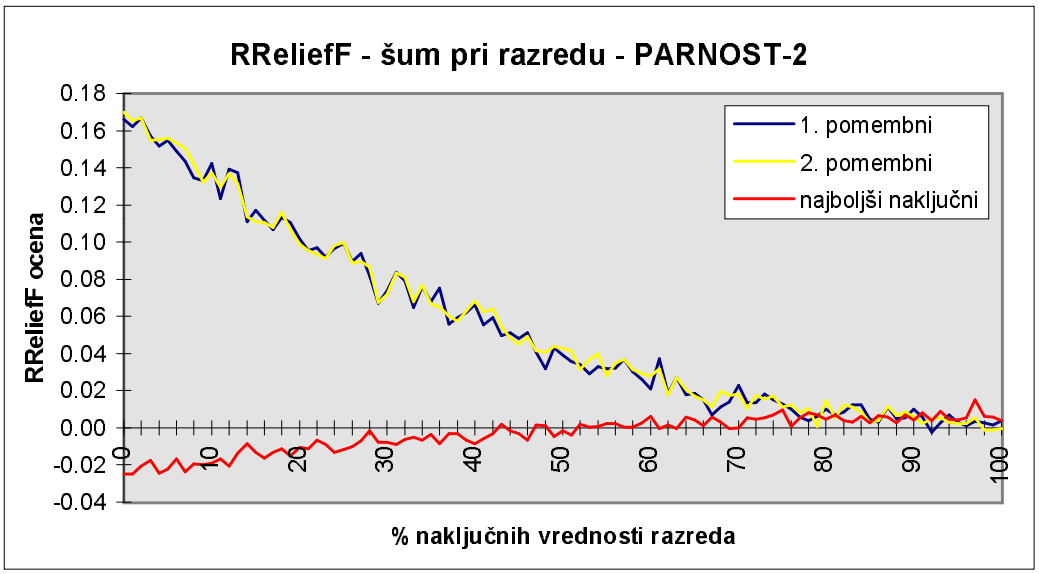

Slika B.4: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem: MODULO-8 z dvema pomembnima atributoma.



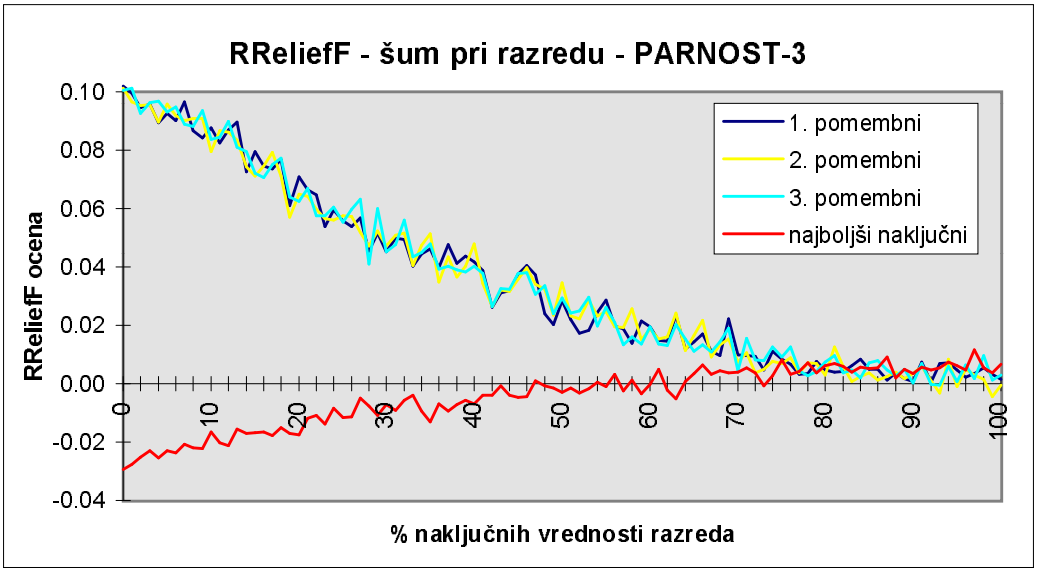

Slika B.5: Ocene kvalitete pri MODULO-8 s tremi pomembnimi atributi. Oznake pomenijo: I pomembni, R naključni, w najslabši in b najboljši atribut.



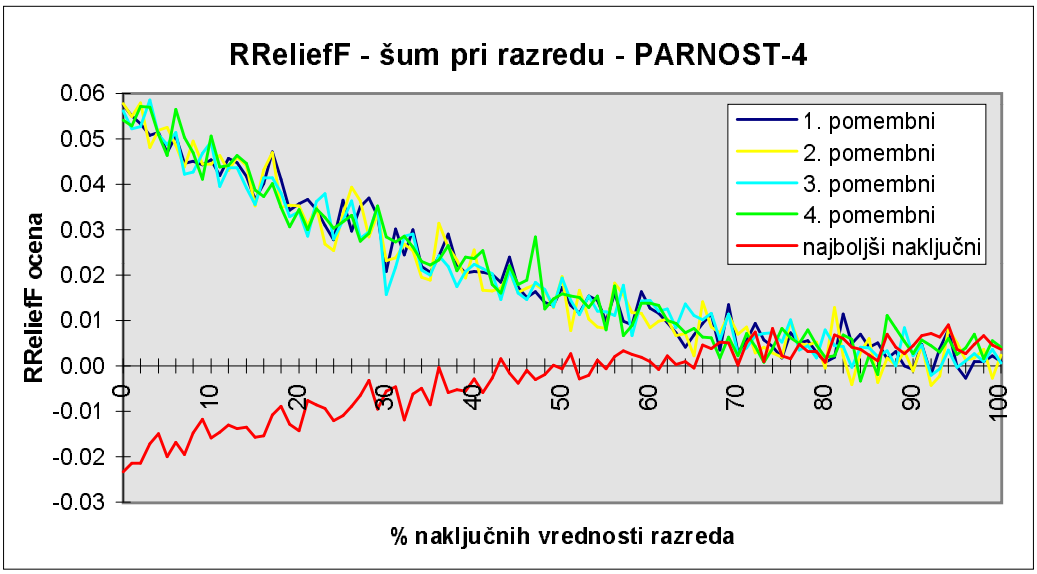

Slika B.6: Ocene kvalitete za MODULO-8 s štirimi pomembnimi atributi. Oznake pomenijo: I pomembni, R naključni, w najslabši in b najboljši atribut.



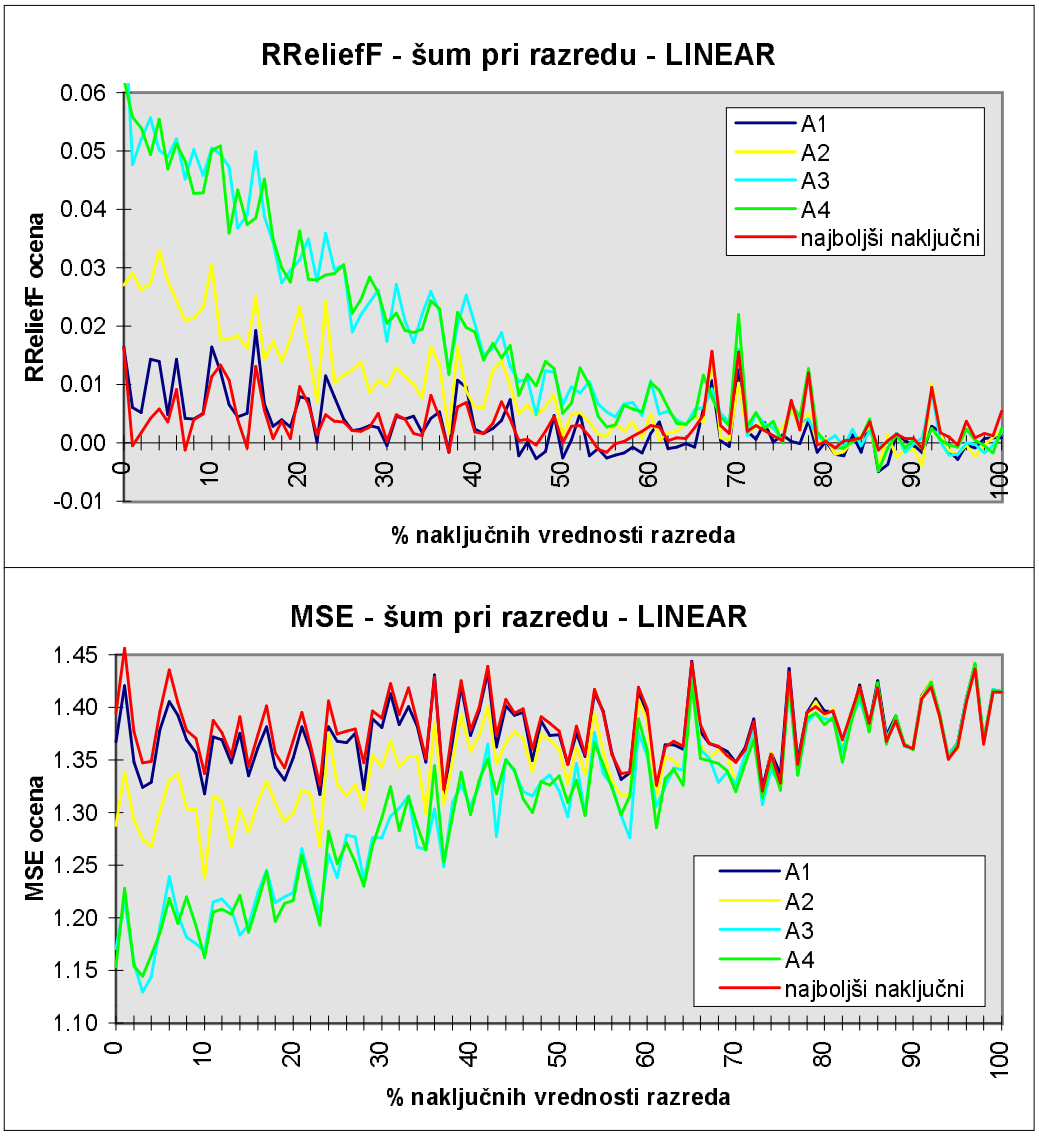

Slika B.7: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem: PARNOST z dvema pomembnima atributoma.



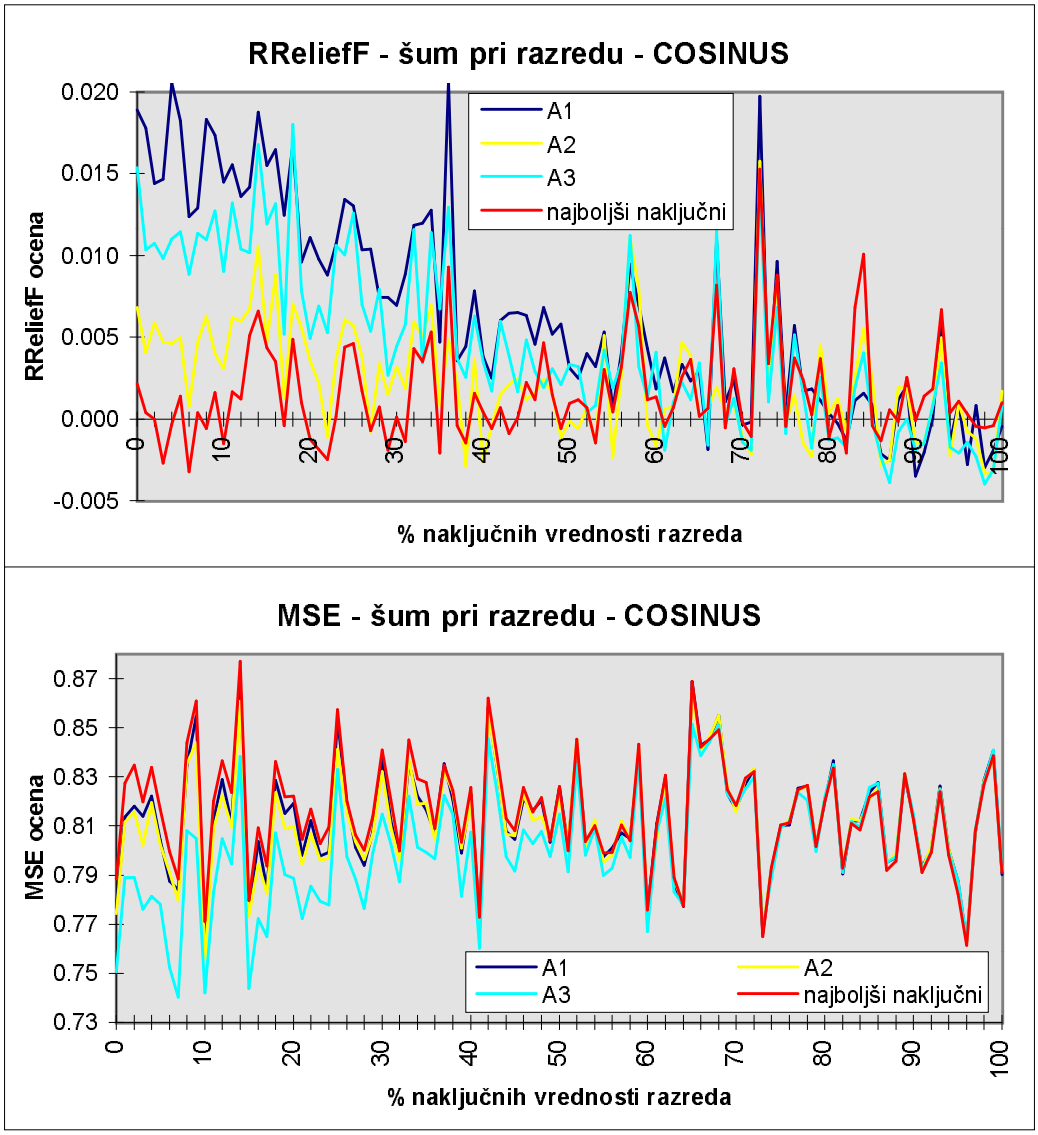

Slika B.8: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem: PARNOST s tremi pomembnimi atributi.




Slika B.9: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem: PARNOST s štirimi pomembnimi atributi.




Slika B.10: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem LINEAR: $C = A_1 - 2A_2 + 3A_3 - 3A_4$



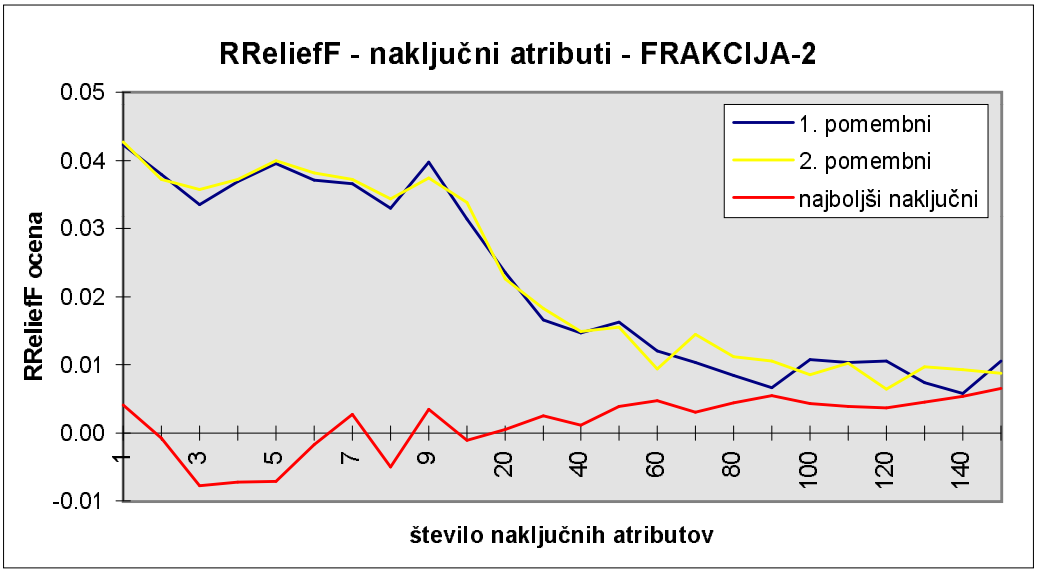

Slika B.11: Ocene kvalitete atributov pri napačnih vrednostih razreda. Problem COSINUS: $C = (-2A_2 + 3A_3)\cos(4\pi A_1)$

# C

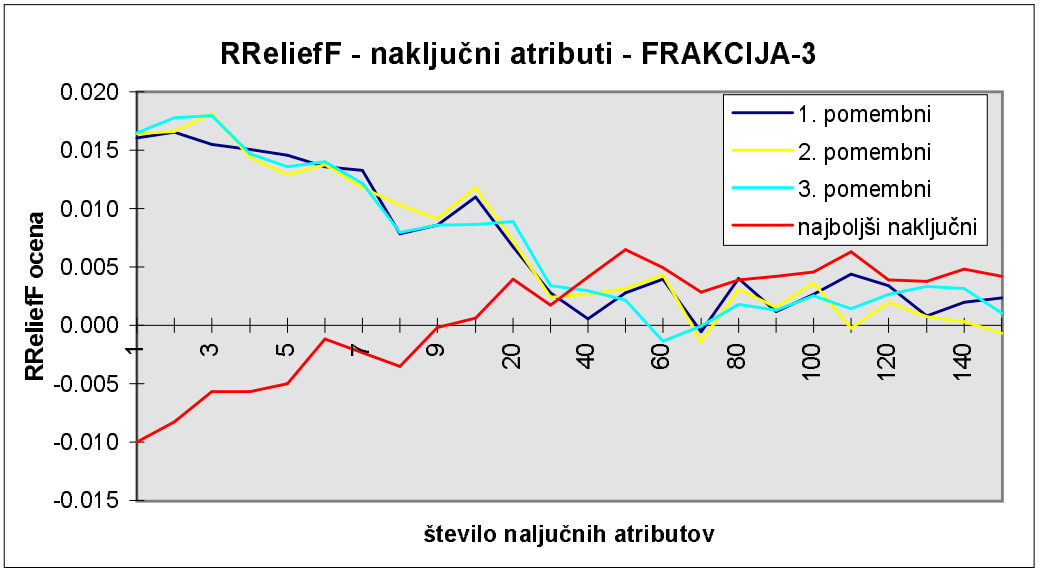
# Ocene in naključni atributi

*Bog ne kocka.*
*Albert Einstein*

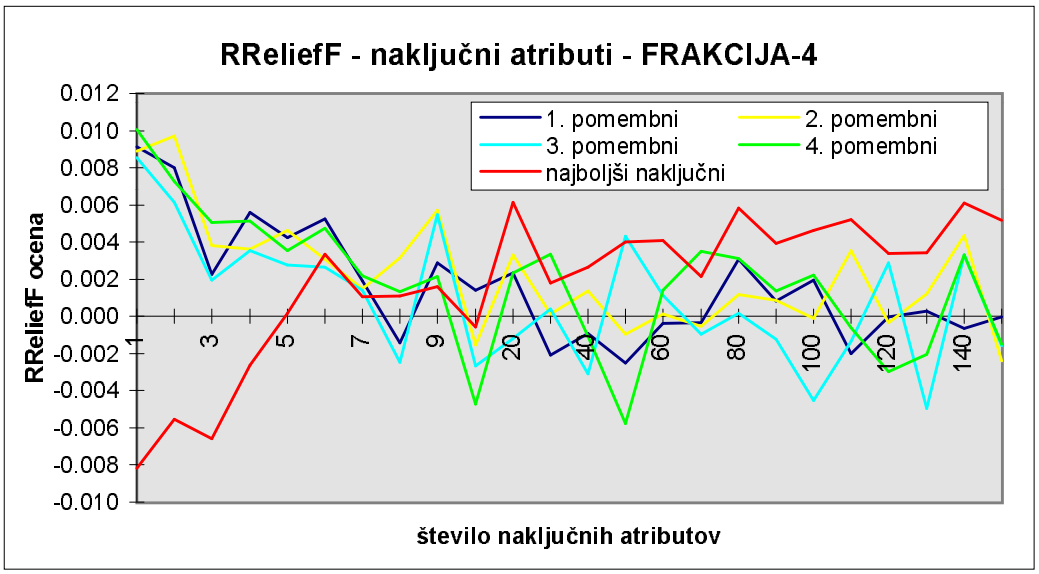
Grafi predstavljajo odvisnosti ocen kvalitete atributov, ki jih izračuna RReliefF, od števila naključnih atributov v opisu problema. MSE ocenjuje vsak atribut neodvisno od ostalih, zato število naključnih atributov ne vpliva na njegovo oceno in grafov ne podajamo.



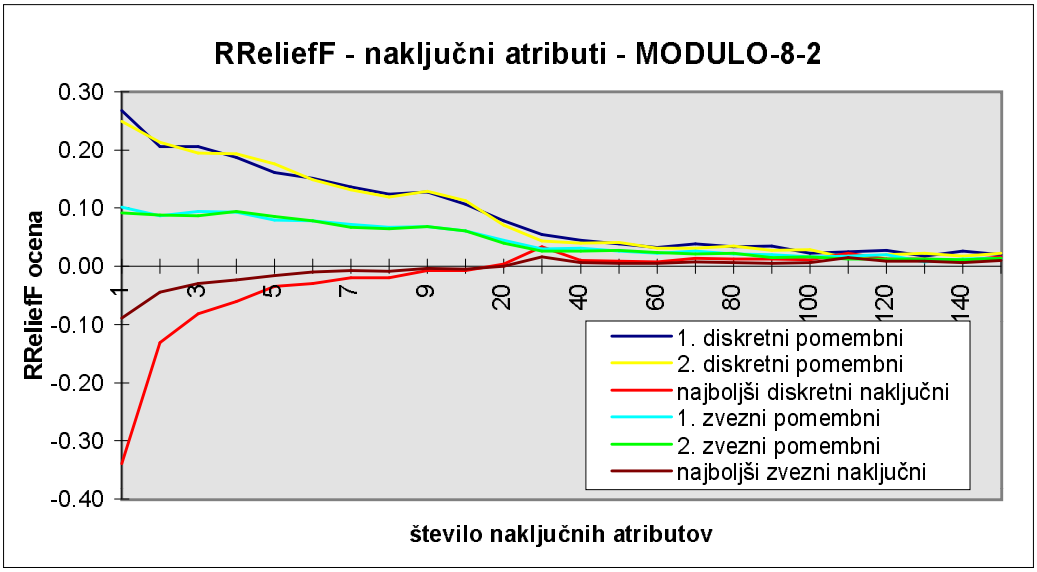

Slika C.1: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem: FRAKCIJA z dvema pomembnima atributoma.



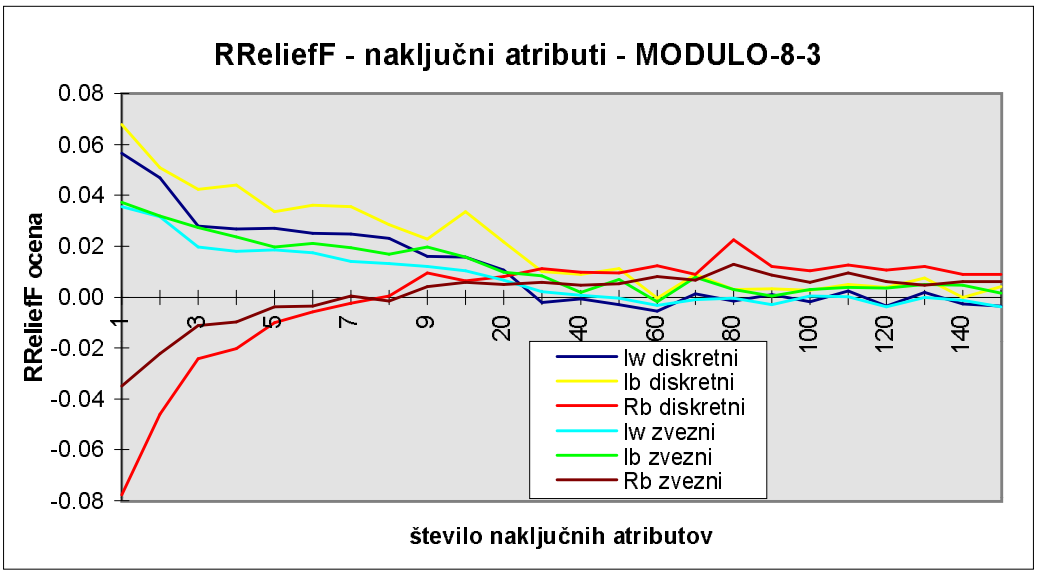

Slika C.2: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem: FRAKCIJA s tremi pomembnimi atributi.



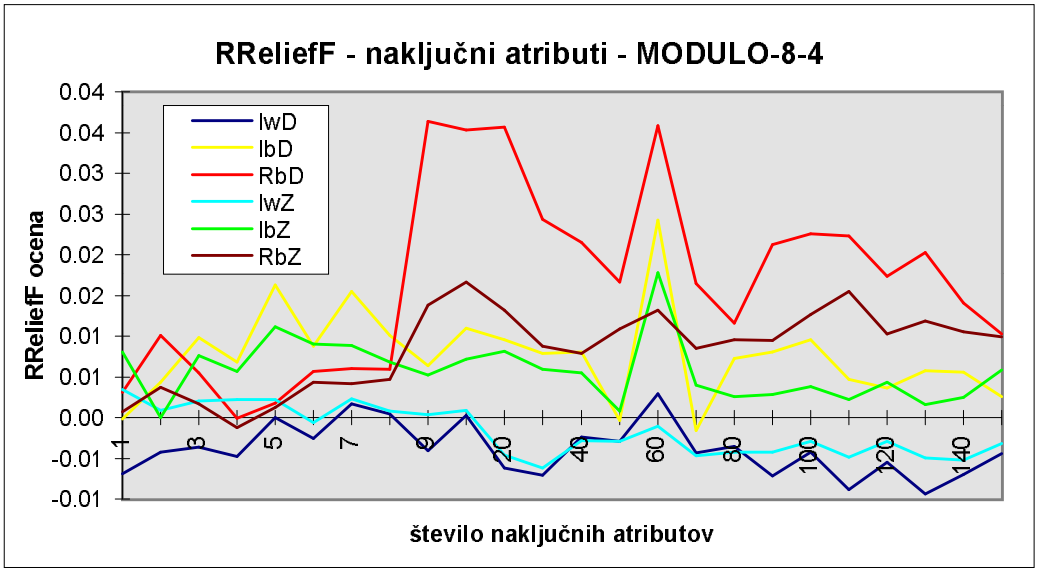

Slika C.3: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem: FRAKCIJA s štirimi pomembnimi atributi.



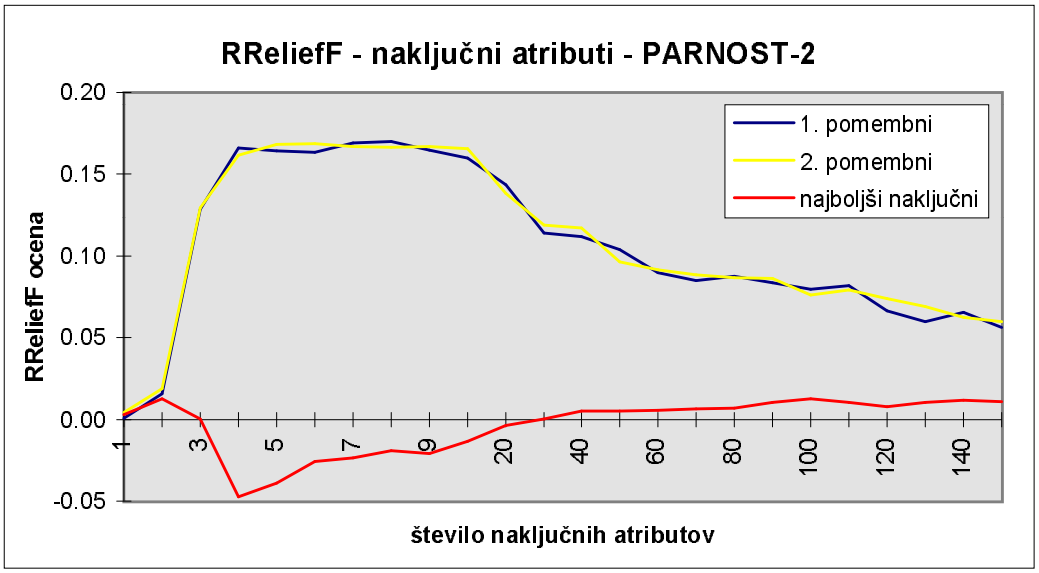

Slika C.4: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem: MODULO-8 z dvema pomembnima atributoma.



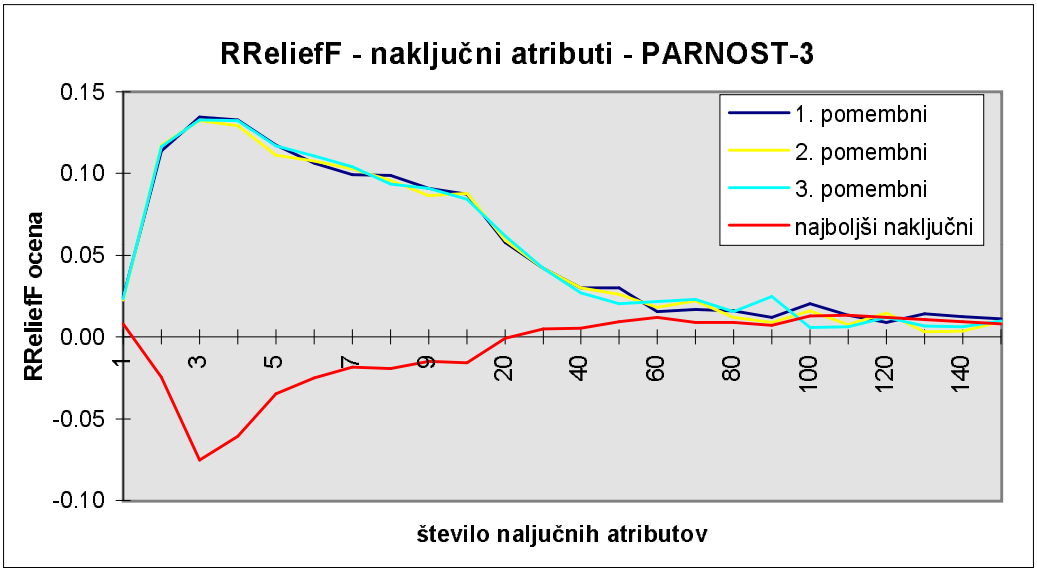

Slika C.5: Ocene kvalitete pri MODULO-8 s tremi pomembnimi atributi. Oznake pomenijo: I pomembni, R naključni, w najslabši in b najboljši atribut.



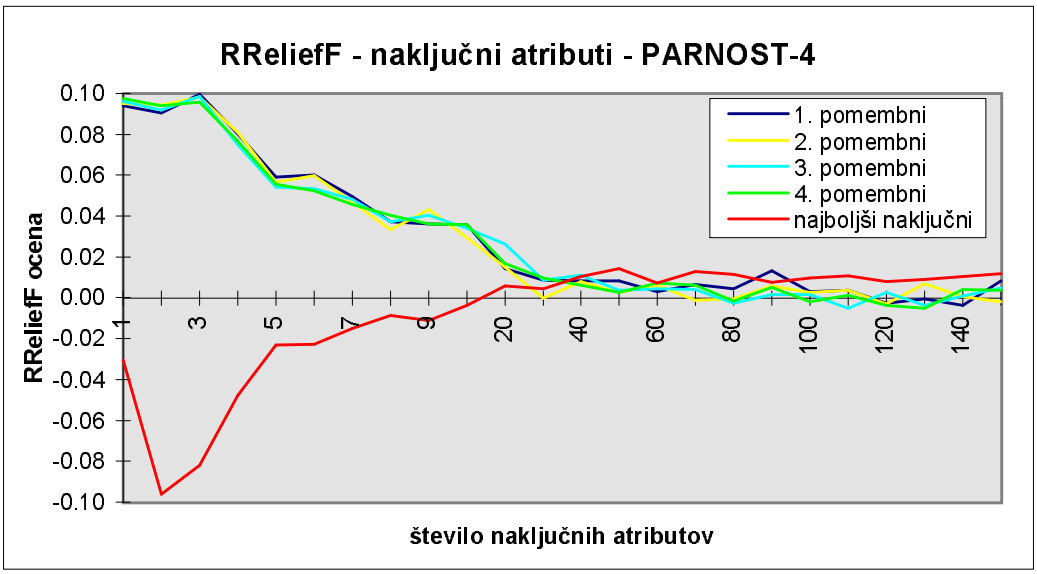

Slika C.6: Ocene kvalitete pri MODULO-8 s štirimi pomembnimi atributi. Oznake pomenijo: I pomembni, R naključni, w najslabši, b najboljši, D diskretni in Z zvezni atribut.



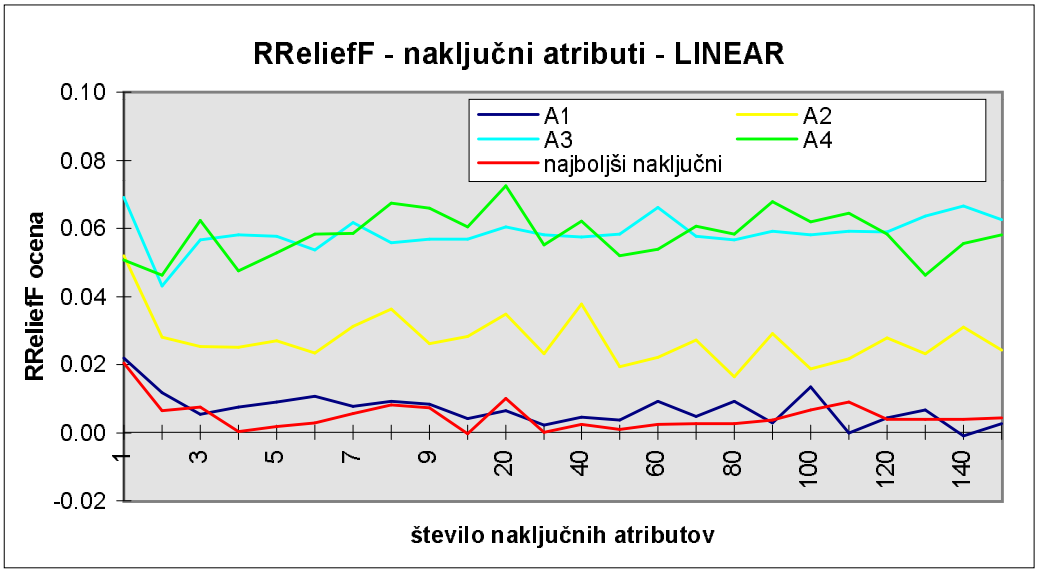

Slika C.7: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem: PARNOST z dvema pomembnima atributoma.



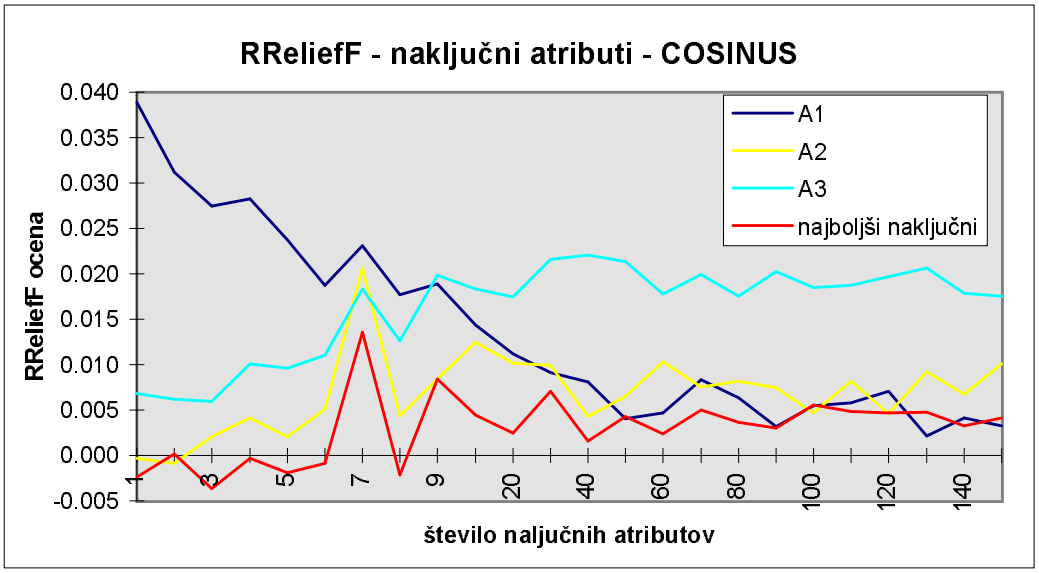

Slika C.8: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem: PARNOST s tremi pomembnimi atributi.




Slika C.9: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem: PARNOST s štirimi pomembnimi atributi.




Slika C.10: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem LINEAR: $C = A_1 - 2A_2 + 3A_3 - 3A_4$




Slika C.11: Ocene kvalitete atributov v odvisnosti od števila naključnih atributov. Problem COSINUS: $C = (-2A_2 + 3A_3)\cos(4\pi A_1)$

*Vse je bilo že povedano. Toda, ker nihče ne posluša, je potrebno vedno znova začenjati.*

*André Gide*