Univerza v Ljubljani, Fakulteta za elektrotehniko
Podiplomski magistrski študijski program 2. stopnje Elektrotehnika
Biomedicinska tehnika - Biomedicinska informatika

NAVODILA

Vaja 9: Binarno razvrščanje

Pripravila: Žiga Bizjak & Tomaž Vrtovec

Navodila

Raziskovalci so za namene preverjanja predpostavke, da je ena izmed bolezni srca in ožilja povezana z sistoličnim (višjim) krvnim tlakom, razvili štiri različne diagnostične teste (metode) merjenja krvnega tlaka. Vsi testi merijo krvni tlak v območju med 70 mmHg in 170 mmHg (nižjih ali višjih vrednosti ne zaznajo), z njimi pa so izmerili krvni tlak pri 2268 osebah, od katerih je bila ena polovica bolnih (1134 oseb), druga polovica pa zdravih (1134 oseb). Rezultate so zapisali v datoteko labData.mat (npr. struktura `D`), v kateri spremenljivka `D.refData` predstavlja referenčne podatke (vrednost 0 predstavlja zdravo, vrednost 1 pa bolno osebo), spremenljivka `D.testData` pa predstavlja rezultate diagnostičnih testov (krvni tlak v mmHg), pri čemer vsaka vrstica `i` predstavlja posamezno osebo, vsak stolpec `j` pa posamezen diagnostični test:

	D.refData	D.testData			
i	(i,1)	(i,1)	(i,2)	(i,3)	(i,4)
⋮	⋮	⋮	⋮	⋮	⋮
71	0	85.5	98.5	114.8	115.0
72	1	144.2	70.6	113.8	115.0
73	0	93.4	145.6	117.9	115.0
⋮	⋮	⋮	⋮	⋮	⋮

1. Napišite funkcijo za binarno razvrščanje rezultatov:

```
def classifyData(iThreshold, iTestData, iRefData):
    # ...
    # your code goes here
    # ...
    return oTP, oTN, oFP, oFN
```

   kjer vhodni argument `iThreshold` predstavlja prag razvrščanja, `iTestData` je vektor rezultatov posameznega diagnostičnega testa, `iRefData` pa vektor referenčnih podatkov. Izhodni argumenti `oTP`, `oTN`, `oFP` in `oFN` predstavljajo število resnično pozitivnih (TP), število resnično negativnih (TN), število lažno pozitivnih (FP) in število lažno negativnih (FN) rezultatov. Podatke razvrščajte glede na `tNum` različnih pragov, pri čemer jih enakomerno razporedite med najmanjšim pragom `tMin` = 70 mmHg in največjim pragom `tMax` = 170 mmHg.

2. Napišite funkcijo za računanje deležev, ki merijo sposobnost binarnega razvrščanja:

```
def computeRates(iTP, iTN, iFP, iFN):
    # ...
    # your code goes here
    # ...
    return oTPR, oTNR, oFPR, oFNR
```

kjer vhodni argumenti `iTP`, `iTN`, `iFP` in `iFN` predstavljajo število resnično pozitivnih (TP), število resnično negativnih (TN), število lažno pozitivnih (FP) in število lažno negativnih (FN) rezultatov. Izhodni argumenti `oTPR`, `oTNR`, `oFPR` in `oFNR` predstavljajo delež resnično pozitivnih rezultatov (TPR oz. občutljivost), delež resnično negativnih rezultatov (TNR oz. specifičnost), delež lažno pozitivnih rezultatov (FPR oz. nespecifičnost) in delež lažno negativnih rezultatov (FNR oz. neobčutljivost).

3. Napišite funkcijo za računanje preostalih vrednosti, ki ravno tako merijo sposobnost binarnega razvrščanja:

```
def computeValues(iTP, iTN, iFP, iFN):
    # ...
    # your code goes here
    # ...
    return oPPV, oNPV, oFDR, oACC
```

kjer vhodni argumenti `iTP`, `iTN`, `iFP` in `iFN` predstavljajo število resnično pozitivnih (TP), število resnično negativnih (TN), število lažno pozitivnih (FP) in število lažno negativnih (FN) rezultatov. Izhodni argumenti `oPPV`, `oNPV`, `oFDR` in `oACC` predstavljajo pozitivno napovedno vrednost (PPV), negativno napovedno vrednost (NPV), delež lažnega odkrivanja rezultatov (FDR) in točnost razvrščanja (ACC).

4. Napišite funkcijo za izris rezultatov razvrščanja:

```
def drawResults(iX, iY, iTitle, iLabelX, iLabelY):
    # ...
    # your code goes here
    # ...
```

kjer vhodna argumenta `iX` in `iY` predstavljata vrednosti na x in y osi izrisa, `iTitle` je naslov izrisa, `iLabelX` in `iLabelY` pa oznake na x in y osi izrisa. Vhodna argumenta `iX` in `iY` naj bosta matriki, pri čemer vsaka vrstica predstavlja posamezni prag razvrščanja, vsak stolpec pa posamezen diagnostični test. Rezultate razvrščanja vseh testov izrišete v isti koordinatni sistem, legendo z generičnimi oznakami (`data1`, `data2`, `data3`, `data4`) pa lahko izrišete s pomočjo python funkcije `legend`.

Vprašanja

Odgovore na sledeča vprašanja zapišite v poročilo, v katerega vstavite zahtevane izrise in programske kode.

1. Razvrstite rezultate na podlagi `tNum` $= 20$ različnih pragov ter priložite slike poteka TPR, TNR, FPR in FNR v odvisnosti od praga razvrščanja, in sicer ločeno sliko za vsako mero sposobnosti razvrščanja, ki vsebuje krivulje vseh diagnostičnih testov.

2. Razvrstite rezultate na podlagi `tNum` = 20 različnih pragov ter priložite slike poteka PPV, NPV, FDR in ACC v odvisnosti od praga razvrščanja, in sicer ločeno sliko za vsako mero sposobnosti razvrščanja, ki vsebuje krivulje vseh diagnostičnih testov.

3. Napišite funkcijo za izračun površine pod krivuljo (AUC):

```
def computeAUC(iX, iY):
    # ...
    # your code goes here
    # ...
    return AUC
```

   kjer vhodna argumenta `iX` in `iY` predstavljata krivuljo oz. njene vrednosti na x in y osi izrisa, izhodni argument `oAUC` pa predstavlja površino pod krivuljo, ki jo izračunate po trapezni metodi.

   Priložite programsko kodo funkcije `computeAUC()`.

4. Razvrstite rezultate na podlagi `tNum` = 20 različnih pragov ter priložite sliko ROC krivulj, v kateri so prikazani vsi štirje diagnostični testi, ter podajte vrednosti površin pod ROC krivuljami. Na podlagi površine pod ROC krivuljo označite uspešnost posameznega diagnostičnega testa.

5. Na podlagi izrisanih potekov in krivulj določite optimalni prag razvrščanja za vsak diagnostični test posebej. Vašo izbiro ustrezno obrazložite.

6. Kateri diagnostični test je najboljši? Kaj lahko sklepate o testu, katerega ROC krivulja poteka popolnoma diagonalno? Kaj lahko sklepate o testu, katerega ROC krivulja poteka skoraj diagonalno? Odgovore ustrezno obrazložite.

7. Za prag razvrščanja izberite 120 mmHg in za prvi diagnostični test zapišite kontingenčno tabelo. Pri istem pragu razvrščanja podajte deleže PR, TNR, FPR in FNR ter vrednosti PPV, NPV, FDR in ACC za vse diagnostične teste (npr. v obliki tabele).

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati k poročilu, prispevajo pa naj k boljšemu razumevanju vsebine.

Napišite funkcijo za računanje praga razvrščanja, s katerim dosežemo izbrano občutljivost razvrščanja:

```
def computeThreshold(iTPR, iTestData, iRefData):
    # ...
    # your code goes here
    # ...
    return oThreshold, oTPR, oFPR
```

kjer vhodni argument `iTPR` predstavlja izbrani delež resnično pozitivnih rezultatov (TPR oz. občutljivost), `iTestData` je vektor rezultatov posameznega testa, `iRefData` pa vektor referenčnih podatkov. Izhodni argument `oThreshold` predstavlja prag, s katerim dosežemo izbrano občutljivost razvrščanja, `oTPR` predstavlja dejansko doseženo občutljivost razvrščanja, `oFPR` pa dejansko doseženo nespecifičnost razvrščanja.

Zapišite dobljene prage razvrščanja ter dejansko doseženo občutljivost in nespecifičnost razvrščanja pri izbranih občutljivostih razvrščanja 90,0%, 95,0% in 97,5%.