2.zadaća – inteligentni sustavi 2

Radila: Nola Čumlievski

1. Izbacivanje varijabli i diskretizacija

```
library(tree)
library(ISLR)
library(data.table)
#izostavljanje varijable stog i id
setDT(edukacija88)[, stog :=NULL]
setDT(edukacija88)[, id :=NULL]
edukacija88
```

Slika 1: Izbacivanje nerelevantnih varijabli

Nakon importa dataseta *„edukacija88”,* na slici 1. vidljiv je kod za izbacivanje varijable *„stog” i „id”* iz navedenog skupa. Varijabla *„stog”* izbačena je iz skupa jer je bilo potrebno izbaciti varijablu zadanu iz prošle zadaće, a varijablu *„id”* jer je u daljnjem postupku izrade stabla, Rstudio koristio varijablu *„id”* za predviđanje vrijednosti varijable *„grade”* koju ne bi trebalo uzeti u obzir u ovoj predikciji. Za izbacivanje varijabli korištena je funkcija *„setDT”* iz biblioteke *„ data.table”.*

```
#diskretizacija dataset-a
binarni=edukacija88[,6:13]
nebinarni=edukacija88[,1:5]
eduDisk=discretizeDF(nebinarni,default = list(method="interval",breaks=3,labels=c("bad","good","excellent")))
edudis=data.frame(eduDisk, binarni)
head(edudis)
```

Slika 2: Diskretizacija nebinarnih varijabli

Kod za diskretizaciju vidljiv je na slici 2. Prije diskretizacije skupa, bilo je potrebno odvojiti binarne i nebinarne varijable u posebne skupove, s obzirom da se binarne varijable ne smiju mijenjati. Za diskretizaciju skupa *„nebinarni”* (skupa nebinarnih varijbli) korištena je funkcija *„discretizeDF”* iz biblioteke *„arules”.* Navedena funkcija za parametre uzima dataset nad kojim izvršavamo diskretizaciju i listu u kojoj navodimo koju metodu želimo koristiti za diskretizaciju, kao i da li želimo da se prikažu integer vrijednosti varijabli ili određeni vektor naziva (label). U ovom primjeru, za prikaz sam izabrala prikaz naziva *„bad”, „good” te „excellent”.* Parametar *„breaks”* određuje graničnu vrijednost za diskretiziranje varijabli. Možemo navesti broj koji označava broj kategoričkih varijabli (u ovom slučaju imamo dvije kategoričke varijable – FAIL i

PASS), a može se navesti bilo koji broj koji određuje u koliko kategorija želimo podijeliti vrijednosti varijabli (u primjeru, varijable su podijeljene u kategorije „bad”, „good” i „excellent” što znači da parametar breaks mora iznositi 3). Nakon diskretizacije nebinarnih varijabli, pomoću funkcije *„data.frame”,* diskretiziran skup *„eduDisk”* i skup *„binarni”* objedinjuju se u jedan data frame *„edudis”*. Prikaz prvih 6 redova skupa možemo vidjeti na slici 3.

```
> head(edudis)
   lectures   quizzes      labs    videos selfassesm forum red kruzna stabla1 dinamicko stabla2 demons grade
1       bad      good      good       bad        bad     1   1      1       1         1       1      1  PASS
2      good      good      good       bad        bad     0   0      0       0         0       0      0  PASS
3      good      good       bad       bad        bad     1   0      1       1         0       0      1  FAIL
4      good excellent      good       bad       good     1   1      1       1         1       1      1  PASS
5       bad excellent       bad excellent       good     0   0      1       0         0       0      1  FAIL
6 excellent excellent excellent      good        bad     0   0      0       0         0       0      0  PASS
```

Slika 3: Prvih 6 redova skupa "edudis"

Kao što je vidljivo na slici 3., varijable *„lectures, „quizzes”, „labs” „videos” i „selfassesm”* su diskretizirane, dok su svim binarnim varijablama sačuvane originalne vrijednosti.

2. Izrada inicijalnog stabla odluke

```
attach(edudis)
#inicijalno stablo odluke
tree.edu=tree(grade~.-grade, edudis)
summary(tree.edu)
plot(tree.edu)
text(tree.edu,pretty=0)
tree.edu
```

Slika 4: Inicijalno stablo odluke

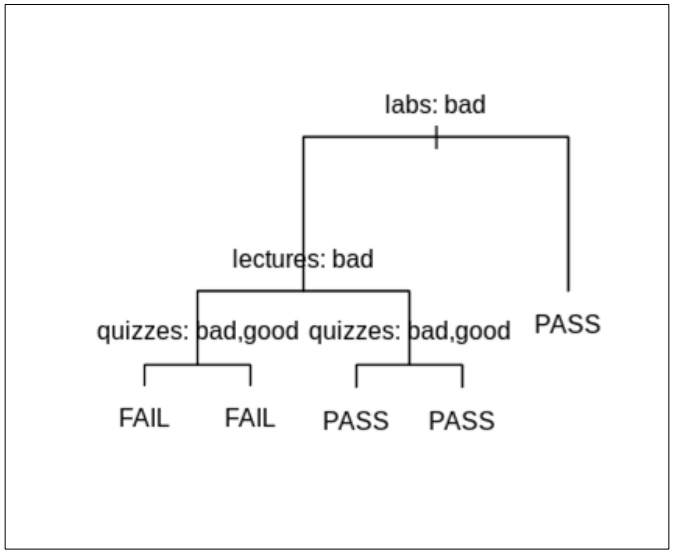
Na slici 4. vidljiv je kod za izradu inicijalnog stabla odluke koje je utemeljeno na cijelom skupu *„edudis”.* Za izradu stabla korištena je funkcija *„tree”* koja se nalazi unutar istoimene biblioteke. Prvi parametar označava koju varijablu predviđamo (u ovom primjeru to je varijabla *„grade”*), te koje varijable želimo koristiti u predikciji (-grade znači da želimo koristiti sve varijable u treniranju osim varijable *„grade”* koju želimo predviditi) i drugi parametar označava skup podataka koji želimo koristiti (*edudis*). Pokretanjem funkcije *„summary”* dobijemo izvještaj prikazan na slici 5. Za prikaz stabla korištena je funkcija *„plot”,* a za prikaz teksta na stablu koristi se funkcija *„text”.* Parametar *„pretty”* unutar funkcije *„text”* označava da se prikazuju imena kategorija za kvalitativne

prediktore, a ne samo slova. Pokrenemo li ime stabla, R izacuje izlaz koji odgovara svim granama stabla.

```
> summary(tree.edu)

Classification tree:
tree(formula = grade ~ . - grade, data = edudis)
Variables actually used in tree construction:
[1] "labs"     "lectures" "quizzes"
Number of terminal nodes:  5
Residual mean deviance:  0.2216 = 15.96 / 72
Misclassification error rate: 0.06494 = 5 / 77
```

Slika 5: Pokretanje funkcije summary

Pokretanjem funkcije „*summary”* dobijemo neke osnovne podatke o stablu odluke. Vidimo da su ispisane varijable koje su korištene unutar funkcije – *„labs”, „quizzes” i „lectures”,* što zapravo predstavlja varijable koje funkcija smatra relevantnim za predviđanje varijable *„grade”*. Devijanca označava mjeru raspršenosti podataka u stablu (što je manja, to je bolja). Također možemo vidjeti stopu pogreške (error rate) koja iznosi 6%.



Slika 6: Inicijalno stablo odluke

Na slici 6. vidljivo je navedeno stablo odluke. Vidimo da najveći utjecaj na ocjenu ima varijabla *„labs”* te stablo od nje započinje grananje. Lijeva strana stabla gleda se za one opservacije za koje

vrijedi da je „*labs”* diskretizacijom klasificiran kao „*bad”*. Vidimo prema stablu, sve opservacije kojima je vrijednost varijable *„labs”* klasificirana kao *„good”* ili *„excellent”* imaju vrijednost varijable *„grade” – „PASS”*. Zatim, ako je vrijednost varijable „*labs”* klasificirana kao „*bad”*, gleda se varijabla „*lectures”* za koju vrijedi: ako je vrijednost također klasificirana kao *„bad”,* studenti su pali kolegij, no ako je klasificirana kao *„good”* ili *„excellent”* student je prošao kolegij.

```
> tree.edu
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 77 97.070 PASS ( 0.32468 0.67532 )
   2) labs: bad 39 50.920 FAIL ( 0.64103 0.35897 )
     4) lectures: bad 24 13.770 FAIL ( 0.91667 0.08333 )
       8) quizzes: bad,good 18  0.000 FAIL ( 1.00000 0.00000 ) *
       9) quizzes: excellent 6  7.638 FAIL ( 0.66667 0.33333 ) *
     5) lectures: good,excellent 15 15.010 PASS ( 0.20000 0.80000 )
      10) quizzes: bad,good 6  8.318 PASS ( 0.50000 0.50000 ) *
      11) quizzes: excellent 9  0.000 PASS ( 0.00000 1.00000 ) *
   3) labs: good,excellent 38  0.000 PASS ( 0.00000 1.00000 ) *
```

Slika 7: Stablo - grananje

Utipkavanjem imena stabla u R-u dobijamo prikaz kriterija cijepanja (npr- labs različito ili jednako „*bad”*), broj opservacija u toj grani, devijanca (mjera raspršenosti podataka u skupu), većinsko predviđanje unutar grane („*FAIL”* ili *„PASS”*) i postotak opservacija koje poprimaju vrijednosti varijable *„grade”*, gdje prvi postotak označava vrijednosti klase „*FAIL”,* a drugi označava postotak vrijednosti klase *„PASS”* (npr. u zadnjem redu vidimo da 100% opservacija pripada klasi „*PASS”*).

3. Izrada stabla odluke koje se temelji na određenom skupu za treniranje te procjena greške testiranja

```
#procjena greške testiranja
set.seed(4)
ucenje=sample(1:nrow(edudis), 43)
test=edudis[-ucenje,]
test_grade=grade[-ucenje]
tree.edu=tree(grade~.-grade,edudis,subset=ucenje)
tree.pred=predict(tree.edu,test,type="class")
table(tree.pred,test_grade)
(10+22)/34
```

Slika 8: Drugo stablo odluke i procjena greške testiranja

Kako bi dobili uvijek isti rezultat prilikom testiranja, koristimo funkciju *„set.seed”*. Zatim kao skup za učenje iniciramo 43 random reda skupa *„edudis”,* što znači da je skup za testiranje suprotnost od skupa za učenje. Također je potrebno inicijalizirati skup koji se sastoji samo od vrijednosti varijable *„grade”* od skupa za testiranje kako bi ga mogli usporediti sa predviđenim vrijednostima (*test_grade*). Za izradu stabla koristimo istu funkciju *„tree”* uz dodatan parametar *„subset”* kojim označavamo da želimo izgradnju stabla samo nad skupom za treniranje/učenje. Za predikciju koristi se funkcija *„predict”* koja kao parametre uzima stablo odluke koje koristimo, skup za koji želimo dobiti predviđene vrijednosti i parametar „*type”* označava koji tip izlaznih podataka želimo (ovdje je odabrana klasa kao output). Na kraju, koristimo funkciju *„table”* kako bi dobili matricu konfuzije s omjerom točno i netočno predviđenih klasa. Pokretanjem funkcije dobije se rezultat prikazan na slici 9.

```
> table(tree.pred,test_grade)
         test_grade
tree.pred FAIL PASS
     FAIL   10    1
     PASS    1   22
```

Slika 9: Točnost predviđanja stabla odluke

Sa slike vidimo da je stablo točno predvidjelo 32 vrijednosti (vrijednosti sa dijagonale) i 2 vrijednosti netočno. Točnost modela možemo provjeriti tako da zbrojimo vrijednosti na dijagonali te

taj broj podijelimo sa ukupnim brojem opservacija u tom skupu – (10+22)/34. Pokretanjem ove linije koda dobijemo da model ima točnost od 0.9411765 što je otprilike 94%.

4. Cross validacija

```
#cross validacija
set.seed(4)
cv.edu=cv.tree(tree.edu, FUN = prune.misclass)
names(cv.edu)
cv.edu
par(mfrow=c(1,2))
plot(cv.edu$size,cv.edu$dev,type="b")
plot(cv.edu$k,cv.edu$dev,type="b")
```

Slika 10: Kod - cross validacija

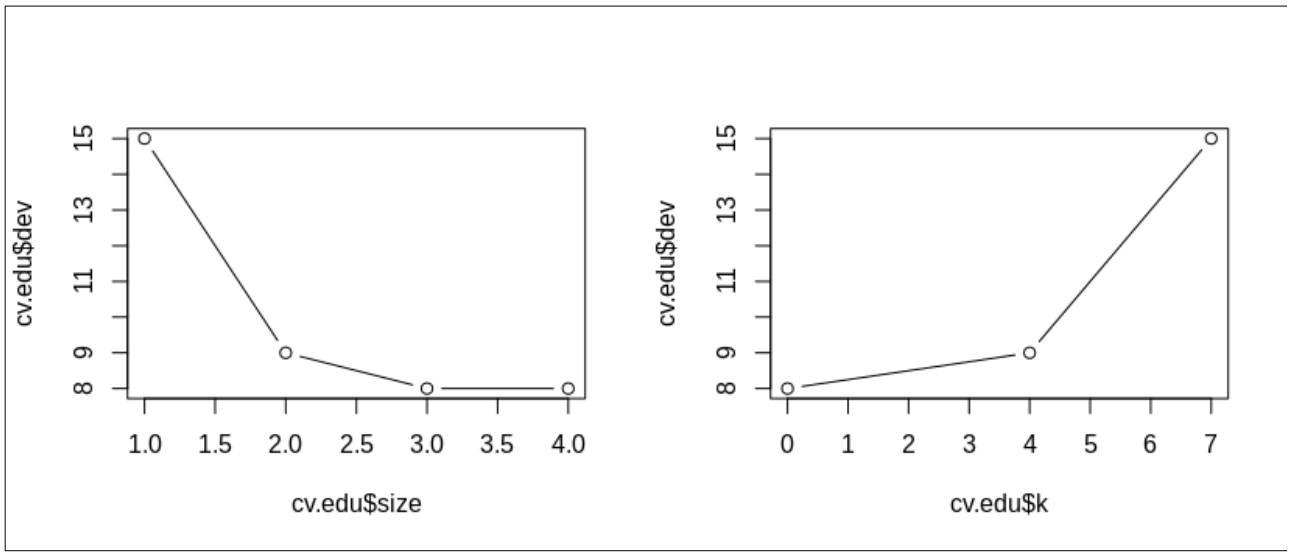
Cross validaciju izvodimo s ciljem određivanja optimalne razine složenosti. Jedna od funkcija koje se mogu koristiti za izračun je funkcija *„cv.tree”,* koja za prvi parametar uzima stalbo odluke koje koristimo, a drugi parametar (FUN=prune.misclass) označava da želimo da se procese cross validacije i orezivanja provodi na temelju stope pogreške klasifikacije. Pokretanjem varijable *„cv.edu”* dobijemo izlaz vidljiv na slici 11.

```
> cv.edu
$size
[1] 4 3 2 1

$dev
[1]  8  8  9 15

$k
[1] -Inf    0    4    7
```

Slika 11: Izlaz cross validacije

Na slici vidimo izlaz cross validacije, „size” označava veličinu čvora, „*dev*” je devijanca (odgovarajuća stopa pogreške), a „*k”* označava vrijednosti parametra cijene složenosti koja se

koristila. Iz prikaza možemo vidjeti da bi optimalno stablo trebalo imati 3 ili 4 čvora, s obzirom da je stopa pogreške kod tih čvorova najmanja (8). Na slici 12. nalazi se grafički prikaz stope pogreške cross validacije.



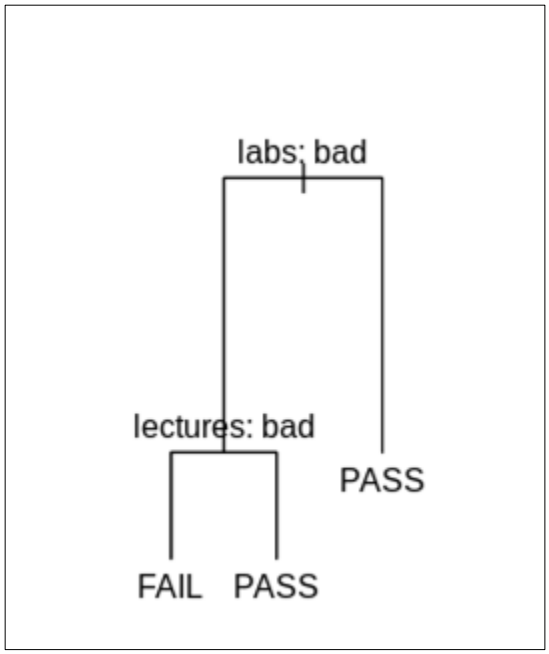
Slika 12: Grafički prikaz stope pokreške

Vidimo da graf prati ispis sa prošle slike. Čvor veličine 1 ima stopu pogreške od 15, veličine 2 stopu pogreške 9, te veličine 3 i 4 stopu pogreške od 8.

```
#orezivanje stabla
prune.edu=prune.misclass(tree.edu, best=3)
plot(prune.edu)
text(prune.edu, pretty = 0)
tree.pred=predict(prune.edu,test,type="class")
table(tree.pred,test_grade)
(10+22)/34
```

Slika 13: Orezivanje stabla na optimalan broj čvorova

Na slici 13. vidimo kod za orezivanje stabla na optimalan broj čvorova. Orezivanje se može izvršiti pmooću funkcije *„prine.misclass"* koja kao argument uzima stablo odluke i broj čvorova na koji želimo orezati stablo (best označava optimalni broj čvorova koji smo dobili pomoću cross validacije). Zatim se koristi funkcija *„plot" i „text"* za izradu grafičkog prikaza stabla.



Slika 14: Orezano stablo

Na slici 14. vidimo prikaz orezanog stabla, koje je u odnosu na početno stablo dosta pojednostavljeno. Koristi samo varijable *„labs" i „lectures"* s obzirom da te dvije varijable imaju najveći utjecaj na ocjenu studenta. U prošlom stablu, imali smo varijablu *„quizzes"* no ona nije imala utjecaj na ocjenu studenta, s obzirom da, ako je varijabla *„lectures"* deklarirana kao *„bad"* student je pao kolegij bez obzira na vrijednost varijable *„quizzes",* te ako je variajbla *„lectures"* deklarirana kao *„good" ili „excellent",* student bi prošao kolegij, bez obzira na varijablu *„quizzes",* što znači da varijabla *„quizzes"* nije relevantna za predviđanje ocjene studenta, što se vidi na orezanom stablu. Pokretanjem funkcije *„table"* dobijemo da je stablo predvidilo klase jednako kao i prvo stablo odluke (isti je postotak točnosti).

```
> table(tree.pred,test_grade)
         test_grade
tree.pred FAIL PASS
     FAIL   10    1
     PASS    1   22
> (10+22)/34
[1] 0.9411765
```

Slika 15: Točnost orezanog stabla

Na slici 15. vidimo da se točnost predviđanja orezanog stabla ne razlikuje od točnosti predviđanja prvog stabla, čemu može biti uzrok manji skup podataka. Međutim, ako postavimo pod parametar *„best"* broj veći od broja optimalnog broja čvorova, možemo vidjeti da smanjena točnost modela.

```
#orezivanje stabla na veću od najbolje vrijednosti
prune.edu2=prune.misclass(tree.edu2,best=5)
tree.pred3=predict(prune.edu2,test,type="class")
table(tree.pred3,test_grade)
(7+23)/34
```

Slika 16: Orezivanje stabla na veći broj čvorova

```
> table(tree.pred3,test_grade)
          test_grade
tree.pred3 FAIL PASS
      FAIL    7    0
      PASS    4   23
> (7+23)/34
[1] 0.8823529
```

Slika 17: Točnost modela orezanog na veći broj čvorova od optimalnog broja

Na slici 17. vidimo da, ako povećamo broj čvorova, postotak točnosti modela se smanjuje (sa 94% točnosti model je spao na 88% točnosti).